ANÁLISIS DE DATOS DE CITOMETRIA DE FLUJO: APLICACIONES DE R-BIOCONDUCTOR

Ramón Tamarit Agusti

Enero 2010

ÍNDICE

1 Citómica y citometría de flujo. Implicaciones bioinformáticas

1.1 Bioinformática y citómica

La bioinformática es la aplicación de tecnología informática a la gestión y análisis de datos biológicos. La bioinformática es una "*ciencia*" interdisciplinar, que requiere el uso o el desarrollo de diferentes técnicas que incluyen informática, matemática aplicada, estadística, ciencias de la computación, inteligencia artificial, química y bioquímica para solucionar problemas, analizar datos, o simular sistemas o mecanismos, **todos ellos de índole biológica y médica, y normalmente (pero no siempre) a nivel molecular** (http://www.ebi.ac.uk/2can/bioinformatics/bioinf_what_1.html).

El ámbito de aplicación de la bioinformática se centra en solucionar o investigar problemas sobre escalas de tal magnitud que sobrepasan el discernimiento humano, haciéndose necesaria la utilización de recursos computacionales. Los principales esfuerzos de investigación en bioinformática se han centrado principalmente en aplicaciones genómicas, proteonómicas, evolutivas y metabolómicas.

La creciente automatización y sofisticación de los clitómetros de flujo ha resultado en la potencialidad de generar de una cantidad de datos similar a la Genómica o Proteónomica, de hecho ya se considera a la citómica [1-8] como una "ómica" más. En la publicación de Valet en 2005 ("*Human cytome project, cytomics, and systems biology: the incentive for new horizons in cytometr*y") [9] se plantean las necesidades tecnológicas bioinformáticas del **Proyecto del Citoma Humano.** Hasta la fecha, los avances bioinformáticos han sido relevantes [10-15], pero aún existen diferencias sustanciales con el resto de "ómicas". Entre otras se encuentran las siguientes: (i) No existe un repositorio público de "datos" de citometría, (ii) El software existente esta principalmente orientado a la "visualización de los datos", las implementaciones orientadas al análisis estadístico y minería de datos son escasas.

El proyecto "Bioinformatics Standards for Flow Cytometry" auspiciado por la FICCS (Flow Informatics and Computational Cytometry Society) es la principal fuente de recursos y avances en Bioinformática aplicada a la citómica y citometría de Flujo. Se centra en dos elementos clave:

- **Diseño de bases de datos relacionales y estructuras de datos**: La información obtenida de los experimentos de citometría tiene que indexarse en bases de datos integradas con el resto de "ómicas". Para ello es necesario desarrollar las ontologías (OBI), los estándares de almacenamiento y transmisión de datos junto con sus metadatos (XML-based standards), los modelos de objetos y los esquemas de base de datos [15].
- **Desarrollo de herramientas de software**: El software de tratamiento de datos debe ser desarrollado en base a los requerimientos de la capa inferior de diseño. Desde el punto de vista bioinformático debe cumplir dos funciones esenciales:(i) encapsular las implementaciones

estadísticas (caja negra para el usuario) facilitando el desarrollo de pasarelas ("pipelines") de minería de datos, y (ii) estandarizar los protocolos de análisis de datos.

1.2 Software para análisis de datos de Citometría de flujo.

El software actual de análisis de datos de Citometría de flujo lo podemos clasificar según la procedencia como:

- **Software propietario del instrumento**: Es el facilitado por el fabricante del instrumento, y normalmente ha estado orientado a la adquisición y visualización "tradicional" de los datos en tiempo real. Actualmente, diversos fabricantes de equipos desarrollan software de tratamiento de datos específicos, sobre todo en el caso de equipos de alto rendimiento ("imagestream", "influxo"), véase por ejemplo el desarrollado por BD Biosciencies, o Amnis.
- **Software de terceros en forma de "programa" ejecutable**. Permite representar y analizar a posteriori los ficheros FCS obtenidos en el instrumento. Como ejemplos de este tipo de software tenemos FlowJo, WinMDI, FCS Express. El principal inconveniente de este tipo es que no permite introducir modificaciones metodológicas en el flujo de trabajo. Su principal ventaja es la facilidad de uso y productividad.
- **Software especifico para el análisis y minería de datos**, como por ejemplo SPSS, MatCad, R, S-Plus e incluso la hoja de cálculo Excel. Este tipo de software tiene el inconveniente de su dificultad de uso ya que en muchos casos hay que programar los protocolos de trabajo. Sin embargo todas las aplicaciones avanzadas de minería de datos de Citometría de flujo se están desarrollando bajo este tipo de plataformas, ya que ofrecen la posibilidad de implementar técnicas avanzadas de minería de datos[16].

