ddpcr: an R package and web application for analysis of droplet digital PCR data [version 1; peer review: 2 approved]

Dean Attali¹,², Roza Bidshahri²,³, Charles Haynes²,³, Jennifer Bryan²,⁴

¹Bioinformatics Training Program, University of British Columbia, Vancouver, Canada
²Michael Smith Laboratories, University of British Columbia, Vancouver, Canada
³Department of Chemical and Biological Engineering, University of British Columbia, Vancouver, Canada
⁴Department of Statistics, University of British Columbia, Vancouver, Canada

Abstract
Droplet digital polymerase chain reaction (ddPCR) is a novel platform for exact quantification of DNA which holds great promise in clinical diagnostics. It is increasingly popular due to its digital nature, which provides more accurate quantification and higher sensitivity than traditional real-time PCR. However, clinical adoption has been slowed in part by the lack of software tools available for analyzing ddPCR data. Here, we present ddpcr – a new R package for ddPCR visualization and analysis. In addition, ddpcr includes a web application (powered by the Shiny R package) that allows users to analyze ddPCR data using an interactive graphical interface.

Keywords
droplet digital PCR, shiny, bioinformatics, personalized medicine, Rpackage, gating, Gaussian mixture models, kernel density estimates

This article is included in the RPackage gateway.

Corresponding author: Jennifer Bryan (jenny@stat.ubc.ca)
Competing interests: No competing interests were disclosed.
Grant information: CH receives support as a Canada Research Chair. This work was further supported by the Canadian Institutes of Health Research (CIHR) Bioinformatics Training Program scholarship.
Copyright: © 2016 Attali D et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Data associated with the article are available under the terms of the Creative Commons Zero "No rights reserved" data waiver (CC0 1.0 Public domain dedication).
How to cite this article: Attali D, Bidshahri R, Haynes C and Bryan J. ddpcr: an R package and web application for analysis of droplet digital PCR data [version 1; peer review: 2 approved] F1000Research 2016, 5:1411 (https://doi.org/10.12688/f1000research.9022.1)
**Introduction**

Droplet digital polymerase chain reaction (ddPCR) accurately quantifies targeted nucleic acid sequences (templates) by randomly partitioning sample DNA into isolated droplets, such that most droplets contain at most one template. The template within each droplet is then amplified and detected in a sequence-specific manner using a hydrolysis probe. The counting of droplets emitting a sequence-specific fluorescent signal permits the number of copies of that sequence present in the sample to be quantified with excellent sensitivity and precision. Different templates, such as wild-type and mutant alleles, may be quantified by using a uniquely labeled probe against each. The most commonly used reporter dyes on the probes are FAM (fluorescein) and HEX™, with the end-point fluorescence amplitudes for the two dyes measured by analyzing each droplet with a two-channel fluorescence detector.

ddPCR data readily lends itself to visualization as a two-dimensional scatter plot (Figure 1), in which the fluorescence amplitudes in both channels are plotted against each other for every droplet. In a ddPCR experiment designed to quantify two different templates, droplets ideally segregate into unique groups (clusters) that may include HEX-positive, FAM-positive, double-positive, and double-negative (empty) clusters. For example, distinct FAM-positive, double-positive, and empty droplet clusters can be seen in Figure 5B. In practice, some droplets record an ambiguous set of fluorescent signals that fall between the distinct positive and negative populations. Such droplets are termed “rain” and can be observed between all clusters. By gating the droplets into groups based on their fluorescence signals, the exact number of template-positive droplets can be counted to provide exact quantification in a digital form.

**Motivation**

Quantification of template abundance from raw ddPCR data begins with assigning each droplet to a unique cluster or to rain. The QuantaSoft program (Bio-Rad, Hercules, CA) is designed to perform these assignments either via manual gating, with the usual disadvantages of subjectivity and non-reproducibility, or automatic gating. The algorithm used in the latter case is proprietary and can produce unsatisfactory results, especially when applied to ddPCR data obtained from formalin-fixed paraffin-embedded (FFPE) samples, as exemplified in Figure 5A.

