LUND
UNIVERSITY

Bachelor Thesis

eulerr: Area-Proportional Euler Diagrams with Ellipses

Johan Larsson

supervised by
Peter Gustafsson

September 20, 2017

Background

The visual display of quantitative information is often most intuitive presentation of data. If done right, visualizations possess the potential to convey intricate relationships that a single statistic or table of numbers never could. This is because data visualizations map abstract quantities to concrete aesthetics, such as shape or color, which lets the observer take a multi-dimensional approach to interpret the presentation.

Such aesthetics, however, are only informative in relation to other shapes. For instance, a disc with a radius of 1 cm and label of *diabetes* says nothing by itself—yet juxtapose it with 2 cm radius disc and label it *overweight* and the graphic starts to become informative. If we then intersect the two discs, thereby generating a visible overlap, we have managed to explain both the relative sizes of two separate categories and their intersection. The diagram we have constructed is a *Euler diagram*, which, incidentally, serves as the topic of this paper.

The Euler diagram, originally proposed by Leonard Euler [1], is the superset of the obiquiteous Venn diagram: a staple introductory text books in statistics and scientific disciplines such as bioinformatics and geology. The difference between Venn and Euler diagrams is that the former require that all $2^n - 1$ possible intersections are present—even if they are empty—whilst Euler diagrams do not.

Euler diagrams may be area-proportional, which is to say that each separate surface of the diagram is proportional to the quantity it intends to measure. (This was the case with the diagram with defined in the second paragraph.) This is the most natural and rational form of a Euler diagram being that only its shapes are necessary to interpreting it, letting us, for instance, discard numbers without crucial loss of information—the same could not be said for a Venn diagram.

Euler diagrams may be fashioned out of any conceivable closed shape. Solutions have been developed for triangles [2], rectangles [2], ellipses [3], smooth curves, polygons [2], and circles [2, 4, 5]. The latter is most common, and appears to be preferred for optimal readability. Yet circles do not always lend themselves to accurate representations. Consider, for instance the combination

$$A = B = C = 2,$$
$$A \cap B = A \cap C = B \cap C = 1$$
$$A \cap B \cap C = 0.$$

There is no way to visualize this relationship perfectly with circles, yet with ellipses there *is* (Figure 1).

With four intersecting sets, circular Euler diagrams are actually impossible, given that $2^4 = 16$ regions are required but the circles can yield at most 14 unique intersections. This is not, however, the case with ellipses in that they they may intersect in up to 4, rather than 2, points. All in all, circles have three degrees of freedom: x-coordinate, y-coordinate, and radius, whereas ellipses have five: x-coordinate, y-coordinate, semi-minor radius, semi-major radius, and rotation.

Elliptical Euler diagrams have been successfully implemented in the **eulerAPE** software [3], yet only for three sets that moreover need to intersect,

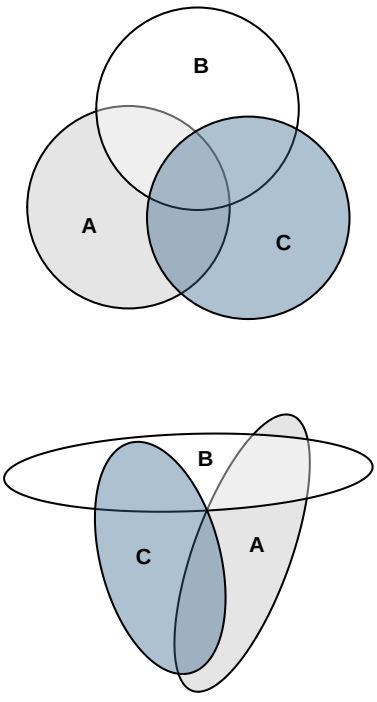

Figure 1: A set relationship depicted with circles and ellipses, showing that ellipses sometimes prevail where circles fail.

the motivation for which being that Euler diagrams with more than three sets often lack adequate solutions and that their high complexity make implementations difficult [6].

