Lund University

Bachelor Thesis

eulerr: Area-Proportional Euler Diagrams with Ellipses

Johan Larsson

supervised by
Peter Gustafsson

December 15, 2017

Contents

Abstract

Euler diagrams may moreover be area-proportional, which is to say that each separate surface of the diagram maps to some quantity. (This was the case with the diagram with defined in the second paragraph.) This is a rational form for a Euler diagram—only its geometries are necessary to interpret it, letting us, for instance, to discard numbers without crucial loss of information; the same cannot be said for a Venn diagram.

1 Background

The visual display of data represents an intuitive form of data presentation. Data visualizations work on multiple dimensions and possess the potential to convey intricate relationships that single statistics or tables never can.

Such visualizations, however, are only effective if their aesthetics convey relationships. Consider, for instance, a disc with a radius of 2 cm labelled *Men*—it says nothing by itself; yet if we juxtapose it with a 1 cm-radius disc labelled *Children*, the graphic starts to become informative: it now displays the relation between two quantities. Now, if we intersect the two discs, so as to produce an overlap, we have successfully visualized the relative proportions of men and children, as well as their intersection. The diagram we have constructed is an *Euler diagram* (Figure 1.1).

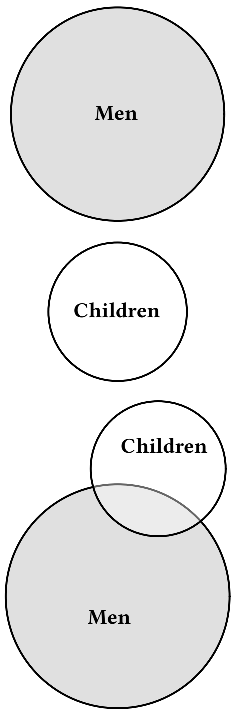


Figure 1.1. The merits of Euler diagrams.

The Euler diagram, originally proposed by Leonard Euler [1], is a superset of the obiquiteous *Venn diagram*: a staple of introductory text books in statistics and research disciplines such as biomedicine and geology. Venn and Euler diagrams differ in that the the former require all intersections to be present—even if they are empty—whilst Euler diagrams do not.

Euler diagrams may moreover be area-proportional, which is to say that each separate surface of the diagram maps to some quantity. (This was the case with the diagram with defined in the second paragraph.) This is a rational form for a Euler diagram—only its geometries are necessary to interpret it, letting us, for instance, to discard numbers without crucial loss of information; the same cannot be said for a Venn diagram.

Area-proportional Euler diagrams may be fashioned out of any closed shape, and have been implemented for triangles [2], rectangles [2], ellipses [3], smooth curves [4], polygons [2], and circles [2, 5, 6]. Circles is the popular choice, and for good reason, since they are easiest to interpret [7]. In spite of this, circles do not always lend themselves to accurate representations. Consider, for instance the following three-set relationship:

$$\begin{gathered} A = B = C = 2, \\ A \cap B = A \cap C = B \cap C = 1 \\ A \cap B \cap C = 0. \end{gathered}$$

There is no way to visualize this relationship perfectly with circles because they cannot be arranged so that the $A \cap B \cap C$ overlap remains empty whilst $A \cap B$, $A \cap C$, and $B \cap C$ are non-empty. With ellipses, however, we can solve this problem since they can both be stretched and rotated, enabling a perfect fit (Figure 1.2). In essence, circles feature three degrees of freedom: a center consisting of x- and y-coordinates h and k, as well as a radius r. Ellipses, meanwhile, have five: the aforementioned h and k, a semi-major axis a, a semi-minor axis b, and an angle of rotation ϕ.

With four or more intersecting sets, exact circular Euler diagrams are in fact impossible, given that we *require* 15 intersections but with four circles can yield at most 13 unique overlaps. This is not the case with ellipses, which may intersect in up to four, rather than two, points. The only implementation of elliptical Euler diagrams is found in **eulerAPE** [3], yet it only supports three sets that are moreover required to intersect. The diagram in Figure 1.3, for instance, would not be possible with **eulerAPE**.

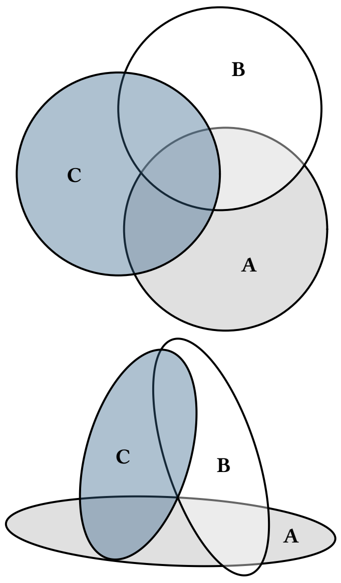


Figure 1.2. A set relationship depicted erroneously with circles and perfectly with ellipses.

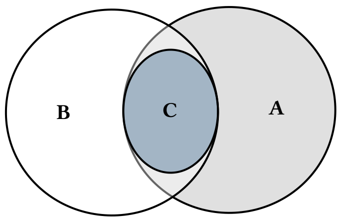


Figure 1.3. A Euler diagram with a subset relationship.

Euler diagrams do not reduce to analytical solutions [8] and have to be solved numerically. Most implementations accomplish this in two steps, first finding a rough initial configuration that is then finalized in a second, more accurate, algorithm. For the initial configuration, **eulerAPE** [9], for instance, uses a greedy algorithm that tries to minimize the error in the three-way intersection. **venneuler** [5] uses multi-dimensional scaling with Jacobian distances, taking only pairwise relationships into account. **venn.js** [10] uses a constrained version of the latter that is instead based on euclidean distances and separately runs a greedy algorithm, picking the best fit out of the two. **Vennerable** [2] uses a simple method of computing the required pairwise distances between circles and then adjusts the largest to find accurate two-way overlaps. All the algorithms use circles in the initial configuration.

Diagrams with more than two sets normally require additional tuning, which is often dealt with in a final configuration step. The prerequisite for this is that we first compute the areas of the overlaps in order to establish how well our diagram fits the input. Calculating overlaps, however, is no trivial task, particularly not for ellipses. This is evident in that most methods resort to approximations such as quad-tree binning [5], polygon intersecting [6], or restricting the algorithm to pairwise overlaps [2]. Frederickson [10], contrastingly, computes areas exactly, yet only for circles. These three approaches moreover allow only circles, whilst Micallef and Rodgers [3], on the other hand, compute areas exactly also for ellipses, though only for a maximum of three.

Compared to approximative methods, exact algorithms require that we know the intersection points of the ellipses. There are two known approaches to this and both treat the ellipses in pairs. One method solve the system of equations formed by the two ellipses, which necessitates solving a fourth-degree polynomial; another method represents the ellipses as conics in projective geometry, which reduces to solving a third-degree polynomial. Both methods are accurate up to floating-point precision.

With all the intersection points at hand, it is possible to derive the areas of the overlaps. Frederickson [11] has published a method for circles and a similar method is used by Micallef and Rodgers [3] for three ellipses. No method, however, has so far been published that generalizes these methods to diagrams of more than three ellipses or ellipses with subset or disjoint relationships.

In the final configuration step, this area-algorithm is used with a numerical optimizer to tune the parameters of the diagram. All previously considered packages treat this as a minimization problem but their optimization targets vary. **venn.js** uses residual sums of squares[1]), **venneuler** uses the *stress* metric[2], which is computed as the residual sums of squares over total sums of squares, **Vennerable** uses Chow's [12] idealistic function[3], and **eulerAPE** uses a proportional loss function[4] that severely punishes missing overlaps—in fact, it becomes undefined if such areas exist.

venn.js relies on a Nelder–Mead optimizer for the final fit, which was originally written specifically for the package. **venneuler**, on the other hand, begins with a steepest descent optimizer with gradient approximation that is then fine-tuned with coordinate descent. **eulerAPE** and **Vennerable**, meanwhile, uses a hill climibing algorithm.

1.1 Aims

Elliptical Euler diagrams have previously not been implemented for more than three sets or for three-set diagrams with subset or disjoint relationships. This is the motivation for this thesis, with which we aim to present a method and implementation for constructing and visualizing Euler diagrams for sets of any numbers using ellipses and exact area computations.

[1] **venn.js**'s loss function is

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

where n is the number of overlaps, ω_i the size of the i:th intersection, and A_i the corresponding overlap's area in the diagram.

[2] **venneuler**'s stress metric is defined as

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

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

[3] This idealistic function (used in **Vennerable**) is defined as

$$\sum_{i=1}^{n}\left(\frac{\omega_i}{\sum_{i=1}^{n}\omega_i} - \frac{A_i}{\sum_{i=1}^{n} A_i}\right)^2 + \sum_{k<j<n}\left(A_i - A_j\right),$$

where the sets and their respective overlaps corresponding to the indices i and j have been ordered so that $i < j \implies \omega_i < \omega_j$.

