eulerr: Area-Proportional Euler Diagrams with Ellipses

Bachelor Thesis

Johan Larsson[1] and Peter Gustafsson[1]

[1]*Lund University*

2017-09-04

Contents

1 Background

Relationships between groups and sets are the focus of many scientific disciplines. In biomedicine, for instance, the overlap between sets of genes is often of interest. In epidemiology, the interactions between diseases are frequently studied. Likewise, the social sciences are often involved in studying demographics where the commonalities between various groups are analyzed.

Visualizations of such relationships are central to understanding them. The commonest visualization is Venn diagrams. Venn diagrams are a subset of Euler diagrams, which were originally proposed by Leonard Euler (1707–1783) [1]. Whereas Venn diagrams require that all 2^n possible intersection are present – even if they are empty – Euler diagrams stipulate no such requirement.

Euler diagrams can be area-proportional, which is the case for non-Venn diagrams. Area-proportional diagrams are most easily understood as their interpretations do not depend on any numbers. This paper will henceforth be concerned only with such designs and the term *Euler diagram* will be considered to refer to area-proportional diagrams.

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

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

There is no way to visualize this relationship perfectly with circles yet with ellipses there *is* in fact a perfect solution (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.

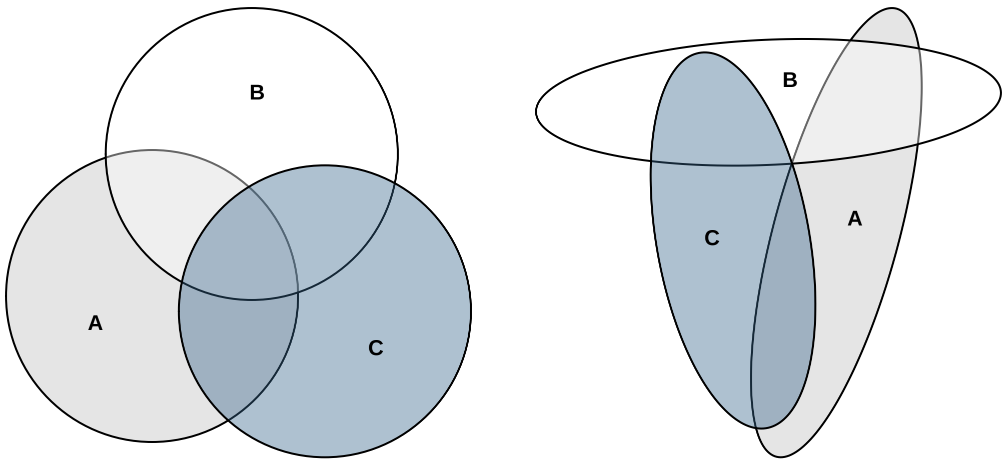

Figure 1. A set relationship depicted with circles and ellipses.

Elliptical Euler diagrams have been successfully implemented in the **eulerAPE** software [3], yet only for three sets that moreover need to intersect, 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 eulerr 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 provide elliptical Euler diagrams for, theoretically, any number of sets.

1.1 Aim

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.

2 Method

A Euler diagram might be conceived of as a model of data, akin to any other statistical model. As such, it requires

- some data,
- a process through with which the model is fit,
- a test to check how well it fits, and finally
- a presentation of the result.

2.1 Input

Every Euler diagram begins with data, which, in one way or another, is always a description of set relationships. **eulerr** tries to be as friendly as possible, accepting many forms (Table 1).

Table 1. Types of data.

Form	Example
Named vector	$A = 10 \quad B = 5 \quad A \cap B = 2$
Matrix of logicals or binaries	$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$
A list of sample spaces	$A = \{ab, bb, bc\}$ $B = \{aa, bc, cc\}$ $C = \{bb, bb, cc\}$

2.2 Initial configuration

For our initial configuration, we work exclusively with circles. We begin by mapping the disjoint set combinations to areas and, given these, figure out the required pairwise distance between the sets to achieve a circle–circle overlap matching these disjoint set combinations. We do this numerically, using the formula for a circle–circle overlap,

$$A = r_1^2 \arccos\left(\frac{d^2 + r_1^2 - r_2^2}{2dr_1}\right) + r_2^2 \arccos\left(\frac{d^2 + r_2^2 - r_1^2}{2dr_2}\right) - \frac{1}{2}\sqrt{(-d + r_1 + r_2)(d + r_1 - r_2)(d - r_1 + r_2)(d + r_1 + r_2)}, \quad (1)$$

where r_1 and r_2 are the radii of the first and second circles respectively and d the distance between them.

