LUND
UNIVERSITY

BACHELOR THESIS

eulerr: Area-Proportional Euler Diagrams with Ellipses

Johan Larsson

supervised by
Peter Gustafsson

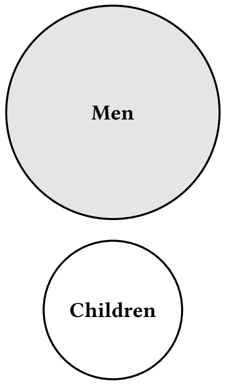
November 1, 2017

Contents

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 men, as well as their intersection. The diagram we have constructed is a *Euler diagram* (Figure 1.1).



```
## [1] 2.24e-09
```

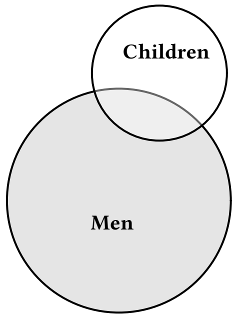


Figure 1.1. The merits of a Euler diagram.

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:

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

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.

```
## [1] 0.0579
## [1] 8.09e-08
```

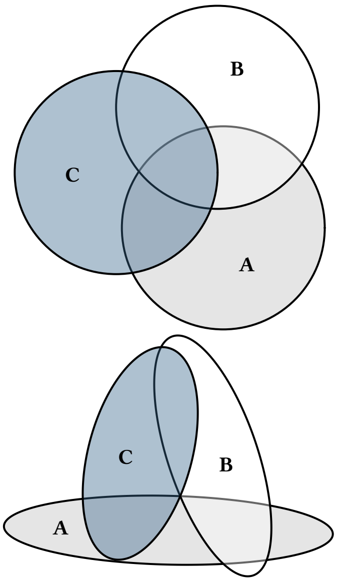


Figure 1.2. A set relationship depicted erroneously with circles and perfectly with ellipses.

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**.

```
## [1] 7.56e-07
```

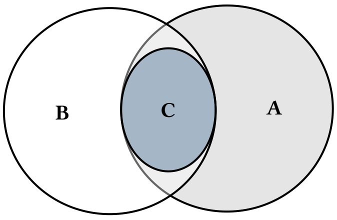


Figure 1.3. A Euler diagram with a subset relationship.

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 estimate 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 minimze the error in the three-way intersection. **venneuler** [5] uses multi-dimensional scaling with Jacobian distances, taking only pairwise relationships into account. **venn.js** Frederickson [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** Swinton [2] uses a simple method of computing the required pairwise distances between circles and the changes the largest to attempt to arrive at the correct two-way overlaps. All the algorithms use circles in the initial configuration.

Diagrams with more than two sets normally require additional tuning, which may be 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 [9], 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 first find all the intersection points between the ellipses, for which there are several approaches. Some rely on solving the system of equations formed by two ellipses to be intersected, which necessitates solving a fourth-degree polynomial; other methods make use of the representation of an ellipse as a conic in projective geometry, which involves solving a third-degree polynomial. These methods vary in execution time but are all accurate up to floating-point precision.

With all the intersection points at hand, the areas of the overlaps can be established. Frederickson [11] has published a method for circles and a similar method is used by Micallef [9] for three ellipses. No method has so far been published that generalizes these methods to diagrams of more than three ellipses or ellipses with subset or disjoint relationships.

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.

2 Method

Constructing a Euler diagram is analagous to fitting a statistical model in that you need

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

In the following sections, we explain how **eulerr** adressess each of these items in turn.

2.1 Input

The data for a Euler diagram is always a description of set relationships. **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 a Euler diagram to each split. This is offered as a convenience function for the user since it may be that the solution is more well-behaved 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.

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

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

[5]

Survived?	No	Yes
Child	52.00	57.00
Adult	1438.00	654.00

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

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

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 a useable form, 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),

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

which we optimize using R's built-in `optimize()`[6]. Convergence is fast and neglible next to our later optimization procedures.

