CYTOMETRY
Journal of Quantitative Cell Science PART A

Robust FCS Parsing: Exploring 211,359 Public Files

Anne E. Bras, Vincent H. J. van der Velden*

Laboratory Medical immunology, Department of Immunology, Erasmus MC, University Medical Center Rotterdam, Rotterdam, the Netherlands

Received 24 January 2020; Revised 24 June 2020; Accepted 29 June 2020

Additional Supporting Information may be found in the online version of this article.

*Correspondence to: *Dr. Vincent H. J. van der Velden, Laboratory Medical immunology, Department of Immunology, Erasmus MC, Dr. Molewaterplein 40, 3015 GD, Rotterdam, the Netherlands
Email: v.h.j.vandervelden@erasmusmc.nl

Published online 15 July 2020 in Wiley Online Library (wileyonlinelibrary.com)

DOI: 10.1002/cyto.a.24187

Abstract

When it comes to data storage, the field of flow cytometry is fairly standardized, thanks to the flow cytometry standard (FCS) file format. The structure of FCS files is described in the FCS specification. Software that strictly complies with the FCS specification is guaranteed to be interoperable (in terms of exchange via FCS files). Nowadays, software interoperability is crucial for eco system, as FCS files are frequently shared, and workflows rely on more than one piece of software (e.g., acquisition and analysis software). Ideally, software developers strictly follow the FCS specification. Unfortunately, this is not always the case, which resulted in various nonconformant FCS files being generated over time. Therefore, robust FCS parsers must be developed, which can handle a wide variety of nonconformant FCS files, from different resources. Development of robust FCS parsers would greatly benefit from a fully fledged set of testing files. In this study, readability of 211,359 public FCS files was evaluated. Each FCS file was checked for conformance with the FCS specification. For each data set, within each FCS file, validated parse results were obtained for the TEXT segment. Highly space efficient testing files were generated. FlowCore was benchmarked in depth, by using the validated parse results, the generated testing files, and the original FCS files. Robustness of FlowCore (as measured by testing against 211,359 files) was improved by re-implementing the TEXT segment parser. Altogether, this study provides a comprehensive resource for FCS parser development, an in-depth benchmark of FlowCore, and a concrete proposal for improving FlowCore. © 2020 The Authors. *Cytometry Part A* published by Wiley Periodicals LLC on behalf of International Society for Advancement of Cytometry.

Key terms

flow cytometry; FCS standard; file format; file parser; bioinformatics

When it comes to data storage, the field of flow cytometry is fairly well standardized. The flow cytometry standard (FCS) file format was introduced in 1984 and is currently the de facto standard. The structure of FCS files (summarized in the Supporting Information Sup. 1) is described in detail in the FCS specification (1–3), a document published by the International Society for Advancement for Cytometry (ISAC) Data Standards Task Force (DSTF).

Software that strictly complies with the FCS specification is guaranteed to be interoperable (in terms of data exchange via FCS files). Nowadays, interoperability is crucial for the flow cytometry eco system, as FCS files are frequently shared (e.g., public repositories and multicenter studies) and modern workflows rely on more than one piece of software (e.g., different acquisition and analysis software).

Given the complexity of software development, unintentional noncompliances are pretty much inevitable and found their way into well-established flow cytometry software. Unfortunately, these noncompliances resulted into numerous noncompliant FCS being generated over time (4) and added seriously to the complexity of building robust FCS parsers (algorithms being able to interpret any FCS file from any resource). Notably, even seemingly small noncompliances might introduce significant complexity (example in the Supporting Information Sup. 2).

Essentially, there are two approaches to this problem: preventing nonconformant FCS writers from reaching production environments (e.g., by using FCS conformance testing tools, like FlowIO, during development), and improving robustness of FCS parsers (e.g., by testing against nonconformant FCS files during development).

Most FCS parsers gained some degree of robustness over time, due to being exposed to countless FCS files, and being optimized for numerous edge cases that occurred over time. Presumably, most software developers build their own collection of nonconformant FCS files along the way. However, to our best knowledge, only a small collection of testing files is publicly available.

Development of robust FCS parsers would greatly benefit from a fully fledged set of testing files, consisting of representative FCS files, along with fully validated parse results. In this study, the readability of 211,359 public FCS files was evaluated. Each FCS file was checked for conformance with the FCS specification. For each data set, within each FCS file, fully validated parse results were obtained for the TEXT segment, by aggregating results from two well-established FCS parsers (FlowCore and FlowJo). Highly space efficient testing files were generated, which truly reflect the heterogeneity in FCS files, as seen nowadays. Finally, based on these testing files, FlowCore was benchmarked in depth, and its robustness (as measured by testing against 211,359 files) was improved.

