eulerr under the hood

Johan larsson

January 1, 2018

1 Introduction

eulerr relies on an extensive machinery to turn user input into a pretty Euler diagram. Little of this requires any tinkering from the user. To make that happen, however, **eulerr** needs to make several well-formed decisions about the design of the diagram on behalf of the user, which is not a trivial task.

This document outlines the implementation of **eulerr** from input to output. It is designed to be an evolving documentation on the innards of the program.

2 Input

Euler diagrams present relationships between sets, wherefore the data must describe these relationships, either directly or indirectly. **eulerr** allows several alternatives for this data, namely,

- intersections and relative complements[1],
- unions and identities[2],
- a matrix of binary (or boolean) indices[3],
- a list of sample spaces[4], or
- a two- or three-way table[5].

As an additional feature for the matrix form, the user may supply a factor variable with which to split the data set before fitting the diagram, which sometimes improves diagrams where the set relationships vary across categories.

Whichever type of input is provided, **eulerr** will translate it to the first and second types, *intersections and relative complements* and *unions and identities*, which will be used in the steps to come.

The Euler diagram is then fit in two steps: first, an initial layout is formed with circles using only the sets' pairwise relationships. Second, this layout is fine-tuned taking all $2^N - 1$ intersections into consideration.

[1] $A \setminus B = 3 \quad B \setminus A = 2 \quad A \cap B = 1$

[2] $A = 4 \quad B = 3 \quad A \cap B = 1$

[3] $$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

[4] $A = \{a, b, c, d\}$
$B = \{a, e, f\}$

[5]

	A	A^c
B	1	2
B^c	3	0

3 Initial layout

For our initial layout, we adopt a constrained version of multi-dimensional scaling (MDS) that has been adapted from **venn.js** [1], which in turn is a modification of an algorithm used in **venneuler** [2]. In it, we consider only the pairwise intersections between sets, attempting to position their respective shapes so as to minimize the difference between the separation between their centers required to obtain an optimal overlap and the actual overlap of the shapes in the diagram.

This problem is unfortunately intractable for ellipses, being that there is an infinite number of ways by which we can position two ellipses to obtain a given overlap. Thus, we restrict ourselves to circles in our initial layout, for which we can use the circle–circle overlap formula (1) to numerically find the required distance, d, for each pairwise relationship.

$$O_{ij} = r_i^2 \arccos\left(\frac{d_{ij}^2 + r_i^2 - r_j^2}{2d_{ij}r_i}\right) + r_j^2 \arccos\left(\frac{d_{ij}^2 + r_j^2 - r_i^2}{2d_{ij}r_j}\right) - \frac{1}{2}\sqrt{(-d_{ij} + r_i + r_j)(d_{ij} + r_i - r_j)(d_{ij} - r_i + r_j)(d_{ij} + r_i + r_j)}, \quad (1)$$

where r_i and r_j are the radii of the circles representing the i^{th} and j^{th} sets respectively, O_{ij} their overlap, and d_{ij} their separation.

Setting $r_i = \sqrt{F_i/\pi}$, where F_i is the size of the i^{th} set, we are able to obtain d numerically using the squared difference between O and the desired overlap as loss function (2),

$$\mathcal{L}(d_{ij}) = \left(O_{ij} - (F_i \cap F_j)\right)^2, \quad \text{for } i < j \leq n, \quad (2)$$

which we optimize using `optimize()`[6] from **stats**.

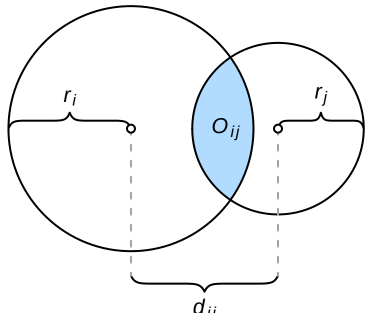


Figure 1. The circle–circle overlap is computed as a function of the discs' separation (d_{ij}), radii (r_i, r_j), and area of overlap (O_{ij}).

[6] According to the documentation, `optimize()` consists of a "combination of golden section search and successive parabolic interpolation."

