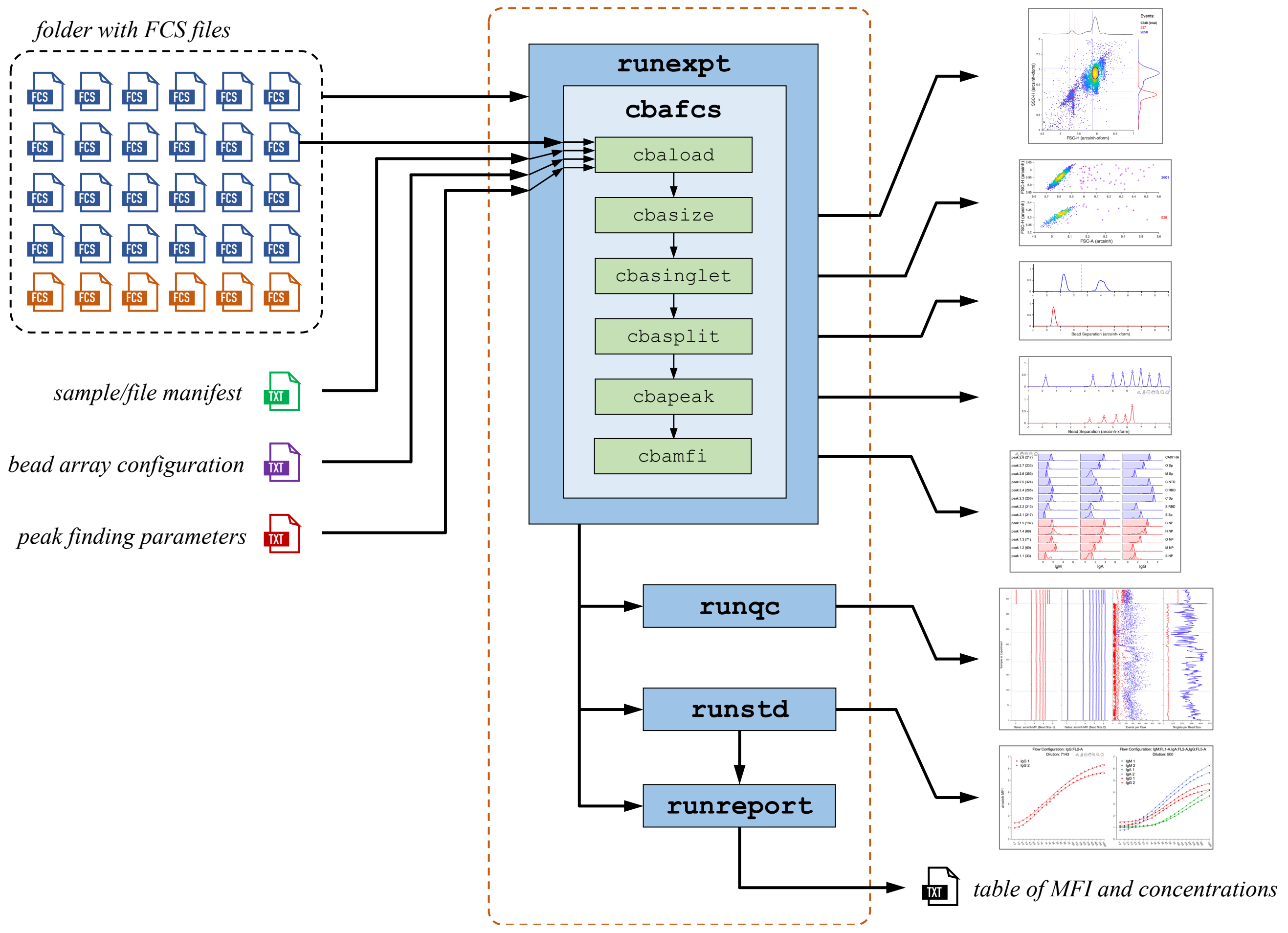
Cytometric Bead Array Informatics Toolbox

