TissueDrawing
Technical details and regression checks

Jonathan Swinton

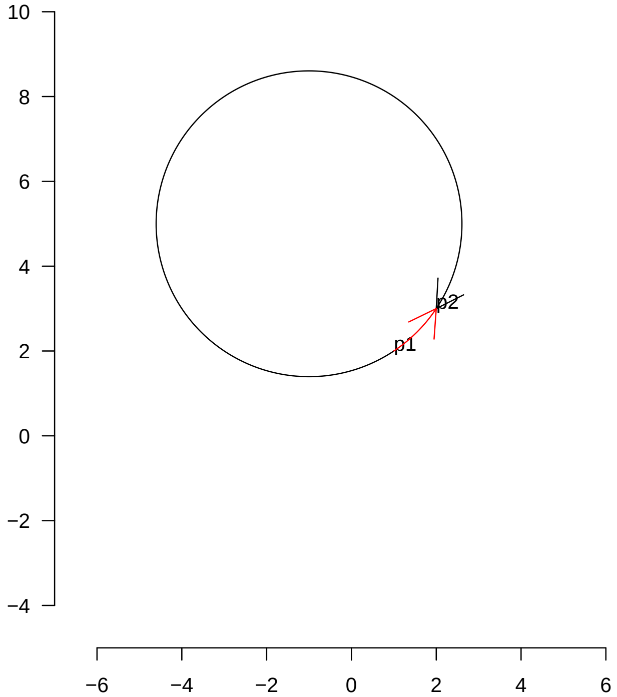
2nd August, 2009

Contents

1 The VDedgeSector object

A sector is a segment of a circle, defined by two points, together with the convention that a right-handed sector goes clockwise.

```
> nodeList <- list(p1 = matrix(1:2, ncol = 2), p2 = matrix(2:3,
+     ncol = 2))
> centre = c(-1, 5)
> fromTheta <- .point.xy.to.theta(nodeList[["p1"]], centre)
> toTheta <- .point.xy.to.theta(nodeList[["p2"]], centre)
> lh <- newEdgeSector(centre = c(-1, 5), hand = 1, from = "p1",
+     to = "p2", fromTheta = fromTheta, toTheta = toTheta, radius = sqrt(13))
> lh <- .normalise.sector(lh)
> VD1 <- new("TissueDrawing", nodeList = nodeList)
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-7, 7), c(-5, 10))
> grid.xaxis()
> grid.yaxis()
> PlotNodes(VD1)
> xy <- .edge.to.xy(lh)
> grid.lines(xy[, 1], xy[, 2], default.units = "native", arrow = arrow())
> lh@hand <- -1
> xy <- .edge.to.xy(lh)
> grid.lines(xy[, 1], xy[, 2], default.units = "native", arrow = arrow(),
+     gp = gpar(col = "red"))
```



We can also split VDedgeSectors

2 The TissueDrawing object

First we test constucting them from scratch.

```
> VD.nodeList <- list(p1 = matrix(1:2, ncol = 2), p2 = matrix(2:3,
+     ncol = 2), p3 = matrix(c(-1, 0), ncol = 2))
> sectorfromto <- function(sector, from, to, nodeList) {
+     sector@from <- from
+     sector@to <- to
+     from.point <- nodeList[[from]]
+     sector@fromTheta <- .point.xy.to.theta(from.point, sector@centre)
+     sector@toTheta <- .point.xy.to.theta(nodeList[[to]], sector@centre)
+     sector <- .normalise.sector(sector)
+ }
> centre = c(-1, 5)
> fromTheta <- .point.xy.to.theta(nodeList[["p1"]], centre)
> toTheta <- .point.xy.to.theta(nodeList[["p2"]], centre)
> lh <- newEdgeSector(centre = c(-1, 5), hand = 1, fromTheta = fromTheta,
+     toTheta = toTheta, radius = sqrt(13))
> lh <- sectorfromto(lh, "p1", "p2", VD.nodeList)
> centre = c(4, 0)
> fromTheta <- .point.xy.to.theta(nodeList[["p1"]], centre)
> toTheta <- .point.xy.to.theta(nodeList[["p2"]], centre)
> rh <- newEdgeSector(centre = c(4, 0), hand = 1, fromTheta = fromTheta,
+     toTheta = toTheta, radius = sqrt(13))
> el <- newEdgeLines(from = "p1", to = "p3", xy = matrix(c(1, 2,
+     -0.5, 0, -1, 0), ncol = 2, byrow = T))
> VD.edgeList <- list(`p1|p2|1` = sectorfromto(lh, "p1", "p2",
+     VD.nodeList), `p2|p1|1` = sectorfromto(lh, "p2", "p1", VD.nodeList),
+     `p1|p2|2` = sectorfromto(rh, "p1", "p2", VD.nodeList), `p2|p1|2` = sectorfromto(rh,
+         "p2", "p1", VD.nodeList), `p1|p3|3` = el, `p3|p1|3` = newEdgeLines(from = "p3",
+         to = "p1", xy = matrix(c(-1, 0, 1, 2), ncol = 2, byrow = T)))
> VD.faceList <- list(`100` = c("p1|p2|1", "-p1|p2|2"), `110` = c("p1|p2|2",
+     "p2|p1|1"), `010` = c("p2|p1|2", "-p2|p1|1"), `001` = c("p1|p3|3",
+     "p3|p1|3"), DarkMatter = c("-p3|p1|3", "-p1|p3|3", "-p2|p1|2",
+     "-p1|p2|1"))
> VD.setList <- list(`1` = c("p1|p2|1", "p2|p1|1"), `2` = c("p1|p2|2",
+     "p2|p1|2"), `3` = c("p1|p3|3", "p3|p1|3"))
> VD.faceSignature <- lapply(names(VD.faceList), function(x) {
+     x
+ })
> names(VD.faceSignature) <- names(VD.faceList)
> VD <- new("TissueDrawing", nodeList = VD.nodeList, edgeList = VD.edgeList,
+     setList = VD.setList, faceList = VD.faceList, faceSignature = VD.faceSignature)
> .validateDrawing(VD)
```

```
Validating a drawing on 3 sets......done
```

```
> VD
```

```
        from to         type npoints centre hand
p1|p2|1   p1 p2 VDedgeSector      NA   -1,5    1
```

```
p2|p1|1   p2 p1 VDedgeSector      NA   -1,5    1
p1|p2|2   p1 p2 VDedgeSector      NA    4,0    1
p2|p1|2   p2 p1 VDedgeSector      NA    4,0    1
p1|p3|3   p1 p3  VDedgeLines       3   <NA>   NA
p3|p1|3   p3 p1  VDedgeLines       2   <NA>   NA
   X1 X2
p1  1  2
p2  2  3
p3 -1  0
                                         faces
100                           p1|p2|1;-p1|p2|2
110                            p1|p2|2;p2|p1|1
010                           p2|p1|2;-p2|p1|1
001                            p1|p3|3;p3|p1|3
DarkMatter -p3|p1|3;-p1|p3|3;-p2|p1|2;-p1|p2|1
                  sig
100               100
110               110
010               010
001               001
DarkMatter DarkMatter
  paste.face..collapse.......
1             p1|p2|1;p2|p1|1
2             p1|p2|2;p2|p1|2
3             p1|p3|3;p3|p1|3
```

```
> .checkPointOnEdge(edge = VD@edgeList[["p1|p2|1"]], point.xy = VD@nodeList[["p1"]])
```

```
[1] TRUE
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-7, 7), c(-5, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VD)
> PlotSetBoundaries(VD)
> PlotNodes(VD)
```

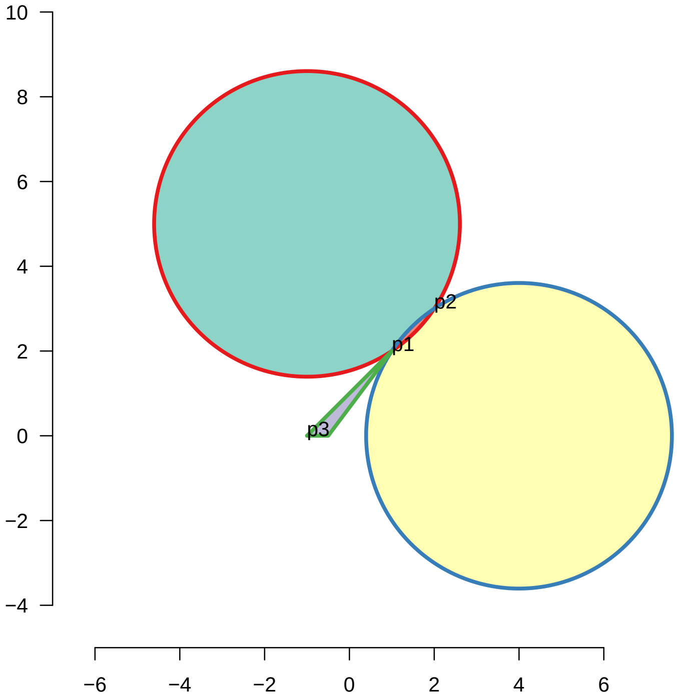


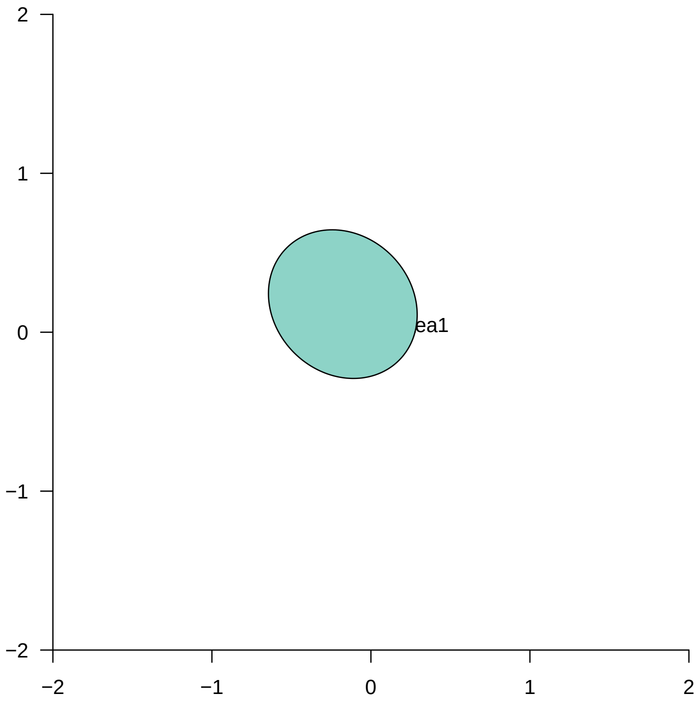
Figure 1: Constructing TissueDrawing objects from scratch

2.1 Ellipses

Ellipses could be coped with specially by finding roots of quartics, but don't bother and just generate them as polygons

```
> VE <- newTissueFromEllipse(f1 = c(0, 0), phi = pi/4, e = 0.5,
+     a = 0.5, Set = 1)
> .validateDrawing(VE)

Validating a drawing on 1 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-2, 2), c(-2, 2))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VE)
> PlotSetBoundaries(VE, gp = gpar(lwd = 2, col = c("red", "blue",
+     "green")))
> PlotNodes(VE)
```



```
> phi <- 0.8
> dex <- 1.7
> dey <- 2.5
> a <- 7.6
> e <- 0.9
> x0 <- c(-0.9, -5)
> VE <- list()
> dx <- 0.2
> VE[[1]] <- newTissueFromEllipse(x0 + c(0, 0), -phi, e, -a, Set = 1,
+     dx = dx)
> VE[[2]] <- newTissueFromEllipse(x0 + c(dex, 0), phi, e, a, Set = 2,
+     dx = dx)
> VE[[3]] <- newTissueFromEllipse(x0 + c(-dey, dey), -phi, e, -a,
+     Set = 3, dx = dx)
> VE[[4]] <- newTissueFromEllipse(x0 + c(dex + dey, dey), phi,
+     e, a, Set = 4, dx = dx)
> TM <- VE[[1]]
> TM2 <- addSetToDrawing(TM, VE[[2]], set2Name = paste("Set", 2,
+     sep = ""))
> TM3 <- addSetToDrawing(TM2, VE[[3]], set2Name = paste("Set",
+     3, sep = ""))
> TM4 <- addSetToDrawing(TM3, VE[[4]], set2Name = paste("Set",
+     4, sep = ""))
> .validateDrawing(TM4)
```

```
Validating a drawing on 4 sets......done
sig 0100 duplicated in faces 01-100;0100
sig 1000 duplicated in faces 1000;1000-1
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-10, 10), c(-8, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM4)
> PlotSetBoundaries(TM4, gp = gpar(lwd = 2, col = c("red", "blue",
+     "green", "yellow")))
> .PlotFaceNames.TissueDrawing(TM4)
```