For a two-set combination, this is all we need to plot an exact diagram, given that we now have the two circles' radii and separation and may place the circles arbitrarily as long as their separation, d, remains the same. This is not, however, the case with more than two sets.

With three or more sets, the circles need to be arranged so that they interfere minimally with one another. In some cases, the set configuration allows this to be accomplished flawlessly, but often, compromises must me made. As is often the case in this context, this turns out to be another optimization problem. It can be tackled in many ways; **eulerr**'s approach is based on a method developed by Frederickson [3], which the author describes as constrained multi-dimensional scaling.

The algorithm tries to position the circles so that the separation between each pair of circles matches the separation required from (2).

If the two sets are disjoint, however, the algorithm is indifferent to the relative locations of those circles as long as they do not intersect. The equivalent applies to subset sets: as long as the circle representing the smaller set remains within the larger circle, their locations are free to vary. In all other cases, the loss function (3) is the residual sums of squares of the optimal separation of circles, d, that we found in (1), and the actual distance in the layout we are currently exploring.

$$\mathcal{L}(h,k) = \sum_{1 \leq i < j \leq N} \begin{cases} 0 & F_i \cap F_j = \emptyset \text{ and } O_{ij} = 0 \\ 0 & (F_i \subseteq F_j \text{ or } F_i \supseteq F_j) \text{ and } O_{ij} = 0 \\ \left(\left(h_i - h_j\right)^2 + \left(k_i - k_j\right)^2 - d_{ij}^2 \right)^2 & \text{otherwise} \end{cases} . \tag{3}$$

The analytical gradient (4) is retrieved as usual by taking the derivative of the loss function,

$$\vec{\nabla} f(h_i) = \sum_{j=1}^{N} \begin{cases} \vec{0} & F_i \cap F_j = \emptyset \text{ and } O_{ij} = 0 \\ \vec{0} & (F_i \subseteq F_j \text{ or } F_i \supseteq F_j) \text{ and } O_{ij} = 0 \\ 4\left(h_i - h_j\right)\left(\left(h_i - h_j\right)^2 + \left(k_i - k_j\right)^2 - d_{ij}^2 \right) & \text{otherwise,} \end{cases} \tag{4}$$

where $\vec{\nabla} f(k_i)$ is found as in (4) with h_i swapped for k_i (and vice versa).

The Hessian (5) for our loss function is given next. However, because the current release of R suffers from a bug[7] causing the analytical Hessian to be updated improperly, the current release of **eulerr** instead relies on the numerical approximation of the Hessian offered by the optimizer.

[7] The current development version of R features a fix for this bug; **eulerr** will be updated to use (5) as soon as it is introduced in a stable version of R.

$$\boldsymbol{H}(h,k) = \sum_{1 \leq i < j \leq N} \begin{bmatrix} 4\left(\left(h_i-h_j\right)^2+\left(k_i-k_j\right)^2-d_{ij}^2\right)+8\left(h_i-h_j\right)^2 & \cdots & 8\left(h_i-h_j\right)\left(k_i-k_j\right) \\ \vdots & \ddots & \vdots \\ 8\left(k_i-k_j\right)\left(h_i-h_j\right) & \cdots & 4\left(\left(h_i-h_j\right)^2+\left(k_i-k_j\right)^2-d_{ij}^2\right)+8\left(k_i-k_j\right)^2 \end{bmatrix}. \tag{5}$$

Note that the constraints given in (3) and (4) still apply to each element of (5) and have been omitted for convenience only.

We optimize (3) using the nonlinear optimizer `nlm()` from the R core package **stats**. The underlying code for `nlm()` was written by Schnabel et al. [4]. It was ported to R by Saikat DebRoy and the R Core team [5] from a previous FORTRAN to C translation by Richard H. Jones. `nlm()` consists of a system of Newton-type algorithms and performs well for difficult problems [6].

The initial layout outlined above will sometimes turn up perfect diagrams, but only reliably so when the diagram is completely determined by its pairwise intersections. More pertinently, we have not yet considered the higher-order intersections in our algorithm and neither have we approached the problem of using ellipses—as we set out to do.