Materials and Methods

The workflow of this study is summarized in Figure 1A and all details are provided in the Supporting Information Sup. 3–24.

Software

Reference results were obtained by aggregating results from FlowJo (general-purpose analysis software, actively developed since 1996) and FlowCore (foundation of the Bioconductor eco system, actively developed since 2002). Presumably, these parsers gained robustness over time, being exposed to many files from many resources. Detailed benchmarks were performed for FlowCore and FlowIO (both open-source, allowing insight in their inner workings). General data handling was performed in R base (version 3.4.1)

Data Collection and Conformity Checks

Public data sets (up to July 2017) were obtained from Cytobank, FlowRepository, and ImmPort (5-7) (Supporting Information Sup. 3). Duplicate files were removed based on MD5 and SHA1 hashes. Remaining files (n = 238,581) were assigned unique identifiers (based on repository name, data set name, and MD5 hash). First, FCS files (n = 211,359) were distinguished from non-FCS files (n = 27,222) (Supporting Information Sup. 4). Next, single data set FCS files (SD-FCS, n = 206,620) were distinguished from multi data set FCS files (MD-FCS, n = 4,739), and absolute offsets were obtained for each data set (n = 9,724) within each MD-FCS (Supporting Information Sup. 5). In total, 216,344 data sets were found among the 211,359 downloaded FCS files (RAW-FCS). Of note, these RAW-FCS were also used in another study (8). Finally, each RAW-FCS was evaluated by FlowIO (4) to check conformance with the FCS specification (Supporting Information Sup. 6).

HEADER Evaluation

HEADER formatting was evaluated, and byte offsets (as stored within each HEADER) were obtained via two methods (Supporting Information Sup. 7). Byte offsets were checked for validity (Supporting Information Sup. 8) and accuracy (Supporting Information Sup. 9), and corrected wherever appropriate, to make sure that each TEXT segment was accurately referenced (the delimiter enclosed portion).

DATALESS-FCS

Four different types of DATALESS-FCS were created (Supporting Information Sup. 10). SLICED-FCS mimic RAW-SD-FCS, as they are identical to RAW-SD-FCS, except for the bytes after the TEXT segment (e.g., the DATA segment) being removed (Fig. 1B). BLANKED-FCS mimic RAW-MD-FCS, as they are identical to RAW-MD-FCS, except for the bytes from the DATA and ANALYSIS segment being replaced by 0x20 bytes (Fig. 1C). INIT-FCS are newly created FCS files, which contain the initial TEXT segment from RAW-FCS, preceded by a HEADER with accurate byte offsets (Fig. 1B). ADD-FCS are similar to INIT-FCS but contain the additional TEXT segment(s) from RAW-MD-FCS (Fig. 1C).

TEXT Segment Parsing

All RAW-FCS and DATALESS-FCS were parsed by FlowCore (1.43.2) and FlowJo (10.0.7, 10.3.0, and 10.4.0), and parse results were stored in terms of key-value tables (Supporting Information Sup. 11). Whether SLICED-FCS and BLANKED-FCS mimicked their RAW-FCS counterparts was evaluated based on the FlowCore and FlowJo results (Supporting Information Sup. 12). FlowJo and FlowCore handled MD-FCS differently, however, by using INIT-FCS and ADD-FCS, instead of RAW-MD-FCS, both parser were forced to return uniformly formatted results, namely one key-value table for each data set (Supporting Information Sup. 13).

TEXT Segment Reference

The parse results were systematically evaluated, and edge cases were documented along the way (Supporting Information Sup. 14). First, the FlowCore and FlowJo results were compared among themselves (version differences), and for each FCS file, the optimal FlowCore and FlowJo result was selected (Supporting Information Sup. 15). Next, the sanitization as applied by FlowJo, was mimicked in the FlowCore results (Supporting Information Sup. 16), allowing a direct comparison between FlowJo and FlowCore (Supporting Information Sup. 17). In case FlowJo and FlowCore returned discrepant results, they were manually reviewed and corrected wherever appropriate. Next, one unambiguous reference file (REF-FCS) was created for each data set (Supporting Information Sup. 18). Finally,

