Automatic analysis of dual-channel Droplet Digital PCR experiments

by

Dean Attali

B.CS., University of Waterloo, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Bioinformatics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

April 2016

Abstract

The ability to quantify the amount of DNA in a sample is an essential technique in biology and is used in many fields of research. Droplet digital polymerase chain reaction (ddPCR) is an advanced technology developed for this purpose that enables more accurate and sensitive quantification than traditional real-time PCR. In ddPCR, nucleic acid (e.g., genomic DNA) within a sample is partitioned into thousands of droplets, along with the reagents needed to amplify and detect one or more DNA target sequences. After amplification takes place in all droplets, each droplet is individually read by a two-colour fluorescence detection system to determine whether or not it contains the target sequence. ddPCR experiments utilizing both fluorescence wavelengths are termed dual-channel, while simpler experiments can make use of only one fluorescence wavelength and are thus classified as single-channel. Droplets containing amplified product exhibit high fluorescence and are said to be positive, while those without product show little or no fluorescence and are considered negative. Using this binary, or digital, classification of droplets, the number of positive and negative droplets can be counted to allow for an absolute quantification of template abundance in the starting sample.

ddPCR instruments are now available commercially and their use is growing. But, there are a very limited number of tools available for downstream data analysis. The key step in ddPCR data analysis is droplet gating: using the end-point fluorescence data to gate, or classify, droplets as either positive or negative for a given template. The proprietary software provided by BioRad Inc., a ddPCR instrument manufacturer, is currently the only program available to automatically analyze dual-channel ddPCR data. However, because this analysis tool often produces poor results, many ddPCR users resort to time-consuming and subjective manual data analyses, emphasizing the clear need for new ddPCR analysis tools.

In this thesis, I devise an algorithm for automatic analysis of dual-channel ddPCR data that can objectively and reproducibly perform droplet gating. The proposed analysis method has been implemented in an R package and is also available as a web application online for easy and open access to any ddPCR user.

Preface

All work in this thesis is my own work, unless otherwise noted. Ryan Brinkman suggested using the local minima in a kernel density estimate in the algorithm described in Section 2.4.4.2. Jenny Bryan gave continuous input along the way and suggested creating a package and a Shiny web application to interact with the R code. She also did most of the experimentation with the 'flexmix' package that is shown in Appendix E. Charles Haynes and Roza Bidshahri provided insights into what features the algorithm and web application should support. All ddPCR experiments were run by Roza Bidshahri.

All chapters in this thesis were written by me. Jenny Bryan, Ryan Brinkman, and Charles Haynes have all provided valuable feedback and suggestions on improving the text of this thesis.

The work in Chapters 2 and 3 was reported in part in a recent co-authored publication in *Journal of Molecular Diagnostics* [1] and in a first-authored paper recently submitted to the peer-reviewed journal *Bioinformatics*.

Figure 1 was obtained from Wikipedia under the CC BY-SA 3.0 license, which permits free copying of the work if it was attributed. The exact details of the license can be found at *https://creativecommons.org/licenses/by-sa/3.0/deed.en*. The figure was created by Wikipedia user Enzoklop under the title "Schematic drawing of the PCR cycle" and is available at *https://commons.wikimedia.org/wiki/File:Polymerase_chain_reaction.svg*. Figure 2 was obtained with permission from Hidson et al. [2] without modification. The original article is available at *http://pubs.acs.org/doi/full/10.1021/ac202028g*. Figure 27 was obtained from Bidshahri et al. [1] with permission from Copyright Clearance Center's RightsLink service (order number 3824060205659). The wording in the text labels on the original image has been modified. All other images are my own work.

The tissue samples used in the analysis in Chapter 4 were obtained with approval from the University of British Columbia Clinical Ethics Research Board, certificate number H14-00577.

Table of Contents

Abstract ii
Preface iii
Table of Contents iv
List of Tables viii
List of Figures ix
List of Abbreviations xiii
Acknowledgements xiv
Dedication xv
Chapter 1: Introduction 1
1.1 Polymerase chain reaction 1
1.2 Real-time polymerase chain reaction 3
1.3 Digital polymerase chain reaction 3
1.3.1 Droplet digital polymerase chain reaction 4
1.3.1.1 Protocol 4
1.3.1.2 Analysis of ddPCR data 5
1.3.1.3 Factors to consider when gating 7
1.3.1.4 Current gating methods 8
1.3.1.4.1 Manual gating 8
1.3.1.4.2 Automatic gating 9
1.4 Project overview 11
Chapter 2: Algorithm for automated ddPCR analysis 12
2.1 Preconditions 12
2.2 Terminology and notation used 13
2.3 Automated analysis of any ddPCR data 14
2.3.1 Step 1: Identify failed wells 14
2.3.1.1 First failure condition 14
2.3.1.2 Second failure condition 15
2.3.1.3 Third failure condition 17
2.3.1.4 Fourth failure condition 17

2.3.2 Step 2: Identify outlier droplets 18
2.3.3 Step 3: Identify empty droplets 21
2.4 Automated analysis of PN/PP assays 22
2.4.1 Step 1: Identify failed experiments 23
2.4.2 Step 2: Identify outlier droplets 24
2.4.3 Step 3: Identify empty droplets 24
2.4.4 Step 4: Gate droplets 24
2.4.4.1 Identify rain droplets 25
2.4.4.2 Assign PN and PP droplets 26
2.4.4.3 Compute PN frequencies and classify wells as PN-positive or PN-negative 31
2.4.4.4 Revisit gating of PN-negative wells 33
Chapter 3: Software tool – *ddpcr* R package 34
3.1 Introduction 34
3.2 Usage 34
3.2.1 Plate objects and plate types 34
3.2.2 Data import 35
3.2.3 Data exploration 37
3.2.4 Data analysis 38
3.3 Implementation technical details 40
3.3.1 A plate object is a list 40
3.3.2 Using S3 for plate objects to override base generic functions 41
3.3.3 Using S3 for plate objects to support inheritance 42
3.3.4 Analysis of different built-in plate types 42
3.3.5 Plate parameters 43
3.3.6 How the analysis steps work 44
3.3.7 Creating new plate types 45
3.3.7.1 Define the parent plate type 45
3.3.7.2 Define the plate type parameters 45
3.3.7.3 Define the potential droplet clusters 45
3.3.7.4 Define the analysis steps 46
3.3.7.5 Changing the algorithm of an existing step 46

3.3.7.6 Define a plot function 46
3.4 The *ddpcr* web application 47
3.4.1 Introduction 47
3.4.2 Main features 47
3.4.2.1 Dataset tab 47
3.4.2.2 Settings tab 48
3.4.2.3 Analyze tab 48
3.4.2.4 Results tab 48
3.4.3 Availability 49
Chapter 4: Case study: *BRAF*-V600 mutations 50
4.1 Introduction 50
4.2 Mutated *BRAF* in cancer 50
4.3 The wild-type negative ddPCR assay for detecting *BRAF* V600 mutations 51
4.4 Data 53
4.5 Results 54
Chapter 5: Conclusion 59
Bibliography 60
Appendices 65
Appendix A Datasets used for testing 65
Appendix B Number of droplets generated per well across different plates 66
Appendix C Applying outlier detection algorithm to different datasets 70
Appendix D Empty droplet detection in different wells 74
Appendix E Other attempted clustering algorithms 77
Appendix F Extended examples of using density kernel estimation for droplet gating 84
F1. Wells with a large number of single-positive droplets, without optimizing density bandwidth 84
F2. Wells with a small number of singly-positive droplets, without optimizing density bandwidth 86
F3. Well without any singly-positive droplets 87
F4. Using too high of a smoothing bandwidth for KDE during droplet gating 88
F5. Using too low of a smoothing bandwidth for KDE during droplet gating 88

F6. Finding the optimal smoothing bandwidth by iteratively increasing its multiplier .. 88
F7. Altering the original calculated gate to a Gaussian gate 90
Appendix G Screenshots from QuantaSoft and the *ddpcr* web tool 93
Appendix H Extending *ddPCR* by adding a new plate type 98
Appendix I Supplementary data for the *BRAF*-V600 assay case study 102

List of Tables

Table 1 – Comparing *BRAF* status and mutation frequency of patient samples across three methods: staining by a pathologist, manual analysis by Roza Bidshahri, and automatic analysis with ddpcr 102
Table 2 – Full results from running *ddpcr* analysis on the case study data 107

List of Figures

Figure 1 – Overview of the main steps of PCR. 2
Figure 2 – Overview of the main steps of ddPCR. 5
Figure 3 – Fluorescence readings from a typical two-channel ddPCR assay. 7
Figure 4 – Comparison of scatterplots for replicate ddPCR experiments conducted in two different wells of a 96-well PCR plate. 8
Figure 5 – Automatic droplet gating of data from a ddPCR experiment amplifying two distincy templates using QuantaSoft. 9
Figure 6 – Automatic droplet gating of a full ddPCR plate using QuantaSoft. 10
Figure 7 – Examples of PN/PP and non-PN/PP assays. 13
Figure 8 – Diagram showing how ddPCR droplet data can be represented as a 2-column matrix per well. 14
Figure 9 – Example of first failure condition: removing wells with insufficient number of read droplets. 16
Figure 10 – Example of second failure condition: checking for proper cluster segregation. 17
Figure 11 – Example of fourth failure condition: checking for sufficiently large non-empty cluster. 18
Figure 12 – Running the failure detection step on an entire plate. 19
Figure 13 – ddPCR data is often extremely skewed. 20
Figure 14 – Outlier detection algorithm. 21
Figure 15 – Identifying empty droplets in a ddPCR experiment. 22
Figure 16 – Identifying empty droplets in PN/PP assays. 24
Figure 17 – Proper droplet gating of a ddPCR experiment. 25
Figure 18 – Identifying rain and filled droplets. 26
Figure 19 – Gating PN (FAM^+) and PP (FAM^+HEX^+) droplets using a kernel density estimator. 28
Figure 20 – Gating droplets in an experiment with a few outlier droplets. 30
Figure 21 – Altering the original calculated gate to a Gaussian gate. 31
Figure 22 – Code to initialize a ddPCR plate using *ddpcr* 36
Figure 23 – Output of a plate summary in *ddpcr* 37

Figure 24 – Exploring basic information about a plate in *ddpcr* 38
Figure 25 – Plotting a ddPCR dataset before and after analysis. 39
Figure 26 – Code to retrieve and set parameters of a ddPCR plate with *ddpcr* 43
Figure 27 – Wild-type negative assay for detecting *BRAF*-V600 mutations 52
Figure 28 – Results of a few wells from the analysis of CRC patient samples. 55
Figure 29 – Comparison of the mutation frequency in each well as calculated automatically by *ddpcr* vs manually by setting custom gates. 56
Figure 30 – Bland-Altman plot showing the difference vs average of the reported MF values by the manual and automatic analysis approaches. 57
Figure 31 – Droplets per well in CRC-41 dataset 66
Figure 32 – Droplets per well in CRC-REPEAT dataset 67
Figure 33 – Droplets per well in PLASMID dataset 67
Figure 34 – Droplets per well in BRAFV600K dataset 68
Figure 35 – Droplets per well in YUMAC dataset 68
Figure 36 – Droplets per well in THYROID dataset 69
Figure 37 – Outlier decetion CRC-41 dataset 70
Figure 38 – Outlier detection in CRC-REPEAT dataset 71
Figure 39 – Outlier detection in PLASMID dataset 72
Figure 40 – Outlier detection in BRAFV600K dataset 73
Figure 41 – Outlier detection in THYROID dataset 73
Figure 42 – Empty droplet identification in CRC-41 dataset 74
Figure 43 – Empty droplet identification in CRC-REPEAT dataset 74
Figure 44 – Empty droplet identification in BRAFV600K dataset 75
Figure 45 – Empty droplet identification in BRAFV600K dataset 75
Figure 46 – Empty droplet identification in YUMAC dataset 76
Figure 47 – Empty droplet identification in THYROID dataset 76
Figure 48 – Raw data in well B11 of the THYROID dataset 77
Figure 49 – Running K-means with 3 centers on the full data of well B11 of the THYROID dataset. The three centers are marked by the blue stars. 78
Figure 50 – Running GMM on the full data of well B11 of the THYROID dataset. 78

Figure 51 – Running K-means on well B11 of the THYROID dataset after removing empty droplets. 79
Figure 52 – Running GMM on well B11 of the THYROID dataset after removing empty droplets. 80
Figure 53 – Running hierarchical divisive clustering on well B11 of the THYROID dataset after removing empty droplets. 80
Figure 54 – Modeling the non-empty droplets in well B11 of the THYROID dataset as two droplet populations. 81
Figure 55 – Running K-means on well B11 of the THYROID dataset after removing empty and rain droplets. 82
Figure 56 – Running K-means on well A02 of the PLASMID dataset after removing empty and rain droplets. 82
Figure 57 – Running GMM on well A02 of the PLASMID dataset after removing empty and rain droplets. 83
Figure 58 – Identifying positive vs negative droplets in well A09 of dataset CRC-41 84
Figure 59 – Identifying positive vs negative droplets in well C08 of dataset CRC-41 84
Figure 60 – Identifying positive vs negative droplets in well E04 of dataset CRC-REPEAT 85
Figure 61 – Identifying positive vs negative droplets in well E05 of dataset CRC-REPEAT 85
Figure 62 – Identifying positive vs negative droplets in well B02 of dataset PLASMID 85
Figure 63 – Identifying positive vs negative droplets in well C03 of dataset PLASMID 85
Figure 64 – Identifying positive vs negative droplets in well B12 of dataset BRAFV600K 86
Figure 65 – Identifying positive vs negative droplets in well C03 of dataset PLASMID 86
Figure 66 – Identifying positive vs negative droplets in well C04 of dataset PLASMID 86
Figure 67 – Identifying positive vs negative droplets in well G02 of dataset CRC-REP 87
Figure 68 – Gating positive and negative droplets using a kernel density estimator in a well without negative droplets. 87
Figure 69 – Using too high of a smoothing bandwidth for KDE during droplet gating. 88
Figure 70 – Using too low of a smoothing bandwidth for KDE during droplet gating. 88
Figure 71 – Finding the optimal smoothing bandwidth for well E05 of the CRC-REP dataset... 89
Figure 72 – Finding the optimal smoothing bandwidth for well C03 of the PLASMID dataset.. 89
Figure 73 – Finding the optimal smoothing bandwidth for well H04 of the PLASMID dataset. 90

Figure 74 – Using a Gaussian gate for well A11 in the CRC-41 dataset..... 90
Figure 75 – Using a Gaussian gate for well F08 in the CRC-41 dataset. 91
Figure 76 – Using a Gaussian gate for well H06 in the CRC-41 dataset..... 91
Figure 77 – Using a Gaussian gate for well A02 in the PLASMID dataset. 91
Figure 78 – Using a Gaussian gate for well A03 in the PLASMID dataset. 92
Figure 79 – Exporting droplet fluorescence data from QuantaSoft..... 93
Figure 80 – Exporting plate metadata from QuantaSoft..... 93
Figure 81 – ddpcr web application: Dataset tab..... 94
Figure 82 – ddpcr web application: Settings tab..... 95
Figure 83 – ddpcr web application: Analyze tab 96
Figure 84 – ddpcr web application: Results tab..... 97
Figure 85 – Raw dataset used in the *BRAF*-V600 case study..... 104
Figure 86 – Fully analyzed dataset in the case study..... 105
Figure 87 – Automatic droplet gating by QuantaSoft on the case study data..... 106

List of Abbreviations

BRAF	v-Raf murine sarcoma viral oncogene homolog B protein kinase
BRAF	gene encoding the BRAF protein
CPD	Copies (of template) per droplet
CRC	Colorectal cancer
CSV	Comma-separated values
dPCR	Digital polymerase chain reaction
ddPCR	Droplet digital polymerase chain reaction
EGF	Epidermal growth factor
EGFR	Epidermal growth factor receptor
FFPE	Formalin-fixed, paraffin-embedded
IQR	Inter-quartile range
KDE	Kernel density estimate
MT	Mutant
NTC	No template control
PCR	Polymerase chain reaction
PN	Positive-negative (singly-positive)
PNPP	Positive-negative/positive-positive
PP	Positive-positive (double-positive)
PTC	Papillary thyroid cancer
qPCR	Quantitative (real-time) polymerase chain reaction
SD	Standard deviation
WT	Wild-type

Acknowledgements

First and foremost, I would like to thank Jenny Bryan. You have been a perfect supervisor and mentor, and much more than that. I especially thank you for supporting and encouraging me in everything I did and helping me start building a name for myself. Your continuous stream of professional and social opportunities was not taken for granted.

I would also like to thank Charles Haynes and Ryan Brinkman, my supervisory committee members. Thank you for your helpful insights and time, for your words of encouragement throughout, and for always being flexible and prompt enough to accommodate my irregular schedule.

Roza Bidshahri had a big impact in my project, and I thank her for all her help and dedication despite having her own thesis and baby to worry about.

Many thanks are owed to Paul Pavlidis and his lab members, most notably Ogan Mancarci. Not only did you provide me with resources and a place to work, but you also made me feel included in your events and helped keep me sane during long nights at the office. I've had many great experiences with the Pavlidis lab, and I thank you all for treating me with so much warmth.

I would like to acknowledge CIHR for funding me for two years and providing me with the great opportunity of experiencing Vancouver while attending a great university.

Finally, my deepest gratitude is extended to my amazing family and wonderful partner, Kesh. You have been there for me every day throughout this journey, and have kept my spirits high through all my personal difficulties. I am truly grateful to have had your endless and selfless support. Thank you.

To my family.

My two parents, two brothers, two sisters,
and two dogs.

I cannot imagine my life without any of you.

Chapter 1: Introduction

The invention of the polymerase chain reaction (PCR) technique by Kary Mullis in the 1980's has revolutionized the field of molecular biology by allowing scientists to selectively amplify specific sequences of DNA to enable their detection in biological studies [3]. A related technique, termed real-time (or quantitative) PCR (qPCR), has since been developed to quantify the amount of each sequence (template) amplified in a sample [4]. The ability to quantify nucleic acid has proven to be applicable in many areas, and is one of the most widely used techniques in medical and biological research laboratories today. However, to provide an absolute measurement of template abundance, qPCR data for amplification of template(s) within a sample must be compared to that for a reference material [5], which can be problematic, particularly when applied to clinical specimens. This limitation is eliminated in droplet digital PCR (ddPCR), which can provide absolute quantification of nucleic acid without the need for comparison to a standard [2]. In addition, when compared to qPCR, ddPCR is considerably more sensitive and, due to the dilution of both templates and PCR-inhibiting analytes in the sample across a large population of droplets, less prone to amplification failures and artifacts. Due to its increased sensitivity and accuracy, the use of ddPCR is becoming increasingly popular. Regrettably, however, there are at present a limited number of tools available for working with ddPCR data.

This thesis outlines the development and application of a set of statistical methods for the automatic analysis of two-channel ddPCR data. The methods are implemented as a software tool that is available in the R programming language [6] as well as on the web. The tool also allows the exploration and visualization of ddPCR data. The algorithm developed is validated through a case study to demonstrate its accuracy and clinical application.

1.1 Polymerase chain reaction

PCR is a protocol for *in vitro* amplification of DNA that can generate millions of copies of a specific DNA sequence in a matter of minutes. PCR uses primers – short nucleotide sequences complementary to a region of interest – to ensure only the target DNA region gets amplified. A

common laboratory and clinical technique, it has found widespread utility in a variety of fields and is being used, for example, in DNA fingerprinting [7], paternity testing [8], DNA sequencing [9], and pathogen detection [10]. Traditional PCR can be operated in many different modalities; some simply optimize standard protocols (e.g., Touchdown PCR [11], Hot-Start PCR [12]) for specific sample types, while others represent more novel approaches (e.g., Assembly PCR [13]). Two modes of PCR that will be discussed in the following sections are quantitative PCR and digital PCR.

PCR consists of a repeated cycle of three processing steps: template denaturation, primer annealing, and primer extension. Denaturation involves heating the reaction to a high temperature so that the two DNA strands encoding the template sequence of interest separate from each other. In the annealing step, the temperature is lowered, often to about 60°C, to allow the forward and reverse primers to anneal to complementary sequences flanking the template to be amplified. In the extension phase, a DNA polymerase then extends each primer by adding nucleotides complementary to the template to create double-stranded copies of the template, also known as amplicons. These three steps of thermal cycling are repeated dozens of times, with each cycle resulting in an exponential growth of amplicons [14]. Figure 1 below shows a schematic of one cycle of the standard PCR process.

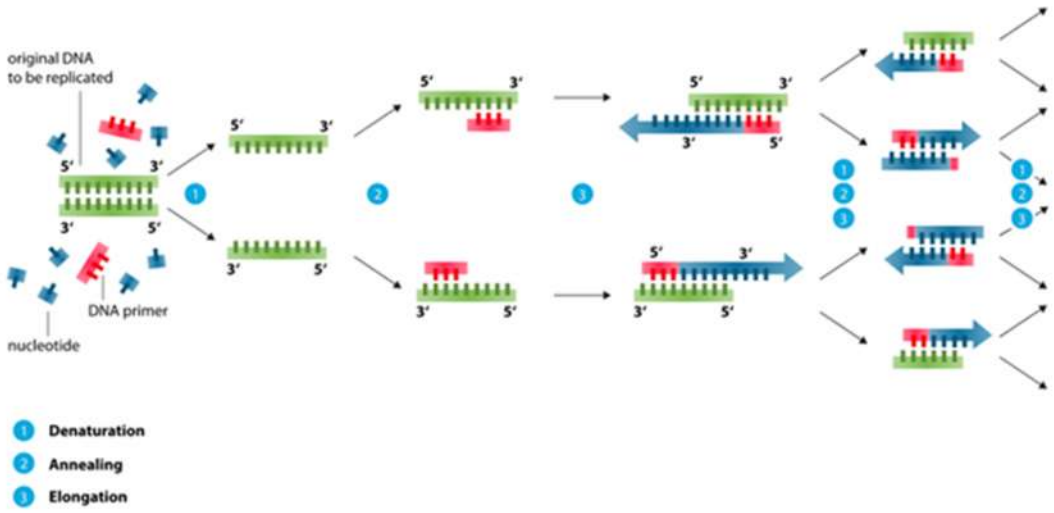

Figure 1 – Overview of the main steps of PCR. (1) During denaturation, the temperature is raised and the two strands of every copy of template come apart. (2) In the annealing phase, the primers anneal to opposing to single-strands of template, and (3) during extension, DNA polymerase synthesizes new copies (amplicons) by adding complementary nucleotides to extend the primers across the template.

1.2 Real-time polymerase chain reaction

Real-time (or quantitative) PCR (qPCR) adds to standard PCR the ability to quantify the initial amount (copies) of template in a sample. This ability has proven useful in a broad range of applications, including gene expression analysis [15], detection of genetically modified organisms in food [16], and cancer phenotyping [17]. While there are other methods for achieving DNA quantification, qPCR is the most commonly used [18].

The qPCR protocol is similar to traditional PCR, with one key addition: a single-stranded dual-labeled hydrolysis probe complementary to a short sequence within the template. One end of the probe is labeled with a fluorescent reporter dye (e.g. fluorescein (FAM)), and the other with a corresponding fluorescence quencher. PCR is conducted in a specialized thermal cycler equipped with fluorescence detectors that can monitor the relative fluorescence (intensity) units (RFU) produced from probe hydrolysis during the amplification reaction, where in theory one reporter dye is released per amplicons created. The amount of fluorescence released during amplification is therefore directly proportional to the amount of template that has been amplified, and this fluorescence is monitored in real time as the thermal cycles progress. Due to the close proximity of the quencher to the dye in the intact probe, the background RFU measured at the start of the reaction is generally undetectable. The quantitation cycle (C_q) is defined as the number of cycles required for the fluorescence signal to be detected. Samples with a higher number of initial copies of template will have a lower C_q value since they will produce more fluorescence per cycle and will be detected at an earlier cycle. Absolute quantitation of starting template concentration can then be achieved by comparing the C_q value for the sample to a standard curve created by conducting the same qPCR experiment on a serial dilution of the template from a stock of known template concentration. The method therefore is not direct, as it requires comparison to a reference and prior knowledge of the template concentration in that reference [4].

1.3 Digital polymerase chain reaction

A technology that builds on qPCR, digital PCR (dPCR) can quantify DNA absolutely with a precision and sensitivity superior to other PCR modalities. While the fundamental principle

behind dPCR was first described in 1992 [19], it was not until 2006 that the technology became commercially available. Since dPCR is a relatively recent development, it is not as widely used as qPCR, though it is gaining popularity. Its increased sensitivity over qPCR makes dPCR a better option in certain applications, such as rare mutation detection [20] and determination of copy number variation [2]. There are different implementations of digital PCR provided by different companies, and this thesis focuses on droplet digital PCR.

In dPCR, copies of template within DNA isolated from a sample are partitioned between a large number of isolated compartments such that each compartment generally contains zero or one copies of the target template. The template in each partition is subjected to PCR amplification in the presence of a Taqman™ dual-labeled probe specific to the template. Partitions containing the target sequence are detected by the end-point fluorescence generated by probe hydrolysis and scored as positive, while partitions without the template do not generate detectable fluorescence and are scored as negative. Total counts of positive partitions can be combined with Poisson statistics to estimate the initial concentration of target template without the need for an external reference [2].

1.3.1 Droplet digital polymerase chain reaction

Droplet digital PCR (ddPCR) is an implementation of dPCR created by Bio-Rad Laboratories Inc. (Hercules, CA) that uses nanoliter-sized water droplets in a water-in-oil emulsion as the reaction compartments. Despite being available for less than five years [2], ddPCR has become the most widely used dPCR platform due to its robust operation, low cost and low input material requirement [21].

1.3.1.1 Protocol

In a ddPCR experiment, an aqueous reaction sample is first prepared by adding 5 to 100 ng of DBSA isolate to be analyzed to the PCR reagents required for target amplification (ddPCR-specific aqueous buffer, nucleotides, forward and reverse primers, dual-labeled hydrolysis probes). The sample is loaded onto a proprietary droplet generator, along with the required oil phase, to partition copies of the target template into *ca.* 20,000 nanoliter-sized droplets using microfluidics. Following the partitioning step, each droplet typically contains either zero or one

copies of the target, though some will inevitable contain more. The target distribution into droplets is assumed to roughly follow a Poisson distribution. The stable droplet-bearing water-in-oil emulsion is then transferred into one reaction well of a 96-well plate (the droplets of each sample are in one well) for PCR thermocycling to independently amplify target templates partitioned among the droplets. The droplets are passed in a single file through a two-colour fluorescent detector that measures and records for each droplet the end-point fluorescence intensity in two optical channels, which is tuned to the excitation/emission wavelengths of a reporter dye (often FAM or HEX) employed on the hydrolysis probes used. Figure 2 below shows an overview of the ddPCR protocol.

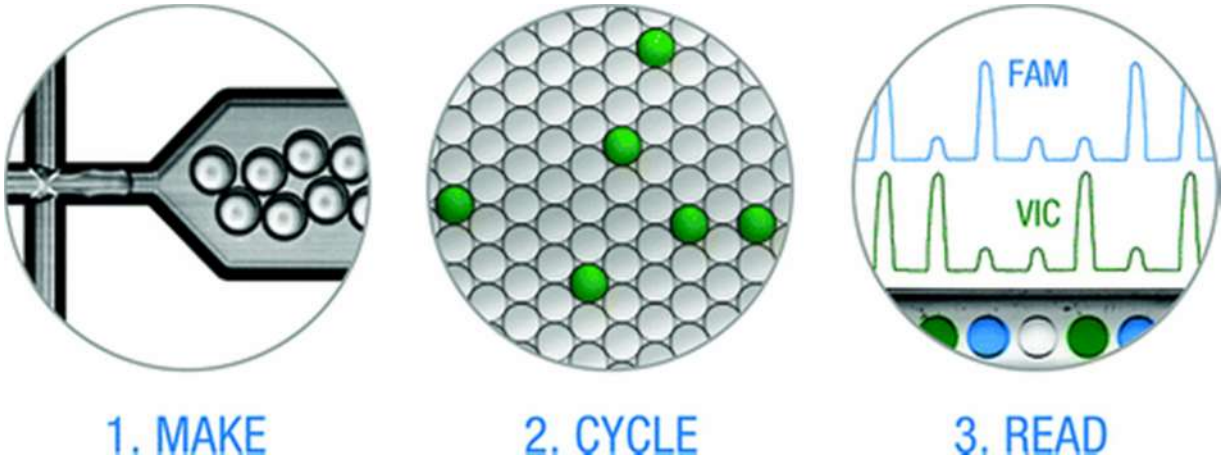

Figure 2 – Overview of the main steps of ddPCR. (1) Sample DNA, along with ddPCR mix, is partitioned into *ca.* 20,000 droplets. (2) Template amplification takes place in all droplets in parallel. (3). The end-point fluorescence in each droplet is detected in two channels using the droplet reader. (Image obtained with permission from Hidson et al. **[2]**)

1.3.1.2 Analysis of ddPCR data

For each droplet, the pair of end-point fluorescence amplitude data recorded in the two optical channels is the main output from the experiment, and the complete data set for all read droplets gets recorded in QuantaSoft (Bio-Rad Inc.), a software tool packaged with the ddPCR system for data retrieval and analysis.

Analysis of this data typically begins with classifying each droplet as either PCR-positive or PCR-negative for each target template analyzed based on the fluorescent intensities recorded for that droplet.

If a droplet initially contains the target template (positive droplet), the template will get amplified and the fluorescent reporter dye on the template-specific probe will be released (and recorded in one of the optical channels) in proportion to the number of template copies produced. A weak background fluorescence signal in that channel will therefore be recorded for droplets without the target template (negative droplets). In this thesis, droplets negative for all target templates analyzed are called 'empty droplets', even though they may contain some partitioned DNA. A cut-off value is determined as a threshold to classify droplets; all droplets with an end-point fluorescence signal above the cut-off imposed in either optical channel are considered positive and the rest negative. The threshold value is also known as a "gate", and the process of choosing the cut-off value is termed "gating".

Since fluorescence is detected by the ddPCR droplet reader in two optical channels, gating can be performed in both channels. For example, if the detector in channel 1 is tuned to detect the FAM fluorophore, and that in channel 2 the HEX fluorophore, then gating in each of the channels will result in each droplet being classified as positive or negative for each channel, and consequently droplets will fall into one of four groups: empty (no template), FAM-positive (FAM^+), HEX-positive (HEX^+), or FAM^+HEX^+, as shown in the resulting scatterplot (Figure 3). In single-channel assays, where only a single fluorescent dye is used, gating is only required in one channel.

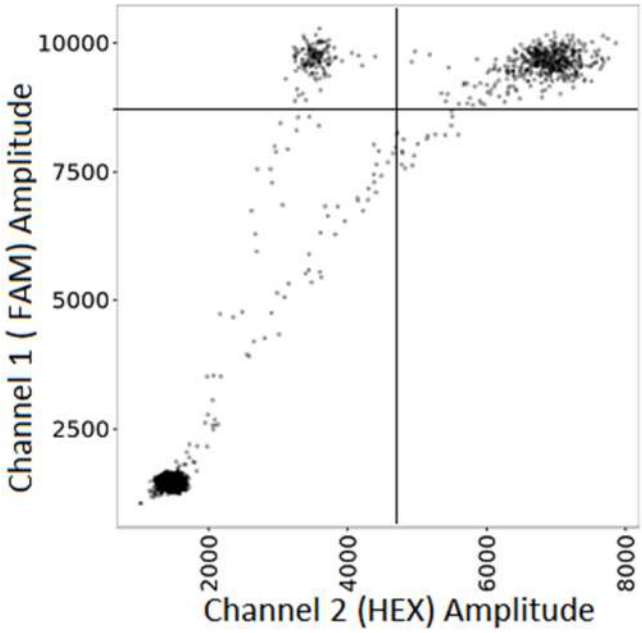

Figure 3 – Fluorescence readings from a typical two-channel ddPCR assay. After the droplet reader records the fluorescence intensity in both channels for each droplet, the droplets can be plotted in a scatterplot such as shown here. Gating can be performed in both channels (black lines), which will divide all the droplets into four groups: empty (bottom left), FAM-positive (top left), HEX-positive (bottom right), or double-positive (top right).

1.3.1.3 Factors to consider when gating

There are a few characteristics of the droplet fluorescence data to note when gating droplets, which can be seen in Figure 4:

- Negative droplets do not result in a reading of zero fluorescence. Instead, droplets without target emit some background fluorescence, due in part to non-specific scattering of light by the sample components and the microfluidic materials used, and by the fact that the quantum efficiency of the quenching agent on the probe is not sufficient to fully arrest fluorescence from the reporter dye when excited by the detector. This background signal can vary between wells on the same plate, as exemplified in Figure 4 which shows that the mean fluorescence amplitudes of the main cluster of the empty droplets vary from well to well.
- Similarly, as demonstrated in Figure 4, the threshold that separates positive from negative droplets can also vary between wells, so a single global gate for an entire plate of replicate or independent ddPCR experiments might not provide the best results.

- In some datasets, positive and negative droplets segregate into distinct dense clusters in the scatterplot. However, inefficient template amplification or probe hydrolysis in a droplet can result in recorded end-point fluorescent values that fall in between the two extremes. These droplets are termed "rain", and their ambiguous signal makes it less clear where gating should occur. Rain droplets appear in many ddPCR analyses, and are particularly prevalent in experiments conducted on DNA isolated from formalin-fixed paraffin-embedded (FFPE) solid tissues such as shown in Figures 3 and 4.

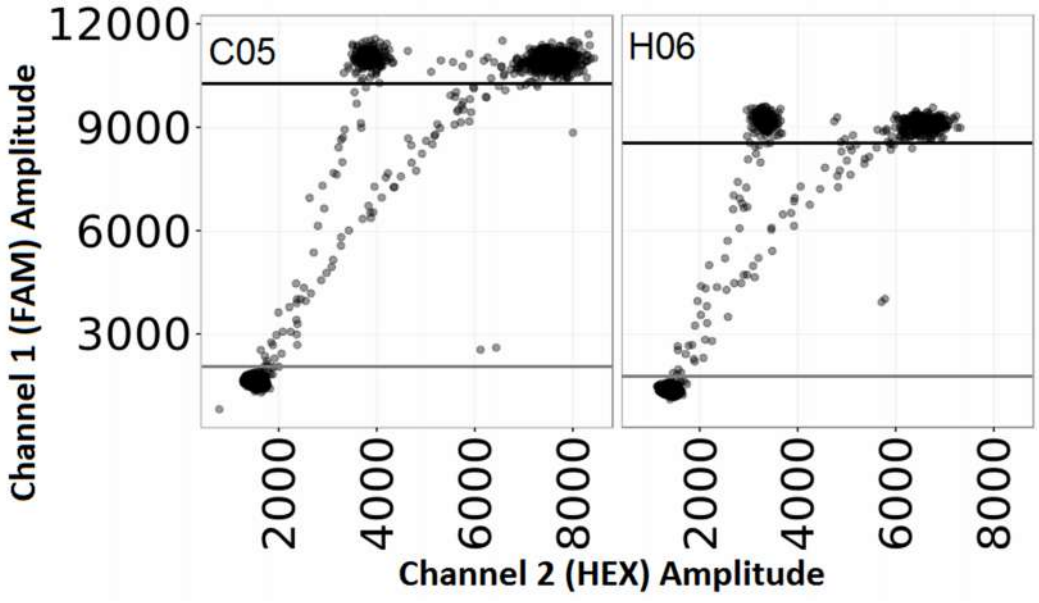

Figure 4 – Comparison of scatterplots for replicate ddPCR experiments conducted in two different wells of a 96-well PCR plate. The fluorescence values that separate positive from negative droplets (black and grey lines) vary between wells. In both channels, the distinction between positive and negative droplets in well C05 occurs at a higher amplitude than in well H06.

1.3.1.4 Current gating methods

Available algorithms for gating can be segregated into those performed manually or performed automatically.

1.3.1.4.1 Manual gating

The pair of end-point fluorescence values (one from each channel) recorded for each droplet are visualized as a two-dimensional scatterplot in which each axis represents the fluorescence

amplitudes recorded in one channel. Manual gating in each channel can then be performed by subjectively positioning the minimum fluorescence threshold (line) for positive droplets. QuantaSoft™ provides a user-controlled tool to select these thresholds [22]. Manual gating of ddPCR data has been employed by many ddPCR users and is currently the most common method of gating [23] [24] [25] [26] [27].

1.3.1.4.2 Automatic gating

Automated gating methods have also been developed with the goal of reducing the time and improving the objectivity and reproducibility of the analysis. QuantaSoft offers a proprietary automatic gating tool that, in principle, can be applied to perform droplet assignments in one or both channels [22]. Arguably the most advanced gating tool available, it performs well in gating an experiment in which only one positive droplet cluster (i.e., ddPCR reactions producing only one amplicon) is expected. It also can be applied to scatterplots for more complex ddPCR experiments (e.g., two distinct amplicons produced) when those plots exhibit very little to no rain. However, it often provides inaccurate gates when the data contains rain, as demonstrated in Figure 5 and Figure 6. As the details of the algorithm are not disclosed, it is not possible to reason about why the algorithm fails in these cases.

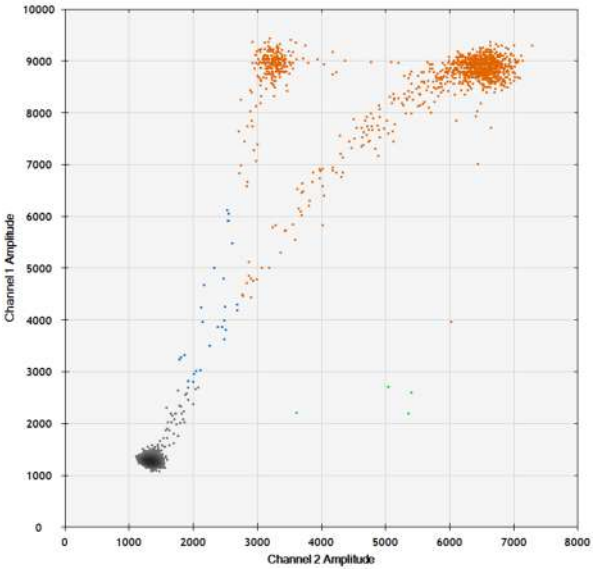

Figure 5 – Automatic droplet gating of data from a ddPCR experiment amplifying two distincy templates using QuantaSoft. QuantaSoft is not able to distinguish between the single-positive and double-positive droplets, grouping them all together (orange droplets).

