Size proportional Venn diagrams in 2 and 3 dimensions: `vennplot(...)` in `R`

by

Zehao Xu

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Wayne Oldford & Prof. Marius Hofert

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

Venn diagrams are popular ways to visualize sets. Wilkinson [48] and Frederickson [18] have introduced statistical models for fitting size-proportional Venn diagram. In this paper, we will improve their methods in several aspects. An `R` function `vennplot()` is available to provide both 2D and 3D layout.

Acknowledgements

I would like to thank my supervisor, Professor R. Wayne Oldford. Thanks for his patience and I do learned a lot from him. He gave me a lot of useful suggestion and technical help in R. This paper is impossible without his help; thanks Prof Martin Lysy, he gives me a lot of help on C++; I am also appreciate Prop Marius Hofert's help on modifying this paper.

Table of Contents

List of Figures

Chapter 1

Introduction

> "Good diagrams clarify. Very good diagrams force the ideas upon the viewer. The best diagrams compellingly embody the ideas themselves."
> Wayne Oldford [9]

Venn diagrams are criticized for the use in probability, but they are good for sets [9]. In this paper, we consider sets. Chow and Ruskey [11] develop ways to construct area-proportional Venn diagrams. This improvement has great ability to convey the information from sets. In recent years variations of Venn diagrams have increasing be used in scientific publications, particularly in genetic applications. For example, Figure 1.1

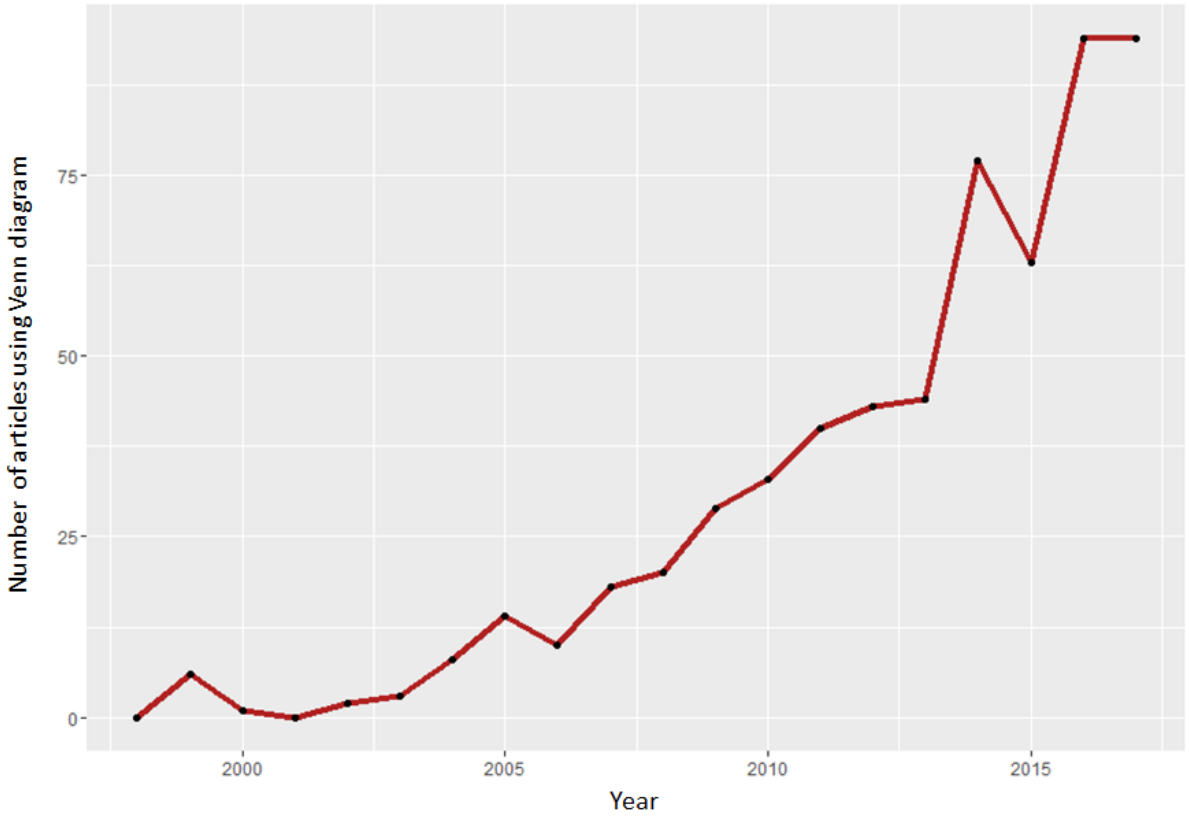


Figure 1.1: Number of articles containing "Venn diagram" over time from the journals *Genetics* and *Nature*

shows the results of an online search for "Venn diagram" over all articles appearing in the journals *Nature* and *Genetics* (including G3: Genes, Genomes, Genetics) from 1998 to 2017. As can be seen, there has been nearly a 10-fold increase since the turn of the century.

1.1 Venn and Euler diagrams

Venn diagrams for m component sets contain all possible 2^m intersections . The are constructed using multiple closed curves, like circles, ellipses, and other irregular polygons which overlap each other to show the various intersections. The interior of a closed curve

represents the elements of the set, while the exterior represents the complement of this set [45]. Figure 1.2 shows some Venn diagram examples:

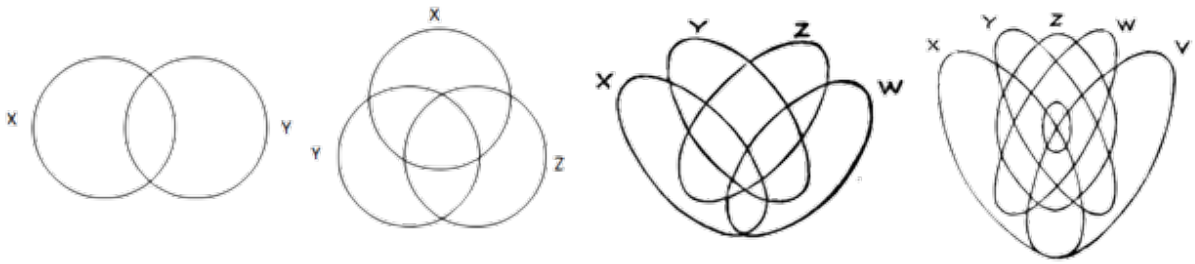


Figure 1.2: Venn diagrams with 2,3,4 and 5 sets, respectively [45]

In contrast to Venn diagrams, Euler [15] only contains the relevant relations when presenting sets. In Venn diagrams, a shaded zone may represent non-intersection, but in Euler diagrams, the corresponding zone is usually missing. Figure 1.3 shows the difference between Euler diagrams and Venn diagrams for two circles.

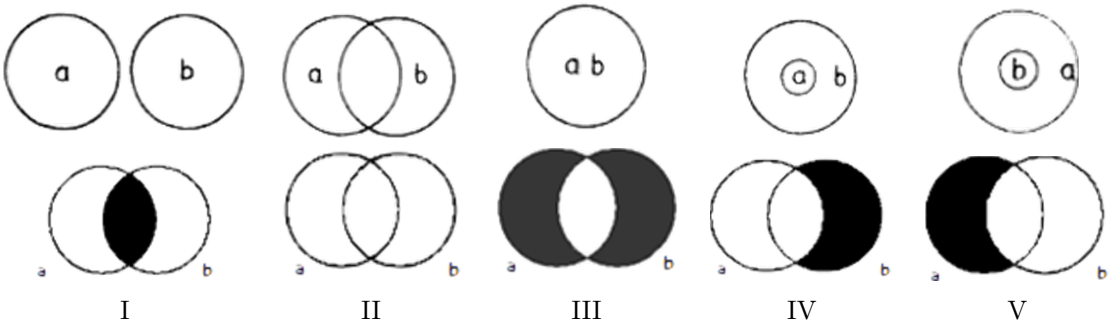


Figure 1.3: Five possible ways contain all the cases of two circles. Euler diagram is above and Venn diagram is below. I) $a \cap b = \varnothing$; II) $a \cap b$; III) a = b; IV) $a \cap b = a$; V) $a \cap b = b$.

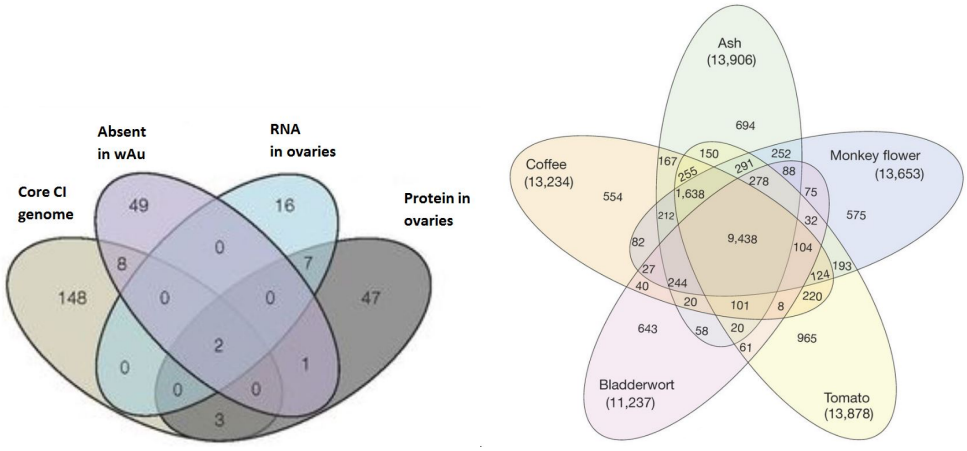
Initially, Venn and Euler diagrams were designed for symbolic logic (not for sets) with different purposes. Euler diagrams are to demonstrate the known content, but Venn diagrams are to derive the content [9]. As formal set theory developed, Venn and Euler

diagrams became useful tools to embody the relationship between sets. In this paper, in keep with common use, we will call them Venn diagrams.

1.2 Examples of Venn diagrams drawn from the scientific literatures

Interest often lies in the number of genes shared by different species, or perhaps by different groups of individuals. In this section, we will show some Venn diagrams examples from different scientific journals, such as "Science", "Nature" and "Association for Computing Machinery".

Figure 1.4 (a) shows Wmel strain gene of Wolbachia pipientis shared in a combination of the four specific criteria "Core CI genome", "Absent in wAu", "RNA in ovaries" and "Protein in ovaries" [27] and (b) describes genes shared by coffee, ash, monkey flower, tomato and bladderwort five asterid species [41]. The common features of these two graphs are that: (1) they use the same size ellipse. (2) the number of ellipses m cuts the diagram into 2^m disjoint areas and the size of each area does not match its count.

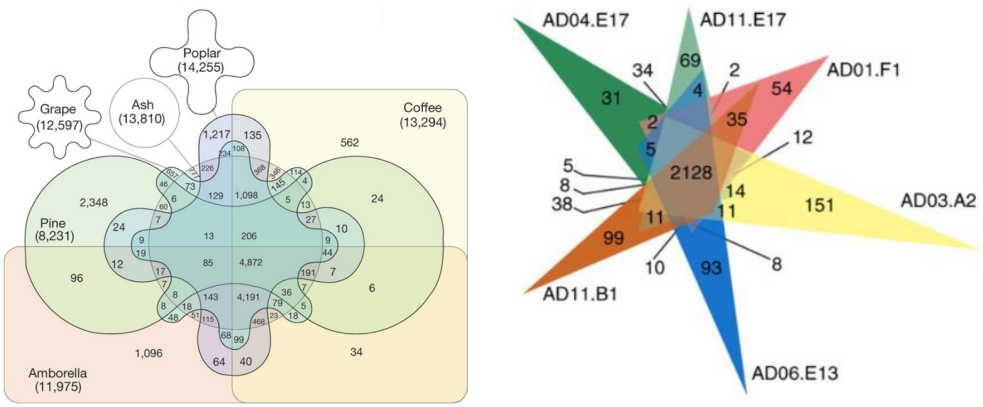


(a) (b)

Figure 1.4: (a) Illustrating the number of unique and shared wMel genes matching these four components. [27]. (b) Genes sharing by five asterid species [41].

In Figure 1.5, both (c) and (d) include six data sets: in (c), irregular polygons are drawn to illustrate genes shared by pine, grape, ash, poplar, coffee and amborella six woody species

[41]. Although it contains all the 2^6 possible intersections, the visualization of interacting characteristics is missing. For example, in the centre, 13 and 4872 share the same area; the total size of "Poplar" is the largest, however, the total area of it is the third smallest. In (d), the diagram is depicted by differently shaped triangles, showing the number of genes shared among six isolates ("AD04.E17", "AD11.E17", "AD01.F1", "AD03.A2", "AD06.E13" and "AD11.B1") from investigative patients [7]. It is difficult for triangles to separate areas into 2^6 pieces, so that we cannot tell what intersection sets the numbers refer to. Meanwhile, the sharp corners make the visualization less aesthetic than (c).

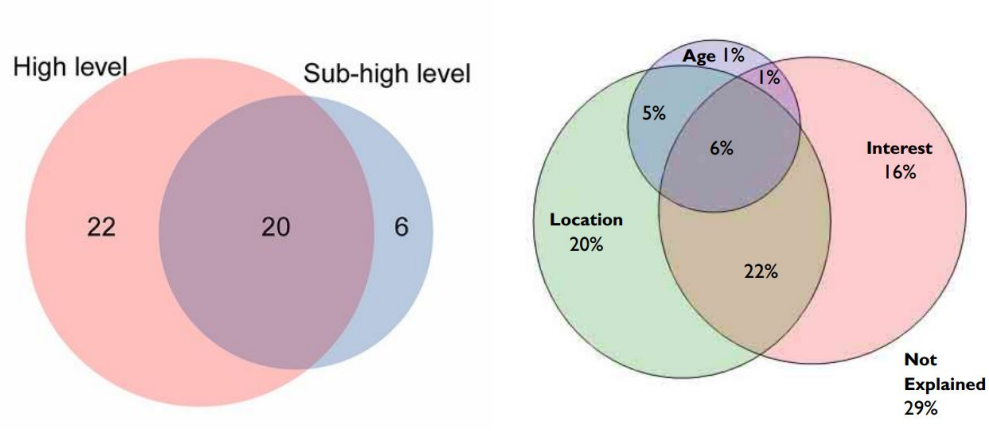


(c) (d)

Figure 1.5: (a) Venn diagram of gene sharing by six woody species [41]. (b) Showing the number of genes shared between isolates from investigative patients[7].

If we look at Venn diagrams (e) and (f) in Figure 1.5, both of them convey the size of intersections directly. In (e), It is clear that majority of "Sub-high level" gene ontologies are shared with "High level" ; only half of "High level" genes partake with the "Sub" ones [49]. In (f), based on the diagram, we can tell location and interest are the two main factors affecting friendships and age impacts just a little (13% in total) [25].

An informal survey of 112 Venn diagrams published in articles of journal *Nature* and *Genetics* in the past two years, review the following common features: (1) close to half of them (49/112) use size-proportional characteristics; (2) over two thirds of them (75/112) use Venn diagram circles and the number of circles is either two or three; (3) in these 75 articles which use circles, 39 contain the property of size-proportion; (4) the 37 articles do not use Venn diagram circles, and 24 have more than four closed curves. In other words,



(e) (f)

Figure 1.6: (e) Illustrating the overlapping of gene ontology between the highland and sub-highland lineages [49]. (f) Explaining the friendships through locations, ages and interests. (data source: www.livejournal.com) [25].

when the number of sets is smaller than four, almost all of them (75/88) make circular Venn diagrams. Besides, amongst 112 Venn diagrams, no more than six sets appear in any diagrams.

Chapter 2

Automated construction of Circular Venn diagrams

Given the increased use of Venn diagrams in the scientific and other literature, it would be of great value to have an automated way to construct these diagrams from data. Here we will consider how we might construct circular Venn diagrams whose visual areas are as nearly proportional to the size of the corresponding sets.

There are two characteristics that are available for us to manipulate: the size, or radius, of each circle and the location of its centre. Suppose we have sets $\mathbf{S} = \{S_1, \ldots, S_m\}$ and sizes $\mathbf{s} = \{s_1, s_2, \ldots, s_m\}$. And we consider the circle (or more generally ball) to be $\mathbf{B} = \{B_1, \ldots, B_m\}$ of area (volume) $\mathbf{b} = \{b_1, \ldots, b_m\}$. Similarly intersections and unions of sets $S_i \cap S_j$ have size s_{ij}.

One might proceed initially at least by choosing radius ρ_i such that $b_i \propto \rho_i^2$. If the distances d_{ij} between centres $\mathbf{c}_i$ and $\mathbf{c}_j$ were known, then we could locate the centres as follows:

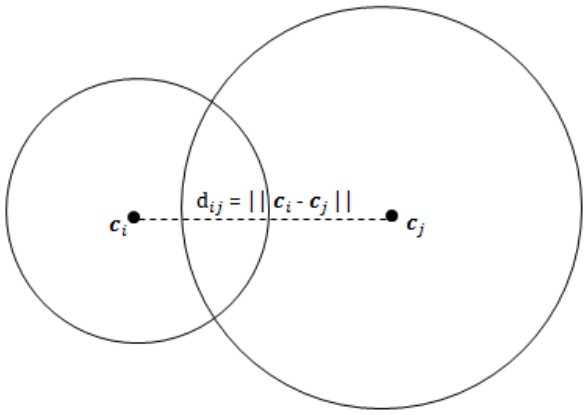


Figure 2.1: locate circles with target distance d_{ij}

An obvious choice for the distance is the Jaccard distance, as selected for example by [48]. Jaccard distance, also known as intersection over union, is used for comparing the distance over sample sets and can be defined as follows [23]:

$$d_{ij} = 1 - \frac{size(S_i \cap S_j)}{size(S_i \cup S_j)} = 1 - \frac{s_{ij}}{s_i + s_j - s_{ij}}$$

which captures the size of the intersection between the two sets.

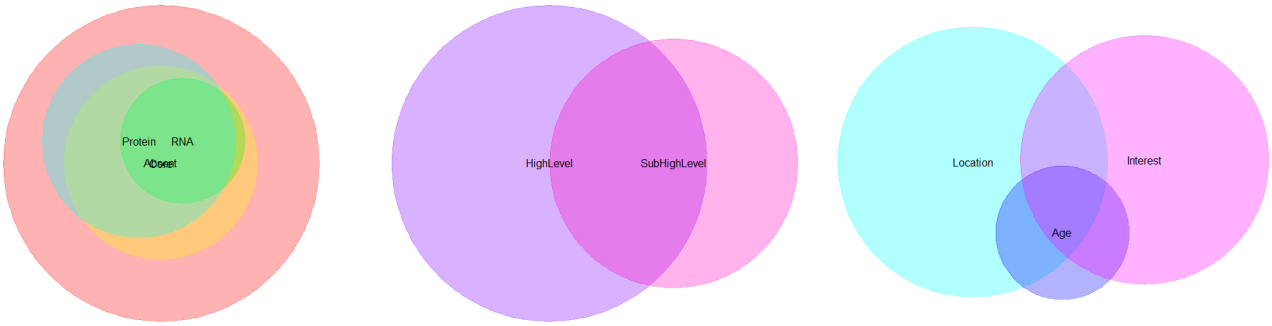
Squared distances $\mathbf{D} = [d_{ij}^2]$ could be used in the Gram matrix, then to the locations [38]

$$\mathbf{G} = (\mathbf{I} - \mathbf{H})\mathbf{C}\mathbf{C}^\mathsf{T}(\mathbf{I} - \mathbf{H}) = -\frac{1}{2}(\mathbf{I} - \mathbf{H})\mathbf{D}(\mathbf{I} - \mathbf{H})$$

where $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m]^\mathsf{T}$, $\mathbf{H} = \frac{1}{n}\mathbf{1}_m\mathbf{1}^\mathsf{T}{}_m$, $\mathbf{1}_m = [1, 1, \ldots, 1]^\mathsf{T}$, $\mathbf{G}$ is the central Gram matrix and $\mathbf{I}$ is the $m \times m$ identity matrix. Letting $\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$ be the eigen decomposition of the Gram matrix, we take $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$ as the initial point configuration. Figure 2.2 shows the initial location of data set in Figure 1.4 (a), Figure 1.6 (e) and (f). For any set $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$, the corresponding disjoint set

$$disjoint(\mathbf{S}) = \mathbf{S}^\star = \{S_1^\star, S_2^\star, \ldots, S_m^\star\}$$

where for all i the size of $S_i^\star$ is $s_i^\star$. In Figure 2.2 (a), circle *Absent*, circle *RNA* and circle *Protein* are totally inside circle *CoreCI*, however, if we look at the original data in Figure 1.4 (a), these three sets are not subsets of S_{CoreCI}. Comparing with Figure 1.6 (e),



(a) (e) (f)

Figure 2.2: The initial point configuration of data set Figure 1.4 (a), Figure 1.6 (e) and (f) by Jaccard distance

Figure 2.2 (e) fits well, except the right side loon of Sub-high level is almost close to the intersections, since the intersection loon should be three times larger than the right one. In Figure 2.2 (f), the disjoint size $s^{\star}_{Location,Interest}$ should earn 22/79 of the total size, which has the largest disjoint area, nevertheless $s^{\star}_{Location}$ and $s^{\star}_{AInterest}$ both are much larger than $s^{\star}_{Location,Interest}$. The initial point configurations of these three are mediocre and could be improved.

Wilkinson suggests an incremental method to improve this configuration by changing the locations of centres with radii stay fixed (`venneuler()`). $\boldsymbol{\mathcal{P}(S)}$ denotes power set excluding the null set.

$$\boldsymbol{\mathcal{P}(S)} = \{S_1, \ldots, S_m, S_{12}, \ldots, S_{12\cdots m}\}$$

Denote by $\mathbf{s}^{\star}_{\mathcal{P}}$ the vector containing the sizes of $\boldsymbol{\mathcal{P}(S)}^{\star}$. Similarly, define $\boldsymbol{\mathcal{P}(B)}$, $\boldsymbol{\mathcal{P}(B)}^{\star}$ and $\mathbf{b}^{\star}_{\mathcal{P}}$ for the corresponding area (volume) of balls.