Given these optimal pairwise distances, we proceed to the next step, where we position the circles representing the sets. This can be accomplished in many ways; **eulerr**

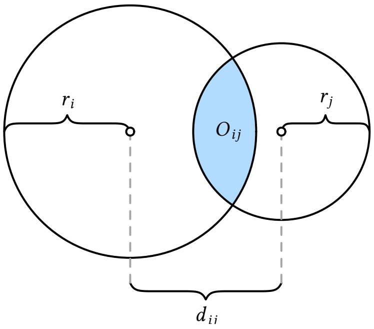


Figure 2.1. The circle–circle overlap is computed as a function of the discs' separation (d_{ij}), radii (r_i, r_j), and area of overlap (O_{ij}).

The algorithm assigns a loss and gradient of zero when the sets and their representations as circles are disjoint or when the sets and circles are subset. In all other cases, the loss function (2.2) 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. The gradient (2.3) is retrieved as usual by taking the derivative of the loss function.

$$\mathcal{L}(\boldsymbol{v}) = \sum_{i=1}^{N-1}\sum_{i=j+1}^{N}\begin{cases} 0 & (F_i \cap F_j = \emptyset) \land (O_{ij} = \emptyset) \\ 0 & \big((F_i \subseteq F_j) \lor (F_i \supseteq F_j)\big) \land (O_{ij} = \emptyset) \\ 4((\boldsymbol{v}_i - \boldsymbol{v}_j)^T(\boldsymbol{v}_i - \boldsymbol{v}_j) - d_{ij}^2)^2 & \text{otherwise} \end{cases} \quad (2.2)$$

$$\vec{\nabla} f(\boldsymbol{v}_i) = \sum_{i=1}^{N}\begin{cases} \vec{0} & (F_i \cap F_j = \emptyset) \land (O_{ij} = \emptyset) \\ \vec{0} & \big((F_i \subseteq F_j) \lor (F_i \supseteq F_j)\big) \land (O_{ij} = \emptyset) \\ 4\left((\boldsymbol{v}_i - \boldsymbol{v}_j)^T(\boldsymbol{v}_i - \boldsymbol{v}_j) - d_{ij}^2\right)(\boldsymbol{v}_i - \boldsymbol{v}_j) & \text{otherwise,} \end{cases} \quad (2.3)$$

where $\boldsymbol{v}_i = \begin{bmatrix} h_i \\ k_i \end{bmatrix}$ and $\land, \lor$ denote the conditional `AND` and `OR` operators respectively.

We optimize (2.2) using the nonlinear optimizer `nlm()` from the R core package **stats**, which is a translation from FORTRAN code developed by Schnabel et al. [12] that uses a mix of Newton and Quasi-Newton algorithms. It makes use of the Hessian, which we presently compute numerically[7].

This initial configuration will be accurate for two-set combinations and optimal—although not necessarily accurate—for three-set combinations that use circles. But for all other combinations there is usually a need to fine-tune the configuration.

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

[7] At the time of writing, there was a bug in the current version of the package, causing the analytic Hessian to be updated incorrectly.

2.3 Final configuration

So far, we have only considered pairwise relationships. To test and improve our layout, however, we need to account for all the relationships and, consequently, all the intersections and overlaps in the diagram. Initially, we restricted ourselves to circles but now extend ourselves also to 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 [13] 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 ellipses are a special case of conics—of which the circle, ellipse, parabola, or hyperbola are members—they can also be represented as such, using the quadric form,

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

Now, if we furthermore convert this, the quadric form, into its matrix equivalent,

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

we have arrived at the representation of an ellipse in projective geometry required to find intersections between ellipses. Following this, we then

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).

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. These points form a geometric shape that can be considered a polygon with elliptical segments formed by successive points of the polygon (Section 2.3).

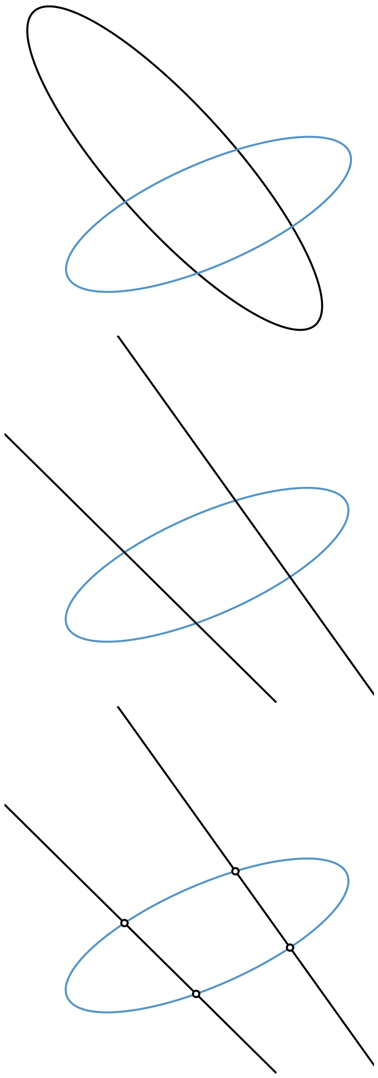


Figure 2.2. The process (from top to bottom) used to intersect two ellipses, here yielding four points.