Alex Rosenberg, 11/7/2024

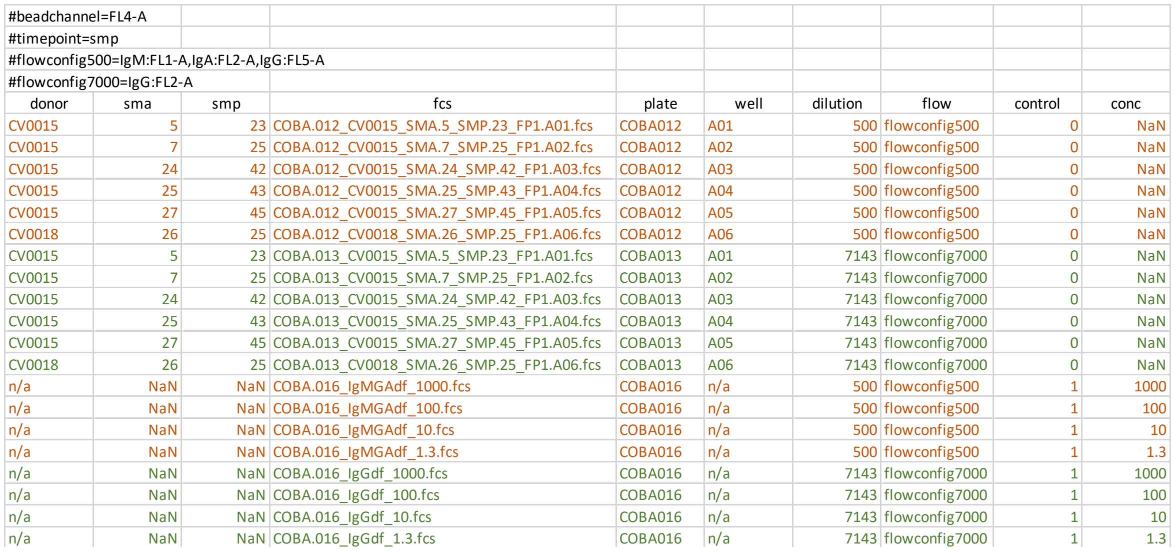
Workflow Overview

The Cytometic Bead Array Informatics Toolbox is a set of programs that automatically processes collections of serum bead array FCS files. The functional unit is the "experiment" which includes a set of samples (that can span several plates) all processed in the same flow cytometry run. These include the standards used to construct the calibration curve. To use this software, the FCS files need to be in a specified folder, and they need to be listed in a tabular sample manifest with additional information (described below). Additionally, two configuration files need to be specified – one to describe the mapping of the beads to peaks (for multiple bead sizes used) and one to set the peak finding parameters to be used for automatic gating. These two configuration files are applied to all the FCS files in the experiment. The first program **runexpt** reads the sample manifest and the configuration files and iterates through all the FCS files in the manifest. For each file, a function **cbafcs** performs multiple gating steps (separating multiple bead sizes, doublet elimination, gating on an optional split channel and identifying peaks for each size) and then quantifies the MFI for each feature and for each isotype on the secondary channels. These results are contained in an array of data structures. The next program **runqc** reads this array of data structures and plots the peak gates for each bead size assess the reproducibility of the peak finding as well as the number of events associated with each feature for each of the FCS files. The program **runstd** automatically identifies the control samples in the experiment (as specified in the manifest), constructs a calibration curve for each isotype, bead size and flow design detected and fits each to a four-parameter logistic function. The next program **runreport** takes the structure array of MFI data for all the samples, and optionally, the calibration curve fit parameters to apply the appropriate calibration curves per isotype and sample to compute the corresponding concentrations, if desired. Additionally, **runreport** compiles all the sample, MFI and concentration data for all features and isotypes into single tabular files (one per flow configuration) suitable for downstream analysis. This workflow is shown below:



Manifest File Format

The sample manifest is a list of FCS files that comprises an experiment. It is a tab-delimited text file with header rows. An example is shown below (not all samples are shown for clarity):



