Lifetime-ratio analysis in the posterior lateral line primordium

Erika Donà, Joseph D. Barry, Guillaume Valentin, Charlotte Quirin,
Anton Khmelinskii, Andreas Kunze, Sevi Durdu, Lionel R. Newton,
Ana Fernandez-Minan, Wolfgang Huber, Michael Knop, Darren Gilmour

October 18, 2014

Contents

1 Introduction

This document outlines the image analysis pipeline used to extract fluorescence intensity ratios (referred to as 'lifetime ratio') from image stacks of the posterior lateral line primordium. All the code required to reproduce the results presented in the main text is supplied. One example primordium dataset was chosen for analysis in this vignette, but analyses of all other primordia were performed in exactly the same way.

For this analysis the following confocal images acquired with identical imaging settings were required:

1. Sample image: dual colour z-stack of a primordium.
2. Sample background image: dual colour z-stack of a sample-free area in the dish.
3. Ratio-normalisation image: dual colour image of purified mCherry-sfGFP fusion protein, diluted in PBS.
4. Ratio-normalisation background image: dual colour image of PBS.

Here green and red fluorescent channels are referred to as GFP and RFP. When ratio comparison occurred among samples acquired with identical conditions, 3 and 4 were not required.

2 Loading required packages

The analysis required the R packages `EBImage` and `parallel`. The following R code may be used to obtain the packages.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite(c("EBImage", "parallel"))
```

The experiment data package `DonaPLLP2013` must also be loaded.

```
> library("DonaPLLP2013")
```

3 Loading image data

Microscopy images of a posterior lateral line primordium (PLLP) were read into R as follows. An example has been bundled with the loaded data package.

```
> xGFP <- readImage(system.file("extdata/cxcr4b_02C1.tif", package="DonaPLLP2013"))
> xRFP <- readImage(system.file("extdata/cxcr4b_02C2.tif", package="DonaPLLP2013"))
```

A maximal projection of the z-stack was used to visualise the sample.

```
> dim(xGFP)
```

```
[1] 2048  300   29
```

```
> xGFP.proj <- apply(xGFP, c(1, 2), max)
> writeImage(normalize(xGFP.proj), "PLLPanalysis-displayImages.jpeg")
```

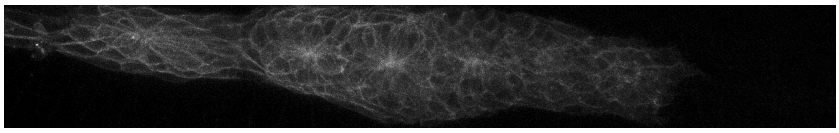

Figure 1: Maximal z-projection of a posterior lateral line primordium (green channel) before any background subtraction or smoothing.

4 Defining parameters and setting units

The voxel dimensions in this image were 0.1318 x 0.1318 x 1 μm, information that was required later in our analysis pipeline. From this, the dimensions of the field of view were calculated. We chose 5 microns as the characteristic length scale for our analysis (corresponding to half the approximate diameter of primordium cells), and converted this to units of pixels for later use.

```
> Lpx <- c(x=0.1318, y=0.1318, z=1)
> Lbox <- Lpx*dim(xGFP)
> Lbox
```

```
       x        y        z
269.9264  39.5400  29.0000
```

```
> Leff <- round(5/Lpx["x"])
> Leff
```

```
 x
38
```

`Lbox` was the length of the measurement box in microns, and `Leff` the characteristic length in units of pixels (must be integer).

5 Background subtraction

The mean of the sample background image was subtracted for each fluorescence channel.

```
> xbckGFP <- readImage(system.file("extdata/bgC1.tif", package="DonaPLLP2013"))
> xbckRFP <- readImage(system.file("extdata/bgC2.tif", package="DonaPLLP2013"))
> xGFP <- xGFP-mean(xbckGFP, na.rm=TRUE)
> xRFP <- xRFP-mean(xbckRFP, na.rm=TRUE)
```

6 Smoothing

Speckling due to Poisson noise was observed in the samples. For visualisation purposes we decided to reduce the pixel noise by use of a low-pass filter. We filtered each z-slice using a 2d Gaussian filter 5 pixels wide as this was approximately the width of the PSF in our imaging conditions. We required the filter size to be an odd integer.