4 Final layout

We now need to account for all the sets' intersections and, consequently, all the overlaps in the diagram. The goal is to map each area uniquely to a subset of the data from the input and for this purpose we will use the sets' intersections and the relative complements of these intersections, for which we will use the shorthand ω. We introduced this form in Section 2, but now define it rigorously in Definition 1.

Definition 1. *For a family of* N *sets,* $F = F_1, F_2, \ldots, F_N$*, and their* $n = 2^N - 1$ *intersections, we define* ω *as the intersections of these sets and their relative complements, such that*

$$\begin{aligned}
\omega_1 &= F_1 \setminus \bigcap_{i=2}^{N} F_i \\
\omega_2 &= \bigcap_{i=1}^{2} F_i \setminus \bigcap_{i=3}^{N} F_i \\
&\vdots \\
\omega_n &= \bigcap_{i=1}^{N} F_i
\end{aligned}$$

with

$$\sum_{i=1}^{n} \omega_i = \bigcup_{j=1}^{N} F_i.$$

Analogously to ω*, we also introduce the* &*-operator, such that*

$$F_i \& F_j = (F_i \cap F_j) \setminus (F_i \cap F_j)^c.$$

The fitted diagram's area-equivalents for ω will be defined as A, so that an exact diagram requires that $\omega_i = A_i$ for $i = 1, 2, \ldots, 2^N - 1$, where N is the number of sets in the input.

In Section 3, we restricted ourselves to circles but now extend ourselves also to ellipses. From now on, we abandon the practice of treating circles separately—they are only a special case of ellipses, and, hence, everything that applies to an ellipse does so equally for a circle.

4.1 Intersecting ellipses

We now need the ellipses' points of intersections. **eulerr**'s approach to this is outlined in Richter-Gebert [7] and based in *projective*, as opposed to *Euclidean*, geometry.

To collect all the intersection points, we naturally need only to consider two ellipses at a time. The canonical form of an ellipse is given by

$$\frac{[(x-h)\cos\phi + (y-k)\sin\phi]^2}{a^2} + \frac{[(x-h)\sin\phi - (y-k)\cos\phi]^2}{b^2} = 1,$$

where ϕ is the counter-clockwise angle from the positive x-axis to the semi-major axis a, b is the semi-minor axis, and h, k are the x- and y-coordinates, respectively, of ellipse's center (Figure 2).

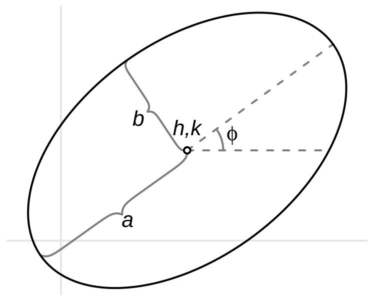


Figure 2. A rotated ellipse with semimajor axis a, semiminor axis b, rotation ϕ, and center h, k.

However, because an ellipse is a conic[8] it can be represented in quadric form,

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

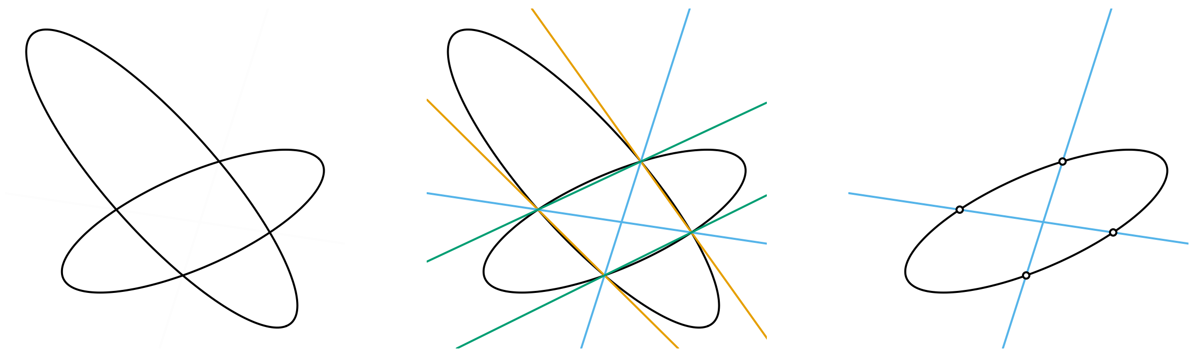
that in turn can be represented as a matrix,

$$\begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix},$$

which is the form we need to intersect our ellipses. We now proceed to

1. form three degenerate conics from a linear combination of the two ellipses we wish to intersect,
2. split one of these degenerate conics into two lines, and
3. intersect one of the ellipses with these lines, yielding 0 to 4 intersection points points (Figure 3).

