Cell Profiles Script Manual

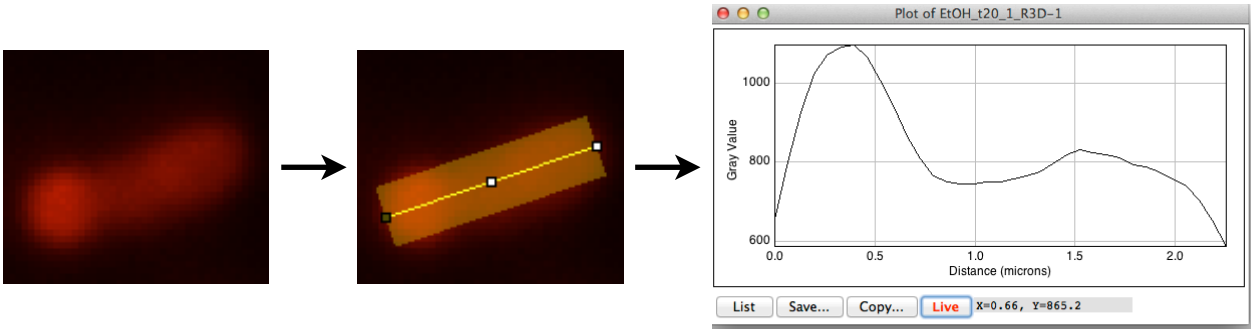
v2.5 - March 2016

Introduction

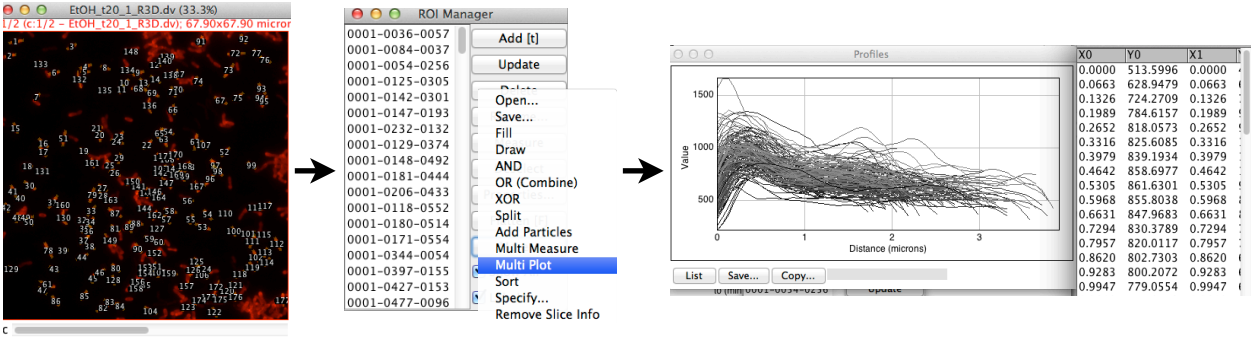
This R script will take a series of fluorescent profiles acquired in ImageJ/Fiji, arrange them from shortest to longest, then use Hadley Wickam's ggplot2 to plot the profiles out as a vertical stack. Each horizontal line represents the fluorescent intensity profile of a single cell. By ordering cells according to length, general localization patterns over the cell cycle can be easily visualized.

Data acquisition in ImageJ

1. Work on only one image at a time. The ROI manager in ImageJ is not so sophisticated that it will keep track of multiple windows.
2. Select cells in your image and save one at a time as ROIs (press 't' once a cell is selected). Use the live profile view to choose your selections carefully.



3. If working with a stack, remove slice info from the ROIs using the ROI manager (or else reduce your stack to one image), and ensure the correct layer is active.
4. Choose multiplot from the ROI manager, then copy the data points from the list view into an Excel spreadsheet.



5. Save the spreadsheet as a .csv file.

note: although the labels for the ggplot graphs assume your source images were calibrated at the um scale, they are trivial to change to other scales.