```
> Lpsf <- round(0.5/Lpx["x"])
> Lpsfodd <- ifelse(Lpsf%%2 == 0, Lpsf+1, Lpsf)
> z <- makeBrush(size=Lpsfodd, shape="gaussian", sigma=Lpsfodd/2)
> x2GFP <- filter2(xGFP, filter=z)
> x2RFP <- filter2(xRFP, filter=z)
> x2GFP.proj <- apply(x2GFP, c(1, 2), max)
> writeImage(normalize(x2GFP.proj), "PLLPanalysis-gaussianSmoothing.jpeg")
```

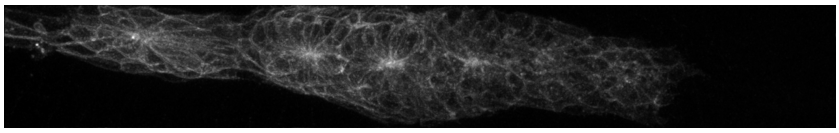

Figure 2: Maximal z-projection of a posterior lateral line primordium after background subtraction and smoothing.

7 Adaptive thresholding

Adaptive thresholding was performed to obtain a mask of the membranes of the primordium. A thresholding value of 0.01 was used in our analysis, while the size length of the thresholding box was set to the characteristic length of a cell. Basic erosion and dilation operations were also performed to remove remaining noise. As these operations modified the thickness of the membrane mask, we used the same filter sizes for the analysis of all of our samples. For this study, we verified that changing the mask thickness over one order of magnitude had negligible effects on the ratio; see Section 15.

We chose to define the mask using only the GFP channel for two reasons. First, the GFP channel was considered as the reference channel in our analysis since it was a better descriptor of protein abundance due to the fast maturation kinetics of sfGFP. Second, sfGFP signal was more restricted to the plasma membrane than the specific RFP used here, TagRFP, which showed greater perdurance in intracellular vesicles.

```
> mk <- function(size) makeBrush(size, shape="disc")
> mask <- thresh(x2GFP, w=Leff, h=Leff, offset=0.01)
```

```
> mask <- erode(closing(mask, mk(Lpsfodd)), mk(Lpsfodd-2))
> maskSlice <- mask[, , 20]
> writeImage(maskSlice, "PLLPanalysis-maskSlice.jpeg")
> maskDensity.proj <- apply(mask, c(1, 2), mean)
> writeImage(normalize(maskDensity.proj), "PLLPanalysis-maskDensity.jpeg")
```

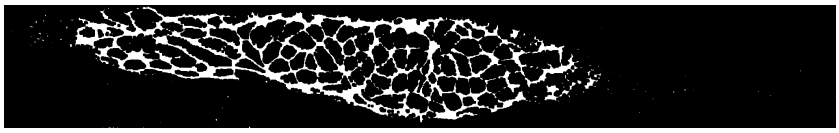

Figure 3: Binarized mask of primordium membrane for z-slice 20. White pixels indicated segmented membrane.

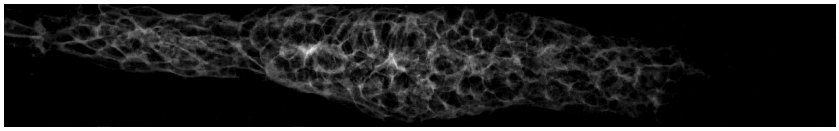

Figure 4: The mean projection in the z-direction of the membrane mask gave an impression of membrane density, and was useful for visualisation purposes.

8 Sample alignment along axis of migration

To measure the RFP/GFP intensity ratio along the principal direction of migration (elongation axis) of the primordium we aligned the samples along this direction. It was convenient to first obtain a 2d mask[1] of the primordium in the xy plane. We did this by filling holes in the image using the `fillHull` function from the package `EBImage`, followed by a maximal intensity projection of the z-stack. The segmentation was improved upon by using additional opening and closing operations. All objects except the largest were subsequently removed as the primordium mask was the object with the largest area in the field of view. Finally, `fillHull` was used again to fill any remaining holes.

