Supplemental Tutorial: Scalable Analysis of Flow Cytometry Data using R/Bioconductor

David J. Klinke II[1,2] and Kathleen M. Brundage[2]

[1]Department of Chemical Engineering
[2]Department of Immunology, Microbiology, & Cell Biology
West Virginia University
Morgantown, WV 25606

April 21, 2009

Contact Info:

david.klinke@mail.wvu.edu	Department of Chemical Engineering
Phone: (304)293-2111 ext 2432	West Virginia University
Fax: (304)293-4139	P.O. Box 6102
	Morgantown, WV 26506-6102

Introduction to Tutorial

The following section is an annotated version of the Results and Discussion section of the corresponding paper. The underlying R scripts are shown interspersed within the corresponding segments of text. As stated in the methods, the Flow cytometry data was analyzed using Bioconductor 2.2, a package implemented in R 2.7.2. The R scripts should also work in subsequent versions as they are released.

Results and Discussion (Annotated Version)

The workflow for data analysis in a typical flow cytometry experiment has evolved with recent technological advances [1] and can be grouped into two key steps, as summarized in Figure 1. First, a pre-processing step was required to ensure that the observed levels of fluorescence were independent and specific measures of the level of expression of the protein of interest, assuming that the antibodies also exhibit specificity. In this study, expression of CD4, CD44, and CD62L were used to characterize the efficiency of $CD4^+CD62L^+$ T cell isolation from Balb/c splenocytes using magnetic microbeads. The second step involved analysis of the cell populations including gating using statistically-based data-driven gates, estimating probability density functions using kernel marginalization, and clustering using principal component analysis.

Installation

Following installation of R [2], basic Bioconductor packages and additional packages that are required to process flow cytometry data were downloaded from the web within R using:

```
>source("http://www.bioconductor.org/biocLite.R")
>biocLite("flowCore")
>biocLite("flowViz")
>biocLite("flowUtils")
>biocLite("geneplotter")
>openVignette()
```

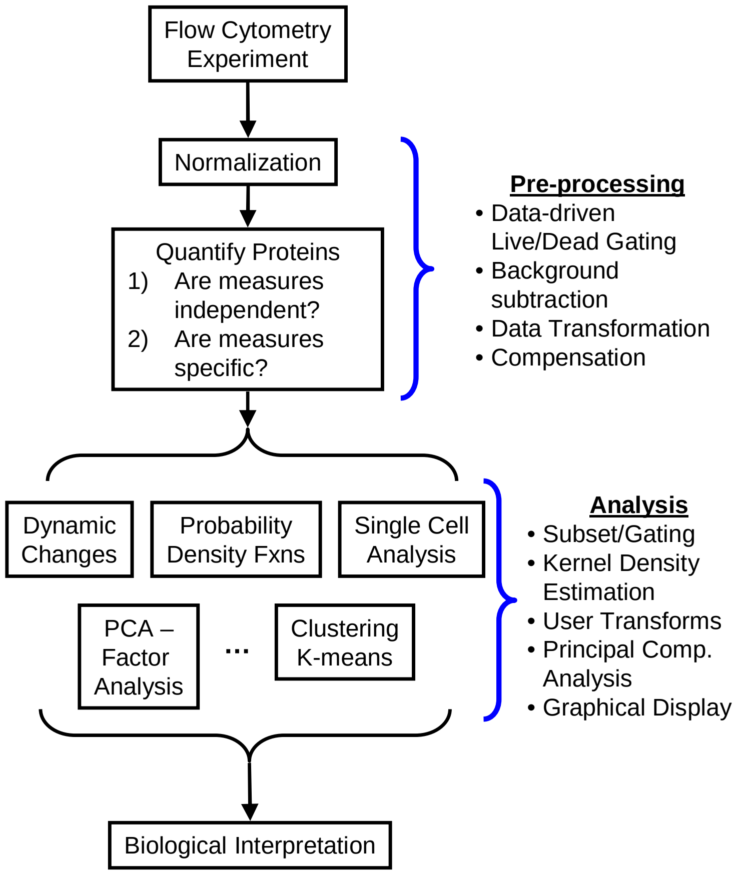

Figure 1: Overview of the steps associated with the use of flow cytometry as a tool in biological research. This manuscript will focus on how Bioconductor can be used during pre-processing and analysis steps.

Additional Bioconductor packages may also be downloaded directly from the website [3]. Additional R packages can be installed using a menu option in the RGui (see 'Packages'->'Install package(s)'). These files need to be downloaded and installed only once. Subsequent sessions can load the packages using `library("package")`.

```
> library(flowCore)
> library(flowViz)
> library(flowUtils)
> library(geneplotter)
> library(colorspace)
> library(grid)
```

Pre-processing

Data Entry

The experimental results were exported from the flow cytometer in FCS3.0 format [4](e.g., `foo.fcs`) following data acquisition. The default working directory is the installation directory for R. It may be more advantageous to change the directory to a working directory (see 'File'->'Change dir') where the `foo.fcs` files are stored. Following the definition of an array, `fclist`, that contains the names of the FCS3.0 files to be analyzed,

```
> fclist <- c("MACSpurity_Tube_001.fcs", "MACSpurity_Tube_002.fcs",
+     "MACSpurity_Tube_003.fcs", "MACSpurity_Tube_004.fcs",
+     "MACSpurity_Tube_005.fcs", "MACSpurity_Tube_006.fcs",
+     "MACSpurity_Tube_007.fcs", "MACSpurity_Tube_008.fcs",
+     "MACSpurity_Tube_009.fcs")
```

the data files were loaded into the R workspace using a single command:

```
> fs <- read.flowSet(fclist, transformation = FALSE)
```

A summary of the loaded `flowSet` can be shown by typing the variable name at the command line:

```
> fs

A flowSet with 9 experiments.

  column names:
  FSC-A SSC-A FITC-A PE-A APC-A Time
```

The information contained within a particular experimental data set (i.e., one fcs file) were read and stored in a `flowFrame`. A `flowFrame` is the name of a meta-object, a digital construct that collects different types of information (i.e., text and numerical data) into a common identifier. A series of `flowFrame`s can also be collected in a `flowSet`. Different functions (e.g., `phenoData()` or `exprs()`) can be used to extract information from these meta-objects, such as the measured fluorescent intensities of the different parameters for each cell and the time that each cell was observed. As the filenames were not descriptive, a list of title names, shown in Supplemental Tables 1 and 2, was created for use in subsequent figures.

```
> Tclist <- c("Pre-sort Unstained", "FITC-CD4 Single", "PE-CD44 Single",
+     "APC-CD62L Single", "Pre-sort Population", "CD4+ Subset",
+     "CD4- Subset", "CD4+CD62L+ Subset", "CD4+CD62L- Subset")
```

Gating on Forward Scatter and Side Scatter

Non-cellular debris and dead cells exhibit non-specific staining. These potentially confounding observations were eliminated by gating on forward scatter and side scatter to help ensure that the fluorescent measurements exhibit specificity for the target of interest. The gates associated with isolating live splenocytes were initialized as follows. First, cells were retained that had Forward Scatter areas between 50,000 and the maximum intensity using a 1-dimensional gate applied to the Forward Scatter parameter.

```
> rectGate <- rectangleGate(filterId = "FSC+", "FSC-A" = c(50000,Inf))
```

Second, the Forward Scatter and Side Scatter parameters were used to create a data-driven gate (`norm2Filter`) that was centered at the median of the specified cell populations in both dimensions and enclosed a region that included 95% of the population (i.e., 2 standard deviations).

```
> morphGate <- norm2Filter(filterId = "MorphologyGate", "FSC-A",
+     "SSC-A", scale = 2)
```

Additional data-driven gates can also be used (e.g., `kmeansFilter`, a data-driven filter that performs one-dimensional k-means clustering), especially for the subsequent analysis step. Further refinement of the gates was achieved by combining individual gates using logical arguments. The logical arguments are applied right to left and combined using ! (NOT), | (OR), and & (AND).

```
> PositiveGate <- morphGate & rectGate
> RejectGate1 <- !morphGate & rectGate
> RejectGate2 <- !rectGate
```

The gates were applied to the entire flowset, although they can also be applied to individual flowFrames.

```
> PosTFS <- Subset(fs, PositiveGate)
> RejTFS1 <- Subset(fs, RejectGate1)
> RejTFS2 <- Subset(fs, RejectGate2)
```

The statistics associated with gating were calculated to determine the number of cells retained for subsequent analysis (see Supplemental Table 1).

```
> Total <- as.numeric(fsApply(fs, nrow, use.exprs = TRUE))
> Live <- as.numeric(fsApply(PosTFS, nrow, use.exprs = TRUE))
> data1 <- data.frame(Files = Tclist, "Total Cells" = Total,
+     "Live Cells" = Live)
> data1 <- transform(data1, Percent = data1[, 3] * 100/data1[,2])
> tabS1 <- as.matrix(data1)

> library(xtable)
> xtable(tabS1, caption = "Number of cells retained for each sample
+     following gating.", label = "Tab:S1", align = c("l", "l", "r",
+     "r", "r"), digits = c(0, 0, 0, 1, 0))
```

	Files	Total.Cells	Live.Cells	Percent
1	Pre-sort Unstained	10000	6784	67.84
2	FITC-CD4 Single	10000	6355	63.55
3	PE-CD44 Single	10000	6508	65.08
4	APC-CD62L Single	10000	6301	63.01
5	Pre-sort Population	10000	6090	60.90
6	CD4+ Subset	10000	7501	75.01
7	CD4- Subset	10000	6150	61.50
8	CD4+CD62L+ Subset	10000	7141	71.41
9	CD4+CD62L- Subset	10000	7163	71.63

Table 1: Number of cells retained for each sample following gating.

