TDS Facility

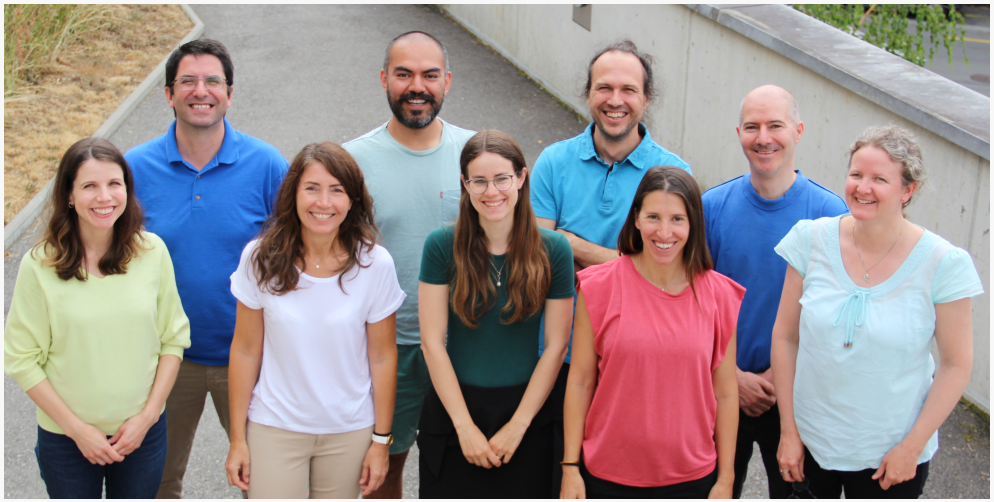
Analysis of flow cytometry data with R
Training for life scientists

João Lourenço, Tania Wyss & Nadine Fournier

Translational Data Science – Facility

SIB Swiss Institute of Bioinformatics

The Translational Data Science Facility



- Part of the **SIB Swiss Institute of Bioinformatics**
- Located at the AGORA Cancer Research Center in **Lausanne**
- Provides **statistics, bioinformatics and computational expertise** to molecular biology and applied research labs.
- Participates in fundamental and translational research by providing expertise in **data analysis** of single-cell and bulk multi-omics, spatial transcriptomics, flow cytometry, etc

For core facility service inquiry: nadine.fournier@sib.swiss
https://agora-cancer.ch/scientific-platforms/translational-data-science-facility/
https://www.sib.swiss/raphael-gottardo-group

Tell us about yourself !

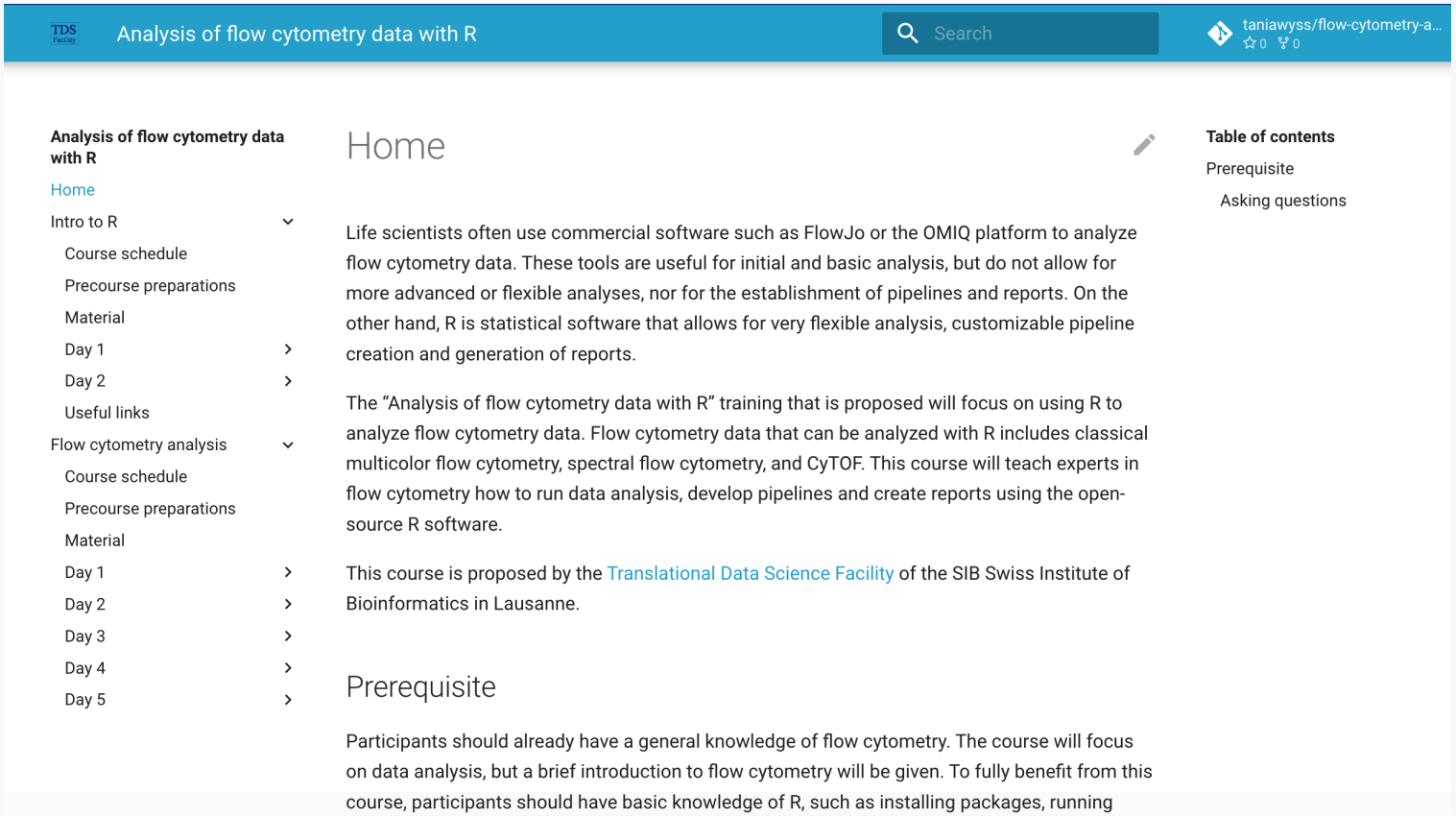
Share about yourself and your research, experience with programming, etc.

Photo by National Cancer Institute, Unsplash

Photo by Scott Graham, Unsplash

Course material

1. Website
https://taniawyss.github.io/flow-cytometry-analysis-with-R/



2. Google doc for exchange of additional information and questions

Outline & Schedule

Day 1 - morning

01 **Introduction**
(9:00 – 10:30)
10:30 – 10:50 Coffee break

02 **Starting to work with flow cytometry data**
(10:50 – 12:30)

Outline & Schedule

Day 1 - afternoon

03

Transformation
(13:30 – 15:30)

15:30 -15:50 Coffee break

04

Automated Quality Control Exercises
(15:50 – 16:50)

16:50 - 17:00 Feedback and end of day

Outline & Schedule

Day 2

05 **Dimensionality reduction**

06 **Clustering and annotation**

07 **Differential testing**

Examples and exercises are integrated in the chapters

Questions and Exercises

Feel free to interrupt with questions by asking them directly or raising your (virtual) hand.

Use the Q&A in Google Doc (or Zoom chat), we will provide answers.

Add a  when you are done with the current exercise.

Exercises in R:

- We will try to debug as much as possible
- We are happy if you share your results or alternative code!

Course Content

Flow cytometry data analysis with R is vast.
We will cover a simple workflow to allow you to:

- get a basic understanding of an analysis workflow
- perform some analysis using R
- give you the tools to expand your workflows according to your needs

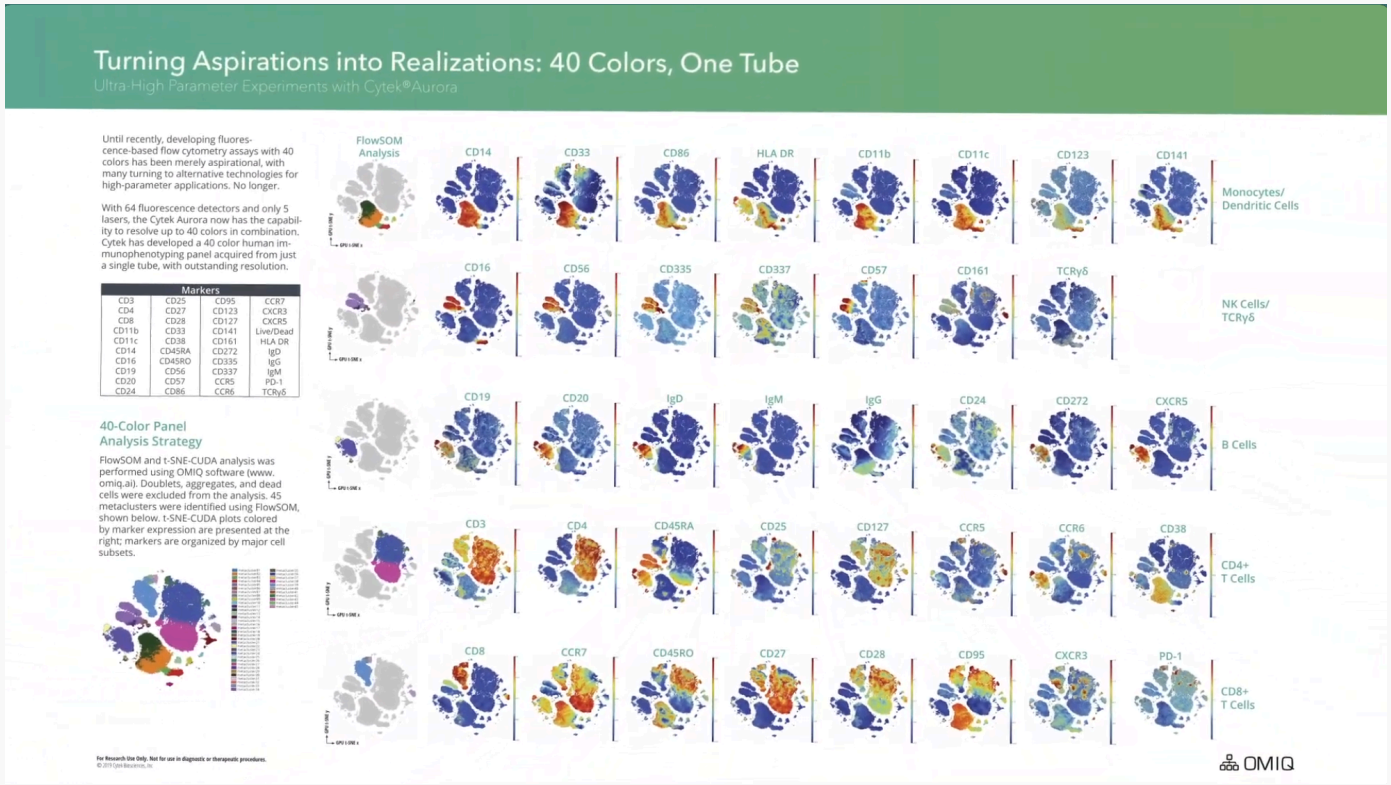
This course is only the first step in your  journey!

Introduction

Why use R for flow cytometry data ?

Types of flow cytometry data:

- conventional flow: 15-20 markers per panel
- spectral : up to 40 markers per panel, deals with cell autofluorescence => complexity of the analysis if using 2D manual gating strategy
- R can facilitate the analysis of datasets with many markers



https://cytekbio.com/pages/aurora

Why use R for flow cytometry data ?

- Commercially available solutions : Cytek's SpectroFlo software, OMIQ
- Online solutions: data privacy issues?

=> R is free and open source

- Allows reproducibility and transparency, everything is hard-coded.
- R offers capabilities to perform analyses beyond the ones of the standard data analysis software via development of packages by the R community.
- Generate PDF or HTML reports

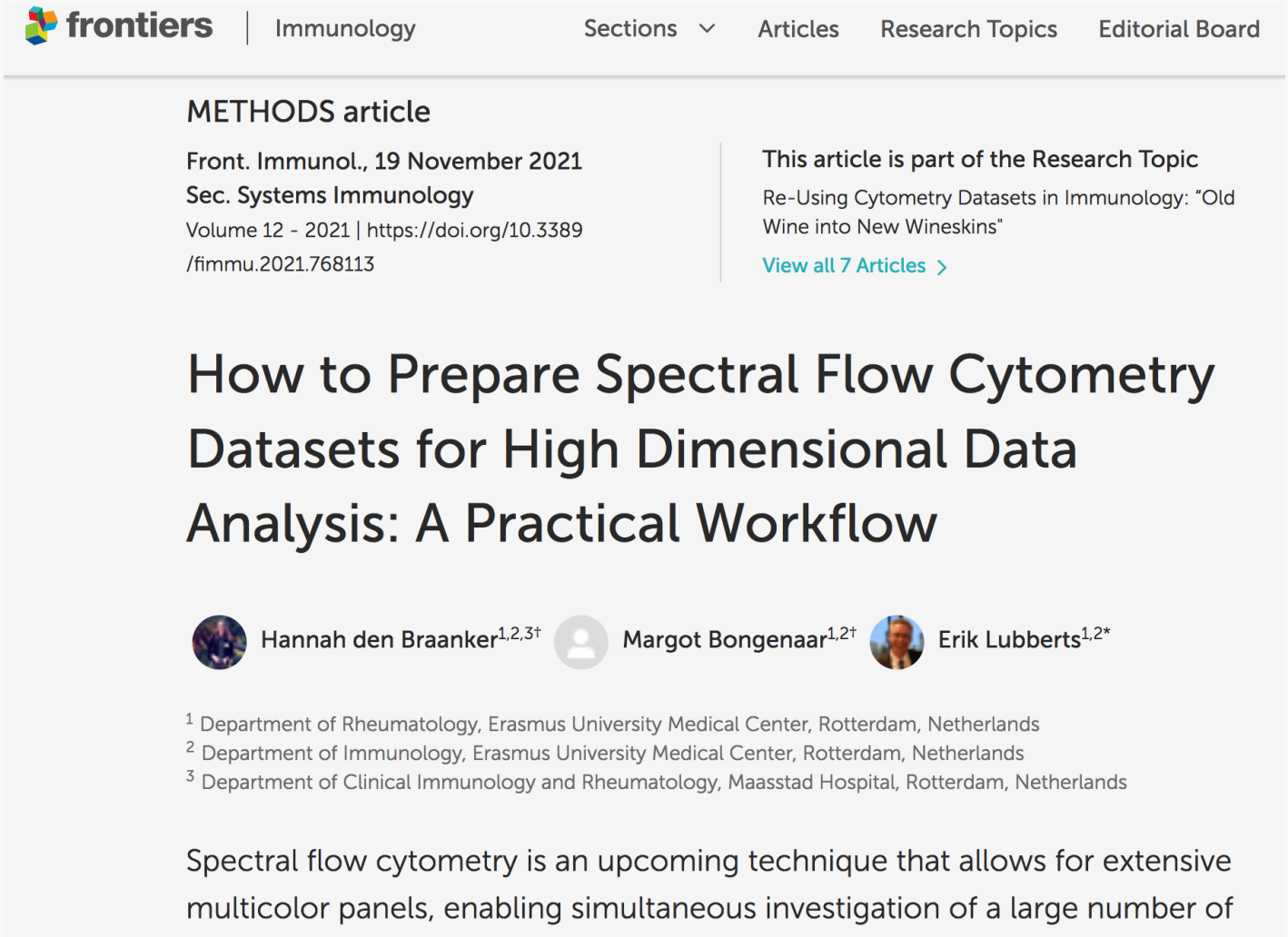
- Analysis with R may be different than the usual 2D gating mind-set.

