

Journal of Mathematics and the Arts

Visually pleasing knot projections

--Manuscript Draft--

Full Title:	Visually pleasing knot projections
Manuscript Number:	TMAA-2020-0042R1
Article Type:	Original Article
Keywords:	knot theory; knot projection; visual appearance
Abstract:	<p>In this short article I discuss the æsthetics of knot projections and introduce software which creates two dimensional knot diagrams optimized for visual appearance. Different aspects of knot appearance are discussed and a framework for objectively optimizing the visual appeal of a knot projection is given. I use the software to create a wide range of knot diagrams.</p>
Order of Authors:	Robin Hankin
Response to Reviewers:	file jma_rebuttal.pdf uploaded

Visually pleasing knot projections

Robin K. S. Hankin

Abstract. In this short article I discuss the aesthetics of knot projections and introduce software which creates two dimensional knot diagrams optimized for visual appearance. The software is in the form of a documented and self-contained suite of functionality written in the R programming language (that is, a “package”): **knotR**. The package leverages the graphical capabilities of the popular vectorised graphics software **inkscape**. Different aspects of knot appearance are discussed and a framework for objectively optimizing the visual appeal of a knot projection is given. I use the software to create a wide range of knot diagrams.

Mathematics Subject Classification (2010). Primary 57K10-XX; 32-XX.

Keywords. Knot projections, Bezier curves, Multidimensional optimization.

1. Introduction

A *mathematical knot* is a smooth, unoriented embedding of a circle \mathbb{S}^1 into \mathbb{R}^3 [1, 2]. Two knots are said to be equivalent if the embedding of one can be continuously deformed to the other; if so, there is a homeomorphism $h: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ which takes one embedded circle to the other. In the art world, knot theory has applications to the study of dance [3], sculpture, [4, 5], and many other branches of mathematical art [6].

It is common to present a knot using diagrams such as Figure 1, in which a two-dimensional projection of the knot is given with broken lines indicating where one strand passes over another. Knot diagrams have a long history: Przytycki [7] discusses an intricate braid drawn around 1700 BC; they figure prominently in *The Book of Kells* and the Lindisfarne gospels, graphic art produced around 800 AD [8]. Such artwork makes extensive use of symmetry [9], and its visual appeal persists to the present day [10]. More recent examples would include bobbin lace [11], in which many threads are

braided together to form a loose fabric. The enduring popularity of such fibre arts attests to the aesthetic of knots and braids in the modern world.

Knot diagrams are ubiquitous in the modern discipline of mathematical knot theory. In the nineteenth century we see Tait [12] publishing an early systematic study of knots including a beautiful plate of elegantly hand-drawn cursive knot diagrams. The tradition continues into the twentieth century with Rolfsen [13] presenting an appendix containing many knot and link projections. Rolfsen’s diagrams are used widely today: Adams [2] reproduces his work extensively, as do many more recent works.

1.1. Computer-generated knot visualisations

Online resources are becoming more widely available, with one prominent example being the *Knot Atlas* [14] presenting extensive knot tables following Rolfsen [13]—for knots up to 10 crossings—and Thistlethwaite [15], for knots of 11 crossings.

Symbolic software such as Mathematica and Maple can be used to produce representations of knots, typically in the form of nonstandard embeddings of the torus into \mathbb{R}^3 ; but as discussed below this approach is suboptimal from the perspective of producing a visually pleasing projection.

Visual appeal of knot projections appears to be important to contemporary mathematicians. Recent work by Taalman (writing as mathgrrl) [16], for example, shows that large amounts of effort have been expended by serious mathematicians making representations of knots attractive. Bosch [4], taking a different approach to knot representations, presents artwork that utilises high-dimensional computer optimization, as here.

No discussion of computer-generated representations of knots would be complete without mentioning the tour de force that is *KnotPlot* [17,18] which includes extensive functionality for visualising knots. Much of the visualization considers (nonstandard) embeddings of a torus in \mathbb{R}^3 ; the configuration is “relaxed” in such a way as to produce a smoother output. *KnotPlot* has two major display modes: any knot may be displayed in either a “beads and sticks” mode or in a “smooth tubes” mode. The software does render knot diagrams but the author states [p77] that “knots with higher crossing numbers (greater than six or seven) will not produce good 2D projections when fully relaxed”, and admits that some of the diagrams are “a bit messy”.

1.1.1. Knot theory in R. The R programming language [19], while usually associated with statistical analysis and data processing, is also a suitable tool for producing visually attractive images. Generative art [that is, use of an algorithm, with or without a random component, in a creative process] is possible in R [20]. More specifically to knot theory, the *Rknots* package [21,22] presents software for folded protein structures, and includes 3D renderings of commonly encountered knotted strands.

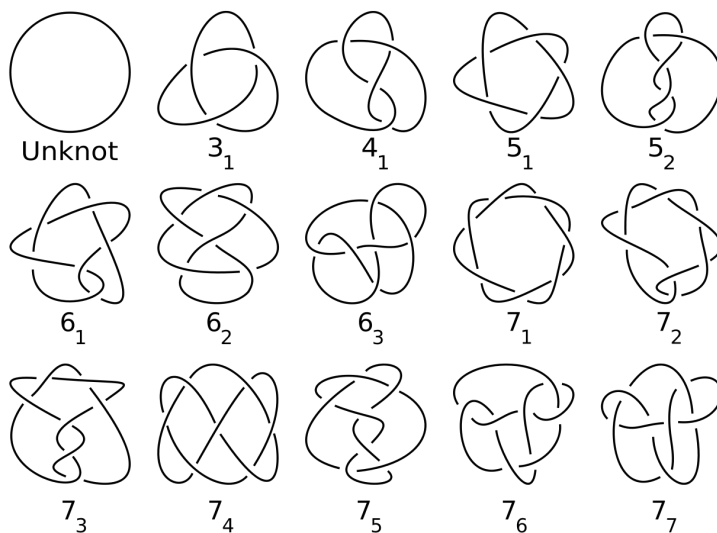


FIGURE 1. A table of prime knots up to seven crossings, labels following [23]. Image taken from [24].

2. The knotR package: Rationale

Consider Figure 1 from an æsthetic perspective; the diagrams are representative of those available under a free license. However, these diagrams are not suitable for high-quality artwork such as posters: they are not vectorized. Many of the underlying knots possess a line of symmetry (at least, the diagrams do if the breaks are ignored), which is not present in the visual representation. Also, several of the strands cross at acute angles. The diagram for knot 7_3 , for example, contains kinks and abrupt changes of curvature which distract from the underlying topological form.

