The "InclusionZone" 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

Thursday 7ᵗʰ November, 2013   2:21pm

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

1

Contents

1 Introduction

Inclusion zones in general are the area surrounding an object in which the sample point (e.g., a fixed-radius plot center) can land and sample the object. Therefore, they have a clearly defined area and perimeter. In addition, for probability proportional to size (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. Container classes "downLogIZs" and "standingTreeIZs" are also shown; they will handle collections of objects of any of the "downLogIZ" or "standingTreeIZ" subclasses, respectively.
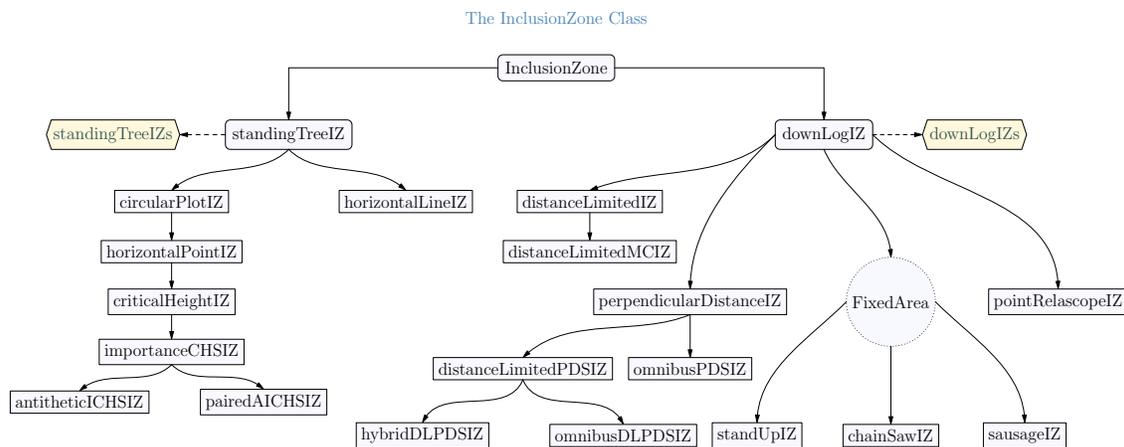

Figure 1: An overview of the "InclusionZone" class. "FixedArea" is not part of the class structure, but simply a subheading for the three protocol subclasses. The two container classes will be elaborated on later.

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, with 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 "standingTreeIZ", 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 "distanceLimitedIZ", by class "downLogIZ", distance 2
Class "fullChainSawIZ", by class "downLogIZ", distance 3
Class "omnibusPDSIZ", by class "downLogIZ", distance 3
Class "distanceLimitedMCIZ", by class "downLogIZ", distance 3
Class "distanceLimitedPDSIZ", by class "downLogIZ", distance 3
Class "omnibusDLPDSIZ", by class "downLogIZ", distance 4
Class "hybridDLPDSIZ", by class "downLogIZ", distance 4
Class "circularPlotIZ", by class "standingTreeIZ", distance 2
Class "horizontalPointIZ", by class "circularPlotIZ", distance 3
Class "horizontalLineIZ", by class "standingTreeIZ", distance 2
Class "criticalHeightIZ", by class "horizontalPointIZ", distance 4
Class "importanceCHSIZ", by class "criticalHeightIZ", distance 5
Class "antitheticICHSIZ", by class "importanceCHSIZ", distance 6
Class "pairedAICHSIZ", by class "importanceCHSIZ", distance 6
```

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 simply helps define 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 "fullChainSawIZ", by class "sausageIZ", distance 2
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
Class "hybridDLPDSIZ", 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:8i6r19vc
  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:o628cnb3
  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)
```
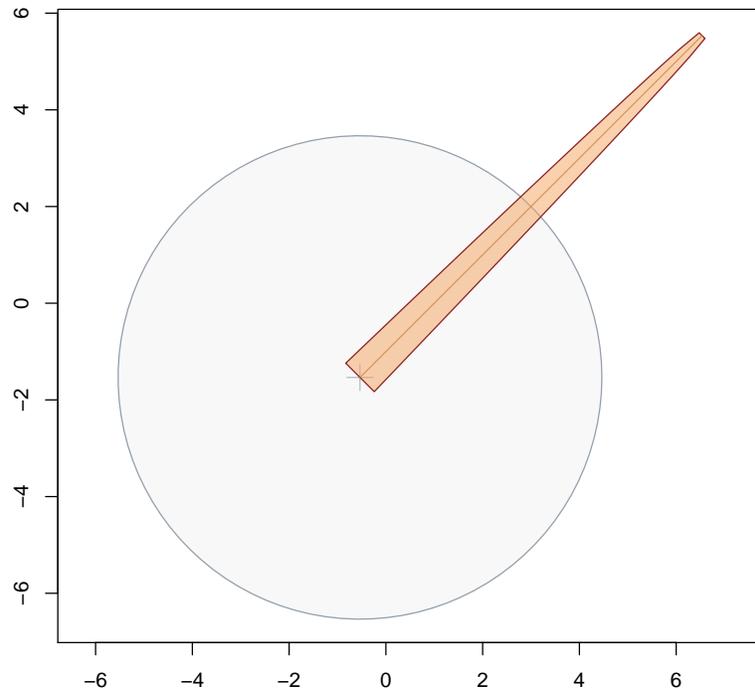
```
[1] TRUE
```

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) = 0.042612638
Section/sliver volume (not expanded) = 0.0077023016
```

```
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:8i6r19vc
  Number of logs: 3466.3947 per acre
  Volume: 26.699217 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): 816.99604 per acre
  Carbon content: 408.49802 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: 0.042612638 cubic feet
      Sliver volume: 0.0077023016 cubic feet
    Total log surface area: 17.811195 square feet
      Bounding bolt surface area: 0.68206195 square feet
      Sliver surface area: 0.12328378 square feet
    Total log coverage area: 5.6666667 square feet
      Bounding bolt coverage area: 0.2164317 square feet
      Sliver coverage area: 0.039120372 square feet
    Total log length: 10 feet
      Bounding bolt length: 0.8804157 feet
    Total log biomass: 84.497622
      Bounding bolt biomass: 1.3039467
      Sliver biomass: 0.23569043
    Total log carbon: 42.248811
      Bounding bolt carbon: 0.65197336
      Sliver carbon: 0.11784521

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) = 1.8407255
Section/sliver volume (not expanded) = 1.8004445
```

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)
```
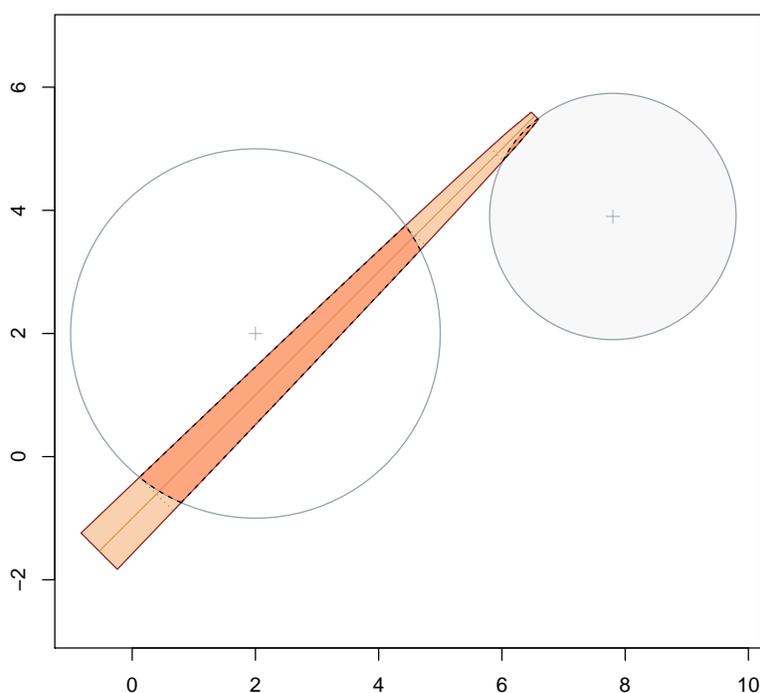

Figure 3: "chainsawIZ" objects comparison.

