(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.

Rebuttal to second third review: overall response

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

Detailed rebuttal: Reviewer #1

Comments from the Editors and Reviewers:

Thank you for this revised manuscript. My apologies for contributing to the slow turnaround of these reviews.

I have given this version the official status of "Minor Revision" to reflect the fact that we all agree that this draft is much closer to publication-ready. In particular, I believe that the remaining reviewer concerns can be addressed with moderate rewriting. However, some of the reviewer feedback falls closer to "Major Revision," so I wanted to acknowledge that a few of the remaining critiques are beyond surface level.

As you will see in the detailed feedback below, one of the reviewers still has substantive concerns about the overall framing of your discussion and how directly it addresses the mathematical and aesthetic choices that went into the software package. I suspect that you can address the concerns in the first few review paragraphs by being more specific about the constraints of your implementation, the level of flexibility you hoped to provide for the user, and the alternate possibilities for future work, but since I don't know what technical decisions went into your coding, I might be misjudging exactly what is suitable.

I look forward to receiving your further revision, and will make sure the editorial process goes more quickly from this point forward.

Best wishes,

Susan Goldstine, Ph.D. Associate Editor Journal of Mathematics and the Arts

Reviewer #1: The paper has improved in terms of its overall presentation in the context of knot renderings, and many detailed fixes have been made as suggested by the reviewers.

However, the paper still has the major weakness of being too much a somewhat ad-hoc technical manual on "How-to-use-R and Inkplot to get some decent knot plots" by using the software package [27].

The paper is lacking good discussions of the key issues and explanations of the choices made, specifically: What makes a knot plot attractive, followed by a thorough discussion of how to best set up a knot representation and an optimization procedure to achieve the desired qualities?

I appreciate the sentiment, which I take to be a comment on the large amount of programming-oriented detail in the submission. This reviewer evidently regards the knots in the submission as "decent", which I take to be a great compliment. It is often difficult to know the correct balance between overly technical descriptions [which indeed often overlaps with the aims of a technical manual] and gushing conceptual big-picture thinking. However, I note that many of the comments below are essentially requests for clarifications of minor technical details of the implementation: and this would indicate that perhaps the balance should be shifted back toward the technical manual end of the spectrum. The other observation I would make here is that the software package [27] is intended to be part of the submission (it is released under the GPL and is thus available for public use) and with this understanding, even a downbeat technical manual would have scholarly value in conjunction with the new software it describes. I want the editor and reviewer to know that in addressing the concerns below I have tried hard to answer not just "what" but also the "why" of the implementation.

Nevertheless, the cue above is a good one. The two questions: (a) *what makes a knot plot attractive?*, and (b) *how best ... to achieve this?* are indeed profound, and were not addressed directly in the old manuscript. In the revision I have prominently stated the questions in the introduction, and given a provocative, if brief, outline of my view.

The problem starts with the basic curve representation:

Why use Bezier curves?

Most of the answer is in the manuscript: "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". However, Bezier curves are a natural choice for a number of reasons and I have added the following sentence: "Bezier curves are a natural choice for working with mathematical knots for several reasons: they are familiar to many graphical workers; they have a natural and intuitive control system (usually called "handles"), and are implemented in many graphical software suites". I do not wish to suggest that Bezier curves are the only parametrizations for working with knots; but they are a natural starting point, and AFAICS they have no serious disadvantages in this context, nor is there any obvious superior method.

Why not use some other spline curve that guarantees G2-continuity?

I don't have a good answer to that, other than to point out that the focus of the manuscript was to investigate the visual appearance of knots using mathematical optimization routines. I do not wish to rule out using other G2-continuous splines, and it might be interesting some day to consider (e.g.) B-splines. But as I say above, Bezier curves are a good starting point.

And, if forced by the chosen tool to use Bezier curves, why not place nodes at the crossing points, (and one or two more for large loopy lobes between subsequent crossing points)? – This would allow the user to place the crossing points far enough apart to start with, and this would make it easy to set the Bezier handles to force right-angle crossings.

Most of this issue is discussed below, at the comment for page 4, lines 53-54. But here I would observe that it is not at all easy to manipulate Bezier handles on two distinct nodes [which happen to coincide spatially] to result in their two respective lines being orthogonal. Observe that the overstrand and understrand of a crossing point may have (for good reasons!) very different curvatures.

Also, the implementation in the "badness function" seems far from optimal. "Total crossing angles()" does not address a key problem directly: If one wants to avoid highly acute crossing angles, then this can be addressed in a much more sensitive way with a function such as 1/ sqr(cross.angle). This would go to infinity in the worst case! It also corresponds more directly to the way that excessive curvature is avoided by minimizing "bending energy," which is the integral over the square of curvature – or: 1/ sqr(curve-radius). These are some "math" issues that readers of JMA would be interested in.

Function `total_crossing_angles()` was poorly named. With crossing angles θ_i, $i = 1, 2, \ldots k$ the function actually returns $\sum_{i=1}^{k} (\cos\theta_i)^2$.

