1. A RATIOMETRIC METHOD FOR THE ANALYSIS OF DIFFERENCES IN PROTEIN EXPRESSION

Preynaptic active zones (AZ) in *Drosophila melanogaster* are 200-300 nm in diameter and reach about 100 nm into the cytoplasm (Hallermann et al., 2010; Liu et al., 2011; Maglione and Sigrist, 2013; Südhof, 2012). This corresponds approximately to the resolution limit of conventional light microscopy (Inoué, 2006). For our analysis, we acquired confocal microscopy images with a pixel size adequate for a lateral resolution of up to 200 nm (88 nm). In this manner, we could capture every synapse in a two-dimensional image plane. In order to cover complete, three-dimensional neuropils, we acquired image stacks containing around 150 single image planes. The spacing between these image planes was 296 nm, which constituted a compromise between the desired coverage of all synapses, optimal axial resolution, and the time required for acquisition. To quantify the differences in staining intensity throughout the image stacks, we calculated the ratio between both antibody signal intensities for each pixel, and thus for each synapse.

The analysis was conducted with help of the related applications Fiji (http://fiji.sc) and ImageJ (http://rsbweb.nih.gov/ij). For semi-automated analysis, a collection of plugins was generated. These plugins can be downloaded, along with their source code and documentation at http://ratios.andlauer.net.

All plugins and functions described here, as well as this manual, have been written by Till Andlauer (*v*1.0, November 3rd 2017). You can contact me via till@andlauer.net.

1.1. Segmentation and thresholding

We wanted to quantify stainings at synapses and not weak background signals between synapses or even outside neuropils. We therefore first segmented the respective neuropil of interest from the rest of the brain (Fig. 1), using the Fiji ImageJ plugin *Segmentation Editor* (Schindelin et al., 2012; Schmid, 2010). We used a similar approach for segmentation of the MB calyx previously (Christiansen et al., 2011; Kremer et al., 2010). Masks for segmentation were saved in the Amira file format.

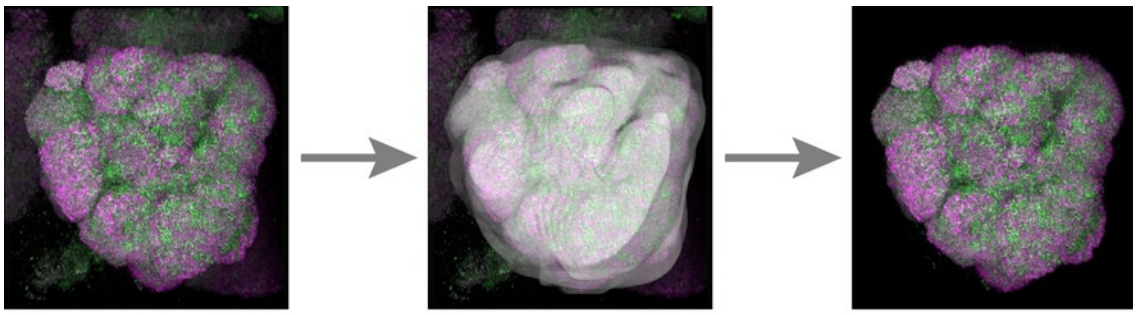

Fig. 1: Illustration of the segmentation of an AL from the surrounding neuropil.

Because all proteins we examined are mainly synaptic, their expression should be strongest at synapses. Thus, we applied a percentile threshold to each image to keep only pixels with high signal intensities.: in this manner, only pixels that were among

the brightest 20% were retained (Fig. 2). Dimmer pixels were set to an intensity value of zero. This 20% threshold was found to deliver an optimal separation of synaptic from non-synaptic signals, based on manual comparisons. Because intensities vary within image stacks, each stack was divided into substacks for the determination of thresholds. This procedure allowed for the identification of threshold levels suitable for each image plane, even when intensities varied.

