ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Katedra biomedicínské informatiky

Vytvoření nástroje pro přenos expertní analýzy cytometrických dat do open source prostředí pro zlepšení lékařské diagnostiky leukemií a jiných onemocnění

Developement of a tool for transfer of expert analysis of cytometric data to an open source environment to improve the medical diagnosis of leukemias and other diseases

Bakalářská práce

Studijní program:	Biomedicínská a klinická technika
Studijní obor:	Biomedicínská informatika

Autor bakalářské práce:	Michaela Součková
Vedoucí bakalářské práce:	Ing. Bohuslav Dvorský
Externí konzultant:	Mgr. Karel Fišer Ph.D.

Kladno 2021

BACHELOR'S THESIS ASSIGNMENT

I. PERSONAL AND STUDY DETAILS

Student's name:	**Součková Michaela**	Personal ID number:	**483028**
Faculty:	**Faculty of Biomedical Engineering**		
Department:	**Katedra biomedicínské informatiky**		
Study program:	**Biomedicínská a klinická technika**		
Branch of study:	**Biomedicínská informatika**		

II. BACHELOR'S THESIS DETAILS

Bachelor's thesis title in English:

Development of a tool for transfer of expert analysis of cytometric data to an open source environment to improve the medical diagnosis of leukemias and other diseases

Bachelor's thesis title in Czech:

Vytvoření nástroje pro přenos expertní analýzy cytometrických dat do open source prostředí pro zlepšení lékařské diagnostiky leukemií a jiných onemocnění

Guidelines:

The aim is to parse Flowjo (v 10.5.3) workspace file in R language to extract stored specifics of flow cytometry data analysis and subsequently transform and process the raw data stored separately in a file of FCS format. If both done correctly we get both, equivalently transformed data and indices of datapoints corresponding to expertly defined datapoints (cell populations) as "gated" by Flowjo user. This is necessary for combination and comparison of expert manual analysis and computational approaches.

Bibliography / sources:

[1] Adan, Alizada, Kiraz, Baran, Nalbant, Flow cytometry: basic principles and applications, Critical Reviews in Biotechnology , ročník 37, číslo 2, 2017
[2] Ashland, OR: Becton, Dickinson and Company, FlowJo™ Software, 2019, https://www.flowjo.com
[3] Chen, Hasan, Libri, Urrutia, Beitz, Rouilly, Duffy, Patin, Chalmond, Rogge, Quintana-Murci, Albert, Schwikowski; Milieu Intérieur Consortium, Automated flow cytometric analysis across large numbers of samples and cell types, Clinical Immunology, ročník 157, číslo 2, 2015
[4] Bagwell CB, Hyperlog-a flexible log-like transform for negative, zero, and positive valued data., Cytometry A, ročník 64(1), číslo 34-42, 2015, Březen
[5] Parks DR, Roederer M, Moore WA, A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data., Cytometry A, ročník 69(6), číslo 541-51., 2006, Červen

Name of bachelor's thesis supervisor:

Ing. Bohuslav Dvorský

Name of bachelor's thesis consultant:

Mgr. Karel Fišer Ph.D.

Date of bachelor's thesis assignment: **15.02.2021**
Assignment valid until: **18.09.2022**

doc. Ing. Zoltán Szabó Ph.D.
Head of department's signature

prof. MUDr. Jozef Rosina, Ph.D., MBA
Dean's signature

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem „Vytvoření nástroje pro přenos expertní analýzy cytometrických dat do open source prostředí pro zlepšení lékařské diagnostiky leukemií a jiných onemocnění“ vypracovala samostatně a použila k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k diplomové práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně 12.5.2021 Michaela Součková

PODĚKOVÁNÍ

Ráda bych poděkovala Ing. Bohuslavu Dvorskému za vstřícnost a pomoc při vedení bakalářské práce. Mé poděkování patří též Mgr. Karlu Fišerovi Ph.D. za cenné rady, věcné připomínky, trpělivost a čas, který mi věnoval při řešení této práce.

ABSTRAKT

Vytvoření nástroje pro přenos expertní analýzy cytometrických dat do open source prostředí pro zlepšení lékařské diagnostiky leukemií a jiných onemocnění

Cílem je analyzovat Workspace soubor softwaru FlowJo (v 10.5.3) v jazyce R a extrahovat uložená specifika analýzy dat průtokové cytometrie a následně transformovat a zpracovat surová data uložená samostatně v souboru formátu FCS. Pokud je obojí provedeno správně, dostaneme jak ekvivaletně transformovaná data, tak indexy datových bodů odpovídajících expertně definovaným bodům (populacím buněk) v „gatech“ určených uživatelem FlowJo. To je nezbytné pro kombinaci a srovnání odborné manuální analýzy a výpočetních přístupů.

Klíčová slova

Průtoková cytometrie, transformace, FlowJo

ABSTRACT

Developement of a tool for transfer of expert analysis of cytometric data to an open source environment to improve the medical diagnosis of leukemias and other diseases

The aim is to parse FlowJo (v 10.5.3) workspace file in R language to extract stored specifics of flow cytometry data analysis and subsequently transform and process the raw data stored separately in a file of FCS format. If both done correctly we get both, equivalently transformed data and indices of datapoints corresponding to expertly defined datapoints (cell populations) as "gated" by FlowJo user. This is necessary for combination and comparison of expert manual analysis and computational approaches.

Keywords

Flow cytometry, transformation, FlowJo

Table of Contents

1 Introduction

With the rapid improvement of technology in the last decades, the progress of various biological and biomedical analytical approaches was inevitable. One of such techniques is flow cytometry, which is a great tool widely used for example in the fields of immunology, molecular biology, or cancer biology. It offers a multi-parametric analysis of single cells in solution, where there is each cell at a time illuminated by a laser, so the cell produces characteristic scattered and fluorescent light signals, which are then detected, converted to electronic signals, and written in a data file with standardized format (.fcs).

Flow cytometry data are still traditionally analyzed visually (although automatic methods have been on the rise lately), which is one of the reasons why the data preprocessing is in most cases necessary and includes compensation to overcome fluorescence spillover and transformations to transform data from linear to log-like space for easier cell populations distinguishment. The main method of flow cytometry analysis is so-called "gating", a process of identifying cellular populations of interest by manually placing boundaries around similarly behaving cytometric events on 2D plots.

There are multiple flow cytometry software applications currently available. They provide useful tools for both preprocessing of the raw data from FCS files and analytical methods like gating. One of the major ones is FlowJo [1], which is the main topic of this thesis. The analysis environment of FlowJo application is presented as "Workspace", containing details about samples and every analytical step made upon them. All the information about the Workspace, needed for reopening the analysis in the FlowJo application, is saved in a proprietary XML-based file (.wsp).

Here appears the question of interoperability among flow cytometry software or other environments, which is important for the evolution and development of new methods or applications and combining or comparing expert manual analysis with computational approaches.

In this thesis, we explore one specific case of reproducing FlowJo (v10.5.3) analysis in the environment of R language using the FlowJo Workspace file. We map the current situation and available R solutions, but since even the best available tools do not match

the original FlowJo analysis precisely in all cases, we also provide our own R structures, to better examine the issue. To achieve that it is necessary first to understand the structure of FlowJo Workspace XML file and determine the importance of the elements and attributes and also understand how FlowJo handles transformations and their application.

2 Overview of the current state of art

2.1 Flow Cytometry

Cytometry is a cell analysis technique, and it refers to the measurement of physical and chemical characteristics like relative size, relative granularity, internal complexity, and relative fluorescence intensity of cells or other biological particles. Simply said, flow cytometry has a unique ability to simultaneously analyze mixed populations of cells for multiple parameters [2].

Cells suspended in a fluid flow one at a time through a focus of exciting light (laser), which is scattered in patterns characteristic to the cells and their components; cells are frequently labeled with fluorescent markers so that light is first absorbed and then emitted at altered frequencies [3]. Light collection optics are focused on the intersection point of cells with the laser beams to pick up fluorescence and scattered light from the cells [4], which is then detected and converted into electronic signals that can be processed by the computer.

The value of the technique lies in the ability to make measurements on large numbers of single cells within a short period of time. The heterogeneity of populations can be revealed and different subsets of cells identified and quantified. Selected cell populations can also be physically sorted for further study [4].

Flow cytometry is used in a variety of fields from molecular biology and virology to medical applications e.g. tumor immunology, genetics, or prenatal diagnosis [5] [6].

2.2 Flow cytometry data analysis

The process of analyzing flow cytometry data can be divided into several steps [4]:

- Signal processing
- Extracting and converting parameters into a *.fcs file
- Data pre-processing - compensation, transformation
- Cell population identification - commonly known as "gating"
- Further analysis

2.2.1 Signal Processing

Photomultiplier tubes are used as light detectors to convert photons to photoelectrons which are then multiplied and subsequently processed by amplifiers and analog-to-digital converter.

