LUND
UNIVERSITY

Bachelor Thesis

eulerr: Area-Proportional Euler Diagrams with Ellipses

Johan Larsson

supervised by
Peter Gustafsson

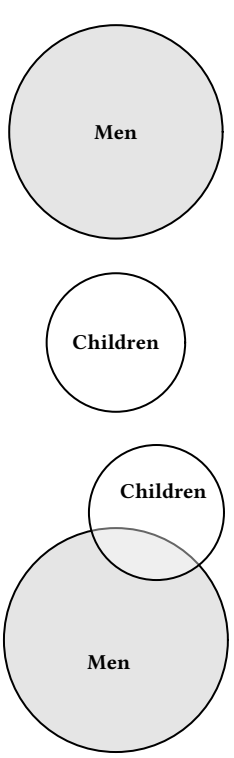
September 29, 2017

Contents

Background

Visualizations are arguably the most intuitive displays of data. Data visualization work on multiple dimensions and, if done right, possess the potential to convey intricate relationships that a single statistic or table of numbers never could.

Yet visualizations are only informative if their aesthetics represent some type of relationship. Consider, for instance, a disc with a radius of 2 cm labelled *Men*[1]—it says nothing by itself, yet if we juxtapose it with a 1 cm-radius disc labelled *Children*[2], the graphic starts to become informative. Proceed to intersect the two discs, so as to generate a visible overlap[3], and we have managed to visualize both the relative proportions of men and children and their intersection. The diagram we have constructed is a *Euler diagram* and is the topic of this paper.



The Euler diagram, originally proposed by Leonard Euler [1], is the 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 also be area-proportional, which is to say that each separate surface of the diagram is proportional 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 geometry is 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 are most common, and for good reason, since they are easily interepreted and preferred by most viewers. In spite of this, circles do not always lend themselves to accurate representations. Consider, for instance the following three-set relationship:

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

There is no way to visualize this relationship perfectly with circles because they cannot be arranged so that the $A \cap B \cap C$ overlap remains zero whilst $A \cap B$, $A \cap C$, and $B \cap C$ stay non-zero. With ellipses, however, we

solve this problem since they can be both stretched and rotated, allowing for a perfect fit (Figure 1).

Put differently, we might say that circles sport three degrees of freedom: a center consisting of x- and y-coordinates h and k, as well as radius r. Ellipses, meanwhile, have five: the aforementioned (h, 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 always impossible, given that $2^4 - 1 = 15$ regions are required but circles yield at most 13 unique intersections. This, however, is not the case with ellipses in that they may intersect in up to 4, rather than 2, points.

Elliptical Euler diagrams are implemented in the **eulerAPE** software [3], yet only for three sets that, additionally, are required to intersect, following the motivation that Euler diagrams with more sets often lack adequate solutions and are complex, making implementations difficult [7].

Euler diagrams do not have analytical solutions [ref] and have to be solved numerically. Most of the current implementations do this in two steps, first finding a rough initial configuration and then finalizing it in a second and more accurate algorithm. For the initial configuration, Micallef [7], for instance, uses a greedy algorithm that tries to minimze the error in the three-way intersection. Wilkinson [5] uses multi-dimensional scaling with jacobian distances, taking only pairwise relationships into account. Frederickson [8] uses a constrained version of the latter that is based on euclidean distances instead and separately runs a greedy algorithm, picking the best fit. All of the algorithms use circles in the initial configuration.

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.

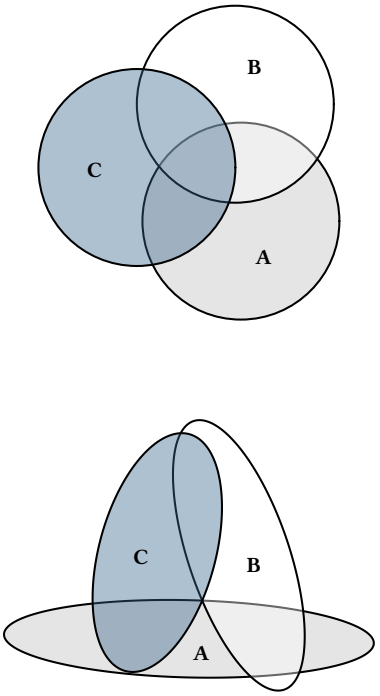


Figure 1: A set relationship depicted erroneously with circles and perfectly with ellipses.