As far as we know, analytical solutions to Euler diagrams do not exist. All existing implementations are instead based on varying numerical methods. Most of these use separate methods for the initial and final diagram configurations. For instance, Micallef [6] uses uses a greedy algorithm, Wilkinson [4] uses multi-dimensional scaling with jacobian distances—a constrained version of which is used in Frederickson [7] in combination with a greedy algorithm. The latter two work only on pairwise relationships between the sets while the first tries to optimize the three-way intersection between the three sets. All use circles for the initial layout.

Diagrams with more than two sets require additional tuning, which is considered in a final configuration step. This is always an optimization procedure with a target loss function and an optimization procedure.

It is this difficulty that we have overcome with **eulerr**, which is the first software to generalize elliptical Euler diagrams to any number of sets.

Aims

The aim of this thesis is to present a method and implementation for constructing and visualizing Euler diagrams for sets of any numbers with ellipses.

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. test to assess the model fit, and
4. a presentation of the result.

In the following sections, we will explain how **eulerr** deals with each of these steps in order to develop an accurate diagram.

Input

The data for a Euler diagram is always a description of set relationships. **eulerr** allows several varities of this data—namely,

- Intersection and relative complements[1]
- Unions and identities[2]
- A matrix of logicals or binaries[3]
- A list of sample spaces[4]
- A two- or three-way table [5]

[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

Whichever type is used, **eulerr** translates it to the first type, *intersections and relative complements* (Definition 1), which is the form used later in the loss functions of the initial and final optimizers.

Definition 1. *For a family of* n *sets,* $F = A, B, \ldots, N$*, we define* ω *as the intersections and their relative complements of these sets, such that*

$$\begin{aligned}
\omega_{ij\cdots n} &= \bigcap_{i=1}^{n} F_i \\
\omega_{ij\cdots n-1} &= \bigcap_{i=1}^{n-1} F_i \setminus \bigcap_{i=1}^{n} F_i \\
&\vdots \\
\omega_i &= A \setminus \bigcup_{i=2}^{n-1} F_i
\end{aligned}$$

with

$$\omega_S = \sum_{i=1}^{n} \omega_i + \omega_j + \cdots + \omega_{ij} + \cdots + \omega_{ij\cdots n} = \bigcup_{i=1}^{n} F_i$$

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 the case that the euler diagram solutions are relatively more well-behaved after such a split.

Fittin the Euler diagram

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 and optionally using ellipses instead of circles.

Initial configuration

For our initial configuration, we work exclusively with circles. We begin by mapping ω to areas and, given these, figure out the required pairwise distance between the sets to achieve a circle–circle overlap matching ω (Definition 1). We accomplish this numerically, using the formula for a circle–circle overlap (1),

$$A_{ij} = r_i^2 \arccos\left(\frac{d^2 + r_i^2 - r_j^2}{2dr_i}\right) + r_j^2 \arccos\left(\frac{d^2 + r_j^2 - r_i^2}{2dr_j}\right) - \frac{1}{2}\sqrt{(-d + r_i + r_j)(d + r_i - r_j)(d - r_i + r_j)(d + r_i + r_j)}, \quad (1)$$

where r_i and r_j are the radii of the circles representing the i:th and j:th sets respectively and d the distance between them.

Although r_i and r_j are known, d is not, wherefore we approximate it numerically. Here, our loss function is the squared difference between A and the desired overlap, $(A \cap B - \omega)^2$, which we then optimize using **R**'s built-in `optimize()`[6]. Convergence is fast—neglible next to our later optimization

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

procedures.