[4] **eulerAPE**'s cost function is defined as

$$\frac{1}{n}\sum_{i=1}^{n}\frac{(\omega_i - A_i)^2}{A_i}.$$

2 Method

Constructing an Euler diagram is much like fitting a statistical model in that we have

1. data,
2. a model to fit the data on,
3. tests to assess the model's fit, and
4. a presentation of the result.

In the following sections, we explain how **eulerr** tackles each item.

2.1 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[5],
- unions and identities[6],
- a matrix of binary (or boolean) indices[7],
- a list of sample spaces[8], or
- a two- or three-way table[9].

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. This is offered for the user's convenience, since it may be that the fit improves after such a split.

Whichever type of input is provided, **eulerr** translates it to the first, *intersections and relative complements* (Definition 2.1), which is the form used later in the loss functions of the initial and final optimizers.

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

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

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

[8] $A = \{ab, bb, bc\}$
$B = \{aa, bc, cc\}$
$C = \{bb, bb, cc\}$

[9]

Survived?	No	Yes
Child	52.00	57.00
Adult	1438.00	654.00

Definition 2.1. *For a family of* N *sets,* $F = F_1, F_2, \dots, F_N$*, and their* $n = 2^N - 1$ *intersections, we define* ω *as the intersections of these sets and their relative complements, such that*

$$
\begin{aligned}
\omega_1 &= F_1 \setminus \bigcap_{j=2}^{N} F_j \\
\omega_2 &= (F_2 \cap F_3) \setminus \bigcap_{j=3}^{n} F_j \\
\omega_3 &= \bigcap_{i=1}^{3} F_i \setminus \bigcap_{j=4}^{N} F_j \\
&\vdots \\
\omega_n &= \bigcap_{j=1}^{N} F_j
\end{aligned}
$$

with

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

Analogously to ω*, and for convenience, we also introduce the & operator as*

$$F_j \& F_k = (F_j \cap F_k) \setminus (F_j \cap F_k)^C = \omega_i,$$

where i *in this instance is the index of the binary identifier of the intersection between* F_j *and* F_k*.*

With the input translated into our form of choice, the Euler diagram is fit in two steps: first, an initial configuration is formed with circles using only the sets' pairwise relationships. Second, this configuration is fine tuned taking all $2^N - 1$ overlaps into account.

2.2 Initial configuration

For our initial configuration, we rely on a constrained version of multi-dimensional scaling (MDS) from **venn.js** [10], which is a modification of a method from **venneuler** [5]. In it, we consider the pairwise relationsships between the sets and attempt to position their respective shapes so as to minimize the difference between the distance between their centers required to obtain an optimal overlap (ω) and the actual overlap between 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, for which we can use the circle–circle overlap formula (2.1) to numerically find the required distance, d, for each set of two ellipses,

$$O_{ij} = r_i^2 \arccos\left(\frac{d_{ij}^2 + r_i^2 - r_j^2}{2d_{ij}r_i}\right) + r_j^2 \arccos\left(\frac{d_{ij}^2 + r_j^2 - r_i^2}{2d_{ij}r_j}\right) - \frac{1}{2}\sqrt{(-d_{ij} + r_i + r_j)(d_{ij} + r_i - r_j)(d_{ij} - r_i + r_j)(d_{ij} + r_i + r_j)}, \quad (2.1)$$

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} the distance between them.

We are looking for d, which we find easily from knowing O and r. Our loss function is the squared difference between O and ω (the desired overlap),

$$\mathcal{L}(d_{ij}) = (O_{ij} - \omega_{ij})^2, \quad \text{for } i < j \neq< n \quad (2.2)$$

which we optimize using R's built-in `optimize()`[10]. Convergence is fast—neglible next to our later optimization procedures. For a two-set combination, this is all we need to plot an exact diagram, given that we now have each the two circles' radii, their separation, and can place the circles arbitrarily on a two-dimensional plane as long as d remains the same. This is not, however, with more than two sets, in which case we proceed to the next step.

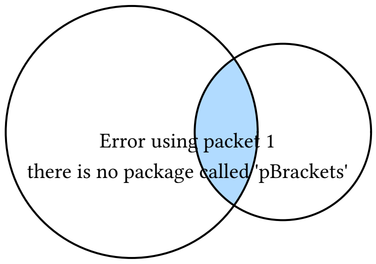


Figure 2.1. The circle–circle overlap is computed as a function of the discs' separation (d_{ij}), radii (r_i, r_j), and area of overlap (O_{ij}).

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

Given these optimal pairwise distances, we now try to arrange the circles representing the sets for our initial layout. This can be accomplished in many ways; **eulerr**'s approach is based on a method developed by Frederickson [13], which the author describes as constrained multi-dimensional scaling—a form of numeric optimization.

The algorithm tries to position the circles on a plane so that the separation between each pair of circles matches the separation required from (2.2). If the two sets are disjoint, however, it is unconcerned with the relative locations of those circles as long as they do not intersect. The same 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 (2.3) is the normal sums of squares between the optimal distance between two sets, d, that we found in (2.1) and the actual distance in the layout we are currently exploring.

$$\mathcal{L}(h,k) = \sum_{0 \leq i < j \leq N} \begin{cases} 0 & F_i \cap F_j = \emptyset \text{ and } O_{ij} = \emptyset \\ 0 & (F_i \subseteq F_j \text{ or } F_i \supseteq F_j) \text{ and } O_{ij} = \emptyset \\ \left(\left(h_i - h_j\right)^2 + \left(k_i - k_j\right)^2 - d_{ij}^2\right)^2 & \text{otherwise} \end{cases}. \quad (2.3)$$

The analytical gradient (2.4) is retrieved as usual by taking the derivative of the loss function:

$$\vec{\nabla} f(h_i) = \sum_{j=1}^{N} \begin{cases} \vec{0} & F_i \cap F_j = \emptyset \text{ and } O_{ij} = \emptyset \\ \vec{0} & (F_i \subseteq F_j \text{ or } F_i \supseteq F_j) \text{ and } O_{ij} = \emptyset \\ 4\left(h_i - h_j\right)\left(\left(h_i - h_j\right)^2 + \left(k_i - k_j\right)^2 - d_{ij}^2\right) & \text{otherwise,} \end{cases} \tag{2.4}$$

where $\vec{\nabla} f(k_i)$ is found as in (2.4) with h_i swapped for k_i (and vice versa). Because it boosts convergence, we also compute the Hessian (2.5):

$$\boldsymbol{H} = \sum_{0 \leq i < j \leq N} \begin{bmatrix} 4\left(\left(h_i-h_j\right)^2+\left(k_i-k_j\right)^2-d_{ij}^2\right)+8\left(h_i-h_j\right)^2 & \cdots & 8\left(h_i-h_j\right)\left(k_i-k_j\right) \\ \vdots & \ddots & \vdots \\ 8\left(k_i-k_j\right)\left(h_i-h_j\right) & \cdots & 4\left(\left(h_i-h_j\right)^2+\left(k_i-k_j\right)^2-d_{ij}^2\right)+8\left(k_i-k_j\right)^2 \end{bmatrix}. \tag{2.5}$$

Note that the constraints given in (2.3) and (2.4) still apply to each element of (2.5) and have been omitted for practical reasons only.

We optimize (2.3) using the nonlinear optimizer `nlm()` from the R core package **stats**. The underlying code for `nlm()` was written by Schnabel et al. [14]. It was ported to R by Saikat DebRoy and the R Core team [15] from a previous translation from FORTRAN to C by Richard H. Jones. `nlm()` consists of a system of Newton-type algorithms and performs well for difficult problems [16].

This initial configuration will be be perfect for any two-set combination using circles but may need to be adjusted if more circles are used. More pertinently, we have not yet allowed for ellipses in our diagrams.

2.3 Final configuration

We now need to account for all the sets' relationships and, consequently, all the intersections and overlaps in the diagram. Initially, we restricted ourselves to circles but now extend ourselves also to ellipses. From here on we abandon the practice of treating circles separately—they are only special forms of ellipses and everything that applies to an ellipse does so equally for a circle.

2.3.1 Intersecting ellipses

As we saw in the Background, we now need to have all the ellipses' points of intersections at hand. **eulerr**'s approach to this is outlined in Richter-Gebert [17] 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. However, because an ellipse is a conic[11] they can be represented via the quadric form

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \tag{2.6}$$

[11] The circle, parabola, and hyperbola are also conics.

(2.6) can in turn 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 a degenerate conic from the solution to the system consisting of the two conics we wish to intersect,
2. split this degenerate conic into a pencil of two lines, and finally
3. intersect the remaining conic with this pencil, yielding 0 to 4 intersection points points (Figure 2.2).

