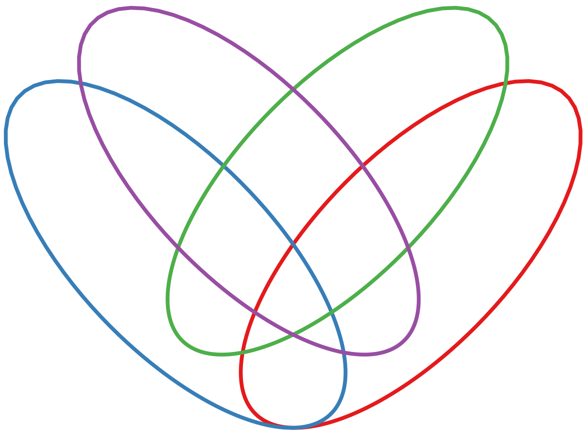
Venn diagrams in R
with the Vennerable package

Jonathan Swinton
jonathan@swintons.net

25th September, 2009



Contents

1 Overview

The Vennerable package provides routines to compute and plot Venn diagrams, including the classic two- and three-circle diagrams but also a variety of others with different properties and for up to seven sets. In addition it can plot diagrams in which the area of each region is proportional to the corresponding number of set items or other weights. This includes Euler diagrams, which can be thought of as Venn diagrams where regions corresponding to empty intersections have been removed.

Figure 1 shows a three-circle Venn diagram of the sort commonly found. To draw it, we use as an example the `StemCell` data of Boyer et al.**?** which lists the gene names associated with each of four transcription factors

```
> library(Vennerable)
> data(StemCell)
> str(StemCell)

List of 4
 $ OCT4 : chr [1:623] "AASDH" "ABTB2" "ACCN4" "ACD" ...
 $ SOX2 : chr [1:1279] "182-FIP" "AASDH" "ABCA5" "ABCB10" ...
 $ NANOG: chr [1:1687] "13CDNA73" "AASDH" "ABCA5" "ABCB10" ...
 $ E2F4 : chr [1:1273] "76P" "7h3" "AAMP" "AATF" ...
```

First we construct an object of class `Venn`:

```
> Vstem <- Venn(StemCell)
> Vstem

A Venn object on 4 sets named
OCT4,SOX2,NANOG,E2F4
0000 1000 0100 1100 0010 1010 0110 1110 0001 1001 0101 1101 0011 1011 0111 1111
   0  109  305   45  644   64  354  287  821   30   78    6  118   16  138   66
```

Although Vennerable can cope with 4-set Venn diagrams, for now we reduce to a three-set object

```
> Vstem3 <- Vstem[, c("OCT4", "SOX2", "NANOG")]
> Vstem3

A Venn object on 3 sets named
OCT4,SOX2,NANOG
000 100 010 110 001 101 011 111
821 139 383  51 762  80 492 353
```

Note how the weights were appropriately updated.

Now a call to plot produces the diagram in Figure 1 showing how many genes are common to each transcription factor.

```
> plot(Vstem3, doWeights = FALSE)
```

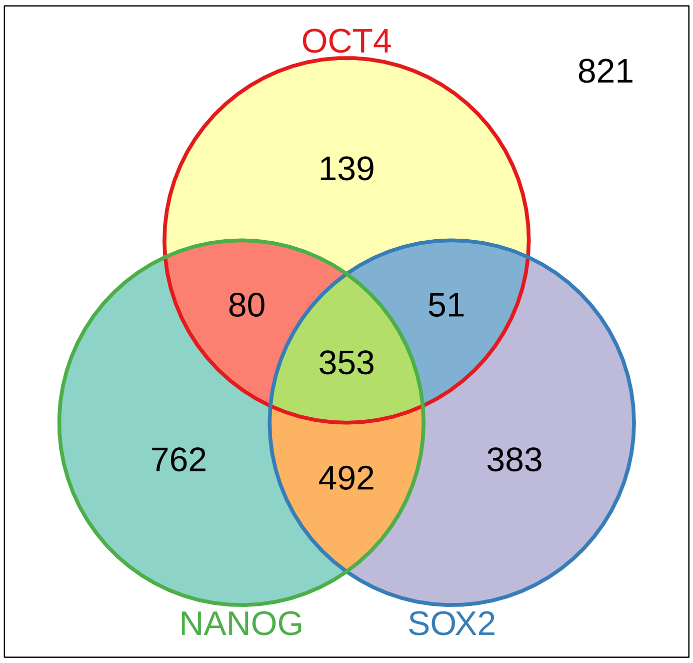


Figure 1: A three-circle Venn diagram

Quite commonly, we may have sets whose intersections we only know by the number of elements. These can be created as `Venn` objects by supplying a named vector of `Weights`:

```
> Vdemo2 <- Venn(SetNames = c("foo", "bar"), Weight = c(`01` = 7,
+     `11` = 8, `10` = 12))
```

Whichever way the `Venn` object is created, we can plot Venn diagrams in which the area of each intersection is proportional to those weights as in Figure 2.

```
> plot(Vdemo2, doWeights = TRUE, type = "circles")
```

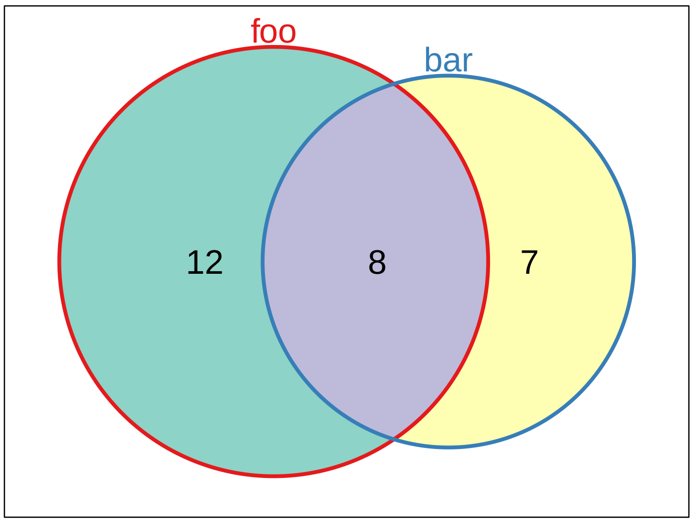


Figure 2: A two-set weighted Venn diagram

For these basic plots, use of the `Vennerable` package may sometimes overkill, but in more complex situations it has useful abilities. First it allows the use of a variety of other shapes for the set boundaries, and up to nine different sets. Secondly it implements a number of published or novel algorithms for generating diagrams in which the area of each region is proportional to, for example, the number of corresponding set elements. Finally it adds a number of graphical control abilities, including the ability to colour individual regions separately.