Given these optimal pairwise distances, we can proceed to the next step, which is to position the circles representing the sets. This can be accomplished in many ways; **eulerr** uses a method from the **venn.js** script [7], namely a constrained version of multi-dimensional scaling (MDS), which in turn is an upgrade to a method from the **venneuler** package [4]. **venneuler** tries to place disjoint and subset 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 sets 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. In other cases, the loss function is the normal sums of squares between the areas from (1) and ω (Definition 1) uses the loss (2) and gradient function (3).

$$\mathcal{L}(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \begin{cases} 0 & I \cap J = \emptyset \wedge A_{ij} = \emptyset \\ 0 & (I \subseteq J) \vee (I \supseteq J) \wedge A_{ij} = \emptyset \\ ((X_i - X_j)^T (X_i - X_j) - D_{ij}^2)^2 & \text{otherwise} \end{cases} \tag{2}$$

$$\vec{\nabla} f(X_i) = \sum_{j=1}^{n} \begin{cases} \vec{0} & I \cap J = \emptyset \wedge A_{ij} = \emptyset \\ \vec{0} & (I \subseteq J) \vee (I \supseteq J) \wedge A_{ij} = \emptyset \\ 4\big((X_i - X_j)^T (X_i - X_j) - D_{ij}^2\big)(X_i - X_j) & \text{otherwise} \end{cases} \tag{3}$$

where $X_i = [x_i, y_i]^T$.

The optimizer transverses over $\binom{n}{2}$ combinations, making the complexity of the task (what?)

Frederickson [7] uses the *Polak–Ribière Conjugate Gradient Method* to optimize the initial layout. The method, however, occasionally encounters local minima, which is why they run several restarts of the initial configuration (in addition to forming a layout using a greedy method). In our case, however, we have opted to instead use the nonlinear optimizer `nlm()` from the **R** core package **stats**, which is a translation from FORTRAN code developed by Schnabel et al. [8] that uses a mix of Newton and Quasi-Newton algorithms. It also uses the Hessian, which we presently compute numerically[7].

This initial configuration will work perfectly two-set combinations and optimally for three-set combinations (if we use circles). But for all other combinations there is almost always a need to fine tune the configuration.

[7] A bug in the current version of the package leads the analytic Hessian to be updated incorrectly.

Final configuration

So far, we have only worked with pairwise relationships and areas. To test and improve our layout, however, we need to account for all relationships and consequently all the various intersections between the shapes we use in the diagram. Initially, we restriced ourselves to circles but we now extend our reach also to ellipses.

Computing the overlaps of ellipses is not trivial, which is evident in that most methods resort to approximations such as quad-tree binning [4] or

polygon intersections [5]. In addition to being imperfect [6], and as we shall see in **??** , sometimes perform poorly.

Other methods [6, 7], however, use algorithms that compute the overlap areas exactly. But unlike most approximative types, all the exact methods require that we first find the points of intersection between the ellipses. **eulerr**'s approach to this is that of a method outlined by Richter-Gebert [9], which is based in projective, rather than euclidean, geometry.

We only need to consider two ellipses at a time. The canonical form of these is given by

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

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) is the center of the ellipse. For our following algorithms to work, we wneed to convert each ellipse to its matrix form,

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

which represents a generalization of the ellipse that can be used to define any conic (circle, ellipse, parabola, or hyperbola). Following this, we then

1. form a degenerate conic from the solution to the system of these two conics,
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 (Figure 2).

After we have discovered all the intersection points, we find the area of a overlap by examining the subset of intersection points that were formed from the ellipses that we are currently exploring that are simultaneously contained within all of these ellipses. These intersection points will form a convex polygon plus elliptical segments (Figure 3).

Since the polygon section is always convex, it is easy to find its area using the the *triangle method*. To find the areas of the elliptical segments, we use an algorithm presented in Eberly [10] in which we begin centering the ellipse on [0, 0] and normalizing its rotation, which is not needed to compute the area. We then integrate the ellipse between the two points making up the start and end of the elliptical segment. This gives us the *elliptical sector* from which we subtract the triangle part to find the segment area ().

$$A(\theta_0,\theta_1)=F(\theta_1)-F(\theta_1)-\frac{1}{2}|x_1y_0-x_0y_1|,$$

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

An illustration is given in Figure 4.