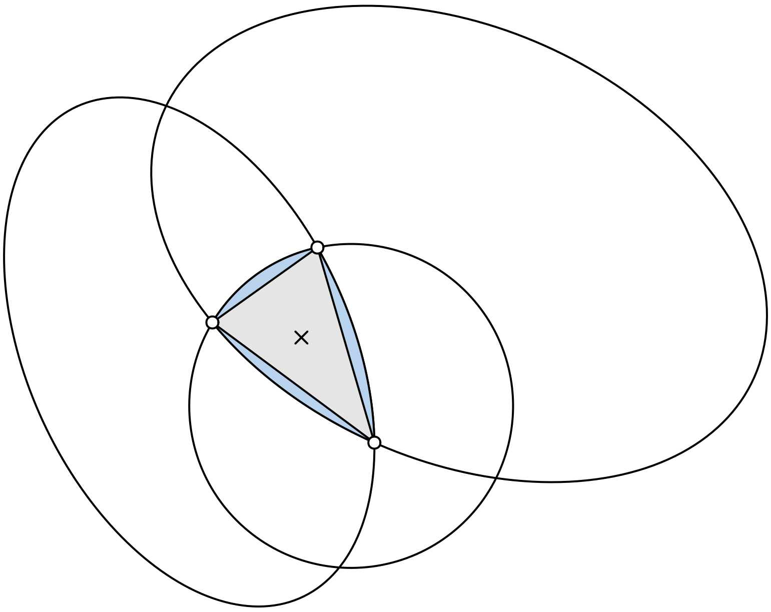


Figure 2.3. The overlap area between three ellipses is the sum of a convex polygon (in grey) and 2–3 ellipse segments (in blue).

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[8]. Then, we acknowledge that each elliptical segment is formed from an arc of the ellipse that is shared by both points[9]. Now that we have two points on an ellipse, we can find the area of the ellipse segment using an algorithm from Eberly [14]. To proceed, we

1. center their ellipse at $(0, 0)$,
2. normalize its rotation, which is not needed to compute the area,
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 (10).

$$\alpha(\theta_0, \theta_1) = F(\theta_1) - F(\theta_0) - \frac{1}{2}\left|x_1 y_0 - x_0 y_1\right|,$$
$$\text{where } F(\theta) = \frac{a}{b}\left[\theta - \arctan\left(\frac{(b-a)\sin 2\theta}{b + a + (b-a)\cos 2\theta}\right)\right]$$

This procedure is illustrated in Figure 2.4.

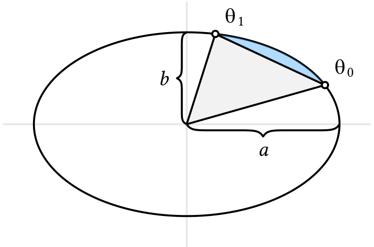


Figure 2.4. The elliptical segment in blue is found by first subtracting the elliptical sector from $(a, 0)$ to θ_0 from the one from $(a, 0)$ to θ_1 and then subtracting the triangle part (in grey).

In a few instances[10], the exact algorithm will break down. This occurs from numerical approximation errors when some 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 an overview),

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

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

[10] 1 out of approximately 7000 in our simulations

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 up to numerical precision. We feed the initial layout computed in Initial configuration to the optimizer, this time allowing the ellipses to rotate and the relation between the semiaxes vary, altogether rendering five parameters to optimize per set and ellipse. 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 sum of squared errors between the ideal sizes (ω from Definition 2.1) and the respective areas of the diagram,

$$\sum_{i=1}^{n} (A_i - \omega_i)^2 \tag{2.4}$$

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] and *diagError* [3].