Availability of R packages

- CRAN
- Bioconductor
- (github)

Spectral flow cytometry analysis workflow

- Workflow based on



frontiers | Immunology Sections Articles Research Topics Editorial Board

METHODS article

Front. Immunol., 19 November 2021
Sec. Systems Immunology
Volume 12 - 2021 | https://doi.org/10.3389/fimmu.2021.768113

This article is part of the Research Topic
Re-Using Cytometry Datasets in Immunology: "Old Wine into New Wineskins"
View all 7 Articles >

How to Prepare Spectral Flow Cytometry Datasets for High Dimensional Data Analysis: A Practical Workflow

Hannah den Braanker[1,2,3†] Margot Bongenaar[1,2†] Erik Lubberts[1,2*]

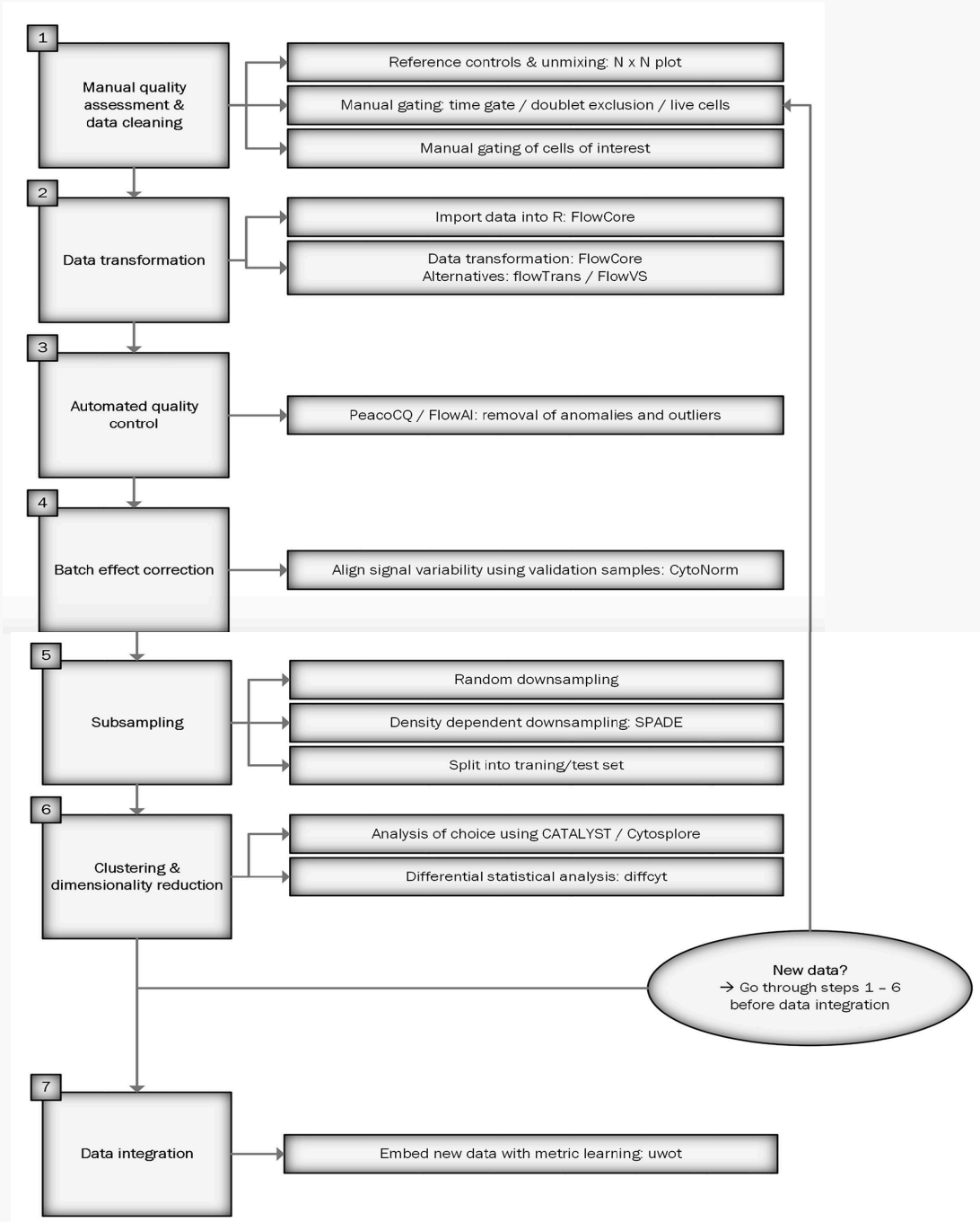
[1] Department of Rheumatology, Erasmus University Medical Center, Rotterdam, Netherlands
[2] Department of Immunology, Erasmus University Medical Center, Rotterdam, Netherlands
[3] Department of Clinical Immunology and Rheumatology, Maasstad Hospital, Rotterdam, Netherlands

Spectral flow cytometry is an upcoming technique that allows for extensive multicolor panels, enabling simultaneous investigation of a large number of

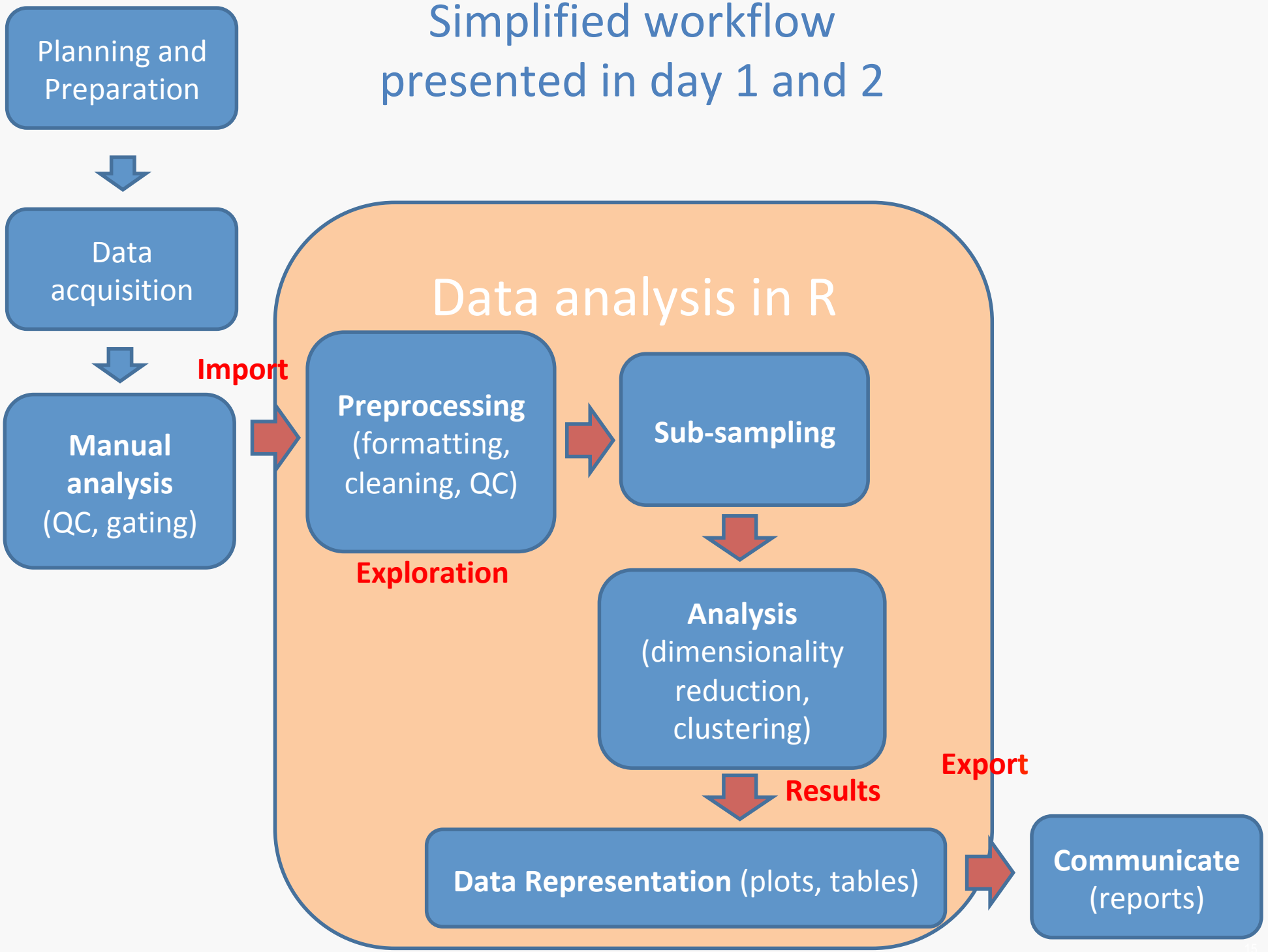
Provide R code to perform the proposed workflow

https://doi.org/10.3389/fimmu.2021.768113

Suggested workflow: Figure 1, den Braanker et al



Simplified workflow presented in day 1 and 2

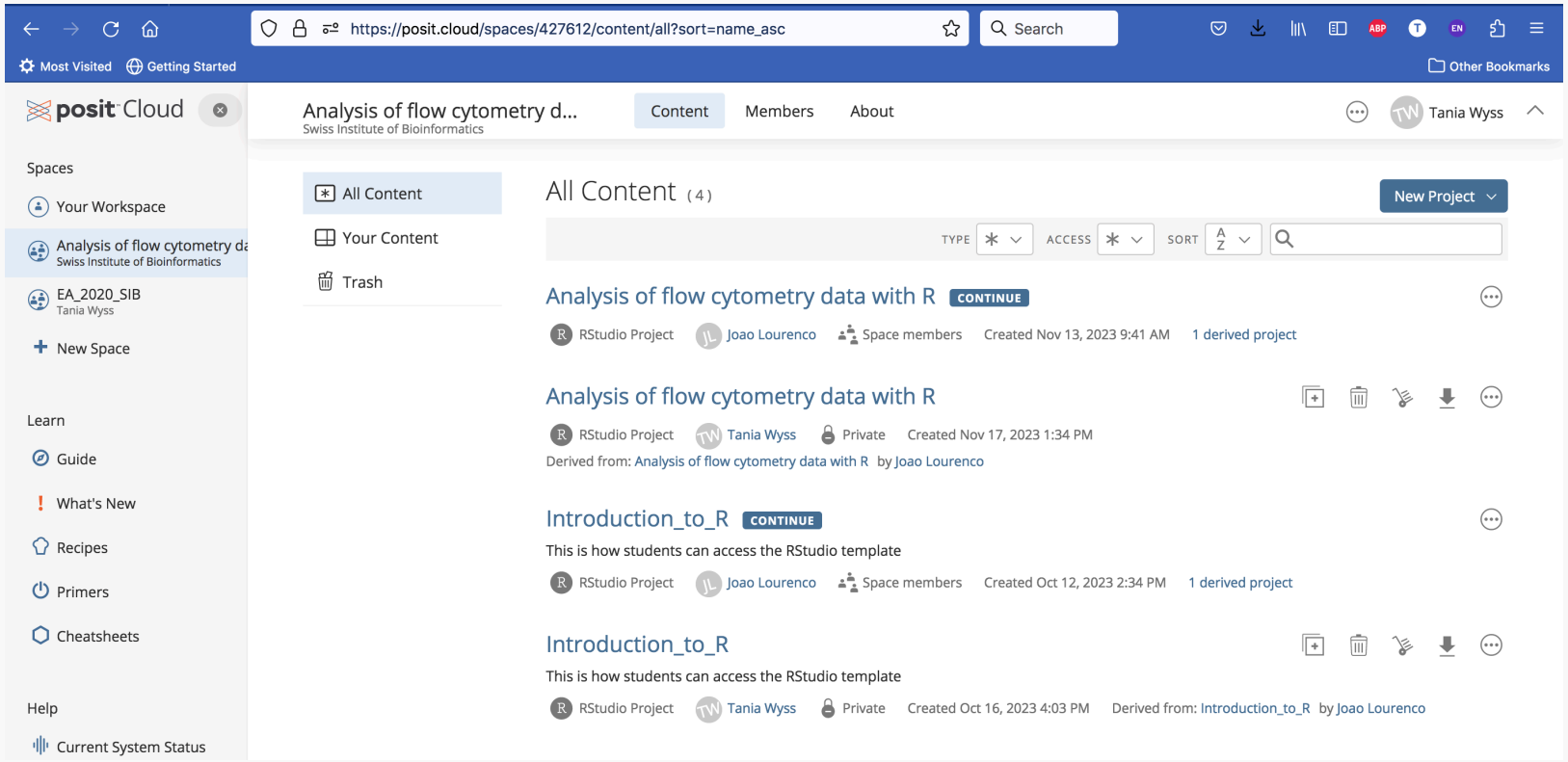


1. Manual QC check and gating
Initial recommendations

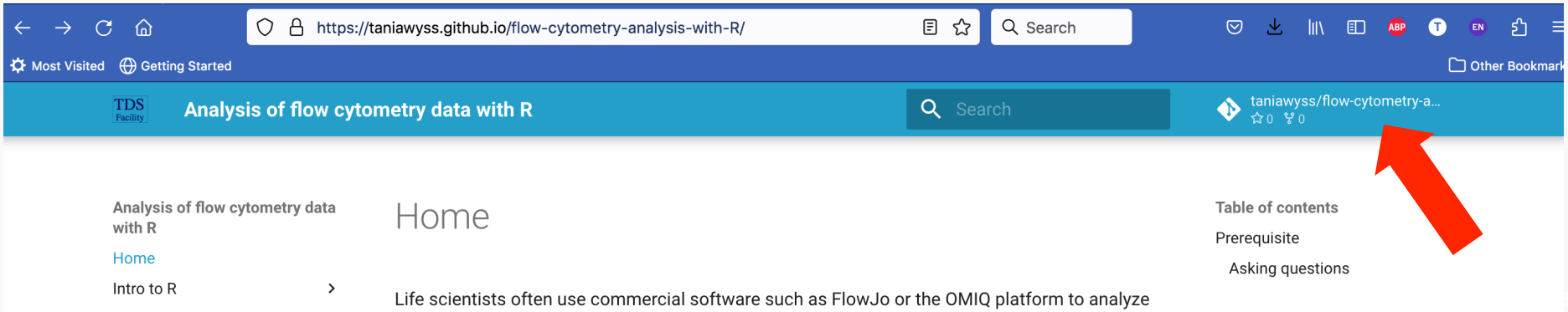
- Well designed panel
- Well designed single-stain controls
- Manual quality checks and gating of each sample: time gate, select single cells, viable cells, cells of interest.

- We will start the basic workflow using fcs files exported from FlowJo after QC checks and initial gating.

R environment and cloud



Github repository



In a Nutshell

- We present some useful packages to build a basic workflow, but “there is more than one way to do it”!

- We encourage you to search for packages that have functions that could suit your needs.

