



R-based **A**nalytical tool for **D**ifferential **A**Nalysis of cy**T**ometry data

User manual
31/01/2024

Authors:
[Alper Çevirgel](#)
[Martijn Vos](#)
[Anne Floor Holtrop](#)

Radiant v1.0
Manual version v.1.0

Contents

1. Introduction.....	2
2. Preparation of the data and working environment.....	4
2.1 Prepare the data in FlowJo	4
2.2 Starting up the R environment	5
2.3 Manually create a folder structure.	5
3. Analysis.....	6
3.1 Data Preprocessing.....	6
<i>Chunk 1: setting environment and folder structure.....</i>	<i>6</i>
<i>Chunk 2: loading data into the working environment.....</i>	<i>7</i>
<i>Chunk 3: cleaning and transformation of the data.....</i>	<i>9</i>
<i>Chunk 4: validation panel file and matching internal and external fcs file names.....</i>	<i>11</i>
<i>Chunk 5: validation metadata and generating diagnostics.....</i>	<i>11</i>
<i>Chunk 6: preparing files and directions for cross-entropy test.....</i>	<i>13</i>
<i>Chunk 7: preparing files and directions for cross-entropy test.....</i>	<i>13</i>
<i>Chunk 8: check CSV values for CytoNorm.....</i>	<i>14</i>
<i>Chunk 9: run CytoNorm.....</i>	<i>14</i>
<i>Chunk 10: run the cross entropy test after normalizing.....</i>	<i>15</i>
3.2 Clustering analysis.....	16
<i>Chunk 1: setting environment and folder structure.....</i>	<i>16</i>
<i>Chunk 2: loading data into the working environment.....</i>	<i>17</i>
<i>Chunk 3: downsampling, clustering and dimension reduction</i>	<i>18</i>
<i>Chunk 4: determination requirement of cluster merging.....</i>	<i>18</i>
<i>Chunk 5: cluster merging.....</i>	<i>19</i>
<i>Chunk 6: evaluation cluster merging.....</i>	<i>21</i>
3.3 Downstream analysis.....	21
<i>Chunk 1: setting environment and folder structure.....</i>	<i>21</i>
<i>Chunk 2: create folder and subset the data if required.....</i>	<i>22</i>
<i>Chunk 3: statistical analysis.....</i>	<i>23</i>

1. Introduction.

Over the last decade, the number of optical channels in flow cytometers has been increasing rapidly allowing for more in-depth analysis. However, this advancement also

renders data analysis more time-consuming and complex. Traditional 2-dimensional gating might overlook certain marker combinations. While paid online tools like Cytobank and OMIQ are available, they offer limited flexibility in their analysis and are costly. Alternatively, publicly available R scripts often lack user-friendliness for those unfamiliar with R. Here we present the RADIANT pipeline which we constructed by combining the latest tools available such as PeacoQC, FlowSOM, CytoNorm, CATALYST, and elements from other packages. Our pipeline includes built-in checks, troubleshooting, and a user manual to make it easier for the users.

The pipeline is structured into three sections: pre-processing, clustering and downstream analysis. It is advised to meticulously adhere to the on-screen instructions as well as the guidance presented within this manual to ensure a smooth workflow. To enhance user comprehension and facilitate progress monitoring, we have incorporated multiple quality checks at various junctures. These checkpoints serve to illuminate the ongoing processes and allow for meticulous examination after each step.

Pipeline Overview:

Pre-processing: **1_Data_Preprocessing_usr_RADIANT_v1.0.rmd**

- preparation of the analysis folders
- data import, data cleaning
- CytoNorm

FlowSOM clustering: **2_FlowSOM_clustering_RADIANT.V1.0_usr.rmd**

- Determining number of Metacluster
- FlowSOM
- Cluster Merging

Downstream analysis: **3_Downstream_analysis_RADIANT.V1.0_usr.Rmd**

- Setting up work environment for statistical analysis

You will need **RADIANT_dependencies_usr_v1.0.R** for all the functions the pipeline is running.

2. Preparation of the data and working environment.

It is important to remember that the R language treats mathematical signs as operations. Therefore, it is extremely important not to use +, -, /, = etc. in your variables, folder names or files to prevent potential problems. This applies to fcs files and antigens you measure in your experiment. Instead, you can use “_” where needed. Therefore, it is important to make it a habit to not include them in file names.

Let's say you cannot reexport your data and you have antigen names including these symbols. Don't worry, there is a fix for that but you need to do some work. (explained below at the end of Chunk 2)

Name of	Examples that could create problems	Use instead
Fcs files	sample-20231222.fcs CD4+-patient1-CTLA-4+.fcs	sample_20231222.fcs CD4p_patient1_CTLA4p.fcs
Antigens	HLA-DR, PD-1	HLADR, HLA_DR, PD1, PD_1
Folders	CD4+ T cells	CD4_T_cells, CD4pos_T_cells

2.1 Prepare the data in FlowJo

- 1) Begin by opening your experiment containing all relevant fcs files in FlowJo. Conduct pre-gating procedures as usual to eliminate background noise.
 - This involves gating out debris and dead cells, retaining only single cells, confirming the absence of compensation or spill-over issues, and reviewing NxN plots to assess marker expressions.
- 2) Export the freshly gated cells, ensuring retention of only uncompensated parameters, with the inclusion of FCS-A and SSC-A.
- 3) When exporting, append a consistent suffix (e.g., "_uncomp") to unify file names for easier selection in R.
- 4) Additionally, incorporate 'TUBE NAME' or '\$FIL' parameters in the file name, simplifying the process and avoiding the need for a custom parameter in the pipeline.
 - Export all the compensation matrices you have applied to the data **as .csv (not .mtx!)**. It happens often that FlowJo does not remove the compensation matrix, even if you selected to export uncompensated parameters. Therefore, re-import your exported files into FlowJo and check if the compensation matrix is present. If so, remove it completely (click the little “-” sign in the bottom left of the compensation screen and re-export the uncompensated parameters.
 - For *spectral cytometers* instead of compensation matrix, unmixing is used. However, often small tweaks in the compensation matrix could be needed after unmixing. In this case, you can export your data (compensated) and use an empty compensation matrix for the pipeline. Therefore, RADIANT will apply an empty compensation matrix to your data, since you already have the clean & compensated fcs files.