1.3 Análisis de datos de Citometría de flujo con R-Bioconductor

La implementación de la capa de análisis y minería de datos impulsada por la FICCS se esta realizando dentro del proyecto Bioconductor, empleando R como herramienta de programación. El resultado es un conjunto de paquetes[4;17-20] (Tabla 1) que resuelven las necesidades básicas de análisis estadístico de cualquier estudio de Citometría de flujo ya sea los tradicionales o los de alto rendimiento.

Las ventajas de R-Bioconductor sobre otras plataformas son incuestionables: Se posee toda la experiencia de las implementaciones de microarrays, es software libre, dispone prácticamente de todas las técnicas estadísticas implementadas a bajo nivel, es multiplataforma, es fácilmente integrable en pipelines. En definitiva, cumple todos los requisitos necesarios para integrarse como herramienta bioinformática para la investigación básica. Los inconvenientes de R quedan relegados a un segundo

término debido a reciente incorporación del paquete iFlow[21] como interface grafica, y a la posibilidad de importar y exportar espacios de trabajo de flowCore a FlowJo[22].

En los apartados siguientes veremos algunos ejemplos prácticos de cómo aplicar los paquetes de R-Bioconductor al tratamiento y análisis de datos de citometría de flujo a pequeña escala. Aplicaciones en donde son necesarias técnicas avanzadas de minería de datos se pueden encontrar en los manuales de cada modulo especifico de Bioconductor.

Módulos bajo flowCore	
flowCore	**Es modulo principal encargado de importar y preprocesar los datos. Los objetos generados por este modulo se pueden analizar mediante las implementaciones del resto de módulos.**
flowViz	Métodos gráficos para la visualización grafica
flowQ	Control de calidad de los datos
flowStats	Métodos estadísticos adicionales a flowCore
flowUtils	Utilidades para integrar modelos de datos de otro mediante XML.
flowClust	Clustering mediante “t mixture models with Box-Cox transformation”
flowMerge	Herramientas para automatizar el modelo de clustering de flowClust, creando filtros automáticos.
flowFP	Creación de huellas dactilares a partir de datos de Citometría de flujo.
flowFlowjo	Importación de espacios de trabajo de FlowJo.
Módulos independientes de flowCore	
prada	Conjunto de herramientas para fenotipado con cellHTS2
cellHTS2	Conjunto de herramientas para análisis de datos de FCHS (flow cytometry high-content screening)
plateCore	Conjunto de herramientas para análisis de datos de FCHS
rflowcyt	Métodos estadísticos adicionales a flowCore

Tabla 1 .- Módulos para el análisis de datos de Citometría de Flujo con R-Bioconductor.

2 Análisis de datos a pequeña escala con Bioconductor.

En este apartados veremos las estructuras y protocolos propuestos por R-Bioconductor, para manejar los datos de citometría de flujo a través de las principales etapas del pre-procesamiento: la compensación, transformación, filtrado, y el posterior análisis de datos. Como ejemplo, revisaremos el código y resultados de un experimento publicado recientemente por David J. Klinke [11], cuyos datos son públicos, y además esta publicado dentro del proyecto bioconductor en forma de "vignette" (tutorial).

2.1 Descripción del experimento.

El objetivo final del experimento es comprobar la eficacia de un kit de separación de Linfocitos CD4+/CD62L+ mediante la técnica de micro-esferas magnéticas. Para comprobar los resultados se emplean tres marcadores: CD4, CD44 y CD62L. Los linfocitos se extraen del bazo de ratones balb/c, del extracto del bazo se prepara una primera muestra control. De la muestra control se realizan citometrías de flujo de: la muestra control ("pre-sort unstained"), de los tres marcadores por separado ("single-stained"), y de todos los marcadores juntos ("pre-sort population").