Cell Profiles package contents

The script now includes three parts:

a) cell profiles function.R
   - this is the bulk of the code to order/align/etc the profiles

b) example plots.R
   - this is the code to load the csv file, call upon the cell profiles function, and generate the ggplot2 graphs

c) profiles.csv
   - example data set from 100 cells, used for example plots

How to use this script: workflow example

1. Open cell profiles function.R. Select all and execute.
   - This loads `cellProfiles()` & `cellProfileTruncate()` for the current R session.
   - Unlike before, do not modify any part of this file.

2. Open example plots.R, set correct working directory and change options.
   a. Execute the first section to install & load the required packages. (this only needs to be done once for each R session)
   b. Edit the `setwd()` command to the correct directory containing your data, and `read.table()` to match the file name of the .csv file.
   c. Set `cellProfileTruncate()` to truncate the ends of your cells, if/as needed.
   d. Edit the `cellProfiles()` command to modify the script's behavior.

3. Execute. Calculate profiles and generate ggplot2 graph
   a. Select from the `setwd()` command through to `dev.new();g` & execute.
   b. If needed, adjust the limits of the color scale by changing the `range=c(0.02,0.98)` option in `cellProfiles()`, then execute again.

NOTE: Everything was tested and operational against the latest version of R, ggplot, and other required packages at the time of release. However, subsequent updates could (and have in the past) cause the script to fail to run. Let me know if this happens, as usually the fixes are fairly simple.

Function details

cellProfiles()

This function adjusts contrast, aligns, and orders cells for graphing.

usage	`cellProfiles(data=NULL,position="center",align="native", reverse=FALSE,contrast="native",range=c(0.02,0.98))`
example	`profileResults <- cellProfiles(data=dtable,contrast="norm")`

argument	examples	description
data	= dtable	the raw data table. requires wide-format data with alternating paired columns of cell length and intensity values. **(required)**
position	= **"center"** = "left"	• **shifts cells so that mid-cell is centered. (default)** • shifts cells so that their origins are aligned.
align	= **"native"** = "orient" = "random" = "reuse"	• **retains original cell orientations. (default)** • flips orientations so brightest halves are all on the right side. this is weighted by a log(1:10) scale for each cell half. • randomly chooses half of the profiles and reverses their orientation. • reuses the last calculated orientations, if they exist.
reverse	= **FALSE** = TRUE	• **no change to final orientations. (default)** • reverses the final orientations.
contrast	= **"native"** = "norm" = "max"	• **no contrast adjustments; native fluorescence values. (default)** • for each cell, divides by its mean fluorescence. • for each cell, rescales the intensities to range from 0 to 1.
range	= **c(0.02,0.98)** = c(0.05,0.995)	a numeric vector of length 2, ranging from 0 to 1. defines the lower and upper percentiles of the data to include in the color scale range.

This function exports a number of variables of use for graphing and other purposes. Access them from the variable you saved the results into (`profileResults` in the example above), in the form: `profileResults$variableName`.

variable	description
lim_or_dtable	a long-format data table used for making ggplot2 'tile' graphs. the fluorescence intensity values in this table are truncated by the limits set by the `range` option.
or_dtable	lim_or_dtable, but without truncation applied.
prop_dtable	or_dtable, but with cell length converted into fraction of total cell length.
form_dtable	the fluorescence intensity data from or_dtable, presented in a wide-format. save this to a csv file in your current directory using: `write.csv(as.matrix(profileResults$form_dtable), file="cellprofiles wide format.csv")`

variable	description
data	the input data table
max	maximum fluorescence value of all cells (pre-truncation)
min	minimum fluorescence value of all cells (pre-truncation)
med	median fluorescence value of all cells (pre-truncation)
mid	midpoint between min & max (pre-truncation)
maxlength	the length of the longest cell