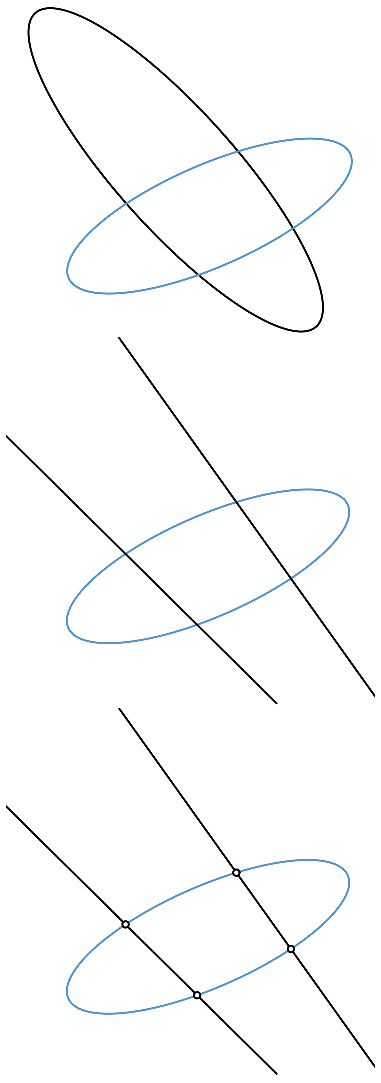


Figure 2.2. The process (from top to bottom) used to intersect two ellipses, here yielding four points.

2.3.2 Computing the areas of the intersections

After we have all the intersection points, we find the overlap by examining the intersection points that are formed from the intersections of the ellipses we are currently exploring and that are simultaneously contained within all of these ellipses. The elliptical arcs that connect these points along with the polygon with sides formed from joining the points with straight lines together make up the area of the overlap (Figure 2.3).

Since the polygon part is always convex, it is easy to find its area using the *triangle method*. To find the areas of the elliptical segments, we first order all the points in clockwise order[12]. Then, we acknowledge that each elliptical segment is formed from an arc of the ellipse that is shared by both points[13]. Now that we have two points on an ellipse, we can find the area of the ellipse segment using an algorithm from Eberly [18]. To proceed, we

[12] It makes no difference if we sort them in counter-clockwise order instead.

[13] Because there is sometimes two arcs connecting the pairs of points, we simply compute both areas and pick the smaller.

1. center their ellipse at $(0, 0)$,
2. normalize its rotation, which is not needed to compute the area,

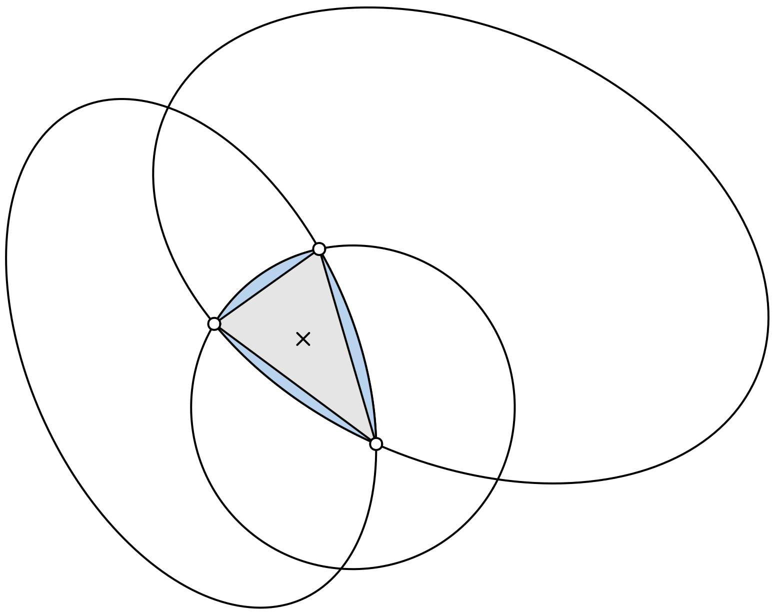


Figure 2.3. The overlap area between three ellipses is the sum of a convex polygon (in grey) and 2–3 ellipse segments (in blue).

3. integrate the ellipse from 0 to ϕ_0 and ϕ_1 to produce two elliptical sectors,
4. subtract the smaller of these sectors from the larger, and
5. subtract the triangle section to finally find the segment area (13).

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

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

This procedure is illustrated in Figure 2.4.

The exact algorith may in rare instances[14], breaks down. The culprit is numerical approximation issues that occur because ellipses are close-to tangent to one another or completely overlap. In these cases, the algorithm will resort to approximation of the involved overlap area by

1. spreading points across the ellipses using Vogel's method (see Labeling 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

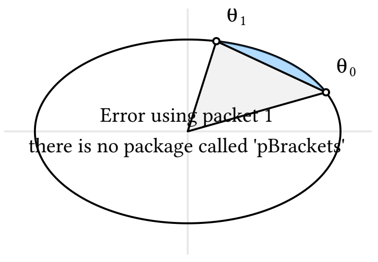


Figure 2.4. The elliptical segment in blue is found by first subtracting the elliptical sector from $(a, 0)$ to θ_0 from the one from $(a, 0)$ to θ_1 and then subtracting the triangle part (in grey).

[14] 1 out of approximately 7000 in our simulations.

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 up to numerical precision.

We feed the initial layout computed in Initial configuration 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** [5], namely *stress*, is the residual sums of squares over the total sums of squares,

$$\text{Stress} = \frac{\sum_{i=1}^{n}(A_i - \beta\omega_i)^2}{\sum_{i=1}^{n} A_i}, \qquad (2.7)$$

where $\beta = \sum_{i=1}^{n} A_i\omega_i / \sum_{i=1}^{n} \omega_i^2$. This boils down to fitting a linear regression through the origin, where β is the coefficient that makes the areas in the diagram proportional to the sizes of the sets.

2.3.3 Last-ditch optimization

If we are using ellipses and are faced with an error in our fit, we offer a final step in which we pass the parameters on to a last-ditch optimizer. The weapon of choice[15] is a *differential evolution algorithm* from the R package **RcppDE** [19], which is a port from C to C++ of the **DEoptim** package [20].

[15] 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 minimi but may be coarse in the sense that they do not reach the precision offer by other methods; this shortcoming can be remedied by a fine tuning from a loca optimizer [21]—again, we employ `nlm()` from **stats** 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 (2.8) above 0.001[16] The reason being that the method is considerably more intensive computationally.

[16] The user, however, is free to choose if and when to activate the optimizer using standard arguments to the main function of the package.

2.4 Goodness of fit

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* [5], which is also the loss metric we use in our final optimization step and is used in **venneuler**, as well as *diagError* [3], which is used by **eulerAPE**.

The stress metric (2.7), which we first defined in the Background and then reiterated in Final configuration,

The stress metric does not reduce to any overt interpretation but can be transformed into a rough analogue of the correlation coefficient via $r = \sqrt{1 - \text{stress}^2}$.

diagError, meanwhile, is given by

$$\text{diagError} = \max_{i=1,2,\ldots,n} \left| \frac{\omega_i}{\sum_{i=1}^{n} \omega_i} - \frac{A_i}{\sum_{i=1}^{n} A_i} \right|, \tag{2.8}$$

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

3 Results

The only R packages that feature area-proportional Euler diagrams are **eulerr**, **venneuler**, **Vennerable**, and **d3VennR**. The latter is an interface to the **venn.js** script that has been discussed previously, but because it features an outdated version of the script and only produces images as html, we call **venn.js** directly using the javascript engine **V8** via the R package of the same name [22]. Only **eulerr**, **venn.js**, and **venneuler** support more than three sets, which is why there are only three-set results for **Vennerable** and **eulerAPE**.

The packages used here were

- **eulerr** 3.1.0,
- **eulerAPE** 3.0.0,
- **venn.js** 0.2.14,
- **venneuler** 1.1-0, and
- **Vennerable** 3.1.0.9000.

The results for **eulerAPE** were computed on a laptop computer[17] The remaining results were computed on an Amazon EC2 cloud-based computing instance[18]

[17] The specification of the computer was

- Microsoft Windows Pro 10 x64
- Intel® Core$^{\text{TM}}$ i7-4500U CPU @ 1.80GHz, 2 cores
- 8 GB memory

[18] This was a m4.large instance with the following specifications:

- Ubuntu 16.04 x64
- 2.4 GHz Intel Xeon®E5-2676 v3 (Broadwell) CPU, 2 cores
- 8 GB memory

3.1 Case studies

We begin our examination of **eulerr** by studying a difficult set relationship from Wilkinson [5],

$$\begin{gathered}
A = 4 \quad B = 6 \quad C = 3 \quad D = 2 \quad E = 7 \quad F = 3\\
A\&B = 2 \quad A\&F = 2 \quad B\&C = 2 \quad B\&D = 1\\
B\&F = 2 \quad C\&D = 1 \quad D\&E = 1 \quad E\&F = 1\\
A\&B\&F = 1 \quad B\&C\&D = 1,
\end{gathered}$$

where we use the & operator as defined in Definition 2.1. We fit this specification with **venneuler** and **eulerr**, in the latter case using both circles and ellipses (Figure 3.1).

This example showcases the improvement gained from using ellipses and also the small benefit that **eulerr** offers relative to **venneuler**.