Figura 1 .- BALB/c. (2009, December 8). In *Wikipedia, The Free Encyclopedia*. Retrieved 09:36, December 16, 2009, from

El proceso de separación consta de dos etapas. Primero se separan los CD4+ de los CD4-. Segundo, la alícuota de CD4+ se trata para separar los CD62L- de los CD62L+. De cada una de las alícuotas obtenidas se realiza una Citometría de flujo ("CD4- subset", "CD4+ subset", "CD4+CD62L+ subset" y "CD4+CD62L- subset").

El CD4 confiere a la célula papel de "helper T-Cell". El CD44 es un marcador de "effector-memory T-cells". Mientras que el marcador CD62L, permite diferenciar los linfocitos que aún no responden a ningún patógeno (naive T-Cell) de los que si (memory T-Cell).

2.2 El flujo de trabajo

El flujo de trabajo de análisis se puede dividir en dos etapas clave, que se resumen en la Figura 2.

Preprocesado: En primer lugar, previo paso del proceso de análisis es necesario asegurar que los niveles observados fluorescencia son independientes entre si (están compensados) y que las medidas específicas del nivel de expresión de la proteína de interés son proporcionales a la expresión de las mismas, suponiendo que los anticuerpos también son específicos.

Análisis: La segunda etapa incluye el análisis de las poblaciones de células mediante gates (filtros) basados en métodos estadísticos (no mediante selección manual). Para ello se estiman las funciones de densidad de probabilidad de los núcleos de población y se agrupan mediante técnicas de clustering.

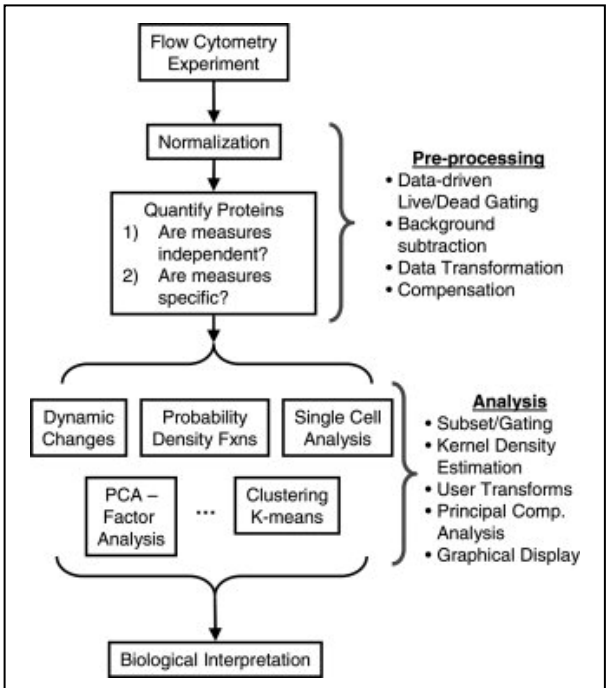


Figura 2. Overview of the steps associated with the use of flow cytometry as a tool in biological research. This manuscript will focus on how Bioconductor can be used during preprocessing and analysis steps

Fuente: **Cytometry Vol.75A, 8 Pages: 699-706**

2.3 Formato de los datos. Ficheros FCS.

Los datos generados por la mayoría de los citómetros de flujo comercial se almacena en el formato Flow Cytometry Standard (FCS) . Dentro de este formato, el estándar de almacenamiento más común es el modo lista. Los ficheros de datos en modo lista tienen una cabecera de texto seguida de los valores experimentales y que finaliza con un bloque de texto en donde se incluyen los protocolos de análisis de datos.