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

Known Subclasses: "fullChainSawIZ"
```

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:8i6r19vc
  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:o47x63py
  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`;
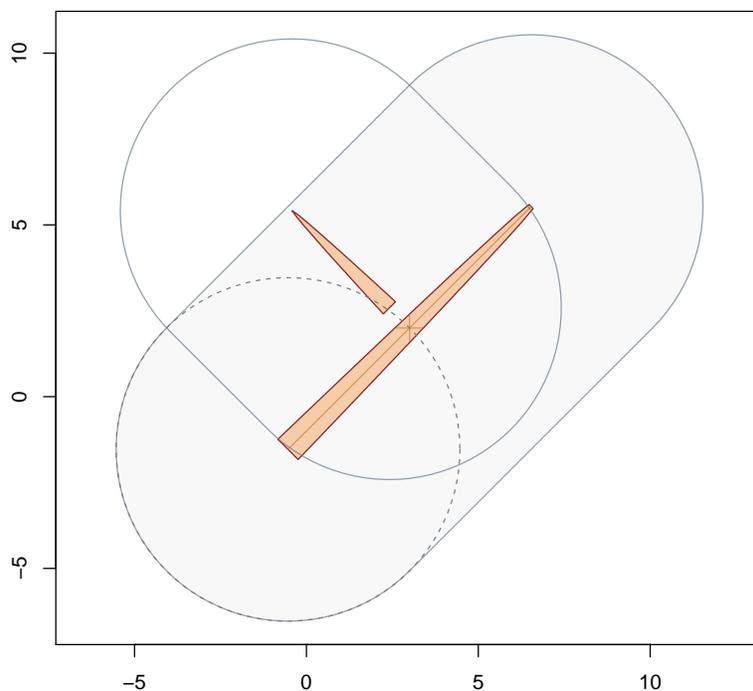
Figure 4: "sausageIZ" objects with "standUpIZ" perimeter for comparison.

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 "fullChainSawIZ" Class

This class incorporates the inclusion zone for the full chain saw method Gove and Van Deusen (2011). The inclusion zone itself is a direct subclass of the "sausageIZ" class discussed in §6 and it adds no new slots. It is given its own class for method dispatch on signature arguments allowing differentiation between this and the sausage method.

```
R> getClass('fullChainSawIZ')
```

```
Class "fullChainSawIZ" [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 "sausageIZ", directly
Class "downLogIZ", by class "sausageIZ", distance 2
Class "InclusionZone", by class "sausageIZ", distance 3
```

7.1 Object creation

The following will produce an inclusion zone that is exactly the same as that for the sausage sampling example in §6. . .

```
R> fcs = fullChainSawIZ(dl, 5)
R> summary(fcs)


Object of class: fullChainSawIZ
-------------------------------------------------------------
Inclusion zone for "fullChainSaw" 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:8i6r19vc
  Number of logs: 243.97919 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

sausageIZ...
  Spatial ID: fcs:uy61i30g
  radius = 5 feet
  area = 178.53982 square feet (0.004099 acres)
  Number of perimeter points: 101 (closed polygon)
```

7.2 Plotting the object

Similarly, plotting the object will result in the same inclusion zone as shown in Figure 4.

8 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
```

8.1 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 homogeneous 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).

8.2 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:8i6r19vc
  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:0982vuhd
  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.

8.3   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)
```
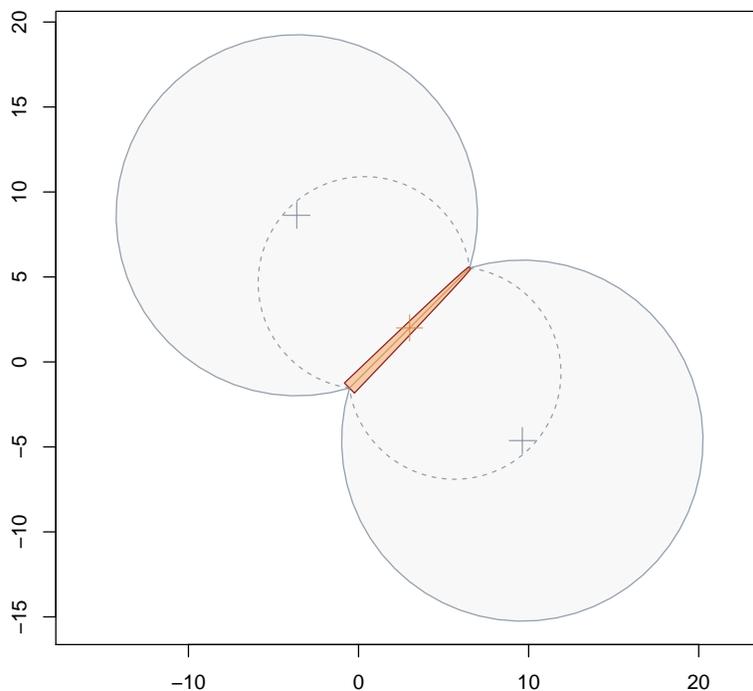
Figure 5: "pointRelascopeIZ" objects with different angles for comparison. Also shown are the dual circle centers for the smaller angle (larger inclusion zone).

9   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
Class "hybridDLPDSIZ", by class "distanceLimitedPDSIZ", distance 2
```

9.1   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 homogeneous 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.

9.2   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:8i6r19vc
  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:t2ce594r
  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 measurable 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).

9.3 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)
```
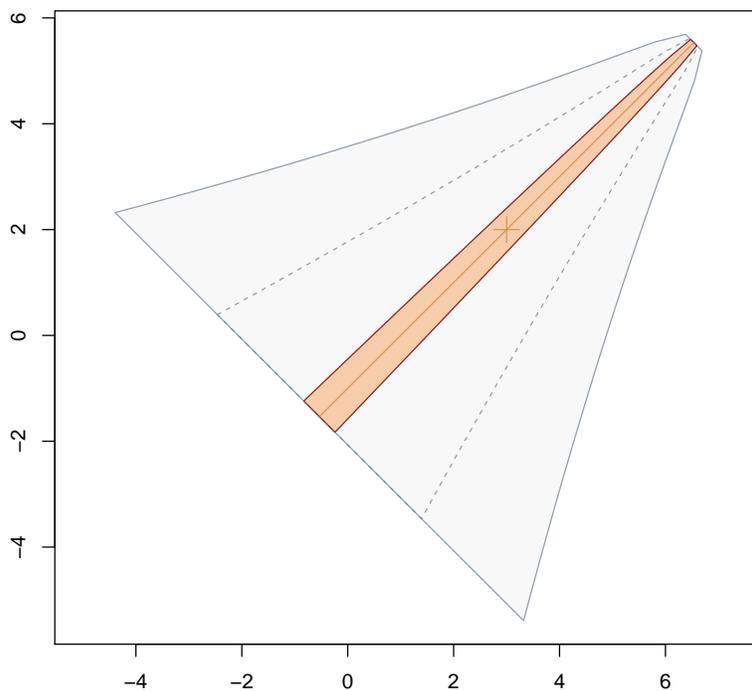

Figure 6: "perpendicularDistanceIZ" objects with different K PDS values for comparison.

9.4 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.

9.4.1 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:8i6r19vc
  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:xq1490li
  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.

9.4.2 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.

10 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 probability proportional 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"
```

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.

- *izPerim*: A matrix representation of the inclusion zone in homogeneous 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.

10.2   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:8i6r19vc
  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:7ozk82h3
  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).

10.3   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)
```
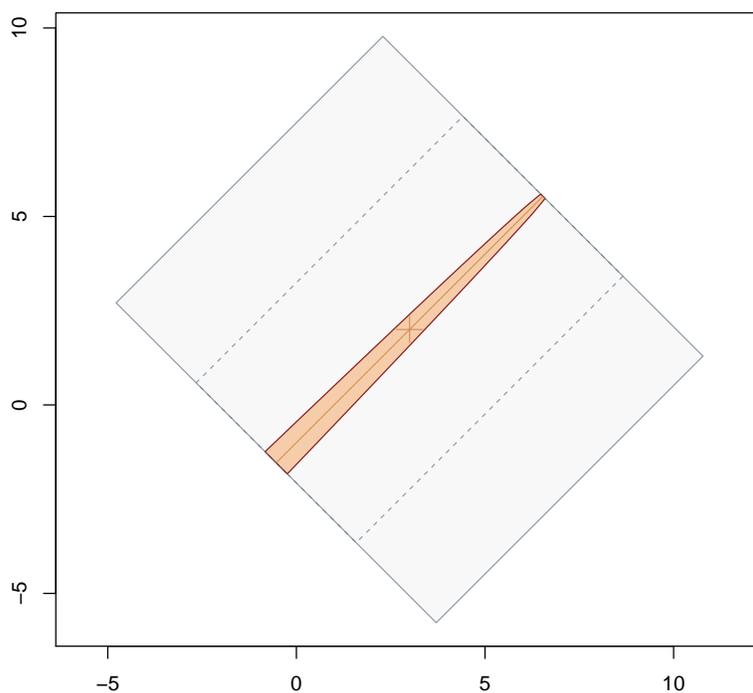

Figure 7: "distanceLimitedMCIZ" object.

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.

11 The "distanceLimitedPDSIZ" Class

Distance limited PDS (DLPDS) was initially developed by Ducey et al. (2013). 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 "down-Log" 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", "hybridDLPDSIZ"
```

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

11.1   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 homogeneous 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 compound 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.