The stress metric is the residual sums of squares over the total sums of squares,

$$\text{Stress} = \frac{\sum_{i=1}^{n}(\omega_i - A_i)^2}{\sum_{i=1}^{n}(A_i - \bar{A})}, \tag{2.5}$$

where $\bar{A}$ is the arithmetic mean of the areas in the diagram.

The stress metric does not lend itself readily to a clear-cut interpretation but can be transformed into a rough analogue of the correlation coefficient by $r = \sqrt{1 - \text{Stress}^2}$.

diagError, meanwhile, is given by

$$\text{diagError} = \max_{i=1,2,\ldots,n} \left| \frac{\omega_i}{\sum_{i=1}^{n} \omega_i} - \frac{A_i}{\sum_{i=1}^{n} A_i} \right|, \tag{2.6}$$

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. 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** 2.0.0.9000,
- **eulerAPE** 3.0.0,
- **venn.js** 0.2.14,
- **venneuler** 1.1-0, and
- **Vennerable** 3.1.0.9000.

The results in Performance and all results from **eulerAPE** were computed on a computer[11] running R version 3.4.2. The remaining results were computed on an Amazon EC2 cloud-based computing instance.

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**.

[11] 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

```
## [1] 0.011
## [1] 2.08e-08
```

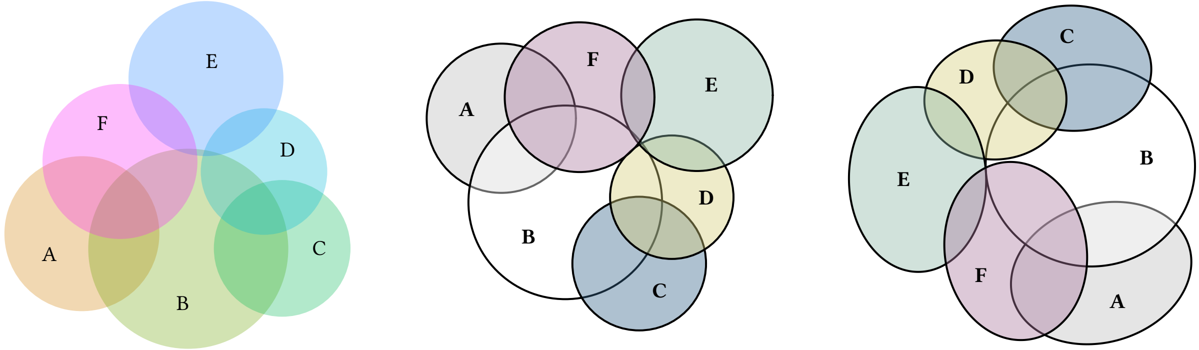


Figure 3.1. A comparison of a Euler diagram generated with **venneuler** with two generated from **eulerr** with circles and ellipses respectively. The stress of the solutions are 0.006, 0.004, and 0.000 respectively.

3.2 Consistency

To compare the consistency among **eulerr**, **venneuler**, **eulerAPE**, **venn.js**, and **Vennerable**, we generate random diagrams of circles and ellipses, compute their areas, and attempt to reproduce the original diagram using the software packages that we are studying. We restrict ourselves to diagrams consisting of between 3 and 8 shapes.

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

$$\begin{aligned} r_i &\sim \mathcal{U}(0.3, 0.6) \\ h_i, k_i &\sim \mathcal{U}(0, 1). \end{aligned} \tag{3.1}$$

For the ellipses, we sample semiaxes (a_i and b_i), coordinates (h_i and k_i), and rotation axes (ϕ_i) from

$$\begin{aligned} h_i, k_i &\sim \mathcal{U}(0, 1) \\ a_i, b_i &\sim \mathcal{U}(0.2, 0.8) \\ \phi_i &\sim \mathcal{U}(0, 2\pi). \end{aligned} \tag{3.2}$$

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* (2.6) 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 p, the proportion of successful diagrams, that is no wider than 2%. The confidence that we generate is the standard asymptotic interval,

