ImageData_Analysis

Vlatka Antolovic

Dependencies

```
library(ggplot2)
library(viridis)
library(plotfunctions)
library(dplyr)
library(paletteer)
library(shades)
library(zoo)
```

Set working directory (full path of the 'Image DataAnalysis' folder)

```
folder <- 'C:/Users/Vlatka/Documents/Image_DataAnalysis/'
setwd(folder)
```

Import functions

```
source('./Functions/Fun_imagePlot.R')
source('./Functions/Fun_fitPlot.R')
source('./Functions/Fun_min_infl.R')
source('./Functions/Fun_combinedPlots.R')
source('./Functions/Fun_flamindo.R')
```

1 Data import

Choose experiment

```
gene1 = 'carA'
gene2 = 'cafA'

#---------carA dates-----------
g1_datum1 = '20211217' # paper: Figure 2
g1_datum2   = '20220722' # paper: Figure S2

#---------cafA dates-----------
g2_datum1 = '20211027' # paper: Figure 3
g2_datum2 = '20211215' # paper: Figure S3
g2_datum3 = '20221107'

# Choose the experiment
gene = gene2
```

```
date = g2_datum2

file_spots = paste0(folder, 'DATA/', date, gene, '.csv')
file_nucl = paste0(folder, 'DATA/', date, gene, '_nucs.csv')
file_flam = paste0(folder, 'DATA/', date, gene, '_flamindo.csv')
```

Import data Import: spot intensity, nuclei position and Flamindo2 intensity.

SPOTS: time, x-coord. and intensity of each spot.

NUCL: time and x-coord. of each nuclei.

FLAMINDO: (time, x) matrix of each image's y-column mean intensity within intensity thresholds corresponding to the intensity of Flamindo2 expressing cells.

```
spots <- read.table(file_spots, sep = ",", header = F); spots = as.data.frame(t(spots))
names(spots) <- c('time', 'x', 'I')

nucl <- read.table(file_nucl, sep = ",", header = F); nucl = as.data.frame(t(nucl))
names(nucl) <- c('time', 'x')

flamindo <- as.matrix(read.table(file_flam, sep = ",", header = F))
```

In some experiments, to save on computation time, the last few timepoints where the population has clearly split, and the stream had peeled away, were not analysed for spots. We remove this extra timepoints from Flamindo2 matrix.

```
if (date == 20221107) {flamindo <- flamindo[-217:-241,]
} else if (date == 20211217) {flamindo <- flamindo[-241,]
}
```

Get 'times'

```
times <- unique(nucl$time)
```

2 Binning

To speed up code and plotting.

```
# Set bin width
binW <- 10 # i.e. 1/3 of the cell length (0.349 um/pixel)

# Bin function
bin <- function(x, binW){ceiling((x-1) / binW)}
```

2.1 Binning spots and nuclei

Assign spots and nuclei to bins.

```
if (binW==1){
  spots$xbin <- spots$x
  nucl$xbin <- nucl$x
} else{
```

```
  spots$xbin <- bin(spots$x, binW)
  nucl$xbin <- bin(nucl$x, binW)
}
```

Create (time, bin) matrices of spot intensity and nuclei counts.

```
pixelNum <- max(nucl$x)
binNum = max(nucl$xbin)

frameNum = length(times)

#--------Spots and Nuclei-----------
# Create (time x bin) matrices

nuclei <- maxI <- matrix(0, nrow = frameNum, ncol = binNum)
for (i in times){
  # nuclei
  tmp <- nucl[nucl$time == i, ] %>% group_by(xbin) %>% tally()
  nuclei[i, tmp$xbin] <- tmp$n
  # median & max intensity
  tmp <- spots[spots$time == i, ] %>% group_by(xbin) %>%
    dplyr::summarise(maxI = max(I))
  maxI[i, tmp$xbin] <- tmp$maxI
}
```

2.2 Binning Flamindo2 data

```
# Already a (time x pos) matrix
if (binW==1){
  flam = flamindo
} else{
  flam <- matrix(nrow = frameNum, ncol = binNum)
  bin = 0
  for (i in seq(1, pixelNum, binW)){
    if (i+binW-1 > dim(flamindo)[2]) {break}
    bin = bin + 1
    flam[, bin] = suppressWarnings( rowMeans(flamindo[, i:(i+binW-1)], na.rm = T) )
  }
}
```

2.3 This part is used only if the images need to be reversed ... to standardise the orientation of the developmental niche.

```
# --- Function 'reverse'
reverse <- function(m) {m <- m[, ncol(m):1]}

# ---Reverse matrices
# If the image needs to be reversed - undifferentiated cells on the left
if (date == 20211027) {nuclei = reverse(nuclei); maxI = reverse(maxI); flam = reverse(flam)
} else if (date == 20220722) {nuclei = reverse(nuclei); maxI = reverse(maxI); flam = reverse(flam)
}
```