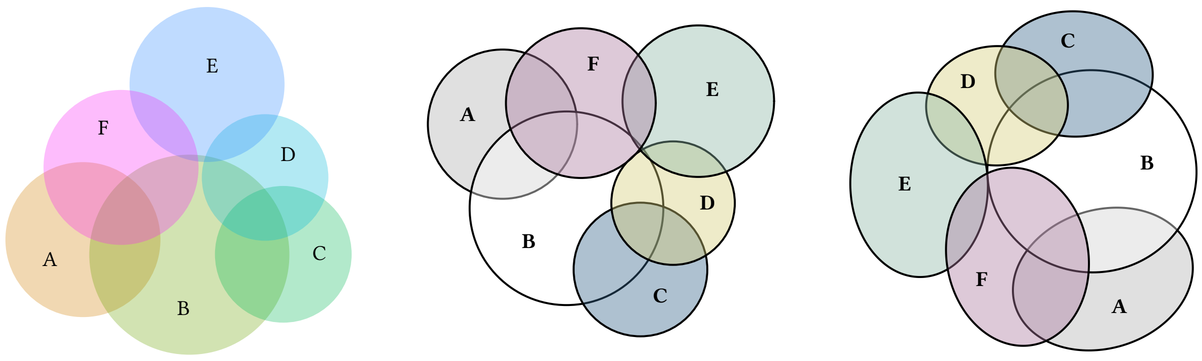


Figure 3.1. A comparison of a Euler diagram generated with **venneuler** with two generated from **eulerr** with circles and ellipses respectively. The stress of the solutions are 0.006, 0.004, and 0.000 respectively.

Micallef and Rodgers [3] featured a diagram from Lenz and Fornoni [23]. We will not dwell upon the inaccuracies of the original diagram here but rather contrast the results of **eulerAPE** with those of **eulerr**. The data from the original diagram was

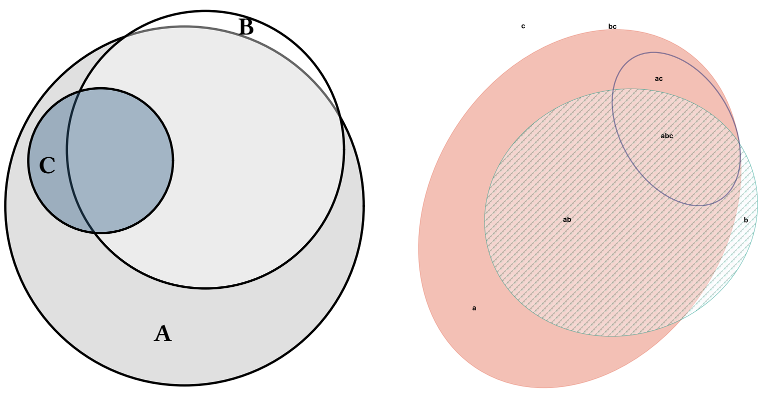
$$A = 0.36, \quad B = 0.03, \quad C = 0,$$
$$A\&B = 0.41, \quad A\&C = 0.04, \quad B\&C = 0, \quad \text{and}$$
$$A\&B\&C = 0.11.$$

However, because **eulerAPE** cannot fit diagrams that lack some intersections, the authors instead replaced 0 with 0.00001. Since the diagram was reproduced with ellipses, we will do the same with **eulerr** but set the areas as they are in the original data.

The fits from both packages are exact (Figure 3.2). Although we told **eulerr** to use ellipses in the fit, the algorithm stuck to circles, which, given that the fit is exact, is the appropriate choice since circles are easier to interpret. The reason **eulerAPE** did not is that it tries to keep the three shapes intersecting (albeit marginally), which cannot be done with circles if the layout is to be exact.

We also note that the labels of the nonexistant overlaps in the diagram from **eulerAPE** have been placed at seemingly arbitrary locations[19].

[19] In the original publication [3], the labels were manually adjusted.



(a) The diagram from **eulerr**. **(b)** The diagram from **eulerAPE**.

Figure 3.2. Diagrams from **eulerAPE** and **eulerr** based on data from a diagram in Lenz and Fornoni [23]. The diagrams have not been modified in any way save for converting the diagram from **eulerAPE** from .svg to .pdf.

	Amelie	Pulp Fiction	Miss Congeniality	Armageddon	Rashomon	Coyote Ugly
Amelie	38,753					
Pulp Fiction	15,197	70,153				
Miss Congeniality	1,829	3,854	37,837			
Armageddon	1,218	6,593	10,536	40,345		
Rashomon	2,087	2,799	132	143	6,209	
Coyote Ugly	610	2,206	5,965	5,699	38	15,611

Table 3.1. The data from the Netflix Prize dataset used in Frederickson [24].

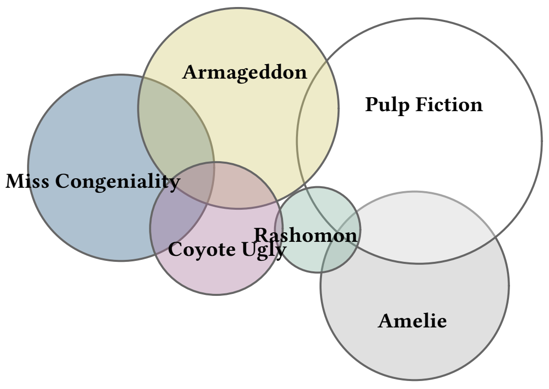
To conclude the case studies part of this thesis, we look at a diagram that was featured on website of the author of **venn.js** [24], which used the *Netflix Prize Dataset*[20] from which he

> [...] picked 6 movies, kind of at random – and then represented them using the set of users that gave the movie a 5 star rating.

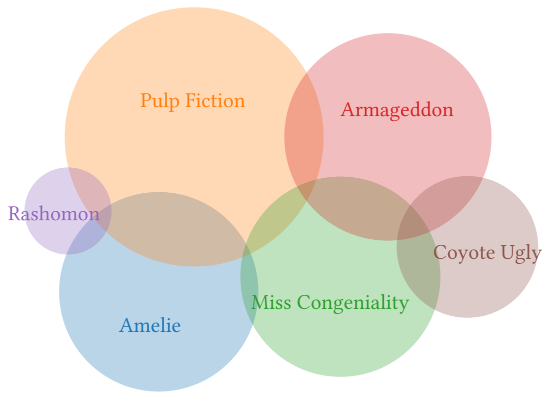
[20] The dataset is no longer available on this webpage, but has been archived at `https://www.kaggle.com/netflix-inc/netflix-prize-data` for those who are interested.

The movies were Amelie, Pulp Fiction, Miss Congeniality, Armageddon, Rashomon, and Coyote Ugly. The data only includes pairwise relationships (Table 3.1).

The fit from **eulerr** is marginally better than that of **venn.js**. The stress and diagError of the **eulerr** diagram (Figure 3.4a) are 0.003 and 0.014 respectively, whilst the same figures are 0.004 and 0.015 for the **venn.js** diagram (Figure 3.4b).



(a) The fit from **eulerr** with a stress of 0.003 and diagError of 0.014.



(b) The fit from **venn.js** with a stress of 0.004 and diagError of 0.015.

Figure 3.4. The Euler diagrams generated from the Netflix Prize data. The difference in the two layouts primarily concern the placements of *Rashomon* and *Miss Congeniality*.

3.2 Consistency

To compare the consistency among **eulerr**, **venneuler**, **eulerAPE**, **venn.js**, and **Vennerable**, we generate random diagrams of circles and ellipses (separately), compute their areas, and attempt to reproduce the original diagrams using the various software. We restrict ourselves to diagrams consisting of between three and eight shapes.

For the circles, we sample radii (r_i) and coordinates (h_i and k_i) from

$$\begin{aligned} r_i &\sim \mathcal{U}(0.2, 0.6) \\ h_i, k_i &\sim \mathcal{U}(0, 1), \end{aligned} \tag{3.1}$$

where N is the number of shapes. For the ellipses, we sample semiaxes (a_i and b_i), coordinates (h_i and k_i), and rotation axes (ϕ_i) from

$$\begin{aligned} h_i, k_i &\sim \mathcal{U}(0, 1) \\ r_i &\sim \mathcal{U}(0.2, 0.6) \\ c_i &\sim \mathcal{U}(1/2, 2) \\ a_i &= r_i c_i \\ b_i &= r_i / c_i \\ \phi_i &\sim \mathcal{U}(0, \pi), \end{aligned} \tag{3.2}$$

where c is the ratio between the semiaxes.

Next, we compute the required areas, ω (from Definition 2.1), for each iteration and fit a Euler diagram using the aforementioned packages. Finally, we compute and return *diagError* (2.8) and score each diagram as a *success* if its *diagError* is lower than 0.01, that is, if no portion of the diagram is 1% off (in absolute terms) from that of the input; note that this is always achievable since our Euler diagrams are formed from sampled diagrams.

For each number of shapes ($i = 3, 4, \ldots, 8$) we run the simulations until we have achieved a 95% confidence interval around $\hat{p}$, the proportion of successful diagrams, that is no wider than 2%. The confidence that we generate is the standard asymptotic interval,

$$\mathcal{I}(\hat{p})_{0.95} = \hat{p} \pm z_{0.95}\sqrt{\frac{\hat{p}(\hat{p}-1)}{n}}$$

The algorithm is formalized in Algorithm 1. The left panel of Figure 3.5 shows the results of our simulation.