11.2   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:8i6r19vc
  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

distanceLimitedPDSIZ...
  PDS type: volume
  Spatial ID: dlpds:30a2f4ju
  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 details...
  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...

$$d_l^* = \sqrt{\frac{4 * D_l}{\pi K_{\text{PDS}}}} \hspace{3cm} \text{PP volume}$$

$$d_l^* = \frac{D_l}{\pi K_{\text{PDS}}} \hspace{3cm} \text{PP Surface Area}$$

$$d_l^* = \frac{D_l}{K_{\text{PDS}}} \hspace{3cm} \text{PP Coverage Area}$$

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
```

11.3   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)
```
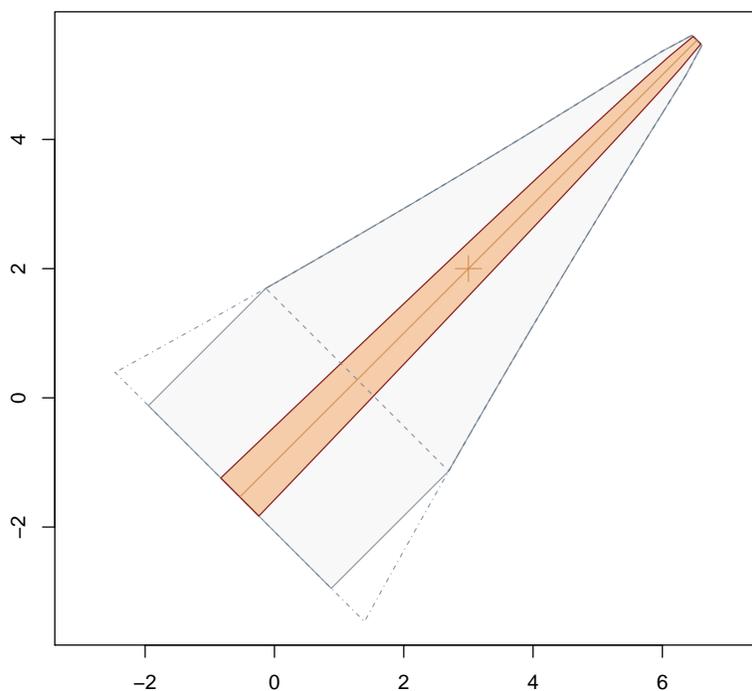

Figure 8: "distanceLimitedPDSIZ" object showing the PDS component delineation (dashed) and the full PDS inclusion zone (dot-dashed) for comparison.

11.4   The "omnibusDLPDSIZ" class

Omnibus DLPDS is discussed in Gove et al. (2013). 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 "distanceLimitedM-CIZ" 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.

11.4.1   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:8i6r19vc
  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

omnibusDLPDSIZ...
  PDS type: volume
  Spatial ID: odlpds:hq23i81d
  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 details...
  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.

11.4.2 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.

11.5 The "hybridDLPSIZ" class

This method is actually the original DLPDS sampling method introduced by Ducey et al. (2013). It is referred to here as a "hybrid" approach because it combines both a Monte Carlo distance limited estimator with a canonical PDS estimator. The overall estimator is design-unbiased as a consequence. It was envisioned in the original account to be applicable to the selection attributes of volume and surface area (though it extends nicely to coverage area as well). The "hybridDLPDSIZ" 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 on the superclass for the class definition.

It follows that, as with "omnibusDLPDSIZ", the objects in the "Part" slots will hold objects corresponding to the above. The `pdsPart` and `pdsFull` slots will hold objects of class "perpendicularDistanceIZ", while the `dlsPart` slot will hold an object of class "distanceLimitedMCIZ". Recall, that either of these may be `Null` if the log does not divide into the two components.

11.5.1 Object creation

Again, the inclusion zone is exactly the same as for the other distance limited protocols, so object creation is the same as well. And again, because of the Monte Carlo aspect of the distance limited component of the inclusion zone, there will be no overall estimate of the attributes. The inclusion zone will be flat in the PDS component, and sloping in the DLMCS component. More will be covered on this in "*The InclusionZoneGrid Class*" vignette.

```
R> iz.hdlpds = hybridDLPDSIZ(dl, pds=pdsEng2, dls=dlsEng3)
R> iz.hdlpds
```

```
Object of class: hybridDLPDSIZ
-------------------------------------------------------------
inclusion zone for down log hybrid 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:8i6r19vc
   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

hybridDLPDSIZ...
   PDS type: volume
   Spatial ID: hdlpds:pm3607kf
   area = 25.755564 square feet (0.0005913 acres)
   Number of perimeter points: 37 (closed polygon)
   (The above summary is for the entire DLPDS region)

hybridDLPDSIZ details...
   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.hdlpds@pdsPart)

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

R> class(iz.hdlpds@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.

11.5.2 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.

12 The "standingTreeIZ" Class

This class forms the base class for methods dealing with sampling standing trees and is analogous to the "downLogIZ" class for downed logs. It is a virtual class, and adds only one slot to the "InclusionZone" class definition, that slot holds the "standingTree" object...

```
R> getClass('standingTreeIZ')


Virtual Class "standingTreeIZ" [package "sampSurf"]

Slots:

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

Name:     puaBlowup puaEstimates     userExtra
Class:      numeric          list          ANY

Extends: "InclusionZone"

Known Subclasses:
Class "circularPlotIZ", directly
Class "horizontalLineIZ", directly
Class "horizontalPointIZ", by class "circularPlotIZ", distance 2
Class "criticalHeightIZ", by class "horizontalPointIZ", distance 3
Class "importanceCHSIZ", by class "criticalHeightIZ", distance 4
Class "antitheticICHSIZ", by class "importanceCHSIZ", distance 5
Class "pairedAICHSIZ", by class "importanceCHSIZ", distance 5
```

13   The "circularPlotIZ" Class

Sampling standing trees with fixed-area circular plots has long been a standard method used in forestry, ecology, and related disciplines. This class is a direct descendant of the "standingTreeIZ" class and adds only one slot to its definition. The slot holds a "circularPlot" class object from the "ArealSampling" superclass.

```
R> getClass('circularPlotIZ')


Class "circularPlotIZ" [package "sampSurf"]

Slots:

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


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


Extends:
Class "standingTreeIZ", directly
Class "InclusionZone", by class "standingTreeIZ", distance 2


Known Subclasses:
Class "horizontalPointIZ", directly
Class "criticalHeightIZ", by class "horizontalPointIZ", distance 2
Class "importanceCHSIZ", by class "criticalHeightIZ", distance 3
Class "antitheticICHSIZ", by class "importanceCHSIZ", distance 4
Class "pairedAICHSIZ", by class "importanceCHSIZ", distance 4
```

13.1   Object creation

The constructor method for this class takes a "standingTree" object and the fixed-area plot radius as the two signature arguments. The plot radius gets converted to an object of class "circularPlot" internally.[2] An example is the following fir a $\frac{1}{10}$th acre plot...

---

[2]The use of a plot radius rather than passing a "circularPlot" object is for consistency with the down woody debris sampling methods. The latter were the first methods written and regretfully, not a lot of forward thought went into their design.

```
R> sttr = standingTree(species='red oak', height=50, dbh=10,
+            topDiam=0, center=c(x=40,y=40),
+            units='English', vol2wgt=34.9, wgt2carbon=0.5)
R> pltRad = function(plotSize) sqrt(43560*plotSize/pi)
R> sttr.cpiz = circularPlotIZ(sttr, plotRadius = pltRad(0.1))
R> summary(sttr.cpiz)


Object of class: circularPlotIZ
---------------------------------------------------------------
inclusion zone for circular plot method
---------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 10 per acre

  Object bounding box...
        min        max
x 2.7822345 77.236513
y 2.7681745 77.231825

standingTree component estimates...
  Spatial ID: tree:j3qf281g
  Number of trees: 10 per acre
  Basal area: 5.4541539 square feet per acre
  Volume: 132.53546 cubic feet per acre
  Surface area: 836.389 square feet per acre
  Biomass (stem): 4625.4875 per acre
  Carbon content: 2312.7438 per acre

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


R> summary(sttr.cpiz@circularPlot)


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

ArealSampling...
  units of measurement:  English
```

```
circularPlot...
  radius = 37.236513 feet
  area = 4356 square feet (0.1 acres)
  spatial units:  NA
  spatial ID: cp:j9f4q2r7
  location (plot center)...
    x coord:  40
    y coord:  40
  Number of perimeter points: 101 (closed polygon)
```

13.2 Plotting the object

As with other classes, the `plot` generic has been extended to facilitate graphical display...