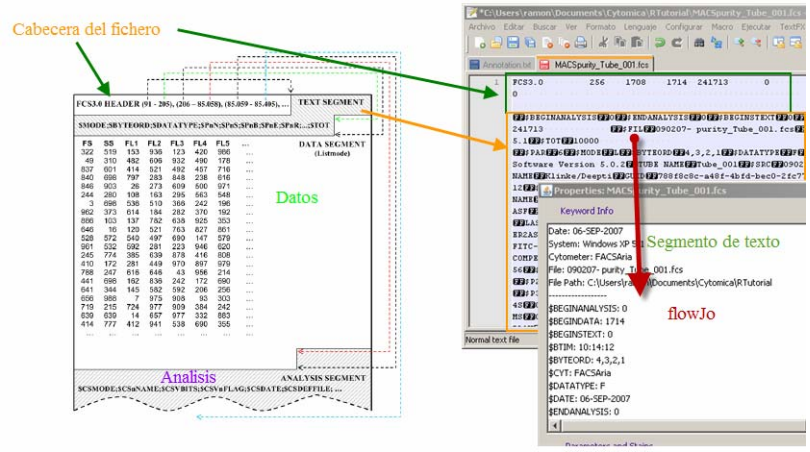


Figura 3.- Esquema de un fichero FCS. Vista desde el notepad de Windows y desde el editor de propiedades de FlowJo

2.4 Manipulación básica de los datos.

La tarea principal del paquete flowCore es la adquisición, representación y manipulación básica de los datos de citometría de flujo. Esto se logra a través de un modelo de datos muy similar a la adoptada por otros paquetes de bioconductor (en concreto con las técnicas de microarrays).

La unidad básica de manipulación e información en flowCore es el flowFrame, que se corresponde con un solo archivo "FCS" (un tubo de experimento).Un flowFrame se compone de los “slots”: De “expresión” que con contienen la información a nivel de eventos (los resultados de fluorescencia de cada célula detectada), y de “parámetros” que contiene los metadatos respectivamente.

Los datos de fluorescencia se almacenan como una matriz y pueden ser fácilmente manipulados mediante los métodos comunes de bioconductor, como por ejemplo el métodos exprs(). El slot de parámetros contiene la información recuperada de los campos de texto del fichero FCS. Por ejemplo los métodos featureNames() y colNames() devuelven (normalmente) los anticuerpos y los flourocromos respectivamente.

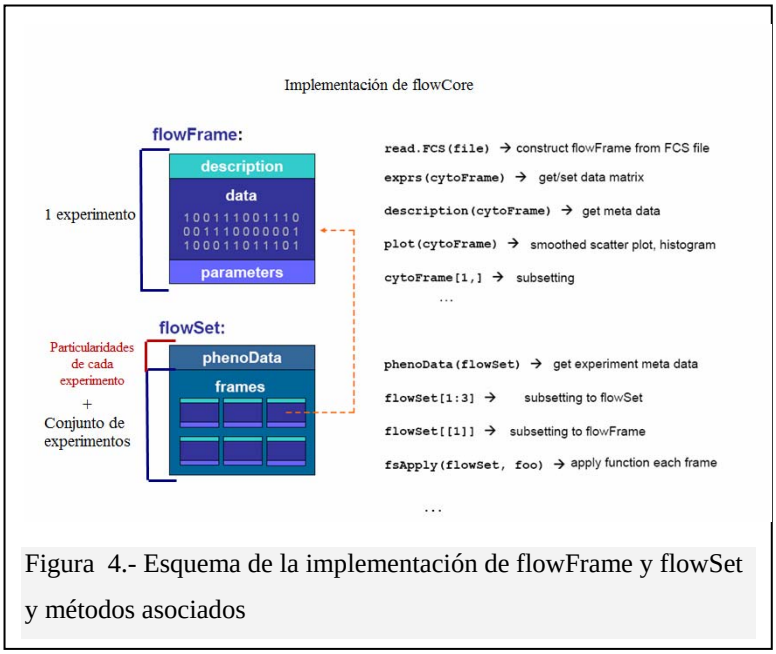


Figura 4.- Esquema de la implementación de flowFrame y flowSet y métodos asociados

La mayoría de los experimentos (varios tubos) consisten en varios objetos flowFrame, que se organizan mediante un objeto flowSet. Esta clase proporciona un mecanismo eficaz para garantizar que los metadatos experimentales se relacionan adecuadamente con cada flowFrame.

