microPublication BIOLOGY

10/26/2026 - *Open Access*

BlindSpot facilitates unbiased data analysis via file renaming

Jenna S Vesey[1], Amy B Liu[1], Siddharth K Sankaranarayanan[1], Amy S Maddox[1], Linnea C Wethekam[1§]

[1]Department of Biology, UNC-Chapel Hill, Chapel Hill, NC, US

[§]To whom correspondence should be addressed: wethekam@unc.edu

Abstract

Bias and subjectivity are to be avoided in scientific research since they can lead to incorrect data interpretation. Methods that enhance the rigor and reproducibility of biomedical research include filename blinding. To overcome the limitations of currently available blinding programs, we developed *BlindSpot*, a publicly available, user-friendly tool for filename renaming. *BlindSpot* is compatible with macOS and Windows, supports both a graphical user interface (GUI) and a command-line interface (CLI), and has features to customize blinding and safeguard data identity.

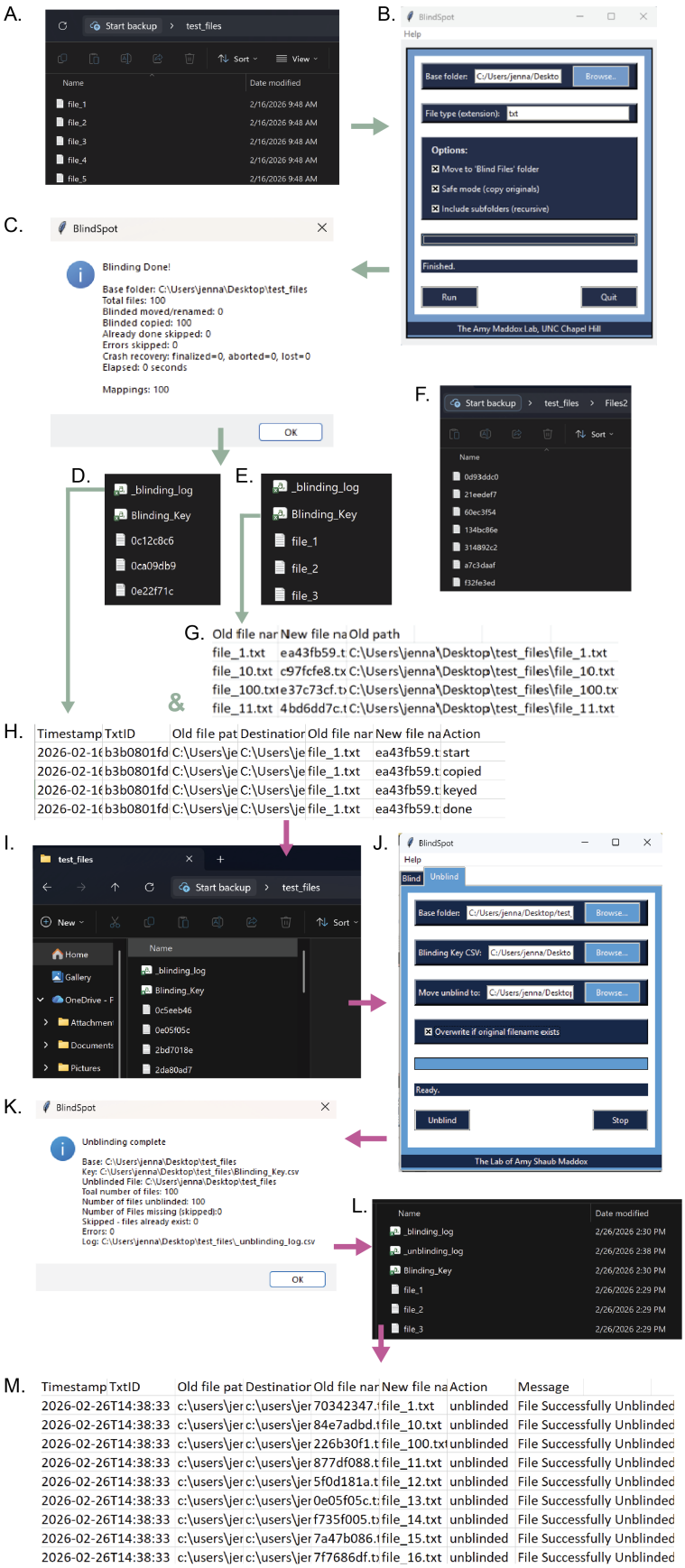


Figure 1. Overview of the BlindSpot pipeline for reproducible filename blinding and restoration:

A - H) blinding pathway (green arrows). I - M) unblinding pipeline (purple arrows). A) User identifies target folder for blinding. B) User accesses the graphical user interface (GUI), selects specified inputs, designates the target folder,

specifies the file type, and makes optional selections. C) Once blinding is complete a pop-up shows a summary of the blinding. D) Example output where with no image duplication or file movement. E) Example output where blinded files were duplicated and moved to a new folder. F) Example including subdirectories with the blinded files having been moved to a new folder. G) Layout of the Blinding_key.csv for data identity restoration. H) Layout of the _blinding_log.csv showing tracked and recorded actions. I) Example folder containing blinded data. J) Unblinding function tab. K) GUI popup after unblinding is complete. L) Output folder after running the unblinding code. M) Printed output _unblinding_log.csv.

Description

Scientists strive for objectivity to ensure the integrity of results, yet a “reproducibility crisis” exists (Bespalov et al., 2020; Baker, 2016). As few as 39% of published studies have been reproduced (Youyou, W., Yang, Y., & Uzzi, B. 2023). One basis of irreproducibility is error introduced by investigator bias (Bello et al., 2014; Jost and Waters 2019). Blinded analysis reduces bias and increases reproducibility (Baker, 2016; Jost and Waters 2019). However, blinding is not consistently reported in the analysis of microscopy images and other datasets, in part because it can be tedious (Begley and Ioannidis 2015; Bello et al. 2014; Bespalov et al. 2020; Karp et al. 2022; Jost and Waters 2019). While most blinding programs are specific to file format, Blinder and ImageJ’s blinding plugin can be used for multiple file types (Cothren et al., 2018; Jaiswal & Lorenz, 2019; Schneider et al., 2012). However, both temporarily rename files within a user interface – either an independent GUI or ImageJ – limiting the portability of the files for image analysis. We developed *BlindSpot* to stably blind file names in a flexible and accessible manner.

In designing *BlindSpot*, we prioritized crash safety and accessibility for non-programmers. *BlindSpot* not only accomplishes filename randomization, but also provides crash-safe bookkeeping that guarantees each file is renamed exactly once. Importantly, *Blindspot* restores a file’s original name after analysis is complete.

To begin, users open *BlindSpot’s* graphical user interface (GUI) on their device and identify the desired folder for blinding and select the required features (Figure 1A, B). The program first confirms files of the specified type are present, and if directed, also searches all subfolders (Figure 1A, B). To blind each file, the program selects a file with the correct extension and verifies that it has not already been blinded, using _blinding_log.csv (Figure 1D-F). If the user elects for the blinded file to be a copy of the original file, the original file is retained and thus protected. If the user does not select this option, a warning message prompts the user to confirm the intention to overwrite filenames. Files are then renamed randomly using the final 8 characters from the universally unique identifier (UUID) package (Leach et al., 2005). Progress of the blinding is logged into _blinding_log.csv throughout the process; Blinding_key.csv is produced after all files are blinded (Figure 1G). Within _blinding_log.csv, each step of the blinding process (Starting, Keyed, Pending Skipped, Error, Copied, Moved, and Done) is logged with a time stamp (Figure 1H). Blinding_key.csv contains the original file name, the corresponding blinded file name, and the directory path of the original file (Figure 1G). If a crash occurs during the blinding of a set of files, upon re-initiation, the program will pick up where it left off, such that each file is blinded once. A progress bar is visible in the GUI as blinding occurs.