Although r_1 and r_2 are known, d is not, wherefore we approximate it numerically. Here, our loss function is the squared difference between A and the desired overlap, $(A - D)^2$, which we then ptimize using the **R**'s built-in `optimize()`: a "combination of golden section search and successive parabolic interpolation." Convergence is fast and neglible in relation to our later, more demanding, operations.

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 is based on a similar 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. However, since 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 0 when, for instance, the set relationsships *and* the candidate ellipses are disjoint. Then, to optimize the pairwise relationsships between sets, **eulerr** uses the following loss and gradient functions.

$$\text{loss} = \sum_i \sum_j \begin{cases} 0 & \text{disjoint}(i, j) \\ 0 & \text{subset}(i, j) \\ ((X_i - X_j)^T (X_i - X_j) - D_{ij}^2)^2 & \text{otherwise} \end{cases}$$

$$\nabla f(X_i) = \sum_j \begin{cases} \vec{0} & \text{disjoint}(i, j) \\ \vec{0} & \text{subset}(i, j) \\ 4((X_i - X_j)^T (X_i - X_j) - D_{ij}^2)(X_i - X_j) & \text{otherwise} \end{cases}$$

Source: Better Venn Diagrams by Ben Fredrickson, which includes a nice interactive demonstration.

Fredrickson uses the *Polak–Ribière Conjugate Gradient Method* to optimize the initial layout. In our experience this method occasionally ends up in local minima, which is why we have opted to use `nlm()` from the **R** core package `stats`, which is a translation from FORTRAN code developed by Schnabel, Koonatz, and Weiss [8] and uses a mixture of algorithms (Newton and Quasi-Newton).

This initial configuration will work perfectly for any 1–2 set combinations and as well as possible with 3 sets if we use circles but for all other combinations there is usually a need to fine tune the configuration.

2.3 Final configuration

2.3.1 Intersection between ellipses

Splitting a conic

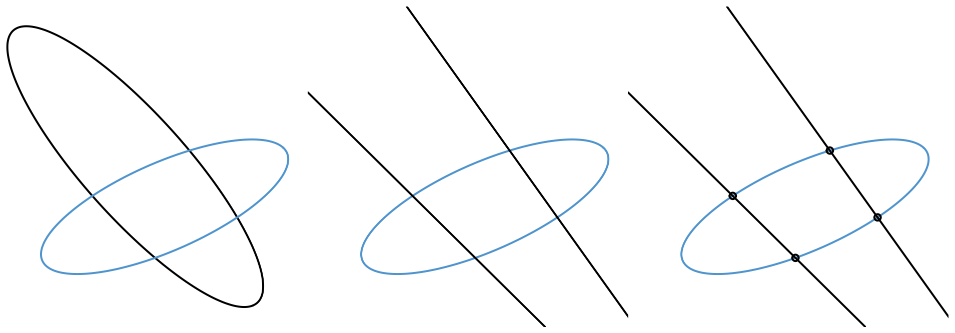

Figure 2. The process (from left to right) used to intersect two ellipses, here yielding 4 points.

2.3.2 Areas of overlapping ellipses

2.4 Visualization

3 Results

3.1 Consistency

3.2 Accuracy

3.3 Performance

4 Discussion

5 Pre-processing

6 Initial configuration

Our initial layout can be setup in a number of ways; **eulerr** uses one of the methods from Fredrickson's venn.js, which features a constrained version of multi-dimensional scaling (MDS) based on that of Wilkinson's **R** package venneuler [4]. **venneuler** tries to place disjoint and subset exactly neck-in-neck and at the exact midpoint of the set respectively. However, since 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 occupy some of that space.