Once the point configuration is given, disjoint area $\mathbf{b}^{\star}_{\mathcal{P}}$ can be approached. Wilkinson introduces a quick and efficient method to access the numerical actual disjoint area $\mathbf{b}^{\star}_{\mathcal{P}}$. Imagine there are m 100×100 bit-squares, one for each circle. In any square, a bit is 1 if the circle for that square covers it, and is zero if it does not. Location of the Venn diagram is the pixel-wise logical disjunction of all m squares, pixels in each disjoint region of the diagram are identified by a unique pattern of the m bits for that location [48].

For better scale, we can force $\sum_i^N s_i^{\star} = 1$ and $\sum_i^N b_i^{\star} = 1$. If fit perfectly, $\mathbf{b}^{\star}_{\mathcal{P}}$ should be equal to the corresponding sizes of the disjoint sets $\mathbf{s}^{\star}_{\mathcal{P}}$. The extent that this is not the case

is captured by fitting the linear model

$$\mathbf{b}^{\star}_{\mathcal{P}} = \mathbf{s}^{\star}_{\mathcal{P}}\beta + \mathbf{r} \tag{2.1}$$

to the given $\mathbf{b}^{\star}_{\mathcal{P}}$ and $\mathbf{s}^{\star}_{\mathcal{P}}$ with $\mathbf{r}$ as a residual vector (perfect fit denotes $\beta = 1$ and $\mathbf{r}$ is a zero vector). The least squares fitted value for β is $\widehat{\beta} = (\mathbf{s}^{\star\mathsf{T}}_{\mathcal{P}}\mathbf{s}^{\star}_{\mathcal{P}})^{-1}\mathbf{s}^{\star\mathsf{T}}_{\mathcal{P}}\mathbf{b}^{\star}_{\mathcal{P}}$ and the estimated residual sum of squares

$$\begin{aligned} RSS &= \widehat{\mathbf{r}}^{\mathsf{T}}\widehat{\mathbf{r}} = (\mathbf{b}^{\star}_{\mathcal{P}} - \mathbf{s}^{\star}_{\mathcal{P}}\widehat{\beta})^{\mathsf{T}}(\mathbf{b}^{\star}_{\mathcal{P}} - \mathbf{s}^{\star}_{\mathcal{P}}\widehat{\beta}) \\ TSS &= \mathbf{b}^{\star\mathsf{T}}_{\mathcal{P}}\mathbf{b}^{\star}_{\mathcal{P}} \end{aligned}$$

We can use $stress(\mathbf{b}^{\star}_{\mathcal{P}})$ as a measure of the quality of the fit, where

$$stress(\mathbf{b}^{\star}_{\mathcal{P}}) = \frac{RSS}{TSS} = \frac{(\mathbf{b}^{\star}_{\mathcal{P}} - \widehat{\mathbf{b}}^{\star}_{\mathcal{P}})^{\mathsf{T}}(\mathbf{b}^{\star}_{\mathcal{P}} - \widehat{\mathbf{b}}^{\star}_{\mathcal{P}})}{\mathbf{b}^{\star\mathsf{T}}_{\mathcal{P}}\mathbf{b}^{\star}_{\mathcal{P}}.} \tag{2.2}$$

The remaining task is to fix the radii and move centres to find a $\mathbf{b}^{\star}_{\mathcal{P}}$ which corresponds to the minimum *stress*. A descent step on each iteration for B_i is roughly proportional to:

$$\frac{\partial stress(\mathbf{b}^{\star}_{\mathcal{P}})}{\partial \mathbf{c}_i} \approx \sum_{k=1}^{N}\sum_{j\neq i}^{m}(\mathbf{c}_i - \mathbf{c}_j)\widehat{r}_k I_{ij}(k) = \sum_{j\neq i}^{m}(\mathbf{c}_i - \mathbf{c}_j)\widehat{\mathbf{r}}^{\mathsf{T}}\mathbf{I}_{ij} \tag{2.3}$$

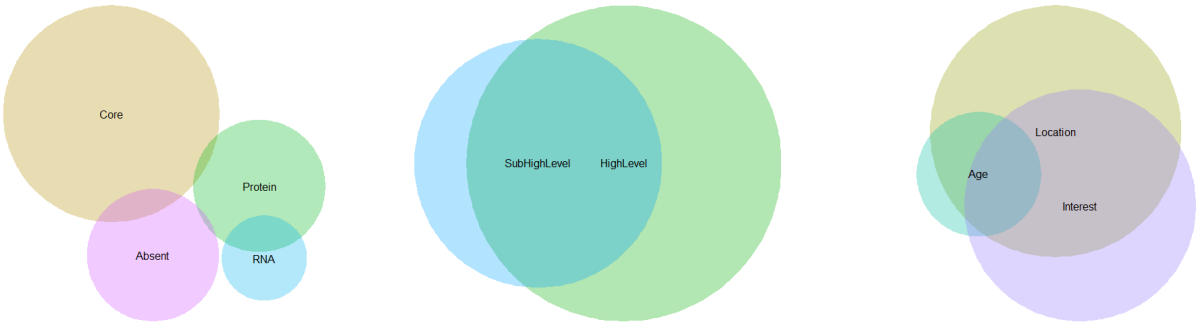
where, $\mathbf{I}_{ij}$ is a length N vector, $i, j \in \{u_1, u_2, \ldots, u_\ell\}$ and the kth element $I_{ij}(k)$ is the indicator function

$$I_{ij}(k) = \begin{cases} 1 & \text{if } \; s^{\star}_k = s^{\star}_{u_1 u_2 \cdots u_\ell} \text{ and } \; i, j \in \{u_1, u_2, \ldots, u_\ell\} \\ 0 & \text{otherwise} \end{cases}$$

And the centre can be updated as

$$\mathbf{c}_i^{(n+1)} = \mathbf{c}_i^{(n)} - \alpha\frac{\partial stress(\mathbf{b}^{\star}_{\mathcal{P}})}{\partial \mathbf{c}_i^{(n)}}. \tag{2.4}$$

where α is 0.01 and n is the count; If the residuals are very large, use a closer approximation to the gradient, computes stress four times (up, down, left, right) for each ball centre by taking small steps of 0.01. The gradient direction goes with the lowest stress values for $\mathbf{c_i}$. Figure 2.3 gives the final layout of these three.



(a) (e) (f)

Figure 2.3: The final point configuration of data set Figure 1.4 (a), Figure 1.6 (e) and (f), with stress, $0.00048, 6 \times 10^{-6}$ and 0.0029

Figure 2.3 (a) is a good fit, aside from the missing four way intersections. Figure (b) and (c) provide a very good fit, however, sometimes, Wilkinson's algorithm fails to handle some sets with complicated intersections. Here is an example, $\mathbf{S}_1 = \{S_1, S_2, \ldots, S_6\}$ and given input disjoint set $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_1^\star = \{S_1^\star, S_2^\star, S_3^\star, S_4^\star, S_5^\star, S_6^\star, S_{34}^\star, S_{35}^\star, S_{13}^\star, S_{14}^\star, S_{25}^\star, S_{15}^\star, S_{26}^\star\}$ with size

$$\mathbf{s}_{\mathcal{I}}^\star = [s_1^\star = 80, s_2^\star = 50, s_3^\star = 100, s_4^\star = 100, s_5^\star = 100, s_6^\star = 40,$$

$$s_{13}^\star = 30, s_{14}^\star = 30, s_{25}^\star = 30, s_{15}^\star = 40, s_{26}^\star = 10]$$

$\boldsymbol{\mathcal{P}}(\boldsymbol{S})_1^\star$ is the corresponding power set, any sets in $\boldsymbol{\mathcal{P}}(\boldsymbol{S})_1^\star \setminus \boldsymbol{\mathcal{I}}(\boldsymbol{S})_1^\star$ are $\varnothing$ and $\mathbf{s}_{\mathcal{P}}^\star = [\mathbf{s}_{\mathcal{I}}^\star, 0, \ldots, 0]$. Based on his algorithm, Figure 2.4 gives the final layout. S_3 and S_4 are totally overlaid, S_6 and S_2 are disjoint. The *stress* of it is 0.598, which denotes a poor fit.

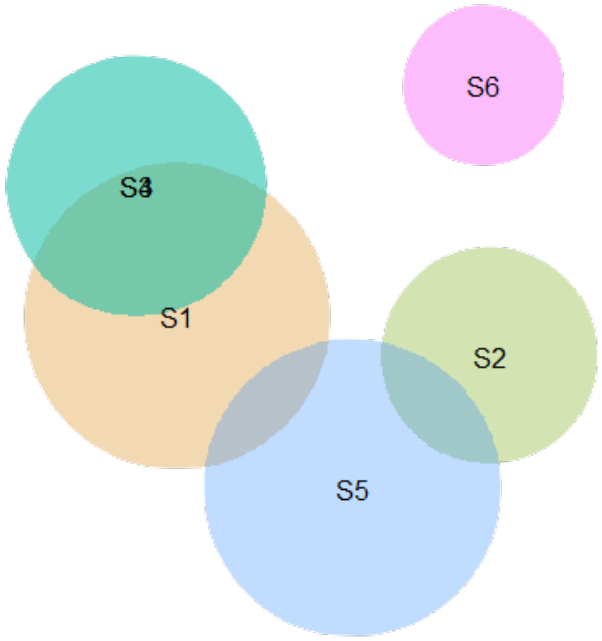


Figure 2.4: The final layout of disjoint set $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_1^\star$ by `venneuler()` of Wilkinson.

Frederickson noticed the failure of Wilkinson's algorithm `venneuler()` in some cases. Thus, he created a new algorithm called "Constrained Multiple Dimension Scaling" (or "Constrained MDS") by locating the sets by optimizing the distances between circles, instead of the intersection areas directly.

He starts with the geometric distance: if $S_i \cap S_j = \varnothing$, then for B_i and B_j, $d_{ij} \geq \rho_i + \rho_j$ and we choose to set $d_{ij} = \rho_i + \rho_j$; if $S_i \subset S_j$, then for B_i and B_j, $d_{ij} \leq \rho_j - \rho_i$ and we choose to set $d_{ij} = \rho_j - \rho_i$; if $S_i \cap S_j \neq \varnothing$, $S_i \not\subset S_j$, $S_j \not\subset S_i$, use $S_i \cap S_j$ to determine the d_{ij},

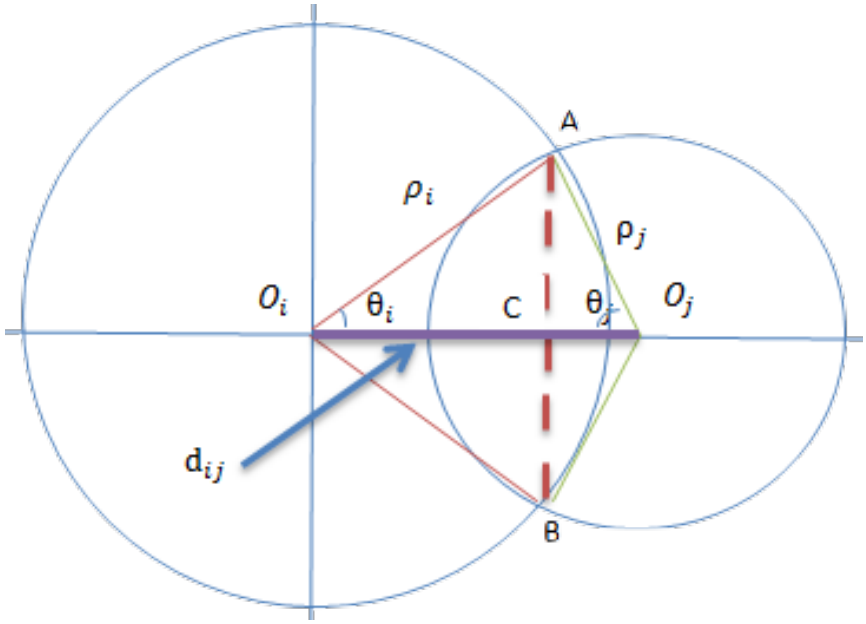


Figure 2.5: Two dimension circles

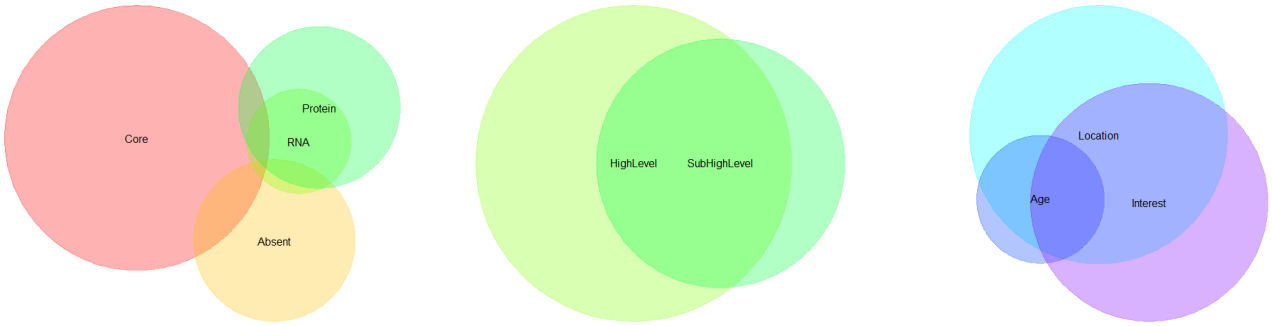
In Figure 2.5, O_i and O_j are the centres of two circles and d_{ij} is the distance between these two centres. A and B are the points of intersection. AB $\perp O_iO_j$ at point C. θ_i and θ_j are two angles of the triangle AO_iO_j. Thus, d_{ij} can be found by

$$d_{ij} = |O_iA|\cos(\theta_i) + |O_jA|\cos(\theta_j)$$

The remaining task is to find θ_i and θ_j. Firstly, $|AC| = |O_iA|\sin(\theta_i) = |O_jA| \times \sin(\theta_j)$. Secondly, area b_{ij} can be separated by line AB into two parts $Area(\widehat{AB}_{left})$ and $Area(\widehat{AB}_{right})$; $Area(\widehat{AB}_{left})$ equals to area of arc $O_j\widehat{AB}$ minus triangle O_jAB and $Area(\widehat{AB}_{right})$ equals to area of arc $O_i\widehat{AB}$ minus triangle O_iAB, where $|O_iA| = |O_iB| = \rho_i$, $|O_jA| = |O_jB| = \rho_j$. Hence, θ_i and θ_j can be found by solving the following equations:

$$\begin{aligned} 0 &= \theta_i\rho_i^2 - \rho_i^2\sin(\theta_i)\cos(\theta_i) + \theta_j\rho_j^2 - \rho_j^2\sin(\theta_j)\cos(\theta_j) - b_{ij} \\ 0 &= \rho_i\sin(\theta_i) - \rho_j\sin(\theta_j) \end{aligned}$$

Use Newton-Raphson to solve for $\widehat{\theta}_i$, $\widehat{\theta}_j$, and hence d_{ij}. Figure 2.6 illustrates the initial point configuration with geometric distance $[d_{ij}]$



(a) (e) (f)

Figure 2.6: The initial point configuration of data set Figure 1.4 (a), Figure 1.6 (e) and (f)

Comparing with Figure 2.2, the initial point configuration with geometric distance shows higher accuracy. However, in Figure 2.2 (a), $s^{\star}_{RNA}$ is almost missing; Figure 2.2 (e) is perfect; Figure 2.2 (f), $s^{\star}_{Age}$ is too large which should be the same with $s^{\star}_{Age,Interest}$.

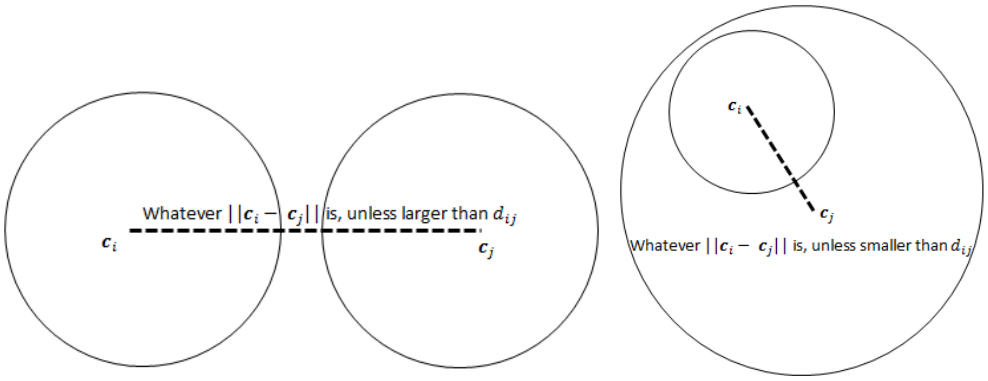
Frederickson improves the initial multidimensional scaling layout by noticing subsets and disjoint circles. The geometric distances $[d_{ij}]$ are fixed at their initial values, however determined from the sets S_i and S_j. This loss places a great deal of importance on the pairwise intersections between sets S_i and S_j and B_i and B_j. For example, when $S_i \cap S_j = \varnothing$ then, arguably, the circles should not intersect so placing them farther apart than the distance d_{ij} incurs no loss on the pairwise intersections, like Figure 2.7 (a). Similarly, if one of S_i or S_j is a subset of the other, then ideally the corresponding B_i and B_j should be entirely inside the other whatever the position of centres provided, like Figure 2.7 (b). Hence, he defines a loss function

$$L(\mathbf{C}) = \sum_{i=1}^{m} \ell(\mathbf{c}_i) \tag{2.5}$$

where for each i

$$\ell(\mathbf{c}_i) = \sum_{j=1}^{m} l(\mathbf{c}_i, \mathbf{c}_j). \tag{2.6}$$

Then update the configuration from its initial position by minimizing a suitably defined



(a) (b)

Figure 2.7: (a) If $S_i \cap S_j = \varnothing$, the way to position B_i and B_j. (b) If $S_i \cap S_j = S_i$ or $S_i \cap S_j = S_j$, the way to position B_i and B_j.

loss:

$$l(\mathbf{c}_i, \mathbf{c}_j) = \begin{cases} 0 & \text{when } S_i \cap S_j = \varnothing \text{ and } (\mathbf{c}_i - \mathbf{c}_j)^\mathsf{T}(\mathbf{c}_i - \mathbf{c}_j) \geq d_{ij}^2, \\ 0 & \text{when } S_i \subset S_j \text{ or } S_j \subset S_i \text{ and } (\mathbf{c}_i - \mathbf{c}_j)^\mathsf{T}(\mathbf{c}_i - \mathbf{c}_j) \leq d_{ij}^2, \\ ((\mathbf{c}_i - \mathbf{c}_j)^\mathsf{T}(\mathbf{c}_i - \mathbf{c}_j) - d_{ij}^2)^2 & \text{otherwise .} \end{cases} \tag{2.7}$$

The objective is to choose a configuration which minimizes this loss. To that end, he differentiates the loss with respect to $\mathbf{c}_i$ and solve. The derivative of $l(\mathbf{c}_i, \mathbf{c}_j)$ function with respect to $\mathbf{c}_i$ is

$$\frac{\partial l(\mathbf{c}_i, \mathbf{c}_j)}{\partial \mathbf{c}_i} = \begin{cases} \mathbf{0} & \text{when } S_i \cap S_j = \varnothing \text{ and } (\mathbf{c}_i - \mathbf{c}_j)^\mathsf{T}(\mathbf{c}_i - \mathbf{c}_j) \geq d_{ij}^2, \\ \mathbf{0} & \text{when } S_i \subset S_j \text{ or } S_j \subset S_i \text{ and } (\mathbf{c}_i - \mathbf{c}_j)^\mathsf{T}(\mathbf{c}_i - \mathbf{c}_j) \leq d_{ij}^2, \\ 4((\mathbf{c}_i - \mathbf{c}_j)^\mathsf{T}(\mathbf{c}_i - \mathbf{c}_j) - d_{ij}^2)(\mathbf{c}_i - \mathbf{c}_j) \text{ otherwise.} \end{cases} \tag{2.8}$$

Thus,

$$\frac{\partial \ell(\mathbf{c}_i)}{\partial \mathbf{c}_i} = \sum_j \frac{\partial l(\mathbf{c}_i, \mathbf{c}_j)}{\partial \mathbf{c}_i}. \tag{2.9}$$

This can be used in a nonlinear conjugate gradient method [40] to find a minimum $L(\mathbf{C})$ as follows.

1. *Initialization*:

   the initial configuration

$$\mathbf{C}^{(0)} \leftarrow [\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_m^{(0)}]^\mathsf{T}$$

   from the eigen decomposition of the gram matrix, the initial loss

$$L(\mathbf{C}^{(0)}) \leftarrow \sum_{i=1}^{m} \ell(\mathbf{c}_i^0)$$

   and the iteration count

$$n \leftarrow 0$$

2. *Outer loop over* n:

   (a) *Inner loop*: for $i = 1, \ldots, m$

   - Determine the conjugate direction $\mathbf{t}_i^{(n)}$:

$$\mathbf{t}_i^{(n)} \leftarrow \begin{cases} -\frac{\partial \ell(\mathbf{c}_i^{(n)})}{\partial \mathbf{c}_i^{(n)}} + \omega^{(n)} \mathbf{t}_i^{(n-1)} & n \geq 1 \\ -\frac{\partial \ell(\mathbf{c}_i^{(0)})}{\partial \mathbf{c}_i^{(0)}} & n = 0 \end{cases}$$

     where

$$\omega_{PR}^{(n)} \leftarrow \frac{\frac{\partial \ell(\mathbf{c}_i^{(n)})}{\partial \mathbf{c}_i^{(n)}}^\mathsf{T} \left(\frac{\partial \ell(\mathbf{c}_i^{(n)})}{\partial \mathbf{c}_i^{(n)}} - \frac{\partial \ell(\mathbf{c}_i^{(n-1)})}{\partial \mathbf{c}_i^{(n-1)}} \right)}{\frac{\partial \ell(\mathbf{c}_i^{(n-1)})}{\partial \mathbf{c}_i^{(n-1)}}^\mathsf{T} \frac{\partial \ell(\mathbf{c}_i^{(n-1)})}{\partial \mathbf{c}_i^{(n-1)}}}$$

     is the Polak-Ribiere choice[37] and

$$\omega^{(n)} \leftarrow \max\left(0, \omega_{PR}^{(n)}\right)$$

     where $\omega^{(n)}$ is a popular choice [40].