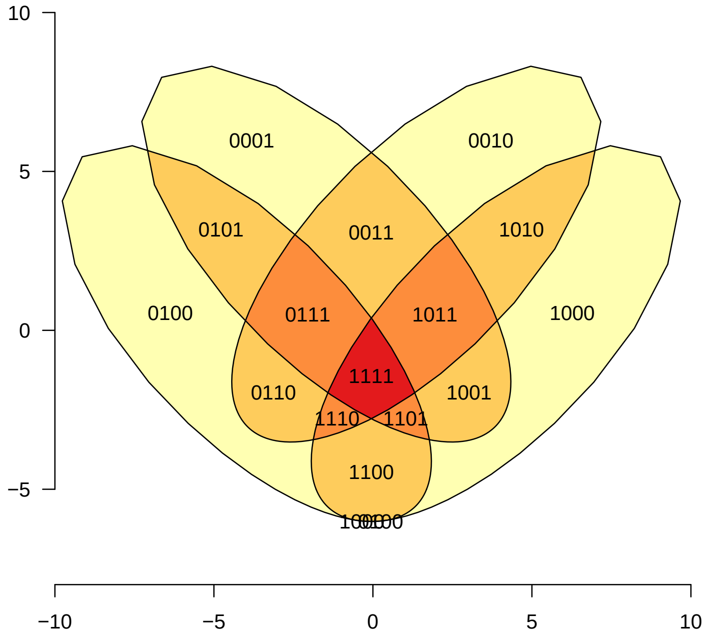


Figure 2: Constructing TissueDrawing objects from scratch

3 Injecting points and edges

We test injecting points

```
> p4 <- matrix(c(7, -2), ncol = 2)
> rownames(p4) <- "p4"
> VD4 <- injectPoint(drawing = VD, edgeName = "p2|p1|2", newPoint = p4)
> .validateDrawing(VD4)
```

```
Validating a drawing on 3 sets......done
```

```
> VD4
```

```
        from to         type npoints centre hand
p1|p2|1   p1 p2 VDedgeSector      NA   -1,5    1
p2|p1|1   p2 p1 VDedgeSector      NA   -1,5    1
p1|p2|2   p1 p2 VDedgeSector      NA    4,0    1
p1|p3|3   p1 p3  VDedgeLines       3   <NA>   NA
p3|p1|3   p3 p1  VDedgeLines       2   <NA>   NA
p2|p4|2   p2 p4 VDedgeSector      NA    4,0    1
p4|p1|2   p4 p1 VDedgeSector      NA    4,0    1
   X1 X2
p1  1  2
p2  2  3
p3 -1  0
p4  7 -2
                                                  faces
100                                    p1|p2|1;-p1|p2|2
110                                     p1|p2|2;p2|p1|1
010                            p2|p4|2;p4|p1|2;-p2|p1|1
001                                     p1|p3|3;p3|p1|3
DarkMatter -p3|p1|3;-p1|p3|3;-p4|p1|2;-p2|p4|2;-p1|p2|1
                  sig
100               100
110               110
010               010
001               001
DarkMatter DarkMatter
  paste.face..collapse.......
1             p1|p2|1;p2|p1|1
2     p1|p2|2;p2|p4|2;p4|p1|2
3             p1|p3|3;p3|p1|3
```

```
> p5 <- matrix(c(-3, 2), ncol = 2)
> rownames(p5) <- "p5"
> VD4 <- injectPoint(VD4, edgeName = "p1|p2|1", newPoint = p5)
> .validateDrawing(VD4)
```

```
Validating a drawing on 3 sets......done
```

```
> VD4
```

```
        from to         type npoints centre hand
p2|p1|1   p2 p1 VDedgeSector      NA   -1,5    1
p1|p2|2   p1 p2 VDedgeSector      NA    4,0    1
p1|p3|3   p1 p3  VDedgeLines       3   <NA>   NA
p3|p1|3   p3 p1  VDedgeLines       2   <NA>   NA
p2|p4|2   p2 p4 VDedgeSector      NA    4,0    1
p4|p1|2   p4 p1 VDedgeSector      NA    4,0    1
p1|p5|1   p1 p5 VDedgeSector      NA   -1,5    1
p5|p2|1   p5 p2 VDedgeSector      NA   -1,5    1
   X1 X2
p1  1  2
p2  2  3
p3 -1  0
p4  7 -2
p5 -3  2
                                                           faces
100                                     p1|p5|1;p5|p2|1;-p1|p2|2
110                                              p1|p2|2;p2|p1|1
010                                     p2|p4|2;p4|p1|2;-p2|p1|1
001                                              p1|p3|3;p3|p1|3
DarkMatter -p3|p1|3;-p1|p3|3;-p4|p1|2;-p2|p4|2;-p5|p2|1;-p1|p5|1
                  sig
100               100
110               110
010               010
001               001
DarkMatter DarkMatter
  paste.face..collapse.......
1     p1|p5|1;p5|p2|1;p2|p1|1
2     p1|p2|2;p2|p4|2;p4|p1|2
3             p1|p3|3;p3|p1|3
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-7, 7), c(-5, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VD4)
> PlotSetBoundaries(VD4, gp = gpar(lwd = 2, col = c("red", "blue",
+     "green")))
> PlotNodes(VD4)
```

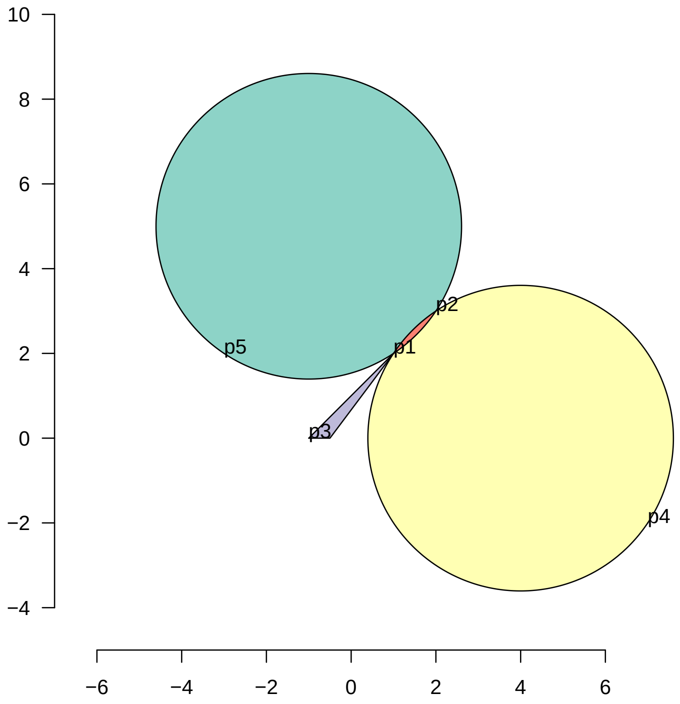


Figure 3: Injecting points

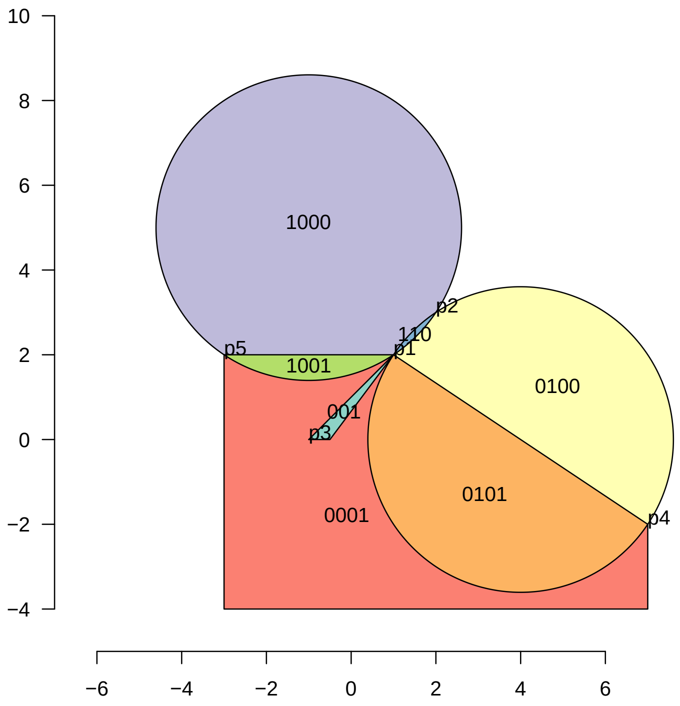
Then we try injecting single edges

```
> p1p4.line <- newEdgeLines(from = "p1", to = "p4", xy = matrix(c(1,
+     2, 7, -2), ncol = 2, byrow = T))
> p5p1.line <- newEdgeLines(from = "p5", to = "p1", xy = matrix(c(-3,
+     2, 1, 2), ncol = 2, byrow = T))
> p4p5.line <- newEdgeLines(from = "p4", to = "p5", xy = matrix(c(7,
+     -2, 7, -4, -3, -4, -3, 2), ncol = 2, byrow = T))
> VD6 <- VD4
> VD6@setList[["4"]] <- c("p4|p5|4", "p5|p1|4", "p1|p4|4")
> VD6@edgeList <- c(VD6@edgeList, list(`p1|p4|4` = p1p4.line, `p5|p1|4` = p5p1.line,
+     `p4|p5|4` = p4p5.line))
> VD6 <- injectEdge(drawing = VD6, newEdgeList = VD6@edgeList["p1|p4|4"],
+     set2Name = "4", addToList = FALSE)
```

```
> VD6 <- injectEdge(drawing = VD6, newEdgeList = list(`p5|p1|4` = p5p1.line),
+     set2Name = "4", addToList = FALSE)
> VD6 <- injectEdge(drawing = VD6, newEdgeList = list(`p4|p5|4` = p4p5.line),
+     set2Name = "4", addToList = FALSE)
> .is.face.within.set(drawing = VD6, faceName = "0101", setName = "2")

[1] TRUE

> .is.face.within.set(drawing = VD6, faceName = "1000", setName = "2")

[1] FALSE

> .is.face.within.set(drawing = VD6, faceName = "0001", setName = "2")

[1] FALSE

> VD6

        from to         type npoints centre hand
p2|p1|1   p2 p1 VDedgeSector      NA   -1,5    1
p1|p2|2   p1 p2 VDedgeSector      NA    4,0    1
p1|p3|3   p1 p3  VDedgeLines       3   <NA>   NA
p3|p1|3   p3 p1  VDedgeLines       2   <NA>   NA
p2|p4|2   p2 p4 VDedgeSector      NA    4,0    1
p4|p1|2   p4 p1 VDedgeSector      NA    4,0    1
p1|p5|1   p1 p5 VDedgeSector      NA   -1,5    1
p5|p2|1   p5 p2 VDedgeSector      NA   -1,5    1
p1|p4|4   p1 p4  VDedgeLines       2   <NA>   NA
p5|p1|4   p5 p1  VDedgeLines       2   <NA>   NA
p4|p5|4   p4 p5  VDedgeLines       4   <NA>   NA
   X1 X2
p1  1  2
p2  2  3
p3 -1  0
p4  7 -2
p5 -3  2
                                                 faces
110                                    p1|p2|2;p2|p1|1
001                                    p1|p3|3;p3|p1|3
DarkMatter                  -p2|p4|2;-p5|p2|1;-p4|p5|4
0101                                   p1|p4|4;p4|p1|2
0100                         -p2|p1|1;p2|p4|2;-p1|p4|4
1001                                   p5|p1|4;p1|p5|1
1000                         p5|p2|1;-p1|p2|2;-p5|p1|4
0001       p4|p5|4;-p1|p5|1;-p3|p1|3;-p1|p3|3;-p4|p1|2
                  sig
110               110
001               001
DarkMatter DarkMatter
0101             0101
0100             0100
```

```
1001              1001
1000              1000
0001              0001
  paste.face..collapse.......
1      p1|p5|1;p5|p2|1;p2|p1|1
2      p1|p2|2;p2|p4|2;p4|p1|2
3              p1|p3|3;p3|p1|3
4      p4|p5|4;p5|p1|4;p1|p4|4
```