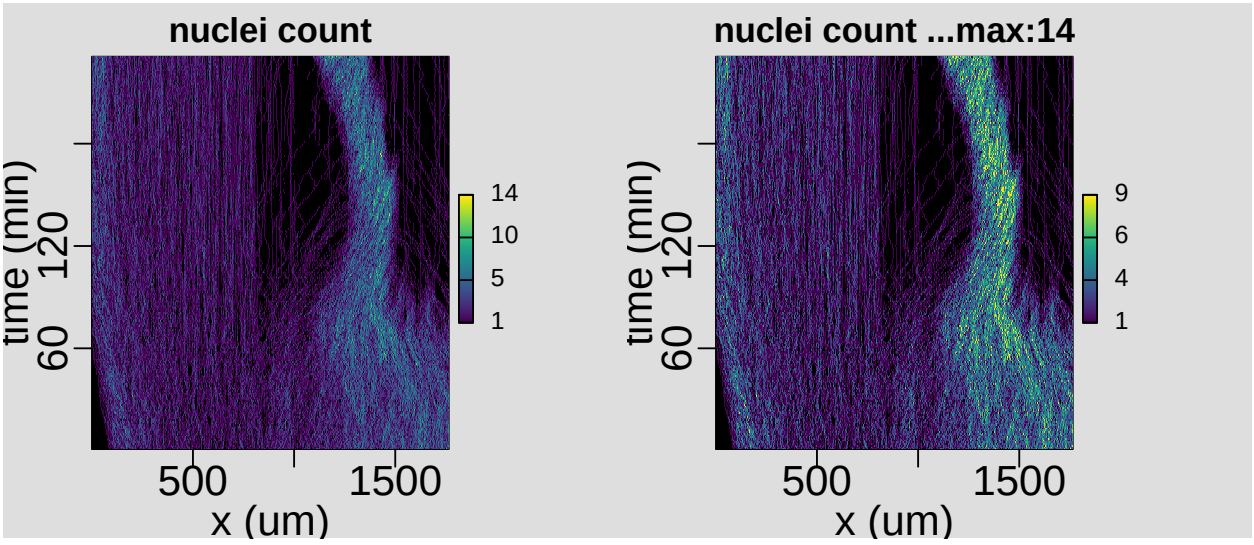
3 Plot values at x-coordinates in times

Imaging settings

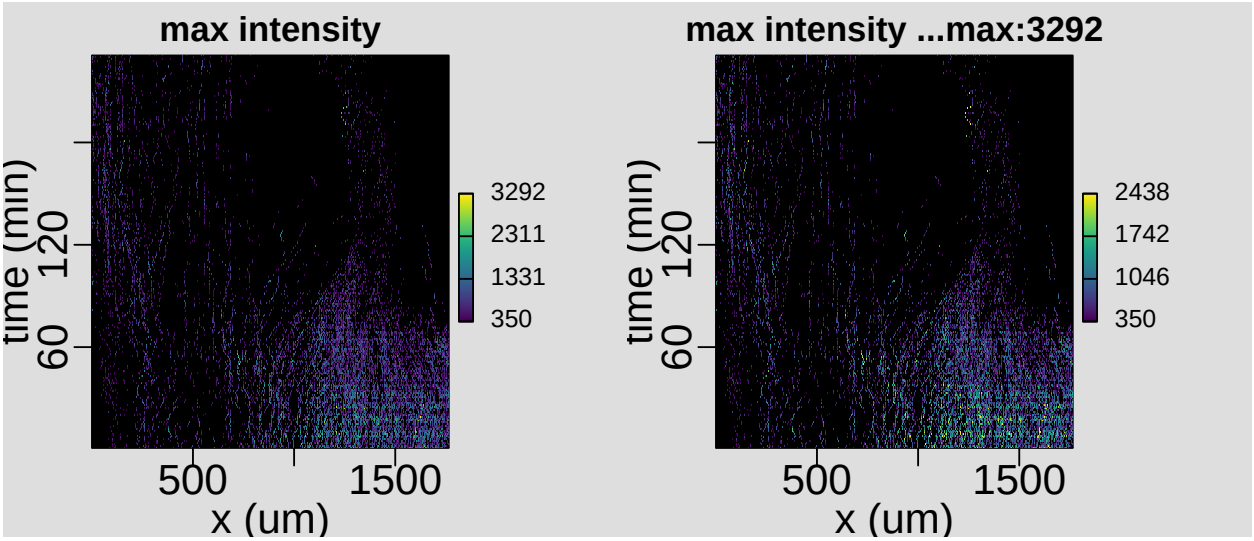
```
#--------um per pixel-----------
umPix = 0.349

#--------min per frame-----------
if (date == 20211027) {dt <- 45/60
} else {dt <- 1
}
```

Plot x-axis represents x-coordinate and y-axis represents time. Depending how the matrices (time, bin) are defined, there is an option to replace zeros with NAs or NAs with zeros, for control over the dynamic range of the data.

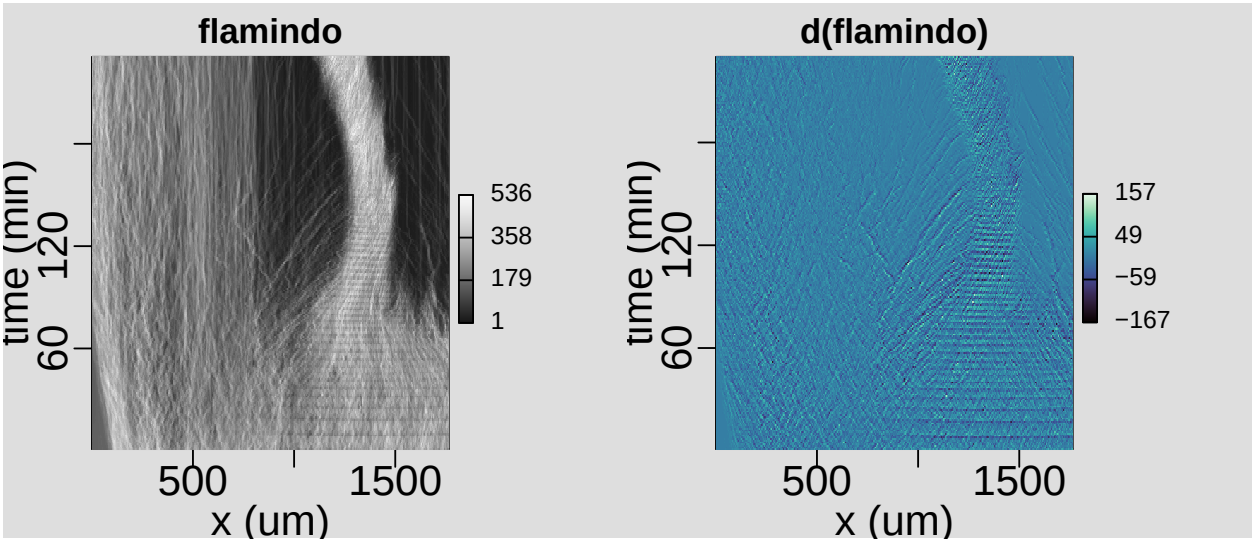
```
# --- Colour map
colMap = viridis(100)

# --- Legend Position
if (date == 20211027) {pos_l =  c(480, 75, 500, 170)
} else{pos_l =  c(520, 75, 540, 150)
}

# --- Plot nuclei
par(mfrow = c(1,2))
plotmat(nuclei, colMap, 'nuclei count', repl_0 = T)
# everything above outlier threshold (Q3+3IQR) represented with the highest colour value:
plotmat_outl(nuclei, colMap, 'nuclei count', repl_0 = T)
```




```
# --- Plot expression intensity
par(mfrow = c(1,2))
plotmat(maxI, colMap, 'max intensity', repl_0 = T)
plotmat_outl(maxI, colMap, 'max intensity', repl_0 = T)
```




```
# --- Plot flamindo signal
# Color palette
pal <- paletteer_c("grDevices::Grays", 10)
colPal = colorRampPalette( pal[1:10] )
#
par(mfrow = c(1,2))
plotmat(flam, colPal(100), 'flamindo')

# --- Flamindo differential
dflam = diff(flam)
#
plotmat(dflam, mako(100), 'd(flamindo)')
```

