**Reviewer 1**

*“novelty: After inspecting the code base on GitHub, my impression was that all this package really does is to wrap the functionality of GroundingDINO and Slim-SAM into some R functions, that are executed with a Python back-end. This is essentially replicating the already existing Grounded-SAM approach by Ren et al. (2024), which is also cited by the author but strangely not further discussed. Also, while the original Ground-SAM repo is admittedly convoluted (https://github.com/IDEA-Research/Grounded-Segment-Anything), there is a greatly streamlined version by autodistill (https://github.com/autodistill/autodistill-grounded-sam), which achieves the same functionality with much less code. This implementation of Ground-SAM significantly lowers the bar for open-set segmentation while maintaining full flexibility during inference and downstream analyses, but also is not mentioned. Given these existing tools, I doubt SegColR offers innovations beyond being an R-port of existing Python packages and pipelines.”*

**Sure, I agree.**

**I would like to mention that originally I had coded a version of this package using reticulate. This is a more true-to-form R implementation of these models. However, the behavior of this implementation was far below what could be achieved using a python implementation and command line. So, although the reticulate version of this package would have been more novel from a coding perspective, the performance was massively downgraded because of an inability to distribute across cores. Thus, my choice to use the command line arguments was not arbitrary and was made specifically to enhance performance. I maintain that most users in my field are far more comfortable in R, to the point that many would have never have used python or command line. The novelty of this work is entirely in allowing these users to have access to a working segmentation software with minimal investment. It will be easier for them to build off of a base of this R tool than to learn a completely new set of coding skills. Line 123**

*“paper framing and structure: The paper provides little introduction to the package itself, focusing more on general context around image analysis and discussing common issues and challenges with non-standardized citizen science data. Only a small section (l116-212) addresses the package's functionality. While I don't disagree with the points in the other sections, they make the paper feel more like a review article than a description of the package. This might be because the function scope of the package is quite limited, adding to my previous comment about limited novelty.”*

**I have added a greater focus on the package points. For example, additional information about the package installation (Line XX),**

**Reviewer 2**

*“This paper describes SegColR, an R package that uses two deep learning models to enable semantic segmentation of biological images for e.g. background masking, and provides some code for performing color clustering on the resulting segmentations. I think the general idea of the package--a simple interface for zero-shot segmentation that allows users to provide their own labels and save the segmentation results--is incredibly useful, and can be applied to a huge range of problems in biological image analysis. Background masking, for example, is easily one of the most tedious tasks that most biologists encounter when trying to quantify images. I look forward to using a version of the package in my own research.*

*However, I think the package/paper as written have some technical and methodological issues that need to be resolved in order for this method to be used widely. I detail my recommendations below, but in summary: 1) I would remove the color clustering function and instead add options for batch processing and exporting to different formats; 2) the existing segmentation code needs to be carefully debugged since many of the examples do not work as written; and 3) the utility of the method should be demonstrated on a small example dataset. These changes would make the package much more usable to a wider audience of biologists, and would allow it to address some important and long-standing bottlenecks in image analysis.”*

**Thank you for taking the time to test the package and your thoughtful comments. I have addressed all of the bug reports you provided and taken your advice on reframing the package to be focused specifically on segmentation rather than color extraction. As such, I have renamed the package SegmentR. Note that the color extraction features remain in the package for the sake of the case study, but segmentation is now the focal point of the work.**

*“I tried out the R package version of SegColR, and I had a colleague who also has extensive computational image processing experience try out the Python scripts. We both ran into issues with installation, and almost every line of the R code as written in the examples and on the GitHub directory throws an error or a warning. Of course occasional warnings/errors are inevitable, but the example code should work, especially if the audience is biologists without extensive coding experience.”*

**Thank you for testing the code and I apologize that it did not work. I did have colleagues try the package at various stages of development, so I am somewhat surprised by the extent of the issues you experienced. Hopefully, this newest version results in a more streamlined experience.**