02 Starting to work with flow cytometry data

Workflow and source of flow cytometry dataset

frontiers in Immunology

METHODS
published: 19 November 2021
doi: 10.3389/fimmu.2021.768113

Check for updates

How to Prepare Spectral Flow Cytometry Datasets for High Dimensional Data Analysis: A Practical Workflow

Hannah den Braanker[1,2,3†], *Margot Bongenaar*[1,2†] *and Erik Lubberts*[1,2*]

[1] *Department of Rheumatology, Erasmus University Medical Center, Rotterdam, Netherlands,* [2] *Department of Immunology, Erasmus University Medical Center, Rotterdam, Netherlands,* [3] *Department of Clinical Immunology and Rheumatology, Maasstad Hospital, Rotterdam, Netherlands*

https://doi.org/10.3389/fimmu.2021.768113

Example of flow cytometry dataset

- Publicly available through the **FlowRepository database** at https://flowrepository.org/, using repository ID FR-FCM-Z4KT
- Data from **31-color spectral flow cytometry** on peripheral blood mononuclear cells (**PBMCs**) from healthy controls
- Data were acquired and unmixed using SpectroFlo® v2.2.0.3 software (Cytek Biosciences, Fremont, California, USA)
- **Resulting unmixed fcs files were pre-processed using manual gating** in FlowJo v10.7 software (BD Biosciences, San Jose, California, USA)

Flow Cytometry Standard (FCS) files

- Data standard for reading and writing data from flow cytometry experiments
- File exported from the cytometer's acquisition software
- Versions: FCS1.o (1984), FCS 2.0 (1990), FCS 3.0 (1997), FCS 3.1 (2010),
- File Format (main segments):
  - HEADER segment (ASCII text): version, …
  - TEXT segment (ASCII text): keywords and values which describe the data format and encoding
  - DATA segment (binary): contains the actual measurements
  - Others …

Data structure

- Array (matrix) with fluorescence and scatter channels represented in columns and individual «events» (cells…) forming the rows

Channels

Events

	FSC-A	FSC-H	SSC-A	B515-A	R780-A	R710-A	R660-A	V800-A	V655-A
[1,]	27700.75	27291.75	177.52585	1984.485	625.0796	1232.1008	748.5101	1553.0295	1350.2565
[2,]	41264.25	39764.25	320.12296	3639.620	539.7032	1433.3112	1470.2659	2217.6750	2305.3516
[3,]	65054.75	57606.25	203.01607	2191.861	198.6541	726.9798	766.2198	802.2521	809.9579
[4,]	30584.00	31664.50	130.68690	1873.409	1304.0895	2528.7083	784.6980	1702.3671	1185.8608
[5,]	39505.75	39626.00	203.25166	2540.620	323.2625	857.1525	715.0004	1117.4775	1746.5798
[6,]	33171.50	34794.00	333.64246	2192.864	1408.8563	2573.5095	1604.2236	2128.1748	1727.5891
[7,]	63711.00	54475.50	1122.48340	3879.044	1730.8085	3573.5652	1691.8744	5106.0596	3578.0332
[8,]	40000.75	40213.50	236.54262	2545.858	1081.6753	2313.5962	1411.0983	2989.7524	1920.4047
[9,]	49286.00	49182.50	78.61845	1601.092	123.2834	493.6364	242.0255	633.3533	759.2227
[10,]	32209.75	33368.25	203.29897	2387.361	1056.0723	1769.4005	939.7758	1693.8635	1579.7000
[11,]	35937.25	36212.50	220.66580	2901.591	1218.1395	3202.3853	1059.7604	2443.0205	2253.0146
[12,]	32905.50	33897.50	233.98033	2726.240	1952.0721	3405.7139	2726.1091	2988.6882	2011.0159
[13,]	36028.50	35845.50	219.18674	3221.668	2542.3389	3895.0371	2283.0444	3331.8298	2479.6580
[14,]	38616.00	38775.00	218.46669	3218.305	582.6801	1022.7971	1255.5858	2150.4185	1993.2681
[15,]	45282.25	42223.25	1173.74487	6941.545	705.4651	1649.9570	1615.0811	4287.2036	3778.2302
[16,]	36246.25	36207.75	189.15569	3049.417	1736.7826	2823.7266	1031.0308	2824.6582	2053.6843
[17,]	29282.75	29884.00	209.64102	1836.197	612.2673	1149.7164	870.3303	1720.2170	1525.6914
[18,]	57757.25	54448.25	1999.17517	12972.877	4364.5908	11298.7070	6745.5039	20934.3457	17057.1934
[19,]	33301.00	33093.50	208.47151	2146.622	429.5022	855.5981	845.9418	1207.8969	1297.2683
[20,]	34478.25	35390.75	211.26921	3060.585	2016.3651	3442.5408	1348.4852	2673.9729	2259.8494
[21,]	29406.25	28219.50	231.55798	3008.380	997.8875	2319.5779	1514.2091	1757.2463	1675.9983
[22,]	49978.50	48517.75	537.04224	3122.343	981.1232	2252.1189	1861.3472	2518.4731	2230.5327
[23,]	39872.50	37620.75	198.75706	2719.222	1657.0939	2945.5713	1025.1293	2203.0527	1670.1367
[24,]	33395.00	35331.75	220.46056	2664.632	690.1926	1483.0898	1736.9537	1397.0316	1982.9124
[25,]	46976.00	47355.25	231.33037	2530.461	537.1376	1194.0681	1072.7083	1531.7494	1766.5841
[26,]	56663.75	51458.25	223.06416	3217.866	398.6222	1279.4880	1207.4561	1268.9905	1553.6884
[27,]	50818.75	48556.25	305.77182	3714.351	577.0732	1364.4095	1064.0983	1633.2513	2077.0466
[28,]	36225.25	36196.75	180.30524	2636.466	946.7570	2138.4143	1695.0502	1807.8429	2057.7292
[29,]	28509.25	30715.50	230.27397	1072.201	1867.2009	1643.1423	882.4811	1201.5806	688.1475
[30,]	37198.75	36200.50	237.67776	3046.719	1376.3452	2580.9287	1326.2197	2599.6101	2196.7258

Intensities

flowCore R Package

- https://bioconductor.org/packages/release/bioc/html/flowCore.html
- Provides data structures and basic functions to deal with flow cytometry data in R
- Installation:

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("flowCore")
```

- Vignette https://bioconductor.org/packages/release/bioc/vignettes/flowCore/inst/doc/HowTo-flowCore.pdf

Reading an FCS file into a *flowFrame*

- *A **flowFrame*** is the basic unit of manipulation
- Corresponds to a single FCS file

The function **read.FCS()** allows to read a single FCS file into R. Example:

```
> FCS_file <- read.FCS(
            filename = "course_datasets/FR_FCM_Z4KT/
T_cells_REU270_alive_T cells.fcs",
            transformation = FALSE,
           truncate_max_range = FALSE)
```

- Important arguments:
  - **filename** is the path to the fcs file
  - **transformation** specifies the type of transformation to be applied. When set to FALSE, no transformation is applied.
  - **truncate_max_range**. Set to FALSE to avoid truncating the extreme positive value to the instrument measurement range.

What is a flowFrame object?

> help(flowFrame)

flowFrame-class {flowCore} R Documentation

'flowFrame': a class for storing observed quantitative properties for a population of cells from a FACS run

Description

This class represents the data contained in a FCS file or similar data structure. There are three parts of the data:

1. a numeric matrix of the raw measurement values with `rows=events` and `columns=parameters`
2. annotation for the parameters (e.g., the measurement channels, stains, dynamic range)
3. additional annotation provided through keywords in the FCS file

Details

Objects of class `flowFrame` can be used to hold arbitrary data of cell populations, acquired in flow-cytometry.

What is a flowFrame object ?

- In R, objects such as flowFrames are **collections of data (variables)** and **methods (functions).**
- They belong to a given **class** (a blueprint for that object)
- Member variables in R objects are called **slots**. There are three slots in a flowFrame: *exprs*, *parameters* and *description*

Slots

`exprs`

Object of class `matrix` containing the measured intensities. Rows correspond to cells, columns to the different measurement channels. The `colnames` attribute of the matrix is supposed to hold the names or identifiers for the channels. The `rownames` attribute would usually not be set.

`parameters`

An `AnnotatedDataFrame` containing information about each column of the `flowFrame`. This will generally be filled in by `read.FCS` or similar functions using data from the FCS keywords describing the parameters.

`description`

A list containing the meta data included in the FCS file.

Summarize a *flowFrame*

```
> FCS_file
```

```
flowFrame object 'T_cells_REU270_alive_T cells.fcs'
with 315735 cells and 39 observables:
                        name       desc      range  minRange  maxRange
$P1                    FSC-A         NA    4194304         0   4194303
$P2                    FSC-H         NA    4194304         0   4194303
$P3                    SSC-A         NA    4194304         0   4194303
$P4                  SSC-B-A         NA    4194304         0   4194303
$P5                  SSC-B-H         NA    4194304         0   4194303
...                      ...        ...        ...       ...       ...
$P35 FJComp-PerCP-eFluor ..       CD127     100000      -111     99999
$P36 FJComp-Spark Blue 55..         CD3     100000         0     99999
$P37     FJComp-Zombie UV-A  Zombie UV     100000      -111     99999
$P38    FJComp-eFluor 660-A     CTLA-4     100000      -111     99999
$P39                   Time         NA        166         0       165
278 keywords are stored in the 'description' slot
```

```
> summary(FCS_file)
```

```
                FSC-A      FSC-H     SSC-A    SSC-B-A   SSC-B-H     SSC-H FJComp-AF-A
Min.         248252.4   200055.0 104527.1   54734.91   43496.0   71728.0  -85312.109
1st Qu.      694239.2   544927.0 371822.5 216510.48 163184.0 290444.0  -11530.027
Median       794500.6   628644.0 453873.5 263345.84 195338.0 348003.0   -7938.739
Mean         809134.7   639853.2 452449.1 263406.04 195808.8 347746.4   -8392.375
3rd Qu.      908499.2   722281.5 527944.0 307300.50 226672.5 402455.0   -4571.343
Max.        1608623.4  1358182.0 946566.2 608151.94 449530.0 791039.0   37643.547
```

```
...
```

```
      Time
 0.00000
39.13176
81.05503
81.00381
121.98465
162.51257
```

Access data elements in a *flowFrame*

- To access data: use the @ operator or a method (function)
- Matrix of expression values (as a matrix)

```
> FCS_file@exprs
```

or

```
> exprs(FCS_file)
```

```
        FSC-A  FSC-H    SSC-A  SSC-B-A SSC-B-H  SSC-H FJComp-AF-A FJComp-APC-A
[1,] 708579.4 593958 331966.4 195681.8  161726 273584  -12322.742   -4990.1958
[2,] 587231.9 489906 323881.8 209247.5  165442 265458  -10672.745   -5642.0508
[3,] 828618.7 662813 487978.5 289251.3  215334 379895   -1366.873   -3940.6289
[4,] 733458.1 606898 447868.5 242895.0  188230 357695   -2092.956    -998.5401
[5,] 576551.5 461784 428876.1 238000.4  175819 326038   -6251.983   -5225.1035
[6,] 762848.1 606807 583804.5 344976.0  251346 444231  -10864.361   -4390.9263
```

```
> colnames(FCS_file)
```

```
 [1] "FSC-A"                  "FSC-H"                  "SSC-A"
 [4] "SSC-B-A"                "SSC-B-H"                "SSC-H"
 [7] "FJComp-AF-A"            "FJComp-APC-A"           "FJComp-APC-Fire 750-A"
[10] "FJComp-APC-Fire 810-A"  "FJComp-APC-R700-A"      "FJComp-BB515-A"
[13] "FJComp-BB700-A"         "FJComp-BUV395-A"        "FJComp-BUV496-A"
[16] "FJComp-BUV563-A"        "FJComp-BUV615-A"        "FJComp-BUV661-A"
[19] "FJComp-BUV737-A"        "FJComp-BUV805-A"        "FJComp-BV421-A"
[22] "FJComp-BV480-A"         "FJComp-BV510-A"         "FJComp-BV570-A"
[25] "FJComp-BV605-A"         "FJComp-BV650-A"         "FJComp-BV711-A"
[28] "FJComp-BV750-A"         "FJComp-BV785-A"         "FJComp-PE-A"
[31] "FJComp-PE-Cy5-A"        "FJComp-PE-Cy7-A"        "FJComp-PE-Dazzle594-A"
[34] "FJComp-PerCP-A"         "FJComp-PerCP-eFluor 710-A" "FJComp-Spark Blue 550-A"
[37] "FJComp-Zombie UV-A"     "FJComp-eFluor 660-A"    "Time"
```

Access data elements in a *flowFrame*

- Metadata (panel)

```
> pData(FCS_file@parameters)
```
or
```
> pData(parameters(FCS_file))
```

	name	desc	range	minRange	maxRange
$P1	FSC-A	NA	4194304	0	4194303
$P2	FSC-H	NA	4194304	0	4194303
...	...	...	...	...	...
$P35	FJComp-PerCP-eFluor ..	CD127	100000	-111	99999
$P36	FJComp-Spark Blue 55..	CD3	100000	0	99999
$P37	FJComp-Zombie UV-A	Zombie UV	100000	-111	99999
$P38	FJComp-eFluor 660-A	CTLA-4	100000	-111	99999
$P39	Time	NA	166	0	165

How to replace the channel names by the antigen names in the expression matrix

- Copy the metadata to a data frame

```
> panel <- pData(FCS_file@parameters)
```

- Copy the names to a new column

```
> pData(FCS_file@parameters)$channel <- panel$name
```

- Replace the names by the antigens

```
> colnames(FCS_file)[!is.na(panel$desc)] <- panel$desc[!
  is.na(panel$desc)]
```

```
> head(exprs(FCS_file)[,10:15])
```

	CD27	LAG-3	CD25	CD49b	CD8	CD4
[1,]	34010.4844	-726.74323	2337.454	622.7443	-1674.8558	54145.7031
[2,]	26705.5781	-447.67514	3196.554	1380.7151	-1855.1270	64054.7617
[3,]	846.9209	246.76016	1000.591	581.6843	-977.6837	-219.4745
[4,]	1110.7271	-507.84625	1215.742	1224.3079	1438.7726	-2148.8167
[5,]	3685.3149	-1773.50989	4398.276	818.2174	-1662.8772	88996.8984
[6,]	505.4961	-21.17858	2401.317	-358.6945	451.0627	-2950.0181

Reading a list of FCS files into a *flowSet*

- *A **flowSet*** is a collection of *flowFrame*
- Convenient way to apply methods to all *flowFrame* simultaneously

The function **read.flowSet()** allows to read several FCS files in a given directory.

Example:

```
> fcs_data <- read.flowSet(path="course_datasets/FR_FCM_Z4KT/",
                           pattern="*.fcs",
                           transformation = FALSE,
                           truncate_max_range = FALSE)
```

- Important arguments:
  - **path** is the path to the folder containing the FCS files
  - **pattern** sets which files to read (* is a wildcard replacing the file names)

You can coerce a list of *flowFrames* into a *FlowSet*, but is less convenient

Slots in a *flowSet*

> help(flowSet)

flowSet-class {flowCore} R Documentation

'flowSet': a class for storing flow cytometry raw data from quantitative cell-based assays

Description

This class is a container for a set of `flowFrame` objects

Slots

`frames`

An `environment` containing one or more `flowFrame` objects.

`phenoData`

An `AnnotatedDataFrame` containing the phenotypic data for the whole data set. Each row corresponds to one of the `flowFrame`s in the `frames` slot. The `sampleNames` of `phenoData` (see below) must match the names of the `flowFrame` in the `frames` environment.

Methods applied to a *flowSet*
List sample names

```
> sampleNames(fcs_data)
```

```
 [1] "T_cells_REU267_alive_T_cells.fcs"           "T_cells_REU268_alive_T_cells.fcs"
 [3] "T_cells_REU269_alive_T_cells.fcs"           "T_cells_REU270_alive_T_cells.fcs"
 [5] "T_cells_REU271_12_july_alive_T_cells.fcs"   "T_cells_REU271_13_april_alive_T_cells.fcs"
 [7] "T_cells_REU271_14_april_alive_T_cells.fcs" "T_cells_REU271_7_apr_alive_T_cells.fcs"
 [9] "T_cells_REU271_9_april_alive_T_cells.fcs"   "T_cells_REU271_alive_T_cells.fcs"
[11] "T_cells_REU272_12_july_alive_T_cells.fcs"   "T_cells_REU272_13_april_alive_T_cells.fcs"
[13] "T_cells_REU272_14_april_alive_T_cells.fcs" "T_cells_REU272_7_apr_alive_T_cells.fcs"
[15] "T_cells_REU272_9_april_alive_T_cells.fcs"   "T_cells_REU272_alive_T_cells.fcs"
```

We can change the sample names:

```
> sampleNames(fcs_data) <- c("REU267","REU268","REU269","REU270",
                             "REU271_12_july","REU271_13_april",
                                    "REU271_14_april","REU271_7_apr",
                             "REU271_9_april","REU271","REU272_12_july",
                             "REU272_13_april","REU272_14_april",
                             "REU272_7_apr","REU272_9_apri","REU272")
```

Phenotypic data

- Extract / replace the data frame (or columns thereof) containing actual phenotypic information from the phenoData slot

