Introducing eulerr

Johan larsson

January 1, 2018

1 Motivation

eulerr generates area-proportional euler diagrams that display set relationships (intersections, unions, and disjoints) with circles or ellipses. Euler diagrams are Venn diagrams without the requirement that all set interactions be present (whether they are empty or not). That is, depending on input, eulerr will sometimes produce Venn diagrams but sometimes not.

R features several packages that produce Euler diagrams; some of the more prominent ones (on CRAN) are

- eVenn,
- VennDiagram,
- venn,
- colorfulVennPlot, and
- venneuler.

The last of these (**venneuler**) was the primary inspiration for this package, along with the refinements that Fredrickson has presented on his blog and made available in his javascript venn.js. eulerAPE, which was the first program to uses ellipses instead of circles, has also been instrumental in the design of **eulerr**. The downside to **eulerAPE** is that it only handles three sets that all need to intersect.

enneuler, on the other hand, will take any number of sets (in theory), yet has been known to produce imperfect solutions for set configurations that have perfect such. And unlike **eulerAPE**, it is restricted to circles (as is **venn.js**).

2 Enter eulerr

eulerr is based on the improvements to venneuler that Ben Fredrickson introduced with **venn.js** but has been programmed from scratch, uses different optimizers, and returns statistics featured in **venneuler** and **eulerAPE** as well as allows a range of different inputs and conditioning on additional variables. Moreover, it can model set relationships with ellipses for any number of sets involed.

2.1 Input

At the time of writing, it is possible to provide input to **eulerr** as either

- a named numeric vector with set combinations as disjoint set combinations or unions (depending on how the argument `type` is set),
- a matrix or data frame of logicals with columns representing sets and rows the set relationships for each observation, or
- a list of sample spaces.

```
library(eulerr)
options(digits = 4)

# Input in the form of a named numeric vector
fit1 <- euler(c("A" = 25, "B" = 5, "C" = 5,
                "A&B" = 5, "A&C" = 5, "B&C" = 3,
                "A&B&C" = 3))

# Input as a matrix of logicals
set.seed(1)
mat <- cbind(
  A = sample(c(TRUE, TRUE, FALSE), 50, TRUE),
  B = sample(c(TRUE, FALSE), 50, TRUE),
  C = sample(c(TRUE, FALSE, FALSE, FALSE), 50, TRUE)
)
fit2 <- euler(mat)
```

2.2 Fit

We inspect our results by printing the eulerr object

```
fit2

##       original fitted residuals regionError
## A           10 10.164    -0.164       0.003
## B            7  7.138    -0.138       0.003
## C            5  5.119    -0.119       0.002
## A&B         15 15.173    -0.173       0.003
## A&C          1  0.571     0.429       0.010
## B&C          2  1.802     0.198       0.005
## A&B&C        5  5.141    -0.141       0.003
##
## diagError: 0.01
## stress:    0.001
```

or directly access and plot the residuals.

```
# Cleveland dot plot of the residuals
lattice::dotplot(resid(fit2), xlab = "",
                 panel = function(...) {
                   panel.abline(v = 0, lty = 2)
                   panel.dotplot(...)
                 })
```

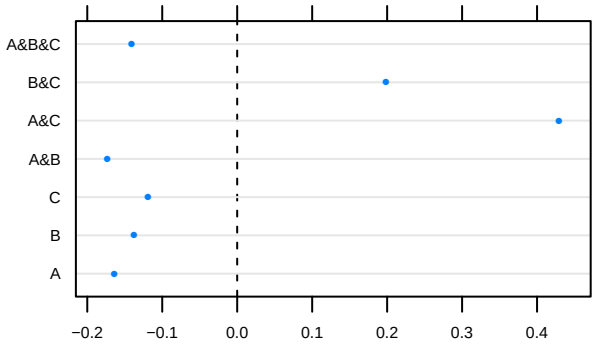

Figure 1. Residuals for the fit diagram.

This shows us that the `A&C` intersection is somewhat overrepresented in `fit2`. Given that these residuals are on the scale of the original values, however, the residuals are arguably of little concern.