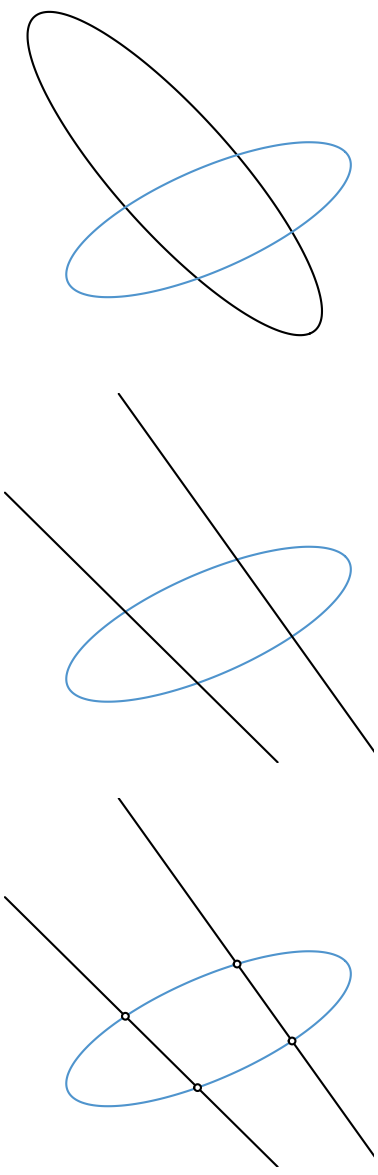

Figure 2: The process (from top to bottom) used to intersect two ellipses, here yielding 4 points.

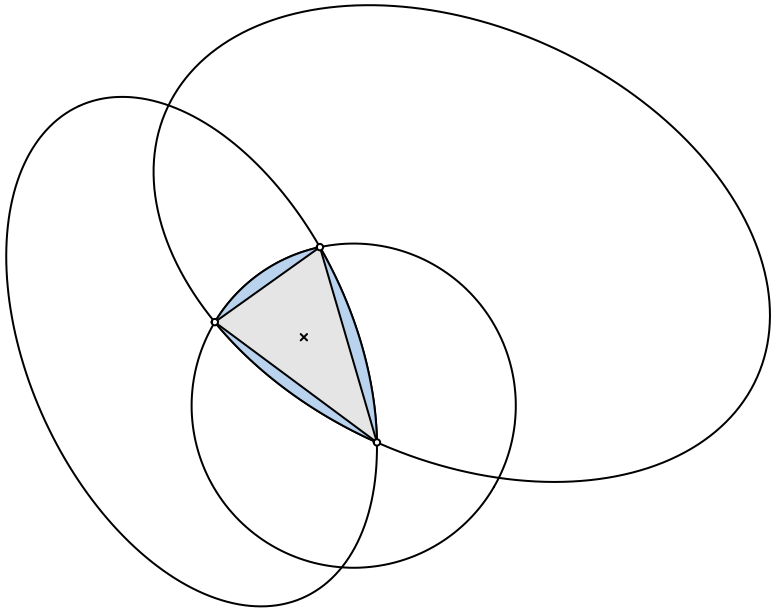

Figure 3: The overlap area between three ellipses is the sum of a convex polygon (in grey) and 2–3 ellipse segments (in blue).

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

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 **??** to the optimizer, this time allowing the ellipses 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 1) and the respective areas of the diagram,

$$\sum_{\omega}(A_i - \omega_i)^2 \tag{4}$$

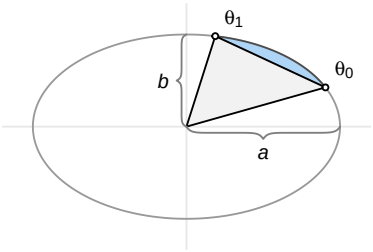

Figure 4: The elliptical segment in blue is found by first subtracting the elliptical sector from $[a, 0]$ to θ_0 from the one from $[a, 0]$ to θ_1 and then subtracting the triangle part (in grey).