```
R> plot(sttr.cpiz, axes=TRUE, cex=2)
R> sttr.cpiz2 = circularPlotIZ(sttr, plotRadius = pltRad(0.05))
R> plot(sttr.cpiz2, showTree=FALSE, izColor=NA, lty='dashed', add=TRUE)
R> plot(circularPlot(pltRad(0.02),centerPoint=c(x=40,y=40), units='English'),
+      add=TRUE, lty='dotted', izColor=NA)
```

The example in Figure 9 displays both the circular plot object and the standing tree object by default in the first command. Next, a second "circularPlotIZ" object has been added to the plot to show some of the options available in the `plot` method for this class found in the third line. It should be obvious by now that there are many ways to create the same graphical effect as is shown in these two lines. The one taken here is really overkill. For example, we could also have simply plotted the perimeter of the `circularPlot` slot in the "circularPlotIZ" object using the `perimeter` method, or the just plotted the slot object itself. Finally, if a separate inclusion zone object is not required for other use, what amounts to the same thing graphically and is simpler would be to just create and plot a `circularPlot` object as is shown in the last line of code. These examples just serve to illustrate the flexibility available that can be used in different circumstances as the need arises.

14 The "horizontalPointIZ" Class

This class encompasses what has come to be known as horizontal point sampling, angle gauge sampling, or classic Bitterlich sampling for standing trees. The inclusion zones are circular and proportional to tree basal area in this case. The class definition is...
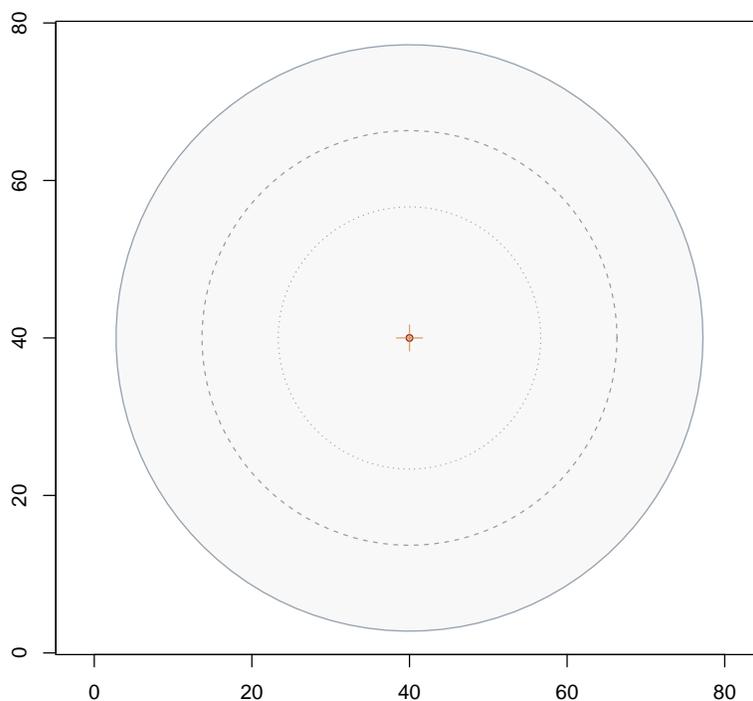
Figure 9: Three "circularPlotIZ" objects with different radii.

```
R> getClass('horizontalPointIZ')


Class "horizontalPointIZ" [package "sampSurf"]

Slots:

Name:     angleGauge circularPlot standingTree   description         units
Class:    angleGauge circularPlot standingTree     character     character

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

Extends:
Class "circularPlotIZ", directly
Class "standingTreeIZ", by class "circularPlotIZ", distance 2
```

```
Class "InclusionZone", by class "circularPlotIZ", distance 3

Known Subclasses:
Class "criticalHeightIZ", directly
Class "horizontalPointCMCIZ", directly
Class "horizontalPointISIZ", directly
Class "horizontalPointCVIZ", directly
Class "importanceCHSIZ", by class "criticalHeightIZ", distance 2
Class "antitheticICHSIZ", by class "criticalHeightIZ", distance 3
Class "pairedAICHSIZ", by class "criticalHeightIZ", distance 3
```

The "horizontalPointIZ" class is a subclass of the "circularPlotIZ", and adds only one slot for the angle gauge information used to create the inclusion via an "angleGauge" object. The "circularPlotIZ" class already knows how to handle circular inclusion zones centered on a "standingTree" object. The "horizontalPointIZ" class takes advantage of this inherent functionality associated with different methods already developed for the "circularPlotIZ" class and simply makes each object's circular plot variable (depending on the gauge angle and tree basal area), rather than fixed. As a consequence, there is no need for a separate `plot` method, for example, since the method for "circularPlotIZ" already knows how to handle circular inclusion zones. If you have followed this reasoning, this is a good example of how inheritance allows use of existing methods applied to subclass objects.

14.1   Object creation

It should not be surprising that the constructor is very similar to that for the superclass, but substituting an "angleGauge" object for the plot radius, which is determined internally based on the tree size and gauge angle.

```
R> aGauge = angleGauge(baf=10, units='English')
R> sttr.hpiz = horizontalPointIZ(sttr, aGauge)
R> summary(sttr.hpiz)


Object of class: horizontalPointIZ
-------------------------------------------------------------
inclusion zone for horizontal point sampling method
-------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 18.334649 per acre
```

```
   Object bounding box...
         min       max
x 12.513845 67.500000
y 12.503461 67.496539

standingTree component estimates...
  Spatial ID: tree:j3qf281g
  Number of trees: 18.334649 per acre
  Basal area: 10 square feet per acre
  Volume: 242.99912 cubic feet per acre
  Surface area: 1533.4899 square feet per acre
  Biomass (stem): 8480.6692 per acre
  Carbon content: 4240.3346 per acre

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

horizontalPointIZ...
  use "summary" on the angleGauge slot for details


R> sttr.hpiz@circularPlot


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

ArealSampling...
  units of measurement:  English

circularPlot...
  radius = 27.5 feet
  area = 2375.8294 square feet (0.05454 acres)
  spatial units:  NA
  spatial ID: hps:83da6yf5
  location (plot center)...
    x coord:  40
    y coord:  40
  Number of perimeter points: 101 (closed polygon)
```

The diameter of the "standingTree" object is 10 inches, and the plot radius factor for a 10 BAF factor prism is 2.75, so the variable plot radius for this tree is 27.5 feet.

14.2   Plotting the object

An object of class "horizontalPointIZ" can be plotted as...

```
R> plot(sttr.hpiz, axes=TRUE, cex=2)
```
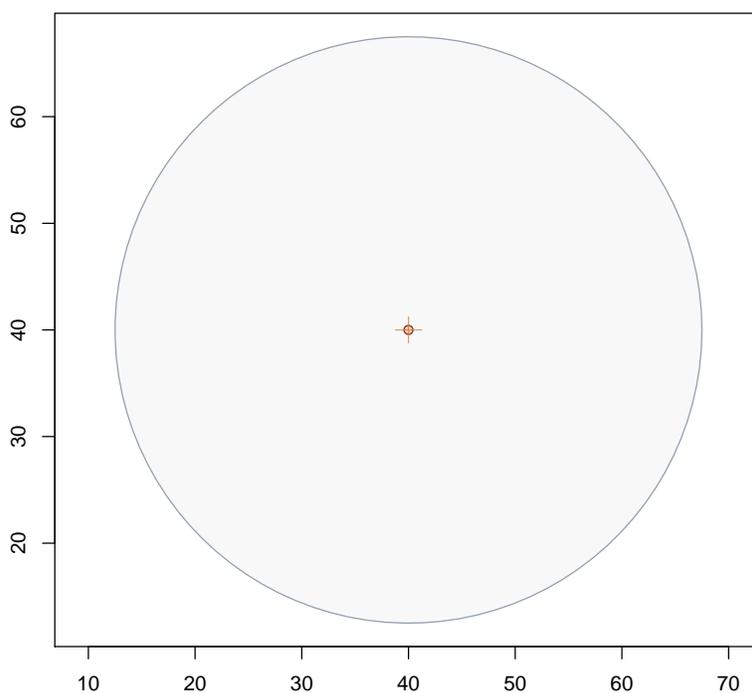

Figure 10: A "horizontalPointIZ" object.

Because the `plot` method for the "circularPlotIZ" class is used via inheritance here, there is nothing new to remark about in Figure 10.

14.3   The horizontalPointCMCIZ Class

This class is for two-stage horizontal point sampling for volume estimation, where crude Monte Carlo is used to subsample the tree for volume. The class definition is...