Some more detailed comments:

Page 3 line 42: "...knots possess a line of symmetry (at least, the diagrams do if the breaks are ignored), which ..." ->> "...knots possess a C2 rotation axis, which ..."

Page 4 lines 22-25: "One plausible technique for creating knot projections is to consider a two-dimensional projection of a knot's embedding E in R3. However, this approach often results in poor visual appearance: cusps or other displeasing effects can occur." ->> True! - But what is the alternative, if the user has an entangled physical knot in her hands?

Being able to render a nice optimized diagram corresponding to a physical knot that one has in one's hands is a reasonable wish. But not one that I can fulfil. Indeed, the package is predicated upon the notion that the diagram is the "real" object, and the (3D) knot itself is irrelevant. Thus much of the package *prevents* the optimization routine from making Reidermeister moves [that is, changes that affect the topology of the diagram but not that of the 3D knot]. The operational reason for this is that the user specifies a particular diagram which the software then optimizes for appearance. The manuscript includes two diagrams of 10_{125}, the Perko pair, which show that sometimes one wants to preserve the topology of the diagram.

Page 4 line 45: Define "nodes" !

"Nodes" is a standard word in the context of Bezier curves. I have not used the word in any other sense in the manuscript and I think the text is clear as it stands.

Page 4 line 48: "However, if we can ensure that the radius of curvature changes only by a small amount,..." ->> HOW can the user do this?

It is not the user that stops large changes, it is the optimization routine and in particular the rather eclectic set of badness functions. Incidentally, the "keep nodes far from crossing points" badness function is instrumental in stopping large changes of this type from occurring.

OLD TEXT: 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.

NEW TEXT: However, because the optimization routine ensures 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.

Page 4 lines 50-52: No need to differentiate here between over- and understrands. I assume they are treated the same in the optimization process.

The referee is correct to say that the over and under strand are treated the same by the optimization routine, but my intent was to introduce and define the words themselves (rather than to differentiate between them).

Page 4 lines 53-54: "We would like strand crossing points to be far from nodes,..." – Why? The curves are also visually continuous across the nodes!

The curves certainly are visually continuous across the nodes—but only in the optimized knots. The motivation for keeping Bezier nodes far from crossing points is twofold. Firstly, as one traces a curve visually, Bezier nodes cannot possibly avoid introducing *some* form of disjointness in the path; and such disjointness is particularly disruptive if on the understrand of a node, as the visual continuity is interrupted. I have added a figure (including two different diagrams of a trefoil knot) to the Gallery that illustrates the reason why nodes' being at crossing points is undesirable. In short, at the node of a Bezier curve, the radius of curvature undergoes a visually jarring discontinuity which looks bad, especially the understrand, which is particularly susceptible to such considerations because of the break. I do appreciate that many viewers will find the distinction invisible or inconsequential; but my informal polling of family and colleagues reveals that, once sensitised to the issue, they cannot unsee the kink.

The second reason is more mundane. With reference to figure 8 (the four figure-of-eight knots diagram), we can see a vertical line of symmetry on which there are two crossing points. Imagine there was a node at one of these crossing points. There would have to be *two* nodes at the same point, one for the overstrand and one for the understrand; the software would somehow have to impose vertical mirror symmetry on the diagram and this would necessitate that the knot class contain yet another symmetry object, alongside the 12 existing ones. These considerations become yet more onerous when considering rotational symmetry as in Figure 9 (which shows 5_1). If there were a node at, say, the intersection of strand 18-19 and 9-10 then somehow the two nodes would have to embody not only mirror symmetry (with respect to 5-6/14-15) but also fivefold rotational symmetry. Simply put, this is a nightmare. The software documentation contains a substantial discussion of the issue, but I do not believe that the manuscript would benefit from what amounts to little more than me grumbling about implementation difficulties.

Page 5 lines 46-51: Why are there SEVEN significant decimal digits? Does the graphic screen have 10 million pixels per line?

This is the default print method used by R (which internally uses IEC-60559 arithmetic, 14 significant figures). Any attempt by me to change the print method would, I believe, be confusing and unnecessary. The point of the table was to show that a knot path could be represented by a few dozen floating-point numbers.

Figure 3, 5, 7: Perhaps fewer than 100 circles per Bezier curve would be better.

Figure 4. Knot 7_6 with strands numbered... – The main use here seems to be to see very clearly the extent of every Bezier curve. Over-/under-crossings seem to become relevant only for the final rendering of the knot curve with the gaps included.

Good point, I have added a sentence to this effect to the caption of figure 4.

Page 7 line 52: "The optimization typically proceeds over R 50" - What is the meaning of this?

It is a 50-dimensional optimization, optimizing a function with fifty parameters (which correspond to different positions of Bezier nodes and handles).

Page 7 line 56: "...function total crossing angles()" The function is not explicitly defined in the paper, but it sounds clearly suboptimal. One bad acute-angle crossing can easily "'hide" in a sum of all angles. It would be better to individually penalize acute angles. A suitable term might be: 1/ sqr(cross.angle). This would go to infinity in the worst case!