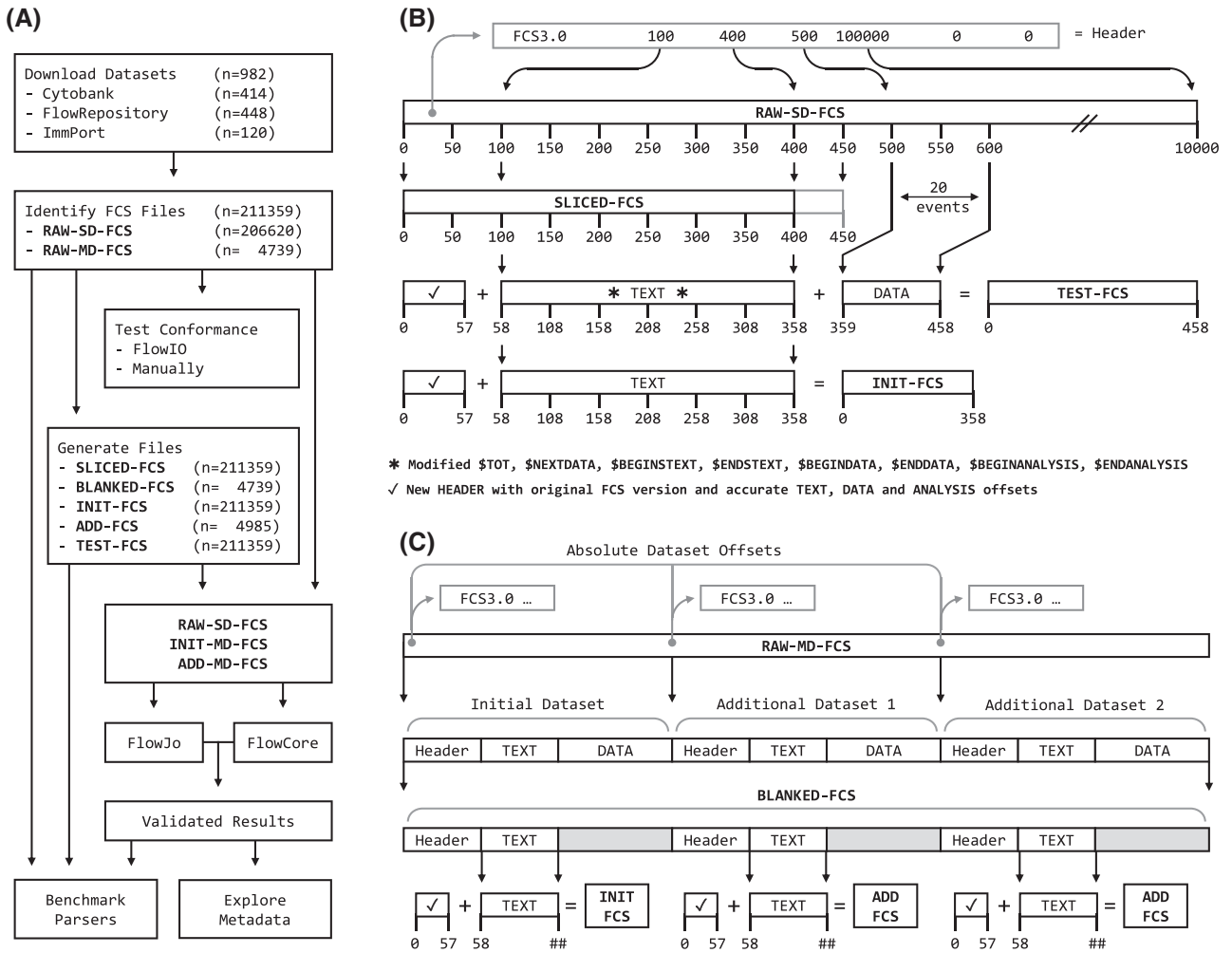

Figure 1. A. In total, 211,359 publicly available FCS files were downloaded (RAW-FCS), consisting of 206,620 single data set files (RAW-SD-FCS) and 4,739 multi data set files (RAW-MD-FCS). Each RAW-FCS was checked for conformance with the FCS specification. For each RAW-FCS, multiple artificial FCS files were created (details in **B** and **C**), allowing in depth evaluation of various aspects of RAW-FCS, in a highly space efficient manner. For each data set, within each RAW-FCS, validated parse results were obtained for the TEXT segment, by aggregating FlowCore and FlowJo parse results. The metadata of 211,359 public FCS files can now be easily explored. Together, the validated parse results and artificial FCS files form a fully fledged testing set for FCS parser development. Performance of FlowCore was benchmarked in-depth. **B**. SLICED-FCS mimic RAW-SD-FCS, as they are identical to RAW-SD-FCS, except for the bytes after the TEXT segment being removed. INIT-FCS are newly created FCS files, which contain the initial TEXT segment from RAW-FCS, preceded by a HEADER with accurate offsets. TEST-FCS contain the first 20 events of the original DATA segment, and a slightly modified version of the original TEXT segment (reflecting the new DATA segment size). **C.** BLANKED-FCS mimic RAW-MD-FCS, as they are identical to RAW-MD-FCS, except for the bytes from the DATA and ANALYSIS segment being replaced by 0x20 bytes (space character in ASCII). ADD-FCS are similar to INIT-FCS, but contain the additional TEXT segment(s) from RAW-MD-FCS.

each REF-FCS was validated against its RAW-FCS counterpart (Supporting Information Sup. 19).