In the digital domain, the signals are processed by filters, baseline restorer, pulse height, pulse width algorithms, and trigger, mainly to reach an optimal "signal to noise ratio" and "dynamic detection range" of the cytometer. The baseline restorer attempts to keep the baseline at zero. In practice, however, baseline restoring is not perfect and can lead to negative values on the histogram axis or introduce a slight distortion of low signals [4].

Also, Photomultiplier tubes (or sometimes used avalanche diodes) "*are unsuitable for wavelength detection, hence the fluorescent light needs to be filtered by optical filters and mirrors. These filters must be carefully chosen because a multiparameter experiment, i.e. an experiment in which multiple parameters (markers) are analyzed, requires that multiple fluorophores are used simultaneously; a consequence of this is spectral overlap or spillover. Conventional flow cytometers circumvent this problem by compensation in order to accurately correlate the physical light properties with the biological properties of the cell.*", says Cossarizza [4] (See section 2.2.3 Data pre-processing).

After baseline restoring, the pulse parameters (height, width, and area) are extracted and converted into a *.fcs file.

2.2.2 FCS file

Flow Cytometry Standard (FCS) is a file standard that provides the specifications needed to completely describe flow cytometry data sets including the instrument used to obtain the data, the sample measured, the data obtained, and the results of analysis of the data [7]. FCS was developed (the first version was published in 1984) and is maintained by the International Society for Advancement of Cytometry (ISAC). The main goal is to create a standardized way of passing experiment data between instruments and computers and thus facilitate the development of software and analytics tools [8].

FCS file, written as a continuous byte stream, can contain one or more data sets, where each data set consists of at least four required segments: HEADER, TEXT, DATA, and ANALYSIS.

- HEADER segment contains information about the version of FCS and pointers specifying where the other three parts begin.
- TEXT segment carries series of key-value pairs. There are no default values for keywords, but some are required for a valid FCS file. These mandatory keywords describe the DATA segment format and encoding, containing all the information necessary to read the data part of the data set and should not be redefined by the user. Users may define other keywords for their own use.
- The format of data in DATA segment can be either ASCII, binary integer, or floating point, as defined in keywords by the user. Data are structured in a matrix with "columns" corresponding to parameters and "rows" corresponding to electronic events [9].
- ANALYSIS segment contains information added to the file after it was collected and stored. Its key-value structure is the same as in the TEXT part. [7]

2.2.3 Data pre-processing

The FCS files, containing all important information about the conducted experiment, can be then easily exported into other data analysis software for flow cytometry. There are several software packages available, e.g. Cytobank, BD FACSDiva™, or FlowJo™, offering various tools for flow cytometry data processing and analysis.

Before we proceed to the data analysis and population identification itself, there are several steps required to eliminate artifacts and optimize scaling for easier visual analysis. Two main operations, which are important for the scope of our work, are mentioned below.

Compensation

Compensation is a mathematical method that overcomes fluorescence spillover, which is described by Cossarizza [4] as "*the amount of signal, measured in median fluorescence intensity (MdFI), that a fluorochrome emits in a secondary detector specific for a different fluorochrome*". The degree of spectral overlap between fluorochromes and detectors can

be recorded and used to create a compensation matrix containing the spillover values which provide effective correction [10].

It is important to mention, that the process of compensation can, and commonly does, result in negative values, which can seem counterintuitive, but negative fluorescence can be caused by e.g. autofluorescence, which occurs to some extent in all cell types. This issue is described in detail by Anja B. Bohn, Bjarne K. Moller, and Mikkel S. Petersen in their article Flow cytometry and compensation of highly autofluorescent cells: the example of mesenchymal stem cells [10].

Transformations

Traditional flow cytometry data analysis is based on visual analysis, therefore the data should be well-arranged. However, the distribution of cytometry data is mostly log-normal, as is common to biological systems.

The log-normal distribution is a continuous probability distribution of a random variable whose logarithm is normally distributed [11]. Therefore, if the data were presented on a linear scale, it would be difficult, if not impossible, to conduct visual analysis, as we can see in Figure 2.1 A and Figure 2.1 B. To visualize the data, flow cytometry data analysis includes the use of one- or two-parameter histograms, that have traditionally been plotted with either a linear or logarithmic scale. The logarithmic transformation has been found to be useful because it allows for the visualization of at least four decades of dynamic range on a single graph, and it makes log-normal distributions appear more symmetrical [12].

However, the logarithmic transformation has few known disadvantages. Zero and negative valued data are undefined and must be truncated to zero, leading to compression of data against the axes and poor visual representation of low intensity or unstained populations. That makes it difficult to appreciate them as a separate population [13]. This

effect has been a source of considerable confusion and has been commonly referred to as the "log artifact" [14].

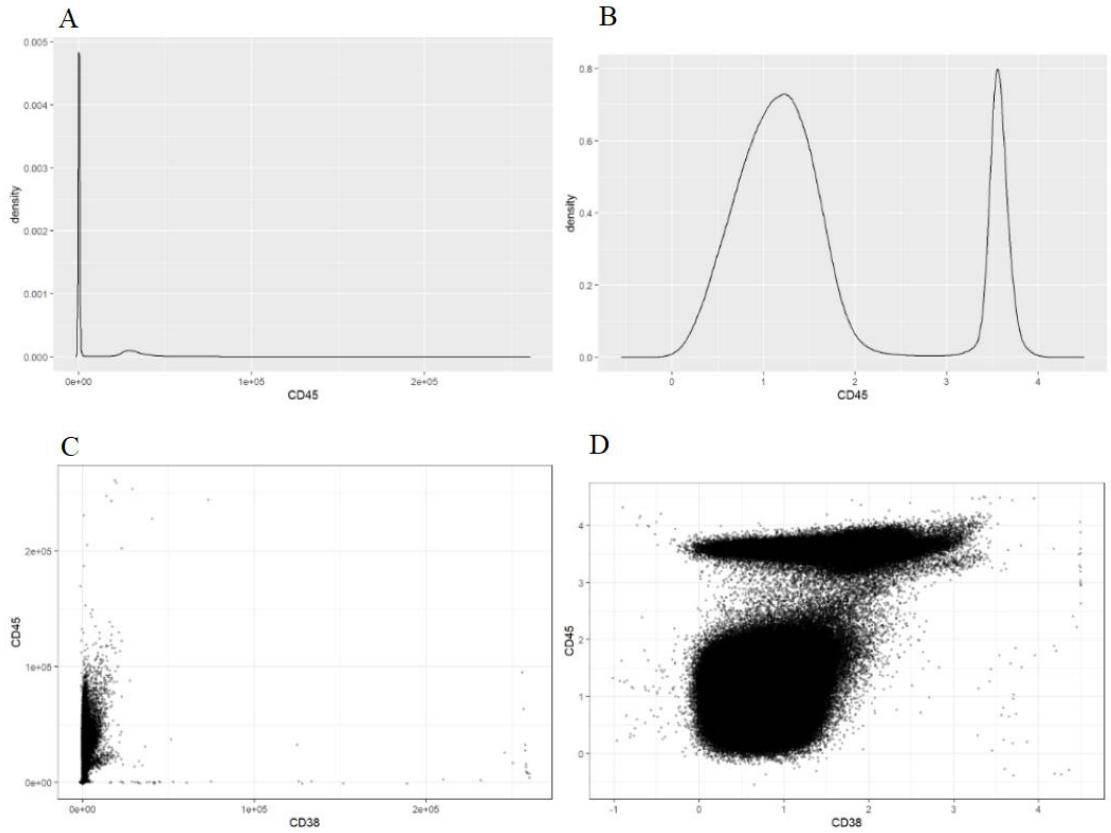

Figure 2.1: A: Density plot of compensated data before applying scale transformation, B: Density plot of compensated and transformed data, C: Two-parameter dot plot of compensated data before transformation, D: Two-parameter dot plot of compensated and transformed data; created by the author

In the field of cytometry, this issue is important because the process of compensation, that commonly precedes the process of transformation and involves subtraction or translocation, resulting in the negative, zero, and positive values, as we mentioned previously. The other characteristic of compensated data is a translocation of a population by subtraction along one or more of its parameters to or near the origin of an axis without significantly changing its variance and logarithmic transform is particularly unsuited for this type of data. An optimal transform for compensated data must be able to handle both characteristics [13].

To deal with the issue of negative and zero values, other transformations have been suggested, including the biexponential, logicle, arcsinh or hyperlog, with their formulas in Table 1.1. These all improve upon the log by allowing one to view a large dynamic range of data on a single plot and also allowing negative values, demonstrated in Figure

2.1 B and Figure 2.1 D. They provide a linear representation of data around zero and a logarithmic representation of the data at higher intensity values, with a smooth transition between the two extremes [15]. The linear scaling of the lower values serves to limit the effective resolution of the histogram, thus minimizing the valley and picket fencing artifacts [12]. The mentioned transformations are described more in detail in section 2.3.