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

```
for $i \leftarrow 3$ to $8$ do
    while length of each $\mathcal{I}(p_s)_{0.95} < 2\%$ and $j < 500$ do
        if circle then
            $h, k \leftarrow \mathcal{U}(0, 1)$
            $r \leftarrow \mathcal{U}(0.3, 0.6)$
            $\omega \leftarrow$ findOverlaps$(h, k, r)$
        else if ellipse then
            $h, k \leftarrow \mathcal{U}(0, 1)$
            $a, b \leftarrow \mathcal{U}(0.2, 0.8)$
            $\phi \leftarrow \mathcal{U}(0, 2\pi)$
            $\omega \leftarrow$ findOverlaps$(h, k, a, b, \phi)$
        $A \leftarrow$ 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 $diagError_j < 0.01$ then successes + 1
        foreach $s \leftarrow$ software package do
            $\hat{p}_s \leftarrow \text{successes}/j$
            $\mathcal{I}(\hat{p}_s)_{0.95} \leftarrow \hat{p} \pm z_{0.95}\sqrt{\frac{\hat{p}_s(\hat{p}_s - 1)}{n}}$
```

Algorithm 3.1. The algorithm used to simulate circles and ellipses, compute their areas, and fit Euler diagrams to these layouts using the different software packages.

The algorithm is formalized in Algorithm 3.1. The left panel of Figure 3.2 shows the results of our simulation.

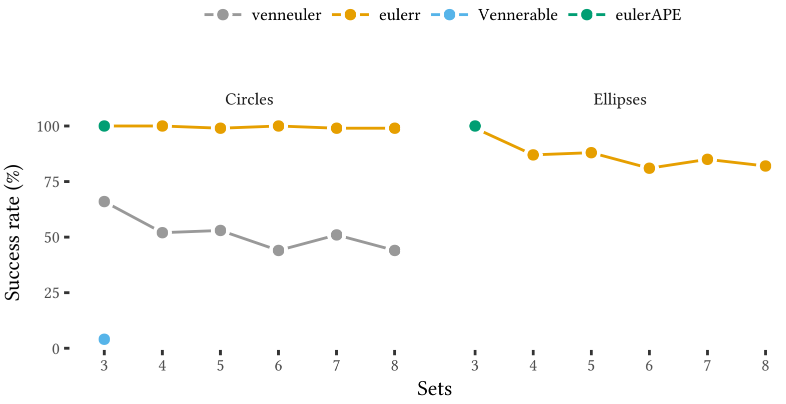


Figure 3.2. Reproducibility tests for ellipses and circles generated from the distributions from (3.1). Note that **Vennerable** and **eulerAPE** only support Euler diagrams for three sets, which is why its data is absent for the other cases.

eulerr outperforms both **Vennerable** and **venneuler** in consistency (Figure 3.2). It is able to reproduce Euler diagrams for almost all of the circles and for all three-set ellipses. For ellipses of four or more sets, the consistency drops considerably, yet remains above ∞%. **Vennerable**, which is only able to produce three-set diagrams, only produces accurate diagrams for *NaN*% of the random layouts and moreover fails with an error in 0 cases.

3.3 Accuracy

In Consistency, we assess the efficacy in reproducing diagrams with exact, but unknown, solutions. In real situations, however, we are often faced with set configurations that lack exact solutions, in which case we want our method to produce an approximation that is as accurate as possible.

```
for $i \leftarrow 3$ to $8$ 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 $j \leftarrow 3$ to $i$ do
            $j \leftarrow$ random index in $\{\omega : \omega \cap F_i \neq \emptyset\}$
            $\omega_j \leftarrow \mathcal{U}(0, 1)$
        $\omega_S \leftarrow \mathcal{U}\{0, i\}$ random elements from $\{\omega : \omega = 0\}$
        $\omega_S \leftarrow \mathcal{U}(0, 1)$
        fit a Euler diagram to $\omega$
```

Algorithm 3.2. The algorithm we use to simulate random set relationships and fit them with the software under study to assess their accuracy.

To assess this, 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 have achieved a desired confidence interval for the estimates. This time, we keep on going until we have a 95% confidence interval no longer than 0.02 in mean diagError. We use the common confidence level 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 3.2

The error in the diagrams generated with **eulerr** is considerably lower than for the other methods. This is most apparent with ellipses but also true for circles compared to **venneuler** and **Vennerable**.

