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

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

**October 27, 2020**

## Contents



## 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, non-orthogonal 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. For details on the experimental setup please refer to Hughesman et al. [1].

# 2 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. Load files from your file system with `readFiles`.

| Ch1 Amplitude | Ch2 Amplitude |
|---|---|
| 2360.098 | 6119.26953 |
| 2396.3916 | 1415.31665 |
| 2445.838 | 6740.79639 |
| 2451.63867 | 1381.74683 |
| 2492.55884 | 1478.19617 |
| 2519.6355 | 7082.25049 |
| ⋮ | ⋮ |

You can also set up a run template to tell ddPCRclust which and how many markers were used in each experiment.

| Well | Sample type | No of markers | Marker 1 | Marker 2 | Marker 3 | Marker 4 |
|---|---|---|---|---|---|---|
| B01 | Blood | 4 | a | b | c | d |
| G01 | FFPE | 4 | a | b | c | d |
| F02 | Blood | 3 | a | | c | d |
| D03 | FFPE | 3 | a | | c | d |
| A04 | FFPE | 4 | a | b | c | d |
| G07 | Cell line | 3 | a | | c | d |
| G08 | Cell line | 3 | a | | c | d |
| E09 | FFPE | 2 | | | c | d |

# 3 Methods

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. ddPCRclust performes 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.

## 3.1 ddPCRclust

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:

```
> library(ddPCRclust)
> # Read files
> exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'),
+                            full.names = TRUE)
> files <- readFiles(exampleFiles[3])
> # To read all example files uncomment the following line
> # files <- readFiles(exampleFiles[1:8])
>
> # Read template
> template <- readTemplate(exampleFiles[9])
> # Run ddPCRclust
> result <- ddPCRclust(files, template)
> # Plot the results
> library(ggplot2)
> p <- ggplot(data = result$B01$data,
+             mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
> p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
+      ggtitle('B01 example')+theme_bw() + theme(legend.position='none')
> p
```

### 3.1.1 Parameters

- files: The input data obtained from the csv files. For more information, please see 2.
- template: A data frame containing information about the individual ddPCR runs. For more information, please see 2.
- numOfMarkers: The number of primary clusters that are expected according the experiment set up. Can be ignored if a template is provided. Else, a vector with length equal to `length(files)` should be provided, containing the number of markers used for the respective reaction.
- sensitivity: The clustering algorithms can be tweaked in order to partition the data into more or fewer clusters. A sensible value lies between 0.1 and 2, the standard is 1. A higher value means the data is divided into more clusters, a lower value means more clusters are merged. This allows fine tuning of the algorithm for exceptionally low or high CPDs.
- similarityParam: If the distance of a droplet between two or more clusters is very similar, the algorithm can't be sure where it belongs. Therefore, it will not be counted for either if the ratio of the distances from cluster 1 and cluster 2 is larger than similarityParam (assuming cluster 2 $>$ cluster 1). The standard is 0.95, i.e. at least 95% similarity. A sensible value lies between 0 and 1, where 0 means none of the 'rain' droplets will be counted and 1 means all droplets will be counted.

- distanceParam: When assigning rain between two clusters, typically the bottom 20% are assigned to the lower cluster and the remaining 80% to the higher cluster. This parameter changes the ratio, i.e. a value of 0.1 would assign only 10% to the lower cluster.
- fast: Run a simpler version of the algorithm that uses only flowDensity (see 3.2.1) and thus is about 10x faster. For clean data, this might already deliver very good results. However, it is mostly intended to get a quick overview over the data.
- multithread: Distribute the algorithm among all CPU cores to speed up the computation if set to TRUE (not available on Windows). Default is FALSE.

## 3.2 Initial clustering

The initial clustering of the data (see step 1 in section 3) is performed by three different clustering packages, which have been adjusted to work on ddPCR data. Each algorithm has its own function, in case users need low level control.

### 3.2.1 flowDensity

Originally designed for gating of flow cytometry data, *flowDensity* [2] 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.

This part of the algorithm can be accessed with the function `runDensity`. The parameters are the same as discussed in 3.1.1.

### 3.2.2 SamSPECTRAL

Since spectral clustering is computationally expensive ($\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space), *SamSPECTRAL* [3] 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.

This part of the algorithm can be accessed with the function `runSam`. The parameters are the same as discussed in 3.1.1.

### 3.2.3 flowPeaks

The third approach uses the *flowPeaks* package [4]. 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.

This part of the algorithm can be accessed with the function `runPeaks`. The parameters are the same as discussed in 3.1.1.

## 3.3 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.

### 3.3.1 Parameters

- results: The result of the ddPCRclust algorithm.
- template: A data frame containing information about the individual ddPCR runs. For more information, please see 2.
- constantControl: The name of a marker that has been used as a constant control. This marker has to be present in *every* row of the provided template (i.e. in every reaction). The results will be normalised against this control.

## 3.4 Exporting results

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

# 4 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 [5]. 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] Curtis B Hughesman, XJ David Lu, Kelly YP Liu, Yuqi Zhu, Catherine F Poh, and Charles Haynes. A Robust Protocol for Using Multiplexed Droplet Digital PCR to Quantify Somatic Copy Number Alterations in Clinical Tissue Specimens. *PloS one*, 11(8):e0161274, 2016.

[2] 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.

[3] 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.

[4] 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.

[5] 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.