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

In the following sections, we explain how **eulerr** deals with each of these steps in turn.

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[4],
- unions and identities[5],
- a matrix of binary (or boolean) indices[6],
- a list of sample spaces[7], or
- a two- or three-way table [8].

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.

Whichever type of input is provided, **eulerr** translates it to the first, *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,* $F = F_1, F_2, \ldots, F_N$*, and their* $n = 2^N - 1$ *intersections, we define* ω *as the intersections of these sets and their*

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

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

[6]
$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

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

[8]

Survived?	No	Yes
Child	52.00	57.00
Adult	1438.00	654.00

relative complements, such that

$$\omega_1 = F_1 \setminus \bigcap_{j=2}^{N} F_j$$

$$\omega_2 = (F_2 \cap F_3) \setminus \bigcap_{j=3}^{n} F_j$$

$$\omega_3 = \bigcap_{i=1}^{3} F_i \setminus \bigcap_{j=4}^{N} F_j$$

$$\vdots$$

$$\omega_n = \bigcap_{j=1}^{N} F_j$$

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.

Initial configuration

For our initial configuration we rely on a constrained version of multi-dimensional scaling (MDS) from **venn.js** [8], 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 (1) to numerically find the required distance, d, for each set of two ellipses,

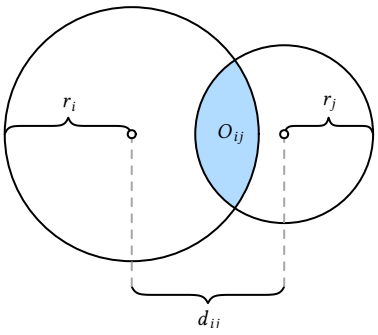


Figure 2: The circle–circle overlap is computed as a function of the discs' separation (d_{ij}), radii (r_i, r_j), and area of overlap (O_{ij}).

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

where r_i and r_j are the radii of the circles representing the i:th and j:th sets respectively, O_{ij} their overlap, and d_{ij} 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()`[9]. Convergence is fast and neglible next to our later optimization procedures.

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

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**

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) is the normal sums of squares between the optimal distance between two sets, d, that we found in (1) and the actual distance in the layout we are currently exploring. The gradient (3) is retrieved as usual by taking the derivative of the loss function.

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

$$\vec{\nabla} f(\boldsymbol{v}_i) = \sum_{i=1}^{N} \begin{cases} \vec{0} & (F_i \cap F_j = \emptyset) \wedge (O_{ij} = \emptyset) \\ \vec{0} & \big((F_i \subseteq F_j) \vee (F_i \supseteq F_j)\big) \wedge (O_{ij} = \emptyset) \\ 4\left((\boldsymbol{v}_i - \boldsymbol{v}_j)^{\mathrm{T}}(\boldsymbol{v}_i - \boldsymbol{v}_j) - d_{ij}^2\right)(\boldsymbol{v}_i - \boldsymbol{v}_j) & \text{otherwise,} \end{cases} \tag{3}$$

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

We optimize (2) using the nonlinear optimizer `nlm()` from the R core package **stats**, which is a translation from FORTRAN code developed by Schnabel et al. [9] that uses a mix of Newton and Quasi-Newton algorithms. It makes use of the Hessian, which we presently compute numerically[10].

[10] There is a bug in the current version of the package, causing the analytic Hessian to be updated incorrectly.

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.

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.