The MDS algorithm from **venn.js** circumvents this by assigning a loss and gradient of 0 when, for instance, the set relationsships *and* the candidate ellipses are disjoint. Then, to optimize the pairwise relationsships between sets, **eulerr** uses the following loss and gradient functions.

$$\text{loss} = \sum_i \sum_j \begin{cases} 0 & \text{disjoint}(i,j) \\ 0 & \text{subset}(i,j) \\ ((X_i - X_j)^T (X_i - X_j) - D_{ij}^2)^2 & \text{otherwise} \end{cases}$$

$$\nabla f(X_i) = \sum_j \begin{cases} \vec{0} & \text{disjoint}(i,j) \\ \vec{0} & \text{subset}(i,j) \\ 4((X_i - X_j)^T (X_i - X_j) - D_{ij}^2)(X_i - X_j) & \text{otherwise} \end{cases}$$

Source: Better Venn Diagrams by Ben Fredrickson, which includes a nice interactive demonstration.

Fredrickson uses the *Polak–Ribière Conjugate Gradient Method* to optimize the initial layout. In our experience this method occasionally ends up in local minima, which is why we have opted to use `nlm()` from the **R** core package `stats`, which is a translation from FORTRAN code developed by Schnabel, Koonatz, and Weiss [8] and uses a mixture of algorithms (Newton and Quasi-Newton).

This initial configuration will work perfectly for any 1–2 set combinations and as well as possible with 3 sets if we use circles but for all other combinations there is usually a need to fine tune the configuration.

7 Final configuration

In order to finalize the configuration we need to be able to compute the areas of the overlaps of the sets, which as it turns out, is *not* trivial. In fact, most of methods rely on approximations of the areas by, for instance, quad-tree binning (**venneuler**) or polygon intersections (**VennMaster** [5]). These methods yield reasonable estimates but, given that the computation may have to run for a vast number of iterations, are usually prohibitive in terms of performance.

venn.js and **eulerAPE** both, however, use exact algorithms. Based on the fact that any intersection of ellipses can be represented as a convex polygon with elliptical segments on the fringes, it is possible to arrive at exact area calculations.

7.1 Intersections

Finding the areas of the overlaps exactly requires that we first know the points at which the different ellipses intersect. **eulerr**'s approach to this is based on a method outlined by Richter-Gebert [9]. **eulerr** owes significant debt to the **R** package **RConics** [10], which has been tremendously helpful in developing and, especially, debugging the algorithm. Some parts of the code are in fact straight-up translations to C++ from the code in **RConics**.

The method is based in *projective geometry* (rather than euclidean). To find the intersection points, the algorithm first

- converts the two ellipses from canonical form to matrix notation. The canonical form of a rotated ellipse 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 *phi* is the counter-clockwise angle from the positive x axis to the semi-major axis *a*. *b* is the semi-minor axis whilst *(h, k)* is the center of the ellipse. This is then converted to the matrix form

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

which may be used to represent any conic. We then
- split one of the ellipses (conics) into a pencil of two lines, and subsequently
- intersect the remaining conic with these two lines, which will yield between 0 and 4 intersection points.

7.2 Areas

The next step is to calculate the area of overlap between all the possible combinations of ellipses. The solution to this was discovered, as far as I know, by Fredrickson who explains it beautifully in a blog post. It relies on finding all the intersection points between the currently examined sets that are also within these sets. It is then trivial to find the area of the convex polygon that these vertices make up. Finding the rest of the area, which is made up of the ellipse segments between subsequent points, requires a bit of trigonometry.

Here, we have used an algorithm from Eberly [11], which computes circle integral between the points on the ellipse minus the area of the triangle made up of the center of the ellipse:

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

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

As our loss function, we use the sum of squared differences between the disjoint set intersections and the areas we have computed and again use the `nlm()` optimizer to layout the set.

This optimization step is the bottleneck of the overall computations in terms of performance, being that we're optimizing over 5 parameters for every ellipse (or 3 in the case of circles) – nevertheless, we're profitting immensely from the implementation in the C++ programming language through **Rcpp** [12] and its plugin for the linear algebra library **Armadillo** [13] which ends up making the code much faster than the java-based **venneuler**.