2 Computation and Annotation

2.1 Computing Venn drawings

The calls to `plot` are really convenience wrappers for two separate functions which compute the geometry of the drawing first, returning an object of class `VennDrawing` and then renders that object. For example

```
> plot(Vstem3, doWeights = TRUE)
```

is equivalent to

```
> C3 <- compute.Venn(Vstem3, doWeights = TRUE)
> grid.newpage()
> plot(C3)
```

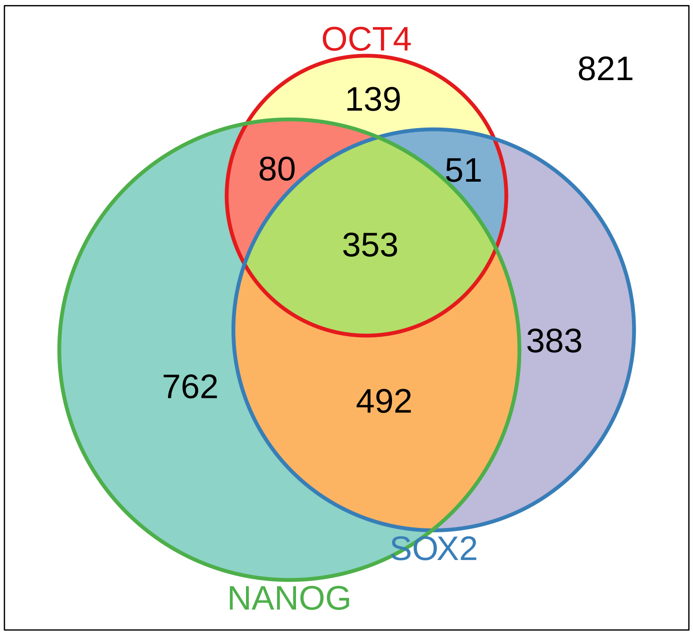


Figure 3: A weighted three-circle Venn diagram

Note the use of a function from the `grid` graphics library package: all of the renderings are created using `grid` objects. The `compute.Venn` function can take a variety of arguments such as `doWeights` controlling the geometry and topology of the drawing, while the `plot` method has a number of arguments controlling annotation and display.

2.2 Annotation parameters

The text displayed in each face is controlled by the `FaceText` element of the `show` parameter list to `plot`. Other elements of the parameter control whether, for example, set names are displayed or faces are individually coloured

```
> grid.newpage()
> plot(C3, show = list(FaceText = "signature", SetLabels = FALSE,
+     Faces = FALSE, DarkMatter = FALSE))
```

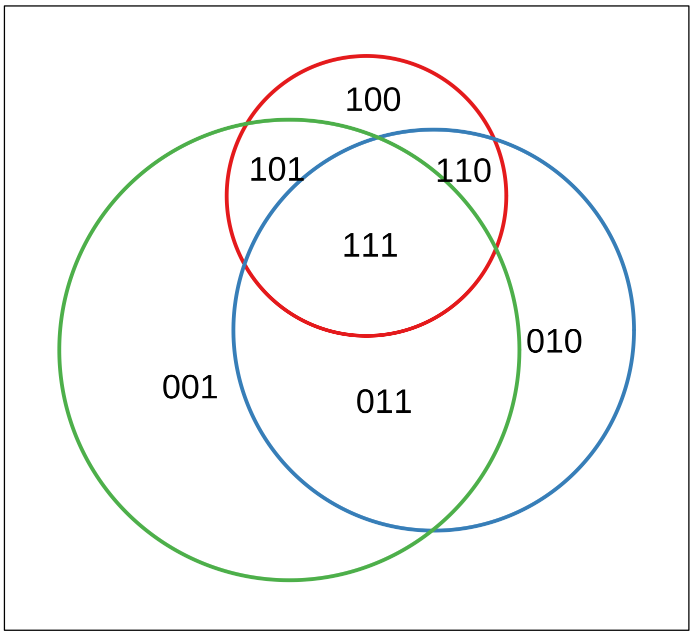


Figure 4: The same Venn diagram with different `show` parameters

2.3 Graphical parameters

The package makes its own decisions about how to colour lines and faces depending on the complexity of the diagram. This can be overridden with the `gpList` argument to `plot`. The default choices are equivalent to

```
> gpList <- VennThemes(C3)
> plot(C3, gpList = gpList)
```

Low-level modifications can be using the `gpList` argument, typically by modifying the value of a call to `VennThemes`. There is more detail on the `VennThemes` man page about the format of `gpList`. More high-level modifications can be made by supplying the `ColourAlgorithm` or `increasingLineWidth` arguments to `VennThemes`.

```
> grid.newpage()
> gp <- VennThemes(C3, colourAlgorithm = "binary")
> plot(C3, gpList = gp, show = list(FaceText = "sets", SetLabels = FALSE,
+     Faces = TRUE))
```

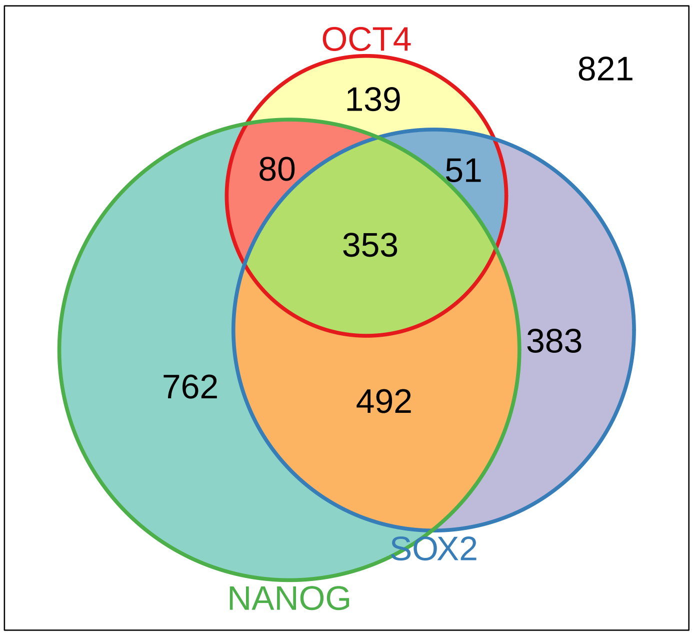


Figure 5: The effect of setting `ColourAlgorithm="binary"` and `FaceText="sets"`

The position and format of the set and face annotation are controlled by the data returned by `VennGetSetLabels` and `VennGetFaceLabels`, respectively, which can be modified and then reembedded in the `VennDrawing` object with `VennSetSetLabels` and `VennSetFaceLabels`.

```
> grid.newpage()
> SetLabels <- VennGetSetLabels(C3)
> SetLabels[SetLabels$Label == "February", "y"] <- SetLabels[SetLabels$Label ==
+     "March", "y"]
> C3 <- VennSetSetLabels(C3, SetLabels)
> plot(C3)
```