Goodness of fit

When there is no perfect solution **eulerr** generates an approximation, which may or may not fit well. It is not, however, obvious what it means for it to "fit well". The loss metric (4) we used in the optimizer does measure this but not in a standadized way. Wilkinson [4] uses the *stress* metric (5),

$$\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} y_i^2} \tag{5}$$

where $\hat{y}_i$ is the ordinary least squares estimate from obtained by regressing the fitted areas on the original areas through the origin. It is not, however, clear what the

Micallef and Rodgers [3], on the other hand, uses *diagError* (6),

$$\max_{i=1,2,\ldots,n}\left|\frac{y_i}{\sum y_i} - \frac{\hat{y}_i}{\sum \hat{y}_i}\right|. \tag{6}$$

Eulerr adopts both measures and lets the user decide which is more appropriate.

Visualization

After we have have ascertained that our Euler diagram is adequate, we turn to visualizing the solution. **eulerr** leverages the **Lattice** graphics system [12] for **R** to offer the user exhaustive control over the output.

Plotting the ellipses is straightforward. Often, however, we would like to label the ellipses and their intersections with text and this is more involved. The reason is that the intersections are often concave and lack a well-defined center.

We know of no analytical solution to this problem. As usual, however, the next-best thing turns out to be a numerical such. First, we locate a point that is inside the required region by spreading points across one of the discs involed in the set combination. To spread points uniformly, we use a modification of *Vogel's method* [13, 14] adapted to ellipses Equation (7).

$$\left(p_k = (\rho_k, \theta_k) = \left(r\sqrt{\frac{k}{n}},\ \pi(3 - \sqrt{5})(k-1)\right)\right)_{k=1}^{n}, \tag{7}$$

Our modification involves stretching, scaling, and rotating the points after they have been formed.

After we spread our points throughout the ellipse and find one point, X, that matches our desired intersection, we then proceed to optimize its position numerically. We look for a position inside the intersection that maximizes the smallest distance to all the ellipses in our diagram to provide as much spacing as possible for the label. This is a maximin problem with a loss function equal to

$$\max \min (X - E_i), \quad i = 0, 1, \ldots, n, \tag{8}$$

where E represent the ellipses.

The method for computing the distance from a point to an ellipses has been described in **?**] and involves a bisection optimizer.

To optimize this, we employ a version of the *Nelder–Mead Method* [15] which is translated from Matlab code by Kelley [16] and adapted for **eulerr** to ensure that convergence is fast and that the simplex remains within the intersection boundarie (since we want the local maximum). The method is visualized in Figure 5.

Layout

A side effect of running an unconstrained optimizer 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 [17] 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 asn appropriate size and aspect ratio of the golden rule.

Aesthetics

Euler diagrams display both quantitative and qualitative aspects of data. The quantitative aspect is covered by the areas of the sets and their intersections, which is the main focus of this paper. The qualitative aspect is the mapping