```
> pData(fcs_data)
                                                    name
REU267                    T_cells_REU267_alive_T cells.fcs
REU268                    T_cells_REU268_alive_T cells.fcs
REU269                    T_cells_REU269_alive_T cells.fcs
REU270                    T_cells_REU270_alive_T cells.fcs
REU271_12_july   T_cells_REU271_12_july_alive_T cells.fcs
REU271_13_april T_cells_REU271_13_april_alive_T cells.fcs
REU271_14_april T_cells_REU271_14_april_alive_T cells.fcs
REU271_7_apr       T_cells_REU271_7_apr_alive_T cells.fcs
REU271_9_april   T_cells_REU271_9_april_alive_T cells.fcs
REU271                    T_cells_REU271_alive_T cells.fcs
REU272_12_july   T_cells_REU272_12_july_alive_T cells.fcs
REU272_13_april T_cells_REU272_13_april_alive_T cells.fcs
REU272_14_april T_cells_REU272_14_april_alive_T cells.fcs
REU272_7_apr       T_cells_REU272_7_apr_alive_T cells.fcs
REU272_9_apri    T_cells_REU272_9_april_alive_T cells.fcs
REU272                    T_cells_REU272_alive_T cells.fcs
```

Add a new column to the phenotypic data

```
> pData(fcs_data)$gender <- c(rep("male",8), rep("female",8))
> pData(fcs_data) # or fcs_data@phenoData@data
```

	name	gender
REU267	T_cells_REU267_alive_T cells.fcs	male
REU268	T_cells_REU268_alive_T cells.fcs	male
REU269	T_cells_REU269_alive_T cells.fcs	male
REU270	T_cells_REU270_alive_T cells.fcs	male
REU271_12_july	T_cells_REU271_12_july_alive_T cells.fcs	male
REU271_13_april	T_cells_REU271_13_april_alive_T cells.fcs	male
REU271_14_april	T_cells_REU271_14_april_alive_T cells.fcs	male
REU271_7_apr	T_cells_REU271_7_apr_alive_T cells.fcs	male
REU271_9_april	T_cells_REU271_9_april_alive_T cells.fcs	female
REU271	T_cells_REU271_alive_T cells.fcs	female
REU272_12_july	T_cells_REU272_12_july_alive_T cells.fcs	female
REU272_13_april	T_cells_REU272_13_april_alive_T cells.fcs	female
REU272_14_april	T_cells_REU272_14_april_alive_T cells.fcs	female
REU272_7_apr	T_cells_REU272_7_apr_alive_T cells.fcs	female
REU272_9_apri	T_cells_REU272_9_april_alive_T cells.fcs	female
REU272	T_cells_REU272_alive_T cells.fcs	female

Manipulating a *flowSet*

- Extract a *flowFrame* from a *flowSet* object using the [[operator

```
> fcs_data[[1]]
```

```
flowFrame object 'T_cells_REU267_alive_T_cells.fcs'
with 265857 cells and 39 observables:
                        name       desc     range  minRange  maxRange
$P1                    FSC-A         NA   4194304         0   4194303
$P2                    FSC-H         NA   4194304         0   4194303
$P3                    SSC-A         NA   4194304         0   4194303
$P4                  SSC-B-A         NA   4194304         0   4194303
$P5                  SSC-B-H         NA   4194304         0   4194303
```

- Create a new *flowSet* object by subsetting with the [operator

```
> fcs_data[1:5]
```

```
A flowSet with 5 experiments.

column names(39): FSC-A FSC-H ... FJComp-eFluor 660-A Time
```

Manipulating a *flowSet*

- Subset a *flowSet* based on a condition

```
> fcs_data_males <- fcs_data[pData(fcs_data)$gender=="male"]
> fcs_data_females <- subset(fcs_data, pData(fcs_data)$gender=="female”)
```

- Split the *flowSet* based on a condition

```
> fcs_data_split <- split(fcs_data, pData(fcs_data)$gender)
> names(fcs_data_split)
[1] "female" "male"
```

Manipulating a *flowSet*

- Combine several *flowSet objects* (or *flowSets* and *flowFrames*)

```
> fcs_data_combined <-
  rbind2(fcs_data_split$female, fcs_data_split$male)
> pData(fcs_data_combined)
```

	name	gender	split
REU271_9_april	T_cells_REU271_9_april_alive_T cells.fcs	female	female
REU271	T_cells_REU271_alive_T cells.fcs	female	female
REU272_12_july	T_cells_REU272_12_july_alive_T cells.fcs	female	female
REU272_13_april	T_cells_REU272_13_april_alive_T cells.fcs	female	female
REU272_14_april	T_cells_REU272_14_april_alive_T cells.fcs	female	female
REU272_7_apr	T_cells_REU272_7_apr_alive_T cells.fcs	female	female
REU272_9_apri	T_cells_REU272_9_april_alive_T cells.fcs	female	female
REU272	T_cells_REU272_alive_T cells.fcs	female	female
REU267	T_cells_REU267_alive_T cells.fcs	male	male
REU268	T_cells_REU268_alive_T cells.fcs	male	male
REU269	T_cells_REU269_alive_T cells.fcs	male	male
REU270	T_cells_REU270_alive_T cells.fcs	male	male
REU271_12_july	T_cells_REU271_12_july_alive_T cells.fcs	male	male
REU271_13_april	T_cells_REU271_13_april_alive_T cells.fcs	male	male
REU271_14_april	T_cells_REU271_14_april_alive_T cells.fcs	male	male
REU271_7_apr	T_cells_REU271_7_apr_alive_T cells.fcs	male	male

Visualizing Cytometry Data with the *ggcyto* Package

- https://www.bioconductor.org/packages/release/bioc/html/ggcyto.html
- Interface to the ggplot2 graphics system
- Installation:

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("ggcyto")
```

- Vignettes

https://www.bioconductor.org/packages/release/bioc/vignettes/ggcyto/inst/doc/Top_features_of_ggcyto.html

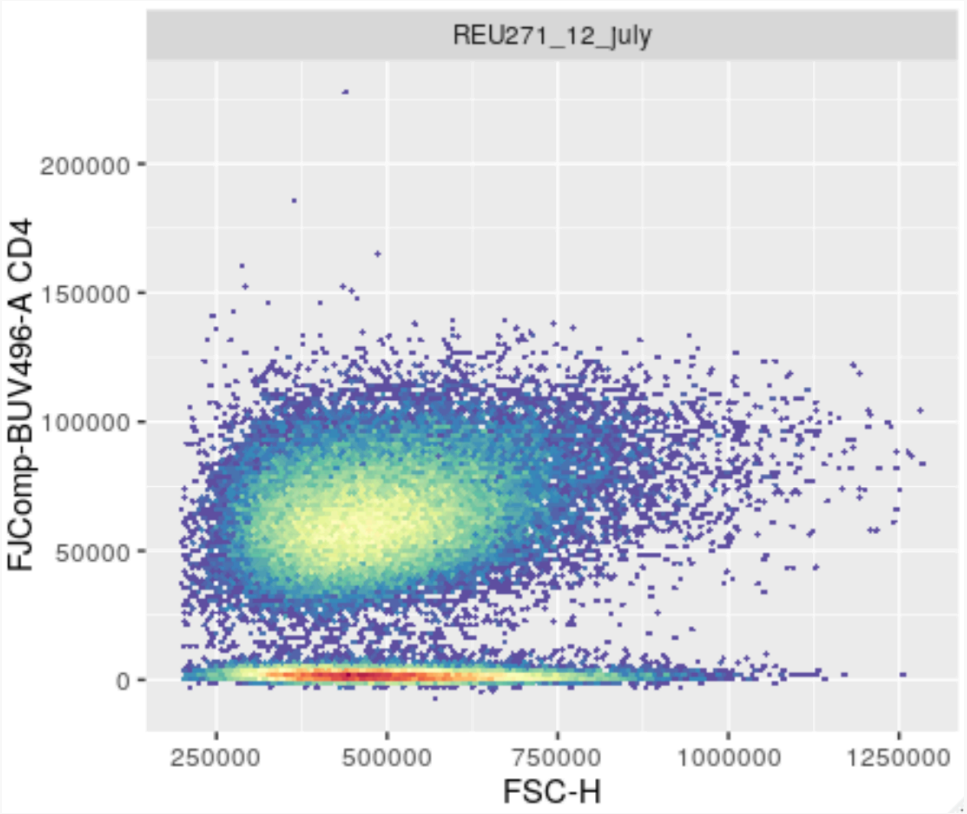
https://www.bioconductor.org/packages/release/bioc/vignettes/ggcyto/inst/doc/ggcyto.flowSet.html

Visualizing a single *flowFrame* within a *flowSet*

The function **autoplot()** can be used to create a **bivariate density plot**.

Example:

```
> autoplot(object = fcs_data[[5]], x="FSC-H", y="FJComp-BUV496-A",
  bins = 2^7)
```

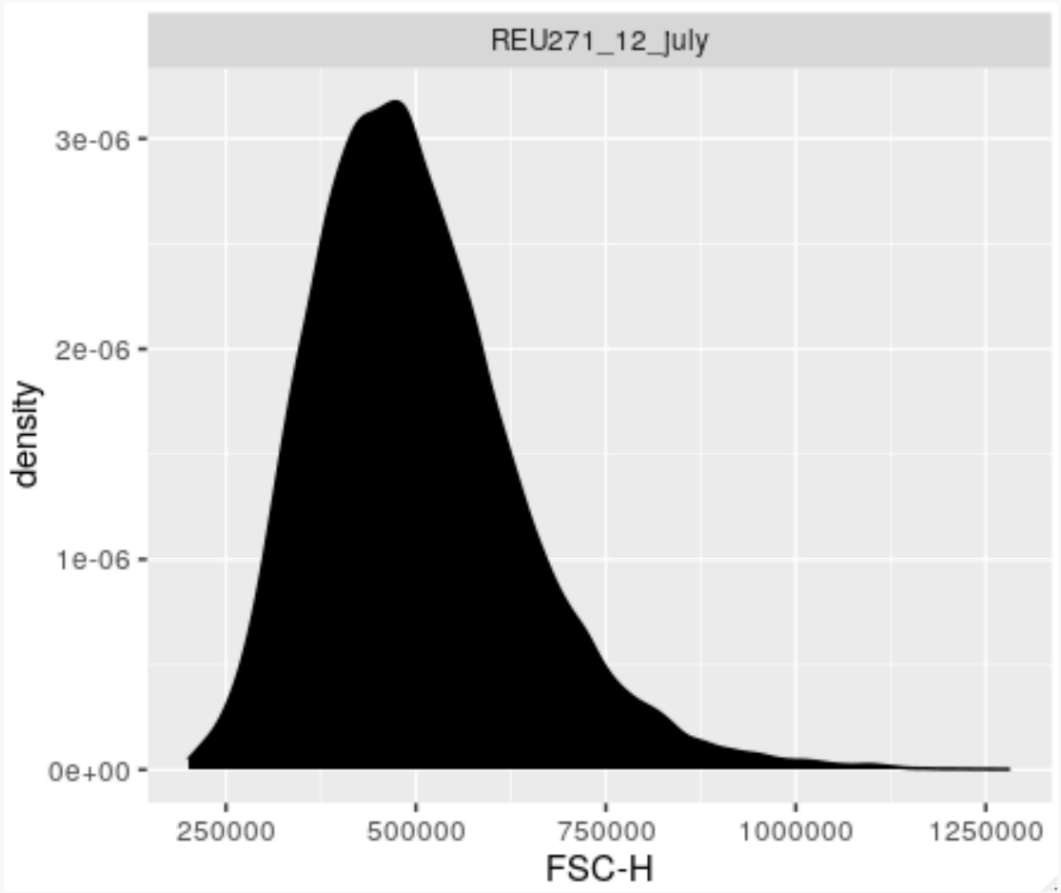


- **bins** sets the granularity of the plot. The higher the number of bins, the finer the granularity

Visualizing a single *flowFrame* within a *flowSet*

Similarly, to get a **univariate densityplot**:

```
> autoplot(object = fcs_data[[5]], x="FSC-H")
```



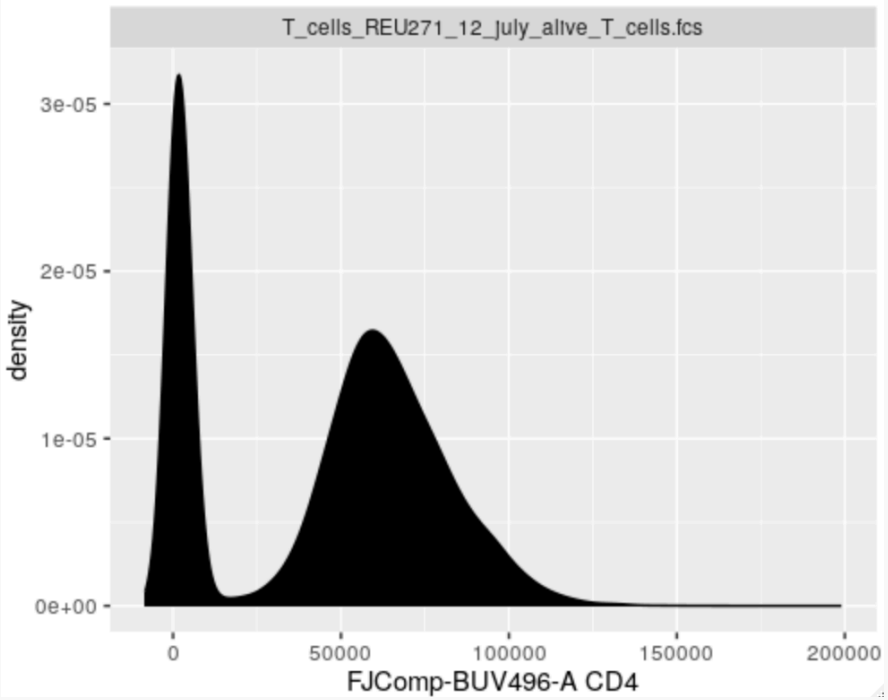
In-line transformation

Use a different scale for the data

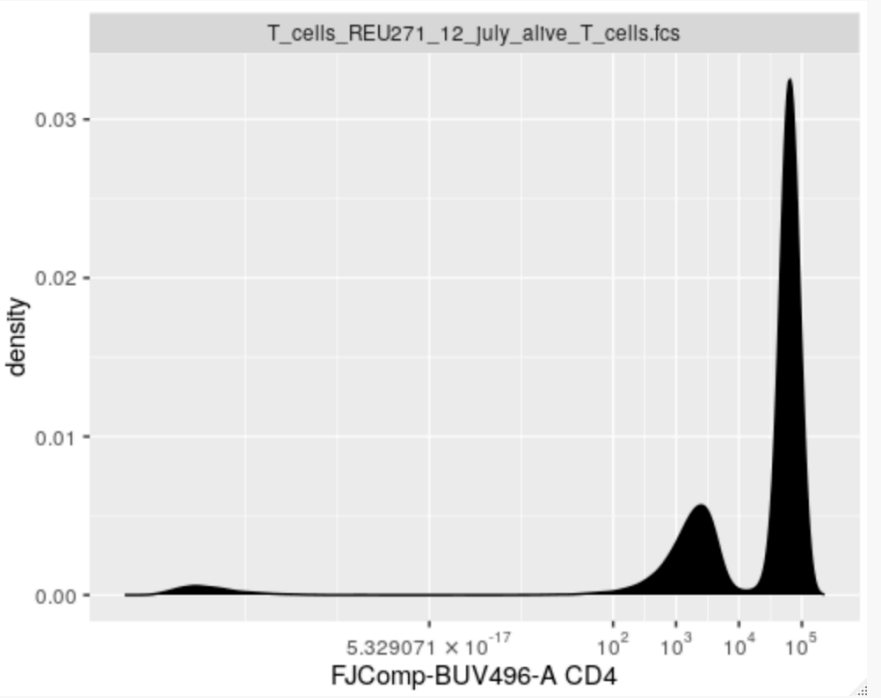
```
> autoplot(fcs_data[[5]],
           x="FJComp-BUV496-A")
```

```
> autoplot(fcs_data[[5]],
  x="FJComp-BUV496-A") +
  scale_x_flowjo_fasinh()
```