Table 2.1: Mathematical formulas of scale transformations supported by FlowJo [13] [14]

Transformation	Formula
Logarithmic	$f(x)=log_b x\frac{r}{d}$
Arcinh	$f(x)=asinh(a+bx)+c$
Hyperlog	$f^{-1}(x)=ae^{bx}+cx-f$
Biexponential	$f^{-1}(x)=ae^{bx}-ce^{dx}+f$
Logicle	$f^{-1}(x)=ae^{bx}-ce^{-dx}+f$

2.2.4 Gating

Flow cytometry data analysis is built upon identifying populations of cells, i.e. cells with similar characteristics such as forward scatter, side scatter and marker expression. This process is called "gating" and consists of placing boundaries around the cells of interest. Manual gating is the traditional and wildly used approach, which identifies cell types by user-defined regions on user-defined 2D plots, see Figure 2.2 for an example. Gating can be hierarchical, meaning the gate can be based on another one and become its child.

The main drawback of the manual approach mentioned by Peng Qiu is that it "*relies on prior knowledge of the protein markers and visual inspection of the data, which is subjective, labor-intensive, and difficult to reproduce*" [16]. Also, with the recent advancement of flow cytometry also grew the amount and complexity of produced data and measured parameters, which makes the analysis much more time-consuming and difficult, when only two parameters can be displayed at a time [17]. These disadvantages motivated the development of automated clustering algorithms for objective and reproducible analysis.

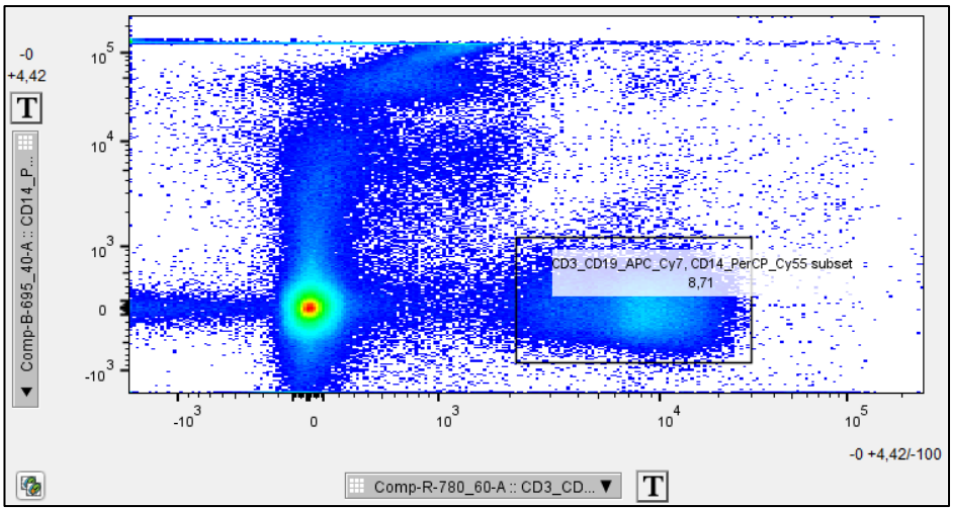

Figure 2.2: Example of manual gating in FlowJo v 10.5.3, created by the author

2.2.5 Further analysis

Reproducibility of analytical steps and interoperability among flow cytometry data analysis tools is important for the development of advanced methods and collaboration in the flow cytometry community. Even though FCS standard supports the description of compensation, transformations, and gating, it is insufficient for capturing the post- acquisition operations and therefore does not meet the need for standardization.

One of the solutions came from International Society for Advancement of Cytometry, as they developed Gating-ML [9].

Gating-ML

Gating-ML ("Gating Markup Language") is an XML-based specification for formal description of gates, but also allows encoding of transformations and compensation, which are crucial for the reconstruction of the analysis as well.

There are three XML schemas available to validate Gating-ML XML documents: Gating- ML.v2.0.xsd, Transformations.v2.0.xsd and DataTypes.v2.0.xsd [18].

Gating-ML supports rectangular gates, quadrant gates, polygon gates, ellipsoid gates, and Boolean collections of any of the types of gates. Gates can be uniquely identified or ordered into a hierarchical structure, also they can be applied on either raw data or transformed events as described by an explicit scale transformation. The supported transformations are logarithmic, polynomial of degree one, square root, asinh, split-scale,

Hyperlog, Logicle and ratio of two parameters, as well as inverse transformations wherever these exist. Firstly Logicle/Biexponential was not included in Gating-ML "*as it is covered by a patent and is licensed under restrictive conditions*" and such components were not supported, according to Josef Spidlen [9], but eventually the Logicle transformation was involved as well.

Gating-ML defines transformations in terms of consistent parametrization to make them more comprehensible and more related to user experience. Each transformation is described with all or some of the following parameters:

- *T* – standing for the top scale value and is always mapped 1
- *M* – the number of decades for logarithmic and log-like transforms
- *W* – controlling the degree of linearization (for Logicle and Hyperlog)
- *A* – specifying an additional range of negative data values

According to Josef Spidlen "*the Logicle, Hyperlog, and parametrized inverse hyperbolic sine transforms with A= 0 will all behave like the logarithmic transform with the same values of T and M for large data values*". Therefore e.g. the Logicle transform can be a reasonable alternative to Hyperlog with the same parameters and vice versa.

Rectangular gates can represent range gates (one dimensional), rectangular gates (two dimensional), box regions (three dimensional) or hyper-rectengulat regions (more than three dimensional). Parameters describing rectangular gates are *min* and *max* attributes. Events, which are considered to be in the gate, lay between them, with the intervals treated as half-open with weak inequality of the lower bound and strict inequality of the upper bound.

Polygon gates are defined by at least three vertices with the boundaries drawn between consequent vertices in the same order as the *vertex* elements carrying their coordinates. The polygon is automatically closed, so the last boundary is drawn from the last to the first vertex. Events inside the polygon and on the boundary are considered to be in the gate [18].

2.3 Transformations

All four transformations, biexponential**,** logicle**,** arcsinh and hyperlog – follow a similar idea, combining linear scaling for values around zero and log-like for higher values, as described below, shown in Figure 2.3. The difference is mainly in their ability to smoothly transition between the two scalings and the flexibility to adapt the linear section.

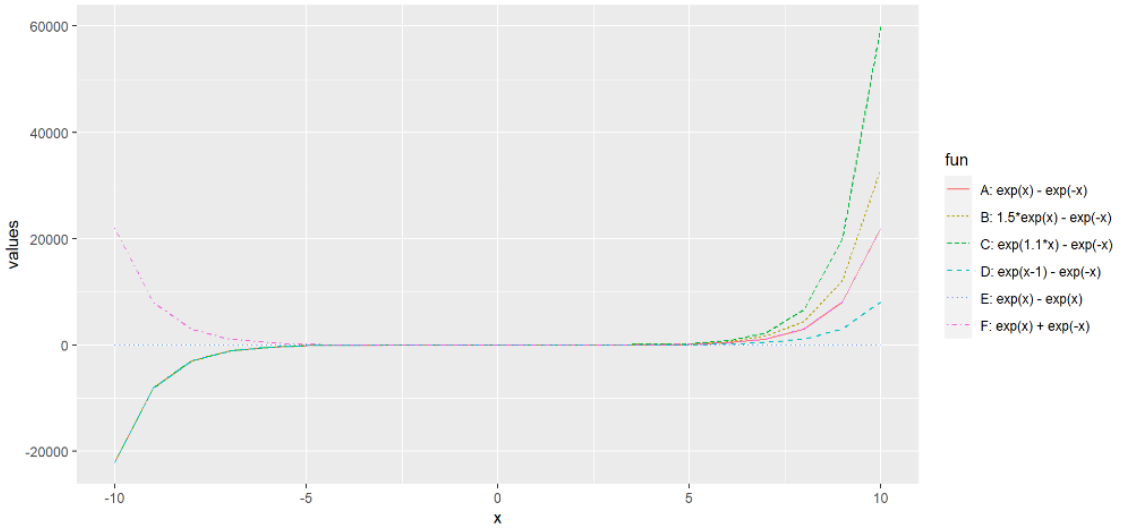

Figure 2.3: Simplified example of biexponential function and the effect of different parametrization (before inversion), created by the author

If the parametrization is picked correctly, each population is represented as one peak, but wrong parameters can distort the display or even split the peaks and lead to misinterpretation of the data, shown in Figure 2.4. Estimation of parameters is possible for some of the transformations.