Custom ad hoc scripts have therefore been developed to automatically gate specific types of ddPCR data by some ddPCR users [28]. As well, there have been two attempts to implement more general third-party tools for automated ddPCR droplet gating. In 2014, Jones et al. created a JavaScript program called 'definetherain' [29], and in 2015 Trypsteen et al. developed a script in R named 'ddpcRquant' [30]. Though more accurate droplet gating than QuantaSoft is claimed in both cases, both of these programs share two common drawbacks. First, both methods rely on the highly undesirable need for control samples to be included on the plate as standards, and then base droplet gating on the (problematic) assumption that the control is representative of all the wells. More importantly, both methods are only applicable to analysis of single-channel data, and therefore cannot be used for any two-channel ddPCR experiments.

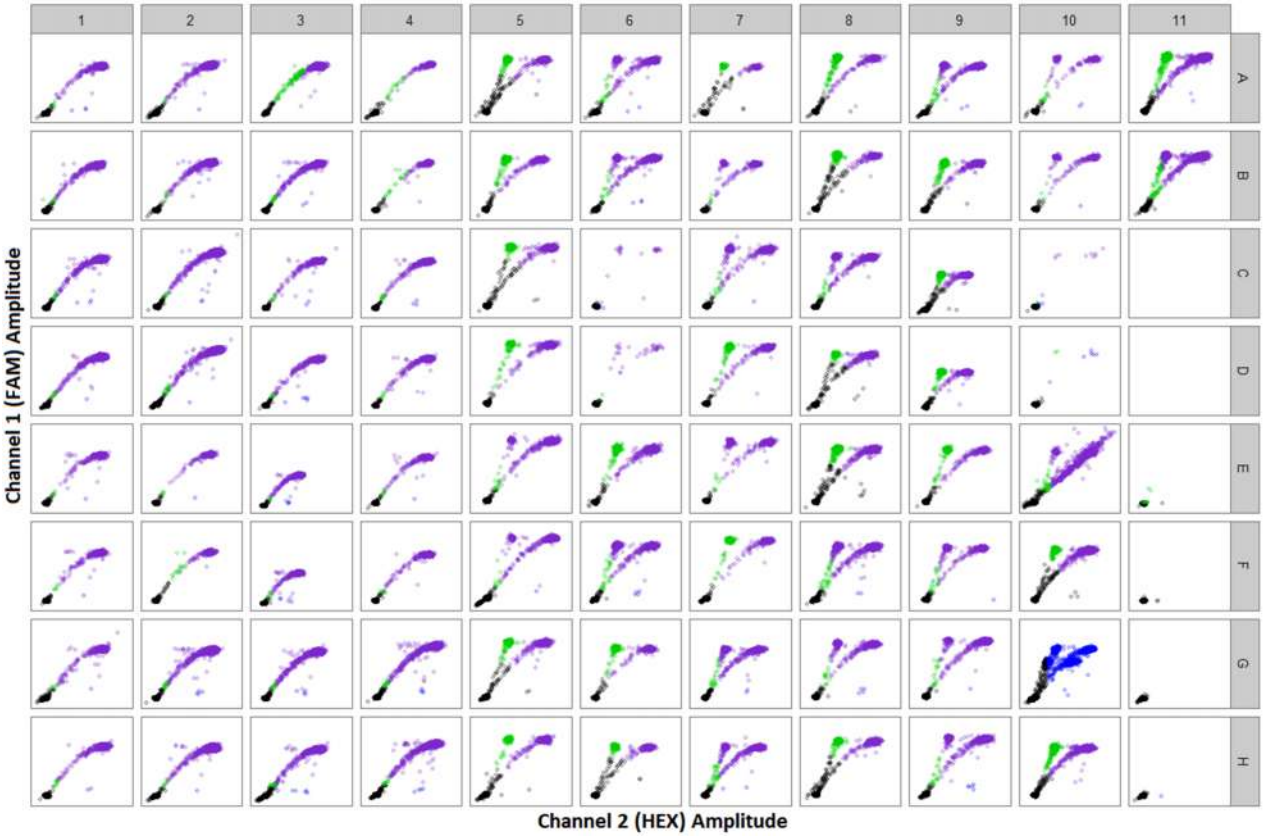

Figure 6 – Automatic droplet gating of a full ddPCR plate using QuantaSoft. The gating is visibly inaccurate in many samples where the single-positive and double-positive droplets were clustered together.

1.4 Project overview

Manual droplet gating of ddPCR data is time consuming and requires a knowledgeable human operator. It is therefore highly subjective and non-reproducible, as well as prone to human error. The need for automatic gating is clear, as Milbury et al. stated last year: "Objective automated gating of droplet event clusters is likely necessary for dPCR practitioners to take advantage of the full potential sensitivity of the technology for routine applications" [31]. Other than QuantaSoft, which has limited reliable applicability, the only other two options for automatic droplet gating operate on single-channel data only. Given the lack of tools available for automatic gating of two-channel ddPCR data, the main aim of this project is to provide a robust tool to analyze dual-channel ddPCR data automatically.

Chapter 2 will explain in detail each step of the algorithmic methods developed. Chapter 3 presents a software implementation of the methods developed in Chapter 2. In Chapter 4, a case study is used to demonstrate the utility of the developed methods by analyzing a ddPCR dataset with the tool described in Chapter 3.

Chapter 2: Algorithm for automated ddPCR analysis

This chapter provides the details of a four-step algorithm to automatically analyze ddPCR data. The first three steps, which can be thought of as pre-processing, can be performed on any ddPCR dataset. The last step, droplet gating, can only be applied for ddPCR assays that match certain criteria such as having one single-positive and one double-positive droplet populations, as explained in Section 2.1.

2.1 Preconditions

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 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 types of assays, I will 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.

While any ddPCR assay can benefit from the first three steps of the analysis algorithm, the fourth step explicitly targets PN/PP assays. More specifically, any ddPCR experiment with the following properties qualifies as a PN/PP assay and can use the full devised algorithm:

1. The assay must be a two-channel ddPCR assay. For single-channel assays, simpler algorithms such as 'definetherain' or 'ddpcRquant' may be suitable options.
2. The only possible states of each droplet are (assuming FAM and HEX dyes): *FAM$^+$HEX$^+$*, *FAM$^+$*, *HEX$^+$*, or *FAM$^-$HEX$^-$*. Assays with more possible droplet populations than these four are not compatible with this algorithm.

3. All wells on the plate have at most three different droplet types: $FAM^{-}HEX^{-}$, $FAM^{+}HEX^{+}$, and exactly one of FAM^{+} or HEX^{+}. There can be no structural expectation of samples with both FAM^{+} and HEX^{+} droplets.
4. The majority of the droplets are empty ($FAM^{-}HEX^{-}$). This requirement is often satisfied in ddPCR experiments, as many are designed to operate at a low CPD (copies per droplet) to ensure no more than one copy of each template per droplets.

Figure 7 shows examples of ddPCR data sets that can and cannot be automatically gated using the software developed in this thesis.

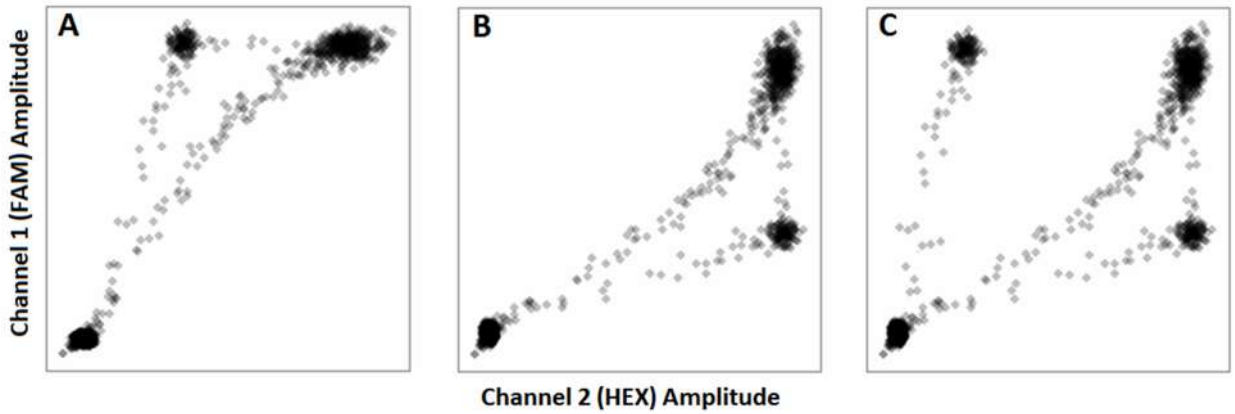

Figure 7 – Examples of PN/PP and non-PN/PP assays. Assays that result in droplet data such as in wells (A) and (B) qualify as PN/PP. (C) is not PN/PP because there are single-positive droplets in both channels.

2.2 Terminology and notation used

There are a few conventions and terminology used throughout the rest of this document:

- The ddPCR system can be set to detect FAM (channel 1) and either HEX or VIC dyes. For brevity, this chapter assumes that a HEX dye is used and read in channel 2, but every reference to a HEX dye could be applied to an assay that instead uses VIC.
- The fluorescence amplitude data for the set of read droplets can be represented as a list of n matrices, where n is the number of wells read on a plate. Each matrix contains two data columns and m_i data rows, where m_i is the number of droplets in the i^{th} well. The pair of data in each row in the matrix represents the end-point fluorescent signal values of a single droplet, and the data in each column are the measured fluorescent amplitudes in

the HEX and FAM channels for the set of read droplets in well i, as depicted in Figure 8. These matrices are the input to the analysis algorithm.

- Many of the numeric values used for parameters within the algorithm are merely illustrative and can be modified by the user. Any number that appears as **bold** signifies one such adjustable parameter value.

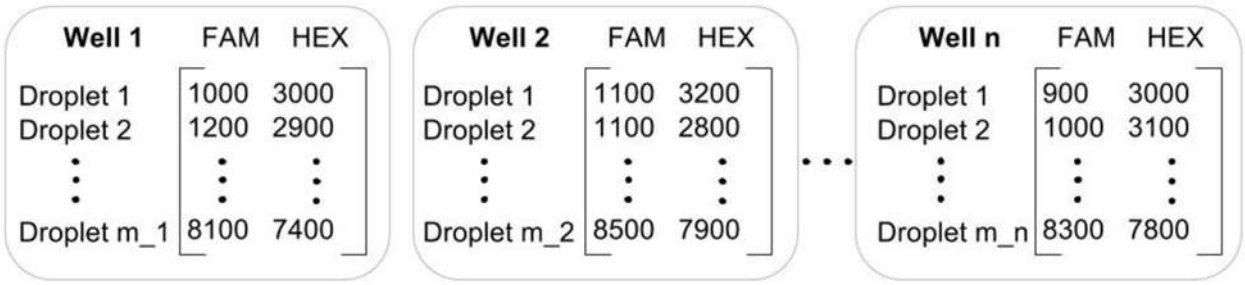

Figure 8 – Diagram showing how ddPCR droplet data can be represented as a 2-column matrix per well.

2.3 Automated analysis of any ddPCR data

Given droplet fluorescence data in the matrix format described above from any ddPCR experiment, three analysis steps can be performed automatically: identifying failed wells, identifying outlier droplets, and identifying negative (empty) droplets.

2.3.1 Step 1: Identify failed wells

The first step in the pipeline is to identify any wells that represent a failed ddPCR run by checking four quality control metrics. The failure could either be caused by human error or by a machine error. While not technically failures, no-template control wells are also deemed failures by this step, because these wells should be excluded from subsequent droplet analysis. The default values of the parameters used in these metrics are chosen to be conservative, which means that wells that seem like borderline failures may pass the checks. The other programs for ddPCR analysis do not perform any quality control checks on the data.

2.3.1.1 First failure condition

The first check is to ensure that the droplet generator has generated an acceptable number of readable droplets. The Bio-Rad QX100 droplet generator creates *ca.* 20,000 droplets per well in

theory. Typically, however, fluorescent amplitude data are recorded for only 12,000 to 15,000 droplets (see Appendix B). The quality of the remaining droplets is too poor (e.g. improper size, unacceptable spectral properties, etc.) based on internal sorting metrics set by the QX100 ddPCR droplet reader. Data for those droplets is therefore discarded by the reader. If the total number of droplets in a well does not exceed a specific threshold value of **5000** (Parameter – **REMOVE_FAILURES:TOTAL_DROPS_T**), the well is considered a failure because of an insufficient number of read droplets. The more droplets there are, the more accurate the results will be, and having too few droplets is likely to yield unreliable results. Figure 9 demonstrates how this test can be beneficial.

2.3.1.2 Second failure condition

Droplet clusters and their characteristics are further evaluated by performing two-dimensional k-means clustering with two centers on the fluorescence values of all read droplets in each well. Each of the two resulting clusters contains a center c and has an associated p, where $0 < p < 1$ is the proportion of the total droplets that belong to the cluster. In a typical ddPCR experiment, a portion of read droplets are positive for at least one template. The cluster with its center closest to the origin (point $(0,0)$ on a standard Cartesian plane) is comprised of empty droplets, while droplets farther away from the origin are positive for at least one template. The distance from each cluster center to the origin is calculated using standard Euclidean distance in order to determine which of the two clusters is the empty-droplet cluster and which is the positive cluster (containing droplets positive in either or both channels).

The second quality control metric ensures that there are at least two distinct droplet clusters and that there is sufficient segregation between the positive droplets and the empty droplets. The Euclidean distance between the centers of the two clusters is calculated and used as a heuristic to determine if the clusters are separated. If the distance between the two clusters is larger than the distance from the empty droplet cluster to the origin, then the clusters are considered far enough apart for robust gating and cluster assignments. This criterion can be defined more formally: let c_E be the center of the empty cluster in two-dimensional space, c_P be the center of the positive cluster, O be the origin point, and $d_{x,y}$ be the Euclidean distance between x and y. An experiment is considered a failed run if $d_{c_E,c_P} < d_{c_E,O}$, as shown in Figure 10.

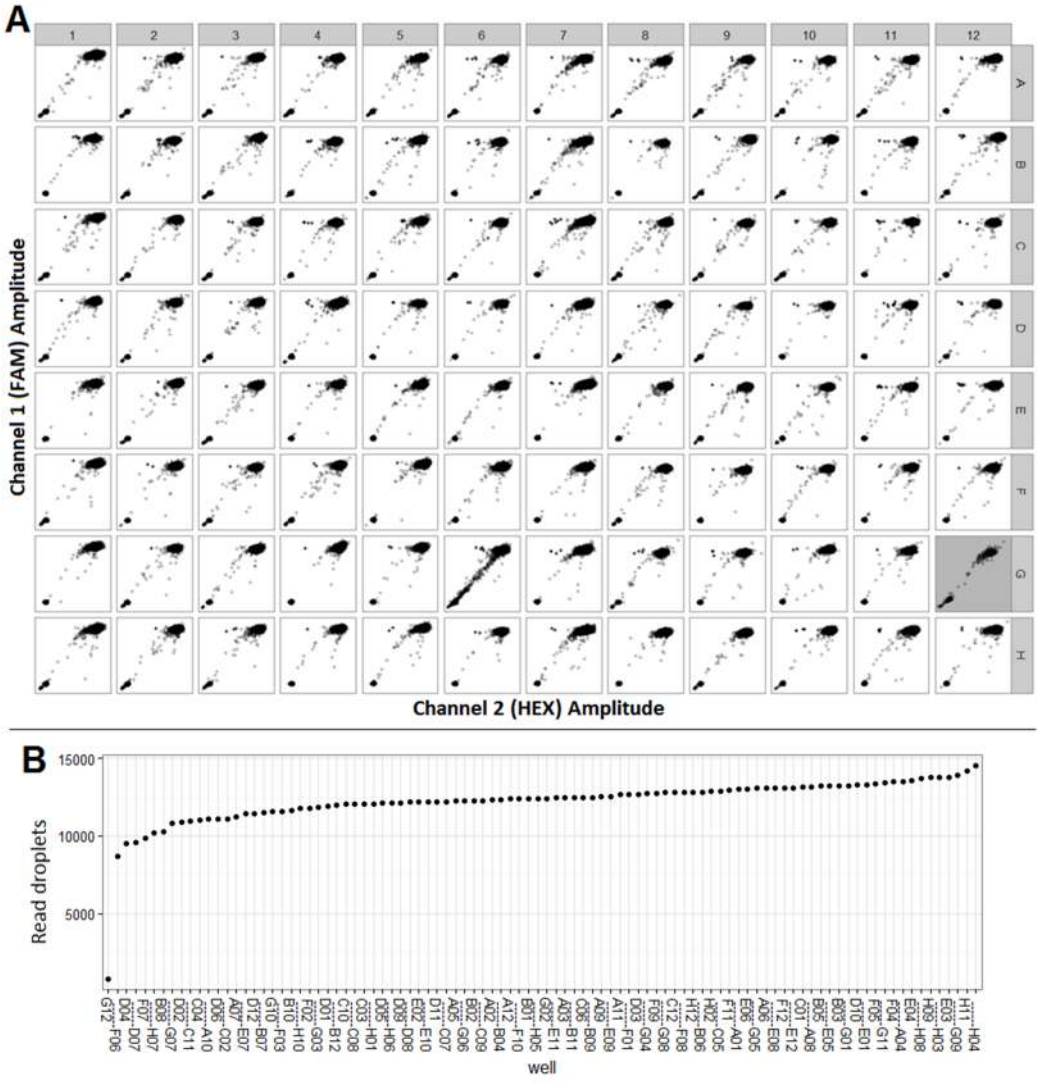

Figure 9 – Example of first failure condition: removing wells with insufficient number of read droplets. (A) Though it is difficult to notice visually when plotting the raw data, well G12 (grey background) contains far fewer droplets than the other wells, and is deemed a failure by the algorithm because of it. (B) By looking at a plot of read droplets per well, it's clear that G12 has insufficient droplets.

Failure to pass this check is indicative that the two droplet clusters are close together in two-dimensional space, which suggests either that there is only one true cluster or that any gating and associated cluster assignments will carry unreasonable levels of uncertainty. Recording only one cluster of droplets represents a failed ddPCR run because it means there are no template-containing droplets, which in turn means that no amplification took place. Wells that contain no-template controls will by design have only empty droplets and will be classified as failures due to this check. Figure 10 shows an example of a well that fails the segregation condition.

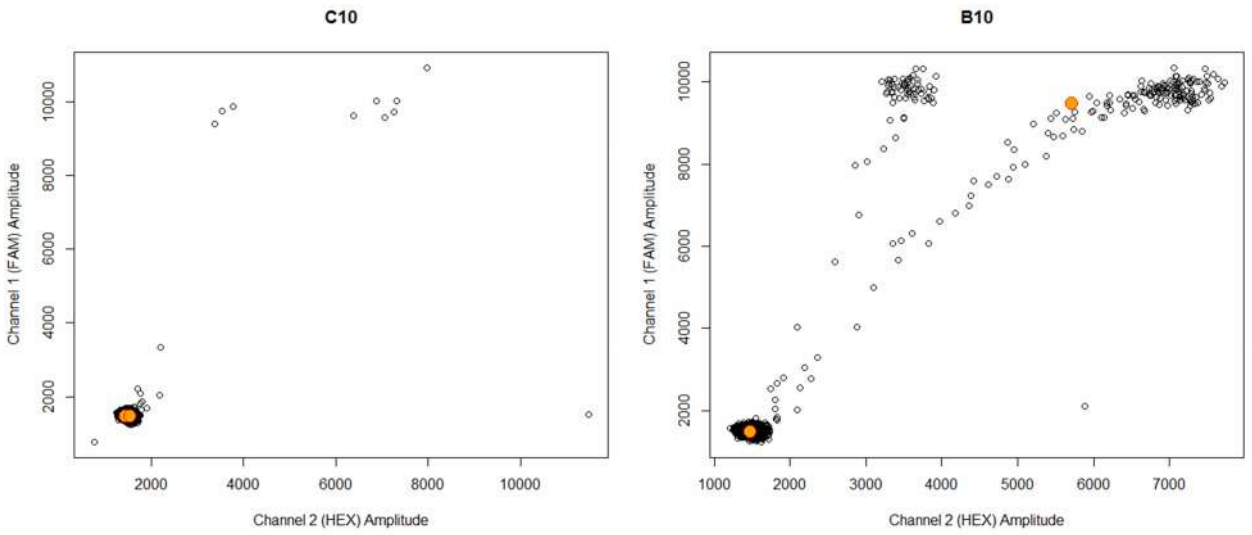

Figure 10 – Example of second failure condition: checking for proper cluster segregation. Well C10 (left-hand scatterplot) is classified as a failure by the algorithm because the two K-means centers (orange circles with red outline) lie extremely close to each other. Well B10 (right-hand scatterplot) passed the segregation condition because the K-means centers are well separated.

2.3.1.3 Third failure condition

Many ddPCR experiments, particularly when serving as a clinical assay, are designed to operate at or near a specific CPD. Thus, the fraction of all read droplets that are empty should be consistent with that CPD. The third metric evaluates the proportion of droplets that are in the empty cluster (p_E) to ensure the fraction of empty droplets is above a threshold, say **0.3** (**REMOVE_FAILURES:EMPTY_LAMBDA_LOW_T**). According to Poisson statistics, the expected p_E given a specific CPD is e^{-CPD}, so the parameter should be set to a value lower than the expected p_E. Failure to meet this criterion indicates a lack of an appropriately sized empty cluster and therefore a failed experiment based on the expected experimental design.

2.3.1.4 Fourth failure condition

Similar to the third condition, the fourth quality control check analyzes the proportion of empty droplets. If p_E is high in value relative to a threshold, say **0.99** (**REMOVE_FAILURES:EMPTY_LAMBDA_HIGH_T**), set based on the intended CPD, the experiment is considered a failure as well. Figure 11 shows an example of a well that is classified a failure because the positive droplets cluster is too small.

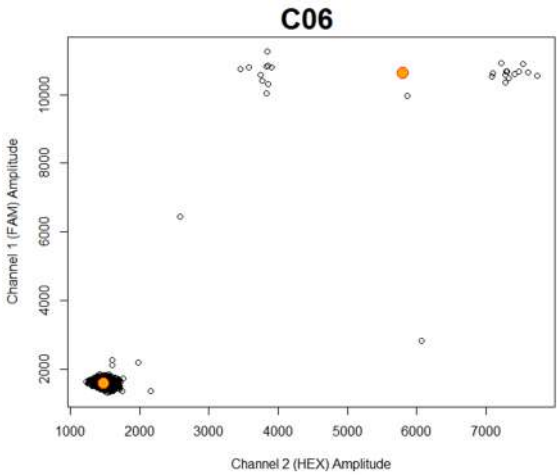

Figure 11 – Example of fourth failure condition: checking for sufficiently large non-empty cluster. The two K-means centers (red circles) in well C06 are far enough apart to pass the segregation test. However, the experiment is deemed a failure because the empty cluster contains 99.83% of the droplets, which is far above the threshold of 99.0%.

Any experiment (well) that does not meet all four criteria is deemed a failed run, and all such wells are flagged as failures so that they can be reviewed manually and are excluded from further automated analysis. Figure 12 shows the results of running the failure detection step on all wells in a plate.

2.3.2 Step 2: Identify outlier droplets

The second step in analyzing ddPCR droplet data is identifying outlier droplets, which are defined as droplets that are read and recorded by the QX100/QX200 droplet detector, yet cannot be assigned to a cluster because their fluorescence intensities are too large. Outlier droplets are rare and most wells do not have any such droplets, but they do occur sometimes. When outlier droplets exist, they are visually easy to spot and remove manually, but they do not have a technical definition, which makes them difficult to detect automatically.

The ddPCR machine's fluorescence detector cannot theoretically observe negative fluorescence, so the fluorescence intensities recorded from the droplets have a strict lower bound of 0. However, there is no upper bound on the value the ddPCR reader may read. As a result, it is possible to observe a small number of droplets with abnormally large HEX or FAM values, which can be considered outlier droplets.

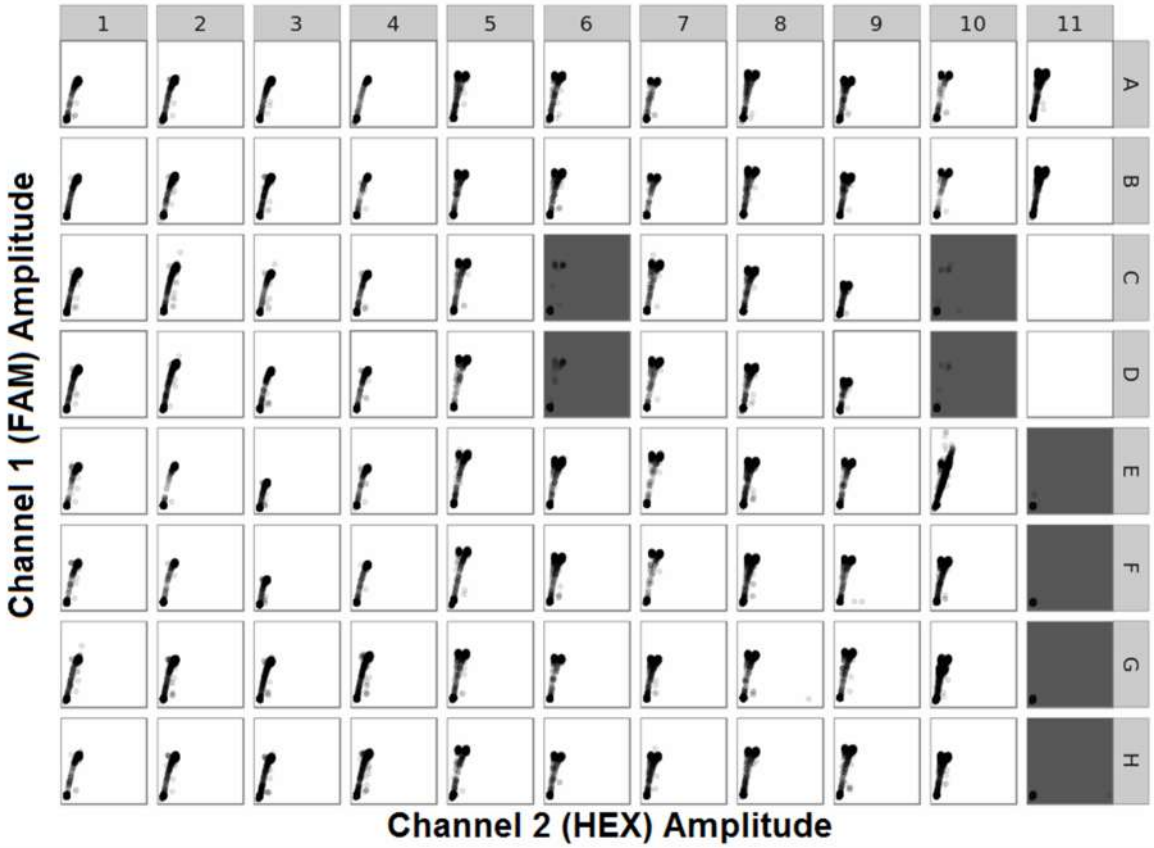

Figure 12 – Running the failure detection step on an entire plate. Eight out of 86 wells were deemed failures (grey background). Four wells (E11, F11, G11, H11) are no template controls, two wells (C10, D10) failed because of insufficient segregation between negative and positive droplets, and two wells (C06, D06) failed because of an insufficiently large cluster of positive droplets.

A common method for detecting outliers in normal distributions is to calculate the first quartile $Q1$, third quartile $Q3$, and interquartile range IQR (defined as $Q3 - Q1$), and define any observations that fall outside the range of $[Q1 - 1.5IQR,\ Q3 + 1.5IQR]$ as outliers [32] [33].

If ddPCR fluorescence data were normally distributed, this constraint could be applied as a one-sided test, as droplets with the minimum RFU of 0 will be considered empty rather than outliers. Thus, only the upper bound of $Q3 + 1.5IQR$ would be applied and used to define outliers. However, as shown in Figure 13, ddPCR data is extremely skewed and non-normal, due to rain and the fact that a large portion of the droplets are empty. Since the assumption of normality is not met, the common outlier detection approach outlined above is not directly suitable.

However, a small modification can be made to make it applicable to ddPCR data. For each of the two channels (for example, FAM and HEX), the top **1** percent (**REMOVE_OUTLIERS:TOP_PERCENT**) of droplets having the highest signal value in that dimension are identified. An outlier threshold is then defined as $Q3 + k * IQR$, where $k = \mathbf{5}$ (**REMOVE_OUTLIERS:CUTOFF_IQR**) and the quartiles are calculated using only the values of the highest 1%. Any droplets that have fluorescence amplitude larger than this threshold are considered outliers. This is essentially a variation of the common outlier detection method with two modifications: only the extreme values in the data are used to calculate the outlier threshold, and the outlier range is extended from $1.5IQR$ to $5IQR$ in order to err on the side of inclusion.

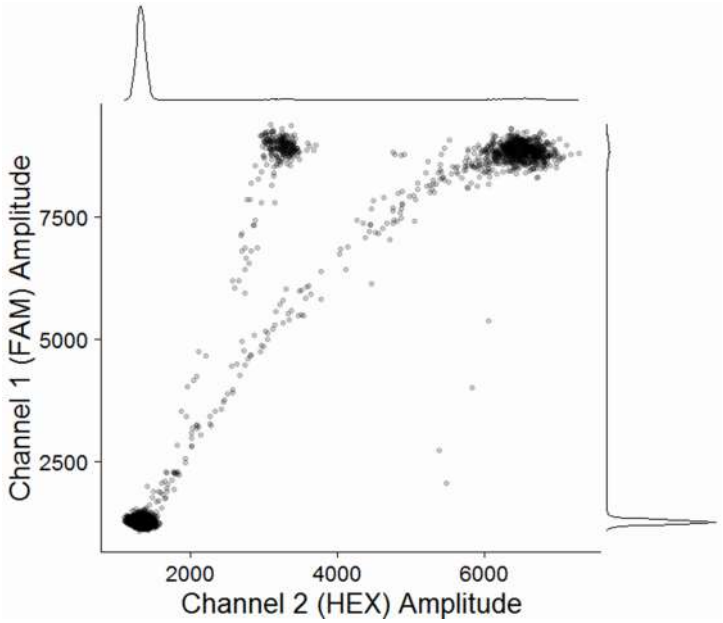

Figure 13 – ddPCR data is often extremely skewed. A typical ddPCR experiment in which the majority of droplets are empty. Droplets have an alpha transparency to better indicate their density. The marginal density plots are further visual aids to show where the majority of droplets lie.

The outlier threshold is set and applied to both fluorescence channels. Any droplet that has a FAM intensity value exceeding the FAM outlier threshold or a HEX value exceeding the HEX outlier threshold is deemed an outlier droplet, and removed from further analysis. The outlier detection process is depicted in Figure 14.

Appendix C shows the results of applying the outlier detection step to different ddPCR datasets.

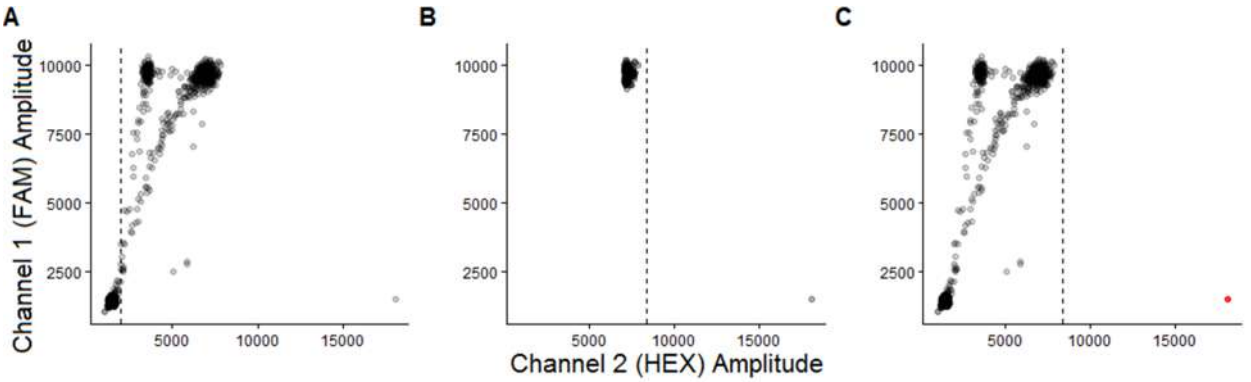

Figure 14 – Outlier detection algorithm. (A) A well with a single outlier in the HEX channel is depicted. The dotted line shows the HEX value of $\boldsymbol{Q3 + 5 * IQR}$, which does not correctly identify outliers. (B) The droplets with the highest 1% of HEX values are retained, and a threshold of $\boldsymbol{Q3 + 5 * IQR}$ of these droplets is calculated (dotted line). (C) The original droplet data is shown with the threshold that is calculated using the top 1% of HEX values. Using this threshold, one droplet (red) is identified as an outlier.

2.3.3 Step 3: Identify empty droplets

Any droplet that does not contain an amplifiable template is expected to emit low (background) fluorescence when read by the detector. Therefore, droplets with very low fluorescent signal in both channels are assumed to be empty droplets and are removed from any downstream analysis.

The removal of empty droplets is important in order to increase the signal-to-noise ratio in the data. Empty droplets can be seen as noise because they do not contain any of the target DNA. Even though a large proportion of the droplets are discarded, no useful information is lost since all the template-containing droplets are retained. Another benefit of removing empty droplets is that reducing the size of the data also allows for faster computations on the remaining droplets.

Empty droplets are identified in each well separately. Similar to the previous step where an outlier threshold value was determined for each of the two fluorescence channels, a threshold for empty droplets is defined in each channel. For each channel, first a two-component Gaussian mixture model is fitted to the fluorescence signals of all droplets in a well (Figure 15-A). Each of the two components is characterized by a mean (center) μ and a standard deviation σ. The Gaussian distribution with the lower mean is assumed to correspond to the empty droplets, and it will likely have a relatively small standard deviation as the empty droplets are generally densely clustered around some background fluorescence. A threshold for empty droplets is then

calculated using the normal model. In a normal distribution, 99.9999% of values lie within 5 standard deviations of the mean. With this knowledge, an empty threshold is defined as $\mu + k * \sigma$, where $k = \mathbf{7}$ (**REMOVE_EMPTY:CUTOFF_SD**). This essentially sets an upper limit on fluorescence values that can be considered empty.

A threshold is calculated for both channels independently in such a manner, and any droplets in the well that emit a lower fluorescence than the thresholds in both channels are deemed empty and not considered in further analysis. A droplet that does not meet the threshold in one channel but is above the empty threshold in the other channel is not considered empty. Figure 15 shows an example of applying this step to an experiment.

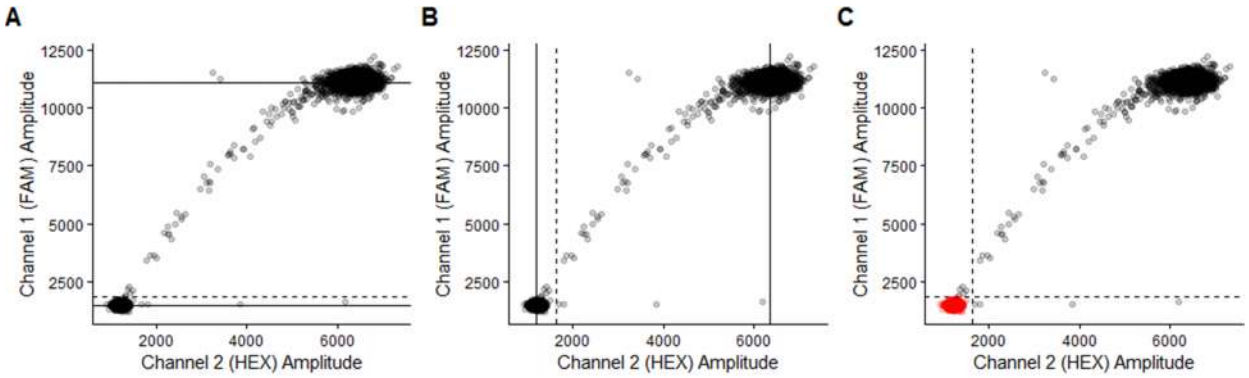

Figure 15 – Identifying empty droplets in a ddPCR experiment. (A) A two-component Gaussian mixture is fitted to the FAM values of all droplets. The two solid black lines represent the FAM values of the two means. The dotted line shows the FAM value of $\boldsymbol{\mu + 7 * \sigma}$ of the lower population. (B) The process in (A) is repeated in the HEX channel. (C) The dotted lines are used as the thresholds for empty droplets; any droplets below these thresholds are classified as empty (red).

2.4 Automated analysis of PN/PP assays

The methods described above for identifying failed experiments, identifying outlier droplets, and assigning empty droplets (sections 2.3.1 – 2.3.3) can be applied to any ddPCR experiment. In contrast, the droplet-gating tool developed in this work specifically applies to PN/PP-type ddPCR experiments. For that broad class of dual-channel ddPCR experiments, the algorithms defined in sections 2.3.1 – 2.3.3 can be further tailored to improve performance given the extra knowledge about the type of assay. Those refinements and the droplet gating method that

together comprise the entire algorithm developed to analyze PN/PP experiments are described below.

The algorithm in this section describes the steps required to analyse a *(FAM^+)/(FAM^+HEX^+)* PN/PP assay. The exact same steps can be taken to analyze a *(HEX^+)/(FAM^+HEX^+)* PN/PP assay instead by simply swapping HEX with FAM, and vice versa, in every step.

2.4.1 Step 1: Identify failed experiments

Similar to the more general method described in 2.3.1, four quality control checks are used to identify a failed PN/PP assay. The first condition, checking for a minimum number of droplets, is the same, while the other three conditions can take advantage of an optimization because there is prior knowledge in a PN/PP assay about the nature of the droplet clusters.

Instead of performing two-dimensional k-means clustering, k-means can be carried out in only one dimension. In the case of a FAM-positive PN/PP assay, k-means is only performed on the FAM signals of all droplets. This results in two clusters, each characterized by a separate center value. The cluster with the lower FAM center is assumed to be that of the empty droplets.

The second condition, which checks for adequate segregation between the two clusters, uses the same calculations as in Section 2.3.1.2, but all calculations are reduced from two dimensions to one. For example, the distance between the centers of the empty and positive clusters (d_{c_E,c_P}) is simply defined as $|x - y|$ since that is the Euclidean distance when applied in one dimension. Similarly, the distance from the center of the empty cluster c_E to the origin is simply the value of c_E itself. Thus, the inequality $d_{c_E,c_P} < d_{c_E,O}$ that determines if an experiment has failed can be simplified in PN/PP assays to $|x - y| < c_E$. Any well for which this inequality holds true is considered a failed experiment.