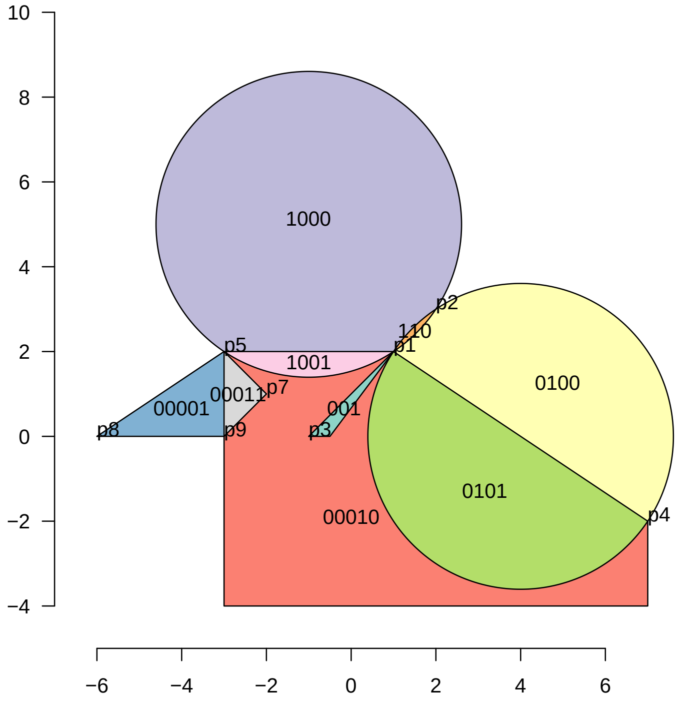
```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-7, 7), c(-5, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VD6)
> PlotSetBoundaries(VD6, gp = gpar(lwd = 2, col = c("red", "blue",
+     "green", "black")))
> .PlotFaceNames.TissueDrawing(VD6)
> PlotNodes(VD6)
```



And now injecting edges of multiple points

```
> VD8 <- VD6
> p7 <- matrix(c(-2, 1), ncol = 2)
> rownames(p7) <- "p7"
> VD8@nodeList[["p7"]] <- p7
> p8 <- matrix(c(-6, 0), ncol = 2)
> rownames(p8) <- "p8"
> VD8@nodeList[["p8"]] <- p8
> p9 <- matrix(c(-3, 0), ncol = 2)
> rownames(p9) <- "p9"
> VD8@nodeList[["p9"]] <- p9
> p5p7.line <- newEdgeLines(from = "p5", to = "p7", xy = matrix(c(-3,
+     2, -2, 1), ncol = 2, byrow = T))
> p7p9.line <- newEdgeLines(from = "p7", to = "p9", xy = matrix(c(-2,
+     1, -3, 0), ncol = 2, byrow = T))
> p9p8.line <- newEdgeLines(from = "p9", to = "p8", xy = matrix(c(-3,
+     0, -6, 0), ncol = 2, byrow = T))
> p8p5.line <- newEdgeLines(from = "p8", to = "p5", xy = matrix(c(-6,
+     0, -3, 2), ncol = 2, byrow = T))
> VD8@edgeList[["p5|p7|5"]] <- p5p7.line
> VD8@edgeList[["p7|p9|5"]] <- p7p9.line
> VD8@edgeList[["p9|p8|5"]] <- p9p8.line
> VD8@edgeList[["p8|p5|5"]] <- p8p5.line
> VD8@setList[["5"]] <- c("p5|p7|5", "p7|p9|5", "p9|p8|5", "p8|p5|5")
> VD8@edgeList[["p4|p5|4"]]@xy

     [,1] [,2]
[1,]    7   -2
[2,]    7   -4
[3,]   -3   -4
[4,]   -3    2

> VD8 <- injectPoint(drawing = VD8, edgeName = "p4|p5|4", newPoint = VD8@nodeList[["p9"]])
> VD8@edgeList[["p9|p5|4"]]@xy

     [,1] [,2]
[1,]   -3    0
[2,]   -3    2

> VD8@edgeList[["p4|p9|4"]]@xy

     [,1] [,2]
[1,]    7   -2
[2,]    7   -4
[3,]   -3   -4
[4,]   -3    0

> VD8 <- injectEdge(drawing = VD8, newEdgeList = VD8@edgeList[c("p5|p7|5",
+     "p7|p9|5")], set2Name = "5", addToList = FALSE)
> VD8 <- injectEdge(drawing = VD8, newEdgeList = VD8@edgeList[c("p9|p8|5",
+     "p8|p5|5")], set2Name = "5", addToList = FALSE)
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-7, 7), c(-5, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VD8)
> PlotSetBoundaries(VD8, gp = gpar(lwd = 2, col = c("red", "blue",
+     "green", "black", "orange")))
> .PlotFaceNames.TissueDrawing(VD8)
> PlotNodes(VD8)
```

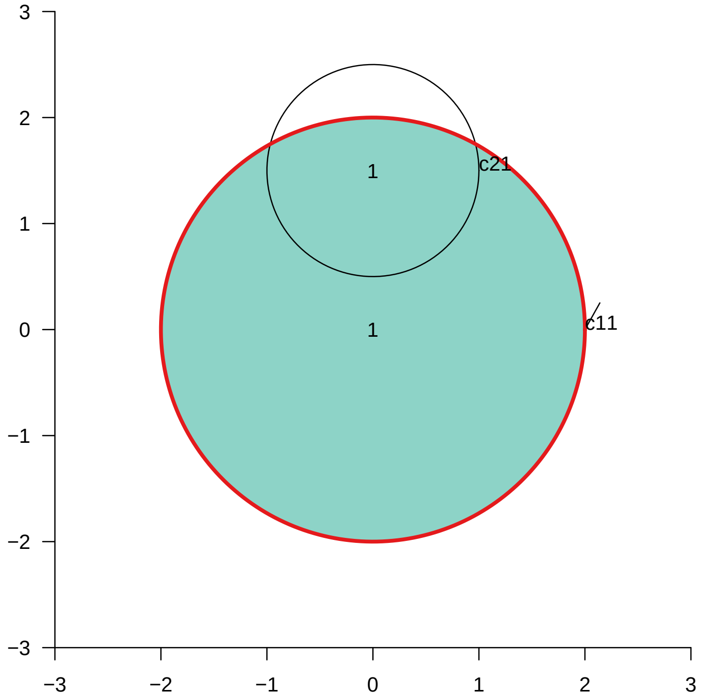


4 Making a simple drawing from a circle

```
> centre.xy <- c(0, 0)
> VDC1 <- newTissueFromCircle(centre.xy, radius = 2, Set = 1)
> VDC2 <- newTissueFromCircle(centre.xy + c(0, 1.5), radius = 1,
+     Set = 2)
> .validateDrawing(VDC2)
```

```
Validating a drawing on 1 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> xy <- .edge.to.xy(VDC1@edgeList[[1]])
> grid.lines(xy[, 1], xy[, 2], default.units = "native", arrow = arrow())
> PlotFaces(VDC1)
> PlotFaces(VDC2, gp = gpar(fill = "red"))
> PlotSetBoundaries(VDC1)
> .PlotFaceNames.TissueDrawing(VDC1)
> PlotNodes(VDC1)
> PlotNodes(VDC2)
> .PlotFaceNames.TissueDrawing(VDC2)
```



5 Circles

```
> r = 0.6
> d = 0.4
> angles <- pi/2 - c(0, 2 * pi/3, 4 * pi/3)
> x <- d * cos(angles)
> y <- d * sin(angles)
> r <- rep(r, 3)
> centres <- matrix(c(x, y), ncol = 2, byrow = FALSE)
> VDC1 <- newTissueFromCircle(centres[1, ], radius = r[1], Set = 1,
+     nodes = 3)
> VDC2 <- newTissueFromCircle(centres[2, ], radius = r[2], Set = 2,
+     nodes = 3)
> TM <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDC2, set2Name = "Set2")
> VDC3 <- newTissueFromCircle(centres[3, ], radius = r[3], Set = 3)
> TM <- addSetToDrawing(drawing1 = TM, drawing2 = VDC3, set2Name = "Set3")
> .validateDrawing(TM)
```

```
Validating a drawing on 3 sets......done
```

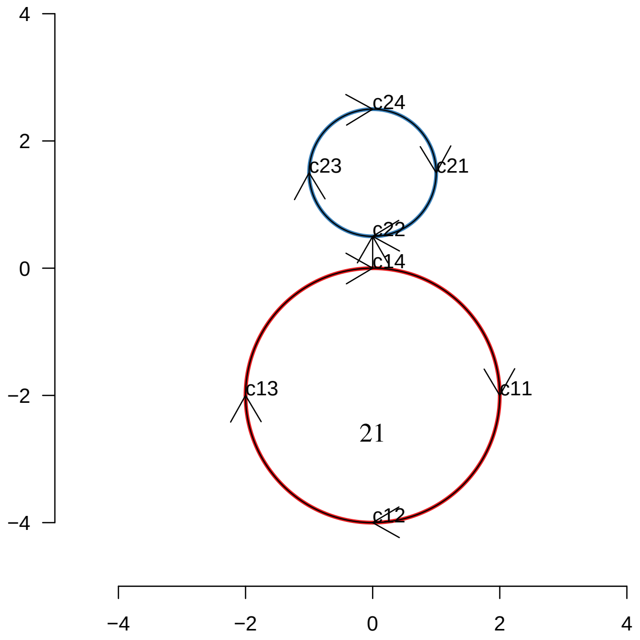
```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-1.5, 1.5), c(-1.5, 1.5))
> grid.xaxis()
> grid.yaxis()
> PlotSetBoundaries(TM)
> PlotNodes(TM)
> shoar(TM)

$`c13|c11|1`
lines[GRID.lines.665]

$`i24|c12|1`
lines[GRID.lines.666]

$`c11|i25|1`
lines[GRID.lines.667]

$`c21|c22|2`
lines[GRID.lines.668]

$`c23|i25|2`
lines[GRID.lines.669]

$`i25|c21|2`
lines[GRID.lines.670]

$`c12|i32|1`
lines[GRID.lines.671]

$`i32|c13|1`
lines[GRID.lines.672]

$`i25|i33|1`
lines[GRID.lines.673]

$`i33|i24|1`
lines[GRID.lines.674]

$`c22|i34|2`
lines[GRID.lines.675]

$`i34|i24|2`
lines[GRID.lines.676]

$`i24|i35|2`
lines[GRID.lines.677]

$`i35|c23|2`
lines[GRID.lines.678]

$`i33|c31|3`
lines[GRID.lines.679]

$`i32|i35|3`
lines[GRID.lines.680]

$`i35|i33|3`
lines[GRID.lines.681]
```

5.1 Non overlapping circles

```
> centre.xy <- c(0, -2)
> VDC1 <- newTissueFromCircle(centre.xy, radius = 2, Set = 1, nodes = 4)
> VDC2 <- newTissueFromCircle(centre.xy + c(0, 3.5), radius = 1,
+     Set = 2, nodes = 4)
> TN2 <- addSetToDrawing(VDC1, VDC2)
> VDC3 <- newTissueFromCircle(c(0, -0.5), radius = 1, Set = 3)
> .validateDrawing(TN2)
```

```
Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-5, 5), c(-5, 5))
> grid.xaxis()
> grid.yaxis()
> PlotSetBoundaries(TN2)
> PlotNodes(TN2)
> shoar(TN2)

$`c11|c12|1`
lines[GRID.lines.736]

$`c12|c13|1`
lines[GRID.lines.737]

$`c13|c14|1`
lines[GRID.lines.738]

$`c14|c11|1`
lines[GRID.lines.739]

$`c21|c22|2`
lines[GRID.lines.740]

$`c22|c23|2`
lines[GRID.lines.741]

$`c23|c24|2`
lines[GRID.lines.742]

$`c24|c21|2`
lines[GRID.lines.743]

$`c14|c22|invisible`
lines[GRID.lines.744]
```



5.2 Example of bug 528

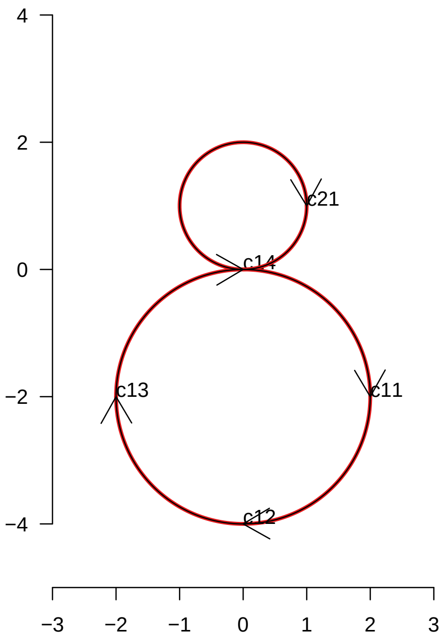
```
> centre.xy <- c(0, -2)
> VDC1b <- newTissueFromCircle(centre.xy, radius = 2, Set = 1,
+     nodes = 4)
> VDC2b <- newTissueFromCircle(centre.xy + c(0, 3), radius = 1,
+     Set = 2)
> TN2b <- (addSetToDrawing(VDC1b, VDC2b))
> TN2b
```

```
          from  to         type npoints centre hand
c11|c12|1  c11 c12 VDedgeSector      NA   0,-2    1
c12|c13|1  c12 c13 VDedgeSector      NA   0,-2    1
c13|c14|1  c13 c14 VDedgeSector      NA   0,-2    1
c14|c11|1  c14 c11 VDedgeSector      NA   0,-2    1
c21|c14|2  c21 c14 VDedgeSector      NA    0,1    1
c14|c21|2  c14 c21 VDedgeSector      NA    0,1    1
               X1 X2
c11  2.000000e+00 -2
c12 -3.673819e-16 -4
c13 -2.000000e+00 -2
c14  1.224606e-16  0
c21  1.000000e+00  1
                                                                       faces
10                                   c11|c12|1;c12|c13|1;c13|c14|1;c14|c11|1
DarkMatter -c13|c14|1;-c12|c13|1;-c11|c12|1;-c14|c11|1;-c21|c14|2;-c14|c21|2
01                                                       c14|c21|2;c21|c14|2
                  sig
10                 10
DarkMatter DarkMatter
01                 01
                 paste.face..collapse.......
Set1 c11|c12|1;c12|c13|1;c13|c14|1;c14|c11|1
Set2                     c21|c14|2;c14|c21|2
```

```
> (.validateDrawing(TN2b))
```

```
Validating a drawing on 2 sets......done
NULL
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-5, 5))
> grid.xaxis()
> grid.yaxis()
> PlotSetBoundaries(VDC1b)
> PlotNodes(VDC1b)
> PlotSetBoundaries(VDC2b)
> PlotNodes(VDC2b)
> shoar(VDC1b)

$`c11|c12|1`
lines[GRID.lines.774]

$`c12|c13|1`
lines[GRID.lines.775]

$`c13|c14|1`
lines[GRID.lines.776]

$`c14|c11|1`
lines[GRID.lines.777]

> shoar(VDC2b)

$`c21|c21|2`
lines[GRID.lines.778]
```