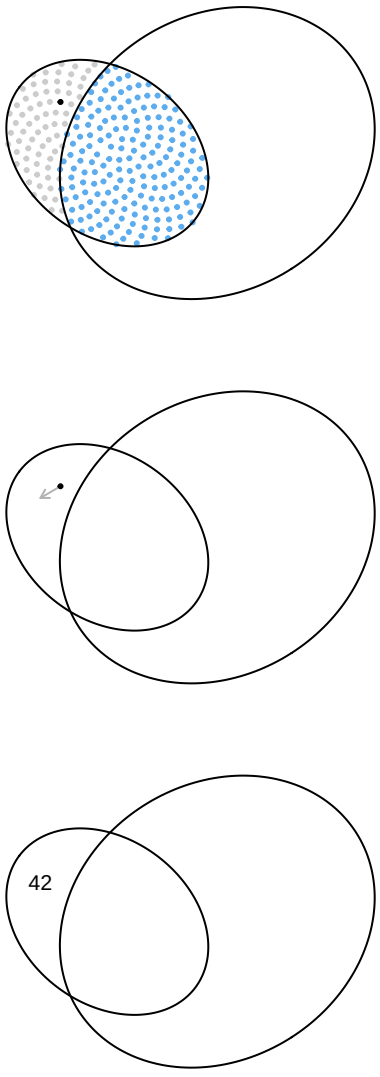

Figure 5: The method eulerr uses to locate an optimal position for a label in three steps from top to bottom: first, we spread sample points on one of the ellipses and pick one inside the intersection of interest, then we begin moving it numerically, and finally place our label.

of each set to a quality or category such as the membership of a group or the ownership of a certain gene.

In the diagram, these qualities needs to be separated through qualities such as

- 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 can interpret the diagram with ease and clarity.

The most common choice among aesthetics is color, which provides useful information without extraneous chart junk [**?**]. The issue with color, however, is that it cannot be perceived perfectly by rought 10% of the population. Moreover, color is often a premium in scientific publications and adds nothing to a diagram of two diagrams.

For these reasons, **eulerr** by default distinguishes ellipses with color using a color palette generated via the **R** package **qualpalr** [18], 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.

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 and since it only produces images as html, we refrain from using it in our results. Only **eulerr** and **venneuler** support more than 3 sets, which is why there are only 3–set results for **Vennerable**.

These results were computed on a PC with the following specs and software configuration:

- Microsoft Windows Pro 10 x64
- Intel® Core™ i7-4500U CPU @ 1.80GHz, 2 cores
- 8 Gb memory
- R 3.4.1, x64

Case studies

We begin our examination of **eulerr** by studying a difficult set relationship from Wilkinson [4],

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

where we use the & operator analogously to ω in Definition 1. We fit this specification with **venneuler** and **eulerr**, in the latter case using both circles and ellipses (Figure 6).

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

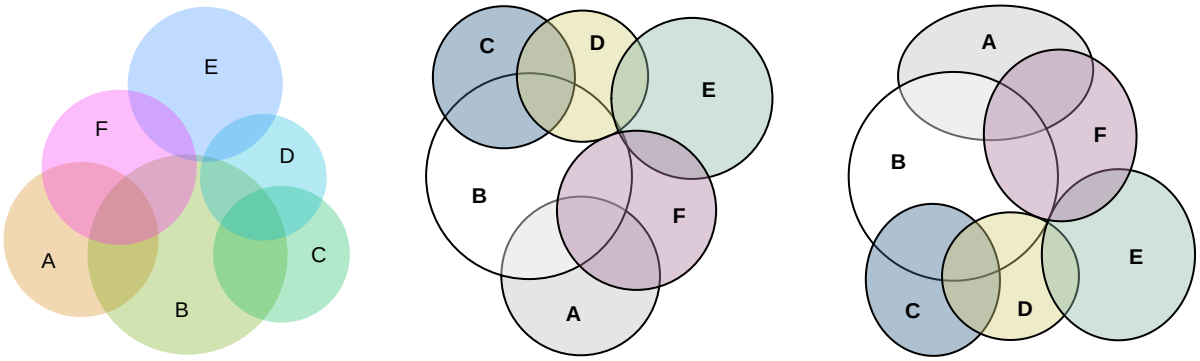

Figure 6: A comparison of a Euler diagram generated with **venneuler** with two generated from **eulerr** with circles and ellipses respectively. The stress of the solutions are 0.006, 0.004, and 0.000 respectively.