TEST-FCS Creation

DATA offsets, as stored in the HEADER and TEXT segment, were checked for validity (Supporting Information Sup. 20). For each primary data set, within each RAW-FCS, one TEST-FCS was created (Supporting Information Sup. 21). Essentially, TEST-FCS contain the original TEXT segment (except for eight key-value pairs being modified), part of the original DATA segment (the first 20 events), and a new HEADER with accurate offsets (Fig. 1B). Validity of the modifications was checked by FlowIO.

FlowCore Improvements and Benchmarks

Two branches of FlowCore were created (Supporting Information sup. 22); one where the existing TEXT segment parser was improved (to automatically handle escapes and empty values), and one where the TEXT segment parser was re-implemented (to improve robustness). FlowCore developers can easily merge either branch. Performance of both branches was evaluated in depth (Supporting Information sup. 23). For

each exception (error or warning), an exemplary case was described in detail.

DATA Segment

FlowCore (with the re-implemented TEXT segment parser) was used to obtain raw DATA segment values, which were subsequently checked for validity (Supporting Information sup. 24). The cyclic redundancy checks (CRC = "digital fingerprint") were checked for validity, and keywords relevant to the DATA segment were explored (Supporting Information sup. 24).

Data Availability and Reproducibility

The appendixes (Supporting Information Sup. 25) include an overview of the enrolled RAW-FCS, an overview of the FlowIO results, the DATALESS-FCS (SLICED-FCS, BLANKED-FCS, INIT-FCS and ADD-FCS), the validated parse results (REF-FCS), the testing files (TEST-FCS), and the newly created FlowCore branches. RAW-FCS can be easily obtained from their original resources, based on their identifiers (Supporting Information Sup. 3).

Results

Conformance Testing

First, for each RAW-FCS, conformance with the FCS specification was checked by FlowIO (Supporting Information Sup. 6). FlowIO exited unexpectedly in 925 files (e.g., files without the $DATATYPE keyword) and exited gracefully in 15,483 files where integrity of the TEXT segment could not be confirmed (a conservative approach to FCS parsing). For the other files (n = 194,951), FlowIO returned results for nine conformance tests. Interestingly, only 1,481 files passed all tests (0.7%), and only 3,765 files failed at most one test (1.8%). Next, the FCS files were grouped by their test results. The largest group (117,228 files, 55%) failed the numeric formatting test and keyword compliance test. The second largest group (53,595 files, 25%) failed the same two tests, and three additional tests. Thus, strictly speaking, conformance with the FCS specification was poor; however, it should be stressed that most noncompliances do not result in FCS files being unreadable.

DATALESS-FCS

For each RAW-FCS (n = 211,359, circa 1.47 terabyte), multiple DATALESS-FCS (n = 432,442, circa 42 megabytes) were created (Fig. 1B,C). FlowCore and FlowJo returned identical key-value tables for DATALESS-FCS and their RAW-FCS counterparts (Supporting Information Sup. 12). Thus, DATALESS-FCS perfectly mimic RAW-FCS when it comes to HEADER and TEXT segment parsing and do this in a space efficient manner. Nevertheless, it should be emphasized that these DATALESS-FCS are corrupt FCS files (e.g., the DATA offsets refer to nonexistent locations). Therefore, TEST-FCS were created in a later stage, which closely resemble RAW-FCS, without the disadvantage of being corrupt.

HEADER Variability

The HEADER contains offsets for the TEXT, DATA and ANALYSIS segment. First, offset formatting was evaluated (Fig. 2A and Supporting Information Sup. 8). Offsets were always right aligned, either with leading spaces or zeros. Absence of ANALYSIS segments was either indicated by empty offsets, negative offsets or both offsets being set to zero. Nevertheless, FlowCore parsed each HEADER successfully.

Besides being well formatted, byte offsets must also be valid (e.g., reference readable bytes) and accurate (e.g., precisely reference a specific segment). Various invalid offsets were found (Fig. 2B and Supporting Information Sup. 9), including invalid ranges (start > end), ranges out-of-range (end > size) or overlapping ranges (segments having bytes in common). Inaccurate TEXT offsets always referenced the entire TEXT segment, plus some trailing bytes (Fig. 2C), which were either padding bytes (e.g., spaces) and/or bytes from the adjacent DATA segment (Fig. 2D).