As an alternative, we could plot the circles in another program by retrieving their coordinates and radii.

```
coef(fit2)

##         h       k     r
## A  0.6842  0.5594 3.144
## B -0.7753  1.2227 3.051
## C -1.8227 -1.2227 2.005
```

2.3 Goodness-of-fit

To tell if we can trust our solution, we use two goodness-of-fit measures: the stress statistic from venneuler [1],

$$\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} y_i^2}$$

where $\hat{y}_i$ is an ordinary least squares estimate from the regression of the fitted areas on the original areas that is being explored during optimization,

and the *diagError* statistic from **eulerAPE** [**?**]:

$$\max_{i=1,2,\ldots,n} \left| \frac{y_i}{\sum y_i} - \frac{\hat{y}_i}{\sum \hat{y}_i} \right|$$

In our example, the diagError is and our stress is 5.865×10^{-4}, suggesting that the fit is accurate.

We can now be confident that eulerr provides a reasonable representation of our input using circles. Were it otherwise, we might try to use ellipses instead. Wilkinson [1] features a difficult combination that it manages to fit with a reasonably small error; with **eulerr**, however, we can get rid of that error entirely (Figure 2).

```
wilkinson2012 <-  c(A = 4, B = 6, C = 3, D = 2, E = 7, F = 3,
                    "A&B" = 2, "A&F" = 2, "B&C" = 2, "B&D" = 1,
                    "B&F" = 2, "C&D" = 1, "D&E" = 1, "E&F" = 1,
                    "A&B&F" = 1, "B&C&D" = 1)
fit3 <- euler(wilkinson2012, shape = "ellipse")
```

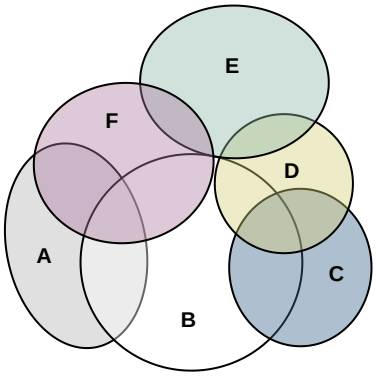

Figure 2. A difficult combination modelled with ellipses.

If we still lack a good fit after having tried ellipses, we would do best to stop here and look for another way to visualize our data. (I suggest the excellent UpSetR package.)

2.4 Visualization

No we get to the fun part: plotting our diagram. This is easy, as well as highly customizable, with **eulerr**. The default parameters (Figure 3) can easily be adjusted to suit anyobody's needs (Figure 4).

```
plot(fit2)
```

```
# Remove fills, vary border type, and switch fontface.
plot(fit2, fill = "transparent", lty = 1:3, fontface = 4)
```

Plotting is provided through the excellent lattice and interfaces all of its bells and whistles to allow for extensive customization of the resulting plots.

eulerr's default color palette is taken from qualpalr—another package that I have developed—which uses color difference algorithms to generate distinct qualitative color palettes.

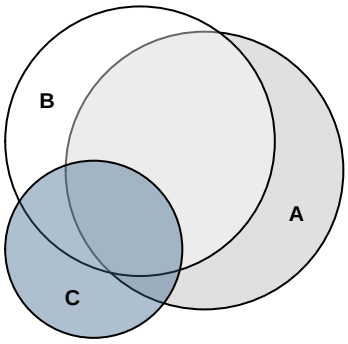

Figure 3. The default graphical parameters.

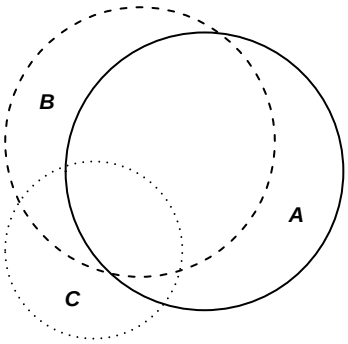

Figure 4. Adjusted graphical parameters.

3 Acknowledgements

eulerr would not be possible without Ben Fredrickson's work on **venn.js** or Leland Wilkinson's **venneuler**.

References

[1] L. Wilkinson. Exact and approximate area-proportional circular Venn and Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):321–331, February 2012. ISSN 1077-2626. doi: 10.1109/TVCG.2011.56.