#beadchannel=FL4-A									
#timepoint=smp									
#flowconfig500=IgM:FL1-A,IgA:FL2-A,IgG:FL5-A									
#flowconfig7000=IgG:FL2-A									
donor	sma	smp	fcs	plate	well	dilution	flow	control	conc
CV0015	5	23	COBA.012_CV0015_SMA.5_SMP.23_FP1.A01.fcs	COBA012	A01	500	flowconfig500	0	NaN
CV0015	7	25	COBA.012_CV0015_SMA.7_SMP.25_FP1.A02.fcs	COBA012	A02	500	flowconfig500	0	NaN
CV0015	24	42	COBA.012_CV0015_SMA.24_SMP.42_FP1.A03.fcs	COBA012	A03	500	flowconfig500	0	NaN
CV0015	25	43	COBA.012_CV0015_SMA.25_SMP.43_FP1.A04.fcs	COBA012	A04	500	flowconfig500	0	NaN
CV0015	27	45	COBA.012_CV0015_SMA.27_SMP.45_FP1.A05.fcs	COBA012	A05	500	flowconfig500	0	NaN
CV0018	26	25	COBA.012_CV0018_SMA.26_SMP.25_FP1.A06.fcs	COBA012	A06	500	flowconfig500	0	NaN
CV0015	5	23	COBA.013_CV0015_SMA.5_SMP.23_FP1.A01.fcs	COBA013	A01	7143	flowconfig7000	0	NaN
CV0015	7	25	COBA.013_CV0015_SMA.7_SMP.25_FP1.A02.fcs	COBA013	A02	7143	flowconfig7000	0	NaN
CV0015	24	42	COBA.013_CV0015_SMA.24_SMP.42_FP1.A03.fcs	COBA013	A03	7143	flowconfig7000	0	NaN
CV0015	25	43	COBA.013_CV0015_SMA.25_SMP.43_FP1.A04.fcs	COBA013	A04	7143	flowconfig7000	0	NaN
CV0015	27	45	COBA.013_CV0015_SMA.27_SMP.45_FP1.A05.fcs	COBA013	A05	7143	flowconfig7000	0	NaN
CV0018	26	25	COBA.013_CV0018_SMA.26_SMP.25_FP1.A06.fcs	COBA013	A06	7143	flowconfig7000	0	NaN
n/a	NaN	NaN	COBA.016_IgMGAdf_1000.fcs	COBA016	n/a	500	flowconfig500	1	1000
n/a	NaN	NaN	COBA.016_IgMGAdf_100.fcs	COBA016	n/a	500	flowconfig500	1	100
n/a	NaN	NaN	COBA.016_IgMGAdf_10.fcs	COBA016	n/a	500	flowconfig500	1	10
n/a	NaN	NaN	COBA.016_IgMGAdf_1.3.fcs	COBA016	n/a	500	flowconfig500	1	1.3
n/a	NaN	NaN	COBA.016_IgGdf_1000.fcs	COBA016	n/a	7143	flowconfig7000	1	1000
n/a	NaN	NaN	COBA.016_IgGdf_100.fcs	COBA016	n/a	7143	flowconfig7000	1	100
n/a	NaN	NaN	COBA.016_IgGdf_10.fcs	COBA016	n/a	7143	flowconfig7000	1	10
n/a	NaN	NaN	COBA.016_IgGdf_1.3.fcs	COBA016	n/a	7143	flowconfig7000	1	1.3

Header rows preceded by a “#” are parsed as attribute-value pairs (separated by a “=” with no additional spaces). the “beadchannel” attribute specifies which channel of the FCS file to use for bead separation. This is applied to all FCS files in the experiment. The next row sets the “timepoint” attribute and specifies which column to interpret as a time point variable (the value must correspond to one of the column headers in the main table). The next two header rows specify two flow configurations used in this experiment. A flow configuration string is constructed as a comma-separated list of secondary isotype – flow channel tuples. This arrangement allows multiple flow configurations to be used for different subsets of samples. In this example, two dilutions are used for each sample. For the 1:500 dilution samples, IgM, IgG and IgA secondaries are used (red rows). For the 1:7000 dilution samples, only IgG secondaries are used (blue rows). The flow configuration used for a particular sample is specified as the value of the “flow” column. Valid values for this column are the attributes of the attribute-value pairs for the flow configuration specifications in the headers (to the left of the “=”). The “control” column specifies whether the FCS file corresponds to a sample used for the standard curves. For standard curve samples, there must be a concentration value in the “conc” column. In general, missing data should be denoted by NaN (“not-a-number”) in numeric columns and “n/a” in text columns. The columns shown above are required although additional columns are permitted.