For circular diagrams, **eulerr** outperforms **Vennerable** and **venneuler** in consistency by considerable margin and is on par with **venn.js** and **eulerAPE** (Figure 3.5). **eulerr** and **eulerAPE** are the only packages that successfully fits all of the circular diagrams (although **eulerAPE** only accepts a subset of our three-set diagrams). **venn.js** fails for 4 sets out of 6000 diagrams. This difference, however, is not significant according to our asymptotic confidence interval.

For ellipses of three shapes, only **eulerr** passes the bar on each occasion. **eulerAPE** fails in 6 cases, which again is not significantly worse.

For ellipses of four or more sets—which only **eulerr** accepts—the consistency drops for each additional set, yet remains above 38%.

```
for $i \leftarrow 3:8$ do
    while length of each $\mathcal{I}(p_s)_{0.95} < 2\%$ and $j < 500$ do
        if circle then
            $h, k \leftarrow \mathcal{U}(0, 1)$
            $r \leftarrow \mathcal{U}(0.3, 0.6)$
            $\omega \leftarrow \texttt{findOverlaps}(h, k, r)$
        else if ellipse then
            $h, k \leftarrow \mathcal{U}(0, 1)$
            $a, b \leftarrow \mathcal{U}(0.2, 0.8)$
            $\phi \leftarrow \mathcal{U}(0, 2\pi)$
            $\omega \leftarrow \texttt{findOverlaps}(h, k, a, b, \phi)$
        end if
        $A \leftarrow \texttt{fitDiagram}(\omega)$
        $\text{diagError}_j \leftarrow \max_{k=1,2,\ldots,2^i-1} \left| \frac{\omega_k}{\sum \omega_k} - \frac{A_k}{\sum A_k} \right|$
        if $\text{diagError}_j < 0.01$ then
            successes + 1
        end if
        for all $s \leftarrow$ software package do
            $\hat{p}_s \leftarrow \text{successes}/j$
            $\mathcal{I}(\hat{p}_s)_{0.95} \leftarrow \hat{p} \pm z_{0.95}\sqrt{\frac{\hat{p}_s(\hat{p}_s-1)}{n}}$
        end for
    end while
end for
```

Algorithm 1. The algorithm used to simulate circles and ellipses, compute their areas, and fit Euler diagrams to these layouts using the different software packages.

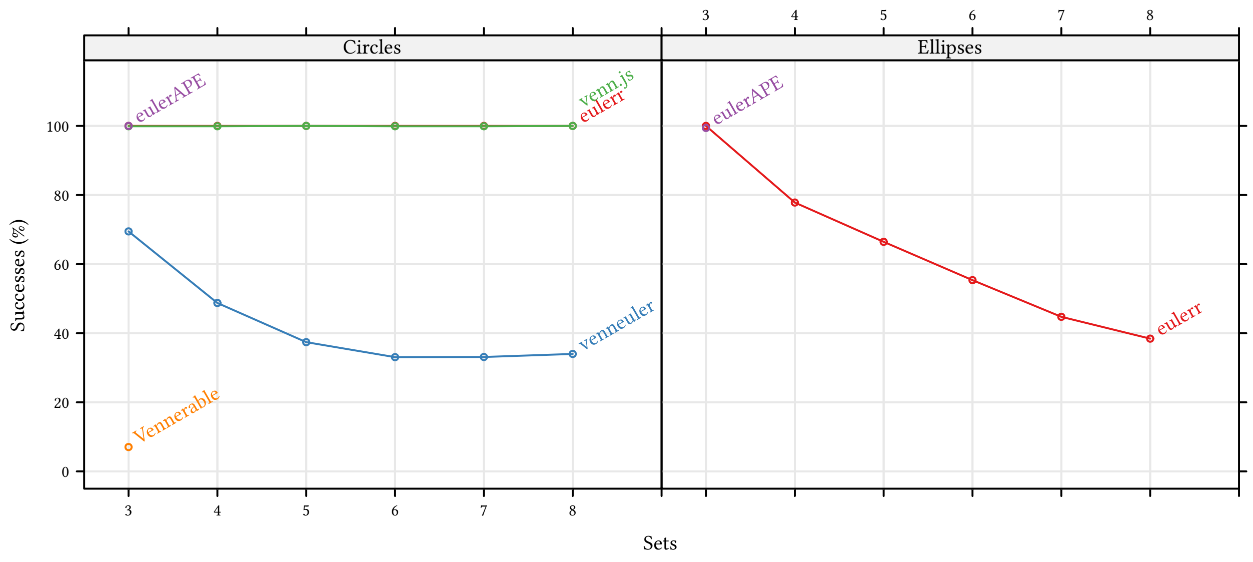


Figure 3.5. Reproducibility tests for ellipses and circles generated from the distributions from (3.1). **Vennerable** and **eulerAPE** only accepts three-set diagrams and are hence otherwise absent..

```
for $i \leftarrow 3:8$ do
    while length of each $\mathcal{I}(\hat{\mu}_s)_{0.95} < 0.02$ and $j < 500$ do
        initialize $\omega = \{\omega_1, \omega_2, \ldots, \omega_{2^i-1}\}$ to zero
        for $j \leftarrow 3:i$ do
            $j \leftarrow$ random index in $\{\omega : \omega \cap F_i \neq \emptyset\}$
            $\omega_j \leftarrow \mathcal{U}(0,1)$
        end for
        $\omega_S \leftarrow \mathcal{U}\{0,i\}$ random elements from $\{\omega : \omega = 0\}$
        $\omega_S \leftarrow \mathcal{U}(0,1)$
        fit an Euler diagram to $\omega$
    end while
end for
```

Algorithm 2. The algorithm we use to simulate random set relationships and fit them with the software under study to assess their accuracy.

Vennerable, which is only able to produce three-set diagrams, only produces accurate diagrams for 7% of the random layouts and moreover fails with an error in 6 cases.

3.3 Accuracy

In Consistency, we assessed the efficacy in reproducing diagrams with exact, but unknown, solutions. Reality, however, often presents us with relationships that lack such exact solutions. Yet, although there might not be an exact solution, one might still exist that is *good enough*, which we then want our model to find for us so that we can examine its adequacy.

To assess the proficiency of finding approximate solutions, we generate random set relationships that may or may not have exact solutions and fit Euler diagram using the software under study. For each $i = 3, 4, \ldots, 8$ sets we initialize $2^i - 1$ permutations of set combinations, select one for each set, and initialize these to a number in $\mathcal{U}(0,1)$. After this, we pick 0 to $\binom{N-2}{1}$ elements from the $2^i - i - 1$ remaining permutations and assign to them a number from $\mathcal{U}(0,1)$ as before.

As in Consistency, we run our simulations until we achieve a 95% confidence interval around the mean diagError no wider than 0.02, but for a minimum of 1000 iterations. The confidence level we use is the based on the t-distribution:

$$\mathcal{I}(\hat{\mu})_{0.95} = \hat{\mu} \pm t_{0.95}\frac{s}{\sqrt{n}}, \tag{3.3}$$

where $\hat{\mu} = \bar{x}$. The algorithm is formalized in Algorithm 2

However, given that neither **Vennerable** nor **eulerAPE** are capable of fitting Euler diagrams that feature subset or disjoint relationships, and because fully intersecting diagrams are more difficult to fit—at least with circles—we run a separate treatment in which we modify Algorithm 2 so that every intersection is initialized to a value in $\mathcal{U}(\epsilon, 1)$, where ϵ is defined as the square root of the current machine's smallest representable difference between one and the smallest value greater than one[21].

[21] The relevant line of code used to generate this tolerance is `sqrt(.Machine$double.eps)`

From the results generated via Algorithm 2, we can surmise that elliptical **eulerr** diagrams achieve the lowest stress and diagError on average Figure 3.6. It is lower regardless of the number of sets, but is relatively more pronounced for sets of three and four shapes.