Such considerations suggest that knot diagrams might be produced by minimization of some objective function that quantifies the visual inelegance of a knot diagram. The precise nature of such an objective function is, of course, a subjective choice but one might require the following desiderata:

- Curvature to be as smoothly changing as possible, with limited maximal curvature
- Strands to cross at right angles
- Crossing points to be separate from one another
- Any symmetry desired in the knot should be enforced exactly, and be visually apparent

Knot diagrams may be created using vectorized graphics software such as inkscape [25]: one specifies a sequence of control points, then interpolates between these points to create a knot diagram with the appropriate topology (inkscape is a widely-used system available under the GPL). One way

of smoothly interpolating between specified points is Bezier curves [26]. A Bezier curve is a visually pleasing polynomial path that can be used to specify the path of a knot projection; here cubic Bezier curves are used. The software discussed here [27] allows one to specify a knot in terms of its Bezier control points within inkscape, import the object into R [19], and then to use numerical optimization techniques to improve the visual appearance of the knot.

One plausible technique for creating knot projections is to consider a two-dimensional projection of a knot's embedding $\mathbb{E} \in \mathbb{R}^3$. However, this approach often results in poor visual appearance: cusps or other displeasing effects can occur.

3. The package in use

The software presented here is written in the “R” programming language [19], which is usually associated with statistical analysis and data processing. However, R possesses a number of features such as object-oriented programming and high-dimensional optimization which make it suitable for producing knot diagrams. In this section, I give workflows for creating two simple knots: firstly 7_6 , followed by the figure-of-eight knot 4_1 which requires imposition of symmetry constraints.

The first step is to create a closed curve in inkscape that shows the rough outline of the knot (Figure 2 shows a screenshot of `7_6_first_draft.svg`, supplied with the package). Note that this file contains only the knot *path*; the over and under information is to be added later. Irvine *et. al.* [11] use the terms *drawing* and *braid word* for analogous concepts in their work on bobbin lace patterns.

Here, knot paths are required to have Bezier handles that are symmetrically placed with regard to nodes (this is a user-settable option in inkscape). One consequence of this design choice is that radius of curvature is not matched exactly across a node; the path is not G2 smooth in CAD terminology. However, if we can ensure that the radius of curvature changes only by a small amount, visual continuity—in the sense of absence of visible interruptions to a smoothly evolving strand—is preserved. At a crossing point it is important that both the unbroken and broken strands (designated “overstrand” and “understrand” respectively) are smooth. We would like strand crossing points to be far from nodes, as this ensures visual continuity of strands at crossing points, especially the understrand.

The knot shown in Figure 2 is clearly suboptimal: even though the nodes are connected by Bezier curves which are individually smooth, the path as a whole is visually disjointed as it has places where the radius of curvature changes abruptly.

Although it is possible in principle to improve the visual appearance of the knot path by hand in inkscape, this is a surprisingly difficult and frustrating task. In order to remedy the flaws of the diagram using an automated

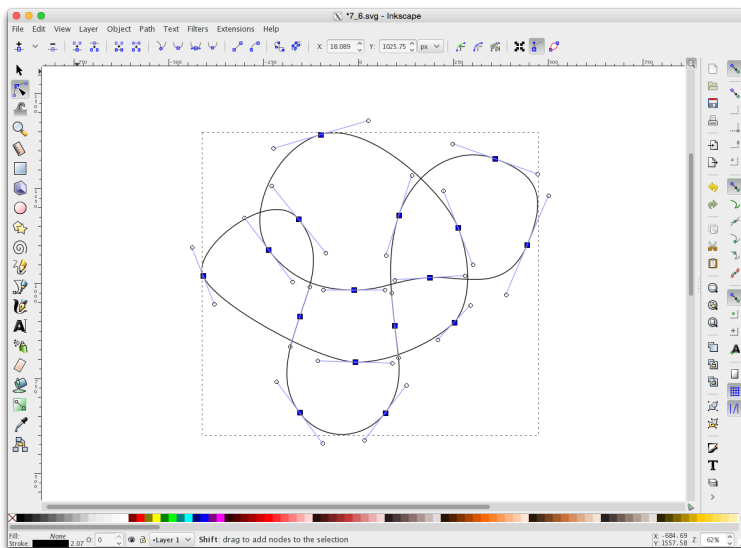


FIGURE 2. Screenshot of inkscape setup for knot 7_6 . Nodes are shown as squares, handles as small circles, symmetrically placed on either side of nodes

system, we first read the .svg file into R using the `reader()` function; a typical R session follows:

```
> library(knotR)
> k76 <- reader(system.file("7_6_first_draft.svg", package="knotR"))
> head(k76)
```

	x	y
[1,]	-98.81963	339.81898
[2,]	-223.87754	303.35366
[3,]	-299.84521	121.06064
[4,]	-236.36319	36.35340
[5,]	-172.88118	-48.35384
[6,]	-92.86186	-69.78212

Object `k76` is stored as an object of class `inkscape`: a two-column matrix with rows corresponding to the node and handle positions of the inkscape path. Above we see only the first six lines of the object. Inkscape representation has a certain amount of redundancy as knot paths have handles which are symmetrically placed with respect to nodes; also, the first node is the same as the last for the loop is closed. The package can coerce `inkscape` objects to other forms, specifically `minobj` objects, which contain no redundancy (the position of each node, as well as one of the handles, is stored); or `controlpoints` objects, which allow for easy construction of Bezier interpolation between nodes.

```

> k76_rough <- reader(system.file("7_6_first_draft.svg",package="knotR"))
> knotplot2(k76_rough, seg=TRUE)