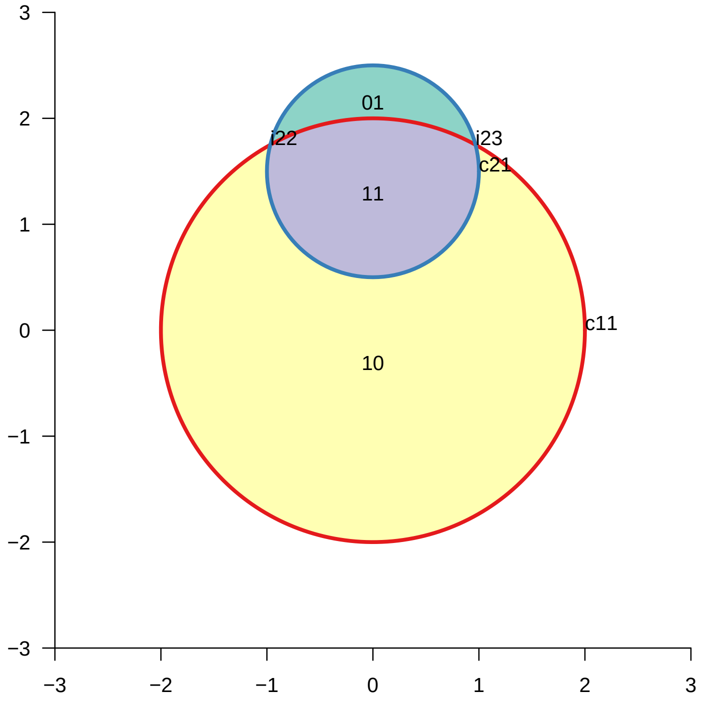


6 Check for the intersection of two edges

```
> centre.xy <- c(0, 0)
> VDC1 <- newTissueFromCircle(centre.xy, radius = 2, Set = 1)
> renameFaces(VDC1, oldName = .faceNames(VDC1, onlyVisible = TRUE),
+     "1")

          from  to         type npoints centre hand
c11|c11|1  c11 c11 VDedgeSector      NA    0,0    1
    X1            X2
c11  2 -4.898425e-16
                faces
1           c11|c11|1
DarkMatter -c11|c11|1
                  sig
1                   1
DarkMatter DarkMatter
     paste.face..collapse.......
Set1                   c11|c11|1

> VDC2 <- newTissueFromCircle(centre.xy + c(0, 1.5), radius = 1,
+     Set = 2)
> edge1 <- VDC1@edgeList[[1]]
> edge2 <- VDC2@edgeList[[1]]
> .findIntersection(edge1, edge2)

           [,1] [,2]
[1,] -0.9682458 1.75
[2,]  0.9682458 1.75

> edge1 <- VD8@edgeList[["p1|p4|4"]]
> edge2 <- VDC2@edgeList[[1]]
> .findIntersection(edge1, edge2)

     [,1] [,2]

> edge1 <- VD8@edgeList[["p1|p4|4"]]
> edge2 <- VD8@edgeList[["p2|p4|2"]]
> .findIntersection(edge1, edge2)

     [,1] [,2]
[1,]    7   -2

> .find.point.within.face(drawing = VD8, faceName = "1001")

         [,1]     [,2]
centroid   -1 1.755971

> .is.point.within.face(VD8, "DarkMatter", p7)

[1] FALSE
```

```
> .is.point.within.face(VD8, "DarkMatter", matrix(c(-100, 100),
+     ncol = 2))

[1] TRUE

> edge1 <- VD8@edgeList[["p1|p4|4"]]
> edge2 <- VD8@edgeList[["p1|p3|3"]]
> .findIntersection(edge1, edge2)

    [,1] [,2]
ict    1    2

> drawing1 <- VDC1
> drawing2 <- VDC2
> VM <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDC2, set2Name = "Set2")
> .validateDrawing(VM)

Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VM)
> PlotSetBoundaries(VM)
> .PlotFaceNames.TissueDrawing(VM)
> PlotNodes(VM)
```



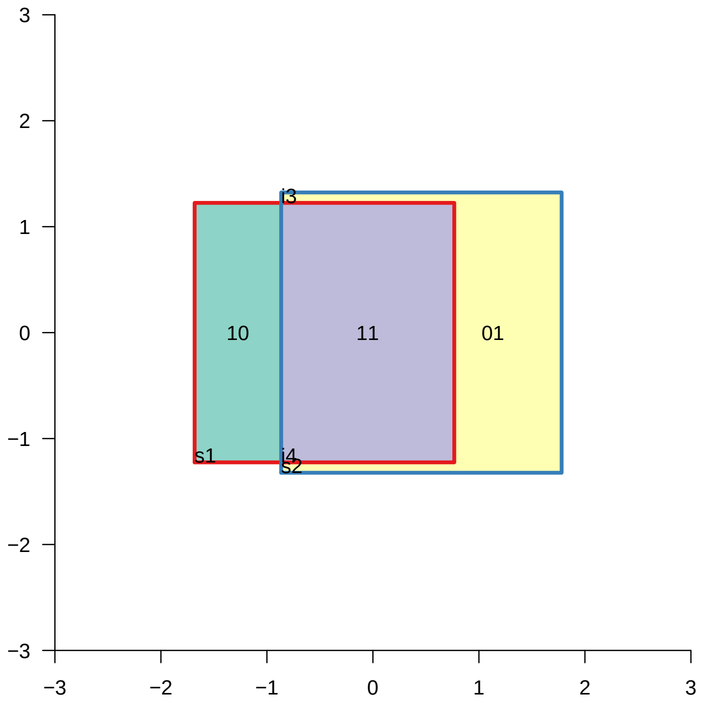
7 addSetToDrawing two polygons

```
> d <- 1
> s1 <- 0.7
> s2 <- 0.6
> d <- 0.9146274
> s1 <- 2.44949
> s2 <- 2.645751
> l1 <- -d/2 - s1/2
> l2 <- d/2 - s2/2
> r1 <- -d/2 + s1/2
> r2 <- d/2 + s2/2
> poly.1 <- matrix(c(l1, -s1/2, l1, s1/2, r1, s1/2, r1, -s1/2),
+     ncol = 2, byrow = TRUE)
```

```
> rownames(poly.1) <- paste("s", 1:4, sep = "")
> poly.2 <- matrix(c(l2, -s2/2, l2, s2/2, r2, s2/2, r2, -s2/2),
+     ncol = 2, byrow = TRUE)
> rownames(poly.2) <- paste("s", 2:5, sep = "")
> VDP1 <- newTissueFromPolygon(points.xy = poly.1, Set = 1)
> VDP2 <- newTissueFromPolygon(points.xy = poly.2, Set = 2)
> TM <- addSetToDrawing(drawing1 = VDP1, drawing2 = VDP2, set2Name = "Set2")
> .validateDrawing(TM)
```

```
Validating a drawing on 2 sets......done
```

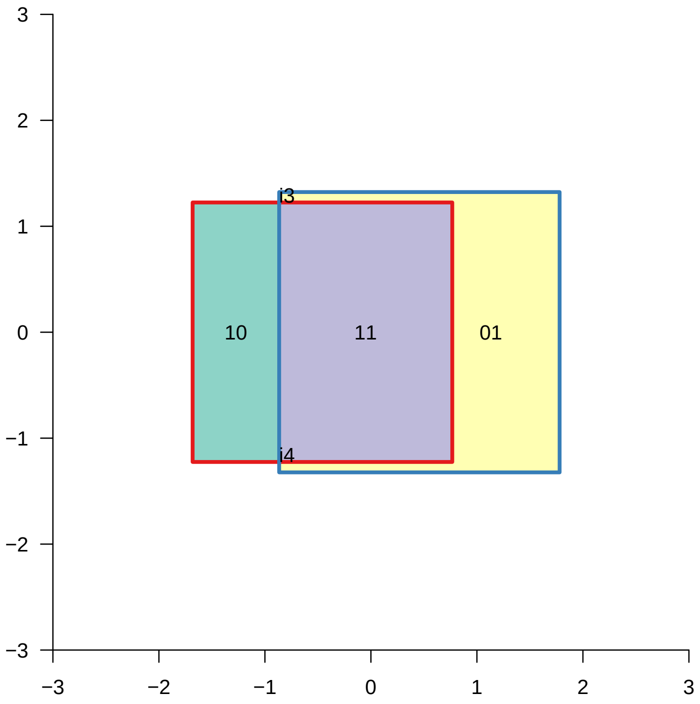
```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM)
> PlotSetBoundaries(TM)
> .PlotFaceNames.TissueDrawing(TM)
> PlotNodes(TM)
```



```
> TMR <- remove.nonintersectionpoints(drawing = TM)
> .validateDrawing(TMR)
```

```
Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TMR)
> PlotSetBoundaries(TMR)
> .PlotFaceNames.TissueDrawing(TMR)
> PlotNodes(TMR)
```

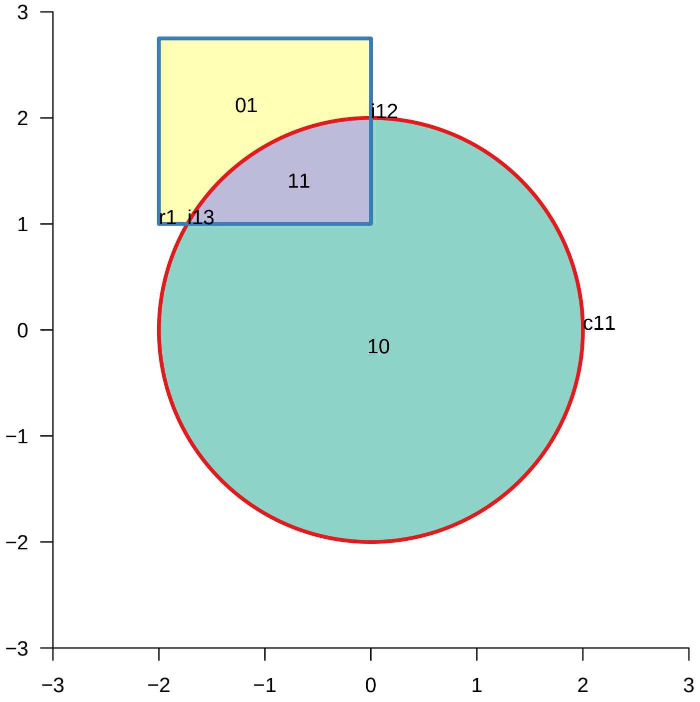


8 addSetToDrawing a polygon and a circle

```
> centre.xy <- c(0, 0)
> poly.xy <- matrix(c(-2, 1, -2, 2.75, 0, 2.75, 0, 1), byrow = TRUE,
+     ncol = 2, dimnames = list(paste("r", 1:4, sep = "")))
> VDP1 <- newTissueFromPolygon(points.xy = poly.xy, Set = 2)
> poly2.xy <- -poly.xy
> rownames(poly2.xy) <- sub("r", "rx", rownames(poly2.xy))
> VDP2 <- newTissueFromPolygon(points.xy = poly2.xy, Set = 3)
> drawing1 <- VDC1
> drawing2 <- VDP1
> VDCPM <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDP1, set2Name = "Set2")
> .validateDrawing(VDCPM)
```