Two third-party tools for automatic gating of ddPCR data have been described to date: ‘definetherain’ by Jones et al. and ‘ddpcRquant’ by Trypsteen et al. However, both are limited to single-channel ddPCR data and are therefore not applicable to increasingly common two-channel experiments such as shown in Figure 1. Given the lack of tools for such analyses, users must currently resort to manual droplet gating.

**Methods**

**Overview**

To improve automated droplet assignments as well as permit visualization of ddPCR datasets, we have developed *ddpcr*, an R package that can be used to explore, visualize, and analyze two-channel ddPCR data. The R language was chosen because it is open-source and cross-platform, which allows anyone to use it freely on any operating system. R is also a popular language in the field of computational biology, and is the main data analysis language for many scientists. To improve access and ease of use, we also implemented an interactive web application using Shiny, through which one can run the analysis using a simple point-and-click interface.

*ddpcr* has been thoroughly tested using R versions 3.2.3 and 3.3.0 on both Windows 7 and Ubuntu 14.04.2 machines. However, the package is likely to run on any machine with a working installation of R.

**Plate object**

The most important object in the *ddpcr* package is the *ddPCR_plate* object, or simply referred to as the “plate object”. A plate object represents all the data for experiments conducted on a 96-well PCR plate. It gets created either by loading ddPCR input data files (see ‘Data import’) into a new plate object, or by loading an existing plate object that was previously saved to disk. A plate object contains all the information required to analyze the droplets within each well of a particular ddPCR plate. A plate object is both the input and output of all the core analysis functions.

**Workflow**

To use the *ddpcr* package, it must first be installed and loaded.

```
install.packages("ddpcr")
library("ddpcr")
```
A very simple analysis workflow using a sample dataset can be performed using the following code, with the result of the code shown in Figure 2:

```r
dir <- sample_data_dir()
my_data <- new_plate(dir, type = plate_types$fam_positive_pnpp)
my_data <- subset(my_data, "F05")
my_data <- analyze(my_data)
plot(my_data, show_drops_empty = TRUE, show_grid_labels = TRUE)
```

While `ddpcr` contains dozens of functions, most analyses will follow a similar pattern: load ddPCR data into R using the `new_plate()` function, run the automated analysis using `analyze()`, and then explore the results using a variety of functions (Figure 3). The `plot()` function is used to visualize a dataset using `ggplot2`, while the `plate_meta()` and `plate_data()` functions return the dataset’s metadata and droplet grouping data as R data frames, respectively. The `save_plate()` function can be called at any time to save the current state of the dataset to disk in a format that can be loaded back into `ddpcr`.

The example code above uses a sample dataset, but in order to use new data, ddPCR data must be exported from QuantaSoft, as described in the next section. For more complex analysis or customizing the analysis parameters, see the full list of functions available by running `?ddpcr`.

Data import
Before beginning analysis on a novel dataset, the first step is to import the ddPCR droplet fluorescence data into R. The raw data obtained from the fluorescence detector is encoded in a proprietary format that cannot be read by any software other than QuantaSoft, so the data must first be opened in QuantaSoft and exported into an accessible file format. QuantaSoft offers an option to export the droplet event data as a set of CSV (comma-separated values) files, as well as an option to export a metadata file that contains information on each well (Supplementary Figure 1 and Supplementary Figure 2). These CSV files are used as the input to `ddpcr`.

Analysis algorithm
The analysis automatically gates droplets into unique clusters using kernel density estimation and Gaussian mixture models applied to the droplet fluorescence amplitudes. The full algorithm

![Figure 2. ddPCR data from well F05 of the sample dataset analyzed using ddpcr.](image)

![Figure 3. Basic workflow for analyzing ddPCR data using the ddpcr package.](image)
Please note that the text contains a reference to a figure (Figure 1), which is not visible in the image provided. The text continues with a detailed explanation of the ddPCR analysis steps and the use of QuantaSoft's ddPCR package in R.