```
> organ.mask <- apply(fillHull(mask), c(1, 2), max)
> organ.mask <- opening(closing(organ.mask, mk(Leff/2)), mk(Leff/2))
> organ.labels <- bwlabel(organ.mask)
> I <- which(table(organ.labels)[-1] < max(table(organ.labels)[-1]))
> organ.mask <- rmObjects(organ.labels, I)
> organ.mask[organ.mask > 0] <- 1
> organ.mask <- fillHull(organ.mask)
> writeImage(organ.mask, "PLLPanalysis-organMask.jpeg")
```

Figure 5: An xy-segmentation of the whole organ is obtained to determine the axis of migration and to ascertain the location of the leading edge.

[1]For condition mem-tFT a 3d mask was required due to the persistence of signal in the muscle layer underneath the primordium, which is to be expected from the slow degradation rate of the membrane-tethered protein.

An example of the resulting segmentation can be seen in Fig. 5. From the xy mask of the primordium, we computed the major axis of inertia using principal component analysis. The angle of this principal axis with respect to horizontal was stored in degrees.

```
> organ.coord <- as.data.frame(which(organ.mask == 1, arr.ind=TRUE),
+                              stringsAsFactors=FALSE)
> colnames(organ.coord) <- c("x", "y")
> organ.coord$x <- (organ.coord$x-1)*Lpx["x"]
> organ.coord$y <- -(organ.coord$y-1)*Lpx["y"]
> organ.pca <- prcomp(organ.coord)
> pc1 <- organ.pca$rotation[, 1]
> pc1.theta <- atan(pc1["y"]/pc1["x"])*(180/pi)
> pc1.theta

       y
-4.01044
```

All images were rotated so that they lay along the axis of migration. Extra parameters were passed to the `rotate` function to ensure that the rotated images fit fully into the new field of view.

```
> shiftOrigin <- c(100, 100)
> expandDim <- 1.2
> xGFP <- rotate(xGFP, angle=pc1.theta, output.dim=expandDim*dim(xGFP)[1:2],
+                output.origin=shiftOrigin)
> xRFP <- rotate(xRFP, angle=pc1.theta, output.dim=expandDim*dim(xRFP)[1:2],
+                output.origin=shiftOrigin)
> x2GFP <- rotate(x2GFP, angle=pc1.theta, output.dim=expandDim*dim(x2GFP)[1:2],
+                 output.origin=shiftOrigin)
> x2RFP <- rotate(x2RFP, angle=pc1.theta, output.dim=expandDim*dim(x2RFP)[1:2],
+                 output.origin=shiftOrigin)
> mask <- rotate(mask, angle=pc1.theta, output.dim=expandDim*dim(mask)[1:2],
+                output.origin=shiftOrigin)
> organ.mask <- rotate(organ.mask, angle=pc1.theta, output.dim=expandDim*dim(organ.mask),
+                      output.origin=shiftOrigin)
```

9 Excluding signal outside of primordium

By inspection of the membrane mask, we observed foreground pixels that lay outside of the primordium. As we were rather interested in cxcr4b turnover within the primordium, we excluded pixels that lay outside of the xy tissue mask of Fig. 5.

```
> organ.mask.rep <- replicate(dim(xGFP)[3], organ.mask)
> I1 <- which(organ.mask.rep == 1)
> I2 <- which(mask == 1)
> I <- intersect(I1, I2)
> mask <- array(0, dim=dim(organ.mask.rep))
> mask[I] <- 1
```

10 Cropping images for computational efficiency

For the sake of computational efficiency in the subsequent running median calculation, we cropped the images so that the field of view only contained the segmented primordium.

```
> I <- which(organ.mask.rep == 1, arr.ind=TRUE)
> intRange <- function(x) {rg=range(x); seq(rg[1], rg[2], by=1)}
> x.range <- intRange(I[, 1])
> y.range <- intRange(I[, 2])
```

```
> xGFP <- xGFP[x.range, y.range, ]
> xRFP <- xRFP[x.range, y.range, ]
> x2GFP <- x2GFP[x.range, y.range, ]
> x2RFP <- x2RFP[x.range, y.range, ]
> mask <- mask[x.range, y.range, ]
> organ.mask <- organ.mask[x.range, y.range]
> writeImage(organ.mask, "PLLPanalysis-cropImages.jpeg")
```

The coordinates of the field of view were also recalculated to account for the cropping.