The third and fourth conditions, which ensure the empty cluster is neither too large nor too small, use the same methods that are used in Sections 2.3.1.3 and 2.3.1.4, with the exception that they operate on one-dimensional rather than two-dimensional clusters.

2.4.2 Step 2: Identify outlier droplets

Outlier droplet detection is performed in the exact same manner as described in Section 2.3.2, with the data for all droplets defined as outliers removed from the analysis.

2.4.3 Step 3: Identify empty droplets

Empty droplets identification is done in a similar fashion to the method described in Section 2.3.3, but an empty threshold is only defined in the FAM channel rather than in both channels. Only the FAM signal is used because in a $(FAM^+)/(FAM^+HEX^+)$ assay all positive droplets must emit some FAM fluorescence signal above the background value, and therefore the FAM channel is enough to discriminate between the empty and positive droplets. Any droplet with a FAM fluorescence amplitude below the FAM empty threshold is considered empty. Figure 16 below shows a summary of this step. Appendix D shows several examples of applying the empty droplet detection algorithm to different wells.

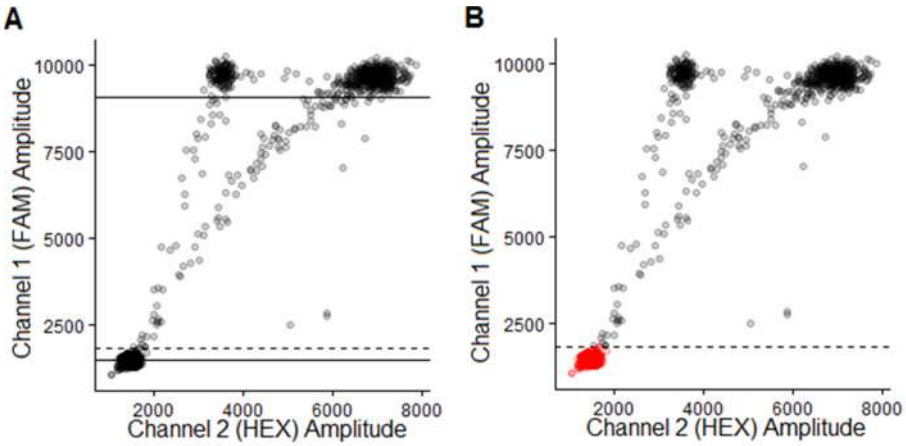

Figure 16 – Identifying empty droplets in PN/PP assays. (A) A two-component Gaussian mixture is fitted to the FAM values of all droplets. The two solid black lines represent the FAM values of the two means. The dotted line shows the FAM value of $\boldsymbol{\mu + 7 * \sigma}$ of the lower population. (B) The dotted line is used as the threshold for empty droplets; any droplets below the line are classified as empty (red).

2.4.4 Step 4: Gate droplets

At this step, all failed experiments, empty droplets, and outlier droplets have been removed. The main remaining step of the analysis pipeline is to gate the remaining droplets and classify them as positive-negative (PN, i.e. FAM^+), positive-positive (PP, i.e. FAM^+HEX^+), or rain. The two

types of positive droplets (PN and PP) are termed "filled" droplets because they contain template DNA, while rain droplets are not considered filled for the purposes of this algorithm (though they may contain template).

Figure 17 below shows an example of a well with all its droplets classified as rain, PN, or PP. For completeness, the empty droplets are shown as well.

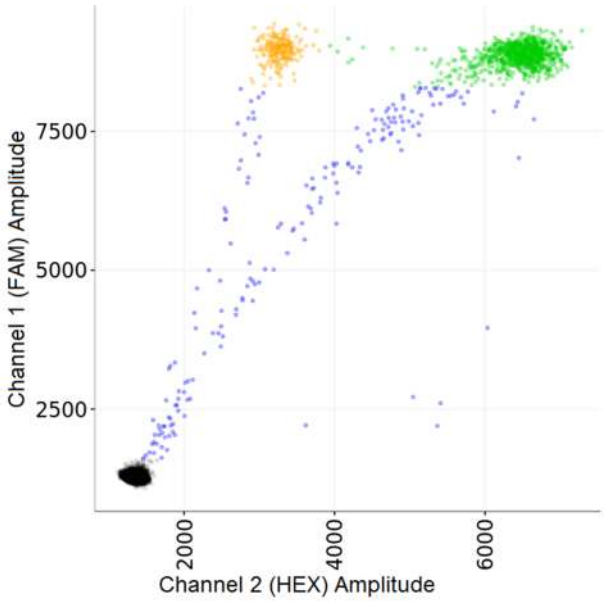

Figure 17 – Proper droplet gating of a ddPCR experiment. Correct gating of this well will group the droplets into four groups: PN (orange), PP (green), rain (blue), and empty droplets (black).

2.4.4.1 Identify rain droplets

While theoretically the only expected non-empty droplets are either singly-positive (PN) or double-positive (PP), clinical ddPCR datasets normally also contain many rain droplets that fall ambiguously between negative (empty) and positive (filled) droplet clusters. Since it is unclear if template is present and amplified within rain droplets, these droplets are often omitted from manual analysis [29]. They are identified and removed by this algorithm as well, though there is currently no definitive recommendation regarding whether or not rain droplets should be included in an analysis. However, because template-positive droplets far outnumber rain droplets, the answer to that question is of more theoretical than practical importance, as rain droplets do not significantly impact the ability to accurately quantify template abundance.

In a similar fashion to the way empty droplets are identified, rain droplets are identified by calculating a unique FAM threshold in each well. For each well, a two-component Gaussian mixture model is fitted to the FAM signals of all droplets in the well. Each of the two components is defined with a mean μ and a standard deviation σ. The filled droplets (FAM^+ and FAM^+HEX^+) all have similar FAM values, while the rain droplets have a much wider range of FAM values that are, by definition, lower. Therefore, the Gaussian distribution with the larger μ is assumed to capture the filled droplets. A FAM threshold is then calculated as $\mu - k * \sigma$ using the mean and standard deviation of the filled Gaussian, where $k = \mathbf{3}$ (**CLASSIFY:CUSTERS_BORDERS_NUM_SD**). Any droplets in the well below this value are considered rain, and all droplets above this threshold are considered the filled droplets and are used in the next step. Figure 18 shows the process used to identify rain droplets and the filled droplets.

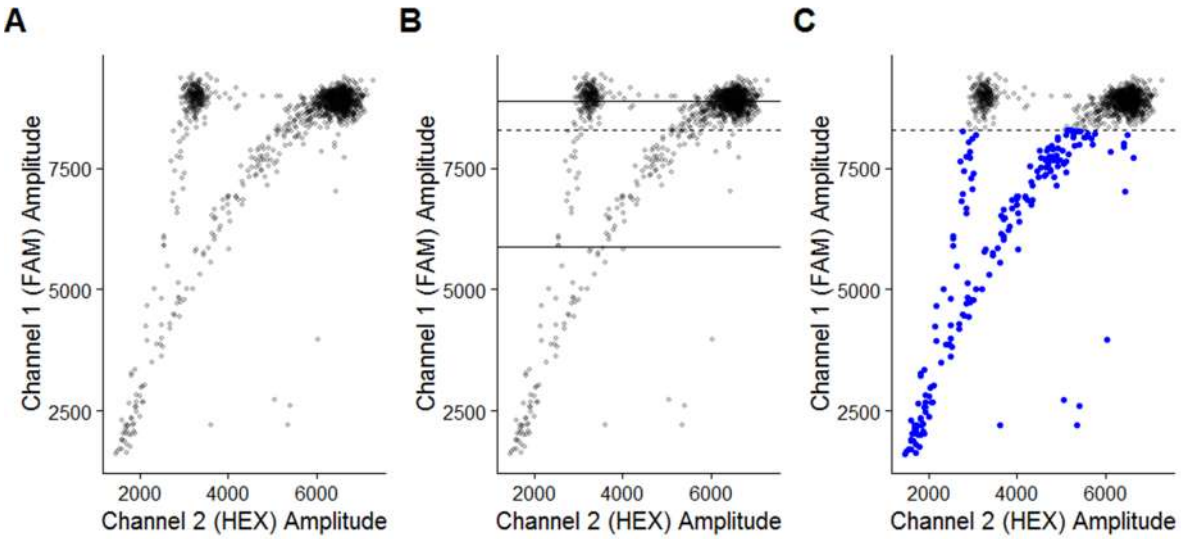

Figure 18 – Identifying rain and filled droplets. (A) A typical PN/PP well is shown, after the empty drops have been removed. (B) A two-component Gaussian mixture is fitted to the FAM values of all droplets. The two solid black lines represent the FAM values of the two means. The dotted line shows the FAM value of μ – 3*σ of the upper (filled droplets) population. (C) The dotted defines the threshold; any droplets below the threshold are classified as rain (blue).

2.4.4.2 Assign PN and PP droplets

After removing all empty and rain droplets, the only remaining droplets at this stage contain either FAM^+ (PN) or FAM^+HEX^+ (PP) droplets.

While all filled droplets have similar FAM values, their HEX values are the best predictor to discriminate between them. Therefore, gating the droplets in this step partitions the filled droplets into two clusters based on their HEX values. While employing simple clustering methods such as k-means to partition the two groups of droplets may seem like a feasible solution, it does not yield good results. Other clustering methods such as k-medoids, divisive hierarchical clustering, and Gaussian mixture models also do not produce desirable results. Appendix E shows representative results for some of these approaches attempted.

Instead of using a standard clustering approach, the two clusters are defined by calculating a gate in the HEX dimension that separates the two template-positive (PN and PP) droplet clusters. First, the kernel density estimate (KDE) of the HEX values is calculated. After the HEX KDE is obtained, the local maxima (peaks) and local minima (troughs) of the density distribution are calculated. The peaks correspond to the HEX values of the two droplet clusters, and the trough between them can be used as the gate between FAM^+ and FAM^+HEX^+ droplets. Figure 19 shows an overview of this concept. This approach is able to correctly identify the PN droplets both when they form a large cluster (as in Figure 19) and when that cluster is comprised of only a few such droplets (Appendices F1 and F2).

The PN/PP assay for which data are shown in Figure 19 is designed to produce a major cluster of PP droplets, and a smaller (or no) cluster of PN droplets. When using an ideal smoothing bandwidth, the density estimate for the HEX values will either have two peaks and one trough or one peak and no troughs. In wells with only one peak, the peak captures the PP droplet cluster. In wells with a second cluster of PN droplets, the larger peak captures the PP droplet cluster, while the lower peak corresponds to the PN droplet cluster, and the trough between the two peaks defines the gate between the two types of droplets, as seen in Figure 19. See Appendix F3 for an example of gating a well without a PN droplet cluster.

When computing the kernel density estimate, the degree of smoothing is an important parameter that strongly affects assignment of droplets to the resulting clusters, so the algorithm attempts to find the optimal smoothing bandwidth by iteratively increasing the bandwidth parameter and assessing the quality of the resulting KDE. To better understand this, we can focus again (and

for the remainder of the chapter) on a PN/PP experiment of the type shown in Figure 19. If the smoothing bandwidth is too high, the PN cluster will not be identified (see Appendix F4), while too low of a bandwidth will result in too many peaks and an inability to identify the correct PN cluster (Appendix F5). The optimal bandwidth for correct gating varies even between wells on the same plate, so an iterative process of finding the optimal smoothing parameter is performed on every well by multiplying the initial bandwidth b by an increasingly larger number k.

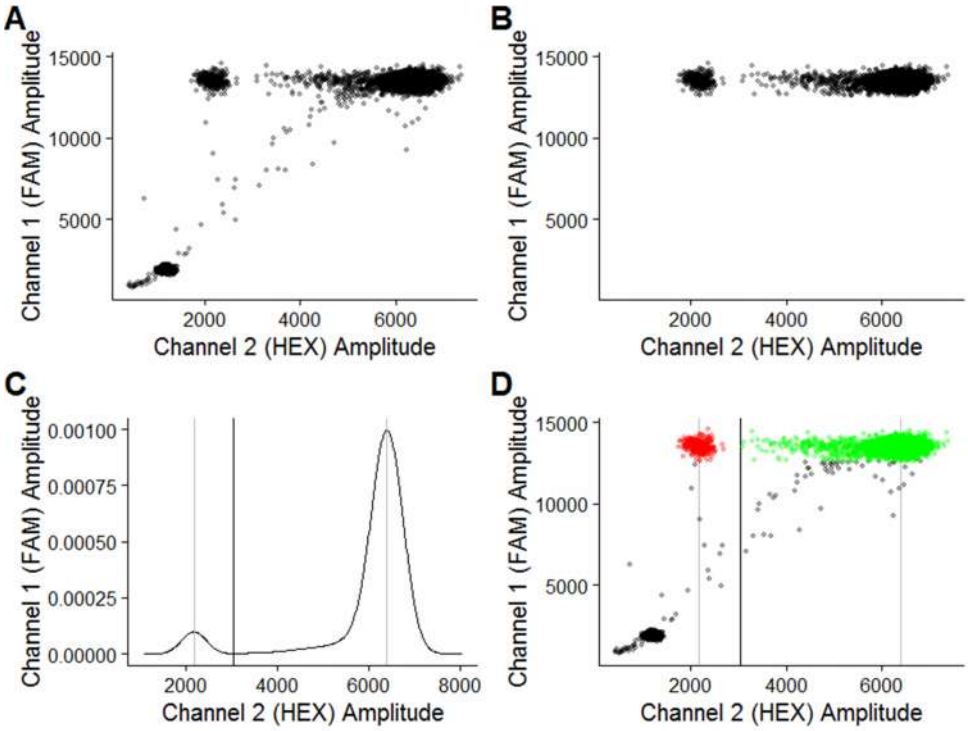

Figure 19 – Gating PN (FAM^+) and PP (FAM^+HEX^+) droplets using a kernel density estimator. (A) A typical PN/PP experiment containing both FAM^+ and FAM^+HEX^+ droplet populations is shown. (B) To calculate the gate between PN and PP droplets, only the filled droplets (not empty and not rain) are retained. (C) A kernel density estimator of the HEX values of the filled droplets is generated. The grey lines correspond to peaks in the HEX density, and the black line corresponds to the trough. The trough value is used as the gate. (D) The full set of filled data for the experiment is shown, using the trough as the threshold between PP (green) and PN (red) droplets.

Initially, k is set to 1 and the bandwidth value b is calculated using the method developed by Sheather & Jones [34], which is a popular bandwidth selection method that minimizes the mean integrated square error by introducing a bias term in the density estimator. If only a single peak is observed, it is indicative that there is only one droplet cluster, and that there are no PN

droplets. In this case, the HEX gate is set to 0, and all droplets are considered PP. If there are two peaks, then a gate is defined at the trough between the two peaks, with the droplets having HEX intensities below the gate assigned to the PN cluster and the droplets with HEX intensities above the thresholds assigned to the PP cluster. If there are more than two peaks, it means the bandwidth used is too narrow, so k is increased in increments of 0.5 until a maximum value of **20** (**CLASSIFY:ADJUST_BW_MAX**). In every such iteration, the initial bandwidth is multiplied by k and a new KDE is calculated. The number of peaks is then inspected and the process is repeated until only two peaks are observed. If the maximum value of k is reached and the KDE still has more than two peaks, the gate is still defined using the left-most trough, but the well is also flagged for manual review. k is only increased up to a certain maximum value because by using too large of a bandwidth, the estimator becomes a very poor representation of the data and is no longer useful. Appendix F6 shows this iterative process of choosing the best k.

When counting the number of peaks at every iteration, there is one important heuristic that is employed. Usually the PN and PP droplets form distinct, dense, cohesive clusters. However, it is possible for a few droplets to have an unusually large HEX value, which can result in another (erroneous) peak, as shown in Figure 20. Though these droplets may be considered outliers, they may not be detected and removed by the outlier detection step, which is designed to be conservative in terms of droplet removal from the data. Since the peak with the highest HEX intensity is taken by the algorithm to be the PP cluster, this spurious sparsely populated peak, if erroneously identified as the PP droplet cluster, can give very inaccurate results. To prevent this problem, the largest peak is always assessed at every iteration of k by counting the number of droplets with a larger HEX value than the peak itself. If this is the correct peak, then there should be many droplets–about half the droplets in the positive cluster–beyond the peak, since the peak should be approximately in the middle of the cluster. If the peak is caused by only a few outlier droplets, then there will be very few droplets beyond it. Thus, every time a KDE is calculated and the peaks are assessed, if less than 10% of the filled droplets are beyond the largest peak, that peak is ignored.

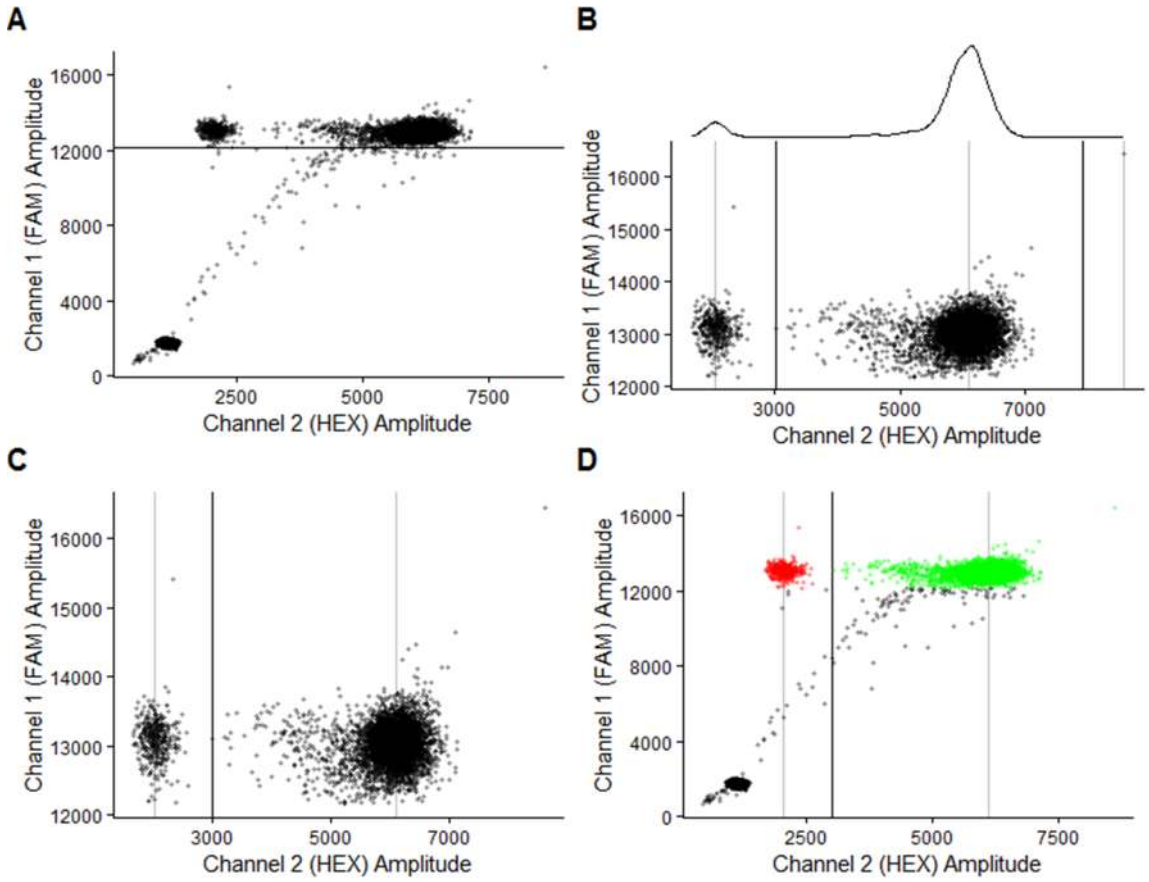

Figure 20 – Gating droplets in an experiment with a few outlier droplets. (A) A PN/PP well is shown, along with the threshold (black line) that separates the filled droplets from the rest. (B) Only the filled droplets are shown, along with the three peaks (grey lines) and two troughs (black lines) of the KDE of the droplets' HEX values. There is one positive outlier on the right that causes a peak to be centered on it. (C) Since less than 10% of the filled droplets are to the right of the right-most peak, it is removed along with the right-most trough. Now there is only one trough, which can be used as the gate between positive and negative droplets. (D) The full well is shown, using the trough as the threshold between positive (green) and negative (red) droplets.

Another heuristic is applied after the HEX gate that divides the PP and PN droplets is calculated. Since the positive cluster is larger and more well-defined, ddPCR operators using manual gating methods tend to place the gate close to the PN cluster, rather than in the middle of the trough separating PN from PP droplet clusters. In order to mimic this behaviour, a second gate is calculated, called the Gaussian gate. The mean μ and standard deviation σ of the HEX intensity values of the PN droplets are calculated, and the Gaussian gate is defined as $\mu + 3\sigma$. If the Gaussian gate has a lower HEX value than the original gate, then it is used instead. Figure 21 shows how this heuristic is applied, with more examples available in Appendix F7.

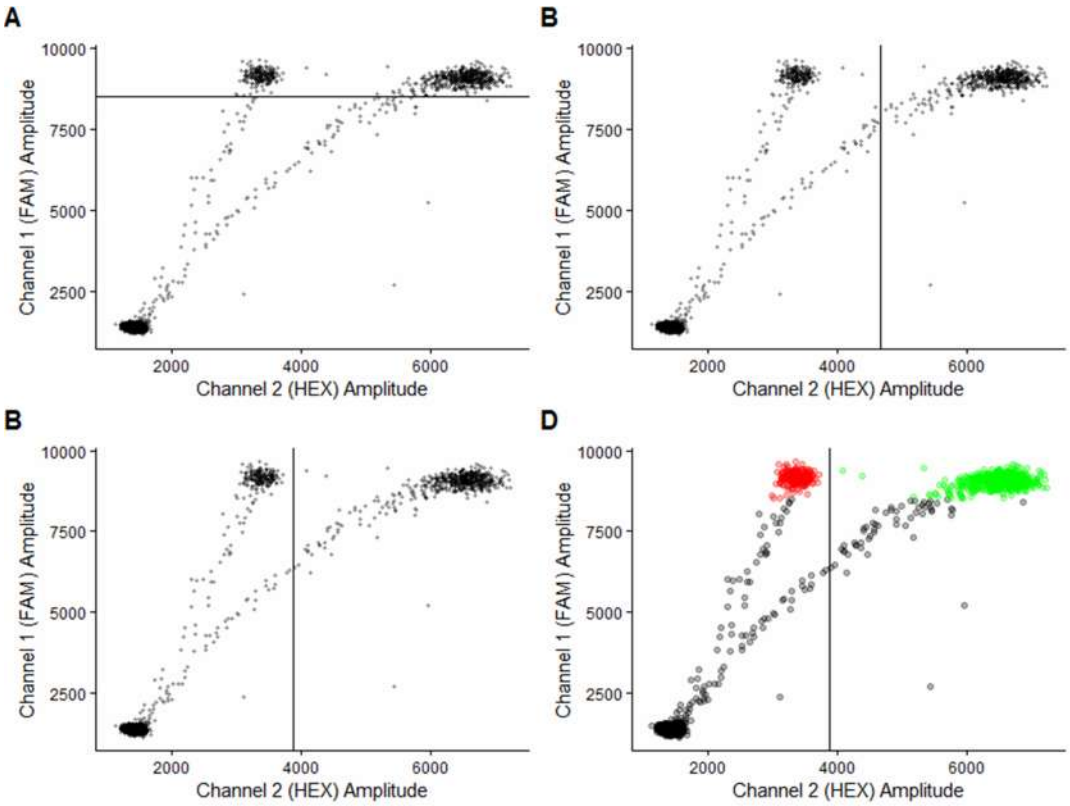

Figure 21 – Altering the original calculated gate to a Gaussian gate. (A) A PN/PP well is shown, along with the threshold (black line) that separates the filled droplets from the rest. (B) The gate calculated using the KDE of the HEX values is shown (black line). This gate appears to be in the middle of the two major droplet clusters. (C) All the droplets that were classified as negative using the original gate are used to calculate a new Gaussian gate (black line). The Gaussian gate has a lower HEX value, so it becomes the gate of choice. (D) The full well is shown, using the Gaussian gate as the threshold between positive (green) and negative (red) droplets.

2.4.4.3 Compute PN frequencies and classify wells as PN-positive or PN-negative

After gating, the numbers of PN and PP droplets can be directly enumerated and the frequency of PN droplets is computed. The PN frequency can have biological significance. For example, the fraction of total copies of a gene for which the template monitored by the HEX-labeled probe is not amplified is measured by the PN frequency. The PN frequency is given by the equation

$$PN\ frequency = \frac{number\ of\ PN\ droplets}{number\ of\ filled\ droplets} \qquad (1)$$

Using the PN frequency, it is possible to classify a sample as PN-positive if it contains a statistically significant number of PN (i.e. HEX-negative) droplets, or PN-negative (only PP

droplets are observed). In statistical terms, a sample scoring PN-negative contains data generated from a single distribution – the PP cluster – while a PN-positive sample contains data from two distributions – the PN and PP clusters. A PN-positive well is defined as having a PN frequency that is statistically significantly higher than **1**% (**CLASSIFY:SIGNIFICANT_NEGATIVE_FREQ**) at a significance level of **0.01** (**CLASSIFY:SIGNIFICANT_P_VALUE**). In other words, each well is tested to see if more than **1**% of the observed fluorescence data can be explained by a distribution of PN droplets. A well that has more than **1**% of its data coming from PN droplets with a p-value lower than **0.01** is considered PN-positive. A binomial test is carried out to test the significance of the PN frequency, with the null hypothesis stating that the real PN frequency is at most **1**%. These two parameters are highly dependent on the application of the ddPCR results, and should be modified accordingly. For example, if a clinically actionable decision can be made only when the PN frequency is above 10%, then **CLASSIFY:SIGNIFICANT_NEGATIVE_FREQ** should be set to 10%.

As an example, if a well has 7 PN droplets and 493 PP droplets for a total of 500 filled droplets, then its PN frequency is 1.4% using the above equation, which exceeds the 1% threshold to quality as PN-positive. However, to see if the well is truly PN-positive, the binomial test is applied. The null hypothesis states that the true PN frequency is lower than 1%. The appropriate binomial test in this situation is as follows:

$$
\begin{aligned}
&Probability\ of\ observing\ at\ least\ 7\ PN\ droplets \\
&in\ a\ well\ with\ PN\ frequency\ of\ 1\% \\
&= P(X \geq 7) \\
&= 1 - P(X < 7) \\
&= 1 - [P(X = 0) + P(X = 1) + \ \dots\ + P(X = 6)] \\
&= 1 - [b(0;\ 500, 0.01) + b(1;\ 500, 0.01) + \ \dots\ + b(6;\ 500, 0.01)] \\
&= 0.237 \\
&> 0.01
\end{aligned}
\qquad (2)
$$

Since the p-value is higher than 0.01, the well is not classified as PN-positive despite having a PN frequency of 1.4% and is instead considered PN-negative.

However, a well with 70 PN droplets and 4930 PP droplets does get classifies as PN-positive despite having the same PN frequency of 1.4%. Carrying out a similar binomial test reveals that the p-value in this case is 0.004, which is well below the required significance level of 0.01, and therefore the well would be described as PN-positive.

2.4.4.4 Revisit gating of PN-negative wells

PN-positive wells have well-defined clusters of PN and PP droplets, making it easy to determine where to gate them. On the other hand, PN-negative wells have very few, if any, PN droplets, making it difficult to accurately identify them. It is possible to leverage the data in the PN-positive wells to more accurately find PN droplets in PN-negative wells. This is only possible if there are enough wells to draw data from, so this step should only be performed if there are at least **4** PN-positive wells in the plate (**RECLASSIFY:MIN_WELLS_NEGATIVE_CLUSTER**).

The basic idea is to see where the PN droplets are located relative to the PP droplets in most wells with a significant number of PN droplets (PN-positive wells), and use this information to predict where PN droplets should appear in PN-negative wells. A PN-to-PP-cluster ratio (N/P ratio) is calculated for every PN-positive well by comparing the largest HEX value of the PN droplets to the median HEX value of the PP droplets. This N/P ratio is a metric used to define where the two droplet clusters are located relative to each other. After calculating the N/P ratio for all PN-positive wells, the **75**th percentile (**RECLASSIFY:BORDER_RATIO_QUANTILE**) of these values is chosen as the consensus N/P ratio. The median HEX value of all filled droplets in each PN-negative well is then multiplied by the consensus N/P ratio, and the resulting value is used as the new gate between PN and PP droplets.

Chapter 3: Software tool – *ddpcr* R package

3.1 Introduction

The algorithm developed in Chapter 2 has been implemented in the R programming language in order to show its viability and to make it immediately usable to any person running ddPCR experiments, regardless of programming expertise level. The R language was chosen because it is open-source and cross-platform, which allows anyone to use it freely. R is also a very popular language in the field of computational biology, and is the main language for data analysis for many scientists.

The algorithm is implemented as part of an R package named '*ddpcr*' which is freely available, along with its source code, on CRAN (http://cran.r-project.org/package=ddpcr) and GitHub (https://github.com/daattali/ddpcr). The package includes a command-line interface for R programmers, as well as a web application developed using the '*shiny*' R package [35] that supports a point-and-click user interface.

3.2 Usage

The *ddpcr* package can be used to explore, analyze, and visualize any ddPCR dataset. It is designed to be easy to use for the average user by making many assumption that allow common use cases to be run with minimal code, while also highly customizable for advanced users by allowing many parameters to be modified and providing advanced functions for defining custom analyses.

3.2.1 Plate objects and plate types

The most important object in the package is the `ddpcr_plate` object, or simply referred to as the "plate object". A plate object represents the droplet data for i independent experiments conducted on a 96-well PCR plate. It gets created either by loading read-droplet ddPCR data from QuantaSoft 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 read

droplets within each well of a particular ddPCR plate. It contains all the read droplet data for each experiment and various metadata of the plate, such as the dataset name and the number of read droplets in each well. The core functionality of the *ddpcr* package is to analyze a ddPCR dataset using the algorithm developed in Chapter 2, and a plate object is both the input and output of the core analysis functions.

As discussed in Section 2.1, there are different types of ddPCR assays, defined by the expected droplet populations. The `ddpcr_plate` type, which is the default plate type, can be used for any plate of results from independent wells running any ddPCR assay. When running the automated analysis on a plate of type `ddpcr_plate`, the algorithm developed for generic ddPCR assays (Section 2.3) is used, and no droplet gating is performed. *ddpcr* also offers the plate types `fam_positive_pnpp` and `hex_positive_pnpp`, which can be used when working with a FAM-positive or HEX-positive PN/PP-type ddPCR assay, and will consequently use the algorithm for PN/PP-type assays described in Section 2.4. Users who prefer to gate all droplets in each well of a plate using manually-defined fluorescence thresholds, similar to the manual gating tool provided by QuantaSoft, can use the plate type `custom_thresholds`.

3.2.2 Data import

Before beginning any analysis with the *ddpcr* package, the first step is to create a plate object by importing the required 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 from QuantaSoft into an accessible file format. QuantaSoft offers an option to export the droplet event data as a set of CSV (comma-separated values) files using the **Export Amplitude and Cluster Data** button in the **Setup** tab (Figure 79 in Appendix G). One CSV file gets generated for each well, with each file containing the fluorescence values in both channels for every recorded droplet. The CSV file for each well has a standardized name consisting of the dataset name followed by the well number and ending with the word "Amplitude", for example "myexperiment_A01_Amplitude.csv". Other than the well files, QuantaSoft also offers an option to export a metadata file that contains information on each well, such as the name of the sample in each well and statistical information about the droplet clusters. Using the **Export**

CSV button in the **Analyze** tab (Figure 80 in Appendix G) generates this metadata file with the same name as the dataset.

After obtaining the CSV files for the plate, they can be used to create a new plate object using the `new_plate()` function. The CSV files for the wells are the only required input, as they contain all the necessary information about the droplets. Each such file contains the matrix described in Section 2.2, and by reading all the well files, the program obtains the list of matrices that the algorithm requires as input. The metadata file is an optional input, and is only used to map each well to its sample name. The `new_plate()` function can be called either with a list of individual CSV files, or with the path to a directory containing all the exported CSV files. In the latter case, the *ddpcr* package automatically finds all the valid well files in the given folder, as well as a metadata file if one exists. For documentation and learning purposes, *ddpcr* includes a sample dataset comprising of a subset of five samples from the cohort of colorectal cancer patient data that will be analyzed in Chapter 4. The `sample_data_dir()` function can be used to retrieve the folder containing the sample data. Figure 22 shows how to initialize a new plate object with the sample dataset, and the resulting output. When calling `new_plate()`, the plate type can be specified, and is assumed to be `ddpcr_plate` if it is omitted.

```
> library(ddpcr)
> dir <- sample_data_dir()
> plate <- new_plate(dir)
Reading data files into plate... DONE (1 seconds)
Initializing plate of type `ddpcr_plate`... DONE (0 seconds)
```

Figure 22 – Code to initialize a ddPCR plate using *ddpcr*

During data import, a small storage optimization is made by converting the fluorescence signal data from real numbers to integers. The QuantaSoft-exported fluorescence data contains two real numbers, one for each fluorescence channel, for each droplet. Assuming an average of 15,000 droplets per well, storing a full plate with 96 wells corresponds to storing 2,880,000 numeric values. Since a real number in R takes up eight bytes of storage, this equates to 23,040,000 bytes, or roughly 23 megabytes (MB). Integers in R only take up four bytes, so by rounding all the fluorescence values to their nearest integer, the size of the data is reduced by a factor of two. While theoretically this involves losing some data, namely the fractional part of each

fluorescence intensity recorded, this loss of information does not make a practical difference as the difference in values is negligible and does not affect the results of the algorithm.

3.2.3 Data exploration

As Figure 23 shows, the most basic way to explore a plate object is by simply printing it to the console. The output will show some basic information about the dataset and current state of the analysis.

```
> plate
                    ddpcr plate
                   -------------
             Dataset name : small
             Data summary : 5 wells; 72,727 drops
               Plate type : ddpcr_plate
Completed analysis steps : INITIALIZE
Remaining analysis steps : REMOVE_FAILURES, REMOVE_OUTLIERS, REMOVE_EMPTY
```

Figure 23 – Output of a plate summary in *ddpcr*

While printing a plate object gives an overview of its main characteristics, there are also several functions that return specific pieces of information about the dataset. Figure 24 below shows a few common functions that can be used to explore the data in a plate object: `name()` returns the name of a dataset, `type()` returns the type of plate object, `wells_used()` returns a list of all the wells that are used in this plate, `plate_data()` shows the droplet fluorescence data, and `plate_meta()` shows some metadata information for each well.

Another important tool for exploring a ddPCR dataset is the scatterplot that visualizes the fluorescence data. When given a plate object as its first argument, the `plot()` function will produce a scatterplot for each well of the plate, showing the fluorescence in both channels of every droplet, with each well in its own panel. The plotting is done using the '*ggplot2*' package [36]. A plate object can be plotted at any time during an analysis, and by plotting it just after initialization, it will show the raw droplet data. While there are dozens of plotting parameters, the default plot produced with a call to `plot(plate)` without any arguments often produces a useful figure.

```
> name(plate)
[1] "small"
> type(plate)
[1] "ddpcr_plate"
> wells_used(plate)
[1] "A01" "A05" "C01" "C05" "F05"
> plate_data(plate)
Source: local data frame [72,727 x 4]

    well   HEX   FAM cluster
   (chr) (int) (int)   (int)
1    A01   577   494       1
2    A01   515   495       1
3    A01   690   645       1
4    A01   929   860       1
5    A01   844   868       1
6    A01   942   907       1
7    A01   985   923       1
8    A01  1058   966       1
9    A01  1058   979       1
10   A01  1095  1002       1
..   ...   ...   ...     ...
> plate_meta(plate, only_used = TRUE)
  well sample row col used drops
1  A01   Dean   A   1 TRUE 15820
2  A05   Dave   A   5 TRUE 13165
3  C01   Mike   C   1 TRUE 14256
4  C05  Emily   C   5 TRUE 14109
5  F05   Mary   F   5 TRUE 15377
```

Figure 24 – Exploring basic information about a plate in *ddpcr*

3.2.4 Data analysis

An analysis of a ddPCR plate object involves transforming the droplet data through the series of steps described in the algorithm detailed in Chapter 2. As shown in Figure 23, it is always possible to see which steps have already run and which steps remain in the analysis. For any plate with the default type of `ddpcr_plate`, such as the plate shown in Figure 23, the analysis steps include removing failed wells, removing outlier droplets, and removing empty droplets. Wells in which a PN/PP-type ddPCR experiment has been conducted may also be subjected to automated droplet gating. Plates of type `custom_thresholds` also have a gating step, but the gates are defined based on values set by the user rather than programmatically.

Analyzing a plate object can be done step-by-step by using the `next_step()` function, which will run the next step in the pipeline, or by using the `analyze()` function to run the full algorithm to completion. Both these functions accept a plate object as input and determine what steps to run based on the plate type. The output from both functions is a plate object with the

droplet clusters and metadata updated based on the results of the algorithm. For example, after the outlier detection step is completed, each outlier droplet in the data is marked as an outlier, and the plate metadata gains a new variable showing how many outlier droplets are in each well. *ddpcr* also allows the user to select only a subset of wells for analysis using the `subset()` function. This can be useful when a full plate of wells is loaded but there is a need to analyze data from specific samples only.

Since each step of the analysis returns a modified version of the plate object, it is possible to inspect a plate object at any time during its analysis. For example, inspecting the plate metadata with `plate_meta()` after a plate is initialized (Figure 24) shows only a handful of metrics, but after running a full analysis on a plate, the metadata contains many variables. Similarly, plotting a plate object after it has been created will simply show the raw droplet data, while plotting a fully analyzed plate will show more information. Figure 25 shows the difference between plotting a FAM-positive PN/PP dataset before and after the automated analysis.