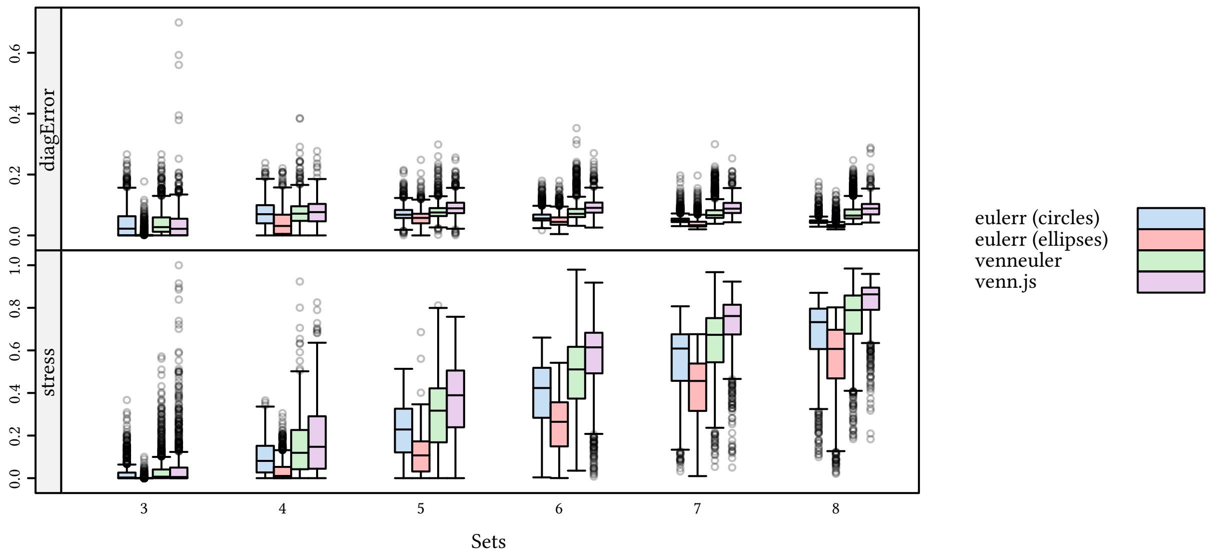


Figure 3.6. Tukey box plots of Euler diagrams based on set relationships that may or may not have perfect solutions, generated from (3.2).

We also find that among the algorithms that fit circular Euler diagrams, **eulerr** offers the lowest median stress across all the sizes of set relationships. For diagError, however, **venn.js** produces diagrams with the least loss for three sets, whilst **eulerr** scores better for all the remaining set sizes, for which **venn.js** is moreover outdone by **venneuler**.

Looking at the results from the simulation of three-set relationships with every intersection present (Figure 3.7), **eulerr** still produces the most accurate diagrams (provided that we use ellipses)—both in terms of stress and diagError. The difference next to **eulerAPE** is very small, however, with differences below 10^{-8}.

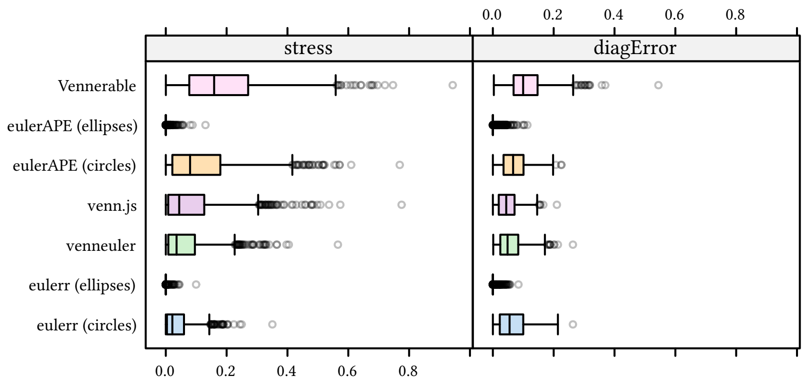


Figure 3.7. Tukey box plots of diagError and stress for Euler diagrams based on set relationships of three sets with every intersection present.

For the circular diagrams, **venn.js** achieves the lowest median diagError of 7.11×10^{-8} followed by **venneuler** at 8.219×10^{-7}, **eulerr** at 0.044, **eulerAPE** at 0.035, and **Vennerable** at 0.044. As for stress, however, the order is partially reversed with respective stress values at 4.581×10^{-14}, 7.834×10^{-12}, 0.022, 0.035, 0.044 for **eulerr**, **venneuler**, **venn.js**, **eulerAPE**, and **Vennerable** respectively.

3.4 Performance

Using the same method as in Accuracy, we generate random set relationships and measure the time it takes for each software package to form a diagram from the fit. We rely on **microbenchmark** for the benchmarks, which randomizes the order in which the packages are called between trials. Because the current version of **venn.js** has not been implemented in R, nor any version of **eulerAPE**, we omit these packages from the performance benchmarks.

The violin plot in Figure 3.8 shows that **eulerr** is speedier for circular diagrams up to seven sets, at which point **venneuler** catches up and then surpasses the performance of **eulerr**. For a three set diagram, for instance, **eulerr** takes a median of 0.007 seconds to find a fit, whilst **venneuler** takes 0.414 seconds, and **Vennerable** 0.056 seconds. For eight sets, meanwhile, **eulerr** takes 2.68 seconds and **venneuler** 2.059.

The computation time for the elliptical diagrams from **eulerr** generally vary more but are still, on average, faster than **venneuler** for up to five sets—albeit with the exception of diagrams of three sets, where the resulting bimodal distribution is a consequence of the time-consuming last-ditch optimizer (Last-ditch optimization) that is activated by default if the fit is not error-free.

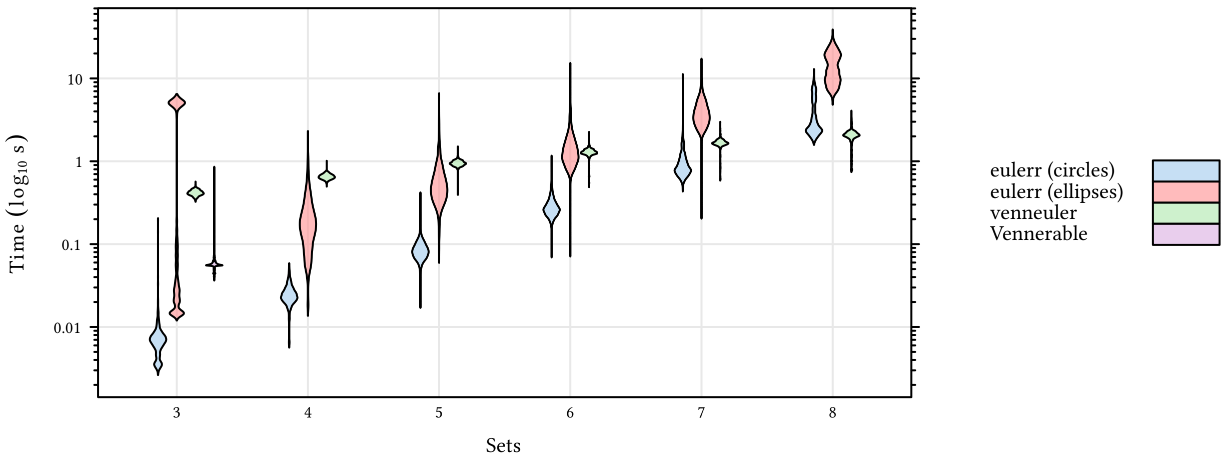


Figure 3.8. Violin plot of the performance of **eulerr**, **venneuler**, and **Vennerable** on random set relationships of three to eight sets. The density smoother is that of Sheather and Jones [25].

3.5 Availability

eulerr is available as an R package on the CRAN network [26] and is installed simply by typing

```r
install.packages("eulerr")
```

A development version and the source code for the project is maintained in a GitHub repository at `https://github.com/jolars/eulerr`. This version can be installed, provided that the **devtools** package is installed, with the following oneliner:

```r
devtools::install_github("jolars/eulerr")
```

The source code for this thesis, meanwhile, is also provided in a GitHub repository at `https://github.com/jolars/eulerrPaper`, which also hosts the code used to generate the results for this thesis is provided.

Finally, there is a **shiny** [27] web application for **eulerr** at `http://jolars.co/eulerr`, which features a slightly slimmed-down version that only allows two types of input, offers less freedom in customizing the diagram, and does not feature the last-ditch optimizer. (See Last-ditch optimization). The web application s available to everyone with a internet connection and somewhat-recent web browser.

4 Discussion

In this paper, we have presented a novel method for generating Euler diagrams for any number of sets using ellipses. We have shown that the package performs better or on par with all the other packages for most of the analyses in this thesis. In addition, the method is speedier than the other R packages for set relationships with up to seven sets, wherafter it is eclipsed by **venneuler**.

eulerr reproduced all the circular diagrams generated in Consistency within our margin of error. No other package managed this task, although the failing diagrams of both **eulerAPE** and **venn.js** numbered in the single digits.

In contrast, **venneuler** was not (by our standards) able to adequately reproduce between 30 and 65% of these diagrams, which, however, was still better than the results on a previous test2 [28]. Some of these diagrams feature disjoint or subset relationships, which **venneuler** have problems fitting on account of its initial optimizer. As we discussed in the Background, **eulerr**, **venn.js**, and **venneuler** all use multi-dimensional scaling for their initial diagram. **venneuler**, however, unnecessarily restricts the locations of disjoint and subset circles. Given, for instance, two disjoint sets, the algorithm will attempt to place them tangent to one another; otherwise, the optimizer will report loss. Likewise, for a subset relationship, **venneuler** tries to place the smaller circle at the exact midpoint of the enclosing shape.

For the fit to be accurate, however, those restrictions are pointless as long as the sets remain disjoint or subset and almost always feature other locations that fulfill those requirements. For **venneuler**, this becomes problematic when the required positions interfere with locations of other sets that need to use that space. The MDS algorithm from **venn.js** and **eulerr** circumvents this by assigning a loss and gradient of zero when the pairwise set intersection *and* the candidate circles are disjoint or subset. This makes it easier for the starting layout to find a good initial layout.