   - Perform a line search for

$$\alpha^{(n)} \leftarrow \arg\min_{\alpha} \quad \ell(\mathbf{c}_i^{(n)} + \alpha \mathbf{t}^{(n)})$$

   - Update the position $\mathbf{c}_i$:

$$\mathbf{c}_i^{(n+1)} \leftarrow \mathbf{c}_i^{(n)} + \alpha^{(n)} \mathbf{t}^{(n)}$$

- end *inner loop*

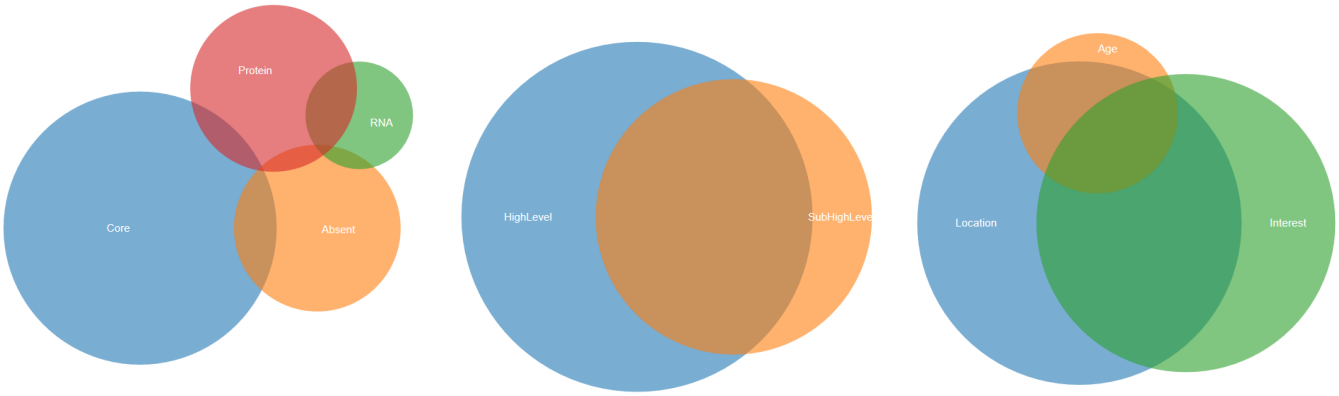
(b) Update outer loop:

$$n \leftarrow n + 1$$
$$L(\mathbf{C}^{(n)}) \leftarrow \sum_{i=1}^{m} \ell(\mathbf{c}_i^{(n)})$$

(c) *Outer loop* ends when $\left|L(\mathbf{C}^{(n)}) - L(\mathbf{C}^{(n-1)})\right| < \epsilon$.

3. Return point configuration $\mathbf{C} \leftarrow \mathbf{C}^{(n)}$

Figure 2.8 shows the final layout of data set in Figure 1.4 (a), Figure 1.6 (e) and (f).



(a) (e) (f)

Figure 2.8: The final point configuration

In Figure 2.8 (a) fits well, aside from the missing four way intersections. Figure 2.8 (e) is a perfect fit and $s^{\star}_{Age}$ in Figure 2.8 (f) seems still too large. These three are thus mostly perfect. Let us look at the fit of the artificial set $\boldsymbol{\mathcal{I}}(\boldsymbol{S})^{\star}_{1}$ (in Frederickson's `venn.js()`, the input combinations are $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_{1}$ instead of $\boldsymbol{\mathcal{I}}(\boldsymbol{S})^{\star}_{1}$, which are different from most programs. Thus, we should transform $\boldsymbol{\mathcal{I}}(\boldsymbol{S})^{\star}_{1}$ to $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_{1}$). It is basically a perfect fit.

Before `venneuler()`, `VennMaster()` [24] and `Chow/Rodgers` algorithm [10] are widely known generalized Venn programs. Wilkinson compares his function `venneuler()` to these two algorithms. In his comparison, `venneuler()` has several adventages: (1) the goodness

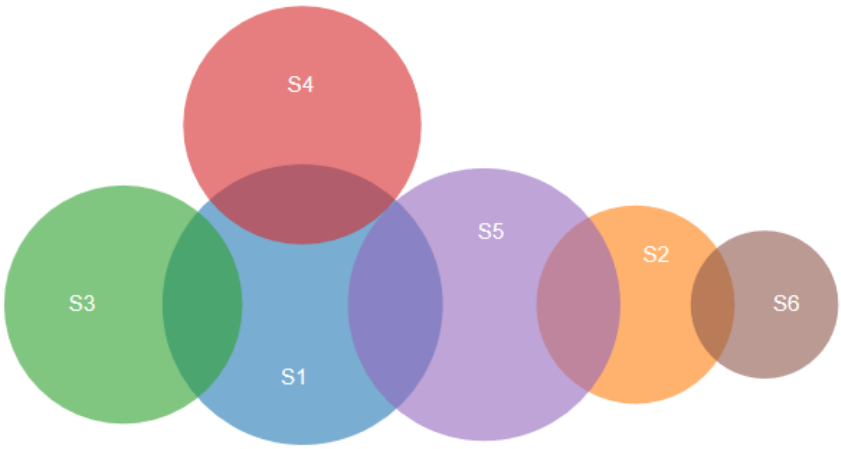
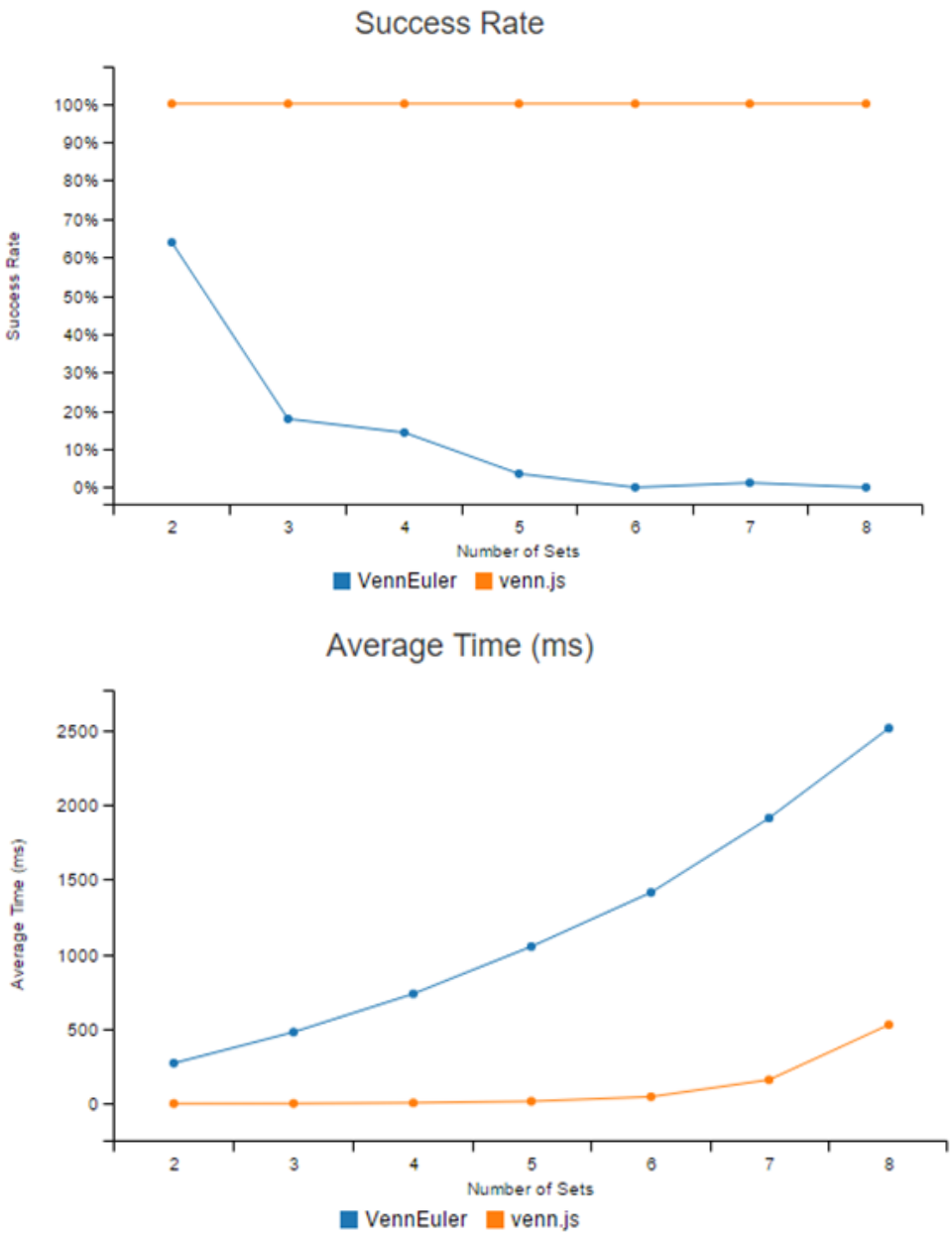


Figure 2.9: The layout of disjoint set $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_1^\star$ by `venn.js()` Frederickson

of fit *stress* is better than the other two. (2) `VennMaster()` relies on the random seed and the solutions give no indication of how well the fit is, which makes it not trustworthy; `Chow/Rodgers` algorithm is limited to 3-ring generalized Venn diagrams, so superset problems cannot be handled. Frederickson pointed out " while `venneuler()` frequently gets a solution that is close to being correct, it rarely gets a solution that is close enough for this test to say it succeeded" [19]. Hence he compares `vennplot()` and `venn.js()` with "success rate".

$$\psi = \max_i \frac{|b_i^\star - s_i^\star|}{s_i^\star}$$

If $\psi \leq 10\%$, he marks this test successful and set "success rate" to the number of successful test divided by the total number. In his comparison, `venn.js()` is far better than `vennplot()`, not only the "rate" but also the running time, with circles from two to eight in Figure 2.10. However, his test begins with possible configurations, real data also includes impossible configurations; like the data in Figure 1.4 (a) or Figure 1.6 (f).



(f)

Figure 2.10: Comparison of success rate and running time [19]

Chapter 3

Shrink or stretch

Note that for the loss function given in equation 2.7 no consideration is given to three and higher way intersections. A simple adjustment to the loss function is to replace the third condition (when neither this nor that holds) by

$$(\lambda(\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) - d_{ij}^2)^2$$

for some fixed value of $\lambda > 0$. Minimizing a loss with this value will still favour Euclidean squared distances between centres that are now **proportional** to the target squared distances d_{ij}^2. The geometric effect of λ is to move circles further apart or closer together.

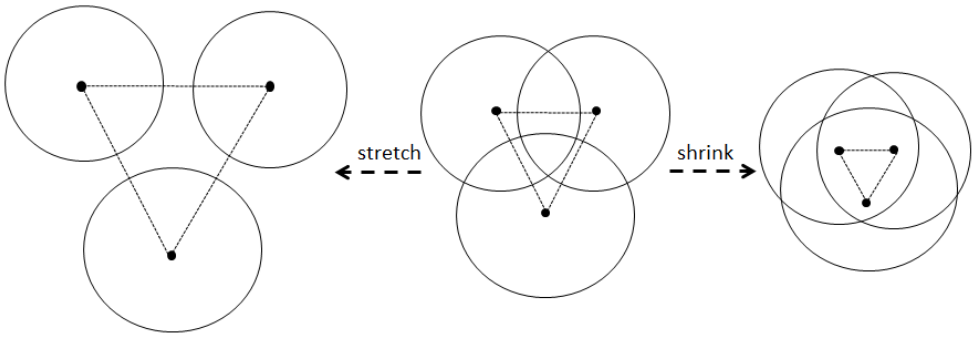


Figure 3.1: Stretch circles with $\lambda = 2$ or shrink circles with $\lambda = 0.5$.

Changes in λ have the effect of creating changes in all intersections. To choose λ we reintroduce the *stress* used by Wilkinson, back and forth, for fixed lambda.

Hence, the model can be defined as:

$$L(\mathbf{C}, \lambda) = \sum_{i=1}^{m} \ell(\mathbf{c_i}, \lambda)$$

For each i

$$\ell(\mathbf{c}_i) = \sum_{j=1}^{m} l(\mathbf{c}_i, \mathbf{c}_j, \lambda)$$

where

$$l(\mathbf{c}_i, \mathbf{c}_j, \lambda) = \begin{cases} 0 \quad \text{when} \;\; S_i \cap S_j = \varnothing \text{ and} \quad (\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) \geq d_{ij}^2, \\ 0 \quad \text{when} \;\; S_i \subset S_j \;\; \text{or} \;\; S_j \subset S_i \;\; \text{and} \quad (\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) \leq d_{ij}^2, \\ (\lambda(\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) - d_{ij}^2)^2 \;\; \text{otherwise.} \end{cases}$$

The derivative of $l(\mathbf{c}_i, \mathbf{c}_j, \lambda)$ with respect to $\mathbf{c}_i$ is:

$$\frac{\partial l(\mathbf{c}_i, \mathbf{c}_j, \lambda)}{\partial \mathbf{c}_i} = \begin{cases} \mathbf{0} \quad \text{when} \;\; S_i \cap S_j = \varnothing \text{ and} \quad (\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) \geq d_{ij}^2, \\ \mathbf{0} \quad \text{when} \;\; S_i \subset S_j \;\; \text{or} \;\; S_j \subset S_i \;\; \text{and} \quad (\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) \leq d_{ij}^2, \\ 4\lambda(\lambda(\mathbf{c}_i - \mathbf{c}_j)^{\mathsf{T}}(\mathbf{c}_i - \mathbf{c}_j) - d_{ij}^2)(\mathbf{c}_i - \mathbf{c}_j) \text{ otherwise.} \end{cases}$$

Thus,

$$\frac{\partial \ell(\mathbf{c}_i, \lambda)}{\partial \mathbf{c}_i} = \sum_j \frac{\partial l(\mathbf{c}_i, \mathbf{c}_j, \lambda)}{\partial \mathbf{c}_i}$$

where $\lambda \in R$. Use a nonlinear conjugate gradient method to find the minimum of $L(\mathbf{C}, \lambda)$. In this way, we can shrink or stretch our layout and obtain the centres $\mathbf{C}$ ($\mathbf{C}$ is determined by λ).

To determine an appropriate value of λ, following Wilkinson, a $stress(\lambda)$ could be measured for the quality of the fit of areas. For any configuration, given λ, the vector of sizes for the disjoint balls will be $\mathbf{b}_{\mathcal{P}}^{\star}(\lambda)$ and

$$\mathbf{b}_{\mathcal{P}}^{\star}(\lambda) = \mathbf{s}_{\mathcal{P}}^{\star}\beta + \mathbf{r}.$$

With this formulation, we can even give different weights to each pair of disjoint balls so that the quality of fit on higher intersections can be captured. Hence, the linear model can be defined as:

$$\mathbf{W}^{\frac{1}{2}}\mathbf{b}_{\mathcal{P}}^{\star}(\lambda) = \mathbf{W}^{\frac{1}{2}}\mathbf{s}_{\mathcal{P}}^{\star}\beta + \mathbf{W}^{\frac{1}{2}}\mathbf{r},$$

where $\mathbf{W} = diag(w_1, w_2, ..., w_N)$ is a $N \times N$ diagonal matrix. The weighted least squares fitted value for β is $\widehat{\beta} = (\mathbf{s}_{\mathcal{P}}^{\star\mathsf{T}}\mathbf{W}\mathbf{s}_{\mathcal{P}}^{\star})^{-1}\mathbf{s}_{\mathcal{P}}^{\star\mathsf{T}}\mathbf{W}\mathbf{b}_{\mathcal{P}}^{\star}(\lambda)$ and the estimated residual sum of squares

$$\begin{aligned} RSS(\beta, \lambda) &= \widehat{\mathbf{r}}^{\mathsf{T}}\mathbf{W}\widehat{\mathbf{r}} = (\mathbf{b}_{\mathcal{P}}^{\star}(\lambda) - \mathbf{s}_{\mathcal{P}}^{\star}\widehat{\beta})^{\mathsf{T}}\mathbf{W}(\mathbf{b}_{\mathcal{P}}^{\star}(\lambda) - \mathbf{s}_{\mathcal{P}}^{\star}\widehat{\beta}) \\ TSS &= \mathbf{b}_{\mathcal{P}}^{\star}(\lambda)^{\mathsf{T}}\mathbf{b}_{\mathcal{P}}^{\star}(\lambda) \end{aligned}$$

We can use $stress(\lambda)$ as a measure of the quality of the fit, where:

$$stress(\lambda) = \frac{RSS(\beta, \lambda)}{TSS}$$

and

$$stress = \arg\min_{\lambda} stress(\lambda)$$

An algorithm to find centre $\mathbf{C}$ and corresponding stress can be given as follows:

1. *Initialization*:
   the initial point configuration

   $$\mathbf{C}^{(0)} \leftarrow [\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_m^{(0)}]^{\mathsf{T}}$$

   the initial λ

   $$\lambda^{(1)} \leftarrow 1$$

   and the initial count

   $$n \leftarrow 1$$

2. Fixing $\lambda^{(n)}$ and computing $stress(\lambda^{(n)})$

   - Minimizing $L(\mathbf{C}; \lambda^{(n)})$ and get $\mathbf{C}^{(n)}$

     $$\mathbf{C}^{(n)} \leftarrow \arg\min_{\mathbf{C}} \; L(\mathbf{C}; \lambda^{(n)})$$

   - Compute each area $\mathbf{b}_{\mathcal{P}}^{\star}(\lambda^{(n)})$ and find $\widehat{\beta}^{(n)}$

     $$\widehat{\beta}^{(n)} \leftarrow \arg\min_{\beta} RSS(\beta; \lambda^{(n)})$$

   $\widehat{\beta}^{(n)}$ can be solved as $(\mathbf{s}_{\mathcal{P}}^{\star\mathsf{T}}\mathbf{W}\mathbf{s}_{\mathcal{P}}^{\star})^{-1}\mathbf{s}_{\mathcal{P}}^{\star\mathsf{T}}\mathbf{W}\mathbf{b}_{\mathcal{P}}^{\star}(\lambda^{(n)})$

- Finding $stress(\lambda^{(n)})$

$$stress(\lambda^{(n)}) \leftarrow \frac{RSS(\widehat{\beta}^{(n)}, \lambda^{(n)})}{\mathbf{b}^{\star\mathsf{T}}_{\mathcal{P}}(\lambda^{(n)})\mathbf{W}\mathbf{b}^{\star}_{\mathcal{P}}(\lambda^{(n)})}$$

3. Update $\lambda^{(n)}$, back and forth, until $\left|stress(\lambda^{(n)}) - stress(\lambda^{(n-1)})\right| \leq \epsilon$, return $\mathbf{C}$ and $stress$

For step 3, updating λ, we suggest *Nelder Mead Algorithm* [31]. Set:

$$\lambda_{+}^{(n)} \leftarrow \lambda^{(n)} + \Delta$$

$$\lambda_{-}^{(n)} \leftarrow \lambda^{(n)} - \Delta$$

where $\Delta \in \Re^{+}$, a small step size. Fixing $\lambda_{+}^{(n)}$ and $\lambda_{-}^{(n)}$ to compute $stress(\lambda_{+}^{(n)})$ and $stress(\lambda_{-}^{(n)})$.

