# *Supplementary Information accompanying the article*

# Violin SuperPlots: Visualising replicate heterogeneity in large datasets.

# M. Kenny and I. Schoen

# **Journal, issue: pages (2021)**

# **Superviolin: a Python CLI to generate Violin SuperPlots**

This package can be used to create Violin SuperPlots. As well as a tutorial on how to use the package, this documentation provides an introduction to Python installation and package management to enable non-programmers to easily setup a Python workspace to use this software.

# **1. Python installation for non-programmers**

If you haven’t used Python before, you will be installing a Python distribution called Anaconda, which is widely used by scientific Python developers. A distribution is a bundle that contains the components necessary to run Python, as well as some domain-specific packages. For example, the Anaconda distribution is intended for use as a data science solution and contains packages for machine learning, scientific data analysis, and much more. It ships with over 200+ packages, including the package dependencies for Superviolin.

However, as this guide is aimed at non-programmers, we will be installing the Miniconda distribution, which has the same functionality as Anaconda, but saves on hard drive space and installation time by only installing the basic packages required for it to function. If there are other packages you want to use in the future, you can install them manually later on.

# **To install Miniconda:**

1. Download the Python 3.x Miniconda installer for your OS at <https://docs.conda.io/en/latest/miniconda.html>.   
   Superviolin requires the installed Python version to be at least 3.7
2. Run the installer and follow the on-screen instructions, using the default settings you are prompted with.
3. Congratulations! You now have a functional Python environment on your computer!

**To install the Superviolin package:**

You can now proceed to installing Superviolin. Python has a so-called package manager which is used to install, update, and remove packages. It ensures that all package dependencies are maintained and will warn you if something would be broken by a new package installation.

Superviolin uses basic functions from Numpy, Scipy, and Pandas which aren’t likely to be changed dramatically, so you don’t need to worry about versions. As long as you installed Python 3.7 or later, Superviolin should install just fine.

Installing Python packages requires you to enter text commands in the terminal (Mac, Linux) or in Anaconda Prompt (Windows). Spacing and upper/lower case must be maintained to ensure the commands run as expected. Most common programming errors occur due to these mistakes and they affect both experienced and beginners alike, though experienced programmers tend to get these errors less often.

1. To install Superviolin, open up the terminal or Anaconda Prompt and enter:

pip install superviolin

The package manager will list the dependencies required and install them for you. Once you see a message stating superviolin was successfully installed, you can proceed to the next section and try out the package with your data.

# **2. Superviolin command overview**

The superviolin CLI has 3 commands:

## **demo**

This command creates a Violin SuperPlot using dummy data that ships with the package. Run superviolin demo from your command prompt after installation to be sure the package is working. It should generate a figure similar to **Figure 1** below. If you get an error, please contact Martin Kenny at [mkenny5@tcd.ie](mailto:mkenny5@tcd.ie) to report your issue so that it can be corrected.

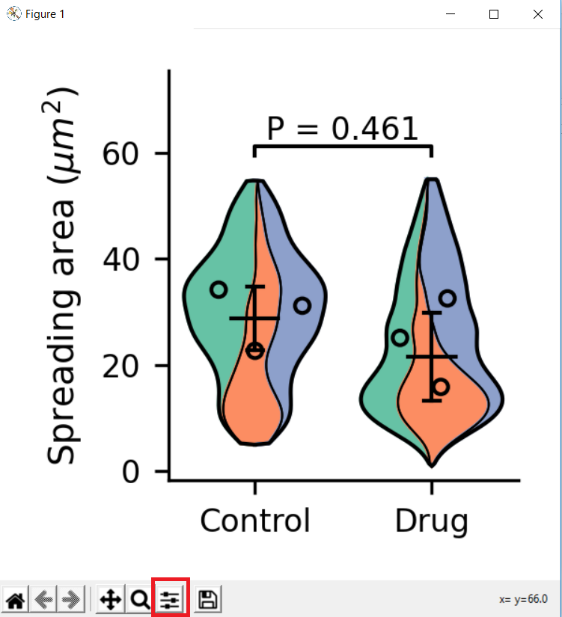
## **init**

The superviolin init command generates an ‘args.txt’ file in the current directory, which will be used to generate a Violin SuperPlot based on the Excel or csv file of your data located in the same folder using arguments specified in this text file.