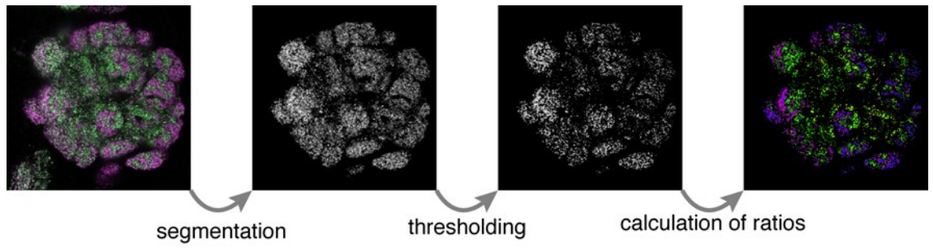

Fig. 2: Illustration of the steps segmentation, thresholding and calculation of ratios.

The algorithm for merging image stacks with segmentation masks and the subsequent thresholding was implemented into the ImageJ plugin *Mask Generator*. This plugin is, together with all other novel algorithms described here, part of the ImageJ package *Ratio Calculator*. To retain only the relatively brightest (synaptic) signal, segmented neuropils were thresholded in the following manner (plugin *Mask Generator*, class *Mask_Generator.java*):

1. *Segmentation*
   - 1.1. Generation of segmentation masks using *Segmentation Editor*.
   - 1.2. RGB merge of the individual channels of the original image stack.
   - 1.3. Conversion of the RGB image into a single 8 bit grayscale channel. Thus, all channels of the stack contribute equally to finding the threshold.
   - 1.4. All pixels in the image stack generated in step 1.3 that are not part of the segmentation mask from step 1.1, are set to an intensity of 0 (*Mask_Generator* function *createMask*).

2. *Percentile thresholding*
   - 2.1. In order to find a threshold that preserves only the brightest pixels in the masked image stack generated in step 1.4, percentile thresholding is used (Doyle, 1962).

     To understand the basic approach, consider an example were only the 20% of pixels that are the brightest should be kept. If all pixels that are among the brightest 20% were part of group *A*, all dimmer pixels would be in group *B*.

     To separate *A* from *B*, intensities of all pixels within the masked stack are counted and the threshold is set to the value separating pixels into the two desired populations *A* and *B*.

   - 2.2. If the image stack contains more than 30 slices (i.e. 30 single image planes), the stack is divided into substacks prior to computing the threshold. Each substack has a size of at least 30 slices.

     This is required, since absolute intensities within a stack vary: the deeper one penetrates into neuronal tissue, the lower the absolute intensity.

     The number of pixels per masked image plane is not constant. In the centre of the stack, masks typically cover a large portion of the image plane. By contrast, towards the upper and lower ends of the stack, often only a small area of the slice is included in the mask.

     Moreover, the pixels at each end of the stack are likely brighter or dimmer than the average intensity of the complete stack. For these two reasons, the outcome of percentile thresholding is dependent on the number of pixels included in the calculation of the optimal threshold.

     Accordingly, if thresholds are calculated for a substack that is too large, thresholds are not optimally representative for the absolute intensities of pixels in all slices. If, however, substacks are too small (1 substack = 1 slice, in an extreme case), calculated thresholds differ strongly between stacks and no homogenous population of pixels is acquired. A threshold calculated from 1000×1000 = 1,000,000 pixels is likely more reliable than one calculated from 10×10 = 100 pixels.

     Thus, a compromise regarding the size of substacks is required: substacks containing a minimum of 30 slices each were found to produce reliable thresholds. Comparisons with manually chosen values confirmed that these thresholds were suited for proper separation of bright, synaptic signal from dim, non-synaptic background staining.

   - 2.3. Substacks are determined in the following manner:
     - 2.3.1. n = number of slices contained in the image stack.
       If n>30, substacks are generated.
     - 2.3.2. Empty slices not containing any masked pixels are temporarily removed; e = number of empty slices.
     - 2.3.3. s = number of substacks = (n-e) / 30. s is rounded down.
       For example, if n=90 and e=7, s=83/30=2.
     - 2.3.4. The last substack can be smaller or larger than the others, to compensate for rounding errors. In the example shown above, the first substack contains 42 slices, the second 41.