Another reason for the mediocre results of **venneuler** could be both that the optimizer terminates prematurely in case the relative reduction in stress is considered negligible between iterations *or* that the number of iterations is generally too low.

eulerr managed to also reproduce all of the elliptical three-set diagrams perfectly but failed for a considerable portion as the number of sets in the input surged. For three-set diagrams, **eulerr** employs a rigorous last-ditch optimizer in case the fit it not adequate[22] after

[22] Here, we define *adequate* as a diagram with diagError below 0.001.

the "final" optimization step. This step is time-consuming (as we saw in Performance), but if activated would yield better results also for sets of more than three shapes.

The elliptical diagrams from **eulerr** are more accuracte for random set relationships (that might lack exact solutions) than all the other packages in this thesis, although the elliptical diagrams from **eulerAPE** were almost as accurate. Elliptical Euler diagrams are more successful since they, as we briefly covered in the Background, allow two addition degrees of freedom for each set. The gain is greatest for three sets and shrinks as we add more sets. This is what we expect: complicated inputs require complicated models and there are many diagrams that even elliptical Euler diagrams cannot fit—consider, for instance, the complicated geometry of the Venn diagram in Figure 4.1 that is required to represent all the intersections between six sets.

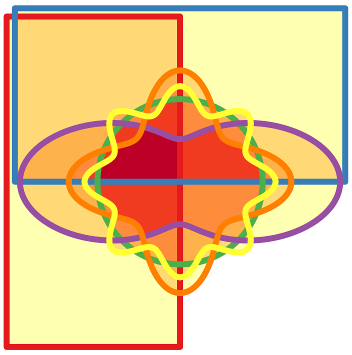


Figure 4.1. A six-set Venn diagram using Edward's method [29]; this diagram was generated with **Vennerable**.

Considering only circular Euler diagrams, **eulerr** remains the best choice for both diagError and stress in all cases except for three-set diagrams, where it ranks best in terms of stress but not diagError[23]; **venn.js** scores lowest. Interestingly—as well as surprisingly, given the results from Consistency—**venn.js** performs worse than **venneuler** in all of the remaining cases. We can only speculate as to reasons for this, but it possible that the Nelder–Mead variant that was written for **venn.js** has trouble finding a good fit; indeed, this was our experience when designing **eulerr**, which at one point featured a variety of the same optimizer; in our case, `nlm()` from R, which uses numerical derivatives, was found to be superior.

[23] It might be appropriate to note that **eulerr** and **venn.js** both attempt to minimize the sums of squares and not diagError.

For three to seven sets, **eulerr** is faster than both **Vennerable** and **venneuler**. The reason for this is partly the implementation in C++ and the use of the Armadillo library [30] provided by the interfaces **Rcpp** [31] and **RcppArmadillo** [32]. The second reason is the exact-area computations that outperform the quadtree binning of **venneuler** for the lower amount of sets. Paradoxically, this is also why the performance of **eulerr** suffers as the number of sets increase beyond seven.

The bottleneck in **eulerr**'s performance is the final optimization. It has to examine every possible intersection when computing the areas. For eight sets, for instance, this means investigating 255 intersections. In theory, the algorithm should converge in $O(2^n)$ time. Wilkinson [5], meanwhile, report convergence in $O(n)$ time. There is evidence of this in Figure 3.8 and it is possible that a fine-tuned version of **venneuler**'s algorithm might outperform **eulerr** also for lower number of sets. Regardless, the increasing computational demand of large number of sets make fitting such diagrams prohibitive at least in exploratory data analysis. Although it is questionable whether there is much use for Euler diagrams for such complicated diagrams—only rarely willl they surrender to an adequate fit—future versions of this algorithm should consider implementing approximate area-calculations when the number of sets is large.

Fitting diagrams is substantially harder with ellipses than with circles. There are many more local minimi that might hobble our optimizer and since our initial configuration only considers circles, such minimi cannot reliably be avoided before the final optimization step. And because we moreover default to a semi-local optimizer in this final step, it is not uncommon that we become stuck before finding the global minimum. The last-ditch optimizer battles such minimi with brute force: it tries a considerable number of permutations and uses the previous fits (initial and semi-final) only to set up constraints for the algorithm. The downside to using this optimizer is that it, in contrast to us, people[24], does not prefer circles over ellipses, which means that we might get diagrams with ellipses when circles would do just as well.

[24] And the other optimizer that begins with an initial layout of circles.

4.1 Conclusion

In the majority of scenarios examined in this thesis, **eulerr** offers solutions with the least error among all of the packages tested. For three sets, its performance is equalled by the elliptical diagrams that **eulerAPE** produce, which, however, impose the restrictions that **eulerr** do not, namely, that there be no disjoint or subset relationships. In addition, **eulerr** offers considerable control over the visual output, which is levied via the **lattice** package for R.

eulerAPE's restriction to three sets is discussed by the authors of the package, who motivate this limitation with the propensity of Euler diagrams with more sets to lack adequate solutions and that their complexity make implementations difficult [9]. Whilst it is true that inputs with more than three sets do not always reduce to adequate Euler diagrams, it is our stance that those that do warrant a software implementation that can help the user find them, given that Euler diagrams are intuitive graphics that are appreciated by most viewers.

The primary shortcoming of **eulerr** lies in its failure to consistently reproduce random elliptical diagrams (see Consistency), implying that the performance in producing random set relationships (see Accuracy) must have potential to improve. This is not an intractable problem and we argue that it can be overcome either by

- relying on brute force global optimizers that thoroughly examine the search space and attempt to optimize, and possibly parallelize, these to make larger diagrams fit within a reasonable time span, or
- designing an algorithm for the initial configuration that works specifically for ellipses.

Whichever direction future algorithms take, we believe that the advances presented in this thesis should serve as another step forward towards more accurate Euler diagrams.

4.2 Acknowledgements

Thanks to blabla.

Appendices

A Visualization

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 [33] for R to offer intuitive and granular control over the output.

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

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

Often, however, we would also like to label the ellipses and their intersections with text and this is considerably more involved.

A.1 Labeling

Labeling the ellipses is difficult because the shapes of the intersections often are irregular, lacking a well-defined center; 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* [34, 35] adapted to ellipses. Vogel's method spreads points across a disc using

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

In our modification, we scale, rotate, and translate the points formed in (A.1) to match the candidate ellipse. We rely, as before, on projective geometry to carry out the transformations in one go:

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

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 maximin problem with a loss function equal to

$$\max_{x, y \in \mathbb{R}^2} \min_{i = 1, 2, \ldots, N} f(x, y, h_i, k_i, a_i, b_i, \phi_i) \tag{A.2}$$

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 in Eberly [36] 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* [37] which has been translated from Kelley [38] and adapted for **eulerr** to ensure that is coverges quickly and that the simplex remains within the intersection boundaries (since we want the local maximum). The method is visualized in Figure A.1.

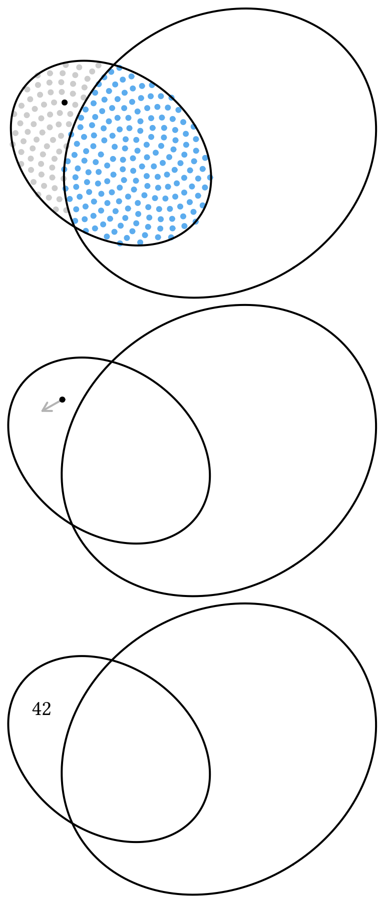


Figure A.1. The method eulerr uses to locate an optimal position for a label in three steps from top to bottom: first, we spread sample points on one of the ellipses and pick one inside the intersection of interest, then we begin moving it numerically, and finally place our label.

A.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 [7], which provides useful information without extraneous chart junk [39]. 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 [40]. 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** [41], 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 A.2.

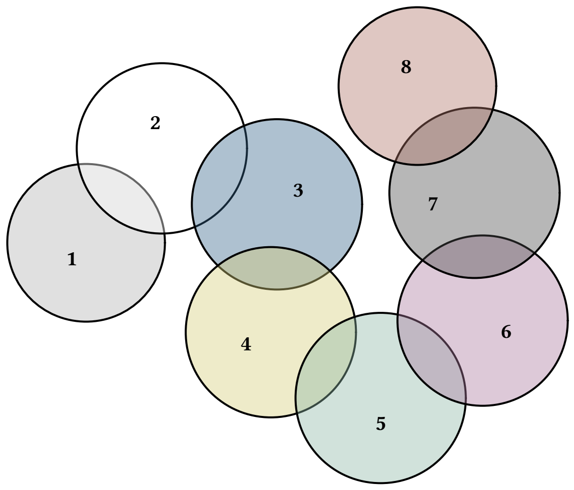


Figure A.2. The eight first colors of the default color palette.

