

# Using RFlowCyt

A.J. Rossini      J.Y. Wan

May 19, 2005

## Abstract

This manual describes the usage of the functions in the *rflowcyt* library package. The main categories are Data Management and Retrieval, Flow Cytometry Visualizations, Exploratory Analysis, Gating, and Flow Cytometry Hypothesis Testing and Statistical Inference. Examples are also shown for each category.

## 1 Data Management and Retrieval

The *rflowcyt* tools (ie, in parentheses) in this category are used for the following tasks:

1. ("read.FCS") to read in FCS binary files into R-objects of S4 class "FCS"
2. ( "[i,j]" ) to extract or subset information from the "data" (a "matrix" object) of the FCS R-object
3. ("[[i]]") to extract "metadata" (which is of S4 class "FCSmetadata") portion of the FCS R-object
4. ("[i,j]<-") to replace "data" information
5. ("addParameter") to add column variables in the "data"
6. ("[[i]<-") to replace or add new information to the "metadata" portion of the FCS R-object
7. ("as") to coerce among "FCS", "data.frame" and "matrix" class objects
8. ("equals") to demonstrate equality between two FCS R objects
9. ("print-methods", "show-methods") print or show methods for "FCS" objects
10. ("checkvars") to check for any discrepancies between the metadata and the data of the "FCS" object
11. ("fixvars") to fix the metadata to reflect the information obtained from the data if there are discrepancies

12. ("summary-methods") to summarize the FCS R-object with Tukey's five number summary of the data and with slot information in the metadata and to output an "FCSsummary" S4 class object

## 1.1 Datasets

There are two types of data sets that are available in the required data package *rfdmin*. The first type of data set consists of raw binary Flow Cytometry Standard (FCS) files, and the second type consists of R-objects of S4 class "FCS" and is the result of reading in an FCS binary file using "read.FCS" function. There are other larger data packages called *rfdfhcrc2* and *rfdorig* which also contains FCS binary files and from which *rfdmin* is obtained.

### 1.1.1 Binary FCS data files

The Flow Cytometry Standard (FCS) binary files consist of a HEADER, a TEXT, a DATA, and an optional ANALYSIS segment. The HEADER in ASCII text gives information about the version of the FCS file and the byte offsets of the beginning and ending of the other segments within the FCS file. The FCS version 3.0 is currently used and has been updated from version 2.0 to accommodate data sets longer than 99,999,999 bytes and allowing for primary and supplemental portions within the TEXT segment, among other changes. Located after the HEADER, the TEXT segment in ASCII text includes summary information in keywords such as the number of observations and names of column variables in the DATA segment. The DATA segment that follows consists of the raw binary data. The optional ANALYSIS segment includes some results of earlier data analyses (Robinson, 2001).

#### A Note about the Data Range

When specific immunofluorescence signals are received and digitized by the analog-to-digital converters (ADCs) of a flow cytometer machine, the measurements are grouped into a number of bins based on the bit-resolution of the ADC. Thus, a  $n$ -bit resolution ADC will group the data into  $2^n$  bins or "channels". Thus, each immunofluorescence measurement variable is actually categorical and has an integer range from 0 to  $2^n$ , depending on the ADC bit-resolution, which is usually 10 or 8 (Robinson, 2001).

The following removes all objects such that the following examples can be run in a cleared workspace.

```
> rm(list = ls())
```

Loading required package: xtable

### 1.1.2 Reading in the FCS binary file: "read.FCS" and "fcs.type"

The subsequent code will allow for us to call the *rflowcyt* library. If the *rflowcyt* library is in the working library location, then the "library.location" is a character string identifying the location of the *rflowcyt* library.

	FCS Version	Source	Machine	bit resolution	Integer range
073100v3.013.fcs	2.0	UW	CELLQuest	10	0–1024
DiVa-262k-FCS3.fcs	3.0	ANU	DiVa	18	0–262,144
DiVa1024.fcs	2.0	ANU	DiVa	10	0–1024
FACSCalibur256.fcs	2.0	ANU	FACSCalibur	8	0–256
FACSCalibur1024.fcs	2.0	ANU	FACSCalibur	10	0–1024
facscan256.fcs	2.0	ANU	facscan	8	0–256
facscan1024.fcs	2.0	ANU	facscan	10	0–1024
FACStar256.fcs	2.0	ANU	FACStar	8	0–256
LSR256.fcs	2.0	ANU	LSR	8	0–256
LSR1024.fcs	2.0	ANU	LSR	10	0–1024

Table 1: Example FCS binary files in 'rfcdorig' package that can be read in using read.FCS

```
> if (!require(rflowcyt)) {
+   stop("Rflowcyt not available?")
+ }
```

```
Loading required package: rflowcyt
Loading required package: survival
Loading required package: splines
Loading required package: KernSmooth
KernSmooth 2.22 installed
Copyright M. P. Wand 1997
Loading required package: MASS
Loading required package: grid
Loading required package: splancs
```

Spatial Point Pattern Analysis Code in S-Plus

```
Version 2 - Spatial and Space-Time analysis
Loading required package: hexbin
Loading required package: colorspace
```

```
Attaching package: 'colorspace'
```

The following object(s) are masked from package:grDevices :

```
hcl
```

```
Loading required package: rfcdmin
Loading required package: fields
fields is loaded use help(fields) for an overview of this library
```

```
> if (!require(rfcdmin)) {
+   stop("rfcdmin not available?")
+ }
```

Alternatively, the data packages *rfcdorig* or *rfcdfhcrc2* could also be called using similar commands as demonstrated above.

In order to read in the FCS binary file, the location of the FCS binary file in the */fcs* directory of the *rfcdmin*, *rfcdorig*, or the *rfcdfhcrc2* package has to be input as a parameter in the calling for "read.FCS".

Table 1 summarizes the current reading information for the raw binary files in the *rfcdorig* data package. Only the "facscan256.fcs" and "SEB-NP22.fcs" binary files are available in *rfcdmin*. For more information about the binary FCS files for the *rfcdfhcrc2* and *rfcdorig* packages, look at the package documentation files using the commands in R:

```
> help(package = "rfcdmin")
```

The ".fcs" files noted in each package are raw binary FCS files which have to be read into R by using the **read.FCS** function. The output is a FCS R object of S4 or S3 class. Currently, the *rflowcyt* package implements functions and methods with the S4 class.

We will find the */fcs* directory location containing the FCS raw binary files within the *rfcdmin* package.

```
> fcs.loc <- system.file("fcs", package = "rfcdmin")
```

After finding the .fcs file location, we will read in the raw binary file "facscan256.fcs" using **read.FCS** and call it "FC.FCSRobj". In order to demonstrate a S3-to-S4 class change, we will incorrectly read in the binary file as an S3 object.

```
> file.location <- paste(fcs.loc, "facscan256.fcs", sep = "/")
> FC.FCSRobj <- read.FCS(file.location, UseS3 = TRUE, MY.DEBUG = FALSE)
```

NOTE: Long names \$PnS are missing.  
Short names \$PnN are assigned to the dataset instead.

Currently, **read.FCS** will read in all the ".fcs" raw, binary files from all three packages: *rfcdmin*, *rfcdorig*, and *rfcdfhcrc2*.

If the "fcs.type" option is specified for a particular flow cytometer machine, the "read.FCS" options will be pre-specified automatically without user input. Depending on the machine from which the FCS binary file was obtained, the "fcs.type" can be specified as one of the following in Table 2.

See also "fcs.type" documentation for details. If your FCS raw, binary files cannot be read into R using "read.FCS", please notify A.J. Rossini (email: blind-globe@gmail.com).

	fcs.type
1	fcs.type.LYSYS
2	fcs.type.cellquest.3.1.FACS.Vantage
3	fcs.type.cellquest.3.1.FACScan
4	fcs.type.cellquest.3.3
5	fcs.type.default
6	fcs.type.LSR256
7	fcs.type.facsan256
8	fcs.type.DiVa1024
9	fcs.type.FACSCalibur1024
10	fcs.type.LSR1024
11	fcs.type.facsan1024

Table 2: Summary of FCS types used by read.FCS

### 1.1.3 FCS R-objects

The following are FCS R-objects which are readily accessible in R and can be used for analysis using the tools in the *rflowcyt* package. Prior to this release, the FCS class has been S3. Now the FCS class among other classes (FCSmetadata, FCSgate (described in the Gating section), FCSggobi (still in working progress) ) are S4. Use **convertS3toS4** to convert S3-class "FCS" to S4-class "FCS" R objects.

To exemplify the conversion, the following demonstrates the conversion of an S3 FCS R-object to an S4 FCS R-object:

```
> FC.FCSObj <- convertS3toS4(FC.FCSObj, myFCSobj.name = "FC.FCSObj",
+   fileName = file.location)
```

A "FCS" R object has slots "data" and "metadata". The "data" component is a matrix in which the rows are the individual cells or observations, and the columns are the different immunofluorescence measurements or variables. The metadata is of S4-class "FCSmetadata" and has slots referring to keywords that are in the TEXT segment of the FCS raw binary file. Information such as variable names ("longnames" and "shortnames") and ranges ("paramranges") are also slots in the metadata component. For more details, see the help files for "FCS" and "FCSmetadata".

### 1.1.4 Opening *rflowcyt* data with FCS R-objects

A simple use of the "data" function will enable the user to access the FCS R-objects within a particular data directory of the *rfcdmin* and other data packages.

We have saved and archived the objects using the "save" function with option "compress=TRUE" in the base library will keep the extension as .RData (ie, NOT RData.gz) eventhough these files were compressed. Table 3 summarizes

	R-object name	FCS raw binary file	HIV-protein stimulated?	HIV status
VRC	unst.1829	1829-28+49d.fcs	no	- (negative control)
.	st.1829	1829-GAG.fcs	yes, GAG-protein	- (negative case)
..	unst.DRT	DRT-28+49d.fcs	no	+ (positive control)
...	st.DRT	DRT-GAG.fcs	yes, GAG-protein	+ (positive case)
FHCRC	MC.053	042404c1.053.fcs	no	- (negative control)
....	MC.054	042404c1.054.fcs	yes, CEF-protein	contact source
.....	MC.055	042404c1.055.fcs	yes, SEB-protein	contact source

Table 3: Summary of FCS R-objects in 'rfcdorig' data package

the two archived FCS R-object datasets in *rfcdorig* data package. The *rfcdmin* data package contains only the VRC data and the MC.053 individual of the FHCRC data and can be retrieved by the following data call:

```
> data(VRCmin)
> data(MC.053min)
```

Alternatively, other FCS R objects that have not been archived in the /data directory of either data package "rfcdorig" and "rfcdfhcrc2" can be currently read in using "read.FCS", by also using the `textbfdata(".FCSObj")` command, where ".FCSObj" is the character string of the FCS R object filename. The ".FCSObj" can then be accessible from the current workspace.

For the *rfcdmin* package, the `textbfdata(".FCSObjmin")` command must be used in order to obtain the `facscan256.FCSObj` in the current workspace :

```
> data("facscan256.FCSObj.currentmin")
```

```
Start/Stop:$version
```

```
[1] "FCS2.0"
```

```
$TextStart
```

```
[1] 256
```

```
$TextEnd
```

```
[1] 1595
```

```
$TextLength
```

```
[1] 1340
```

```
$DataStart
```

```
[1] 1792
```

```
$DataEnd
```

```
[1] 51791
```

```

$DataLength
[1] 50000

$AnalysisStart
[1] 0

$AnalysisEnd
[1] 0

[1] "Obtaining the metadata."
[1] "$BYTEORD 4,3,2,1"
[1] "file endian is big"
[1] "platform endian is little"
[1] "FCS object endian is swap"
[1] "Reading in data"
[1] "Checking the size and range of data"
[1] "Data passes range and size check"
[1] "Constructing data.frame from N= 50000"
[1] "Get names"
NOTE: Long names $PnS are missing.
Short names $PnN are assigned to the dataset instead.
[1] "Assigning 5 names to dataframe dims 10000 x 5"
[1] "Constructing return value"

```

To see other ".FCSRobj" or ".FCSRobjmin" filenames, the user can implement one of the following commands :

```
> data(package = "rfcdmin")
```

## 1.2 Other S4 class R-objects

Besides the already defined "FCS" class, other S4 class R-objects include "FCSmetadata", "FCSsummary", and "FCSgate". The "FCS" class is the class of all ".fcs" files that are read into R using "read.FCS". The "FCSmetadata" is the class of the metadata slot of an "FCS" R-object. The "FCSsummary" class is the class of the output of the summary method implemented on a FCS R-object. The "FCSgate" class contains the "FCS" class and extends it to include gating information (ie, the information about the indexing of row observations for subsequent extraction).

The following is a brief summary of the available, generic methods associated with each class object.

### 1.2.1 "new" Generic Method

Default objects can be made by using the **new(object-contents, S4-class-name)** method, where object-contents refer to the contents of each slot for the

specified S4-class-name.

The following commands produce default class objects with no slot information:

```
> new.FCS <- new("FCS")
> new.FCSmetadata <- new("FCSmetadata")
> new.FCSsummary <- new("FCSsummary")
> new.FCSgate <- new("FCSgate")
```

### 1.2.2 "coercian" Generic Method

Currently, there are only coercian methods to and from the "FCS" class and the "matrix" and "data.frame" classes.