The results of the gating on the forward scatter and side scatter characteristics of the splenocytes are shown in Supplemental Figure 2. The live cells are shown in blue using a contour overlay that indicates the density of the spots. Dot plots of the rejected cells were superimposed on the figures and shown in red.

```
> opar <- par(mfrow = c(2, 2), mar = c(4, 4, 2, 2))
> Ptxt = c("A", "B", "C", "D")
> for (i in 5:8) {
+     plot(PosTFS[[i]], c("FSC-A", "SSC-A"), xlab = "FSC", xlim = c(0,
+         262144), ylab = "SSC", ylim = c(0, 262144), nrpoints = 1000)
+     title(main = Ptxt[i - 4], outer = FALSE, adj = 0, cex.main = 2)
+     points(exprs(RejTFS1[[i]][, 1]), exprs(RejTFS1[[i]][, 2]),
+         pch = ".", col = "red")
+     points(exprs(RejTFS2[[i]][, 1]), exprs(RejTFS2[[i]][, 2]),
+         pch = ".", col = "red")
+ }
```

Compensating for Fluorescent Spillover

Given the difficulty of determining appropriate compensation values 'on-the-fly', the current generation of flow cytometers incorporate two advancements for the analysis of flow cytometry data. First, contemporary software drivers for flow cytometers include an algorithm for automatically calculating the fluorescent compensation matrix. Second, raw data is uncompensated providing the opportunity to adjust compensation values after data collection. The initial estimate for the compensation matrix was extracted from the text description of `MACSPurity_Tube_001.fcs`. This initial estimate of the compensation matrix was based upon prior experiments and was used to observe the data during acquisition. To illustrate compensation using R/Bioconductor, it was not optimized.

```
> spillM <- description(PosTFS[[1]])$SPILL
> spillM
```

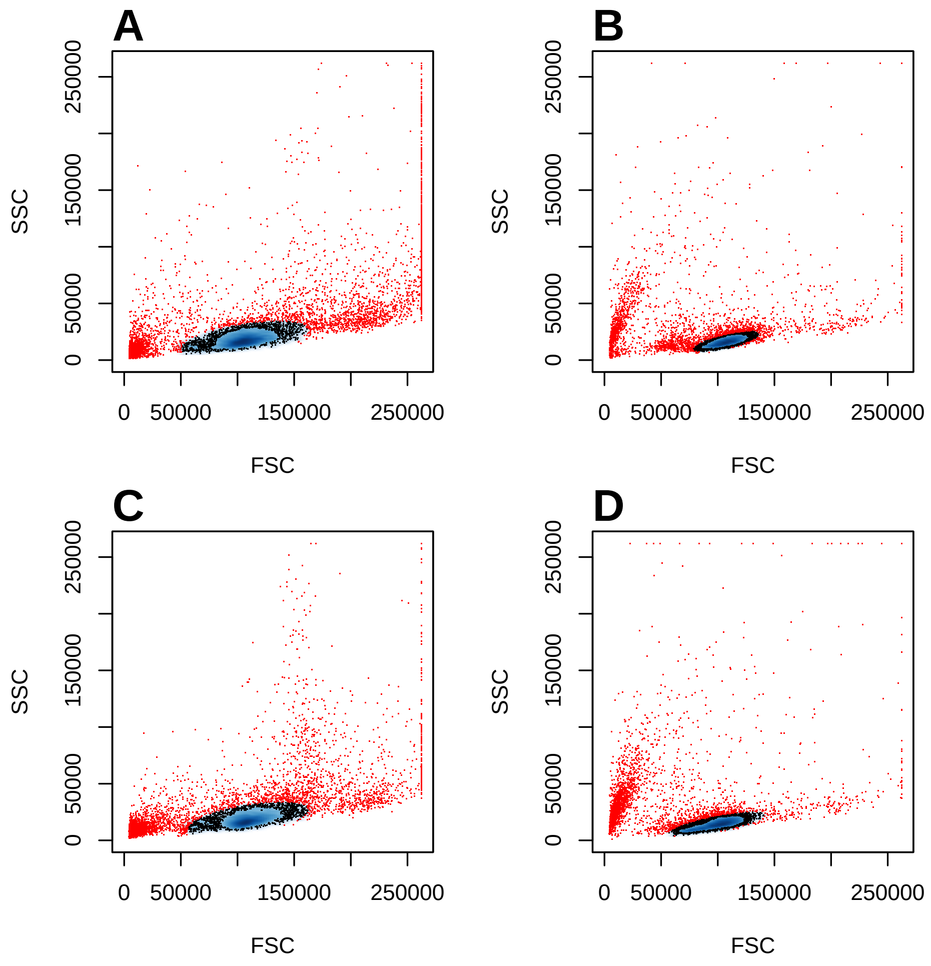

Figure 2: Gating on Forward Scatter and Side Scatter parameters to identify live splenocytes that were included in subsequent analysis: Pre-sort fraction (A), CD4$^+$ fraction (B), CD4$^-$ fraction (C), and CD4$^+$CD62L$^+$ fraction (D). The live cells are shown in blue. Rejected cells appear as red dots.

```
          FITC-A PE-A APC-A
[1,] 1.000000000 0.12     0
[2,] 0.017999996 1.00     0
[3,] 0.002999996 0.00     1
```

To illustrate how the compensation matrix can be refined at any time following data acquisition, unstained and single-stained controls were used to estimate the compensation matrix. The adjusted compensation matrix, expressed in terms of a fraction of the primary signal and shown below, was used to modify the fluorescent measurements.

This adjusted spillover matrix, `fij`, was calculated as follows. Fluorescent spillover of the primary parameter into secondary parameters was assumed to be a linear function of the primary parameter. The observed parameters (O_{ij}) were linearly combined using

$$T_{ij} = O_{i1} \cdot f_{1j} + O_{i2} \cdot f_{2j} + O_{i3} \cdot f_{3j} \tag{1}$$

to estimate the true fluorescent intensity (T_{ij}) for parameter j in experiment i, where f_{kj} is the fraction of parameter k that spills over into parameter j. Upon rewriting Equation 1 in matrix notation and re-arranging the terms, the compensation matrix, $\mathbb{F}$, was estimated from the single stain controls using

$$\mathbb{F} = \mathbb{O}^{-1} \cdot \mathbb{T} \tag{2}$$

by assuming that the observed intensities in the primary parameters provide an estimate of the true fluorescent intensities (i.e., $T_{ij} = O_{ij}$ if $i = j$ else $T_{ij} = 0$). The matrix of the observed intensities (i.e., $\mathbb{O}$) summarized the median values obtained from the single-stain experiments. Prior to calculating the median values, the background fluorescence was subtracted from the raw values. The background fluorescence, corresponding to the median intensity of an unstained parameter, was obtained from the unstained and single-stain experiments (see Supplemental Table 2).

```
> TMedians <- as.matrix(fsApply(PosTFS, each_col, median)[, -(1:2)])
> rownames(TMedians) <- c(1:length(PosTFS))
> data2 <- data.frame(Files = Tclist, "FITC-A" = TMedians[, 1],
+     "PE-A" = TMedians[, 2], "APC-A" = TMedians[, 3])
> tabS2 <- as.matrix(data2)

> library(xtable)
> xtable(tabS2, caption = "Median fluorescence in each parameter shown for
+     each sample.", label = "Tab:S2", align = c("l", "l", "r", "r", "r"),
+     digits = c(0, 0, 2, 2, 2))
```

	Files	FITC.A	PE.A	APC.A
1	Pre-sort Unstained	62.40	51.48	68.88
2	FITC-CD4 Single	68.64	95.16	29.52
3	PE-CD44 Single	71.76	762.84	19.68
4	APC-CD62L Single	46.80	12.48	266.91
5	Pre-sort Population	100.62	837.72	227.55
6	CD4+ Subset	1837.68	1184.04	838.86
7	CD4- Subset	84.24	533.52	172.20
8	CD4+CD62L+ Subset	1723.80	946.92	632.22
9	CD4+CD62L- Subset	1698.84	1461.72	573.18

Table 2: Median fluorescence in each parameter shown for each sample.