A.3 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 [42] 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 also increases the chance of having similar layouts for different function calls even when we do not fix the random seed.

B Usage

`euler()` and `plot()` are the only functions that a user of **eulerr** need concern themselves with. In Input, we described the various forms of input that the function can be supplied with. Using the first form, we will showcases how a Euler diagram is fit. For this example, we use a diagram from a publication by Junta et al. [43] that was also showcased in Wilkinson [5]. We load the package, specify our diagram, and fit it using `euler()` as follows.

```
library(eulerr)
junta_2009 <- c("SE" = 13, "Treat" = 28, "Anti-CCP" = 101,
                "DAS28" = 91, "SE&Treat" = 1, "SE&DAS28" = 14,
                "Treat&Anti-CCP" = 6, "SE&Anti-CCP&DAS28" = 1)
fit1 <- euler(junta_2009)
```

Printing the results provides a summary of the fit, including the *stress* and *diagError* metrics that were introduced in Goodness of fit.

```
fit1 # or equivalently print(fit1)

##                         original fitted residuals regionError
## SE                            13 12.780     0.220       0.000
## Treat                         28 27.525     0.475       0.000
## Anti-CCP                     101 99.287     1.713       0.002
## DAS28                         91 89.457     1.543       0.001
## SE&Treat                       1  0.983     0.017       0.000
## SE&Anti-CCP                    0  0.000     0.000       0.000
## SE&DAS28                      14 13.763     0.237       0.000
## Treat&Anti-CCP                 6  5.898     0.102       0.000
## Treat&DAS28                    0  0.000     0.000       0.000
## Anti-CCP&DAS28                 0  0.000     0.000       0.000
## SE&Treat&Anti-CCP              0  0.000     0.000       0.000
## SE&Treat&DAS28                 0  0.000     0.000       0.000
## SE&Anti-CCP&DAS28              1  0.000     1.000       0.004
## Treat&Anti-CCP&DAS28           0  0.000     0.000       0.000
## SE&Treat&Anti-CCP&DAS28        0  0.000     0.000       0.000
##
## diagError: 0.004
## stress:    0
```

The fit is more or less equivalent to that of **venneuler** [5]. There *is* an error but it is small at a diagError of 0.004. We could, however, try to improve the fit using ellipses:

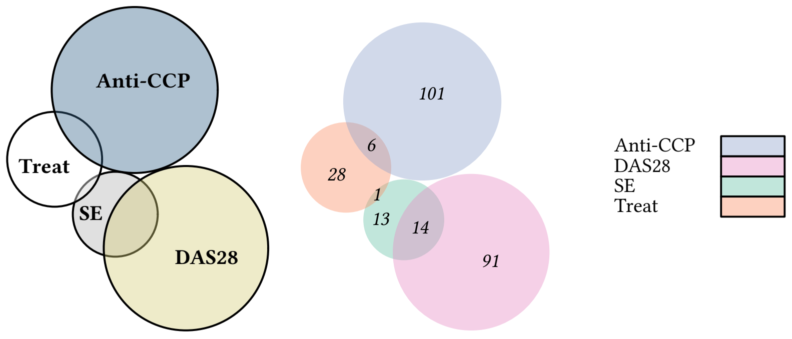
```r
# Fit the data using ellipses instead
fit2 <- euler(junta_2009, shape = "ellipse")

# Compare the fits on diagerror
fit1$diagError - fit2$diagError


## [1] 0
```

Comparing the two fits in diagError, however, shows that we have not bettered the fit in any meaningful way. Our next goal is to visualize the layout, which we do both using the default options and by customizing the fit, adding quantities, replacing the sets' labels with a key, removing lines, and changing the fills using the **RColorBrewer** package (Figure B.1).

```r
p1 <- plot(fit1)
p2 <- plot(fit1,
           quantities = list(fontface = 3),
           fill = RColorBrewer::brewer.pal(4, "Set2"),
           border = "transparent",
           auto.key = list(space = "right")) # key on the right
```



(a) The default settings. **(b)** Custom plot settings.

Figure B.1. The same fit visualized in two distinct ways.

Bibliography

[1] Leonhard Euler. *Letters of Euler to a German princess, on different subjects in physics and philosophy*. Murray and Highley, 1802. URL http://archive.org/details/letterseulertoa00eulegoog.

[2] Jonathan Swinton. *Vennerable: Venn and Euler area-proportional diagrams*, 2011. URL https://github.com/js229/Vennerable. R package version 3.1.0.9000.

[3] 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.

[4] Luana Micallef and Peter Rodgers. eulerForce: force-directed layout for Euler diagrams. *Journal of Visual Languages and Computing*, 25(6): 924–934, December 2014. ISSN 1045-926X. doi: 10.1016/j.jvlc.2014.09.002.

[5] 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.

[6] Hans A. Kestler, André Müller, Johann M. Kraus, Malte Buchholz, Thomas M. Gress, Hongfang Liu, David W. Kane, Barry R. Zeeberg, and John N. Weinstein. VennMaster: area-proportional Euler diagrams for functional GO analysis of microarrays. *BMC Bioinformatics*, 9:67, January 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-67.

[7] 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.

[8] Stirling Christopher Chow. *Generating and drawing area-proportional Euler and Venn diagrams*. Ph.D. dissertation, University of Victoria, Victoria, BC, Canada, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.2678&rep=rep1&type=pdf.

[9] Luana Micallef. *Visualizing set relations and cardinalities using Venn and Euler diagrams*. Ph.D. dissertation, University of Kent, September 2013. URL https://kar.kent.ac.uk/47958/.

[10] 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.

[11] Ben Frederickson. Calculating the intersection area of 3+ circles, November 2013. URL http://www.benfrederickson.com/calculating-the-intersection-of-3-or-more-circles/.

[12] Stirling Chow and Peter Rodgers. Constructing area-proportional Venn and Euler diagrams with three circles. In *Euler Diagrams Workshop 2005*, pages 182–196, August 2005. URL http://www.cs.kent.ac.uk/pubs/2005/2354.

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

[14] 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.

[15] 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/.

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] Jeroen Ooms. *V8: Embedded JavaScript Engine for R*, 2017. URL https://CRAN.R-project.org/package=V8. R package version 1.5.

[23] Oliver Lenz and Alessia Fornoni. Chronic kidney disease care delivered by US family medicine and internal medicine trainees: results from an online survey. *BMC medicine*, 4:30, December 2006. ISSN 1741-7015. doi: 10.1186/1741-7015-4-30.

[24] Ben Frederickson. Venn diagrams with D3.js, June 2015. URL http://www.benfrederickson.com/venn-diagrams-with-d3.js/.

[25] Simon Sheather and M. C. Jones. A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation. *Journal of the Royal Statistical Society*, 53(3):683–690, January 1991. doi: 10.2307/2345597. DOI: 10.2307/2345597.

[26] R Core Team. The Comprehensive R Archive Network, November 2017. URL https://CRAN.R-project.org/.

[27] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2017. URL https://CRAN.R-project.org/package=shiny. R package version 1.0.5.

[28] Ben Frederickson. venn.js compared to venneuler, June 2015. URL http://benfred.github.io/venn.js/tests/venneuler_comparison/.

[29] A. W. F. Edwards. *Cogwheels of the Mind: The Story of Venn Diagrams*. Johns Hopkins University Press, Baltimore, USA, 1 edition, April 2014. ISBN 978-0-8018-7434-5. URL https://jhupbooks.press.jhu.edu/content/cogwheels-mind.

[30] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. *The Journal of Open Source Software*, 1(2): 26, 2016. doi: 10.21105/joss.00026. URL http://joss.theoj.org/papers/10.21105/joss.00026.

[31] Dirk Eddelbuettel and Romain François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL http://www.jstatsoft.org/v40/i08/.

[32] Dirk Eddelbuettel and Conrad Sanderson. RcppArmadillo: accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. doi: 10.1016/j.csda.2013.02.005.

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

[34] 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.

[35] 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.

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

[37] 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.

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

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

[40] 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.

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

[42] 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.

[43] Cristina Moraes Junta, Paula Sandrin-Garcia, Ana Lúcia Fachin-Saltoratto, Stephano Spanó Mello, Renê D. R. Oliveira, Diane Meyre Rassi, Silvana Giuliatti, Elza Tiemi Sakamoto-Hojo, Paulo Louzada-Junior, Eduardo Antonio Donadi, and Geraldo A. S. Passos. Differential gene expression of peripheral blood mononuclear cells from rheumatoid arthritis patients may discriminate immunogenetic, pathogenic and treatment features. *Immunology*, 127(3):365–372, July 2009. ISSN 1365-2567. doi: 10.1111/j.1365-2567.2008.03005.x.