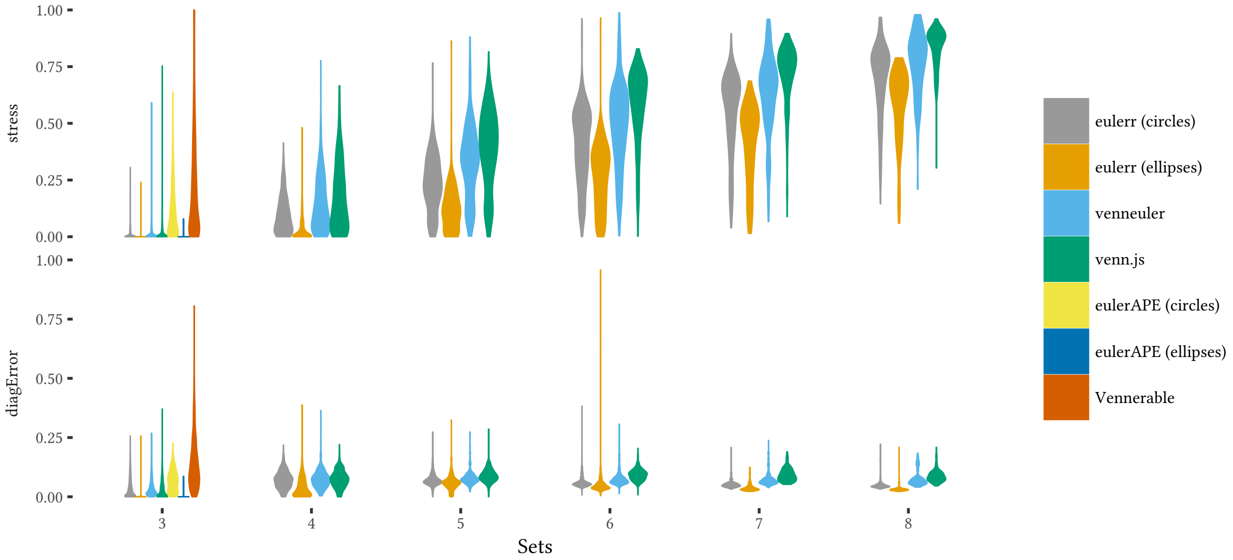


Figure 3.3. Accuracy tests of set relationships that may or may not have perfect solutions, generated from (3.2).

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** to compute this, subtracting function call times to produce strict measurement. In addition, we randomize the order in which the packages are called between trials.

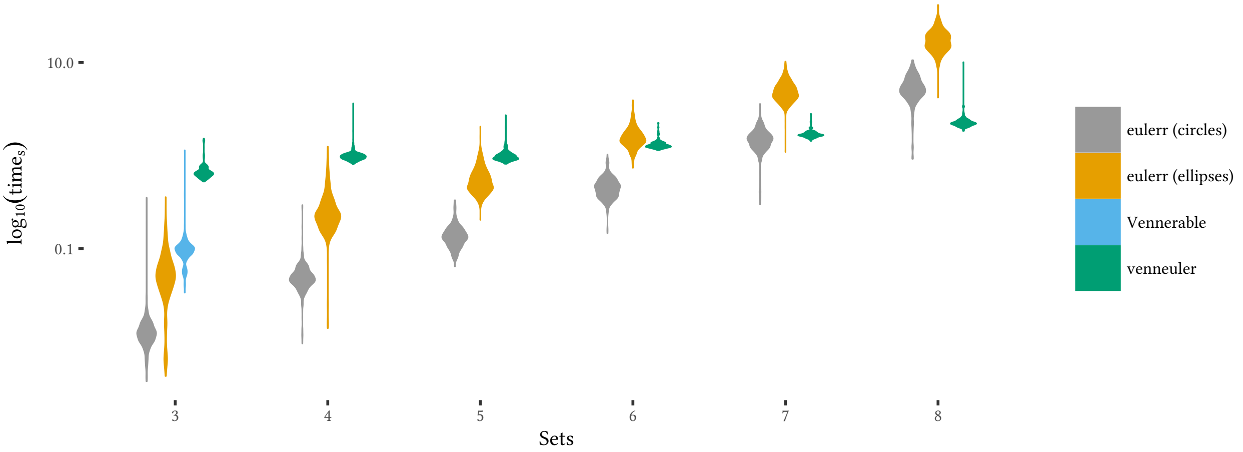


Figure 3.4. Performance of **eulerr**, **venneuler**, and **Vennerable** on random set relationships of 3 to 8 sets

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 method is superior in both accuracy and consistency next to all other software packages for R—even when the method is restricted to circles. In addition, the method is speedier than the competition for set relationships with up to six sets, wherafter it performs worse than **venneuler**.