One of the challenges with single-stained controls is that the cell population used for the experiment may be heterogeneous (e.g., splenocytes) and may bias the estimate of the median. Partitioning the singly-stained cells into high and low expression groups via a `kmeansFilter` was used to improve the estimate of $\mathbb{O}$.

```
> # flowFrames need to be specified in a particular order in the flowSet
> # 1. unstained control
> # 2. single stain for first column after SSC-A
> # 3. single stain for second column after SSC-A
> # etc
> #Use a kmeansFilter to select high expression groups
> kmfilt1 <- kmeansFilter("kmfilt1", "FITC-A" = c("Low", "High"))
> FITC.high <- fsApply(PosTFS[2], function(x) split(x, kmfilt1)$High)
> kmfilt2 <- kmeansFilter("kmfilt2", "PE-A" = c("Low", "High"))
> PE.high <- fsApply(PosTFS[3], function(x) split(x, kmfilt2)$High)
> kmfilt3 <- kmeansFilter("kmfilt3", "APC-A" = c("Low", "High"))
> APC.high <- fsApply(PosTFS[4], function(x) split(x, kmfilt3)$High)
>
> #Combine resulting flowFrames into a flowSet
> FiltFS = flowSet(PosTFS[[1]], FITC.high[[1]], PE.high[[1]], APC.high[[1]])
>
> #Calculate the background intensity for each parameter
> CMed = as.matrix(fsApply(FiltFS, each_col, median)[, -c(1:2,6)])
>
> #Sweep out medians determined from unstained control from single stained
> #controls
> bFiltFS <- transform(FiltFS, "FITC-A" = `FITC-A` - min(CMed[,1]),
+     "PE-A" = `PE-A` - min(CMed[, 2]), "APC-A" = `APC-A` - min(CMed[,3]))
>
> #Capture medians from single-stained flowFrames
> FObs = as.matrix(fsApply(bFiltFS[c(2:4)], each_col, median)[,-c(1:2,6)])
>
> #Estimate compensation spillover matrix and echo result
> fij = solve(FObs) %*% diag(diag(FObs))
> fij

              [,1]       [,2]          [,3]
FITC-A  1.00386296 -0.1229754 -0.0137533848
PE-A   -0.03153383  1.0038630  0.0004320279
APC-A   0.00000000  0.0000000  1.0000000000

> # Apply calculated compensation matrix to flowSet
> bPosTFS <- transform(PosTFS, "FITC-A" = `FITC-A` - min(CMed[,
+     1]), "PE-A" = `PE-A` - min(CMed[, 2]), "APC-A" = `APC-A` -
+     min(CMed[, 3]))
> cPosTFS <- transform(bPosTFS, cFITC = fij[1, 1] * `FITC-A` +
+     fij[2, 1] * `PE-A` + fij[3, 1] * `APC-A`, cPE = fij[1, 2] *
+     `FITC-A` + fij[2, 2] * `PE-A` + fij[3, 2] * `APC-A`, cAPC = fij[1,
+     3] * `FITC-A` + fij[2, 3] * `PE-A` + fij[3, 3] * `APC-A`)
```

Linear-Log Data Transformation

A logarithmic transform is a common approach used to cope with the wide dynamic range of the fluorescent measurements obtained by flow cytometry. However, fluorescent compensation and subtraction of background fluorescent creates negative values. Plotting data on logarithmic axes truncate the negative values and can lead to incorrect assessment of the compensation for fluorescence spillover [5]. Various alternative methods for displaying fluorescent values have been proposed [5, 6, 7]. A common theme for these different solutions is to use a transform that is linear around zero and non-linear in other regions. In the following section, a simple data transformation is implemented.

Similar to a recent transform proposed by Battye [7], one of the simplest data transformations is to convert the raw data using a linear relationship at lower values and a logarithmic relationship at higher values:

$$\hat{Y} = \begin{cases} M_{linear} \cdot (X_{raw} - b) & \text{if } X_{raw} < transition \\ log_{10}\left(M_{log} \cdot (X_{raw} - b)\right) & \text{if } X_{raw} \geq transition, \end{cases} \tag{3}$$

where $\hat{Y}$ is the transformed "parameter intensity" and X_{raw} is the raw fluorescence value. A smooth transition between these two relationships is ensured by setting the values and the slopes of the linear and logarithmic relationships equal at the transition point. These two constraints provide sufficient information to determine values for the two unknowns: M_{linear} and M_{log}. In addition, we can add an additional constraint that the transformed variable must equal zero when the raw variable equals zero. This shifts the transformed variable for both the linear and logarithmic relationships by $-b$. Prior implementation of this split scale transform required specifying five parameters. Implementing this split scale transform in R/Bioconductor required specifying two values: the median of the untransformed population and the distance (dist) in raw data units between the median adjusted values and the transition point. This relationship was encoded as a function to be reused multiple times within the script:

```
> linlogTransform = function(transformationId, median = 0, dist = 1,
+     ...) {
+     tr <- new("transform", .Data = function(x) {
+         idx = which(x <= median + dist)
+         idx2 = which(x > median + dist)
+         if (length(idx2) > 0) {
+             x[idx2] = log10(x[idx2] - median) - log10(dist/exp(1))
+         }
+         if (length(idx) > 0) {
+             x[idx] = 1/dist * log10(exp(1)) * (x[idx] - median)
+         }
+         x
+     })
+     tr@transformationId = transformationId
+     tr
+ }
```

```
> lnlgT <- linlogTransform(transformationId = "splitscale", median = 0,
+     dist = 100)
>
> #Calculate X-labels for graphs
> lnlgTGraphs <- linlogTransform(transformationId = "splitscale",
+     median = 0, dist = 100)
> Xloc <- lnlgTGraphs(c(-200, -150, -100, -50, 0, 50, 100, 150,
+     200, 250, 400, 550, 700, 850, 1000, 2500, 4000, 5500, 7000,
+     8500, 10000, 25000, 40000, 55000, 70000, 85000, 1e+05))
> Xlab <- c(-200, " ", -100, " ", 0, " ", 100, " ", " ", " ", " ",
+     " ", " ", " ", expression(10^3), " ", " ", " ", " ", " ",
+     expression(10^4), " ", " ", " ", " ", " ", expression(10^5))
```

The transforms were applied to the measured fluorescent values. The transition value was held constant for all of the parameters at a value of 100. The resulting transformed values are deposited within the `flowFrame` in a new parameter.

```
> cPosTFS <- transform(cPosTFS, CD4 = lnlgT(cFITC), CD44 = lnlgT(cPE),
+     CD62L = lnlgT(cAPC))
```

Finally, confirmation of the appropriate compensation for fluorescence spillover is shown in Supplemental Figure 3. Together, these pre-processing steps ensured that the parameter intensities were independent and specific measures of the corresponding levels of protein expression, assuming antibody specificity.

```
> Plim = c(-0.5, 2.75)
>
> #Set up themes for all subsequent lattice figures
> trellis.par.set(theme = col.whitebg())
> lw <- list(ylab.axis.padding = list(x = 0.5), left.padding = list(x = 0.1,
+     units = "inches"), right.padding = list(x = 0, units = "inches"),
+     panel = list(x = 1.5, units = "inches"))
> lh <- list(bottom.padding = list(x = 0, units = "inches"), top.padding <-
+     list(x = 0, units = "inches"), panel = list(x = 1.5, units = "inches"))
>
> lattice.options(layout.widths = lw, layout.heights = lh)
>
> # Plot results from spillover compensation in three panels - tp1, tp2, tp3
> tp1 <- xyplot(CD44 ~ CD4 | name, cPosTFS[c(1:4)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 4), aspect = 1, xlab = "CD4",
+     xlim = Plim, ylab = "CD44", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
+     strip = strip.custom(factor.levels = Tclist[c(1:4)]), panel = function(x,
+         frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-0.5, 2.5), c(0, 0))
+         llines(c(0, 0), c(-0.5, 2.5))
+     })
> tp2 <- xyplot(CD62L ~ CD4 | name, cPosTFS[c(1:4)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 4), aspect = 1, xlab = "CD4",
+     xlim = Plim, ylab = "CD62L", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
+     strip = strip.custom(factor.levels = Tclist[c(1:4)]), panel = function(x,
+         frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-0.5, 2.5), c(0, 0))
+         llines(c(0, 0), c(-0.5, 2.5))
+     })
> tp3 <- xyplot(CD44 ~ CD62L | name, cPosTFS[c(1:4)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 4), aspect = 1, xlab = "CD62L",
+     xlim = Plim, ylab = "CD44", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
+     strip = strip.custom(factor.levels = Tclist[c(1:4)]), panel = function(x,
+         frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-0.5, 2.5), c(0, 0))
+         llines(c(0, 0), c(-0.5, 2.5))
+     })
> plot(tp1, position = c(0, 0, 0.33, 1), more = TRUE)
> plot(tp2, position = c(0.33, 0, 0.66, 1), more = TRUE)
> plot(tp3, position = c(0.66, 0, 1, 1), more = FALSE)
```