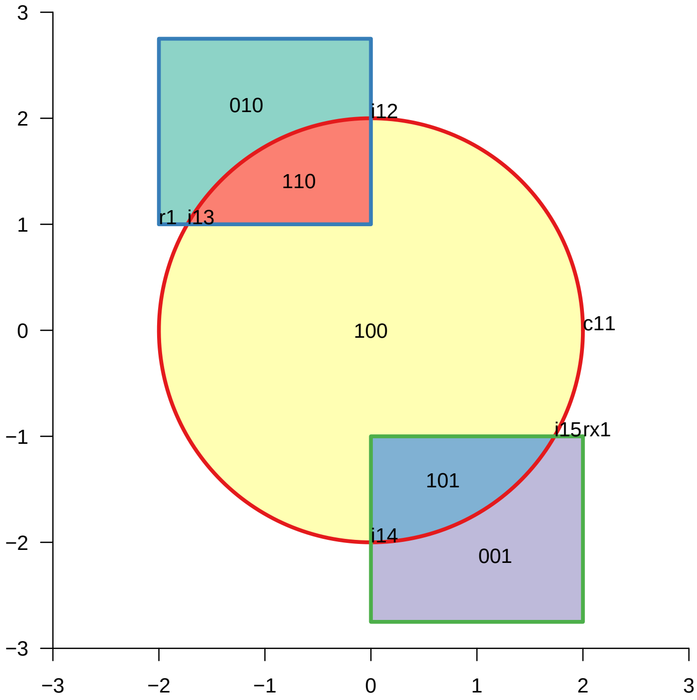
```
Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VDCPM)
> PlotSetBoundaries(VDCPM)
> .PlotFaceNames.TissueDrawing(VDCPM)
> PlotNodes(VDCPM)
```



```
> VDCPM2 <- addSetToDrawing(drawing1 = VDCPM, drawing2 = VDP2,
+     set2Name = "Set3")
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VDCPM2)
> PlotSetBoundaries(VDCPM2)
> .PlotFaceNames.TissueDrawing(VDCPM2)
> PlotNodes(VDCPM2)
```



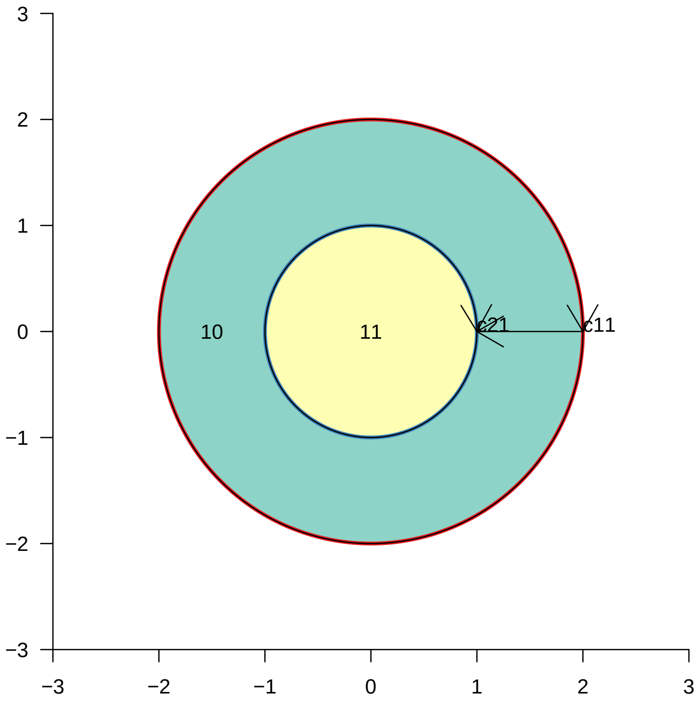
9 Invisible edges

```
> centre.xy <- c(0, 0)
> VDC3 <- newTissueFromCircle(centre.xy, radius = 2, Set = 1)
> VDC4 <- newTissueFromCircle(centre.xy, radius = 1, Set = 2)
> VDI <- addSetToDrawing(drawing1 = VDC3, drawing2 = VDC4, set2Name = "Set2")
> .validateDrawing(VDI)
```

```
Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VDI)
> PlotSetBoundaries(VDI)
> .PlotFaceNames.TissueDrawing(VDI)
> PlotNodes(VDI)
> shoar(VDI)
```

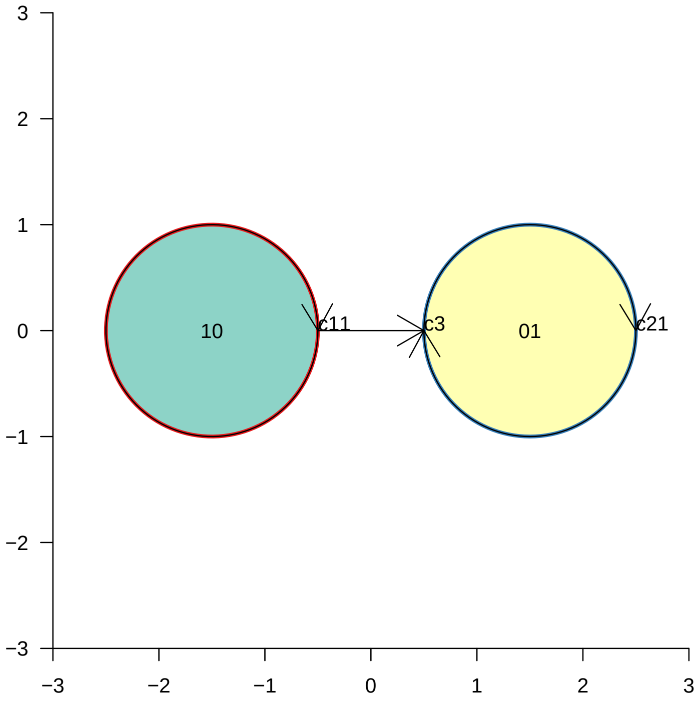
```
$`c11|c11|1`
lines[GRID.lines.1063]

$`c21|c21|2`
lines[GRID.lines.1064]

$`c11|c21|invisible`
lines[GRID.lines.1065]
```



The code only attemtps to inject invisible edges between known points, so we have to give the algorithm a hint by inserting such known points in the right place

```
> centre.xy <- c(-1.5, 0)
> VDC5 <- newTissueFromCircle(centre.xy, radius = 1, Set = 1)
> VDC6 <- newTissueFromCircle(centre.xy + c(3, 0), radius = 1,
+     Set = 2)
> VDC6 <- injectPoint(VDC6, "c21|c21|2", newPoint = matrix(c(0.5,
+     0), ncol = 2, dimnames = list("c3")))
> VDO <- addSetToDrawing(drawing1 = VDC5, drawing2 = VDC6, set2Name = "Set2")
> .validateDrawing(VDO)

Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VDO)
> PlotSetBoundaries(VDO)
> .PlotFaceNames.TissueDrawing(VDO)
> PlotNodes(VDO)
> lapply(VDO@edgeList, function(lh) {
+     xy <- .edge.to.xy(lh)
+     grid.lines(xy[, 1], xy[, 2], default.units = "native", arrow = arrow())
+ })

$`c11|c11|1`
lines[GRID.lines.1096]

$`c21|c3|2`
lines[GRID.lines.1097]

$`c3|c21|2`
lines[GRID.lines.1098]

$`c11|c3|invisible`
lines[GRID.lines.1099]
```

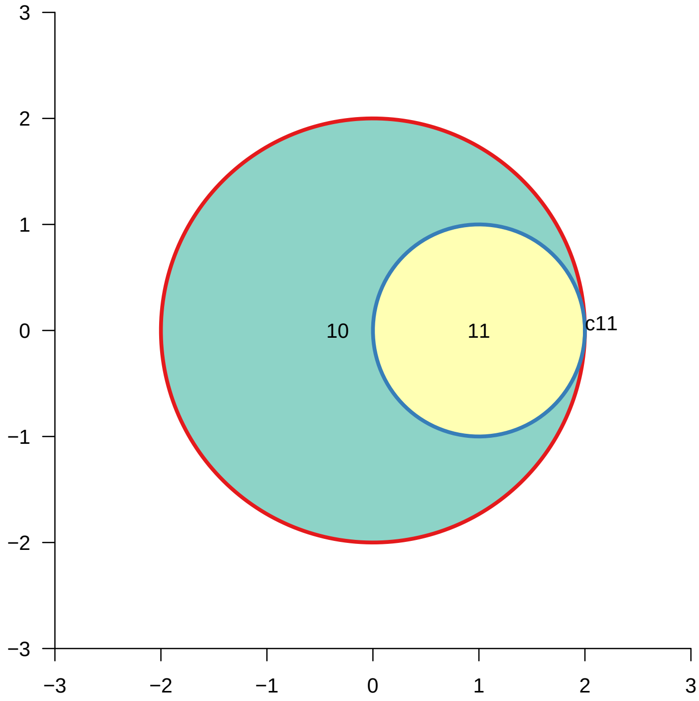


10 Tangents

```
> centre.xy <- c(0, 0)
> VDC7 <- newTissueFromCircle(centre.xy, radius = 2, Set = 1)
> VDC8 <- newTissueFromCircle(centre.xy + c(1, 0), radius = 1,
+     Set = 2)
> VDT <- addSetToDrawing(drawing1 = VDC7, drawing2 = VDC8, set2Name = "Set2")
> .validateDrawing(VDT)
```

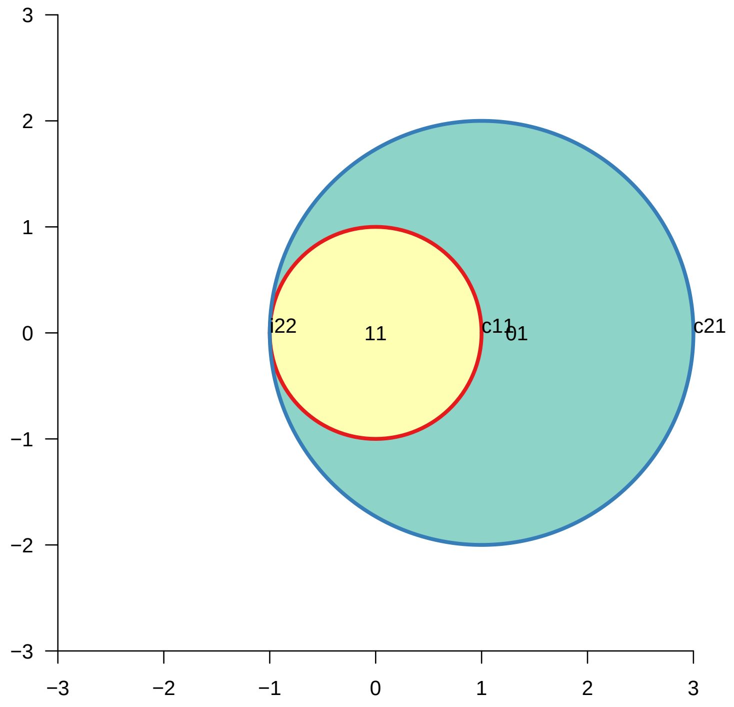
```
Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VDT)
> PlotSetBoundaries(VDT)
> .PlotFaceNames.TissueDrawing(VDT)
> PlotNodes(VDT)
```



```
> centre.xy <- c(0, 0)
> VDC9 <- newTissueFromCircle(centre.xy, radius = 1, Set = 1)
> VDC10 <- newTissueFromCircle(centre.xy + c(1, 0), radius = 2,
+     Set = 2)
```

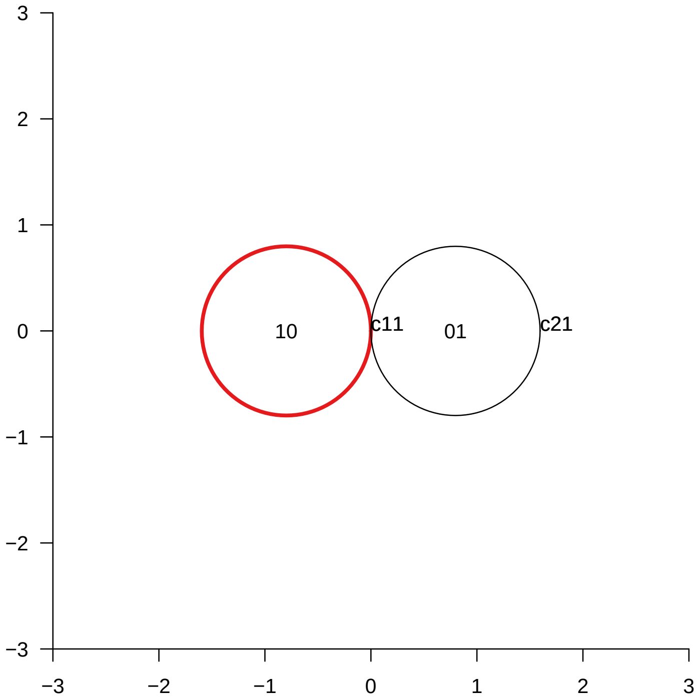
```
> VDT2 <- addSetToDrawing(drawing1 = VDC9, drawing2 = VDC10, set2Name = "Set2")
> .validateDrawing(VDT2)

Validating a drawing on 2 sets......done


> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(VDT2)
> PlotSetBoundaries(VDT2)
> .PlotFaceNames.TissueDrawing(VDT2)
> PlotNodes(VDT2)
```



```
> r1 = 0.797884560802865
> r2 = 0.797884560802865
> d = 1.59576912160573
> r = c(r1, r2)
> centres <- matrix(c(-d/2, 0, d/2, 0), ncol = 2, byrow = TRUE)
> VDC1 <- newTissueFromCircle(centres[1, ], radius = r[1], Set = 1)
> VDC2 <- newTissueFromCircle(centres[2, ], radius = r[2], Set = 2)
> VDT <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDC2, set2Name = "Set2")
> .validateDrawing(VDT)
```