- 5) In order to properly transform the data, a reference file is needed (.fcs). You should have 1 reference file per batch. If you are working with multiple types of tissue you need 1 reference file per tissue per batch.
 - A reference file can either be a technical control (same donor taken along every batch), or a concatenated .fcs file of cells from all samples in a batch if you don't have a technical control. The reference file should be uncompensated as well.

2.2 Starting up the R environment

To optimize the pipeline for user-friendliness, certain compromises were introduced in terms of adaptability. While adept users can introduce modifications in the dependencies file, this manual is tailored for individuals with limited R experience and therefore adheres to specific limitations. The pipeline possesses the capability to carry out numerous tasks automatically, but specific prerequisites must be met for this automation to occur. Therefore, it is important to meticulously adhere to the subsequent steps. To execute the pipeline, you have two options: you can either operate in R (version 4.x) with RStudio on your local device but we highly recommend using R servers.

2.3 Manually create a folder structure.

- 1) Create a folder where all the files will be located by the pipeline. In this example, we will call it "Analysis", but "RADIANT Analysis", "Study Name" or any other name is sufficient!
- 2) Inside this folder, make a subfolder for each different (subset of a) dataset you are going to work with, for example CD8_Tcells and CD4_Tcells. The name of this folder will be the `cellType` variable in the beginning of the pipeline.
- 3) Inside the CD8_Tcells folder copy the complete folder ("00 RADIANT files") from here: "S:\O&O-VO\DATA\RADIANT pipeline", don't edit the ones in the source folder (in case you accidentally break something).

Your folder structure should now be similar to : `~\Analysis\cellType\00 RADIANT files`. Don't start uploading any files just yet!

3. Analysis.

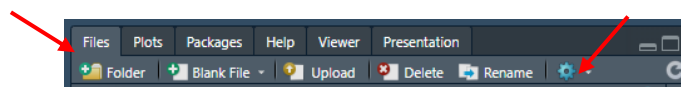
3.1 Data Preprocessing.

Chunk 1: setting environment and folder structure.

DURATION: if you've run the installation process before, this chunk takes only moments. Run it **twice** to easily check if every package is loaded correctly. If you've not run this before, this chunk can take up to several hours.

- 1) Open the "1_Data_Preprocessing_usr_RADIANT_v1.0.rmd" file in RStudio, make sure to use the copied version on your own device/server to avoid altering the original source.
- 2) In the first chunk of the pipeline set the working directory of your primary analysis folder in the "wd" variable (see chapter 2.1 step 1). For example: `wd = "~/Analysis"`.

To find the location, click on files in the right bottom panel on Files, move to the right folder, then click on the blue settings button and find the "Copy Folder Path to Clipboard" option. This will copy your folder path which you can now past into your "wd" variable. Make sure to keep it between "quotation marks".



- 3) Enter the name of the subfolder where your data is located in the `cellType` variable. For example: `cellType = "CD8"`.
- 4) You can also provide an additional identifier in the `expID` variable. This can, for example, be the experiment, or study ID. For example: `expID = "FFX"`.
- 5) In the `file_pattern` variable fill in the shared suffix (can even be a single letter, or number) of your FCS files. For example: `file_pattern = "uncomp.fcs"`. Your files all should have this extension, which you decided during exportation of the files. This way we can select fcs files from other types of files in folders during the analysis steps.
- 6) For the `ds` variable, please put whether or not you're working with (conventional) flow cytometry (`ds = "FC"` data, or spectral flow cytometry (`ds = "SFC"`) data, as for cleaning with PeacoQC later on we have to treat spectral flow cytometry a little bit different.
- 7) Next, hit ctrl + shift + enter to run the chunk.

Installing and loading the packages

R will automatically check if there are any packages that need to be installed in order for the pipeline to run properly. If all packages are properly installed, they are loaded in automatically. If packages are missing, you will be asked if the packages should be installed automatically. Open the console on the bottom left of the screen and type either "y", or "n". Then click enter. If you don't want the packages to be installed automatically, you'll have to do it manually. Be aware that the installation may take quite a long time, depending on how many packages need to be installed. If you run this chunk for the very first time, it can take several hours.

A common error we encountered when using R servers is something like the image below. In this case go to your R library (for our servers, example path: N:\data\R_library) and delete the

00LOCK (followed by the name of the package that gives the error) folder. Then rerun the chunk. This error can occur multiple times, but just keep deleting the 00LOCK folder and rerunning the chunk. If you cannot delete the folder, you can move it to another folder somewhere else (you can create a “trash” folder for that, somehow moving can sometimes be allowed followed by deleting it, but not directly in that folder).

```
Error in unzip(zipname, exdir = dest) :  
cannot open file 'N:/data/R_library/file2a2064061f60/colospace/hclwizard/www/images/pal_red-purple.png': Invalid argument  
In addition: warning messages:  
1: package(s) not installed when version(s) same as current; use `force = TRUE` to re-install: 'BiocStyle'  
2: In file.copy(savedcopy, lib, recursive = TRUE) :  
problem copying N:/data/R_library/00LOCK/cli\libs\x64\cli.dll to N:/data/R_library/cli\libs\x64\cli.dll: Permission denied
```

Uploading files if required

If all packages are installed and loaded, it checks whether or not there already was a folder structure, and whether or not they were filled. If they were filled it gives that as an output, and suggests to change a variable in the next chunk (called `to_import`) to change to either a, b or c, see the note on screen. If no files were found, please upload all the required files.

- 1) A new folder has been created in the cellType folder: 01 FCS files, within this folder you can place your FCS files in the 00 Raw files folder.
 - a. There will be a folder “01 FCS files”, in the subfolder “00 Raw files” copy all your fcs files you exported previously here. Make sure these are the raw files that are NOT compensated and NOT transformed!
 - a. If you’re working on the server, make sure to save all files in a ZIP file to upload them in one go.
- 2) There will also be a folder called “Reference files” in the cellType folder. Inside this folder there are two subfolders called “Compensation matrices” and “FCS files”. Put your compensation matrix, or matrices in the “Compensation matrices” folder, and your Reference file(s) in the “FCS files” folder.

Chunk 2: loading data into the working environment.