Function `total_crossing_angles()` was poorly named. With crossing angles θ_i, $i = 1, 2, \ldots k$ the function actually returns $\sum_{i=1}^{k} (\cos\theta_i)^2$.

Page 8 line 20, and footnote: The "devil" and the "interesting math" is in the details! The footnote is not sufficient to explain the trade-offs that are being made in the optimization process.

Page 9, line 62: What is "Bosch-type symmetry:..." ? Please explain.

Bosch-type symmetry refers to Bosch (2010) which was cited in the previous sentence.

old text: The package implements Bosch-type symmetry…

new text: The package implements symmetry in a similar way to that of Bosch (2010)…

>> If all nodes fulfill the specified symmetry constraints (as shown on page 11, lines 15-22 and on lines 35-43), but the connection diagram does not completely adhere to the specified symmetry - What happens? - Does the package check this and let the user know about the discrepancy?

This situation cannot occur: symmetric nodes imply a symmetric diagram.

Page 10 lines 56-58: "Minimizing the badness is not entirely straightforward..." – What is the problem? I think that if all the points are moved jointly within the specified symmetry constraints, an optimized symmetrical solution will result.

The problem is reconstructing the knot from the minimal set of of nodes and the various types of imposed symmetry. This is not difficult exactly, but not entirely straightforward either.

Also, optimizing high-dimensional functions is rarely straightforward: the search space is large; in my field of computational ecology we talk of the "curse of dimensionality". One has to contend with a wide variety of frustrating issues such as local extrema, expensive function evaluation, and not least the inadequacies of Euclidean distance as a metric for the domain. Dimensionally cursed problems such as that addressed by `knotR` are sometimes literally intractable. My considered view is that Nelder-Mead simplex technique used in the package is unreasonably effective.

The above two comments suggest, that it might be advantageous to use some hierarchy in the representation of symmetrical knots: Only specify the curve for one unique fragment of the knot curve, and compose the complete curve by instancing that fragment with mirroring and rotation as required to construct the complete knot! The optimization would then only have to be done for this unique fragment (with proper end-conditions). If this same hierarchy is also used in the graphical display of the knot curve, the user could easily make adjustments to just a few nodes, while always automatically obtaining the desired symmetrical result.

This is exactly what I have done! The referee gives an accurate summary of the code's methodology.

Page 10 lines 55: "In the package, the badness weightings may be altered easily,..." – What is the user interface for this? Are there sliders in the display that allow interactive adjustments of the weights?

I have not implemented sliders. The weightings may be changed in function `badness()`, and documentation (including examples) is given at `badness.Rd`.

Figure 9: The fact that "angle-crossing penalty" had to be enhanced by a factor of 100 to achieve a more desirable (in my opinion) result, illustrates the fact that the "total crossing angles()"-function is not a good choice for

optimization.

As discussed above, `total_crossing_angles()` was a somewhat misleading name. The factor of 100 ensured that even tiny deviations from exactly 90° were heavily penalised.

I am missing a description how the two breaks at every crossing point are being graphically implemented.

New figure added (Figure 6) which illustrates the implementation

Conclusions:

RE: “Further work might include functionality to deal with links.” >> Readers may think that this would be a trivial extension. Please add a paragraph like the one you wrote in response to Reviewer 1, to explain why the extension to links may be more difficult than one might think.

Done, new subsection “some reflections on links” at the end, under “conclusions and further work”

RE: “In the broader context of optimization in art, we observe that numerical optimization techniques can produce aesthetically pleasing results, an observation that might find uptake by graphic artists.” >> This only holds with the proviso that we can think of a good mathematical formulations to capture the various features that capture aesthetics appropriately. - This then becomes the real “art.”

An insightful observation.

OLD TEXT 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.

NEW TEXT 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. Indeed, one could argue that real art lies in the capturing of mathematical formulations of aesthetics.

Gallery:

Knot diagrams 8_3 through 8_6 in Figure 12 stand out with their helices too tightly wound. The two criteria on page 4, lines 54 and 55 do not seem to be sufficiently enforced. I suggest adjusting these figures in the spirit of Figure 9b. This may happen automatically with a better "badness function" that penalizes non-orthogonal crossing angles more directly.

The referee is correct to say that changing the badness function would result in figures less like 9a and more like 9b. But I would take issue with the phrase "too tightly wound". I have polled my colleagues who appear to be evenly split between preferring 9a and 9b. The comparison appears to elicit strong opinions here at AUT. Proponents of 9b point to the broader winding along the central helix, while those of 9a point to the more liquid and languid appearance, and in particular emphasise the absence of retrograde curvature at the bottom. I have added a brief discussion of these issues to the caption of figure 12 [which is now figure 15 of the resubmission]. I would observe that the `knotR` package makes such discussions possible.

[27]: Incomplete reference. Perhaps give a URL.

Canonical URL now added

Reviewer #2: Firstly I would like to apologise for my late response on this. Reviewing this edited paper had not got onto my radar.

Secondly all the points in my original report have now been addressed. I do feel that the literature review could have gone a little bit further beyond my stated suggestions, but it is now sufficient to give the paper more context.