1. *Loop over* n:

   (a) *Previous preparation*:
   the initial $\boldsymbol{\lambda}^{(n)}$ vector

   $$\boldsymbol{\lambda}^{(n)} \leftarrow [\lambda^{(n)}, \lambda_{+}^{(n)}, \lambda_{-}^{(n)}]$$

   the corresponding $\mathbf{stress}^{(n)}$

   $$\mathbf{stress}^{(n)} \leftarrow [stress(\lambda^{(n)}), stress(\lambda_{+}^{(n)}), stress(\lambda_{-}^{(n)})]$$

   and assuming:

   $$stress(\lambda_1) \leq stress(\lambda_2) \leq stress(\lambda_3)$$

   where, $\lambda_i \in \boldsymbol{\lambda}^{(n)}$, $stress(\lambda_i) \in \mathbf{stress}^{(n)}$ and $i = \{1, 2, 3\}$. λ_0 can be computed as:

   $$\lambda_0 \leftarrow \frac{\lambda_1 + \lambda_2}{2}$$

   λ_r is:

   $$\lambda_r \leftarrow 2\lambda_0 - \lambda_3$$

   (b) Update *Loop*
   i). the $\boldsymbol{\lambda}$:

- **if**$(stress(\lambda_1) \leq stress(\lambda_r) < stress(\lambda_2))$ **then** do *reflection*

$$\boldsymbol{\lambda}^{(n+1)} \leftarrow [\lambda_1, \lambda_2, \lambda_r]$$

- **else if**$(stress(\lambda_r) < stress(\lambda_1))$ **then** do *expansion*

$$\lambda_e \leftarrow 2\lambda_r - \lambda_0$$

and get corresponding $stress(\lambda_e)$

  - **if** $(stress(\lambda_e) < stress(\lambda_r))$ **then**

$$\boldsymbol{\lambda}^{(n+1)} \leftarrow [\lambda_1, \lambda_2, \lambda_e]$$

  - **else**

$$\boldsymbol{\lambda}^{(n+1)} \leftarrow [\lambda_1, \lambda_2, \lambda_r]$$

- **else if**$(stress(\lambda_r) \geq stress(\lambda_2))$ **then** do *contraction*:

$$\lambda_c \leftarrow \frac{\lambda_0 + \lambda_3}{2}$$

  - **if**$(stress(\lambda_c) \leq stress(\lambda_3))$ **then**

$$\boldsymbol{\lambda}^{(n+1)} \leftarrow [\lambda_1, \lambda_2, \lambda_c]$$

ii). the count:

$$n \leftarrow n + 1$$

(c) *Shrink* $\boldsymbol{\lambda}$:
With $\boldsymbol{\lambda}^{(n)}$, we can get corresponding $\mathbf{stress}^{(n)}$, and assuming:

$$stress(\lambda_1) \leq stress(\lambda_2) \leq stress(\lambda_3)$$

where, $\lambda_i \in \boldsymbol{\lambda}^{(n)}$, $stress(\lambda_i) \in \mathbf{stress}^{(n)}$ and $i = \{1, 2, 3\}$.
Shrink λ_j, where $j = \{2, 3\}$

$$\lambda_j \leftarrow \frac{\lambda_j + \lambda_1}{2}$$

(d) *Loop* ends when

$$\left|\max(\boldsymbol{\lambda}^{(n)}) - \min(\boldsymbol{\lambda}^{(n)})\right| \leq \epsilon$$

or

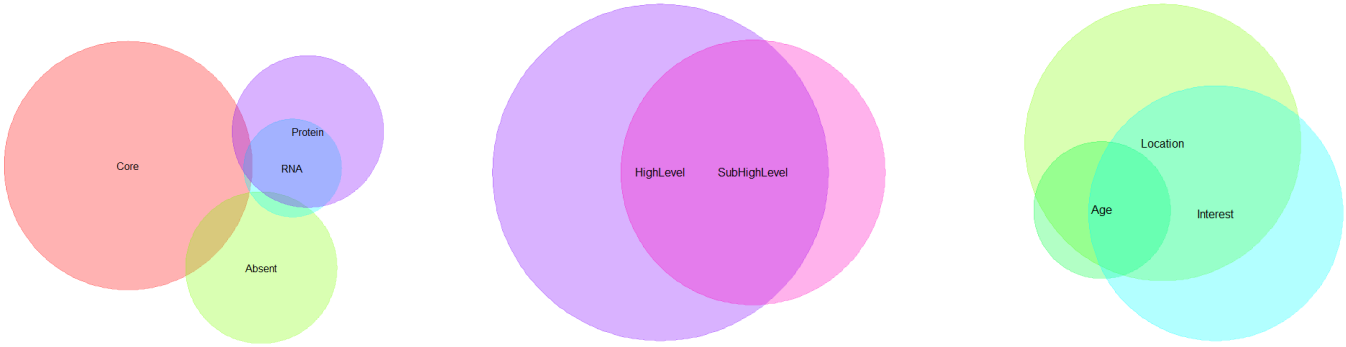
$$\left|\max(\mathbf{stress}^{(n)}) - \min(\mathbf{stress}^{(n)})\right| \leq \epsilon$$

(e) $stress = \min(\mathbf{stress}^{(n)})$

.

We have one more method for finding λ, see the appendix and multiple methods implemented in `vennplot()`.

Figure 3.2 shows the layout of data sets as in Figure 1.4 (a), Figure 1.6 (e) and (f): comparing `venneuler()` and `venn.js()`. Figure 3.2 (a) fails to capture the global struc-



(a) (e) (f)

Figure 3.2: (a), (e) and (f) corresponds to Figure 1.4 (a), Figure 1.6 (e) and (f)

ture; (e) is almost perfect; (f) gives too large area to $s^{\star}_{Age}$, the stress of these three are 0.017, 1.46×10^{-6} and 0.0028. Let us look at the fit of the artificial set $\boldsymbol{\mathcal{I}}(\boldsymbol{S})^{\star}_{1}$, see Figure 3.3

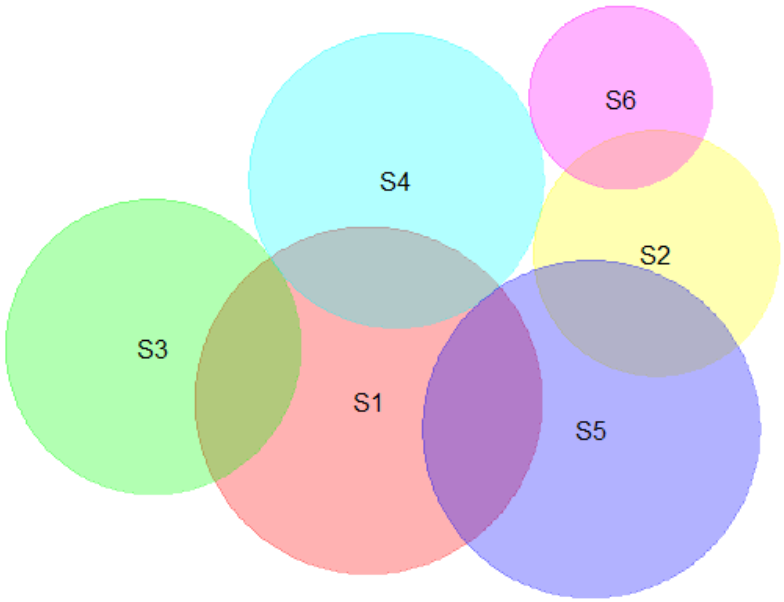


Figure 3.3: The layout of disjoint set $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_1^\star$ by `vennplot()`

$stress(\mathbf{C}, \lambda)$ of this layout is close to 0.00016 and $L(\mathbf{C}, \lambda)$ is 9.6×10^{-9}. It is a good fit, since all the intersections can match $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_1^\star$.

Figure 3.4 shows scatter plots for these four data sets, the x-label is $\mathbf{s}^\star / \sum_i^N \mathbf{s}^\star$ and y-label is $\mathbf{s}^\star / \sum_i^N \mathbf{s}^\star - \mathbf{b}^\star / \sum_i^N \mathbf{b}^\star$. In the scatter plot, for "Set 1", blue dots perform worse than the other two; for "Set 2", blue dots and purple dots are coincident and better than the red ones; for "Set 3", all of them are very similar; for "Set 4", blue and purple ones are coincidence and better than the red ones. And we can also compute ψ for each data set: for `venneuler()`, $\boldsymbol{\psi}_{ve} = [1, 0.0148, 1.012, 1]$ (excluding ∞ value), $\boldsymbol{\psi}_{vj} = [0.53, 0.0017, 0.86, 0.0262]$, for `vennplot()`, $\boldsymbol{\psi}_{vp} = [2.16, 0.0017, 0.809, 0.0262]$.

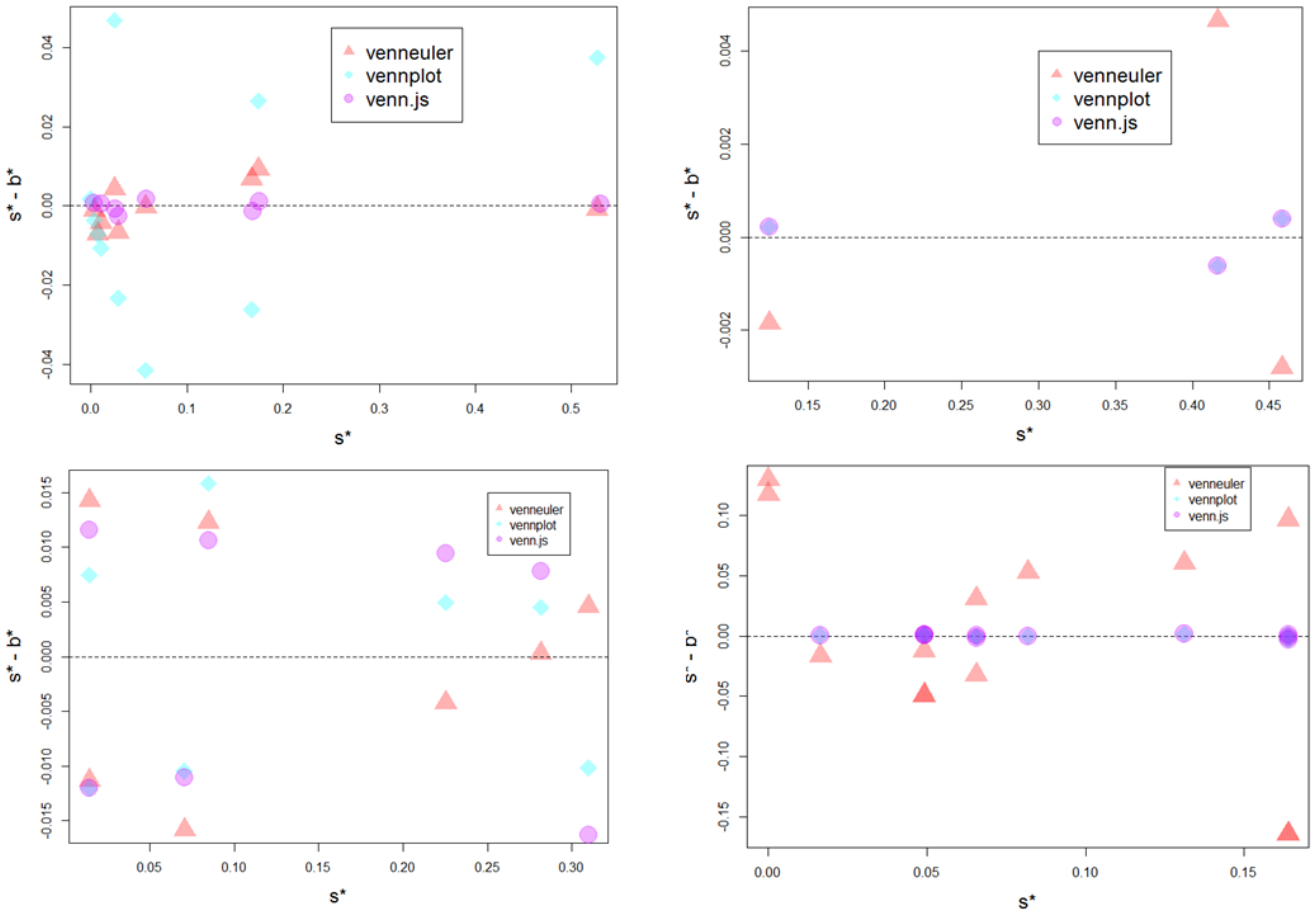


Figure 3.4: Scatter plot. Red dots represent `venneuler()`, blue dots represent `vennplot()` and purple dots represent `vennjs()`; imaginary line represents $y = 0$. Four sets are on behalf of Figure 1.4 (a), Figure 1.6 (e), (f) and $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_1^\star$

Going back to data set 3.2 (a), none of these three algorithms could capture the four way intersections. The main problem is the data set and dimension, the steepest descent fails to reach a minimum or the minimum only provides a mediocre result. Hence, we could increase 2 dimensions to 3 dimensions to illustrate Venn diagrams.

Chapter 4

Three dimension Venn diagram

Venn diagrams can also be illustrated in three dimensions. Let $p \in \{2, 3\}$ be the dimensionality of the Venn representation. If $p = 3$, balls B are spheres; $b = size(B)$ denotes their volume; $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m]^\mathsf{T}$ is the $m \times 3$ matrix of ball centres; ρ_i is the radius of the ball i, $\rho_i \propto b_i^{\frac{1}{3}}$.

The geometric distance $[d_{ij}]$ can be accessed as follows: It is similar to $p = 2$. In Figure

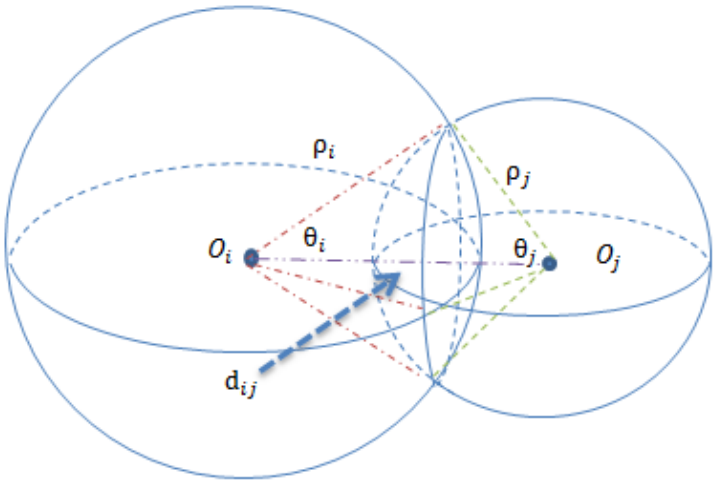


Figure 4.1: Dimension $p = 3$

4.1, O_i and O_j are the centres of these two spheres. A and B are the points of intersection and line AB is the diameter of the intersect plane, so AB $\perp O_iO_j$ at point C. θ_i and θ_j are two angles of the triangle AO_iO_j. Thus, d_{ij} can be found by

$$d_{ij} = |O_iA|\cos(\theta_i) + |O_jA|\cos(\theta_j).$$

The remaining task is to find θ_i and θ_j. Firstly, $|AC| = |O_iA|\sin(\theta_i) = |O_jA|\sin(\theta_j)$. Secondly, volume b_{ij} can be separated by a circle plane, with centre C and radius $|AC|$ ($|BC|$), into two parts, $SphereCap_{left}$ and $SphereCap_{right}$:

$$\begin{aligned} SphereCap_{left} &= \tfrac{\pi(|O_jA|-|O_jC|)}{6}(3\,|AC|^2 + (|O_jA| - |O_jC|)^2) \\ SphereCap_{right} &= \tfrac{\pi(|O_iA|-|O_iC|)}{6}(3\,|AC|^2 + (|O_iA| - |O_iC|)^2) \end{aligned}$$

where $|O_iA| = |O_iB| = \rho_i$, $|O_jA| = |O_jB| = \rho_j$, $|AC| = |BC| = \rho_i\sin(\theta_i)$, $|O_iC| = \rho_i\cos(\theta_i)$ and $|O_jC| = \rho_j\cos(\theta_j)$; $SphereCap_{left}$ and $SphereCap_{right}$ can be expressed as

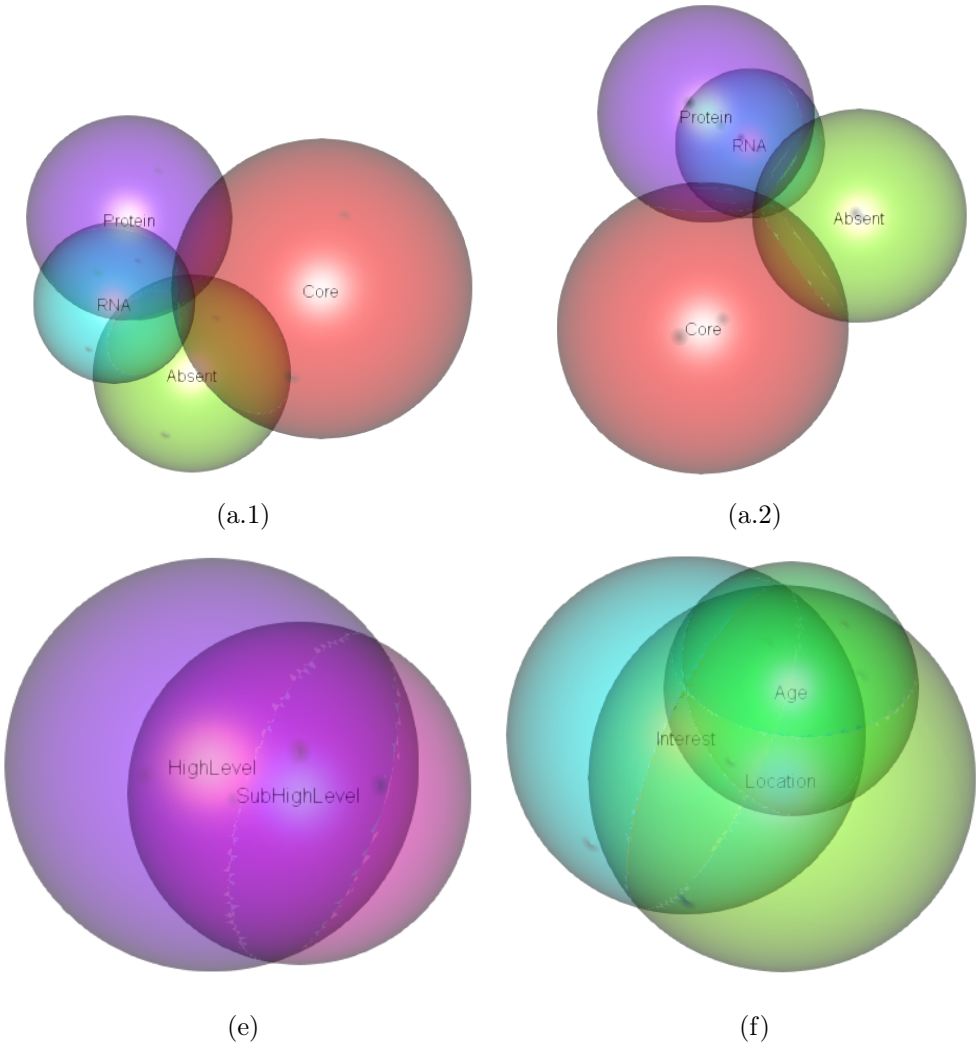
$$\begin{aligned} SphereCap_{left} &= \tfrac{\pi(\rho_j-\rho_j\cos(\theta_j))}{6}(3\rho_j\sin(\theta_j)^2 + (\rho_j - \rho_j\cos(\theta_j))^2), \\ SphereCap_{right} &= \tfrac{\pi(\rho_i-\rho_i\cos(\theta_i))}{6}(3\rho_i\sin(\theta_i)^2 + (\rho_i - \rho_i\cos(\theta_i))^2). \end{aligned}$$

Then, we can add them up to get b_{ij}; after simplifying, θ_i and θ_j can be found by solving the following equations:

$$\begin{aligned} 0 &= \tfrac{\pi}{3}\rho_i^3(1 - cos(\theta_i))^2(2 + cos(\theta_i)) \\ &\quad + \tfrac{\pi}{3}\rho_j^3(1 - cos(\theta_j))^2(2 + cos(\theta_j)) \;-\; b_{ij} \\ 0 &= \rho_i sin(\theta_i) - \rho_j sin(\theta_j) \end{aligned}$$

The Newton-Raphson method can be applied for $\widehat{\theta}_i$, $\widehat{\theta}_j$, and hence d_{ij}.

If the Venn diagrams are illustrated as three dimensional balls, we can also use Wilkinson's method [48] to compute volumes but expand a 100 × 100 bit-squares plane to a 100 × 100 × 100 bit box. Each "pixel" has either value 1 or 0. Then go through all the pixels and sum them up to get each part's volume. Figure 4.2 shows the three-dimensional layout.



(a.1) (a.2)

(e) (f)

Figure 4.2: (a.1) and (a.2) are the same layout but observed by different angles of data set Figure 1.4 (a); (e) and (f) are the corresponding 3D layout of Figure 1.6 (e) and (f)

We can find that the 3D fit is better. Especially for the four circle one, the four way intersection can be observed. Most of the time, three dimensional Venn diagrams could fit better: first, when we choose the Gram matrix as initial configuration, we could extract the first three columns of $\mathbf{\Lambda}^{\frac{1}{2}}$; second, when we minimize $L(\mathbf{C}, \lambda)$, the gradient have two radians to move, thus, circles have more directions to move and they are more likely to reach the minimum.

Chapter 5

Unspecified intersections

In this section, we will discuss a "special case": some intersections are unspecified, but their higher order intersections exist. Furthermore, we are just interested in the fit of the input set $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})$, instead of $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})$.

Suppose we have a set $\mathbf{S} = \{S_1, S_2, S_3\}$, the input given disjoint subsets $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})^\star_2 = \{S^\star_1, S^\star_2, S^\star_3, S^\star_{123}\}$ with size

$$\mathbf{s}^\star_{\mathcal{I}} = [s^\star_1 = 20, s^\star_2 = 20, s^\star_3 = 20, s^\star_{123} = 1]$$

5.0.1 Common case

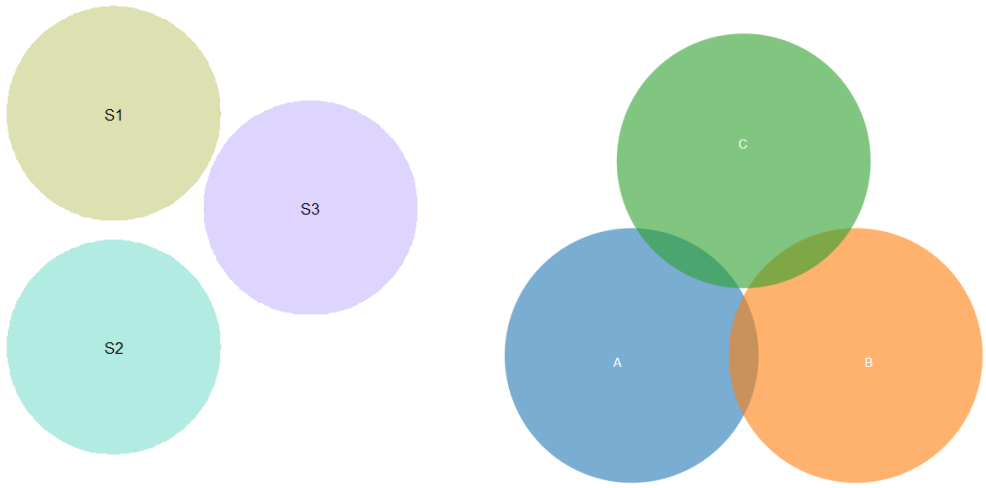
In a "common" case, any sets in $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})_2 \setminus \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_2$ are $\varnothing$. We have

$$\mathbf{s}_{\mathcal{I}} = [s_1 = 21, s_2 = 21, s_3 = 21 s_{12} = 1, s_{13} = 1, s_{23} = 1, s_{123} = 1].$$

Figure 5.1 shows the layout of `venneuler()` and `venn.js()` with input $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})^\star_2$. Neither of these two diagrams show the three way intersection.