[8] The circle, parabola, and hyperbola are the other types of conics.



(a) Our objective is two intersect these two ellipses.

(b) Three degenerate conics (orange, teal, and blue) are formed from the linear combination of our two ellipses.

(c) One of the degenerate conics is split it into two lines and intersected with one of the ellipses to yield four intersection points.

Figure 3. The process used to intersect two ellipses, here yielding four points. This figure was inspired by an example from Richter-Gebert [7].

4.2 Overlap areas

Using the intersection points of a set of ellipses that we retrieved in Section 4.1, we can now find the overlap of these ellipses. We are only interested in the points that are *contained within all of these ellipses*, which together form a geometric shape consisting of a convex polygon, the sides of which are made up of straight lines between consecutive points, and a set of elliptical arcs—one for each pair of points (Figure 4).

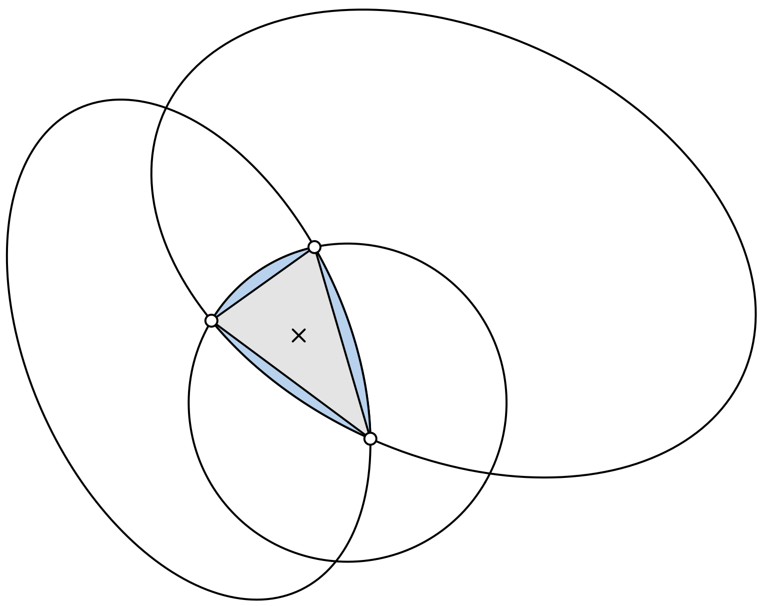


Figure 4. The overlap area between three ellipses is the sum of a convex polygon (in grey) and 2–3 ellipse segments (in blue).

We continue by ordering the points around their centroid. It is then trivial to find the area of the polygon section since it is always convex [8]. Now, because each elliptical segment is formed from the arcs that connect successive points, we can establish the segments' areas algorithmically [9]. For each ellipse and its related pair of points (located at angles θ_0 and θ_1 from the semimajor axis), we proceed to find its area by

1. centering the ellipse at $(0, 0)$,
2. normalizing its rotation, which is not needed to compute the area,
3. integrating the ellipse over $[0, \theta_0]$ and $[0, \theta_1]$, producing elliptical sectors $F(\theta_0)$ and $F(\theta_1)$,
4. subtracting the smaller ($F(\theta_0)$) of these sectors from the larger ($F(\theta_0)$), and
5. subtracting the triangle section to finally find the segment area,

$$F(\theta_1) - F(\theta_0) - \frac{1}{2}\left|x_1 y_0 - x_0 y_1\right|,$$

$$\text{where } F(\theta) = \frac{a}{b}\left[\theta - \arctan\left(\frac{(b-a)\sin 2\theta}{b+a+(b-a)\cos 2\theta}\right)\right].$$

This procedure is illustrated in Figure 5. Note that there are situations where this algorithm is altered, such that when the sector angle ranges beyond π—we refer the interested reader to Eberly [9].