Computing overlaps of ellipses is not trivial, which is evident in that most methods resort to approximations such as quad-tree binning [5] or polygon intersections [6]. Contrastingly, Micallef [7], Frederickson [8] do compute overlap areas exactly but, unlike the approximative types,

require that we first find all intersection points between the ellipses. **eulerr**'s approach to this is outlined in Richter-Gebert [10] and based in *projective*, as opposed to *euclidean*, geometry.

To collect all the intersection points, we naturally 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 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 3).

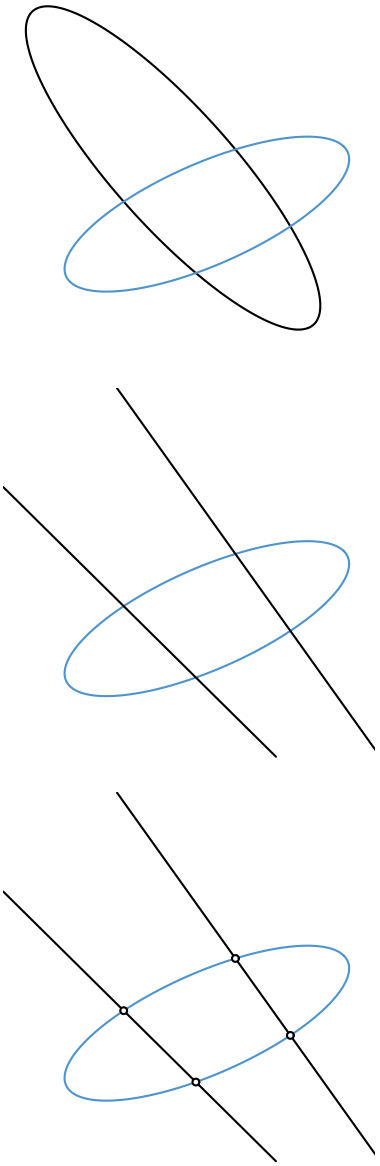


Figure 3: The process (from top to bottom) used to intersect two ellipses, here yielding 4 points.

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 (Figure 4).

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[11]. Then, we acknowledge that each elliptical segment is formed from an arc of the ellipse that is shared by both points[12]. Now that we have two points on an ellipse, we can find the area of the ellipse segment using an algorithm from Eberly [11]. 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,

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

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

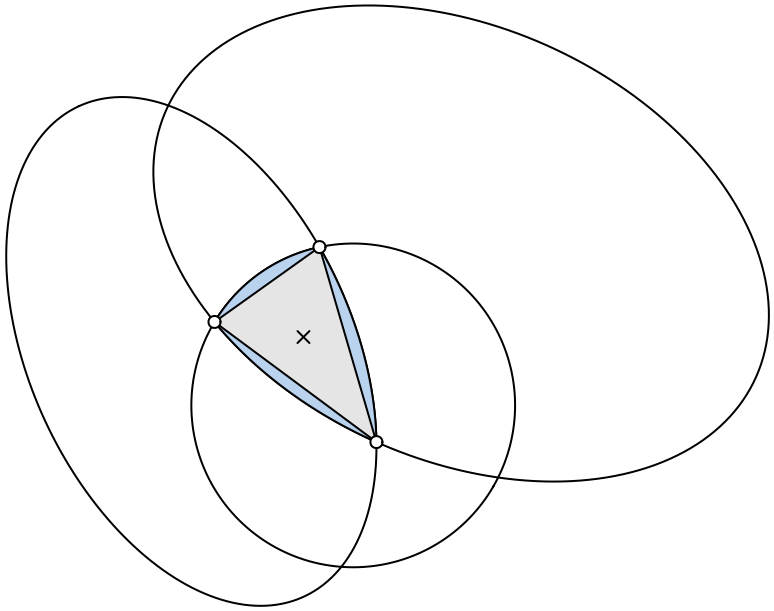


Figure 4: The overlap area between three ellipses is the sum of a convex polygon (in grey) and 2–3 ellipse segments (in blue).

4. subtract the smaller of these sectors from the larger, and
5. subtract the triangle section to finally find the segment area ().

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

This procedure is illustrated in Figure 5.