*“In R: I already had Anaconda installed, but when I ran setup\_conda\_environment(), the function looked only for a Miniconda install. I ended up install Miniconda as well (which did work after cleaning up some of the failed conda environments), but either tell users to install Miniconda or test this out on some other machines with Anaconda installed.”*

**This is odd, as I had tested an Anaconda3 machine. The function search\_conda\_locations() attempts to search for the conda install and I am unsure why it was not able to find the installation. search\_conda\_locations() code examines common locations where conda may be installed (Home, usr/local, and /opt) and then checks for both miniconda3 and anaconda3 by appending the typical path to conda (“miniconda3/bin/conda” or “anaconda3/bin/conda”). After further testing I was unable to replicate the error where miniconda was required and anaconda3 did not work. I suspect it may have been related to the python-based plotting, so I have removed that functionality temporarily. Furthermore, I have added an additional section in the paper focused on installation of the package and have included that Miniconda had been reported to work when Anaconda did not.**

*“In Python: My colleague installed the Python environment using the provided .yaml file, but had to install some additional packages as well before the environment built. He recommends restricting the .yaml file for Python users to the minimal packages needed, and perhaps mentioning that since only these few packages are needed, it's not essential to use the provided .yaml file to build that specific environment (e.g. could just direct users to set up a conda environment with packages X, Y, and Z, since most of us have done this).”*

**There is an alternative .yml file for installing the package with only the necessary packages named “environment\_general.yml”. This should be, and has been made, the default installation procedure. I had originally used a more specific environment because solving the package dependencies can take a long time. However, after testing I found that the specific environment led to more errors and thus, I had changed the default behavior in the R installation to use the general environment. I have now entirely removed the option for a “specific environment” installation to avoid confusion.**

*“This is especially important for the R package version of the software, since the author specifies that this package is intended for people without much coding experience. I usually see users without much coding experience give up when this level of troubleshooting is required. A perfectly smooth installation process is not required (and probably impossible given the Python packages involved), but it could be made much smoother than this, and there should be more instructions provided.”*

**Agreed, thank you for the feedback.**

*The code in the paper is outdated (presumably the package structure was updated in between the paper submission and this review).*

*For example, the first process\_masks\_and\_extract\_colors() usage (for the grouper image) throws an error because ground\_results is now just a list of the command input/output and relevant filepaths for a segmentation call, and the provided error message ("No masks meet the score threshold") is not informative. This works fine on the GitHub README though, so just update things like this.*

**Yes, that is the case the code was updated while the paper was under consideration. As per your suggestion, color extraction is no longer a focal point of this paper and thus this example is no longer included.**

*I could not get the flower/bee example to work as written. The first step is fine (see image), except that for some reason the package appends periods to the ends of all my labels (I provided the labels as just "a flower" and "a bee"--if the periods are required for the labeling to work it should be stated). But providing a score threshold of 0.5 for the color segmentation only resulted in the larger flower being displayed.*

**Adding a period at the end of a search has been reported to improve performance of the GroundingDINO model and in my experience this improvement is substantial. In my own testing I had always included the period in my prompt and thus forgotten that I had added code which automatically adds the period if it is not there. I have now made a note of this in the code as well as in the paper.**

*Oddly, when I provided the labels as just "bee" and "flower", the bee was included in the color analysis (although the confidence scores remained very similar), although the flower with a confidence of 0.63 still did not appear.*

**This bug has been resolved.**

*The segmentation features of the package have the potential to be enormously useful, but the color clustering function seems ancillary, and I don't think there is a real use case for it that isn't covered already by dedicated software with more options. I recommend removing the color clustering function and instead improving the options surrounding the segmentations, namely, make it easier to batch process images and provide more export options.*

**Thank you for your thoughtful feedback. The reason that color clustering was a focal piece of this paper originally was because of my own use case for the software. However, I agree with the reviewer that it is an ancillary feature for most use cases. As such, I have taken several steps to reframe the paper and software around segmentation in R. The package is now called SegmentR and all references to SegColR have been removed.**