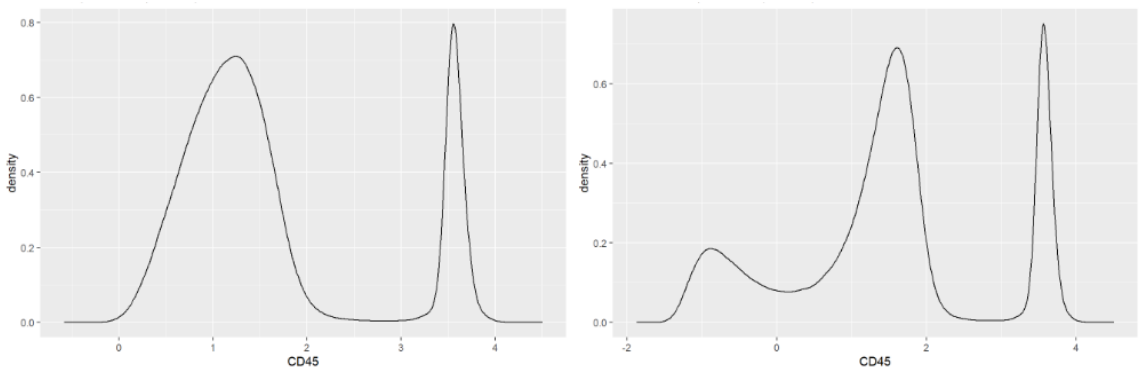

Figure 2.4: On the right is a density plot of correctly transformed data, on the left is a density plot of data transformed with incorrect parametrization, splitting one population into two peaks, created by the author

2.3.1 Biexponential

Biexponential scaling helps visualize data that is compressed against the low x- and y- axes. It provides display settings that are, in effect, a combination of linear and logarithmic scaling techniques. For values around zero, the scaling is displayed as if it was linear. For large values, the biexponential function displays the data with more log-like properties. Therefore, distributions with small magnitude values and negative values can be shown alongside large magnitude, and large variance distributions [19]. The biexponential transform provides additional flexibility by allowing the linear portion of the scale to be asymmetric around zero [15].

According to flowCore function *biexponentialTransform()* documentation, the *biexponential* is an overparameterized inverse of the hyperbolic sine. The function takes the form

$$f^{-1}(x) = a * e^{b*(x-w)} - c * e^{-d*(x-w)} + f$$

with default parameters selected to correspond to the hyperbolic sine (a = 0.5, b = 1, c = 0.5, d = 1, f = 0, w = 0) [20]. Parameters *a* and *b* affect one side of the curve, while c and d the other. If either *a, b, c,* or d gained negative value, the shape of the function would change dramatically, and it would lose its desired effect. Therefore, they should be kept positive, or other parameters should be changed accordingly. Selecting the parameters too low can lead to artificial population splitting when one population, represented with one peak, is graphically split into two. That should be minimized to unambiguously appreciate separate clusters.

In FlowJo documentation we can find that there are three settings that can be set when optimizing histogram with biexponential transformation: *n* stands for negative decades, *p* stands for positive decades and *w* for width basis, where *w* is considered as crucial.

The value of *w* determines the number of channels to be compressed into linear space around zero. The space of linear does not change, but rather the number of channels or bins being compressed into the linear space [19]. But no equation or definition of the biexponential transformation was included.

2.3.2 ArcSinh

ArcSinh transformation serves a similar purpose as biexponential, in fact, it is its generalized version. The adjustable argument for this transformation is called the “scale

argument." For data values on either side of zero to the magnitude of the scale argument, data are displayed in a linear-like fashion tracking with the raw data values. For values beyond the scale argument, data are displayed in a log-like fashion. Arcsinh values are calculated by applying the arcsinh equation divided by the scale argument to the measured intensity value. The scale argument defines the size of the linear region around 0 [21]. Inverse hyperbolic sine transform class arcsinhTransform in flowCore represents a transformation defined by the function

$$f(x, a, b, c) = \text{asinh}(a + b * x) + c$$

where all parameters *a*, *b* and *c* should be selected positive. Parameter *a* corresponds to a shift about 0, parameter *b* corresponds to a scale factor. The transformation would normally be used to convert a linear valued parameter to the natural logarithm scale. By default, *a* and *b* are both equal to 1 and *c* to 0 [20]. Selecting the parameters too low can also lead to artificial population splitting into two peaks, similarly to biexponential transform.

2.3.3 Hyperlog

Hyperlog is a log-like function developed by Bruce Bagwell that approximates the log transform with a function that matches the log transform above zero but has an inverse that can be used for ranges below zero [19]. In 2003 the HyperLog was released in software and later presented as a log-like transform, that admits negative, zero, and positive values, optimized for compensated data. The HyperLog transform was engineered as a hybrid function. One component of the function is ideally suited to the multiplicative nature of compensated data, and the other to its translocated nature. Its ability to smoothly transition between exponential and linear scales gives it desirable features, depending on the analysis or graphics needs.

Bagwell's Hyperlog is defined [13]:

$$HL(x, b) = root(EH(y, b) - x)$$

Where the EH is the inverse of hyperlog transform and is given by:

$$EH(y,b) = 10^{\frac{d}{r}*y} + b * \frac{d}{r} - 1, \qquad for\ y \geq 0$$

$$EH(y,b) = -10^{\frac{-d}{r}*y} + b * \frac{d}{r} + 1, \qquad for\ y < 0$$

A parameter in flowCore definition corresponds to the *r* parameter in Bruce Bagwell's definition, which stands for the analog-to-digital resolution. Coefficient *b* affects the transform through the origin and both *a* and *b* should be selected positive. Setting $b = 0$ often splits negative populations into two peaks and increasing the coefficient can eliminate that. An appropriate *b* coefficient can be either set manually or calculated.

2.3.4 Logicle

In 2006 David R. Parks, Mario Roederer and Wayne A. Moore reacted to the need of a better approach to scaling and displaying flow cytometry data than the traditional logarithmic or linear scaling could produce. They developed criteria for defining a new scaling function and concluded that particular generalizations of the hyperbolic sine function (sinh), which they called Logicle functions, can best meet those criteria [14]. Therefore, with correct parameters, Arcsinh is completely equivalent to Logicle transformation and it is not strictly necessary to implement them as separate classes [20]. Logicle is defined by the function

$$f(x,T,W,M,A) = root(B(y,T,W,M,A) - x)$$

Where B is a modified biexponential function [18]

$$B(y,T,W,M,A) = ae^{by} - ce^{-dy} - f$$

2.4 FlowJo

FlowJo™ is a flow cytometry software application currently maintained by FlowJo LLC, a subsidiary of Becton Dickinson. The application provides an integrated environment for management, display, manipulation, analysis and publication of the large data stream, accepting data from various flow cytometers or single-cell sequencing technology (supporting not only .fcs files but also e.g. .acs, .csv or .zip). FlowJo provides useful tools

for creating and applying compensation matrices, transforming data, visualizing results and also manual identification of cell populations (gating).

Supported transformations are *biexponential*, *logical*, *arcsinh*, *hyperlog* and *miltenyi*, with *biexponential* being defaulty applied on data which require transforming, offering the user an option to change it.

Provided gating options include:

- Rectangle,
- Quad (dividing data into four rectangular gates sharing a common center point, used to divide non-overlapping populations),
- Ellipse (for round populations),
- Polygon,
- Drawing with a pencil,
- Curly Quad (with an exponential curve in each dimension),
- Spider Gates (abutting polygons sharing common vertices),
- Auto Gate (creates a polygon matching the distribution of the events in these two dimensions).

2.4.1 FlowJo Workspace

The FlowJo environment is presented as Workspace, which contains a list of all samples, including their analysis (e.i. gates and statistics) and also graphic and tabular layouts. The Workspace is described in a single XML-based document with .wsp or .jo file extensions, and its main purpose is to save everything needed for reopening the environment in FlowJo application window. However, there have been attempts to use the Workspace for reproducing the analysis in other platforms.

Unfortunately, this Workspace document does not entirely follow the Gating-ML specification, even though the structure is very similar. One of the reasons can be the fact, that Gating-ML does not include biexponential or Logicle transformations [9] or other operations, which are often used in FlowJo.

We did not manage to find any official documentation or description of the Workspace file, which makes reproducing the FlowJo analysis in another platform more difficult.

Therefore, in this work the meaning assigned to each element or attribute in the file is partly based on our assumptions and partly on Gating-ML documentation.

2.5 Currently available tools parsing FlowJo Workspace file in R

There are numerous tools in environments of several programming languages for importing the FlowJo Workspace file and reproducing the analytical steps that it describes. Here we focus on packages working in R.

A wildly used group of R packages focusing on flow cytometry data analysis in R was developed and is maintained by Raphael Gottardo's Research Lab (RGLab). These packages offer many useful tools for analysis and overall management of flow cytometry data (e.g. packages flowCore, OpenCyto and more), also including the ability to import and parse FlowJo Workspace (packages flowWorkspace, CytoML and cytolib which backs the R packages flowCore, flowWorkspace, CytoML, and others). Most other available tools import these packages and use them for the parsing itself, or do not guarantee compatibility with new versions of FlowJo (in our case v 10.5.3).

2.5.1 CytoML

CytoML is a R/Bioconductor package created by RGLab for the import and export of gated cytometry data, supporting not only files (.xml or .wsp) from FlowJo, but also from Diva or Cytobank, therefore allowing sharing data across different platforms. With usage of other packages Cyto-ML maps the different analytic objects, including compensation matrices and transformations, from Workspace files to the core cytometry data structures (which are defined in flowWorkspace package) in R and provides options for visualization (ggcyto package) and modification (openCyto package) of the imported data [22].