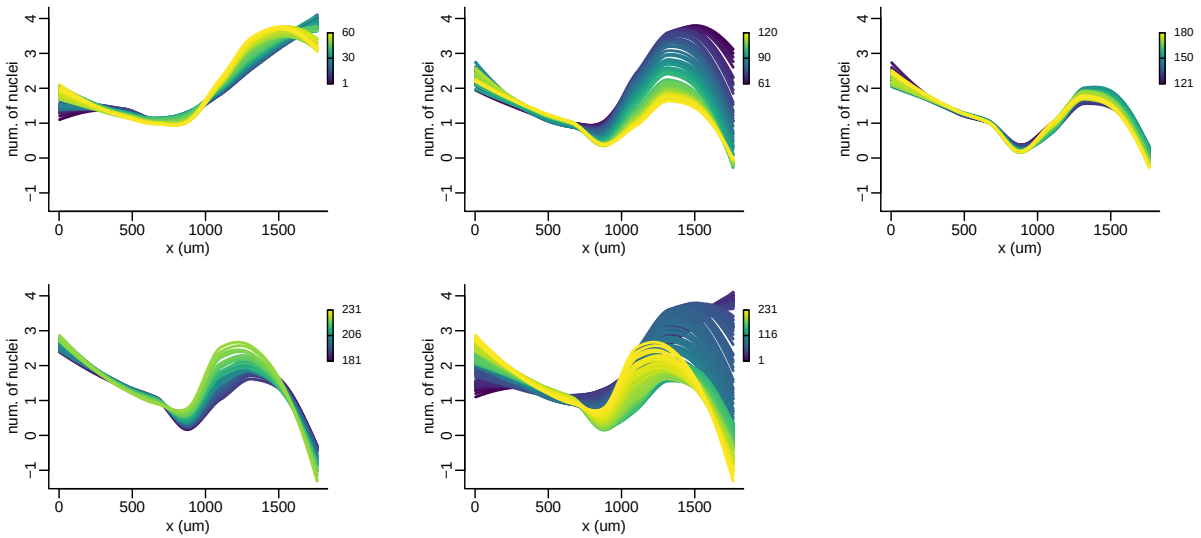


4 Cell distribution

Identify the area of cell population separation (population splits in the 'middle', while cells either go onto develop or are left behind)

4.1. Fit a curve of nuclei distribution across the fields' x-coords. Nuclei distribution in time.

```
#---------loess----------
# default fit (loess with span = .75)
fit_nucl <- lw(nuclei)

#---------Plot fitted curves in time-----------
par(mfrow = c(2,3))
# Plot every hour
for (i in round(seq(1, frameNum, 60) / dt)){
  if (i > frameNum) {break}
  plotfit_time(fit_nucl, i, i+59, 'num. of nuclei')
}

# All
plotfit_time(fit_nucl, 1, frameNum, 'num. of nuclei')
```



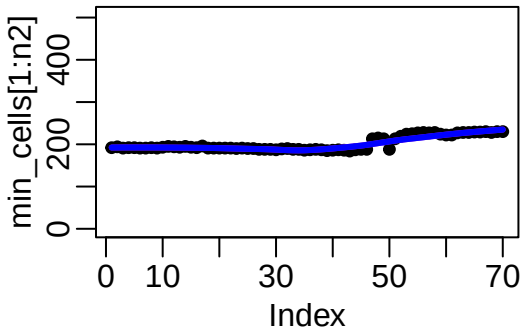
4.2 Population separation Timerange of interest

```
# defining analysis boundaries, where n2 - upper time limit of inflection analysis
# n3 - upper time limit of time series plots (so oscillations can be visualised)
if (date == 20211217) {n2 <-  210; n3 <- 120
} else if (date == 20211027) {n2 <- 100; n3 <- 100
} else if (date == 20220722) {n2 <- 120; n3 <- 120
} else if (date == 20221107 | date == 20211215) {n2 <- 70; n3 <- 70
}
```

Identify the minimum of nuclei counts at each timepoint. This is an approximation for the location of cell population separation (i.e. either join a stream or be left behind).

```
# --- Find minima
min_cells <- vector()
for (i in 1:frameNum){
  min_cells[i] <- find_min(fit_nucl, i)
```

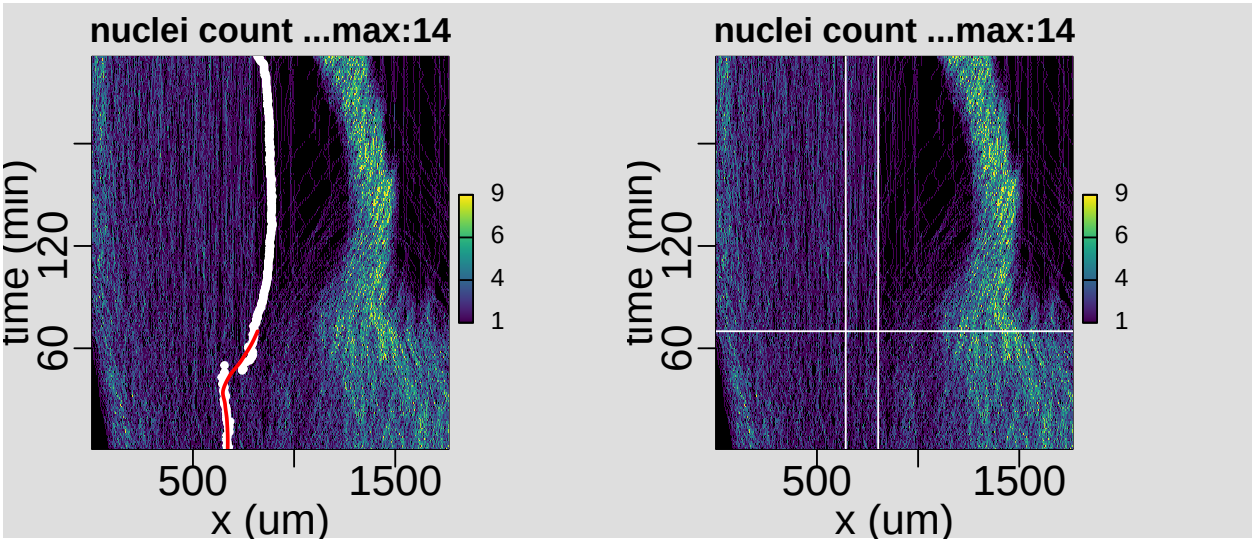
```
}

# --- Check outliers
hout <- function(x){quantile(x, .75, na.rm = T) + 1.5 * IQR(x, na.rm = T)}
lout <- function(x){quantile(x, .25, na.rm = T) - 1.5 * IQR(x, na.rm = T)}

# Range of detected minima in a timerange of interest
# m_range: without outliers (range of min_cells[1:n2] values with outliers removed)
m_range <- min_cells[1:n2] < hout(min_cells[1:n2]) & min_cells[1:n2] > lout(min_cells[1:n2])

# Range of detected minima (cell depleted area)
min_area <- range(min_cells[1:n2][m_range], na.rm = T)

# --- Curve through minimums in analysis range
# all
x = 1:n2; y = min_cells[1:n2]; min_fit <- loess(y ~ x, span = .75)
# outliers removed
x = x[m_range]; y = y[m_range]; min_fit_outl <- loess(y ~ x, span = .75)

# --- Plot cell minima curve
par(mgp = c(1.5, .5, 0), mar = c(2.5, 2.5, 1.5, 1.5))
plot(min_cells[1:n2], pch = 19, cex = .75, ylim = c(0, binNum))
# all
lines(min_fit$x, min_fit$fitted, lwd = 3, col = 'purple')
# outliers removed
lines(min_fit_outl$x, min_fit_outl$fitted, lwd = 3, col = 'blue')
```



Plot over image Plot the identified nuclei count minima over the x-t plot.