- 1) As described before, change the variable `to_import` to the corresponding files you want to load. If it is your first run, keep at “a” (which results in importing raw fcs files).
- 2) Run the chunk by hitting ctrl + shift + enter.

You will see a table popping up as an output for you to see what has been detected in your files. The name column are the names of the **fluorophores** and the desc column are **the antibodies you coupled to them**. It’s a check for you to see what the pipeline has found in the applied data. To see all click on NEXT on the right bottom of the output.

If you want to change your antigen names you could try the code below:

```
pData(parameters(fs[[1]]))[,1:2]$desc #This will give you the list of antigens you currently have
```

In my case row 17 was HLA-DR and row 22 was PD-1, then I can use these code below to re-write HLA-DR into HLADR, PD-1 into PD

```
for (i in 1:length(fs)) {  
  pData(parameters(fs[[i]]))$desc[17] <- "HLADR"  
  pData(parameters(fs[[i]]))$desc[22] <- "PD1" }
```

Then rerun `pData(parameters(fs[[1]]))[,1:2]$desc` run to see if the changes worked.

Chunk 3: cleaning and transformation of the data.

1. Set `writePanel` to **TRUE** if you want to automatically write a panel file. This is an excel file where you can indicate how the pipeline should treat each marker (type/state)(see below, Uploading the panel file). If you already have a panel file from a previous run you should set this to **FALSE**, as the existing file will be overwritten otherwise
2. If you're working with multiple compensation matrices and/or multiple reference files you need to map these to the correct sample files. The pipeline will automatically write the required excel files for you to fill in if you set `write_files` to **TRUE**. After you've filled in and saved these files (and re-uploaded them to the server) you can set `write_files` to **FALSE** and re-run the chunk. If you already have mapping files from a previous run you can also set `write_files` to **FALSE**. If you leave it on **TRUE**, pre-existing files will be overwritten by empty ones.
3. Indicate if you want to perform cleaning and/or transformation in the `cleaningMethod` and `transformationMethod` variables. For `cleaningMethod` the options are "PeacoQC", or "None". For `transformationMethod` options are "Logicle", or "None". For the transformation you should only set it to "None" if you're working with pre-transformed data, e.g. from a previous run. In order to perform clustering, the data needs to be transformed, so you should NOT continue with raw data. If `to_import` in the previous chunk was "c" (i.e. cleaned and transformed files), put these to "None".
4. FCS files have an internal file name parameter, as well as an external one (the one you gave it when exporting from FlowJo). These don't always match. Kind of like a word document called "Document A" containing the text "File name: Document B". R only has a list of fcs files and a list of the external file names in the FCS files folder. We want to make sure the external file name is matched to the correct fcs file, so that we can overwrite the internal file name with the external one. To do this R looks at two internal parameters ('TUBE NAME', '\$TBNM' and '\$FIL') and tries to find these parameters in the external file name. If you used either of these parameters when exporting the fcs files from FlowJo the pipeline will find a match and automatically overwrite the internal file name with the external one. If you didn't you can add a parameter you used to export the fcs files to `customFileNameVar`, e.g. `customFileNameVar = "$SRC"`, otherwise leave it at **NA**.
5. Next you need to provide some information about your compensation matrices and reference files. Set the `multComp` variable to **TRUE** if you have multiple compensation matrices (**FALSE** if not). Set the `multRef` variable to **TRUE** if you have multiple reference files (**FALSE** if not).
6. Now you need to specify which channels you want to use for data cleaning and transformation. This list should also contain the markers you want to run clustering analysis on. You can include more markers than you want to run analysis on, but not less. Also, it is recommended to include "FSC-A" and "SSC-A". Fill in the desired markers in the `markers_of_interest` variable.
7. For more information on the `MADValue` and `ITLimitValue` you can read the paper by Emmaneel *et al.* 2021 ([PeacoQC: Peak-based selection of high quality cytometry data - Emmaneel - - Cytometry Part A - Wiley Online Library](#)). The defaults are 6 for the `MADValue` and 0.55 for the `ITLimitValue`.

For the cleaning part the data is sectioned in bins. In the end, bins that contain noise are removed. Therefore, a higher number of bins increases resolution and may result in less data loss, but could be less accurate if the total number of events is low. The

PeacoQC algorithm itself determines how many bins are used, based on a user defined number of events per bin (default is 150). In the RADIANT pipeline we allow users to select a minimum (`minBin`) and maximum number of bins (`maxBin`), as well as an adjustable default number of cells per bin (`dEvtsPerBin`). If the total number of cells exceeds the maximum user defined number of bins that are allowed to be filled with the user defined number of cells per bin, the algorithm will automatically increase the number of cells per bin, so as not to exceed the user defined maximum number of bins.

For example: if the total number of cells is 10.000, the user defined maximum number of bins is 100 and the user defined number of cells per bin is 50, the algorithm will automatically increase the number of cells per bin to 100, so as to maintain 100 bins. This will be assessed per sample. On the other hand, if there are not enough total cells to fill the user defined minimum number of bins with the user defined number of cells per bin, the number of cells per bin is automatically reduced for all samples, so that for all samples at least the minimum number of bins is filled. For some samples that will then mean the maximum number of bins is exceeded, which will result in the algorithm automatically increasing the number of cells per bin per sample.

Please take into account that the more bins you have, the longer it takes to clean the data! It is advisable to leave these variables on their default value, and only later start optimizing them together with someone who understands it at a deeper level.

8. PeacoQC is capable of automatically writing plots to show where it removed data. Writing these plots can take quite some time, especially if there is a lot of data. We therefore made it possible to turn this feature off completely, or only let PeacoQC write plots if more than a certain percentage of data gets removed. If you want to write plots for all the files set `plotPQC` to **TRUE**, if you do not want to write plots at all set to **FALSE**, if you want to only write plots if more than a specific percentage gets removed, set to any integer between 1 and 100.

9. Hit ctrl + shift + enter to run the chunk

After running PeacoQC: After the cleaning in PeacoQC folder you can find the results of the cleaning. Excel sheet will summarize % of cells removed and total number of cells before and after cleaning per each sample. If a very high percentage of cells are removed from the analysis, please check time plots of your samples to see if there was something wrong during the acquisition of samples.