```

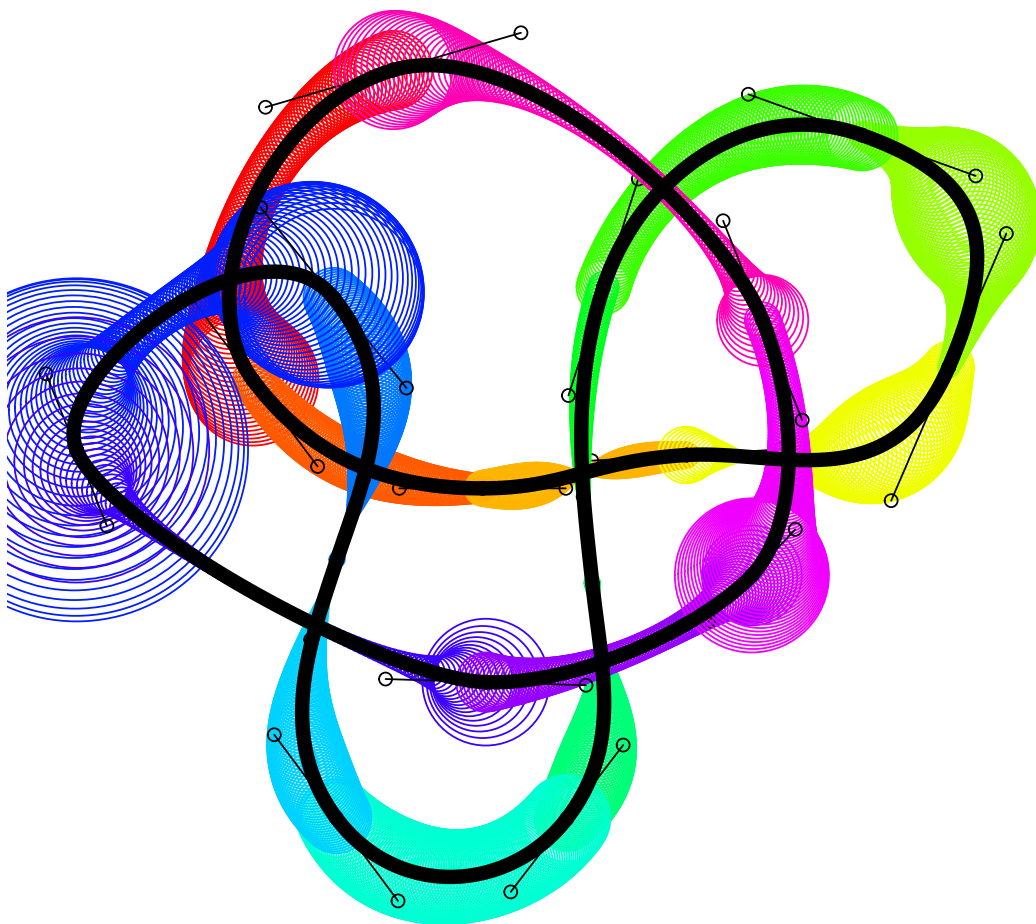


FIGURE 3. The *path* of (unoptimized) knot 7_6 , showing Bezier handles as thin straight lines and circles. The coloured circles have a radius proportional to the curvature (that is, the reciprocal of the radius of curvature) along the strand. Colouring is arbitrary, one color for each Bezier segment; solid blotches result from densely overlapping circles. Note large curvature at loop on left

```
knotplot2(k7_6, text=TRUE, lwd=1, circ=0, rainbow=TRUE)
```

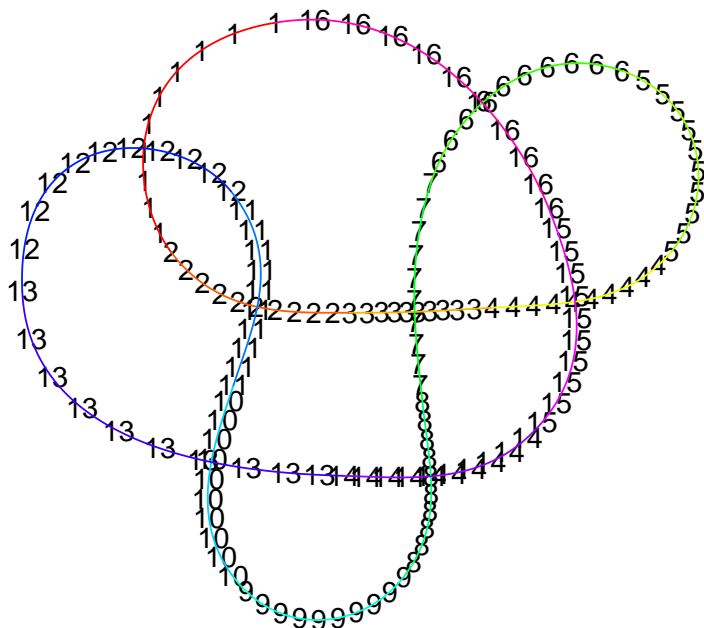


FIGURE 4. Knot 7_6 with strands numbered so that the sense of the crossings can be established. For example, strand 7 passes over strand 3. In the package, this fact is specified by the `overunder` object, as discussed in the text, having a row 7 3 (and not 3 7 which would be the crossing having the opposite sense)

To beautify it we need to specify a function of the path that quantifies its displeasingness, and then minimize this objective function using numerical methods. In the package, minimization is performed using either Nelder-Mead [28] or a Newton-type method [29] (gradient information is not available). The optimization typically proceeds over \mathbb{R}^{50} and in the context of the package, the methods appear to be broadly comparable.

Two examples of desiderata for such an objective function might be to keep the strands crossing at right angles, and the overall bending energy. These are evaluated in the package by functions `total_crossing_angles()` and `total_bending_energy()` respectively:

```
> b <- as.controlpoints(k76_rough)
> total_crossing_angles(b)
```

```
[1] 0.3145033
```



```
> total_bending_energy(b)
```

```
[1] 0.1276257
```

The knots supplied in the package minimize a weighted sum of these and other badnesses¹. The weightings for the various badnesses are, of course, subjective; but the system discussed here allows the user to change the weightings used and compare results. I present a short discussion in section 3.3 below. Numerically, the badnesses are evaluated by function `badness()`:

```
> badness(k76_rough)
```

```
[1] 6.16279
```

This function may be minimized by numerical optimization:

```
> o <- nlm(badness, as.knotvec(k76_rough))
```

```
> k7_6 <- as.minobj(o$estimate)
```

```
> badness(k7_6)
```

```
[1] 3.550152
```

(the above takes about an hour of CPU time: it is optimizing a function of 64 real variables, and the objective function takes a few seconds to evaluate). However, the result is visually smoother and thus arguably more attractive (Figure 5).

To specify the senses of the knot's crossings, we create an overunder object which is a two-column matrix:

```
> ou76 <- matrix(c(
+   12,01,
+   02,11,
+   07,03,
+   04,15,
+   16,06,
+   14,08,
+   10,13
+   ),byrow=TRUE,ncol=2)
```

With reference to Figure 4, each row of `ou76` corresponds to a crossing; the first element gives the overstrand and the second the understrand; thus strand 12 passes over strand 1, strand 2 passes over strand 11, and so on. The result is shown in figure 6.