8 Layout

Since the optimization steps are unconstrained, we run the risk of ending up with dispersed layouts. To fix this, we use the SKYLINE-BL rectangle packing algorithm [14] to pack the disjoint clusters of ellipses (in case there are any) into a heuristically chosen bin.

At the time of writing this algorithm is crudely implemented – for instance, it does not attempt to rotate the rectangles (boundaries for the ellipses) or attempt to use. Since we're dealing with a rather simple version of the rectangle packing problem, however, it seems to do the trick.

9 Output

Before we get to plotting the solution, it is useful to know how well the fit from **eulerr** matches the input. Sometimes euler diagrams are just not feasible, particular for combinations with many sets, in which case we should stop here and look for another design to visualize the set relationships.

It is not, however, obvious what it means for a euler diagram to "fit well". **venneuler** uses a metric called *stress*, which is defined as

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

where $\hat{y}_i$ is an ordinary least squares estimate from the regression of the fitted areas on the original areas that is being explored during optimization.

Meanwhile, **eulerAPE** [3] uses *diagError*:

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

Both metrics are given the user after the diagram has been fit, together with a table of residuals.

```
##        original fitted residuals region_error
## A           1.0  1.038    -0.038        0.021
## B           1.0  1.038    -0.038        0.021
## C           1.0  1.038    -0.038        0.021
## A&B         0.5  0.302     0.198        0.040
## A&C         0.5  0.302     0.198        0.040
## B&C         0.5  0.302     0.198        0.040
## A&B&C       0.0  0.247    -0.247        0.058
##
## diag_error:  0.058
## stress:      0.049
```

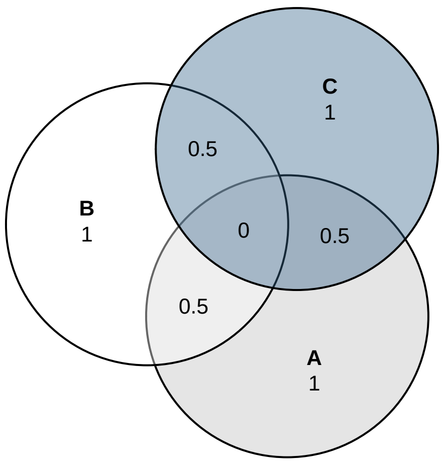

Figure 3. A plot with circles.

It is clear that this is not a good fit, which we can find out just by looking at the plot. This is a good example of when ellipses come in handy.

```
##       original fitted residuals region_error
## A          1.0    1.0         0            0
## B          1.0    1.0         0            0
## C          1.0    1.0         0            0
## A&B        0.5    0.5         0            0
## A&C        0.5    0.5         0            0
## B&C        0.5    0.5         0            0
## A&B&C      0.0    0.0         0            0
##
## diag_error:  0
## stress:      0
```

Much better.

10 Plotting

Let's face it: euler diagrams are naught without visualization. Here, **eulerr** interfaces the elegant Lattice graphics system [15] to grant the user extensive control over the output, and allow for facetted plots in case such a design was used in fitting the euler configuration.

10.1 Labelling

Most users will want to label their euler diagrams. One option is to simply add a legend

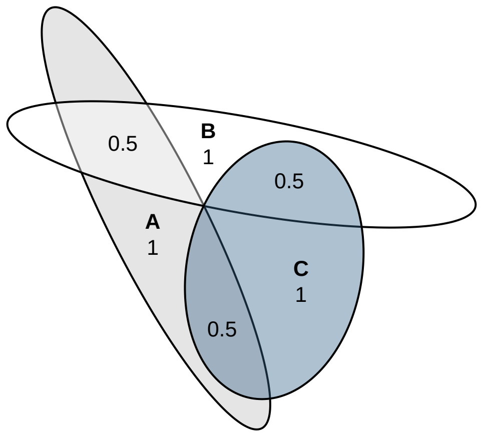

Figure 4. A plot with ellipses.