Uploading the panel file:

As you'll see, a panel file will have automatically written to your cellType folder if you put writePanel to TRUE. The excel file consists of 4 columns: "index_val", "fcs_colname", "antigen" and "marker_class". The "fcs_colname" column denotes the channel, as specified in the FCS file and the "antigen" column shows the marker, as specified by the user in DIVA. You only need to edit the "marker_class" column. In this column you need to specify what kind of marker each channel contains. There are three options: "none", "type", or "state". You should fill in "none" for all parameters that should not be part of the analysis. These include at least FSC, SSC, Time and Original_ID if you've run PeacoQC.

index_val	fcs_colname	antigen	marker_class
1	FSC-A	FSC-A	none
2	FSC-H	FSC-H	none
3	FSC-W	FSC-W	none
4	SSC-A	SSC-A	none
5	SSC-H	SSC-H	none
6	SSC-W	SSC-W	none
7	APC-A	CD57	type
8	APC-H7-A	Live_Dead	none
9	APC-R700-A	CD3	none
10	BUV396-A	CD8	none
11	BUV661-A	CD4	none
12	BUV737-A	CD27	type
13	BV421-A	PD1	state
14	BV510-A	CD38	state
15	BV605-A	TIGIT	state
16	BV650-A	CCR6	type
17	BV711-A	CXCR3	type
18	BV786-A	CD45RO	type
19	FITC-A	CXCR5	type
20	PE-A	CD95	type
21	PE-CF594-A	CCR7	type
22	PE-Cy7-A	CCR4	type
23	PerCP-Cy5-5-A	CD28	type
24	Time	Time	none
25	Original_ID	Original_ID	none

In our example we are working with CD8+ T-cells, which were pre-gated in FlowJo based on live/dead-, CD3+, CD8+ and CD4-, so we will set these channels to "none" as well, as they do not play a role in differentiating subsets anymore. For the markers that are left we have to decide if they tell something about the type, or the state of the cell. Type markers will be used in flowSOM clustering to identify distinct subsets whereas state markers will not be included in the clustering but their MFI will be reported for each cluster.

After clustering, "state" markers will be used show differential expression in clusters. Which markers exactly qualify for either "type", or "state", is up to the user to decide. Do not leave FALSE in any of the cells. You have to fill in either "none", "state", or "type".

Once you've filled in the file, don't forget to save it (and re-upload it to the server if required).

Chunk 4: validation panel file and matching internal and external fcs file names.

- 1) In order to annotate the data after clustering a metadata file needs to be provided. The pipeline will automatically write one. In order to do so, set writeMeta to **TRUE**. If you already have a metadata file, you should set this to FALSE, otherwise the existing file will be overwritten.
- 2) Hit ctrl + shift + enter to run the chunk

Uploading the metadata file

A new metadata file will have been written to your cellType folder. There are 5 columns: "file_name", "sample_id", "patient_id", "sample_control" and "batch". These columns **CANNOT** be deleted. The "sample_id" column should contain a unique entry for every sample. When you have technical controls in your dataset please indicate these by entering "control" in the "sample_control" column. Normal samples can be labeled as "sample". You can add as many columns as you like with additional information (e.g. condition, timepoint, gender, age, etc.).

Once you've filled in the file, don't forget to save it. Be aware to not leave any FALSE in it, and please label your batches with number (i.e., 1, 2, 3 rather than for example batch 1, batch 2, batch 3)

Chunk 5: validation metadata and generating diagnostics.

- 1) The pipeline allows for automatic writing of plots to pdf files. To enable this, set `writeFiles` to `TRUE`.
- 2) If you have technical controls included in your dataset, set `TCS` to `TRUE`. This will exclude these files from clustering, as they can skew the results massively.
- 3) Part of the diagnostics includes analyzing the non-redundancy score and multi-dimensional scaling. You can set a variable from your metadata to color the plots by in `NRSColorBy` and `MDSColorBy`.
- 4) Lastly, the pipeline will generate a heatmap showing you the expression of every marker of every sample. This heatmap will be clustered using both row and column dendrograms. You can select variables from your metadata to annotate the rows in `exprHMRowAnno`.
- 5) Hit ctrl + shift + enter to run the chunk

IMPORTANT: you've now created diagnostic plots that can be found in the result folder, or you did this in a previous run. You can check each file. Certain color and groupings are used based on the `NRSColorBy`, `MDSColorBy` and `exprHMRowANNO` you specified at the beginning of this chunk. If you've never run this chunk before, the default settings are in such a way to have the highest probability of semi-automatically detecting batch effects.

REDO this chunk until the groupings make sense, and continue to chunk 6.

IMPORTANT: if you don't have technical controls, and/or want to run normalization with `cytonorm`, you can skip the rest of this document and move on to clustering analysis, otherwise, continue the last chunks.

Chunk 6: preparing files and directions for cross entropy test.

This segment is founded on a pipeline outlined in the publication available at [https://www.cell.com/cell-reports-methods/pdfExtended/S2667-2375\(22\)00295-8](https://www.cell.com/cell-reports-methods/pdfExtended/S2667-2375(22)00295-8).

Essentially, the statistical test is designed to evaluate the similarity or dissimilarity among all cells originating from different batches or within technical controls, specifically within a high-dimensional space in this context. In practical terms, one anticipates non-significant results for technical controls, and in the case of a randomized study, no discernible differences between batches are expected. Additionally, within the third RMD file of this pipeline, this test can be applied to compare various groups based on the columns of your metadata file. In essence this code copies the files of interest (either all or a subset) based on a column of the metadata file, renames them by giving a prefix (e.g., all healthy cells get healthy_ as prefix, and all diseased cells diseased_), groups all cells of the different prefixes together and runs the cross entropy test to determine whether or not the groups differ in the high-dimensional space.