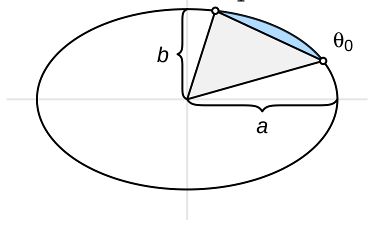


Figure 5. The elliptical segment in blue is found by first subtracting the elliptical sector from $(a, 0)$ to θ_0 from the one from $(a, 0)$ to θ_1 and then subtracting the triangle part (in grey).

Finally, the area of the overlap is then obtained by adding the area of the polygon and all the elliptical arcs together.

Note that this does not yet give us the areas that we require, namely A: the area-equivalents to the set intersections and relative complements from Definition 1. For this, we must decompose the overlap areas so that each area maps uniquely to a subspace of the set configuration. This, however, is simply a matter of transversing down the hierarchy of overlaps and subtracting the higher-order overlaps from the lower-order ones. For a three-set relationship of sets A, B, and C, for instance, this means subtracting the $A \cap B \cap C$ overlap from the $A \cap B$ one to retrieve the equivalent of $(A \cap B) \setminus C$.

The exact algorithm may in rare instances[9], break down, the culprit being numerical precision issues that occur when ellipses are tangent or completely overlap. In these cases, the algorithm will approximate the area of the involved overlap by

[9] 1 out of approximately 7000 in our simulations.

1. spreading points across the ellipses using Vogel's method (see Section 6.1 for a brief introduction),
2. identifying the points that are inside the intersection via the inequality

$$\frac{[(x-h)\cos\phi + (y-k)\sin\phi]^2}{a^2} + \frac{[(x-h)\sin\phi - (y-k)\cos\phi]^2}{b^2} < 1,$$

   where x and y are the coordinates of the sampled points, and finally
3. approximating the area by multiplying the proportion of points inside the overlap with the area of the ellipse.

With this in place, we are now able to compute the areas of all intersections and their relative complements, ω, up to numerical precision.

4.3 Final optimization

We feed the initial layout computed in Section 3 to the optimizer—once again we employ `nlm()` from **stats** but now also provide the option to use ellipses rather than circles, allowing the "circles" to rotate and the relation between the semiaxes to vary, altogether rendering five parameters to optimize per set and ellipse (or three if we restrict ourselves to circles). For each iteration of the optimizer, the areas of all intersections are analyzed and a measure of loss returned. The loss we use is the same as that in **venneuler** [2], namely *stress*,

$$\frac{\sum_{i=1}^{n}(A_i - \beta\omega_i)^2}{\sum_{i=1}^{n} A_i}, \qquad (6)$$

where

$$\beta = \frac{\sum_{i=1}^{n} A_i\omega_i}{\sum_{i=1}^{n} \omega_i^2}.$$

This is equivalent to linear regression through the origin, where β is the slope of the regression line (Figure 6).

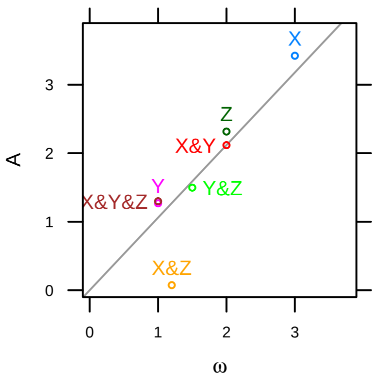


Figure 6. Optimizing via stress is analogous to least-squares linear regression through the origin. ω is the set of unique quantities in the input (Definition 1) and A the respective areas in the diagram.

4.4 Last-ditch optimization

If the fitted diagram is still inexact after the procedure in Section 4.3, we offer a final step in which we pass the parameters on to a last-ditch optimizer. The weapon of choice[10] is a *differential evolution algorithm* from the R package **RcppDE** [10]—a port of the **DEoptim** package [11] from C to C++.

[10] We conducted thorough benchmarking, that we opt not to report here, to decide upon an algorithm for this step.

The solutions offered by **RcppDE** often avoid local minima but may be inefficient in local search regions; this shortcoming can be remedied by fine tuning with a local optimizer [12]—once more, we rely on `nlm()` to serve this purpose.