Original scale (raw intensity measurements)



flowJo inverse hyperbolic sine

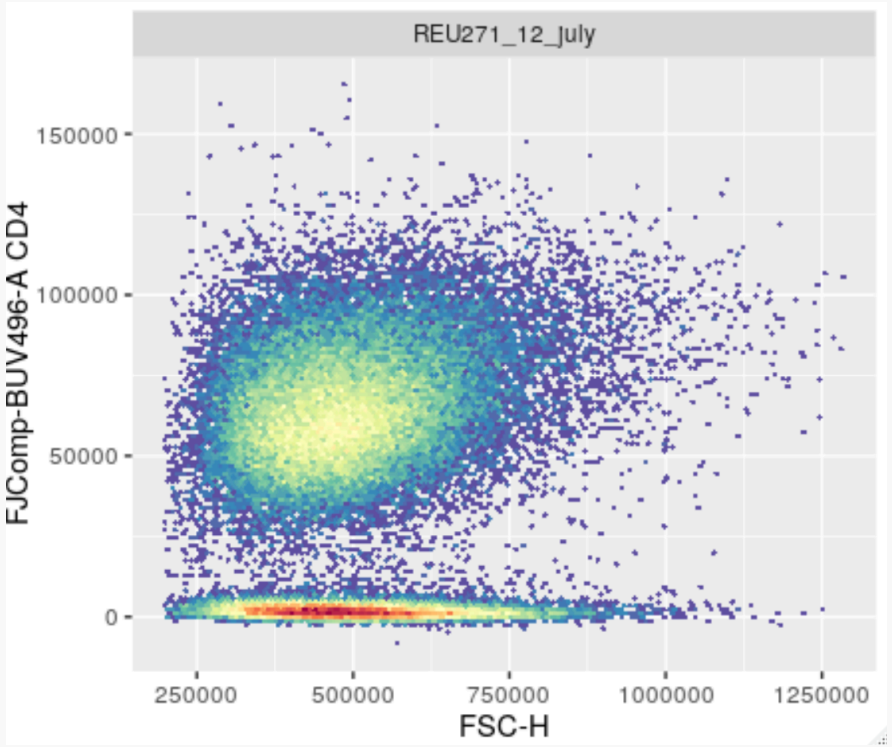


In-line transformation

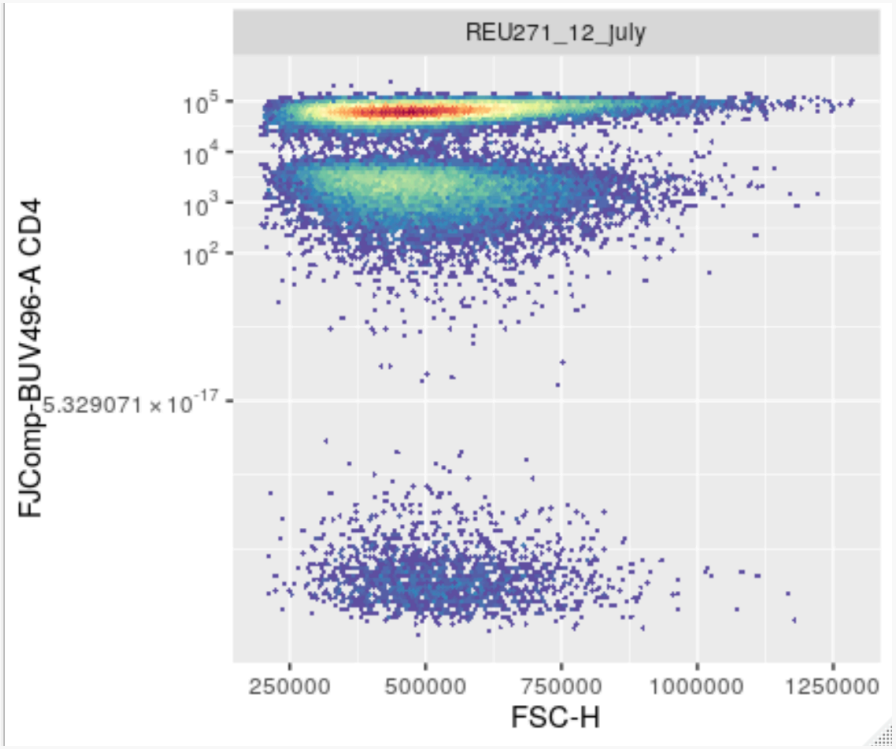
Example in a **bivariate density plot**.

```
> autoplot(object = fcs_data[[5]], x="FSC-H", y="FJComp-BUV496-A",
  bins = 2^7) + scale_y_flowjo_fasinh()
```

Original scale



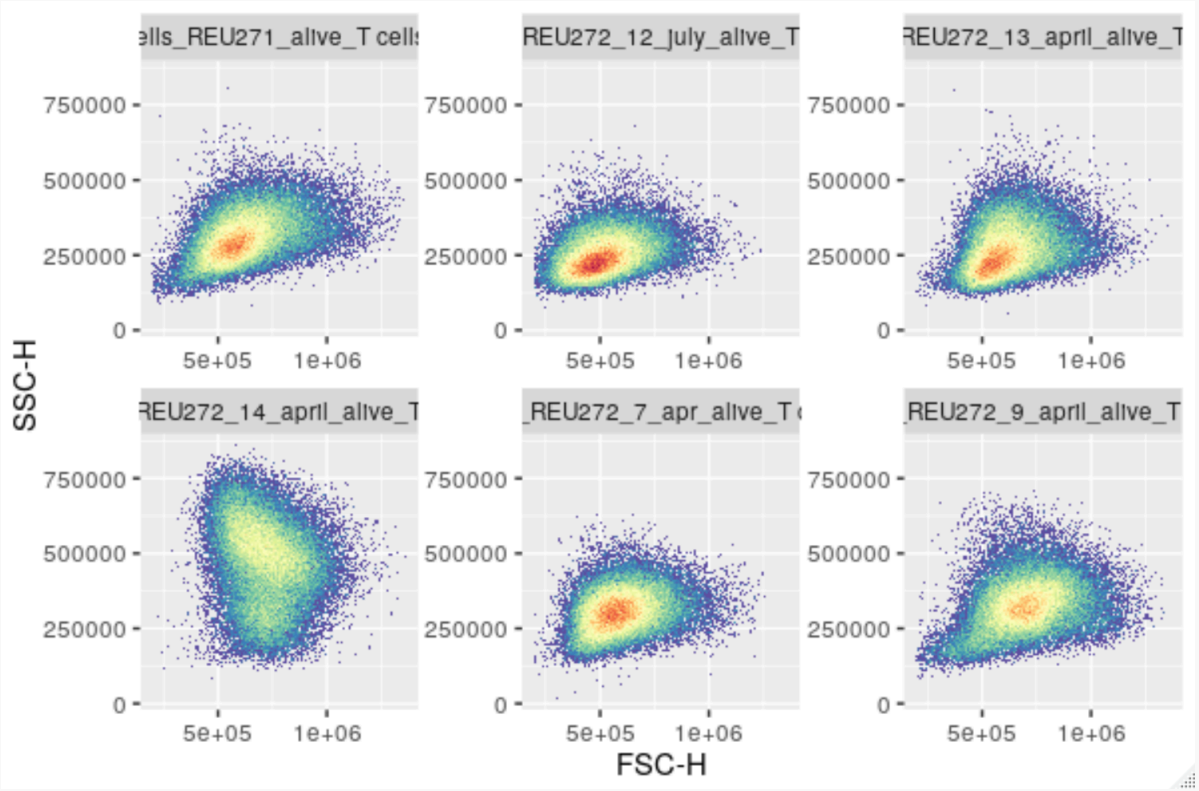
flowJo inverse hyperbolic sine on the y axis



Visualizing a *flowSet*

The syntax is basically the same for *flowSet* objects, with the output now being a grid of plots corresponding to each *flowFrame*. Example:

```
> autoplot(object = fcs_data[10:15], x="FSC-H", y="SSC-H", bins = 2^7)
```



Let's practice – 1

In this exercise we will use a 36-color spectral flow cytometry dataset from a study performed in the context of Covid-19 research. Only a subset from 5 healthy donors will be used. For each healthy donor, there are three time points, as indicated in FCS file names. Data was downloaded through the Flow Repository database (FR-FCM-Z3WR) at https://flowrepository.org/id/FR-FCM-Z3WR. FCS files were pre-gated on live $CD3^{+}CD19^{-}$ T cells in FlowJo.

Create a new script in which you will

1) Import the FCS files (course_datasets/FR_FCM_Z3WR/) into a flowSet. Do not transform or truncate the values
2) Create a data frame with the list of channels and corresponding antigens, and view it . **Hint:** get the antigens from the parameters of one of the flowFrame in the set
3) Add a new column to the phenotypic data with the time point of the sample. View the phenotypic data
4) Convert the channel names in the expression matrices to the corresponding antigen names (where applicable).
5) Create a bivariate density plot showing «FSC-H» against «HLA-DR» for all samples from day 0. Apply a flowJo inverse hyperbolic sine scale to the y axis («HLA-DR»)

In a nutshell

- **FCS files** include the cell measurements and metadata
- FCS files can be imported into the R environment with the **flowCore package**.
- flowCore provides data structures, such as ***flowFrame*** and ***flowSet***, and basic functions to deal with flow cytometry data.
- The **ggcyto package** implements methods for visualization of *flowFrame* and *flowSet* objects, including an interface to the ggplot2 graphics system

Transformation

Construct a data frame of the panel

Retrieve the list of channels and corresponding antigens

```
> fcs_colname <- colnames(fcs_data)
> antigen <- pData(parameters(fcs_data[[1]]))$desc
```

Marker classes:

- Cell “type” markers (used to define clusters representing cell populations)

```
> marker_class <- rep("none", ncol(fcs_data[[1]]))
> marker_class[c(8:31,33:36,38)] <- "type”
```

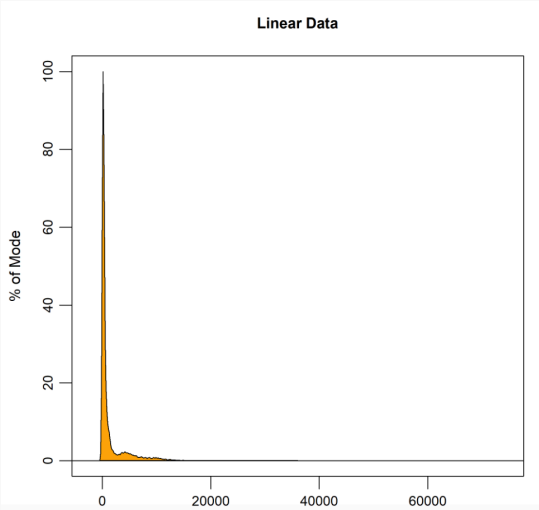
- Cell "state” markers (used for testing differential states within cell populations)

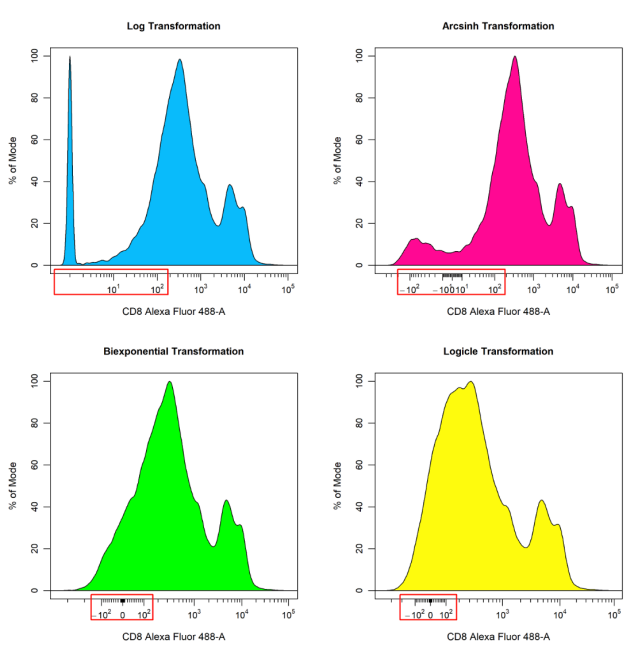
```
> marker_class[32] <- "state”
> marker_class <- factor(marker_class,
                         levels=c("type","state","none"))
```

Put everything together in a data frame

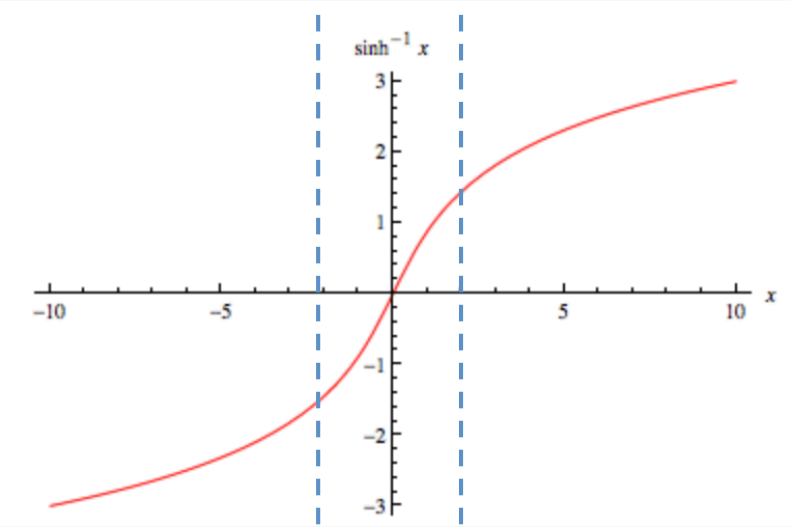
```
> panel <- data.frame(fcs_colname, antigen, marker_class,
                      row.names = NULL)
```

Transformation functions





Inverse hyperbolic sine transformation (arcsinh)

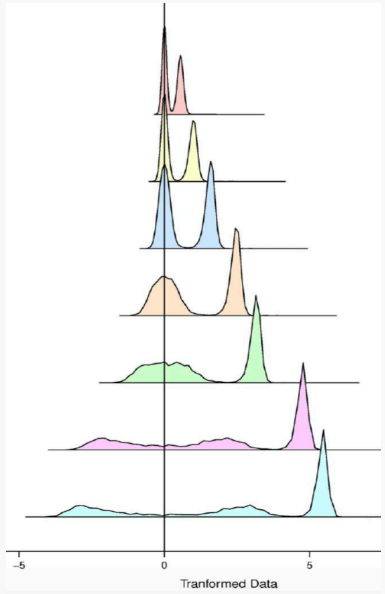


Log-like linear Log-like

Adapted from Folcarelli et al. 2021

Cofactor:

scale argument that controls the behaviour of the function around zero



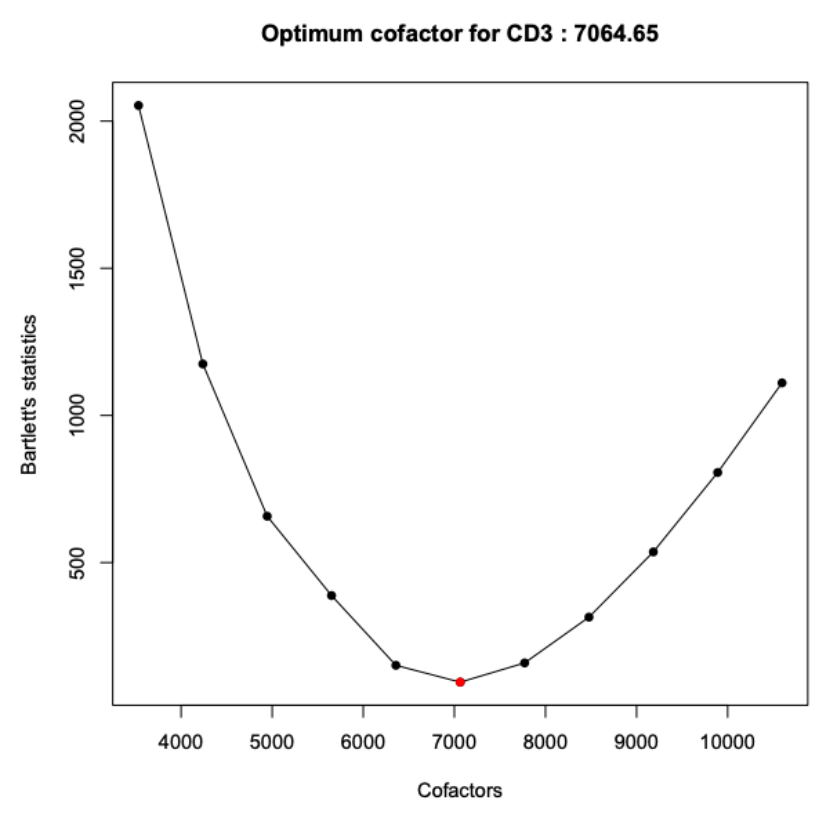
Ray and Pyne 2021

$$arcsinh(x) = \log\left(\frac{x}{c} + \sqrt{\left(\left(\frac{x}{c}\right)^2 + 1\right)}\right)$$