A user makes a FCS R-object by the coercian method "as" exemplified in the following code, where data2 is a matrix or data.frame identifying the rows as the cell observations and the columns as the different variables:

```
> data2 <- rbind(1:10, 2:11, 3:12)
> data2.matrix <- as(data2, "matrix")
> data2.df <- as.data.frame(data2)
> test.FCSObj <- as(data2.matrix, "FCS")
> test.FCSObj2 <- as(data2.df, "FCS")
> original.matrix <- as(test.FCSObj2, "matrix")
> original.matrix <- as(test.FCSObj2, "data.frame")
> metadata <- new("FCSmetadata", size = dim(data2)[1], nparam = dim(data2)[2],
+   fcsinfo = list(comment = "This is a pseudo FCS-R object.))
> test.FCSObj@metadata <- metadata
> test.FCSObj
```

```
Original Object of class `FCS' from: None
Object name: None
Dimensions 3 by 10
```

### 1.2.3 "is" Generic Method

The S4 R-object class can be verified by using the "is" method.

```
> is(MC.053, "matrix")

[1] FALSE

> is(MC.053, "FCS")

[1] TRUE

> is(MC.053@metadata, "FCSmetadata")

[1] TRUE
```

```
> is(MC.053, "FCSgate")
```

```
[1] FALSE
```

The FCSsummary class is exemplified below:

```
> sum.FCS <- summary(MC.053)
```

#### I. Data reports:

A. Dimension Check: Dimensions: (row X col): 126795 X 7

B. Data Column Names & Univariate Summary:

Using Tukey's method for the five number summary

	column	min	lower-hinge	median	upper-hinge	max	mean	sd
Forward Scatter	1	0	430	504	687	1023	568.045	196.049
Side Scatter	2	0	70	94	275	1023	185.913	185.470
IFNgamma FITC	3	0	194	236	308	955	246.718	81.965
CD69 PE	4	0	291	353	422	1023	357.339	108.511
CD8 PerCP	5	0	111	186	286	986	237.867	187.510
<NA>	6	0	0	0	0	1023	0.906	9.737
CD3 APC	7	0	276	740	806	1023	571.022	288.345

#### II. Metadata Variable/Slot reports:

A. Metadata Slots:

	slotnames	description	values
1	mode	Mode	L
2	size/\$TOT	number of cells/rows	126795
3	nparam/\$PAR	number of column params	7
4	shortnames/\$PnN	Shortnames of column parameters	see below
5	longnames/\$PnS	Longnames of column parameters	see below
6	paramranges/\$PnR	Ranges/max of column parameters	see below
7	filename	original FCS filename	042402c1.053.fcs
8	objectname	name of current object	MC.053
9	original	current object original status	TRUE
10	fcsinfo	misc. metadata info	see part II B.

\$ColumnParametersSummary

	\$PnN	\$PnS	\$PnR
[1,]	"FSC-H"	"Forward Scatter"	"1024"
[2,]	"SSC-H"	"Side Scatter"	"1024"
[3,]	"FL1-H"	"IFNgamma FITC"	"1024"
[4,]	"FL2-H"	"CD69 PE"	"1024"
[5,]	"FL3-H"	"CD8 PerCP"	"1024"
[6,]	"FL1-A"	NA	"1024"
[7,]	"FL4-H"	"CD3 APC"	"1024"

B. Metadata 'fcsinfo' slot length= 103 & slot names:

```
$fcsinfoNames
 [1] "$BYTEORD"          "$DATATYPE"          "$NEXTDATA"
 [4] "$SYS"              "CREATOR"            "$P1B"
 [7] "$P1E"              "$P2B"               "$P2E"
[10] "$P3B"              "$P3E"               "$P4B"
[13] "$P4E"              "$P5B"               "$P5E"
[16] "$P6B"              "$P6E"               "$P7B"
[19] "$P7E"              "PATIENT ID"         "SAMPLE ID"
[22] "$CYT"              "CYTNUM"             "$BTIM"
[25] "$ETIM"             "BD$AcqLibVersion"  "BD$NPAR"
[28] "BD$P1N"            "BD$P2N"             "BD$P3N"
[31] "BD$P4N"            "BD$P5N"             "BD$P6N"
[34] "BD$P7N"            "BD$WORD0"           "BD$WORD1"
[37] "BD$WORD2"          "BD$WORD3"           "BD$WORD4"
[40] "BD$WORD5"          "BD$WORD6"           "BD$WORD7"
[43] "BD$WORD8"          "BD$WORD9"           "BD$WORD10"
[46] "BD$WORD11"         "BD$WORD12"          "BD$WORD13"
[49] "BD$WORD14"         "BD$WORD15"          "BD$WORD16"
[52] "BD$WORD17"         "BD$WORD18"          "BD$WORD19"
[55] "BD$WORD20"         "BD$WORD21"          "BD$WORD22"
[58] "BD$WORD23"         "BD$WORD24"          "BD$WORD25"
[61] "BD$WORD26"         "BD$WORD27"          "BD$WORD28"
[64] "BD$WORD29"         "BD$WORD30"          "BD$WORD31"
[67] "BD$WORD32"         "BD$WORD33"          "BD$WORD34"
[70] "BD$WORD35"         "BD$WORD36"          "BD$WORD37"
[73] "BD$WORD38"         "BD$WORD39"          "BD$WORD40"
[76] "BD$WORD41"         "BD$WORD42"          "BD$WORD43"
[79] "BD$WORD44"         "BD$WORD45"          "BD$WORD46"
[82] "BD$WORD47"         "BD$WORD48"          "BD$WORD49"
[85] "BD$WORD50"         "BD$WORD51"          "BD$WORD52"
[88] "BD$WORD53"         "BD$WORD54"          "BD$WORD55"
[91] "BD$WORD56"         "BD$WORD57"          "BD$WORD58"
[94] "BD$WORD59"         "BD$WORD60"          "BD$WORD61"
[97] "BD$WORD62"         "BD$WORD63"          "BD$LASERMODE"
[100] "CalibFile"         "P7THRESVOL"         "$FIL"
[103] "$DATE"
```

```
> is(sum.FCS, "FCSsummary")
```

```
[1] TRUE
```

### 1.3 Descriptive tools for FCSmetadata class R-objects

For the rest of this manual we will use one example FCS R object "st.1829" to demonstrate the tools that are available in *rflowcyt*.

Some descriptive tools for the "FCSmetadata" class is **show**, **print**, and **summary**, which all can be exemplified in the following code:

```
> show(st.1829@metadata)
```

```
FACSmetadata for original FCS object: st.1829 from original file 1829_GAG.fcs  
with 126675 cells and 8 parameters.
```

The following code would output the metadata as a string and is not shown because of its lengthy output.

```
> summary(st.1829@metadata)
```

The slots for an FCSmetadata are summarized in Table 4. Alternative character indices include:

"\$TOT" "size"

"\$PAR" "nparam"

"\$PnN" "shortnames"

"\$PnS" "longnames"

"\$PnR" "paramranges"

	slotnames	description
1	mode	Mode
2	size	number of cells/rows
3	nparam	number of column parameters
4	shortnames	shortnames of column parameters
5	longnames	longnames of column parameters
6	paramranges	Ranges/Max value of the columns
7	filename	original FCS filename
8	objectname	name of the current object
9	original	current object original status
10	fcsinfo	misc. metadata info

Table 4: FCSmetadata slot descriptions

Slots and slot components of the metadata can be retrieved by using , "[i]", or "[[i]]". Currently to extract metadata information, we can use a single character string index being one of the slotnames in Table 4 or one of the slotnames in

the "fcsinfo" slot. In the case that there is a common slotname that is also in the "fcsinfo" slot, only the slot from Table 4 will be retrieved.

A single numeric index or a vector of numeric indices refers to only the slot positions of the "fcsinfo" slot.

The following example extracts the column parameter ranges or maximum value. A similar extraction can be implemented with "shortnames" and "longnames" extraction.

```
> rng1 <- st.1829@metadata@paramranges
> rng1 <- st.1829@metadata["paramranges"]
> rng1

[1] 1024 1024 1024 1024 1024 1024 1024 1024

> rng2 <- st.1829@metadata[["$PnR"]]
> rng2

[1] 1024 1024 1024 1024 1024 1024 1024 1024

> rng1.1 <- st.1829@metadata[["$P1R"]]
> rng1.1 <- st.1829@metadata[["$P1S"]]
```

Items in the FCSmetadata can be replaced by using "[...]<-" or "[[...]]<-".

The following example will replace the "longnames" with dummy names.

```
> st.1829@metadata["longnames"]

[1] "FSC-Height"          "Side Scatter"          "CD8 FITC"
[4] "IFN, IL2, TNF PE"   "CD4 perCP"             " "
[7] "CD3 APC"             "Time (204.80 sec.)"

> st.1829@metadata["longnames"] <- rep("dummy", length(st.1829@metadata["longnames"]))
> st.1829@metadata["$P3S"] <- "wrongname"
> st.1829@metadata["longnames"]

[1] "dummy"      "dummy"      "wrongname" "dummy"      "dummy"      "dummy"
[7] "dummy"      "dummy"
```

When using the replacement method for a FCSmetadata R-object (ie, "[...]<-" or "[[...]]<-"), if the slotname is not found, then a new slot with the current character index is made under the "fcsinfo" slot. In the following example, we will add a new slot named "newslot" to the metadata.

```
> st.1829@metadata[["newslot"]] <- "wow this is cool"
> st.1829@metadata@fcsinfo[["newslot"]]

[1] "wow this is cool"
```

## 1.4 Descriptive tools for FCS class R-objects

To access the "data" or the "metadata" components, use either the `metaData` or the tools **metaData** to extract the metadata component and **fluors** to extract the data component.

```
> meta1 <- st.1829@metadata
> meta1 <- metaData(st.1829)

> data1 <- st.1829@data
> data1 <- fluors(st.1829)
> summary(data1)
```

FSC.Height	Side.Scatter	CD8.FITC	IFN..IL2..TNF.PE
Min. : 125.0	Min. :167.0	Min. : 0.0	Min. : 0.0
1st Qu.: 339.0	1st Qu.:410.0	1st Qu.:155.0	1st Qu.:222.0
Median : 441.0	Median :473.0	Median :227.0	Median :285.0
Mean : 489.9	Mean :475.6	Mean :235.2	Mean :276.1
3rd Qu.: 680.0	3rd Qu.:543.0	3rd Qu.:274.0	3rd Qu.:340.0
Max. :1023.0	Max. :969.0	Max. :856.0	Max. :882.0

CD4.perCP	X.	CD3.APC	Time..204.80.sec..
Min. : 0.0	Min. : 0.0000	Min. : 0.0	Min. : 0.0
1st Qu.:121.0	1st Qu.: 0.0000	1st Qu.:199.0	1st Qu.:140.0
Median :264.0	Median : 0.0000	Median :272.0	Median :291.0
Mean :256.8	Mean : 0.8412	Mean :321.8	Mean :292.5
3rd Qu.:371.0	3rd Qu.: 0.0000	3rd Qu.:399.0	3rd Qu.:444.0
Max. :948.0	Max. :1023.0000	Max. :969.0	Max. :599.0

A set of descriptive tools are attached to the FCS R-object. The method "print" (using an "FCS" object in its signature) will automatically give a short summary of the FCS R-object without printing out all the contents of the data and the metadata. The following examples are different incantations of the "print" method for FCS objects:

```
> print(unst.1829)
```

```
Original Object of class `FCS' from: 1829_28+49d.fcs
Object name: unst.1829
Dimensions 197025 by 8
```

```
> print(MC.053)
```

```
Original Object of class `FCS' from: 042402c1.053.fcs
Object name: MC.053
Dimensions 126795 by 7
```

A longer and more detailed summary with statistics for the column variables can be displayed by using the "summary" method, whose output is a "FCSsummary" S4 class.

```
> out.sum <- summary(st.1829)
> print(out.sum)
```

To extract and replace slots of the metadata of a "FCS" object, use only "[..]" and "[...]<-" , respectively.

```
> shortnames.1829 <- st.1829[["shortnames"]]
> shortnames.1829
```

```
[1] "FSC-H" "SSC-H" "FL1-H" "FL2-H" "FL3-H" "FL2-A" "FL4-H" "Time"
```

```
> st.1829[["$PnR"]]
```

```
[1] 1024 1024 1024 1024 1024 1024 1024 1024
```

```
> st.1829[["$P1R"]] <- 0
```

```
> st.1829[["paramranges"]]
```

```
[1] 0 1024 1024 1024 1024 1024 1024 1024
```

```
> st.1829[["newslot"]]
```

```
[1] "wow this is cool"
```

```
> st.1829[["newslot"]] <- "this is even cooler"
```

```
> st.1829[["newslot"]]
```

```
[1] "this is even cooler"
```

To extract and replace components within the "data" matrix of a "FCS" object, use only "[..]" and "[...]<-" , respectively.

```
> firstten.1829 <- as(st.1829[1:10, ], "matrix")
```

```
> firstten.1829
```

	FSC-Height	Side Scatter	CD8 FITC	IFN, IL2, TNF PE	CD4 perCP	CD3 APC
1	341	408	154	238	232 0	532
2	690	564	265	371	255 0	313
3	335	455	562	128	106 0	744
4	367	550	165	283	113 0	240
5	190	495	219	334	107 0	284
6	441	414	194	229	159 0	339
7	144	443	199	296	0 0	261
8	730	509	257	344	366 0	247
9	542	480	243	337	278 0	326
10	305	463	61	113	472 0	563
Time (204.80 sec.)						
1	0					
2	0					

```

3           0
4           0
5           0
6           0
7           0
8           0
9           0
10          0

> firstobs.1829 <- as(st.1829[1, 1], "matrix")
> firstobs.1829

  FSC-Height
1          341

> st.1829[1, 1] <- 99999999
> as(st.1829[1, 1], "matrix")

  FSC-Height
1          1e+08

> st.1829[1, 1] <- firstobs.1829
> as(st.1829[1, 1], "matrix")

  FSC-Height
1          341

> st.1829[1, 1]

  Non-original Object of class `FCS' from: 1829_GAG.fcs
  Object name: st.1829
  Dimensions 1 by 1

```

Note that the "original" slot within the "metadata" is only changed to FALSE when the "data" is changed.