Array Configuration File

This file links the beads to particular proteins. It is a text file of attribute-value pairs, where the attributes and values are separated by whitespace then an equals-sign then more whitespace. An example is shown below:

```
bead_mfg_1             = BeadCo Inc.
bead_mfg_2             = BeadCo Inc.
bead_catalog_number_1 = B235735
bead_catalog_number_2 = B235737
bead_size_1            = 4um
bead_size_2            = 5um
bead_peak_order_1      = S NP,M NP,O NP,H NP,C NP
bead_peak_order_2      = S Sp,S RBD,C Sp,C RBD,C NTD,M Sp,O Sp,CA07 HA
bead_ctrl_order_1      = IgM,IgA,IgG,IgM,IgA,IgG,IgM,IgG
bead_ctrl_order_2      = IgM,IgA,IgG,IgM,IgA,IgG,IgM,IgA,IgG
```

This software is currently set up for up to two bead sizes – it can be extended to more bead sizes. This file can capture the manufacturer and catalog # for each bead series used (in this case two). The last four lines describe to format of the assay – the peak orders for the proteins used and those for the secondary isotype controls. Note that the proteins and isotype controls can have different numbers of peaks as in this example.

The bead array can be configured to double the number of proteins detected by dividing both bead sizes into two pools, half of each labeled for detection on an otherwise unused flow channel (a "split channel"). If, for example, only the 5um beads were split into two pools, then the bead peak order lines in the array configuration file would appear as follows:

```
bead_peak_order_1     = ag1,ag2,ag3,ag4,ag5,ag6
bead_peak_order_2_1   = ag7,ag8,ag9,ag10,ag11,ag12,ag13,ag14,ag15
bead_peak_order_2_2   = ag16,ag17,ag18,null,ag19,ag20,null,ag21,ag22
```

The two bead sizes can have different numbers of members. If a bead size is split, however, the two resulting bins need to have the same number of proteins (use the keyword "null" for unused beads).

Peak Detection Profile

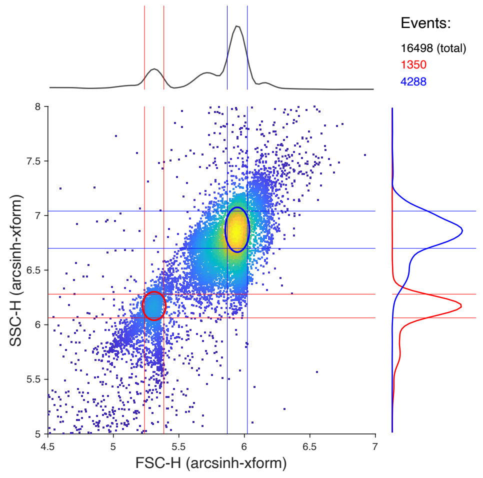
This file contains parameters used for the gating process. It is a text file of attribute-value pairs, where the attributes and values are separated by whitespace then an equals-sign then more whitespace. An example is shown below:

```
beadread_asinh_scale_factor  = 3000;
beadsize_xfsc_range          = [4.5 7]
beadsize_yssc_range          = [5 8]
beadsize_oval_gate_scale     = 1.2
beadsize_bin_size            = .025
beadsize_loess_smooth        = .15
beadsize_min_peak_height     = .06
beadsize_min_peak_prominence = .07
beadsing_dbclus_epsilon      = .04
beadsing_dbclus_min_points   = 50
beadpeak_range               = [-1 9]
beadpeak_bin_size            = .01
beadpeak_loess_smooth        = .04
beadpeak_min_peak_height     = .03
beadpeak_min_peak_prominence = .03
beadreac_bin_size            = .01
beadreac_loess_smooth        = .08
beadreac_range               = [-1 7]
```