**Implementation**

**Plate objects are lists.** Every S3 object in R has a base type upon which it is built. The plate object is implemented as an S3 object of class `ddpcr_plate` with the R list as its base type. Using a list allows for an easy way to bundle together the several different R objects describing a plate into one. All information required to analyze a plate is part of the plate object. Every plate object contains a set of nine elements that together fully describe and reproduce the current state of the dataset: `plate_data`, `plate_meta`, `name`, `params`, `status`, `clusters`, `steps`, `dirty`, `version`.

**Using S3 to override base generic functions.** Since the plate object is an S3 object, it can benefit from the use of generic functions. There are three common generic functions that the plate object implements: `print()`, `plot()`, and `subset()`. The `print()` method does not take any extra arguments and is used to print a summary of a plate object in a visually appealing way to the console. It gives an overview of the most important parameters of the plate such as its name and size. The `plot()` method generates a scatter plot of every well in the dataset and can be highly customizable using the many arguments it supports. While the base `plot()` method in R uses base R graphics, the `plot()` method for `ddpcr_plate` objects uses the `ggplot2` package. The `subset()` generic is overriden by a method that is used to retain only a subset of wells from a larger plate.

**Plate types.** A ddPCR assay can be characterized by the droplet populations that are expected to arise after amplification. For example, in a (FAM+)/(FAM-HEX-) assay (such as Figure 1) it is expected that most of the non-empty droplets will either be FAM-HEX+ or FAM+, but not HEX+. Similarly, a (HEX+)/(FAM-HEX-) assay means that there are expected to be no droplets that are only FAM+.

To describe these two types of assays, we define the term “PN/PP” (positive-negative/positive-positive). This name is a reflection of the expected populations of non-empty droplets: one population of singly-positive droplets (such as HEX+ or FAM+), and one population of double-positive droplets.

This characterization of a ddPCR experiment defines the plate type of a plate object, and it determines what type of analysis to run on the data. The default and most basic plate type is `ddpcr_plate`, which can be used for any ddPCR dataset. Running the analysis on a plate of this type will perform the first few analysis steps of identifying failed wells, outlier droplets, and empty droplets, but will not carry out the automated gating. Since in PN/PP-type experiments there is a rough expectation of where the droplets should be, automated gating can ensue on plates of that type.

**Using S3 to support inheritance.** Inheritance means that every plate type has a parent plate type from which it inherits all its features, while specific behaviour can be added or modified. In `ddpcr`, transitive inheritance is implemented, which means that features are inherited from all ancestors rather than only the most immediate one. Multiple inheritance is not supported, meaning that each plate object can only have one parent.

The notion of inheritance is an important part of the `ddpcr` package, as it allows ddPCR data from different assay types to share many properties. For example, PN/PP assays are first treated using the analysis steps common to all ddPCR experiments, and then gated with an assay-specific step, so PN/PP assays can be thought of as inheriting the analysis from general ddPCR assays. Furthermore, the two types of PN/PP assays share many similarities, so they both inherit from a common PNPP plate type. Another benefit of inheritance in `ddpcr` is that it allows users to easily extend the functionality of the package by adding custom ddPCR plate types to gate different types of experiments. More information, including a fully worked example, on how to add a new plate type can be found in the package vignette (see ‘Software availability’).

**Shiny web application**

The `ddpcr` package includes a web application that allows users to perform an analysis of ddPCR data in an interactive visual environment. The web application, written using the Shiny package v0.11, implements most of the features available in the `ddpcr` package and makes them accessible via a simple point-and-click interface. The Shiny application can be a useful tool for persons not comfortable with R programming or simply as a more convenient way to perform an analysis. However, since the web application only supports a curated subset of the `ddpcr` functions, it is not as powerful as using the command-line interface.

The `ddpcr` Shiny application includes four main tabs that mimic the natural flow of a ddPCR analysis (Figure 4): upload a dataset, configure analysis parameters, analyze the plate, and explore the results. At any point during the session, the current plate object can be downloaded and saved, and can be loaded into either the R command-line or the web application at a later time to continue the analysis.
The application is freely available online at http://daattali.com/shiny/ddpcr and is hosted on a server located in San Francisco, California. All data that is uploaded to the application is deleted when a user session ends, and none of the data is stored permanently. However, some users may prefer to run the application locally, which can be done using the `ddpcr::launch()` function.