Changing the metadata itself will not alter the status of the "original" slot.

```

> st.1829[["original"]]

[1] FALSE

```

The function "dim.FCS" retrieves the dimensions of the "data" matrix of a FCS object.

```

> dim.1829 <- dim.FCS(st.1829)
> dim.1829

[1] 126675      8

```

A data parameter column can be appended to the "data" matrix of a "FCS" object by using the method "addParameter", which will also result in the change of the "original" metadata slot to be "FALSE".

```
> column.to.add <- rep(0, dim.1829[1])
> st.1829 <- addParameter(st.1829, colvar = column.to.add, shortname = "test",
+   longname = "example", use.shortname = FALSE)
```

## 1.5 Checking Validity of the FCS R-object and Fixing errors

The method "checkvars" checks the ranges, dimensions, and the column variable names of the data against what is specified in the metadata. If details are not specified in the metadata, then the available information is added to the metadata. The output is a boolean as to whether the object passes the check. The option, **MY.DEBUG=TRUE**, allows us to view the checking statements.

```
> st.1829.checkstat <- checkvars(st.1829, MY.DEBUG = TRUE)

[1] "Class is FCS"
[1] "Object has data"
[1] "Object has metadata"
[1] "Object has a name:st.1829"
[1] "Data Dimension Check: Dimensions: (row X col)"
[1] "   Data: (126675 X 9)"
[1] "   Metadata: (126675 X 9)"
[1] "Names Check:"
      Data Parameter Names st.1829@metadata@longnames
[1,] "FSC-Height"         "dummy"
[2,] "Side Scatter"      "dummy"
[3,] "CD8 FITC"          "wrongname"
[4,] "IFN, IL2, TNF PE" "dummy"
[5,] "CD4 perCP"         "dummy"
[6,] " "                  "dummy"
[7,] "CD3 APC"           "dummy"
[8,] "Time (204.80 sec.)" "dummy"
[9,] "example"           "example"
[1] "   st.1829@metadata@longnames do not match with that of the data."
[1] "Range Check: Column parameters are within specified metadata range."
      Data Ranges st.1829@paramranges
FSC-Height           1023           1023
Side Scatter          969           969
CD8 FITC              856           856
IFN, IL2, TNF PE     882           882
CD4 perCP             948           948
                     1023           1023
```

CD3 APC	969	969
Time (204.80 sec.)	599	599
example	0	0

```
> st.1829.checkstat
```

```
[1] FALSE
```

Because st.1829 has been altered such that there is a discrepancy between the "metadata" and the "data" portions of this FCS object, "fixvars" will be used to correct major errors.

```
> if (st.1829.checkstat == FALSE) {
+   st.1829 <- fixvars(st.1829, MY.DEBUG = TRUE)
+ }
```

```
[1] "Class is FCS"
[1] "Object has data"
[1] "Object has metadata"
[1] "Object has a name: st.1829"
[1] "Data Dimension Check: Dimensions: (row X col)"
[1] "   Data: (126675 X 9)"
[1] "   Metadata: (126675 X 9)"
[1] "Names Check:"
      Data Parameter Names st.1829@metadata@longnames
[1,] "FSC-Height"          "dummy"
[2,] "Side Scatter"       "dummy"
[3,] "CD8 FITC"           "wrongname"
[4,] "IFN, IL2, TNF PE"  "dummy"
[5,] "CD4 perCP"         "dummy"
[6,] " "                  "dummy"
[7,] "CD3 APC"           "dummy"
[8,] "Time (204.80 sec.)" "dummy"
[9,] "example"           "example"
[1] "   st.1829@metadata@longnames do not match with that of the data."
[1] "Names Fix: Replacement of the metadata parameter(s):"
      [,1]
[1,] "$P1S"
[2,] "$P2S"
[3,] "$P3S"
[4,] "$P4S"
[5,] "$P5S"
[6,] "$P6S"
[7,] "$P7S"
[8,] "$P8S"
[1] "   from the old name(s) of the original metadata:"
      [,1]
```

```

[1,] "dummy"
[2,] "dummy"
[3,] "wrongname"
[4,] "dummy"
[5,] "dummy"
[6,] "dummy"
[7,] "dummy"
[8,] "dummy"
[1] " to the following name(s) from the data:"
     [,1]
[1,] "FSC-Height"
[2,] "Side Scatter"
[3,] "CD8 FITC"
[4,] "IFN, IL2, TNF PE"
[5,] "CD4 perCP"
[6,] " "
[7,] "CD3 APC"
[8,] "Time (204.80 sec.)"
[1] "Range Check: Column parameters are within specified metadata range."
      Data Ranges st.1829@paramranges
FSC-Height          1023          1023
Side Scatter         969          969
CD8 FITC             856          856
IFN, IL2, TNF PE    882          882
CD4 perCP            948          948
                    1023          1023
CD3 APC              969          969
Time (204.80 sec.)  599          599
example              0           0

```

## 1.6 Equality between FCS objects

Two FCS objects can be checked for equality by using the "equals" method. The default check is to verify the equality of the the "metadata" (except for the "filename" and the "objectname") and all the elements of the "data".

```
> equals(st.1829, unst.1829)
```

```
[1] FALSE
```

The "check.filename" and "check.objectname" set to TRUE will allow the equality verification of the "filename" and "objectname" slots in the "metadata".

```
> equals(st.1829, st.1829, check.filename = TRUE, check.objectname = TRUE)
```

```
[1] TRUE
```

## 1.7 Extraction of the FCS R-object

The original FCS R-object can be retrieved by using the function "get", if the original object is on the current workspace and has been unchanged. Alternatively, the original FCS R-object can be obtained by reading in the binary, fcs file from the /fcs directory (if this raw binary file exists) of the data package "rfcdmin".

```
> st.1829 <- get(st.1829[["objectname"]])
> original.FC.FCSRobj <- read.FCS(FC.FCSRobj[["filename"]], MY.DEBUG = FALSE)
```

NOTE: Long names \$PnS are missing.  
Short names \$PnN are assigned to the dataset instead.

## 2 Flow Cytometry Visualizations

In this section, we include visualization tools that help analyze the multivariate flow cytometry data. Because each cell has multiple immunofluorescence and light scatter measurements, we have made alternatives to visualize, beyond the ordinary bivariate scatterplots, the cell distributions based on the different measurements. The common approach in the field circumvents the visualization of data on all variables by selecting a subset of "interesting" cells by a sequential progression of 1 and 2 dimensional gating steps. Gating refers to the selection of a region of cells or observations in a bivariate or univariate plot by placing boundaries around the region. These boundaries or thresholds based on a particular immunofluorescence or light scatter measurement are referred to as **gates**. The sequence of gating steps is based on certain pairs of measurements or individual measurement, in which the gated region in a previous step is subsequently gated further in the next gating step. First we discuss the bivariate and multivariate plotting tools and then the gating tools.

### 2.1 Bivariate Plotting Tools

The basic bivariate plots are the **ContourScatterPlot** with hexagonal binning without contours or rectangular binning with superimposed contour levels and the **parallel.coordinates** plot which is either an **ImageParCoord** or a **JointImageParCoord** plot.

#### 2.1.1 ContourScatterPlot

The **plotvar.FCS** has the options of plotting specified variables from an FCS R-object. A univariate histogram or ContourScatterPlot with hexagonal binning or rectangular binning can be shown with the appropriate specified options. Here we will demonstrate with the FCS R object "unst.1829" the uses of **plotvar.FCS**.

```
> plotvar.FCS(unst.1829, varpos = 1)
```

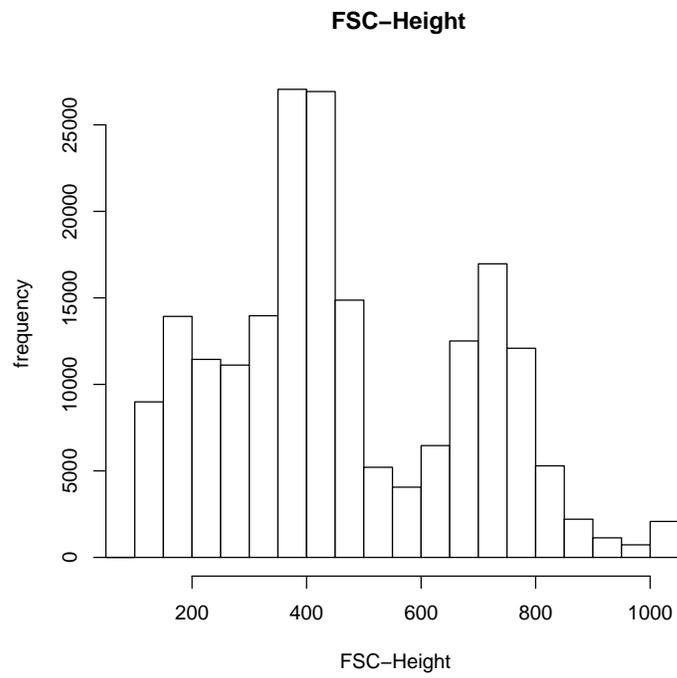


Figure 1: plotvar.FCS: Plotting a single variable histogram with the "unst.1829" FCS R object

```
> print(plotvar.FCS(unst.1829, varpos = c(3, 4)))
```

plotvar.FCS: Plotting a bivariate ContourScatterPlot with hexagonal binning with the "unst.1829" FCS R object. The Bioconductor hexbin package is needed for this type of plot. This works without Sweave, but fails within Sweave.

```
> plotvar.FCS(unst.1829, varpos = c(3, 4), hexbin.CSPlot = FALSE)
```

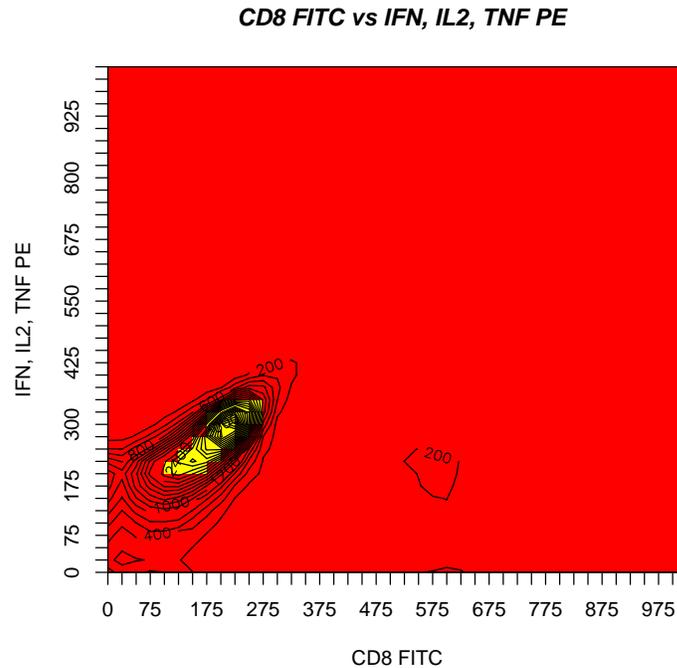


Figure 2: `plotvar.FCS`: Plotting a bivariate `ContourScatterPlot` with rectangular binning with the "unst.1829" FCS R object

The function **ContourScatterPlot** will make an image plot using rectangular bins of counts produced by the function `make.grid` by default. Also by default, there are superimposed contour levels that are also drawn on the plot with rectangular image binning. The "make.grid" function is used by the "ContourScatterPlot" function to make a count matrix for the number of observations in a two-dimensional grid layout. This function will output a matrix of counts ("z") as well as the total number of observations ("n.cells") within this matrix. The count matrix for the image plot has 25 unit cut-offs and can be changed by the "x.grid" and "y.grid" options. Alternatively, if there is a **status** or binary response variable for the data, other values such as the difference in counts, proportions, normalized proportions, and z statistics can be calculated by `make.density` for the rectangular bins of the image plot. Currently, a roughly estimated color legend is available for this rectangular binning with the `legend.CSP` function.

Alternatively, however, there is an option for hexagonal binning with an appropriate legend. Note that the Bioconductor "hexbin" package is necessary for this plot option. The hexagonal binning does not have superimposed contour

levels nor does it have the option to estimate other values besides counts in its bins.

We will demonstrate the use of **ContourScatterPlot** to make the same plots exemplified earlier with **plotvar.FCS**. These plots are not shown.

The following code extracts the third and the fourth column variables of the FCS R object "unst.1829".

```
> xvar <- as(unst.1829[, 3], "matrix")
> yvar <- as(unst.1829[, 4], "matrix")
```

The ContourScatterPlot function is implemented to make a plot with hexagonal binning and a legend. Other parameters such as binning style and number of bins can also be specified in the signature.

```
> ContourScatterPlot(xvar, yvar, xlab = unst.1829[["longnames"]][3],
+   ylab = unst.1829[["longnames"]][4], main = "Individual unst.1829",
+   hexbin.plotted = TRUE)
```

A plot can be made that has rectangular binning. The color of the image map (via the "image.col" option) can be changed as well as the size of the rectangular bins by "x.grid" and "y.grid" options. A legend can be displayed in a separate plot by setting the option "plot.legend.CSP" = TRUE.

```
> ContourScatterPlot(xvar, yvar, xlab = unst.1829[["longnames"]][3],
+   ylab = unst.1829[["longnames"]][4], main = "Individual 042402c1.053",
+   hexbin.plotted = FALSE, numlev = 25, image.col = heat.colors(15))
```

## 2.2 Multivariate Plots

The FCS R-object can be plotted using the generic "plot.FCS" or "plot" command which will make a pairs plot (by default) or a parallel coordinates plot. Here we show a default pairs plot using rectangular binning :

The same plot can be made using hexagonal binning; the code is shown, but the plot will not be displayed. **This is currently broken.**

Additional parameters for the pairsplot of a data matrix can be referenced by the **pairs.CSP** function. Currently a color legend can be plotted in the lower panels for **pairs.CSP** only for the rectangular binning. There is currently no legend available for pairs.CSP using hexagonal binning.