- 1) If you haven't executed this code previously, set `crossEntropyTest` to **TRUE**. Evaluate whether you wish to reduce the number of markers (refer to the note displayed on the screen after running the code).
- 2) If you previously specified `SSC-A` and/or `FSC-A` in the `markers_of_interest` variable, these will be automatically excluded as they are unnecessary for the cross-entropy test. The remaining markers will be displayed on the screen after running the code. If you agree with this selection, respond with 'y' (yes); otherwise, respond with 'n' (no). If you respond, the code will pause, and you'll need to manually adjust the `markers_of_interest` variable, following a similar format as shown here: `c("CD8", "CD4", "CTLA4")`.
- 3) You can adjust the seed for reproducibility using the `fcs.seed.base` variable. Avoid changing this value between runs to ensure consistent and reproducible results.
- 4) The `filesCE` variable is set to use the data loaded in the current environment (`to_import` variable). However, you can modify it to point to a folder containing FCS files.
 - a. Raw files = `fcs`
 - b. Transformed files = `trns`
 - c. Cleaned and transformed files = `pqcd`
 - d. Normalized files = `cn`
- 5) The code generates prefixes based on unique inputs in a column of the metadata file. To assess files for batch effects, keep this set to "batch." In the downstream analysis, you'll have the option to explore further.
- 6) As the current objective is to identify batch effects relevant to CytoNorm, focus on control samples. Subset the files based on the "`sample_control`" group and retain only those labeled as "`control`." These files will be assigned a prefix based on unique values in the "`batch`" column.
- 7) Hit `ctrl + shift + enter` to run the chunk.

On screen there will be an output, copy this according to the messages on screen, and paste it in the `fcs.channel.label` in the next chunk.

Chunk 7: preparing files and directions for cross entropy test.

- 1) Copy the output from the previous chunk and add to the `fcs.channel.label` variable.
- 2) `FCS.cluster.n` => use more clusters for more diverse cell collections. For example, for a broad immune phenotyping panel use 40-50 clusters. For pre-gated cell types (e.g., CD8s), use 10-12 clusters.
- 3) In theory you can alter the significance threshold, only do this if you understand the reason behind the variable.
- 4) You can alter the number of iterations in the UMAP the test uses, default is 1000, the higher you pick this the longer the code takes, same goes for the `fcs.dmr.data.sample.n.per.condition` variable, this can either be a variable or NULL (then it takes all the cells), keep this around 9000 to not let the code run too long.
- 5) Hit ctrl + shift + enter to run the chunk.

NOTE: All results will be saved in the `~wd/cellType/03 Results/RunNo/Cross_Entropy` folder. If you have >2 controls, you have to check the `umap_ce_diff_result_condition.txt` manually. Otherwise a table will be presented on the screen and a note of how many significant values have been found.

Chunk 8: check CSV values for CytoNorm.

Deciding the number of som clusters for CytoNorm model training is an important parameter to take into consideration. This value can be decided based on the `cvsCheck` results. In general, as developers of the package suggest this number is much smaller than the number of meta clusters you use.

In the pipeline, `testCVcluster` is set to `c(3:15,20,25)`, which will give you the results for CytoNorm models using 3 to 15, 20 and 25 som clusters. You can change this according to your own needs.

When you run the code, in the heatmaps you will see CV values for each cluster and all the batches. If the CV value is high (>2) this could mean that this cluster is only observed in that certain batch and may be skewing your CytoNorm model. Find the number of clusters that give the lowest CV results. For more details see the [CytoNorm manuscript](#).

If you don't want to run CytoNorm or if you've run it before, put `performCN` to **FALSE** and continue to the next chunk.

- 1) Put `verbose` to **FALSE** if you don't want messages on screen.
- 2) Put `cvsCheck` to **TRUE** if you haven't checked these files for this data yet.
- 3) Only change the other variables if you understand what impact they have on your data (for more information see the CytoNorm paper:
 - a. <https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.23904>
- 4) Hit ctrl + shift + enter to run the chunk.

Chunk 9: run CytoNorm.

This chunk is based on: <https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.23904>.

CytoNorm, which is based on the hypothesis that batch effects can be modeled on technical controls (TCs) that are run alongside each batch. In short, the process involves clustering the cells of the TCs to identify the different cell types. Next, marker expressions are calculated for each cluster for each batch, representing the different levels of expression. These expression patterns are divided into bins, and a goal distribution is established based on aligning the data across the different groups (i.e. for each cell type-batch-marker combinations). The values of the corresponding groups in the samples of interest are subsequently transformed using Hermite splines to match the desired distribution.

- 1) Set `write_files` to `TRUE` if you want to write diagnostics plots after running CytoNorm, default TRUE
- 2) If you don't want to run CytoNorm or if you've ran it before, `performCN` in the previous chunk was put to `FALSE`. If so, continue to chunk 11, as chunk 10 will also not be required for your analysis.
- 3) Alter `nClustersCN` based on your findings in chunk 8.
- 4) Alter `normParamLimit` to an interval for which you expect the most of marker expression to be. You can check the diagnostic plots created before to decide min & max. If you observe very long tails with a few events in your diagnostics plots, check your `normParamLimit` or CytoNorm som model.
- 5) Only alter the other variables if you understand what impact they have on your data (for more information see the CytoNorm paper mentioned before).

Chunk 10: run the cross-entropy test after normalizing.

1. This chunk is the same as running CE before, but now it will be for after normalization. You need to run this code when you have everything still in your working environment (don't close it before), as it utilizes all the variables created before.
2. Hit ctrl + shift + enter to run the chunk.

3.2 Clustering analysis.

Remember to:

1. Open the "2_FlowSOM_clustering_RADIANT.V1.0_usr.rmd" file in RStudio.
2. Use a copied version of the file to avoid altering the original source.
3. Pay attention to both the displayed notes and the instructions provided in this manual for comprehensive guidance.

Chunk 1: setting environment and folder structure.

