

Display cell images with Rcell (Version 1.1-6)

Alan Bush

November 3, 2011

1 Introduction

Rcell uses the functions of **EBImage** package to manipulate and display the images processed by **Cell-ID**. The main purpose of the functions described in this document is to get a quick look at cells in different conditions, channels and times. **cimage** function crops images from single cells and displays them according to a user define arrangement.

If you haven't done so, read the "Getting Started with Rcell" document before proceeding.

```
> vignette("Rcell")
```

Make sure you have the **EBImage** package installed in your system. This package is quite hard to install, follow instructions from the VCell-ID-Rcell-Installation-Guide at <http://sourceforge.net/projects/cell-id/files/> or from <http://bioconductor.wustl.edu/bioc/html/EBImage.html>. To test if the package is working correctly try the following commands. A picture of Lena should be displayed.

```
> library(EBImage)
> example(display)
```

2 Display cell images

If you haven't done so, load the **Rcell** package and the filtered example dataset with

```
> library(Rcell)
> data(ACL394filtered)
```

When analyzing a dataset, you usually want to take a look at the cells images that are generating the data points. This helps interpret the data and gives you confidence on the result. To visualize a random set of cells from a image, you have to specify position, channel and t.frame (if you are dealing with a time course). For example, to visualize some BF images of cells from position 29 and t.frame 11 use the following commands¹.

```
> img1 <- cimage(X, subset = pos == 29 & t.frame == 11, channel = "BF", bg.col = "white")
```

This function displays the image shown in Figure 1, and returns a Image object (saved to **img1**).

As all **Rcell** functions, the first argument of **cimage** is the cell.data object that you wish to visualize. This function first subsets the cell.data object X according to the *subset* argument, as many other **Rcell** functions. This is useful to select cells and times, but you can't use this argument to select the channel you want to see. Instead you can use the *channel* argument for this. Note that you can select several channels

¹To save space, only some images of the example dataset were included in the package. Changing the *subset* or the *channel* arguments might result in errors if the specified images are not found.

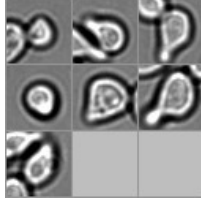


Figure 1: BF images of random cells selected from position 29, t.frame 11

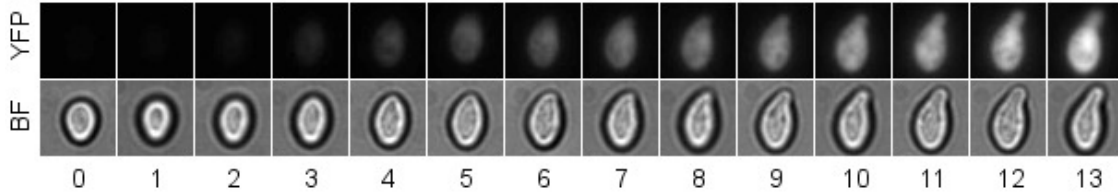


Figure 2: Time course strips for cell 5 of position 29

(see below). `cimage` then takes a random sample of cells from those which passed the `subset` argument. The default sample size is seven, but you can specify it with the `N` argument. If you set `N` to `NA`, no sampling is applied and all selected cells are shown. The position each cell took in the image was arbitrary in Figure 1, they were just tiled together to make a square arrangement. But position can have a meaning. A normal way to display cell images is to show a time course strips, where different channels are stacked one over the other. `cimage` can easily produce this sort of images (Figure 2).

```
> img2 <- cimage(X, channel ~ t.frame, subset = pos == 29 & cellID == 5, channel = c("BF",
+      "YFP"), bg.col = "white")
```

The second argument `cimage` is the *formula* that specifies the position of individual images. The first term indicates the y variable, `channel` in this example, so different channels will have different y coordinates. The right term specifies which variable is going to be used as the x coordinate, `t.frame` in this case. In this example a single cell was explicitly selected with the `subset` argument. When you select more than one cell per group, you have to specify how you want them to be layout on the image. To specify different cells within a sample you can use three dots (...), as shown in Figure 3.

```
> img3 <- cimage(X, ... + channel ~ t.frame, subset = pos == 29, channel = c("BF",
+      "YFP"), N = 4, bg.col = "white")
```

Note that you can use more than one variable in each term of the formula, separated by the plus operator (+). The order matters, the last variable to the right varies faster. In this example (Figure 3) `channel` is animated in each cell (you can think of the dots as different cells).