   - 2.4. Calculation of the actual thresholds was adapted from Gabriel Landini's *AutoThresholder* class, part of the ImageJ code. In brief, histograms for each substack are calculated, followed by determination of the threshold value according to the *percentile* algorithm (Doyle, 1962), ported to MATLAB by Antti Niemisto in 2004 (GPL license). The percentage of pixels to be retained can be chosen by the user. Computation of thresholds was implemented in the *Mask_Generator* function *execThresh*.
   - 2.5. Next, thresholds calculated for substacks are interpolated, in order to assign an individual threshold to each slice. This had the following reason: if thresh-

olds varied between substacks, neighbouring slices at the ends of substacks could receive vastly different thresholds.

For example, slice *A* is the last slice of substack *S1*, slice *B* the first slice of substack *S2*. Slice *B* follows directly after slice *A* and the signal in both is probably highly similar. The threshold found for substack *S1* is 50, the one for substack *S2* is 60. Thus, without interpolation, the threshold for slice *A* is 50 and for slice *B* 60. By contrast, if thresholds are interpolated, both slices *A* and *B* receive the threshold value 55. At the same time, the centre slices of substacks *S1* and *S2* retain the values 50 and 60, respectively.

2.5.1. The percentile threshold determined in 2.4 is assigned to the centre slice of each substack.

2.5.2. Individual thresholds are interpolated for each single slice, so that a smooth transition from one threshold/slice to the next is achieved, spanning the borders of substacks. The thresholds calculated in 2.4 and assigned to the centre slices thus merely constitute the minimum/maximum values of the threshold distribution.

2.5.3. The average threshold of all slices is shown to the user as an indication.

2.6. Finally, the individual thresholds are applied to each slice, substacks are recombined into one stack and empty slices that were temporarily removed from the ends of the stack (step 2.3.2) are added again.

2.7. The final product of *Mask_Generator* is a 3D mask, indicating the positions of the pixels to be included in the calculation of ratios.

The thresholds were determined from a signal combining all channels of the recording (steps 1.2, 1.3).

3. For masks of antibody stainings containing two channels, *Mask Generator* was used with the percentile threshold 0.8.
For masks of the GFP signal, *Mask Generator* was used with the percentile threshold 0.95.

1.2. Calculation of ratios

8-bit images contain 256 (2^8) monochrome intensity values ranging from 0 to 255. Calculating the ratio between two 8-bit values in all 65536 (256×256) possible cases yields only 39641 different values. This is the case because there are, for example, 256 different possibilities for calculating the ratio 1 (1/1, 2/2, 3/3, etc.), 128 different ways of calculating 0.5 (1/2, 2/4, 3/6, etc.), and so forth.

Ratio values were calculated for every pixel of an image stack. To visualize ratios, all ratio values were ranked. Accordingly, the ratio 1/1 corresponds to the middle rank and 255/0 to the maximum rank. The resulting data was displayed as a 16 bit image stack and colours were mapped to the ranks (Fig. 3). This allowed us to highlight synapse populations with especially strong differences in CAZ protein intensity.

Fig. 3: Colour mapping for the visualisation of ratio values.
Ratio: Brp$^{C\text{-}Term}$ / *other*

Ratios were calculated in the following manner (plugin *Ratio Calculator*, class *Ratio_Calculator.java*):

1. *Generation of ranks (function rankGenerator, called by calcRatio)*

1.1. To save more memory during execution of the plugin, individual ratio values are not stored at all, unless requested by the user. Instead, merely the frequency with which each ratio value occurs is counted. However, the original ratio values can easily be reconstructed for each pixel, if the ratio image stack is saved in 16 bit format.

1.2. Each real ratio number is mapped in ascending order to an integer, ranging from 1 to 39641. Accordingly, the ratio 1/1 corresponds to 19821 and 255/0 to 39641.

1.3. Handling of the value 0

- Raw ratio values range from 0 (0/255) over 1 (1/1) to infinity (255/0). However, the value 0 has to be handled especially. First, divisions by zero (e.g., $x/0$) do not produce a real number. Second, a ratio of 0/0 has to be considered as 1, because in such a case the signal of both stainings is equally low. Third, it is desired that ratios show a smooth, uniform distribution, without large steps at the end of the distribution.
- In order to create a relatively uniform sequence of ratios, any ratio $0/n$, $1 \le n \ge 255$, was defined as $1/(255\times4) = 0.000928$ instead of 0; any ratio $n/0$ was defined as $255\times4 = 1020$ instead of *infinity*.
- To avoid divisions by 0, an alternative calculation of ratios is possible: the ratio r of two values a and b can also be calculated by the formula $r = (a\text{-}b)/(|a|+|b|)$ instead of $r = a/b$. This is implemented as an optional choice for the user.