Analysis

As highlighted in Figure 1, subsequent data analysis can take multiple paths depending on the research question. To illustrate one path using R/Bioconductor, flow cytometry was used to demonstrate the efficiency of cell sorting using magnetic microbeads. Enrichment of a CD4$^+$CD62L$^+$ T cell population from mouse splenocytes involve two main steps: enrichment of a CD4$^+$ subset using negative selection and subsequent enrichment of a CD4$^+$CD62L$^+$ subset using positive selection. Five aliquots were obtained from the pre-sort population and after each stage of the isolation protocol: pre-sort, CD4$^+$, CD4$^-$, CD4$^+$CD62L$^+$, and CD4$^+$CD62L$^-$. Expression of CD4, CD62L, and CD44 within these groups is shown in Figures 5 and Supplemental Figure 4.

To calculate statistics for the aliquots, a statistically-based data-driven threshold was used define whether a cell was positive for expressing the protein of interest. The threshold was defined as the level of expression for which 95% of the unstained cells exhibited a lower level of expression.

```
> #define positive limits
> # CD44 - from CD4 single-stained control experiment
```

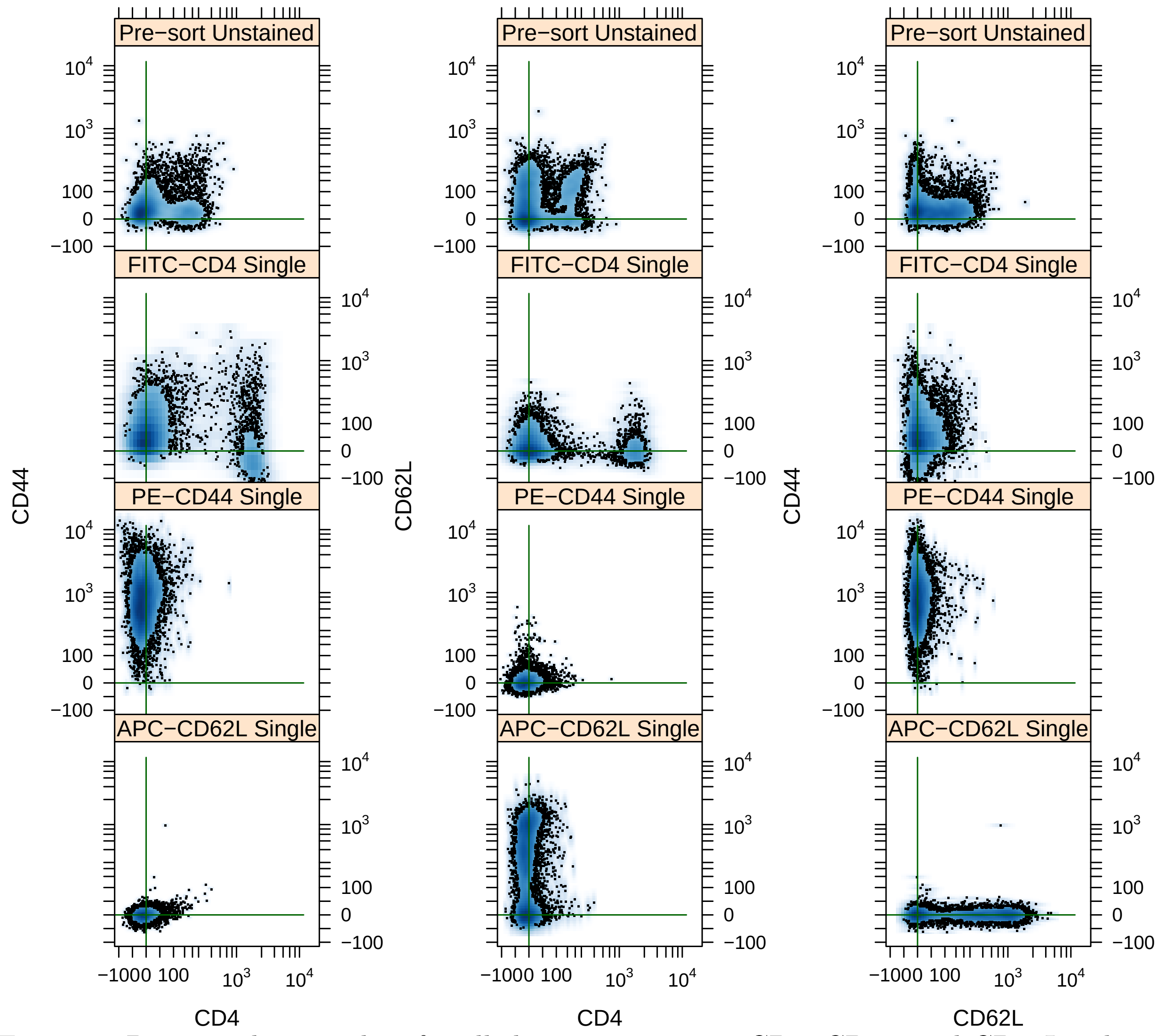

Figure 3: Pairwise density plots for all three parameters - CD4, CD44, and CD62L - shown separately for an unstained aliquot of the pre-sort population and singly stained controls: FITC-CD4, PE-CD44, and APC-CD62L. Each panel corresponds to a particular pair: CD44 versus CD4 (left panel), CD62L versus CD4 (center panel), and CD44 versus CD62L (right panel).

```
> CD441 <- density(exprs(cPosTFS[[2]])[, 11])
> CumV <- 0
> limCD441 <- 0
> while (CumV < 0.95) {
+     limCD441 <- limCD441 + 1
+     CumV <- sum(CD441$y[1:limCD441])/sum(CD441$y)
+ }
```

```
>
> # CD44 - from CD62L single-stained control experiment
> CD442 <- density(exprs(cPosTFS[[4]])[, 11])
> CumV <- 0
> limCD442 <- 0
> while (CumV < 0.95) {
+     limCD442 <- limCD442 + 1
+     CumV <- sum(CD442$y[1:limCD442])/sum(CD442$y)
+ }
> ValCD44 <- max(c(CD441$x[limCD441], CD442$x[limCD442]))
>
> #Estimate high value for CD44 - from CD4+CD62L+ fraction
> CD443 = density(exprs(cPosTFS[[8]])[, 11])
> CumV <- 0
> limCD443 <- 0
> while (CumV < 0.95) {
+     limCD443 <- limCD443 + 1
+     CumV <- sum(CD443$y[1:limCD443])/sum(CD443$y)
+ }
> HiValCD44 <- CD443$x[limCD443]
>
> # CD4 - from CD44 single-stained control experiment
> CD41 = density(exprs(cPosTFS[[3]])[, 10])
> CumV <- 0
> limCD41 <- 0
> while (CumV < 0.95) {
+     limCD41 <- limCD41 + 1
+     CumV <- sum(CD41$y[1:limCD41])/sum(CD41$y)
+ }
>
> # CD4 - from CD62L single-stained control experiment
> CD42 = density(exprs(cPosTFS[[4]])[, 10])
> CumV <- 0
> limCD42 <- 0
> while (CumV < 0.95) {
+     limCD42 <- limCD42 + 1
+     CumV <- sum(CD42$y[1:limCD42])/sum(CD42$y)
+ }
> ValCD4 <- max(c(CD41$x[limCD41], CD42$x[limCD42]))
>
> # CD62L - from CD4 single-stained control experiment
> CD621 = density(exprs(cPosTFS[[2]])[, 12])
> CumV <- 0
> limCD621 <- 0
> while (CumV < 0.95) {
```

```
+     limCD621 <- limCD621 + 1
+     CumV <- sum(CD621$y[1:limCD621])/sum(CD621$y)
+ }
>
> # CD62L - from CD44 single-stained control experiment
> CD622 = density(exprs(cPosTFS[[3]])[, 12])
> CumV <- 0
> limCD622 <- 0
> while (CumV < 0.95) {
+     limCD622 <- limCD622 + 1
+     CumV <- sum(CD622$y[1:limCD622])/sum(CD622$y)
+ }
> ValCD62 <- max(c(CD621$x[limCD621], CD622$x[limCD622]))

> # Pairwise plots for pre-sort, CD4+, and CD4- aliquots
> tp1 <- xyplot(CD44 ~ CD4 | name, cPosTFS[c(5:7)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 3), aspect = 1, xlab = "CD4",
+     xlim = Plim, ylab = "CD44", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
+     strip = strip.custom(factor.levels = Tclist[c(5:7)]), panel = function(x,
+         frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-1, 2.75), c(ValCD44, ValCD44))
+         llines(c(-1, 2.75), c(HiValCD44, HiValCD44), lty = 2)
+         llines(c(ValCD4, ValCD4), c(-1, 2.75))
+     })
> tp2 <- xyplot(CD62L ~ CD4 | name, cPosTFS[c(5:7)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 3), aspect = 1, xlab = "CD4",
+     xlim = Plim, ylab = "CD62L", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
+     strip = strip.custom(factor.levels = Tclist[c(5:7)]), panel = function(x,
+         frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-1, 2.75), c(ValCD62, ValCD62))
+         llines(c(ValCD4, ValCD4), c(-1, 2.75))
+     })
> plot(tp1, position = c(0, 0, 0.5, 1), more = TRUE)
> plot(tp2, position = c(0.5, 0, 1, 1), more = FALSE)


> # Pairwise plots for CD4+CD62L+ and CD4+CD62L- aliquots
> tp1 <- xyplot(CD44 ~ CD4 | name, cPosTFS[c(8:9)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 2), aspect = 1, xlab = "CD4",
+     xlim = Plim, ylab = "CD44", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
```