¹Function `badness()` includes various “housekeeping” badnesses which are used to make sure that the minimum found by `nlm()` is topologically identical to the starting configuration. Function `non_crossing_strand_close_approach_badness()`, for example, makes non-crossing strands “repel” one another so as not to introduce spurious intersections.

```
knotplot2(k7_6)
```

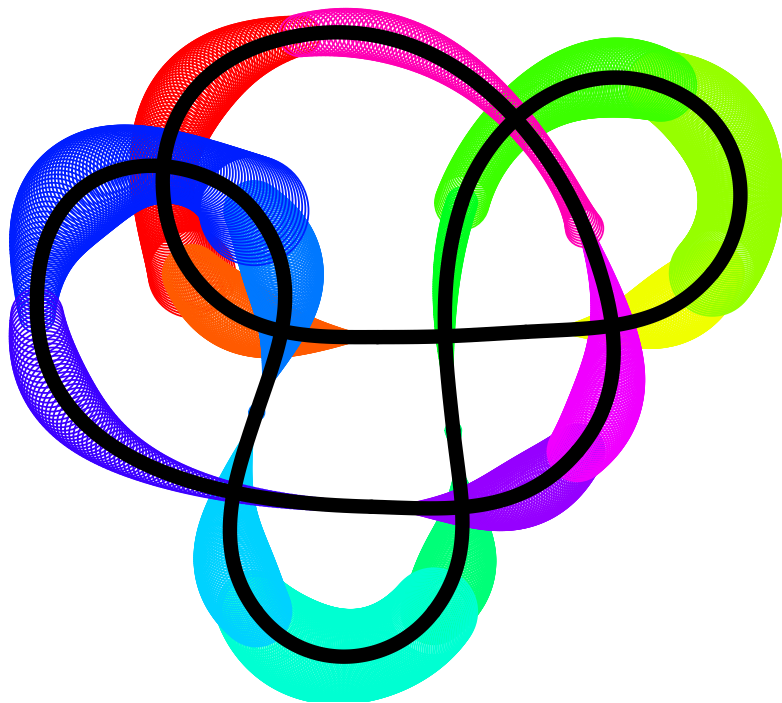


FIGURE 5. Knot 7_6 , post-optimization

3.1. Symmetry

If a knot diagram can be drawn with a particular symmetry, one common response is to demand that this symmetry be respected exactly. Bosch [4], for example, considered rotational symmetry to be sufficiently important to impose five-fold rotational symmetry in one of his works; similar comments apply to mirror symmetry. Many of the knots in Figure 1 have an axis of symmetry, or possess rotational symmetry; some have both and thus have a dihedral symmetry group. The package implements Bosch-type symmetry:

```
knotplot(k7_6)
```

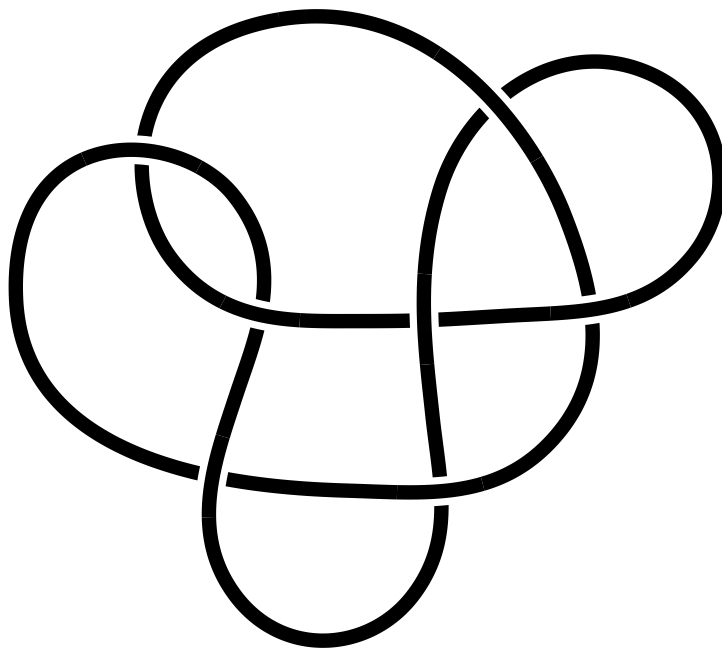


FIGURE 6. Knot 7_6 , post-optimization with breaks indicating underpassing strands

one can impose these symmetry relations on knots, and optimize the resulting symmetrical knot.

Minimizing the badness is not entirely straightforward on account of the induced redundancy, which is characterized using a symmetry object specific to the knot under consideration. However, symmetry constraints do reduce the dimensionality of the optimization problem.

I will consider the figure-of-eight knot 4_1 as an example. Using Figure 7, top left, as reference, the appropriate symmetry object is defined as follows:

```

> fig8 <- reader(system.file("4_1_first_draft.svg",package="knotR"))
> Mver8 <- matrix(c(
+   02,03,
+   09,07,
+   05,11,
+   10,06
+   ),ncol=2,byrow=TRUE)
> sym8 <- symmetry_object(fig8, Mver=Mver8, xver=8)

```

(Matrix `Mver8` specifies that nodes 2 and 3 are symmetric, as are nodes 9 and 7, and so on; `xver=8` forces node 8 to be on the axis of symmetry). The results are shown in Figure 7. Note that 4_1 may be rendered in a form that has two mirror lines, and this is available in the package as object `k4_1a`.

3.2. Rotational symmetry

Consider knot 5_1 . This knot has D_5 symmetry with five mirror lines. The package includes functionality to impose appropriate symmetry constraints. Using Figure 8 as reference, we have:

```

> sym51 <- symmetry_object(k5_1,
+   Mver = cbind(11,13),
+   xver = c(2,12),
+   Mrot = rbind(
+     c(12,04,16,08,20),
+     c(13,05,17,09,01),
+     c(11,03,15,07,19),
+     c(02,14,06,18,10)
+   ))

```

Thus, using the same notation as before, nodes 11 and 13 are symmetrical about the vertical axis, nodes 2 and 12 are on the vertical axis. The `Mrot` argument specifies sets of nodes that map to themselves under rotation. The top line of `Mrot` indicates that nodes 12, 4, 16, 8, and 20 are concyclic. An example of a rotationally symmetric knot is given in Figure 11.

3.3. Subjective choice for weighting

One of the benefits of the `knotR` software is that it allows the relative importance of the different aspects of appearance to be assessed. In the package, the badness weightings may be altered easily, and in this way we can investigate the visual impact of the badness components. The default weightings were originally chosen as a bland compromise, but it is easy to place greater weight on one or other component. Figure 9 shows the same knot optimized using different weightings; we see the effect of imposing right-angled strand crossing angles on the remainder of the knot. Readers will have to judge for themselves which version they prefer.