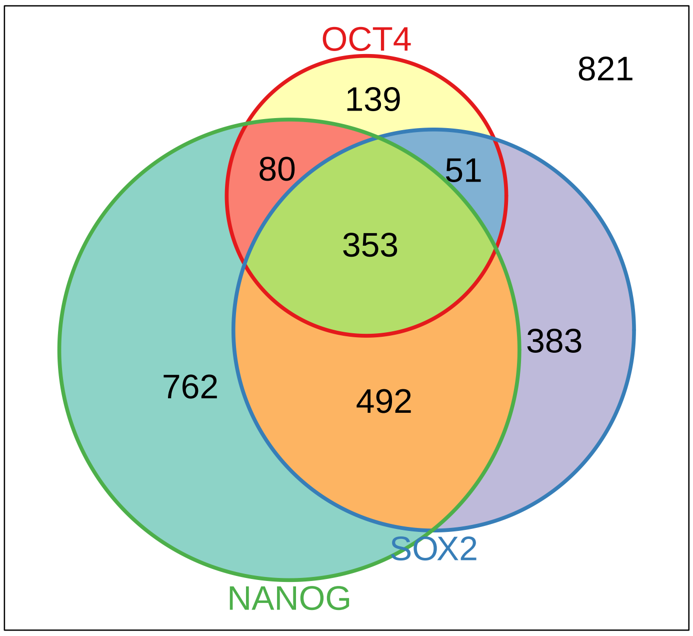


Figure 6: Modifying the position of annotation

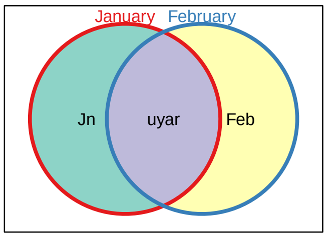
3 Unweighted Venn diagrams

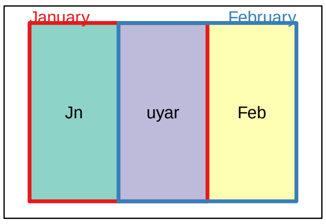
For another running example, we use sets named after months, whose elements are the letters of their names.

```
> setList <- strsplit(month.name, split = "")
> names(setList) <- month.name
> Vmonth3 <- VennFromSets(setList[1:3])
> Vmonth2 <- Vmonth3[, c("January", "February"), ]
```

3.1 Unweighted 2-set Venn diagrams

For two sets, a diagram can be drawn using either circles or squares, as controlled by the `type` argument. This is shown in Figure **??**.[1].





plot(V,type=circles,...)　　plot(V,type=squares,...)

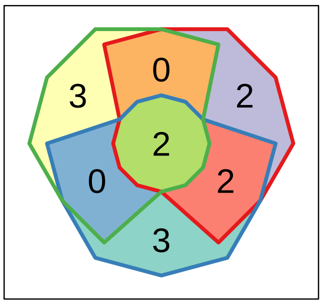
Figure 7: Unweighted 2-set Venn diagrams with `type=circles` or `type=squares`

[1]Here and in the rest of this vignette, much of the code to plot the Figures, which is mainly devoted to layout, is not shown. However it can always be found by inspecting the source code of the vignette at `PACKAGETREE/Vennerable/doc/Venn.Rnw` where `PACKAGETREE` is directory where the package was installed.

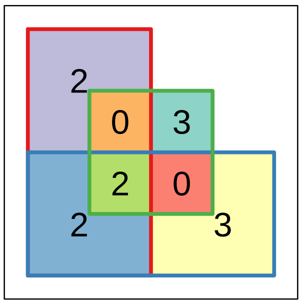
3.2 Unweighted 3-set Venn diagrams

For three sets, the `type` argument can be `circles`, `squares`, `ChowRuskey`, `triangles` or `AWFE`. We have already seen the circles plot. The AWFE plot is an implementation of the elegant ideas of **?**. The Chow-Ruskey plot is from **?**, and is a redrawing of the AWFE plot in such a way that there is an algorithm which will allow all of the faces to be adjusted in area without disrupting the topology of the diagram. The triangles plot is fairly obvious, for example to reference**?**, but I have not seen it implemented elsewhere.

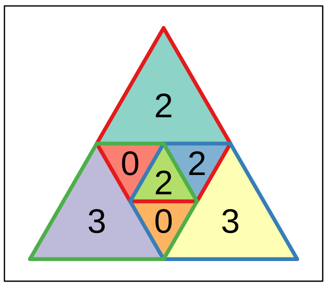
This example of the squares plot is not *simple*, in the sense of **?**, because the set boundaries don't cross transversally. Topologically, there is only one simple Venn diagram of order 3 (in a way that **?** makes precise).



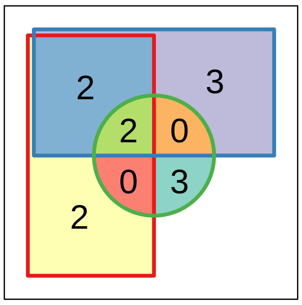
plot(Vmonth3,type="ChowRuskey",...)



plot(Vmonth3,type="squares",...)



plot(Vmonth3,type="triangles",...)



plot(Vmonth3,type="AWFE",...)

3.3 Unweighted 4-set Venn diagrams

For four sets, the `type` argument can be `ChowRuskey`, `AWFE`,`squares` or `ellipses`.

The squares plot is said by Edwards **?** to have been introduced by Lewis Carroll **?**. The ellipse plot was suggested by Venn **?**.

Note how the package makes an attempt to identify a point within each face where the annotation can be plotted, but doesn't make a very good choice for very non-concave or elongated faces.

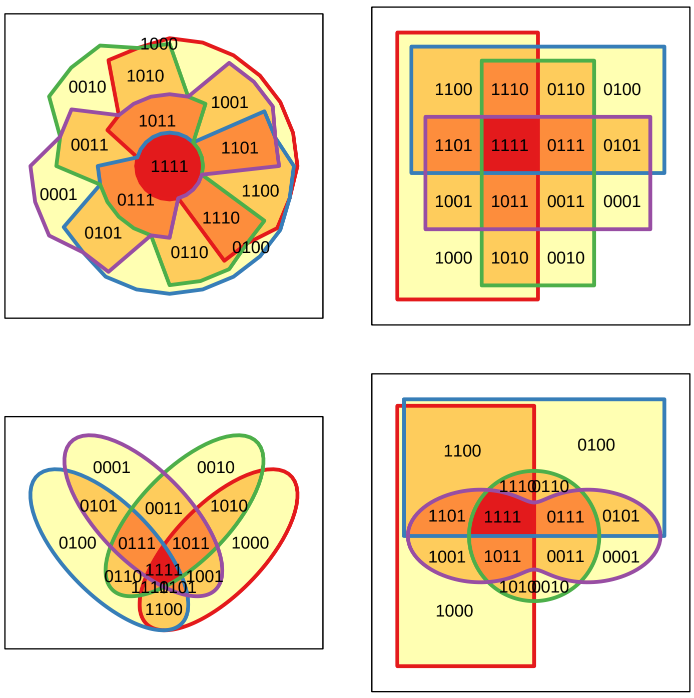


Figure 8: Venn diagrams on four sets drawn with the `type` argument set to `ChowRuskey`, `squares`, `ellipses`, and `AWFE`.

A number of variants on the `squares` type are implemented. Currently they can only be accessed by passing the parameters `s` or `likesquares` to the low level creation function `compute.S4` directly, which is what is done in Figure **??**.