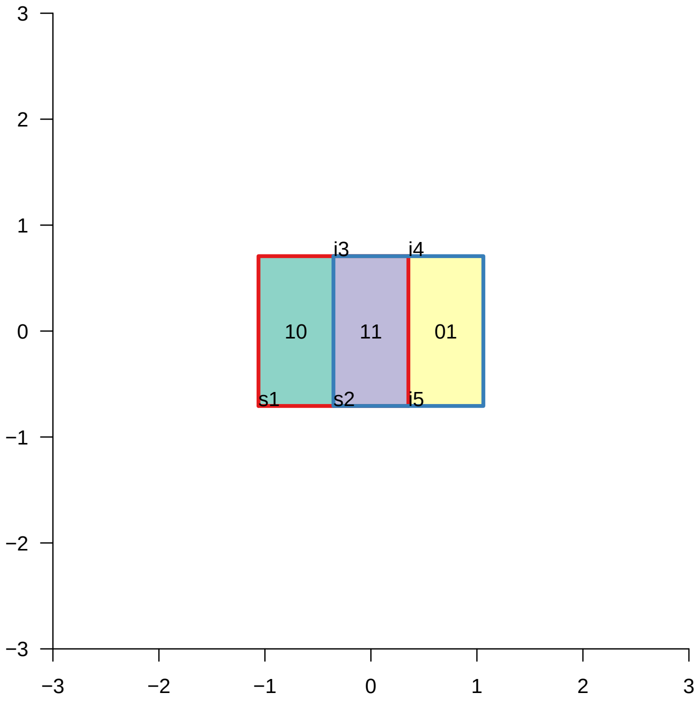
```
Validating a drawing on 2 sets......done
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotSetBoundaries(VDC1)
> PlotSetBoundaries(VDC2, gp = gpar(col = "red"))
> PlotNodes(VDC1)
> PlotNodes(VDC2)
> .PlotFaceNames.TissueDrawing(VDT)
> PlotNodes(VDT)
```



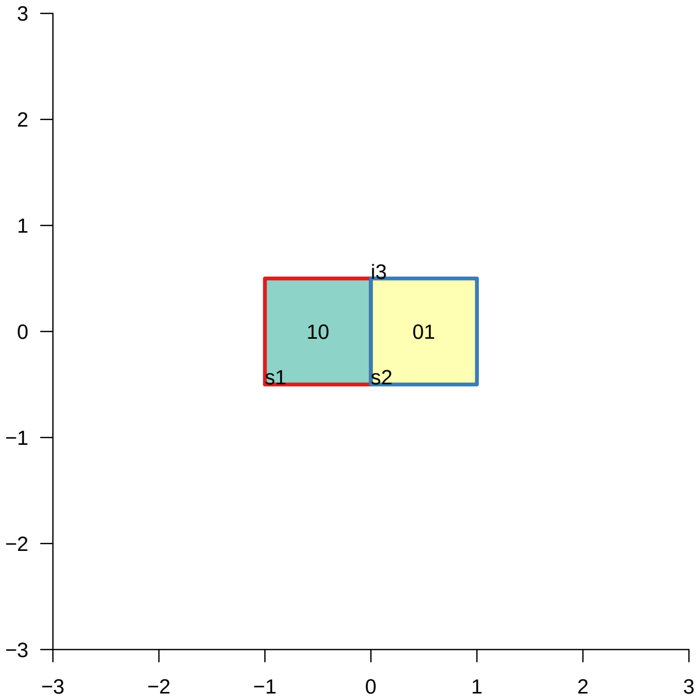
```
> l1 <- -1.06066
> r1 <- 0.3535534
> l2 <- -0.3535534
> r2 <- 1.06066
> s1 <- 1.414214
> s2 <- 1.414214
> poly.1 <- matrix(c(l1, -s1/2, l1, s1/2, r1, s1/2, r1, -s1/2),
+     ncol = 2, byrow = TRUE)
> rownames(poly.1) <- paste("s", 1:4, sep = "")
> poly.2 <- matrix(c(l2, -s2/2, l2, s2/2, r2, s2/2, r2, -s2/2),
+     ncol = 2, byrow = TRUE)
```

```
> rownames(poly.2) <- paste("s", 2:5, sep = "")
> VDP1 <- newTissueFromPolygon(points.xy = poly.1, Set = 1)
> VDP2 <- newTissueFromPolygon(points.xy = poly.2, Set = 2)
> TM <- addSetToDrawing(drawing1 = VDP1, drawing2 = VDP2, set2Name = "Set2")

> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM)
> PlotSetBoundaries(TM)
> .PlotFaceNames.TissueDrawing(TM)
> PlotNodes(TM)
```



```
> d <- 1
> s1 <- 1
> s2 <- 1
> l1 <- -d/2 - s1/2
> l2 <- d/2 - s2/2
> r1 <- -d/2 + s1/2
> r2 <- d/2 + s2/2
> poly.1 <- matrix(c(l1, -s1/2, l1, s1/2, r1, s1/2, r1, -s1/2),
+     ncol = 2, byrow = TRUE)
> rownames(poly.1) <- paste("s", 1:4, sep = "")
> poly.2 <- matrix(c(l2, -s2/2, l2, s2/2, r2, s2/2, r2, -s2/2),
+     ncol = 2, byrow = TRUE)
> rownames(poly.2) <- paste("s", 2:5, sep = "")
> VDP3 <- newTissueFromPolygon(points.xy = poly.1, Set = 1)
> VDP4 <- newTissueFromPolygon(points.xy = poly.2, Set = 2)
> TM3 <- addSetToDrawing(drawing1 = VDP3, drawing2 = VDP4, set2Name = "Set2")
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM3)
> PlotSetBoundaries(TM3)
> .PlotFaceNames.TissueDrawing(TM3)
> PlotNodes(TM3)
```

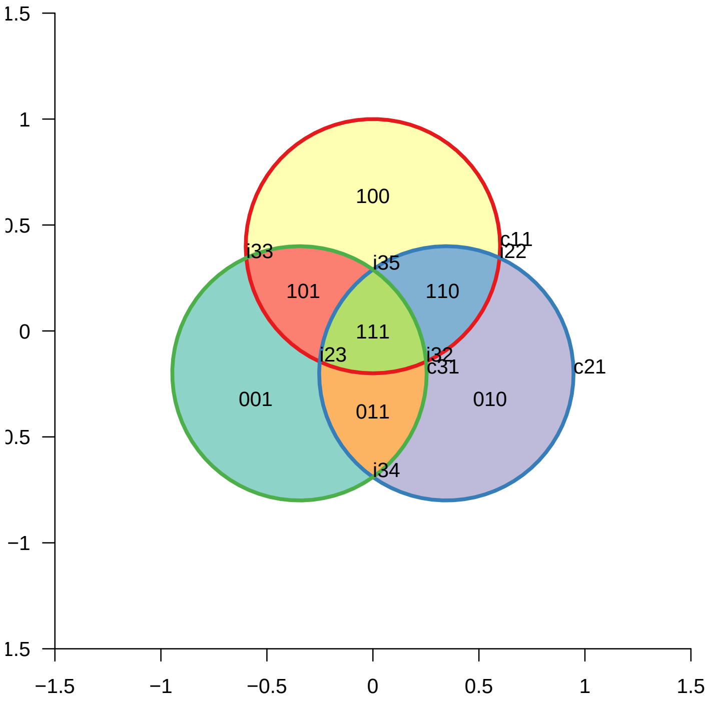


11 Three circles

11.1 Canonical

```
> r <- 0.6
> d <- 0.4
> angles <- pi/2 - c(0, 2 * pi/3, 4 * pi/3)
> x <- d * cos(angles)
> y <- d * sin(angles)
> r <- rep(r, 3)
> centres <- matrix(c(x, y), ncol = 2, byrow = FALSE)
> VDC1 <- newTissueFromCircle(centres[1, ], radius = r[1], Set = 1)
> VDC2 <- newTissueFromCircle(centres[2, ], radius = r[2], Set = 2)
> TM3 <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDC2, set2Name = "Set2")
> VDC3 <- newTissueFromCircle(centres[3, ], radius = r[3], Set = 3)
> TM3 <- addSetToDrawing(drawing1 = TM3, drawing2 = VDC3, set2Name = "Set3")
```

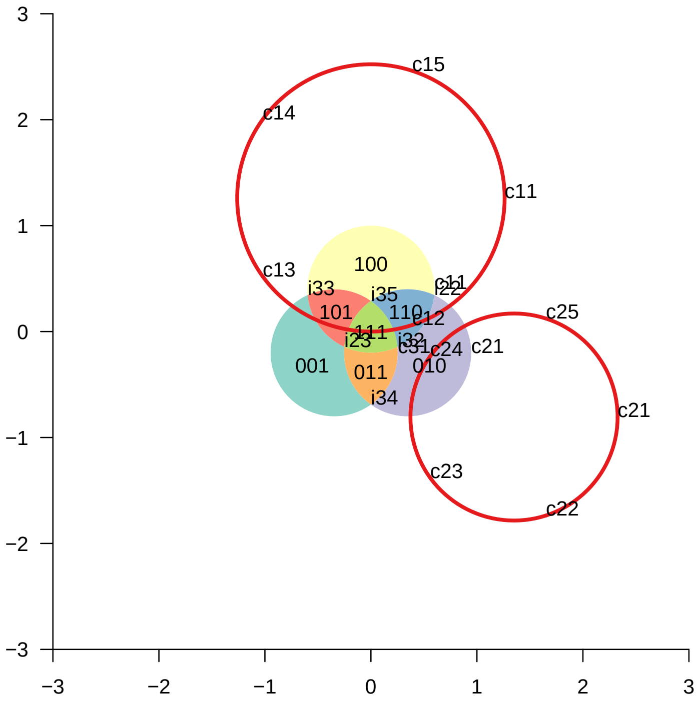
```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-1.5, 1.5), c(-1.5, 1.5))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM3)
> PlotSetBoundaries(TM3)
> .PlotFaceNames.TissueDrawing(TM3)
> PlotNodes(TM3)
```



12 One tangent point

```
> r <- c(1.261566, 0.977205, 1.492705)
> x <- c(0, 1.350138, -1.086542)
> y <- c(1.2615663, -0.8066661, -0.4028718)
> centres <- matrix(c(x, y), ncol = 2, byrow = FALSE)
> VDC1 <- newTissueFromCircle(centres[1, ], radius = r[1], Set = 1,
+     nodes = 5)
> VDC2 <- newTissueFromCircle(centres[2, ], radius = r[2], Set = 2,
+     nodes = 5)
> TM <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDC2, set2Name = "Set2")
> VDC3 <- newTissueFromCircle(centres[3, ], radius = r[3], Set = 3)
> TM <- addSetToDrawing(drawing1 = TM, drawing2 = VDC3, set2Name = "Set3")
```

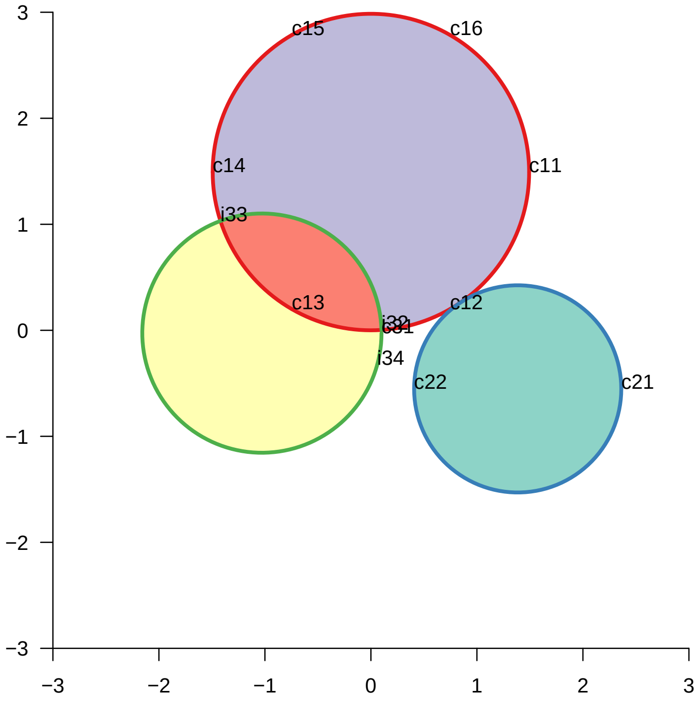
```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM3)
> PlotSetBoundaries(VDC1)
> PlotSetBoundaries(VDC2)
> PlotNodes(VDC1)
> PlotNodes(VDC2)
> .PlotFaceNames.TissueDrawing(TM3)
> PlotNodes(TM3)
```