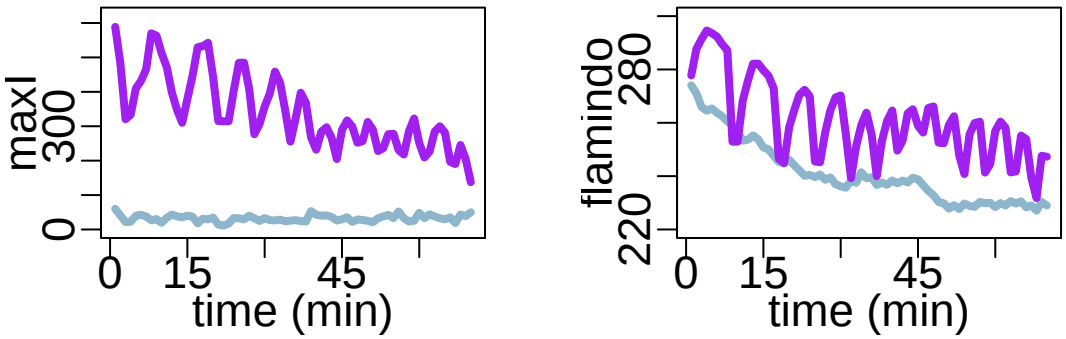
```
par(mfrow = c(1,2))
# Plot detected minima and fit
plotmat_outl(nuclei, colMap, 'nuclei count', repl_0 = T)
points(min_cells, 1:frameNum, pch = 19, cex = .5, col = 'white')
lines(min_fit_outl$fitted, min_fit_outl$x, lwd = 2, col = 'red')

# Plot the positions of area divide (vertical lines) and time point of analysis
# boundary (horizontal line)
plotmat_outl(nuclei, colMap, 'nuclei count', repl_0 = T)
abline(h = n2, col = 'white', lw = 1)
```

```
abline(v=min_area[1], col = 'white', lw = 1)
abline(v=min_area[2], col = 'white', lw = 1)
```



4.3 Two zones of behaviour Compare the average spot intensity and Flamindo2 intensity between undifferentiated (blue) and differentiating (purple) area, at each timepoint.

```
# area divide:
range1 = 1:min_area[1];  range2 = min_area[2]:binNum
# (recalculated within the splitarea function; used later with flamindo?)

# ---  Plot spot intensity and Flamindo signal on either side of
# population division
# y-range for maxI and flam
if (date == 20211217) {y_int <- c(0, 350); y_flam <- c(100,300)
} else if (date == 20211027) {y_int <- c(0, 130); y_flam <- c(150,250)
} else if (date == 20220722) {y_int <- c(0, 150); y_flam <- c(100,400)
} else if (date == 20221107) {y_int <- c(0, 620); y_flam <- c(230,335)
} else if (date == 20211215) {y_int <- c(0, 620); y_flam <- c(220,300)
}

# blue: close to feeding front; purple: after the separation
par(mfrow = c(1,2))
splitarea(maxI, 1, n2, 'maxI', y_int, tseq = 15)
splitarea(flam, 1, n2, 'flamindo', y_flam, tseq = 15)
```

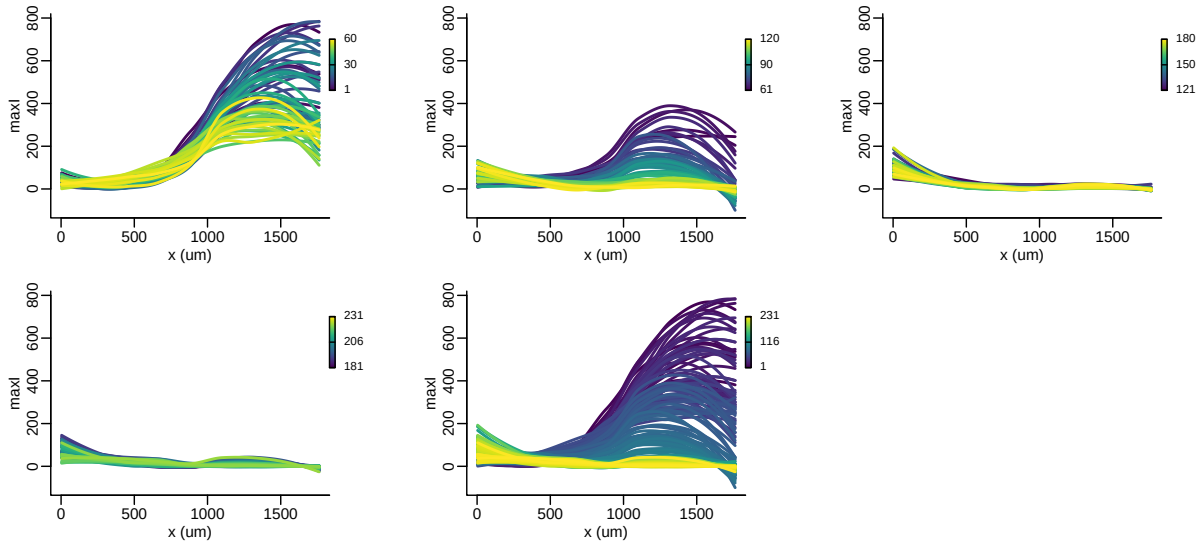


5 Gene induction (cafA or carA)

Identify the location of cells inducing stong gene expression.