For more details on this see the help pages for `compute.S4`.

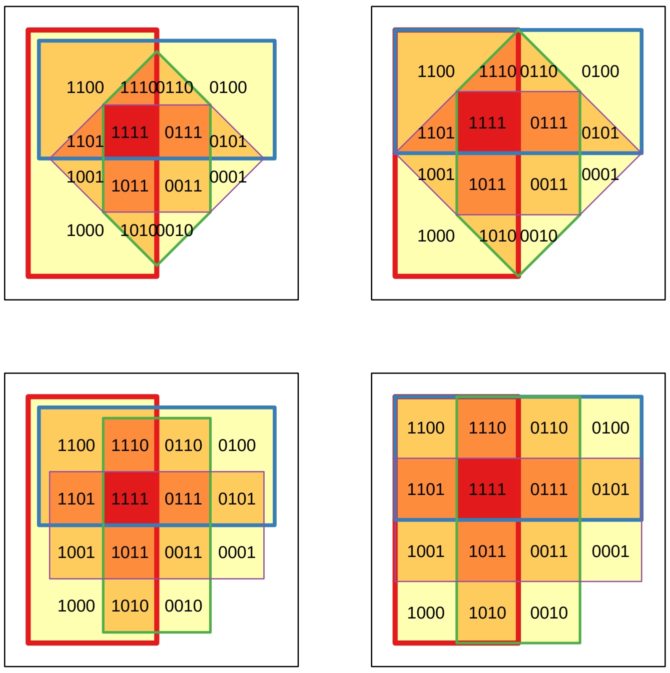


Figure 9: Four variants on the four-squares

3.4 Unweighted Venn diagrams on more than four sets

The package implements a variant of the Edwards construction**?**, which can in principle generate Venn diagrams on an arbitrary number of sets n. The currently implemented algorithm only computes up to 8 sets for the classic construction.

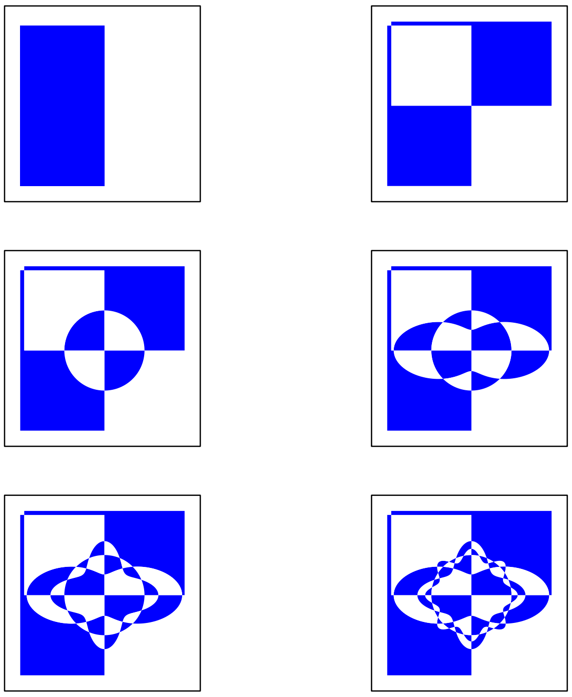


Figure 10: Edwards constructions for five to eight sets

A variant on the Edwards construction I developed as both quicker to compute with, because it is based on straight lines, and slightly easier to visualise high-order intersections in, is shown in Figure **??**. It can be drawn by using `type="battle"` for up to 9 sets.

```
> plot(Venn(n = 9), type = "battle", show = list(SetLabels = FALSE,
+     FaceText = ""))
```

Figure 11: The battlement variant of the Edwards construction on 9 sets with the `type=battle` argument

4 Weighted Venn diagrams

There are repeated requests to generate Venn diagrams in which the areas of the faces themselves are meant to carry information, mainly by being proportional to the intersection weights. Even when these diagrams can be drawn, they are not often a success in their information-bearing mission. But we can try anyway, through use of the argument `doWeights=TRUE`. First of all we consider the case when all the visible intersection weights are nonzero.

4.1 Weighted 2-set Venn diagrams for 2 Sets

4.1.1 Circles

It is always possible to get an exactly area-weighted solution for two circles as shown in Figure **??**.

```
> V3.big <- Venn(SetNames = LETTERS[1:3], Weight = 2^(1:8))
> Vmonth2.big <- V3.big[, c(1:2)]
> plot(Vmonth2.big)
```

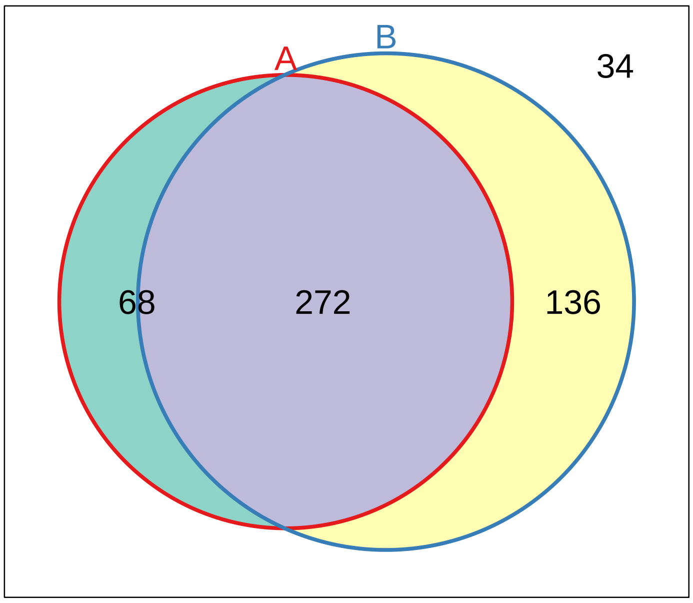


Figure 12: Weighted 2d Venn

4.1.2 Squares

As for circles, square weight-proportional diagrams can be simply constructed.

```
> plot(Vmonth2.big, type = "squares")
```

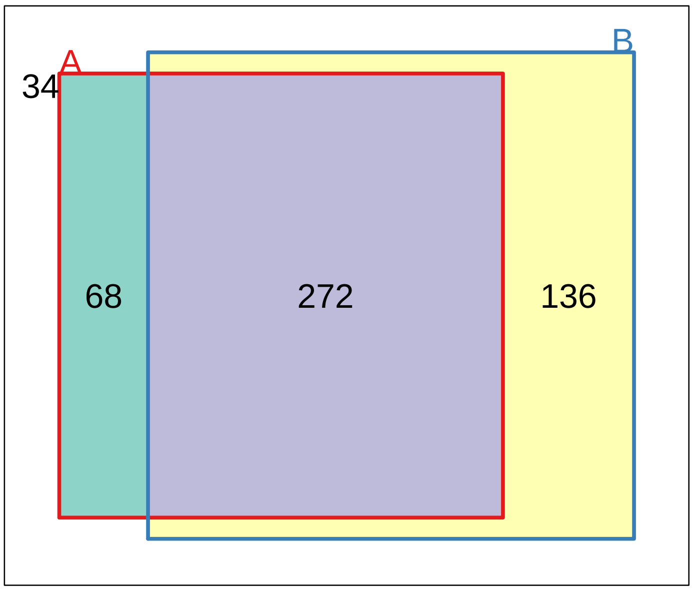


Figure 13: Weighted 2d Venn squares

4.2 Weighted 3-set Venn diagrams

4.2.1 Circles

There is no general way of creating area-proportional 3-circle diagrams. While these attempts at these diagrams are quite commonly seen, they must almost always be inexact.