The parallel coordinates plot tracks each observation whose value is plotted on the vertical, y-axis through a series of variables on the horizontal, x-axis. The observation is tracked by a line from one variable to the next. The order of the column variables on the horizontal axis is the order that is presented in the input data matrix.

Here we make a parallel coordinates plot for the data portion of the "st.1829" FCS R-object. Because there are too many cell or row observations, we only show the first 10 observations in this parallel coordinates plot.

```
> plot(unst.1829)
```

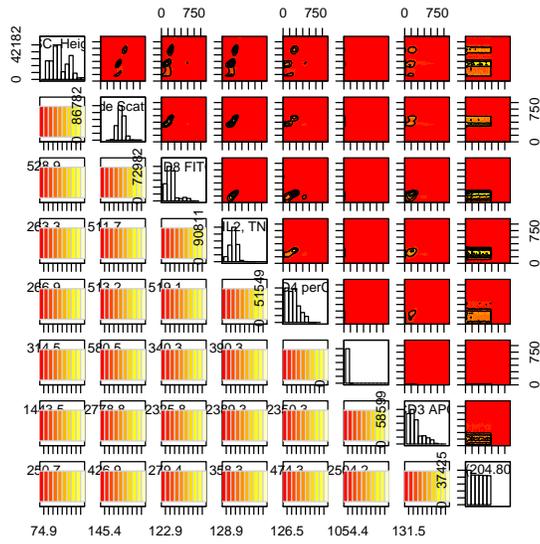


Figure 3: unst.1829: Default Pairs plotting with rectangular bins

```
> par(mfrow = c(1, 1))
> row.obs <- 1:10
> parallel.coordinates(as(unst.1829[row.obs, ], "matrix"))
```

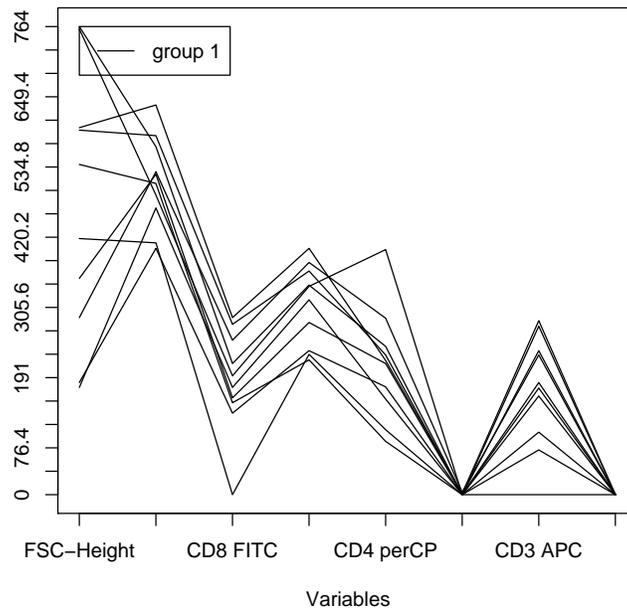


Figure 4: Parallel Coordinates plot of the first ten observations in the "data" of unst.1829.

It is important to note that all column variables in this plot must have the same range and scaling. We can force scaling on a [0,1] scale by using the option "scaled" set to TRUE. We can also give group certain observations by color ("group.col"), type ("group.lty"), and width ("group.lwd") of line. New observations can also be added at a time by setting "superimpose" to be TRUE or by using the function "add.parallel.coordinates". The following example shows these other options:

```
> row.obs <- 1:10
> parallel.coordinates(as(unst.1829[row.obs, ], "matrix"), scaled = TRUE,
+   group = c(rep(1, 5), rep(2, 5)))
```

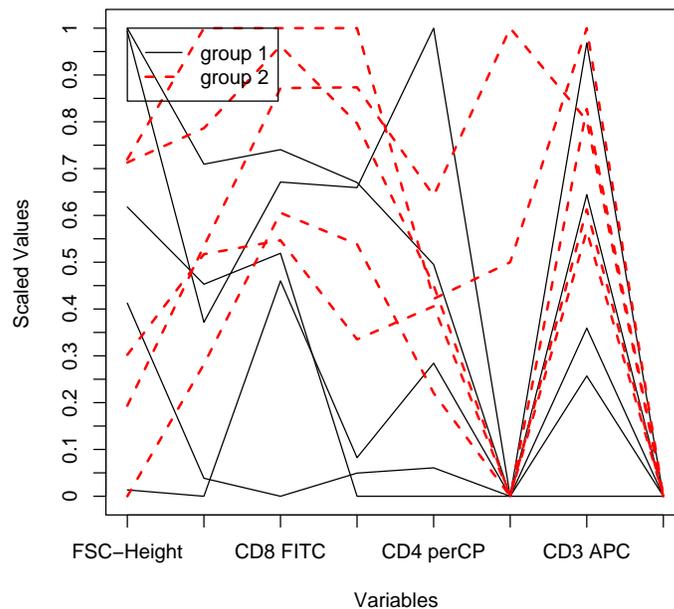


Figure 5: Scaled Parallel Coordinates plot of the first ten observations in the data of unst.1829, where the first 5 observations are in one group, and the next five observations are in the second group.

Because there are many cell or row observations, an ImageParCoord or JointImageParCoord plot can be used to show all of the row observations by binning on the y-axis and having the different column variables as labels on the x-axis. There are superimposed parallel.coordinates lines on the colored binning that demonstrate the movement of observations from one bin of one variable to another bin of the next variable. In an ImageParCoord, these lines represent moves

only between two adjacent variables, and in a JointImageParCoord, the lines represent movement among all of the variables. The plots are subject to change with the ordering of the column variables as labels on the x-axis of the plots.

Additional parallel.coordinates lines can be added to any existing plot using the **add.parallel.coordinates** function.

The following series of graphs exemplify the Image parallel.coordinates plots. Only the first 5 column variables and the first 1000 observations will be shown.

```
> output1 <- ImageParCoord(unst.1829@data[1:1000, 1:5], num.bins = 16,  
+   title = "1000 obs 16 bins 5 trans", ntrans = 5, legend.plotted = TRUE,  
+   image.plotted = TRUE, lines.plotted = TRUE, MY.DEBUG = FALSE)
```

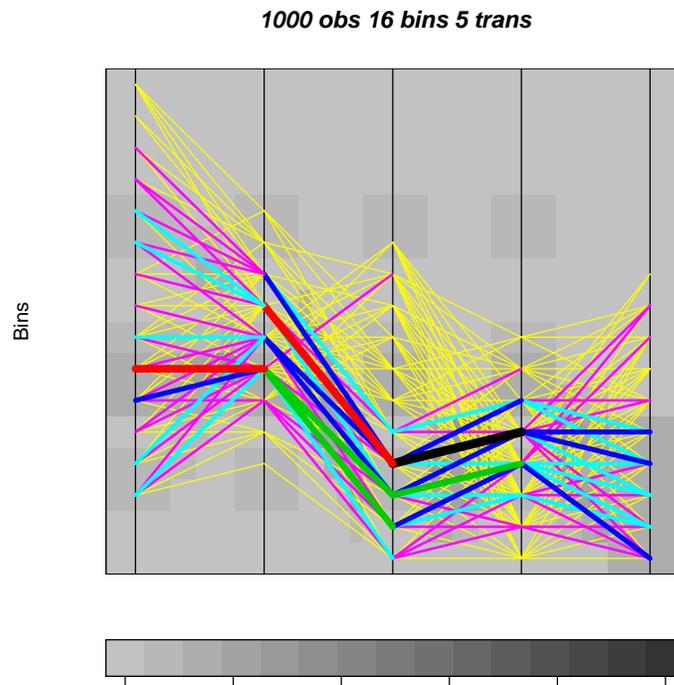


Figure 6: This plot is an Image Parallel Coordinates plot of the first 1000 observations and the first 5 column variables in the "data" of unst.1829.

```
> output2 <- ImageParCoord(unst.1829@data[1:1000, 1:5], num.bins = 16,  
+   title = "1000 obs 16 bins 5 trans", ntrans = 5, legend.plotted = TRUE,  
+   image.plotted = FALSE, lines.plotted = FALSE, MY.DEBUG = FALSE)
```

**Key for counts signified by the line**

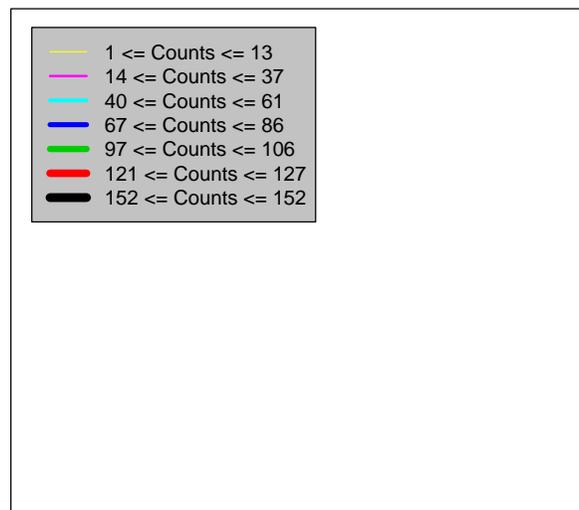


Figure 7: This plot is the legend for the lines of the Image Parallel Coordinates plot of the first 1000 observations and the first 5 column variables in the "data" of unst.1829. Each line color and width represents the number of observations that have moved from one bin of one variable to the adjacent bin of the next variable. The lines only represent movement between two variables.

The functions **ImageParCoord** and **JointImageParCoord** can also plot histograms and traditional parallel coordinates plots as diagnostics in addition to or separately from the image parallel coordinates plots when the option **MY.DEBUG=TRUE**.

```
> output3 <- JointImageParCoord(unst.1829@data[1:1000, 1:5], num.bins = 16,
+   title = "1000 obs 16 bins 5 trans", ntrans = 5, legend.plotted = FALSE,
+   MY.DEBUG = FALSE)
```

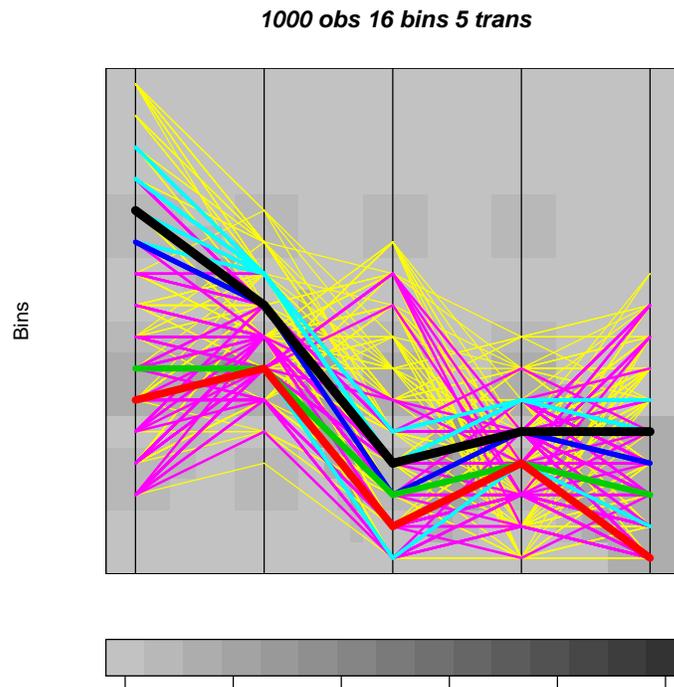


Figure 8: This is a Joint Image Parallel Coordinates plot of the first 1000 observations and the first 5 column variables in the "data" of unst.1829.

### 2.3 Dynamic Plotting Tools

Another multi-dimensional tool is "xgobi.FCS" which uses the "xgobi" library. We will leave the example for the user because the tool is interactive. Generally, by default **xgobi.FCS** will show the first 15 observations across all variables in the input data of the FCS R-object in a high-level multi-dimensional plot, in which the user is able to shift among sets of variables, color certain observations, and rotate visual perspectives of these observations amongst these variables. The function "xgobi.FCS" allows the user to input the FCS R-object, subset amongst the row observations, and subset amongst the column variables to

```
> output4 <- JointImageParCoord(unst.1829@data[1:1000, 1:5], num.bins = 16,  
+   title = "1000 obs 16 bins 5 trans", ntrans = 5, legend.plotted = TRUE,  
+   image.plotted = FALSE, lines.plotted = FALSE, MY.DEBUG = FALSE)
```

**Key for counts signified by the line**

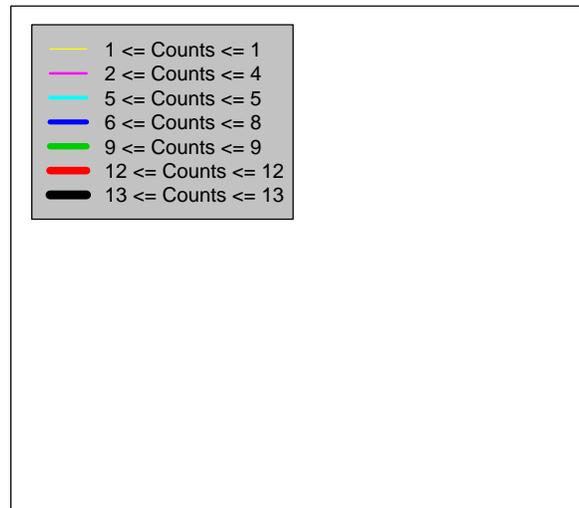


Figure 9: This legend is for the previous Joint Image Parallel Coordinates plot of the first 1000 observations and the first 5 column variables in the "data" of unst.1829. Each different line represents a different number of cells that have moved from bin to bin across all the variables. The lines represent the movement of cells across all variables jointly. An extra black line was added by the function "add.parallel.coordinates" function. This function can also be used with "ImageParCoord" and "parallel.coordinates" plots to add additional lines.

show in an "xgobi" plot. Currently "ggobi" S4 objects are still being constructed and would extend "xgobi" with more dynamic plotting and subsetting features.