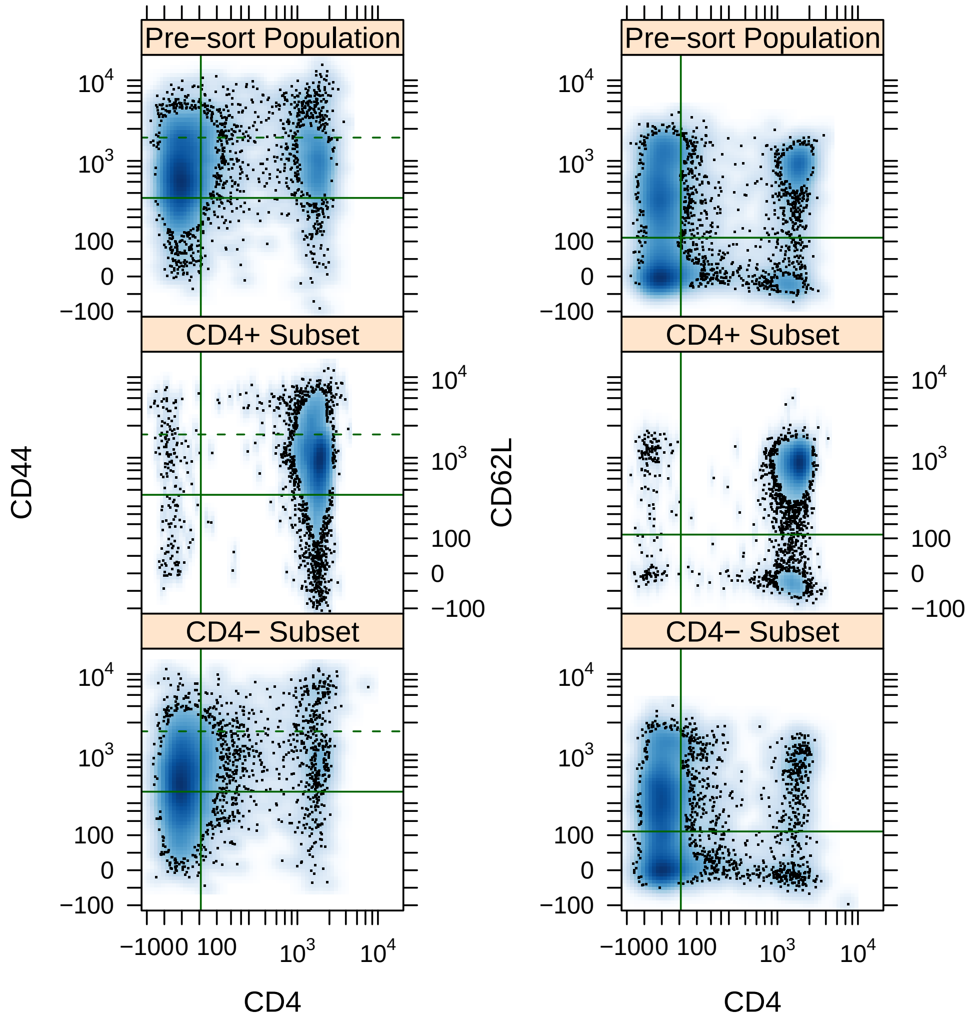

Figure 4: Pairwise density plots for CD4, CD62L, and CD44 expression shown separately for aliquots obtained from pre-sort, CD4$^+$, and CD4$^-$ fractions. Each panel corresponds to a particular pair: CD44 versus CD4 (left panel) and CD62L versus CD4 (right panel). The solid lines indicate the expression threshold for a cell to be associated with positive expression. Ninety five percent of the unstained cell fraction was contained below the threshold. The dotted line indicates the upper limit of CD44 expression for 95% of the CD4$^+$CD62L$^+$ fraction.

```
+     strip = strip.custom(factor.levels = c("CD4+CD62L+", "CD4+CD62L-")),
+     panel = function(x, frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-1, 2.75), c(ValCD44, ValCD44))
+         llines(c(-1, 2.75), c(HiValCD44, HiValCD44), lty = 2)
+         llines(c(ValCD4, ValCD4), c(-1, 2.75))
+     })
> tp2 <- xyplot(CD62L ~ CD4 | name, cPosTFS[c(8:9)], nrpoints = 1000,
+     labels = FALSE, layout = c(1, 2), aspect = 1, xlab = "CD4",
+     xlim = Plim, ylab = "CD62L", ylim = Plim, scales = list(x = list(at = Xloc,
+         labels = Xlab), y = list(at = Xloc, labels = Xlab, rot = 0)),
+     strip = strip.custom(factor.levels = c("CD4+CD62L+", "CD4+CD62L-")),
+     panel = function(x, frames, channel.x, channel.y, ...) {
+         panel.xyplot.flowset(x, frames, channel.x, channel.y, ...)
+         llines(c(-1, 2.75), c(ValCD62, ValCD62))
+         llines(c(ValCD4, ValCD4), c(-1, 2.75))
+     })
> plot(tp1, position = c(0, 0, 0.5, 1), more = TRUE)
> plot(tp2, position = c(0.5, 0, 1, 1), more = FALSE)
```