By default, this last-ditch step is activated only when we have a three-set diagram with ellipses and a diagError (7) above 0.001[11] The reason being that the method is considerably more computationally intensive.

[11] The choice of if and when this last-ditch optimizer is activated is left to the user via simple commands to the main function of the package.

5 Goodness of fit

Every Euler diagram must be investigated for its adequacy in representing the input. Exact Euler diagrams are not always possible When **eulerr** cannot find a perfect solution, it offers an approximate one instead, the adequacy of which has to be measured in a standardized way. For this purpose we adopt two measures: *stress* [2], which is also the loss metric we use in our final optimization step and is used in **venneuler**, as well as *diagError* [13], which is used by **eulerAPE**.

The stress metric is not easily grasped but can be transformed into a rough analogue of the correlation coefficient via $r = \sqrt{1 - \text{stress}^2}$.

diagError, meanwhile, is given by

$$\max_{i=1,2,\dots,n} \left| \frac{\omega_i}{\sum_{i=1}^n \omega_i} - \frac{A_i}{\sum_{i=1}^n A_i} \right|, \tag{7}$$

which is the maximum *absolute* difference of the proportion of any ω to the respective unique area of the diagram.

6 Plotting

Once we have ascertained that our Euler diagram fits well, we can turn to visualizing the solution. For this purpose, **eulerr** leverages the **Lattice** graphics system [14] for R to offer intuitive and granular control over the output.

Plotting the ellipses is straightforward using the parametrization of a rotated ellipse,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} h + a\cos\theta \\ k + b\sin\theta \end{bmatrix}, \quad \text{where } \theta \in [0, 2\pi],\ a, b > 0.$$

Most users will also prefer to label the ellipses and their intersections with text and this, however, is considerably more involved.

6.1 Labeling

Labeling the ellipses is complicated since the shapes of the intersections often are irregular, lacking well-defined centers; we know of no analytical solution to this problem. As usual, however, the next-best option turns out to be a numerical one. First, we locate a point that is inside the required region by spreading points across the discs involved in the set intersection. To distribute the points, we use a modification of *Vogel's method* [15, 16] adapted to ellipses. Vogel's method spreads points across a disc using

$$p_k = \begin{bmatrix} \rho_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} r\sqrt{\frac{k}{n}} \\ \pi(3-\sqrt{5})(k-1) \end{bmatrix} \quad \text{for } k = 1, 2, \ldots, n. \tag{8}$$

In our modification, we scale, rotate, and translate the points formed in (8) to match the candidate ellipse. We rely, as before, on projective geometry to carry out the transformations in one go:

$$p' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix},$$

where h, k translates, ϕ rotates, and a, b stretches the ellipse.

After we spread our points throughout the ellipse and find a point, p'_i, that is contained in our desired intersection, we proceed to optimize its position numerically. The position we are looking for is that which maximizes the distance to the closest ellipse in our diagram to provide as much margin as possible for the label. This is a maximization problem with a loss function equal to

$$\mathcal{L}(x, y) = \min_{i=1,2,\ldots,N} f(x, y, h_i, k_i, a_i, b_i, \phi_i) \tag{9}$$

where f is the function that determines the distance from a point (x, y) to the ellipse defined by h, k, a, b and ϕ.

Similarly to fitting Euler diagrams in the general case, there appears to be no analytical solution to computing the distance from a point to an ellipse. The numerical solution we use has been described by Eberly [17] and involves solving the roots to a quartic polynomial via a robust bisection optimizer.

To optimize the location of the label, we employ a version of the *Nelder–Mead method* [18], which has been translated from a Matlab code by Kelley [19] and adapted for **eulerr** to ensure that it converges quickly and that the simplex remains within the intersection boundaries (since we want the local maximum). The method is visualized in Figure 7.