DURATION: if you've run the installation process before, this chunk takes only seconds. Run it **twice** to easily check if every package is loaded correctly. If you've not run this before, this chunk can take up to several hours

1. Go back to the first rmd file: "1_Data_Preprocessing_usr_ RADIANT_v1.0.rmd" located within the "00 RADIANT files" directory
2. Copy lines 10-22 (example given below) and place them in the second rmd file in the same place.
3. Hit ctrl + shift + enter to run the chunk.

```
9
10 ### ADJUSTABLE VARIABLES ###
11
12 # set the working directory so the computer knows where the files are
13 wd = "~/Work/RADIANT_trials/Analysis"
14
15 # naming the experiment variables
16 cellType = "BMR3_Bcells" # select the folder containing the data you want to analyse. Valid entries
17 # should match a folder name in wd specified above
18 expID = "BMR3" # add the name of the study, or experiment.
19 runNo = "2: final check hopefully" # add either a number or text, change this for each NEW round you
20 # run to prevent cluttering and overwriting files.
21 file_pattern = "uncomp.fcs" # shared digit/character at the end of fcs files and fcs extension
22
23 # data source
24 ds = "FC" # valid input: if normal flow cytometry = "FS", if spectral flow cytometry = "SFC"
25
26 ### CODE ###
27
28 # importing custom functions
29 #source(paste0(wd, "/", cellType, "/00 RADIANT files/DIFFUSER_dependencies_usr.R"))
30 RMD = "Downstream_analysis"
31 source("~/Work/radiant-pipeline/RADIANT_dependencies_usr_v1.0.R")
32 # install missing packages, install them if required
33 autoInstallpkg()
34 # load required packages
35 autoLoadpkg()
```

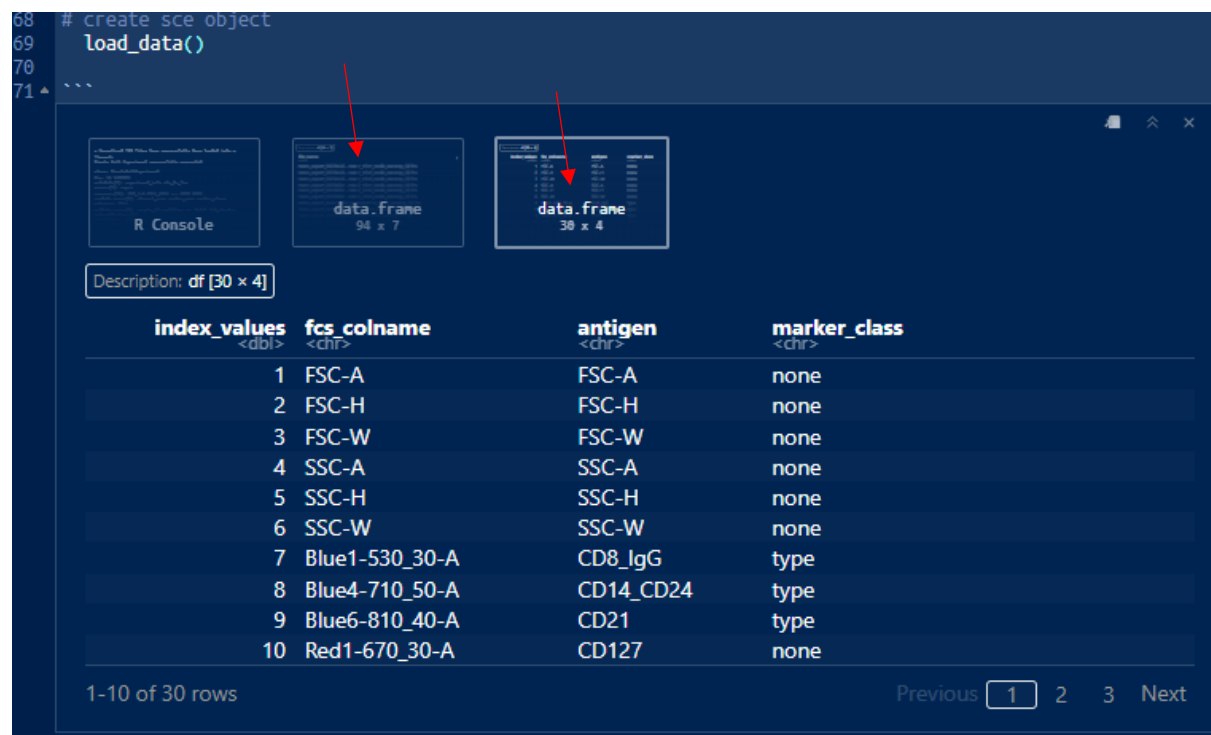
NOTE: You can modify the `runNo` variable as desired. However, it's crucial to maintain consistency with all other variables from the previous file. Any discrepancies might result in numerous errors. Since the files originate from the previous RMD file, in case you encounter significant errors, simply rerun this code and begin anew. Once preprocessing is completed, refrain from making further modifications to that file.

Chunk 2: loading data into the working environment.

DURATION: the more fcs files and events you have the longer it takes.

1. As described before, change the variable `to_import` to the corresponding files you want to load. You can choose: "b" (= transformed files), "c" (= cleaned and transformed files) or "d" (= normalized files). Be aware you are NOT allowed to use the raw files.
 - a. The options you have are given as an output in chunk 1: the files that were detected.
- 2) Hit ctrl + shift + enter to run the chunk.

NOTE: as a check you will have an output of both the metadata file and panel file. They are there as a check for you, if pre-processing went correctly, they should be fine.



```
# create sce object
load_data()
```

Environment: **df** 94 x 7, **df2** 30 x 4

Description: **df** [30 x 4]

index_values <dbl>	fcs_colname <chr>	antigen <chr>	marker_class <chr>
1	FSC-A	FSC-A	none
2	FSC-H	FSC-H	none
3	FSC-W	FSC-W	none
4	SSC-A	SSC-A	none
5	SSC-H	SSC-H	none
6	SSC-W	SSC-W	none
7	Blue1-530_30-A	CD8_IgG	type
8	Blue4-710_50-A	CD14_CD24	type
9	Blue6-810_40-A	CD21	type
10	Red1-670_30-A	CD127	none

1-10 of 30 rows

Previous **1** 2 3 Next

Chunk 3: downsampling, clustering and dimension reduction

DURATION: Depends on the number of cells and whether or not you used UMAP, tSNE or both.

1. If there are any markers you still want to exclude from the analysis you can add them to the `excluded_markers` variable.
2. Next you can indicate the minimum (`min_cells`) and maximum (`max_cells`) number of cells per file that should be included in the FlowSOM clustering. If there is a sample with less than the minimum number of cells, that sample will be excluded from the analysis.
3. The clustering is based on FlowSOM. This method will first cluster the data in 100 SOMCodes, which are then merged into metaclusters. The maximum number of metaclusters is user defined and can be adjusted using the `metClust` variable. The seed for FlowSOM can also be adjusted using the `seedFS` variable. Make sure to only pick one seedFS value for all of your experiments.