```
> Lbox <- Lpx*dim(xGFP)
> Lbox

       x        y        z
226.0370  39.2764  29.0000
```

Figure 6: The organ mask was rotated so that it is aligned with the axis of migration, and cropped so that only the primordium was in the field of view. Compare with the pre-rotated and uncropped Fig. 5.

11 Running median

Before measuring the RFP/GFP fluorescence intensity ratio across the principal axis of our specimen, a running median on the raw data within the mask was computed to reduce the effects of pixel noise and chromatic aberration. Taking the median also reduced the contribution of RFP-rich vesicles to the RFP signal, which were sometimes observed close to the membrane, and therefore could fall within the membrane mask. This approach had merit provided that the volume of the box we were taking the median in was sufficiently larger than the volume of the vesicle that was lying on the membrane. Here we defined the size length of the cube for the running median to be 10 microns and the isotropic grid points on which to center this cube were spaced by 5 microns.

The following additional functions were required. `getCoordinates` receives a continuous range in microns and returns the corresponding discrete pixel coordinates. `getCubeIntensity` calculates the running median at specified locations. `runningMedian` is a wrapper function that calls `getCoordinates` and `getCubeIntensity` at each point on a spatial grid.

As the running median algorithm was straightforward to parallelize, `runningMedian` made use of the `mclapply` function from the `parallel` package to distribute the computation across available cores.

```
> getCoordinates <-
+ function(s, xrange, yrange, zrange) {
+     Ix=which(s$x >= min(xrange) & s$x <= max(xrange))
+     Iy=which(s$y >= min(yrange) & s$y <= max(yrange))
+     Iz=which(s$z >= min(zrange) & s$z <= max(zrange))
+     return(list(x=Ix, y=Iy, z=Iz))
+ }
> getCubeIntensity <-
+ function(x0, x, y, z, spatial, Lcube) {
+     crd <- getCoordinates(spatial,
+                           xrange=c(x-Lcube/2, x+Lcube/2),
+                           yrange=c(y-Lcube/2, y+Lcube/2),
+                           zrange=c(z-Lcube/2, z+Lcube/2))
+     cube.median <- median(imageData(x0)[crd$x, crd$y, crd$z], na.rm=TRUE)
```

```
+     return(list(median=cube.median))
+ }
> runningMedian <-
+ function(x, grid.x, grid.y, grid.z, Lx, Ly, Lz, nCores=2, Lcube) {
+     grid <- as.list(data.frame(t(expand.grid(grid.x, grid.y, grid.z))))
+     spatial <- list(x=(1:dim(x)[1]-1)*Lx,
+                     y=(1:dim(x)[2]-1)*Ly,
+                     z=(1:dim(x)[3]-1)*Lz)
+     chooseCores <- function(numCoresWanted) {
+         if(.Platform$OS.type == "windows") return(1)
+         return(numCoresWanted)
+     }
+     dataIntensities <-
+         mclapply(grid, function(s) getCubeIntensity(x0=x, x=s[1], y=s[2], z=s[3],
+                                                     spatial=spatial, Lcube=Lcube),
+                  mc.cores=chooseCores(nCores),
+                  mc.preschedule=FALSE)
+     dataIntensities=unlist(dataIntensities)
+     result.median=array(dataIntensities,
+                         dim=c(length(grid.x), length(grid.y), length(grid.z)))
+     return(result.median)
+ }
```

The spatial grid was defined in x, y and z coordinates of the specimen, the dimensions of the cube in which the running median was calculated, and the spacing between points on our grid.

```
> Ljump <- 5
> Lcube <- 10
> grid.x <- seq(from=0, to=Lbox["x"], by=Ljump)
> grid.y <- seq(from=0, to=Lbox["y"], by=Ljump)
> grid.z <- seq(from=0, to=Lbox["z"], by=Ljump)
```

Any pixel values in the GFP and RFP images that did not lie on the membrane mask were set to NA so that they did not contribute to the calculation.