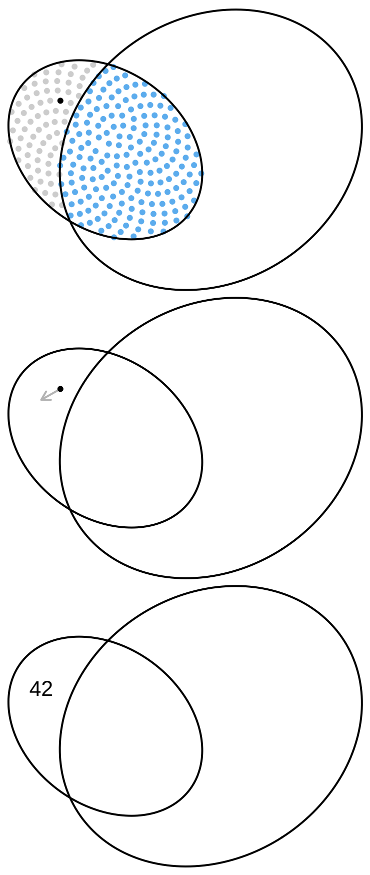


Figure 7. The method eulerr uses to locate an optimal position for a label in three steps from top to bottom: first, we spread sample points on one of the ellipses and pick one inside the intersection of interest, then we begin moving it numerically, and finally place our label.

6.2 Aesthetics

Euler diagrams display both quantitative and qualitative data. The quantitative aspect is the quantities or sizes of the sets depicted in the diagram and is visualized by the relative sizes, and possibly the labels, of the areas of the shapes—this is the main focus of this paper. The qualitative aspects, meanwhile, consist of the mapping of each set to some quality or category, such as having a certain gene or not. In the diagram, these qualities can be separated through any of the following aesthetics:

- color,
- border type,
- text labelling,
- transperancy,
- patterns,

or a combination of these. The main purpose of these aethetics is to separate out the different ellipses so that the audience may interpret the diagram with ease and clarity.

Among these aesthetics, the best choice (from a viewer perspective) appears to be color [20], which provides useful information without extraneous chart junk [21]. The issue with color, however, is that it cannot be perceived perfectly by all—8% of men and 0.4% of women in European Caucasian countries, for instance, suffer the most common form, red–green color deficiency [22]. Moreover, color is often printed at a premium in scientific publications and adds no information to a diagram of two shapes.

For these reasons, **eulerr** defaults to distinguishing ellipses with color using a color palette generated via the R package **qualpalr** [23], which automatically generates qualitative color palettes based on a perceptual model of color vision that optionally caters to color vision deficiency. This palette has been manually modified to fullfil our other objectives of avoiding using colors for two sets. The first eight colors of the pallete are visualized in Figure 8.

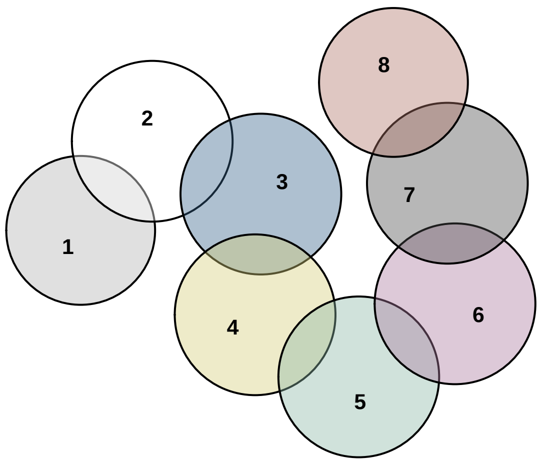


Figure 8. The eight first colors of the default color palette.

7 Normalizing dispered layouts

If there are disjoint clusters of ellipses, the optimizer will often spread these out more than is necessary, wasting space in our diagram. To tackle this, we use a SKYLINE-BL rectangle packing algorithm [24] designed specifically for **eulerr**. In it, we surround each ellipse cluster with a bounding box, pack these boxes into a bin of appropriate size and aspect ratio, and adjust the coordinates of the ellipses in the clusters to compact our diagram. As a bonus, this increases the chance of having similar layouts for different function calls.

References

[1] Ben Frederickson. venn.js: area proportional Venn and Euler diagrams in JavaScript, November 2016. URL https://github.com/benfred/venn.js. original-date: 2013-05-09T17:13:20Z.

[2] 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.

[3] Ben Frederickson. A better algorithm for area proportional venn and euler diagrams, June 2015. URL http://www.benfrederickson.com/better-venn-diagrams/.