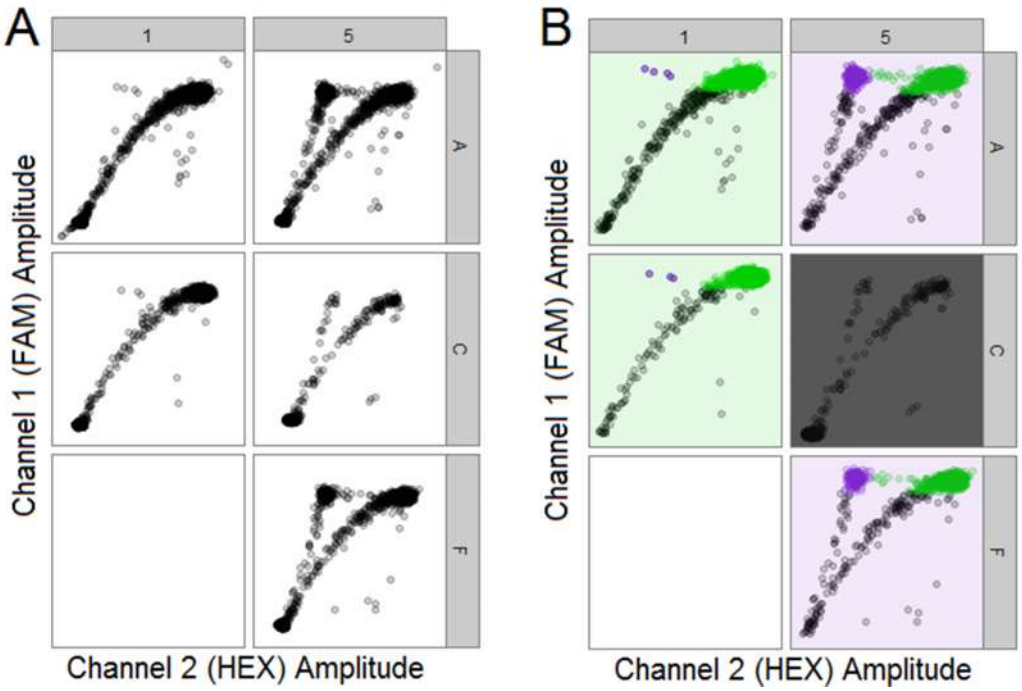

Figure 25 – Plotting a ddPCR dataset before and after analysis. (A) The raw fluorescence data is shown. (B) After analyzing the plate with *ddpcr*, the plot function differentiates between positive droplets (green) and negative (purple), as also differentiates between failed wells (grey background), positive wells (green background), and negative wells (purple background).

3.3 Implementation technical details

The *ddpcr* package makes extensive use of the S3 class inheritance system in R. The plate object is implemented as an S3 object of class `ddpcr_plate` with the R `list` as its base type. S3 objects are used for two main purposes: to override the behaviour of some base R functions when applied to plate objects, and to allow plate objects to inherit characteristics of other plate objects.

3.3.1 A plate object is a list

Every S3 object has a base type upon which it is built. The base type of a plate object is a list, as it 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 the following nine items:

- **plate_data –** A data.frame containing the fluorescence amplitude values and the cluster assignment for each droplet. The fluorescence values remain static throughout an analysis, while the cluster assignments start as "undefined" and are assigned specific clusters during the analysis.
- **plate_meta –** A data.frame containing the metadata information about each well. Each analysis step adds more variables to the metadata.
- **name –** A string representing the name of the dataset.
- **status –** An integer indicating the number of steps in the analysis pipeline that the plate has already run through.
- **params –** A list containing all the parameters used in the analysis algorithm.
- **clusters –** A vector of names of all the possible clusters a droplet can be assigned to. For example, any plate with a PN/PP type has seven possible clusters: undefined (a droplet that has not been analyzed yet), failed (a droplet in a failed well), outlier, empty, rain, positive (PP), negative (PN).
- **steps –** A named list containing the steps required to analyze the given plate. The name of each item in the list is a short descriptive name of the analysis step, and the value is the name of the function that contains the code to run the step. The items in the list of steps are ordered in the same order that the analysis is carried out.

- **dirty** – A Boolean flag indicating whether or not the analysis parameters have been modified since the analysis was run. If any parameters have changed, the `dirty` flag is set to true and the user is informed that the plate should be re-analyzed in order for the new parameters to take effect.
- **version –** A number representing the version of *ddpcr* at the time of plate creation. This information is not directly used by the program, but it can be useful when inspecting an old plate object to know whether it was created with the most up to date version of the package.

Except for the version and dirty flag, all other seven items in the plate's list have corresponding getter and setter functions for convenience. For example, the name of a dataset can be retrieved with a call to `name(plate)` and can be modified with a call to `name(plate) <- "my data"`. The only information about a plate object that is not inside the plate's list is the plate type. The plate's type is already stored as the plate object's class in order for S3 to function, so including the plate type in the list would be redundant.

3.3.2 Using S3 for plate objects 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()` and `plot()` generics are very often implemented for S3 classes, and they are especially useful for plate objects. 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 type and name, as well as the current analysis state of a plate. The `plot()` method generates a scatterplot of every well in the plate object and can be highly customizable using the many arguments it implements. While the base `plot()` method uses base R graphics, the `plot()` method for `ddpcr_plate` objects uses the '*ggplot2*' package [36]. The `subset()` generic is overridden by a method that can be called to retain a subset of wells from a larger plate. The `subset()` method uses two additional arguments, `wells` and `samples`, to specify which wells to select.

3.3.3 Using S3 for plate objects to support inheritance

Inheritance means that every plate type has a parent plate type from which it inherits all its features, but 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, FAM-positive and HEX-positive assays (defined in Section 2.1) are both PN/PP assays that share many similarities, so they can be thought of as inheriting many properties of a PN/PP assay. Another benefit of the *ddpcr* inheritance is that it allows any user to easily extend the functionality of the package by adding custom ddPCR plate types.

3.3.4 Analysis of different built-in plate types

The most basic plate type is `ddpcr_plate`, and every plate type inherits from it, either directly or by inheriting from other plate types that are descendants of `ddpcr_plate`. This is useful because it means that all functionality that is common to all ddPCR experiments is implemented once for `ddpcr_plate`, and every other plate type automatically inherits all the methods that `ddpcr_plate` has. When a new plate object is created (with the `new_plate()` function) without specifying a plate type, the `ddpcr_plate` type is assumed.

Calling the `analyze()` function on any plate will result in running the ddPCR data through a series of steps that are defined for the given plate type, and the read droplets will be assigned to one of several clusters associated with the given plate type. The exact set of analysis steps and potential clusters are determined by the plate type. Plates of type `ddpcr_plate` have four possible droplet clusters: “undefined” for any droplet that has not been assigned a cluster yet, “failed” for droplets in failed wells, “outlier” for outlier droplets, and “empty” for droplets without any template in them. Plates of type `ddpcr_plate` have 4 analysis steps:

"INITIALIZE", "REMOVE_FAILURES", "REMOVE_OUTLIERS", and "REMOVE_EMPTY". This means that any plate that is created will perform these basic steps of removing failed wells, outlier droplets, and empty droplets. Other plate types inherit the same clusters and steps by default, and can alter them to be more appropriate for each specific type. For example, PN/PP plate types use the same clusters, as well as three more: "rain", "positive" (PP), and "negative" (PN). PN/PP plate types also use the same list of steps as the base type, with the addition of steps at the end to perform droplet gating.

3.3.5 Plate parameters

Every ddPCR plate object has adjustable parameters associated with it (the `params` object in a plate's underlying list). There are general parameters that apply to the plate as a whole and are used throughout the entire analysis pipeline, and each analysis step also has its own set of parameters that are used for that step specifically. As shown in Figure 26, it is possible to view all parameters of a plate, view all parameters for a specific step, retrieve the value of a specific parameter, and edit specific parameters. After making any modifications to a plate's parameters, the plate needs to be analyzed again in order for the new parameters to take effect.

```
> str(params(plate))
List of 4
 $ GENERAL         :List of 4
  ..$ X_VAR          : chr "HEX"
  ..$ Y_VAR          : chr "FAM"
  ..$ DROPLET_VOLUME: num 0.00091
  ..$ RANDOM_SEED    : num 8
 $ REMOVE_FAILURES:List of 3
  ..$ TOTAL_DROPS_T       : num 5000
  ..$ EMPTY_LAMBDA_LOW_T : num 0.3
  ..$ EMPTY_LAMBDA_HIGH_T: num 0.99
 $ REMOVE_OUTLIERS:List of 2
  ..$ TOP_PERCENT: num 1
  ..$ CUTOFF_IQR : num 5
 $ REMOVE_EMPTY    :List of 1
  ..$ CUTOFF_SD: num 7
> str(params(plate, "REMOVE_OUTLIERS"))
List of 2
 $ TOP_PERCENT: num 1
 $ CUTOFF_IQR : num 5
> params(plate, "REMOVE_OUTLIERS", "TOP_PERCENT") <- 4
> params(plate, "REMOVE_OUTLIERS", "TOP_PERCENT")
[1] 4
```

Figure 26 – Code to retrieve and set parameters of a ddPCR plate with *ddpcr*

3.3.6 How the analysis steps work

The most important functionality of the *ddpcr* package is the ability to programmatically analyze a plate's droplet data. As described above, a plate object contains a list of ordered steps that can be accessed via the `steps()` command. The value of each element in the list refers to the name of the function that implements that analysis step. For example, the last step in the analysis of a basic plate of type `ddpcr_plate` is "remove_empty", which means that the `remove_empty()` function is automatically called when the analysis pipeline reaches this step.

One of the several properties of a plate object that is stored in the plate's underlying list object is its status. The status of a plate is simply an integer that indicates how many analysis steps have been taken. When a plate is created, the first step of initialization runs automatically, and the plate's status is set to one. To analyze a plate, the `analyze()` or `next_step()` functions are used. The `next_step()` runs the next step by examining the plate's list of steps and its current state to determine what function to call. The `analyze()` function can be used to run the full analysis pipeline on a plate, as it simply calls `next_step()` repeatedly until the analysis is completed.

There are several commonalities among all functions that implement the analysis steps:

- All the step functions accept a plate object as input and return a plate object as output. No other parameters are required to pass to any step function because all parameters should be contained within the plate object's `params`.
- All the steps functions are implemented as generics in order to allow new plate types to override their logic.
- To ensure reproducibility, the random seed is set at the beginning of every analysis step. This guarantees that even if the same step is called twice in a row, it will produce identical results.
- All step functions must update the status of the returned plate object to reflect the recently run step.

- All step functions call the `step_begin()` and `step_end()` functions to allow the user to see what step is currently being run and to calculate how long each step takes.

3.3.7 Creating new plate types

The inheritance system in *ddpcr* allows new plate types to be created by simply implementing a few functions. When creating a new plate type, any of the parameters, clusters, and analysis steps of any existing plate type can be modified, and new ones can be added. All the functions that are required to be added in order to create a new plate type are S3 generics and must be named appropriately, and they all accept one argument: a plate object. The following sections describe how to create a new plate type, and a fully worked example with working code that follows these rules can be found in Appendix H.

3.3.7.1 Define the parent plate type

Every ddPCR plate has a parent plate type from which it inherits all its properties. When creating a new plate type, the `parent_plate_type()` function should be defined, and its return value should simply be the parent plate type. If this function is not defined, the parent plate type is assumed to be the base type of `ddpcr_plate`. Inheriting from a parent plate means that the same clusters, analysis steps, and parameters will be used by default.

3.3.7.2 Define the plate type parameters

Every ddPCR plate type has a set of default parameters that are used during the analysis. When creating a custom plate type, if the new plate type needs a different set of parameters than its parent type, the `define_params()` function should be defined, and its return value should be the parameters specific to this plate. In the function body, a call to `NextMethod("define_params")` can be used to get a list of the parent type's parameters so that only new parameters need to be added rather than having to redefine all the parameters.

3.3.7.3 Define the potential droplet clusters

Every ddPCR plate type has a set of potential clusters the droplets can be assigned to. When creating a custom plate type, if the new plate type uses a different set of clusters than its parent type, the `define_clusters()` function should be defined, and its return value should be the

cluster names. In the function body, a call to `NextMethod("define_clusters")` can be used to get a list of the clusters available in the parent type so that only new droplet clusters need to be added rather than having to define them all.

3.3.7.4 Define the analysis steps

Every ddPCR plate type has an ordered set of steps that are run to analyze the data. When creating a new plate type, if the new plate type has different analysis steps than its parent type, the `define_steps()` function should be defined, and its return value should be a named list of the analysis steps. In the function body, a call to `NextMethod("define_steps")` can be used to get a list of the steps available in the parent type so that only new steps need to be added rather than having to define all of them. Any step that is returned in this list must have an associated function with the same name, and must follow the conventions outlined in Section 3.3.6.

3.3.7.5 Changing the algorithm of an existing step

While the `define_steps()` function is used to define new analysis steps that are unique to a new plate type, it can also be desirable to simply change the implementation of an existing step. For example, any ddPCR plate will, by default, use the `remove_failures.ddpcr_plate()` function as the algorithm for identifying failed wells. However, if a new plate type has a specific way of flagging failed wells, this step can be overwritten by defining a new S3 method for the new plate type.

3.3.7.6 Define a plot function

Given the fact that all plate types inherit from `ddpcr_plate`, calling `plot()` on any newly defined plate type will result in running the default plot function for `ddpcr_plate`. This plot function will generally be sufficient to convey the information in a new ddPCR plate type. If there is anything specific that needs to be added to plots of a new plate type, then a `plot()` function can be defined. It is recommended to build on top of the output of the default plot function rather than building a completely new plot. This can be done by calling `NextMethod("plot")` and adding plot elements on top of its output.

3.4 The *ddpcr* web application

3.4.1 Introduction

The *ddpcr* package includes a web-based application that allows users to perform an analysis of ddPCR data in an interactive visual environment. The web application implements most of the features available in the *ddpcr* package and makes them accessible via a simple point-and-click interface. The application is written using the '*shiny*' [35] R package. The main purpose of the web application is to allow users who are not familiar with the R programming language to use the *ddpcr* package. For users who are comfortable with R, the web application can still be a useful tool as a graphical user interface can be easier to operate than R commands. However, since the web application only supports a curated subset of the *ddpcr* functions, it is not as powerful as using the R command-line interface.

3.4.2 Main features

The *ddpcr* web app consists of four separate tabs that represent the steps taken for a typical ddPCR analysis, and a fifth tab with basic information about the application. 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 order of the first four tabs mimics the natural flow of a ddPCR analysis: upload a dataset, set plate parameters, analyze the plate, and explore the results. Appendix G includes screenshots from each of the tabs. The following sections describe the functionality available in each of the four main tabs.

3.4.2.1 Dataset tab

When the application loads, the Dataset tab appears and allows the user to select ddPCR data to analyze. There are three options for data selection: uploading a new dataset from QuantaSoft-exported CSV files, loading a plate object that was previously saved by *ddpcr*, or using a sample dataset. After obtaining a dataset, the user is instructed to move on to the Settings tab.

3.4.2.2 Settings tab

The Settings tab allows the user to customize the plate object before running the automated analysis. Basic plate properties such as the plate type and dataset name can be set, and all the algorithm parameters can be modified as well. The Settings tab also shows which wells on a plate are used, and the user can choose to only retain a subset of the available wells.

3.4.2.3 Analyze tab

After choosing a plate type and customizing the parameters of a plate in the Settings tab, the next step is to run the automatic analysis on the plate. The Analyze tab provides the interface to allow the user to begin the analysis, and it shows the output from the analysis in real-time. The output from the analysis includes a brief description of every step and the elapsed time of an analysis step.

3.4.2.4 Results tab

The Results can be explored both before and after the analysis, but it is much more informative afterwards. The Results tab offers a few ways to explore the data in a plate object.

First, the full plate metadata is shown as a table, which gives an overview of the results in each well. Many of the metadata variables are numeric, such as the number of droplets in a well, the number of outliers in a well, and the concentration of a well. Selecting multiple rows in the metadata table results in another table that shows the mean and standard error of all the numeric variables in the selected wells. The Results tab also offers a tool to create histograms, density plots, or box plots for any of the numeric metadata variables, as a way to visually explore a metric across the wells in a plate.

Another way to explore a plate object is by plotting the droplets in a scatterplot, with the fluorescence intensity in each of the two channels plotted in each axis. Plotting a plate object shows all available information about the plate, including the droplet clusters, which wells failed, and which wells in a PN/PP assay are deemed positive vs negative. The plotting tool supports dozens of parameters that allow the creation of highly customizable figures that can be tailored to a user's exact needs.

The Results tab also offers a way to see the droplet data in a tabular format, by showing the two fluorescence amplitudes of every droplet along with the droplet's cluster assignment. While this information is not useful for a user to inspect visually because of the sheer amount of information contained in the table, it is useful mainly for the purposes of exporting it and analyzing the droplets and their clusters using a different program.

All forms of results can be downloaded by the user. Any tables that are available in the Results tab are available as CSV files, and any figures generated can be saved as PNG image files.

3.4.3 Availability

The *ddpcr* web app is freely available online at http://daattali.com/shiny/ddpcr/. The online application 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 not to upload any data to a web server for security reasons. To accommodate this need, the web application can also be launched locally on a user's personal machine by running the `launch()` command from the *ddpcr* package.

Chapter 4: Case study: *BRAF*-V600 mutations

4.1 Introduction

To demonstrate its utility, the *ddpcr* software tool was used to analyze ddPCR data from a novel ddPCR assay developed by Bidshahri et al. [1] The ddPCR assay detects mutations in a specific codon in the v-Raf murine sarcoma viral oncogene homolog B (*BRAF*) gene, which are prognostic of several different cancers. Applying automated droplet gating on PN/PP-type ddPCR data for genomic DNA (gDNA) isolated from patient tissue specimens (biopsies) can be used to make clinically actionable decisions regarding the best course of treatment of those cancer patients. With the use of the *ddpcr* tool, such droplet gating is possible in an objective and reproducible manner.

4.2 Mutated *BRAF* in cancer

BRAF is a human proto-oncogene that codes for the B-Raf protein. B-Raf is a 766-amino acid protein kinase that constitutes a key component of the mitogen-activated protein kinase (MAPK) pathway – a chain of reactions controlling cell proliferation. In healthy cells, the B-Raf MAPK pathway is activated only when epidermal growth factor (EGF) binds to the epidermal growth factor receptor (EGFR) and promotes phosphorylation of downstream effectors. Under normal conditions, the pathway is activated only when the growth signal is received. However, there are several mutations that can occur in the *BRAF* gene and cause the gene product B-Raf protein to become constitutively active – transmitting signal down the MAPK pathway regardless of whether or not EGF is bound to EGFR. These mutations therefore promote uncontrolled cell growth, which can develop into cancerous tumours.

Most of these oncogenic mutations in *BRAF* are nucleotide substitutions in the V600 codon, which codes for valine in the wild-type (normal) form of human B-Raf. The most commonly observed substitution is a *BRAF*-V600E mutation (valine gets substituted for glutamic acid), but several other substitutions are also seen in that codon. Mutated (MT) *BRAF*-V600 alleles have been implicated in the pathogenesis of several types of cancers and have been found in

approximately 10% of colorectal (CRC) [37], 45% of papillary thyroid (PTC) [38], and 50% of melanoma tumours [39].

CRC patients can be treated with anti-EGFR antibodies that block activation of the MAPK pathway, thereby limiting the otherwise uncontrolled cell growth. However, these drugs are only effective for patients with wild-type (WT) *BRAF*. Patients harbouring a MT *BRAF* do not benefit from anti-EGFR drugs and generally require alternate treatment. For instance, the Food and Drug Administration recently approved vemurafenib [40] and dabrafenib [41] as treatments for melanoma patients with MT *BRAF* V600E. These drugs are MT B-Raf inhibitors that bind to a B-Raf harbouring a V600 mutation and hinder its cell proliferation activity. Similarly, research suggests that in PTC, disease progression and relapse rate are correlated with MT *BRAF*-V600, and hence *BRAF* status can be a prognostic biomarker [42]. Therefore, correct detection of MT *BRAF* is crucial for determining the proper course of treatment for cancers in which *BRAF* plays an integral role.

4.3 The wild-type negative ddPCR assay for detecting *BRAF* V600 mutations

Bidshahri et al. [1] have recently developed a ddPCR assay capable of differentiating MT from WT *BRAF*-V600 in gDNA recovered from tumour biopsies. This assay uses two hydrolysis probes added to the ddPCR mix: a consensus probe and a WT-specific probe. The consensus probe uses a FAM reporter dye to detect amplification of a highly conserved sequence downstream of the V600 mutation hot-spot on the *BRAF* gene. The consensus probe therefore binds to both MT and WT *BRAF* alleles. The WT-specific probe uses a HEX reporter dye and is engineered to bind across the V600 codon in such a way that it will only hybridize to WT *BRAF*. Therefore, any MT *BRAF* allele hybridizes to the FAM-labeled consensus probe only, while WT *BRAF* alleles hybridize to both probes. Consequently, after PCR amplification, droplets containing MT *BRAF* can be distinguished from droplets with WT *BRAF* based on their FAM and HEX fluorescence. It is a PN/PP-type assay where droplet positive in both fluorescence channels contain WT BRAF, while droplets emitting strong FAM fluorescence but weak HEX fluorescence contain MT BRAF. Figure 27 demonstrates how the assay works.

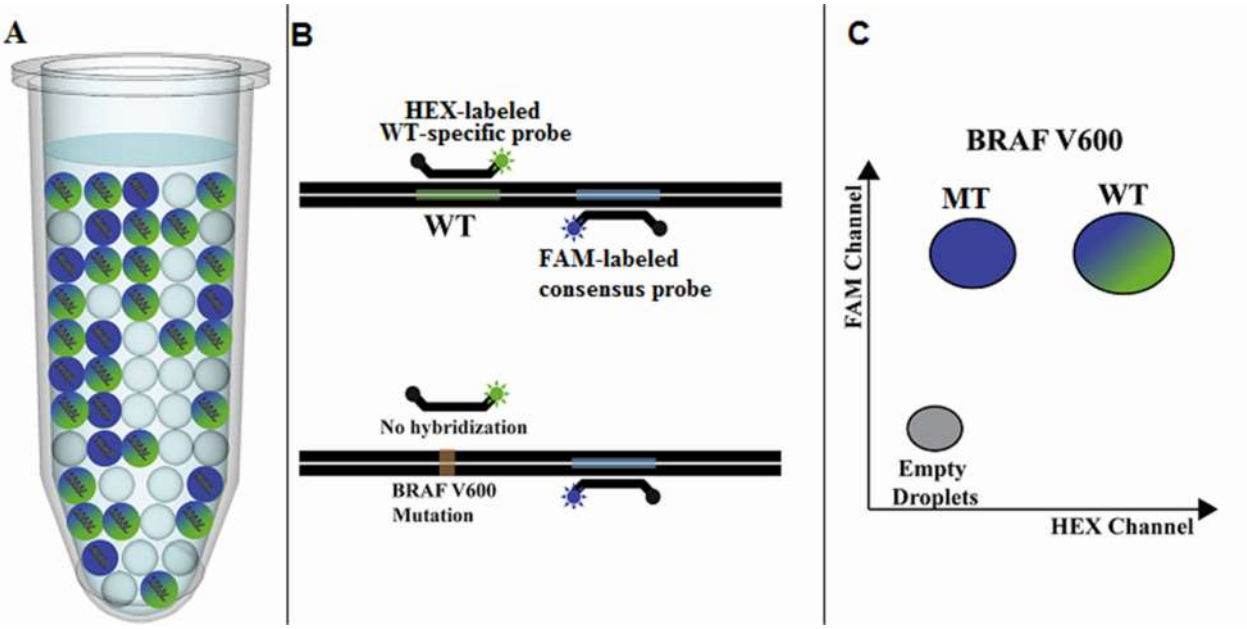

Figure 27 – Wild-type negative assay for detecting *BRAF*-V600 mutations. (A) After ddPCR droplet partitioning, each droplet contains either no *BRAF* template (grey), mutant *BRAF* (blue), or wild-type *BRAF* (blue-green). (B) A FAM-labeled consensus probe hybridizes to any *BRAF* allele downstream of the V600 codon, and a HEX-labeled WT-specific probe binds only to wild-type *BRAF* across the V600 codon. (C) The fluorescence reading of the droplets can be visualized in a scatterplot. The droplets containing wild-type *BRAF* segregate in the FAM^+HEX^+ quadrant and droplets with mutant *BRAF* form another cluster in the FAM^+HEX^- quadrant.

This ddPCR assay is a FAM-positive PN/PP-type assay. The PP droplets (FAM^+HEX^+) initially contain a copy of the WT *BRAF* gene and will be called "WT droplets" for the remainder of this chapter. Similarly, the PN droplets (FAM^+HEX^-) initially contain a MT *BRAF* copy and will be henceforth described as "MT droplets". Any double-negative (empty) droplets are droplets that do not contain any copies of the *BRAF* gene.

By gating the droplets in this assay, it is possible to count the number of droplets with MT and WT *BRAF*. From that information, the mutant frequency (MF) within the sample can be easily determined using the equation

$$MF = \frac{number\ of\ MT\ alleles\ (droplets)}{number\ of\ MT\ alleles + number\ of\ WT\ alleles} \qquad (3)$$

The mutation frequency of a sample shows what proportion of the *BRAF* copies in the sample DNA is mutated at V600. In order to make clinically actionable decisions, the *BRAF* status of a sample needs to have a binary classification of either MT-positive or WT. A sample with a low MF (i.e., below the MT *BRAF* limit of detection (LOD)) is scored as WT, while that with a statistically significant MF is scored MT-positive for clinical purposes. Several assays using various different PCR-based technologies have been developed for this purpose, and each offers a unique LOD for MT *BRAF* detection. For example, the Cobas® 4800 BRAF V600 Mutation Test (Roche Molecular Diagnostics, Pleasanton, CA) and the BRAF RGQ PCR Kit (Qiagen, Manchester, UK) both score a sample as MT-positive when the measured MF is above 5%.

4.4 Data

Formalin-fixed, paraffin-embedded (FFPE) tumour biopsy samples from a cohort of 41 CRC patients were obtained from Lion's Gate Hospital (North Vancouver, BC, Canada) (with approval from the University of British Columbia Clinical ethics Research Board, certificate number H14-00577). Each patient's tumour was individually verified by a certified pathologist as either MT or WT *BRAF* using an immunohistochemical staining assay (Ventana Medical Systems, Tucson, AZ). The pathologist determined that 16 out of the 41 had WT *BRAF*, while 25 patients were MT *BRAF* positive (Table 1).

86 wells of a 96-well plate were loaded: gDNA isolated from each of the 41 tumour samples was placed into pairs of adjacent wells (replicates), and four additional wells were used as no template control (NTC) by not loading any DNA into them. Each well also contained all the necessary reagents for a ddPCR experiment, including the primers and specific probes designed for the wild-type negative ddPCR assay for detecting *BRAF* V600 mutations. After running this plate through the ddPCR protocol, the droplet fluorescence data was loaded into QuantaSoft and exported into CSV files. The set of CSV files containing the data for this plate is collectively referred to as the dataset. The raw droplet fluorescence data is seen in Figure 85 (Appendix I).

4.5 Results

Since the *BRAF* WT-negative ddPCR assay is a PN/PP assay, the *ddpcr* tool can be used to automatically perform droplet gating on the given dataset and calculate the MF in each well. The *ddpcr* tool can also be used to classify each well as MT-positive or WT using the method described in Section 2.4.4.3.

The dataset was loaded into the *ddpcr* web tool and analyzed using all the default options. The web tool was used instead of the command-line simply because generating customized figures is easier with the interactive tool, although identical results can be obtained using the command-line functions. The results from the automated droplet gating can be seen in Figure 86 (Appendix I), and the full metadata of each well is available in Table 2. The full analysis ran in 67 seconds 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. Results from a few select wells are shown in Figure 28.

The four NTC wells (E11, F11, G11, H11 in Figure 28) were correctly identified as failed wells. These wells are classified as failed because they do not contain any template and droplet gating should not be performed on them. Four other wells (C06, D06, C10, D10 in Figure 28) were also classified as failed, and did not get further analyzed. These four wells have samples from two patients (suggested by the fact that they are adjacent pairs, and verified by looking at the results in Table 2), and they all failed the quality control metrics because they did not have enough non-empty droplets (see Section 2.3.1.4).

Out of the remaining 78 wells, 32 wells were classified as WT (e.g. F04 in Figure 28) and 46 as MT-positive (e.g. F06 in Figure 28). Every pair of wells with gDNA from the same patient gave a consistent *BRAF* status. To assess the clinical utility of the *ddpcr* tool, the *BRAF* classification reported by *ddpcr* for these 78 wells were compared to those determined by a pathologist using an independent method. Except for the two patients that could not be compared because their ddPCR wells failed, complete agreement was observed between the pathologist's results and the *ddpcr*-reported *BRAF* status, as shown in Table 1.

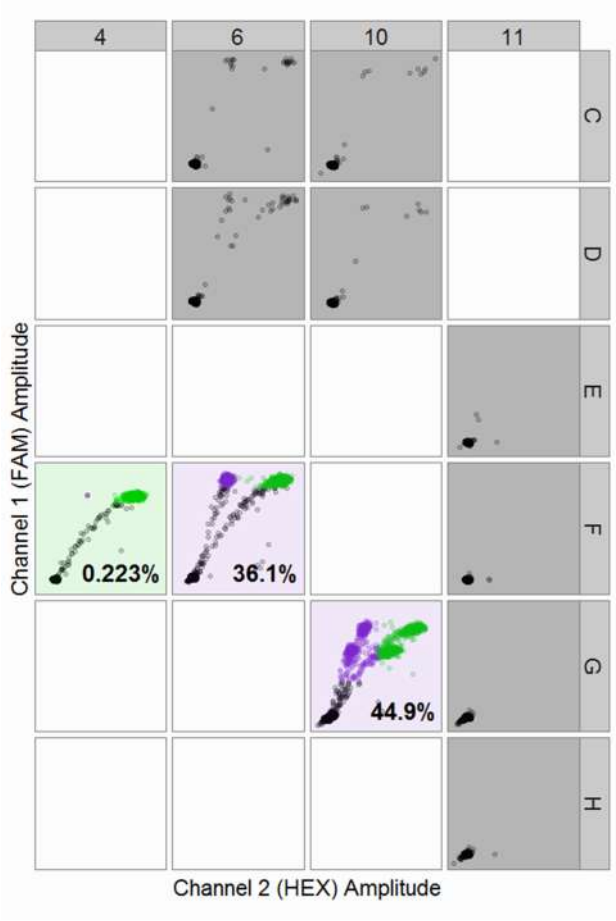

Figure 28 – Results of a few wells from the analysis of CRC patient samples. Wells with a grey background indicate failed wells. Wells E11, F11, G11, H11 are NTC, while wells C06, D06, C10, D10 failed because of insufficient non-empty droplets. Wells with light green background (F04) indicate wild-type *BRAF* while wells with light purple background (F06, G10) indicate mutant *BRAF*. The numbers specify the calculated MF.

For comparison of *ddpcr* against QuantaSoft, droplet gating of the same data using QuantaSoft's automated gating is shown in Figure 87. In many samples (e.g. well C08), QuantaSoft was not able to distinguish between the MT and WT droplets, and grouped all the PN and PP droplets together. In some samples (e.g. well D08), QuantaSoft classified nearly all rain droplets as empty. Overall, QuantaSoft's gating performed visibly worse than *ddpcr* and resulted in droplet gating that are not useful for analysis.

To further assess the accuracy of the droplet gating and MF calculation, a trained ddPCR operator (Roza Bidshahri) independently performed manual droplet gating on all the wells using QuantaSoft. The gating was done solely based on visual inspection of the droplets. After

manual gating, the number of droplets in each cluster was reported by QuantaSoft, and the MF was calculated for each well. After removing all eight wells that failed the automated analysis and therefore did not have comparable results, the MF values calculated for the other 78 wells were compared between the manual analysis and the automated analysis by *ddpcr* (Table 1). As seen in Table 1, all mutant frequencies calculated by *ddpcr* were found to be very similar to the MF calculated after manual analysis, except for well G10. The raw fluorescence data in well G10 (Figure 28) shows that there are four droplet populations instead of two, which suggests that an error occurred in that well, perhaps from splash-back contamination.

A few visual and quantitative tests were carried out on the results to further assess the quality of the automated droplet gating. Figure 29 shows a scatterplot comparing the MF calculated using both methods. If the two methods had perfect agreement with each other, all points would lie exactly on the $y = x$ diagonal line. Since all the points have only minor deviations from the diagonal line except for one point (corresponding to sample G10), the scatterplot suggests that the MF calculated by both methods is almost equal. The Pearson correlation (linear correlation) between the MF values of the two approaches is 0.989, which suggests there is an almost perfect linear correlation between the two approaches.

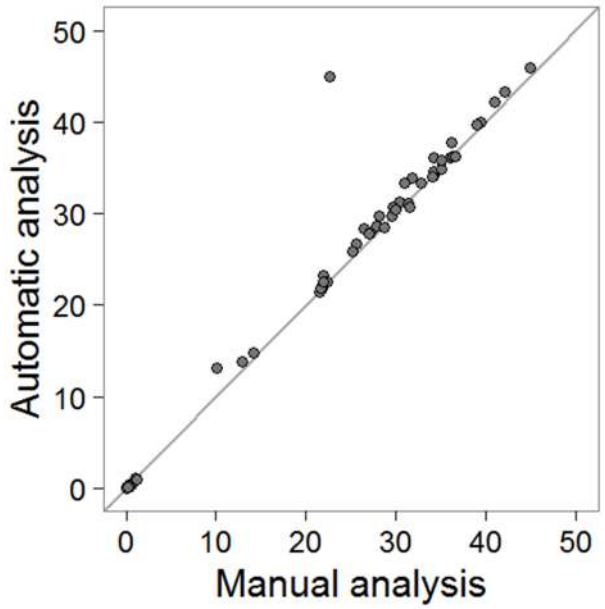

Figure 29 – Comparison of the mutation frequency in each well as calculated automatically by *ddpcr* vs manually by setting custom gates. For reference, the grey line has equation $\boldsymbol{y = x}$; points along the line signify perfect agreement between manual and automatic calculation.

When comparing the manual analysis to the automated analysis, it is important to note that both methods only provide approximations to the true MF, and none of them are a gold standard. However, while the manual analysis is not meant to be 100% accurate, it still provides clinically useful results. Therefore, if the MF value calculated automatically for a given sample does not match exactly the one calculated manually, it can still be considered correct if the difference is small enough to be practically trivial. The definition of "practically trivial" depends on the context; in the case of measuring *BRAF* mutation frequencies, it is reasonable to claim that a mean difference of 1% between the reported MF of the two approaches is clinically insignificant, so if the two methods produce results that are no more different than 1%, they can be regarded as practically in agreement with each other. It is important to note also that in cases where the two values are similar but not identical, it is impossible to know which of the two methods produced a more accurate result. Bland-Altman plots are commonly used in biostatistics to visualize the difference between two quantitative methods to analyze their agreement with each other [43]. For each sample, the difference between the two methods is plotted against the mean of the two methods. Figure 30 shows a Bland-Altman plot comparing the MF values calculated manually and automatically. The plot shows that there is an agreement between the two methods mostly within 1%.

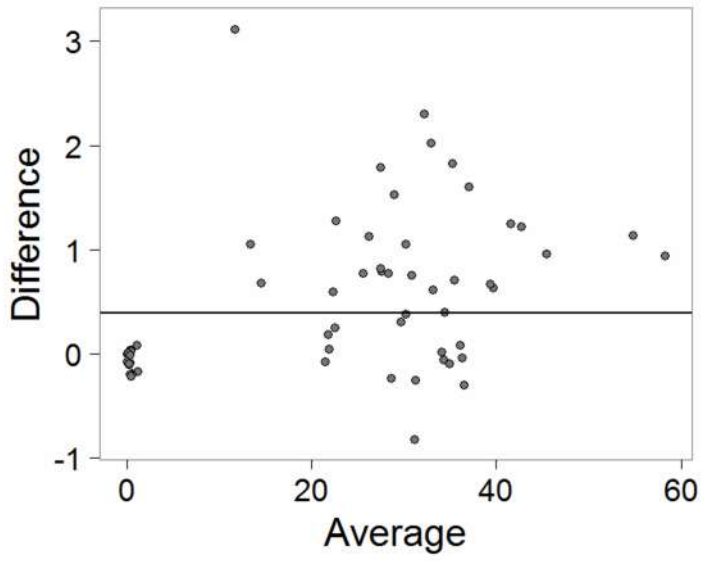

Figure 30 – Bland-Altman plot showing the difference vs average of the reported MF values by the manual and automatic analysis approaches. Bland-Altman plots are used in biostatistics to compare the agreement between two measurement techniques. The solid line represents the mean difference between the measurements on all samples. Well G10 is excluded from the plot.

The mean difference between the two methods is 0.39, which means that on average the value reported by the automatic analysis is 0.39% higher than the manual analysis. To test whether or not this mean difference is significant, a regular test of means (ie. via a paired t-test or paired Mann-Whitney U test) is not applicable because it only tests if two means are equal but not if they are clinically equivalent. Instead, a paired two one-sided test (TOST) of equivalence is performed. A TOST is used to determine whether the measurements between two different methods are close enough to be considered equivalent for practical matters by performing two one-sided tests that test if the differences fall significantly outside a region of similarity [44]. Since performing a TOST involves two separate tests, there are two null hypotheses to be tested. Defining the region of similarity to be 1% (ie. a difference of 1% between the tests is considered equivalent) results in the following two null hypotheses: $H_{0\_1}: \mu \geq 1$ and $H_{0\_2}: \mu \leq 1$, where μ is the mean difference between the two methods. Since the distribution of differences between the methods are not normally distributed (Shapiro-Wilk test p-value of 1.5×10^{-17}), Mann-Whitney U tests are used instead of t-tests to test the two null hypotheses. The p-values for the one-sided tests are 4.3×10^{-8} and 1.8×10^{-14} respectively, so both null hypotheses are rejected. By rejecting both null hypotheses, we can conclude that $-1 \leq \mu \leq 1$, which suggests the automatic analysis is in agreement with the manual analysis to within 1% similarity.

Chapter 5: Conclusion

Droplet digital PCR is a new technology that holds great promise in clinical diagnostics. It is increasingly popular due to its digital nature, which provides more accurate nucleic acid quantification and higher sensitivity than traditional qPCR. The growth of ddPCR usage, however, has been slowed by the lack of tools to reliably analyze data produced by ddPCR experiments.