```
> I <- which(mask == 0)
> xGFP.maskOnly <- xGFP
> xRFP.maskOnly <- xRFP
> xGFP.maskOnly[I] <- NA
> xRFP.maskOnly[I] <- NA
> resultGFP <- runningMedian(x=xGFP.maskOnly,
+                            grid.x=grid.x, grid.y=grid.y, grid.z=grid.z,
+                            Lx=Lpx["x"], Ly=Lpx["y"], Lz=Lpx["z"],
+                            nCores=4, Lcube=Lcube)
> resultRFP <- runningMedian(x=xRFP.maskOnly,
+                            grid.x=grid.x, grid.y=grid.y, grid.z=grid.z,
+                            Lx=Lpx["x"], Ly=Lpx["y"], Lz=Lpx["z"],
+                            nCores=4, Lcube=Lcube)

> GFP.profile=apply(resultGFP, 1, median, na.rm=TRUE)
> RFP.profile=apply(resultRFP, 1, median, na.rm=TRUE)
> plot(-grid.x, rev(GFP.profile),
+      xlab="Distance from leading edge (microns)",
+      ylab="Fluorescence intensity (a.u.)",
+      type="b",
+      axes="F",
+      pch=1,
+      ylim=c(0, range(GFP.profile, RFP.profile, na.rm=TRUE)[2]),
```

```
+      col="darkgreen")
> points(-grid.x, rev(RFP.profile), type="b", pch=2, col="red")
> axis.at.x <- seq(-200 ,0 , by=25)
> axis(1, at=axis.at.x, labels=-axis.at.x)
> axis(2)
> legend("topright", legend=c("GFP", "RFP"), pch=1:2, col=c("darkgreen", "red"))
```

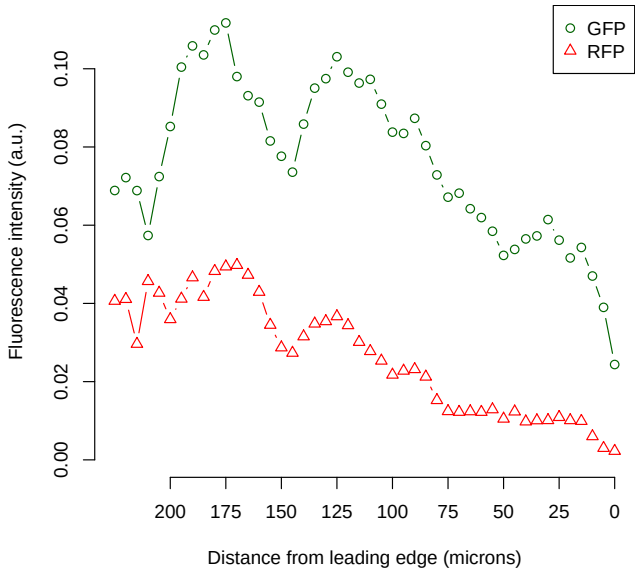

Figure 7: Median RFP and GFP fluorescence intensities across the primordium.

We visualised signal in the direction of migration by taking an additional running median along this axis (Fig. 7).

```
> resultRatio <- resultRFP/resultGFP
> ratio.profile <- apply(resultRatio, 1, median, na.rm=TRUE)
> plot(-grid.x, rev(ratio.profile),
+      xlab="Distance from leading edge (microns)",
+      ylab="RFP/GFP fluorescence intensity ratio (-)",
+      type="b",
+      axes="F")
> axis.at.x <- seq(-200, 0, by=25)
> axis(1, at=axis.at.x, labels=-axis.at.x)
> axis(2)
```

To obtain a localised readout for cxcr4b stability, we took the ratio of the RFP to GFP signal; see Fig. 8. For graph interpretation note that the higher the RFP/GFP ratio, the higher the age of the protein.

12 Correcting for day-to-day variability

From day-to-day, there can be variability in signal due to intentional changes in laser power by the experimenter, undesirable daily fluctuations in the chosen laser power, as well as other unknown factors. Here we used a solution of purified sfGFP-mCherry fusion protein as a daily control. In Section 14 we explain why it is appropriate to normalise the TagRFP/sfGFP ratio of each imaged primordium by the mCherry/sfGFP fluorescence intensity of the control.

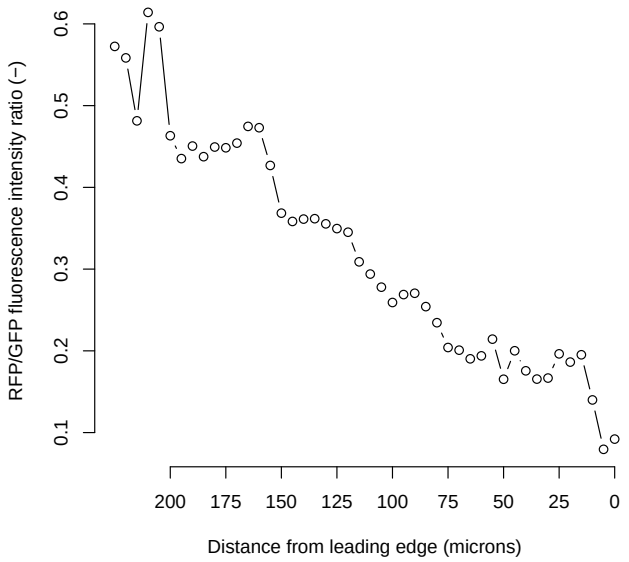

Figure 8: The RFP/GFP intensity ratio increased towards the back of the primordium, indicating an increased stability, or longer lifetime, of cxcr4b receptors.

Before calculating the mCherry/sfGFP ratio from the ratio-normalisation image, we performed background subtraction.