The first parameter is the scaling factor used for the hyperbolic arcsine transform of the raw flow data. The parameters beginning with "beadsize" are used for forward scatter vs. side scatter plot used to identify the different bead sizes. The parameters beginning with "beadsing" are used for the density-based clustering for singlet detection. The parameters beginning with "beadpeak" are used for the individual bead gates for both size beads. The parameters beginning with "beadreac" are used for the secondary antibody detection channels analysis. [I will add a section indicating what each parameter does]

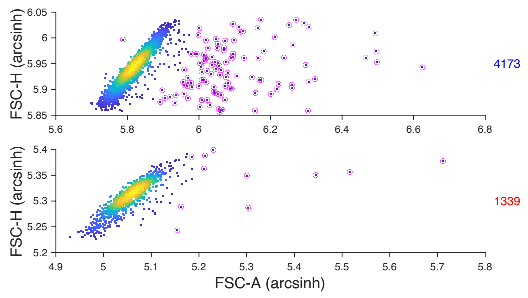
Visualizations

For a single instance of running **cbafcs**, you can generate visualizations that are useful for troubleshooting. These capture all stages of processing. These plots are suppressed in batch mode. The first plot shows the forward scatter vs. side scatter used for identifying different bead sizes in the flow data.

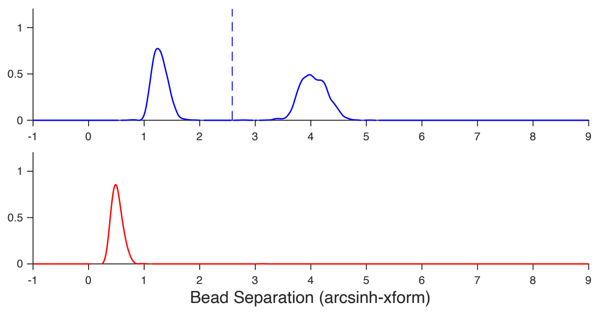


This plot shows a 2-D density scatterplot along with the identification of the expected peaks on the forward-scatter axis and the corresponding peaks on the side-scatter axis used to construct the oval gates corresponding to the multiple bead sizes.

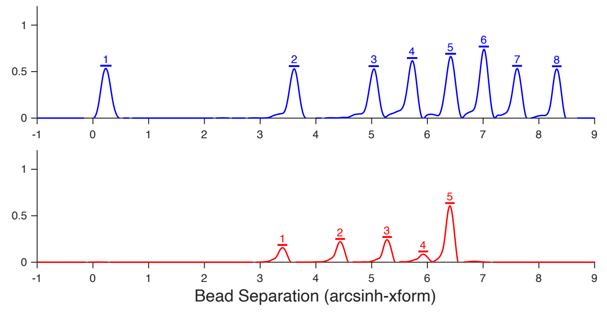
The next plot shows the forward scatter area vs. height plots for each bead-size gate used to identify singlets to retain for further consideration. Density-based clustering is used at this stage to identify outliers to remove (shown with magenta circles).



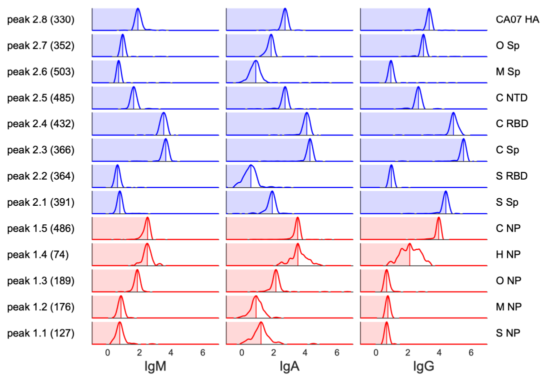
If an optional split channel is used (to split one or both bead sizes by an otherwise unused flow channel), a plot is shown indicating thew location of the splitting gate in hyperbolic arcsine transformed MFI. In this example, the larger (5um) beads were split (blue).