The most widely used tool for ddPCR data analysis is QuantaSoft, a proprietary program provided by the creators (BioRad Inc.) of the QX100 and QX200 ddPCR equipment that allows users to analyze data either manually or automatically. However, this program often produces poor results when automatic analysis is selected. To date, only two third-party software tools have been developed for ddPCR analysis, and both tools operate only on single-channel ddPCR experiments. As a result, most ddPCR users working with two-channel assay rely on completely manual analysis simply because software for ddPCR analysis is severely lacking. With dual-channel ddPCR experiments becoming increasingly more common, there is a clear need for software to able to automatically and objectively analyze those datasets.

The current work describes and validates algorithmic methods for automatic analysis of two-channel ddPCR data. These methods have been implemented in R as an R package and are also available through a web interface. Any ddPCR user working with a two-channel experiment can use the developed tool in their analysis.

To demonstrate clinical utility of the developed algorithm, the web tool was used to classify tumour samples from CRC patients as containing either mutant or wild-type *BRAF*. Comparing the results obtained from the automatic analysis to those from two independent methods has verified that the *ddpcr* tool automatically produces results consistent with other more time-consuming benchmark methods.

Bibliography

[1] R. Bidshahri, D. Attali, K. Fakhfakh, K. McNeil, A. Karsan, J. Won, R. Wolber, J. Bryan, C. Hughesman and C. Haynes, "Quantitative Detection and Resolution of BRAF V600 Status in Colorectal Cancer Using Droplet Digital PCR and a Novel Wild-Type Negative Assay," *The Journal of Molecular Diagnostics,* vol. 18, no. 2, pp. 190-204, 2016.

[2] B. Hindson, K. Ness, D. Masquelier, P. Belgrader, N. Heredia, A. Makarewicz, I. Bright, M. Lucero, A. Hiddessen and T. Legler, "High-throughput droplet digital PCR system for absolute quantitation of DNA copy number," *Analytical chemistry,* vol. 83, no. 22, pp. 8604-8610, 2011.

[3] J. M. Bartlett and D. Stirling, "A short history of the polymerase chain reaction," *PCR Protocols,* pp. 3-6, 2003.

[4] C. Heid, J. Stevens, K. J. Livak and M. Williams, "Real time quantitative PCR," *Genome research,* vol. 6, no. 10, pp. 986-994, 1996.

[5] S. Dhanasekaran, M. Doherty and J. Kenneth, "Comparison of different standards for real-time PCR-based absolute quantification," *Journal of immunological methods,* vol. 354, no. 1, pp. 34-39, 2010.

[6] R Core Team, "R: A Language and Environment for Statistical Computing," R Foundation for Statistical Computing, Vienna, Austria, 2015.

[7] L. Roewer, "DNA fingerprinting in forensics: past, present, future," *Investigative genetics,* vol. 4, no. 1, p. 1, 2013.

[8] S. D. Pena and R. Chakraborty, "Paternity testing in the DNA era," *Trends in Genetics,* vol. 10, no. 6, pp. 204-209, 1994.

[9] E. R. Mardis, "Next-generation DNA sequencing methods," *Annual Review of Genomics and Human Genetics,* vol. 9, pp. 387-402, 2008.

[10] S. Yang and R. E. Rothman, "PCR-based diagnostics for infectious diseases: uses, limitations, and future applications in acute-care settings," *The Lancet infectious diseases,* vol. 4, no. 6, pp. 337-348, 2004.

[11] R. Don, P. Cox, B. Wainwright, K. Baker and J. Mattick, "'Touchdown'PCR to circumvent

spurious priming during gene amplification," *Nucleic acids research,* vol. 19, no. 14, p. 4008, 1991.

[12] N. Paul, J. Shum and T. Le, "Hot start PCR," *RT-PCR Protocols: Second Edition,* pp. 301-318, 2010.

[13] W. Stemmer, A. Crameri, K. Ha, T. Brennan and H. Heyneker, "Single-step assembly of a gene and entire plasmid from large numbers of oligodeoxyribonucleotides," *Gene,* vol. 164, no. 1, pp. 49-53, 1995.

[14] G. Schochetman, C.-Y. Ou and W. K. Jones, "Polymerase chain reaction," *The Journal of infectious diseases,* vol. 158, no. 6, pp. 1154-1157, 1988.

[15] J. Jozefczuk and J. Adjaye, "Quantitative Real-Time PCR-Based Analysis of Gene Expression," *Methods in enzymology,* vol. 500, pp. 99-109, 2011.

[16] M.-A. Fraiture, P. Herman, I. Taverniers, M. De Loose, D. Deforce and N. Roosens, "Current and new approaches in GMO detection: challenges and solutions," *BioMed research international,* vol. 2015, 2015.

[17] P. S. Bernard and C. T. Wittwer, "Real-time PCR technology for cancer diagnostics," *Clinical Chemistry,* vol. 48, no. 8, pp. 1178-1185, 2002.

[18] K. Nielsen, H. Mogensen, J. Hedman, H. Niederstatter, W. Parson and N. Morling, "Comparison of five DNA quantification methods," *Forensic Science International: Genetics,* vol. 2, no. 3, pp. 226-230, 2008.

[19] P. Sykes, S. Neoh, M. Brisco, E. Hughes, J. Condon and A. Morley, "Quantitation of targets for PCR by use of limiting dilution," *Biotechniques,* vol. 13, no. 3, pp. 444-449, 1992.

[20] G. Pohl and I.-M. Shih, "Principle and applications of digital PCR," *Expert review of molecular diagnostics,* vol. 4, no. 1, pp. 41-47, 2004.

[21] M. Baker, "Digital PCR hits its stride," *Nature Methods,* vol. 9, no. 6, pp. 541-544, 2012.

[22] Bio-Rad, "Droplet Digital PCR Applications Guide," [Online]. Available: http://www.bio-rad.com/webroot/web/pdf/lsr/literature/Bulletin_6407.pdf. [Accessed 06 04 2016].

[23] D. Pretto, D. Maar, C. Yrigollen, J. Regan and F. Tassone, "Screening newborn blood spots for 22q11. 2 deletion syndrome using multiplex droplet digital PCR," *Clinical chemistry,* vol. 61, no. 1, pp. 182-190, 2015.

[24] C. Roberts, W. Jiang, J. Jayaraman, J. Trowsdale, M. Holland and J. Traherne, "Killer-cell Immunoglobulin-like Receptor gene linkage and copy number variation analysis by droplet digital PCR," *Genome Med,* vol. 6, no. 3, p. 20, 2014.

[25] J. Beaver, D. Jelovac, S. Balukrishna, R. Cochran, S. Croessmann, D. Zabransky, H. Y. Wong, P. V. Toro, J. Cidado and B. Blair, "Detection of cancer DNA in plasma of patients with early-stage breast cancer," *Clinical Cancer Research,* vol. 20, no. 10, pp. 2643-2650, 2014.

[26] V. Taly, D. Pekin, L. Benhaim, S. Kotsopoulos, D. Le Corre, X. Li, I. Atochin, D. Link, A. Griffiths and K. Pallier, "Multiplex picodroplet digital PCR to detect KRAS mutations in circulating DNA from the plasma of colorectal cancer patients," *Clinical chemistry,* vol. 59, no. 12, pp. 1722-1731, 2013.

[27] J. Pavsi, J. Zel and M. Milavec, "Assessment of the real-time PCR and different digital PCR platforms for DNA quantification," *Analytical and bioanalytical chemistry,* vol. 408, no. 1, pp. 107-121, 2016.

[28] M. Strain, S. Lada, T. Luong, S. Rought, S. Gianella, V. Terry, C. Spina, C. Woelk and D. Richman, "Highly precise measurement of HIV DNA by droplet digital PCR," *PloS one,* vol. 8, no. 4, p. e55943, 2013.

[29] M. Jones, J. Williams, K. Gartner, R. Phillips, J. Hurst and J. Frater, "Low copy target detection by Droplet Digital PCR through application of a novel open access bioinformatic pipeline,'definetherain'," *Journal of virological methods,* vol. 20, no. 2, pp. 46-53, 2014.

[30] W. Trypsteen, M. Vynck, J. De Neve, P. Bonczkowski, M. Kiselinova, E. Malatinkova, K. Vervisch, O. Thas, L. Vandekerckhove and W. De Spiegelaere, "ddpcRquant: threshold determination for single channel droplet digital PCR experiments," *Analytical and bioanalytical chemistry,* vol. 407, no. 19, pp. 5827-5834, 2015.

[31] C. Milbury, Q. Zhong, J. Lin, M. Williams, J. Olson, D. Link and B. Hutchison, "Determining lower limits of detection of digital PCR assays for cancer-related gene mutations," *Biomolecular detection and quantification,* vol. 1, no. 1, pp. 8-22, 2014.

[32] P. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 1, no. 1, pp. 73-79,

2011.

[33] S. Seo, "A review and comparison of methods for detecting outliers in univariate data sets," M.S. thesis, University of Pittsburgh, 2006.

[34] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society,* pp. 683-690, 1991.

[35] W. Chang, J. Cheng, J. Allaire, Y. Xie and J. McPherson, "shiny: Web Application Framework for R," R package version 0.13.1.9000, http://shiny.rstudio.com, 2016.

[36] H. Wickham, "ggplot2: Elegant Graphics for Data Analysis," Springer-Verlag New York, http://had.co.nz/ggplot2/book, 2009.

[37] C. Vaughn, S. ZoBell, L. Furtado, C. Baker and W. Samowitz, "Frequency of KRAS, BRAF, and NRAS mutations in colorectal cancer," *Genes, Chromosomes and Cancer,* vol. 50, no. 5, pp. 307-312, 2011.

[38] M. Yarchoan, V. LiVolsi and M. Brose, "BRAF mutation and thyroid cancer recurrence," *Journal of Clinical Oncology,* vol. 33, no. 1, pp. 7-8, 2015.

[39] P. Ascierto, J. Kirkwood, J.-J. Grob, E. Simeone, A. Grimaldi, M. Maio, G. Palmieri, A. Testori, F. Marincola and N. Mozzillo, "The role of BRAF V600 mutation in melanoma," *J Transl Med,* vol. 10, no. 85, 2012.

[40] P. Chapman, A. Hauschild, C. Robert, J. Haanen, P. Ascierto, J. Larkin, R. Dummer, C. Garbe, A. Testori and M. Maio, "Improved survival with vemurafenib in melanoma with BRAF V600E mutation," *New England Journal of Medicine,* vol. 364, no. 26, pp. 2507-2516, 2011.

[41] A. Menzies and G. Long, "Dabrafenib and trametinib, alone and in combination for BRAF-mutant metastatic melanoma," *Clinical Cancer Research,* vol. 20, no. 8, pp. 2035-2043, 2014.

[42] R. Tufano, G. Teixeira, J. Bishop, K. Carson and M. Xing, "BRAF mutation in papillary thyroid cancer and its value in tailoring initial treatment: a systematic review and meta-analysis," *Medicine,* vol. 91, no. 5, pp. 274-286, 2012.

[43] M. Bland and D. Altman, "Statistical methods for assessing agreement between two methods of clinical measurement," *The Lancet,* vol. 327, no. 8476, pp. 307-310, 1986.

[44] A. Munk, "Testing Statistical Hypotheses of Equivalence," *Journal of the American Statistical Association,* vol. 99, no. 465, p. 293, 2004.

[45] F. Di Nicolantonio, M. Martini, F. Molinari, A. Sartore-Bianchi, S. Arena, P. Saletti, Mazzucchelli, L. Mazzucchelli, M. Frattini and S. Siena, "Wild-type BRAF is required for response to panitumumab or cetuximab in metastatic colorectal cancer," *Journal of Clinical Oncology,* vol. 26, no. 35, pp. 5705-5712, 2008.

Appendices

Appendix A Datasets used for testing

Throughout the appendices, there are many examples of wells from different ddPCR plates that were used for developing and testing the algorithms. There are six different datasets that will be referred to:

- **CRC-41**: FFPE tumour samples from 41 colorectal cancer patients. This is the dataset used in the use case study in Chapter 4. The internal name for this dataset is 2-26-2014-BRAFWTNEGASSAY-FFPEDNA-CRC-1-41.
- **CRC-REPEAT**: A repeated experiment of CRC-41 performed with a higher input concentration, using a subset of 16 out of the 41 patients. The internal name for this dataset is 3-5-2014_BRAFWTNEGASSAY_FFPEDNA_CRCRepeat.
- **PLASMID**: *BRAF*-V600K synthetic plasmids and a YUMAC cell line that was homozygous for V600K. The internal name for this dataset is 2013-06-26_brafv600k_plasmid_cellline_lod.
- **YUMAC**: A run on the YUMAC cell line (homozygous for V600K) blends at various frequencies were created by combining WT DNA and MT YUMAC DNA. The internal name for this dataset is 2013-8-13_yumac-cell-line_lod.
- **BRAFV600K**: Same YUMAC dataset, but a synthetic plasmid DNA was used. The internal name for this dataset is 2013-8-8_brafv600k_lod.
- **THYROID**: DNA extracted from 7 thyroid FFPE scrolls. The internal name for this dataset is 2014-06-06_BRAFWTNEGASSAY_FFPEThyroidscrolls.

Appendix B Number of droplets generated per well across different plates

According to Bio-Rad's ddPCR manual, the droplet generator creates 20,000 droplets per well, but observing ddPCR experiment results reveals that typically 12,000 to 15,000 droplets are generated in each well, though this number varies between each run. The following six box plots show the distribution of number of droplets per well in six different ddPCR experiments.

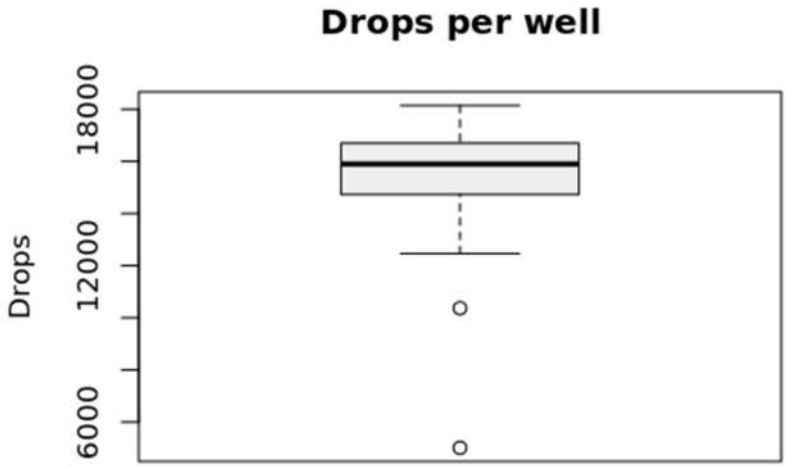

Figure 31 – Droplets per well in CRC-41 dataset

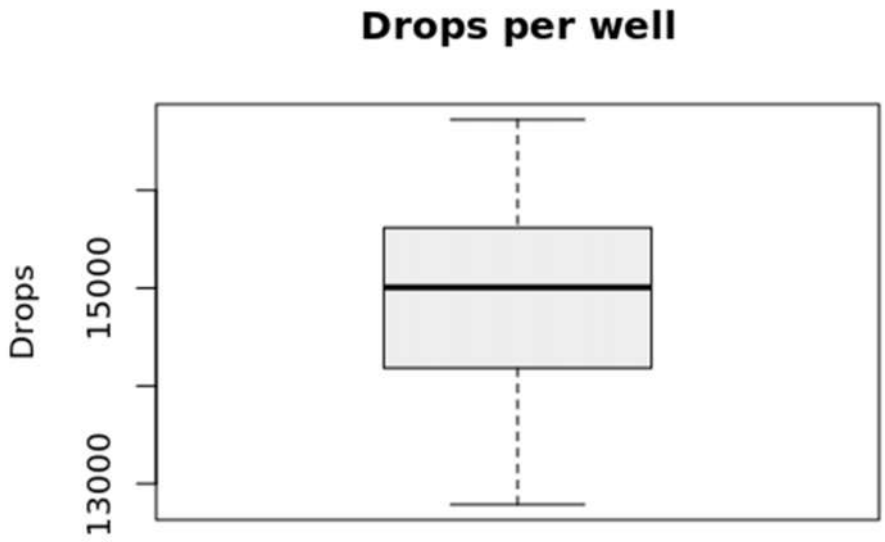

Figure 32 – Droplets per well in CRC-REPEAT dataset

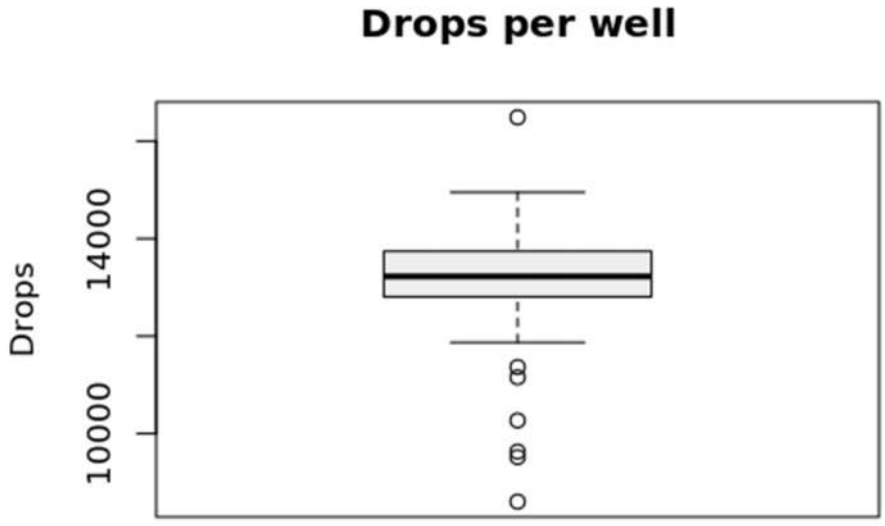

Figure 33 – Droplets per well in PLASMID dataset

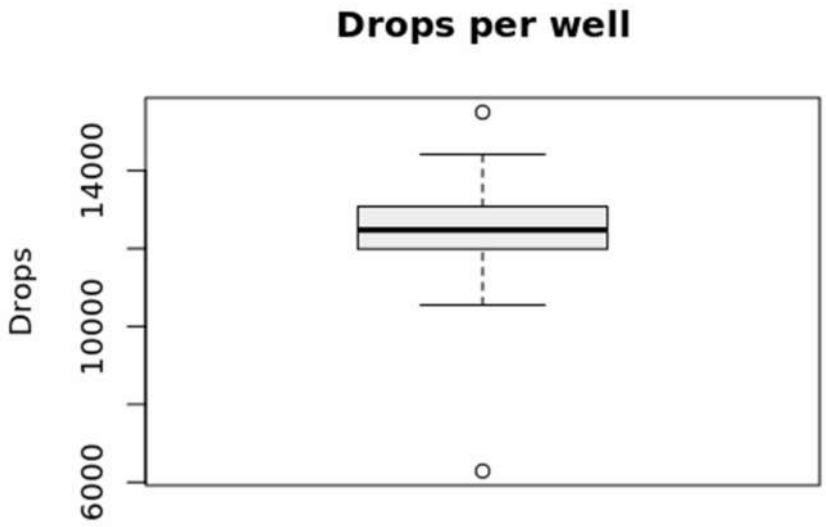

Figure 34 – Droplets per well in BRAFV600K dataset

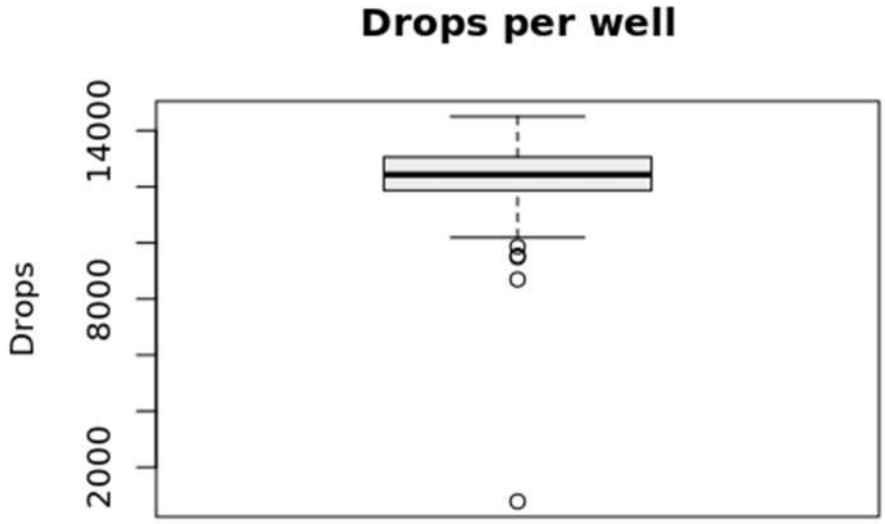

Figure 35 – Droplets per well in YUMAC dataset

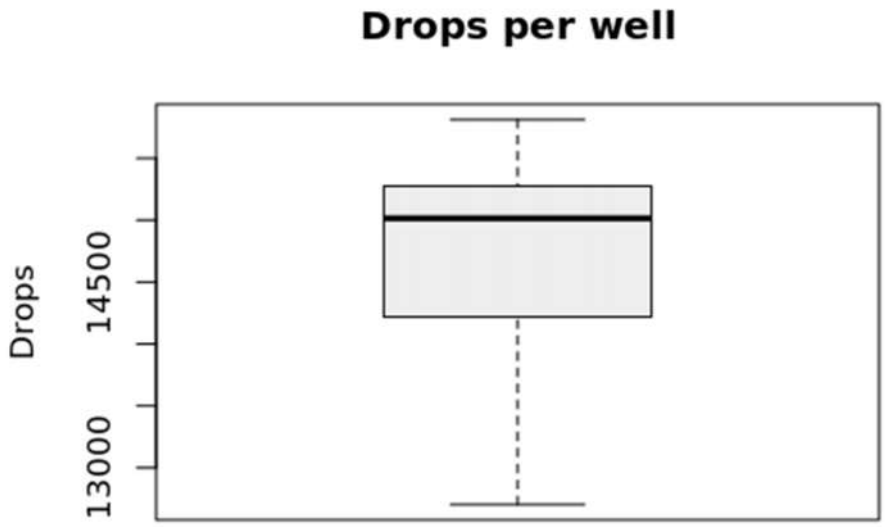

Figure 36 – Droplets per well in THYROID dataset

Appendix C Applying outlier detection algorithm to different datasets

The outlier detection step has been applied to different ddPCR plates. Wells with grey backgrounds are failed wells. Orange droplets are droplets identified as outliers.

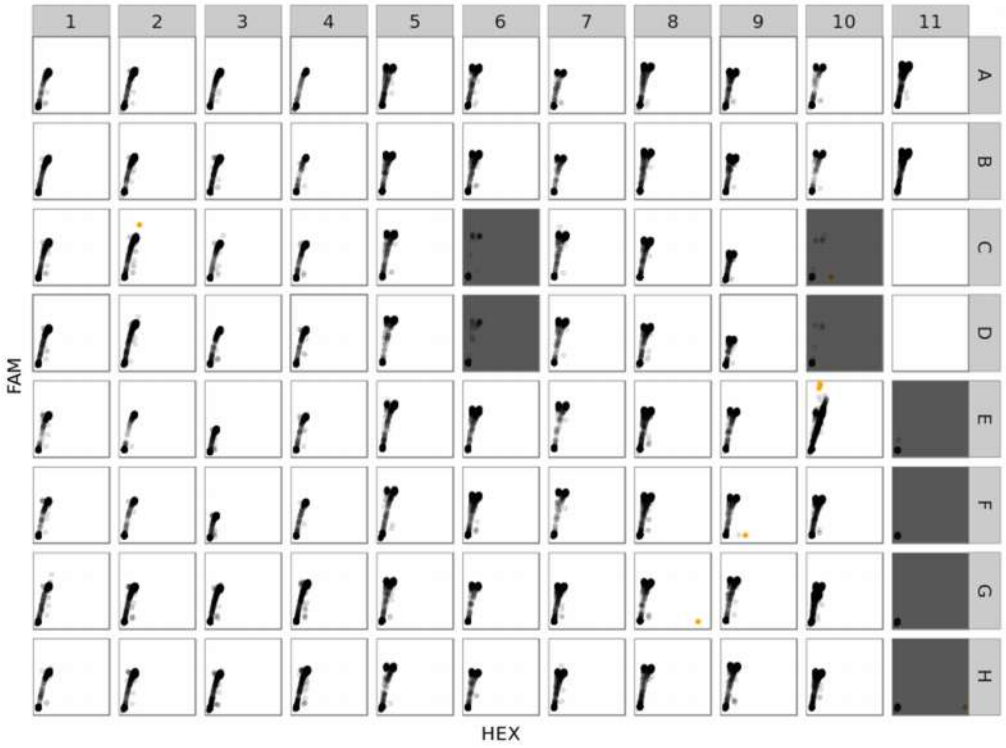

Figure 37 – Outlier decetion CRC-41 dataset

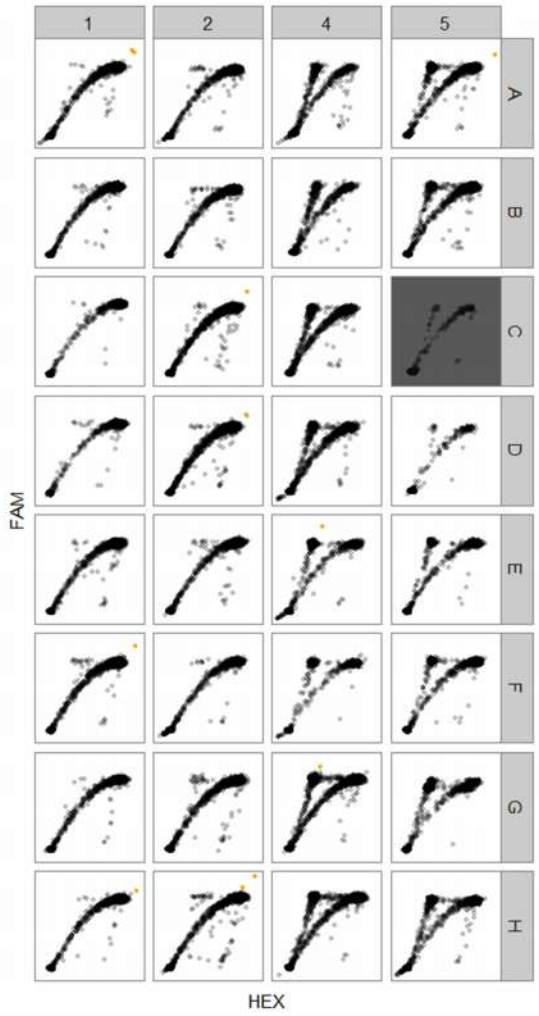

Figure 38 – Outlier detection in CRC-REPEAT dataset

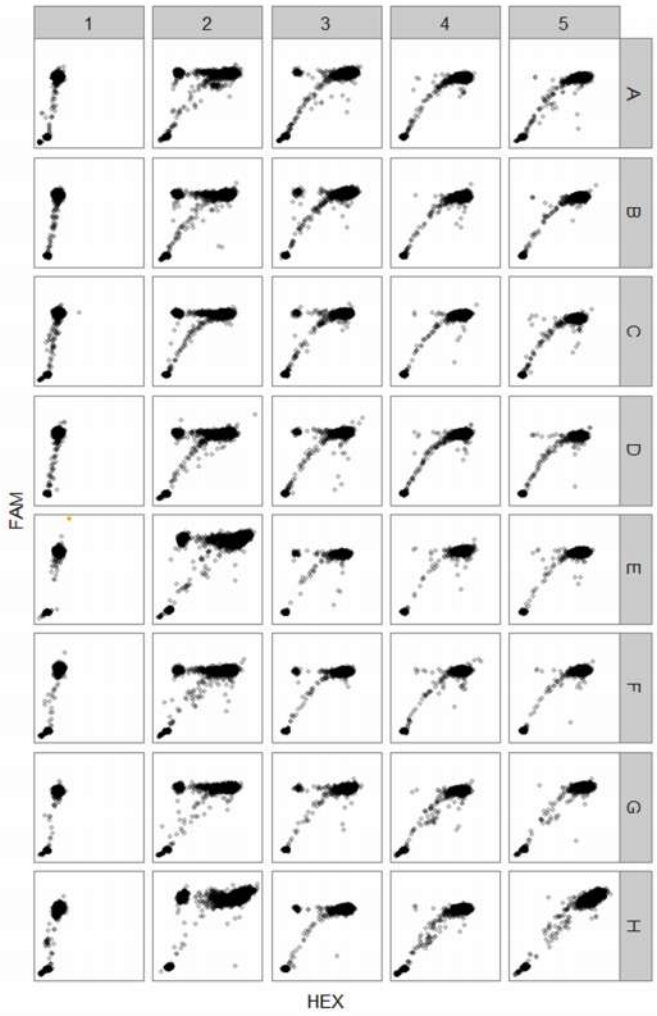

Figure 39 – Outlier detection in PLASMID dataset

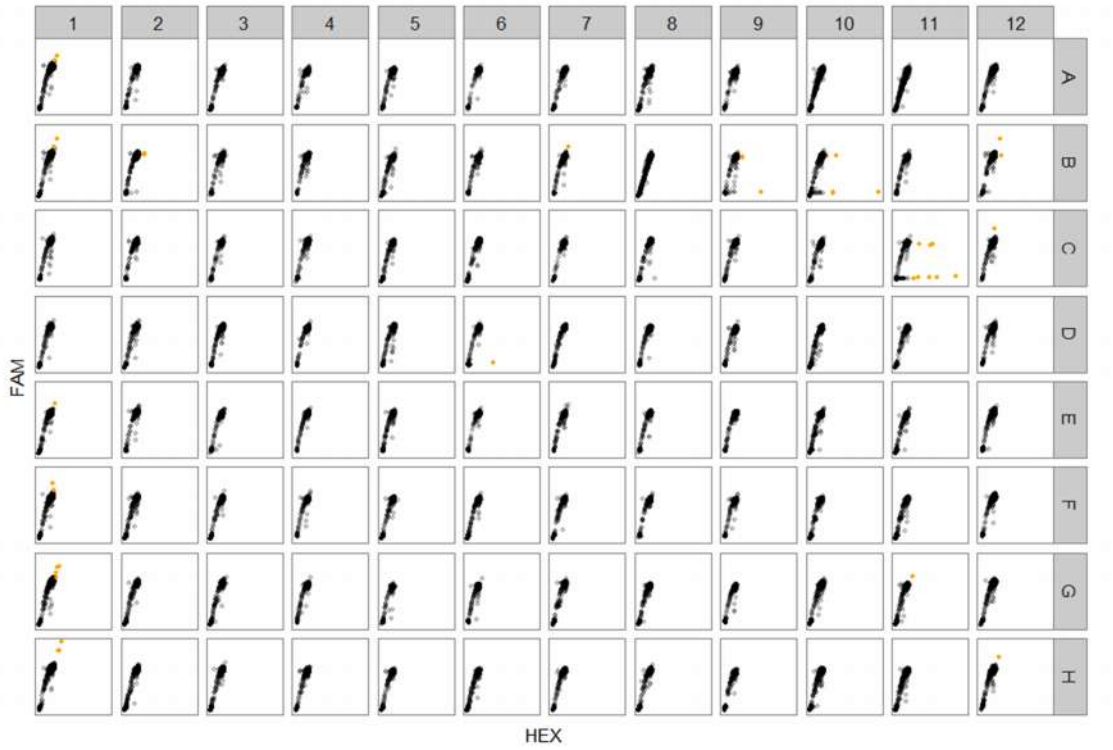

Figure 40 – Outlier detection in BRAFV600K dataset

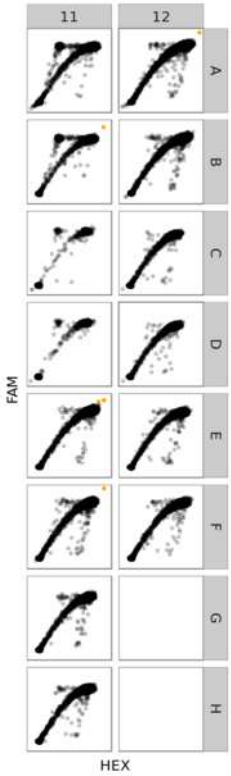

Figure 41 – Outlier detection in THYROID dataset

Appendix D Empty droplet detection in different wells

The following figures show the results of identifying empty droplets in three wells from six different ddPCR plates. The droplets classified as empty are marked in red.