```
R> showClass('horizontalPointCMCIZ')
```

```
Class "horizontalPointCMCIZ" [package "sampSurf"]

Slots:

Name:           mcsObj              antithetic                  proxy
Class: MonteCarloSampling              logical              character

Name:         angleGauge           circularPlot           standingTree
Class:         angleGauge           circularPlot           standingTree

Name:        description                  units                   bbox
Class:          character              character                 matrix

Name:            spUnits               puaBlowup            puaEstimates
Class:                CRS                numeric                    list

Name:           userExtra
Class:                ANY

Extends:
Class "MonteCarloSamplingIZ", directly
Class "horizontalPointIZ", directly
Class "horizontalPointMonteCarloSamplingIZ", directly
Class "circularPlotIZ", by class "horizontalPointIZ", distance 2
Class "standingTreeIZ", by class "horizontalPointIZ", distance 3
Class "InclusionZone", by class "horizontalPointIZ", distance 4
```

This class is a combination of the "MonteCarloSamplingIZ" and "horizontalPointIZ" class definitions via inheritance. The first three slots in the above class definition are supplied by the "MonteCarloSamplingIZ" class, and the rest are supplied by the "horizontalPointIZ" class. All of the later slots will therefore define a valid "horizontalPointIZ" object, and the usual `plot` and `summary` routines will display the same information. The "MonteCarloSamplingIZ" slots carry information that will eventually be used to grid the volume surface in the corresponding `izGrid` constructor. More details on this can be found in Gove (2013) and Gove (2011).

14.4   The horizontalPointISIZ Class

This class is for two-stage horizontal point sampling for volume estimation, where importance sampling is used to subsample the tree for volume. The class definition is...

```
R> showClass('horizontalPointISIZ')


Class "horizontalPointISIZ" [package "sampSurf"]

Slots:

Name:              mcsObj            antithetic                 proxy
Class: MonteCarloSampling              logical             character

Name:           angleGauge          circularPlot          standingTree
Class:           angleGauge          circularPlot          standingTree

Name:          description                 units                  bbox
Class:           character             character                matrix

Name:              spUnits              puaBlowup          puaEstimates
Class:                  CRS               numeric                  list

Name:             userExtra
Class:                  ANY


Extends:
Class "MonteCarloSamplingIZ", directly
Class "horizontalPointIZ", directly
Class "horizontalPointMonteCarloSamplingIZ", directly
Class "circularPlotIZ", by class "horizontalPointIZ", distance 2
Class "standingTreeIZ", by class "horizontalPointIZ", distance 3
Class "InclusionZone", by class "horizontalPointIZ", distance 4
```

The same information applies here as to crude Monte Carlo subsampling, see § 14.3 for details.

14.5   The horizontalPointCVIZ Class

This class is for two-stage horizontal point sampling for volume estimation, where control variate sampling is used to subsample the tree for volume. The class definition is...


```
R> showClass('horizontalPointCVIZ')


Class "horizontalPointCVIZ" [package "sampSurf"]
```

```
Slots:

Name:                   mcsObj              antithetic                  proxy
Class: MonteCarloSampling                      logical              character

Name:               angleGauge            circularPlot          standingTree
Class:              angleGauge            circularPlot          standingTree

Name:              description                   units                  bbox
Class:               character               character                matrix

Name:                  spUnits                puaBlowup           puaEstimates
Class:                     CRS                 numeric                   list

Name:                userExtra
Class:                     ANY


Extends:
Class "MonteCarloSamplingIZ", directly
Class "horizontalPointIZ", directly
Class "horizontalPointMonteCarloSamplingIZ", directly
Class "circularPlotIZ", by class "horizontalPointIZ", distance 2
Class "standingTreeIZ", by class "horizontalPointIZ", distance 3
Class "InclusionZone", by class "horizontalPointIZ", distance 4
```

The same information applies here as to crude Monte Carlo subsampling, see § 14.3 for details.

15   The "horizontalLineIZ" Class

The "horizontalLineIZ" class is for horizontal line sampling of standing trees with an angle gauge (e.g., Gregoire and Valentine, 2008, p. 359). The inclusion zones are rectangular and proportional to tree diameter at breast height. The class definition is...

R> showClass('horizontalLineIZ')


```
Class "horizontalLineIZ" [package "sampSurf"]

Slots:

Name:        angleGauge        lineSegment            width            length
```

```
Class:      angleGauge      lineSegment         numeric          numeric

Name:           izPerim             area        perimeter           pgArea
Class:           matrix          numeric SpatialPolygons          numeric

Name:      standingTree      description            units             bbox
Class:     standingTree        character        character           matrix

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

Extends:
Class "standingTreeIZ", directly
Class "InclusionZone", by class "standingTreeIZ", distance 2

15.1 Class slots

The following slots have been added to this subclass; note that it is a direct subclass of "standingTree". . .

- *angleGauge*: An object of class "angleGauge" used in conjunction with the tree diameter to determine the inclusion zone width.

- *lineSegment*: An object of class "lineSegment" used to determine the length of the inclusion zone.

- *width*: The width of the inclusion zone.

- *length*: The length of the inclusion zone.

- *izPerim*: A matrix representation of the inclusion zone in homogeneous 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.

15.2   Object creation

The constructor method for this class takes "standingTree" and "angleGauge" objects as well as the line length as the signature arguments. The line `orientation` defaults to north (0 degrees). An object of class "lineSegment" is created internally from the `lineLength` and `orientation` arguments, centered on the tree location. An example follows. . .

```
R> aGauge = angleGauge(baf=75, units='English')
R> sttr.hliz = horizontalLineIZ(sttr, angleGauge = aGauge, lineLength = 50,
+                               orientation = 345, units='English')
R> summary(sttr.hliz)
```

```
Object of class: horizontalLineIZ
-------------------------------------------------------------
inclusion zone for horizontal line sampling method
-------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 43.379627 per acre

  Object bounding box...
        min        max
x 23.830102 56.169898
y 13.252902 66.747098

standingTree component estimates...
  Spatial ID: tree:j3qf281g
  Number of trees: 43.379627 per acre
  Basal area: 23.659916 square feet per acre
  Volume: 574.93387 cubic feet per acre
  Surface area: 3628.2242 square feet per acre
  Biomass (stem): 20065.192 per acre
  Carbon content: 10032.596 per acre

horizontalLineIZ...
  Inclusion zone length = 50 feet
  Inclusion zone width = 20.08316 feet
  Inclusion zone area = 1004.158 square feet
  Note: use "summary" on the angleGauge & lineSegment slots for details
```

The diameter of the "standingTree" object is 10 inches, and the plot radius factor for a 75 BAF factor prism is 1, so the variable plot width for this tree is 20.1 feet as shown above.[3]

15.3   Plotting the object

An object of class "horizontalLineIZ" can be plotted as, e.g.,...

```
R> plot(sttr.hliz, axes=TRUE, cex=1.5)
```
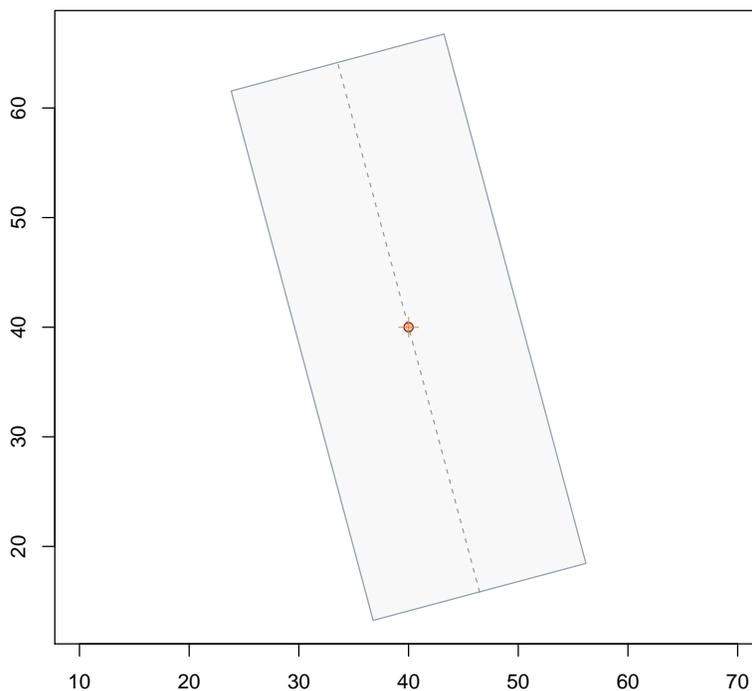

Figure 11: A "horizontalLineIZ" object.