TEXT Segment Variability

In essence, the TEXT segment is generated by alternately concatenating keys (e.g., "$k1" and "$k2") and values (e.g., "v1" and "v2") and by using a delimiter (e.g., "/") for delimitation. The TEXT segment (e.g., "/$k1/v1/$k2/v2/") is essentially parsed by splitting at each delimiter occurrence (e.g., "$k1," "v1," "$k2," and "v2") and associating the keys and values (e.g., "$k1" = "v1" and "$k2" = "v2"). In case the delimiter of choice occurs within any value, it must be escaped prior to TEXT segment generation (e.g., "CD4/CD8" to "CD4//CD8"), guaranteeing that single delimiters (e.g., "/") represent delimiters, and consecutive delimiters (e.g., "//") represent escapes. Importantly, because concatenation of empty values also result in consecutive delimiters (e.g., "$k4" = "" results in "/$k4//"), empty values are strictly prohibited by the FCS specification, thereby preventing ambiguity (details in Supporting Information Sup. 2).

Different delimiters were used, including printable (Fig. 2E) and nonprintable characters (e.g., tabs and returns). Despite strict definitions in the FCS specification, various noncompliances were identified, including TEXT segments with double closing delimiters (Fig. 2F) and TEXT segments without any closing delimiters (Fig. 2G). Consecutive delimiters occurred frequently, and either represented empty values (Fig. 2H) or escaped delimiters (Fig. 2I). Remarkably, the vast majority of consecutive delimiters represented empty values, despite being strictly prohibited by the FCS specification.

TEXT Segment Parsing

All TEXT segments (n = 216,344), as stored across all FCS files (n = 211,359), were parsed by three FlowJo versions (10.0.07, 10.3, and 10.4), and parsed twice by FlowCore (version 1.42.2, either with the "assume empty values" setting enabled or disabled). Multiple parsers were used, because neither parsed all TEXT segments successfully. FlowJo and FlowCore exited occasionally, however, at least one parse result was obtained for each TEXT segment.