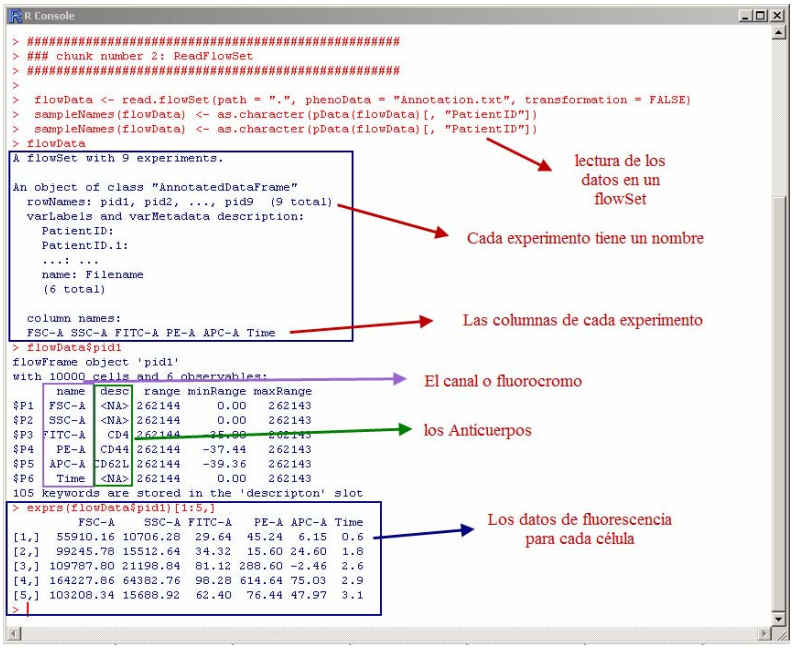


Figura 5.- manipulación de los flowFrame y flowSet con los métodos de flowCore.

2.5 Visualización de los datos.

Gran parte de la visualización más sofisticada de los flowFrame y objetos flowSet, se lleva a cabo por el paquete flowViz. La lista de métodos de visualización de flowViz es muy extensa, prácticamente se pueden reproducir todos los gráficos existentes en la bibliografía.

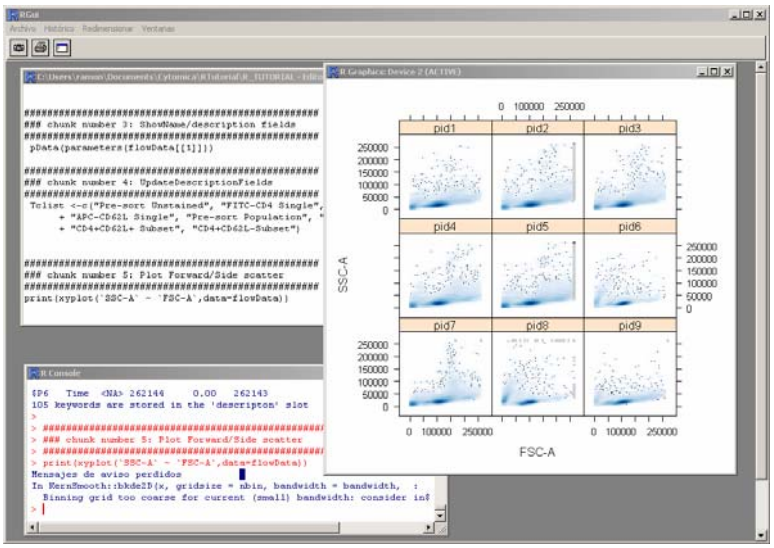


Figura 6.- Ejemplo de visualización de un panel de experimentos mediante flowViz.

2.6 Gatting - Filtrado

La tarea más común en el análisis de los datos de citometría de flujo es el filtrado (o gatting), ya sea para obtener estadísticas de resumen sobre el número de eventos que cumplan determinados criterios o para realizar nuevos análisis en un subconjunto de los datos. La mayoría de las operaciones de filtrado son una composición de una o más operaciones. La definición de los filtros (“gates”) en flowCore sigue la “Gating Markup Language Candidate Recommendation” Spidlen et al. (2008), por lo que cualquier estrategia de filtrado de flowCore puede ser reproducida por cualquier otro software que siga el estándar, y viceversa, por ejemplo en flowJo.