The example code for the S3 "xgobi.FCS" is shown below but is left for the user to run separately. By default, only the first 15 rows and half of the column variables are shown. If "subset.row" and "subset.col" are specified, then these rows and columns will be displayed for the user to view interactively. In the second example, the first 6000 rows with the first 2 column variables are shown.

```
> xgobi.FCS(unst.1829, title = "unst.1829 default subset")
> xgobi.FCS(unst.1829, subset.row = 1:6000, subset.col = 1:2, title = "unst.1829: 6000 rows,
```

### 3 Gating

	slotnames	description
1	gate	matrix of column indices for row selection
2	history	vector of strings describing columns in gate
3	extractGatedData.msg	vector of strings describing extraction of the data
4	current.data.obs	vector of the original row positions in current data
5	data	matrix of column variables for rows denoting cells
6	metadata	FCSmetadata object

Table 5: FCSgate slot descriptions

	slotnames	description
1	uniscut	univariate single cut
2	bipcut	bivariate polygonal cut
3	bidcut	bivariate double cut
4	biscut	bivariate single cut
5	biscut.quadrant	values denoting the quadrant to be selected
6		+/, +/+, +/+, -/+, +/-

Table 6: Types of Gating

The **FCSgate** class extends the S4 FCS class. The slots of the S4 FCSgate class are summarized in Table 5. There are three aspects to gating that are summarized below:

**Create Gating Index** Initially, a gating index will be created. This binary index will denote the selection of row observations in the "data" and will be appended as a column to the "gate" matrix. The extension of the FCS object to a FCSgate object results from the S4 methods **createGate** and **icreateGate**, an interactive method with user prompts for option values.

	slotnames	description
1	gateNum	column position in 'gate' matrix
2	gateName	name of gate index
3	type	type of gating
4	biscut.quadrant	quadrant selected, if gating type is 'biscut'
5	data.colpos	'data' column variable positions used in gating
6	data.colnames	'data' names of the column variables used in gating
7	IndexValue.In	value of the gating index denoting inclusion
8	gatingrange	vector of gating thresholds
9	prev.gateNum	gateNum of previous gating, if any
10	prev.gateName	gateName of previous gating, if any
11	comment	comment by user for this gating index

Table 7: Description of 'extractGateHistory' output: Gating Details

Table 6 summarizes the types of gates or cuts that can be used to select the data. Currently, there are only gates involving one (ie, univariate) or two (ie, bivariate) column variables of the "data". A "single" or "double" cut refers to the number of thresholds for each variable. For an example, if there is a "bidcut", then there are two thresholds for each of the two variables. The group of observations lying within these bivariate thresholds are chosen. In the bivariate polygonal cut "bipcut", the selection ranges describe a polygonal shape which could be a square or any other closed linear shape description.

**Extract Gated Data** In order to collapse the "data" given the row selection index, the method **extractGatedData** will subset the "data" according to a specific value of the selection index (ie, IndexValue.In) and to a particular column in the "gate" matrix. Information about the extraction will be updated in the corresponding element of the "extractGatedData.msg" vector. The "metadata" will also be updated in terms of row "size" and the "original" flag will be set as FALSE. The "current.data.obs" will also be subset according to the selection index. In summary, the S4 method **extractGatedData** handles "data" collapsing with a corresponding row selection index of a FCSgate class object.

**Extract Gating Information** The **extractGateHistory** will output a list of values and details of a particular gating index. Table 7 summarizes the descriptions of the gating information that is extracted.

The following subsections exemplify the creation of a gating or selection binary index, the extraction or subsetting of the "data" using this newly created gating index, the extraction of gating details, a description of bivariate gating schemes, and other gating functions for high-dimensional plots.

See 5.1 for details about subsequent analyses after gating (Roederer and Hardy, 2001).

### 3.1 Creating Gate Index

Using `createGate` or the interactive `icreateGate` will result in a binary index that will be appended to the gate matrix. We will use the FCS R-object `unst.1829` for a following demonstration of gating.

First a bivariate double cut gate will be implemented and will capture the observations between 300 and 600 of the FSC-Height, first column variable of "data", and the Side Scatter, second column variable of "data".

```
> gate.range.x <- c(300, 600)
> gate.range.y <- c(300, 600)
> unst.1829.gate1 <- createGate(unst.1829, varpos = c(1, 2), gatingrange = c(gate.range.x,
+   gate.range.y), type = "bidcut", comment = "first gate")
```

In order to see the gate, we use "plotvar.FCS" and "showgate.FCS".

Currently, the "showgate.FCS" does not work with "plotvar.FCS" with the "hexbin.CSPlot=TRUE" option. The following is a hexbin ContourScatterPlot of the complete data before extraction on the created gate. Note that the gating thresholds are not shown.

```
> par(mfrow = c(1, 1))
> data.vars <- 1:2
> plotvar.FCS(unst.1829.gate1, varpos = data.vars, plotType = "ContourScatterPlot",
+   hexbin.CSPlot = TRUE)
```

(Again, Sweave errors cause the above not to work here).

The gate for the can be shown with the original data with the following code:

```
> data.vars <- 1:2
> plotvar.FCS(unst.1829.gate1, varpos = data.vars, plotType = "ContourScatterPlot",
+   hexbin.CSPlot = FALSE)
> showgate.FCS(unst.1829.gate1@data[, data.vars], gatingrange = c(gate.range.x,
+   gate.range.y), Index = unst.1829.gate1@gate[, 1], type = "bidcut",
+   pchtype = ".")
```

Alternatively, the corresponding `icreateGate` could be implemented that would make a plot and prompt the user for information about the type of gate desired. If parameters such as the type of gate and the `gatingrange` are known before looking at the data, these options can be input into `icreateGate`, and the plot will be shown.

The following plot and implementation describes the use of setting a univariate single cut gate for selection of cells that are  $\geq 500$  in value for the 4th "data" column variable from those selected by the first gate. The previous gate is the first column of "gate" and the selection value is 1 (ie, `prev.gateNum = 1` and `prev.IndexValue.In = 1`). Setting "prompt.all.options" to FALSE will suppress other interactive prompts for the title and gating color of the plot.

For a completely interactive gating session, the user can implement `icreateGate` on a FCS R-object and input all plotting and gating options after each prompt.

```

> unst.1829.gate2 <- icreateGate(unst.1829.gate1, varpos = 4, gatingrange = 500,
+   type = "uniscut", prev.gateNum = 1, prev.IndexValue.In = 1,
+   comment = "", MY.DEBUG = FALSE, prompt.all.options = FALSE)

[1] " plotvar.FCS: Making univariate histogram; Please Wait..."

```

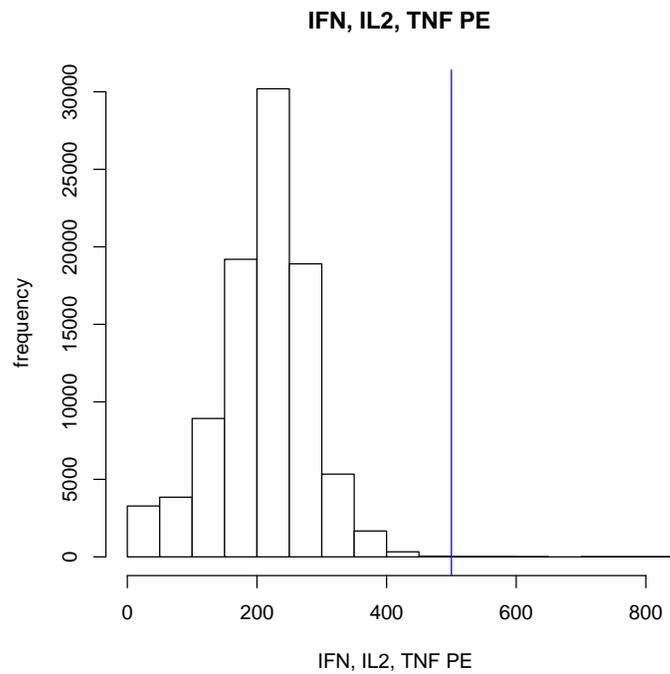


Figure 10: unst.1829: The gating index for fourth column variable of the data is shown. The row observations beyond the vertical gate of 500 of uniscut are selected with an IndexValue.In=1.

```
> icreateGate(unst.1829)
```

### 3.2 Data Extraction from Gate Index

The extraction or row subsetting of the "data" matrix corresponding to a gating index is implemented by **extractGatedData**.

The following extraction the "data" will use the first gating index (ie, the first column of the "gate" matrix specified with gateNum=1) and the selection value of 1 (ie, selection of observations with IndexValue.In=1).

```
> unst.1829.subset1.1 <- extractGatedData(unst.1829.gate2, gateNum = 1,
+   IndexValue.In = 1, MY.DEBUG = FALSE)
> unst.1829.subset1.2 <- extractGatedData(unst.1829.gate1, gateNum = 1,
+   IndexValue.In = 1, MY.DEBUG = FALSE)
```

Both the "unst.1829.gate1" and "unst.1829.gate2" are FCSgate objects with the same "data" but different "gate" matrices. The generic method "equals" will only evaluate the equality of the "data" and "metadata" slots and not of the "gate" matrix for FCSgate objects.

```
> equals(unst.1829.subset1.1, unst.1829.subset1.2, check.filename = FALSE,
+   check.objectname = FALSE)
```

```
[1] TRUE
```

Extraction using the second column index of the "gate" matrix (ie, gateNum=2) and selecting those with IndexValue.In=1 could be implemented on either a previously extracted FCSgate object or the FCSgate object without extraction. The output "unst.1829.subset2.1" and "unst.1829.subset2.2" should have the same "data" and "metadata" slots evaluated by "equals".

```
> unst.1829.subset2.1 <- extractGatedData(unst.1829.subset1.1,
+   gateNum = 2, IndexValue.In = 1, MY.DEBUG = FALSE)
> unst.1829.subset2.2 <- extractGatedData(unst.1829.gate2, gateNum = 2,
+   IndexValue.In = 1, MY.DEBUG = FALSE)
> equals(unst.1829.subset2.1, unst.1829.subset2.2, check.filename = FALSE,
+   check.objectname = FALSE)
```

```
[1] TRUE
```

### 3.3 Extraction of Gating Details from "history"

The use of **extractGateHistory** extracts information for a particular gate index. The list output provides an easy way to access the information that can be used as input for the functions "createGate", "icreateGate", and "extractGatedData" in subsequent gating implementations.

The extraction of gating information before gated data extraction is shown in the for gates 1 and 2.

```

> info.gate1 <- extractGateHistory(unst.1829.gate2, gateNum = 1)
> info.gate1

$gateNum
[1] 1

$gateName
[1] ""

$type
[1] "bidcut"

$biscut.quadrant
NULL

$data.colpos
[1] 1 2

$data.colnames
[1] "FSC-Height" "Side Scatter"

$IndexValue.In
[1] 1

$gatingrange
[1] 300 600 300 600

$prev.gateNum
[1] NA

$prev.gateName
[1] NA

$comment
[1] "first gate"

> info.gate2 <- extractGateHistory(unst.1829.gate2, gateNum = 2)
> info.gate2

$gateNum
[1] 2

$gateName
[1] "uniscut.v4"

$type
[1] "uniscut"

```

```
$biscut.quadrant
NULL
```

```
$data.colpos
[1] 4
```

```
$data.colnames
[1] "IFN"
```

```
$IndexValue.In
[1] 1
```

```
$gatingrange
[1] 500
```

```
$prev.gateNum
[1] 1
```

```
$prev.gateName
[1] ""
```

```
$comment
[1] ""
```

The extraction of gating information after implementing "extractGatedData" provides the following output for gates 1 and 2, respectively:

```
> info.gate1.1 <- extractGateHistory(unst.1829.subset2.1, gateNum = 1)
> info.gate1.1
```

```
$gateNum
[1] 1
```

```
$gateName
[1] ""
```

```
$type
[1] "bidcut"
```

```
$biscut.quadrant
NULL
```

```
$data.colpos
[1] 1 2
```

```
$data.colnames
```

```

[1] "FSC-Height" "Side Scatter"

$IndexValue.In
[1] 1

$gatingrange
[1] 300 600 300 600

$prev.gateNum
[1] NA

$prev.gateName
[1] NA

$comment
[1] "first gate"

> info.gate2.1 <- extractGateHistory(unst.1829.subset2.1, gateNum = 2)
> info.gate2.1

$gateNum
[1] 2

$gateName
[1] "uniscut.v4"

$type
[1] "uniscut"

$biscut.quadrant
NULL

$data.colpos
[1] 4

$data.colnames
[1] "IFN"

$IndexValue.In
[1] 1

$gatingrange
[1] 500

$prev.gateNum
[1] 1

```

```
$prev.gateName  
[1] ""
```

```
$comment  
[1] ""
```

Suppose the next gate is a bivariate double cut on the 5th and 6th column variables of the "data" matrix. If this gate is implemented from the previous first gate, then this extracted information "info.gate1" is used as well as "info.gate1.1" to identify the previous gating information (ie, "previous.gateNum" and "previous.IndexValue.In" in the example).

```
> gate.range.x <- c(200, 300)  
> gate.range.y <- c(100, 500)  
> previous.gateNum <- info.gate1$gateNum  
> previous.IndexValue.In <- info.gate1$IndexValue.In  
> unst.1829.gate3 <- createGate(unst.1829.gate2, varpos = c(1,  
+ 2), gatingrange = c(gate.range.x, gate.range.y), type = "bidcut",  
+ prev.gateNum = previous.gateNum, prev.IndexValue.In = previous.IndexValue.In,  
+ comment = "first gate")  
> extractGateHistory(unst.1829.gate3, gateNum = 3)
```

```
$gateNum  
[1] 3
```

```
$gateName  
[1] "bidcut.v1v2"
```

```
$type  
[1] "bidcut"
```

```
$biscut.quadrant  
NULL
```

```
$data.colpos  
[1] 1 2
```

```
$data.colnames  
[1] "FSC-Height" "Side Scatter"
```

```
$IndexValue.In  
[1] 1
```

```
$gatingrange  
[1] 200 300 100 500
```

```

$prev.gateNum
[1] 1

$prev.gateName
[1] ""

$comment
[1] "first gate"

```

Subsequent data extraction can be made on the FCSgate object "unst.1829.gate3" using "extractGatedData" given a particular gate index column in the "gate" matrix.