**Use case**

**Dataset 1.** Raw ddPCR data from application of the ddPCR assay against BRAF-V600 mutations

http://dx.doi.org/10.5256/f1000research.9022.d126032

This data can be loaded and displayed in QuantaSoft™. Column 12 on the plate is from a different experiment and is not considered part of the dataset.

**Dataset 2.** The set of exported CSV files of the data presented in Dataset 1

http://dx.doi.org/10.5256/f1000research.9022.d126033

We have applied ddpcr to data (Dataset 1) from a novel ddPCR assay against somatic point mutations in the BRAF-V600 codon that was applied to FFPE specimens from a cohort of colorectal cancer (CRC) patients. V600 mutations are observed in approximately 10% of colorectal tumours and their detection in CRC patients helps determine disease prognosis and treatment regimen. Through its droplet gating algorithm, ddpcr accurately identified droplet clusters and the number of droplets within each to provide the information needed to compute the frequency of mutated BRAF genes (Supplementary Figure 3).

To assess the accuracy of results from ddpcr, we compared BRAF-V600 mutation frequencies determined from the output of ddpcr with results obtained by two independent methods. V600 mutation frequencies computed from automated ddpcr results were within 3% of those obtained by manual analysis of the ddPCR data by an experienced operator (Supplementary Figure 4 and Supplementary Table 1). In addition, the BRAF-V600 status for each sample in the entire cohort was classified as mutant or wild-type by a certified pathologist using an immunohistochemical staining assay. We obtained complete agreement between the pathologist’s binary classification of BRAF status and that determined using ddpcr.

We also analyzed the same dataset using QuantaSoft version 1.6.6. FAM-positive and double-positive droplets were not recognized as distinct clusters in 9 out of the 16 mutant-positive BRAF samples (Figure 5A).

**Discussion**

We present ddpcr, an R package that allows users to analyze ddPCR data and explore the results, both programmatically using R and via an interactive web application. To demonstrate clinical utility, a case study performed on a cohort of CRC patients showed that BRAF-V600 mutation frequencies determined using ddpcr are verified using two independent methods. The analysis runtime was 17 seconds, observed on a 64-bit Ubuntu 14.04.2 machine.
with 512MB of RAM and a single core Intel(R) Xeon(R) CPU E5-2630 at 2.30GHz. The package documentation includes details on extending the package, explanations of the algorithms used, and a walkthrough of a fully worked example.

Data availability
F1000Research: Dataset 1. Raw ddPCR data from application of the ddPCR assay against BRAF-V600 mutations, 10.5256/f1000research.9022.d12603

F1000Research: Dataset 2. The set of exported CSV files of the data presented in Dataset 1., 10.5256/f1000research.9022.d12603

Dataset 1 is also available as a sample dataset within the ddpcr package. To access the data via the web application, select the tab Use sample dataset, choose Large dataset, and then click Load data. To access the data in R, run the following command to store the dataset as a plate object: my_data <- ddpcr::sample_plate("large").

Software availability
Software available from: http://cran.r-project.org/package=ddpcr or https://github.com/daattali/ddpcr

The free web tool can be accessed online at: (http://daattali.com/shiny/ddpcr); or run locally via the ddpcr package with the command ddpcr::launch().

Figure 5. Comparison between droplet gating in (A) QuantaSoft and (B) ddpcr. Both tools analyzed the same ddPCR experiment (well F05) from an assay designed to quantify wild-type (double-positive) and mutant (FAM-positive) alleles of the BRAF gene. (A) QuantaSoft failed to assign the double-positive and FAM-positive droplets into unique clusters, instead assigning all droplets recording a high FAM signal to a single cluster; (B) ddpcr assigned droplets into one of three uniquely identified clusters (double-positive (green), FAM-positive (orange), and empty (black)), or rain (blue).

Latest source code: https://github.com/daattali/ddpcr

Archived source code at time of publication: https://dx.doi.org/10.6084/m9.figshare.3423725

License: MIT

Author contributions
DA wrote the code and produced the figures. RB ran the ddPCR experiments. JB provided ideas and feedback for analysis algorithm. CH provided ideas and feedback for the functionality of the ddpcr package. DA wrote the manuscript with feedback from JB, RB, and CH. All authors approved the final manuscript.