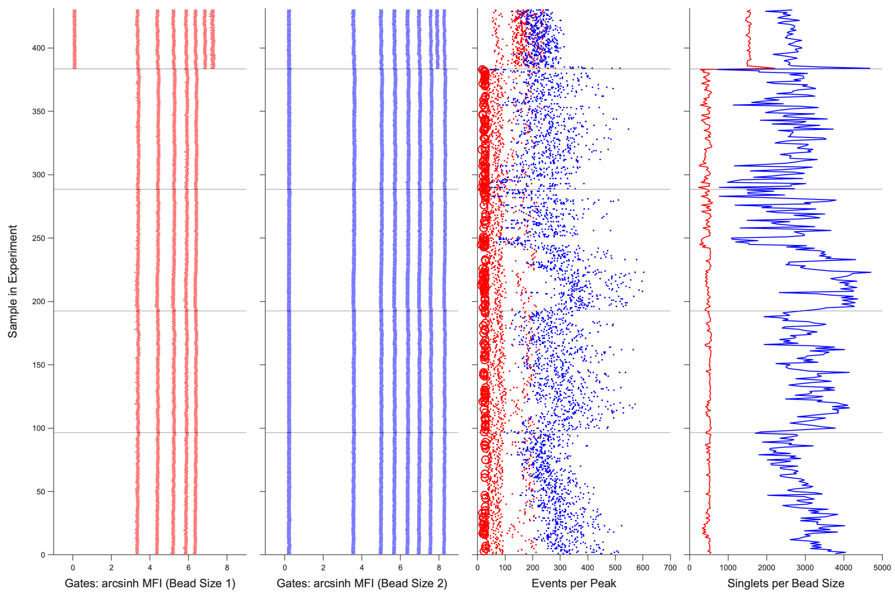
The next plot shows the hyperbolic arcsine transformed MFI of the bead channel (x-axis) for the different bead sizes along with the called peaks and corresponding gates. This plot can be useful for adjusting the peak-finding parameters if necessary.



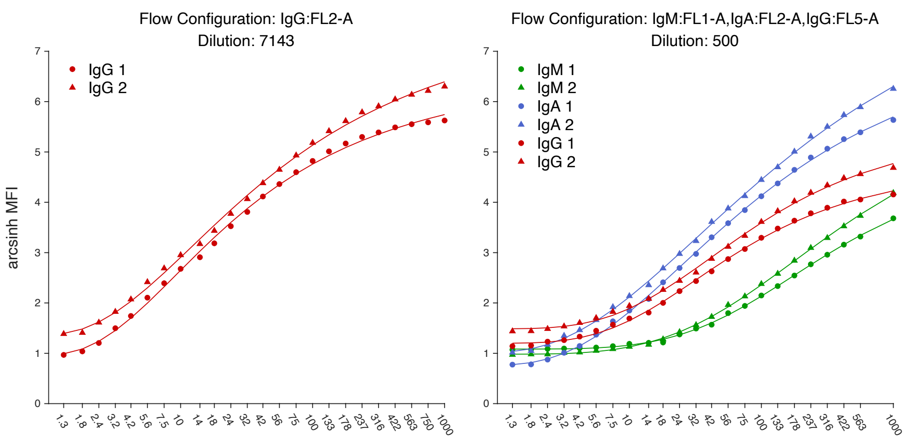
Finally, the hyperbolic arcsine transformed MFI (x-axis) are shown for each feature in all of the secondary isotype channels:



After executing **runqc** on the output of **runexpt**, a visualization showing the gating properties of all samples in the experiment is shown. In the left plots, the positions of the gates based on the automatically detected peaks are shown (hyperbolic arcsine transformed MFI on the x-axis) for each sample (y-axis) for the entire experiment (plates demarcated by horizontal lines). A separate plot is made for each bead size. The next plot shows the numbers of events per feature (bead sizes indicated by color). Points below a specified number of events are annotated with a circle for a sample/feature. The right-most plot shows the number of singlets for each bead size (indicated by color).



Finally, after running **runstd**, a plot of the standard curve data and the corresponding fits are plotted for each flow configuration. Standard curve fits are computed separately for each bead size.