4.1. Fit a curve of spot intensity accross the fields' x-coords. Gene expression across the field' x-coords in time.

```
#---------loess----------
# default fit (loess with span = .75)}
fit_maxI <- lw(maxI)

#---------Plot fitted curves in time-----------
par(mfrow = c(2,3))
# Plot every hour
for (i in seq(1, frameNum, 60) / dt){
  if (i > frameNum) {break}
  plotfit_time(fit_maxI, i, i+59, 'maxI')
}

# All
plotfit_time(fit_maxI, 1, frameNum, 'maxI')
```

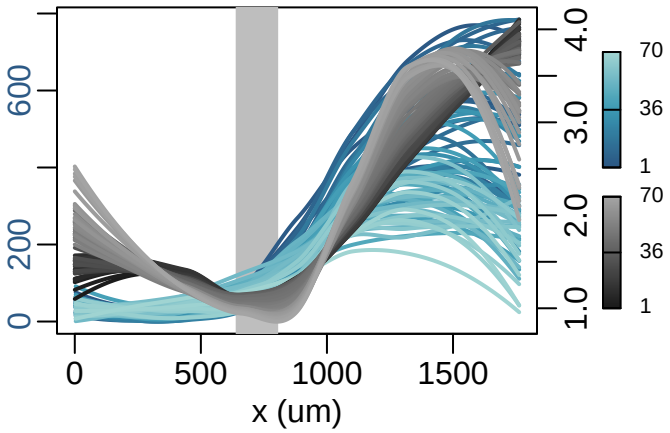


Overlay with cell distribution Gray vertical box: range of nuclei counts minima.

Gray curves (dark to light - through time): nuclei counts distribution over x-axis.

Blue curves (dark to light): gene expression distribution over x-axis.

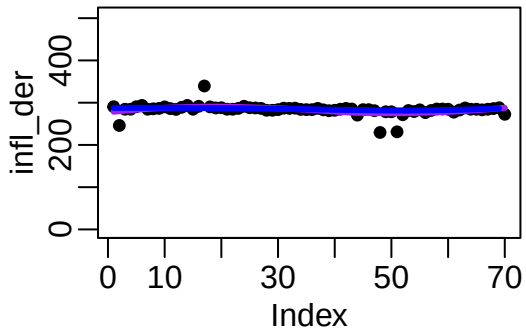
```
#---------Colour palettes-----------
pal <- rev(paletteer_c("ggthemes::Blue-Teal", 10)); pal <- pal[1:9]
colPal1 <- colorRampPalette(pal);
pal <- paletteer_c("grDevices::Grays", 10); pal <- pal[1:6]
colPal2 = colorRampPalette(pal);

# --- Plot settings (cafA = datum1)
# Legend
if (date == 20211217) {x = c(610, 630); yg = c(210, 500); yn = c(0, .9)
} else if (date == 20221107) {x = c(610, 630); yg = c(350, 650); yn = c(2, 4.5)
} else if (date == 20211215) {x = c(600, 620); yg = c(400, 700); yn = c(1, 2.2)
} else if (date == 20211027) {x = c(520, 540); yg = c(70, 150); yn = c(-.5, .9)
} else if (date == 20220722) {x = c(600, 620); yg = c(75, 150); yn = c(0, .8)
}
#
pos_l_gene = c(x[1], yg[1], x[2], yg[2])
pos_l_nuc = c(x[1], yn[1], x[2], yn[2])

# Plot (timepoints of interest)
plot_mix(fit_nucl, 1, n2, if_legend1 = T)
```



4.2 Gene induction Find inflection point. Approximation for the location of cells starting to strongly induce the observed gene's expression.

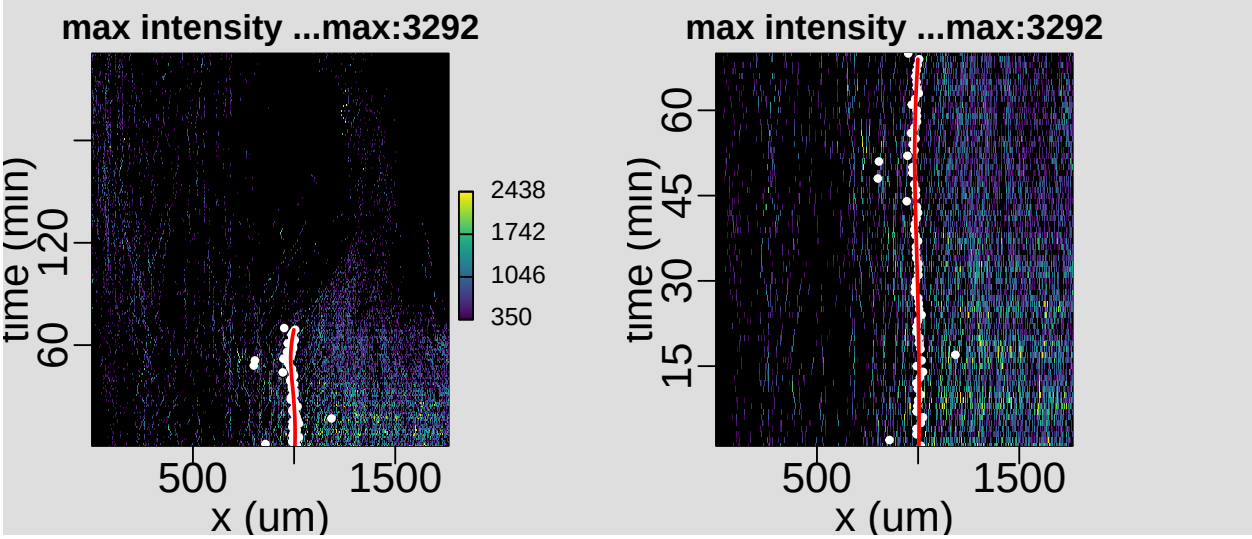
```
#---------Find inflection-----------
win = 30 # (default)

# Inflection points
infl_der = vector()
for (i in 1:n2){
  infl_der[i] = find_infl(fit_maxI, i, win = win, if_plot = F, if_result = T)
}
```

```r
# i_range: without outliers (range of infl_der values with outliers removed)
i_range <- infl_der < hout(infl_der) & infl_der > lout(infl_der)

# --- Curve through infl. coord.
# all
x = 1:n2; y = infl_der; infl_fit <- loess(y ~ x, span = .75)
# outliers removed
x = x[i_range]; y = y[i_range]; infl_fit_outl <- loess(y ~ x, span = .75)

# --- Plot infl. coor. curve
par(mgp = c(1.5, .5, 0), mar = c(2.5, 2.5, 1.5, 1.5))
plot(infl_der, pch = 19, cex = .75, ylim = c(0, binNum))
# all
lines(infl_fit$x, infl_fit$fitted, lwd = 3, col = 'purple')
# outliers removed
lines(infl_fit_outl$x, infl_fit_outl$fitted, lwd = 3, col = 'blue')
```



Plot over image Plot the identified position of cells inducing gene expression over the x-t plot.

```r
# --- All times
par(mfrow = c(1,2))
plotmat_outl(maxI, colMap, 'max intensity', repl_0 = T)
# Identified points of infl.
points(infl_der, 1:n2, pch = 19, cex = .5, col = 'white')
lines(infl_fit_outl$fitted, infl_fit_outl$x, lwd = 2, col = 'red')

# --- Timepoints of interest (zoom-in)
plotmat_outl(maxI, colMap, 'max intensity', repl_0 = T, n1 = 1, n2 = n2,
             tseq = 15, if_legend = F)
# Identified points of infl.
points(infl_der, 1:n2, pch = 19, cex = .5, col = 'white')
lines(infl_fit_outl$fitted, infl_fit_outl$x, lwd = 2, col = 'red')
```



Overlay with cell distribution Gray vertical box: range of nuclei counts minima.