  We have used the ratio $r = a/b$ for our analyses. The formula $r = (a\text{-}b)/(|a|+|b|)$ produces a uniformly distributed ratio, ranging from

-1 to +1. However, information is lost: the calculation yields, in the case of a=4 and b=1, r=0.6. By contrast, the formula $r = a/b$ produces, in the same case, the result r=4. In the latter case, it is directly recognizable that a is enriched 4-fold over b.

1.4. A ratio/rank table is generated containing all possible 8 bit intensity values for pixels a and b, the value r of the ratio a/b, as well as the corresponding, uniformly distributed rank (see step 1.1).
A random example from this matrix is: a = 1, b = 2, *ratio* = 0.5, *rank* = 9911.
The complete table can be displayed by executing the plugin *Show ratio table*.

2. *Calculation of ratios (function calcRatio)*

2.1. For each pixel (voxel) of the 3D image stack, the algorithm checks whether the pixel is part of the mask calculated by *Mask_Generator*. If the pixel is not part of the mask, the pixel is ignored.

2.2. If the pixel is part of the mask, absolute intensity values for both channels are determined. Corresponding ratios and ranks are looked up in the matrix generated by *rankGenerator* and thus do not need to be calculated again.

2.3. Sometimes it cannot be avoided that one or both of the channels are oversaturated during image acquisition. All oversaturated pixels have an intensity of 255, irrespective of the original brightness of that spot. This could possibly distort the data and bias the intensity distribution towards the value 255. Therefore, all ratios derived from the value 255 could and should be omitted from the calculation of statistics. In our analyses, ratios were ignored if one of the two channels was saturated for the respective pixel (i.e. had a value of 255) to avoid skewed statistics.

2.4. If the user chooses to save ratios as an image, all ratio values are stored as 16 bit values in form of an image stack.

2.5. For calculation of ratio statistics and histograms, ratio values are not stored. Instead, frequencies of ratios are directly counted for all possible 39641 bins. This approach saves a large amount of memory. In this manner, statistics and histograms can be calculated even for huge stacks, without running out of memory. Thus, in case of large image stacks, the option to generate a ratio image should be deselected.

2.6. The ratio image is displayed with a special colour code (lookup table), shown in Fig. 3. Ratios close to 1:1 are shown in black, negative values, in descending order, in magenta-blue-cyan, positive values, in ascending order, in green-yellow-red.
This lookup table can be downloaded as the file *Ratio Spectrum.lut*.

3. *Calculation of statistics and histograms*

3.1. To summarize ratios of an image stack, rank-based statistics are calculated: minimum, lower quartile, median, upper quartile, and maximum values (function *calcStats*).

3.2. Histograms are computed and normalized by the total amount of data, to ease comparisons between image stacks containing different amounts of

slices and different masks (function *calcHisto*). Subsequently, histograms can be scaled with a multiplication factor specified by the user.

3.3. Statistics, as well as normalized and original histogram values, are saved and can be used for additional analyses (see below).

1.3. Ratios specific for cell types

We used cell-specific expression of GFP to allocate individual puncta of a certain ratio to defined neuron types. We expressed Brp$^{short\text{-}GFP}$ constructs with cell-specific Gal4 drivers; Brp$^{short\text{-}GFP}$ is a fragment of Brp that depends on endogenous Brp for localization to AZs and accumulates proportionally to full-length Brp isoforms (Fouquet et al., 2009; Kremer et al., 2010; Schmid et al., 2008). Brp$^{short\text{-}GFP}$ therefore labels all presynapses of the cells it is expressed in that contain endogenous Brp. Moreover, expression of the construct does not stimulate the generation of synapses *de novo* (Kremer et al., 2010). Thus, by examining only synapses marked by Brp$^{short\text{-}GFP}$, we were able to observe CAZ protein ratios at identified synapses.