```
> solGFP <- readImage(system.file("extdata/1_100_mCherryC1.tif",
+                     package="DonaPLLP2013"))
> solGFPbck <- readImage(system.file("extdata/PBSC1.tif", package="DonaPLLP2013"))
> solRFP <- readImage(system.file("extdata/1_100_mCherryC2.tif",
+                     package="DonaPLLP2013"))
> solRFPbck <- readImage(system.file("extdata/PBSC2.tif", package="DonaPLLP2013"))
> solRFP <- solRFP-mean(solRFPbck)
> solGFP <- solGFP-mean(solGFPbck)
```

We then formed the average mCherry/sfGFP fluorescence intensity ratio of the solution and divided the sample ratio by this number.

```
> solRatio <- mean(solRFP)/mean(solGFP)
> resultRatioCorrected <- resultRatio/solRatio
```

13 Tissue-scale ratio measurement

To obtain an estimate of whole-primordium receptor stability, the median ratio was computed across the sample. In each case this was done on the solution-corrected ratios.

```
> print(median(resultRatioCorrected, na.rm=TRUE))
```

```
[1] 0.1902899
```

14 Theoretical explanation for ratio correction

For a given fluorophore, fluorescence intensity is proportional to the number of mature fluorescent proteins present. The proportionality factor f is constant for a particular fluorophore and is related to its

brightness. h is the laser power used on a given day, in suitable units. We also consider additive noise ξ (normally referred to as background), and multiplicative noise $e^{\epsilon} \approx 1$ with $\epsilon \sim N(0, \sigma^2)$, which encompasses daily laser power fluctuations or other unknown effects. We denote the fluorescence intensity measurement I per volume per unit time for a given fluorescence channel as

$$I = e^{\epsilon} f h X + \xi \ . \tag{1}$$

In Section 5 background subtraction was performed on each channel to reduce the additive noise. We assume that the resulting corrected image $\hat{I}$ had negligible additive noise, and therefore ignore the ξ term. Denoting the TagRFP/sfGFP ratio calculated in Section 11 as R, we have

$$R = \frac{\hat{I}_r}{\hat{I}_g} = \frac{e^{\epsilon_{561}} f_{\mathrm{TagRFP}} h_{561} X_r}{e^{\epsilon_{488}} f_{\mathrm{sfGFP}} h_{488} X_g} \tag{2}$$

The subscripts r and g refer to the RFP and GFP channels, respectively. 561 and 488 denote the excitation wavelengths in nanometers of the RFP and GFP channels, respectively.

Background subtraction was also performed for the solution of mCherry-sfGFP tandem fluorescent timer, resulting in the following expression for the ratio:

$$R^s = \frac{\hat{I}^s_r}{\hat{I}^s_g} = \frac{e^{\epsilon_{561}} f_{\mathrm{mCherry}} h_{561} X^s_r}{e^{\epsilon_{488}} f_{\mathrm{sfGFP}} h_{488} X^s_g} \tag{3}$$

The superscript s here indicates terms that are representative of the mCherry-sfGFP solution. f_{mCherry} is the proportionality factor of mCherry. For an ideal solution of fully mature fluorophores, where no misfolding has occurred, $X^s_r/X^s_g = 1$ but in reality, this number will not be exactly equal to 1. Instead we assume it has a constant purity, $P = X^s_r/X^s_g$, as the solution was always prepared from the same batch, and had a large number of molecules. Therefore, by dividing eq. 2 by eq. 3, the multiplicative noise and other linear terms cancel out, resulting in

$$\hat{R} = \frac{R}{R^s} = c\frac{X_r}{X_g} \tag{4}$$

, where we have defined a new constant $c = f_{\mathrm{TagRFP}}/(P f_{\mathrm{mCherry}})$. $\hat{R}$ denotes the control-normalised ratio.

All image data presented in the main text was corrected for additive and multiplicative noise, as described in this section. The normalisation by the control solution of mCherry-sfGFP eliminated any time-dependent noise factors (on the scale of days) and thus allowed us to group measurements of the same condition taken on different days, and to directly compare ratio profiles across different genetic conditions.

We note that as long as the laser powers remain constant during a given microscope session for the imaging of both sample and control, in principle one does not need to explicitly know their values, as they cancel out during the control normalisation step. Although not incorporated into the above equations, the same is true for other linear terms such as exposure time. Nevertheless, where possible, it is good practice to record these values for future reference.

15 The effect of mask thickness on the ratio

Earlier we remarked that changes in the membrane mask thickness due to opening and closing morphological operations, did not substantively change the RFP/GFP fluorescence intensity ratio. We demonstrated this on tissue-scale measurements by progressively thickening, and then progressively thinning the membrane mask. After each change in thickness, the tissue-scale RFP/GFP ratio was measured for pixels lying on the mask.