5.0.2 Special case

The highest order of $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})^\star_2$ is three and all the two way disjoint intersections are unspecified. If we simply set all sets in the complement set of $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})^\star_2$ to $\varnothing$, it will be hard to



(a) (b)

Figure 5.1: (a) `venneuler()` (b) `venn.js()`

capture the three way intersection. Thus, from $\boldsymbol{\mathcal{I}}(\boldsymbol{S})^{\star}_{2}$, assume all unspecified are $\varnothing$ to construct $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_{2}$. For example,

$$\mathbf{s}^{\star}_{\mathcal{I}} = [s^{\star}_{1} = 20,\ s^{\star}_{2} = 20,\ s^{\star}_{3} = 20,\ s^{\star}_{123} = 1]$$

$$\rightarrow \mathbf{s}_{\mathcal{I}} = [s_{1} = 21, s_{2} = 21, s_{3} = 21, s_{123} = 1]$$

Then, assume all unspecified $s_{ij\ldots\ell}$ are to be estimated and constrain unspecified $k-$way intersections. E.g. for two way intersections:

$$\begin{cases} s_{123} \leq \widehat{s}_{12} \leq min(s_{1}, s_{2}), \\ s_{123} \leq \widehat{s}_{13} \leq min(s_{1}, s_{3}), \\ s_{123} \leq \widehat{s}_{23} \leq min(s_{2}, s_{3}), \end{cases}$$

A simple solution can be had by noticing that the following would hold in general:

$$s_{12} \propto s_{123}$$

$$s_{123} \propto s_{1234}$$

and so on, with constant of proportionality ≥ 1 in each case. Hence, $\widehat{s}_{12} = \mu_1 s_{123}$, $\widehat{s}_{13} = \mu_2 s_{123}$ and $\widehat{s}_{23} = \mu_3 s_{123}$ and the constraints are

$$\begin{cases} \mu_1 \leq \frac{min(s_1,s_2)}{s_{123}}, \\ \mu_2 \leq \frac{min(s_1,s_3)}{s_{123}}, \\ \mu_3 \leq \frac{min(s_2,s_3)}{s_{123}}, \\ \mu_i \geq 1 \quad i = \{1,2,3\}. \end{cases}$$

If the order increases, the number of μ will increase exponentially. For simplicity, we choose to set $\mu = \mu_1 = \mu_2 = \mu_3$. In this way, we can shorten three parameters to just one and we have

$$\widehat{s}_{12} = \widehat{s}_{13} = \widehat{s}_{23} = \mu s_{123}.$$

where $1 \leq \mu \leq \frac{min(s_1,s_2,s_3)}{s_{123}}$. Since d_{ij} is a function of ρ_i, ρ_j, s_{ij} or $\widehat{s}_{ij}$, and we can use $[\widehat{d}_{ij}]$ to represent geometric distance matrix, where $[\widehat{d}_{ij}]$ is determined by μ.

We need to mention that if any two way disjoint intersections are given, for example, $s^\star_{12} = 2$ and $s_{12} = 2 + 1 = 3$, then s_{12} is determined and will not be generated.

Model redefinition

The model is the same as before but we replace $[d_{ij}]$ to by $[\widehat{d}_{ij}]$. Hence, we add a parameter μ to our model.

$$L(\mathbf{C}, \lambda; \mu) = \sum_{i=1}^{m} \ell(\mathbf{c_i}, \lambda; \mu) = \sum_{i=1}^{m} \sum_{j=1}^{m} l(\mathbf{c}_i, \mathbf{c}_j, \lambda; \mu).$$

Minimize $L(\mathbf{C}, \lambda; \mu)$ to get centre $\mathbf{C}$ and compute area $\mathbf{b}^\star_{\mathcal{I}}$. Then,

$$\mathbf{W}^{\frac{1}{2}} \mathbf{b}^\star_{\mathcal{I}}(\lambda) = \mathbf{W}^{\frac{1}{2}} \mathbf{s}^\star_{\mathcal{I}} \beta + \mathbf{W}^{\frac{1}{2}} \mathbf{r}.$$

Weighted RSS and $stress$ can be expressed as

$$RSS(\beta, \lambda; \mu) = (\mathbf{b}^\star_{\mathcal{I}}(\lambda) - \beta \mathbf{s}^\star_{\mathcal{I}})^{\mathsf{T}} \mathbf{W} (\mathbf{b}^\star_{\mathcal{I}}(\lambda) - \beta \mathbf{s}^\star_{\mathcal{I}}),$$

$$stress(\lambda; \mu) = \frac{RSS(\beta, \lambda; \mu)}{\mathbf{b}^\star_{\mathcal{I}}(\lambda)^{\mathsf{T}} \mathbf{W} \mathbf{b}^\star_{\mathcal{I}}(\lambda)}.$$

Here, we have two parameters μ and λ. We will start with parameter μ, then λ, because it is meaningless to shrink or expand with a poor geometric distance.

Due to $\mu \geq 1$, one can apply the following algorithm for finding a good centre $\mathbf{C}$:

1. *Initialization*:
   the initial point configuration
   $$\mathbf{C}^{(0)} \leftarrow [\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_m^{(0)}]^{\mathsf{T}}$$
   the initial μ
   $$\mu^{(1)} \leftarrow 1 \text{ and } \mu^{(1)} \Rightarrow [\widehat{d}_{ij}^{(1)}]$$
   the initial λ
   $$\lambda^{(1)} \leftarrow 1$$
   and the initial count
   $$n \leftarrow 1$$
2. Fix $\lambda^{(1)}$, update $\mu^{(n)}$ until *stress* reaches minimum, then return
   $$[\widehat{d}_{ij}] = [\widehat{d}_{ij}^{(n)}]$$
3. Fix $[\widehat{d}_{ij}]$ and shrinking or stretching λ to find the minimum *stress*, then return $\mathbf{C}$.

For step 2, we recommend *Line Search* [6]:

- *Loop over* n:
  1. Fix $\lambda^{(1)}$ and $\mu^{(n)}$
     $$\mathbf{C}^{(n)} \leftarrow \arg\min_{\mathbf{C}} L(\mathbf{C}; \lambda^{(1)}, \mu^{(n)})$$
  2. Compute area $\mathbf{b}_{\mathcal{I}}^{\star}(\lambda^{(1)}, \mu^{(n)})$ and
     $$\widehat{\beta}^{(n)} \leftarrow \arg\min_{\beta} RSS(\beta; \lambda^{(1)}, \mu^{(n)})$$
     get corresponding
     $$stress(\lambda^{(1)}; \mu^{(n)}) \leftarrow \frac{RSS(\widehat{\beta}^{(n)}, \lambda^{(1)}, \mu^{(n)})}{\mathbf{b}^{\star\mathsf{T}}_{\mathcal{I}}(\lambda^{(1)}, \mu^{(n)})\mathbf{W}\mathbf{b}^{\star}_{\mathcal{I}}(\lambda^{(1)}, \mu^{(n)})}$$

3. Update *Loop*

$$\mu^{(n+1)} \leftarrow \mu^{(n)} + \Delta$$

$$n \leftarrow n + 1$$

4. *Loop* ends when $stress(\lambda^{(1)}; \mu^{(n-1)}) \leq stress(\lambda^{(1)}; \mu^{(n)})$

- Return $\mu \leftarrow \mu^{(n-1)}$ and get corresponding $[\widehat{d}_{ij}] \leftarrow [\widehat{d}_{ij}^{(n-1)}]$

Nelder Mead Algorithm [31] can also help to find μ and corresponding $[\widehat{d}_{ij}]$. However, in practice, *Line Search* performs better and faster.

Figure 5.2 shows the Venn diagram of $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_2^\star$.

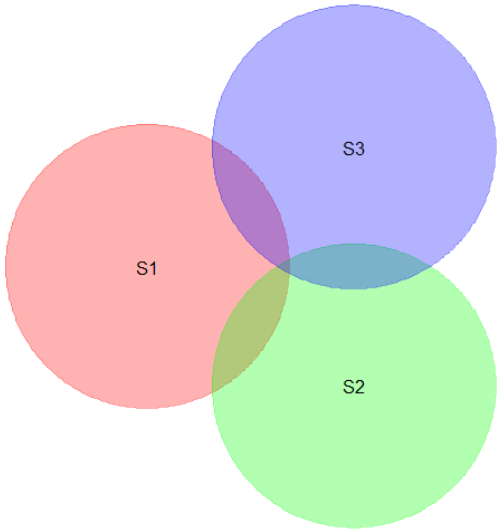


Figure 5.2: `vennplot(twoWayGenerate = TRUE)`

In general

If any two way intersections are missing, such as $S_i \cap S_j$, but there are some higher than two way intersection sets $\{S_1 \cap S_2 \cap \cdots \cap S_M, S_1 \cap S_2 \cap \cdots \cap S_L, \ldots\} \subseteq S_i \cap S_j$ and $s_1 \cap s_2 \cap \cdots \cap s_M = \max(\{s_1 \cap s_2 \cap \cdots \cap s_M, s_1 \cap s_2 \cap \cdots \cap s_L, \ldots\})$. Based on the generation we mentioned before, we can use $s_1 \cap s_2 \cap \cdots \cap s_M$ to estimate $s_i \cap s_j$, $\widehat{s}_{ij} = s_{12\cdots M} \times \mu^{M-2}$

5.0.3 Weight

We can also give the three way intersection very high weight to realize its layout. For example: `vennplot(s*_I, weight = c(1,1,1,10))` and Figure 5.3 shows the layout. Comparing with the "Special case" algorithm, this way is a bit tricky, since we just give higher weight to some intersections we want to see. In Figure 5.3, the three way intersection is larger than the one in Figure 5.2. In the scatter plot in Figure 5.4, for blue dots, the three intersection one is better with sacrificing other disjoint pieces.

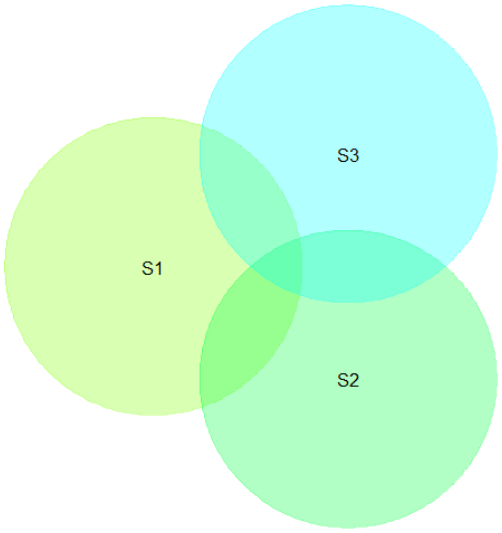


Figure 5.3: `vennplot()` with high weight on three way intersection

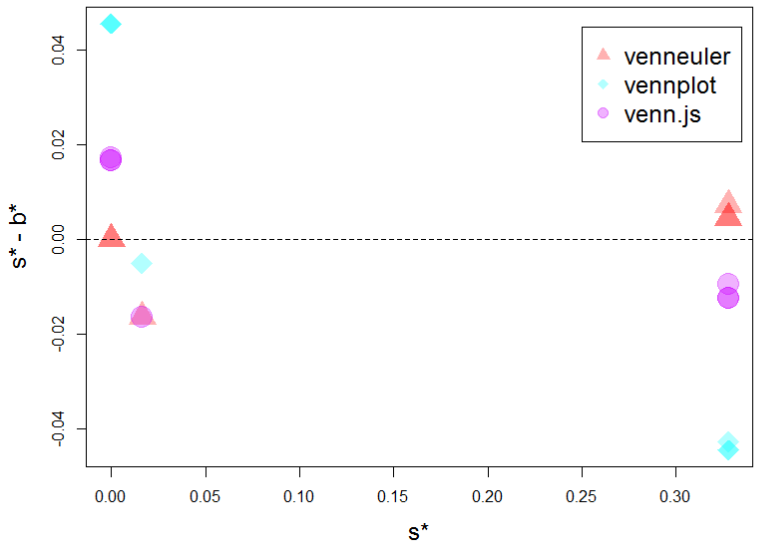


Figure 5.4: scatter plot with high weight on three way intersection, red dots represent `venneuler`, blue ones represent `vennplot()`, purple ones represent `venn.js`

Chapter 6

Undirected connected components

Suppose we have a complicated data set $\mathbf{S} = \{S_1, S_2, \ldots, S_{12}\}$, the input given disjoint subsets $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_1^\star$

$$\begin{aligned}\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_3^\star \;=\; \{&S_1^\star, S_2^\star, S_3^\star, S_5^\star, S_{23}^\star, S_{24}^\star, S_{45}^\star, S_{56}^\star,\\ &S_{126}^\star, S_{234}^\star, S_7^\star, S_8^\star, S_9^\star, S_{10}^\star, S_{11}^\star, S_{12}^\star,\\ &S_{78}^\star, S_{7,10}^\star, S_{9,12}^\star, S_{8,11}^\star, S_{10,12}^\star, S_{9,11}^\star\}\end{aligned}$$

with its size $\mathbf{s}_{\mathcal{I}}^\star$:

$$\begin{aligned}\mathbf{s}_{\mathcal{I}}^\star \;=\; [&s_1^\star = 200, s_2^\star = 100, s_3^\star = 100, s_5^\star = 550,\\ &s_{2,3}^\star = 100, s_{2,4}^\star = 100, s_{4,5}^\star = 50, s_{5,6}^\star = 100, s_{1,2,6}^\star = 200, s_{2,3,4}^\star = 100,\\ &s_7^\star = 232, s_8^\star = 378, s_9^\star = 296, s_{10}^\star = 423, s_{11}^\star = 698, s_{12}^\star = 337,\\ &s_{7,8}^\star = 133, s_{7,10}^\star = 165, s_{9,12}^\star = 151, s_{8,11}^\star = 89, s_{10,12}^\star = 212, s_{9,11}^\star = 113]\end{aligned}$$

$\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})_3^\star$ is the corresponding power set, any sets in $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})_3^\star \setminus \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_3^\star$ are $\varnothing$ and $\mathbf{s}_{\mathcal{P}}^\star = [\mathbf{s}_{\mathcal{I}}^\star, 0, \ldots, 0]$. Figure 6.1 shows the layout of `venneuler()` and `venn.js()`.

(a) (b)

Figure 6.1: (a) `venneuler()` (b) `venn.js()`

Neither of these two seems perfect. If we look at $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_3^\star$, we can find $\{S_1, S_2, \ldots, S_6\}$ are connected by some intersections, $\{S_7, S_8, \ldots, S_{12}\}$ are connected by some intersections. However, $\{S_1, S_2, \ldots, S_6\}$ and $\{S_7, S_8, \ldots, S_{12}\}$ are totally two groups without any connection, like Figure 6.2. Hence, for any set, we can divide, conquer each group and then unite.

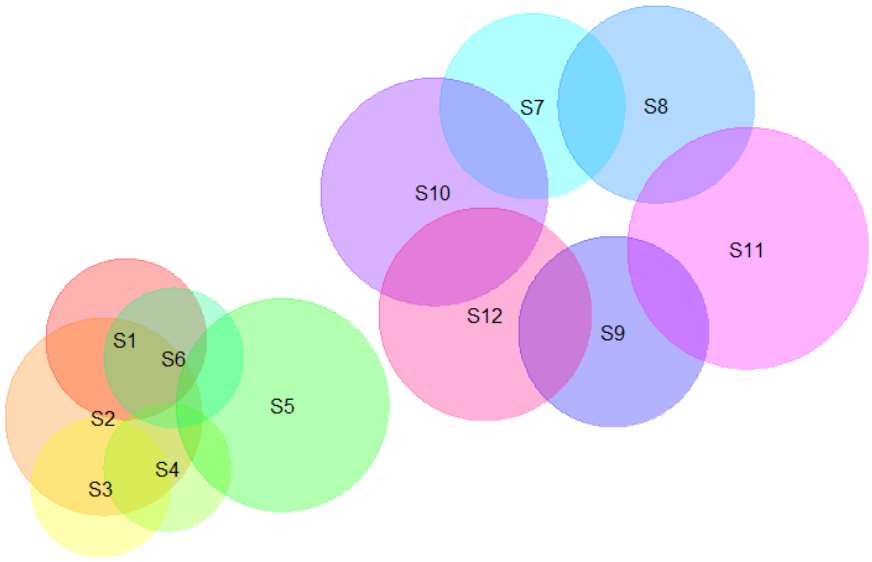


Figure 6.2: `vennplot()`

Let higher disjoint set $\boldsymbol{\mathcal{H}(\mathcal{S})} = \boldsymbol{\mathcal{I}(\mathcal{S})} \backslash \mathbf{S} = \{H_1, \dots H_{\mathcal{N}}\}$ be the higher order intersection set (the order is larger than 1) with size $\mathcal{N}$, where $\mathcal{N} = N - m$.

Undirected connected components $\boldsymbol{\mathcal{G}} = \{\mathbf{V}, \mathbf{E}\}$, where $\mathbf{V}$ is a set whose elements are called nodes and $\mathbf{E}$ is the undirected edges connecting nodes. In the given set $\boldsymbol{\mathcal{I}(\mathcal{S})}$, any order k intersections, where $k \geq 2$, can be treated as $\binom{k}{2}$ edges ; such as $S_1 \cap S_2 \cap S_3$ implies S_1, S_2 and S_3 must be connected thus $\mathbf{E} = \{S_1 \cap S_2 \cap S_3\}$ and $\mathbf{V} = \{S_1, S_2, S_3\}$

Suppose we have η groups, let $\boldsymbol{\mathcal{G}}_1 = \{\mathbf{V}_1, \mathbf{E}_1\}, \dots, \boldsymbol{\mathcal{G}}_\eta = \{\mathbf{V}_\eta, \mathbf{E}_\eta\}$, where $1 \leq \eta \leq m$. In each $\mathbf{V}_i$, any two sets can be connected by edges $\mathbf{E}_i$ and we have:

$$\mathbf{V}_1 \cup \mathbf{V}_2 \cup \ldots \cup \mathbf{V}_\eta = \mathbf{S} = \{S_1, S_2, \dots, S_m\}$$

$$\mathbf{E}_1 \cup \mathbf{E}_2 \cup \ldots \cup \mathbf{E}_\eta = \boldsymbol{\mathcal{H}(\mathcal{S})}$$

For any $\boldsymbol{\mathcal{G}}_i$ and $\boldsymbol{\mathcal{G}}_j$

$$\mathbf{V}_i \cap \mathbf{V}_j = \varnothing$$

$$\mathbf{E}_i \cap \mathbf{E}_j = \varnothing$$

where $i \neq j$

6.1 Divide $\boldsymbol{\mathcal{G}}$

Assume we have a data set $\mathbf{S} = \{S_1, S_2, ..., S_m\}$, a size N input set $\boldsymbol{\mathcal{I}}(\boldsymbol{S})$ and a size $\mathcal{N}$ higher order set $\boldsymbol{\mathcal{H}}(\boldsymbol{S})$. We can use the following algorithm to divide groups. Before we start, let us introduce some functions which can help us better understand this algorithm:

- *separate* function: input is a high order disjoint intersection; output is each set. e.g. $separate(S_1 \cap S_2 \cap S_3) = \{S_1, S_2, S_3\}$
- *unique* function: returns a vector but with duplicate elements removed. e.g. $unique(S_1, S_2, S_3, S_1) = \{S_1, S_2, S_3\}$
- *any* function: give a set of logical vectors, is at least one of the values true. e.g. $\mathbf{z} = \{A, B, C\}$, $\mathbf{Z} = \{A, D, E\}$, then $any(\mathbf{z} \in \mathbf{Z}) = TRUE$; $\mathbf{z} = \{A, B, C\}$, $\mathbf{Z} = \{D, E\}$, then $any(\mathbf{z} \in \mathbf{Z}) = FALSE$; $\mathbf{z} = \{TRUE, FALSE\}$, then $any(\mathbf{z} = TRUE) = TRUE$.
- *which* function: give the true index of a logical object. e.g. $\mathbf{z} = \{TRUE, FALSE, TRUE\}$;$which(\mathbf{z} = TRUE) = \{1, 3\}$
- *length* function: get the length of vectors. e.g. $\mathbf{z} = \{TRUE, FALSE, TRUE\}$; $length(\mathbf{z}) = 3$.