## **plot**

The superviolin plotcommand renders the Violin SuperPlot as a figure. This layout can be edited prior to saving.

*NOTE: To run any of these commands, enter* superviolin command *into your terminal or Anaconda Prompt.*



**Figure 1.** *Violin Superplot generated by the* demo *command.*

**3. Generating a Violin SuperPlot**

Superviolin comes with a command-line interface (CLI) so you can generate SuperPlots using the terminal or Anaconda Prompt. You can also import the “superplot” class into your Python scripts and extend its functionality to create your own specific brand of Violin SuperPlot, but first, let’s understand how it works. This section contains a simple how-to for non-programmers.

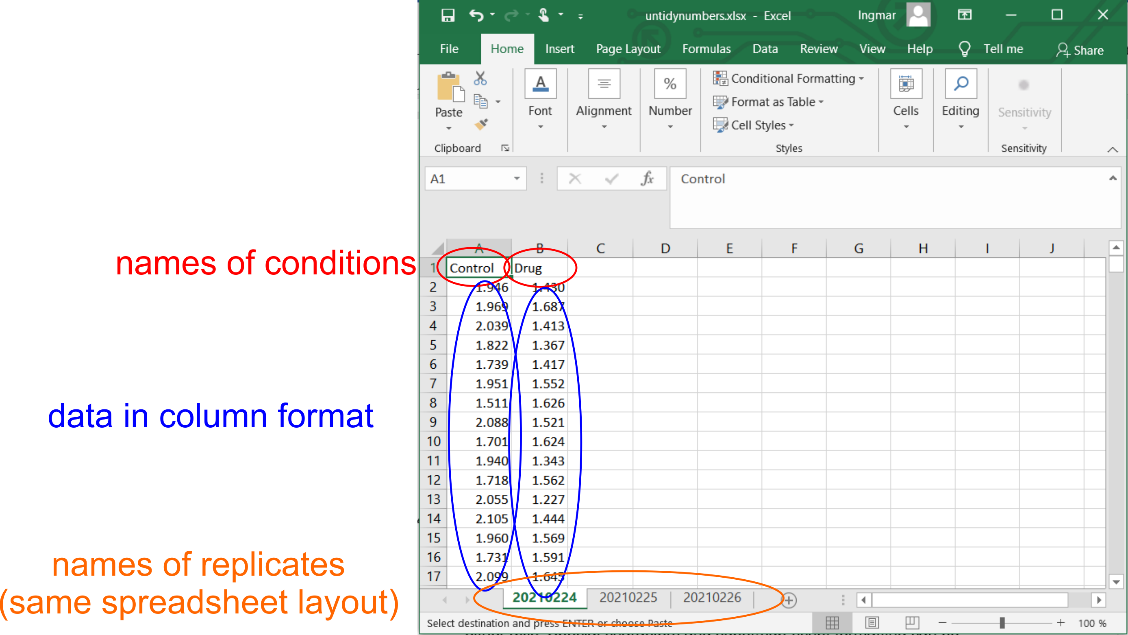
1. Prepare your data as a comma separated value (\*.csv) or Excel spreadsheet file and place it into an empty folder. The data should be arranged in either (A) the tidy data format (see **Table 1**) where you have a column for each variable/property and a row for each observation, or (B) ‘untidy’ data in an Excel workbook where each spreadsheet is an experimental replicate containing columns for each condition (see **Figure 2**).