The `Vennerable` package makes an attempt at produce approximate ones. Figure **??** shows a dataset taken from Chow and Ruskey **?**

```
> Vcombo <- Venn(SetNames = c("Female", "Visible Minority", "CS Major"),
+     Weight = c(0, 4148, 409, 604, 543, 67, 183, 146))
> plot(Vcombo)
```

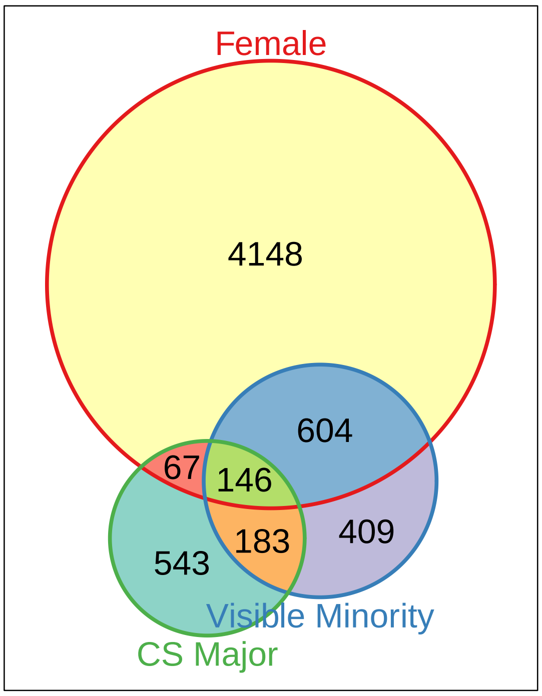


Figure 14: 3D Venn diagram

The algorithm used is to compute the individual circles to have the exact area necessary for proportionality, and then compute each of the three pairwise distances between centres necessary for the correct pairwise areas. If these distances do not satisfy the triangle inequality the largest is reduced until they do. Then the circles are arranged with their centres separated by these (possibly modified) distances.

4.2.2 Squares

There is are a number of possible algorithms to generate exact Venn diagrams based on polygons. With `type=squares` the package uses an algorithm almost identical to that suggested by **?**, which tries to generate rectangles as the set boundaries if possible

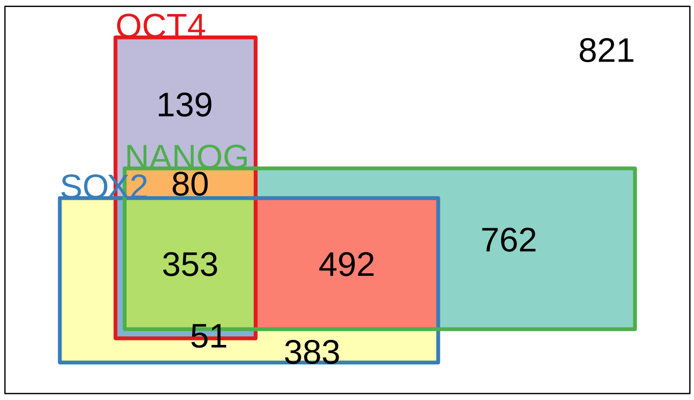


Figure 15: Weighted 3-set Venn diagram based on the algorithm of **?**

```
> V3a <- Venn(SetNames = month.name[1:3], Weight = 1:8)
> plot(V3a, type = "squares", show = list(FaceText = "weight",
+     SetLabels = FALSE))
```

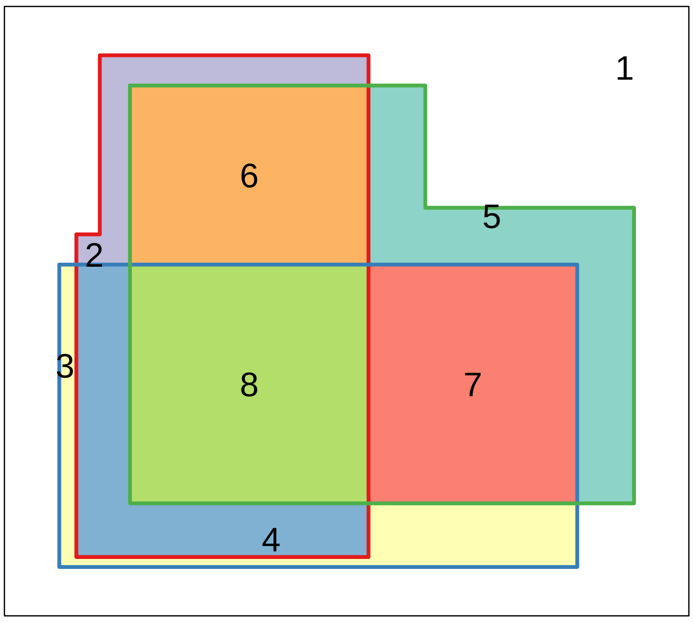


Figure 16: Weighted 3-set Venn diagram based on the algorithm of **?**. This time the algorithm fails to find rectangles.

4.2.3 Triangles

The triangular Venn diagram on 3-sets lends itself nicely to an area-proportional drawing under some contraints on the weights (detailed elsewhere).

```
> grid.newpage()
> C3t <- compute.Venn(V3a, type = "triangles")
> plot(C3t, show = list(SetLabels = FALSE, DarkMatter = FALSE))
```

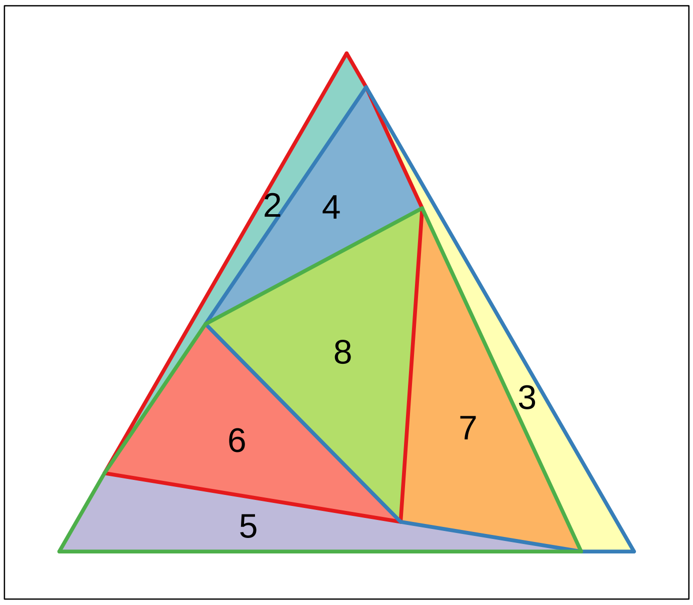


Figure 17: Weighted Triangular Venn diagram

4.3 Chow-Ruskey diagrams for 3 or more sets

The general Chow-Ruskey algorithm **?** for area-proportional faces can be implemented in principle for an arbitrary number of sets provided the weight of the common intersection is nonzero. In practice the package is limited (to $n = 9$) by the size of the largest AWFE diagram it can compute.

```
> plot(V3a, type = "ChowRuskey", show = list(SetLabels = FALSE,
+     DarkMatter = FALSE))
```