Competing interests
No competing interests were disclosed.

Grant information
CH receives support as a Canada Research Chair. This work was further supported by the Canadian Institutes of Health Research (CIHR) Bioinformatics Training Program scholarship.

Acknowledgments
We would like to thank Dr. Ryan Brinkman and his lab members for their time and advice.
Supplementary material

Figure 1. Exporting droplet fluorescence data from QuantaSoft.
Click here to access the data

Figure 2. Exporting plate metadata from QuantaSoft.
Click here to access the data

Figure 3. Automated ddpcr droplet gating results for raw assay output for the cohort of 32 CRC patient samples. The numbers show the calculated BRAF-V600 mutation frequency, defined as the ratio of FAM-positive droplets to the sum of FAM-positive and double-positive droplets. Background colours: green = sample is classified as wild-type, purple = sample is classified as mutant, grey = failed ddPCR run.
Click here to access the data

Figure 4. Comparison of the mutation frequency in each patient sample as calculated automatically by ddpcr vs that determined independently manually by an expert technician. The grey line represents the $y = x$ line.
Click here to access the data

Table 1. Comparison of the mutation frequency in each patient sample as calculated automatically by ddpcr vs that determined independently manually by an expert technician.
Click here to access the data

References

Open Peer Review

Current Peer Review Status: ✔️ ✔️

Version 1

Reviewer Report 14 September 2016

https://doi.org/10.5256/f1000research.9706.r15716

© 2016 Hazlitt S et al. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Stephanie L. Hazlitt
Ministry of Environment - Province of British Columbia, Victoria, BC, Canada

Andy Teucher
Ministry of Environment - Province of British Columbia, Victoria, BC, Canada

The software tool article ‘ddpcr: an R package and web application for analysis of droplet digital PCR data’ is well-written and includes sufficient detail for the reader to assess the tool's construction, implementation and outputs. The ddpcr R package (v1.5) functions well and includes clearly-written, detailed vignettes to support the user and understand what is happening 'under the hood'. In addition, the source code is openly available on GitHub, which allows advanced R users to investigate details of the implementation. In general, we recommended more explicit links in the paper to supporting vignettes and the package README.

In addition to ddpcr being a useful tool for standard data science steps (get data, visualize data, analyze data), it also provides the ddPCR community with a well-documented, new analytical methodology for ddpcr clustering and hence opportunities for assessing robustness around clustering techniques in general in this field. This contribution could be made more clear in the Introduction and/or Motivation sections.

Figure 5--referenced multiple times in the Introduction--could be moved alongside the text for the reader.

The Methods structure could be arranged to follow standard data science steps and Figure 3: (1) get data [raw, export, csv] (2) import data [new_plate()] (3) analyze data [analyze()]. There is no section on (4) visualize data [plot()], and the likely user cycle of analyze-visualize-change analysis parameters-analyze-visualize. With clustering, I am guessing this will be a common and important set of repeat steps. I would end the section with the example workflow code to reinforce the steps.

The Analysis sub-section might benefit from some supporting references for the selected analyses (e.g. kernel density estimation, Gaussian mixture models), url links to the ddpcr::analysis vignette in GitHub, and if the paper word-limit allowed, a bit more detail in the paper itself, especially for the 'assign droplets into clusters' step.
The examples show rain being identified in the FAM (empty vs filled; vertical direction), but not between FAM+ and FAM+HEX+ (mutant vs wildtype; horizontal direction). Is this because the algorithm doesn't allow for this, or because all of the droplets in that range could be classified as FAM+ or FAM+HEX+?

Also in the Analysis sub-section, a description of the next_step() function would be useful for those who want to understand each step in the process. A potential enhancement to the package would be a plot function/method for each step that visually displays the results of each step in the analysis.