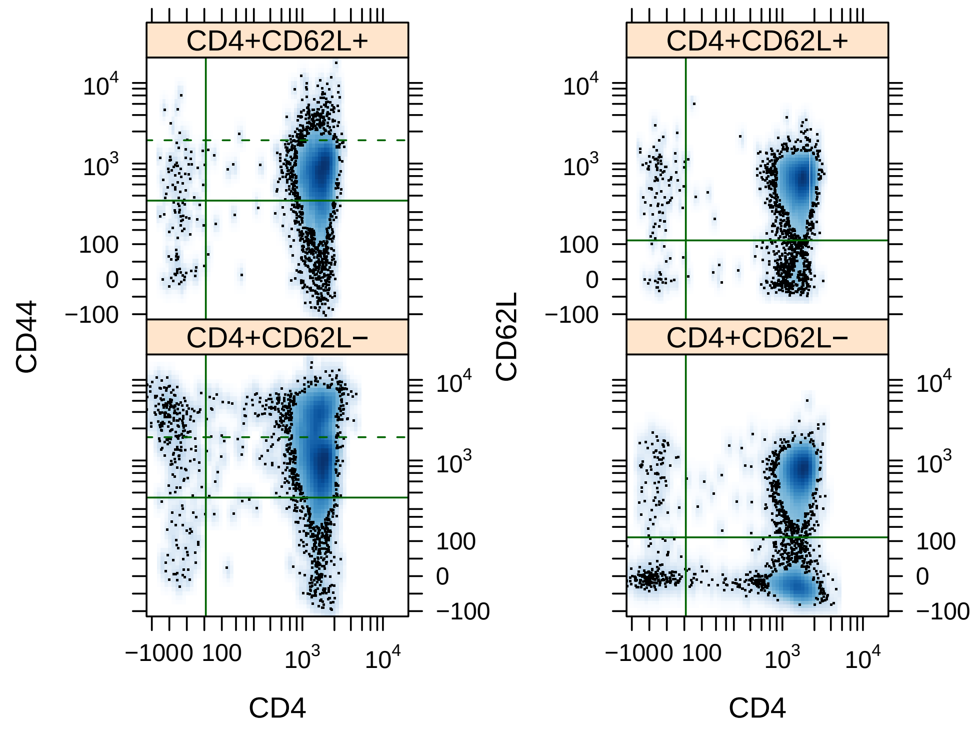

Figure 5: Pairwise density plots for CD4, CD62L, and CD44 expression shown separately for aliquots obtained from CD4$^+$CD62L$^+$, and CD4$^+$CD62L$^-$ fractions. Each panel corresponds to a particular pair: CD44 versus CD4 (left panel) and CD62L versus CD4 (right panel). The solid lines indicate the expression threshold for a cell to be associated with positive expression. Ninety five percent of the unstained cell fraction was contained below the threshold. The dotted line indicates the upper limit of CD44 expression for 95% of the CD4$^+$CD62L$^+$ fraction.

```
> # Calculate statistics for gating
> CD4PGate <- rectangleGate(filterId = "CD4+", CD4 = c(ValCD4, Inf))
> CD44HGate <- rectangleGate(filterId = "CD44hi", CD44 = c(HiValCD44, Inf))
> CD62PGate <- rectangleGate(filterId = "CD62L+", CD62L = c(ValCD62, Inf))
> Total = vector("list", 5)
> CD4PP = vector("list", 5)
> CD4CD62PP = vector("list", 5)
> CD44tCD4CD62PP = vector("list", 5)
> for (i in 5:9) {
+     CD4P = Subset(cPosTFS[[i]], CD4PGate)
+     CD4PCD62P = Subset(cPosTFS[[i]], CD62PGate & CD4PGate)
+     CD4PCD62CD44HP = Subset(cPosTFS[[i]], CD44HGate & CD62PGate &
+         CD4PGate)
+     Total[[i-4]] <- nrow(cPosTFS[[i]])
+     CD4PP[[i-4]] <- nrow(CD4P) * 100/Total[[i-4]]
+     CD4CD62PP[[i-4]] <- nrow(CD4PCD62P) * 100/Total[[i-4]]
+     CD44tCD4CD62PP[[i-4]] <- nrow(CD4PCD62CD44HP) * 100/Total[[i-4]]
+ }
> data3 <- data.frame(Fractions = Tclist[c(5:9)], "Total Cells" =
+     as.numeric(Total), "CD4$^+$ (%)" = as.numeric(CD4PP),
+     "CD4$^+$CD62L$^+$ (%)" = as.numeric(CD4CD62PP),
+     "CD4$^+$CD62L$^+$CD44$^{high}$" = as.numeric(CD44tCD4CD62PP))
> tab3 <- as.matrix(data3)

> library(xtable)
> xtable(tab3, caption = "Efficiency statistics for na\"ive
+     CD4$^+$CD62L$^+$ T cell isolation from Balb/c splenocytes",
+     label = "Tab:3", align = c("l", "l", "r", "r", "r", "r"),
+     digits = c(0, 0, 0, 2, 2, 2))
```

	Fractions	Total Cells	CD4$^+$	CD4$^+$CD62L$^+$	CD4$^+$CD62L$^+$CD44^{high}
1	Pre-sort Population	6090	30.46	18.92	3.50
2	CD4+ Subset	7501	97.77	85.14	6.12
3	CD4- Subset	6150	21.53	11.22	1.54
4	CD4+CD62L+ Subset	7141	98.46	90.46	4.08
5	CD4+CD62L- Subset	7163	96.52	65.00	6.69

Table 3: Efficiency statistics for naïve CD4$^+$CD62L$^+$ T cell isolation from Balb/c splenocytes

As an alternative, a Bayesian framework could be used for gating such that the gate could be refined based upon new data. In practice, classification of a cell into a subset can be obtained by calculating the ratio of the marginalized density of a particular aliquot relative to the marginalized density of a negative control population at a given level of parameter intensity [8]. The disadvantage of this approach is that the particular parameter intensity used for gating would depend on each aliquot.

As shown in Table 3, magnetic bead enrichment from the starting population of Balb/c splenocytes was used to obtain a population of cells that were >98.459599495869% positive for CD4$^+$ and >90.463520515334% positive for both CD4$^+$ and CD62L^{high} (i.e., naïve CD4$^+$ T cells). As the population of CD4$^+$ CD62L^{high} splenocytes may contain a mixture of both central memory and naïve T cells, the activation marker CD44 was used assess the contribution of the central memory pool. Greater than

```
> Res <- 100 - as.numeric(CD44tCD4CD62PP[[4]])
```

95.92% of CD4$^+$CD62L^{hi} cells were observed by flow cytometry to express intermediate to low levels of CD44, consistent with a naïve T cell population (i.e., CD4$^+$CD62L^{hi}CD44^{lo}). The results suggest that contribution of the central memory population (i.e., CD4$^+$CD62L^{hi}CD44^{hi}) was minor. In comparison, a high level of CD44 expressions was observed in the CD4$^+$ CD62L^{lo} population, consistent with an effector T cell population (i.e., CD4$^+$CD62L^{lo}CD44^{hi}). These different T cell subsets can be clearly identified in Figure 6.

```
> tp1 <- levelplot(CD62L ~ CD44, cPosTFS[6], n = 100, contour = TRUE,
+     aspect = 1, labels = FALSE, colorkey = FALSE, col.regions = gray(50:0/50),
+     xlab = "CD44", xlim = Plim, ylab = "CD62L", ylim = Plim,
+     scales = list(x = list(at = Xloc, labels = Xlab), y = list(at = Xloc,
+         labels = Xlab, rot = 0)))
> tp2 <- levelplot(CD62L ~ CD44, cPosTFS[8], n = 100, contour = TRUE,
+     aspect = 1, labels = FALSE, colorkey = FALSE, col.regions = gray(50:0/50),
+     xlab = "CD44", xlim = Plim, ylab = "CD62L", ylim = Plim,
+     scales = list(x = list(at = Xloc, labels = Xlab), y = list(at = Xloc,
+         labels = Xlab, rot = 0)))
> plot(tp1, position = c(0, 0, 0.5, 1), more = TRUE)
> plot(tp2, position = c(0.5, 0, 1, 1), more = FALSE)
```

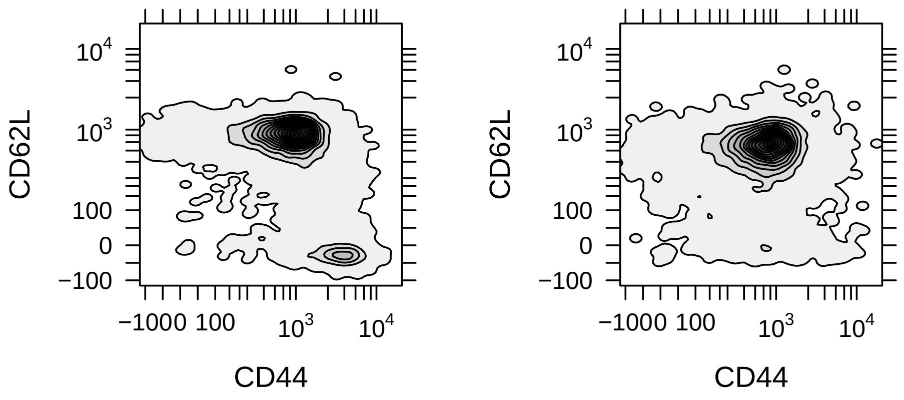

Figure 6: A smoothed contour plot for CD62L versus CD44 expression. The CD4$^+$ population (left panel) is comprised of two populations: a CD62L$^+$CD44^{med} subset and a CD62L$^-$CD44^{high} subset. The CD62L$^-$CD44^{high} subsets was eliminated from the CD4$^+$CD62L$^+$ population (right panel) upon sorting using a anti-CD62L antibody. The contours are colored by density estimation.