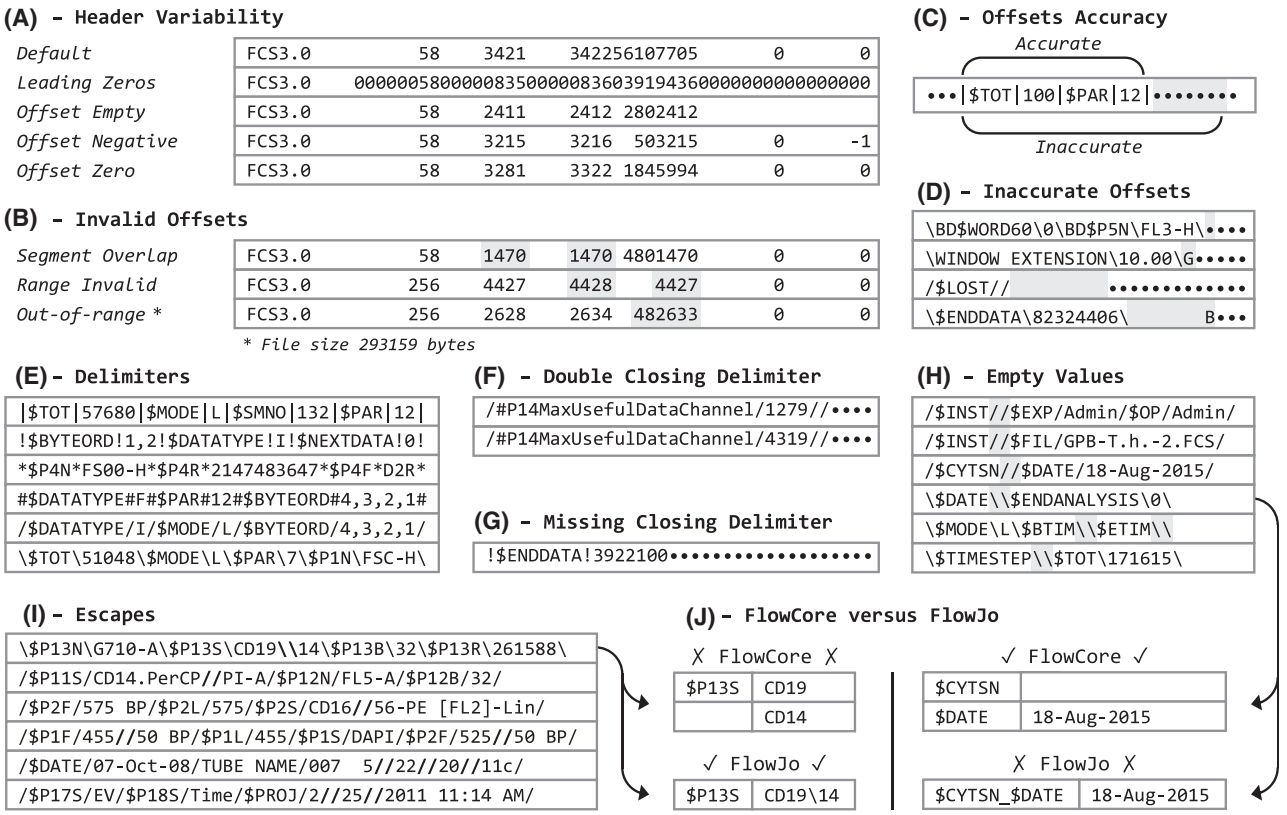

Figure 2. HEADER and TEXT segments were checked against the FCS specification. **A**. Offsets were always right aligned either with leading spaces or leading zeros. Absence of ANALYSIS segments was indicated by empty offsets, negative offsets (incorrect) or both offsets being zero. **B**. Various invalid offset pairs were found, including invalid ranges (start > end), ranges out-of-range (end > size) and overlapping ranges (segments having bytes in common). **C**. Inaccurate TEXT offsets always resulted in additional bytes being included. **D**. These were mostly padding bytes and/or bytes from the adjacent segment. **E**. Various delimiters were used. **F**. Some TEXT segments featured double closing delimiters. **G**. Others featured no closing delimiter at all. **H**. Many empty values were identified, despite being strictly prohibited by the FCS specification. **I**. Escapes were less frequently found, as compared to empty values. **J**. FlowCore and FlowJo alternately misinterpreted the consecutive delimiters (empty values vs. escapes).

First, the discrepancies within FlowJo and within FlowCore were evaluated (Supporting Information Sup. 15). For each TEXT segment, each FlowJo version returned identical results, except for 897 cases (0.4%), where FlowJo 10.0.7 misinterpreted the TIMESTEP keyword. The "assume empty values" setting (FlowCore) affected results in 1,617 cases (0.7%); for these, the optimal parse result was manually selected.

Next, the discrepancies between FlowJo and FlowCore were manually evaluated (Supporting Information Sup. 17). Parse results were discrepant in 1,530 cases (0.7%). These discrepancies were either caused by misinterpretation of consecutive delimiters (escapes vs. empty values) or by differences in data wrangling (e.g., FlowJo added sequential numbers to duplicate channels). Notably, FlowCore and FlowJo alternately misinterpreted consecutive delimiters (Fig. 2J), thus neither of them handled consecutive delimiters flawlessly. Except for three TEXT segments with double closing delimiters (Fig. 2E), either FlowJo or FlowCore or FlowJo returned correct results.

Concordant parse results (FlowJo vs. FlowCore) were assumed to be correct (and proven to be correct at a later stage, see below). Discrepant parse results were manually evaluated, and involved decisions (in terms of selecting the correct parse results) were obvious (various representative examples in the Supporting Information Sup. 17).

Reference Results

By aggregating the FlowJo and FlowCore parse results, and manually fixing three edge cases, solid parse results were obtained for each data set. Interestingly, the vertical bar ("|") never occurred within any key or value, and the tilde ("~") never occurred as value. Therefore, by using vertical bars as delimiters, and tildes as placeholders for empty values, unambiguous reference FCS files (REF-FCS) could be created. In other words, for each data set (n = 216,344), as stored across each RAW-FCS (n = 211,359), one REF-FCS was created, which is easy to parse (no ambiguity), and contains fully validated parse results (Supporting Information Sup. 18).

Validity of each REF-FCS was confirmed by escaping the original delimiter (e.g., replacing "/" by "//" in case "/" was used in the RAW-FCS counterpart), removing the

placeholders for empty values ("∼"), inserting the original delimiter (e.g., replacing "|" by "/"), and making sure the final result matched (byte by byte) with the RAW-FCS counterpart (Supporting Information Sup. 19). This way, validity of the parse results themselves, was confirmed as well.

Test-FCS

Essentially, TEST-FCS contain the first 20 events of the original DATA segment, and a slightly modified version of the original TEXT segment. In order to successfully extract the first 20 events from RAW-FCS, the DATA offsets as stored in the HEADER and TEXT segment of RAW-FCS, were validated (Supporting Information Sup. 20). Some irregularities were identified, including DATA offsets being inaccurate (e.g., of by one byte), and discrepancies between the DATA offsets as stored in the HEADER and the TEXT segment. In the end, valid DATA offsets were identified for 211,350 RAW-FCS, allowing 211,350 TEST-FCS to be generated, which passed the relevant checks by FlowIO as well (Supporting Information Sup. 21).