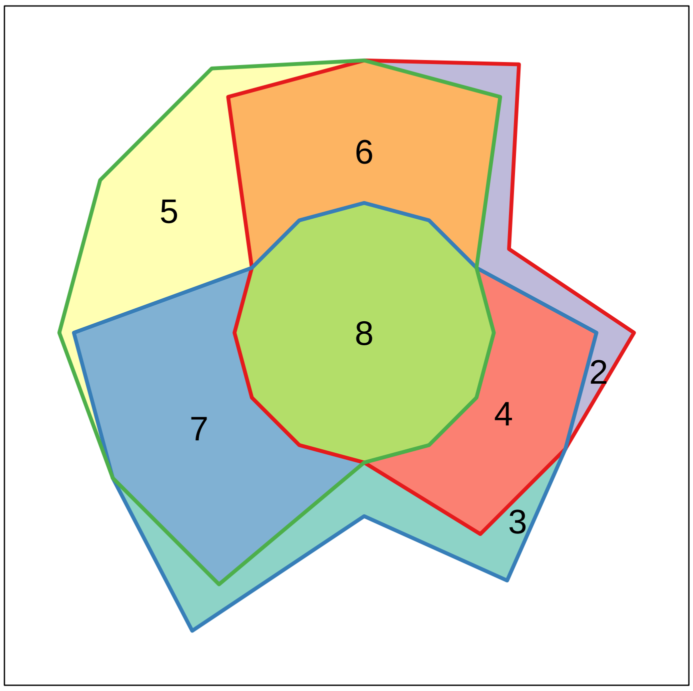


Figure 18: Chow-Ruskey weighted 3-set diagram

```
> V4a <- Venn(SetNames = LETTERS[1:4], Weight = 16:1)
> plot(V4a, type = "ChowRuskey", show = list(SetLabels = FALSE,
+     DarkMatter = FALSE))
```

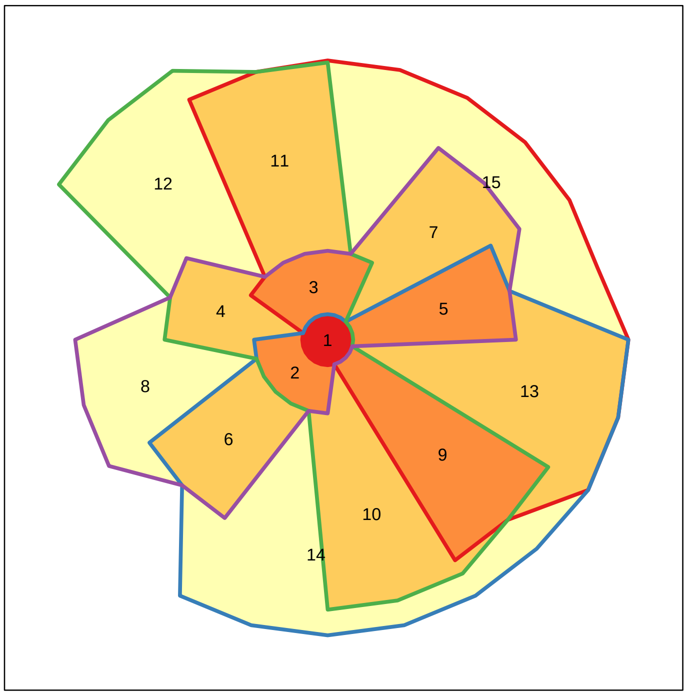


Figure 19: Chow-Ruskey weighted 4-set diagram

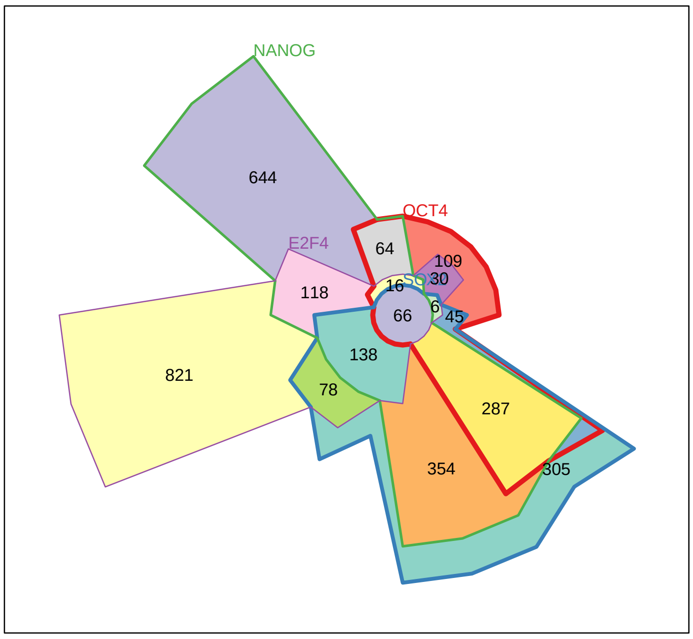


Figure 20: Chow-Ruskey weighted 4-set diagram for the stem cell data

5 Euler diagrams

A *Euler diagram* is one in which regions of zero-weight are not displayed at all (but those which are displayed are not necessarily area-proportional). This can be achieved, for some geometries, by use of the `doEuler=TRUE` argument.

As we have seen, for some geometries it is not possible to enforce exact area-proportionality when requested by the `doWeight=TRUE` argument, and an attempt is made to produce an approximately area-proportional diagram. In particular, regions whose weight is zero may appear with nonzero areas. These two flags can interact in weird and uncomfortable ways depending on exactly which intersection weights are zero.

