ddPCRclust — An R package and Shiny app for automated analysis of multiplexed ddPCR data

Benedikt G. Brink, Justin Meskas, Ryan R. Brinkman

December 29, 2017

Contents

1 Introduction . . . 1
2 Installation . . . 2
3 Methods . . . 2
4 Usage . . . 2
4.1 Input data: . . . 3
4.2 Clustering: . . . 3
4.2.1 flowDensity . . . 3
4.2.2 SamSPECTRAL . . . 4
4.2.3 flowPeaks . . . 4
4.3 Copies per droplet: . . . 5
4.4 Exporting results: . . . 5
5 Visual Interface . . . 5

1 Introduction

Droplet digital PCR (ddPCR) is an emerging technology for quantifying DNA. By partitioning the target DNA into $\sim$20 000 droplets, each serving as its own PCR reaction compartment, ddPCR has significantly increased sensitivity compared to other technologies for DNA quantification. However, manual analysis of the data is time consuming and algorithms for automated analysis of non-orthogonal, multiplexed ddPCR data are unavailable, presenting a major bottleneck for the advancement of ddPCR transitioning from low-throughput to high-throughput. During a ddPCR run, each genetic target is fluorescently labelled with a combination of two fluorophores (typically HEX and FAM), giving it a unique footprint in a two-dimensional space represented by the intensities per colour channel. The position of each droplet within this space reveals how many and, more importantly, which genetic targets it contains. Thus, droplets that contain the same targets cluster together. The number of positive droplets for each target determine its abundance.

ddPCRclust is an R package for automated analysis of multiplexed ddPCR data. It can automatically analyse and visualise multiplexed ddPCR experiments with up to four targets per reaction. Results are on par with manual analysis, but only take minutes to compute instead of hours. The accompanying Shiny app ddPCRvis provides easy access to the functionalities of ddPCRclust through a web-browser based GUI.

2 Installation

The algorithm was implemented in R and can be installed as a package. You can install this package like any other package from GitHub using devtools:

```
> library(devtools)
> install_github("bgbrink/ddPCRclust")
```

Or directly from Bioconductor:

```
> ## try http:// if https:// URLs are not supported
> source("https://bioconductor.org/biocLite.R")
> biocLite("ddPCRclust")
```

3 Methods

As aforementioned, data from ddPCR consists of a number of different clusters $l_1, \ldots, l_k$ and their respective centroids $c_1, \ldots, c_k$, where k is the number of clusters. All droplets $(x_1, \ldots, x_m)$ represent one or more genetic targets $t_1, \ldots, t_n$, where m is the number of droplets and n is the number of targets. Each cluster l_i is defined as a group of droplets that contain an identical combination of targets. We define four steps to successfully analyze this data:

1. Find all cluster centroids c.
2. Assign one or multiple targets t to each cluster l based on c.
3. Allocate the rain and assign a cluster label l to each droplet x.
4. Determine the number of positive droplets for each target t and calculate the CPDs.

4 Usage

The main function of the package is `ddPCRclust`. This function runs the algorithm with one or multiple files, automatically distributing them among all CPU cores (no parallelisation on Windows). We provide eight exemplary ddPCR files along with this package. Analyse them using the following commands.

```
> # Run ddPCRclust
> library(ddPCRclust)
> exampleFiles <- list.files(paste0(find.package("ddPCRclust"), "/extdata"), full.names = TRUE)
> result <- ddPCRclust(files = exampleFiles[3], template = exampleFiles[9])
```

```
> # Plot the results
> library(ggplot2)
> p <- ggplot(data = result$results$B01$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
> p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
+     ggtitle("ddPCRclust example")+theme_bw() + theme(legend.position="none")
> p
```

4.1 Input data:

The input data are one or multiple CSV files containing the raw data from Bio-Rad's droplet digital PCR systems (QX100 and QX200). Each file can be represented as a two-dimensional data frame. Each row within the data frame represents a single droplet, each column the respective intensities per colour channel.

4.2 Clustering:

The initial clustering to find the centroids is based on three different approaches; flowDensity [1], SamSPECTRAL [2], and flowPeaks [3]. Each approach has its own function within ddPCRclust, provided users need more granular control.

4.2.1 flowDensity

Originally designed for gating of flow cytometry data, *flowDensity* identifies cell populations in a dataset using characteristics of the density distribution (e.g. the number, height and width of peaks and the slope of the distribution curve). Parameters can be adjusted on a population-specific basis. We use the density function to find local peaks above a threshold, which represent the centres of clusters. The method comprises the following steps:

1. Remove all x where $(x_1, x_2) < 0.125 \cdot max(x_1, x_2)$. The bottom $12.5\,\%$ of the data space is known to contain the negative population, i.e. the droplets without any of the targets of interest.
2. Find the highest density peaks with $max(x_1)$ and $max(x_2)$, respectively. We define these as the two outer primary clusters y and z, since the primary clusters empirically contain the majority of the non-negative events.
3. Rotate the data with $\theta = |atan(\frac{y_2 - z_2}{y_1 - z_1})|$.
4. Cut the rotated data above the two outer clusters in a staircase manner and find all density peaks.
5. Take the previously removed data and repeat steps 2 and 4, until all clusters are found.