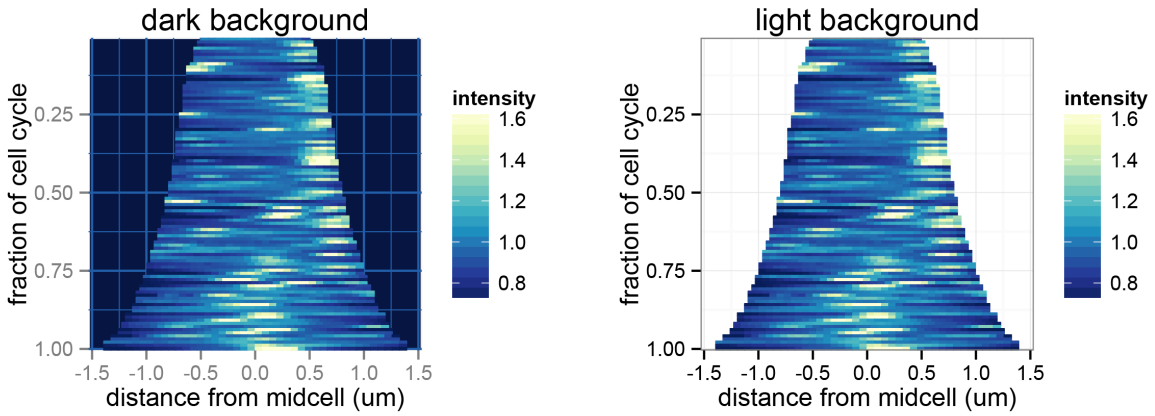
cellProfileTruncate()

This function removes a set number of rows from the beginning and end of each cell. Use to manually correct for profile acquisition artifacts that can occur at cell poles when using wide profile lines (poles can appear darker than they should, since the rectangular line profile will include a significant proportion of dark background at the cell poles).

usage	`cellProfileTruncate(data=NULL,adjust)`
example	`dtable <- cellProfileTruncate(data=raw_dtable,1)`

argument	examples	description
data	=raw_dtable	the raw data table. requires wide-format data with alternating paired columns of cell length and intensity values. **(required)**
adjust	2 0	a positive integer representing the number of rows to trim from the top and bottom of each cell profile. **(required)**

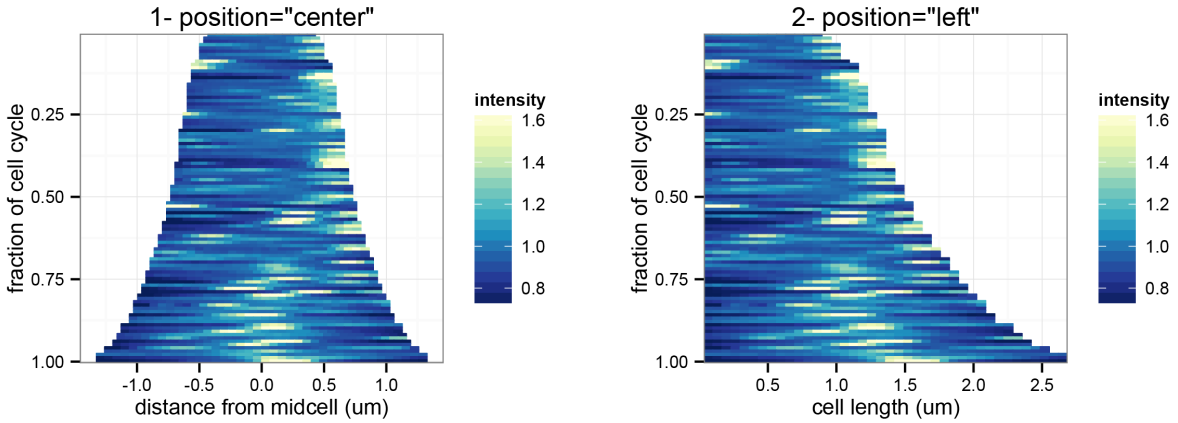
As an alternative to slightly truncating the cell poles, the graph can be given a dark background color into which the problem regions will blend. To the ggplot code, add:
`+theme(panel.background=element_rect(fill="#061542"),panel.grid.minor=element_line(color="#225EA8"),panel.grid.major=element_line(color="#225EA8"))`