13 Three circles

```
> r <- c(1.492705, 0.977205, 1.128379)
> x <- c(0, 1.384666, -1.028597)
> y <- c(1.49270533, -0.55257134, -0.02662434)
> centres <- matrix(c(x, y), ncol = 2, byrow = FALSE)
> VDC1 <- newTissueFromCircle(centres[1, ], radius = r[1], Set = 1,
+     nodes = 6)
> VDC2 <- newTissueFromCircle(centres[2, ], radius = r[2], Set = 2,
+     nodes = 2)
> TM <- addSetToDrawing(drawing1 = VDC1, drawing2 = VDC2, set2Name = "Set2")
> VDC3 <- newTissueFromCircle(centres[3, ], radius = r[3], Set = 3)
> TM3 <- addSetToDrawing(drawing1 = TM, drawing2 = VDC3, set2Name = "Set3")
> TV3 <- .merge.faces.invisibly.split(TM3)
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-3, 3), c(-3, 3))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TV3)
> PlotSetBoundaries(TV3)
> PlotNodes(TM3)
```

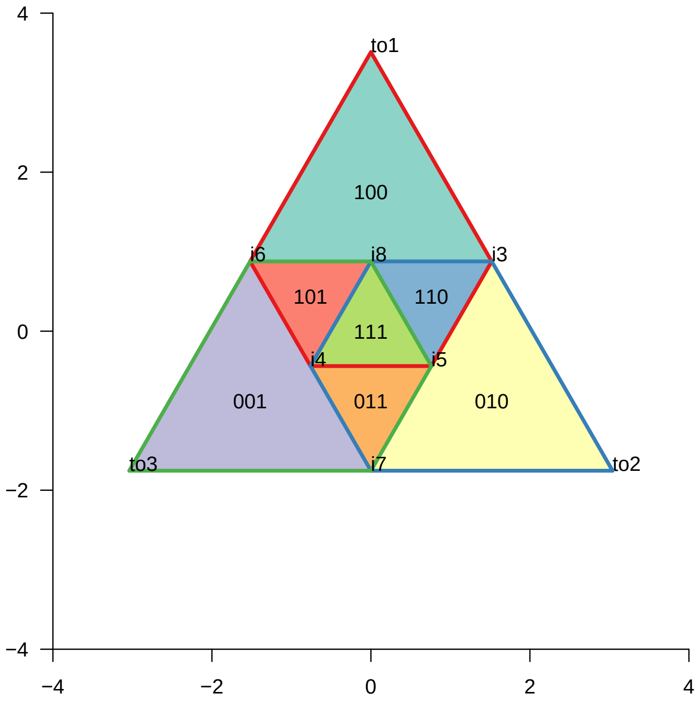


14 Triangles

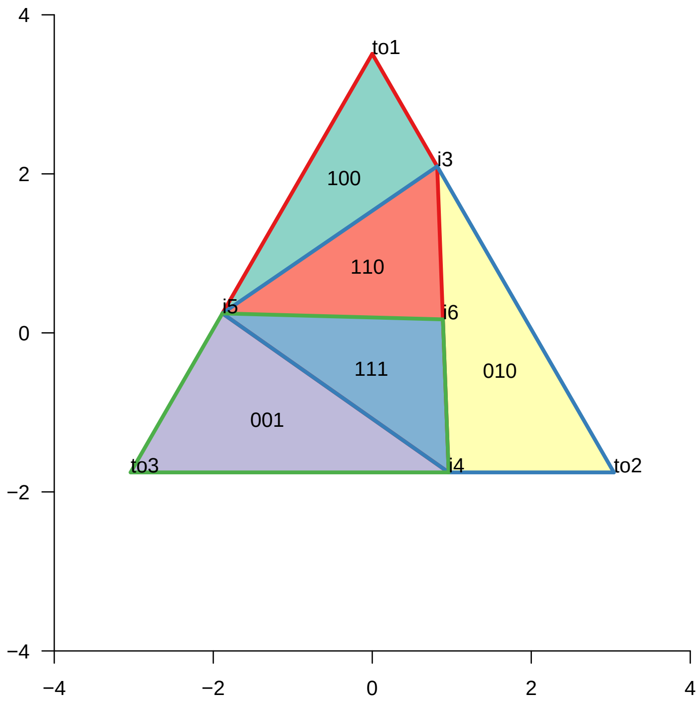
```
> .inscribetriangle.feasible <- function(wghts) {
+     w0 <- 1 - sum(wghts)
+     stopifnot(all(wghts <= 1) & all(wghts >= 0) & w0 >= 0)
+     wa <- wghts[1]
+     wb <- wghts[2]
+     wc <- wghts[3]
+     Delta <- w0^2 - 4 * wa * wb * wc
+     return(Delta >= 0)
+ }
> .inscribetriangle.compute <- function(wghts) {
+     wa <- wghts[1]
+     wb <- wghts[2]
+     wc <- wghts[3]
+     stopifnot(.inscribetriangle.feasible(wghts))
+     pa <- (1 - wc)
+     pb <- (wb + wc - wa - 1)
+     pc <- wa * (1 - wb)
+     sc <- if (wa > 0) {
+         (-pb - sqrt(pb^2 - 4 * pa * pc))/(2 * pa)
+     }
+     else if (wb + wc < 1) {
+         (1 - wb - wc)/(1 - wc)
+     }
+     else {
+         0
+     }
+     sb <- if (sc > 0) {
+         1 - wa/sc
+     }
+     else {
+         wc/(1 - wb)
+     }
+     sa <- wb/(1 - sc)
+     c(sc, sa, sb)
+ }
> .inscribetriangle.inscribe <- function(xy, wghts) {
+     scalef <- NA
+     isfeasible <- .inscribetriangle.feasible(wghts)
+     if (!isfeasible) {
+         scalef <- 4 * wghts[1] * wghts[2] * wghts[3]/(1 - sum(wghts))^2
+         scalef <- scalef^(1/3)
+         wghts <- wghts/(scalef * 1.001)
+         isfeasible <- .inscribetriangle.feasible(wghts)
+         stopifnot(!isfeasible)
+     }
+     if (!isfeasible)
+         return(list(feasible = FALSE))
+     scab <- .inscribetriangle.compute(wghts)
+     inner.xy <- (1 - scab) * xy + scab * (xy[c(2, 3, 1), ])
+     return(list(feasible = TRUE, inner.xy = inner.xy, scalef = scalef))
+ }

> WeightUniverse <- 18
> WeightVisible <- 16
> WeightInvisible <- WeightUniverse - WeightVisible
> w0ratio <- WeightInvisible/WeightVisible
> wa <- 0.25
> wb <- 0.25
> wc <- 0.25
> outer.weights <- c(wa, wb, wc)
> outer.innerw <- 1 - sum(outer.weights)
> outer.inner.ratios <- outer.weights/outer.innerw
> outer.feasible <- .inscribetriangle.feasible(outer.weights)
> wab <- 0.0625
> wbc <- 0.0625
> wca <- 0.0625
> wabc <- 0.0625
> inner.weights <- c(wab, wbc, wca)
> inner.innerw <- wabc
> sf <- (sum(inner.weights) + inner.innerw)
> Weight.Inner <- sf * WeightVisible
> if (sf > 0) {
+     inner.weights <- inner.weights/sf
+     inner.feasible <- .inscribetriangle.feasible(inner.weights)
+ } else {
+     inner.feasible <- FALSE
+ }
> side <- sqrt(4 * WeightVisible/(3 * sqrt(3)))
> angles <- pi/2 - c(0, 2 * pi/3, 4 * pi/3)
> outer.xy <- t(sapply(angles, function(a) c(x = side * cos(a),
+     y = side * sin(a))))
> inner <- .inscribetriangle.inscribe(outer.xy, wghts = outer.weights)
> inner.xy <- inner$inner.xy
> innest <- .inscribetriangle.inscribe(inner.xy, wghts = inner.weights)
> innest.xy = innest$inner.xy
> outest.xy <- outer.xy * sqrt(1 + w0ratio)
> rownames(outer.xy) <- paste("to", 1:3, sep = "")
> rownames(inner.xy) <- paste("ti", 1:3, sep = "")
> rownames(innest.xy) <- paste("tt", 1:3, sep = "")
> outline.a.xy <- do.call(rbind, list(outer.xy[1, , drop = FALSE],
+     inner.xy[1, , drop = FALSE], innest.xy[1, , drop = FALSE],
+     innest.xy[2, , drop = FALSE], inner.xy[3, , drop = FALSE]))
> outline.b.xy <- do.call(rbind, list(outer.xy[2, , drop = FALSE],
+     inner.xy[2, , drop = FALSE], innest.xy[2, , drop = FALSE],
+     innest.xy[3, , drop = FALSE], inner.xy[1, , drop = FALSE]))
> outline.c.xy <- do.call(rbind, list(outer.xy[3, , drop = FALSE],
+     inner.xy[3, , drop = FALSE], innest.xy[3, , drop = FALSE],
+     innest.xy[1, , drop = FALSE], inner.xy[2, , drop = FALSE]))
```

```
> VDP1 <- newTissueFromPolygon(points.xy = outline.a.xy, Set = 1)
> VDP2 <- newTissueFromPolygon(points.xy = outline.b.xy, Set = 2)
> VDP3 <- newTissueFromPolygon(points.xy = outline.c.xy, Set = 3)
> TMT <- addSetToDrawing(drawing1 = VDP1, drawing2 = VDP2, set2Name = "Set2")
> TMT <- addSetToDrawing(drawing1 = TMT, drawing2 = VDP3, set2Name = "Set3")
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-4, 4), c(-4, 4))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TMT)
> PlotSetBoundaries(TMT)
> .PlotFaceNames.TissueDrawing(TMT)
> PlotNodes(TMT)
```



```
> WeightUniverse <- 18
> WeightVisible <- 16
> WeightInvisible <- WeightUniverse - WeightVisible
> w0ratio <- WeightInvisible/WeightVisible
> wa <- 0.166666667
> wb <- 0.25
> wc <- 0.25
> outer.weights <- c(wa, wb, wc)
> outer.innerw <- 1 - sum(outer.weights)
> outer.inner.ratios <- outer.weights/outer.innerw
> outer.feasible <- .inscribetriangle.feasible(outer.weights)
> wab <- 0.166666667
> wbc <- 0
> wca <- 0
> wabc <- 0.166666667
> inner.weights <- c(wab, wbc, wca)
> inner.innerw <- wabc
> sf <- (sum(inner.weights) + inner.innerw)
> Weight.Inner <- sf * WeightVisible
> if (sf > 0) {
+     inner.weights <- inner.weights/sf
+     inner.feasible <- .inscribetriangle.feasible(inner.weights)
+ } else {
+     inner.feasible <- FALSE
+ }
> side <- sqrt(4 * WeightVisible/(3 * sqrt(3)))
> angles <- pi/2 - c(0, 2 * pi/3, 4 * pi/3)
> outer.xy <- t(sapply(angles, function(a) c(x = side * cos(a),
+     y = side * sin(a))))
> inner <- .inscribetriangle.inscribe(outer.xy, wghts = outer.weights)
> inner.xy <- inner$inner.xy
> innest <- .inscribetriangle.inscribe(inner.xy, wghts = inner.weights)
> innest.xy = innest$inner.xy
> outest.xy <- outer.xy * sqrt(1 + w0ratio)
> rownames(outer.xy) <- paste("to", 1:3, sep = "")
> rownames(inner.xy) <- paste("ti", 1:3, sep = "")
> rownames(innest.xy) <- paste("tt", 1:3, sep = "")
> outline.a.xy <- do.call(rbind, list(outer.xy[1, , drop = FALSE],
+     inner.xy[1, , drop = FALSE], innest.xy[1, , drop = FALSE],
+     innest.xy[2, , drop = FALSE], inner.xy[3, , drop = FALSE]))
> outline.b.xy <- do.call(rbind, list(outer.xy[2, , drop = FALSE],
+     inner.xy[2, , drop = FALSE], innest.xy[2, , drop = FALSE],
+     innest.xy[3, , drop = FALSE], inner.xy[1, , drop = FALSE]))
> outline.c.xy <- do.call(rbind, list(outer.xy[3, , drop = FALSE],
+     inner.xy[3, , drop = FALSE], innest.xy[3, , drop = FALSE],
+     innest.xy[1, , drop = FALSE], inner.xy[2, , drop = FALSE]))
> VDP1 <- newTissueFromPolygon(points.xy = outline.a.xy, Set = 1)
> VDP2 <- newTissueFromPolygon(points.xy = outline.b.xy, Set = 2)
> VDP3 <- newTissueFromPolygon(points.xy = outline.c.xy, Set = 3)
> TMT <- addSetToDrawing(drawing1 = VDP1, drawing2 = VDP2, set2Name = "Set2")
> TMT <- addSetToDrawing(drawing1 = TMT, drawing2 = VDP3, set2Name = "Set3")

> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-4, 4), c(-4, 4))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TMT)
> PlotSetBoundaries(TMT)
> .PlotFaceNames.TissueDrawing(TMT)
> PlotNodes(TMT)
```