The `channel.subset` argument allows you to do complex selection of `channels` and `t.frame`. For example you might be interested in the YFP channel, but would like to see the cell boundary found by Cell-ID on a BF image for a single time frame (Figure 4).

```
> img4 <- cimage(X, ... ~ channel + t.frame, subset = pos == 29, channel.subset = channel ==
+      "YFP" | (channel == "BF.out" & t.frame == 11), N = 4, bg.col = "white")
```

You can select the “out” images generated by Cell-ID by appending “.out” to the channel name.

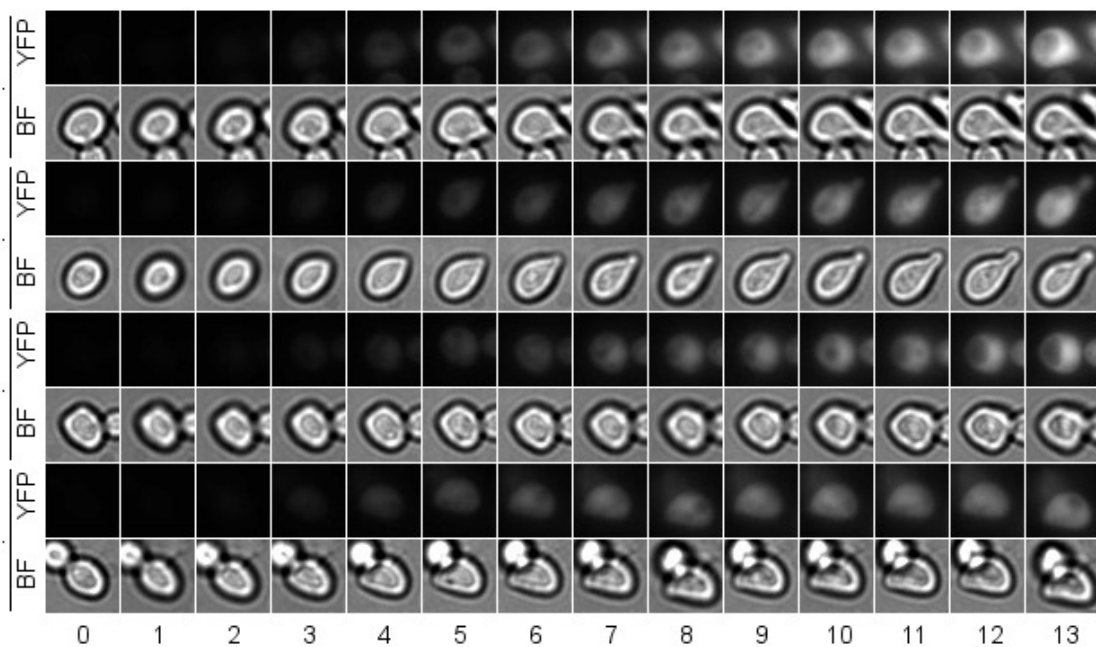


Figure 3: Time course strips for 4 randomly chosen cells

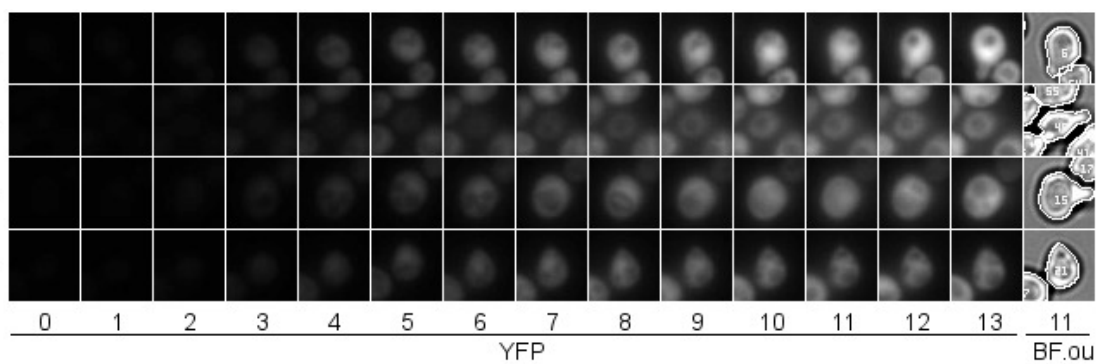


Figure 4: YFP time course strips for 4 randomly chosen cells, with a single BF image