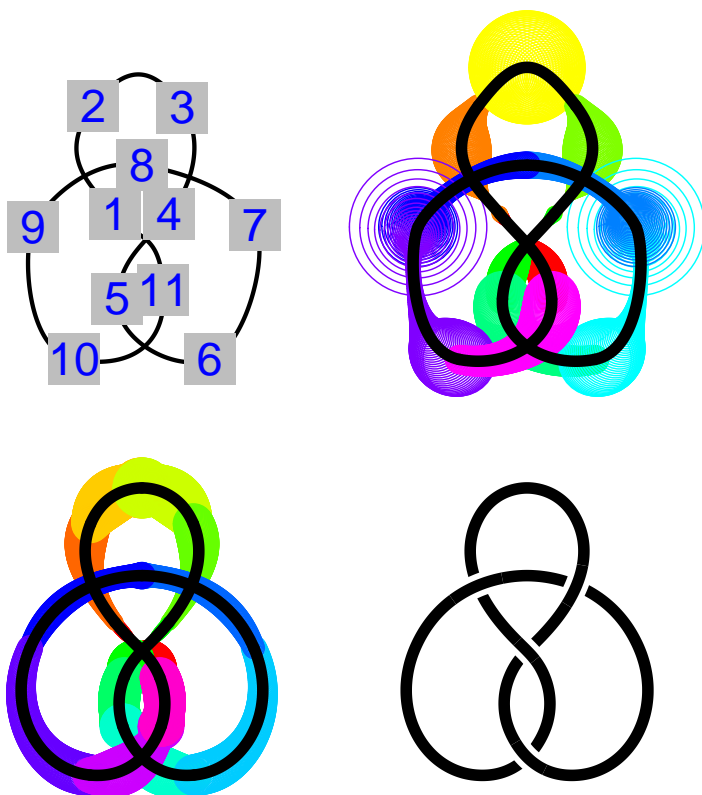


FIGURE 7. Figure eight knots drawn using different plotting methods. Top left, knot path with node numbers shown in order to facilitate definition of the symmetry object; top right, result of symmetrizing the rough path; lower left, the optimized knot with imposed vertical symmetry, with curvature plotted; lower right, knot plotted with overstrand and understrand indicated using line breaks

4. Conclusions and further work

The `knotR` package allows the user to create rough diagrams of knots using the inkscape suite of software, and subsequently polish up such diagrams in terms of a customizable objective function using numerical optimization techniques. Further work might include functionality to deal with links. In the broader context of optimisation in art, we observe that numerical optimisation techniques can produce aesthetically pleasing results, an observation that might find uptake by graphic artists.

```
> knotplot2(k5_1,node=TRUE,width=FALSE)
```

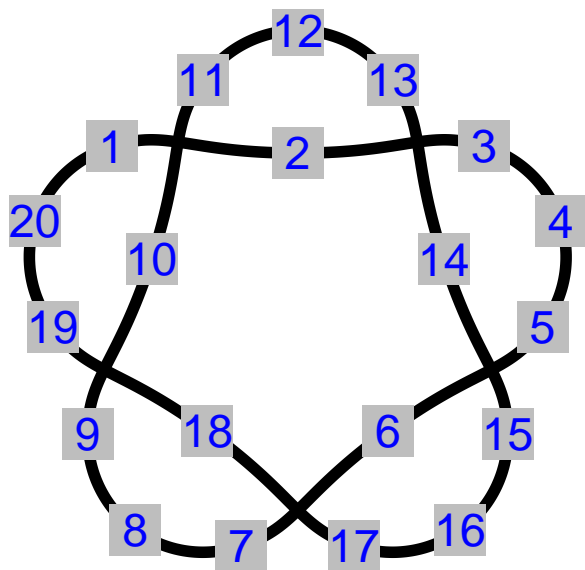


FIGURE 8. Knot 5₁ shown with node numbers

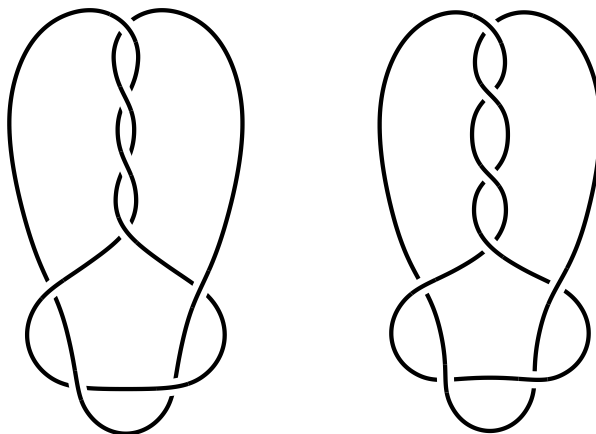


FIGURE 9. Comparison between optimized knot projections for 8_3 using different badness weightings. On the left we see the default knot, and on the right we see the result of increasing the angle-crossing penalty by a factor of 100, effectively forcing all strand crossing angles to be $\pi/2$. Which one is preferable is, of course, a subjective choice

5. Gallery

There now follows a selection of pleasing knot diagrams taken from datasets provided with the package, illustrating the relative ease with which knot diagrams may be created and optimized. Figure 10 shows two forms of a 10-crossing knot once erroneously considered to be topologically distinct; figure 11 shows an ornamental knot with exact five-fold symmetry (but not dihedral symmetry); and figure 12 shows all prime knots with eight or fewer crossings, following Rolfsen [13]. The package includes all knots to nine crossings.

```
par(mfcol=1:2)
knotplot(perko_A)
knotplot(perko_B)
```

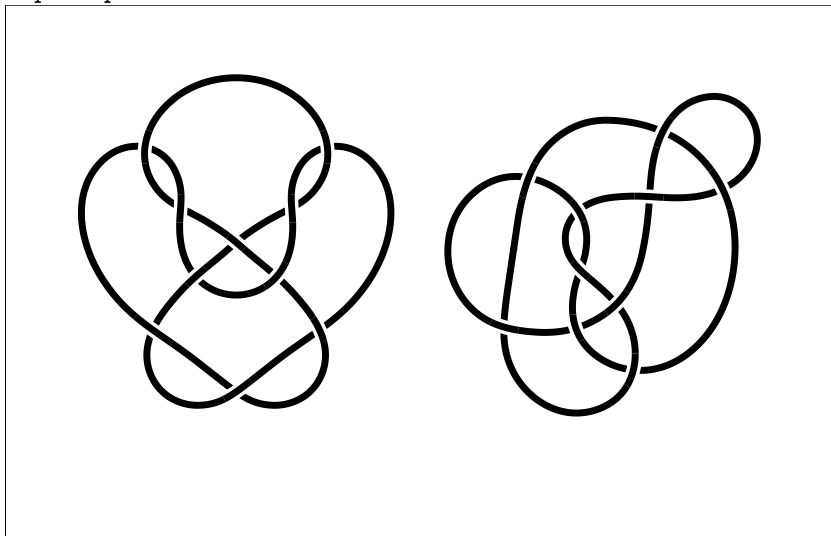


FIGURE 10. Two representations of knot 10_{125} , known as the Perko Pair. The software requires the user to specify the symmetry (mirror or rotational) of a knot projection and has no notion of topological invariance of a knot