Los filtros más simples, son los “gates” geométricos, que corresponden a los que se suelen encontrar en el software interactivo de la citometría de flujo como son los: filtros marginales, rectangulares, poligonales y elipsoidales. Adicionalmente, se introduce el concepto filtros generados por la distribución estadística de los datos o “data-driven gates”, concepto que no se encuentra bien definido en el software comercial de citometría de flujo.

En el enfoque de “data-driven gates”, los parámetros necesarios se calculan sobre la base de las propiedades de los datos subyacentes, por ejemplo, mediante un ajuste a distribución determinada o por la estimación de la densidad de la población. Por ejemplo, el filtro *norm2Filter* es un método robusto para encontrar la región que más se asemeja a una distribución normal bivariada, y el filtro *kmeansFilter*, Identifica las poblaciones sobre la base de un agrupamiento k-dimensional. Este último filtro permite separar múltiples poblaciones.

Veamos la aplicación práctica en el caso que nos ocupa. Dado que los restos no celulares y las células muertas no muestran tinción específica, estas observaciones se pueden eliminar mediante filtros en los canales FSC y SSC. Los filtros asociados a los linfocitos vivos son:

- Partículas con una intensidad de FSC mayor de 50000
- Un filtro estadístico *norm2Filter* con los parámetros de dispersión frontal y lateral para crear una distribución normal (en dos dimensiones que se centre en la mediana de la población de células y que encierre una región que incluya el 95% de la población (es decir, 2 desviaciones estándar).

La Figura 1 muestra como desde R podemos fácilmente crear y modificar los filtros, así como combinarlos. Una de las ventajas de los filtros estadísticos de R, es que se pueden fácilmente aplicar a múltiples experimentos mediante el objeto flowSet y de esta forma son prácticamente independientes de las posibles variaciones experimentales (cambios en el voltaje, temperatura, cinéticas) entre muestras.

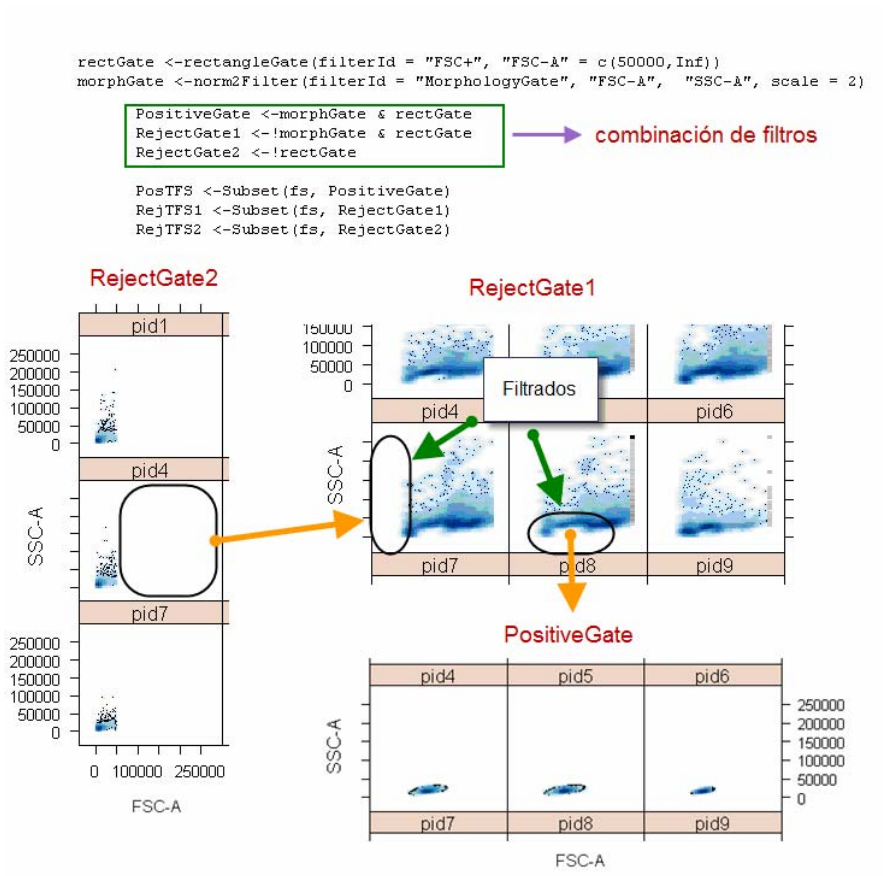


Figura 1.- Estrategias de filtrado en Bioconductor. El filtro rejectGate2, a la izquierda muestra los eventos con FSC<50000. rejectGate1 es una combinación de rejectGate2 y el filtro morphGate (norm2filter). El positiveGate selecciona únicamente las células vivas.