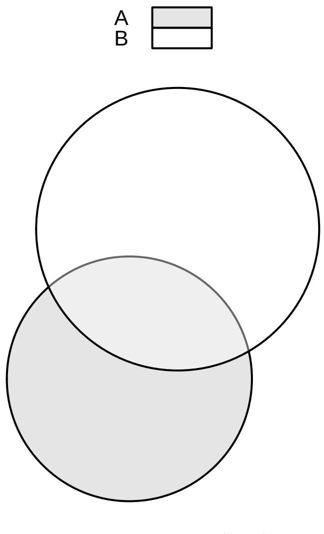

Figure 5. A simple plot.

but many will want to label their diagrams directly, perhaps with counts.

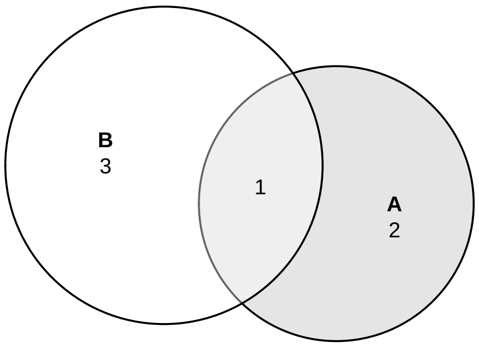

Figure 6. A plot with counts.

In this case, layout out the diagram becomes considerably more involved. Finding a reasonable spot for the text inside the diagram only lends itself to an easy solution if the shape of the intersection has a center-of-gravity inside ellipse, in which case an average of some of the points might suffice. This is often not the case, however, and we need a better solution. Specifically, what we need is a method to find the point inside the circle overlap for the counts and circle complement to the intersection for our labels.

So far, we have not been able to derive at an analyitcal solution for finding a good point, or for that matter a reliable way of finding *any* point that is in the required intersection or complement. As is often the case, the next-best thing turns out to be a numerical one. 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 *Vogel's method* [16, 17]

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

which is actually based on the golden angle.

After this, we scale, translate, and rotate the points so that they fit the desired ellipse.

After we've spread our points throughout the ellipse and found one that matches our desired combination of ellipses/sets, we then proceed to optimize its position numerically. For this, we use version of the *Nelder–Mead Method* [18] which we've translated from Matlab code by Kelley [19] and customized for **eulerr** (in particular to make sure that the simplex does not escape the intersection boundaries since we for this problem *want* the local minimum).

10.2 Coloring

Per default, the ellipses are filled with colors. The default option is to use an adaptive scheme in which colors are chosen to provide a balance between dinstinctiveness, beauty, and consideration for the color deficient. The color palette has been generated from qualpalr (developed by the

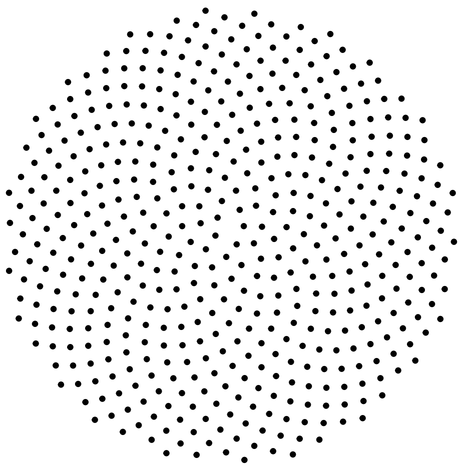

Figure 7. Spreading points on a disc with Vogel's method.

author), which automatically generates qualitative color palettes based on a model of color perception.

References

[1] Leonhard Euler, *Letters of Euler to a German Princess, on Different Subjects in Physics and Philosophy*, English. Murray and Highley, 1802. [Online]. Available: http://archive.org/details/letterseulertoaooeulegoog (visited on 08/14/2017).

[2] J. Swinton, *Vennerable: Venn and Euler area-proportional diagrams*. [Online]. Available: https://github.com/js229/Vennerable.

[3] L. Micallef and P. Rodgers, "eulerAPE: Drawing Area-Proportional 3-Venn Diagrams Using Ellipses," *PLOS ONE*, vol. 9, no. 7, e101717, 2014-jul-17, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0101717. (visited on 12/10/2016).