References

- [1] V. Manturov. *Knot Theory*. Chapman and Hall, 2004.
- [2] C. C. Adams. *The knot book: an elementary introduction to the mathematical theory of knots*. American Mathematical Society, 2004.
- [3] M. Khorami. Space harmony: a knot theory perspective on the work of Rudolph Laban. *Journal of Mathematics and the Arts*, 14(3):239–257, 2020.
- [4] R. Bosch. Simple-closed-curve sculptures of knots and links. *Journal of Mathematics and the Arts*, 4(2):57–71, 2010.


```
> knotplot(ornamental20,gap=15)
```

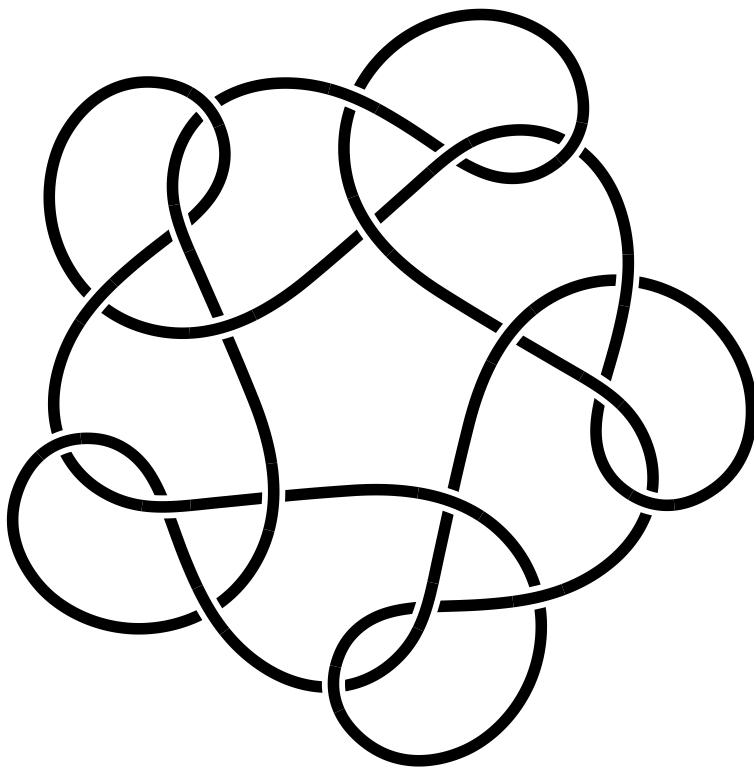


FIGURE 11. An ornamental knot exhibiting fivefold rotational symmetry C_5 ; note the absence of mirror symmetry which would impose D_5

- [5] A. Widmark. Stixhexaknot: a symmetric cylinder arrangement of knotted glass. *Journal of Mathematics and the Arts*, 14(1-2):167–169, 2020.
- [6] G. W. Hart and N. Jonoska. Knotting mathematics and art. *Journal of Mathematics and the Arts*, 2(1):47–51, 2008. Conference in low-dimensional topology and mathematical art, University of South Florida, Tampa, FL, 1-4 November 2007.
- [7] J. H. Przytycki. Classical roots of knot theory. *Chaos, Solitons & Fractals*, 9(4/5):531–545, 1998.
- [8] G. Bain. *Celtic art: the methods of instruction*. MacLellan & Co. (reprinted by Dover), 1973.
- [9] P. R. Cromwell. The distribution of knot types in Celtic interlaced ornament. *Journal of Mathematics and the Arts*, 2(2):61–68, 2008.

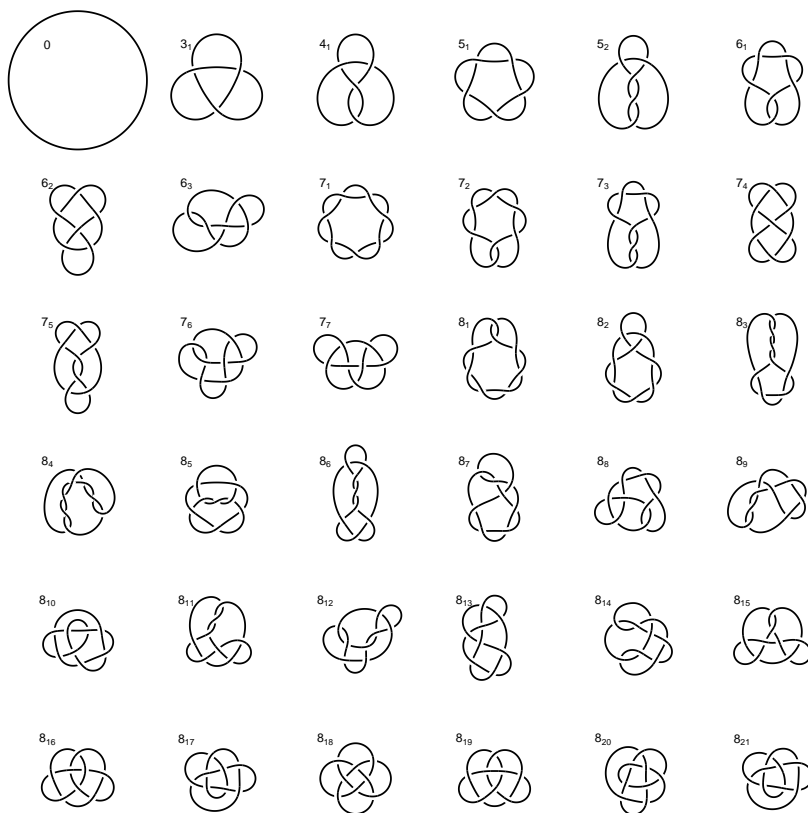


FIGURE 12. All prime knots with 8 or fewer crossings, notation following Rolfsen [13]

- [10] R. Antonsen and L. Taalman. Categorizing Celtic knot designs. In *Bridges 2021 Conference Proceedings*, pages 87–94, (Virtual), August 2021. The Bridges Organization.
- [11] V. Irvine, T. Biedl, and C. S. Kaplan. Quasiperiodic bobbin lace patterns. *Journal of Mathematics and the Arts*, 14(3):177–198, 2020.
- [12] P. G. Tait. The first seven orders of knottiness. *Transactions of the Royal Society of Edinburgh*, 32:327–342, 1884.
- [13] D. Rolfsen. *Knots and links*. Publish or Perish Press, 1976.
- [14] The knot atlas. online. http://katlas.org/wiki/Main_Page.