Example Graphs

Position

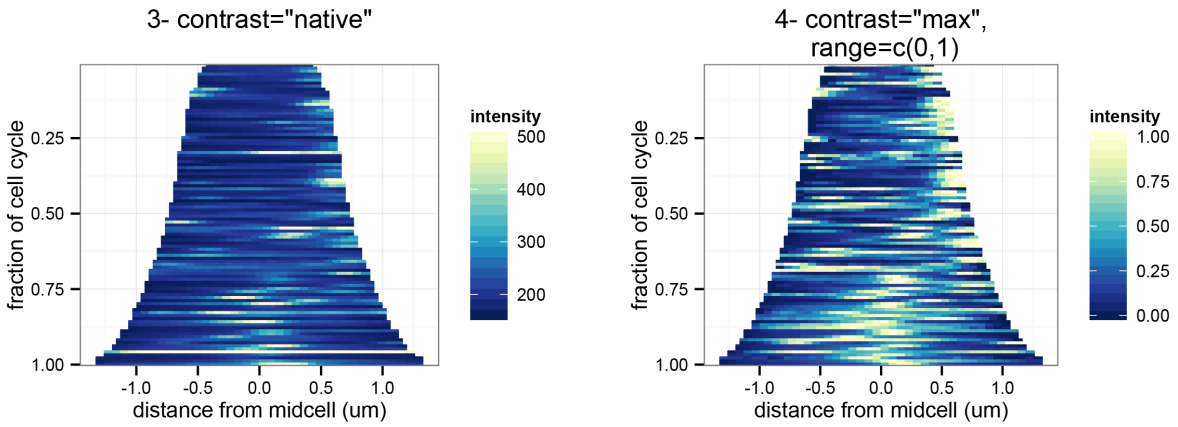
Centered graphs make it easier to see midcell localization patterns, although the x-axis is somewhat less intuitive. If for some reason you need to have cells on the right, use the ggplot2 option `+scale_x_continuous(trans="reverse")`.