Consistency

To compare the consistency among **eulerr**, **venneuler**, and **Vennerable**, we generate $i = 3, 4, \ldots 8$ circles with radii (r_i) and coordinates (x_i and y_i) given from

$$\begin{aligned} r_i &\sim \mathcal{U}(0.3, 0.6) \\ x_i, y_i &\sim \mathcal{U}(0, 1). \end{aligned}$$

We run this simulation through 100 iterations for each i. Next, we compute the required areas, ω (from Definition 1), for each iteration and fit a Euler diagram using the aforementioned packages. Finally, we compute and return *diagError* (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 theoretically achievable since our Euler diagrams are formed from sampled circles that all have perfect solutions. The left panel of Figure 7 shows the results of our simulation. Next, we repeat this experiment with ellipses instead of circles, this time with semiaxes (a_i and b_i), coordinates (x_i and y_i), and rotation axes (ϕ_i) drawn from

$$\begin{aligned} x_i, y_i &\sim \mathcal{U}(0, 1) \\ a_i &\sim \mathcal{U}(0.2, 0.8) \\ b_i &\sim \mathcal{U}(0.2, 0.8) \\ phi &\sim \mathcal{U}(0, 2\pi). \end{aligned}$$

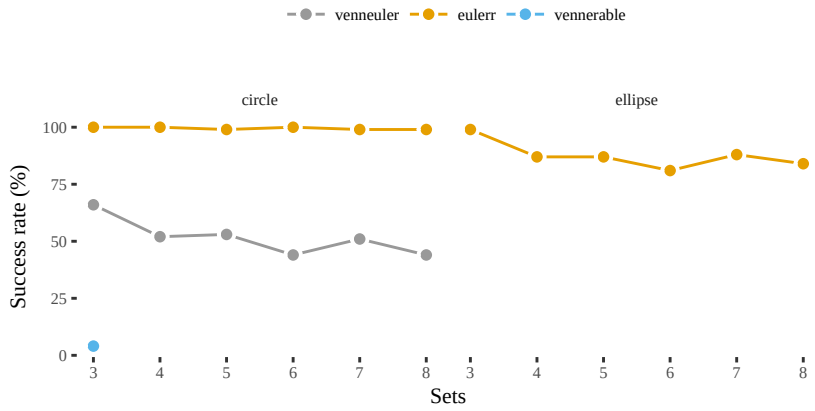

Figure 7: Reproducibility tests for ellipses and circles generated from the distributions in ?? . Note that **Vennerable** only supports Euler diagrams for three sets, which is why its data is absent for the other cases.

We plot this in the right panel of Figure 7.