- [15] J. Hoste, M. Thistlethwaite, and J. Weeks. The first 1701936 knots. *The Mathematical Intelligencer*, 20(4):33–48, 1998.
- [16] L. Taalman. Mathematical art galleries. Online, 2019.
- [17] R. G. Scharein, 1997.
- [18] R. G. Scharein. *Interactive topological drawing*. PhD thesis, The University of British Columbia, 1998.
- [19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.
- [20] Katharina Brunner. *generativeart: Create Generative Art with R*, 2021. R package version 1.0.
- [21] F. Comoglio and M. Rinaldi. A topological framework for the computation of the homfly polynomial and its application to proteins. *PLoS ONE*, 6(4), 2011.
- [22] Federico Comoglio and Maurizio Rinaldi. *Rknots: Topological Analysis of Knotted Proteins, Biopolymers and 3D Structures*, 2016. R package version 1.3.2.
- [23] J. W. Alexander and G. B. Briggs. On Types of Knotted Curves. *Annals of Mathematics*, 28(1/4):562–586, 1926.
- [24] Wikipedia contributors. Knot theory — Wikipedia, the free encyclopedia, 2021. [Online; accessed 16-September-2021].
- [25] Inkscape Project. Inkscape software. <https://inkscape.org>.
- [26] Aaron Olsen. *bezier: Bezier Curve and Spline Toolkit*, 2014. R package version 1.1.
- [27] Robin K. S. Hankin. *knotR: Knot Diagrams using Bezier Curves*, 2020. R package version 1.0-4.
- [28] J. A. Nelder and R. Mead. A simplex algorithm for function minimization. *Computer Journal*, 7:308–313, 1965.
- [29] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

Robin K. S. Hankin

55 Wellesley Street East, Auckland 1010, New Zealand

e-mail: hankin.robin@gmail.com

Rebuttal to second editorial review: overall response

(Below, the reviewer's comments are in black, and my replies to the issues are in blue. I have indicated changes to the manuscript where appropriate.

Short story: I have accommodated all the comments with rewording. The resulting document is, I believe, stronger and more scholarly than before and I recommend it to you.

Detailed rebuttal: Reviewer #1

This short article is a tutorial example of how to use Inkscape and KnotR to beautify 2-dimensional knot diagrams found in the knot tables. The cost function used in optimizing a given diagram tries to minimize the overall bending energy of the knot curve, preserve all shown 2D symmetries, and render all strand crossings as close to 90 degrees as possible. The paper does not discuss the underlying mathematics by which this optimization is achieved, but relies on the power built into the software packages mentioned. Also, the user has to characterize explicitly the desired symmetries; there is no “magic” power that discovers the maximal possible symmetry that a given knot can assume. This exemplified by Figure 7, where the Figure-8-knot is drawn with a single mirror axis, even though this same knot can readily be drawn with two mirror lines; – and also in Figure 9 where a symmetrical and an asymmetrical rendering of the same knot is presented.

This is a good characterisation of the submission, but the comment that there is no “magic” power ascertaining maximal possible symmetry that a given knot may assume is a good one, and one that I had not explicitly realised. I have added a brief discussion of this issue to the caption of Figure 9 that discusses the Perko pair:

OLD text “Two representations of knot 10_{125} , known as the Perko Pair”

NEW text “Two representations of knot 10_{125} , known as the Perko Pair. The software requires the user to specify the symmetry (mirror or rotational) of a knot projection and has no notion of topological invariance of a knot”

The referee observes that 4_1 has a symmetrical rendering with two mirror lines and this is available in the package, as object `k4_1a`. I have added a description of this to the manuscript.

This paper is clearly useful to someone who specifically wants to clean up a given rendering of a particular knot, but it offers little to readers who are not familiar with Inkscape and KnotR.

In any software, one makes use of previously written work. One rarely writes an operating system from scratch, for example. The knotR package leverages the design capabilities of inkscape in, I think, a useful and informative way; and it further uses the numerical abilities of R in an efficient and natural way.

Detailed comments about the presentation:

Abstract: It should mention the software packages that are used in this paper.

Done. The abstract is now completely rewritten.

P1, line 46: “Knot theory” – ? – Is this an incomplete reference?

Corrected, this was a typo

P2, line 43: “uniform” curvature is not possible, since many knots have inflection points. It is better to ask for ‘smoothly changing’ curvature, and to limit maximal curvature.

Corrected:

OLD text Curvature to be as uniform as possible

NEW text Curvature to be as smoothly changing as possible, with limited maximal curvature

P2, line 46: What does it mean for symmetry to be “present” in the knot?

OLD text Any symmetry present in the knot should be enforced exactly, and be visually apparent

NEW text Any symmetry desired in the knot should be enforced exactly, and be visually apparent

P3, line 31: Define “visual continuity”. Are you referring to tangent continuity, $G1$ –, or $G2$ – continuity?

The path does not have $G2$ continuity because the radius of curvature has a discontinuity at Bezier nodes. But the change of curvature should be small, I have added a short discussion to clarify the issue.

P3, line 33: The “understrand” has not been designated yet.

fixed

Figure 2: Specify what order Bezier curves are being used.

fixed

P4, line 18: A gentle introduction to “R” would help to make the following easier to understand.

Fixed, a short introductory passage and a reference to R core now added

P4, lines 27-32: What is the meaning of this? - Six points defining two handle pairs? (Why the extreme precision with 8 digits?)

Only the first six lines are shown for brevity, I have changed the text accordingly: “Above we see only the first six lines of the object”

Figure 3: What is the color assignment? - one color for each Bezier segment? How many circles are drawn for each segment? (Say that solid blotches result from densely overlapping circles.)

The original caption was lacking detail.

OLD text “The *path* of (unoptimized) knot 7_6 , showing Bezier handles as thin straight lines and circles. The coloured circles have a radius proportional to the curvature (that is, the reciprocal of the radius of curvature) along the strand; note large curvature at loop on left”

NEW text “The *path* of (unoptimized) knot 7_6 , showing Bezier handles as thin straight lines and circles. The coloured circles have a radius proportional to the curvature (that is, the reciprocal of the radius of curvature) along the strand. Colouring is arbitrary, one color for each Bezier segment; solid blotches result from densely overlapping circles. Note large curvature at loop on left”

Each segment has one circle per control point (which defaults to $n = 100$) but this is user-settable.

Figure 4: Why is this required? – Does the user have to specify the type of crossing via an “overunder” object? – (Lines 53-62?)