The following procedure can help us find the $\boldsymbol{\mathcal{G}}$s

1. **if** $\mathcal{N} = 2^m - m - 1$ **then**
   $\eta = 1$; $\mathbf{V}_1 = \mathbf{S}$ and $\mathbf{E}_1 = \boldsymbol{\mathcal{H}}(\boldsymbol{S})$
2. **else if** $\mathcal{N} = 1$ **then**
   Where $\boldsymbol{\mathcal{H}}(\boldsymbol{S}) = \{H_1\}$, assume H_1 is the order k intersection, thus $separate(H_1)$ is a k size set and $\eta = m - k + 1$. $S \setminus separate(H_1) = \{\gamma_1, \gamma_2, \ldots, \gamma_{m-k}\}$. Hence $\mathbf{V}_i = \gamma_i$, $\mathbf{E}_i = \varnothing$, where $1 \leq i \leq m - k$ and $\mathbf{V}_\eta = seperate(H_1)$, $\mathbf{E}_\eta = H_1$.
3. **else**
   (a) $\boldsymbol{\mathcal{H}}(\boldsymbol{S}) = \{H_1, H_2, \ldots H_\mathcal{N}\}$
   (b) *Outer loop:* $i = 1, \ldots, \mathcal{N}$:
      i. $\mathbf{Boolean}_1 = \{bool_1, \ldots, bool_\mathcal{N}\}$ and $bool_u = FALSE$, where $1 \leq u \leq \mathcal{N}$;
      ii. *Inner loop*: for $j = i, \ldots, \mathcal{N}$:

- $bool_j = any(separate(H_i) \in separate(H_j))$

iii. $\mathbf{A} = which(\mathbf{Boolean}_1 = TRUE)$; $length(\mathbf{A}) = a$ and $\mathbf{A} = \{A_1, \ldots, A_a\}$

iv. $\mathbf{V}_i = unique(separate(H_{A_1}), \ldots, separate(H_{A_a}))$ and $\mathbf{E}_i = \{H_{A_1}, \ldots, H_{A_a}\}$

v. **if** $i > 1$ **then**
$\mathbf{Boolean}^{(2)} = \{bool_1, \ldots, bool_{i-1}\}$ and $bool_u = FALSE$, where $1 \leq u \leq i-1$

A. *Inner loop*: for $j = 1, \ldots, i-1$:

- $bool_j = any(\mathbf{V}_j \in \mathbf{V}_i)$

B. $\tau = which(\mathbf{Boolean}^{(2)} = TRUE)$; $length(\tau) = 0$ or 1

C. **if**$(length(\tau) = 1)$ **then**

- $\mathbf{V}_\tau = unique(\mathbf{V}_\tau, \mathbf{V}_i)$ and $\mathbf{E}_\tau = unique(\mathbf{E}_\tau, \mathbf{E}_i)$
- $\mathbf{V}_i = \mathbf{E}_i = \varnothing$

(c) Get rid of all the emptysets and reduce $\mathcal{N}$ to ν, where $1 \leq \nu \leq \mathcal{N}$. $\forall\ i$, $\mathbf{V}_i \neq \varnothing$ and $\mathbf{E}_i \neq \varnothing$, where $1 \leq i \leq \nu$

(d) **if**$(\mathbf{V}_1 \cup \ldots \cup \mathbf{V}_\nu = \mathbf{S})$ **then**

- $\nu = \eta$

else

- $\mathbf{S} \setminus (\mathbf{V}_1 \cup \ldots \cup \mathbf{V}_\nu) = \{\gamma_1, \ldots \gamma_{\eta-\nu}\}$
- $\mathbf{V}_{\nu+j} = \gamma_j$ and $\mathbf{E}_{\nu+j} = \varnothing$, where $1 \leq j \leq \eta - \nu$

4. $\mathcal{G}_i = \{\mathbf{V}_i, \mathbf{E}_i\}$, where $1 \leq i \leq \eta$ and return $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_\eta\}$

6.2 Detect case

If $\boldsymbol{\mathcal{I}(S)} \neq \boldsymbol{\mathcal{P}(S)}$, but any sets in $\boldsymbol{\mathcal{P}(S)} \setminus \boldsymbol{\mathcal{I}(S)}$ are $\varnothing$. Then, we could just set $\mu = 1$ and say $\boldsymbol{\mathcal{G}}_i$ is "common case"; else, we could use the following procedure to detect which "case" ("common" or "special") $\boldsymbol{\mathcal{G}}_i$ belongs to.

A function *order* can be defined as:

- *order*: give the order of an intersection. e.g. $order(S_1 \cap S_2 \cap S_3) = 3$

For each $\boldsymbol{\mathcal{G}}_i = \{\mathbf{V}_i, \mathbf{E}_i\}$, where $1 \leq i \leq \eta$ and $\mathbf{E}_i = \{e_1, \ldots, e_\kappa\}$ with length κ, e_j means edges (intersections) belonging to undirected connected component $\boldsymbol{\mathcal{G}}_i$, where $1 \leq j \leq \kappa$. $\mathbf{L}$ is the two way intersection index and defined as: $\mathbf{L} = which(order(\mathbf{E}_i) = 2)$; $length(\mathbf{L}) = l$.

1. **if**$(l = \kappa)$ **then** $\boldsymbol{\mathcal{G}}_i$ belongs to "common case"
2. **else**
   - $\mathbf{L} = \{L_1, \ldots, L_l\}$, where $1 \leq L_1 \leq L_l \leq \kappa$
   - $\boldsymbol{\mathcal{O}}_2 = \{e_{L_1}, \ldots, e_{L_l}\}$
   - $\mathbf{T} = which(order(E_i) > 2)$ and $length(\mathbf{T}) = t$
     - $t + l = \kappa$
   - $\mathbf{T} = \{T_1, \ldots, T_t\}$
   - $\boldsymbol{\mathcal{O}}_r = \{e_{T_1}, \ldots, e_{T_t}\}$
     - $\boldsymbol{\mathcal{O}}_2 \cup \boldsymbol{\mathcal{O}}_r = \mathbf{E}_i$

   (a) *Outer Loop*: $j = 1, \ldots, t$

   i. Assuming e_{T_j} is the order k intersection, where $order(e_{T_j}) = O > 2$ and it implies $\binom{O}{2}$ edges. Set $q = \binom{O}{2}$ and $\mathbf{Q}$ is an order two intersections list which e_{T_j} connotes.

$$\mathbf{Q} = \{Q_1, \ldots, Q_q\}$$

   ii. $\mathbf{Boolean} = \{bool_1, \ldots, bool_q\}$ and $bool_u = FALSE$, where $1 \leq u \leq \mathcal{Q}$

   iii. *Inner Loop*: $u = 1, \ldots q$
   - $bool_u = any(Q_u \in \boldsymbol{\mathcal{O}}_2)$

   iv. **If** $(any(\mathbf{Boolean} = FALSE))$ **then**
   - $\boldsymbol{\mathcal{G}}_i$ belongs to "special case"
   - **break** *Outer Loop*

   else $\boldsymbol{\mathcal{G}}_i$ to be determined

   (b) **If** $\boldsymbol{\mathcal{G}}_i$ still to be determined **then**
   $\boldsymbol{\mathcal{G}}_i$ belongs to "common case"

6.3 Unite $\boldsymbol{\mathcal{G}}$

$\boldsymbol{\mathcal{G}} = \{\boldsymbol{\mathcal{G}}_1, \ldots, \boldsymbol{\mathcal{G}}_\eta\}$ and $\boldsymbol{\mathcal{G}}_i = \{\mathbf{V}_i, \mathbf{E}_i\}$. $\mathbf{V}_i$ is the size m_i data set and $\mathbf{B}_i$ is the corresponding balls with radiuses $\mathbf{R}_i$. Hence, $\mathbf{R}_i$ is a size $m_i \times 1$ vector. After **Detect case**, conquer one by one and get $[\mathbf{C}_1, \ldots, \mathbf{C}_\eta]^\mathsf{T}$. Thus $\mathbf{C}_i$ is a $m_i \times p$ matrix and $p = 2$ or 3.

$$\sum_{i=1}^{\eta} m_i = m$$

The following procedures can help us to lay out $\mathbf{C}_i$ together but with reasonable distances.

1. *Initialization*:
   - Put $\mathbf{C}_1$ in a rectangle box, which can load all these balls.
   - **if** $(\eta = 1)$ return $\mathbf{C}_i$
     else go to next *Outer Loop*
2. *Outer Loop*: $i = 2, \ldots, \eta$
   (a) Randomly select one point $\mathbf{x}$ in this box. Then pick one coordinate (row) $\mathbf{c}_j$ in $\mathbf{C}_i$ (with its radius ρ_j), where $1 \leq j \leq m_i$. This coordinate $\mathbf{c}_j$ must have either the largest (x or y or z) or smallest (x or y or z).

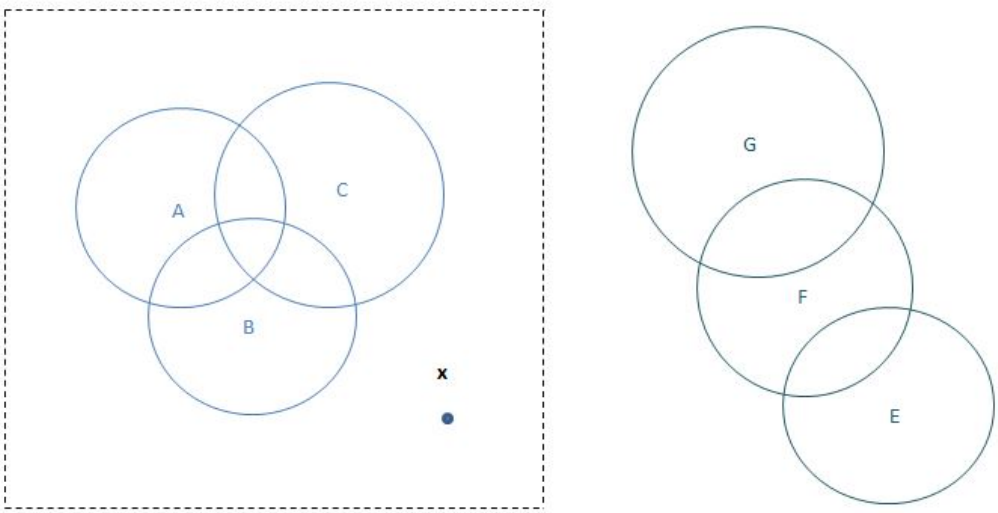


Figure 6.3: The choice of $\mathbf{c}_j$ and $\mathbf{x}$

In Figure 6.3, the imaginary line is a rectangular box and $\mathbf{x}$ is the point we randomly generate. $\mathbf{c}_j$ we choose is the centre ball E, which has the largest x (smallest y).

(b) Translate $\mathbf{C}_i$ to $\mathbf{C}'_i$:

$$\mathbf{C}'_i = \mathbf{C}_i + \mathbf{1}_{m_i}\mathbf{x} - \mathbf{1}_{m_i}\mathbf{c}_j$$

where $\mathbf{1}_{m_i} = [1, 1, \ldots, 1]^\mathsf{T}$ with size m_i

- All the distances $\{d_{1x}, d_{2x}, ..., d_{m_i x}\}$ between $\mathbf{x}$ and $\mathbf{C}_{i-1}$ are larger than $\mathbf{R}_{i-1} + \mathbf{1}^\mathsf{T}_{m_{i-1}}\rho_j$, where $\mathbf{1}_{m_{i-1}} = [1, 1, \ldots, 1]^\mathsf{T}$ with size m_{i-1}.
- At least one of distances $\{d_{1x}, d_{2x}, ..., d_{m_i x}\}$ is smaller than $\mathbf{R}_{i-1} + \mathbf{1}^\mathsf{T}_{m_{i-1}}\rho_j + \mathbf{1}^\mathsf{T}_{m_{i-1}}\delta$, where $\delta \in \Re^+$

If any of these conditions does not match, then go back to first step of *Outer Loop* and get another random point $\mathbf{x}$.

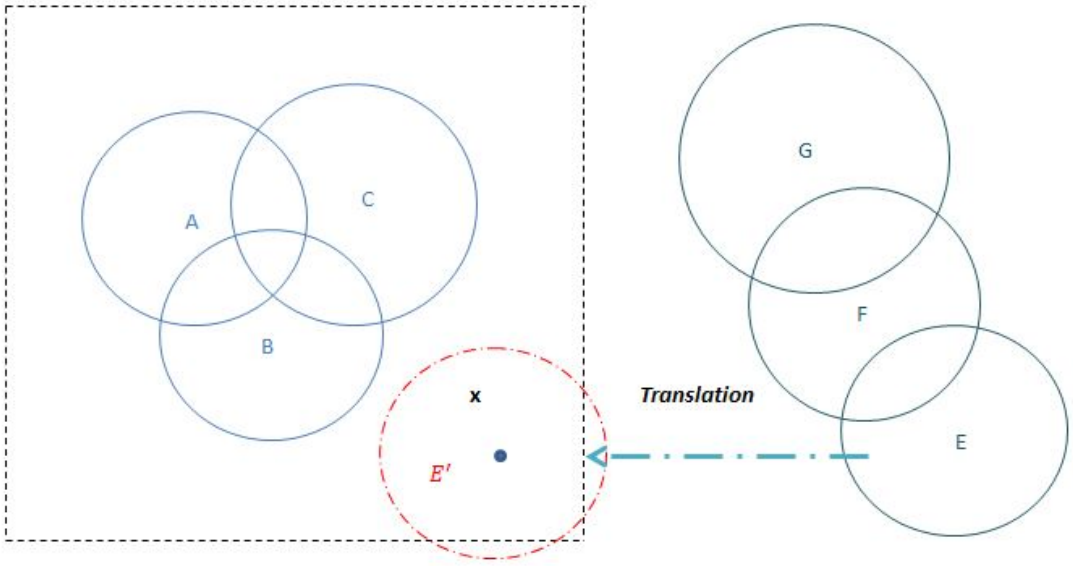


Figure 6.4: Translation $\mathbf{c}_j$

(c) Rotate $\mathbf{C}'_i$ until $\mathbf{C}_{i-1}$ and $\mathbf{C}'_i$ are totally separated.

i. Define a $m_{i-1} \times m_i$ matrix $\boldsymbol{\mathcal{D}}$

$$\begin{aligned}\boldsymbol{\mathcal{D}} &= \textstyle\sum_{k=1}^{p} \left(\mathbf{C}_{i-1}\mathbf{e}_k\mathbf{1}^\mathsf{T}_{m_i} - \mathbf{1}_{m_{i-1}}\mathbf{e}_k^\mathsf{T}\mathbf{C}_i^\mathsf{T}\right) \circ \left(\mathbf{C}_{i-1}\mathbf{e}_k\mathbf{1}^\mathsf{T}_{m_i} - \mathbf{1}_{m_{i-1}}\mathbf{e}_k^\mathsf{T}\mathbf{C}_i^\mathsf{T}\right) \\ &\quad - \left(\mathbf{R}_{i-1}\mathbf{1}^\mathsf{T}_{m_i} + \mathbf{1}_{m_{i-1}}\mathbf{R}_i^\mathsf{T}\right) \circ \left(\mathbf{R}_{i-1}\mathbf{1}^\mathsf{T}_{m_i} + \mathbf{1}_{m_{i-1}}\mathbf{R}_i^\mathsf{T}\right)\end{aligned}$$

where e_k is a p-dimension standard basis, $e_k = [0, \ldots, 1, \ldots, 0]^{\mathsf{T}}$ only the k th element is 1.

ii. **if** (all elements in $\boldsymbol{\mathcal{D}}$ are equale or larger than 0)
Return $\mathbf{C}'_i$

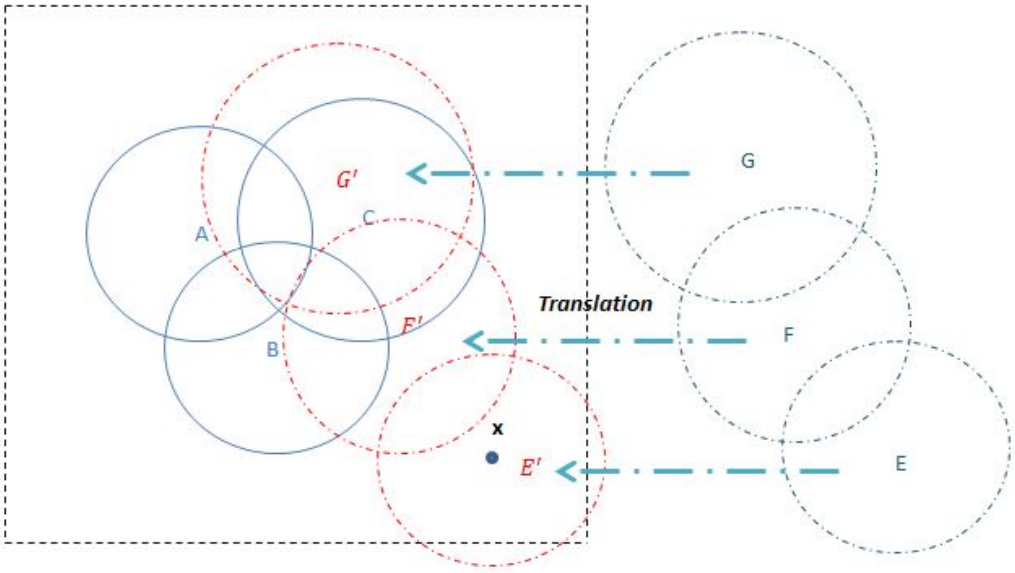


Figure 6.5: Translation

In Figure 6.5, not all elements in $\boldsymbol{\mathcal{D}}$ are equale or larger than 0. Hence, we need to do rotation until they are totally separated.

iii. **else**

- *Inner Loop*: $\theta = \frac{\pi}{18}, \frac{2\pi}{18}, \ldots, 2\pi$
- – $\mathbf{C}'_i \leftarrow \mathbf{C}'_i \times \boldsymbol{\mathcal{R}}$
  - ∗ for p = 2

$$\boldsymbol{\mathcal{R}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

* for p =3

$$\boldsymbol{\mathcal{R}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) \\ 0 & \sin(\theta_1) & \cos(\theta_1) \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_1) & 0 & \cos(\theta_2) \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

θ_1, θ_2 and θ_3 are not necessarily equal. For simplify, we can set $\theta_1 = \theta_2 = \theta_3 = \theta$

- Caculate $\boldsymbol{\mathcal{D}}$
- **if** (all elements in $\boldsymbol{\mathcal{D}}$ are equale or larger than 0) **then break** the *Inner Loop* and return $\mathbf{C}'_i$
  else $\theta \leftarrow \theta + \frac{\pi}{18}$ and repeat the *Inner Loop*

- If $\theta = 2\pi$ and $\boldsymbol{\mathcal{D}}$ still doesn't meet the conditions, then go back to random point selection and pick a new $\mathbf{x}$

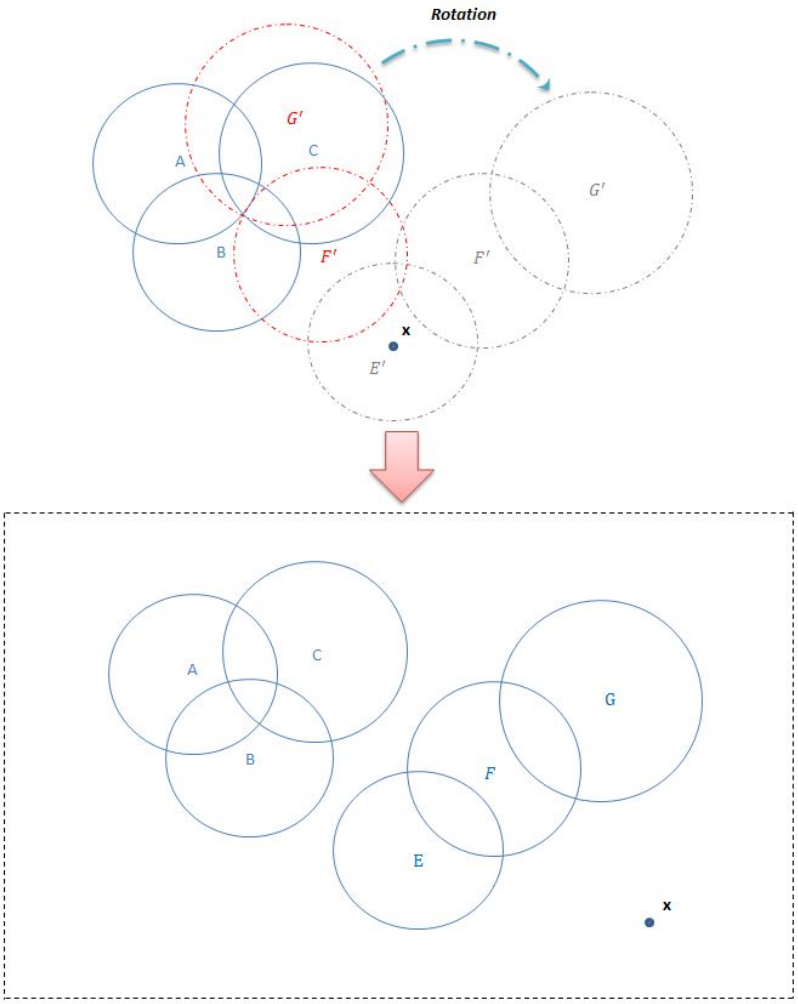


Figure 6.6: Rotation

(d) $\mathbf{C}_i = [\mathbf{C}_{i-1}^{\mathsf{T}}, \mathbf{C}_i'^{\mathsf{T}}]^{\mathsf{T}}$

(e) $i \leftarrow i + 1$

3. Return $\mathbf{C} = \mathbf{C}_\eta$ and $\mathbf{C}$ is final layout with all united groups.

Chapter 7

Examples

Figure 7.1 is an example of factor data on human encountering with great white sharks. The data is collected by Doctor Pierre-Jerome Bergeron [5]. In this example, it shows the relationship among nationality, time and fatality.

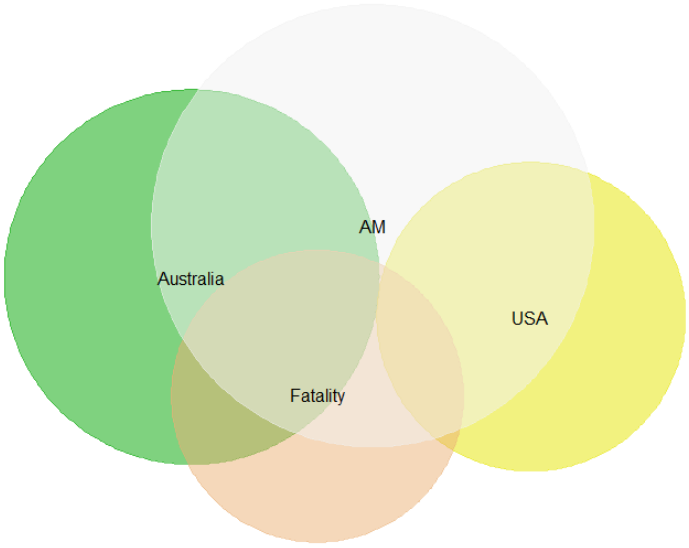


Figure 7.1: sharks data frame

Here, the supplementary sets of “AM”, “Australia and USA”, “Fatality” are “PM”, “oth-

ers", "Survive", respectively. So, any disjoint parts either fall into sets or supplementary sets. The *stress* of this example is 0.06922327.