5.1 2-set Euler diagrams

5.1.1 Circles

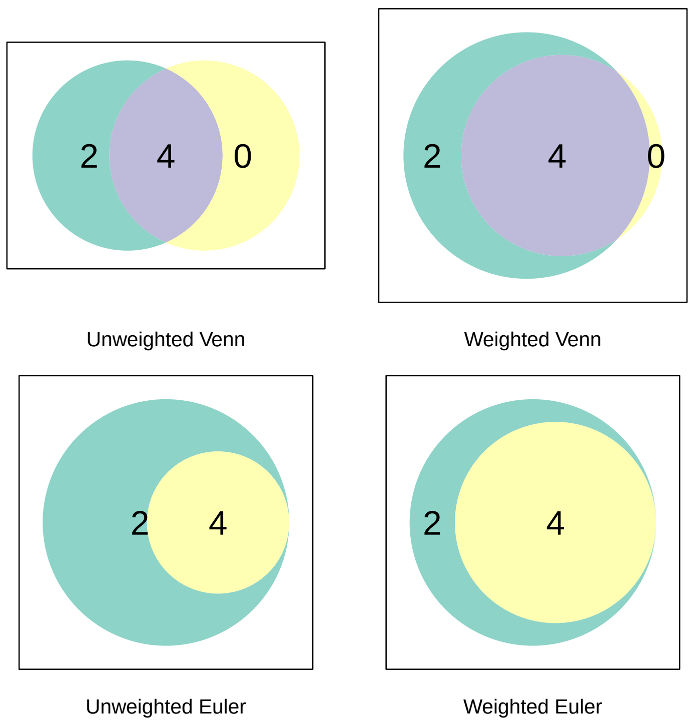


Figure 21: Effect of the `doEuler` and `doWeights` flags for a `Venn` object with `Weights(V)["01"]=0`

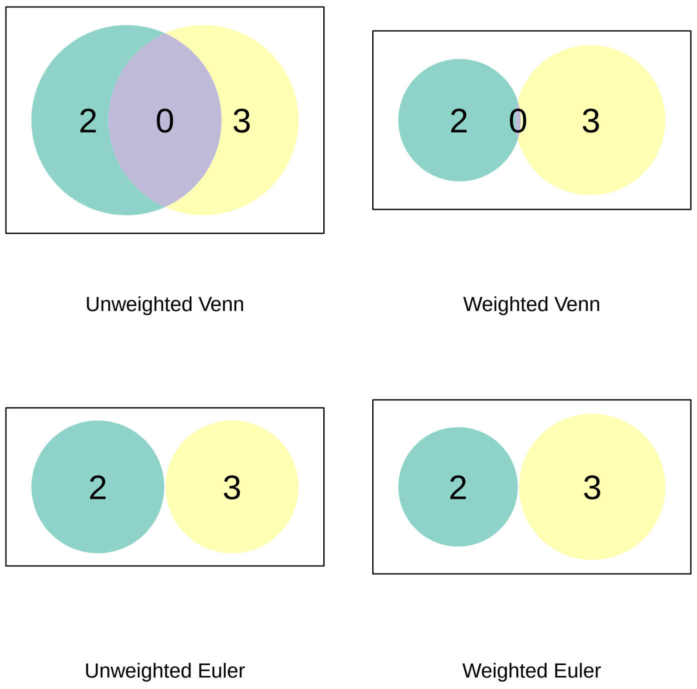
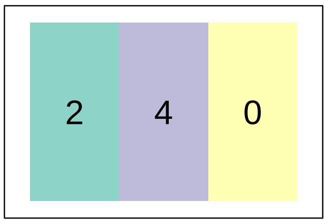


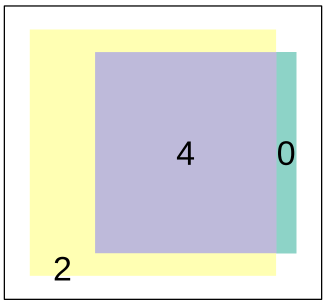
Figure 22: As before for when the intersection set has zero weight

5.1.2 Squares

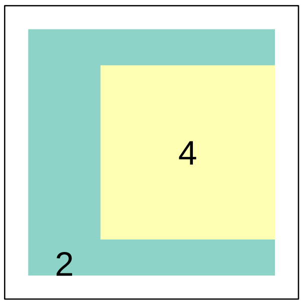
As for circles, the idea of a weighted Venn diagram when some of the weights are zero doesn't make much sense in theory but might be useful for making visual points.



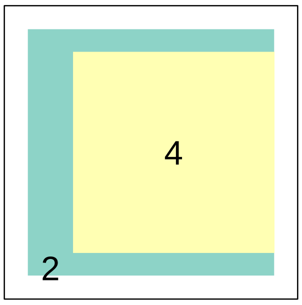
Unweighted Venn



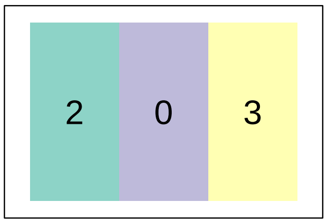
Weighted Venn



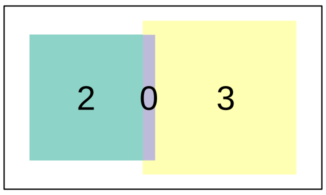
Unweighted Euler



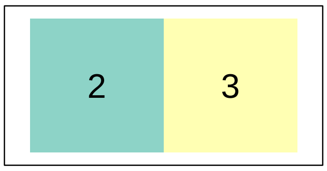
Weighted Euler



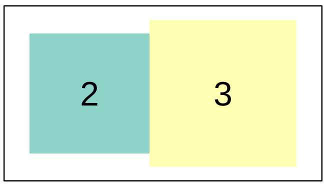
Unweighted Venn



Weighted Venn



Unweighted Euler



Weighted Euler

5.2 3-set Euler diagrams

5.2.1 Circles

There is currently no effect of setting `doEuler=TRUE` for three circles, but the `doWeights=TRUE` flag does an approximate job. There are about 40 distinct ways in which intersection regions can have zeroes can occur, but here are some examples.

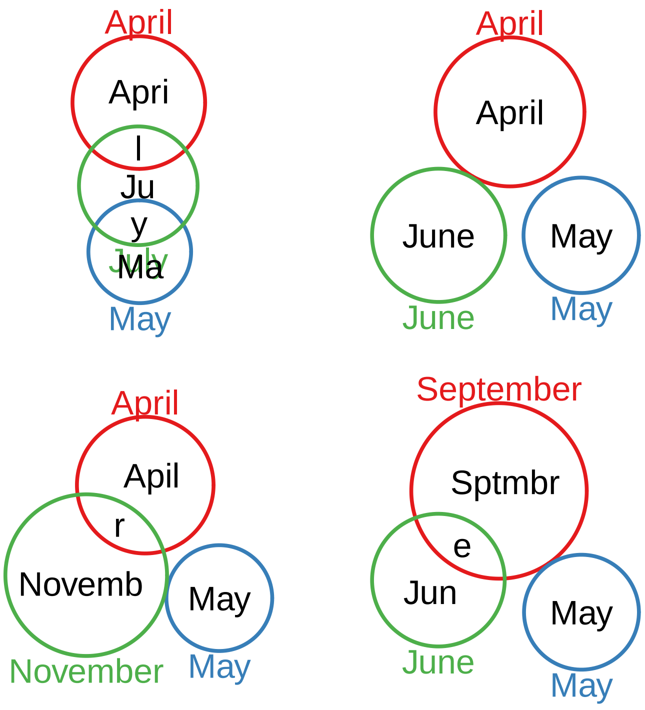


Figure 23: Weighted 3d Venn empty intersections.