The `overunderobject` specifies which of two strands at a crossing is the overstrand and which the understand. I have added a definition of this and a brief discussion in the figure caption.

P10, line 17: “... a vertical line of symmetry.” - Better to say that this figure has D_5 symmetry with five mirror lines.

Done.

P11, line 26: It seems that all the functionality needed to deal with links is already present in the software packages mentioned, as long as one can let the program go around more than one single closed loop.

I have spent many many hours pondering how to deal with links. As the referee says, much of the required functionality is (in principle) already present in the software. Each component would have its own badness, and in addition there would be $n(n-1)/2$ inter-component interaction terms measuring features such as closeness between non-intersecting strands of different components. Several problems stood out: firstly, the problem of inter-component intersection points. Currently this is a (symmetrical) Boolean matrix with rows and columns corresponding to strands, and entry (i, j) being whether or not strand i intersects with strand j . The link generalization would be a symmetric matrix of intersection matrices: an object with four indices and entry (i, j, k, l) indicating whether strand j of component i intersects strand k of component l . I found this object to be unwieldy to work with and difficult to manipulate. The second problem was the imposition of symmetry. Sometimes one might desire that some components of a link have a particular kind of symmetry and others to have a different kind (or no!) symmetry. One unexpectedly difficult problem was that one might have a subset of links that individually possess no symmetry but collectively possess mirror-, rotational-, or indeed dihedral- symmetry. Thistlethwaite's L4a1 is problematic: one would expect the two components to be identical except for a 90-degree rotation, and I was unable to implement this in the context of the package. L6a3 is similarly difficult. We would need new badnesses too. Consider L6a4; the smaller component should be not only as circular as possible (?), but also one might desire the overall length to be smaller than that of the other component.

My considered view is that dealing with links would be possible in principle but, due to a number of non-interesting technical reasons (which are not really suitable for inclusion in a scholarly publication), an order of magnitude harder than the single-component links considered in the submission. I'm not saying it's impossible, but just very difficult and, for me, firmly in the category of "further work". I hope that's OK.

Fig. 10: Why is the absence of mirror symmetry pointed out here?

Some knots have dihedral symmetry but not this one. Discussion added.

Fig. 11: Why is this table shown? It is not mentioned in the main text. The figures are "better" than in Figure 1, but still not optimal in the spirit of this paper. "Rolfsen" needs a reference.

Rolfen now cited, and a discussion of the table now given to place it in context.

References to some of the background material are somewhat random. E.g. For JMA readers, “The Knot Book” by Colin Adams would be a good introductory reference.

Ref [10] Give a more up-to-date access date.

Ref [11] Give URL of Inkscape: <https://inkscape.org/>

Ref [14] Give URL of R Core Team: <https://www.r-project.org/>

All amended as suggested

Detailed rebuttal: Reviewer #2

This lovely article addresses the interesting question of how to produce pleasing and informative images of two dimensional knot diagrams. In addition to the discussion it describes how to use a combination of the software packages Inkscape and R to optimise knots.

At the moment, however the paper is a little too close to simply being a software manual for the software, and needs in particular to describe the techniques used by the software to create the optimised images.

I think this is a perfectly reasonable comment for the original manuscript. I believe the revision is a very much improved document.

In addition there is a rather consistent use of absolute terms for pleasing vs ugly knots. Aesthetics is a complex field, but generally does not provide strong answers.

Absolute terms now rephrased more carefully.

While the choices made in this paper are reasonable, they are simply stated, rather than including a justification for why they are chosen. They certainly only apply to a particular kind of knot diagram. This can be seen in the final image showing all the knots with up to 8 crossings. While some of the resulting diagrams are beautiful, I particularly enjoyed 8_10 and 8_17, some are not so pleasing to me. In particular the algorithm given likes to over-tighten when the line wraps round itself. This can be seen particularly in 8_3, but also in 8_5, 8_6 and 8_11. This is itself a personal preference not an absolute, but a discussion of how to adjust the set-up of the software to make personal aesthetic choices would strengthen the paper.

This is indeed an insightful comment. My original view was that there was a single set of weightings for the various badnesses, that most people

would agree upon; and my task as software engineer was to find the projection that optimizes these standard weightings.

Reflecting on the referee's comment, I realise with a shock that my weightings do not reflect some universal ideal. Rather, my choice of weightings are as subjective as any other aspect of the software and, as such, are peculiar to me. This is obvious in retrospect but was not clear to me at all until just now.

One of the benefits of the `knotR` software is that it allows this kind of conversation to occur; I have added a discussion to the manuscript, just before the conclusions section. There, I present projections of 8_3 but with different penalties for angle crossing terms. We see the results and can compare like with like, perhaps coming to a deeper understanding of pleasingness of projections in the process.

The paper also needs to do a far stronger job of placing itself within the relevant literature. Simply showing that a single diagram from wikipedia has weaknesses is not sufficient. The nineteenth century series of papers "On Knots" by Peter Guthrie Tait in the Transactions of the Royal Society of Edinburgh contain far superior images for example. This is not the first paper to look at making pleasing knot diagrams, the software `knotplot` for example offers many tools to do a similar optimisation to the one discussed here, but is not cited, Laura Taalman and Colin Abrams among others have multiple papers addressing the creation of different knot representations and optimising their form. More broadly the idea of using optimisation in mathematical art has been used widely, the work of Robert Bosch and Craig Kaplan springing immediately to mind.

For these reasons the current paper is not really strong enough for publication. In order to get strong enough I would propose three major pieces of work:

- 1) Conduct a thorough literature review of work on the creation of knot diagrams and their aesthetics and place this paper into that context as well as the broader context of optimisation in art.

I have added a literature review as suggested that discusses knot diagrams from early Celtic artwork through to mathematica.

The referee makes the perfectly reasonable suggestion of discussing the broader

context of optimization of art. I will address this comment colloquially. It is obvious to me, and clearly to the referee as well, that numerical optimization techniques are well-suited to “rounding off rough edges” from a wide range of applications; and further that numerical optimization will produce “nice looking” results. I have cited some of the more prominent workers in the field, whose work is closest to 2D knot projections.

2) Include a discussion of why the aesthetic choices made for the optimisation are pleasing as well as how to adjust the methodology in order to make different personal choices. It might even be interesting to develop some distinct styles of diagrams (by changing the “badness” function, and use the methods to create optimised images for each style.

I have included a figure that compares a knot with two different weightings

3) Give basic information about the specific methods being used for the optimisation rather than just using the function given in R as a black box.

Brief discussion added: the package includes Nelder-Mead by default but the user has the option to use a Newton-type method.

With these changes the paper could be an interesting one, discussing the context of knot diagrams more broadly and providing practical help to help create sets for oneself.