HOWEVER: The t-SNE and UMAP algorithms rely on random initialization (random seed), leading to varied visualizations based on the chosen seed. While t-SNE is more sensitive to seed selection than UMAP, both algorithms can produce different overall cluster arrangements. Interpretation should be cautious, and it's advisable to run the algorithms multiple times with diverse random seeds to ensure stability.

4. There is also an option for T-SNE and/or UMAP analysis. This can be indicated in the `anType` variable, which can be set to "UMAP" and/or "T-SNE" (c("UMAP", "T-SNE")). Be aware the differences between the 2 is, if you have no idea, start with UMAPs as these take considerably less computational power.
 - o UMAP: [statQuest-UMAP](#)
 - o tSNE: [statQuest-tSNE](#)
 - o comparing the 2: <https://pair-code.github.io/understanding-umap/>
5. The seed for UMAP/T-SNE can also be adjusted, using the `clusterSeed` variable.
6. Lastly, you can indicate how many cells per sample you want to use for `UMAP` (`UMAP_cells`), or `T-SNE` (`TSNE_cells`) analysis. Recommended is 500 cells. T-SNE can take a long time if too many cells are added.
7. Hit ctrl + shift + enter to run the chunk

Chunk 4: determination requirement of cluster merging.

- 1) Put `Plot_sequences` to TRUE if you have never ran it before.
2. Set the `'start_nr'` parameter to the minimum number of cell types you anticipate. For instance, it's unlikely to have just 2 clusters in a dataset. If you're uncertain, you can set it to 6. Please note that the `'metClust'` variable you calculated earlier represents the total number of clusters you're testing. So, if you set `'metClust'` to 50 and `'start_nr'` to 6, you'll need to generate 44 plots (either UMAP or tSNE) in your analysis.
3. For the maximum number of clusters you expect, leave it to `metClust` in your first run.

4. Hit ctrl + shift + enter to run the chunk

HOW TO DETERMINE THE NUMBER OF CLUSTERS RIGHT NOW AND WHETHER OR NOT TO MERGE THEM: There are many different methods proposed for calculating the number of clusters but at the end of the day it highly depends on research questions and biological interpretation. We recommend starting with a high number of clusters and reducing to number gradually to see which distinct clusters are kept and which ones are merged.

Often, some clusters could be separated by a single variable and it may take a very high number of meta clusters to separate those cells as a distinct cluster. For example, I was looking at marker expressions in UMAPs and saw that CD31 was expressed in a smaller subset of naïve CD4+ T cells. However, I had to increase the number of meta clusters up to 45 to be able to see these cells separating as a cluster. Then with that many meta clusters, I also saw on heatmaps that there were clusters with extremely similar marker expressions, suggesting over-clustering of the data.

Let's say that this CD31+ naïve CD4+ T cell subset was very important for my research question. In this situation, you can go ahead and over-cluster your data and then manually merge these similar clusters using the cluster merging tool that is explained below.

In this case, it is recommended that first you identify what is the more or less ideal number of meta clusters you need (let's say 25) and identify merging which clusters in your over-clustered data set would give you this ideal number of final 25 clusters.

Chunk 5: cluster merging.

1. If after the previous chunk you determined you have to over-cluster and merge, put `performClusterMerging` to **TRUE**. The number of clusters before merging should be the value of `customMeta`.

NOTE: if you put it to FALSE, ignore the other variables but do run all the chunks coming.

NOTE: even if you don't intend to combine clusters but wish to rename them, for example, when you perform FlowSOM clustering multiple times with the same seed, and cluster 1 in one run isn't the same as cluster 1 in another run but they are both named 1 and 5 respectively, you can utilize this method to assign new names to your clusters.

2. If you haven't ran it before, you have to create a merging file, then put `writeCM` to **TRUE**. Now it will create it. Open the file (`merging_table1.xlsx`), adapt it as required (see down below for an example).
3. As an extra check, you can plot some diagnostic plots before (`plotDiagnosticsBefore`) and after (`plotDiagnosticsCM`) merging/rename via putting the corresponding values to **TRUE**.
4. Hit ctrl + shift + enter to run the chunk

If you run the code there will be an error now mentioning either of the following:

- a. You let the pipeline create a merging file, you now have to fill it in, and rerun the code (DON'T FORGET TO PUT `writeCM` to **FALSE**).

```
Excel file for merging clusters successfully created.  
Error in clusterMergingFile(customMeta, writeCM) :  
  Adapt your merging file! And put writeCM to FALSE!
```

- b. You do have a merging file, but you either didn't fill it in completely (as there are **NA** values left) or did not upload it correctly.

```
Error in clusterMergingFile(customMeta, writeCM) :  
  The file merging_table1.xlsx already exists, but it contains NA values. You should update the file with non-NA  
values or set writeCM to TRUE to overwrite it.
```

Uploading the merging file:

You can locate the merging_table1.xlsx file in the following folder: 03 Results/runNo/01 Clustering/01 Cluster merging of x. Upon opening the file, you will find two columns: `original_cluster` and `new_cluster`. The number of rows in the `original_clusters` column corresponds to the `customMeta` you specified (indicating the clusters you either want to merge due to over-clustering or those you've chosen to rename). You can either rename the clusters in the `new_cluster` column accordingly or assign them the name of the merged cluster. Whether you use numbers or names (such as 'naïve B-cells,' for example) is your choice.

	A	B
1	original_cluster	new_cluster
2		1 #N/A
3		2 #N/A
4		3 #N/A
5		4 #N/A
6		5 #N/A
7		6 #N/A
8		7 #N/A
9		8 #N/A
10		9 #N/A

Once you've filled in the file, don't forget to save it (and re-upload it to the server if required).
NOTE: do not leave `NA` in any of the cells.

Chunk 6: evaluation cluster merging.

1. This chunk is an evaluation chunk. You can create a new figure every time you change the variable `name_CM_result`, and you can save it as well if you have put `save_plots` to `TRUE`.
2. There are several variables to alter. Read the notes on screen and play around to see what happens. Remember to change the `name_CM_result` variable if you want to save it rather than overwrite it!
3. Hit `ctrl + shift + enter` to run the chunk

The files can be saved in the 02 Frequency boxplots folder!

3.3 Downstream analysis.

Remember to:

1. Open the "3_Downstream_analysis_RADIANT.V1.0_usr.rmd" file in RStudio.
2. Use a copied version of the file to avoid altering the original source.
3. Pay attention to both the displayed notes and the instructions provided in this manual for comprehensive guidance.