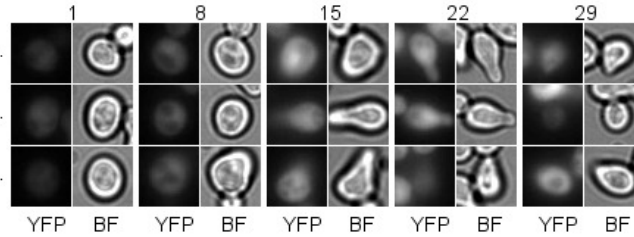


Figure 5: sample against channel, faceted by position

3 Faceting your image plot

In the same way as for `cplot`, you can define *facets* for the image plot. The facets are specified with formula notation, just as the positions of the images within a facet. If only one term of the formula is specified, the facets will be wrapped around the image to save space² (Figure 5).

```
> img5 <- cimage(X, ... ~ channel, facets = ~pos, subset = t.frame == 11 & pos %in%
+   c(1, 8, 15, 22, 29), channel = c("YFP", "BF"), bg.col = "white", N = 3,
+   facets.nx = 5)
```

4 Plotting images against continuous variables

An interesting plot can be obtained if we choose the position of the image according to a continuous variable. First suitable bins of the continuous variables have to be created, we can use the `cut` function for this.

```
> X <- transform(X, cut.fft.stat = cut(fft.stat, 20))
> X <- transform(X, cut.f.tot.y = cut(f.tot.y, 20))
```

Once these variables are created we can use them to arrange the images of the cells (Figure 6).

```
> img6 <- cimage(X, cut.f.tot.y ~ cut.fft.stat, facets = ~channel, subset = t.frame ==
+   11 & pos %in% c(1, 8, 15, 22, 29), channel = c("YFP", "BF.out"), bg.col = "white",
+   N = 1)
```

References

- Pau, Fuchs et al. (2010). EBIImage: an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26(7):979-981.
- Colman-Lerner, Gordon et al. (2005). Regulated cell-to-cell variation in a cell-fate decision system. *Nature*, 437(7059):699-706.
- Chernomoretz, Bush et al. (2008). Using Cell-ID 1.4 with R for Microscope-Based Cytometry. *Curr Protoc Mol Biol.*, Chapter 14:Unit 14.18.

²In this case the *facets.nx* argument can be used to indicate the number of facets columns

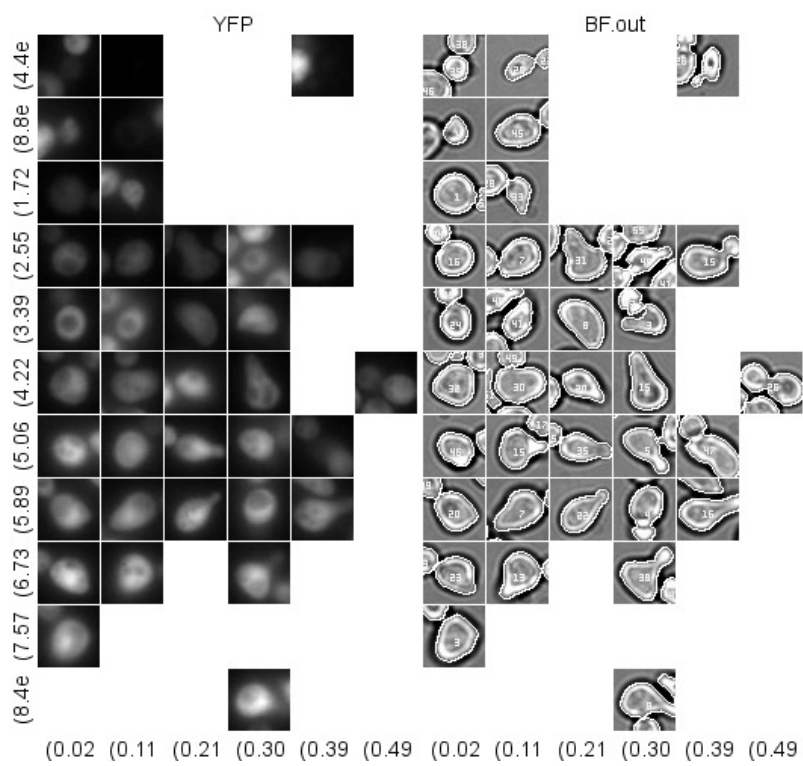


Figure 6: $f.tot.y$ vs $fft.stat$, faceted by channel