Fifteen-ring data set with multiple groups $\mathbf{S}_5 = \{S_A, S_B, \ldots, S_P\}$, the input given disjoint subsets $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_4^\star$ with size $\mathbf{s}_{\mathcal{I}}^\star$:

$$\begin{aligned}\mathbf{s}_{\mathcal{I}}^\star \;=\; & [s_A^\star = 80, s_B^\star = 50, s_C^\star = 100, s_D^\star = 100, s_E^\star = 100, s_F^\star = 40,\\ & s_{A,C}^\star = 30, s_{A,D}^\star = 30, s_{B,E}^\star = 30, s_{A,E}^\star = 40, s_{B,F}^\star = 10,\\ & s_G^\star = 60, s_H^\star = 50, s_I^\star = 100, s_J^\star = 40, s_K^\star = 50, s_L^\star = 100,\\ & s_M^\star = 30, s_O^\star = 50, s_P^\star = 60, s_{G,H}^\star = 20, s_{K,L}^\star = 20, s_{L,M}^\star = 20, s_{O,P}^\star = 30]\end{aligned}$$

$\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})_4^\star$ is the corresponding power set, any sets in the complement of $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_4^\star$ are $\varnothing$. Figure 7.2 shows the layout with *stress* 0.00014.

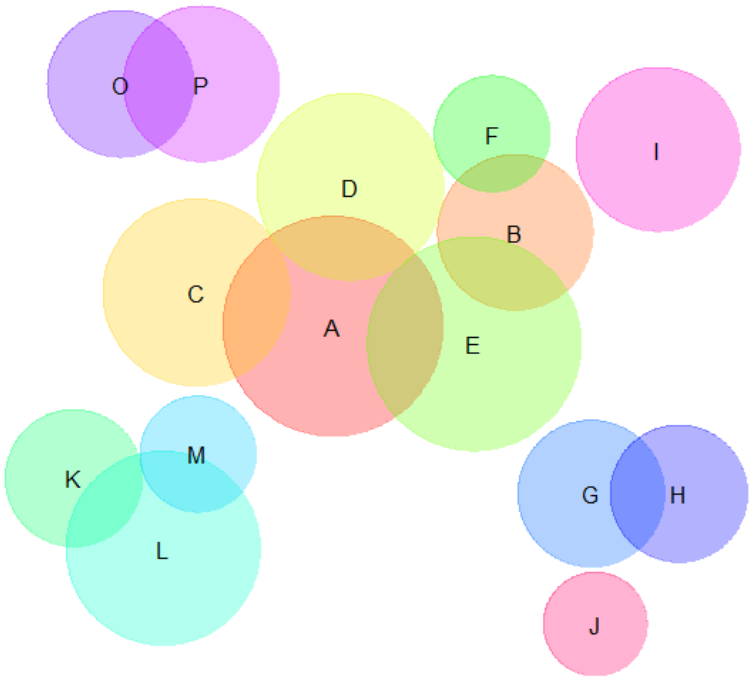


Figure 7.2: Multiple groups in one data set

However, not all data are suitable to be illustrated by area proportional Venn diagrams. For example, Figure 7.3 shows the size proportional Venn diagram of the data set in Figure 1.4 (b).

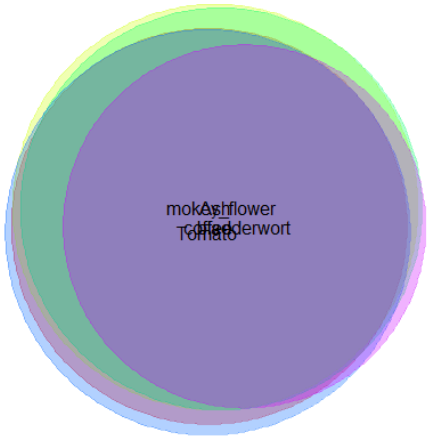


Figure 7.3: Figure 1.4 (b) data set

The *stress* is 0.021 which should indicate a good quality Venn diagram. Nevertheless, we can only tell these five species share a large number of genes.

Chapter 8

Comparison with other Venn algorithms

We compare `vennplot()` with other popular approaches to the circular area-proportional Venn algorithms `venneuler()` and `venn.js()`.

We generate 100 instances of $\mathbf{S}_{\mathcal{P}} = \{S_1, \ldots, S_m\}$ for each value of $m = 3, 4, \ldots, 8$. For each $\mathbf{S}_{\mathcal{P}}$ generates its disjoint set $s^{\star}_{\mathcal{P}}$ independently from $U(0, 10)$. Compare results of `vennplot()` with `venneuler()` and venn.js() on two criteria.

$$abs = \sum_{i}^{N} |b_i^{\star} - s_i^{\star}| \qquad stress = \frac{RSS}{TSS}$$

For each criterion will look at differences in results `venneuler()`$-$`vennplot()`, `venn.js()`$-$`vennplot()`. The larger this difference the more favoured is `vennplot()`. Figure 8.1 illustrates the comparison with `venneuler()`.

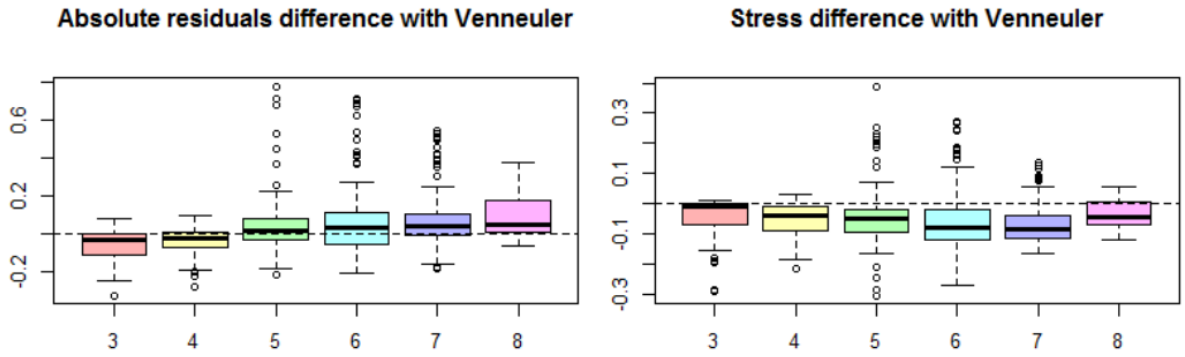


Figure 8.1: `venneuler()`- `vennplot()`, *abs* and *stress*

The *abs* of `vennplot()` is better after four circles, the *stress* is worse but comparable. We may also interested in how well the fit of each way intersection.

$$stress(k) = \frac{\sum_{i \in k-way} \widehat{r}_i^2}{\sum_{i \in k-way} s_i^2}$$

m is the highest order and $1 \leq k \leq m-1$ (when $k = m$, $s^{\star}_{1...m} = s_{1...m}$); $stress(k)$ is the corresponding *stress* and comparison is shown in Figure 8.2.

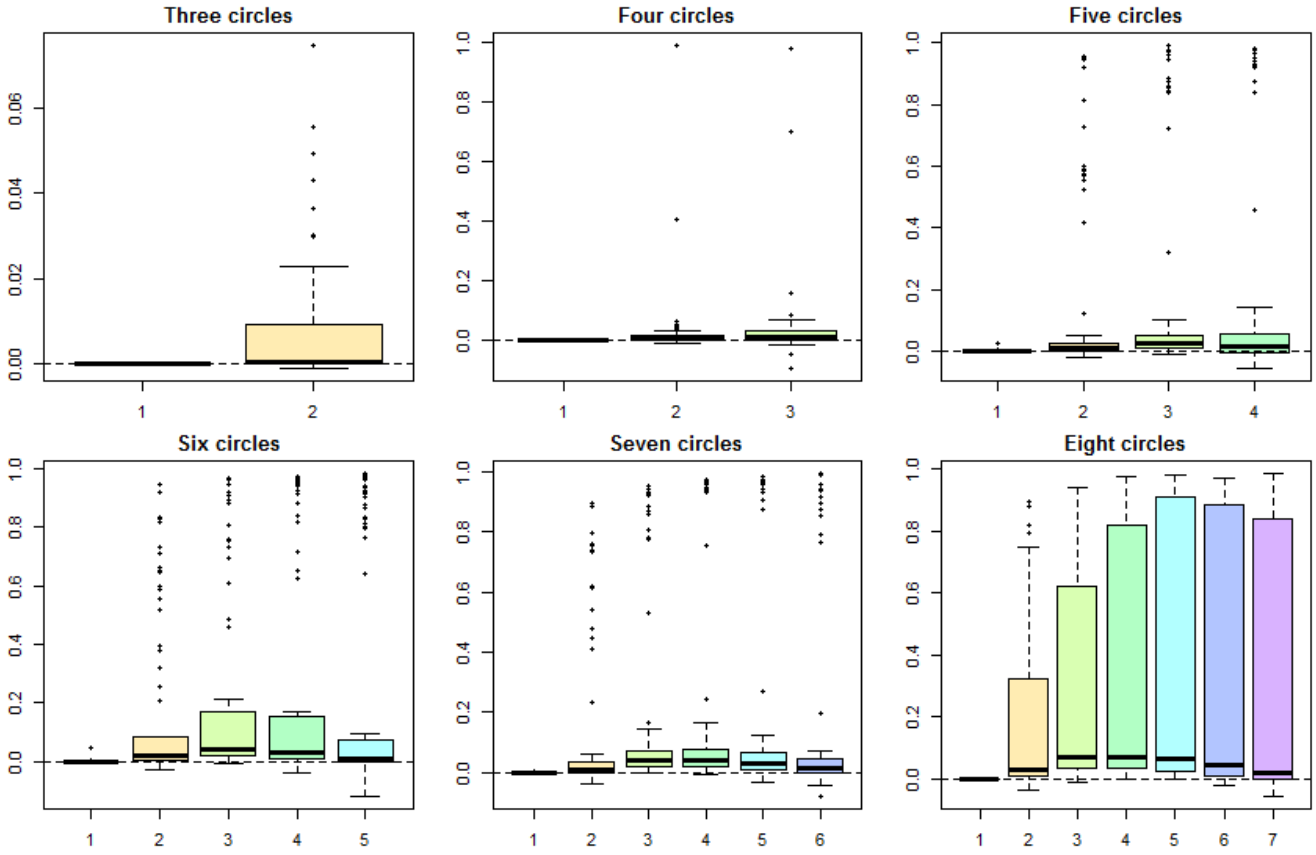


Figure 8.2: `venneuler()` - `vennplot()`, $stress(k)$, where $1 \leq k \leq m-1$

We can find `vennplot()` does better on the fit of $stress(k)$, from three circles to eight circles. However, if `vennplot()` did well on all intersections but bad in *stress*, which should indicate we fit really poor on $s_{1...m}$ and the poor $s_{1...m}$ explodes up the *stress*. Figure 8.3 illustrates the comparison of $stress(m)$.

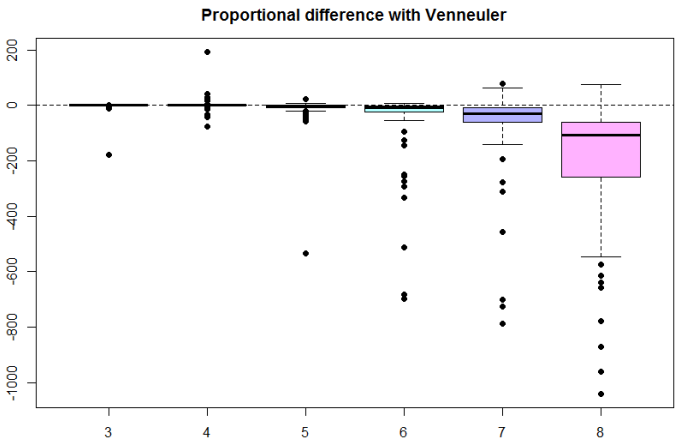


Figure 8.3: `venneuler()`- `vennplot()`, $stress(m)$

However, sometimes, people may also interested in the fit of total size, rather than disjoint part. This can be captured by fitting the linear model:

$$\mathbf{b}_{\mathcal{P}} = \mathbf{s}_{\mathcal{P}}\beta + \mathbf{r}$$

criteria stress is defined as ϕ and the boxplot shows the comparison with `venneuler()`, given the random generation data set.

$$\phi = \frac{\widehat{\mathbf{r}}^{\mathsf{T}}\widehat{\mathbf{r}}}{\mathbf{b}_{\mathcal{P}}{}^{\mathsf{T}}\mathbf{b}_{\mathcal{P}}}$$

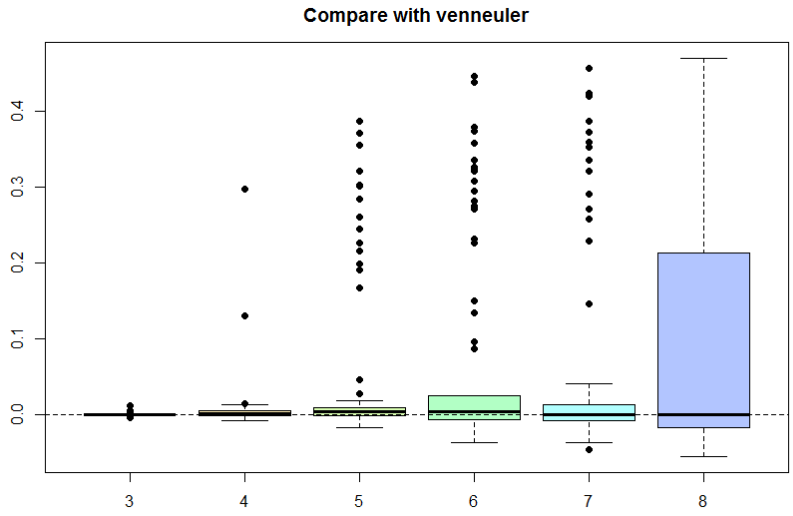


Figure 8.4: `venneuler()`- `vennplot()`, ϕ

Since we cannot grab the centres and radii from `javascript`, we implement Frederickson's algorithm in `R` (suppose `venn.js()` is totally based on his algorithm on his website [18]), use random initial configuration as he did. Figure 8.5 shows the comparison on two criteria with random generated data sets (the same as before).

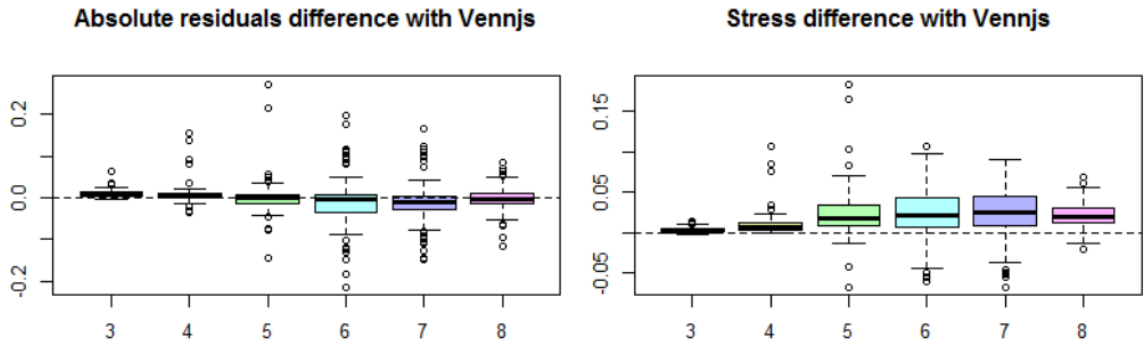


Figure 8.5: `venn.js()` - `vennplot()`, *abs* and *stress*

The boxplot of *abs* are roughly symmetric around zero, which means they should be very similar based on this criteria; *stress* is slightly better (that is because after minimizing 2.5, we import λ to minimize *stress*). Frederickson did improve performance on

possible configurations, however, based on some impossible point configurations (like what we randomly generated), it may not surpass `venneuler()`.

However, if there are multiple groups in one data set, the "three step rule" of `vennplot()`, divide, conquer and unite could really improve the performance. For example, Figure 8.6 shows the layout of $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_4^\star$ by `venneuler()` with *stress* 0.37. We can also generate each data set 100 times which contains two to five groups. In each group, circles vary from one to five (if the maximum circle is over five, after four groups, `venneuler()` is more likely to terminate). Figure 8.7 illustrates the comparison.

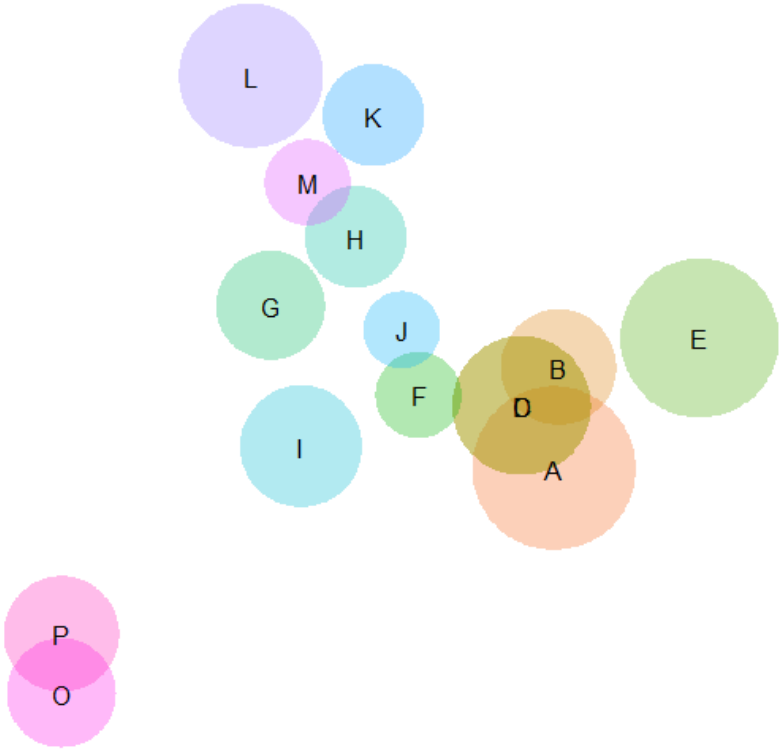


Figure 8.6: For set $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})_4^\star$

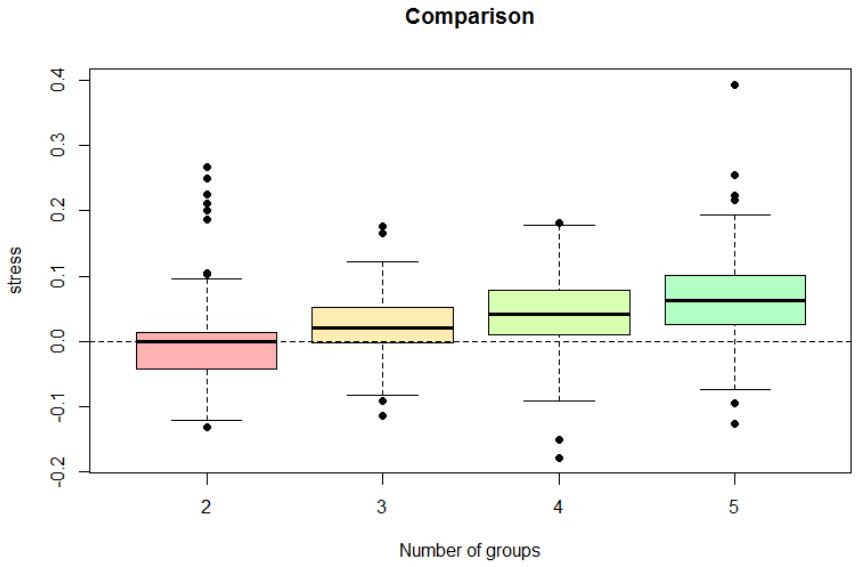
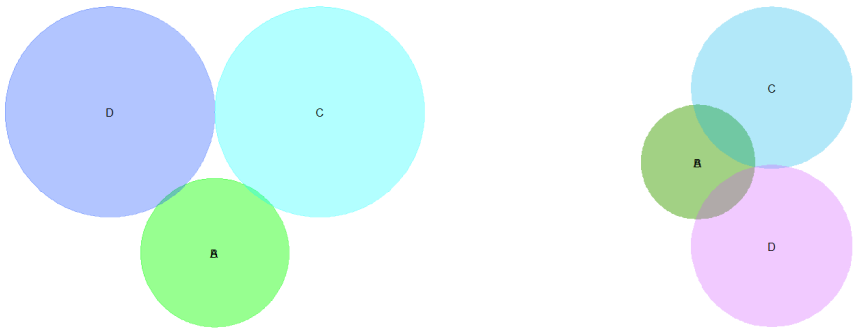


Figure 8.7: `venneuler()`- `vennplot()`, *stress* multiple groups

For `venn.js()`, if the *stress* of `vennplot()` is smaller in a single group, it would not perform worse in multiple groups.

For Wilkinson's algorithm, the steepest descent, equation 2.3 has a problem: if the initial configurations of balls (for example, b_i and b_j) are totally overlaid with each other, then, this steepest descent for $\mathbf{c}_i$ and $\mathbf{c}_j$ would be zero, thus b_i and b_j could be hard to separate. An example is $\mathbf{S}_5 = \{S_A, S_B, S_C, S_D\}$, the disjoint subsets $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})^\star_5$ with size $\mathbf{s}^\star_{\mathcal{I}} = [s^\star_A = 3, s^\star_B = 3, s^\star_C = 10, s^\star_D = 10, s^\star_{AB} = 2]$, $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})^\star_5$ is the corresponding power set and any sets in $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})^\star_5 \setminus \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})^\star_5$ are $\varnothing$. Figure 8.8 shows the initial configuration and final layout of `venneuler()`.



(a) (b)

Figure 8.8: (a) is the initial configuration and (b) is the final one

For "Constrained MDS" (Frederickson's algorithm), the main problem is that equation 2.7 only focuses on the two way intersection but does not consider higher way intersections. Thus, he priorities to one way and two way intersections.

Except `venneuler()` and `venn.js()`, there is another good **R** function `eulerr()` with similar generalized Venn algorithms. The function is also based on Wilkinson and Frederickson's algorithm, but with different optimizers. Also worth noting, in `eulerr()`, radii are not fixed and taken as an optimizer. This behavior can largely decrease the *abs* and *stress*, Figure 8.9.