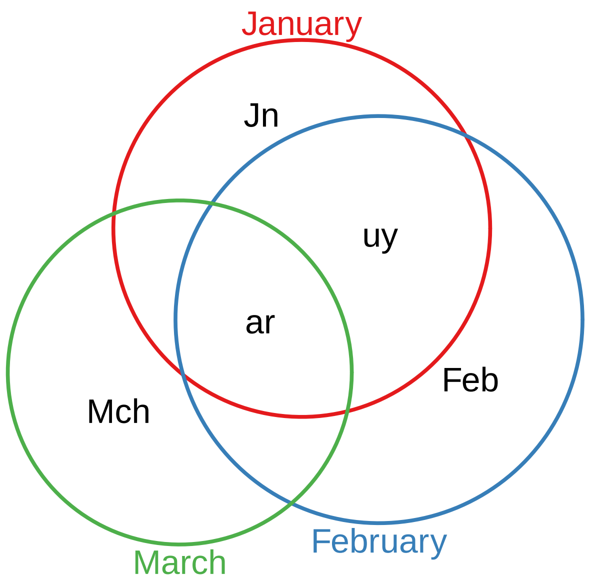


Figure 24: Approximate weighted 3d Venn showing element set membership

5.2.2 Triangles

The doEuler flag has no effect for triangles; all the weighted diagrams produced are Euler diagrams.

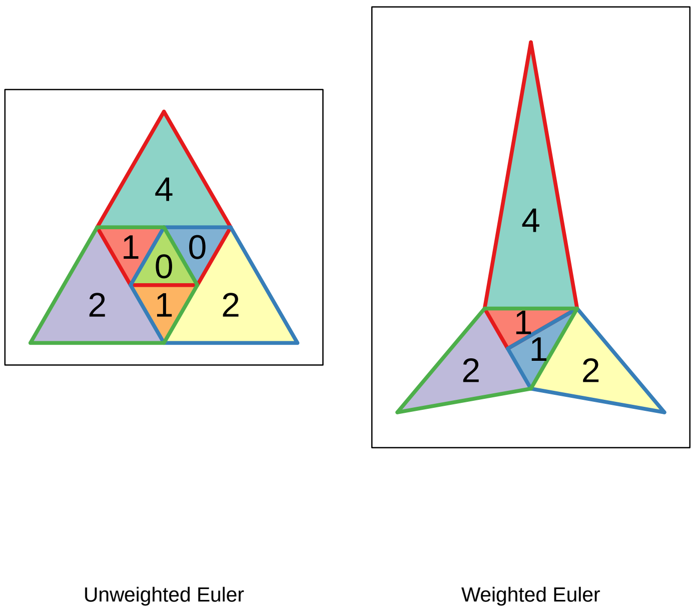
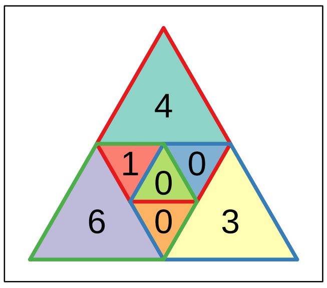


Figure 25: 3d Venn triangular with two zero weights plotted with the doWeights flag FALSE and TRUE



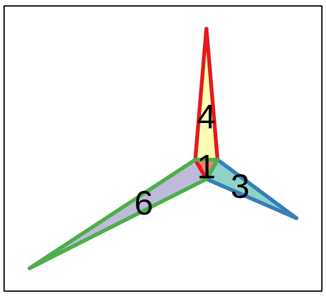


Figure 26: 3d Venn triangular with three zero weights plotted with the `doWeights` flag FALSE and TRUE

5.3 4-set Euler diagrams

5.3.1 Chow-Ruskey diagrams

The `doEuler` flag has no effect for Chow-Ruskey because all the weighted diagrams produced are already Euler diagrams.

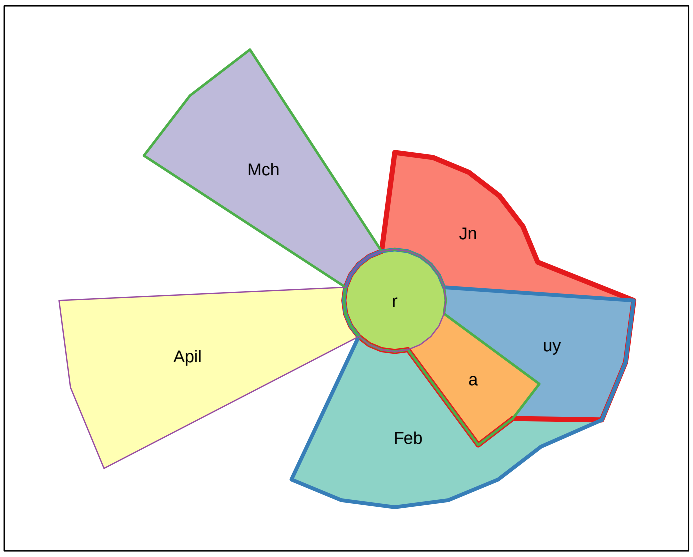


Figure 27: Chow-Ruskey diagram with some zero weights

6 Some loose definitions

Figure 1 illustrates membership of three sets, in order OCT4, SOX2 , NANOG. Genes which are members of the SOX2 set but not the OCT4 or NANOG sets are members of an *intersection subset* with *indicator string* or *signature* `010`.

Given n sets of elements drawn from a universe, there are 2^n intersection subsets. Each of these is a subset of the universe and there is one corresponding to each of the binary strings of length n. If one of these indicator strings has a 1 in the i-th position, all of members of the corresponding intersection subset must be members of the i-th set. Depending on the application, the universe of elements from which members of the sets are drawn may be important. Elements in intersection set `00...`, which are in the universe but not in any known set, are called (by me) *dark matter*, and we tend to display these differently.

A diagram which produces a visualisation of each of the sets as a connected curve in the plane whose regions of intersection are connected and correspond to each of the 2^n intersection subsets is an *unweighted Venn diagram*. Weights can be assigned to each of the intersections, most naturally being proportional to the number of elements each one contains. *Weighted Venn diagrams* have the same topology as unweighted ones, but (attempt to) make the area of each region proportional to the weights. This may not be possible, if any of the weights are zero for example, or because of the geometric constraints of the diagram. Venn diagrams based on 3 circles are unable in general to represent even nonzero weights exactly, and cannot be constructed at all for $n > 3$.

Diagrams in which only those intersections with non-zero weight appear are *Euler diagrams*, and diagrams which go further and make the area of every intersection proportional to its weight are weighted Euler diagrams. For more details and rather more rigour see first the online review of Ruskey and Weston **?** and then the references it contains.

7 This document

Author	Jonathan Swinton
SVN id of this document	Id: Venn.Rnw 58 2009-09-23 22:57:05Z js229 .
Generated on	25th September, 2009
R version	R version 2.9.0 (2009-04-17)