### 3.4 Gating Schemes

The "FHCRC.HVTNFCS" and the "VRC.HVTNFCS" are functions that implement "icreateGate" and "extractGatedData" as example gating procedures (Roederer and Hardy, 2001).

The user will be prompted for gating and plotting input with the following examples and associated FCS R objects (shown and not demonstrated).

```

> MC.053.gt <- FHCRC.HVTNFCS(MC.053)
> MC.054.gt <- FHCRC.HVTNFCS(MC.054)
> MC.055.gt <- FHCRC.HVTNFCS(MC.055)
> st.1829.gt <- VRC.HVTNFCS(st.1829)
> unst.1829.gt <- VRC.HVTNFCS(unst.1829)
> st.DRT.gt <- VRC.HVTNFCS(st.DRT)
> unst.DRT.gt <- VRC.HVTNFCS(unst.DRT)

```

If the user decides to implement one of the example gating schemes on his or her own FCS R object, the column variable positions can be adjusted for each gate implementation such that the variables to be gated may remain the same. The following example shows that for gate 2, column variable positions 7 and 5 refer to cd3 and cd8, respectively for that "data" matrix of "MC.053", the FCS object to be gated. Likewise, column variable positions that correspond to cd69 and INFgamma are 4 and 3.

```

> data(MC.053min)
> MC.053[["longnames"]]
> FHCRC.HVTNFCS(MC.053, gate2.vars = c(7, 5), gate3.vars = c(4,
+      3))

```

### 3.5 Other Image Gating

There are other gating procedures that can be implemented on high-dimensional plots. The **gate.IPC** interactive function allows the user to click on upper and lower bin boundaries for a particular variable to subset. The subsequent graphs

represent this subset of points that move from one variable to the next. The following code will be left for the user to implement as an exercise.

```
> st.DRT2 <- st.DRT
> st.DRT2@data <- st.DRT@data[1:1000, ]
> gate.IPC(st.DRT2, 3, hist.plotted = FALSE, image.plotted = TRUE,
+   para.plotted = FALSE, lines.plotted = TRUE, MY.DEBUG = FALSE)
```

Currently, there is still work in progress to gate on the dynamic plots **ggobi** and **xgobi**. See Section 2.3 for basic plotting usage.

## 4 Exploratory Data Analysis

The user may decide to use more qualitative means to investigate the data. The Patient Rule Induction Method (PRIM) allows the extraction of **rules** defined as subsets that maximizes or minimizes a target function which is usually specified as the mean of a binary label (Friedman and Fisher, 1998). In the flow cytometry setting, this target function is the mean of binary HIV-protein stimulated ( $Y=1$ ) or unstimulated status ( $Y=0$ ) for a particular immunofluorescence data subset or box, which ultimately estimates a rule through iterative trimmings of the box in the greedy, top-down Peeling Step and iterative additions into the box during the patient Expansion Step. A Cross-Validation Step implements the same Peeling and Expansion Steps on Testdata Sets. Hence, the estimated rules aim at finding distributional differences between the HIV-protein stimulated and unstimulated cells in a multi-dimensional setting where many different immunofluorescence measurements are made on the same sample of cells from an individual in an HIV vaccine trial. Again, the results of PRIM are only exploratory because it is a qualitative process that needs subjective, sound judgments to arrive at conclusions for each step of PRIM. PRIM is regarded as a tool for hypothesis generation rather than for inference.

Please refer to the "PRIM.pdf" manual in the *rfcprim* package for details regarding the functions used on the "data" component of the FCS R-objects.

## 5 Flow Cytometry Statistical Testing and Inference

The testing tools in this section are used to evaluate differences between HIV-protein stimulated and unstimulated scenarios, particularly in the IFN $\gamma$  measurement after gating described by Roederer and Hardy (2001).

Each subsection describes particular tests that are implemented by **run-flowcytests** and other functions.

## 5.1 Probability Binning

The current S3-class object **ProbBin.FCS** describes the equal probability binning of a univariate, immunofluorescence measurement (usually of IFN-gamma) after the implementation of a series of gating schemes across different immunofluorescence measurements. *Equal probability binning* ensures that there are equal number of observations, **N**, within a bin across all bins constructed by cut-offs or integer breakpoints of the immunofluorescence measurement. The final bin may contain more or less than **N**, the pre-specified number within each bin. The function, **breakpoints.ProbBin.FCS**, makes the breakpoints or cut-offs for equal probability binning in two ways:

**combined** based on the combination of the univariate distributions (usually of INF-gamma) of both the HIV-protein stimulated and unstimulated samples of cells

**by.control** based on only the unstimulated HIV-protein sample. These breakpoints are then used to make histogram objects from both the HIV-protein stimulated and unstimulated cell samples from an individual (Roederer and Hardy, 2001).

	slotnames	description
1	unst.hist	unstimulated histogram
2	st.hist	stimulated histogram
3	PB	'combined'/'by.control'
4	N.in.bin	number per bin for cut-off construction
5	varname	name of distribution/variable

Table 8: Description of 'ProbBin.FCS' S3 list output

The **ProbBin.FCS** object is a S3 list of the following components in Table 8

We will construct two gated objects as described in Section 3. The stimulated gated object is "st.DRT.gt" and the unstimulated gated object is "unst.DRT.gt". Here we will only gate on the bivariate double cut that extracts the lymphocytes from the Forward Scatter and Side Scatter measurements. Then we will extract the "IFN-gamma" measurement from each sample and then construct a ProbBin.FCS object.

The following implements a "biscut" gate and plots the image with the gate.

We could choose to implement subsequent gates; each gate that is dependent on the selection of a previous gate. We leave further gating as an exercise for the user. Below is an extraction of the data from the cd3+ lymphocytes (ie, from the second gate of cd3+ cells based on the selection of lymphocytes in the first gate).

```
> unst.DRT.ex <- extractGatedData(unst.DRT.gt, gateNum = 2)
> st.DRT.ex <- extractGatedData(st.DRT.gt, gateNum = 2)
```

```
> unst.DRT.gt <- icreateGate(unst.DRT, varpos = c(1, 2), gatingrange = c(300,
+   650, 300, 500), type = "bidcut", comment = "", MY.DEBUG = FALSE,
+   prompt.all.options = FALSE)
```

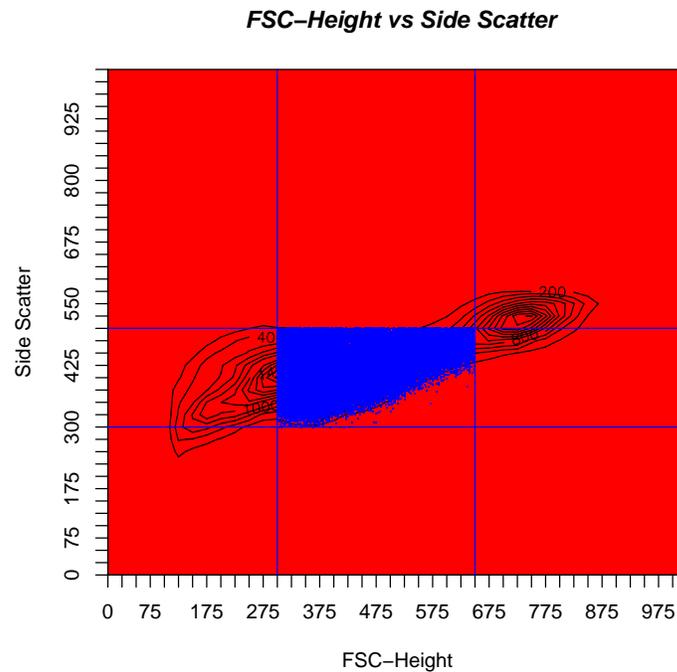


Figure 11: unst.DRT.gt: The gating index for first two column variables of the data is shown for the selection of the central cluster of lymphocytes. The colored points in the center of the bidcut are selected with an IndexValue.In = 1.

```
> st.DRT.gt <- icreateGate(st.DRT, varpos = c(1, 2), gatingrange = c(300,
+   650, 300, 500), type = "bidcut", comment = "", MY.DEBUG = FALSE,
+   prompt.all.options = FALSE)
```

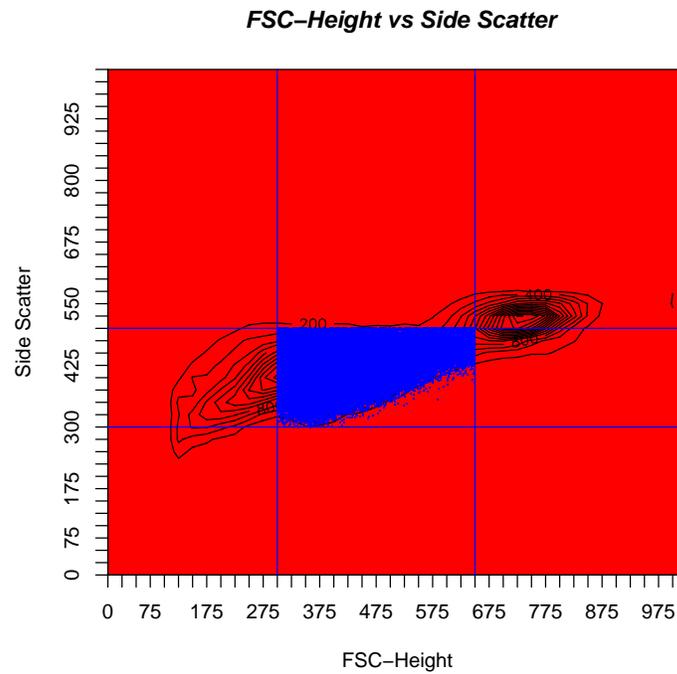


Figure 12: `st.DRT.gt`: The gating index for first two column variables of the data is shown for the selection of the central cluster of lymphocytes. The colored points in the center of the bidcut are selected with an `IndexValue.In=1`.

```

> unst.DRT.gt <- icreateGate(unst.DRT.gt, varpos = c(7, 5), gatingrange = c(500,
+   1024, 0, 1024), type = "bidcut", prev.gateNum = 1, prev.IndexValue.In = 1,
+   comment = "", MY.DEBUG = FALSE, prompt.all.options = FALSE)

```

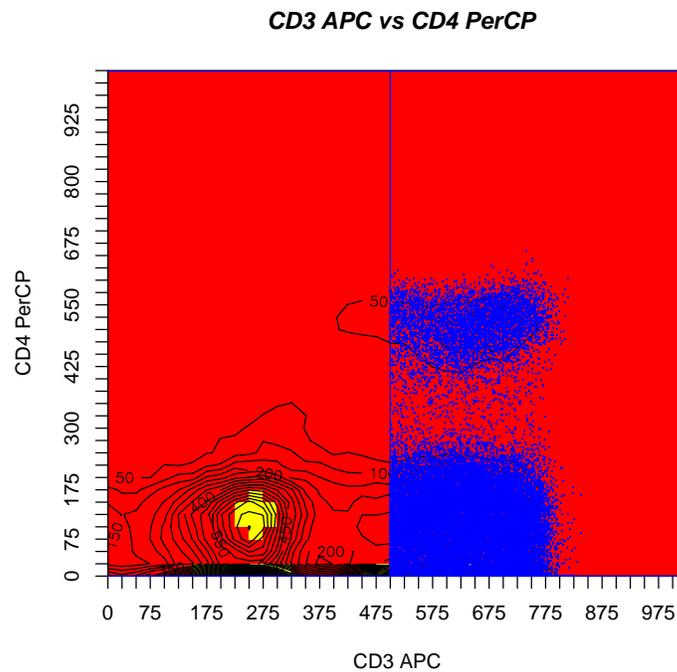


Figure 13: unst.DRT.gt: The gating index for 7th and 5th column variables of the data is shown for the selection of cd3+ cells based on the previous gating and selection of lymphocytes (ie, prev.gateNum=1, prev.IndexValue.In=1). The colored points of the bidcut gate are selected with an IndexValue.In = 1.

```

> st.DRT.gt <- icreateGate(st.DRT.gt, varpos = c(7, 5), gatingrange = c(500,
+   1024, 0, 1024), type = "bidcut", prev.gateNum = 1, prev.IndexValue.In = 1,
+   comment = "", MY.DEBUG = FALSE, prompt.all.options = FALSE)

```

**CD3 APC vs CD4 PerCP**

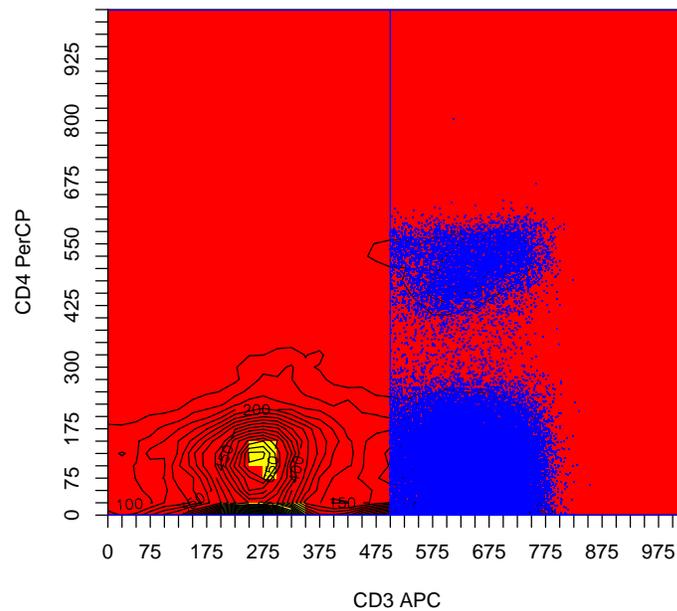


Figure 14: `st.DRT.gt`: The gating index for the 7th and 5th column variables of the data is shown for the selection of `cd3+` based on the previous gating and selection of lymphocytes (ie, `prev.gateNum=1`, `prev.IndexValue.In = 1`). The colored points the bidcut gate are selected with an `IndexValue.In = 1`.