Arcsinh transformation with flowVS

https://bioconductor.org/packages/release/bioc/html/flowVS.html

- Variance stabilization (VS) method based on maximum likelihood (ML) estimation
- Built on top of arcsinh
- Stabilizes the within-population variances for each channel



- Bartlett's statistic (Y axis) is computed from density peaks after data is transformed by different cofactors (X axos)
- An optimum cofactor is obtained where the statistic is minimum

Downsample the data for parameter estimation

Define a function that downsamples all flowFrame objects within a flowSet

```
> Downsampling_flowSet <- function(x, samplesize , replace=TRUE,
prob=NULL){
  if(missing(samplesize))
    samplesize <- min(flowCore::fsApply(x,nrow))
  flowCore::fsApply(x, function(ff){
    i <- sample(nrow(ff), size = samplesize, replace=replace, prob)
    ff[i,]
  })
}
```

Create a downsampled flowSet

```
> fcs_data_small <- Downsampling_flowSet(x=fcs_data,
                                         samplesize = 2000)
```

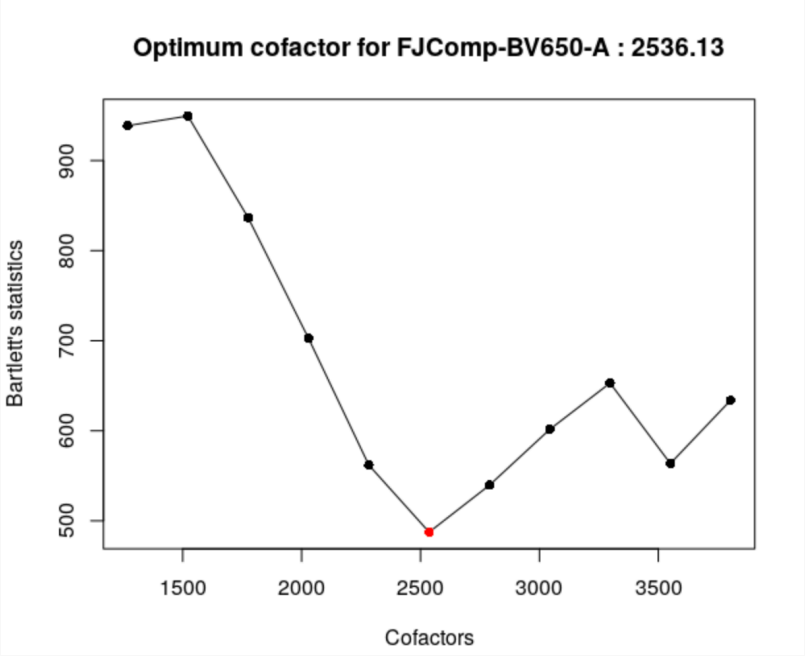
Arcsinh transformation with flowVS

Select markers to be transformed

```
> markerstotransf <- panel$fcs_colname[panel$marker_class!="none"]
```

Estimate cofactors based on the downsampled data

```
> cofactors <- estParamFlowVS(fcs_data_small, channels=markerstotransf)
```



Arcsinh transformation with flowVS

Check cofactors

```
> cofactordata <- data.frame(markerstotransf, cofactors)
> head(cofactordata)
```

	markerstotransform	cofactors
1	FJComp-APC-A	9.21203
2	FJComp-APC-Fire 750-A	15939.97018
3	FJComp-APC-Fire 810-A	14227.99068
4	FJComp-APC-R700-A	323.02254
5	FJComp-BB515-A	1168.48848
6	FJComp-BB700-A	504.56595

Transform the original data

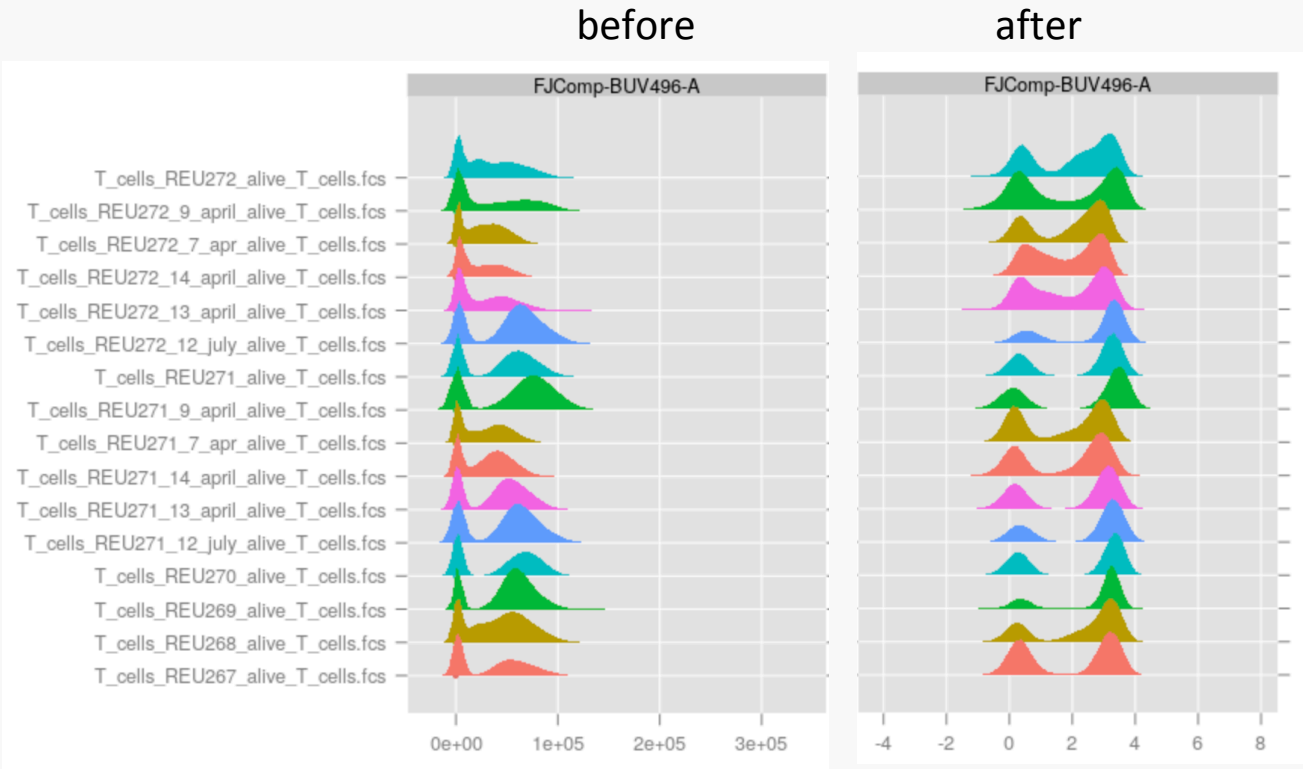
```
> fcs_transform <- transFlowVS(fcs_data,
                               channels = markerstotransf,
                                          cofactors)
```

FlowViz: Visualization for flow cytometry

https://bioconductor.org/packages/release/bioc/html/flowViz.html

Check transformation with density plots

```
> densityplot(~`FJComp-BUV496-A`, fcs_data) # before
> densityplot(~`FJComp-BUV496-A`, fcs_transform) # after
```



Alternative: Arcsinh transformation with fixed (supplied) cofactors

Create a vector of cofactors

```
> cofactor <- 3000
> l <- length(markerstotransf)
> cofactors <- rep(cofactor, l)
```

Transform

```
> fcs_transform <- transFlowVS(fcs_data,
                               channels = markerstotransf,
                                cofactors)
```

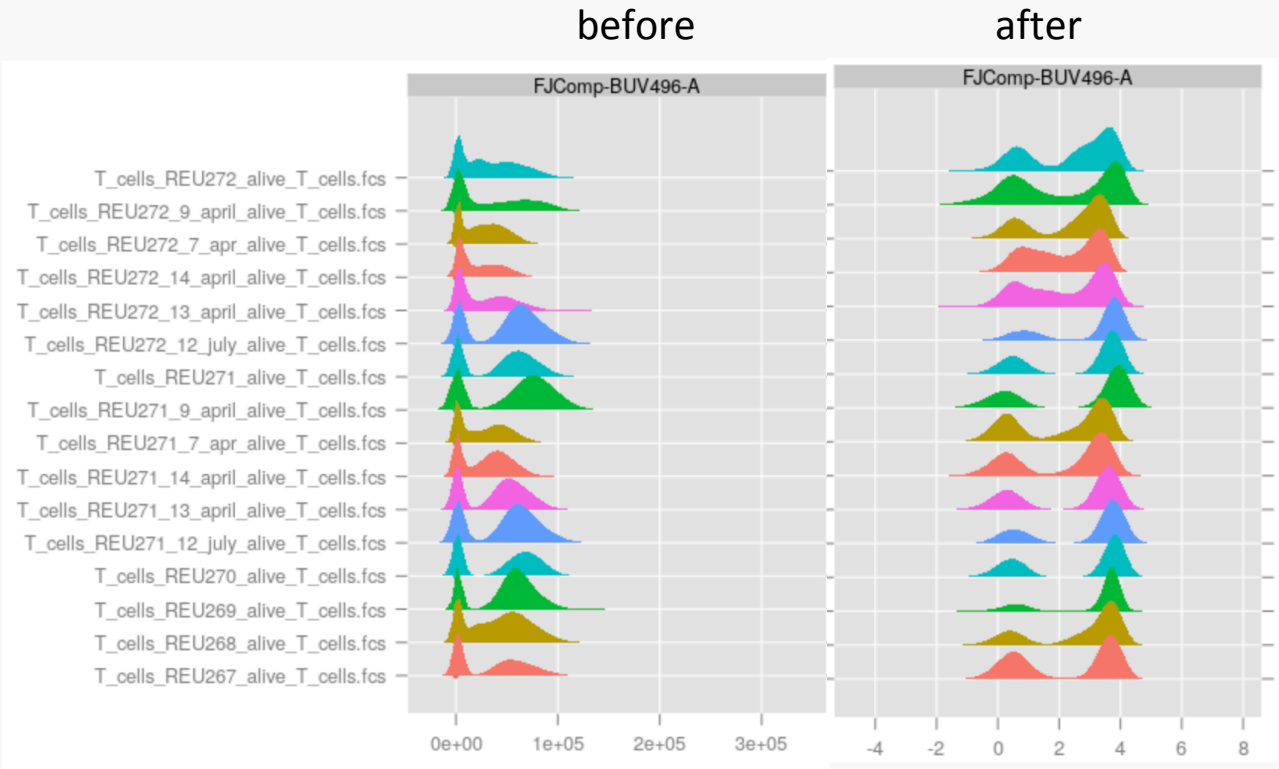
Check transformation with density plots

```
> densityplot(~`FJComp-BUV496-A`, fcs_data) # before
> densityplot(~`FJComp-BUV496-A`, fcs_transform) # after
```

Alternative: Arcsinh transformation with fixed (supplied) cofactors

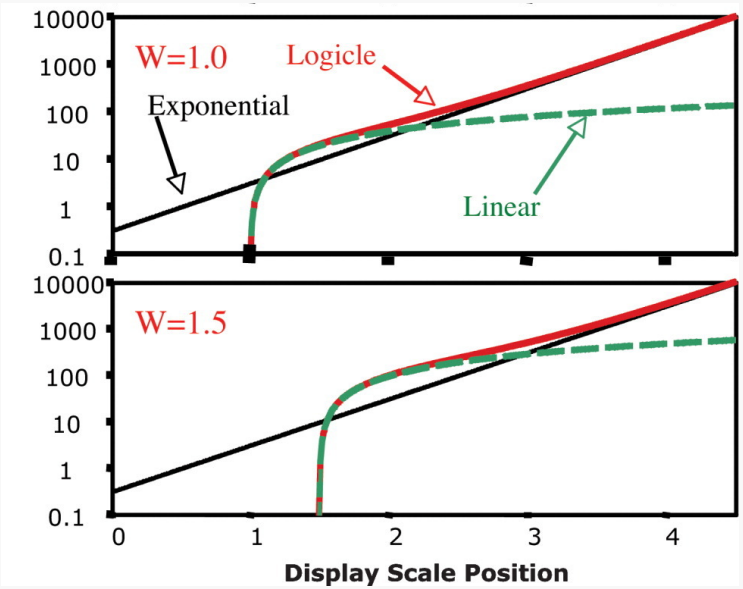
Check transformation with density plots

```
> densityplot(~`FJComp-BUV496-A`, fcs_data) # before
> densityplot(~`FJComp-BUV496-A`, fcs_transform) # after
```



Logicle transformation

$$B(x,T,W,M,A) = \begin{cases} T10^{-(M-W-A)}\left(10^{x-W-A} - p^2 10^{-\frac{x-W-A}{p}} + p^2 - 1\right), y \geq W + A \\ -T10^{-(M-W-A)}\left(10^{W+A-y} - p^2 10^{-\frac{W+A-y}{p}} + p^2 - 1\right), y < W + A \end{cases}$$



Parks et al. 2006c

W: Linearization width. Slope of transformation at zero. — estimated from the data

T: Top of the scale data (t = 4E6 in the case of spectral flow cytometry data)

M: With of the transformed data

A: Additional negative range to be included

} set by experimental circumstances

Logicle transformation with flowCore

Estimate parameters and transform

```
> fcs_list <- list()
> for(i in 1:16){
   ff <- fcs_data[[i]]
   algcl <- estimateLogicle(ff,
                            channels = markerstotransform,
                            m=6,
                            t = 4E6)
   fcs_list[[i]] <- transform(ff, algcl)
}
```

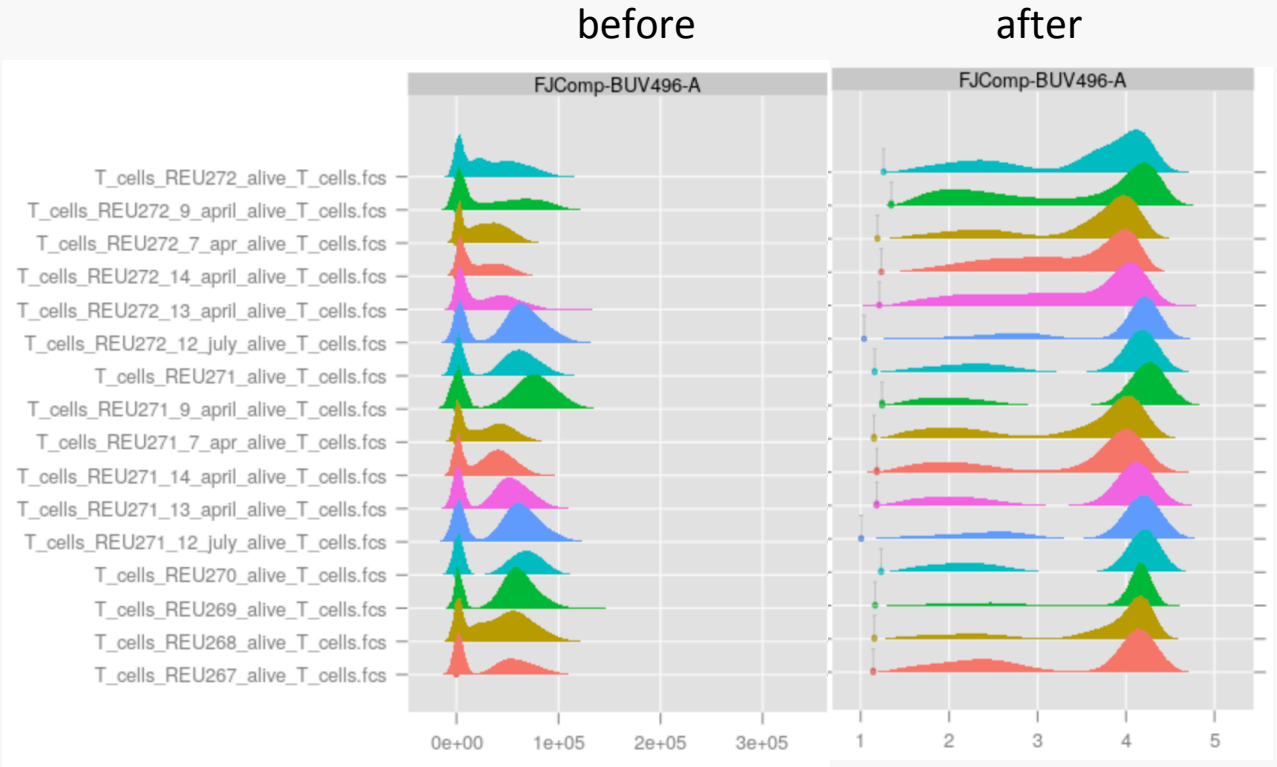
Recreate the flowSet from the list of flowFrames

```
>   names(fcs_list) <- sampleNames(fcs_data)
> fcs_transformed <- as(fcs_list, "flowSet")
```

Logicle transformation with flowCore

Check transformation with density plots

```
> densityplot(~`FJComp-BUV496-A`, fcs_data) # before
> densityplot(~`FJComp-BUV496-A`, fcs_transform) # after
```



Let's practice – 2

We will use the flowSet created in the previous exercise, and transform the data using two sets of cofactors: fixed and estimated using a function from the flowVS package.