Chunk 1: setting environment and folder structure.

DURATION: if you've run the installation process before, this chunk takes only seconds. Run it **twice** to easily check if every package is loaded correctly. If you've not run this before, this chunk can take up to several hours (so start it either on the server on whatever time you prefer, or at the end of a workday if you do it on your own PC, as it requires A LOT of power).

5. Go back to the first rmd file: "'2_FlowSOM_clustering_usr_RADIANT_v1.0.rmd" " located within the "00 RADIANT files" directory
6. Copy lines 10-22 and place them in the second rmd file in the same place.
7. Hit `ctrl + shift + enter` to run the chunk.

NOTE: You CANNOT modify the `runNo` variable this time. If you do this, you will run into errors immediately, as several variables from clustering are used here, even if you didn't perform clustering. SO, either redo the clustering with another `runNo` name, or keep the same as before.

Chunk 2: create folder and subset the data if required.

1. If you performed renaming/cluster merging in the previous rmd file, you have to put `sce_to_use` to `sce_merge` (and `performClusterMerging` to **TRUE**), otherwise keep at `sce` and `performClusterMerging` to **FALSE**.
2. The plots can be written to pdf files automatically (`write_files` = **TRUE**) and/or plotted in R (`plot_graphs` = **TRUE**).
3. Next you can indicate if you want to subset the data before plotting (e.g. only look at a specific timepoint). You can do this by setting `subsetSCE` to either **TRUE**, or **FALSE**. If set to **TRUE**, you can indicate which variables from the metadata you want to subset for in `subsetCat`. You can then specify exactly what you want to subset for by adding the variable and its condition from the metadata to the `sce_trunc_subs` variable, like this:

```
sce_trunc_subs = filterSCE(sce, age_class == "Adult", Timepoint == "T1")
```

In this example the data was subsetted for adults and timepoint 1.

4. It is also possible to export fcs files containing the UMAP coordinates and FlowSOM metacluster ID's. If you want that to happen set `write_FCS` to **TRUE**. A new folder will be created in your results containing individual fcs files and a concatenated fcs file. Please be aware that the export is limited to the number of cells you've put into your UMAP analysis (`UMAP_cells` in the previous clustering rmd file).
5. For the metacluster abundance plots you can indicate what variable from the metadata you want to use to group the data by (`groupVar`), what variable the shapes of the individual data points should indicate (`shapeVar`) and by what variable the plots should be faceted (`facetVar`). That last one will result in a separate plot for every faceting variable. Additionally you can indicate if you want boxplots and/or violin plots for the metacluster abundance plots (`rcaPlotType`).
6. To order your data, based on what you put into the `groupVar` variable, you can order those outputs in the `groupOrder` variable in a `c(x,y,z)` manner. To find unique values in that column of your data if you're not sure, run: **`unique(sce$groupVar)`** or **`unique(sce_merge$groupVar)`** depending on if you merged or not. Be sure to change `groupVar` to the name of the column! So for example: `unique(sce_merge$batch)`.
7. Next you can indicate how many metaclusters you want to plot. The maximum is the value you gave to `metClust` in the previous chunk.
8. Hit ctrl + shift + enter to run the chunk. You can adjust the parameters and rerun the chunk as often as you like. The files will be written to separate folders every time, so you don't have to worry about anything being overwritten. However, remember that in the next chunk statistical analysis can be performed, so if you want to keep that in the folder run chunk 3 before redoing chunk 2.

Chunk 3: statistical analysis.

1. If you want to run statistics, put `runStatistics` to **TRUE**.
2. With the PWC variable you can indicate which test to run for pairwise comparisons. Valid entries are "wilcox.test" and "t.test"
3. With the AOV variable you can indicate which test to run for analysis of variance. Valid entries are "kruskal.test" and "anova". If pairedData is TRUE and AOV is not anova a friedman test will be performed.
4. If you have paired data between the groups indicated at groupVar, use pairedData = **TRUE**.
5. With the multiCompAdjust you can indicate the correction method for multiple comparisons. Valid entries are: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr"
6. Hit ctrl + shift + enter to run the chunk.

REFERENCES

- Nowicka, M., Krieg, C., Crowell, H. L., Weber, L. M., Hartmann, F. J., Guglietta, S., Becher, B., Levesque, M. P., & Robinson, M. D. (2017). CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research*, 6, 748. <https://doi.org/10.12688/f1000research.11622.3>
- Ashhurst, T. M., Marsh-Wakefield, F., Putri, G. H., Spiteri, A. G., Shinko, D., Read, M. N., Smith, A. L., & King, N. J. C. (2022). Integration, exploration, and analysis of high-dimensional single-cell cytometry data using Spectre. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 101(3), 237–253. <https://doi.org/10.1002/cyto.a.24350>
- Emmaneel, A., Quintelier, K., Sichien, D., Rybakowska, P., Marañón, C., Alarcón-Riquelme, M. E., Van Isterdael, G., Van Gassen, S., & Saeys, Y. (2022). PeacoQC: Peak-based selection of high quality cytometry data. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 101(4), 325–338. <https://doi.org/10.1002/cyto.a.24501>
- Van Gassen, S., Gaudilliere, B., Angst, M.S., Saeys, Y. and Aghaeepour, N. (2020), CytoNorm: A Normalization Algorithm for Cytometry Data. *Cytometry*, 97: 268-278. <https://doi.org/10.1002/cyto.a.23904>
- Van Gassen, S., Callebaut, B., Van Helden, M. J., Lambrecht, B. N., Demeester, P., Dhaene, T., & Saeys, Y. (2015). FlowSOM: Using self-organizing maps for visualization and interpretation of

cytometry data. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 87(7), 636–645. <https://doi.org/10.1002/cyto.a.22625>

Roca, C. P., Burton, O. T., Neumann, J., Tareen, S., Whyte, C. E., Gergelits, V., Veiga, R. V., Humblet-Baron, S., & Liston, A. (2023). A cross entropy test allows quantitative statistical comparison of t-SNE and UMAP representations. *Cell reports methods*, 3(1), 100390. <https://doi.org/10.1016/j.crmeth.2022.100390>