(A) For tidy data, the *Condition* column specifies if an observation belongs to a different condition (control vs drug, different drug concentrations, wild type vs mutant, etc.), the *Value* column(s) specify the value(s) of the measured variable(s) (spreading area, circularity, intensity, etc.), and the *Replicate* column specifies which replicate the observation is from (donor ID, date, well, etc.).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | Area | | | Ellipticity | Replicate |
| Control | | 25.2 | 1.434 | | Donor 1 |
| Control | | 17.1 | 1.017 | | Donor 1 |
| Control | | 32.3 | 1.157 | | Donor 1 |
| Control | | 29.7 | 1.481 | | Donor 1 |
| … | | … | … | | … |
| Drug | | 26.6 | 1.418 | | Donor 1 |
| Drug | | 12.4 | 1.592 | | Donor 1 |
| Drug | | 22.8 | 1.717 | | Donor 1 |
| … | | … | … | | … |
| Control | | 28.9 | 1.481 | | Donor 2 |
| Control | | 35.3 | 1.294 | | Donor 2 |
| … | | … | … | | … |
| Drug | | 9.8 | 1.767 | | Donor 2 |
| Drug | | 32.2 | 1.376 | | Donor 2 |
| … | | … | … | | … |
| Control | | 34.1 | 1.374 | | Donor 3 |
| … | | … | … | | … |
| Drug | | 25.5 | 1.810 | | Donor 3 |
| … | | … | … | | … |

**Table 1:** *Example of a data file in tidy format (either csv or Excel file).*

(B) For ‘untidy’ data, column names have to be exactly the same across replicates to allow the software to combine the data for plotting; the data in each column doesn’t have to be paired data, i.e. can have different number of entries. Column names should not contain special characters or sub-/super-script.



**Figure 2:** *Example of a data file in an ‘untidy’ format (several Excel spreadsheets).*

1. Open the terminal or Anaconda Prompt. Navigate to the folder containing the data file by typing & copy-pasting cd drive:\full\path\to\your\data\folder into the command line (replace the ‘drive:\...’ by your actual path).
2. Run superviolin init to generate the args.txt file in this folder.
3. Edit the args.txt file with the parameters specific to your data file. Optional arguments are detailed in section 6. The required parameters are:
4. **filename**, the name of your data file. Must be in CSV or Excel format.
5. **data\_format**, whether the data follows the format in table 1 (tidy) or figure 2 (untidy).

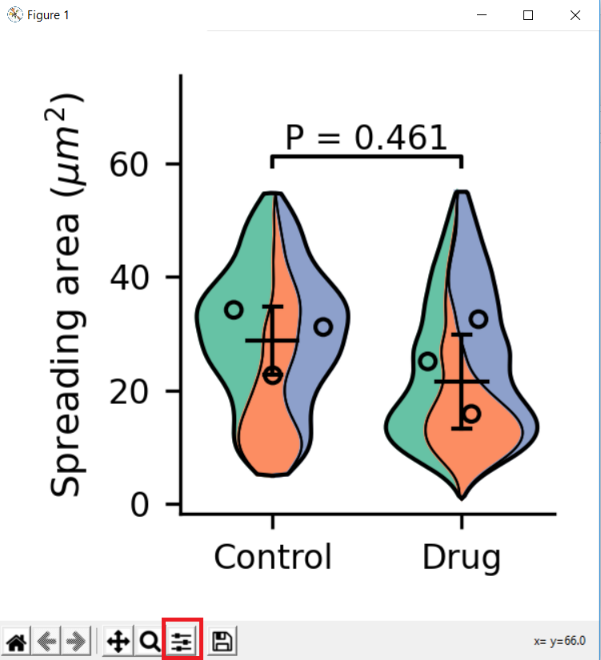
For tidy data, additional inputs are required:

1. **condition**, the column in your dataset which specifies the experimental conditions.
2. **value**, the variable to be plotted from your dataset.
3. **replicate**, the name of the replicate column in your dataset.
4. Save the args.txt file.
5. Run superviolin plot to generate your Violin SuperPlot which will open in the “Figure 1” window. Please note that the Figure window needs to be closed to be able to access the command line / Anaconda prompt again.