Contrast

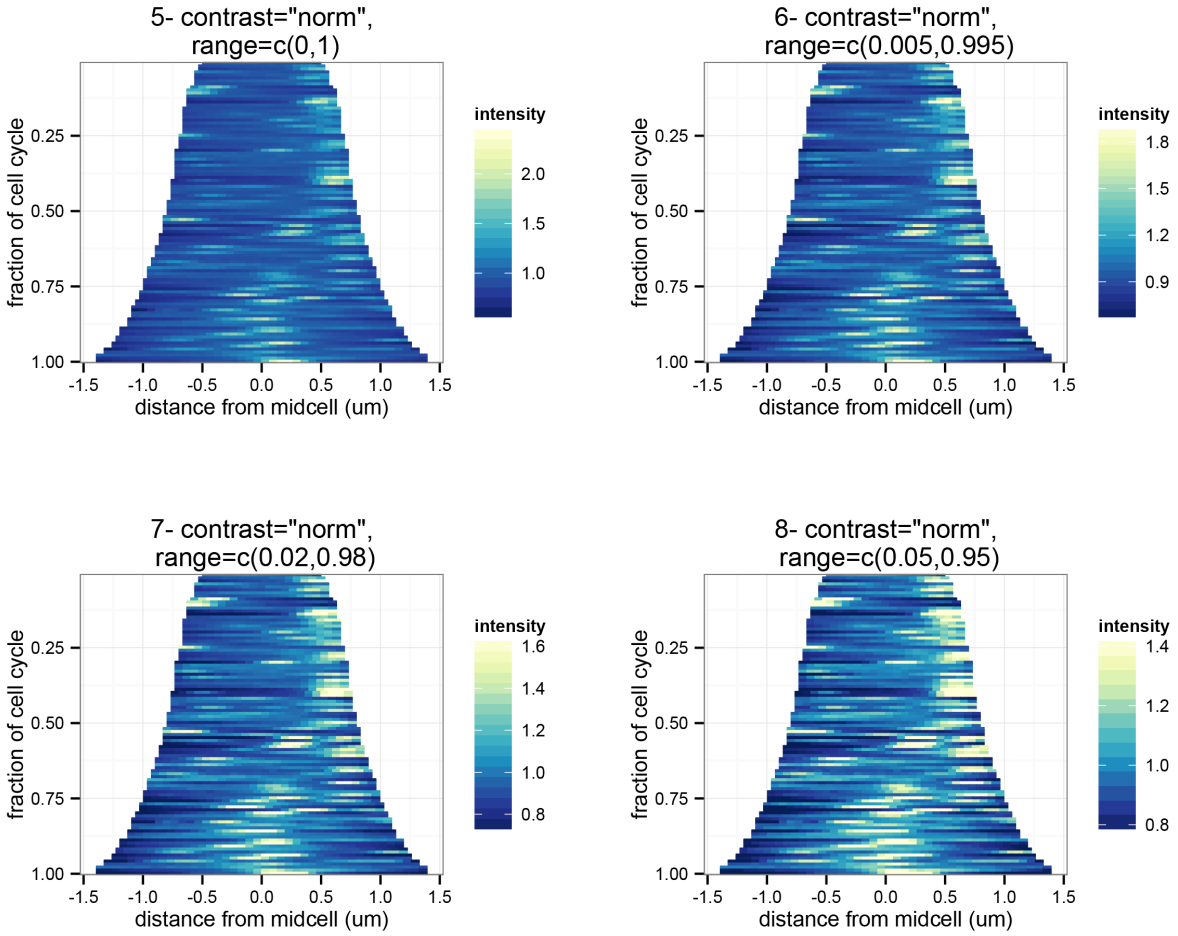
Ideally, native contrast could always be used. However, variations in fluorescence intensity between cells can make visualization difficult, and particularly intense fluorescence can throw off the color scale (although this is mitigated somewhat by the default `range=c(0.02,0.98)`). Note the bright streaks in graph 3.

A simple alternative to native contrast is to simply stretch the range of fluorescence values in each cell to fit between 0 and 1. This maximizes contrast for each cell, but can over-emphasize cells with low contrast (eg, cells with diffuse fluorescence). In graph 4, `range=c(0,1)` was used because further clipping of the color scale is unnecessary.



One way to resolve differences in fluorescence is by dividing each value in the cell by the cell's average fluorescence. This emphasizes *differences* in profile fluorescence, and makes it easier to see the fluorescence patterns in all cells.