We decide to analyze the IFN-gamma distribution among the selected cells. We obtain this measurement, IFN.unst and IFN.st, from the HIV-protein unstimulated and stimulated samples of individual DRT, respectively.

```
> IFN.unst <- unlist(as(unst.DRT.ex[, 4], "matrix"))
> IFN.st <- unlist(as(st.DRT.ex[, 4], "matrix"))
```

These two distributions are used to implement probability binning "by.control" with 100 observations in each bin based on the control, unstimulated group:

```
> PB.by.control <- ProbBin.FCS(IFN.unst, IFN.st, 100, varname = unst.DRT[["longnames"]][4],
+   PBspec = "by.control", MY.DEBUG = FALSE)
```

Alternatively, these two IFN distributions could have been used to implement probability binning constructed by the "combined" data having 100 observations in each bin:

```
> PB.combined <- ProbBin.FCS(IFN.unst, IFN.st, 100, varname = unst.DRT[["longnames"]][4],
+   PBspec = "combined", MY.DEBUG = FALSE)
```

To verify the "ProbBin.FCS" class objects, the following code using `is` can be used:

```
> is(PB.by.control, "ProbBin.FCS")
[1] TRUE
> is(PB.combined, "ProbBin.FCS")
[1] TRUE
```

We show the following "ProbBin.FCS" plots of the "PB.by.control" object.

The statistics associated with testing the two distributions for differences, assuming the null of no difference between the stimulated and unstimulated samples can be referenced in (Roederer et al., 2001; Baggerly, 2001). The summary of a "ProbBin.FCS" object will produce statistics that test the difference between the distributions of the stimulated and unstimulated samples. See Section 5.2.

```
> summary(PB.by.control)
```

Test of distribution difference: Probability Binning & PB metric

Null Hypothesis: Unstimulated/Control Data Histogram/Bins are the statistically the same as the Stimulated Data Histogram/Bins; both samples are from the same distribution

Alternative Hypothesis: Unstimulated/Control Data Histogram/Bins are significantly different from the Stimulated Data Histogram/Bins;

```
> plot(PB.by.control, plots.made = "unstimulated", freq = TRUE)
```

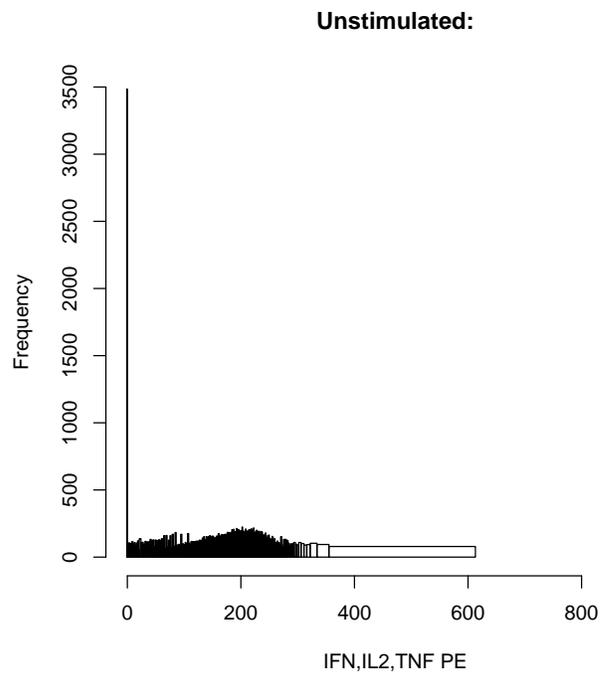


Figure 15: PB.by.control: The histogram shows the equal probability that was implemented on the unstimulated or control IFN $\gamma$  data. Here the counts in each bin are about 100

```
> plot(PB.by.control, plots.made = "stimulated", freq = TRUE)
```

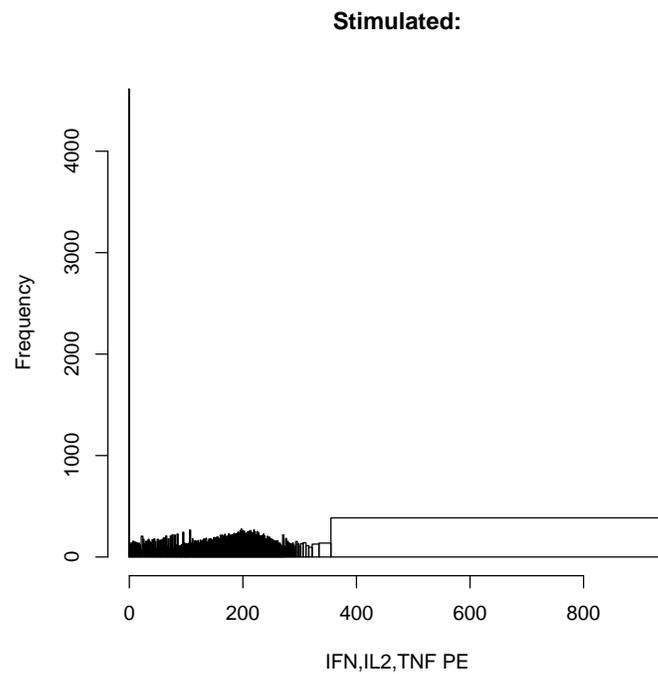


Figure 16: PB.by.control: The histogram shows the equal probability that was implemented on the unstimulated or control IFN $\gamma$  data of which whose breaks are applied to the stimulated data (which is shown in the above histogram). Here the counts in each bin can be shown setting the options `freq=TRUE` and `labels=TRUE`, which will prompt a warning because the binning is not equidistant.

the stimulated and unstimulated samples are from different distributions

Bins obtained from Probability binning with 100  
in each bin in the control dataset

Note: The counts in the first bin may be greater than 100  
because of abundance of zero data.

The counts in the bins are not shown because there are too many bins.

Number of Control: 38380

Number of Stimulated: 48304

Test1: T.chi.unadj

=max(0, (PBmetric-mean(PBmetric.unadj))/ SD(PBmetric.unadj)) statistic  
standard normal approximation test: Mario Roederer:  
unadjusted PB metric (PBmetric.unadj): 0.01033143  
Statistic used to assess significance of PB metric= max(0, unadjusted PB metric)  
= (T.chi.unadj): 8.388503  
one-sided p value (p.val.1tail.z.unadj): 2.461853e-17  
two-sided p value (p.val.2tail.z.unadj): 4.923706e-17

Test2: Adjusted PB metric statistic chi-squared test: Keith A. Baggerly:

adjusted PB metric (PBmetric.adj): 441.916

degrees of freedom (PB.df): 260

upper tail p value (p.val.1tail.chi.adj): 1.357756e-11

Test3: Adjusted T.chi.unadj standard normal approximation test: Keith A. Baggerly:

Adjusted T.chi.unadj (T.chi.adj): 7.977543

one-sided p value (p.val.1tail.z.adj): 7.46373e-16

two-sided p value (p.val.2tail.z.adj): 1.492746e-15

Test4: Pearson's Chi-Squared Test:

Pearson's Chi-squared test

data: cbind(c.i, s.i)

X-squared = 432.4261, df = 260, p-value = 9.778e-11

upper tail p value when df= 260 : pearson.p.val.PBdf= 9.778019e-11

> summary(PB.combined)

Test of distribution difference: Probability Binning & PB metric

Null Hypothesis: Unstimulated/Control Data Histogram/Bins are the  
statistically the same as the Stimulated Data Histogram/Bins;  
both samples are from the same distribution

Alternative Hypothesis: Unstimulated/Control Data Histogram/Bins  
are significantly different from the Stimulated Data Histogram/Bins;  
the stimulated and unstimulated samples are from different distributions

Bins obtained from Probability binning with 100  
in each bin in the combined (control & stimulated) dataset  
Note: The counts in the first bin may be greater than 100  
because of abundance of zero data.  
The counts in the bins are not shown because there are too many bins.  
Number of Control: 38380  
Number of Stimulated: 48304

Test1: T.chi.unadj  
=max(0, (PBmetric-mean(PBmetric.unadj))/ SD(PBmetric.unadj)) statistic  
standard normal approximation test: Mario Roederer:  
unadjusted PB metric (PBmetric.unadj): 0.01259746  
Statistic used to assess significance of PB metric= max(0, unadjusted PB metric)  
= (T.chi.unadj): 9.209693  
one-sided p value (p.val.1tail.z.unadj): 1.635294e-20  
two-sided p value (p.val.2tail.z.unadj): 3.270588e-20

Test2: Adjusted PB metric statistic chi-squared test: Keith A. Baggerly:  
adjusted PB metric (PBmetric.adj): 538.8427  
degrees of freedom (PB.df): 318  
upper tail p value (p.val.1tail.chi.adj): 1.289152e-13

Test3: Adjusted T.chi.unadj standard normal approximation test: Keith A. Baggerly:  
Adjusted T.chi.unadj (T.chi.adj): 8.756982  
one-sided p value (p.val.1tail.z.adj): 1.002741e-18  
two-sided p value (p.val.2tail.z.adj): 2.005483e-18

Test4: Pearson's Chi-Squared Test:

Pearson's Chi-squared test

data: cbind(c.i, s.i)  
X-squared = 522.0144, df = 318, p-value = 4.044e-12

upper tail p value when df= 318 : pearson.p.val.PBdf= 4.044095e-12

## 5.2 Testing for the difference between two univariate distributions

This section describes the tools used to test for the difference between the HIV-protein stimulated sample and the HIV-protein unstimulated sample in terms of the distribution of an immunofluorescence measurement and, in particular, of

the IFN-gamma measurement. There have been four main testing approaches that are outlined below. The null hypothesis is the assumption that both samples originate from the same distribution (ie, there is no difference in two distributions), and the alternative is that they are from different distributions (ie, the stimulated scenario compared to the unstimulated scenario are different in terms of cell densities).

**WLR.flowcytest** The weighted log rank test (by default when rho=0) tests the difference in survival curves of the stimulated and unstimulated scenarios when all measurements are regarded as having the "event" and "time" is considered to be the IFN-gamma or other immunofluorescence measurement. Thus, at every point on the immunofluorescence, the curves are tested for differences. A plot of the survival curves for both samples is also optionally output.

**KS.flowcytest** Kolmogorov-Smirnoff test also evaluates the difference in distributions for the control and the stimulated samples, but may be more sensitive and result in a higher false positive rate when there are a larger number of data points.

**ProbBin.flowcytest** Statistics proposed by Keith A. Baggerly and Mario Roederer include Chi-squared and Normal tests for the PB metric via probability binning (both based on the control data only ("by.control") and based on the combined dataset of both the stimulated and the control samples ("combined") (Roederer et al., 2001; Baggerly, 2001).

**pkci2.flowcytest** The method, proposed by Zoe Moodie, PhD, tests the difference of the upper tails of the two distributions rather than the range of the distribution for IFN-gamma or other univariate immunofluorescence measurement.

**runflowcytests** This function will run all of the aforementioned tests either separately or together in one call.

As a single example implementing all of the testing tools, we will only demonstrate the testing with the "runflowcytests". Further documentation for each individual test can be obtained in the help documentation for the following tests: "WLR.flowcytest", "KS.flowcytest", "ProbBin.flowcytest", "pkci2.flowcytest". Please note that "ProbBin.flowcytest" provides the same statistical output as "summary.ProbBin.FCS".

```
> output.runflowcytests <- runflowcytests(IFN.unst, IFN.st, KS.plotted = FALSE,  
+     WLR.plotted = FALSE, PBObj.plotted = FALSE)
```

FLOWCYTEST: Weighted Log Rank Test

```
experimental.status=0 (control)
```

experimental.status=1 (stimulated)

Call:

```
survdif(formula = Surv(fluorescence) ~ experimental.status,  
  data = my.dataframe, na.action = na.action.WLR, rho = rho.test)
```

	N	Observed	Expected	(O-E) <sup>2</sup> /E	(O-E) <sup>2</sup> /V
experimental.status=0	38380	38380	38094	2.15	3.93
experimental.status=1	48304	48304	48590	1.68	3.93

Chisq= 3.9 on 1 degrees of freedom, p= 0.0475

FLOWCYTEST: KOLMOGOROV-SMIRNOV

Two-sample Kolmogorov-Smirnov test

data: controldata and stimuldata  
D = 0.0178, p-value = 2.625e-06  
alternative hypothesis: two.sided

FLOWCYTEST: BAGGERLY & ROEDERER STATS

Number of observations in each bin: 100  
Dataset used for Probability Binning: by.control

Test of distribution difference: Probability Binning & PB metric

Null Hypothesis: Unstimulated/Control Data Histogram/Bins are the statistically the same as the Stimulated Data Histogram/Bins; both samples are from the same distribution  
Alternative Hypothesis: Unstimulated/Control Data Histogram/Bins are significantly different from the Stimulated Data Histogram/Bins; the stimulated and unstimulated samples are from different distributions

Bins obtained from Probability binning with 100  
in each bin in the control dataset

Note: The counts in the first bin may be greater than 100  
because of abundance of zero data.