Una vez establecidos los filtros se puede utilizar código R para obtener cualquier estadística asociada a las poblaciones.

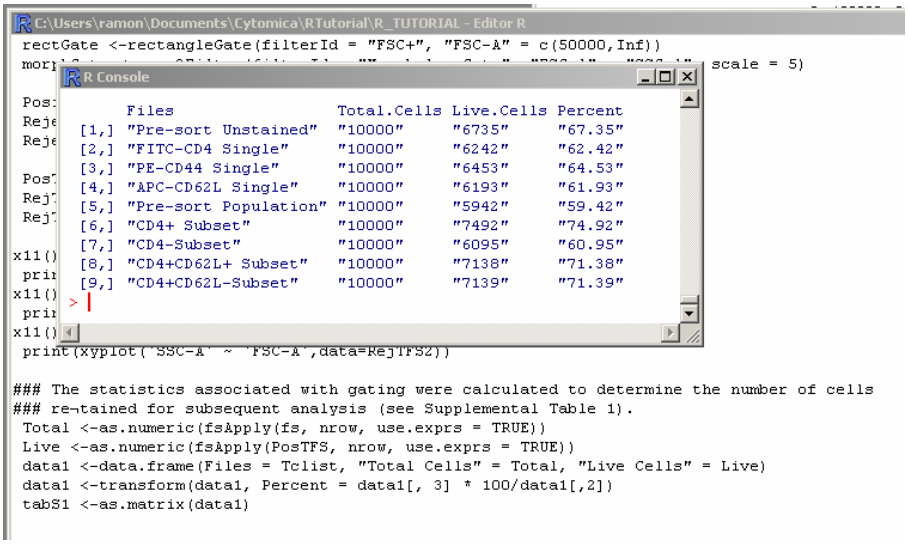


Figura 7.- Ejemplo de manipulación de datos con R. Se calcula el porcentaje de células vivas en cada una de las muestras del ejemplo 1

A efectos operativos, la programación de las estadísticas es uno de los mayores inconvenientes de Bioconductor frente al software comercial con interfaces graficas como flowJo.

2.7 Compensación y corrección de fondo.

Dada la dificultad de determinar los valores de una compensación adecuada en tiempo real (durante la realización del experimento), la generación actual de citómetros de flujo de incorpora dos avances para el análisis de los datos de citometría de flujo. En primer lugar, los controladores de software de los citómetros de flujo incluyen un algoritmo para calcular automáticamente la matriz de compensación de fluorescencia. En segundo lugar, los datos en bruto se almacenan en el fichero FCS sin compensación, se ofrece así la oportunidad de ajustar los valores de compensación después de la recolección de datos. En el ejemplo que nos ocupa la estimación inicial de la matriz de compensación empleada en el experimento se puede extraer de los metadatos de texto de MACSPurity_Tube_001.fcs. Esta es estimación inicial de la matriz de compensación se basa en experimentos anteriores y se utiliza para observar los datos durante la adquisición.

Sin embargo desde R se puede optimizar el cálculo de la matriz de compensación mediante estimación estadística. El siguiente ejemplo ilustra cómo se puede optimizar la matriz de compensación en cualquier momento después de la adquisición de datos, a partir de la muestra de control (unstained) y de las muestras que contienen un solo fluorocromo (single-stained) . La matriz de compensación ajustada, expresada en términos de porcentaje de la señal primaria, se utiliza para modificar las mediciones de fluorescencia. Esta matriz de compensación, Fij, se calcula de la siguiente manera.