To restore original filenames, *BlindSpot* uses blinding_key.csv, the location of the blinded files, and the user’s desired output location to run this program. First, all files represented in the key are identified by searching the designated directory and its subfolders (Figure 1I, J). File names not found are ‘Skipped.’ Unblinded files are marked ‘Done.’ Finally, if all of the program’s required conditions are met, the file is unblinded and placed in the user-specified directory (Figure 1L). Users can choose to overwrite the original files with the blinded modified file (Figure 1J), though situations where multiple files would result in the same filename are not automatically resolved. Unblinding creates a log summary and greets the user with a random quote after the software is done with its run (Figure 1L, M).

To incorporate *BlindSpot* tool into extended workflows, the ‘blind’ and ‘unblind’ functions can be added to python workflows with the use of PyPI packages and command line interface. Documentation on how to call and utilize *BlindSpot* in the command line is present in the Readme.md on GitHub. Directions for downloading the software is available in the GitHub page in: “BlindSpot Installation Instructions”. Future development will focus on integrating *BlindSpot* with popular imaging analysis software and improving compatibility with automated and computational pipelines.

Methods

BlindSpot was written in Python version 3.14.2 using Visual Studio Code. It was formatted using the program package “black” to enable easy readability (Langa & Black contributors, 2026). *BlindSpot* was compiled with the use of bin for Windows and bash for Mac using PyInstaller (v 6.19.0) and tkinter via command line functionality (*PyInstaller development team. PyInstaller (Version 6.19.0)).* The program uses the following natively installed python packages: os, csv, uuid, time, shutil, random, threading, pathlib, tkinter, datetime, webbrowser, and sys. The logo lil_timmy.png was drawn in MS paint (v Paint 11.2512.211.0, Microsoft Corporation) and animated with the use of Canva’s gif function.

To test the robustness of *BlindSpot*, pytest was used (pytest developers. Version 9.0.2), a Python package that allows functions to be tested independently. This program was run through stress tests to confirm that each of 32 elements of *BlindSpot* can function and recover amid any errors (see Blinder_test.py, Unblinder_test.py, Crash_log_test.py, and File_processing_test.py on GitHub). File_processing_test.py documents the testing of these three functions: file naming, file folder, and path cross-compatibility among operating systems. Fourteen functions test the integrity of the crash log. It checks that the software identifies files that have been processed, recovery in the case of a system crash, and that all user-selected options would be consistent across each file run. A test of nine features assessed integrity of the blinder function (Blinder_test.py). Each of these functions ensure that regardless of user settings, the program calls each function correctly and accurately for every file. Each of the twenty-five tests ran in under one minute, including trials with 1000 files, though duration of our tests reflected file size. The unblinding function was also tested to ensure that unblinded files would be properly matched with original files the corresponding name. Tests were performed on both Windows and MacOS. Web application and code can be accessed at this link: https://github.com/maddox-lab/BlindSpot.git. Detailed instructions on how to download BlindSpot can also be found in this folder labeled “BlindSpot Installation Instructions”.

AI usage: AI assistance (ChatGPT-5.3, OpenAI) was used to suggest debugging strategies and interpret error messages during development. No AI assistance was used to write or edit the manuscript. The pylint package helped identify errors for formatting and structure for further review (pylint contributors, n.d.). The Black formatter was used for code formatting (Langa, Ł., & contributors to Black, 2026). The code was tested under human scrutiny and checked against current literature regarding the code for accuracy of function and layout. All code functionality and design decisions were verified by the authors.

Acknowledgements: We would like to thank members of the Amy Shaub Maddox Lab for helpful discussions during the development of BlindSpot.