In terms of consistency and accuracy, there are several reasons for why **eulerr** improves upon the method in **venneuler** and **Vennerable**. The primary reason lies in the use of ellipses rather than circles. Ellipses feature two additional degrees of freedom and are therefore able to accurately represent a larger variety of relationships.

Moreover, the initial optimizer in **eulerr** circumvents a rule in **venneuler** that puts unnecessary restrictions on layouts that include disjoint or subset relationships. The initial optimizer in venneuler places disjoint and subset circles exactly neck-in-neck and at the exact midpoint of the set respectively. Yet, because we are indifferent about where in the space outside (or respectively inside) the circles are placed, that behavior becomes problematic since it might interfere with locations of other sets that need to use that space. The MDS algorithm from **venn.js** circumvents this by assigning a loss and gradient of zero when the pairwise set intersection *and* the candidate circles are disjoint or subset.

For 3 to 7 sets, **eulerr** is speedier than both **Vennerable** and **venneuler**. The reasons for this is partly to do with the implementation in C++ using high-performance interfaces such as **Rcpp** [15] and **RcppArmadillo** [16]. For few sets, the exact-area calculations that **eulerr** features also promote better performance; yet, paradoxically, this also happens to be the reason for why the performance of **eulerr** suffers as the number of sets surge. The bottleneck is the final optimizer. It has to examine every possible intersection when computing the areas, thus converging in $O(2^n)$ time. Wilkinson [5], meanwhile, report convergence in $O(n)$ time. This is clearly evidenced in Figure 3.4. Future versions of this algorithm might consider implementing approximate area-calculations when the number of sets is large.

This method was first published for circles in a blog post [11] and in a scholarly paper for up to three ellipses [9] but has to our knowledge not previously been generalized to any number of ellipses.

The authors motivate this limitation by the propensity of Euler diagrams with more sets to lack adequate solutions and that their complexity make implementations difficult [9].

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 [17] 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* [18, 19] 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 tocomputig the distance from a point to an ellipses. The numerical solution we use has been described in [ref] and involves a bisection optimizer.

To optimize this, we employ a version of the *Nelder–Mead Method* [20] which has been translated from Kelley [21] and adapted for **eulerr** to ensure that quick convergence and that the simplex remains within the intersection boundaries (since we want the local maximum). The method is visualized in Figure A.1.

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 [22]. The issue with color, however, is that it cannot be perceived perfectly by roughly 10% of all people. Moreover, color is often printed at a premium in scientific publications and adds nothing 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 slightly to fullfil our other objectives of avoiding using colors for two sets.

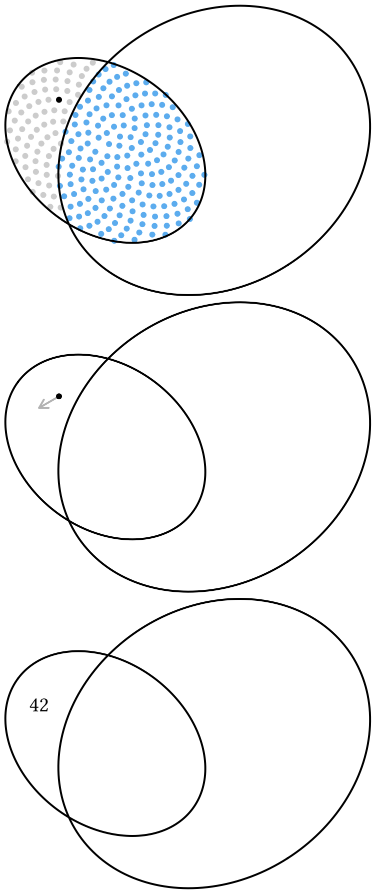


Figure A.1. The method eulerr uses to locate an optimal position for a label in three steps from top to bottom: first, we spread sample points on one of the ellipses and pick one inside the intersection of interest, then we begin moving it numerically, and finally place our label.

```
## [1] 8.43e-08
```