Marginalized Probability Density Functions

The fluorescent intensities can be presented in the form of a function that describes the probability of observing particular parameter intensity. This function is referred to as a probability distribution function (PDF). A PDF function is similar to a histogram but is normalized to the total number of observed events, facilitating comparisons among experimental conditions and groups. The PDFs for each time point were obtained by kernel density estimation using the function `density` [9]. Kernel density estimation is a non-parametric smoothing technique used to estimate probability density functions from independent samples drawn from the population of interest. While it shares some similarity with estimating a density function using a normalized histogram, the kernel method exhibits less bias in estimating the density function. The bias in a histogram estimator with a bin width h is of order h. In contrast, the kernel is centered at each point and, by using a symmetric kernel, yields a leading bias term for the kernel estimate of order h^2. Default values for the bandwidth were used. Representative PDFs for CD4 and CD62L expression are shown in Figure 7.

```
> # Set up parameters for ranges used for x and y axis in figures
> yrng <- c(0, 4)
> xrng <- c(-0.5, 2.5)
>
> # Superimpose the PDFs on the same figure
> opar <- par(mfcol = c(2, 2), mar = c(4, 4, 2, 2))
> Pidx = c(5, 6, 8, 1)
> Plty = c(1, 2, 3, 4)
> PCols <- c("red", "darkgreen", "blue", "black")
>
> # This function is a lower-level function that requires numerical
> # input. The command, exprs(cPosTFS[[1]])[,10], extracts the
> # numerical data associated with column 10 from the first flowFrame
> # in flowSet cPosTFS.
> # CD4 Plots
> plot(density(exprs(cPosTFS[[Pidx[1]]])[, 10], na.rm = TRUE, kernel = "rect"),
+     col = PCols[1], xlab = "CD4", xlim = xrng, ylab = "Density",
+     main = "", ylim = yrng, xaxt = "n", lwd = 2, lty = 1)
> title(main = "A", outer = FALSE, adj = 0, cex.main = 2)
> axis(1, Xloc, labels = Xlab)
> for (i in 2:length(Pidx)) {
+     lines(density(exprs(cPosTFS[[Pidx[i]]])[, 10], na.rm = TRUE,
+         kernel = "rect"), col = PCols[i], lwd = 2, lty = Plty[i])
+ }
>
> # CD62L Plots
> plot(density(exprs(cPosTFS[[Pidx[1]]])[, 12], na.rm = TRUE, kernel = "rect"),
+     col = PCols[1], xlab = "CD62L", xlim = xrng, ylab = "Density",
+     ylim = yrng, xaxt = "n", main = "", lwd = 2, lty = Plty[1])
> title(main = "B", outer = FALSE, adj = 0, cex.main = 2)
> axis(1, Xloc, labels = Xlab)
> for (i in 2:length(Pidx)) {
+     lines(density(exprs(cPosTFS[[Pidx[i]]])[, 12], na.rm = TRUE,
```

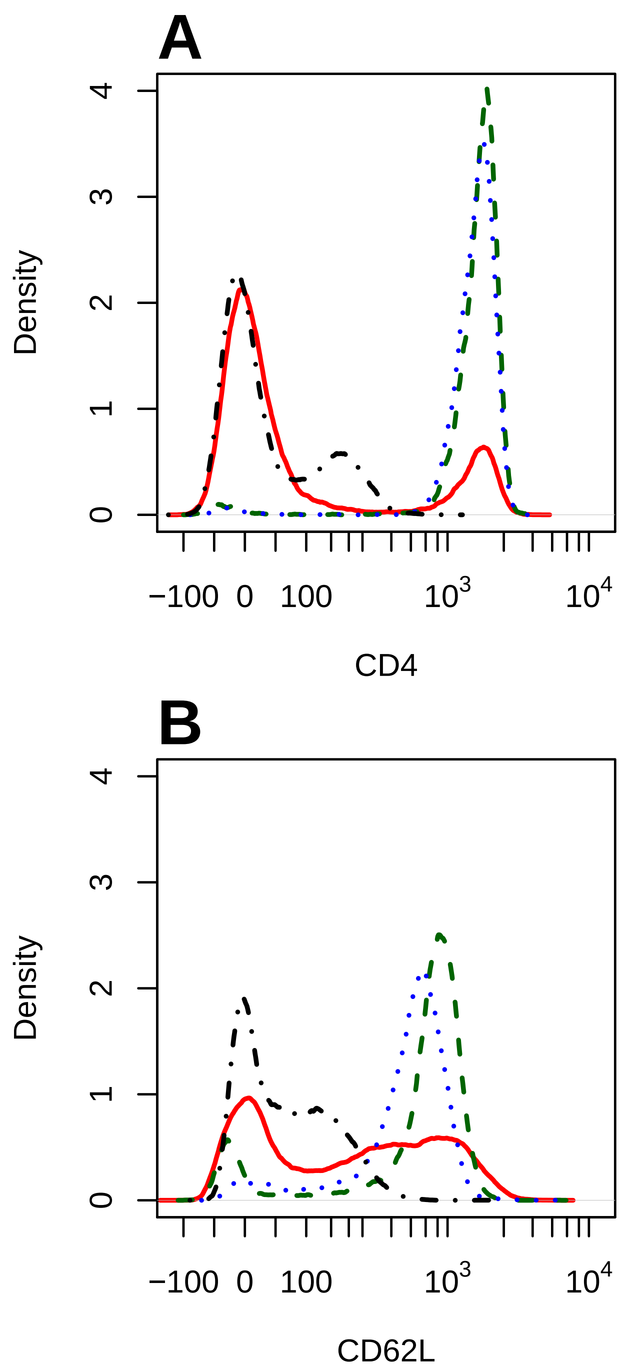

Figure 7: Marginalized probability density functions for CD4 (A) and CD62L (B) expression shown for the different aliquots obtained during MACs cell sorting of Balb/c splenocytes: Unstained fraction (dot-dashed), Pre-sort fraction (solid), CD4$^+$ fraction (dashed), and CD4$^+$CD62Lfraction (dotted).

```
+           kernel = "rect"), col = PCols[i], lwd = 2, lty = Plty[i])
+ }
```

More generally, a PDF is a continuous function that summarizes the distribution in protein expression or activity within a population of cells. The presence of a bimodal PDF distribution indicates that multiple subsets exist within a population. Quantifying the relative contribution of a particular subset can be achieved by deconvoluting a bimodal PDF in terms of a series of overlapping probability distributions (e.g., overlapping Gaussians) that have different median parameter intensities. An analogous approach is used to quantify the cell cycle phase distribution of cells following DNA staining [10, 11].

Principal Component Analysis

The CD4$^+$ and CD4$^+$CD62L$^+$ fractions were further characterized using principal component analysis [12]. Principal component analysis (PCA) is a multivariate statistical technique that allows for the discovery of variables that form a coherent subset and are relatively independent of other subsets of variables. Variables that vary in synchrony with other variables are lumped together into independent principal components. The utility of this approach is in creating a lower-dimensional description of the population, such as multi-dimensional scaling or clustering (e.g., [13]).

To illustrate the approach, three principal components (PCs) were created from the three variables - CD44, CD62L, and CD4 - that characterize the cell population using the R function `princomp`. As principal component analysis is a linear modeling technique, extreme values can influence the quality of the results. Thus the lin-log transformed variables were used in the analysis. The resulting scoring coefficients, shown in Table 4, were used to calculate the principal component values for another cell fraction using:

$$PC_{i,p} = C1_i * v1_p + C2_i * v2_p + \ldots + Cn_i * vn_p, \tag{4}$$

where v's are variable values for the p cell and C's are the scoring coefficients for the i^{th} principal component (PC) and n^{th} variable. A scoring coefficient is related to a correlation coefficient such that a value for the CD62L scoring coefficient of 0.711 in PC_1 means that 50.6% ($100 \cdot 0.711^2$) of the variance in CD62L expression is represented in PC_1. The difference in sign between the scoring coefficients for CD44 and CD62L in PC_1 indicates that these two variables are inversely related in the dataset. PC_1 versus PC_2 projections for the CD4$^+$ and CD4$^+$CD62L$^+$ fractions are shown in Figure 8. The difference in the two populations at a low value for PC_1 corresponds to the elimination of the CD44^{high} subset in the CD4$^+$CD62L$^+$ fraction, as seen in Figure 6 and inferred from the PC loading coefficients.

```
> # Assemble PCA observations from CD4+ subset
> PCAobs = exprs(cPosTFS[[6]])[, c(10:12)]
>
> #calculate covariance matrix for observations then PCA
> covfs1 <- cov(PCAobs, use = "complete.obs")
> fs1PCA <- princomp(PCAobs, subset = complete.cases(PCAobs), cor = TRUE,
+     scores = TRUE)
>
> #Print out loadings of PCA
> PCAload <- loadings(fs1PCA)
> data4 <- data.frame(Parameters = c(rownames(PCAload), "Std Dev"),
+     PC1 = c(PCAload[1:3], fs1PCA$sdev[1]), PC2 = c(PCAload[4:6],
+         fs1PCA$sdev[2]), PC3 = c(PCAload[7:9], fs1PCA$sdev[3]))
> tab4 <- as.matrix(data4)
```

```
> library(xtable)
> xtable(tab4, caption = "Summary statistics for Principal Component
+     Analysis of CD4+ Fraction", label = "Tab:4", align = c("l", "l",
+     "r", "r", "r"), digits = c(0, 0, 2, 2, 2))
```