[4] L. Wilkinson, "Exact and Approximate Area-Proportional Circular Venn and Euler Diagrams," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 321–331, Feb. 2012, ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.56. (visited on 04/09/2016).

[5] H. A. Kestler, A. Müller, J. M. Kraus, M. Buchholz, T. M. Gress, H. Liu, D. W. Kane, B. R. Zeeberg, and J. N. Weinstein, "VennMaster: Area-proportional Euler diagrams for functional GO analysis of microarrays," *BMC Bioinformatics*, vol. 9, p. 67, Jan. 2008, ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-67. [Online]. Available: https://doi.org/10.1186/1471-2105-9-67 (visited on 08/06/2017).

[6] L. Micallef, "Visualizing Set Relations and Cardinalities Using Venn and Euler Diagrams," en, phd, University of Kent, Sep. 2013. [Online]. Available: https://kar.kent.ac.uk/47958/.

[7] B. Frederickson, *Venn.js: Area proportional Venn and Euler diagrams in JavaScript*, original-date: 2013-05-09T17:13:20Z, Nov. 2016. [Online]. Available: https://github.com/benfred/venn.js (visited on 08/27/2017).

[8] R. B. Schnabel, J. E. Koonatz, and B. E. Weiss, "A Modular System of Algorithms for Unconstrained Minimization," *ACM Trans. Math. Softw.*, vol. 11, no. 4, pp. 419–440, Dec. 1985, ISSN: 0098-3500. DOI: 10.1145/6187.6192. [Online]. Available: http://doi.acm.org/10.1145/6187.6192 (visited on 08/06/2017).

[9] J. Richter-Gebert, *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*, en, 1st ed. Berlin, Germany: Springer, Feb. 2011, Google-Books-ID: F_NP8Kub2XYC, ISBN: 978-3-642-17286-1.

[10] E. Huber, *RConics: Computations on Conics*, Dec. 2014. [Online]. Available: https://CRAN.R-project.org/package=RConics.

[11] D. Eberly, *The Area of Intersecting Ellipses*, Nov. 2016. [Online]. Available: https://www.geometrictools.com/Documentation/AreaIntersectingEllipses.pdf (visited on 08/07/2017).

[12] D. Eddelbuettel and R. François, "Rcpp: Seamless R and C++ integration," *Journal of Statistical Software*, vol. 40, no. 8, pp. 1–18, 2011. [Online]. Available: http://www.jstatsoft.org/v40/i08/.

[13] D. Eddelbuettel and C. Sanderson, "Rcpparmadillo: Accelerating r with high-performance c++ linear algebra," *Computational Statistics and Data Analysis*, vol. 71, pp. 1054–1063, Mar. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.csda.2013.02.005.

[14] J. Jylänki, *A Thousand Ways to Pack the Bin - A Practical Approach to Two-Dimensional Rectangle Bin Packing*, Feb. 2010. [Online]. Available: http://clb.demon.fi/files/RectangleBinPack.pdf (visited on 08/07/2017).

[15] D. Sarkar, *Lattice: Multivariate Data Visualization with R*, ser. Use R! New York, USA: Springer, 2008, ISBN: 978-0-387-75968-5. [Online]. Available: http://www.springer.com/us/book/9780387759685.

[16] M. K. Arthur, "Point Picking and Distributing on the Disc and Sphere," en, US Army Research Laboratory, Weapons and Materials Research Directorate, Abedeen, USA, Final ARL-TR-7333, Jul. 2015, p. 58. [Online]. Available: www.dtic.mil/get-tr-doc/pdf?AD=ADA626479.

[17] H. Vogel, "A better way to construct the sunflower head," *Mathematical Biosciences*, vol. 44, no. 3-4, pp. 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*, vol. 7, no. 4, pp. 308–313, Jan. 1965, ISSN: 0010-4620. DOI: 10.1093/comjnl/7.4.308. [Online]. Available: https://academic.oup.com/comjnl/article/7/4/308/354237/A-Simplex-Method-for-Function-Minimization.

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