It might be useful to synchronize the language used to describe the clusters with the attribute language in output objects in the package. For example, in the paper and vignettes clusters are "double positive, FAM+, HEX+, or double negative". In clusters() the clusters are POSITIVE NEGATIVE RAIN EMPTY etc. The plate_data function outputs clusters as numbers 1 through 7. The plate_meta function outputs clusters as mutant_num and wildtype_num. With a bit of sleuthing it is clear you can define the resulting cluster names with new_plate(), which is useful but may not be intuitive for many users.

Other minor suggestions for potential enhancements after a test drive of ddpcr:
- Plotting:
  - an addition of a Legend for the plot() would be a very useful
  - show_grid_labels = TRUE as default?
  - visually distinguish rain from empty? Figure 5 in the paper shows this, but the plot generated by running the code in the 'Workflow' section does not.
- The plate_meta data.frame in a ddpcr_plate object could be a tibble for better printing
- The plate_meta data.frame (after analysis) could also display the rain count

Competing Interests: No competing interests were disclosed.

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.
Figure 3 should be moved alongside the text in the paragraph "Analysis algorithm".

Figure 5 is a powerful demonstration of the rationale for the library's creation and could perhaps stand to be moved up in the body of the paper. Standard practice is to provide a reason for the user to care about a method (e.g. figure 5), then describe the nuts and bolts of the implementation. There is a reason that this is standard practice.

Time permitting, this reviewer has sought and obtained some much more challenging samples. In a revised or separate publication, it would be of interest to compare the software's performance on these ddPCR runs (where 1 in 10000 cells carries a mutation, but that 1 in 10000 is verified by multiple orthogonal sequencing methods) and determine whether the algorithmic flexibility of the ddpcr package would allow users to automate what is currently a highly technical process for detecting rare mutations. Unfortunately, the manufacturer has been unhelpful in converting this data to a usable format within the timeline of the review, so I cannot in good conscience delay it further.

Outside of the scope of this paper, but relevant to the introduction, reproducibility and comparability would be much improved if the raw .QLP files could be parsed by R itself, making exchange and sharing of data far easier. Inquiries placed to Bio-Rad regarding the file format and the QuantaSoft package went unanswered, suggesting that (as with Illumina and their .IDAT format) the eventual solution will be to reverse engineer the format and brute-force the problem. It is unfortunate that some scientific instrument vendors appear to value imaginary profits over actual scientific merit, but such is life. Reverse engineering the file format is not a reasonable request here, but represents a future direction to further improve reproducibility of this important analytical technique. Existing tools rely upon ddPCR data provided in .CSV format; the first to elide this requirement (and that of a $5000 software package merely to review output) will be a notable step towards transparency for an increasingly powerful genotyping technique.

In conclusion, the ddpcr implementation and its extensive documentation (here in this paper and in the copious examples provided with the package) represent a solid foundation for further methodological improvements to an important assay platform.

**Competing Interests:** No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 19 Aug 2016

Dean Attali, University of British Columbia, Vancouver, Canada

Thank you for the thorough and prompt review. I will address the comments once I get a review from the second reviewer (it's taking longer than expected to find someone!)

**Competing Interests:** No competing interests were disclosed.
Stefan Rödiger, Brandenburg University of Technology Cottbus - Senftenberg, Germany

The presented work by Attali et al. is an interesting contribution to the growing knowledge about dPCR and the analysis thereof. The authors state that the “…clinical adoption has been slowed in part by the lack of software tools available for analyzing dPCR data.”. Actually, the environment of analysis of dPCR data is much larger than described. For the sake of completeness we would like to raise the awareness to the dpcR package, which is the first R package (available from CRAN since September 2013) devoted to analysis of dPCR data. The functionality of the dpcR packages is targeted at users of droplet dPCR and chamber dPCR. Similarly, to ddpcr offers dpcR a shiny GUI, visualization tools (ggplot2, ...), simulation tools, import functionality and a collection of data. For further reading please refer to following publications:


In addition, there are further studies using the R statistical computing environment for dPCR data, which might be of interest for the presented study:


We hope this information is a valuable information for the work presented by Attali et al..

Respectfully,

Stefan Rödiger on behalf of all package authors
**Competing Interests:** There are no competing interests to disclose

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com