15 Three squares

```
> ss1 <- c(-2.04988805276466, 1.4142135623731, 1.41421356237309,
+     -1.77228856812726, -1.77228856812726, -2.04988805276466,
+     -2.04988805276466, -2.04988805276466, 3.8936089116869, 3.8936089116869,
+     1.77228856812726, 1.77228856812726)
> ss2 <- c(-2.25237500351774, 3.88908729652601, 3.88908729652601,
+     -2.25237500351774, -2.16799518941608, -2.16799518941608,
+     1.4142135623731, 1.41421356237309)
> ss3 <- c(-1.4142135623731, 4.56252232622749, 4.56252232622749,
+     2.08764859207457, 2.08764859207457, -1.4142135623731, -1.4142135623731,
+     -1.4142135623731, 2.08764859207457, 2.08764859207457, 3.53553390593274,
+     3.53553390593274)
> SS1 <- matrix(ss1, ncol = 2, byrow = FALSE)
> rownames(SS1) <- paste("sa", 1:6, sep = "")
> SS2 <- matrix(ss2, ncol = 2, byrow = FALSE)
> rownames(SS2) <- paste("sb", 1:4, sep = "")
> SS3 <- matrix(ss3, ncol = 2, byrow = FALSE)
> rownames(SS3) <- paste("sc", 1:6, sep = "")
> VDP1 <- newTissueFromPolygon(points.xy = SS1, Set = 1)
> VDP2 <- newTissueFromPolygon(points.xy = SS2, Set = 2)
> VDP3 <- newTissueFromPolygon(points.xy = SS3, Set = 3)
> TM <- addSetToDrawing(drawing1 = VDP1, drawing2 = VDP2, set2Name = "Set2")
> TM <- addSetToDrawing(drawing1 = TM, drawing2 = VDP3, set2Name = "Set3")
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-7, 7), c(-5, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM)
> PlotSetBoundaries(TM, gp = gpar(lwd = 2, col = c("green", "red")))
> PlotNodes(TM)
> .PlotFaceNames.TissueDrawing(TM)
> PlotSetBoundaries(VDP3, gp = gpar(lwd = 2, col = c("green")))
```

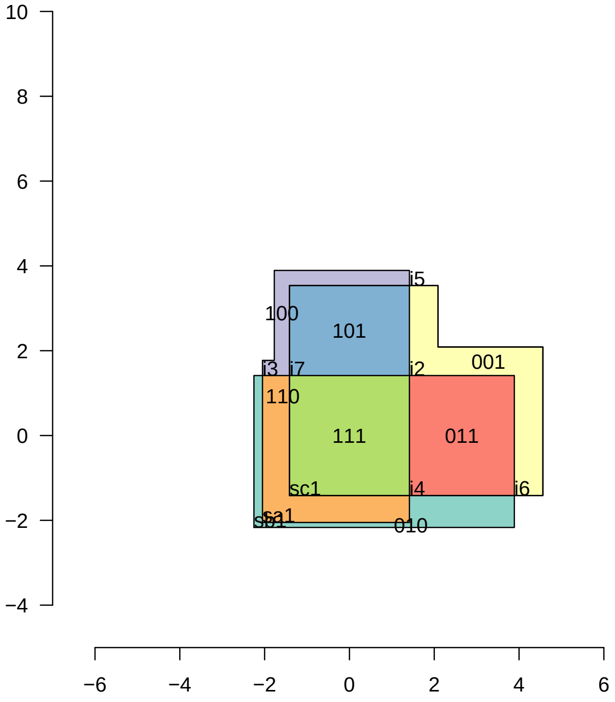


Figure 4: Injecting points

16 Noncontigous subsets

```
> px1 <- matrix(c(-5, -3, -5, 3, 5, 3, 5, -3), ncol = 2, byrow = TRUE)
> rownames(px1) <- paste("pa", 1:nrow(px1), sep = "")
> px2 <- matrix(c(-3, -5, -3, 5, 3, 5, 3, -5), ncol = 2, byrow = TRUE)
> rownames(px2) <- paste("pb", 1:nrow(px2), sep = "")
> VX1 <- newTissueFromPolygon(px1, Set = 1)
> VX2 <- newTissueFromPolygon(px2, Set = 2)
> TM <- addSetToDrawing(VX1, VX2, set2Name = "Set2")
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-10, 10), c(-10, 10))
> grid.xaxis()
> grid.yaxis()
> PlotNodes(TM)
> PlotSetBoundaries(TM, gp = gpar(lwd = 2, col = c("green", "red",
+     "blue")))
> .PlotFaceNames.TissueDrawing(TM)
```

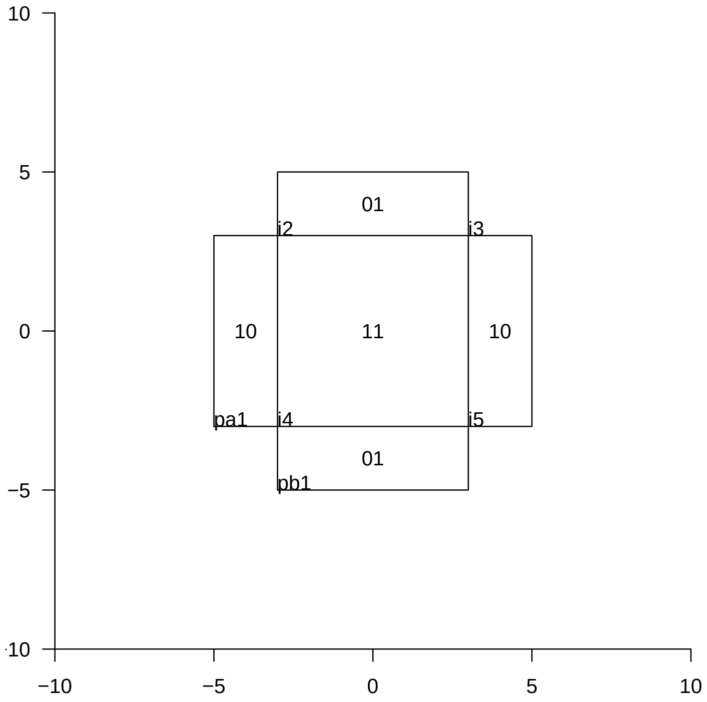


Figure 5: Injecting points

17 Ellipses

```
> phi <- 0.8
> dex <- 1.7
> dey <- 2.5
> a <- 7.6
> e <- 0.9
> x0 <- c(-0.9, -5)
> E <- list()
> E[[1]] <- newTissueFromEllipse(f1 = x0 + c(0, 0), phi = -phi,
+     dx = 0.1, e = e, a = -a, Set = 1)
> E[[2]] <- newTissueFromEllipse(x0 + c(5 + dex, -2), phi, e, a,
+     dx = 0.1, Set = 2)
> TM <- E[[1]]
> TM <- addSetToDrawing(TM, E[[2]], set2Name = "Set2")
```

```
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-10, 10), c(-10, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM)
> PlotSetBoundaries(E[[2]], gp = gpar(lwd = 2, col = c("red", "red",
+     "blue")))
> PlotNodes(TM)
> .PlotFaceNames.TissueDrawing(TM)
> PlotSetBoundaries(TM, gp = gpar(lwd = 2, col = c("green")))
```

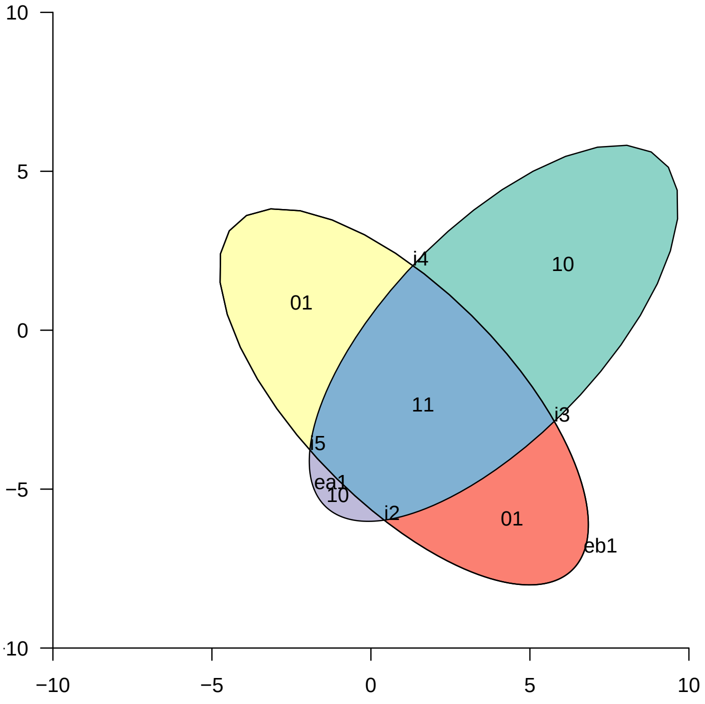


Figure 6: Injecting points

```
> phi <- 0.8
> dex <- 1.7
> dey <- 2.5
> a <- 7.6
> e <- 0.9
> x0 <- c(-0.9, -5)
> dx <- 0.1
> E <- list()
> E[[1]] <- newTissueFromEllipse(f1 = x0 + c(0, 0), dx = dx, phi = -phi,
+     e = e, a = -a, Set = 1)
> E[[2]] <- newTissueFromEllipse(x0 + c(dex, 0), dx = dx, phi,
+     e, a, Set = 2)
> E[[3]] <- newTissueFromEllipse(x0 + c(-dey, dey), dx = dx, -phi,
+     e, -a, Set = 3)
> E[[4]] <- newTissueFromEllipse(x0 + c(dex + dey, dey), dx = dx,
+     phi, e, a, Set = 4)
> TM <- E[[1]]
> TM <- addSetToDrawing(TM, E[[2]], set2Name = "Set2")
> grid.newpage()
> pushViewport(plotViewport(c(1, 1, 1, 1)))
> makevp.eqsc(c(-10, 10), c(-10, 10))
> grid.xaxis()
> grid.yaxis()
> PlotFaces(TM)
> PlotSetBoundaries(TM, gp = gpar(lwd = 2, col = c("green", "red",
+     "blue")))
> PlotNodes(TM)
> .PlotFaceNames.TissueDrawing(TM)
```

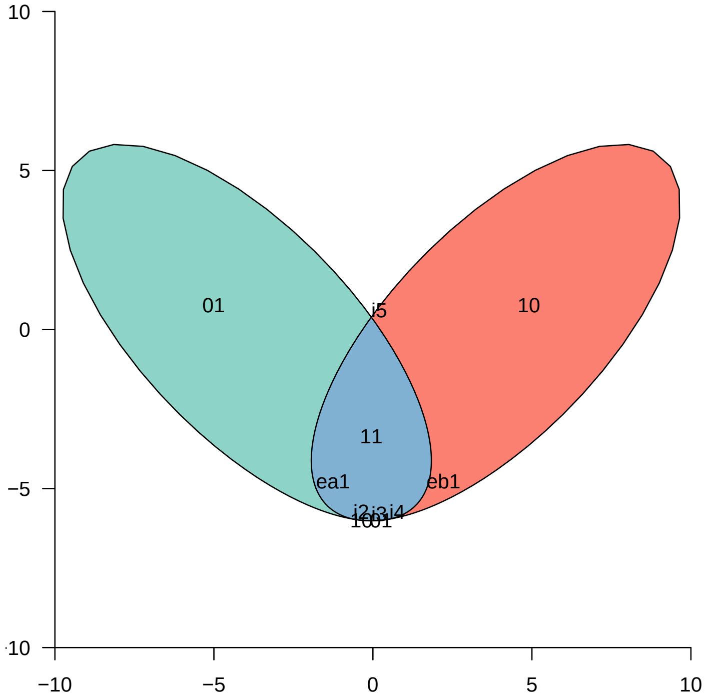


Figure 7: Injecting points

18 This document

Author	Jonathan Swinton
CVS id of this document	Id: TissueDrawingTest.Rnw 31 2009-08-01 22:01:48Z js229 .
Generated on	2nd August, 2009
R version	R version 2.9.0 (2009-04-17)

[1]

References

[1] A. W. F. Edwards. *Cogwheels of the Mind: The Story of Venn Diagrams*. The John Hopkins University Press, Baltimore, Maryland, 2004.