2.5.2 FlowWorkspace

FlowWorkspace is another RGLab R package, which is described as a tool which "*is designed to store, query and visualize the hierarchical gated flow data. […] Gating hierarchies, groups of samples, compensation, and transformation are performed so that the output matches the flowJo analysis.*" FlowWorkspace uses CytoML package for

the process of parsing the FlowJo workspace and serves explicitly for interacting with the imported data.

However, the output is not always equivalent to the original analysis, meaning that the number of events (e.g. cells) belonging to a particular gate described in Workspace differs from the number of events for the same gate in the output. The developers mentioned the possibility of slight differences due to numeric flaws, which should not be significant. Nevertheless, these little deflections might, in some cases, be crucial, since the count of the cells of interest in the sample can be very low.

Furthermore, in some cases, the difference can be significantly higher, with the output count being even double the value of the FlowJo count. This issue was the main reason, why FlowWorkspace was not considered as fully sufficient tool for parsing the Workspace. As we will discuss later, this problem was solved in the process.

3 Aims

FlowJo software plays a major role in the field of flow cytometry and reconstructing the FlowJo analysis, recorded in a proprietary xml-based file, in R environment would be very useful for the advancement of flow cytometry analysis, facilitating the development of new approaches. Many attempts appeared during last years, with FlowWorkspace R package as the currently best tool available, but their results always matched the FlowJo analysis with smaller or larger deviations, thus the need of a reliable solution still remains.

The aims of this work are:

1. Map the current situation of reconstructing the FlowJo analysis in R and currently available solutions.
2. Understand the structure of FlowJo Workspace file and determine the importance of XML elements and attributes.
3. Transfer the relevant structures of Workspace in R objects.
4. Examine how FlowJo handles transformations and their application.
5. Develop R tool with foundational structures and functionality for reproducing described analysis.

4 Methods

The process which starts with importing FCS and Workspace files and ends with correctly compensated, transformed and gated data stored in R structure can be described in three main steps, all in R:

1. Parsing Workspace file containing the description of conducted analytical operations and extract important information (see section 4.4)
2. Apply compensation and transformations on raw data from FCS file (see section 4.5)
3. Apply gating and defining which data points (cells) were gated (see section 4.6)

The only criterium, which we can use to evaluate the accuracy of the result, is the count of events inside the gates. That information can be found in Workspace file and is used for comparison with the count we reach in our solution.

4.1 R

R is a system created for statistical computation and graphics consisting of a programming language and an environment with graphics, debugger and the ability to run R scripts. R is suitable for processing large data sets, complex statistical procedures and offers many tools for various kinds of data presentations. Also, it does not only support functions written in R but it is possible for the user to interface to procedures written in the C, C++, or FORTRAN languages for efficiency [23].

4.2 R packages

CRAN (The Comprehensive R Archive Network) is a network of ftp and web servers around the world providing up-to-date packages and code with documentation for R. Currently, the CRAN package repository features 17401 available packages [23]. CRAN packages used for our work (except the base packages) are *XML* and *xml2*, offering tools for parsing and exploring XML documents.

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data, using R language. Contained bioinformatics packages are open-source and development-free and in our work we use *flowWorkspace*, *CytoML* and *flowCore*, mainly for compensation and transformations.

4.3 Testing files

FlowJo Workspace files used for testing and examples in this thesis were created in FlowJo v10.5.3. All transformations or gates applied on raw data are not correct in terms of flow cytometry analysis, since the accuracy of the analytical steps is not necessary here and beyond the scope of this thesis.

FCS files, containing measured patient data, used for testing and the examples in this thesis can not be published, but in the enclosed script we included example FCS file from FlowWorkspaceData package.

4.4 Parsing Workspace

As mentioned earlier, FlowJo Workspace file is XML-based and its structure is inspired by Gating-ML standard, yet does not follow it entirely, without providing any official documentation.

The Workspace file contains information about everything needed to reconstruct the analysis in FlowJo software, including graphics or layouts. In our case we only need parameters of transformations and gating. Compensation is done using CytoML functionality, so extracting its parameters is not necessary. All the needed data can be found in elements named *Population*, which are children of element *SampleList* and then *transforms* element, which is a child of element *Transformations*. *Population* element contain details about gated population, an example is shown in Code 4.1. The attributes carry its name and the count of events inside the certain gate. Children element *Gate* then describes the gate itself – its type (e.g. rectangle or polygon), ID, FCS dimensions, on which the gate is applied, and parameters needed for reconstruction of the gate depending on its type. FlowJo offers a little bit different set of gate types then Gating-ML, but the description of those, which are common to both are very similar. In case of hierarchical gating, the subpopulations are placed inside their parent population.

```
<Population name="population1"  count="136" >
      <Gate gating:id="ID2061669528" >
            <gating:RectangleGate>
                  <gating:dimension gating:min="-
                  1827.1807162836017"  gating:max="-
                  780.9045038129277" >
                        <data-type:fcs-dimension data-
                        type:name="Comp-R-780_60-A" />
                  </gating:dimension>
                  <gating:dimension gating:min="-
                  456.195630362662"
                  gating:max="818.5352182916166" >
                        <data-type:fcs-dimension data-
                        type:name="Comp-B-610_20-A" />
                  </gating:dimension>
            </gating:RectangleGate>
      </Gate>
      <Subpopulations>
            <Population>
                  <Gate gating:parent_id="ID2061669528">
                  </Gate>
            </Population>
      </Subpopulations>
</Population>
```

Code 4.1: Simplified example of Population element

Each *transforms* element represents one transformation applied on one FCS dimension, holding transformation type, its parameters and name of the dimension (as an attribute of *data-type:parameter* element). An example of *transforms* element is shown in Code 4.2.

```
<transforms:biex transforms:length="256"
transforms:maxRange="262144.0000291775"  transforms:neg="0"
transforms:width="-100"  transforms:pos="4.418539922" >
      <data-type:parameter data-type:name="R-780_60-A" />
</transforms:biex>
```

Code 4.2: Example of transforms element describing biexponential transformation

4.5 Applying compensation and transformations

With Workspace parameters extracted, we can proceed to compensation and transformations, which are applied on raw data from FCS file. The best results were reached, when these steps were applied in the following order: compensation, transformation, and subsequently gating.

Opening and reading the FCS file is done by FlowCore function *read.FCS()*, storing FCS in a *flowFrame* structure, which allow extracting raw measured data to a matix. For handling the whole process of compensation, we decided to use CytoML function *compensate()*. Both functions are considered flawless. Transformations parameters, obtained earlier, are then used to transform the raw data and also the parameters describing gating.

In our work we mainly focus on *linear* and *biexponential* transformations, which FlowJo defaultly uses on displayed data. Gating-ML does not support *biexponential* transformation and also the description of majority of the other transformations differ.

Each transformation recorded in XML Workspace file has attributes carrying information about its kind, numeric parametrization values and name of dimension/parameter on which it is applied. The main encountered issue was determining the importance of each FlowJo transformation parameter. Neither *linear* nor *biexponential* transformation is included in Gating-ML description and their parameters do not match the parameters of their mathematical functions if they are known.

The only source of information is the open source packages solving *biexponential* transformation and their solutions, thus we used FlowWorkspace function *flowjo_biexp()*.

Table 4.1: The difference between parameters of scale transformations supported by FlowJo

Transformation	Parameters of mathematical formula	FlowJo parameters
Linear		minRange, MaxRange
Biexponential	a, b, c, d, f, w	maxRange, neg, width, pos
Arcsinh	a, b, c	maxRange, T, A, M, W
Hyperlog	b	maxRange, T, A, M W
Logicle	t, a, m, w	T, A, M, W

4.6 Applying gating

FlowJo supports 6 types of gating, but in our solution there are only two of them included – the rectangular and polygon. The implementation follows the rules described by Gating-ML specification.

5 Implementation

The main goal of our solution is to create a tool, which imports the FCS and Workspace files as an input and creates a suitable representation of the data and conducted analysis as an output. That requires parsing the Workspace file, importing FCS data and applying compensation, transformation and gating, as discussed above.

Our solution combines object oriented and procedural programming, where the main classes are *SamplePopulations*, *RectPopulation* and *PolygonPopulation.*

5.1 SamplePopulations class

The *SamplePopulations* class is a R6 class that serves to store data both from FCS and Workspace file, initialize the process of reconstructing the analytical steps described in Workspace and present the outcome.

The object of *SamplePopulations* class requires two arguments when constructing a new instance:

- The first argument is *fcsfile* – full path to the XML FlowJo workspace file
- The second is *wsfile* – full path to the FCS file

It has 6 public methods accessible for user after initialization:

- *getPopulations()* – returns a list of *Population* instances containing all popaulations described in Workspace
- *getAllPopsLogical()* – returns a matrix where each column represents one population's *gatedLogical* vector, thus it contains information about which events belong to the gates
- *compareAllCounts()* – returns a data frame with the counts of events that should belong to the gates according to Workspace XML and the counts of events we reached
- *printHierarchy()* – returns a list of strings, where each describes a path to the population in the gating hierarchy
- *getFcsDims()* – returns a matrix containing data from FCS file after compensation and transformation

During the initialization of a *SamplePopulations* instance the FCS file is opened and the raw measured data compensated and transformed. The Workspace file and modified data are then passed to a function *getGateClasses(),* which returns a list of *Population* instances. Both steps – transformation and creating *RectPopulation* or *PolygonPopulation* instances, require parsing the Workspace file first and then using the extracted parameters.