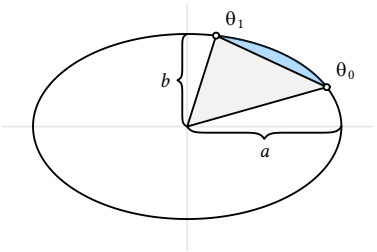


Figure 5: The elliptical segment in blue is found by first subtracting the elliptical sector from $(a, 0)$ to θ_0 from the one from $(a, 0)$ to θ_1 and then subtracting the triangle part (in grey).

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 semi-axes 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,

$$\sum_{i=1}^{n}(A_i - \omega_i)^2 \tag{4}$$

Goodness of fit

When **eulerr** cannot find a perfect solution it offers an approximation, 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,

$$\text{Stress} = \frac{\sum_{i=1}^{n}(\omega_i - A_i)^2}{\sum_{i=1}^{n}(A_i - \bar{A})}, \tag{5}$$

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

$$\text{diagError} = \max_{i=1,2,\ldots,n} \left| \frac{\omega_i}{\sum_{i=1}^{n} \omega_i} - \frac{A_i}{\sum_{i=1}^{n} A_i} \right|, \tag{6}$$

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

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**.

All of the results were computed on a PC running R version 3.4.2[13]

[13] Specifically, the PC featured the following configuration:

- Microsoft Windows Pro 10 x64
- Intel® Core™ i7-4500U CPU @ 1.80GHz, 2 cores
- 8 Gb memory
- R 3.4.2, x64

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 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**.

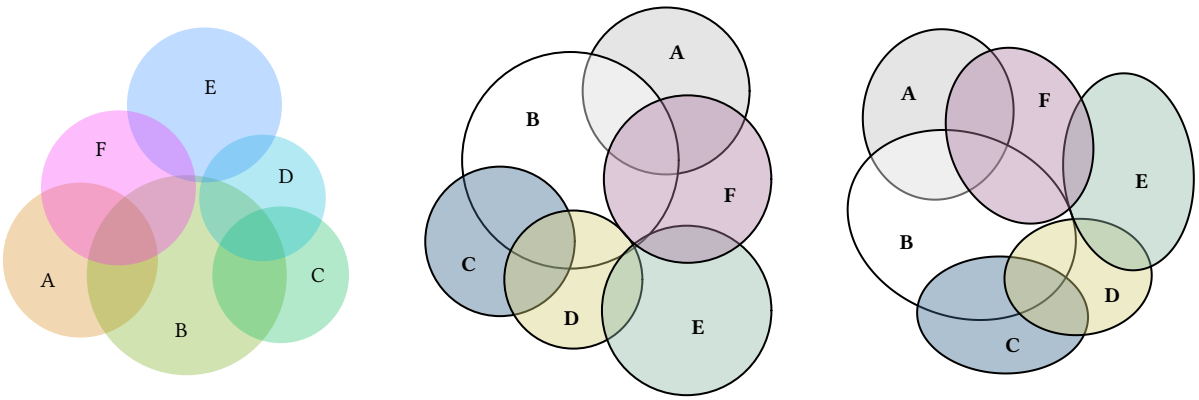


Figure 6: A comparison of a Euler diagram generated with **venneuler** with two generated from **eulerr** with circles and ellipses respectively. The stress of the solutions are 0.006, 0.004, and

Consistency

To compare the consistency among **eulerr**, **venneuler**, and **Vennerable**, we simulate random circles and ellipses, compute their areas, and attempt to reproduce the original diagram using the software packages that we are studying. We run the simulation through 100 iterations for each number of sets, $i = 3, 4, \ldots, 8$.

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

$$\begin{aligned} r_i &\sim \mathcal{U}(0.3, 0.6) \\ h_i, k_i &\sim \mathcal{U}(0, 1). \end{aligned} \tag{7}$$

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

$$\begin{aligned} h_i, k_i &\sim \mathcal{U}(0, 1) \\ a_i, b_i &\sim \mathcal{U}(0.2, 0.8) \\ \phi_i &\sim \mathcal{U}(0, 2\pi). \end{aligned} \tag{8}$$

The algorithm is formalized in Algorithm 1.