Create a new script in which you will

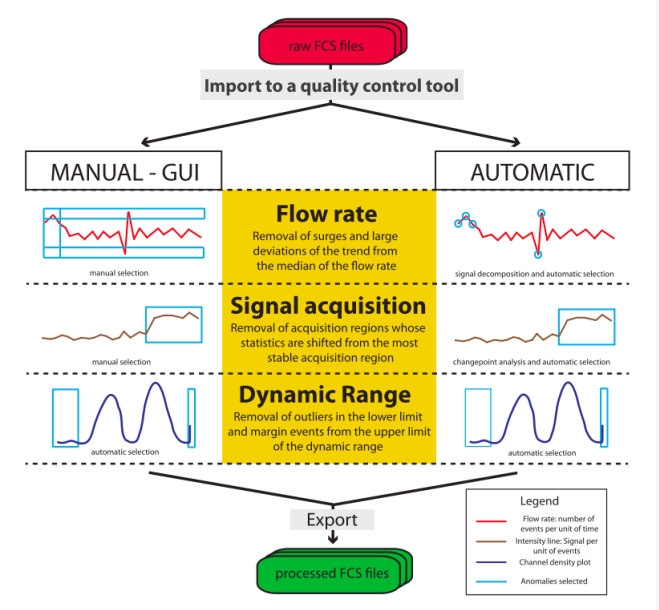
1) Load the flowSet object saved at the end of the previous exercise
2) Read the «course_datasets/FR_FCM_Z3WR/panel.csv» file into a data frame. The last column contains the marker classes («none», «type» or «state»)
3) Downsample the flowSet to 2'000 cells per flowFrame (you can find the downsampling function in the «course_datasets/function_for_downsampling_flowSets.R» file)
4) Transform the «type» and «state» markers using both Logicle (**hint**: use the downsampled flowSet for parameter estimation; start with default parameters, and adjust if needed) and arcsinh transformations (fixed cofactors of 3000).
5) Compare the transformation in the first flowFrame using density plots.

Automated Quality Control

Automated Quality Control with flowAI

https://bioconductor.org/packages/release/bioc/html/flowAI.html

- QC can be performed
  - automatically using flow_auto_qc()
  - interactively using flow_iQC()
- Evaluates three properties:
  - flow rate (FR)
  - signal acquisition (FS)
  - dynamic range (FM)
- Generates a report for each FCS file



Monaco et al., Bioinformatics (2016)

Run flowAI with default parameters

```
> fcs_QC <- flow_auto_qc(fcs_transform,
                         folder_results = “flowAI_results”)
```

Or, if pre-gated time gate, skip FR step

```
> fcs_QC <- flow_auto_qc(fcs_transform, remove_from = "FS_FM")
```

Select from which of the three steps the anomalies have to be excluded

Tabular report (Qcmini.txt)

Name file	n. of events	% anomalies	analysis from	% anomalies flow Rate	% anomalies Signal	% anomalies Margins
T_cells_REU267_alive_T_cells	265857	40.59	Flow Rate, Flow Signal and Flow Margin	39.9	0	1.16
T_cells_REU268_alive_T_cells	322400	1.81	Flow Rate, Flow Signal and Flow Margin	0	0	1.81
T_cells_REU269_alive_T_cells	304007	32.99	Flow Rate, Flow Signal and Flow Margin	31.97	0.33	1.51
T_cells_REU270_alive_T_cells	315735	41	Flow Rate, Flow Signal and Flow Margin	31.94	20.11	1.82
T_cells_REU271_12_july_alive_T_cells	80735	24.95	Flow Rate, Flow Signal and Flow Margin	23.61	0	1.76
T_cells_REU271_13_april_alive_T_cells	123141	33	Flow Rate, Flow Signal and Flow Margin	31.79	0	1.89
T_cells_REU271_14_april_alive_T_cells	107483	17.88	Flow Rate, Flow Signal and Flow Margin	16.14	0	2.08
T_cells_REU271_7_apr_alive_T_cells	204176	36.56	Flow Rate, Flow Signal and Flow Margin	35.55	0	1.59
T_cells_REU271_9_april_alive_T_cells	162794	26.63	Flow Rate, Flow Signal and Flow Margin	25.71	0	1.24
T_cells_REU271_alive_T_cells	207198	39.65	Flow Rate, Flow Signal and Flow Margin	25.95	24.61	1.38
T_cells_REU272_12_july_alive_T_cells	160439	9.72	Flow Rate, Flow Signal and Flow Margin	7.98	0	1.92
T_cells_REU272_13_april_alive_T_cells	171627	33.5	Flow Rate, Flow Signal and Flow Margin	31.82	0	2.54
T_cells_REU272_14_april_alive_T_cells	241584	31.59	Flow Rate, Flow Signal and Flow Margin	30.17	0	2.05
T_cells_REU272_7_apr_alive_T_cells	277724	39.36	Flow Rate, Flow Signal and Flow Margin	38.62	0	1.26
T_cells_REU272_9_april_alive_T_cells	258444	18.48	Flow Rate, Flow Signal and Flow Margin	16.75	0	2.12
T_cells_REU272_alive_T_cells	263909	19.39	Flow Rate, Flow Signal and Flow Margin	0	18.19	1.57

Example of a flowAI report

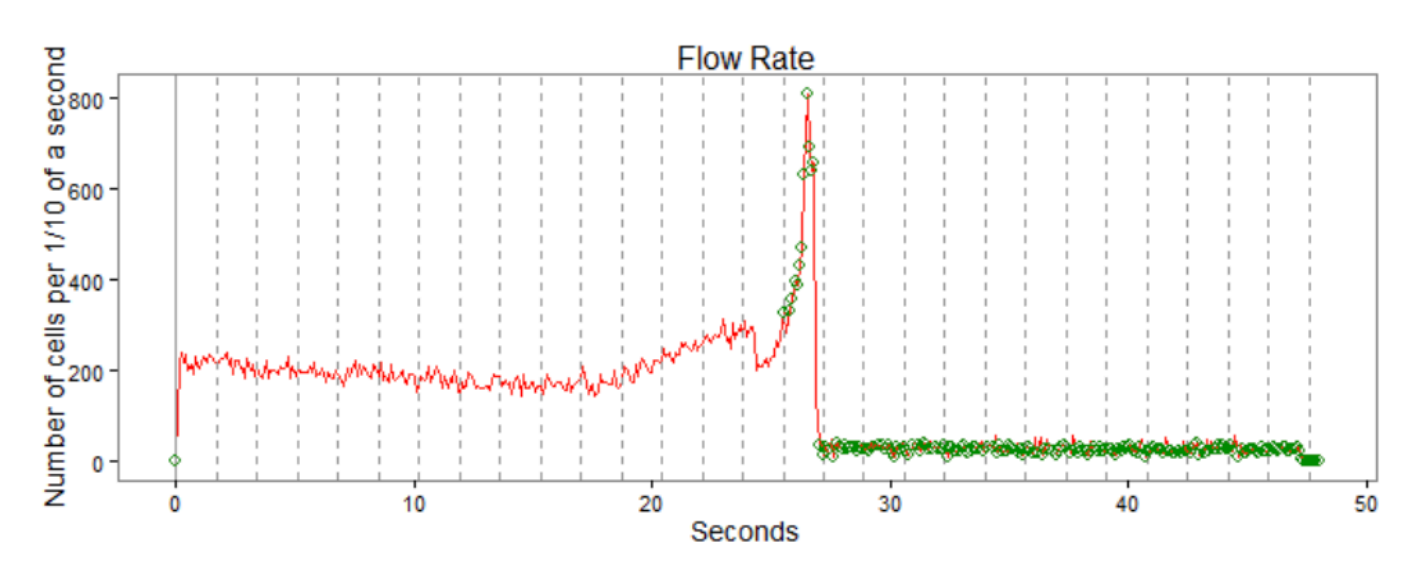
Quality control analysis

Summary

> The anomalies were removed from: Flow Signal and Flow Margin
> Anomalies detected in total: **0.11** %
> Number of high quality events: 157787

Example of a flowAI report

Flow rate check (FR)



https://bioconductor.org/packages/release/bioc/vignettes/flowAI/inst/doc/flowAI.html

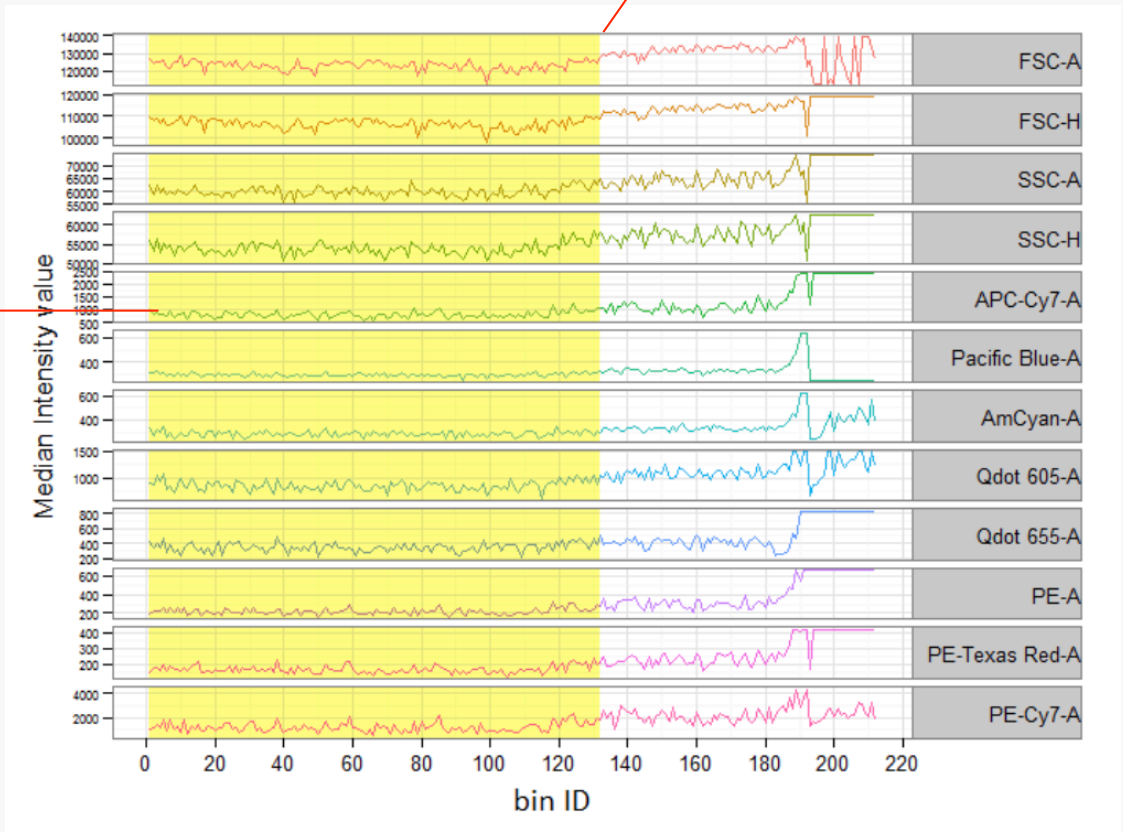
- This plot reconstructs the flow rate with a resolution of 1/10 of a second
- Anomalies are circled in green
- *alphaR*: the level of statistical significance used to accept anomalies (default value is 0.01). Decrease the value to make check less sensitive

Example of a flowAI report

Signals acquisition check (FS)

Detected change point

Median of the signal per bin of events



The mean and standard deviation of the medians should remain constant

https://bioconductor.org/packages/release/bioc/vignettes/flowAI/inst/doc/flowAI.html

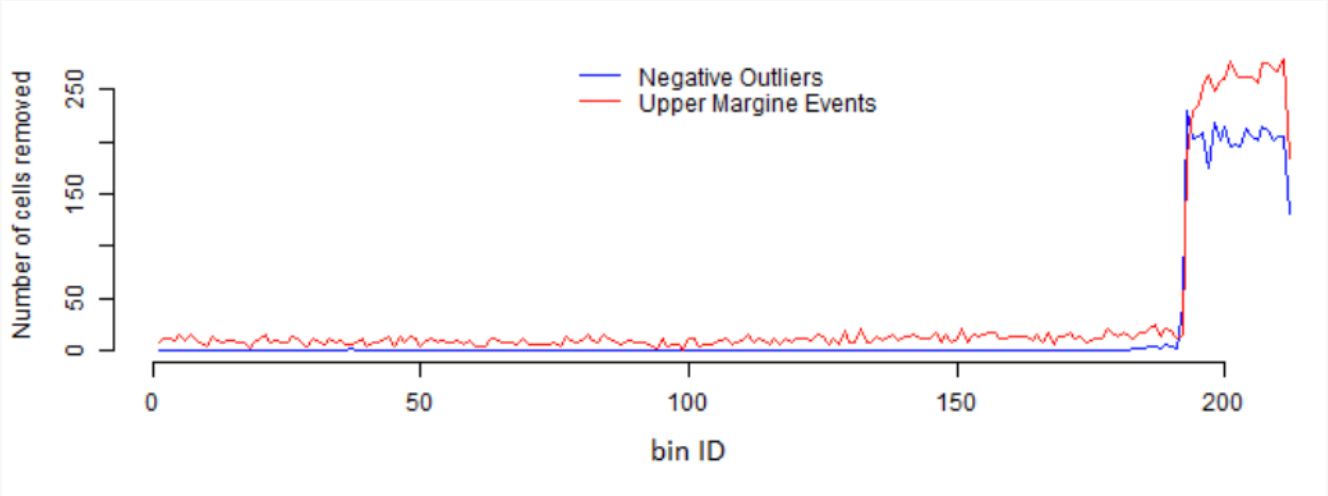
The region that passed the QC is highlighted in yellow

pen_values: Penalty for the changepoint detection algorithm (default is 500). The higher the penalty value the less strict is the detection of the anomalies.