5.2 Parsing Workspace file

Using packages *XML* and *xml2* functionality we parse the document in three separate functions:

- *xmlToPopsList()* returns a list containing each single population described in *Population* elements as a nested list carrying all the information stored in XML.
- Similarly *getTransforms()* returns list of transformations.
- *getPopPath()* provides a list of strings, where each string describes a path to the certain population in the gating hierarchy.

Packages *XML* and *xml2* offer an easy way of parsing XML into R list, however FlowJo Workspace has all its data stored in attributes, for which the packages are not suitable. Therefore, some adjustments are done by few helper functions.

For an easier manipulation with the date later on, the structure is not kept strictly as it is in the XML file. The main changes in populations compared to the original structure (among others) are:

- Each element in the returned list describes one population in Workspace file. In the original file, subpopulations in the gating hierarchy are child elements of other population. In the list, the children are moved as separate populations with an information about their ancestors.
- As said earlier, attributes containing all the data are moved to a separate list element called *attrs* for an easier access.
- Elements names are parsed without their name spaces.
- Each population stored in the list is named after its attribute *name.*

5.3 Transformation

Both raw measured data and gating parameters need to be transformed before they are used. The measured data is passed to *transformDims()* and parameters to *transformParams()*. The *transform()* function then uses the parsed information about transformations from the Workspace, so that the data is transformed according to the description in one of functions *linearTrans()*, *biexTrans()*, *logTrans()*, *fasinhTrans()*, *hyperlogTrans()*, *logicleTrans()* or *miltenyiTrans()*.

We fully implemented only *linear* and *biexponential* transformations, the rest is prepared for additional implementation. All the mentioned functions take as arguments the data, which need to be transformed, and numeric parameters from Workspace.

- *LinearTrans()*, taking arguments *minRange* and *maxRange*, moves all the data points, which value is under *minRange*, to the value of *minRange*, and data points above *maxRange* to the value of *maxRange*.
- *BiexTrans()* with arguments *length*, *maxRange*, *neg*, *pos*, *width* apply *flowjo_biexp()* from FlowWorkspace package.

5.4 Population classes

RectPopulation or *PolygonPopulation* represent one population (one *Population* XML element in Workspace) as one R6 object, both offering same methods, but their initial arguments and properties differ depending on their gating type (rectangle gate or polygon gate) – similar to concept of abstract classes and inheriting in other object-oriented languages.

There are more gate types supported by FlowJo, but in our solution we only included rectangular and polygon types, leaving an option to additionally implement the rest.

Methods and fields which are all common to both classes are:

- *xmlCount* – numeric value describing the count of events in the population taken from Workspace file,
- *populationName* – string value describing the name of population taken from Workspace file,
- *gateID* – string value describing the ID of gate taken from Workspace file,

- *gatedLogical* – vector of logical values, where each value represents one measured event from FCS file, 1 indicating, that the event belongs to the gate and 0 otherwise,
- *compareCounts* – dataframe containing the count of events insinde the gate taken from XML and the count reached in out solution ,
- *plotGate()* – plots the data with gate boundaries.

The arguments common to both classes are:

- *xmlCount* – numeric value describing the count of events in the population taken from Workspace file,
- *populationName* – string value describing the name of population taken from Workspace file,
- *gateID* – string value describing the ID of gate taken from Workspace file,
- *dims* – matrix with FCS data used for gating,
- *parent* – *gatedLogical* of the parent population.

RectPopulation class has additional arguments:

- *min* – numeric array of min values described in attribute gaitng:min,
- *max* – numeric array of max values described in attribute gaitng:max.

PolygonPopulation then:

- *vertices.x* - numeric array of vertices x coordinates,
- *vertices.y* - numeric array of vertices y coordinates.

The private method *processGate(),* used during initialization of an instance, applies the actual gating and selects the data points, which belong to the gate. Both rectangular and polygon gates follow the rules of Gating-ML, where the points on verteces or under lines belong to the gate. The method also includes checking the parent population, if exists, and modifying the child population accordingly in function *checkGatingHierachy(),* which is a simple intersection of both populations.

6 User documentation

Creating a new instance of *SamplePopulations* class is the only step, which user needs to make, as shown in Code 6.1. After that, they can access and work with all the data and information extracted from FlowJo Workspace and FCS file.

```
> wsfile <- “exampleWorkspace.wsp"
> fcsfile <- "exampleFCS.fcs"

> pops <- SamplePopulations$new(fcsfile = fcsfile, wsfile =
wsfile)
```

Code 6.1: Creating a new instance of SamplePopulations class

With the *SamplePopulations* instance saved in a variable, it is very easy to get the comparisone of how many events belong to the gates (Code 6.2), to get a matrix indicating which data points belong to the gates (Code 6.3), to print the hierarchy of the papulations (Code 6.4) or to get a list of *RectPopulation* and *PolygonPopulation* instances, carrying the information about populations (Code 6.5).

```
> pops$compareAllCounts()
  Population_Name         Gate_ID Count_in_XML Processed_Count
1       population1  ID1748209691          414             414
2       population2  ID1439605380         6123            6123
3       population3  ID1421260072         4404            4404
4       population4   ID573931730        10446           10432
5       population5   ID116683766         9002            8992
6       population6   ID202101879         5664            5649
7       population7  ID2018262693         4731            4722
8       population8  ID1987121774          483             486
9       population9  ID1405072533        60772           60772
10     population10    ID29715193         1706            1706
11     population11   ID409322866         9684            9658
12     population12  ID1719223922         2726            2718
```

Code 6.2: compareAllCounts() displaying the resulting counts of events inside gates compared to the count described in Workspace file

```
> pops$getAllPopsLogical()
      population1 population2 population3 population4 …
[1,]           0           0           0           0
[2,]           0           0           0           0
[3,]           0           0           0           0
…
```

Code 6.3: getAllPopsLogical() containing the matrix showing which events belong to the gates

```
> pops$printHierarchy()
[[1]]
[1] "/population1"
[[2]]
[1] "/population2"
[[3]]
[1] "population2/population3"
[[4]]
[1] "/population4 "
[[5]]
[1] "/population4/population5"
[[6]]
[1] "/population6"
[[7]]
[1] "/population6/population7"
[[8]]
[1] "/population6/population7/population8"
…
```

Code 6.4: Displaying of populations hierarchy

Each population from the list of populations can be accessed with double brackets, as shown in Code 6.5.

```
> populations <- pops$getPopulations()
> pop <- populations[[2]]
> pop$plotGate()

> pop$gateID
[1] "ID468195769"

> pop$populationName
[1] "population2"

> pop$compareCounts
     Population_Name     Gate_ID Count_in_XML Processed_Count
   1      Population2 ID468195769         6123            6123

> pop$gatedLogical
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
…
```

Code 6.5: Managing single population

7 Results

We tested our solution on three FlowJo Workspace files, containing analysis on the same set of raw data, and compared the count of events inside the gates – first the count from FlowJo Workspace file, which represents the correct amount, second the result of our implementation and third the count provided by FlowWorkspace package.

7.1 Common populations

First testing Workspace file consists of 12 populations with examples of hierarchical gating, where population3 is a child of population2, population5 is a child of population4, population8 is a child of population7, which is a child of populatuon6, as showed in Code 6.4. Table 7.1 represents our results and Figure 7.1 shows graphic gates in FlowJo application.