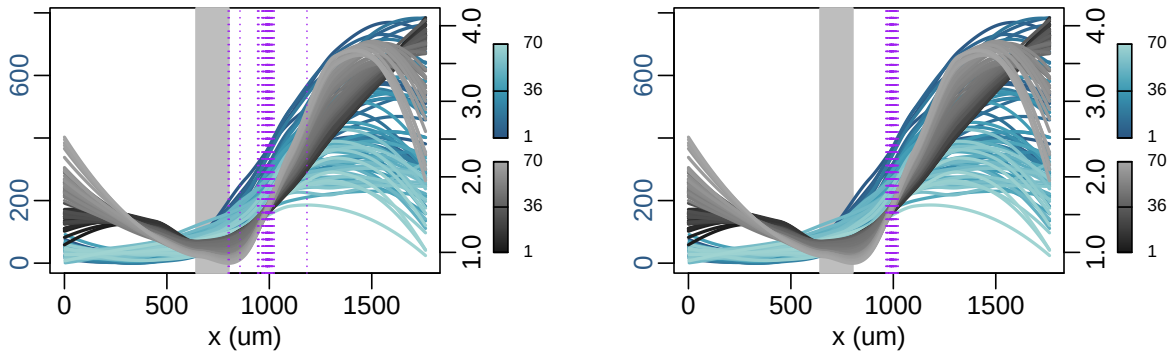
Purple vertical dotted lines: locations of gene induction.

Gray curves (dark to light): nuclei counts distribution over x-axis.

Blue curves (dark to light): gene expression distribution over x-axis.

Left panel: all inflection points. Right panel: outlier points removed.

```
par(mfrow = c(1,2))

# Plot
plot_mix(fit_nucl, 1, n2)
abline(v = infl_der, lty = 3, col = 'purple')

# --- w/o outliers
plot_mix(fit_nucl, 1, n2)
abline(v = infl_der[i_range], lty = 3, col = 'purple')
```



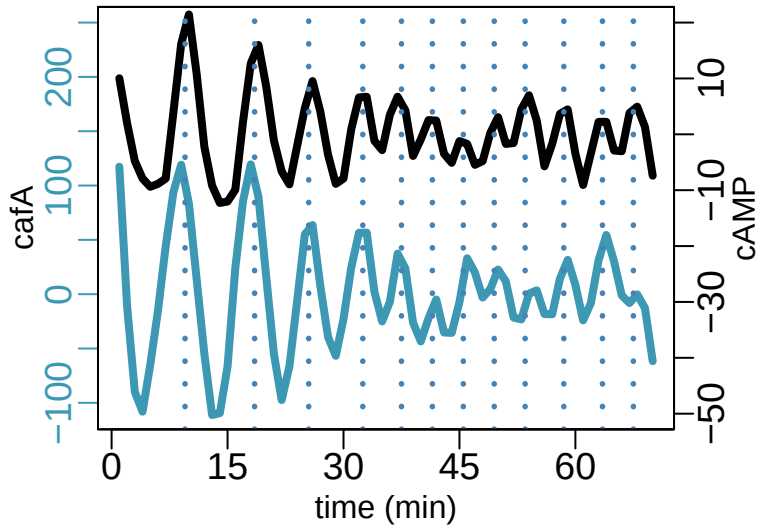
6 cAMP signaling (Flamindo2)

6.1 Overlay cAMP signal with gene transcription Use Savitzky-Golay filtering to smooth and background correct spot intensity and Flamindo2 signal. Additionally, transform Flamindo2 signal into cAMP

signal.

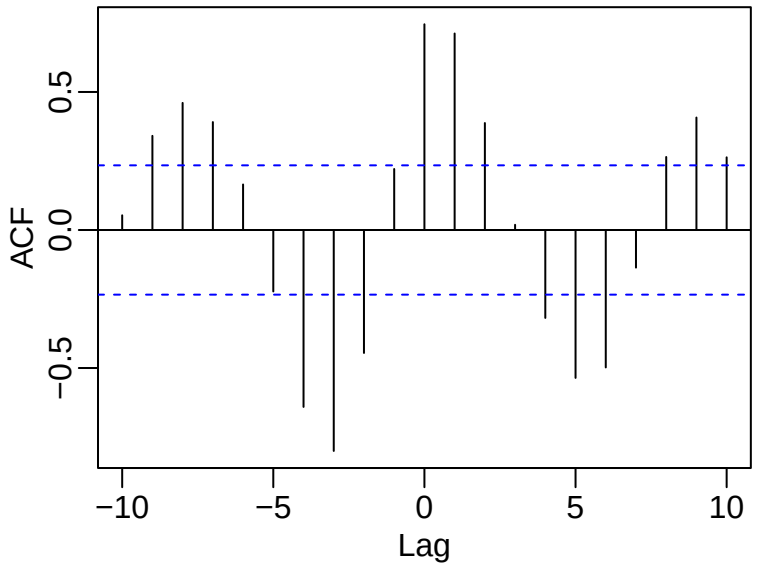
Plot average value of spot intensity and cAMP signaling in the area right of the cell population separation ('range2'), at each timepoint in time range of interest.

```
# ----- plotting cAMP vs spot intensity
# Settings
win = 1; dwin = 2; noise = -3
ylim = c(-5,5)

if (date == 20211217) {glim = c(-60,150); clim = c(-40, 12)
} else if (date == 20211215) {glim = c(-110,250); clim = c(-50, 20)
} else if (date == 20211027) {glim = c(-30,80); clim = c(-50, 20)
} else if (date == 20220722) {
  win <- 2; dwin <- 4; noise = -5
  glim = c(-22,70); clim = c(-50, 30)
} else if (date == 20221107) {glim = c(-80,230); clim = c(-30, 15)
}

# identify timepoints of flamindo minima (for timepoint identification)
min_flam <- minima(flam, 1, n3, win = dwin, if_plot = F)

# Plot
# black: Flamindo2 signal
# teal: gene transcription
# vertical lines: Flamindo2 local minima (cAMP maxima)
n = 1
sgolay_line(lw = 4, fsize = 1.2, tseq = 15, clim = clim, glim = glim,
            add_flam_line = T, n = 3, p = 1)
```