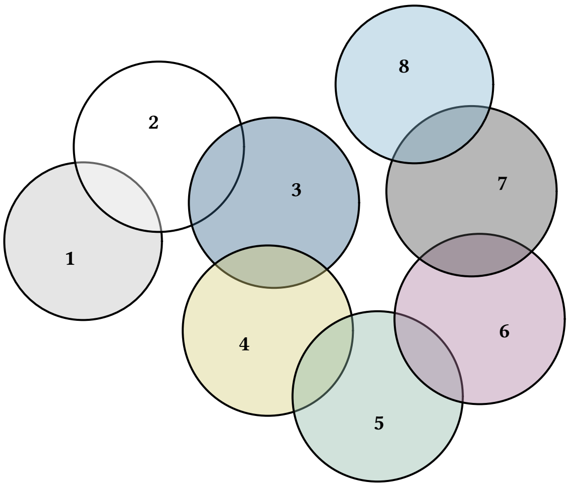


Figure A.2. The eight first colors of the default color palette.

A.3 Normalizing dispered layouts

A side effect of running an unconstrained optimizer (See Final configuration) is that we almost invariably produce overdispersed layouts if there are disjoint clusters of ellipses. To solve this, we use a SKYLINE-BL rectangle packing algorithm [24] which is designed specifically for **eulerr**. In it, we surround each ellipse cluster with a bounding box and pack these boxes into a bin with an appropriate size and aspect ratio of the golden rule.

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 may input a diagram from Junta et al. [25] that was showcased in Wilkinson [5] 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)


## [1] 0.00392
```

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

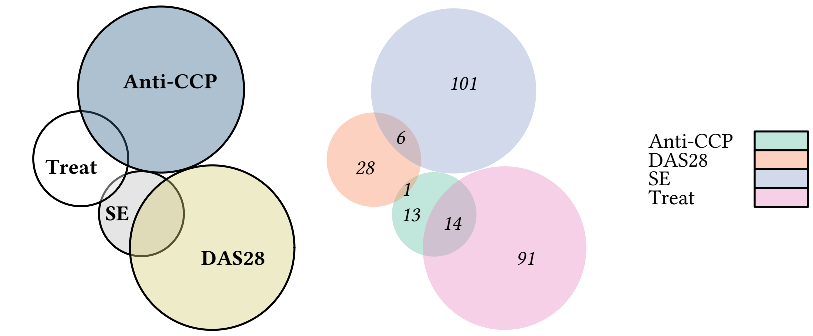
```
fit1 # or equivalently print(fit1)


##                         original fitted residuals regionError
## SE                            13     13         0       0.000
## Treat                         28     28         0       0.000
## Anti-CCP                     101    101         0       0.002
## DAS28                         91     91         0       0.001
## SE&Treat                       1      1         0       0.000
## SE&Anti-CCP                    0      0         0       0.000
## SE&DAS28                      14     14         0       0.000
## Treat&Anti-CCP                 6      6         0       0.000
## Treat&DAS28                    0      0         0       0.000
## Anti-CCP&DAS28                 0      0         0       0.000
## SE&Treat&Anti-CCP              0      0         0       0.000
## SE&Treat&DAS28                 0      0         0       0.000
## SE&Anti-CCP&DAS28              1      0         1       0.004
## Treat&Anti-CCP&DAS28           0      0         0       0.000
## SE&Treat&Anti-CCP&DAS28        0      0         0       0.000
##
## diagError: 0.004
## stress:    0
```

The fit is more or less equivalent to that of **venneuler** Wilkinson [5]. We could try to fit the diagram using ellipses instead.

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


## [1] 0.00392


# 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 counts, replacing the sets' labels with a key, removing lines, and changing the fills using the **RColorBrewer** package (Figure B.1).

```
p1 <- plot(fit1)
p2 <- plot(fit1,
           counts = 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*. URL https://github.com/js229/Vennerable.

[3] Luana Micallef and Peter Rodgers. eulerAPE: drawing area-proportional 3-Venn diagrams using ellipses. *PLOS ONE*, 9(7):e101717, 2014-jul-17. 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. URL http://dx.doi.org/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. URL https://doi.org/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] 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. URL http://doi.acm.org/10.1145/6187.6192.

[13] 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.

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

[15] 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/.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] Johan Larsson. qualpalr: automatic generation of qualitative color palettes, 2016. URL https://cran.r-project.org/package=qualpalr.

[24] Jukka Jyl?nki. 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.

[25] 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.