FlowCore Improvements and Benchmarks

Two branches of FlowCore were created (Supporting Information Sup. 22). The first branch improved user-friendliness, by automatically optimizing the "assume empty values" setting. Thus, this branch reflects FlowCore performance, as experienced by users who always optimize the "assume empty values" setting by hand. The second branch features a new TEXT segment parser, which was implemented based on the lessons learned in this study. Both branches were benchmarked, in terms of TEXT and DATA segment parsing, based on RAW-FCS and TEST-FCS (Supporting Information Sup. 23).

Within RAW-FCS, the first branch returned invalid TEXT results for 29 cases, and no TEXT results for 941 cases (due to invalid offsets). Within TEST-FCS, the first branch returned invalid TEXT results for the same 29 cases and succeeded in the aforementioned 941 cases (because offsets were corrected). The second branch always returned valid TEXT results and provided informative warnings (e.g., empty values and/or trailing bytes). The TEXT segment parser by FlowIO was also evaluated; and returned invalid TEXT results for 1,522 RAW/TEST-FCS with empty values.

Within RAW-FCS, the first branch failed to return DATA results for 965 cases, mostly due to discrepant DATA offsets in the HEADER and TEXT segment. The second branch, succeeded in approximately two-thirds of these cases (where either the HEADER or TEXT segment contained valid DATA offsets) and failed in the remainder as well (where neither contained valid DATA offsets, with respect to file size). Within TEST-FCS, both branches always succeeded, and always threw valid warnings (e.g., absence of required keywords, or duplicate channel names). Altogether, the FlowCore DATA segment parser performed well (as long the HEADER and TEXT segment parser provided accurate information).

DATA Segment

For each RAW-FCS which could be parsed by FlowCore, the raw DATA values were checked for validity, by checking against the $PnR keywords (which specify the maximum range for each parameter) (Supporting Information Sup. 24). The majority of RAW-FCS contained values in line with the $PnR keywords. The remainder mostly consisted of RAW-FCS from specific data sets (e.g., data set I-0091, where the $PnR for fluorescent parameters were incorrect), and RAW-FCS where the $PnR for the time parameter was incorrect. Rarely, a subset of fluorescent channels exceeded the $PnR range. The majority of RAW-FCS featured identical $PnR for each channel, and the remainder mostly consisted of files with different $PnR for each channel type (e.g., different $PnR for scatter and florescent data). Notably, integrity of DATA segments could not be confirmed, as none of the RAW-FCS featured a CRC checksum.

Discussion

Public repositories facilitate reproducible research (re-analyze data for original purpose) and secondary analysis (analyze data for other purpose). Strictly speaking, reproducibility does not depend on software interoperability. However, in practical terms, reproducibility heavily depends on software interoperability, due to scarcity of legacy software platforms (e.g., fairly hard to obtain a PowerMac with CellQuest). Software interoperability is also crucial for secondary analysis (e.g., being able to use state-of-the-art software) and the repositories themselves (e.g., being able to extract metadata for indexing). Altogether, robust FCS parsers are key to public repositories.

Nonconformant FCS files are certainly not a thing of the past. First, cytometrists keep using legacy software, either because upgrades are expensive, or software cannot be upgraded anymore (e.g., acquisition software). Second, nonconformant FCS files were not only generated by legacy software, but also by recent versions of actively developed software (e.g., CyTOF Software). Third, most cytometrists are not aware of the problem. Thus, the need for robust FCS parsers will persist into the foreseeable future.

This study provides a comprehensive resource for parser development, including a fully fledged set of testing files: DATALESS-FCS are identical to RAW-FCS when it comes to HEADER and TEXT segment parsing, but corrupt when it comes to DATA segment parsing. TEST-FCS closely resemble RAW-FCS when it comes to TEXT segment and DATA segment parsing, but fail to represent offset-related non-compliances. REF-FCS are unambiguous FCS files, to be used as reference for TEXT segment parsing, but also corrupt when it comes to DATA segment parsing. Thus, DATALESS-FCS, TEST-FCS, and REF-FCS serve different purposes, and together, reflect the heterogeneity as found in 211,359 public FCS files, within a footprint of 223 megabytes. Also, for each file as discussed in the supplement, the RAW-FCS is provided, thereby providing a small series (n = 248) of well-described representative files (Supporting Information Sup. 25). Of note, these FCS files may serve other purposes as well, for

example, serve as test cases for FCS indexing systems, or serve as a solid starting point for studies on naming conventions (Supporting Information Sup. 26).