Simply changing the filename in the args.txt file will allow you to apply these settings to other CSV or Excel files with the same structure, provided the new filename is in the same directory as the args.txt file.

**4. Editing your Violin SuperPlot**

Most of the editing of the Violin SuperPlot is done directly by arguments in the args.txt file. The basic Violin SuperPlot generated by the program can then be further adapted to edit the amount of whitespace around your plot. **Figure 3** shows the Violin SuperPlot generated by the superviolin demo command. The whitespace around the plot is already optimized within the code, but these spacings can be edited by clicking on the button (highlighted by the red box) to open a menu of sliders corresponding to the whitespace to the top, bottom, left, and right of the plot.



**Figure 3.** *Layout of the Violin Superplot figure window.*

**5. Saving your Violin SuperPlot**

Once satisfied with the plot, you have the option to save it either as a rendered image or a vector graphics image. Clicking the save icon  will open a dialog to choose the location for saving your figure and allow you to select the appropriate output format from the dropdown menu.

Rendered images: preferred are lossless formats (TIFF, PNG). The plot is saved at the resolution specified during creation of the plot (see section 6 below, i.e. in the “args.txt” file).

For maximal resolution and further editing in other programs, it is recommended to save the figure in SVG vector graphics format.

**6. Optional arguments in args.txt used to customize the Violin SuperPlot**

* + 1. **xlabel** and **ylabel** indicate the labels you wish to use for the x and y axes. These arguments can be left unchanged if you don’t wish to have a label on either axis. Special characters and super/sub-script formatting can be specified using Python’s string formatting: <https://pythonforundergradengineers.com/unicode-characters-in-python.html>

For example, the y-axis label in figure 1 below, “spreading area (µm2)” is written into the args.txt file as *Spreading area ($\mu$$m^2$)*, where $\mu$ and the $m^2$ correspond to the greek letter µ and the superscript m2, respectively.

* + 1. **order**, the order of the experimental conditions to be displayed on the x axis. This argument MUST be separated by a single comma and single space otherwise it will throw an error. The default order is alphabetical.

*In the above example: Control, Drug*

* + 1. **Middle\_vals**, the central measure of each replicate (mean, median, or robust mean). Displayed as circles over each replicate stripe. Default is mean. Accepted values are *mean*, *median*, or *robust*.
    2. **Centre\_val**, the central tendency (*mean* or *median*) calculated from replicate means (or medians) and shown between the error bars on the plot. Default is *mean*.
    3. **Error\_bars**, whether to use the standard deviation (*SD*) or 95% confidence interval (*CI*) for plotting error bars. Default is *SD*.
    4. **Bw**, the smoothening factor for stripe outlines. Should be between 0 and 1. The default value is *None* (automatic determination of the smoothening factor).
    5. **Statistics**, choose whether to display statistics on the plot. Only shows statistics for 2 or 3 experimental conditions. Either *yes* or *no*.
    6. **Ylimits**, the minimum and maximum values to display on the y-axis. Should be in the form of two numbers separated by a comma and a space, like *low, high*. If this argument is to be ignored, leave the value as *None*.
    7. **Total\_width**, the percent width of each violin in decimal form. Should be between 0 and 1. Default is 0.8.
    8. **Linewidth**, the width of lines (in pixels) used to plot summary statistics and the outline of each violin. Default is 1.
    9. **Cmap**, the colourmap specifying the colours used for the replicates. *Set2* is the default colourmap, but users can specify a list of colours or a colourmap of their choosing from <https://matplotlib.org/stable/gallery/color/named_colors.html> or <https://matplotlib.org/stable/tutorials/colors/colormaps.html>
    10. **Dpi**, the dots per inch used for saving the plots. Plots are rendered at 300dpi for display purposes, but can be saved at a different resolution, specified by this argument. The default value for saving is *600* dpi, but can be changed to reduce file size or increase resolution, e.g. *300* or *1200* dpi.