*The author already cites a few other packages that could be used in downstream analysis, e.g. patternize, pavo, recolorize, and micaToolbox/QCPA. All of these could take the results of SegColR as input, since they rely on (or can use) segmented images. For example, the colordistance package (not cited here but seems relevant) uses background-masked images to perform color clustering of RGB images, similar to the color clustering function in SegColR. But in that package it's the first step in a pipeline for quantitatively comparing color distributions, and provides more options than k means clustering or pre-specified color centers. In a similar vein, images processed in micaToolbox currently require and ImageJ region-of-interest (ROI) mask to restrict an analysis to e.g. just a fish and not the background. Patternize, recolorize, and pavo also all can take background-masked images but provide no method for masking the background besides (if I recall) ignoring particular RGB colors.*

**Yes, these packages are better suited for color analysis, and I agree that SegmentR would be better utilized as a complement to these more specialized packages.**

*The color clustering as shown in the examples also appears to be quite low-information. In the first example, the grouper shown is clearly tinted green, presumably because it was photographed at a depth of at least a few meters down in seawater. So what do we learn by taking the average RGB color of the grouper? The third example (in Fig. 3), which focuses specifically on the color segmentation of the fish, describes the fish-body-only segmentation in 4D as "better matching the original image", but what criteria are we using to say that it is a better match? What information do I gain from the lighter colors around the eye, given how uneven the lighting is in this image to start with (looks like it was taken with flash)? The original recolorize paper specifies that users need some biological criteria for the color classes. I support the use of databases like iNaturalist to get more phenotype information, but we should be explicit about what kind of quantitative data this can realistically provide.*

**I agree, and though color is no longer a central point of the paper, I would like to share my thoughts. There is likely little information from a set of images in an absolute sense because color is going to be distorted by lighting conditions, water depth, etc. However, because we are starting from a place that is information rich, a well-defined low-information regime could have as much meaning post-distortion as they would have if we had perfect original representations. This is because although noise is added by the process of taking the image, broad categorizations can generally be maintained. I have decided to include this in the limitations section of the paper as I believe it is something that should be considered whenever gathering data from noisy and unstandardized sources.**

*All this to say, I would rather have a really good segmentation package that lets me export segmentation results in a variety of formats to dedicated color clustering softwares than have another color clustering function that does a subset of the analysis found in existing methods. I recommend allowing users to export segmentations as 1) a PNG (or TIFF, etc) with certain regions transparent (e.g. set the background and fins to transparent, export a separate image with each fin separately on a transparent background, etc); 2) a binary mask; 3) an ImageJ ROI (or just a set of XY coordinates); 4) at least one widespread annotation format like COCO JSON. This is up to the author, of course, but these are the export formats I see used most commonly in eco/evo problems (of which I see a lot). It could be quite a powerful but easy-to-generate example to run this package on a small dataset (see below), export the results to another package, and analyze the results from that package, vs. compare how long it takes to run without SegColR (e.g. manual annotation).*

**Thank you for sharing your experience. These have been added as outputs and the example is now a comparison via manual annotation.**

*Running the package on a very small sample dataset (even 3-5 images) will enable the author the demonstrate the utility of the package for a concrete problem, even a simple one, as well as test out batch processing options, since this method is really mostly useful in cases where users have more images than they would be willing to label manually (usually 2+ images). These can be taken from iNaturalist, but I would also encourage the author to find a biologist in their home department/university (I am sure the EEB department will have many of them) who needs to quickly label and mask a series of images. Not only would this demonstrate utility for an existing biological problem, but it would help the author troubleshoot the ways that a biologist without extensive computational experience actually uses their package, which will improve uptake in the community. Given the number of bugs I found in testing the package even for the examples, I think getting another user to test out the package could be really valuable. In general, I’m not totally sure why this package is pitched as being primarily useful for images from iNaturalist. All kinds of biological images need a quick segmentation solution.*

**Thank you for the excellent suggestions. I have replaced my previous examples with a single focused example based on small set of specimen images.**