	Parameters	PC_1	PC_2	PC_3
1	CD4	0.1901	0.9532	0.2349
2	CD44	-0.6774	0.3006	-0.6714
3	CD62L	0.7106	0.0315	-0.7029
4	Std Dev	1.2272	1.0056	0.6948

Table 4: Summary statistics for Principal Component Analysis of CD4+ Fraction

```
> # Predict the corresponding PCs for new data
> scoreCD4 <- predict(fs1PCA, exprs(cPosTFS[[6]])[, c(10:12)])
> scoreCD4CD62L <- predict(fs1PCA, exprs(cPosTFS[[8]])[, c(10:12)])
>
> opar <- par(mfcol = c(1, 1), mar = c(4, 4, 2, 2))
> # Plot results for PCs 1 and 2
> plot(scoreCD4[, 1:2], pch = 21, col = "blue", bg = "blue", cex = 0.5,
+     xlab = "Principal Component 1", ylab = "Principal Component 2")
> cols2 <- densCols(scoreCD4CD62L[, 1:2], nbin = 30, colramp =
+     colorRampPalette(c("white", "red")))
> points(scoreCD4CD62L[, 1:2], pch = 22, cex = 0.5, lwd = 0.25,
+     bg = cols2, col = "red")
```

As mentioned above, PCA identifies linear relationships embedded within high dimensional data. As the number of dimension increases in a flow cytometry experiment, generating and analyzing each pairwise comparison between parameters becomes an onerous task. In addition, a three-way relationship among parameters can be difficult to identify from two-dimensional projections. PCA may be particularly helpful in focusing the analysis to specific combinations of parameters that exhibit interesting relationships, such as the inverse relationship between CD44 and CD62L. Depending on the motivating question, more complex non-linear relationships can be also investigated in R using computationally intensive techniques such as Gaussian Mixture Models (e.g., MCLUST [14]).

In summary, R/Bioconductor is a versatile platform for the analysis of complex data, such as polychromatic flow cytometry data. The value of flow cytometry to inform biological questions requires a multi-step process where the quality of the data can be ensured. As illustrated here, this process for quality control, whether in a high-throughput or low-throughput

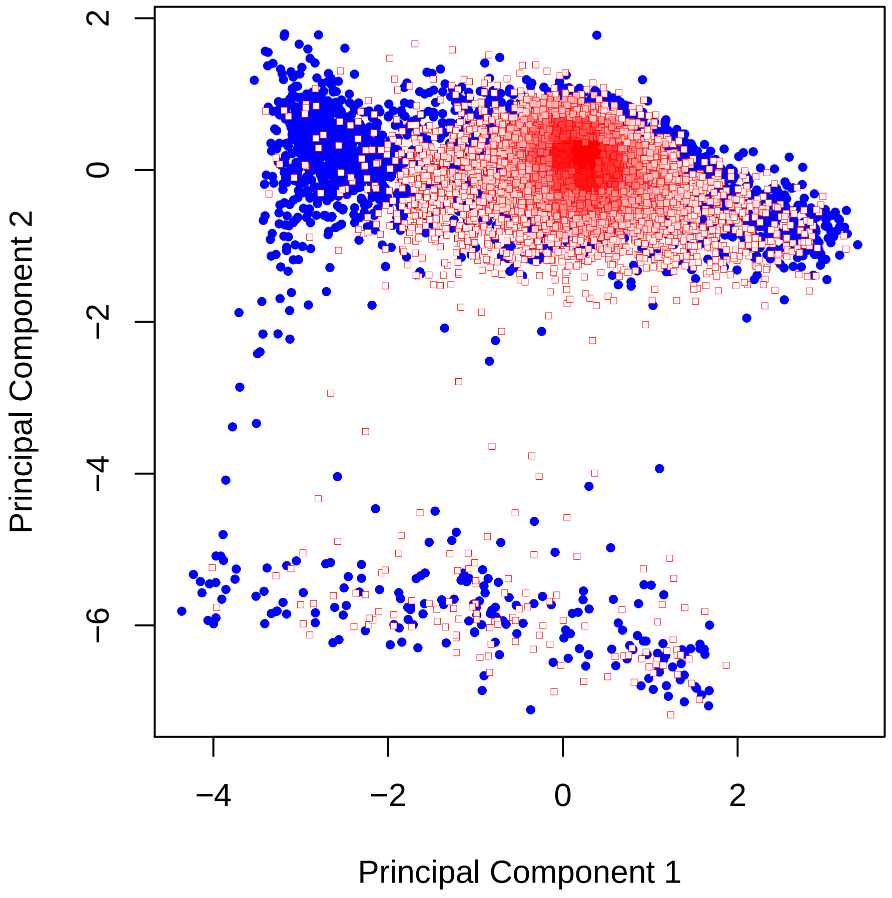

Figure 8: Projections of the CD4$^+$ (filled circles) and CD4$^+$CD62L$^+$ (squares) fractions within the subspace defined by principal component 1 and principal component 2.

setting, is aptly suited to R/Bioconductor. A compendium of text, data, and R scripts provides a clear-cube, rather than black box, approach to the analysis and interpretation of flow cytometry data. The additional effort required to learn this new computational tool is rewarded by the ability to apply a large suite of statistical and graphical tools to your dataset. Specifically, processing can be streamlined by establishing a common workflow in the form of R script templates for typical flow cytometry experiments. Subjectivity can be minimized via use of data-driven gates. Scientific judgement can be focused quickly on embedded trends within this high-dimensional data. Ultimately, the existing analysis algorithms within the R platform provide a rich resource for asking complex questions using polychromatic flow cytometry.

References

[1] L. A. Herzenberg, J. Tung, W. A. Moore, L. A. Herzenberg, and D. R. Parks. Interpreting flow cytometry data: a guide for the perplexed. *Nat.Immunol.*, 7(7):681–685,

2006.

[2] R Development Core Team. R: A language and environment for statistical computing. `http://www.r-project.org`, 2005.

[3] R. Gentleman, V. J. Carey, D. J. Bates, B. M. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, S. Guenther, C. Smith, G. K. Smyth, L. Tierney, Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and informatics. `http://www.bioconductor.org`, 2004.

[4] L. C. Seamer, C. B. Bagwell, L. Barden, D. Redelman, G. C. Salzman, J. C. Wood, and R. F. Murphy. Proposed new data file standard for flow cytometry, version fcs 3.0. *Cytometry*, 28(2):118–122, 1997.

[5] D. R. Parks, M. Roederer, and W. A. Moore. A new "logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry A*, 69(6):541–551, 2006.

[6] C. B. Bagwell. Hyperlog-a flexible log-like transform for negative, zero, and positive valued data. *Cytometry A*, 64(1):34–42, 2005.

[7] F. L. Battye. A mathematical simple alternative to the logarithmic transform for flow cytometric fluorescence data displays. 2005 isac samuel a.latt conference, queensland, australia. `http://www.wehi.edu.au/cytometry/Abstracts/AFCG05B.html`, 2005.

[8] P. Hall and M. P. Wand. On nonparametric discrimination using density differences. *Biometrika*, 75:541–547, 1988.

[9] S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *J Roy Statist Soc B*, 53:683–690, 1991.

[10] P.N. Dean and J. H. Jett. Mathematical analysis of dna distributions derived from flow microfluorometry. *J Cell Biology*, 60:523–527, 1974.

[11] H. Wang and S. Huang. Mixture-model classification in dna content analysis. *Cytometry A*, 71A:716–723, 2007.

[12] R. Khattree and D. N. Naik. *Multivariate Data Reduction and Discrimination with SAS Software.* SAS Institute Inc., Cary, N.C., 2000.

[13] E. Lugli, M. Pinti, M. Nasi, L. Troiano, R. Ferraresi, C. Mussi, G. Salvioli, V. Patsekin, J. P. Robinson, C. Durante, M. Cocchi, and A. Cossarizza. Subject classification obtained by cluster analysis and principal component analysis applied to flow cytometric data. *Cytometry Part A*, 71A(5):334–344, 2007.

[14] C. Fraley and A. E. Raftery. Enhanced model-based clustering, density estimation, and discriminant analysis software: Mclust. *J Classification*, 20(2):263–286, 2003.