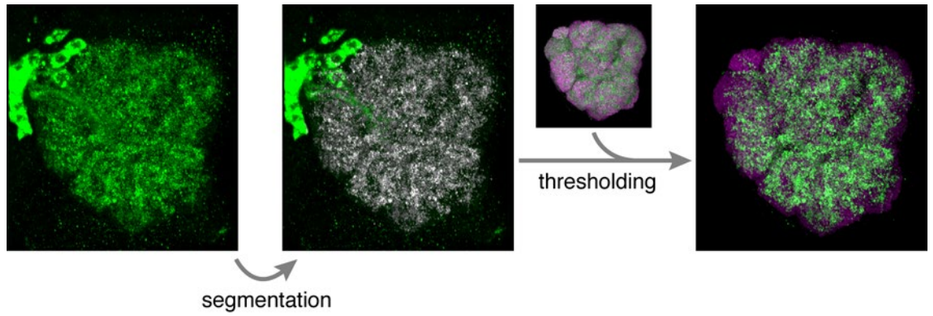

Fig. 4: Illustration of segmentation and thresholding of GFP signals.
Note that the antibody staining of the complete neuropil was just added as a reference here, it does not influence calculation of the threshold for the GFP signal.

For the GFP-based analysis, first the complete neuropil was segmented, as previously described (Fig. 2). Next, segmentation was refined for GFP-positive synapses; cell bodies and axons strongly labelled by GFP were removed (Fig. 4). Subsequently, a 95% percentile threshold was applied to the segmented GFP signal. Thus, only GFP-positive pixels belonging to the brightest 5% were kept. Thereafter, ratios were calculated once for the complete neuropil and once for the GFP-positive mask.

1.4. Analyses of ratiometric results and statistics

1.4.1. Analyses using the ImageJ plugin *Ratio Analysis*

Analyses of ratio data can be conducted using the plugin *Ratio Analysis* (class *Ratio_Analysis.java*). For each such analysis, text files containing the output as well as scripts are generated that can be used for plotting the data with GnuPlot. However, **use of the ImageJ *Ratio Analysis* plugin is discouraged**. These functions are still included in the package for compatibility but they are not actively maintained. Instead, the functions provided for *R* should be used (see below).

1. *Calculation of mean statistics (function basicAnalysis)*
   - *Ratio Calculator* computes the following summary statistics for each image stack: minimum, lower quartile, median, upper quartile, and maximum ratio value. For the calculation of mean statistics, a data set consists of a number of such values. For example, the median ratios of ten different image stacks, all belonging to the same genotype. Each image stack is one sample.
   - *Ratio Analysis* calculates mean values and standard errors of the mean (SEMs) of these parameters for a number of image stacks. If s is the standard deviation of a data set and n is the number of samples, the SEM e is $e = s/\sqrt{n}$.
   - Alternatively, median values and standard errors of the median can be calculated, according to Sachs, 2003:
     standard error of the median = $e = (a\text{-}b) / 3.4641$,
     where a is the $a = (n/2 + \sqrt{(3\times n)}/2)$th observation
     and b is the $b = (n/2 - \sqrt{(3\times n)}/2)$th observation,
     each rounded up to the next integer; n = number of samples.
   - As a third possibility, median values and median absolute deviations can be calculated. The median absolute deviation corresponds to the median of the differences of the individual values to the median. Thus, first, each value is subtracted from the median. Second, the median of these values is the median absolute deviation.

2. *Calculation of mean histograms (function basicAnalysis)*
   - Either mean or median values for the frequency bins of several histograms are calculated. Each individual histograms is generated by *Ratio Calculator* from one image stack and constitutes one sample of a data set.
   - The available options for calculation of errors are as described for *mean statistics* (SEM, standard error of the median, median absolute deviation). In the analysis presented here, mean frequencies plus SEMs were used.
   - In addition, the function generates an image similar to a contact sheet, which shows a comparison of all individual histograms.

3. *Comparison of several data sets (groups) (function compAnalysis)*
   - This module uses data generated by module 1, *mean statistics*.
   - It generates scripts that allow GnuPlot to plot combined graphs for several data sets (e.g., for several genotypes).