Figure 11 shows the results. Note that the line segment is shown as the dashed line through the tree (the segment center is not shown by default as it would overlap the tree in the display). The

---

[3]The 75 BAF angle gauge is a nice BAF to use with horizontal line sampling because checking borderline trees is simple with a plot radius of one foot for every inch in diameter.

inclusion zone is centered on the tree with the desired orientation. The `cex` argument is used to enlarge the location of the center of the tree.

16   The "criticalHeightIZ" Class

Critical height sampling (CHS) is a design-unbiased method for estimating volumes of standing trees based on horizontal point sampling (§ 14). The inclusion zones are exactly the same as in HPS, when the reference height is at breast height (BH). The reference height can also be at the butt (ground level), in which case the inclusion zone will be larger than that for HPS. The class definition is...

```
R> getClass('criticalHeightIZ')


Class "criticalHeightIZ" [package "sampSurf"]

Slots:

Name:   referenceHeight       angleGauge      circularPlot       standingTree
Class:          character       angleGauge      circularPlot       standingTree

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

Name:            puaBlowup      puaEstimates         userExtra
Class:             numeric              list               ANY

Extends:
Class "horizontalPointIZ", directly
Class "circularPlotIZ", by class "horizontalPointIZ", distance 2
Class "standingTreeIZ", by class "horizontalPointIZ", distance 3
Class "InclusionZone", by class "horizontalPointIZ", distance 4

Known Subclasses:
Class "importanceCHSIZ", directly
Class "antitheticICHSIZ", by class "importanceCHSIZ", distance 2
Class "pairedAICHSIZ", by class "importanceCHSIZ", distance 2
```

Note that "criticalHeightIZ" is a direct subclass of "horizontalPointIZ". The only new slot is the `referenceHeight` slot, which is a character string that must match one of the following...

```
R> .StemEnv$referenceCHSIZ
```

```
[1] "butt" "dbh"
```

Please see § 14 for the other slot definitions.

16.1 Object creation

The constructor is very similar to that for the superclass...

```
R> aGauge = angleGauge(baf=10, units='English')
R> sttr.chsiz = criticalHeightIZ(sttr, aGauge, referenceHeight='dbh')
R> summary(sttr.chsiz)
```

```
Object of class: criticalHeightIZ
---------------------------------------------------------------
inclusion zone for critical height sampling method
---------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 10 per acre

  Object bounding box...
        min        max
x 12.513845 67.500000
y 12.503461 67.496539

standingTree component estimates...
  Spatial ID: tree:j3qf281g
  Number of trees: NA per acre
  Basal area: NA square feet per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Biomass (stem): NA per acre
  Carbon content: NA per acre

circularPlotIZ...
  use "summary" on the circularPlot slot for details
```

```
horizontalPointIZ...
  use "summary" on the angleGauge slot for details

criticalHeightIZ...
  Inclusion zone reference height is: dbh
```

One thing to note is that all of the estimates are `NA` in the above object. There are two reasons for this. First, critical height is only used for estimating tree volume, so all other attributes are not applicable. Second, the estimates are different for each "ring" of radius that we step back away from the tree within the inclusion zone, and depend on the grid cell size. Therefore the estimates for the tree get calculated in the `izGrid` method for CHS (see *The "InclusionZoneGrid" Class* vignette for details), producing the variable surface within each tree's inclusion zone.

16.2 Plotting the object

Plotting the object is the same as for "horizontalPointIZ", and in this case, will look exactly the same as the zone in Figure 10, since we have sampled at BH.

16.3 The "importanceCHSIZ" Class

Importance sampling can be combined with critical height in the hopes of reducing the variance of the CHS estimator (which is already pretty good). This class adds an importance sampling variant that chooses an upper stem cross-sectional area through a taper function that makes use of the critical diameter associated with the critical height for a given sample point. Details are too much to cover here and are found in Lynch and Gove (2013). Suffice it to say that the importance sampling point so chosen is constrained by the critical height, producing similar sampling surfaces, but with better variance.

This class is a subclass of "criticalHeightIZ" and is defined as...

```
R> getClass('importanceCHSIZ')


Class "importanceCHSIZ" [package "sampSurf"]

Slots:

Name:  referenceHeight    angleGauge    circularPlot    standingTree
Class:       character     angleGauge    circularPlot    standingTree
```

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

Name:         puaBlowup      puaEstimates      userExtra
Class:          numeric              list            ANY


Extends:
Class "criticalHeightIZ", directly
Class "horizontalPointIZ", by class "criticalHeightIZ", distance 2
Class "circularPlotIZ", by class "criticalHeightIZ", distance 3
Class "standingTreeIZ", by class "criticalHeightIZ", distance 4
Class "InclusionZone", by class "criticalHeightIZ", distance 5


Known Subclasses: "antitheticICHSIZ", "pairedAICHSIZ"
```

Note that there are no new slots added to this class, please see § 16 for details.

16.3.1 Object creation

The constructor is very similar to that for the superclass...

```
R> sttr.ichsiz = importanceCHSIZ(sttr, aGauge, referenceHeight='dbh')
R> summary(sttr.ichsiz)


Object of class: importanceCHSIZ
-------------------------------------------------------------
inclusion zone for importance CH sampling method
-------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 10 per acre

  Object bounding box...
         min        max
x 12.513845 67.500000
y 12.503461 67.496539

standingTree component estimates...
  Spatial ID: tree:j3qf281g
```

```
  Number of trees: NA per acre
  Basal area: NA square feet per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Biomass (stem): NA per acre
  Carbon content: NA per acre

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

horizontalPointIZ...
  use "summary" on the angleGauge slot for details

criticalHeightIZ...
  Inclusion zone reference height is: dbh
```

16.3.2   Plotting the object

Plotting the object is the same as for "horizontalPointIZ", and in this case, will look exactly the same as the zone in Figure 10, since we have sampled at BH.

16.4   The "antitheticICHSIZ" Class

Antithetic sampling can be used to move the importance sampling point (based on CHS) on the tree bole so that it is easier to measure. Details are again given in Lynch and Gove (2013). This is a subclass of "importanceCHSIZ"...

```
R> getClass('antitheticICHSIZ')


Class "antitheticICHSIZ" [package "sampSurf"]

Slots:

Name:   referenceHeight      angleGauge      circularPlot      standingTree
Class:        character      angleGauge      circularPlot      standingTree


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


Name:        puaBlowup      puaEstimates         userExtra
```

```
Class:          numeric          list          ANY

Extends:
Class "importanceCHSIZ", directly
Class "criticalHeightIZ", by class "importanceCHSIZ", distance 2
Class "horizontalPointIZ", by class "importanceCHSIZ", distance 3
Class "circularPlotIZ", by class "importanceCHSIZ", distance 4
Class "standingTreeIZ", by class "importanceCHSIZ", distance 5
Class "InclusionZone", by class "importanceCHSIZ", distance 6
```

Note that there are no new slots added to this class, please see § 16 for details.

16.4.1 Object creation

The constructor is very similar to that for the superclass...

```
R> sttr.aichsiz = antitheticICHSIZ(sttr, aGauge, referenceHeight='dbh')
R> summary(sttr.aichsiz)


Object of class: antitheticICHSIZ
--------------------------------------------------------------
inclusion zone for antithetic ICH sampling method
--------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 10 per acre

  Object bounding box...
        min        max
x 12.513845 67.500000
y 12.503461 67.496539

standingTree component estimates...
  Spatial ID: tree:j3qf281g
  Number of trees: NA per acre
  Basal area: NA square feet per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Biomass (stem): NA per acre
```

```
   Carbon content: NA per acre

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

horizontalPointIZ...
  use "summary" on the angleGauge slot for details

criticalHeightIZ...
  Inclusion zone reference height is: dbh
```

16.4.2   Plotting the object

Plotting the object is the same as for "horizontalPointIZ", and in this case, will look exactly the same as the zone in Figure 10, since we have sampled at BH.

16.5   The "pairedAICHSIZ" Class

A paired antithetic sampling estimator was again given in Lynch and Gove (2013), consisting of two measurements per tree. This, like "antitheticICHS", is a subclass of "importanceCHSIZ"...