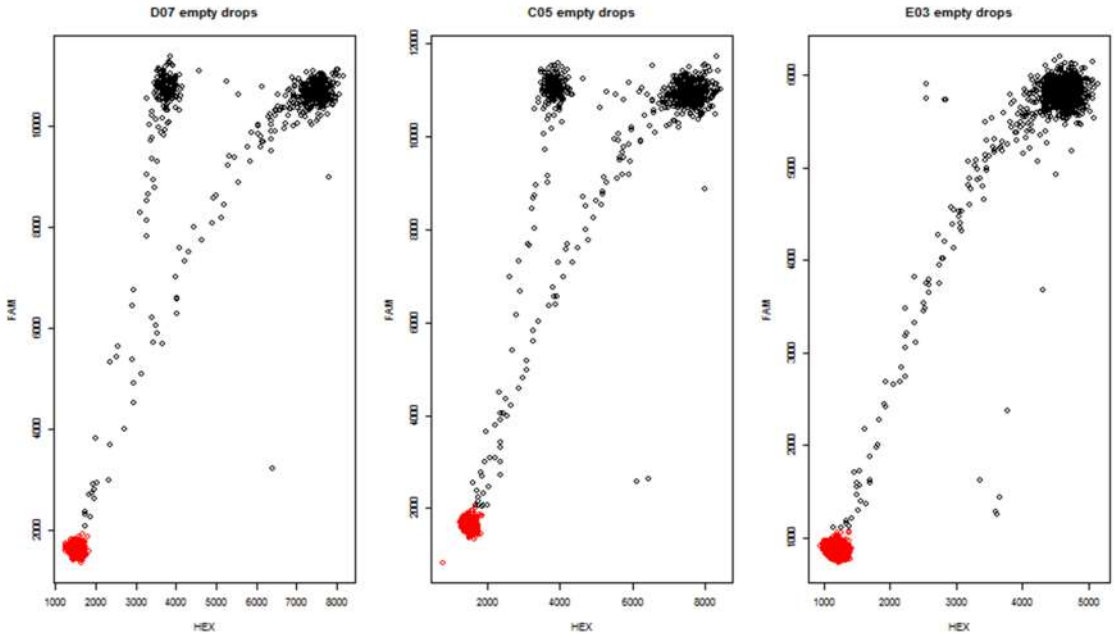

Figure 42 – Empty droplet identification in CRC-41 dataset

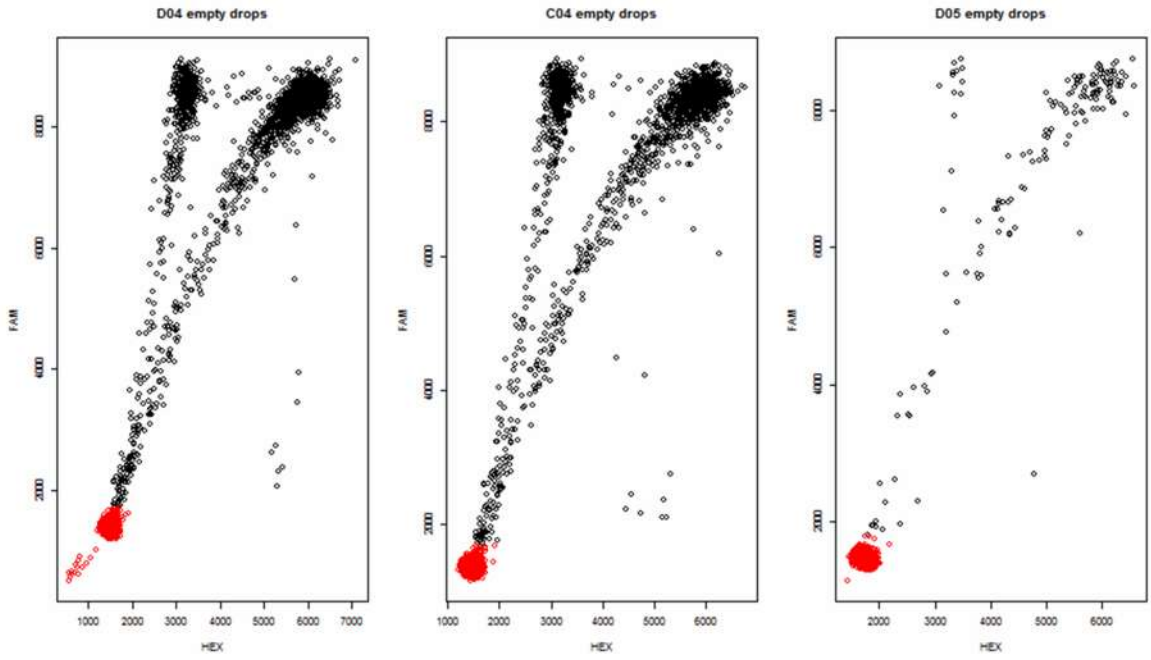

Figure 43 – Empty droplet identification in CRC-REPEAT dataset

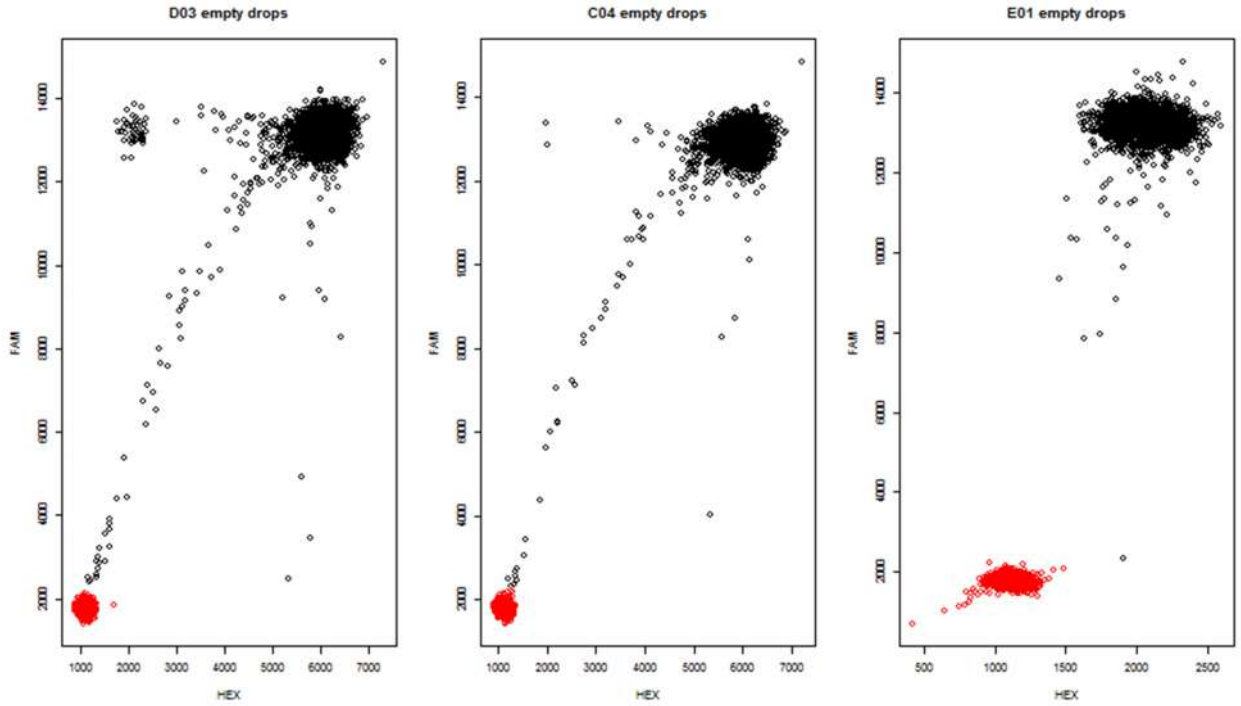

Figure 44 – Empty droplet identification in BRAFV600K dataset

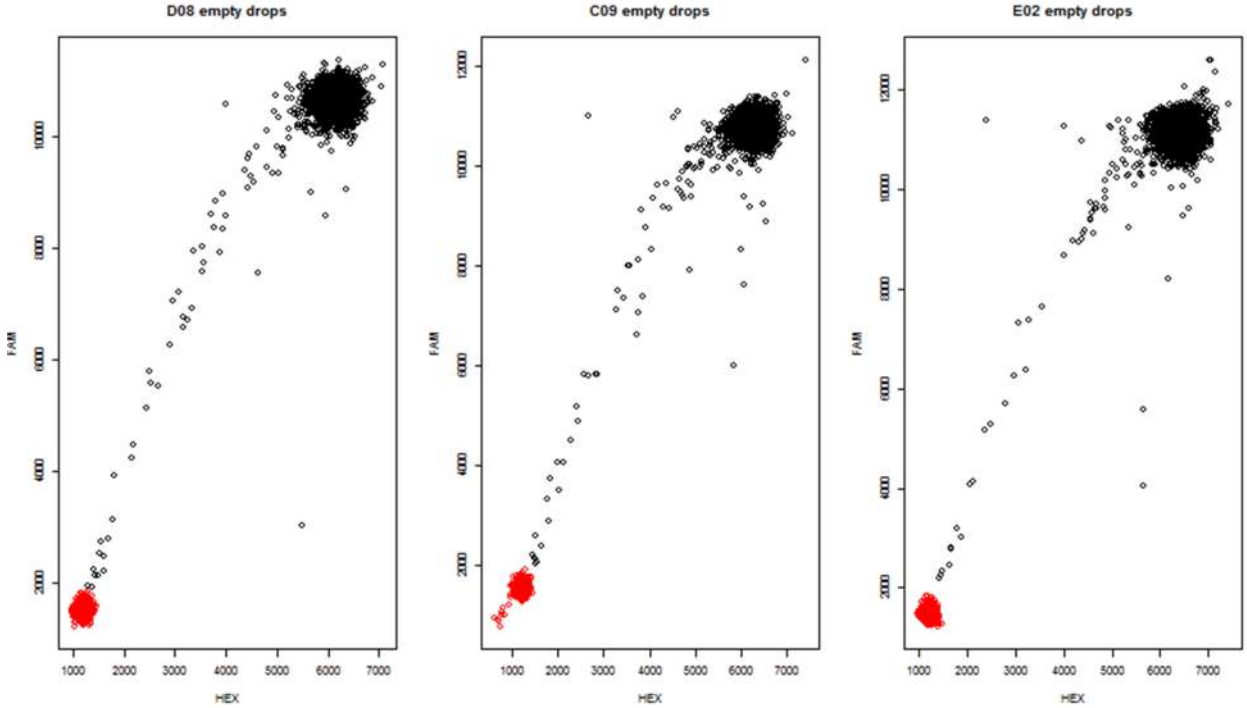

Figure 45 – Empty droplet identification in BRAFV600K dataset

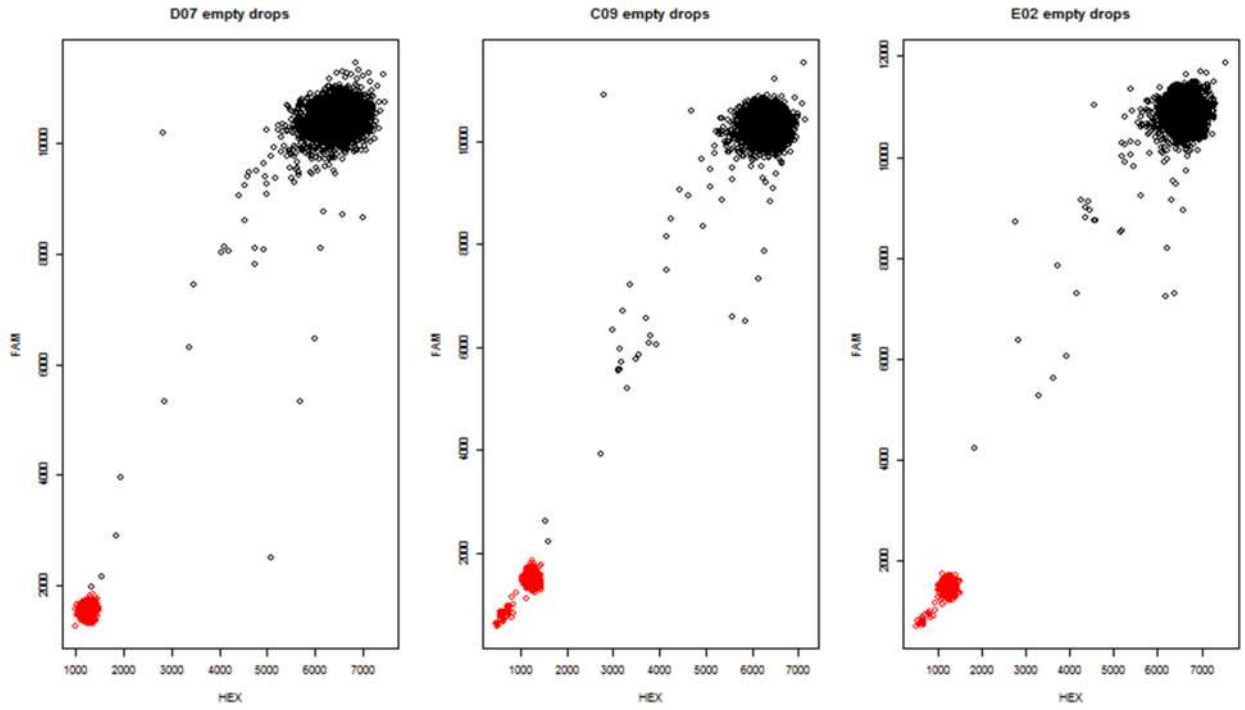

Figure 46 – Empty droplet identification in YUMAC dataset

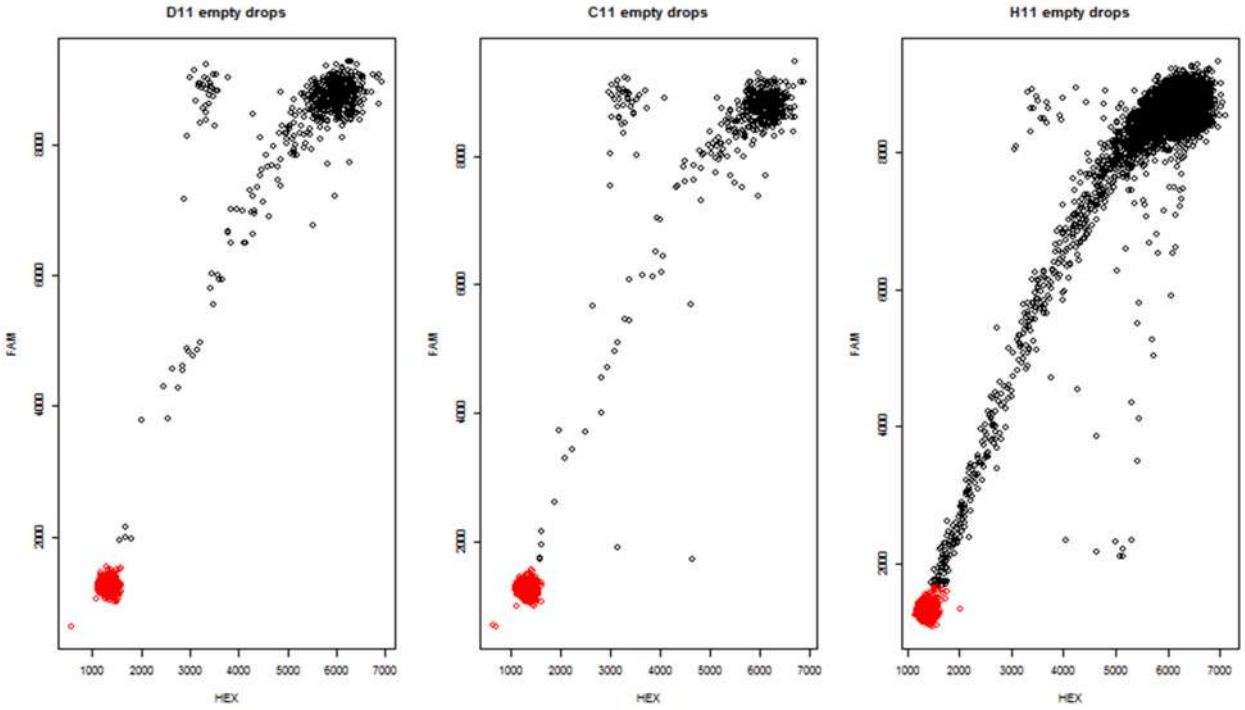

Figure 47 – Empty droplet identification in THYROID dataset

Appendix E Other attempted clustering algorithms

The algorithm devised in Chapter 2 for clustering droplets into empty, rain, single-positive, and double-positive droplets was developed only after simpler approaches were attempted. The initial main goal was merely to identify the single- and double-positive droplets. Figure 48 shows the data in well B11 of the THYROID dataset, which will serve as the example for most of the methods used in this appendix.

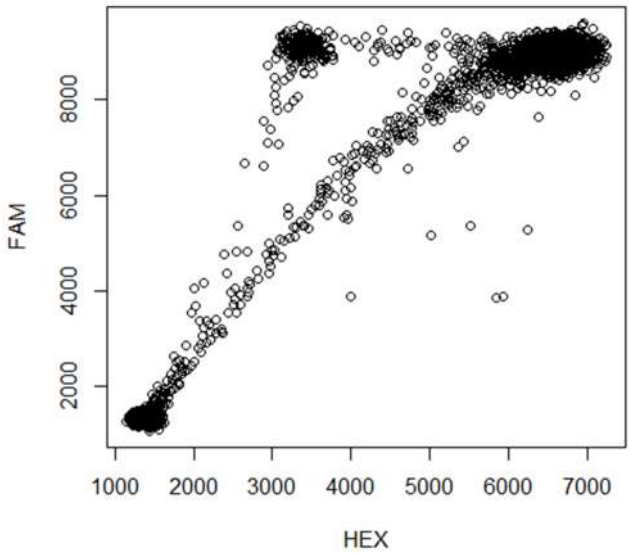

Figure 48 – Raw data in well B11 of the THYROID dataset

To the human eye, it seems clear that there are distinct clusters of single-positive and double-positive droplets. The exact threshold between them may not be objectively clear, but this data looks as though a standard clustering algorithm should be able to find the two clusters of interest.

The first method attempted was the K-means clustering algorithm, due to its popularity and simplicity. Figure 49 shows the result of running K-means on the well data. While the cores of the two clusters of interest were successfully clustered into different groups, the single-positive cluster also includes a lot of rain droplets and many droplets that would be better clustered as double-positive. I tried altering the algorithm parameters, such as providing the initial cluster centers as the three corners that are closest to each cluster, but the results did not change.

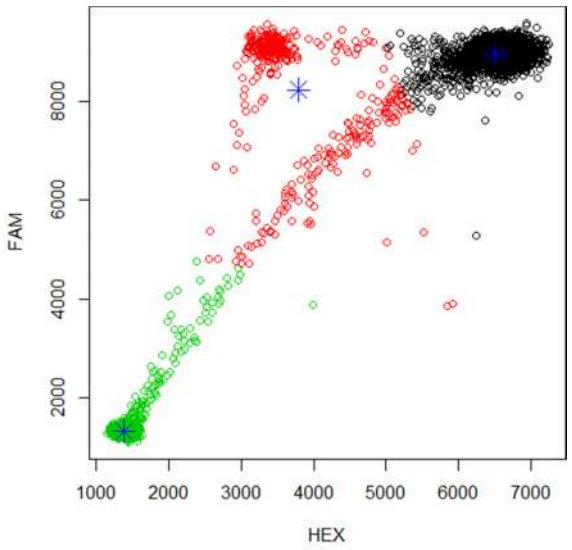

Figure 49 – Running K-means with 3 centers on the full data of well B11 of the THYROID dataset. The three centers are marked by the blue stars.

The next clustering method that I tried was a two-dimensional Gaussian Mixture Model (GMM). Running GMM on a well takes over five minutes, yet the results are still poor. Figure 50 shows the clustering results using GMM. The core of the double-positive cluster was identified, but the single-positive cluster includes all the rain droplets and many double-positive droplets. Different parameters were modified to try and produce better results, such as the shape and relative size of each cluster, but I could not get satisfying results with GMM.

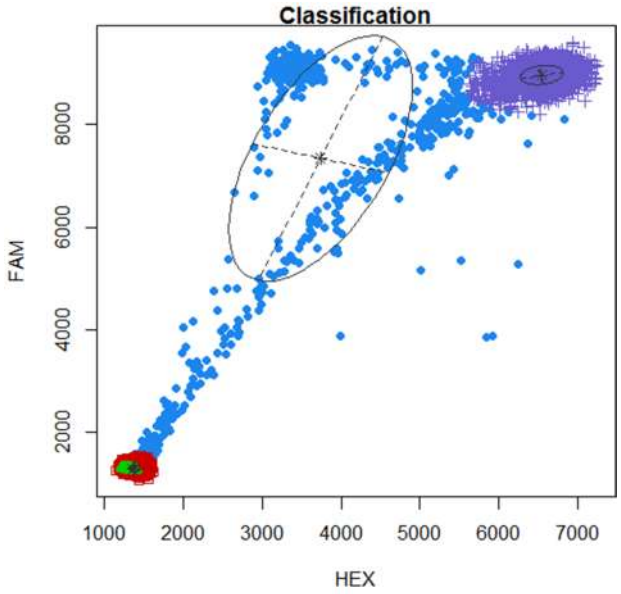

Figure 50 – Running GMM on the full data of well B11 of the THYROID dataset.

The next method I tried was Partitioning Around Medoids (PAM), which is a more robust version of the K-means algorithm. Despite being more robust, this method produced identical clustering as K-means.

Another method I wanted to try is divisive hierarchical clustering. Unfortunately, running this algorithm on a single well took over one hour, so it was not explored any further.

I hypothesized that one reason the clustering algorithms did not perform well on the raw well data is because the empty droplets are a much larger cluster than the others and they might be skewing the results. Similarly, some clustering algorithms were very slow to run because of the large number of droplets. Therefore, I wanted to see if the same clustering methods used earlier can produce better results after removing the empty droplets.

Figure 51 shows the result of running K-means on the same data after removing empty droplets. The result looks very similar to the ones achieved when running K-means on the full data. The PAM clustering method produced identical results to K-means. Figure 52 shows the results from using GMM, which are different than before, but the single-positive droplets are still grouped together with the rain droplets that are very close to the double-positive droplets. Figure 53 shows the results obtained from hierarchical clustering, which did not even separate the single- and double-positive droplets into different groups.

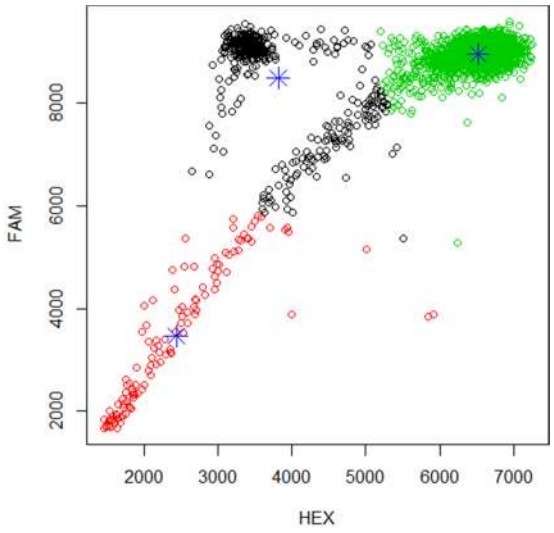

Figure 51 – Running K-means on well B11 of the THYROID dataset after removing empty droplets.

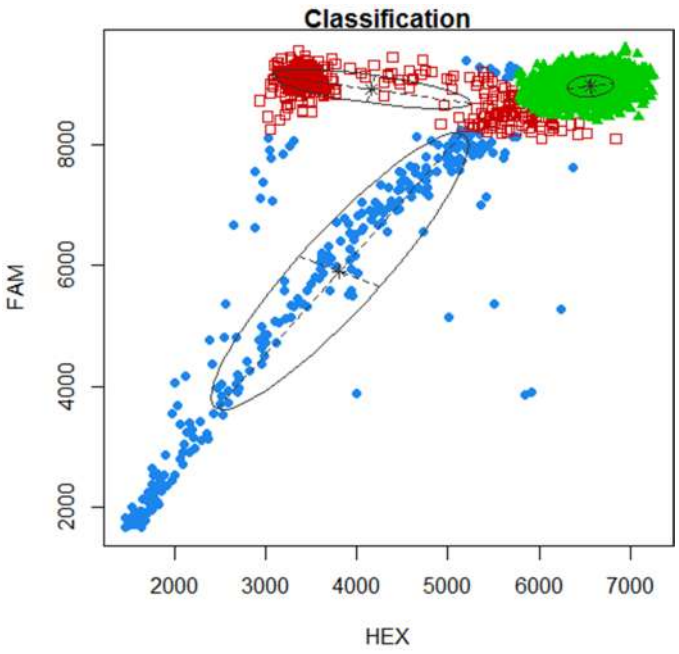

Figure 52 – Running GMM on well B11 of the THYROID dataset after removing empty droplets.

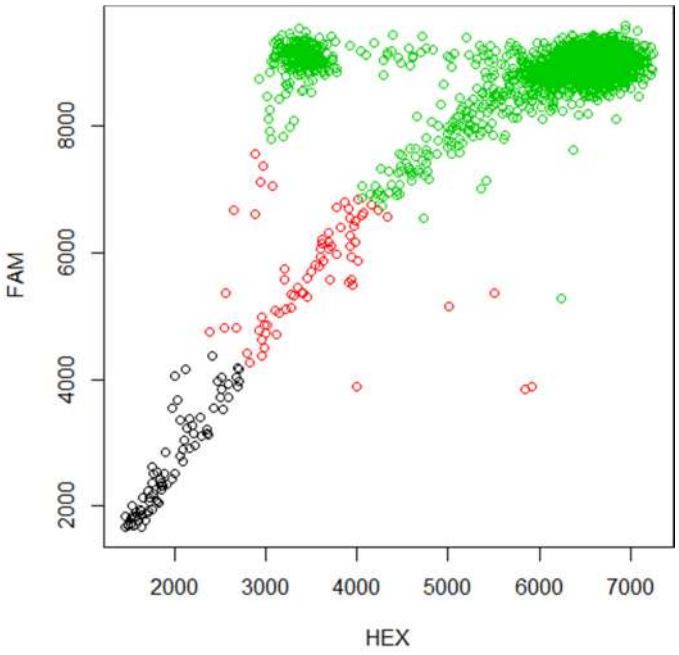

Figure 53 – Running hierarchical divisive clustering on well B11 of the THYROID dataset after removing empty droplets.

Another idea I tried was instead of clustering the droplets, to try to model them as two populations. This idea was suggested by Jenny Bryan. One population of droplets should account for the double-positive droplets and their rain, and the other population should account for the single-positive droplets and their rain. Figure 54 shows the results obtained with this method using different parameters.

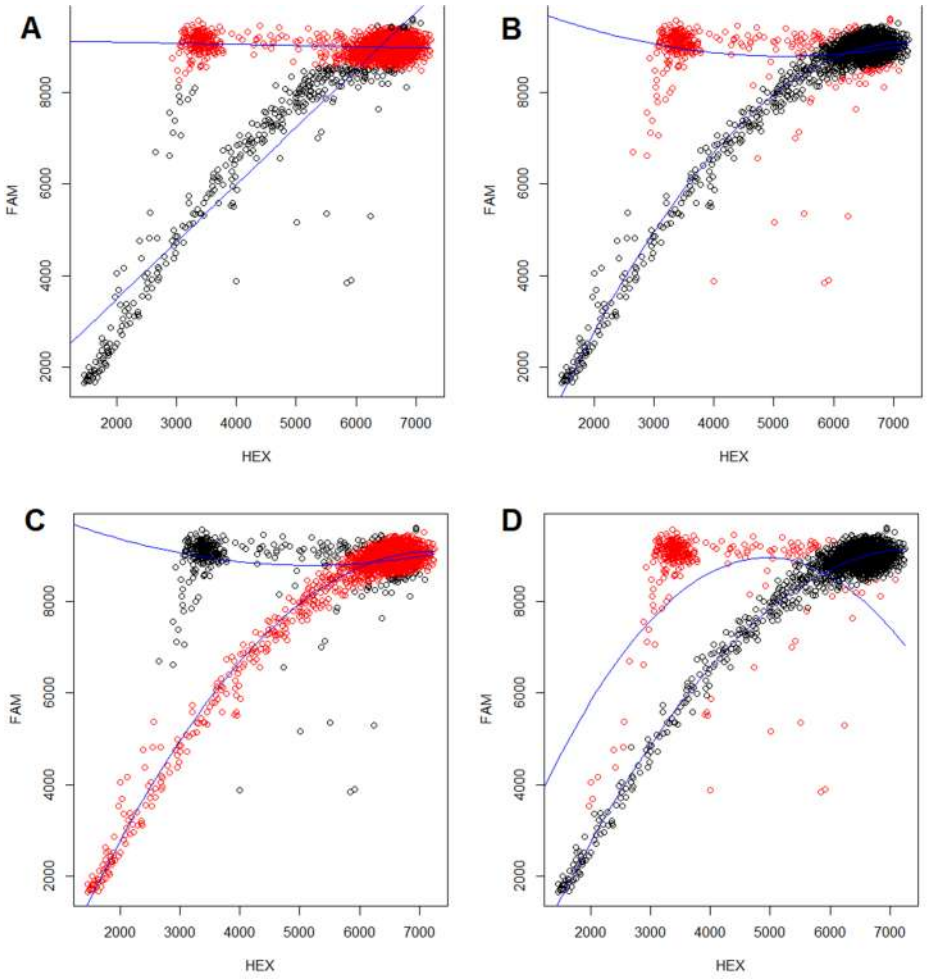

Figure 54 – Modeling the non-empty droplets in well B11 of the THYROID dataset as two droplet populations. (A) The first and most basic attempt at defining two droplet populations. The two blue lines show the curve that defines each population. These results are clearly poor because both clusters are grouped together. (B) A quadratic term was added to the model in order to allow some more complexity and flexibility in modeling the droplets. The resulting droplet group assignments do not differ much. (C) The initial clustering of the droplets was provided to the algorithm as a parameter, but despite starting with the correct clustering, the algorithm converged on a similarly poor result as before. (D) Since there are about five times more droplets in the major clusters than there are rain droplets, a higher weight was given to all the rain droplets to account for the droplet density. This resulted in better clustering, but it is still not satisfactory. More parameters and combinations of these parameters were also tried.

I theorized that the clustering algorithms were failing to cluster the droplets because the rain droplets add too much noise to the data. After removing both empty and rain droplets, I wanted to see if a standard clustering approach could be used to cluster the remaining filled droplets. Figure 55 shows the result of running K-means on the filled droplets.

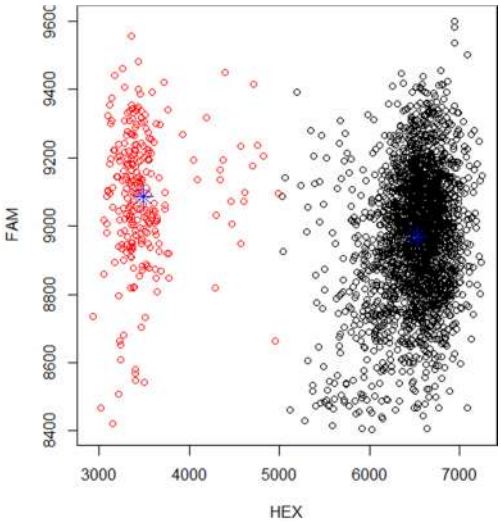

Figure 55 – Running K-means on well B11 of the THYROID dataset after removing empty and rain droplets.

K-means was able to differentiate the two clusters, but because of the nature of K-means, the dividing line between them was in the middle, which is not necessarily a good border for the clusters. To see this more clearly, Figure 56 shows a different well, where the double-positive cluster is wider than the single-positive cluster, and therefore the K-means result is not valid.

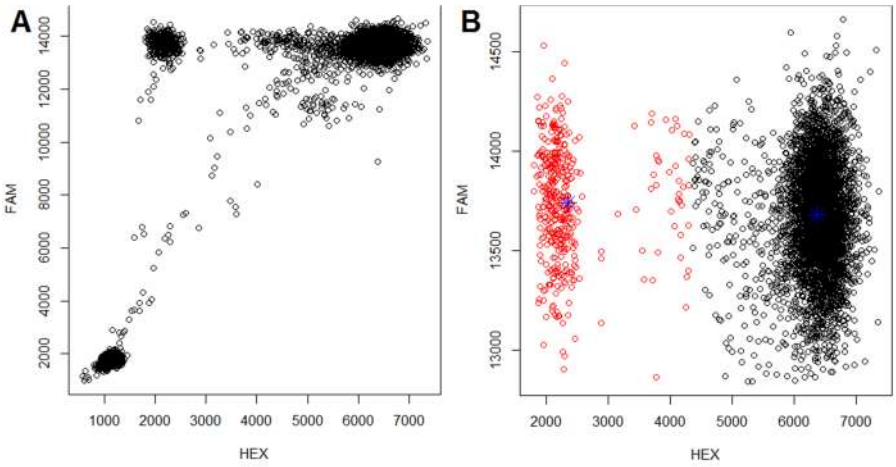

Figure 56 – Running K-means on well A02 of the PLASMID dataset after removing empty and rain droplets. (A) The raw data in the well. (B) The K-means clustering of the filled droplets.

Using the PAM algorithm on this data produced identical results to K-means. Figure 57 shows the result of using GMM to cluster the same data. Running hierarchical clustering on this well took over thirty minutes, so it was not explored any further. Even when removing all the noise from the empty and rain droplets, standard clustering algorithms do not yield acceptable results.

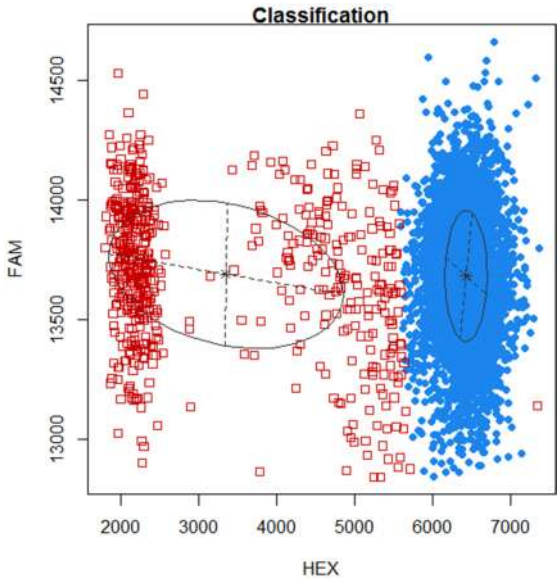

Figure 57 – Running GMM on well A02 of the PLASMID dataset after removing empty and rain droplets.

Appendix F Extended examples of using density kernel estimation for droplet gating

F1. Wells with a large number of single-positive droplets, without optimizing density bandwidth

These figures show examples of running the algorithm for gating negative and positive droplets in wells with a large number of single-positive droplets. In these examples, the gating algorithm does not optimize the density kernel bandwidth. In these figures, the sub-figures labeled A through D always show the same steps for every well. See **Figure 19** for an explanation of each label A-D.

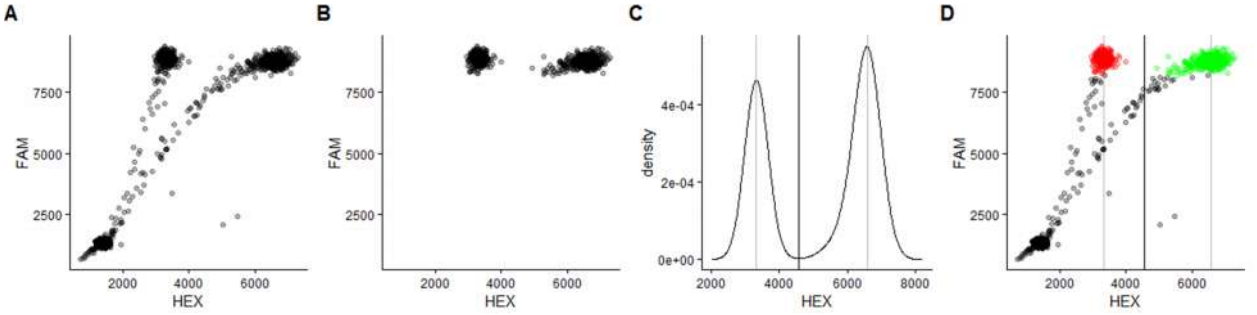

Figure 58 – Identifying positive vs negative droplets in well A09 of dataset CRC-41

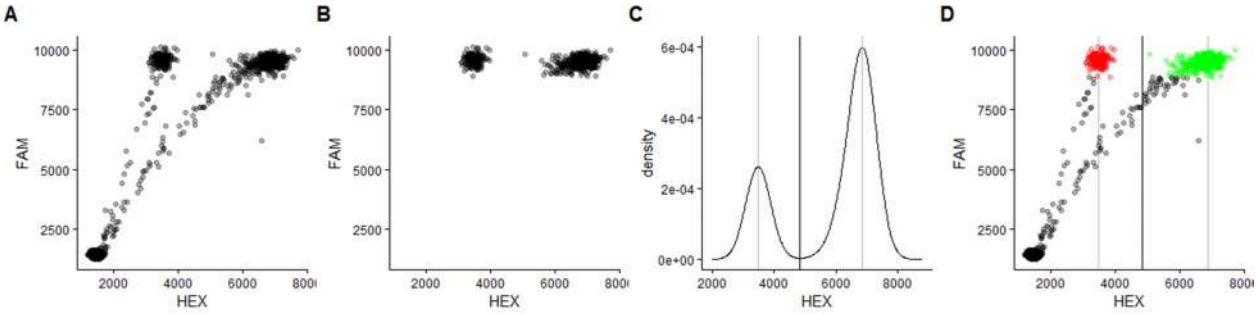

Figure 59 – Identifying positive vs negative droplets in well C08 of dataset CRC-41

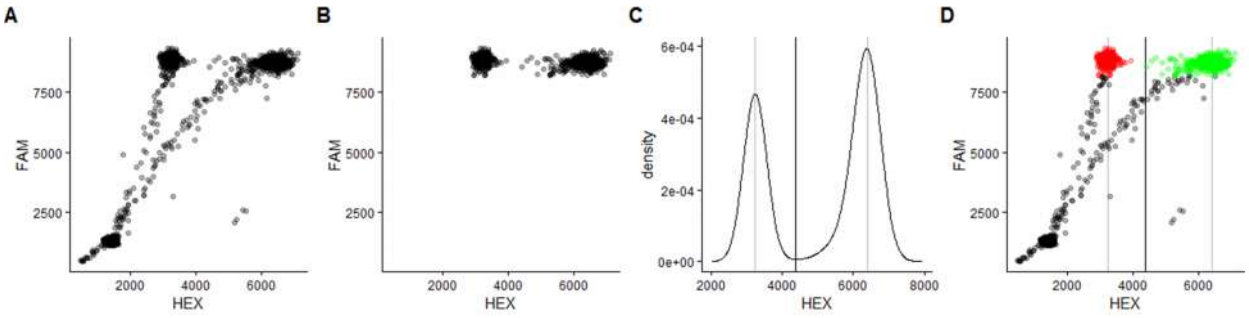

Figure 60 – Identifying positive vs negative droplets in well E04 of dataset CRC-REPEAT

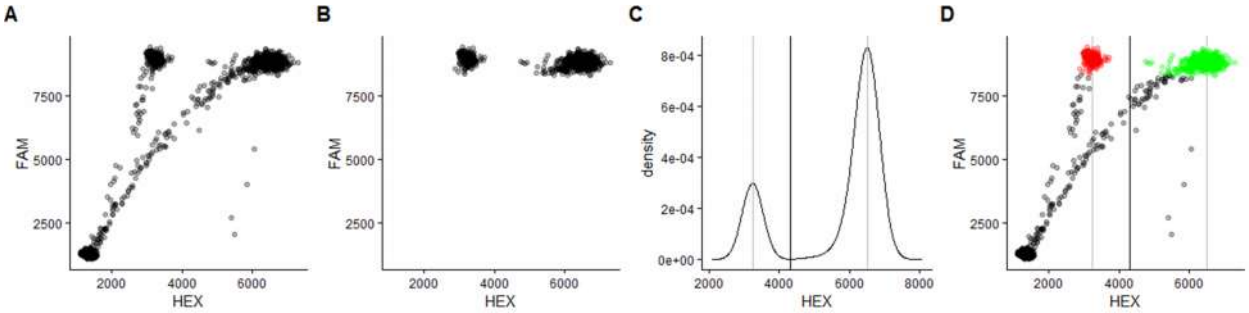

Figure 61 – Identifying positive vs negative droplets in well E05 of dataset CRC-REPEAT

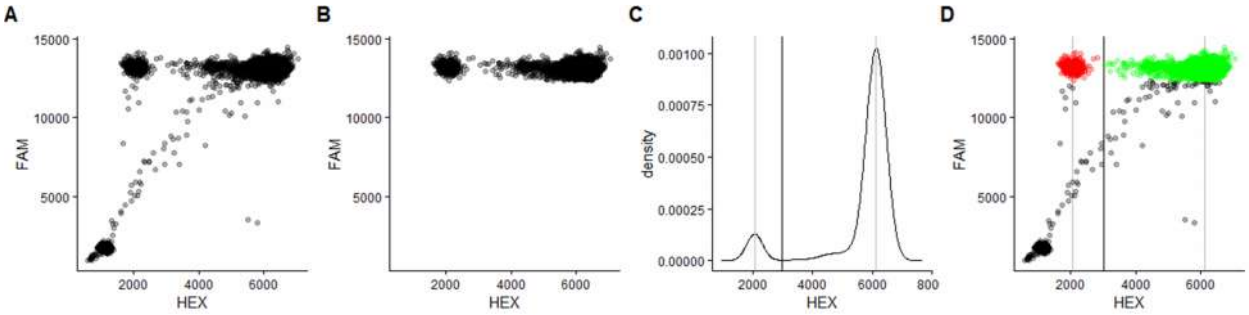

Figure 62 – Identifying positive vs negative droplets in well B02 of dataset PLASMID

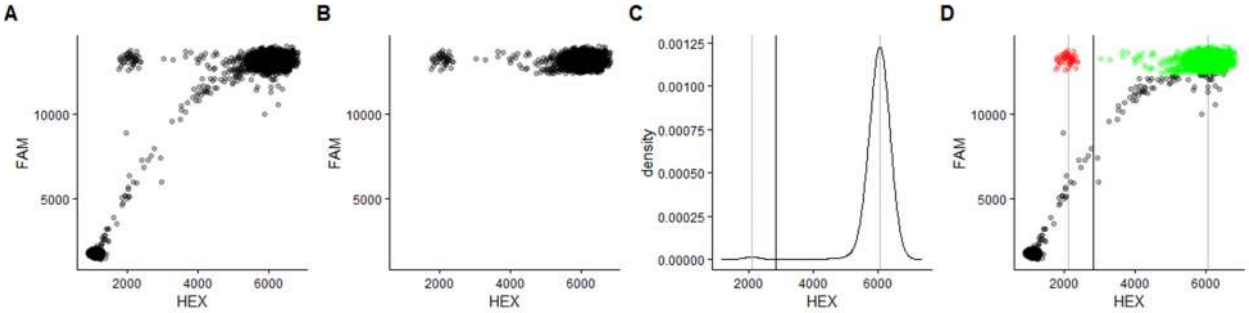

Figure 63 – Identifying positive vs negative droplets in well C03 of dataset PLASMID

F2. Wells with a small number of singly-positive droplets, without optimizing density bandwidth

These figures show examples of running the algorithm for gating negative and positive droplets in wells with a large number of single-positive droplets. In these examples, the gating algorithm does not optimize the density kernel bandwidth. In these figures, the sub-figures labeled A through D always show the same steps for every well. See **Figure 19** for an explanation of each label A-D.