   - In addition, the data sets are being compared in groups of two, by either Mann-Whitney U-tests or T-tests. For these tests, the Apache Commons Library *Math* v3.3.0 is employed, classes *MannWhitneyUTest* and *TestUtils*, respectively.

4. *Normalization of a data set to a reference data set (module "Ratio of two groups", function normAnalysis)*
   - Both data sets need to be dependent on each other. A possible application is the normalization of a subset of a neuropil to the complete neuropil. For example, a GFP-labelled subset is compared to the complete neuropil surrounding it. Thus, the first sample of data set 1 is derived from a subset of pixels used for calculation of the first sample in data set 2.
   - For each sample, values of the first set (e.g. the median ratio of the GFP label, first brain) are divided by the corresponding values of the second set (e.g. the median ratio of the entire neuropil, first brain).
   - Options for uncertainties (SEM, standard error of the median, median absolute deviation) are analogous to the ones in module 1, *mean statistics*.
   - Two data sets are compared, each carrying an uncertainty of its own. Thus, final uncertainties are calculated using Gaussian error propagation (Papula, 2008). Calculation of the uncertainty e in case of division of two values x,y with corresponding uncertainties s_x,s_y: $e = \sqrt{((s_x/x)^2 + (s_y/y)^2)}$.

5. *Generation of fingerprints (function fingerAnalysis)*

   This module uses data generated by module 4, *compAnalysis*.

   Ranks are calculated in a similar manner as in *Ratio Calculator*. However, here, 8 bit ranks are computed, ranging from 0-255. Moreover, a different type of ratio is used, as explained in the next items.

   5.1. The input file contains pairs of samples that are to be compared:
   e.g., line 1: a, statistics of complete neuropil, genotype 1;
   line 2: b, statistics of pixels labelled by GFP, genotype 1.

   5.2. Median values are used to calculate the ratio $r = (|b|\text{-}|a|)/(|a|+|b|)$.

   5.3. The corresponding 8 bit rank is looked up in the rank matrix.

   5.4. Calculated ratios are saved as text files and ranks are displayed/saved as images, with the same colour code (lookup table) used for display of ratio values in *Ratio Calculator* (see Fig. 3). Ratios close to 1:1 are shown in black, negative values in magenta-blue-cyan, positive values in green-yellow-red.
   These images constitute a signature for each analysed situation and are thus called *fingerprints*.

6. *Calculation of relative distributions (function propAnalysis)*

   This modules determines the relative distribution of ratio values, i.e. the relative proportions of low, balanced, and high ratio values in a data set.

   6.1. Here, ratio values between 0.95 and 1.05 are considered as balanced, ratios <0.95 as low, and ratios >1.05 as high.

6.2. Frequencies of ratios are counted for each of these three bins, final frequencies are relative to the total amount of data.

6.3. In addition, this module calculates the number of pixels that were used for the calculation of ratios in each sample, as well as the mean number of pixels included in a complete data set.

6.4. Data generated by this module was used for calculating the Bayesian conditional probability that a single, random spot with a certain property r (e.g. a high ratio a/b) is part of a certain population t of pixels: $p(t|r) = p(r|t) \times p(t)/p(r)$.

- $p(t)$ is the probability that any pixel in the neuropil is part of the population t. For example, this can constitute the relative amount of pixels labelled by GFP within a neuropil. This value can thus be computed by dividing the numbers of pixels positive for GFP by the total number of pixels covering this neuropil.
- $p(r)$ is the probability that any random pixel in the neuropil has the distinct ratio r. This corresponds to the proportion of all pixels with this property r in the complete neuropil (e.g., a high ratio a/b).
- Finally, $p(r|t)$ is the probability that any pixel positive for GFP (and thus part of t) has the distinct ratio r, which corresponds to the proportion of all pixels with this property r in the GFP-labelled subset t.
- Thus, all three probabilities required for the calculation of conditional probabilities are estimated by this module.