```
R> getClass('pairedAICHSIZ')


Class "pairedAICHSIZ" [package "sampSurf"]

Slots:

Name:   referenceHeight      angleGauge      circularPlot      standingTree
Class:        character       angleGauge      circularPlot      standingTree

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

Name:          puaBlowup      puaEstimates        userExtra
Class:           numeric              list              ANY

Extends:
Class "importanceCHSIZ", directly
Class "criticalHeightIZ", by class "importanceCHSIZ", distance 2
Class "horizontalPointIZ", by class "importanceCHSIZ", distance 3
```

```
Class "circularPlotIZ", by class "importanceCHSIZ", distance 4
Class "standingTreeIZ", by class "importanceCHSIZ", distance 5
Class "InclusionZone", by class "importanceCHSIZ", distance 6
```

Note that there are no new slots added to this class, please see § 16 for details.

16.5.1   Object creation

The constructor is very similar to that for the superclass...

```
R> sttr.paichsiz = pairedAICHSIZ(sttr, aGauge, referenceHeight='dbh')
R> summary(sttr.paichsiz)
```

```
Object of class: pairedAICHSIZ
-------------------------------------------------------------
inclusion zone for paired antithetic ICH sampling method
-------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 10 per acre

  Object bounding box...
        min        max
x 12.513845 67.500000
y 12.503461 67.496539

standingTree component estimates...
  Spatial ID: tree:j3qf281g
  Number of trees: NA per acre
  Basal area: NA square feet per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Biomass (stem): NA per acre
  Carbon content: NA per acre

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

horizontalPointIZ...
```

```
  use "summary" on the angleGauge slot for details

criticalHeightIZ...
  Inclusion zone reference height is: dbh
```

16.5.2   Plotting the object

Plotting the object is the same as for "horizontalPointIZ", and in this case, will look exactly the same as the zone in Figure 10, since we have sampled at BH.

16.5.3   You've seen one. . .

It is easy to wonder why one would want all these different classes of objects (plus CHS) if they produce the same inclusion zones? The answer is that the sampling surfaces within these protocols are different because the estimators are different. The associated "InclusionZone" subclass objects are the container for the surfaces in a sense. This will become clear in *The "InclusionZoneGrid" Class* vignette, and it is no different than what we have already encountered with some of the protocols with PDS or the DLMCS methods. So in the larger scheme of things within `sampSurf`, having different classes for each of these protocols facilitates the subsequent programming of constructor methods in a strict `S4` object oriented sense.

17   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. 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. This means there needs to be class and constructor methods, along with summary, plot, etc. for this collection of objects.

17.1   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", "standingTreeIZs"
```

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.

17.2   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"
```

17.2.1   Class construction

In keeping with the previous naming conventions, the constructor function for this class is **down-LogIZs**. 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. However, it is perhaps simpler to use the alternative constructor that works directly on a "downLogs" object collection—both methods are illustrated in the following...

```
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> izs.sa = downLogIZs(dls, iZone = 'sausageIZ', plotRadius=5)
```

Both constructors are equally useful, the second form simply builds the list internally using `lapply`, then calls the first constructor on the `list` object. Note that the code in the second and third lines can be combined into one command, but is separated here for clarity.

17.2.2 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 12. . .

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

In the second example, where we use synthetic logs, we generate Figure 13 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 13, 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.

17.3 Class "standingTreeIZs"

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

```
R> showClass('standingTreeIZs')
```
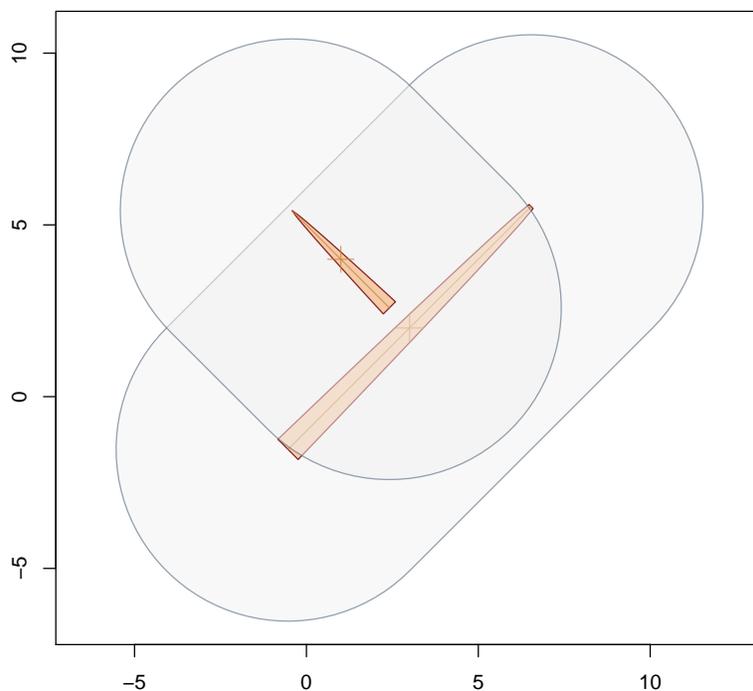
```
Class "standingTreeIZs" [package "sampSurf"]
```

Figure 12: "downLogIZs" object with "sausageIZ" objects in the container.

```
Slots:

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

Extends: "izContainer"
```

The class is the standing tree analog to the "downLogIZs" class presented above.

17.3.1   Class construction

Because the mechanisms for class construction from measured trees or synthetically generated collections are analogous to those for "downLogIZs", only a simple example will be given here. For more details see the previous section and substitute the appropriate standing tree-related functions.
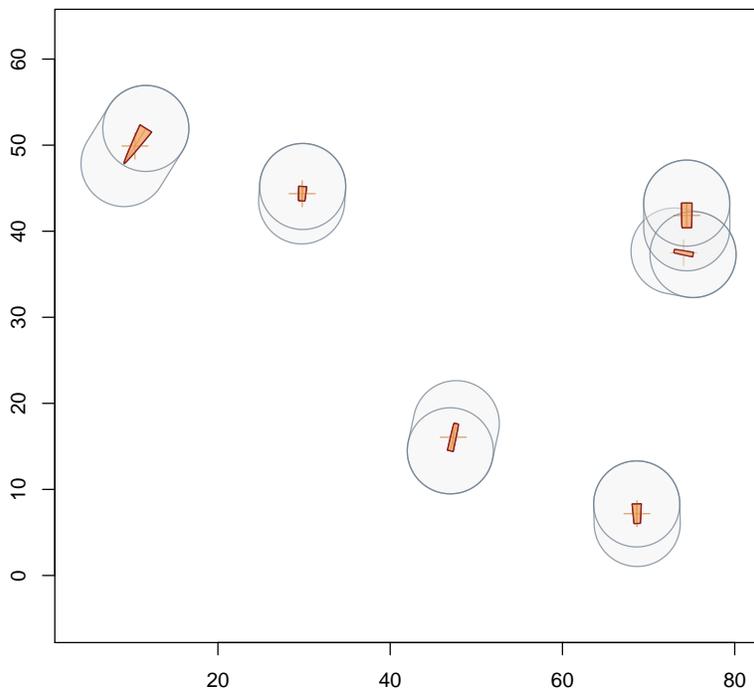
Figure 13: "downLogIZs" object of "sausageIZ" objects, with "standUpIZ" objects superimposed.