[4] Robert B. Schnabel, John E. Koonatz, and Barry E. Weiss. A modular system of algorithms for unconstrained minimization. *ACM Trans Math Softw*, 11 (4):419–440, December 1985. ISSN 0098-3500. doi: 10.1145/6187.6192.

[5] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

[6] John C. Nash. *Nonlinear parameter optimization using R tools*. Wiley, Chichester, West Sussex, 1 edition, May 2014. ISBN 978-1-118-56928-3.

[7] Jürgen Richter-Gebert. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*. Springer, Berlin, Germany, 1 edition, February 2011. ISBN 978-3-642-17286-1.

[8] Darel R. Finley. Ultra-easy algorithm with C code sample. http://alienryderflex.com/polygon_area/, December 2006. URL http://alienryderflex.com/polygon_area/.

[9] David Eberly. The area of intersecting ellipses, November 2016. URL https://www.geometrictools.com/Documentation/AreaIntersectingEllipses.pdf.

[10] Dirk Eddelbuettel. *RcppDE: Global Optimization by Differential Evolution in C++*, 2016. URL https://CRAN.R-project.org/package=RcppDE. R package version 0.1.5.

[11] Katherine M. Mullen, David Ardia, David L. Gil, Donald Windover, and James Cline. DEoptim: An R package for global optimization by differential Evolution. *Journal of Statistical Software*, 40(6), April 2011. doi: 10.18637/jss.v040.i06. URL https://www.jstatsoft.org/article/view/v040i06.

[12] Yang Xiang, Sylvain Gubian, Brian Suomela, and Julia Hoeng. Generalized simulated annealing for global optimization: the GenSA package. *The R Journal*, 5(1):13–28, June 2013. URL https://journal.r-project.org/archive/2013/RJ-2013-002/index.html.

[13] Luana Micallef and Peter Rodgers. eulerAPE: drawing area-proportional 3-Venn diagrams using ellipses. *PLOS ONE*, 9(7):e101717, July 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0101717.

[14] Deepayan Sarkar. *Lattice: multivariate data visualization with R*. Use R! Springer, New York, USA, 1 edition, 2008. ISBN 978-0-387-75968-5. URL http://www.springer.com/us/book/9780387759685.

[15] Mary K. Arthur. Point picking and distributing on the disc the sphere. Final ARL-TR-7333, US Army Research Laboratory, Weapons and Materials Research Directorate, Abedeen, USA, July 2015. URL www.dtic.mil/get-tr-doc/pdf?AD=ADA626479.

[16] H. Vogel. A better way to construct the sunflower head. *Mathematical Biosciences*, 44(3-4):179–189, 1979. doi: 10.1016/0025-5564(79)90080-4.

[17] Eberly. Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid, November 2016. URL https://www.geometrictools.com/Documentation/DistancePointEllipseEllipsoid.pdf.

[18] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7 (4):308–313, January 1965. ISSN 0010-4620. doi: 10.1093/comjnl/7.4.308. URL https://academic.oup.com/comjnl/article/7/4/308/354237/A-Simplex-Method-for-Function-Minimization.

[19] C. T. Kelley. *Iterative methods for optimization*. Number 18 in Frontiers in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1 edition, 1999. ISBN 0-89871-433-8.

[20] Andrew Blake. *The impact of graphical choices on the perception of Euler diagrams*. Ph.D. dissertation, Brighton University, Brighton, UK, February 2016. URL http://eprints.brighton.ac.uk/15754/1/main.pdf.

[21] Edward R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, USA, 2 edition, May 2001. ISBN 978-1-930824-13-3.

[22] Jennifer Birch. Worldwide prevalence of red-green color deficiency. *Journal of the Optical Society of America. A, Optics, Image Science, and Vision*, 29(3): 313–320, March 2012. ISSN 1520-8532.

[23] Johan Larsson. *qualpalr: Automatic Generation of Qualitative Color Palettes*, 2016. URL https://cran.r-project.org/package=qualpalr. R package version 0.3.1.

[24] Jukka Jylänki. A thousand ways to pack the bin – a practical approach to two-dimensional rectangle bin packing, February 2010. URL http://clb.demon.fi/files/RectangleBinPack.pdf.