1.4.2. Analyses using functions for *R*

Functions for the analysis of ratio data in *R* (https://cran.r-project.org/) are provided. These functions only require frequency tables generated by *Ratio Calculator* as an input. Such tables are generated by checking the option *Show unbinned original frequencies*. When saved as a file, the default file name is *Original Data.xls*. The *R* function *genQuant()*, part of the file *ratioFunctions.R*, expands these rank frequencies to frequencies of ratios and calculates summary statistics.

1. *Calculation of weighted ratio quantiles using genQuant() and calcQuant()*
   1.1. Unbinned frequency files belonging to one experimental group are loaded using *readFiles()*.
   1.2. Ranks are merged with the file *ratios.txt* containing a translation table from ranks to ratios. This file can be generated using the plugin *Show ratio table* in ImageJ.
   1.3. Calculation of weighted quantiles using *wtd_quant()*. The method used is described in the book *Relative Distribution Methods in the Social Sciences* (Handcock and Morris, 1999).
   1.4. Cell-specific (GFP-based) ratio quantiles are normalized to the respective ratio quantiles of the surrounding neuropile.

2. *Further analyses and plots*
   2.1. Median ratios can be plotted using boxplot_ab() and boxplot_gal4()

   2.2. Median ratios can be compared using mwu_ABratios() or mwu_gal4ratios()
   2.3. Histograms can be generated using generateHisto()
   2.4. Histograms and violin plots can be plotted using plotBot ()

1.5. Additional modules

1. For masks of antibody stainings containing three channels (plus the segmentation mask), *4-Channel Mask Generator* (class *Mask_Generator_Triple.java*) can be used. This plugin functions analogously to the standard *Mask Generator*.
2. For the calculation of intensity statistics, *Intensity Calculator* can be used (class *Intensity_Calculator.java*), which functions analogously to *Ratio Calculator*.
3. For the analysis of intensity data, *Intensity Analysis* can be used (class *Intensity_Analysis.java*), which functions analogously to *Ratio Analysis*.

1.6. References

Christiansen, F., Zube, C., Andlauer, T.F.M., Wichmann, C., Fouquet, W., Owald, D., Mertel, S., Leiss, F., Tavosanis, G., Farca Luna, A.J., et al. (2011). Presynapses in Kenyon Cell Dendrites in the Mushroom Body Calyx of *Drosophila*. J Neurosci 31, 9696–9707.

Doyle, W. (1962). Operations useful for similarity-invariant pattern recognition. Journal of the ACM 9, 259–267.

Hallermann, S., Kittel, R.J., Wichmann, C., Weyhersmüller, A., Fouquet, W., Mertel, S., Owald, D., Eimer, S., Depner, H., Schwärzel, M., et al. (2010). Naked Dense Bodies Provoke Depression. J Neurosci 30, 14340–14345.

Handcock and Morris (1999). Relative Distribution Methods in the Social Sciences. Springer Verlag, ISBN 0387987789.

Inoué, S. (2006). Foundations of Confocal Scanned Imaging in Light Microscopy. In Handbook of Biological Confocal Microscopy, J.B. Pawley, ed. (New York: Springer), pp. 1–19.

Kremer, M.C., Christiansen, F., Leiss, F., Paehler, M., Knapek, S., Andlauer, T.F.M., Förstner, F., Kloppenburg, P., Sigrist, S.J., and Tavosanis, G. (2010). Structural long-term changes at mushroom body input synapses. Curr Biol 20, 1938–1944.

Liu, K.S.Y., Siebert, M., Mertel, S., Knoche, E., Wegener, S., Wichmann, C., Matkovic, T., Muhammad, K., Depner, H., Mettke, C., et al. (2011). RIM-binding protein, a central part of the active zone, is essential for neurotransmitter release. Science 334, 1565–1569.

Maglione, M., and Sigrist, S.J. (2013). Seeing the forest tree by tree: super-resolution light microscopy meets the neurosciences. Nat Neurosci 16, 790–797.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., et al. (2012). Fiji: an open-source platform for biological-image analysis. Nature Methods 9, 676–682.

Schmid, B. (2010). Computational tools for the segmentation and registration of confocal brain images of *Drosophila melanogaster*. Doctoral Thesis. Julius-Maximilians-Universität Würzburg.

Südhof, T.C. (2012). The presynaptic active zone. Neuron 75, 11–25.