Table 7.1: Resulting counts of events inside common gates with hierarchical structure

Population name	Gate features	XML count	Result count	FlowWorkspace count
Population1	Biex – rectangular	414	414	414
Population2	Biex – rectangular	6123	6123	6123
Population3	Biex – rectangular – child	4404	4404	4404
Population4	Biex – polygon	10446	10432	10436
Population5	Biex – rectangular – child	9002	8992	8996
Population6	Biex – polygon	5664	5649	5652
Population7	Biex – rectangular – child	4731	4722	4724
Population8	Biex – rectangular – child	483	486	486
Population9	Linear – rectangular	60772	60772	60772
Population10	Linear – rectangular	1706	1706	1706
Population11	Linear – polygon	9684	9658	9658
Population12	Linear – polygon	2 726	2718	2718

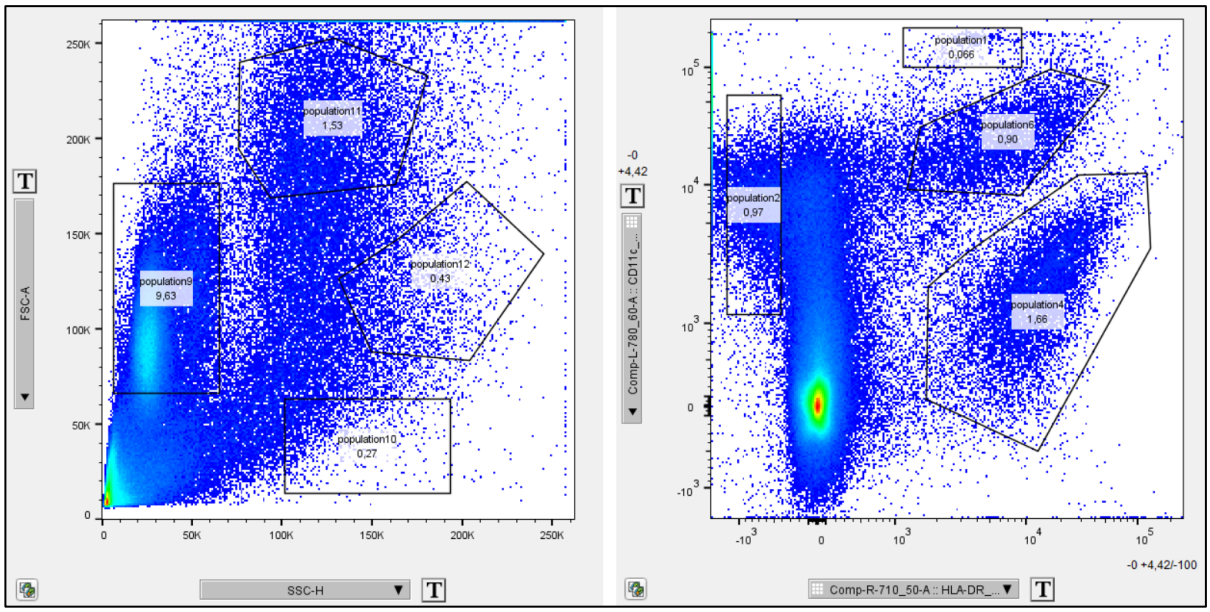

Figure 7.1: Common gates in FlowJo

7.2 Populations gated on the graph edge

The second file shows populations gated on or over the edges of the graph, which FlowJo allows, and carries 6 populations. Table 7.2 represents our results and Figure 7.2 shows graphic gates in FlowJo application.

Table 7.2: Resulting counts of events inside gates on the graph edge

Population name	Gate features	XML count	Result count	FlowWorkspace count
Population1	Biex – rectangular	235	235	235
Population2	Biex – rectangular	8800	5545	8800
Population3	Biex – rectangular	8708	8708	8708
Population4	Biex – polygon	1275	1458	1389
Population5	Linear – rectangular	371	371	371
Population6	Linear – rectangular	13073	13073	13073
Population7	Linear – rectangular	1117	1117	2253

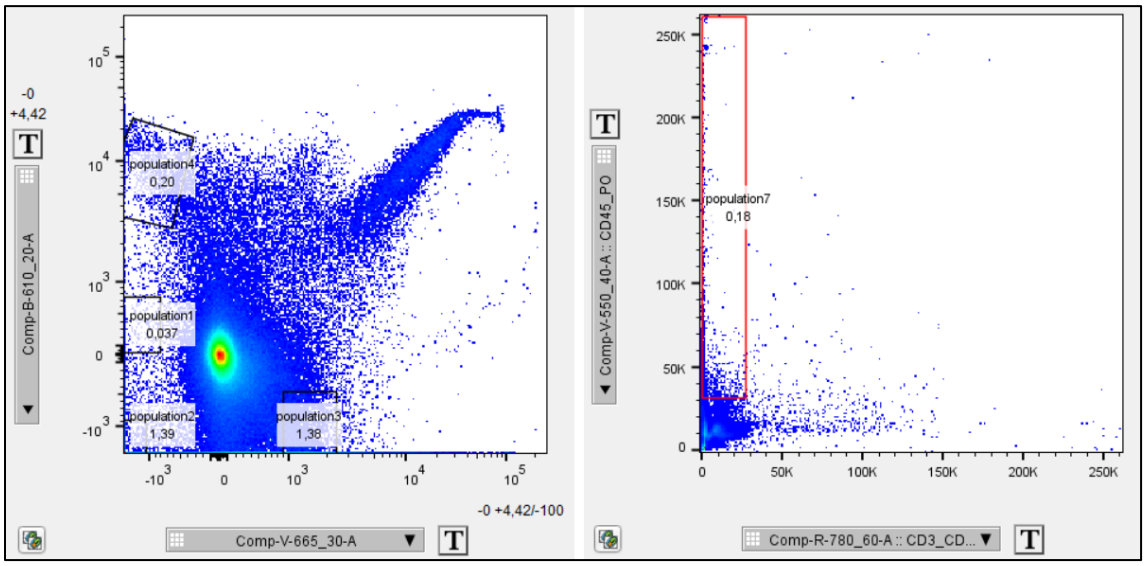

Figure 7.2: Gates on the grapth edge in FlowJo

7.3 Events under gate boundaries

The third Workspace file then represents smaller papulations with events well visible on the boundaries of the gates, to examine their behavior, or with no events on the boundaries at all. Table 7.3 represents our results and Figure 7.3 shows graphic gates in FlowJo.

Table 7.3: Resulting counts of events inside gates with events on the gate boundaries

Population name	Gate features	XML count	Result count	FlowWorkspace count
Population1	Biex – rectangular	6	6	6
Population2	Biex – rectangular	24	24	24
Population3	Biex – polygon	27	28	28
Population4	Biex – polygon	11	9	9
Population5	Biex – polygon	9	11	11
Population6	Linear – rectangular	10	10	10
Population7	Linear – rectangular	27	27	27
Population8	Linear – polygon	36	39	39
Population9	Linear – polygon	52	48	48
Population10	Biex – polygon – non	12	12	12
Population11	Biex – polygon – non	21	21	21
Population12	Biex – polygon – non	22	22	22

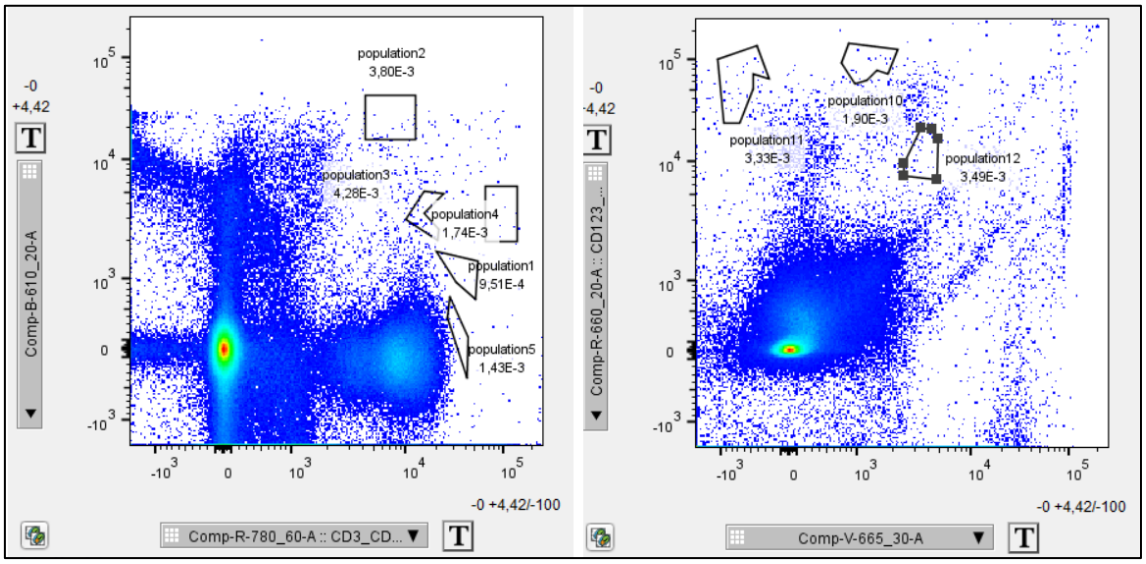

Figure 7.3: Gates with visible events on the gate boundaries on the left, Gates with no events on the gate boundaries on the right

8 Discussion

We mapped the currently available R packages solving this issue of reproducing the FlowJo analysis in R and found the FlowWorkspace package as the best option, since the great number of other R tools managing cytometric data utilize this package in their own approaches.

However, even FlowWorkspace does not match the original FlowJo analysis precisely, where the count of events inside the reproduced gates sometimes differs from the count in the original gates. The difference is mostly relatively small and in most cases insignificant, but for example in terms of leukemia diagnosis, where the count of cells we search for can be very low, the count difference may increase in importance. Less often but not rarely, the counts differ significantly, with the FlowWorkspace count being even twice the original FlowJo count.

These inaccuracies were the main motivation for this thesis, thus it is necessary to mention, that during our analysis of the current situation we contacted the developers of FlowWorkspace and they responded with an answer referring to an already existing solution solving the large deviations mentioned above.