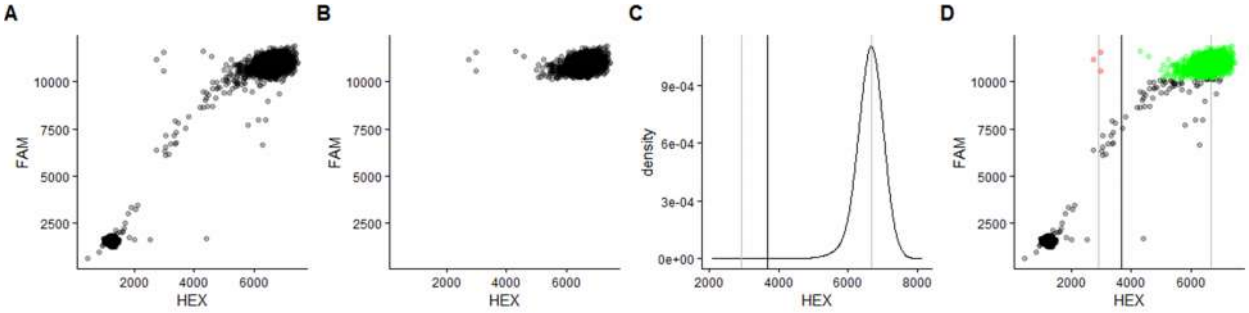

Figure 64 – Identifying positive vs negative droplets in well B12 of dataset BRAFV600K

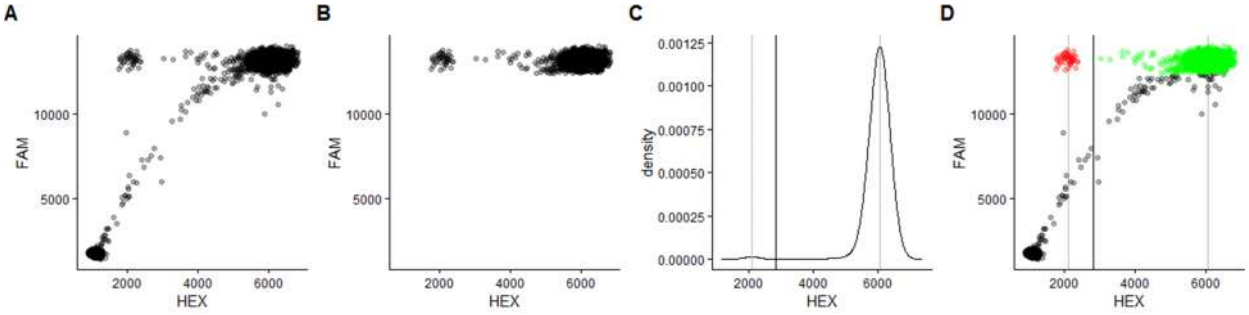

Figure 65 – Identifying positive vs negative droplets in well C03 of dataset PLASMID

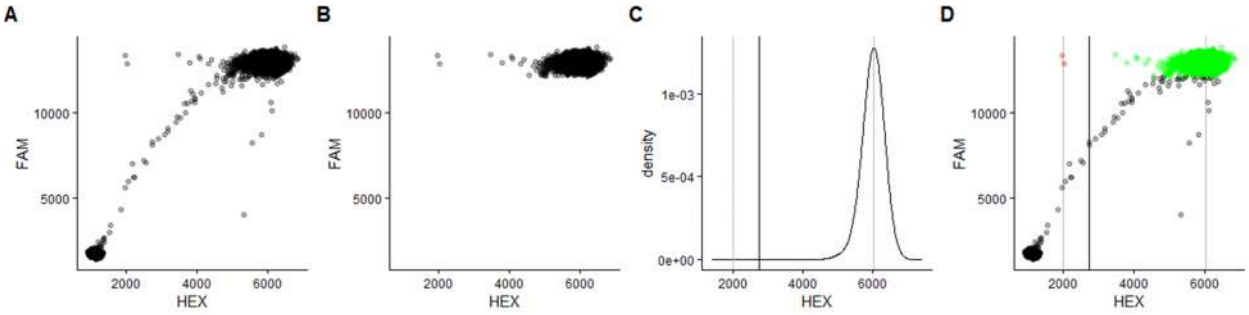

Figure 66 – Identifying positive vs negative droplets in well C04 of dataset PLASMID

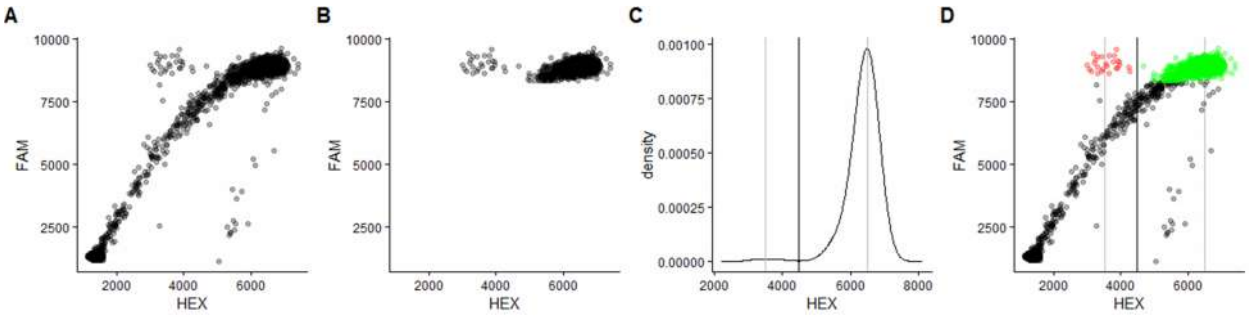

Figure 67 – Identifying positive vs negative droplets in well G02 of dataset CRC-REP

F3. Well without any singly-positive droplets

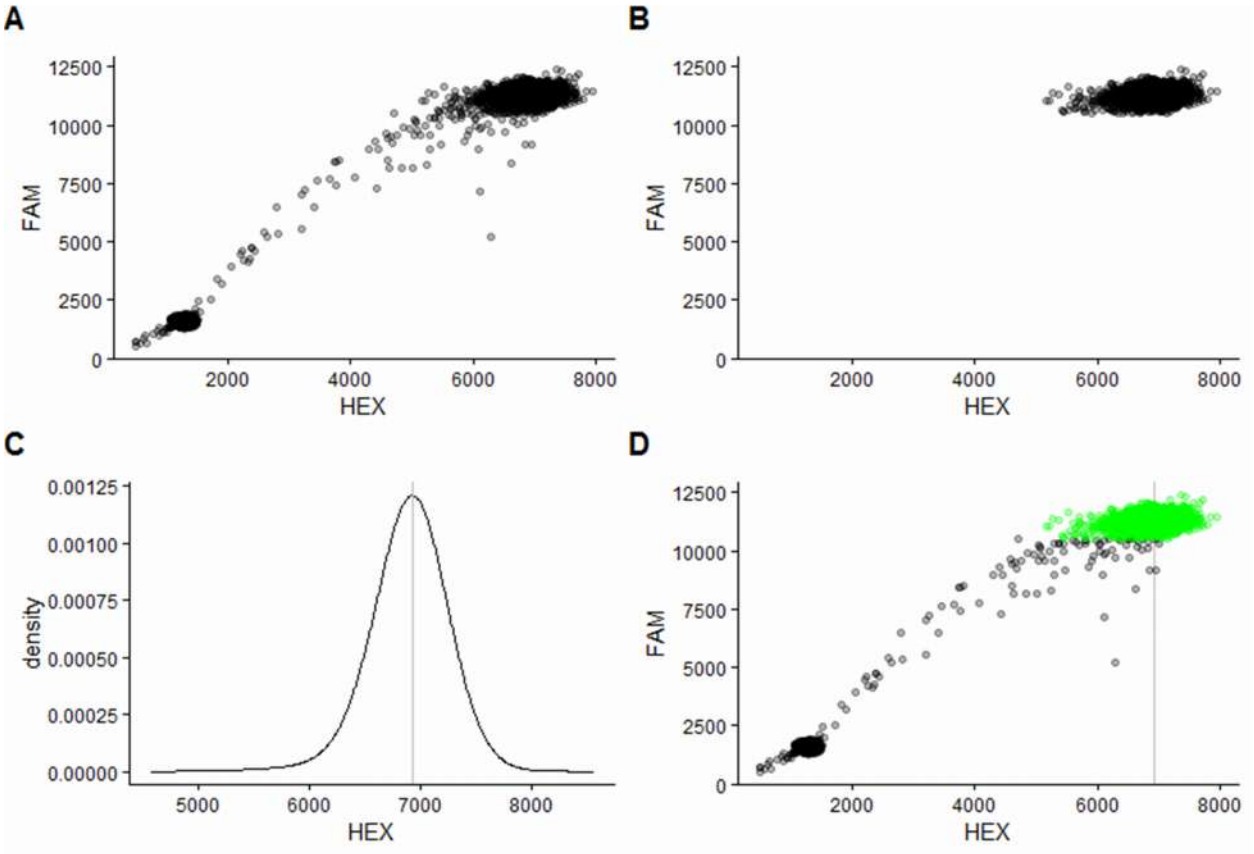

Figure 68 – Gating positive and negative droplets using a kernel density estimator in a well without negative droplets. (A) A typical PN/PP well containing only a positive droplet population is shown. (B) To calculate the gate between positive and negative droplets, only the filled droplets are retained. (C) A kernel density estimator of the HEX values of the filled droplets is generated. The grey line corresponds to the peak in the HEX density. Since there is no population of negative droplets, only one peak exists, and there are no troughs. (D) The full is shown, assigning all filled droplets as positive (green).

F4. Using too high of a smoothing bandwidth for KDE during droplet gating

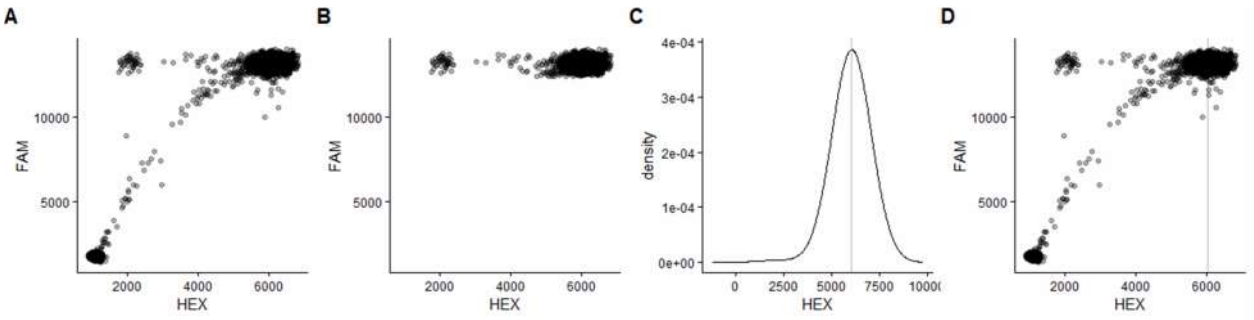

Figure 69 – Using too high of a smoothing bandwidth for KDE during droplet gating. A KDE is generated with the default bandwidth smoothing factor multiplied by 15. Even though there is a visible cluster of singly-positive droplets in the well, the high degree of smoothing causes the larger cluster to completely overshadow it, and only one peak exists. A lower smoothing bandwidth is required to find the correct clusters.

F5. Using too low of a smoothing bandwidth for KDE during droplet gating

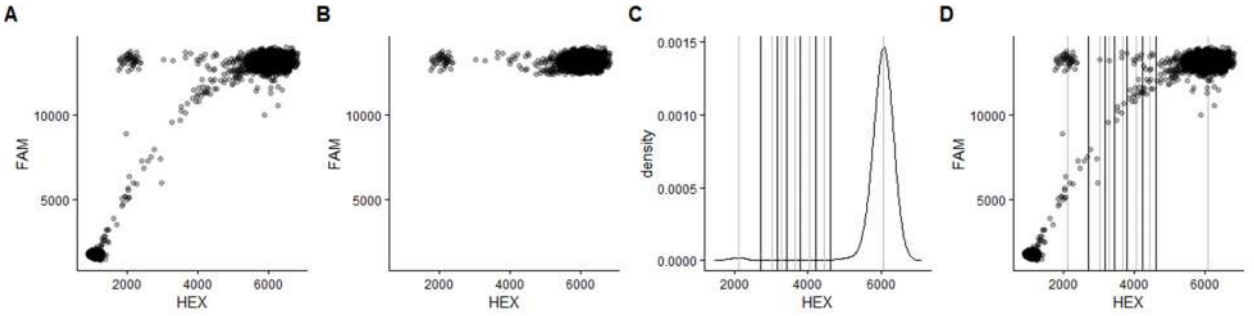

Figure 70 – Using too low of a smoothing bandwidth for KDE during droplet gating. A KDE is generated with the default bandwidth smoothing factor multiplied by 2. The low degree of smoothing is too sensitive and results in too many peaks. A higher smoothing bandwidth is required to focus on the most significant clusters.

F6. Finding the optimal smoothing bandwidth by iteratively increasing its multiplier

The following figures show the iterative process of finding the optimal smoothing bandwidth for the KDE used in droplet gating. All figures have the same set of labels: (A) is the raw droplet data, along with the threshold (black line) that separates the filled droplets from the rest. The last box shows the raw droplet data, along with the peaks and troughs of the optimal KDE and the positive (green) and negative (red) droplets are coloured. The rest of the boxes zoom in on the

filled droplets and show the KDE of the HEX values using increasingly larger smoothing parameters. The peaks (grey lines) and troughs (black lines) in the KDE are shown. The optimal smoothing parameter is reached when there is only one trough, which acts as the gate between positive and negative droplets.

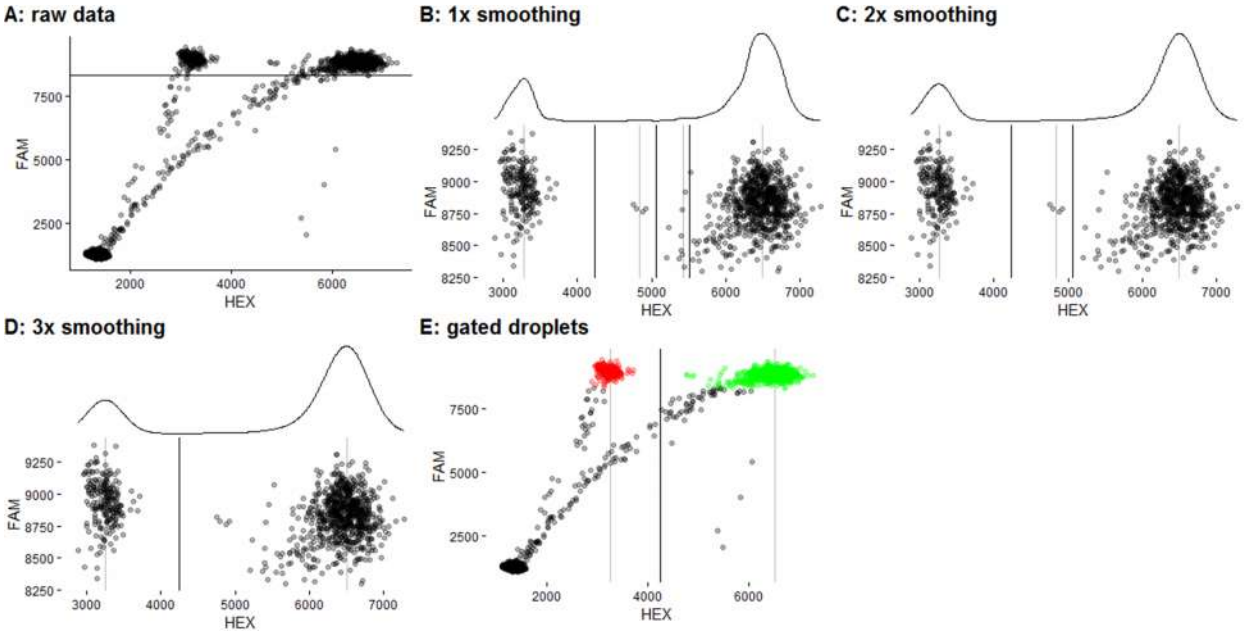

Figure 71 – Finding the optimal smoothing bandwidth for well E05 of the CRC-REP dataset.

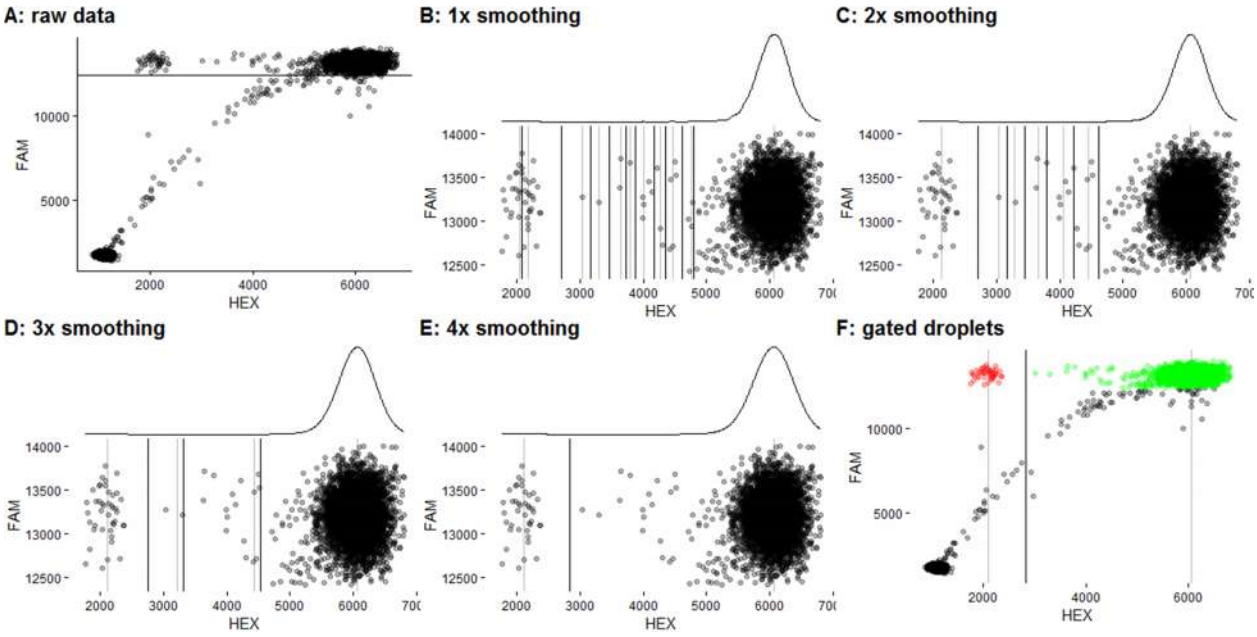

Figure 72 – Finding the optimal smoothing bandwidth for well C03 of the PLASMID dataset.

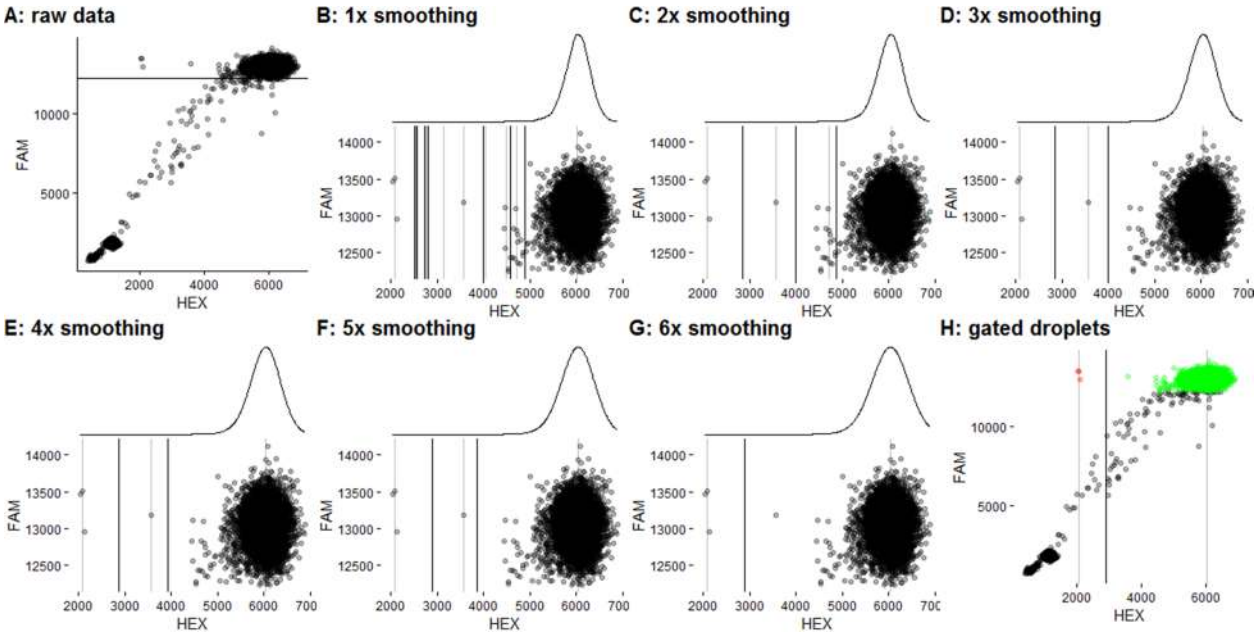

Figure 73 – Finding the optimal smoothing bandwidth for well H04 of the PLASMID dataset.

F7. Altering the original calculated gate to a Gaussian gate

The following figures show wells where the original gate calculated using the KDE is replaced by the Gaussian gate. All figures share the same labels: (A) shows the raw data with a black line showing the filled droplets threshold, (B) shows the original KDE gate, (C) shows the Gaussian gate, (D) shows all the positive (green) and negative (red) droplets discriminated using the Gaussian gate.

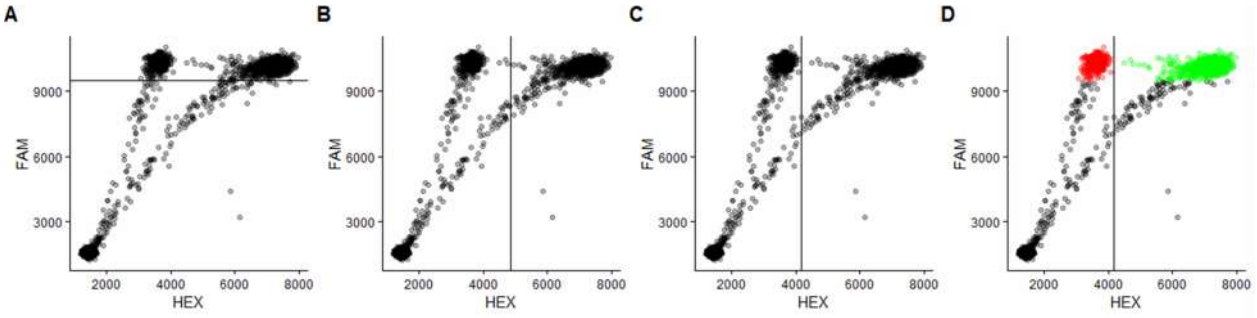

Figure 74 – Using a Gaussian gate for well A11 in the CRC-41 dataset.

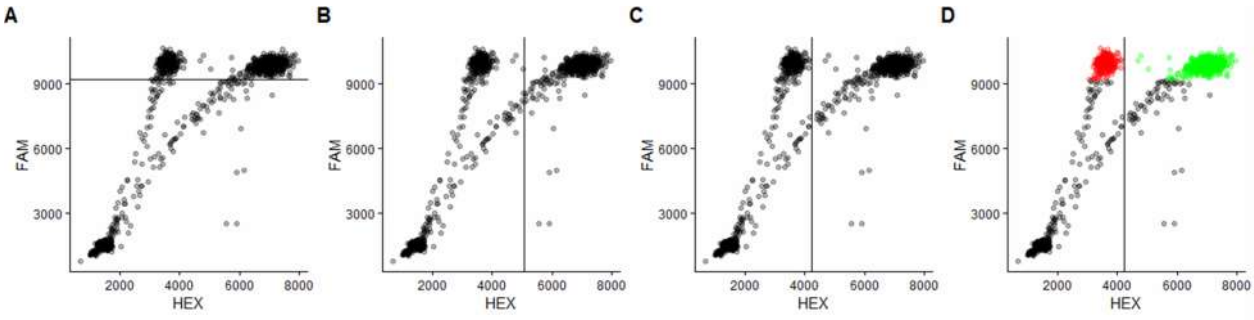

Figure 75 – Using a Gaussian gate for well F08 in the CRC-41 dataset.

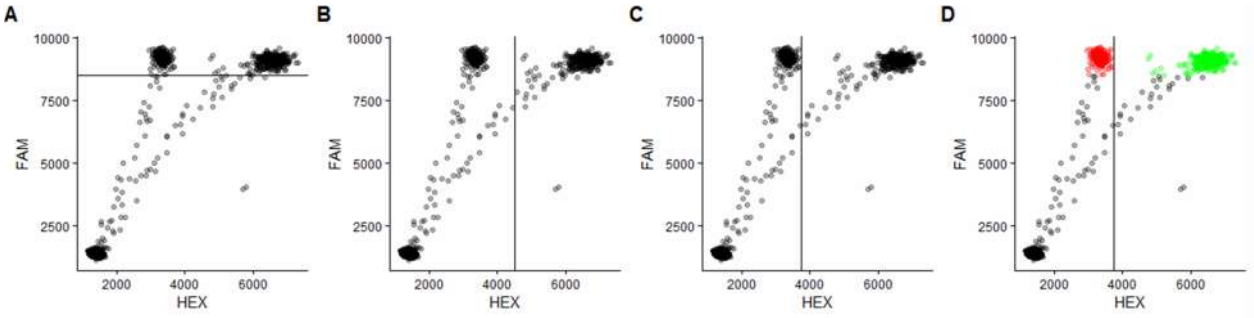

Figure 76 – Using a Gaussian gate for well H06 in the CRC-41 dataset.

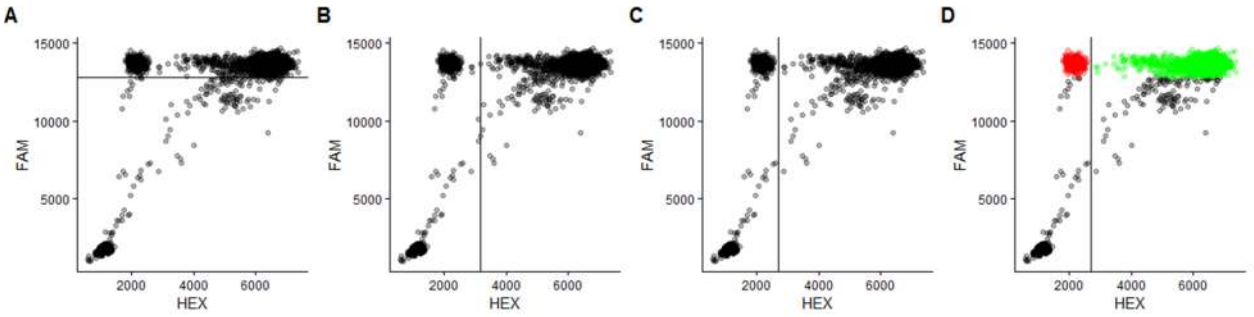

Figure 77 – Using a Gaussian gate for well A02 in the PLASMID dataset.

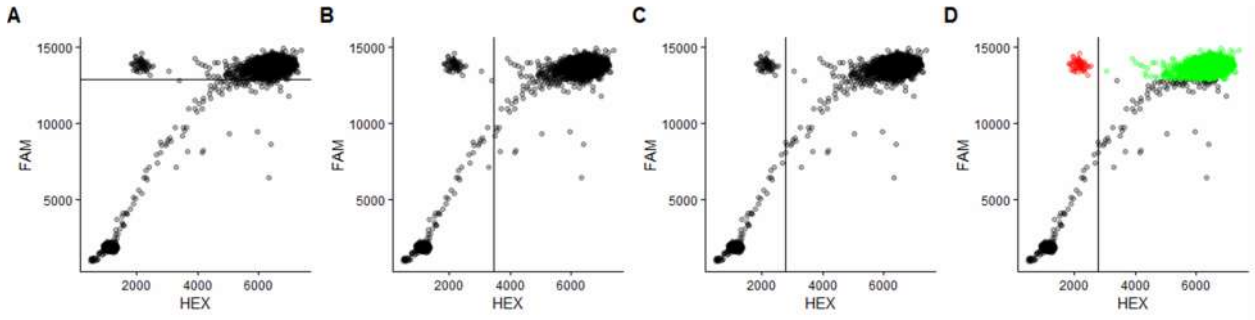

Figure 78 – Using a Gaussian gate for well A03 in the PLASMID dataset.