Accuracy

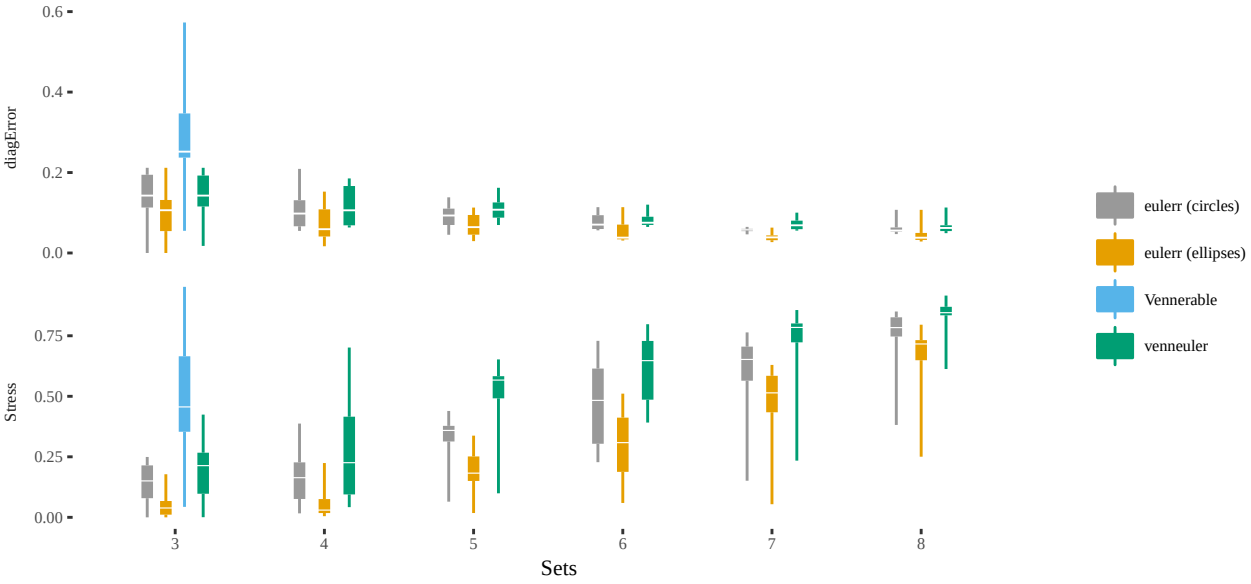

Figure 8: Accuracy tests of set relationships that may or may not have perfect solutions

Performance

Discussion

In terms of performance, we saw that..

The performance of **eulerr** is bottlenecked by the final optimizer because it has to examine every possible intersection when the areas are computed, thus converging in $O(2^n)$ time. Wilkinson [4], meanwhile, report convergence in $O(n)$ time.

The redeeming quality of **eulerr** lies in its implementation in the C++ programming language via the **R** packages **Rcpp** [19] and **RcppArmadillo** [20].

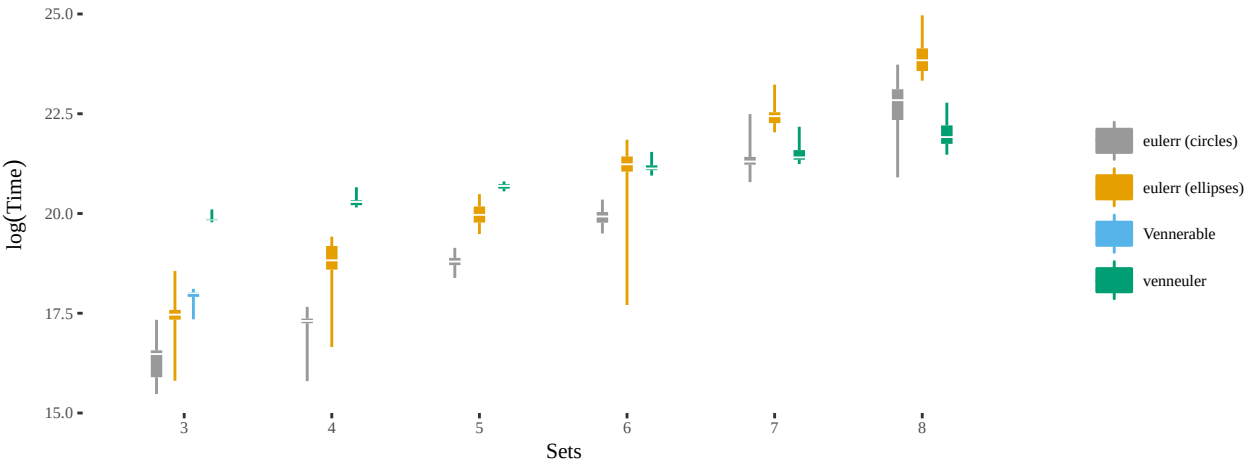

Figure 9: Performance tests.

References

[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] 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.

[5] 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.

[6] Luana Micallef. *Visualizing Set Relations and Cardinalities Using Venn and Euler Diagrams*. phd, University of Kent, September 2013. URL https://kar.kent.ac.uk/47958/.

[7] 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.

[8] 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.

[9] 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. Google-Books-ID: F_NP8Kub2XYC.

[10] David Eberly. The Area of Intersecting Ellipses, November 2016. URL https://www.geometrictools.com/Documentation/AreaIntersectingEllipses.pdf.

[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] 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.

[13] Mary K. Arthur. Point Picking and Distributing on the Disc and 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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/.

[20] 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.