The counts in the bins are not shown because there are too many bins.  
Number of Control: 38380

Number of Stimulated: 48304

Test1: T.chi.unadj  
=max(0, (PBmetric-mean(PBmetric.unadj))/ SD(PBmetric.unadj)) statistic  
standard normal approximation test: Mario Roederer:  
unadjusted PB metric (PBmetric.unadj): 0.01033143  
Statistic used to assess significance of PB metric= max(0, unadjusted PB metric)  
= (T.chi.unadj): 8.388503  
one-sided p value (p.val.1tail.z.unadj): 2.461853e-17  
two-sided p value (p.val.2tail.z.unadj): 4.923706e-17

Test2: Adjusted PB metric statistic chi-squared test: Keith A. Baggerly:  
adjusted PB metric (PBmetric.adj): 441.916  
degrees of freedom (PB.df): 260  
upper tail p value (p.val.1tail.chi.adj): 1.357756e-11

Test3: Adjusted T.chi.unadj standard normal approximation test: Keith A. Baggerly:  
Adjusted T.chi.unadj (T.chi.adj): 7.977543  
one-sided p value (p.val.1tail.z.adj): 7.46373e-16  
two-sided p value (p.val.2tail.z.adj): 1.492746e-15

Test4: Pearson's Chi-Squared Test:

Pearson's Chi-squared test

data: cbind(c.i, s.i)  
X-squared = 432.4261, df = 260, p-value = 9.778e-11

upper tail p value when df= 260 : pearson.p.val.PBdf= 9.778019e-11

FLOWCYTEST: BAGGERLY & ROEDERER STATS

Number of observations in each bin: 100  
Dataset used for Probability Binning: combined

Test of distribution difference: Probability Binning & PB metric

Null Hypothesis: Unstimulated/Control Data Histogram/Bins are the  
statistically the same as the Stimulated Data Histogram/Bins;  
both samples are from the same distribution  
Alternative Hypothesis: Unstimulated/Control Data Histogram/Bins  
are significantly different from the Stimulated Data Histogram/Bins;  
the stimulated and unstimulated samples are from different distributions

Bins obtained from Probability binning with 100  
in each bin in the combined (control & stimulated) dataset  
Note: The counts in the first bin may be greater than 100  
because of abundance of zero data.  
The counts in the bins are not shown because there are too many bins.  
Number of Control: 38380  
Number of Stimulated: 48304

Test1: T.chi.unadj  
=max(0, (PBmetric-mean(PBmetric.unadj))/ SD(PBmetric.unadj)) statistic  
standard normal approximation test: Mario Roederer:  
unadjusted PB metric (PBmetric.unadj): 0.01259746  
Statistic used to assess significance of PB metric= max(0, unadjusted PB metric)  
= (T.chi.unadj): 9.209693  
one-sided p value (p.val.1tail.z.unadj): 1.635294e-20  
two-sided p value (p.val.2tail.z.unadj): 3.270588e-20

Test2: Adjusted PB metric statistic chi-squared test: Keith A. Baggerly:  
adjusted PB metric (PBmetric.adj): 538.8427  
degrees of freedom (PB.df): 318  
upper tail p value (p.val.1tail.chi.adj): 1.289152e-13

Test3: Adjusted T.chi.unadj standard normal approximation test: Keith A. Baggerly:  
Adjusted T.chi.unadj (T.chi.adj): 8.756982  
one-sided p value (p.val.1tail.z.adj): 1.002741e-18  
two-sided p value (p.val.2tail.z.adj): 2.005483e-18

Test4: Pearson's Chi-Squared Test:

Pearson's Chi-squared test

data: cbind(c.i, s.i)  
X-squared = 522.0144, df = 318, p-value = 4.044e-12

upper tail p value when df= 318 : pearson.p.val.PBdf= 4.044095e-12

FLOWCYTEST: PKCI2

Test pkci2: Standard Normal approximation of two-sample binomial statistics

[1] "k.hat, 377 ,is the gate/percentile based on the control data"  
[1] " and the user specified critical proportion of, crit"  
[1] "0.00629 ,ps.hat is the proportion of stimulated data above k.hat"  
[1] "0.00099 , pc.hat is the proportion of the control data above k.hat,"

Null:  $H_0: \text{ps.hat} = \text{pc.hat}$  OR  $\text{ps.hat} - \text{pc.hat} = 0$   
One-sided Alternative:  $H_{1.1}: \text{ps.hat} - \text{pc.hat} > 0$  OR  $\text{ps.hat} > \text{pc.hat}$   
Two-Sided Alternative:  $H_{1.2}: \text{ps.hat} - \text{pc.hat} \neq 0$

Standard Normal Z Statistic: 13.4601429047812

One sided p-value: 1.34198315289743e-41

Two sided p-value: 2.68396630579485e-41

95 % Confidence Interval: ( 0.004531 , 0.006076 )

One sided Test: $H_{1.1}$  (1=reject  $H_0$ , 0=cannot reject  $H_0$ ): 1

Two sided Test: $H_{1.2}$  (1=reject  $H_0$ , 0=cannot reject  $H_0$ ): 1

The plots and output for the "KS.flowcytest" and the "WLR.flowcytest" are shown with the code on the following pages. The plots for the "Prob-Bin.flowcytest" is similar to those shown in Figure 15 and Figure 16.

```
> output.KSflowcytest <- KS.flowcytest(IFN.unst, IFN.st, KS.plotted = TRUE,  
+   MY.DEBUG = FALSE)
```

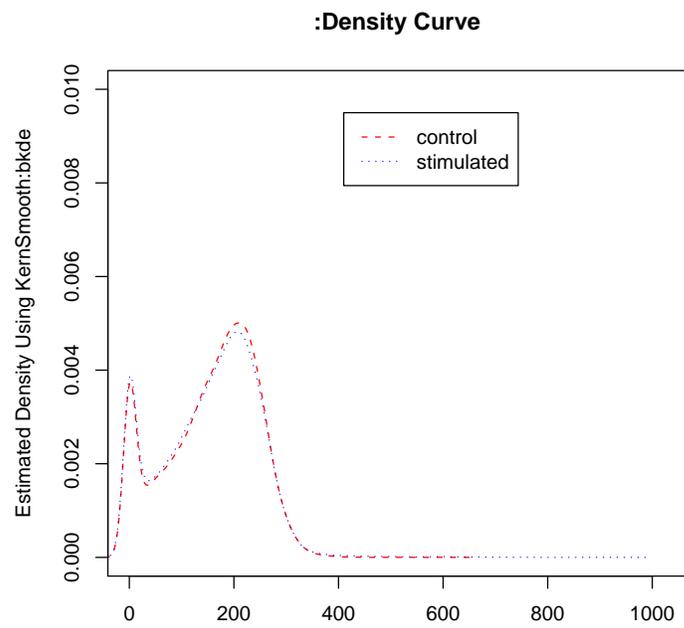


Figure 17: KS.flowcytest plot shows the distributions of the stimulated and unstimulated samples.

```
> output.WLRflowcytest <- WLR.flowcytest(IFN.unst, IFN.st, WLR.plotted = TRUE,
+     MY.DEBUG = FALSE)
```

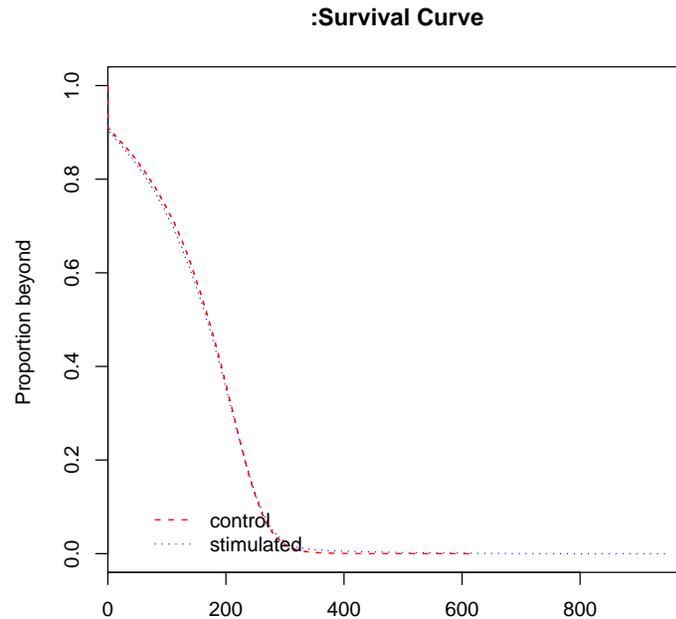


Figure 18: WLR.flowcytest plot shows the survival curves for the two distributions if every data point was regarded as being an event, and time was regarded as the IFN-gamma measurement.

### 5.3 ROC curves for testing tails of two distributions

For each individual there is a pair of data corresponding to a HIV-protein stimulated sample and a HIV-protein unstimulated/control sample. For each individual who is either HIV-positive or negative, the 99.9-th percentile for the unstimulated sample and the percent positive for the stimulated sample based on this control-based 99.9-th percentile was calculated. Here we exemplify the calculations for the "IFN.st" and the "IFN.unst" obtained from the gating for the HIV-negative individual 1829.

First, using "percentile.FCS", we obtain the 99.9-th percentile based on the control, unstimulated sample.

```
> unst.percentile <- percentile.FCS(IFN.unst, percent = 0.999)
```

Now using "PercentPos.FCS", we obtain the percent positives for both the

unstimulated and the stimulated samples, respectively, using the "unst.percentile". Note that the percent positive for the control sample is about 1 - 0.999.

```
> PercentPos.FCS(IFN.unst, percentile = unst.percentile)$percent.pos
```

```
[1] 0.001068265
```

```
> PercentPos.FCS(IFN.st, percentile = unst.percentile)$percent.pos
```

```
[1] 0.006417688
```

To evaluate which HIV-protein stimulation results in the most sensitive detection of HIV-positive status as well as the lowest chance of falsely concluding HIV-positive status based on a stimulated sample's higher 99.9th percentile control-based percent positive (ie, according to the approach used in "pcki2.flowcytest"). Zoe Moodie, PhD, constructed the ROC (Receiver Operating Characteristic) HIV-protein-specific curves in which the cut-offs are based on the combined stimulated and unstimulated percent positives obtained by the previous methods.

The "PerPosROCmin" data in the "rfcdmin" package exemplifies the percent positives obtained to plot the ROC curve.

Here we retrieve the example data provided by Zoe Moodie, PhD.

```
> data(PerPosROCmin, package = "rfcdmin")
```

The function "ROC.FCS" shows the ROC curve and sensitivity, specificity output after the implementation of the functions "percentile.FCS" and "PercentPos.FCS" to obtain the percentiles and the percent positives, respectively, for each individual's HIV-protein stimulated and unstimulated pair for a particular immunofluorescence measurement.

## 6 Future Updates

Most notable future updates include converting the testing and the gating into generic S4 class objects. Currently these objects are all S3.

The dynamic plotting functions will also be converted to S4 generic objects with additional visualization tools and methods.

Future work with PRIM include using the algorithm with real datasets and displaying the results with the tools provided in the "rflowcyt" package.

## References

Keith A. Baggerly. Probability binning and testing agreement between multivariate immunofluorescence histograms: Extending the chi-squared test. *Cytometry*, 2001.

```

> GAG <- ROC.FCS(hivpos.gag, hivneg.gag)
> POLA <- ROC.FCS(hivpos.pola, hivneg.pola, lineopt = 2, colopt = 2,
+   overlay = TRUE)
> POLB <- ROC.FCS(hivpos.polb, hivneg.polb, lineopt = 4, colopt = 3,
+   overlay = TRUE)
> legend(0.7, 0.7, c("gag", "polA", "polB"), col = c(1, 2, 3),
+   lty = c(1, 2, 4))

```

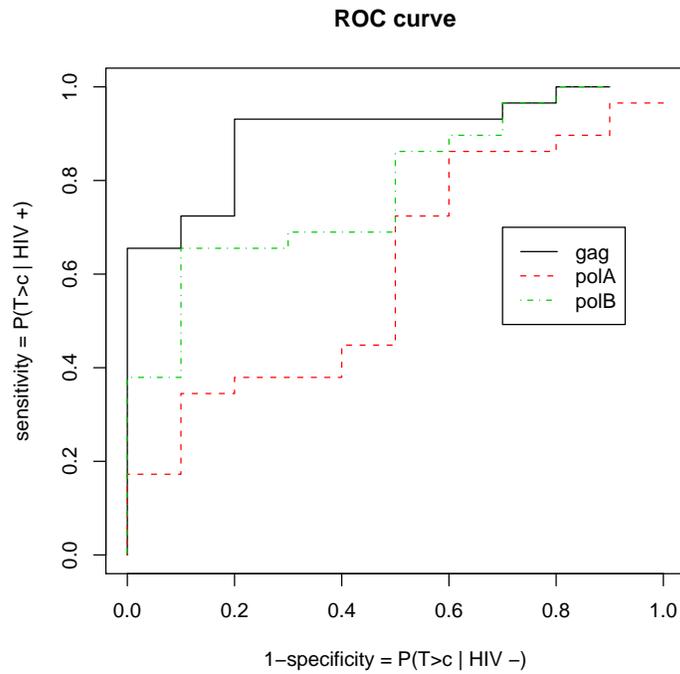


Figure 19: The ROC curves are based on the different HIV-proteins used for the stimulation of immune responses. Here the GAG appears to achieve greater sensitivity at a lower 1-specificity when evaluating the difference in immune responses between an HIV-infected and HIV-noninfected profiles using the `pkci2.flowcytest` approach.

- Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. Technical report, Stanford Statistics, 1998.
- J. Paul Robinson, editor. *Current Protocols in Cytometry*. John Wiley & Sons, Inc., 2001.
- Mario Roederer and Richard R. Hardy. Frequency difference gating: A multivariate method for identifying subsets that differ between samples. *Cytometry*, 2001.
- Mario Roederer, Adam Treister, Wayne Moore, and Leonore A. Herzenberg. Probability binning comparison: A metric for quantitating univariate distribution differences. *Cytometry*, 2001.