In the implementation part of this thesis we designed R structures and functions, which are able to parse the FlowJo Workspace file and reconstruct the most common analytical steps made in FlowJo, including biexponential transformation and rectangular or polygon gating. FlowJo approach to biexponential transformation however differs from the description from other resources, without any official description, therefore we decided to use the FlowWorkpsace function *flowjo_biexp()* in our solution. Our implementation of gating followed the instructions from Gating-ML specification.

Criterium, evaluating the precision of our solution, was the resulting count of events inside the gates compared to the count of events taken from the Workspace file. When testing our result, we also included the counts counted by FlowWorkspace for comparison.

In the Table 7.1, Table 7.2 and Table 7.3 we can see, that reconstructing the rectangular gates, both by our solution or by FlowWorkspace, always resulted in the correct count of events inside the gates, no matter the scaling (linear or biexponential).

On the other hand, reproducing polygon gates very often lead to slightly inaccurate results. That was possibly caused by the events lying under the gate boundary, as we can see in the Table 7.3, where Population10, Population11 and Population12 are gates without any events under the boundary line and the rest of populations with at least one such event. Similarly, to rectangle gates, type of scaling did not affect the result in any significant way and considering the accuracy of reconstructing the rectangle gates and inaccuracy of reproducing the polygon gates in both linear and biexponential space, we assume, that these particular transformations are not the source of the differences between resulting counts. The problem presumingly lies in reconstructing the polygon gates which neither we nor FlowWorkspace managed to apply the same way as FlowJo does.

An exception were gates that crossed the graph edge. In these cases, our solution reached worse outcomes more often, if the gates had been applied on biexponentially scaled data, as we represent in Table 7.2. FlowWorkspace sometimes also resulted in significantly different counts, when the gates reached over the graph edge. These cases proved to be the reason of its larger resulting deviations mentioned earlier, when we contacted the developers of FlowWorkspace during our analysis of the current situation and they responded with an answer refering to an already existing solution to this issue. Following their instructions mostly lead to more accurate results.

9 Conclusion

In this thesis we focused on reproducing FlowJo (v 10.5.3) flow cytometry analysis in the environment of R language based on the FlowJo Workspace file and FCS file. The aim was to survey currently available solutions and provide our own solution, to better understand the issue.

To reach that, we had to determine the importance of elements in the XML Workspace file, parse the file and store the relevant parts in R structures. After pre-processing the raw data we applied “gating”, as described in Gating-ML specification.

Our solution did not exceed already available tools, but served to indentify the main sources of the resulting deviations, when using these packages. This knowledge can be applied during the expert manual analysis and help to avoid or reduce the risk of significantly incorrect results.

References

[1] Ashland, OR: Becton, Dickinson and Company, FlowJo™ Software Verze 10.5.3 [software]. 2019 [cit. 2012-02-11]. Dostupné z: https://www.flowjo.com

[2] MCKINNON, Katherine M. Flow Cytometry: An Overview. *Current Protocols in Immunology*. 2018, **120**(1). ISSN 1934-3671. Dostupné z: doi:10.1002/cpim.40

[3] MILLER, Benjamin a Marie O'TOOLE. *Miller-Keane Encyclopedia and Dictionary of Medicine, Nursing, and Allied Health*. 7th ed. 2003. ISBN 9781455726240.

[4] COSSARIZZA, Andrea, Hyun-Dong CHANG, Andreas RADBRUCH et al. *European Journal of Immunology*. 2017, **47**(10). ISSN 0014-2980. Dostupné z: doi:10.1002/eji.201646632

[5] DAREVSKY, Ilya S., Robert W. MURPHY, Ross D. MACCULLOCH, Cheryl SMITH, Nikolai ORLOV, Leslie A. LOWCOCK a Darlene E. UPTON. Flow cytometry in biodiversity surveys: methods, utility, and constraints. *Amphibia-Reptilia*. 1997, **18**(1), 1-13. ISSN 0173-5373. Dostupné z: doi:10.1163/156853897X00260

[6] MINARD, Austin, Andrea SAJEWSKI, Justin COOK et al. Identification of MRSA infection in blood using photoacoustic flow cytometry. *Photons Plus Ultrasound: Imaging and Sensing 2019*. SPIE, 2019, , 220-. ISBN 9781510623989. Dostupné z: doi:10.1117/12.2510210

[7] Data file standard for flow cytometry. *Cytometry*. 1990, **11**(3), 323-332. ISSN 0196-4763. Dostupné z: doi:10.1002/cyto.990110303

[8] SPIDLEN, Josef, Wayne MOORE, David PARKS et al. Data File Standard for Flow Cytometry, version FCS 3.1. *Cytometry Part A*. 2009, **9999**. ISSN 15524922. Dostupné z: doi:10.1002/cyto.a.20825

[9] SPIDLEN, Josef, Robert C. LEIF, Wayne MOORE, Mario ROEDERER a Ryan R. BRINKMAN. Gating-ML: XML-based gating descriptions in flow cytometry. *Cytometry Part A*. 2008, **73**(12), 1151-1157. ISSN 15524922. Dostupné z: doi:10.1002/cyto.a.20637

[10] BOHN, Anja B, Bjarne K MOLLER a Mikkel S PETERSEN. Flow cytometry and compensation of highly autofluorescent cells: the example of

mesenchymal stem cells. *Stem Cell Biology and Research*. 2015, **2**(1). ISSN 2054-717X. Dostupné z: doi:10.7243/2054-717X-2-4

[11] HOLGATE, P., E. L. CROW a K. SHIMIZU. Lognormal Distributions: Theory and Applications. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*. 1989, **152**(2). ISSN 09641998. Dostupné z: doi:10.2307/2982924

[12] NOVO, David a James WOOD. Flow cytometry histograms: Transformations, resolution, and display. *Cytometry Part A*. 2008, **73**(8), 685-692. ISSN 15524922. Dostupné z: doi:10.1002/cyto.a.20592

[13] BAGWELL, C. Bruce. Hyperlog?A flexible log-like transform for negative, zero, and positive valued data. *Cytometry Part A*. 2005, **64**(1), 34-42. ISSN 1552-4922. Dostupné z: doi:10.1002/cyto.a.20114

[14] PARKS, David R., Mario ROEDERER a Wayne A. MOORE. A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry Part A*. 2006, **69**(6), 541-551. ISSN 1552-4922. Dostupné z: doi:10.1002/cyto.a.20258

[15] FINAK, Greg, Juan-Manuel PEREZ, Andrew WENG a Raphael GOTTARDO. Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics*. 2010, **11**(1). ISSN 1471-2105. Dostupné z: doi:10.1186/1471-2105-11-546

[16] QIU, Peng. Computational prediction of manually gated rare cells in flow cytometry data. *Cytometry Part A*. 2015, **87**(7), 594-602. ISSN 15524922. Dostupné z: doi:10.1002/cyto.a.22654

[17] LUGLI, Enrico, Mario ROEDERER a Andrea COSSARIZZA. Data analysis in flow cytometry: The future just started. *Cytometry Part A*. 2010, **77**(7), 705-713. ISSN 15524922. Dostupné z: doi:10.1002/cyto.a.20901

[18] SPIDLEN, Josef a Ryan BRINKMAN. *Gating-ML 2.0: International Society for Advancement of Cytometry (ISAC)standard for representing gating descriptions in flow cytometry*. 2015. Dostupné také z: http://flowcyt.sourceforge.net/gating/latest.pdf

[19] *FlowJo v10 Documentation* [online]. Ashland (Oregon): FlowJo, LLC, 2021 [cit. 2021-05-10]. Dostupné z: https://docs.flowjo.com/flowjo/

[20] ELLIS, B, Perry HAALAND, Florian HAHNE et al. *FlowCore: Basic structures for flow cytometry data* [online]. [cit. 2021-05-10]. Dostupné z: https://rdrr.io/bioc/flowCore/

[21] About the Arcsinh transform. *CytoBank* [online]. Cytobank, Inc., 2020 [cit. 2021-05-10]. Dostupné z: https://support.cytobank.org/hc/en-us/articles/206148057-About-the-Arcsinh-transform

[22] FINAK, Greg, Wenxin JIANG a Raphael GOTTARDO. CytoML for cross-platform cytometry data sharing. *Cytometry Part A*. 2018, **93**(12), 1189-1196. ISSN 1552-4922. Dostupné z: doi:10.1002/cyto.a.23663

[23] HORNIK, Kurt. R FAQ: Frequently Asked Questions on R. *The Comprehensive R Archive Network* [online]. Welthandelsplatz (Vienna): Institute for Statistics and Mathematics [cit. 2021-05-12]. Dostupné z: https://cran.r-project.org/

Attachment A: Content of the enclosed CD

- Scanned bachelor’s thesis assignment
- Complete bachalor thesis
- Directory R
    - exampleUsageScript.R
    - SamplePopulationClass.R
    - GatingClasses.R
    - listToPopulations.R
    - parseXML.R
    - transformations.R
    - dependencies.R
    - exampleWorkspace.wsp