1: **for** $i = 3, 4, \ldots, 8$ **do**
2: **for** $j = 1, 2, \ldots, 100$ **do**
3: **if** Circle **then**
4: $h, k \leftarrow \mathcal{U}(0, 1)$
5: $r \leftarrow \mathcal{U}(0.3, 0.6)$
6: $\omega \leftarrow$ `findOverlaps`(h, k, r)
7: **else if** Ellipse **then**
8: $h, k \leftarrow \mathcal{U}(0, 1)$
9: $a, b \leftarrow \mathcal{U}(0.2, 0.8)$
10: $\phi \leftarrow \mathcal{U}(0, 2\pi)$
11: $\omega \leftarrow$ `findOverlaps`(h, k, a, b, ϕ)
12: **end if**
13: $A \leftarrow$ `fitDiagram`(ω)
14: diagError $\leftarrow \max_{k=1,2,\ldots,2^i-1} \left| \frac{\omega_k}{\sum \omega_k} - \frac{A_k}{\sum A_k} \right|$
15: Stress $\leftarrow \frac{\sum(\omega - A_k)^2}{\sum(A_k - \bar{A})^2}$
16: **end for**
17: **end for**

where n is the

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

eulerr outperforms both **Vennerable** and **venneuler** in consistency Figure 7. It is able to reproduce Euler diagrams for almost all of

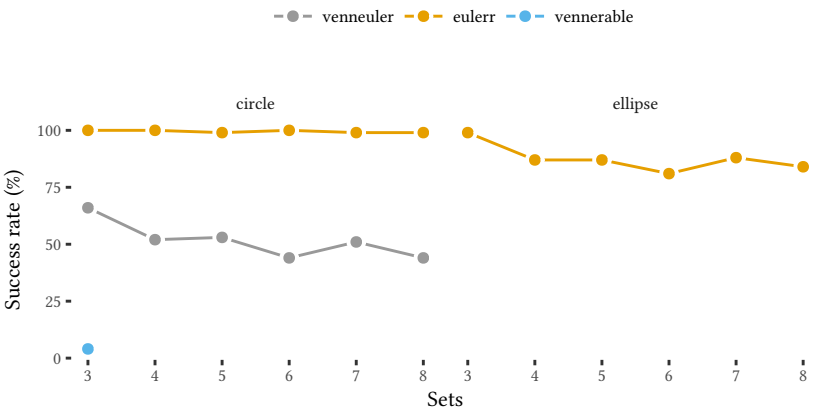


Figure 7: Reproducibility tests for ellipses and circles generated from the distributions from (7). Note that **Vennerable** only supports Euler diagrams for three sets, which is why its data is absent for the other cases.

the circles and for all 3-set ellipses. For ellipses of 4 or more sets, the consistency drops considerably, yet remains above 81%. **Vennerable**, which is only able to produce 3-set diagrams, only produces accurate diagrams for 0.04% of the random layouts and moreover fails with an error in 1 cases.

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.

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 $N = 3, 4, \ldots, 8$ sets we initialize $2^N - 1$ permutations of set combinations and select one for each set, initializing these to a numer in $\mathcal{U}(0, 1)$. After this, we pick 0 to $\binom{N-2}{1}$ elements from the $2^N - N - 1$ remaining permutations and assign to them a random in $\mathcal{U}(0, 1)$ as before. The algorithm is formalized in Algorithm 2

for $N = 3, 4, \ldots, 8$ **do**
  initialize $\omega = \{\omega_1, \omega_2, \ldots, \omega_{2^N-1}\}$ to zero
  for $i = 1, 2, \ldots, N$ **do**
    $j \leftarrow$ random index in $\{\omega : \omega \cap F_i \neq \emptyset\}$
    $\omega_j \leftarrow \mathcal{U}(0, 1)$
  end for
  $\omega_S \leftarrow \mathcal{U}\{0, N\}$ random elements from $\{\omega : \omega = 0\}$
  $\omega_S \leftarrow \mathcal{U}(0, 1)$
  fit a Euler diagram to ω
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.