Perform cross correlation analysis on the Savitzky-Golay filtered signals

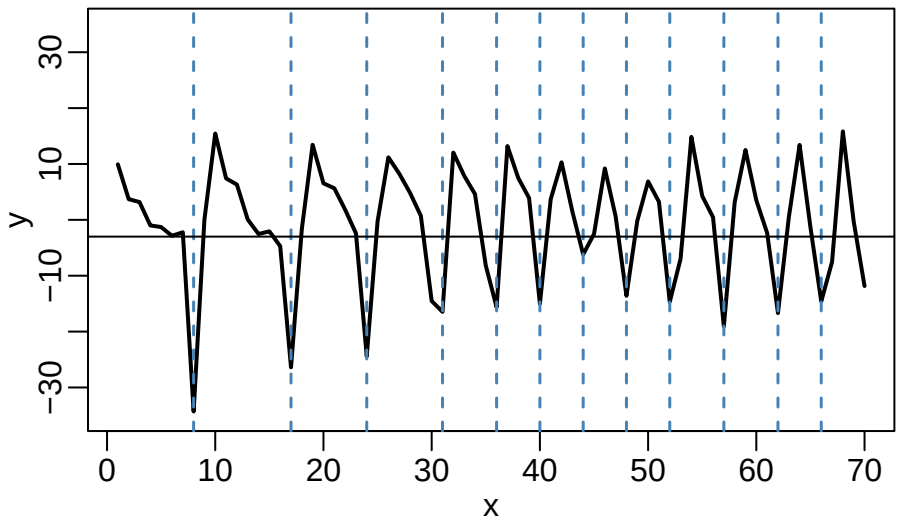
```
plot_ccf(n = 3, p = 1, bg = 15, lag = 10)
```



6.2 Find local minima of mean d(Flamindo2) Identify timepoints where cAMP waves pass through the population: The lowest intensity of Flamindo2 corresponds to the highest intracellular cAMP concentration.

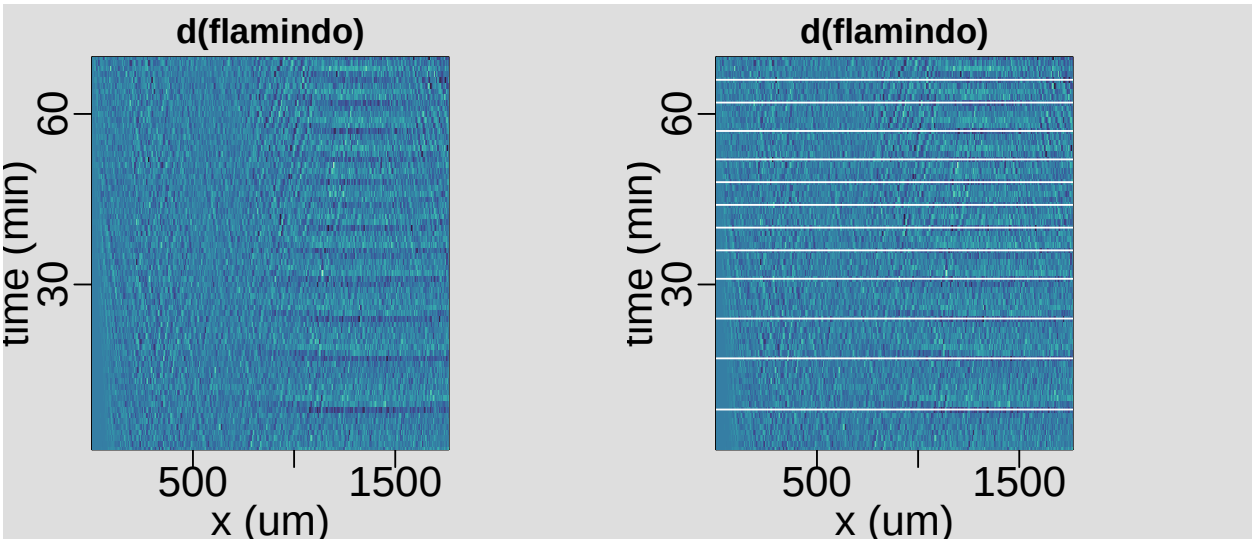
We use the differential of Flamindo2 intensities as a more reliable signal.

```
# Define dflam minima
# Plot check - blue vertical lines at the identified local minima
min_dflam <- minima(dflam, 1, n2, range2, win = win, c(-35, 35), if_diff = T,
                    if_plot = T, noise = noise)
abline(h = noise) # noise cut-off
```



Plot over image Plot the identified local d(Flamindo2) minima over the x-t plot.

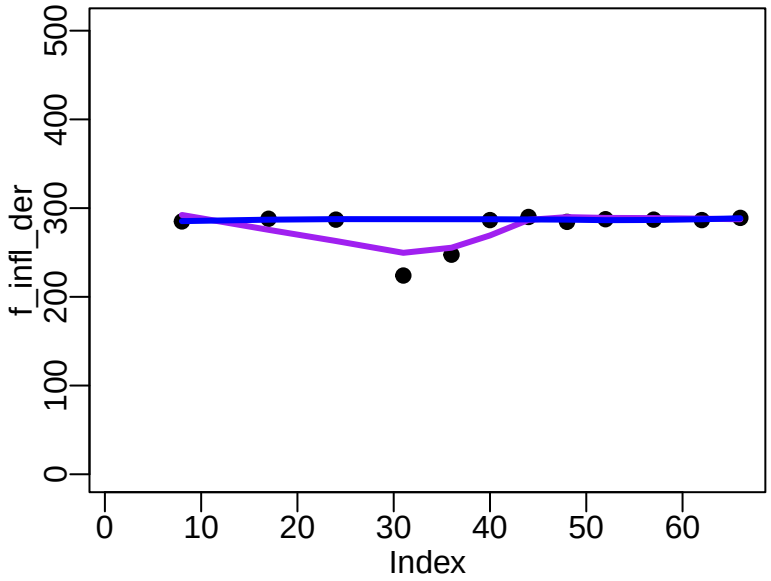
```
par(mfrow = c(1,2))
plotmat(dflam, mako(100), 'd(flamindo)', n2 = n2, tseq = 30,
            if_legend = F)
# --- Plot over image
plotmat(dflam, mako(100), 'd(flamindo)', n2 = n2, tseq = 30,
            if_legend = F)
abline(h = min_dflam, col = 'white', lty = 1, lwd = 1)
```