```
R> etract = Tract(c(x=209,y=209),cellSize=0.5, units='English')
R> ebuffTr = bufferedTract(30, etract)
R> sttrs = standingTrees(10, ebuffTr, dbhs=c(5,12), units='English')
R> sttrs.hpizs = standingTreeIZs(lapply(sttrs@trees, 'horizontalPointIZ',
+                          angleGauge = aGauge))
R> sttrs.hpizs


Container object of class: standingTreeIZs
-------------------------------------------------------------

-------------------------------------------------------------
There are 10 inclusion zones in the population
Inclusion zones are of class: horizontalPointIZ
Units of measurement:  English
Summary of inclusion zone areas in square feet...
```

```
          Min.       1st Qu.       Median         Mean      3rd Qu.         Max.
     637.49000    918.43000   1266.30000   1624.90000   2112.00000   3163.90000
           var          SDev
   875037.26846    935.43427


Encapulating bounding box...
         min        max
x 27.423438 186.85999
y 38.961710 194.29711
```

```
R> sapply(sttrs.hpizs@iZones, class)
```

```
            tree.1                tree.2                tree.3                tree.4
"horizontalPointIZ" "horizontalPointIZ" "horizontalPointIZ" "horizontalPointIZ"
            tree.5                tree.6                tree.7                tree.8
"horizontalPointIZ" "horizontalPointIZ" "horizontalPointIZ" "horizontalPointIZ"
            tree.9               tree.10
"horizontalPointIZ" "horizontalPointIZ"
```

Finally, let's just look at how to extract a few of the components of the collection—remember, diameters are stored internally in the same units as length (feet in this example)...

```
R> sapply(sttrs@trees,function(x)x@dbh)*12 #dbh in inches
```

```
 tree.1  tree.2  tree.3  tree.4  tree.5  tree.6  tree.7  tree.8  tree.9 tree.10
   5.39    5.18    8.99    7.21    7.39    5.89   11.54   11.43    9.57    7.11
```

```
R> sapply(sttrs@trees,function(x)x@dbh)*aGauge@PRF #plot radii (ft)
```

```
 tree.1  tree.2  tree.3  tree.4   tree.5  tree.6  tree.7  tree.8  tree.9 tree.10
14.8225 14.2450 24.7225 19.8275 20.3225 16.1975 31.7350 31.4325 26.3175 19.5525
```

```
R> sttrs.hpizs@iZones$tree.1@circularPlot@radius #plot radius for tree 1
```

```
[1] 14.8225
```

```
R> sapply(sttrs.hpizs@iZones,function(x)x@circularPlot@radius) #plot radii (ft)
```

```
 tree.1  tree.2  tree.3  tree.4  tree.5  tree.6  tree.7  tree.8  tree.9 tree.10
14.8225 14.2450 24.7225 19.8275 20.3225 16.1975 31.7350 31.4325 26.3175 19.5525
```

The first line shows how to extract the DBH values from the "standingTrees" object. The second illustrates how to calculate the plot radii from this information. However, it has already been calculated and stored in the inclusion zone for each of the trees, recalculation is therefore unnecessary. This information is embedded a bit deep into the object as shown in the penultimate line, but can be easily accessed. Since the `iZones` slot is a simple list, we can use the `sapply` function to cycle over all the individual tree inclusion zones in the list and extract the `radius` slot from the "circularPlot" object stored within each tree's inclusion zone (see §14 for more information).

17.3.2 Plotting the object

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

```
R> plot(sttrs.hpizs, axes=TRUE)
```

17.4 Container coercion

It may be useful at times to coerce or change the container collection from its current class to something simpler. For example, it is not necessarily straightforward to work backwards and extract the collection of "Stem" objects from an "izContainer" object. To this end, the general `S4` `as` method has been adapted to container classes to coerce them to simpler forms. The following applies to objects that are a subclass of "izContainer".

For example, the "sampSurf" class stores an "izContainer" object in one of its slots. One of the constructors for the "sampSurf" objects automates the generation of the synthetic "Stem" population, and the only place it is accessible in the "sampSurf" object is within this "izContainer" slot.[4] But it might be useful to have the stem information available for some other purpose. To extract the stem information from a subclass object of "izContainer", apply the following steps...

```
R> stdTrees = as(sttrs.hpizs, 'standingTrees')
R> class(stdTrees)
```

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

---

[4]See *The "sampSurf" Class* vignette for more details.
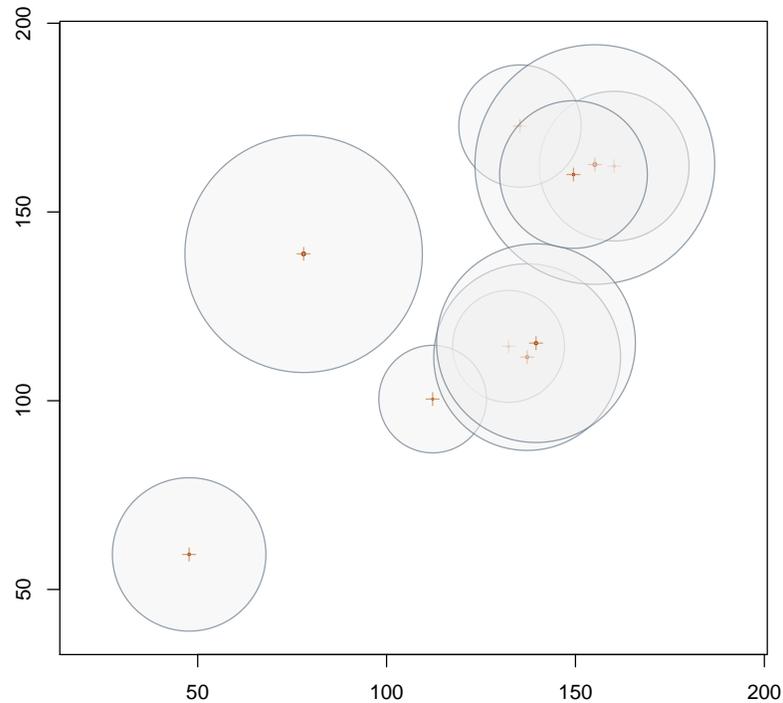
Figure 14: "standingTreeIZs" object under horizontal point sampling.

```
R> as(stdTrees, 'data.frame')
```

```
          species height   dbh topDiam solidType          x          y
1         hemlock  14.23  5.39    3.34       1.5 132.288697 114.388693
2              sm   7.62  5.18    2.76       3.7 112.188595 100.429216
3  shagbark hickory 13.84  8.99    6.81       5.7 137.206651 111.549651
4              rm   6.26  7.21    2.99       5.7 160.241366 162.141079
5  shagbark hickory  9.25  7.39    6.13       9.8  47.735707  59.281651
6              rm   8.46  5.89    3.68       2.4 135.316579 172.752471
7              BW  13.34 11.54    6.21       9.7 155.124993 162.566105
8              rm  10.09 11.43    7.90       1.4  78.045849 138.896218
9              BW  11.58  9.57    6.45       7.9 139.550466 115.245929
10             sm  13.37  7.11    5.07       5.7 149.476420 159.917420
```

The first line of code extracts all of the individual "standingTree" objects and bundles them into a

"standingTrees" container for easy handling, discarding all of the "InclusionZone" subclass information in the process. Then this collection is coerced to a "data.frame" in the last line, with diameters appropriately converted from feet to inches.

Once we know how this works, it can be simpler to just string the commands together as in the following "downLogIZs" object collection created earlier...

```
R> as( as(izs.sa, 'downLogs'), 'data.frame')
```

```
          species logLen buttDiam topDiam solidType        x          y
1     Picea glauca   3.23     8.23    5.89       6.1 47.325841 16.0621027
2               wp   1.68    11.08    8.64       2.9 29.782993 44.3609677
3 shagbark hickory   4.80    19.21    0.58       2.5 10.366026 49.9013624
4               wp   2.24     6.48    4.61       4.0 74.060887 37.4906937
5          hemlock   2.87    14.67   12.83       6.3 74.432935 41.8320043
6               rm   2.27    12.81    8.86       2.1 68.650088  7.1794063
   logAngle logAngle.D
1 1.3555693   77.668399
2 4.6409483  265.906749
3 4.1581264  238.243091
4 2.9574916  169.451788
5 4.7146564  270.129913
6 4.7368793  271.403191
```

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. 23, 28

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. 33, 41

J. H. Gove. *Monte Carlo Sampling Methods in* **sampSurf**, 2013. URL http://CRAN.R-project.org/package=sampSurf. **sampSurf** package vignette. 51

J. H. Gove. *The "InclusionZoneGrid" Class*, 2011. URL http://CRAN.R-project.org/package=sampSurf. **sampSurf** package vignette. 51

J. H. Gove and P. C. Van Deusen. On fixed-area plot sampling for downed coarse woody debris. *Forestry*, 84(2):109–117, 2011. 6, 9, 13, 14, 17

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. 19

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. 19

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. 29, 31

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. 39

T. G. Gregoire and H. T. Valentine. *Sampling strategies for natural resources and the environment.* Applied environmental statistics. Chapman & Hall/CRC, N.Y., 2008. 53

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*, 2013. (In review). 59, 61, 63

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. 23

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. 23