Software Installation

This software was developed on Matlab R2020a for MacOS. The following toolboxes are required:

- Statistics and Machine Learning Toolbox
- Curve Fitting Toolbox
- Signal Processing Toolbox

All Matlab files should be in the directory `$cbaroot/m/`. Both `$cbaroot/m/` and `$cbaroot/m/util/` should be on the Matlab path (i.e. added to the list of directories where m-files are expected to be found). The required software is shown below. In this example, `$cbaroot = cba`:

```
cba
└──── m
      ├──── cbafcs.m
      ├──── cbaheader.m
      ├──── cbaload.m
      ├──── cbamfi.m
      ├──── cbapdf.m
      ├──── cbapeak.m
      ├──── cbasinglet.m
      ├──── cbasize.m
      ├──── cbasplit.m
      ├──── cbatxt.m
      ├──── runexpt.m
      ├──── runqc.m
      ├──── runreport.m
      ├──── runstd.m
      └──── util
            ├──── L4P.m
            ├──── L4P_LICENSE
            ├──── dscatter.m
            ├──── fca_readfcs.m
            ├──── fca_readfcs_license.txt
            ├──── gatechannel.m
            ├──── oval.m
            └──── validatefile.m
```

Function Reference

runexpt

Wrapper function to read a sample manifest and perform automated gating and reactivity quantification on a collection of samples.

usage:

```
X = runexpt('/path/to/fcs/files/', 'arraydesign.txt', 'profile.txt', 'manifest.txt')
```

input arguments:

1. *absolute path to directory with FCS files*
2. *bead array configuration text file name**
3. *peak detection parameter text file name**
4. *sample manifest text file name**

**needs path if file not in current directory*

return value is a structure array (each element corresponding to a row in the manifest file) with the following fields:

- `donor` *(donor identifier)*
- `days` *(time point)*
- `fcs` *(fcs file name)*
- `plate` *(plate name)*
- `well` *(well coordinate)*
- `dilution` *(serum dilution)*
- `iscontrol` *(control flag: 1 = yes, 0 = no)*
- `concentration` *(concentration used for standard curve sample or NaN)*
- `results` *(a structure array with the outputs for antibody reactivity quantification function* `cbafcs`*)*

The FCS files listed in the manifest file (4th input argument) must exist in the specified FCS file folder (1st input argument).

cbafcs

Function to determine antibody reactivities from a single FCS file.

usage:

```
R = cbafcs('/path/to/fcs/files/samp.fcs', 0, 'arraydesign.txt', 'FL4-A', ...
      'IgM:FL1-A,IgA:FL2-A,IgG:FL5-A', 'profile.txt', 'display', 'off')
```

usage when implementing a split channel (see fourth input argument, where the bead channel is specified followed by the split channel with a colon delimiter):

```
R = cbafcs('/path/to/fcs/files/samp.fcs', 0, 'arraydesign_split.txt', 'FL4-A:FL3-A', ...
      'IgM:FL1-A,IgA:FL2-A,IgG:FL5-A', 'profile.txt', 'display', 'off')
```

input arguments:

1. *full path to FCS file*
2. *flag to indicate whether data is from a control sample (1 = yes, 0 = no)*
3. *bead array configuration text file name*
4. *bead channel in FCS file (and optional split channel using colon delimiter)*
5. *flow configuration (isotype – flow channel pairs)*

6. *peak detection parameter text file name*
7. *additional optional arguments as parameter-value pairs:*
    `'display'` *('yes' or 'no') to render figures*
    `'txt'` *('yes' or 'no') to produce a text output file of results*
    `'pdf'` *('yes' or 'no') to produce a PDF report*
    `'outdir'` *to specify an output directory other than current directory*

return value is a structure with the fields:

- `version` *(CBA Toolbox software version)*
- `fcsname` *(FCS file name with full path)*
- `analysisdate` *(analysis time stamp)*
- `profile` *(peak detection parameter file used)*
- `arrayconf` *(bead array configuration file used)*
- `settings` *(a structure with values from the peak detection parameter file)*
- `peakorder1` *(peak order antigen mapping for bead size 1)*
- `peakorder2` *(peak order antigen mapping for bead size 2)*
- `beadchan` *(flow cytometry channel used for bead peak separation)*
- `flowconf` *(flow configuration: isotype – flow channel pairs)*
- `header` *(header values extracted from FCS file)*
- `chan` *(list of FCS file channel labels)*
- `raw` *(a 2-D event-by-channel array of raw FCS file data)*
- `event` *(table of FSC, SSC, bead, isotype channels, arcsinh-transformed data, including assigned bead sizes/peaks)*
- `counts` *(structure with count statistics for each bead size – total and singlets)*
- `out` *(output table with gate MFIs, features, event counts, isotype reactivities)*

runqc

Plotting utility to generate QC visualization for an experiment showing bead gate positions and events per sample per gate.

usage:

```
runqc(X, 30)
```

input arguments:

1. *structure array output from* `runexpt`
2. *a threshold number of events per gate to highlight on plot*

return values:

- *none*

runstd

Function to identify control samples in an experiment and generate standard curves and 4-parameter logistic fits and to plot the curves – NOTE: does not yet work with split channels

usage:

```
Q = runstd(X)
```

input argument:

1. *structure array output from* `runexpt`

return value is a structure array (one per flow config/dilution) with fields:

- `flow` *(flow configuration: isotype – flow channel pairs)*
- `dilution` *(serum dilution factor)*
- `secondaries` *(list of isotypes)*
- `curve` *(a table of FSC file names, concentrations and arcsinh MFI per isotype/bead-size combination)*
- `isort` *(row indices top sort concentration curve table)*
- `lp4` *(a structure array containing the 4-PL fit parameters for each isotype/bead-size combination)*
- `fitparam` *(a table of bead size, isotype and 4-PL fit parameters (a simplified version of the* `lp4` *field)*

runreport (to replace **runcalc**)

Function to generate tabular reports of batch runs (from `runexpt`) – can create MFI tables and optionally, back-calculated arbitrary concentration units if a standard curve object (from `runstd`) was specified. This function generates a table for each flow config and can optionally output to text files.

Usage:

```
Y = runreport(X, 'myexpt')
```

returns a table for each flow config, and writes these tables to text files in the current directory

```
Y = runreport(X, 'myexpt', 'outdir', '/Users/afr/Desktop/')
```

specifies where to write the text files

```
Y = runreport(X, 'myexpt', 'stdcurve', Q)
```

adds back-calculated arbitrary concentration

```
Y = runreport(X, 'myexpt', 'stdcurve', Q, 'txt', 'no')
```

suppresses writing to output text file

input arguments

1. *standard array output from runexpt*
2. *name of experiment (text)*
3. *additional optional arguments as parameter-value pairs:*
   `'stdcurve'` *a standard curve object produced by* `runstd`
   `'txt'` *('yes' – default - or 'no') to produce a text output file of results*
   `'outdir'` *to specify an output directory other than current directory*

Return value is a structure array (one per flow config) where each element has a table with the columns:

- *experiment name*
- *donor*
- *day*
- *FCS file name*
- *plate*
- *well*
- *dilution factor*
- *flow configuration*
- *group of columns: events for each array feature*
- *group of columns: untransformed MFI for each feature/isotype*
- *group of columns: arcsinh MFI for each array feature/isotype*
- *group of columns: concentration for each array feature/isotype (if a standard curve object was provided)*

runcalc - DEPRECATED

Function to compute antibody concentrations from arcsinh MFI data output for an experiment and associated standard curve data.

usage:

```
Y = runcalc(X, Q, 'myexpt')
```

input arguments:

1. *structure array output from* `runexpt`
2. *standard curve structure array output from* `runstd`
3. *name to give this experiment*

return value is a structure array (one per flow config/dilution) where each element has a table with columns:

- *experiment name*
- *donor*
- *day*
- *FCS file name*
- *plate*
- *well*
- *dilution factor*
- *flag indicating control (should all be 0)*
- *concentration (should all be NaN)*
- *flow configuration*
- *group of columns: events for each array feature*
- *group of columns: arcsinh MFI for each array feature/isotype*
- *group of columns: concentration for each array feature/isotype*