Appendix G Screenshots from QuantaSoft and the *ddpcr* web tool

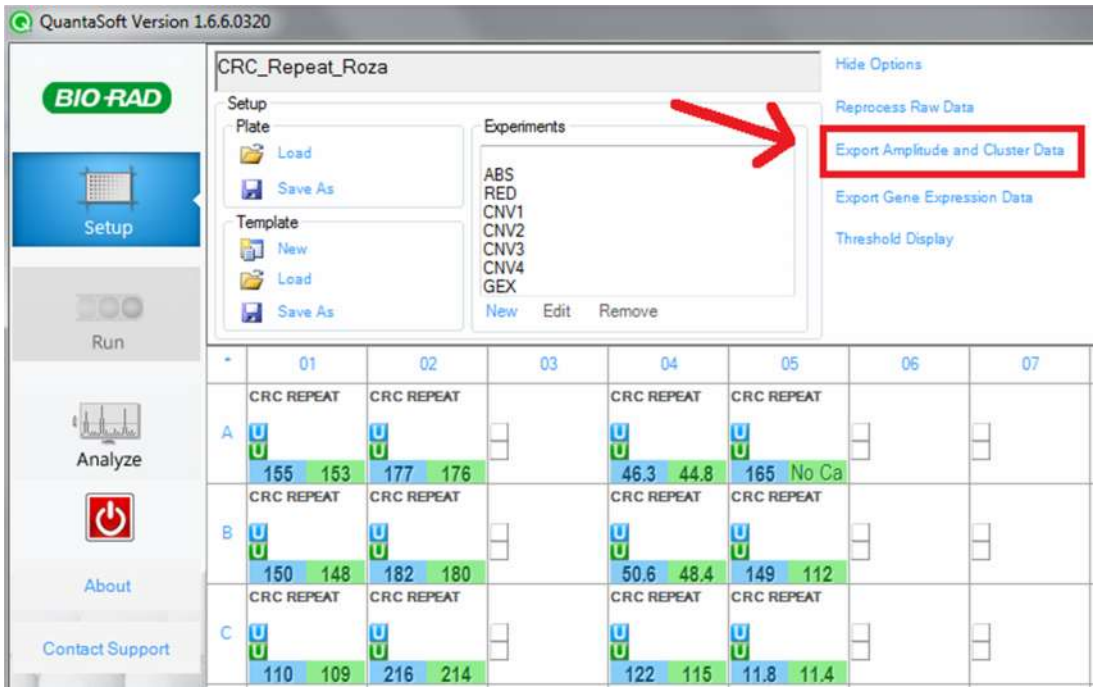

Figure 79 – Exporting droplet fluorescence data from QuantaSoft

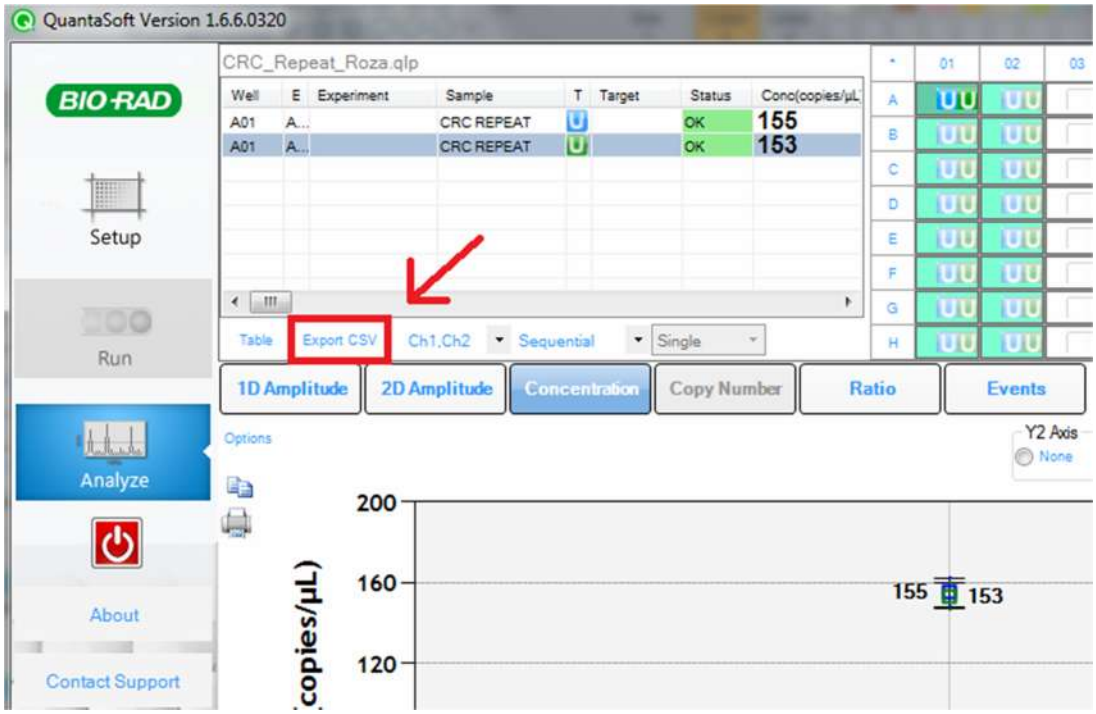

Figure 80 – Exporting plate metadata from QuantaSoft

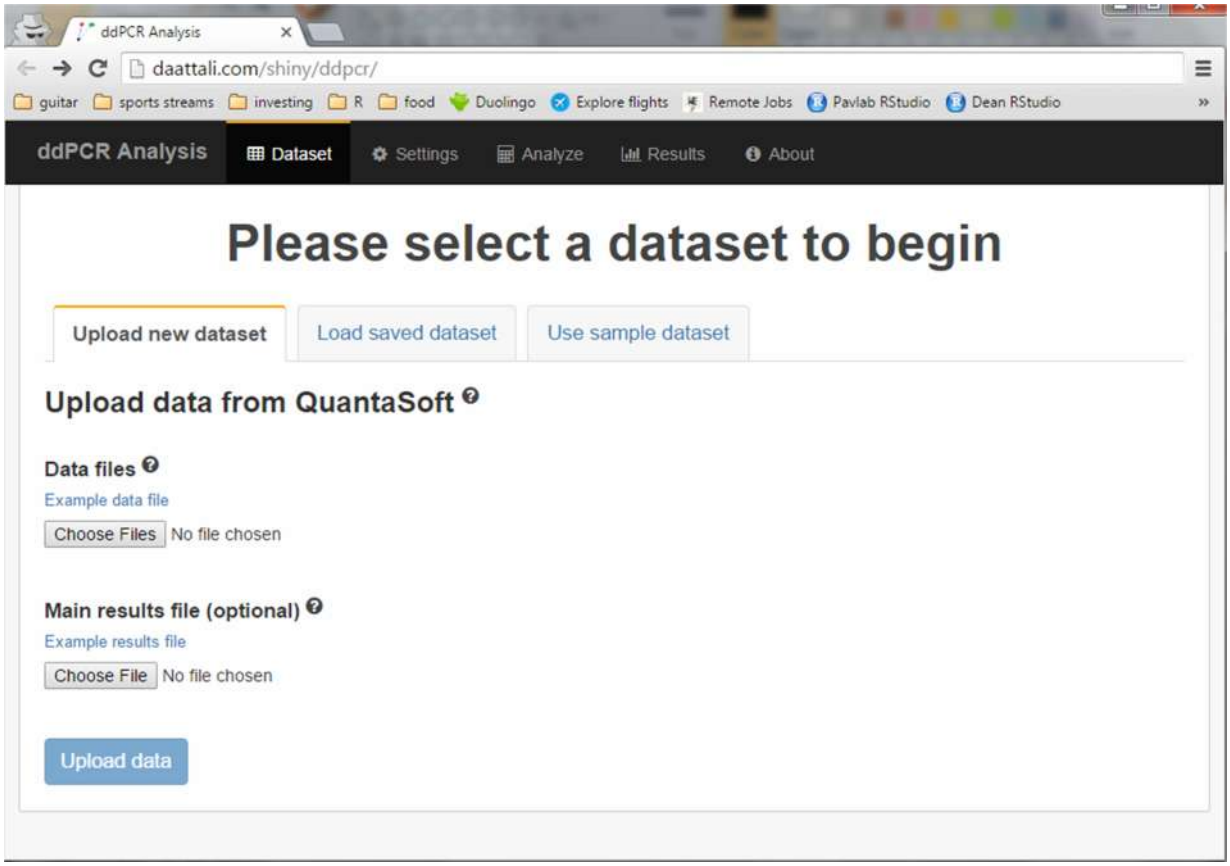

Figure 81 – ddpcr web application: Dataset tab

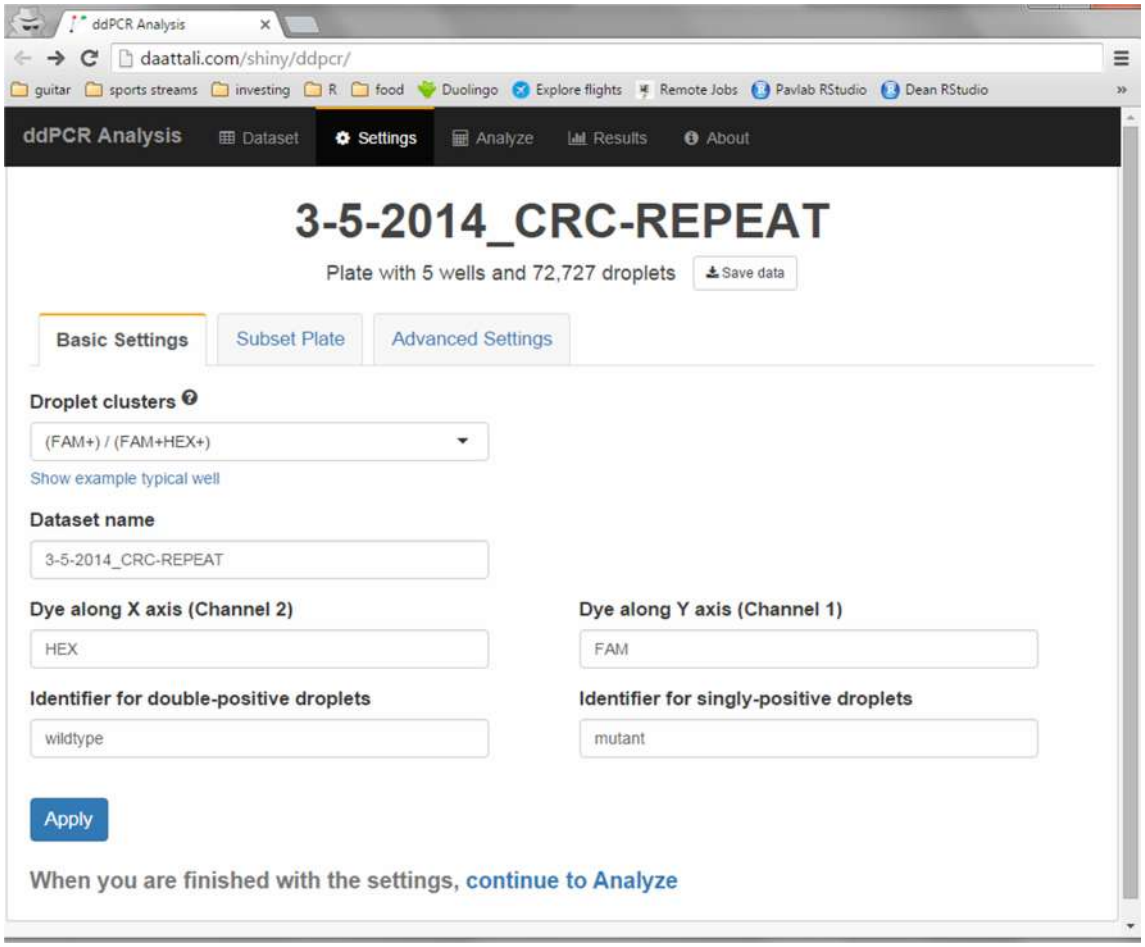

Figure 82 – ddpcr web application: Settings tab

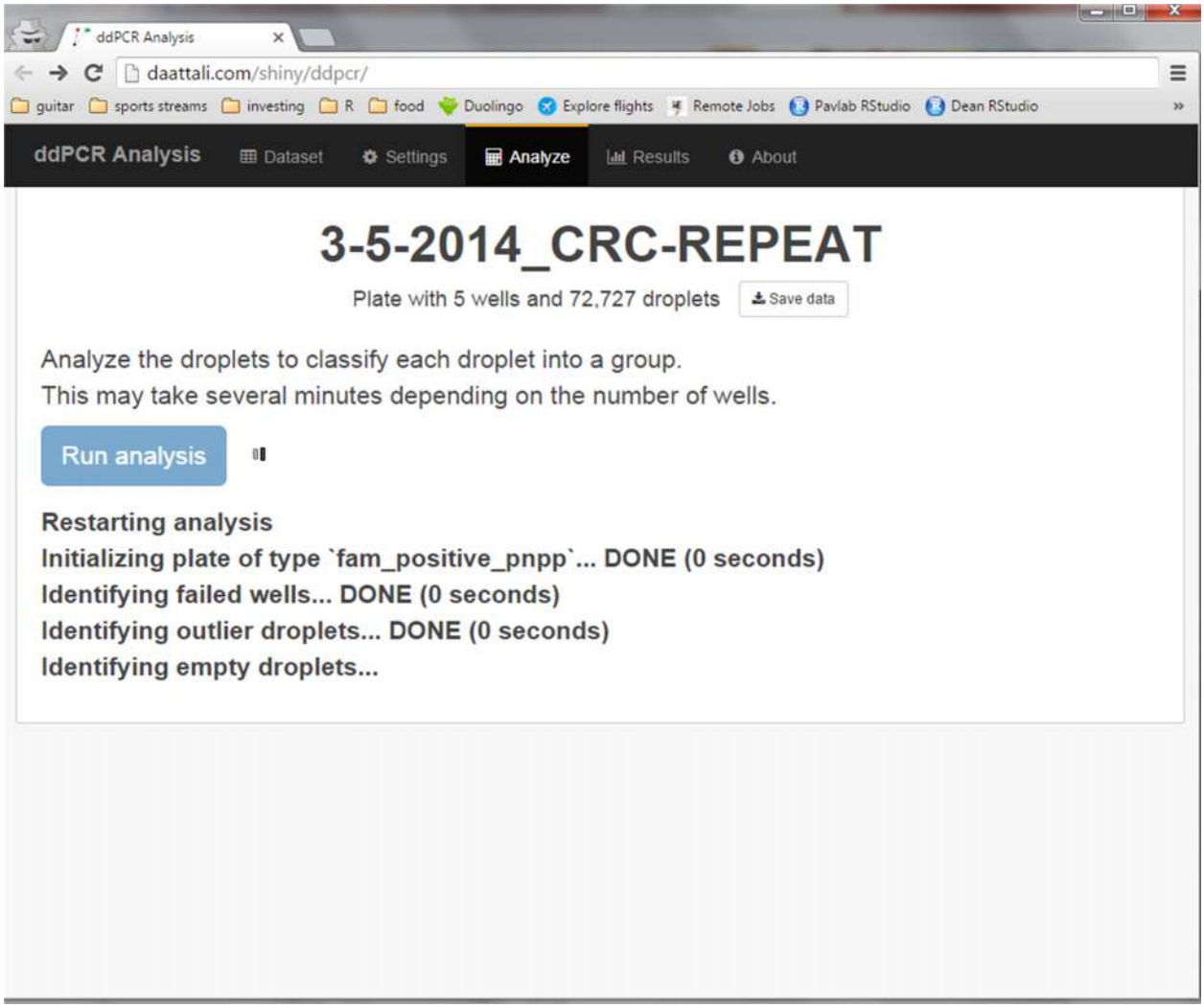

Figure 83 – ddpcr web application: Analyze tab

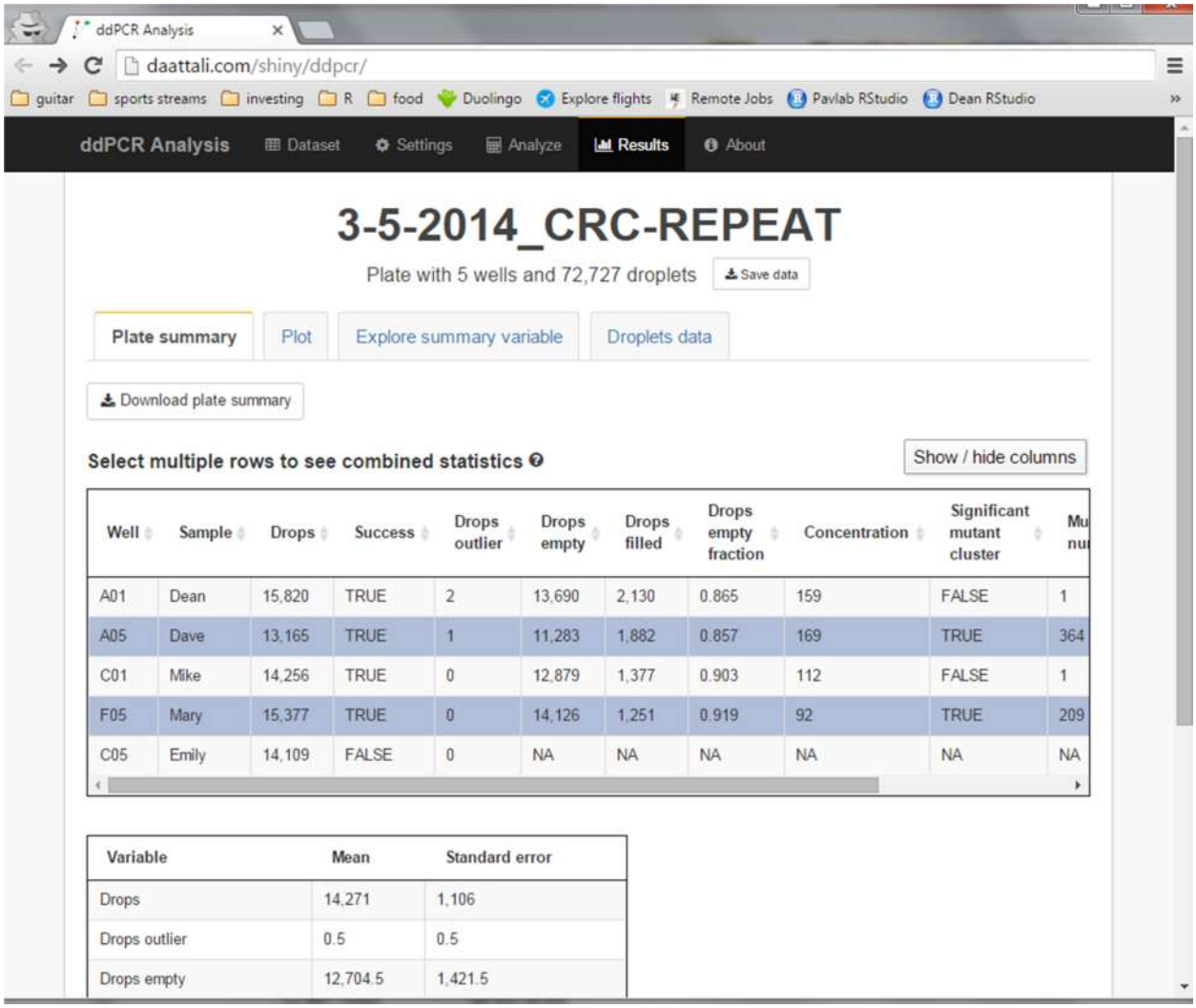

Figure 84 – ddpcr web application: Results tab

Appendix H Extending *ddPCR* by adding a new plate type

As a simple exercise to show how easy it is to create a new plate type, this section will walk through the steps required to create a new plate type. The name of the new plate type is `fam_border`. In experiments of this type, we are interested in running the typical pre-processing (identify failed wells, outliers droplets, and empty droplets), followed by simply classifying the remaining droplets as either FAM-positive or FAM-negative, with the border between positive and negative being a predefined value. This plate type is clearly not very useful but it is good for demonstration purposes.

The first thing to do is to define the parent plate type, which in this case will be `ddpcr_plate`.

```
parent_plate_type.fam_border <- function(plate) {
  "ddpcr_plate"
}
```

The next step is to define the parameters of this plate type. ddPCR plates of this type are expected to have at least 14,000 droplets per well as a QA metric. If any well has fewer droplets, it is considered a failure. There is already a parameter for this, so we just need to modify it. We also add a new parameter for the border to use for gating the droplets, and give it a default value.

```
define_params.fam_border <- function(plate) {
  params <- NextMethod("define_params")

  new_params <- list(
    'REMOVE_FAILURES' = list(
      'TOTAL_DROPS_T' = 14000  # overwriting an existing parameter
    ),
    'GATE' = list(
      'FAM_BORDER' = 5000  # defining a new parameter
    )
  )
  params <- modifyList(params, new_params)

  params
}
```

Next we define the potential droplet clusters. We use the same clusters as the default plate type and add two more, to denote droplets in the FAM-positive and FAM-negative sections.

```
define_clusters.fam_border <- function(plate) {
  clusters <- NextMethod("define_clusters")

  c(clusters,
    'FAM_POSITIVE',
    'FAM_NEGATIVE'
  )
}
```

Next we define the analysis steps. We use all the default analysis steps (pre-processing steps), and add a gating step.

```
define_steps.fam_border <- function(plate) {
  steps <- NextMethod("define_steps")

  c(steps,
    list(
      'GATE' = 'gate_droplets'
    ))
}
```

We need to create the `gate_droplets` function with the actual logic that will perform the step. Notice the use of the following helper functions that are especially useful when writing analysis step functions: `step`, `check_step`, `step_begin`, `step_end`, `unanalyzed_clusters`, `status`. You can look up the documentation for each of these functions to learn more.

```
gate_droplets <- function(plate) {
  # make sure this step was not called prematurely
  current_step <- step(plate, 'GATE')
  check_step(plate, current_step)

  # show an informative message to the user
  step_begin("Classifying droplets as FAM-positive or negative")

  data <- plate_data(plate)
  border <- params(plate, 'GATE', 'FAM_BORDER')

  # get a list of clusters that have not been considered yet in the analysis
  # this is useful so that we only look at droplets that have not yet been
  # assigned to a cluster
  unanalyzed_clusters <- unanalyzed_clusters(plate, 'FAM_POSITIVE')

  # get the indices of all droplets that are FAM-positive and negative
  unanalyzed_idx <- data$cluster %in% unanalyzed_clusters
  fam_pos <- unanalyzed_idx & data$FAM >= border
```

```
  fam_neg <- unanalyzed_idx & data$FAM < border

  # assign each droplet to its cluster
  data[fam_pos, 'cluster'] <- cluster(plate, 'FAM_POSITIVE')
  data[fam_neg, 'cluster'] <- cluster(plate, 'FAM_NEGATIVE')

  # update the data on the plate object
  plate_data(plate) <- data

  # record how many drops in each well are in each cluster
  # and add this info to the plate's metadata
  drops_per_cluster <-
    plyr::ddply(data, ~ well, function(x) {
      data.frame(
        'drops_positive' = sum(x$cluster == cluster(plate, 'FAM_POSITIVE')),
        'drops_negative' = sum(x$cluster == cluster(plate, 'FAM_NEGATIVE'))
      )
    })
  plate_meta(plate) <-
    dplyr::left_join(
      plate_meta(plate),
      drops_per_cluster,
      by = "well"
    )

  # VERY IMPORTANT - do not forget to update the status of the plate
  status(plate) <- current_step
  step_end()

  plate
}
```

Now the new plate type is ready to be used. We can also add a plot function if there are any customizations to the plot that are specific to this plate type. The default plot function of ddPCR plates goes a long way, but one thing we can add is a line showing the division border.

```
plot.fam_border <- function(x, ..., show_border = FALSE) {
  # Plot a regular ddpcr plate
  p <- NextMethod("plot", x)

  # Show the custom thresholds
  if (show_border) {
    border <- params(x, 'GATE', 'FAM_BORDER')
    p <- p +
      ggplot2::geom_hline(yintercept = border)
  }
  p
}
```

The one last thing we can add is a convenient way for the user to set the border parameter.

```
`fam_border<-` <- function(plate, value) {
  params(plate, 'GATE', 'FAM_BORDER') <- value
  plate
}
```

Now we can create a new ddPCR plate with the new type and run a full analysis of it.

```
library(ddpcr)
plate <- new_plate(dir = sample_data_dir(), type = "fam_border")
fam_border(plate) <- 8000
plate <- analyze(plate)
plot(plate, show_drops_empty = TRUE)
plot(plate, col_drops_fam_negative = "red",
     col_drops_fam_positive = "blue", show_border = TRUE)
plate_meta(plate, only_used = TRUE)
```

Appendix I Supplementary data for the *BRAF*-V600 assay case study

Table 1 – Comparing *BRAF* status and mutation frequency of patient samples across three methods: staining by a pathologist, manual analysis by Roza Bidshahri, and automatic analysis with ddpcr

sample	well	BRAF status by pathologist	BRAF status by automatic analysis	Mutant frequency by manual analysis	Mutant frequency by automatic analysis
#1	A01	WT	WT	0	0
#1	B01	WT	WT	0.15	0.16
#10	A02	WT	WT	0	0
#10	B02	WT	WT	0.1	0.1
#11	C02	WT	WT	0.08	0.09
#11	D02	WT	WT	0	0
#13	E02	WT	WT	0	0
#13	F02	WT	WT	0.36	0.38
#15	G02	WT	WT	0.27	0.16
#15	H02	WT	WT	0.31	0.3
#20	A03	WT	WT	0	0
#20	B03	WT	WT	0.4	0.32
#22	C03	WT	WT	0.35	0.38
#22	D03	WT	WT	0	0
#26	E03	WT	WT	0.4	0.2
#26	F03	WT	WT	0.6	0.4
#3	C01	WT	WT	0.08	0.08
#3	D01	WT	WT	0.08	0
#31	G03	WT	WT	0.41	0.22
#31	H03	WT	WT	0.58	0.36
#35	A04	WT	WT	0	0
#35	B04	WT	WT	0	0
#38	C04	WT	WT	0.32	0.33
#38	D04	WT	WT	0.38	0.41
#40	E04	WT	WT	0.65	0.46
#40	F04	WT	WT	0.21	0.22
#41	G04	WT	WT	0.36	0.39
#41	H04	WT	WT	0.21	0.12
#5	E01	WT	WT	1.05	1.13
#5	F01	WT	WT	1.22	1.05
#8	G01	WT	WT	0.31	0.32
#8	H01	WT	WT	0.15	0.16
#12	C06	MT	MT	40.02	NA
#12	D06	MT	MT	25.02	NA
#14	E06	MT	MT	34.26	34.2
#14	F06	MT	MT	36.02	36.1
#16	G06	MT	MT	36.34	36.3
#16	H06	MT	MT	34.2	34.6
#17	A07	MT	MT	36.2	37.8
#17	B07	MT	MT	26.51	28.3

sample	well	BRAF status by pathologist	BRAF status by automatic analysis	Mutant frequency by manual analysis	Mutant frequency by automatic analysis
#18	C07	MT	MT	27.11	27.9
#18	D07	MT	MT	32.79	33.4
#19	E07	MT	MT	40.95	42.2
#19	F07	MT	MT	31.88	33.9
#2	A05	MT	MT	44.94	45.9
#2	B05	MT	MT	42.08	43.3
#21	G07	MT	MT	27.93	28.7
#21	H07	MT	MT	26.98	27.8
#24	A08	MT	MT	54.16	55.3
#24	B08	MT	MT	57.76	58.7
#25	C08	MT	MT	25.57	26.7
#25	D08	MT	MT	21.92	23.2
#27	E08	MT	MT	35	34.9
#27	F08	MT	MT	34.08	34.1
#28	G08	MT	MT	30.45	31.2
#28	H08	MT	MT	29.49	29.8
#29	A09	MT	MT	39.37	40
#29	B09	MT	MT	39.03	39.7
#30-2	A11	MT	MT	31.35	31.1
#30-2	B11	MT	MT	31.52	30.7
#32	E09	MT	MT	34.27	36.1
#32	F09	MT	MT	35.09	35.8
#33	G09	MT	MT	22.35	22.6
#33	H09	MT	MT	21.86	21.9
#34	A10	MT	MT	31	33.3
#34	B10	MT	MT	28.17	29.7
#36	C10	MT	MT	30.01	NA
#36	D10	MT	MT	28.58	NA
#37	E10	MT	MT	10.08	13.2
#37	F10	MT	MT	21.48	21.4
#39	G10	MT	MT	22.7	44.9
#39	H10	MT	MT	36.6	36.3
#4	C05	MT	MT	29.65	30.7
#4	D05	MT	MT	28.74	28.5
#6	E05	MT	MT	14.12	14.8
#6	F05	MT	MT	12.85	13.9
#7	G05	MT	MT	30.02	30.4
#7	H05	MT	MT	25.13	25.9
#9	A06	MT	MT	21.72	21.9
#9	B06	MT	MT	22	22.6

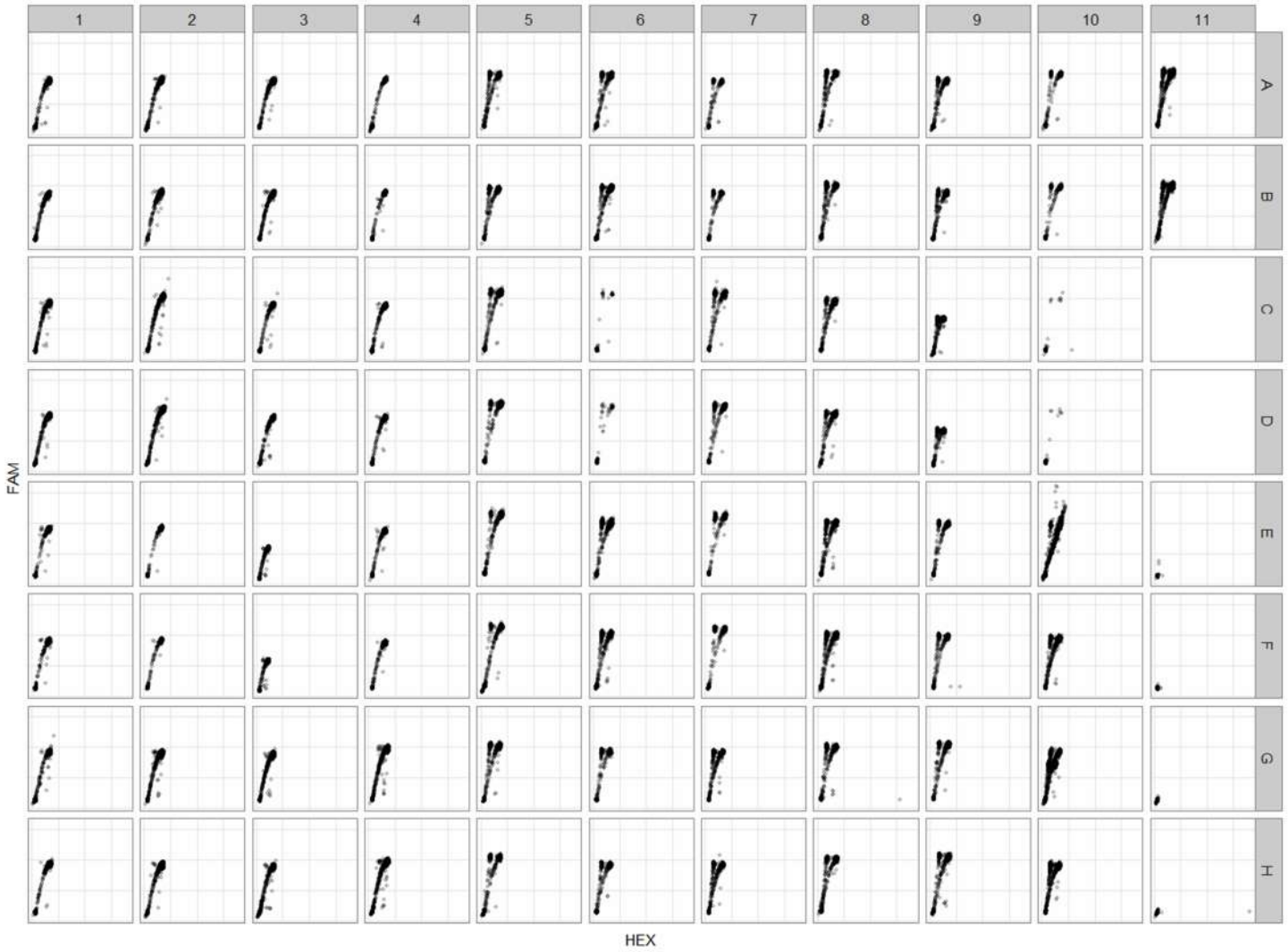

Figure 85 – Raw dataset used in the *BRAF*-V600 case study

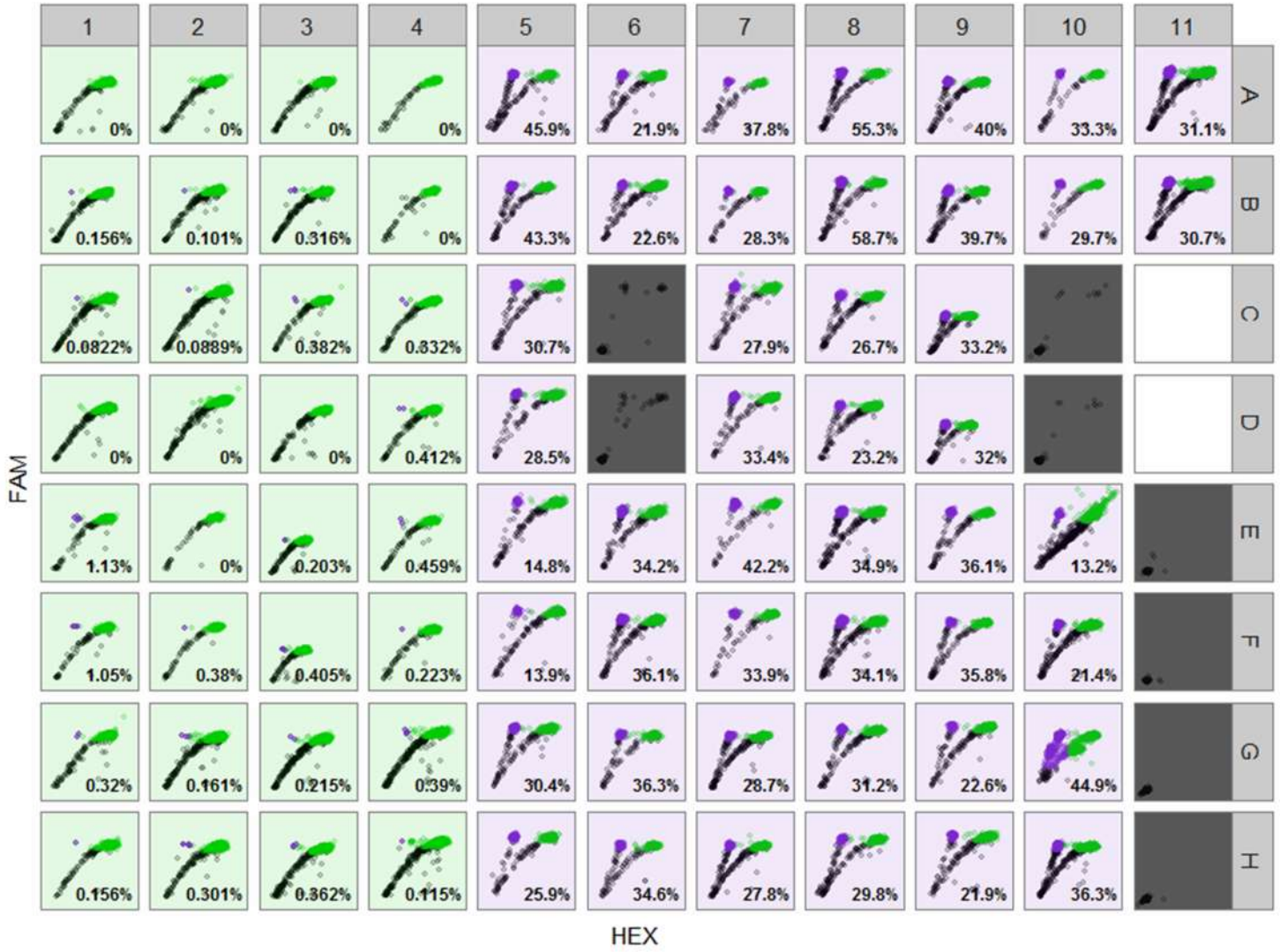

Figure 86 – Fully analyzed dataset in the case study

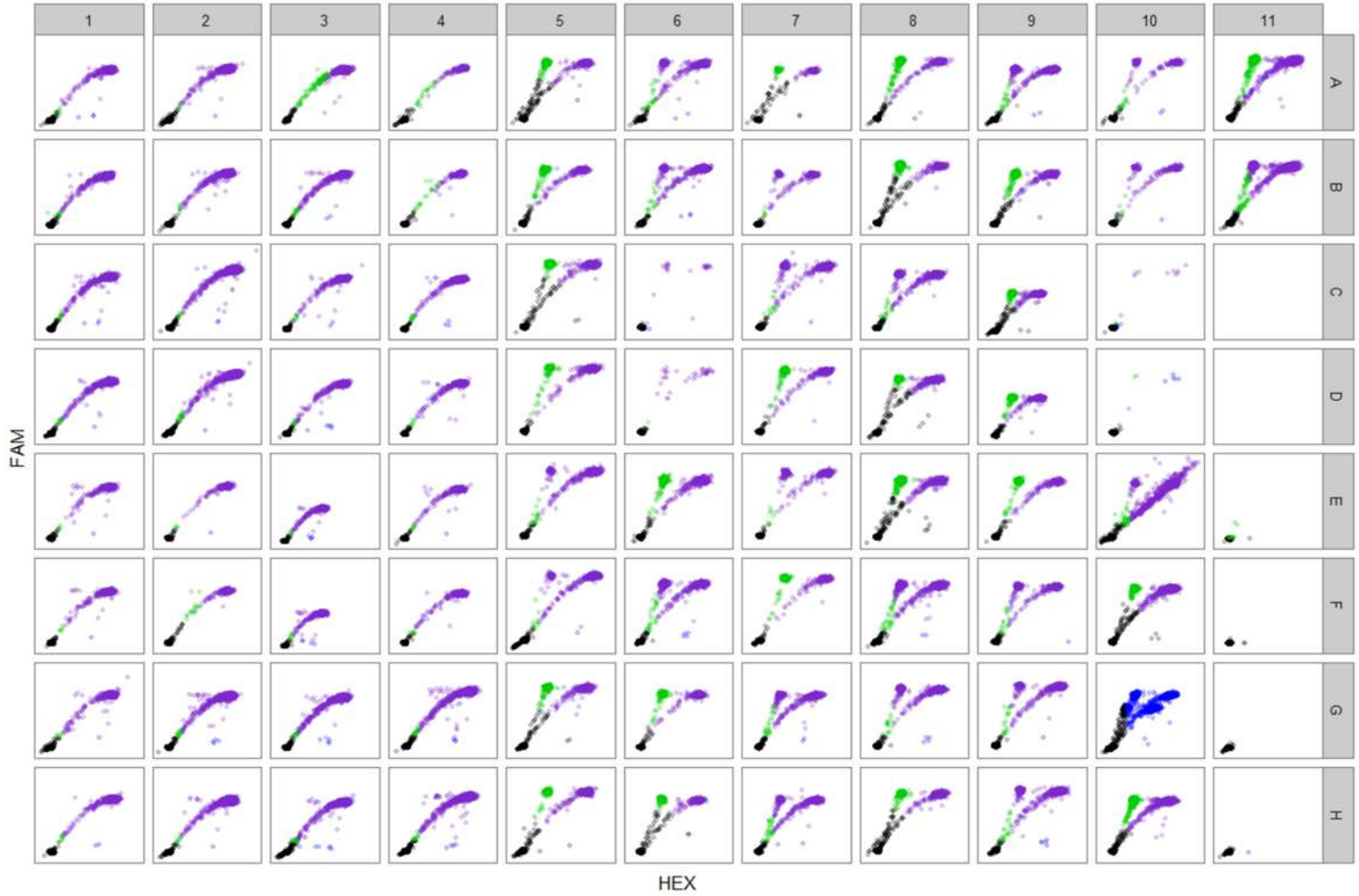

Figure 87 – Automatic droplet gating by QuantaSoft on the case study data

Table 2 – Full results from running *ddpcr* analysis on the case study data

Well	Sample	# drops	Success	# drops outlier	# drops empty	Fraction of empty drops	Concentration	Mutant BRAF status	# mutant drops	# wild-type drops	Mutant frequency
A01	#1	14576	TRUE	0	13884	0.953	52	FALSE	0	608	0
A02	#10	15509	TRUE	0	14437	0.931	78	FALSE	0	966	0
A03	#20	16309	TRUE	0	15284	0.937	71	FALSE	0	893	0
A04	#35	14860	TRUE	0	14652	0.986	15	FALSE	0	166	0
A05	#2	13879	TRUE	0	13273	0.956	49	TRUE	220	259	45.9
A06	#9	14591	TRUE	0	13893	0.952	54	TRUE	131	468	21.9
A07	#17	13868	TRUE	0	13612	0.982	19	TRUE	76	125	37.8
A08	#24	15280	TRUE	0	14637	0.958	47	TRUE	288	233	55.3
A09	#29	14994	TRUE	0	14118	0.942	65	TRUE	307	461	40
A10	#34	14126	TRUE	0	13890	0.983	18	TRUE	61	122	33.3
A11	#30-2	16222	TRUE	0	14570	0.898	118	TRUE	445	984	31.1
B01	#1	17458	TRUE	0	16691	0.956	49	FALSE	1	642	0.156
B02	#10	16398	TRUE	0	15268	0.931	78	FALSE	1	990	0.101
B03	#20	17542	TRUE	0	16436	0.937	71	FALSE	3	945	0.316
B04	#35	14555	TRUE	0	14316	0.984	17	FALSE	0	189	0
B05	#2	15790	TRUE	0	15200	0.963	41	TRUE	211	276	43.3
B06	#9	13655	TRUE	0	12925	0.947	59	TRUE	140	479	22.6
B07	#17	14388	TRUE	0	14136	0.982	19	TRUE	56	142	28.3
B08	#24	16219	TRUE	0	15568	0.96	44	TRUE	311	219	58.7
B09	#29	15344	TRUE	0	14431	0.94	67	TRUE	314	476	39.7
B10	#34	16671	TRUE	0	16435	0.986	15	TRUE	57	135	29.7
B11	#30-2	16256	TRUE	0	14648	0.901	114	TRUE	423	954	30.7
C01	#3	15279	TRUE	0	13903	0.91	103	FALSE	1	1215	0.0822
C02	#11	16619	TRUE	1	15310	0.921	90	FALSE	1	1124	0.0889
C03	#22	17585	TRUE	0	16970	0.965	39	FALSE	2	522	0.382
C04	#38	17689	TRUE	0	16995	0.961	43	FALSE	2	601	0.332
C05	#4	16699	TRUE	0	15995	0.958	47	TRUE	185	418	30.7
C07	#18	16339	TRUE	0	15748	0.964	40	TRUE	140	362	27.9
C08	#25	14801	TRUE	0	14023	0.947	59	TRUE	174	477	26.7
C09	#30	15238	TRUE	0	14339	0.941	66	TRUE	262	528	33.2
D01	#3	15601	TRUE	0	14235	0.912	101	FALSE	0	1234	0
D02	#11	15785	TRUE	0	14482	0.917	95	FALSE	0	1130	0

Well	Sample	# drops	Success	# drops outlier	# drops empty	Fraction of empty drops	Concentration	Mutant BRAF status	# mutant drops	# wild-type drops	Mutant frequency
D03	#22	17628	TRUE	0	16944	0.961	43	FALSE	0	564	0
D04	#38	15096	TRUE	0	14510	0.961	43	FALSE	2	484	0.412
D05	#4	15353	TRUE	0	14743	0.96	44	TRUE	152	381	28.5
D07	#18	15869	TRUE	0	15309	0.965	39	TRUE	162	323	33.4
D08	#25	15139	TRUE	0	14405	0.952	54	TRUE	142	471	23.2
D09	#30	13811	TRUE	0	12992	0.941	66	TRUE	233	495	32
E01	#5	15706	TRUE	0	15188	0.967	36	FALSE	5	436	1.13
E02	#13	16887	TRUE	0	16607	0.983	18	FALSE	0	236	0
E03	#26	18142	TRUE	0	17048	0.94	67	FALSE	2	984	0.203
E04	#40	14324	TRUE	0	13813	0.964	40	FALSE	2	434	0.459
E05	#6	17324	TRUE	0	16797	0.97	33	TRUE	62	358	14.8
E06	#14	14075	TRUE	0	13190	0.937	71	TRUE	268	515	34.2
E07	#19	15542	TRUE	0	15101	0.972	31	TRUE	166	227	42.2
E08	#27	16662	TRUE	0	15557	0.934	75	TRUE	327	610	34.9
E09	#32	16271	TRUE	0	15816	0.972	31	TRUE	134	237	36.1
E10	#37	5013	TRUE	3	4066	0.811	230	TRUE	72	475	13.2
F01	#5	12591	TRUE	0	12142	0.964	40	FALSE	4	376	1.05
F02	#13	17362	TRUE	0	17045	0.982	19	FALSE	1	262	0.38
F03	#26	17538	TRUE	0	16471	0.939	69	FALSE	4	983	0.405
F04	#40	15929	TRUE	0	15415	0.968	35	FALSE	1	447	0.223
F05	#6	16282	TRUE	0	15796	0.97	33	TRUE	54	334	13.9
F06	#14	16262	TRUE	0	15197	0.935	73	TRUE	335	592	36.1
F07	#19	14787	TRUE	0	14391	0.973	30	TRUE	116	226	33.9
F08	#27	17032	TRUE	0	15894	0.933	76	TRUE	331	641	34.1
F09	#32	16329	TRUE	1	15862	0.971	32	TRUE	133	239	35.8
F10	#37	12461	TRUE	0	11408	0.915	97	TRUE	189	694	21.4
G01	#8	17009	TRUE	0	16303	0.958	47	FALSE	2	623	0.32
G02	#15	17092	TRUE	0	15043	0.88	140	FALSE	3	1863	0.161
G03	#31	16594	TRUE	0	15472	0.932	77	FALSE	2	927	0.215
G04	#41	17147	TRUE	0	15908	0.928	82	FALSE	4	1021	0.39
G05	#7	16890	TRUE	0	16173	0.958	47	TRUE	186	426	30.4
G06	#16	14731	TRUE	0	14195	0.964	40	TRUE	172	302	36.3
G07	#21	16622	TRUE	0	15638	0.941	66	TRUE	235	584	28.7

Well	Sample	# drops	Success	# drops outlier	# drops empty	Fraction of empty drops	Concentration	Mutant BRAF status	# mutant drops	# wild-type drops	Mutant frequency
G08	#28	17814	TRUE	1	17070	0.958	47	TRUE	196	432	31.2
G09	#33	14404	TRUE	0	13734	0.953	52	TRUE	126	432	22.6
G10	#39	10369	TRUE	0	9308	0.898	118	TRUE	456	559	44.9
H01	#8	16900	TRUE	0	16184	0.958	47	FALSE	1	641	0.156
H02	#15	15814	TRUE	0	14035	0.888	130	FALSE	5	1655	0.301
H03	#31	14012	TRUE	0	13024	0.929	80	FALSE	3	826	0.362
H04	#41	16603	TRUE	0	15520	0.935	73	FALSE	1	865	0.115
H05	#7	14315	TRUE	0	13693	0.957	48	TRUE	140	401	25.9
H06	#16	16513	TRUE	0	15923	0.964	40	TRUE	178	336	34.6
H07	#21	14537	TRUE	0	13666	0.94	67	TRUE	200	519	27.8
H08	#28	16208	TRUE	0	15526	0.958	47	TRUE	167	393	29.8
H09	#33	16551	TRUE	0	15802	0.955	50	TRUE	138	492	21.9
H10	#39	15361	TRUE	0	13803	0.899	117	TRUE	489	857	36.3
C06	#12	14512	FALSE	0	NA	NA	NA	NA	NA	NA	NA
C10	#36	16404	FALSE	1	NA	NA	NA	NA	NA	NA	NA
D06	#12	17548	FALSE	0	NA	NA	NA	NA	NA	NA	NA
D10	#36	16903	FALSE	0	NA	NA	NA	NA	NA	NA	NA
E11	NTC	14524	FALSE	0	NA	NA	NA	NA	NA	NA	NA
F11	NTC	16862	FALSE	0	NA	NA	NA	NA	NA	NA	NA
G11	NTC	16742	FALSE	0	NA	NA	NA	NA	NA	NA	NA
H11	NTC	14789	FALSE	1	NA	NA	NA	NA	NA	NA	NA