Clusters are then labelled based on their rotated position and lastly the rain is assigned.

We provide eight exemplary ddPCR files along with this package. Analyse one of them using the following commands.

```
> # Run the flowDensity based approach
> library(ddPCRclust)
> exampleFiles <- list.files(paste0(find.package("ddPCRclust"), "/extdata"), full.names = TRUE)
```

```
> file <- read.csv(exampleFiles[3])
> densResult <- runDensity(file = file, numOfMarkers = 4)
> # Plot the results
> library(ggplot2)
> p <- ggplot(data = densResult$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
> p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
+     ggtitle("flowDensity example")+theme_bw() + theme(legend.position="none")
> p
```

4.2.2 SamSPECTRAL

Since spectral clustering is computationally expensive ($\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space), *SamSPECTRAL* uses density based pre-processing to reduce the number of edges in the graph. To do so, a faithful sampling algorithm builds m communities, which are then connected to a graph where the edges represent the similarity between corresponding communities. The spectrum of this graph is subsequently analysed using classical spectral clustering to find the clusters. Finally, the clusters are combined based on their similarity in the community graph and a cluster number for each event in the original data is returned. We use this implementation of spectral clustering and choose m encompassing 5 % of the data, which has empirically proven to be a good compromise between accuracy and speed. However, users can choose a different value if necessary. Clusters are then labelled based on their position and lastly the rain is assigned.

We provide eight exemplary ddPCR files along with this package. Analyse one of them using the following commands.

```
> # Run the SamSPECTRAL based approach
> library(ddPCRclust)
> exampleFiles <- list.files(paste0(find.package("ddPCRclust"), "/extdata"), full.names = TRUE)
> file <- read.csv(exampleFiles[3])
> samResult <- runSam(file = file, numOfMarkers = 4)
> # Plot the results
> library(ggplot2)
> p <- ggplot(data = samResult$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
> p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
+     ggtitle("SamSPECTRAL example")+theme_bw() + theme(legend.position="none")
> p
```

4.2.3 flowPeaks

The third approach uses the *flowPeaks* package [3]. The *flowPeaks* algorithm first uses a two step k-means clustering with a large k, in order to partition the dataset into many compact clusters. The result is then used to generate a smoothed density function. All local peaks are exhaustively found by exploring the density function and the clusters are merged according to their local peaks. Clusters are then labelled based on their position and lastly the rain is assigned.

We provide eight exemplary ddPCR files along with this package. Analyse one of them using the following commands.

```
> # Run the flowPeaks based approach
> library(ddPCRclust)
> exampleFiles <- list.files(paste0(find.package("ddPCRclust"), "/extdata"), full.names = TRUE)
> file <- read.csv(exampleFiles[3])
> peaksResult <- runPeaks(file = file, numOfMarkers = 4)
> # Plot the results
> library(ggplot2)
> p <- ggplot(data = peaksResult$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
> p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
+      ggtitle("flowPeaks example")+theme_bw() + theme(legend.position="none")
> p
```

4.3 Copies per droplet:

Once all droplets are correctly assigned, the actual copies per droplet (CPDs) for each target are calculated by the function `calculateCPDs` according to Equation 1,

$$CPD_i = -ln(1 - \frac{C_i}{C_T}) \quad (1)$$

where C_i is the total number of positive droplets for target i and C_T the total droplet count.

4.4 Exporting results:

The results can be exported using `exportPlots`, `exportToExcel`, and `exportToCSV`.

5 Visual Interface

Furthermore, we developed ddPCRvis, a GUI that gives access to the aforementioned functionalities of the ddPCRclust package directly through a web browser, powered by R Shiny [4]. It also enables the user to check the results and manually correct them if necessary. The interface is available online at https://bibiserv.cebitec.uni-bielefeld.de/ddPCRvis/ or for download at https://github.com/bgbrink/ddPCRvis/.

References

[1] Mehrnoush Malek, Mohammad Jafar Taghiyar, Lauren Chong, Greg Finak, Raphael Gottardo, and Ryan R Brinkman. flowDensity: reproducing manual gating of flow cytometry data by automated density-based cell population identification. *Bioinformatics*, 31(4):606–607, 2015.

[2] Habil Zare, Parisa Shooshtari, Arvind Gupta, and Ryan R Brinkman. Data reduction for spectral clustering to analyze high throughput flow cytometry data. *BMC bioinformatics*, 11(1):403, 2010.

[3] Yongchao Ge and Stuart C Sealfon. flowPeaks: a fast unsupervised clustering for flow cytometry data via K-means and density peak finding. *Bioinformatics*, 28(15):2052–2058, 2012.

[4] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2017. R package version 1.0.3. URL: https://CRAN.R-project.org/package=shiny.