```
> membraneRatio <- array(dim=5)
> I <- which(mask == 1)
> membraneRatio[3] <- median(xRFP[I])/median(xGFP[I])
> calcRatioOnModifiedMembrane <- function(xGFP, xRFP, mask, morph=erode, steps=1) {
+  for (i in seq_len(steps)) mask <- morph(mask, mk(2))
+  I <- which(mask == 1)
+  return(median(xRFP[I])/median(xGFP[I]))
```

```
+ }
> membraneRatio[1] <- calcRatioOnModifiedMembrane(xGFP, xRFP, mask, erode, 2)
> membraneRatio[2] <- calcRatioOnModifiedMembrane(xGFP, xRFP, mask, erode, 1)
> membraneRatio[4] <- calcRatioOnModifiedMembrane(xGFP, xRFP, mask, dilate, 1)
> membraneRatio[5] <- calcRatioOnModifiedMembrane(xGFP, xRFP, mask, dilate, 2)
> plot(1:5, membraneRatio, xlab="Mask ID",
+      ylab="RFP/GFP fluorescence intensity ratio (-)",
+      ylim=c(0, max(membraneRatio)))
```

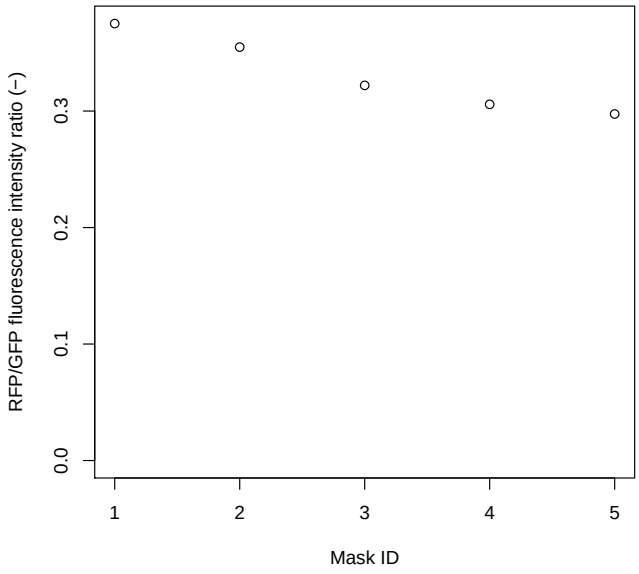

Figure 9: Whole-primordium median RFP/GFP fluorescence intensity ratios for membrane masks of varying thickness. The mask thickness increases from left to right.

Fig. 9 showed that as the thickness of the membrane is varied over a range of the order of a micron, the RFP/GFP ratio changed by only approximately 10%. Any additional thickening or thinning of the mask outside of this range resulted in an inaccurate segmentation, and so these cases were not considered.