Extended Data

Description: Zip folder of the GitHub repository containing both PC and MacOS BlindSpot Software and all accompanying documentation.. Resource Type: Software. File: BlindSpot-main.zip. DOI: 10.22002/5zfem-kkx10

References

Baker M. 2016. 1,500 scientists lift the lid on reproducibility. Nature 533: 452-454. DOI: 10.1038/533452a

Begley CG, Ioannidis JPA. 2015. Reproducibility in Science. Circulation Research 116: 116-126. DOI: 10.1161/CIRCRESAHA.114.303819

Bello S, Krogsbøll LT, Gruber J, Zhao ZJ, Fischer D, Hróbjartsson Ar. 2014. Lack of blinding of outcome assessors in animal model experiments implies risk of observer bias. Journal of Clinical Epidemiology 67: 973-983. DOI: 10.1016/j.jclinepi.2014.04.008

Bespalov A, Wicke K, Castagné V. 2019. Blinding and Randomization. Handbook of Experimental Pharmacology,Good Research Practice in Non-Clinical Pharmacology and Biomedicine : 81-100. DOI: 10.1007/164_2019_279

Cothren S, Meyer J, Hartman J. 2018. Blinded Visual Scoring of Images Using the Freely-available Software Blinder. BIO-PROTOCOL 8: 10.21769/bioprotoc.3103. DOI: 10.21769/BioProtoc.3103

Jaiswal, A., & Lorenz, H. (2019). Blind analysis tools [ImageJ plugin]. *ImageJ*. Retrieved February 19, 2026, from https://imagej.github.io/plugins/blind-analysis-tools

Jost APT, Waters JC. 2019. Designing a rigorous microscopy experiment: Validating methods and avoiding bias. Journal of Cell Biology 218: 1452-1466. DOI: 10.1083/jcb.201812109

Karp NA, Pearl EJ, Stringer EJ, Barkus C, Ulrichsen JC, Percie du Sert N. 2022. A qualitative study of the barriers to using blinding in in vivo experiments and suggestions for improvement. PLOS Biology 20: e3001873. DOI: 10.1371/journal.pbio.3001873

Langa, Ł., & Black contributors. (2026). *Black: The uncompromising Python code formatter* [Computer software].https://github.com/psf/black

Leach, P. J., Salz, R., & Mealling, M. H. (2005). *A universally unique identifier (Uuid) urn namespace* (Request for Comments RFC 4122). Internet Engineering Task Force. https://datatracker.ietf.org/doc/rfc4122/

PyInstaller development team. (n.d.). PyInstaller (Version 6.19.0) [Computer software]. https://github.com/PyInstaller/PyInstaller

Pylint contributors. (n.d.). *Pylint* (Version 4.0.4) [Computer software]. https://github.com/pylint-dev/pylint

pytest developers. (n.d.). *pytest* (Version 9.0.2) [Computer software]. https://github.com/pytest-dev/pytest

Schneider CA, Rasband WS, Eliceiri KW. 2012. NIH Image to ImageJ: 25 years of image analysis. Nat Methods 9(7): 671-5. PubMed ID: 22930834

Youyou W, Yang Y, Uzzi B. 2023. A discipline-wide investigation of the replicability of Psychology papers over the past two decades. Proceedings of the National Academy of Sciences 120: 10.1073/pnas.2208863120. DOI: 10.1073/pnas.2208863120

Funding:
Supported by National Institute of General Medical Sciences (United States) 5K12GM000678 to Linnea C Wethekam.
Supported by National Institute of General Medical Sciences (United States) R35GM144238 to Amy Shaub Maddox.
Supported by National Science Foundation (United States) 2153790 to Amy Shaub Maddox.

Conflicts of Interest: The authors declare that there are no conflicts of interest present.

Author Contributions: Jenna S Vesey: conceptualization, methodology, software, writing - original draft, validation. Amy B Liu: methodology, software, validation. Siddharth K Sankaranarayanan: methodology. Amy S Maddox: funding acquisition, project administration, writing - review editing. Linnea C Wethekam: conceptualization, funding acquisition, project administration, writing - review editing.

Reviewed By: Tara May Finegan, Anonymous

History: Received April 20, 2026 **Revision Received** August 3, 2026 **Accepted** October 22, 2026 **Published Online** October 26, 2026 **Indexed** November 9, 2026

Copyright: © 2026 by the authors. This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International (CC BY 4.0) License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Citation: Vesey JS, Liu AB, Sankaranarayanan SK, Maddox AS, Wethekam LC. 2026. BlindSpot facilitates unbiased data analysis via file renaming. microPublication Biology. 10.17912/micropub.biology.002154