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**.

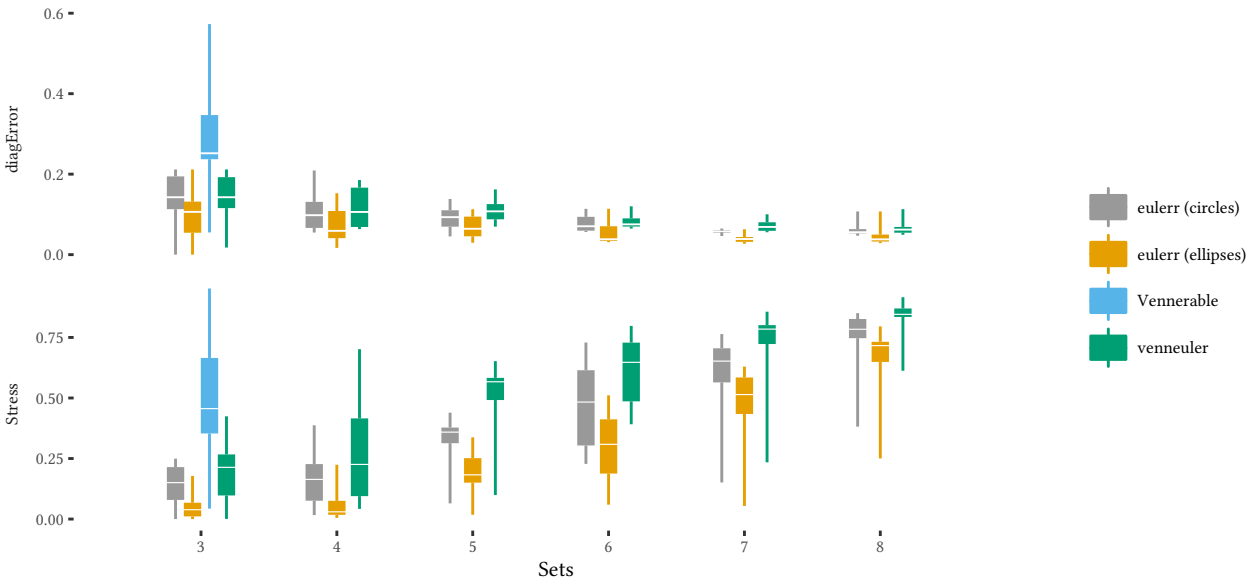


Figure 8: Accuracy tests of set relationships that may or may not have perfect solutions, generated from (8).

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.