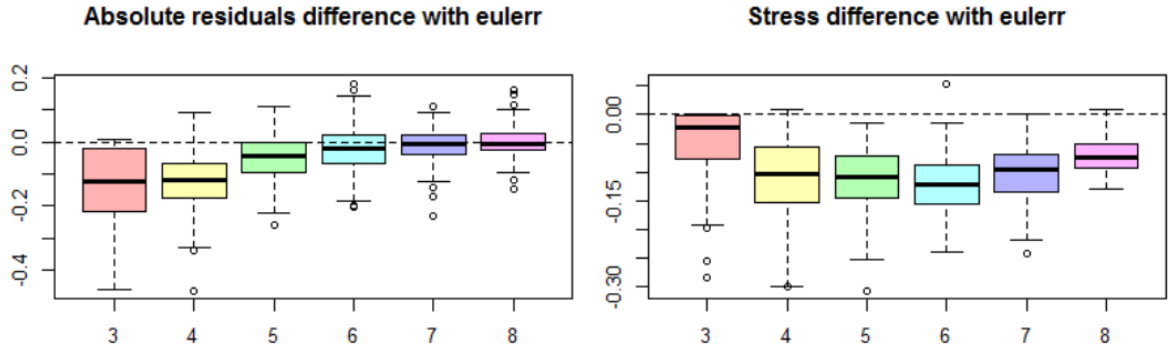


Figure 8.9: `eulerr()` - `vennplot()`

However, it will sacrifice the fit of other intersections, shown in Figure 8.10 and Fig-

ure 8.11. Sometimes, too many parameters can lead the program to fail providing Venn diagrams (like $\boldsymbol{\mathcal{I}}(\boldsymbol{S})_4^\star$) [26].

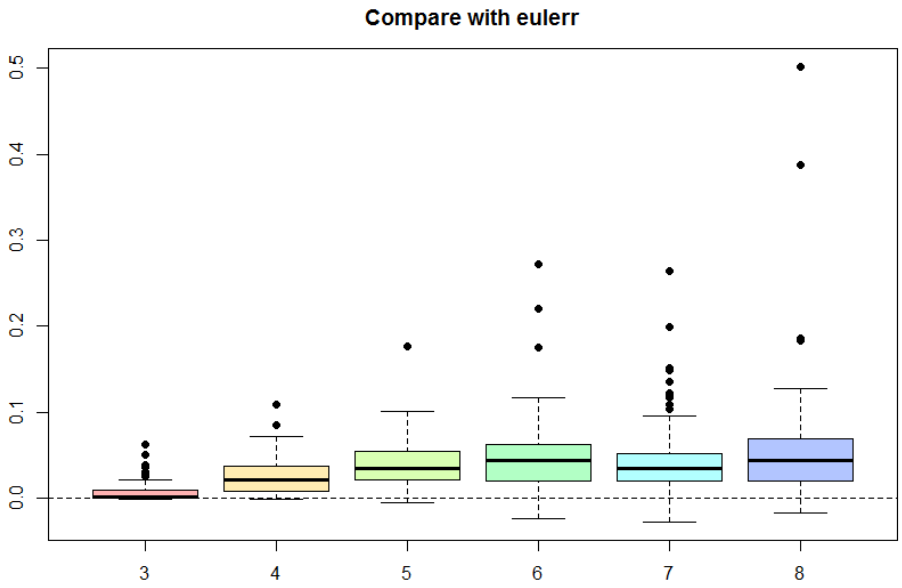


Figure 8.10: `eulerr()` - `vennplot()`, ϕ

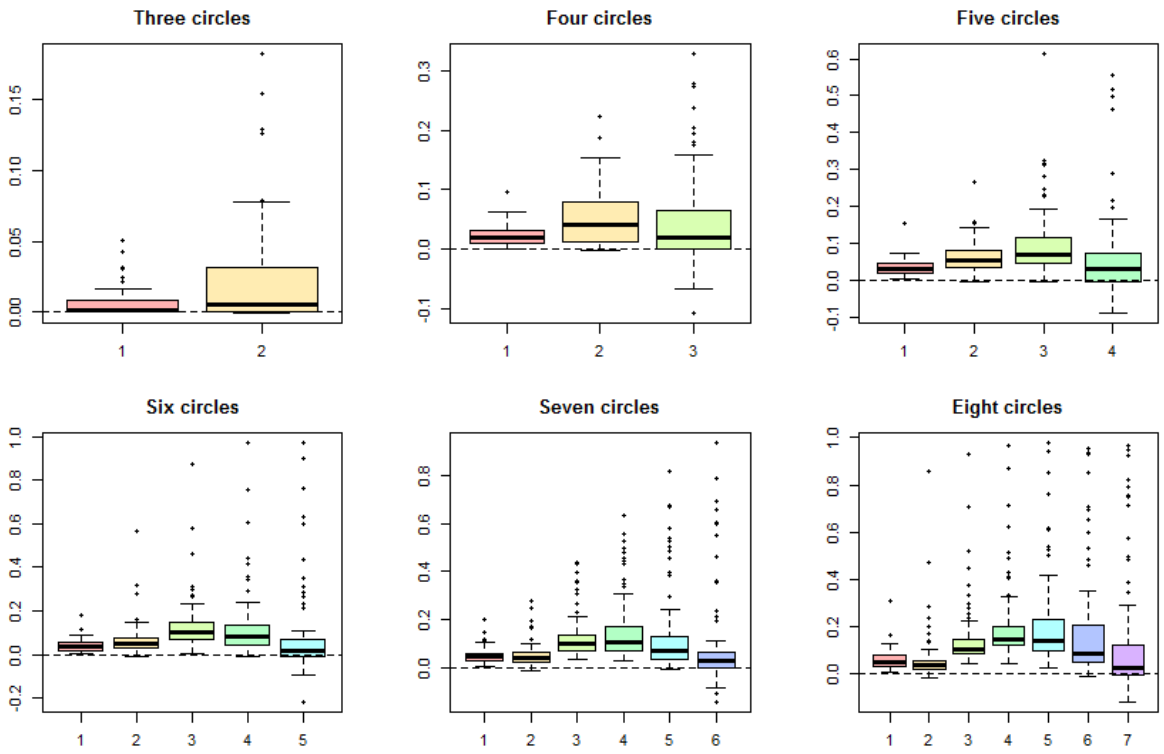


Figure 8.11: `eulerr()` - `vennplot()`, $stress(k)$ where $1 \leq k \leq m-1$

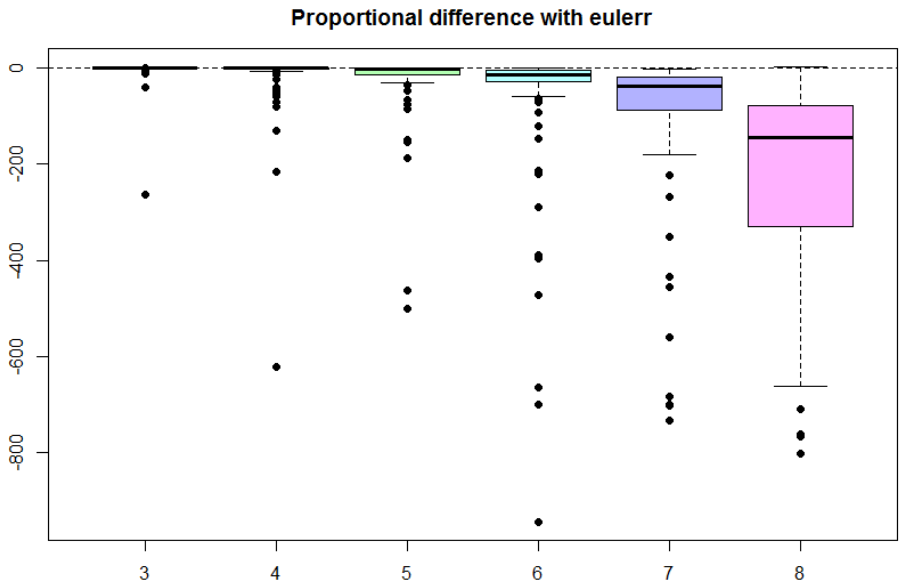


Figure 8.12: `eulerr()` - `vennplot()`, $stress(m)$

The experiment of multiple groups comparison is much more likely to terminate by `eulerr()`, hence we do not show the comparison.

Chapter 9

Discussion

Wilkinson's algorithm tends to give disjoint areas the same weight. Hold radii fixed and move centres. Frederickson's algorithm starts from distances with noticing the subsets and disjoint circles, and then, turns this problem (area proportional Venn diagram fit) into a multiple dimensional scaling minimizing problem. Our algorithm tries to balance two of them, seeking the minimum "MDS" and capturing a fine structure. Moreover, our model is the first one with noting groups automatically. The layout is impressive when there are multiple groups in one Venn diagram. The three-dimensional Venn diagrams can provide a nice visualization in case two dimensional Venn diagrams fail (like Figure 1.4 (a)).

For these three criteria (ψ, *abs* and *stress*), they can be taken as a reference, but not truth. The criteria ψ may be good for some configurations, but for real data, all tests may fail to reconstruct ($\psi \leq 10\%$). And this criteria is not bounded, if any disjoint areas overlay a tiny bit, ψ can go to ∞. *abs* and *stress* are bounded criteria, however, sometimes, they are not trustworthy either. There is an example on Frederickson's website, where $\mathbf{S}_6 = \{S_A, S_B\}$, the power set $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{S}})^\star_6$ with size $\mathbf{s}^\star_{\mathcal{P}} = [s^\star_A = 98, s^\star_B = 48, s^\star_{AB} = 0]$. Figure 9.1 shows the layout of `venneuler()`. The *abs* and *stress* are 0.0137, 7.3×10^{-5}, which should indicate a good fit. However, we can find the there is an intersection between S_A and S_B, very small but noteworthy [19]. So, this gives us a warning, a good Venn diagram must have a very small criteria, but a small criteria does not indicate a good Venn diagram.

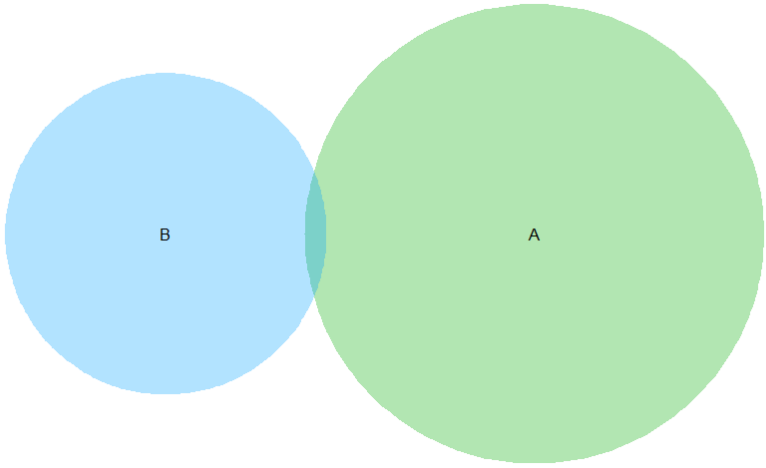


Figure 9.1: small *stress* but fit bad

Chapter 10

Appendix

- *Line Search* [6] for finding λ:

$$\lambda_+^{(n)} \leftarrow \lambda^{(n)} + \Delta$$

$$\lambda_-^{(n)} \leftarrow \lambda^{(n)} - \Delta$$

where $\Delta \in \Re^+$, a small step size. Fixing $\lambda_+^{(n)}$ and $\lambda_-^{(n)}$ to compute $stress(\lambda_+^{(n)})$ and $stress(\lambda_-^{(n)})$.

  - **if**$(\min(stress(\lambda_+), stress(\lambda_-), stress(\lambda^{(n)})) = stress(\lambda^{(n)}))$
    Return $\mathbf{C} \leftarrow \mathbf{C}^{(n)}$
  - **else**
    1. **if** $(\min(stress(\lambda_+), stress(\lambda_-), stress(\lambda^{(n)})) = stress(\lambda_+))$ which means shrinkage can decrease *stress*. Thus,

$$\lambda^{(n)} \leftarrow \lambda_+ \text{ and } stress(\lambda^{(n)}) \leftarrow stress(\lambda_+)$$

    (a) update λ

$$\lambda^{(n+1)} \leftarrow \lambda^{(n)} + \Delta$$

    (b) update count n

$$n \leftarrow n + 1$$

    (c) Fixing $\lambda^{(n)}$ and compute $stress(\lambda^{(n)})$
    (d) Repeat i to iii until $stress(\lambda^{(n-1)}) \leq stress(\lambda^{(n)})$

(e) Return $\mathbf{C} \leftarrow \mathbf{C}^{(n-1)}$ and $stress = stress(\lambda^{(n-1)})$

2. **else** which means expansion can decrease *stress*. Thus,

$$\lambda^{(n)} \leftarrow \lambda_{-}$$

And λ can be updated as follows:

$$\lambda^{(n+1)} \leftarrow \lambda^{(n)} - \Delta$$

The rest procedure is similar with (a). ii to v.

- *Nelder Mead Algorithm* [31] for finding μ:

1. Set

$$\begin{aligned} \mu_{+}^{(n)} &\leftarrow \mu^{(n)} + \Delta \\ \mu_{++}^{(n)} &\leftarrow \lambda^{(n)} + 2\Delta \end{aligned}$$

where $\Delta \in \Re^{+}$ and $\boldsymbol{\mu}$ can be defined as:

$$\boldsymbol{\mu}^{(n)} \leftarrow [\mu^{(n)}, \mu_{+}^{(n)}, \mu_{++}^{(n)}]$$

Thus we can get corresponding estimated distance matrix. Fixing $\lambda^{(1)}$, **stress** is:

$$\mathbf{stress}^{(n)} = [stress(\lambda^{(1)}; \mu^{(n)}), stress(\lambda^{(1)}; \mu_{+}^{(n)}), stress(\lambda^{(1)}; \mu_{++}^{(n)})]$$

2. The rest procedure is similar with "*Nelder Mead Algorithm* [31] for finding λ": fix $\lambda^{(1)}$, do *Loop* but replace $\boldsymbol{\lambda}^{(n)}$ to $\boldsymbol{\mu}^{(n)}$

3. Return

$$\mu = \frac{\sum(\boldsymbol{\mu}^{(n)})}{3}$$

and corresponding distance $[\widehat{d}_{ij}]$

Bibliography

[1] Pax6 gene, July 2014.

[2] D. Ashlock, E.Y. Kim, and L. Guo. Multi-clustering: Avoiding the natural shape of underlying metrics. *Smart Engineering System Design: Neural Networks, Evolutionary Programming, and Artificial Life*, 15:453–461, 2005.

[3] Vic Barnett, editor. *Interpreting multivariate data*. John Wiley & Sons, 1981.

[4] Margaret E. Baron. A Note on the Historical Development of Logic Diagrams: Leibniz, Euler and Venn. *The Mathematical Gazette*, 1969.

[5] Pierre Jerome Bergeron. sharkattackinfo.com, 2017.

[6] M. J Box, D Davies, and W. H Swann. *Non-linear optimization techniques*. Edinburgh: Published for Imperial Chemical Industries Ltd by Oliver Boyd, 1969, 1969.

[7] Allyson L. Byrd, Clay Deming, Sara K. B. Cassidy, Oliver J. Harrison, Weng-Ian Ng, Sean Conlan, NISC Comparative Sequencing Program, Yasmine Belkaid, Julia A. Segre, and Heidi H. Kong. Staphylococcus aureus and Staphylococcus epidermidis strain diversity underlying pediatric atopic dermatitis. *Science*, 2017.

[8] Hanbo Chen and Paul C Boutros. Venndiagram: a package for the generation of highly-customizable Venn and Euler diagrams in R. *BMC Bioinformatics*, 2011.

[9] W.H. Cherry and R.W. Oldford. Picturing Probability: the poverty of venn diagrams, the richness of Eikosograms. *(unpublished manuscript)*, 2003.

[10] S. Chow and P. Rodgers. Constructing area-proportional Venn and Euler diagrams with three circles. *Euler Diagrams Workshop*, 2005.

[11] S. Chow and F. Ruskey. Drawing Area-Proportional Venn and Euler Diagrams. *Graph Drawing*, 2003.

[12] William S. Cleveland and Robert McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods . *American Statistical Association*, 1984.

[13] William S. Cleveland and Robert McGill. Graphical Perception and Graphical Methods for Analyzing Scientific Data. *Science*, 1985.

[14] William S. Cleveland and Robert McGill. Graphical Perception : The Visual Decoding of Quantitative Information on Graphical Displays of Data . *Royal Statistical Society*, 1987.

[15] L. Euler. *Lettres a Une Princesse d'Allemagne, volume 2*. Charpentier, 1843.

[16] Leonhard Euler. Letters cii through cviii. In D. Brewster, editor, *Letters of Euler on different subjects in Natural Philosophy addressed to a German Princess. (Translated from Lettres à une Princesse d'Allemagne)*, volume I, pages 337–366. Harper and Brothers, New York, 1840 (1761 original).

[17] Ben Frederickson. Calculating the intersection area of 3+ circles, 2013.

[18] Ben Frederickson. A better algorithm for area proportional Venn and Euler diagrams, 2015.

[19] Ben Frederickson. Comparison with venneuler, 2015.

[20] John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 1973.

[21] Catherine Q. Howe and Dale Purves. Natural-scene geometry predicts the perception of angles and line orientation. *Proceedings of the National Academy of Sciences of the United States of America*, 102(4):1228–1233, 2005.

[22] C.B. Hurley and R.W. Oldford. *PairViz: Visualization using Eulerian tours and Hamiltonian decompositions*, 2011. R package version 1.2.1.

[23] Paul Jaccard. Distribution de la Flore Alpine dans le Bassin des Dranses et dans quelques regions voisines. *Bulletin de la Socieete vaudoise des sciences naturelles*, 1901.

[24] H. A. Kestler, A. Muller, J. M. Kraus, M. Buchholz, T. M. Gress, H. Liu, D. W. Kane, B. Zeeberg, and J. N. Weinstein. VennMaster: Area proportional euler diagrams for functional GO analysis of microarrays. *BMC Bioinformatics*, 2008.

[25] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Structure and evolution of blogspace. *Structure and evolution of blogspace*, 2004.

[26] Johan Larsson. An introduction to eulerr, 2017.

[27] Daniel P. LePage, Jason A. Metcalf, Sarah R. Bordenstein, Jungmin On, Jessamyn I. Perlmutter, J. Dylan Shropshire, Emily M. Layton, Lisa J. Funkhouser-Jones, John F. Beckmann, and Seth R. Bordenstein. Prophage WO genes recapitulate and enhance Wolbachia-induced cytoplasmic incompatibility. *Nature*, 2017.

[28] R Beau Lotto and Dale Purves. The empirical basis of color perception. *Consciousness and Cognition*, 11(4):609 – 629, 2002.

[29] Luana Micallef and Peter Rodgers. (eulerape: Drawing area-proportional 3-venn diagrams using ellipses. *PLOS*.

[30] B. Minaei-bidgoli, A. Topchy, and W. F. Punch. A comparison of resampling methods for clustering ensembles. In *the International Conference on Artificial Intelligence*, pages 939 – 945, 2004.

[31] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 1969.

[32] R. Wayne Oldford. Constraint-oriented programming with application to statistical graphics. In *Bulletin of the International Statistical Institute*, volume 54, pages IP–21/3 1–18, Cairo, Egypt, 1991. International Statistical Institute.

[33] R. Wayne Oldford. Self-calibrating quantile–quantile plots. *The American Statistician*, 70(1):74–90, February 2016.

[34] Richmond Wayne Oldford. *qqtest: Self calibrating quantile quantile plots for visual testing*, 2014. R package version 1.1.1.

[35] R.W. Oldford. Mental models and interactive statistics: Design principles. In *Computing Science and Statistics*, volume 31, pages 254–262. Interface Foundation of North America, 1999.

[36] D.J. Parkhurst and E. Niebur. Scene content selected by active vision. *Spatial Vision*, 16:125–154, 2003.

[37] E Polak and G Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Mathematical Modelling and Numerical Analysis*, 1969.

[38] Adam Rahman and Wayne Oldford. Euclidean distance matrix completion and point configurations from the minimal spanning tree. *(unpublished manuscript)*, 2016.

[39] Frank Ruskey and Mark Weston. A survey of venn diagrams. *THE ELECTRONIC JOURNAL OF COMBINATORICSl*, 2005.

[40] Jonathan R Shewchuk. Technical report. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, 1994.

[41] Elizabeth S. A. Sollars, Andrea L. Harper, Laura J. Kelly, Christine M. Sambles, Ricardo H. Ramirez-Gonzalez, David Swarbreck, Gemy Kaithakottil, Endymion D. Cooper, Cristobal Uauy, Lenka Havlickova, Gemma Worswick, David J. Studholme, Jasmin Zohren, Deborah L. Salmon, Bernardo J. Clavijo, Yi Li, Zhesi He, Alison Fellgett, Lea Vig McKinney, Lene Rostgaard Nielsen, Gerry C. Douglas, Erik Dahl Kjaer, J. Allan Downie, and David Boshier. Genome sequence and genetic diversity of European ash trees. *Nature*, 2016.

[42] S.S. Stevens. On the psychophysical law. *Psychological Review*, 1957.

[43] W. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 1952.

[44] Antony Unwin, Martin Theus, and Heike Hofmann. *Graphics of large datasets: visualizing a million*. Springer, 2006.

[45] John Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, X(5):1–18, July 1880.

[46] A. Weingessel, E. Dimitriadou, and K. Hornik. An ensemble method for clustering. In *Conference of Directions in Statistical Computing*, 2003.

[47] Wikipedia. List of alternative names for the human species.

[48] Leland Wilkinson. Exact and approximate area-proportional circular Venn and Euler diagrams. *IEEE Trans Vis Comput Graph*, 2012.

[49] Dongsheng Zhang, Mengchao Yu, Peng Hu, Sihua Peng, Yimeng Liu, Weiwen Li, Congcong Wang, Shunping He, Wanying Zhai, Qianghua Xu, and Liangbiao Chen. Genetic Adaptation of Schizothoracine Fish to the Phased Uplifting of the Qinghai - Tibetan Plateau. *Genetics*, 2017.