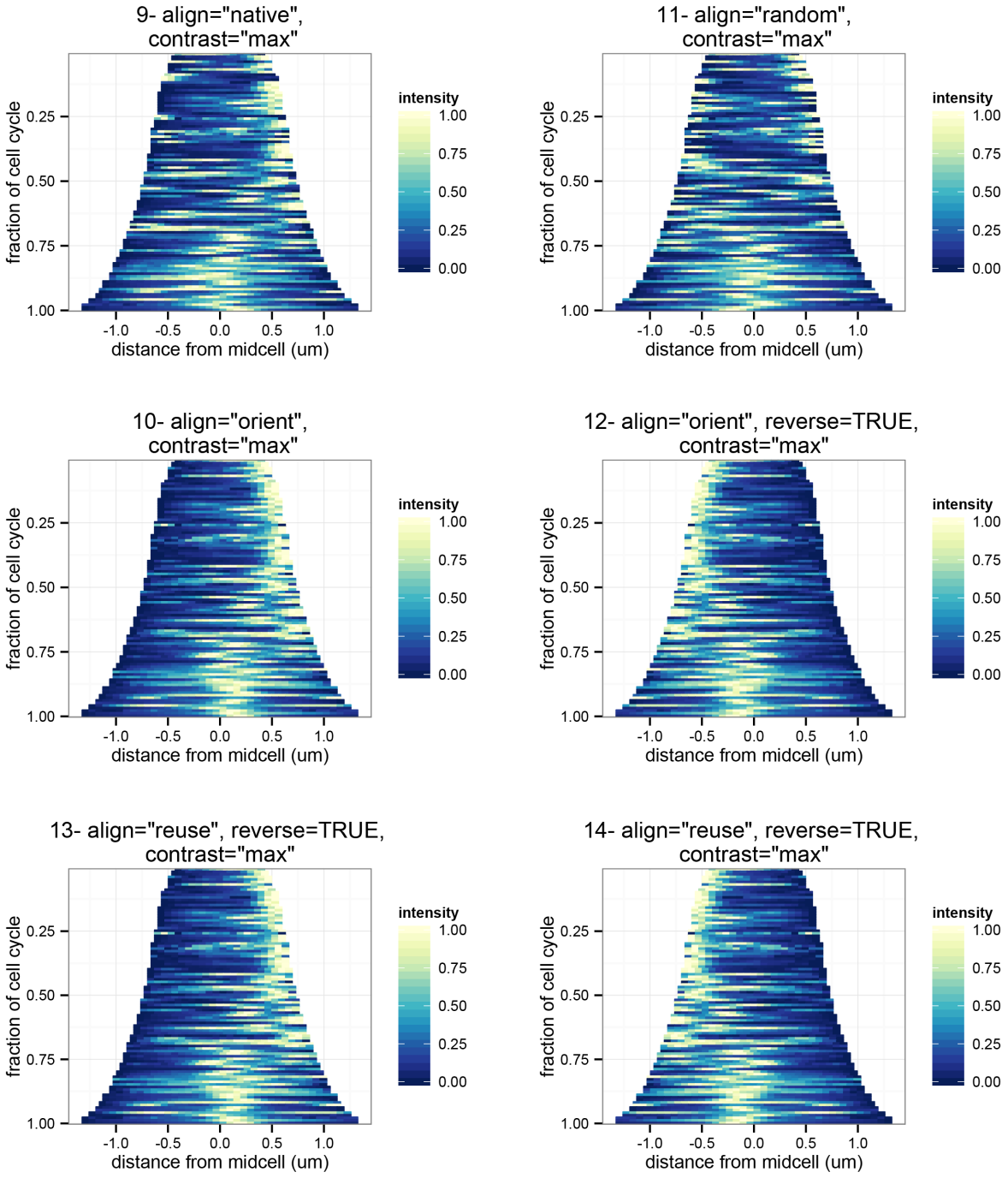
Graphs 5-8 demonstrate how different range settings can have significant impacts on the contrast of the final graph. The range often needs to be fine-tuned for each data set, depending on the distribution of high & low fluorescence values.



Alignment

There are several options for adjusting the orientations of individual cell profiles. Aside from the native orientation (graph 9), cells can be randomly flipped (graph 11), or aligned so that the brightest sides of all cells are together (graph 10).

The orientations of all cells can be flipped by setting `reverse=TRUE` (compare graph 10 to graph 12). The last calculated orientations can also be reused (graphs 13 & 14).



Additional graph types & examples

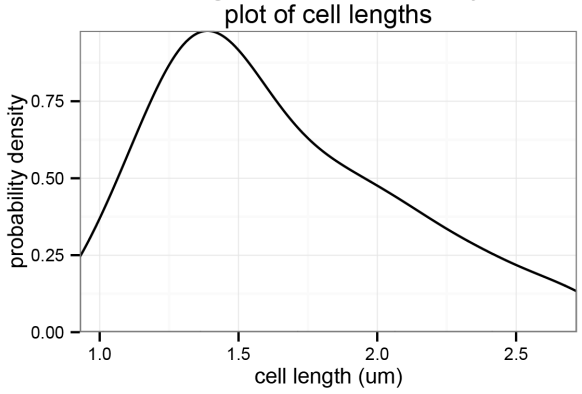
This data can also be used to make other types of graphs; refer to the code in example plots.R for full details.