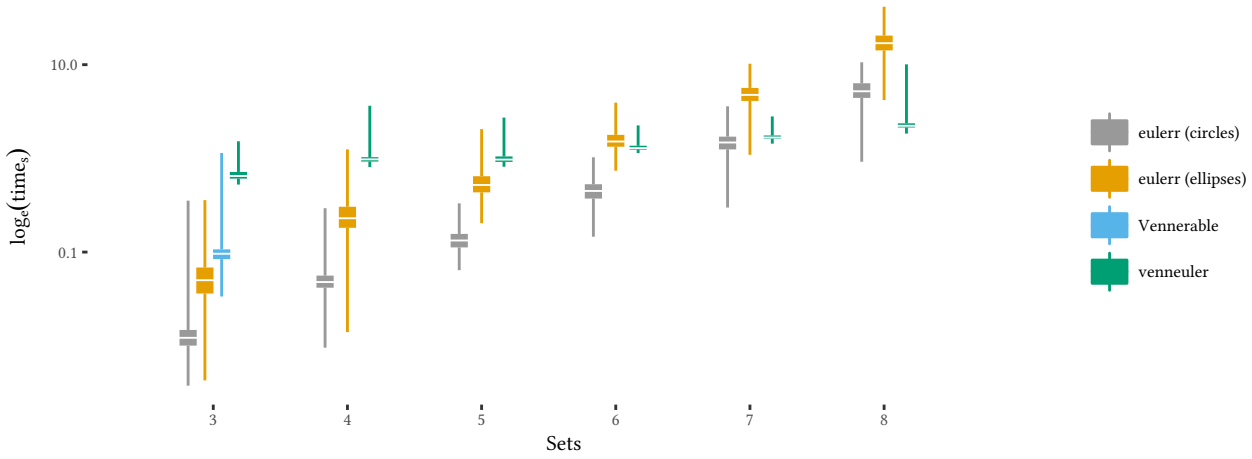


Figure 9: Performance of **eulerr**, **venneuler**, and **Vennerable** on random set relationships of 3 to 8 sets

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** [12] and **RcppArmadillo** [13]. 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 9. 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 [14] and in a scholarly paper for up to three ellipses [**?**] but has to our knowledge not previously been generalized to any number of ellipses.

Appendices

Appendix 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 [15] for R to offer easy, as well as 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 like to label the ellipses and their intersections with text, which is considerably more involved.

Labeling

Labeling the ellipses is not straightforward—the intersections are often irregular and lack a well-defined center. We know of no analytical solution to this problem. As usual, 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* [16, 17] adapted to ellipses. Vogel's method spreads points across a disc using

$$p_k = \begin{bmatrix} \rho_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} r\sqrt{\frac{k}{n}} \\ \pi(3 - \sqrt{5})(k - 1) \end{bmatrix} \quad \text{for } k = 1, 2, \ldots, n. \tag{9}$$

In our modification, we use the points formed in (9) and scale, rotate, and translate them 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 smallest distance to all the ellipses in our diagram to provide as much margin as possible for the label. This is a maximin problem with a loss function equal to

$$\max_{x,\, y \in \mathbb{R}^2} \min_{i = 1, 2, \ldots, N} \delta(x, y, E_i) \tag{10}$$

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* [18] which has been translated from Kelley [19]—originally written in Matlab—and adapted for **eulerr** to ensure that convergence is fast and that the simplex remains within the intersection boundaries (since we want the local maximum). The method is visualized in Figure 10.

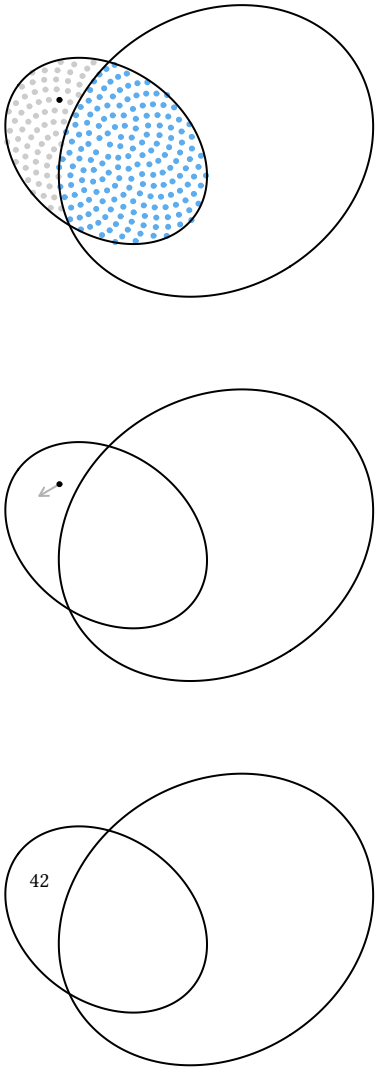


Figure 10: The method eulerr uses to locate an optimal position for a label in three steps from top to bottom: first, we spread sample points on one of the ellipses and pick one inside the intersection of interest, then we begin moving it numerically, and finally place our label.

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.

A common choice among these 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** [20], 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.

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 [21] 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.

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] 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/.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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/.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

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

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

[21] 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.