6.3 Find the location of cAMP relay border Find inflection points in d(flamindo2) fit across x-dimension, at timepoints of identified minima (max cAMP signalling).

```
# Define fit
fit_dflam <- lw(dflam)

#---------Results [only at points of minimum]-----------
f_infl_der = vector()
for (i in min_dflam){
  f_infl_der[i] = find_infl(fit_dflam, i, dd_sign = 'neg',
                  y_lim = c(-100,60), if_plot = F, if_result = T)
}

# f_range: without outliers (range of f_infl_der values with outliers removed)
f_range <- f_infl_der < hout(f_infl_der) & f_infl_der > lout(f_infl_der)

# --- Curve through infl. coord.
# all minima
x = min_dflam; y = f_infl_der[min_dflam]; f_infl_fit <- loess(y ~ x, span = .75)
# far outliers removed
x = which(f_range); y = f_infl_der[x]; f_infl_fit_outl <- loess(y ~ x, span = .75)

# --- Plot infl. coords. & curve
par(mgp = c(1.3, .4, 0), mar = c(2.2, 2.2, .2, .2))
plot(f_infl_der, pch = 19, cex = 1, ylim = c(0, binNum))
# all
lines(f_infl_fit$x, f_infl_fit$fitted, lwd = 3, col = 'purple')
```

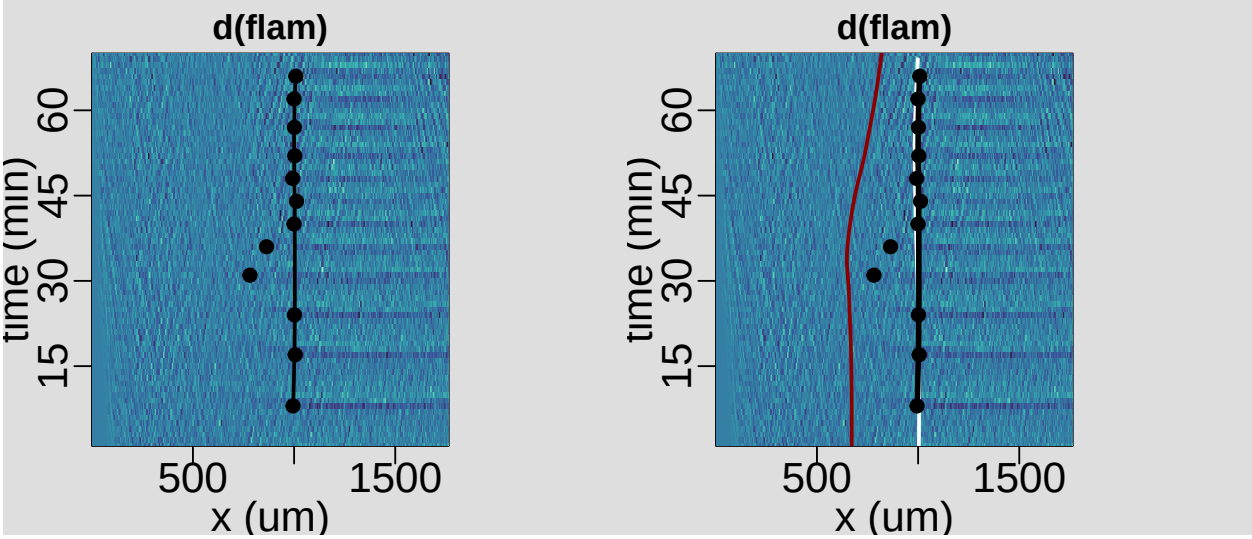
```
# outliers removed
lines(f_infl_fit_outl$x, f_infl_fit_outl$fitted, lwd = 3, col = 'blue')
```



Plot over image Left panel: Plot the identified cAMP relay border over the x-t plot.

Right panel: Plot the identified point of cell population separation (red), gene induction (white) and cAMP relay border (black) over the x-t plot.

```
par(mfrow = c(1,2))
# diff(flamindo)
plotmat(dflam, mako(100), 'd(flam)', n2 = n2, tseq = 15, if_legend = F)
# Identified points. of infl.
points(f_infl_der, 1:tail(min_dflam,1), pch = 19, cex = 1, col = 'black')
# Fit
lines(f_infl_fit_outl$fitted, f_infl_fit_outl$x, lwd = 2, col = 'black')

# diff(flamindo)
plotmat(dflam, mako(100), 'd(flam)', n2 = n2, tseq = 15, if_legend = F)
# Nuclei min fit
lines(min_fit$fitted, min_fit$x, lwd = 2, col = 'darkred')
# Fit
lines(infl_fit_outl$fitted, infl_fit_outl$x, lwd = 2, col = 'white')
# Identified points. of infl.
points(f_infl_der, 1:tail(min_dflam,1), pch = 19, cex = 1, col = 'black')
# Fit
lines(f_infl_fit_outl$fitted, f_infl_fit_outl$x, lwd = 3, col = 'black')
```



Session information

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] Cairo_1.6-0        lmtest_0.9-40      signal_0.7-7       matrixStats_0.61.0
##  [5] zoo_1.8-10         shades_1.4.0       paletteer_1.4.0    dplyr_1.0.8
##  [9] plotfunctions_1.4  viridis_0.6.2      viridisLite_0.4.0  ggplot2_3.3.6
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.2 xfun_0.35        purrr_0.3.4      rematch2_2.1.2
##  [5] ggthemes_4.2.4   lattice_0.20-44  colorspace_2.0-3 vctrs_0.3.8
##  [9] generics_0.1.2   htmltools_0.5.4  yaml_2.3.5       utf8_1.2.2
## [13] rlang_1.0.6      pillar_1.7.0     glue_1.6.1       withr_2.5.0
## [17] lifecycle_1.0.1  stringr_1.4.0    munsell_0.5.0    gtable_0.3.0
## [21] codetools_0.2-18 evaluate_0.15    knitr_1.41       fastmap_1.1.0
## [25] fansi_1.0.2      highr_0.9        scales_1.2.0     farver_2.1.0
## [29] gridExtra_2.3    digest_0.6.29    stringi_1.7.6    grid_4.1.1
```

```
## [33] cli_3.1.1        tools_4.1.1      magrittr_2.0.2   tibble_3.1.6
## [37] crayon_1.5.1     pkgconfig_2.0.3  ellipsis_0.3.2   MASS_7.3-54
## [41] rmarkdown_2.18   rstudioapi_0.13  R6_2.5.1         prismatic_1.1.0
## [45] compiler_4.1.1
```

```
Sys.Date()
```

```
## [1] "2023-09-14"
```