From the raw data alone, the distribution of cell length can be graphed as a gaussian kernel density (graph 18) or as a histogram (not shown, but simply change “density” to “histogram” in the code for graph 18).

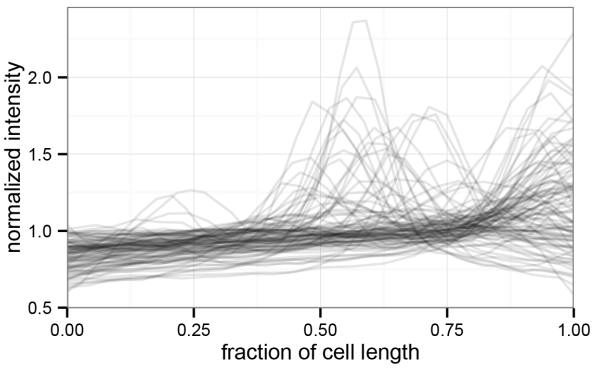
The fluorescence profile plots can also be viewed as overlaid line plots of each cell (graph 15). These line plots can be colored to show differences in localization trends between the shorter cells (red) and longer cells (blue) (graph 16).

The mean line plots for each category, along with confidence intervals, can also be plotted (graph 17). For the averages to plot properly, each profile is first interpolated to the same number of points (this ensures that they are aligned for the purposes of averaging), then the cells are split evenly into three groups of short, medium, and long cells. The 95% confidence interval is generated from a nonparametric bootstrap that does not assume normality.

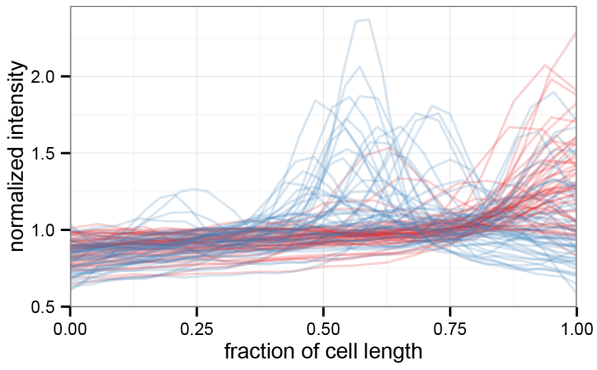
18- gaussian kernel density plot of cell lengths



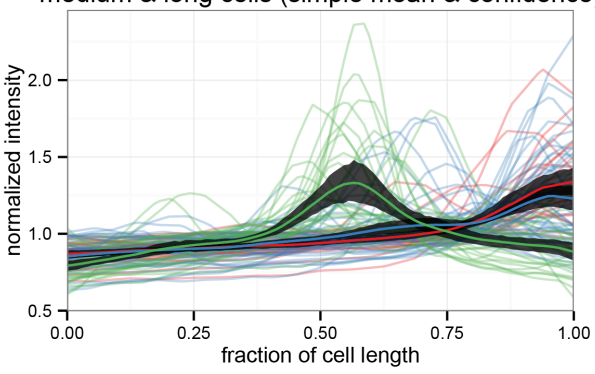
15- line plots of individual profiles



16- line plots split between short & long cells



17- line plots split between short medium & long cells (simple mean & confidence)



Two-color colocalization graphs

Demographs can be constructed from two separate signals, and even overlaid to reveal colocalization trends over the cell cycle. Refer to the code in example plots.R for full details.

Below is an example with one sample data aligned to the left for one channel, and right for the other channel. Where they overlap, the green & red (or orange and blue) colors combine to yellow (or white).