Example of a flowAI report

Dynamic range check (FM)

Frequency of events removed



https://bioconductor.org/packages/release/bioc/vignettes/flowAI/inst/doc/flowAI.html

Same bins as in signal acquisition check

- Upper limit: the maximum value of the dynamic range (maximum pre-set by the manufacturer)
- Lower limit: values below zero for the scatter channels and all the outliers in the negative range for the immunofluorescence channels

Let’s practice – 3

We will continue with the Logicle transformed flowSet created in the last exercise, and apply the flowAI quality control algorithm to remove low quality cells.

Create a new script in which you will

1) Load the flowSet object from exercice 2 («/course_datasets/FR_FCM_Z3WR/fcs_transform_logicle.RData»)
2) Run the flowAI quality control algorithm. Set the output directory to «course_datasets/FR_FCM_Z3WR/flowAI_res»
3) Load the «Qcmini.txt» report created by flowAI and view it.
4) Check the html report for sample 1A9B20_0. What happened ?

Automatic Quality Control with PeacoQC

https://bioconductor.org/packages/release/bioc/html/PeacoQC.html

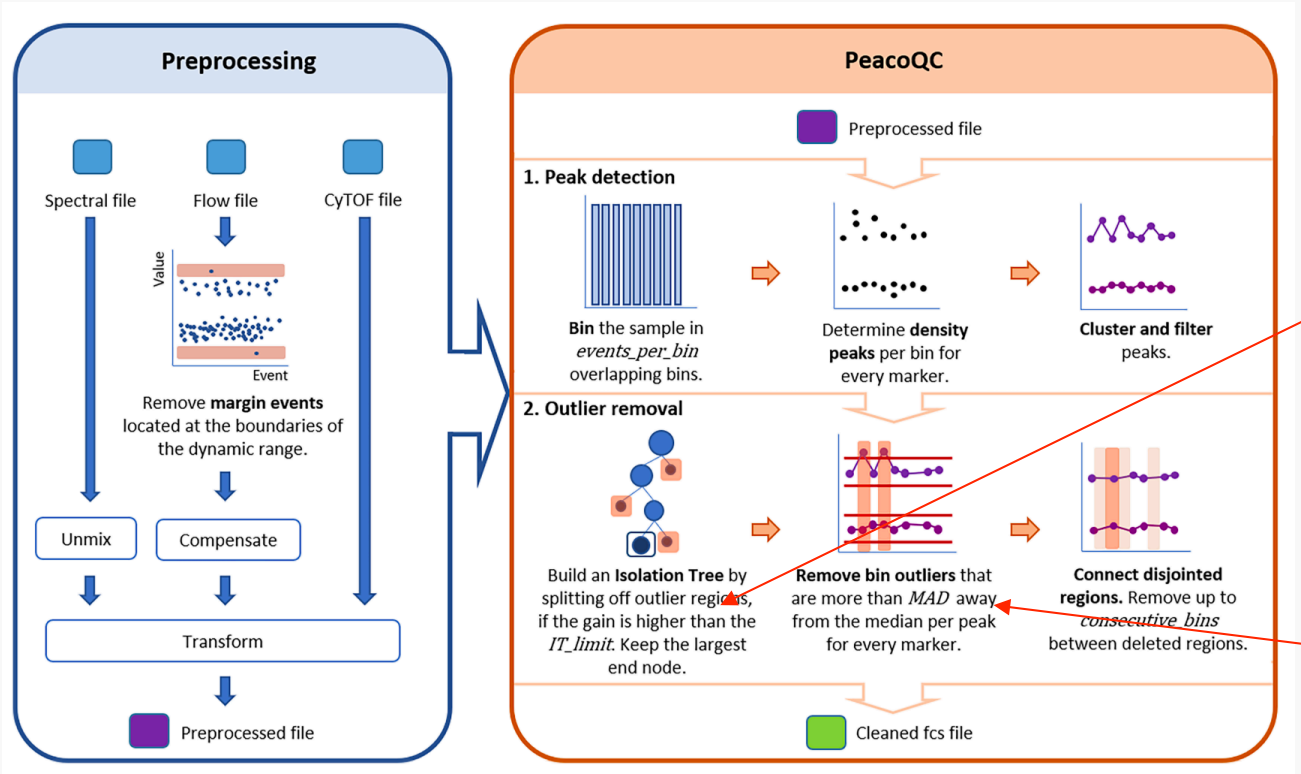
- **P**eak **E**xtraction **A**nd **C**leaning **O**riented **Q**uality **C**ontrol (**PeacoQC**),
- Tool for pre-processing (e.g. transformation) and quality control
- Removes outliers and unstable events introduced due to e.g. clogs, speed changes etc.
- Includes functions for visualising QC results

Run PeacoQC and save the cleaned flowframe as an fcs file

```
> peacoqc_res <- PeacoQC(ff=ff,
                         channels=channels,
                         determine_good_cells="all",
                         save_fcs=TRUE,
                         plot=TRUE,
                         output_directory = "PeacoQC")
```

The filtered *flowFrame* is stored in peacoqc_res$FinalFF and can be used for further analysis

Automatic Quality Control with PeacoQC



Gain limit of the isolation tree (**IT_limit**).
Lower values make the algorithm more strict

Median Absolute Deviations (**MAD**).
Lower values make the algorithm more strict

Two Steps:

- Peak detection: samples are binned and density peaks are determined for every marker and clustered
- Outlier removal: filter based on an Isolation Tree (**IT**) and remove peaks based on their **MAD** distance and connect disjointed regions

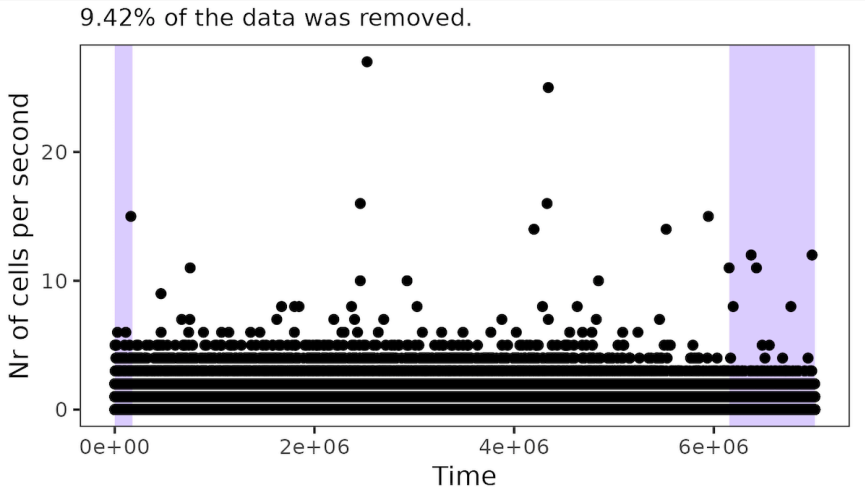
Automatic Quality Control with PeacoQC

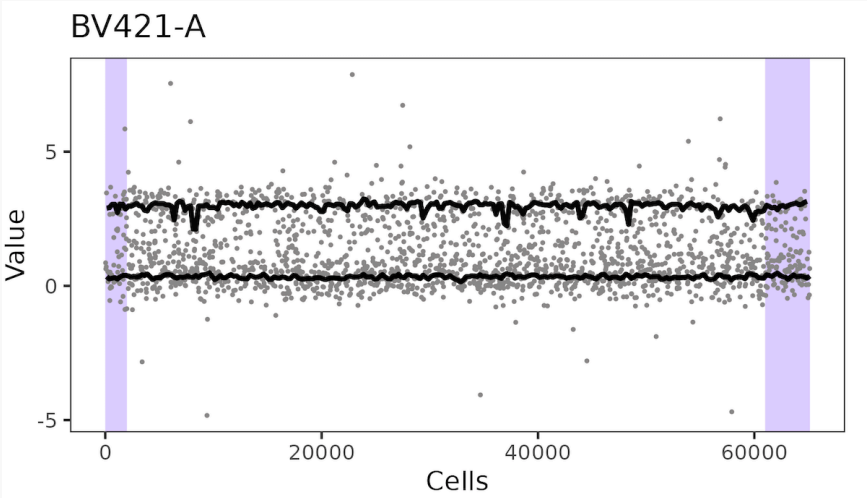
Run peacoQC first on one file to optimize the **parameters**

```
> peacoQC_res <- PeacoQC(ff= fcs_transform[[1]],
                         channels=markerstotransf,
                         determine_good_cells = "all",
                         save_fcs = FALSE,
                         plot=TRUE,
                         output_directory = "PeacoQCresults",
                         IT_limit = 0.65,
                         MAD=8)
```

Example of PeacoQC report and plots

Filename	Nr. Measurements before cleaning	Nr. Measurements after cleaning	% Full analysis	Analysis by	% IT analysis	% MAD analysis	% Consecutive cells	MAD	IT limit	Consecutive bins	Events per bin	Increasing/Decreasing channel
TF_ 062CD8.fcs	232632	230500	0.91646893	all	0	0.91646893	0	5	0.55	5	1000	No increasing or decreasing effect
TF_ 0B943B_0.fcs	65136	59000	9.42028986	all	0	9.42028986	0	5	0.55	5	500	No increasing or decreasing effect
TF_ 0B943B_14.fcs	142496	132000	7.36582079	all	0	7.36582079	0	5	0.55	5	1000	No increasing or decreasing effect
TF_ 0B943B_3.fcs	51600	48250	6.49224806	all	0	6.49224806	0	5	0.55	5	500	No increasing or decreasing effect
TF_ 22CBD6.fcs	286160	264000	7.74391949	all	0	7.74391949	0	5	0.55	5	1500	No increasing or decreasing effect
TF_ 24305F_1.fcs	138144	135000	2.27588603	all	0	2.27588603	0	5	0.55	5	1000	No increasing or decreasing effect
TF_ 2AD75E_14.fcs	108416	107750	0.61430047	all	0	0.61430047	0	5	0.55	5	500	No increasing or decreasing effect
TF_ 2AD75E_7.fcs	168392	155000	7.95287187	all	0	7.06209321	1.4846311	5	0.55	5	1000	No increasing or decreasing effect
TF_ 36EA16.fcs	91896	84750	7.77618177	all	0	7.77618177	0	5	0.55	5	500	No increasing or decreasing effect





Automatic Quality Control with PeacoQC

After choosing the right parameters, apply to all samples

```
> for(i in 1:16){
  peacoqc_res <- PeacoQC(fcs_transform[[i]],
                         markerstotransf,
                         determine_good_cells = "all",
                         IT_limit=0.55,
                         MAD=5,
                         save_fcs = TRUE,
                         plot=TRUE,
                         output_directory = "PeacoQCresults")
}
```

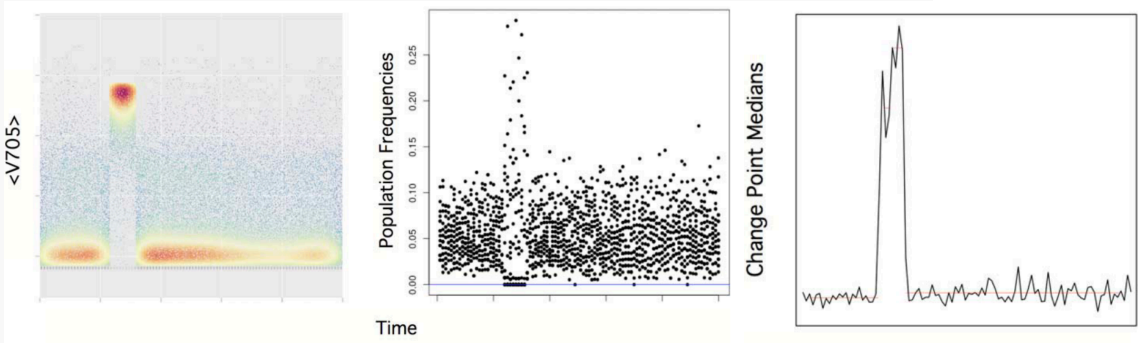
Construct new flowSet from the cleaned fcs files

```
> fcs_clean <- read.flowSet(path= "PeacoQCresults/fcs_file",
                            transformation=FALSE,
                            truncate_max_range = FALSE)
```

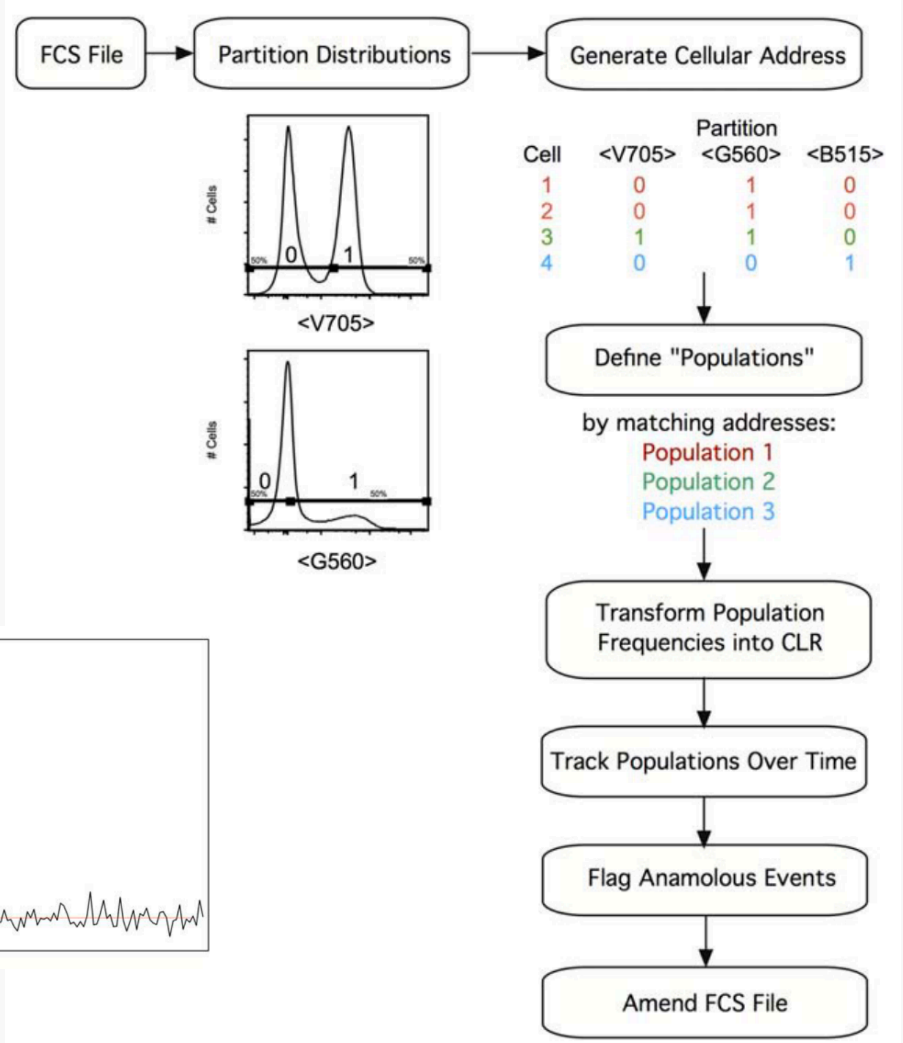
Automated Quality Control with flowClean

https://www.bioconductor.org/packages/release/bioc/html/flowClean.html

- Groups cells into populations, which are tracked over the acquisition period
- Changepoint analysis: detects significant changes in frequencies
- Classifies cells (wether collected in a “good” or “bad” time interval)



Fletez-Brant et al., Cytometry Part A (2016)



Fletez-Brant et al., Cytometry Part A (2016)

Automated Quality Control with flowClean

Run flowClean algorithm

```
> fcs_list <- list()
> marker_indexes <- match(markerstotransf,colnames(fcs_transform))
> for(i in 1:16){

   fcs_list[[i]] <- clean(fF = fcs_transform[[i]] ,
                          vectMarkers = marker_indexes)


   }
```

Construct new flowSet

```
> names(fcs_list) <- sampleNames(fcs_transform)
> fcs_QC <- as(fcs_list, "flowSet")
```

The result is an flowSet identical to the input flowSet, but with a new parameter 'GoodVsBad', which can be used to select the quality cells ("good" if < 10000, "bad" otherwise)