Obviously, these testing files cannot be taken as principle safeguard for FCS parser development. Theoretically, creativity in FCS file generation is limitless. However, in practice, the variety in nonconformities is fairly limited. Most nonconformities are understandable from technical point of view (e.g., offsets being off by one byte) and occurred in independently developed software (e.g., acquisition software from Fluidigm and Beckman Coulter). Therefore, the same nonconformities might occur in future software as well, if only because they go unnoticed (existing FCS parsers handle them silently). Other nonconformities, that truly break existing FCS parsers, are more likely to be noticed in time and therefore less likely to reach production environments.

FlowCore succeeded in the vast majority of RAW-FCS, as long the "assume empty values" setting was optimized. In this study, two modifications to FlowCore were proposed, namely improving the existing TEXT segment parser (by adding the ability to automatically optimize the "assume empty values" setting), and replacing the existing TEXT segment parser (thereby improving robustness, and descriptiveness of exceptions). FlowCore, together with the re-implemented TEXT segment parser, successfully parsed each RAW-FCS.

In principle, the nature of consecutive delimiters (escapes vs. empty values) cannot be resolved without context (e.g., "/A//B//C/D/" is ambiguous, while "/$BTIM//$ETIM//$CYT/CYTOF/" is obvious), therefore parsers must build upon assumptions. FlowIO assumes that consecutive delimiters represent escapes (in line with the specification) while FlowCore (by default) assumes that they represent empty values (in line with reality, being more frequent). By definition, this causes FlowIO to misinterpret empty values, and FlowCore to misinterpret escapes; regularly without warning (Supporting Information Sup. 22). In contrast, our code resolves the nature of consecutive delimiters, by assuming that escapes never occur in keys, which yielded better performance, at least in 211,359 public FCS files. Thus, resolving the nature of consecutive delimiters (by default), and allowing users to override (in case of failure), seems optimal.

FlowIO and FlowJo handle anomalies differently. FlowIO errs on the side of caution (always informing users), while FlowJo errs on the side of usability (rarely informing users). FlowCore, together with our TEXT segment parser, provides a middle ground, by continuing in case of minor anomalies (e.g., trailing spaces), and halting in case of major anomalies (e.g., odd number of tokens).

Thus, strictly speaking, the majority of FCS files are non-compliant, according to FlowIO. Fortunately, cytometrists are rarely affected by these noncompliances, because existing FCS parsers (e.g., FlowCore and FlowJo) handle them well. Altogether, this study provides a comprehensive resource for FCS parser development, an in-depth benchmark of FlowCore, and a concrete proposal for improving FlowCore.

Acknowledgments

The authors gratefully acknowledge Cytobank, FlowRepository, and ImmPort for maintaining the public databases. The authors gratefully acknowledge everyone who contributed to these database. The research for this manuscript was performed within the framework of the Erasmus Postgraduate School Molecular Medicine.

Author Contributions

Anne Bras: Conceptualization; data curation; formal analysis; investigation; methodology; project administration; validation; writing-original draft; writing-review and editing. **Vincent van der Velden:** Funding acquisition; resources; validation; writing-original draft; writing-review and editing.

Conflicts of Interest

None were disclosed by the authors.

Literature Cited

1. Murphy RF, Chused TM. A proposal for a flow cytometric data file standard. Cytometry 1984;5(5):553–555.
2. Seamer LC, Bagwell CB, Barden L, Redelman D, Salzman GC, Wood JC, Murphy RF. Proposed new data file standard for flow cytometry, version FCS 3.0. Cytometry 1997;28(2):118–122.
3. Spidlen J, Moore W, Parks D, Goldberg M, Bray C, Bierre P, Gorombey P, Hyun B, Hubbard M, Lange S, et al. Data file standard for flow Cytometry, version FCS 3.1. Cytometry Part A 2010;77A(1):97–100.
4. Koblizek M, Lebedeva A, flowIO FK. Flow cytometry standard conformance testing, editing, and export tool. Cytometry Part A 2018;93A(8):848–853.
5. Bhattacharya S, Dunn P, Thomas CG, Smith B, Schaefer H, Chen J, Hu Z, Zalocusky KA, Shankar RD, Shen-Orr SS, et al. ImmPort, toward repurposing of open access immunological assay data for translational and clinical research. Sci Data 2018;5:180015.
6. Spidlen J, Breuer K, Rosenberg C, Kotecha N, Brinkman RR. FlowRepository: A resource of annotated flow cytometry datasets associated with peer-reviewed publications. Cytometry Part 2012;81A(9):727–731.
7. Chen TJ, Kotecha N. Cytobank: Providing an analytics platform for community cytometry data analysis and collaboration. Curr Top Microbiol Immunol 2014;377:127–157.
8. Bras AE, van der Velden VHJ. Lossless compression of Cytometric data. Cytometry Part A 2019;95A(10):1108–1112.