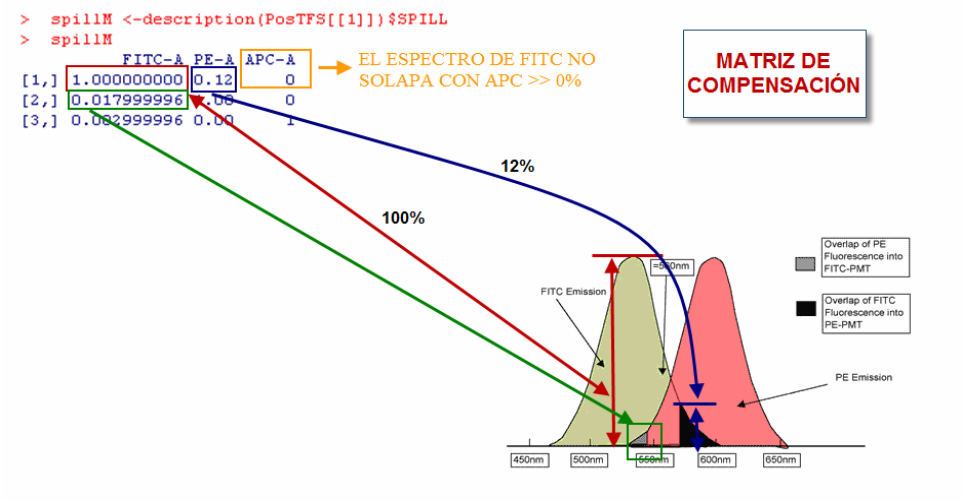


Figura 8Matriz de compensación obtenida de los metadatos del fichero MACSPurity_Tube_001.fcs

El desdoblamiento del canal parámetro primario en los secundarios se supone que será una función lineal del parámetro primario. Los valores compensados son combinación lineal de los observados (O_{ij}) en los experimentos single-stain:

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

Esta ecuación en formato matricial queda: $T = O^{-1} F$

El problema del cálculo de la matriz de compensación se resume a resolver la ecuación (o sistema de ecuaciones) anterior.

La matriz de las intensidades observadas (es decir, O) se estima de los valores de la mediana de cada experimento, single-stain. Antes de calcular los valores de la mediana, la fluorescencia de fondo se resta de los valores en bruto.

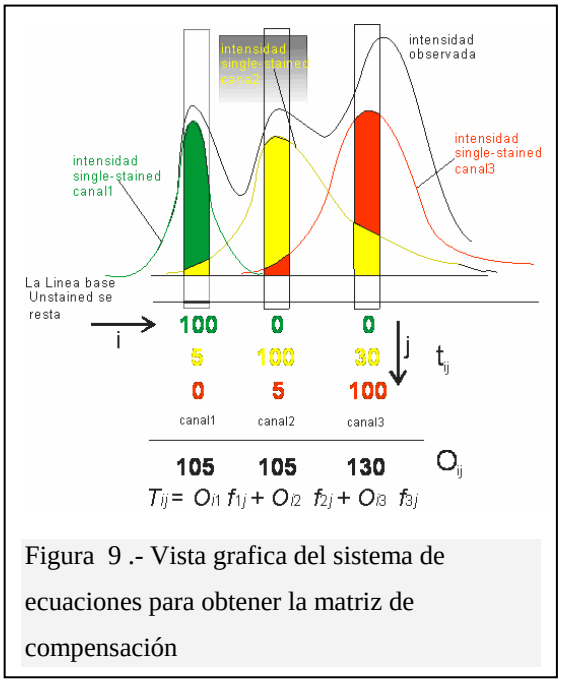


Figura 9 .- Vista grafica del sistema de ecuaciones para obtener la matriz de compensación

```
> 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)
```

```
> tabS2
  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" " 30.75"
3 "PE-CD44 Single"       "  70.20" " 759.72" " 19.68"
4 "APC-CD62L Single"     "  46.80" "  12.48" "273.06"
5 "Pre-sort Population"  "  99.84" " 834.60" "232.47"
6 "CD4+ Subset"          "1837.68" "1184.04" "838.86"
7 "CD4-Subset"           "  82.68" " 533.52" "173.43"
8 "CD4+CD62L+ Subset"    "1723.80" " 946.92" "632.22"
9 "CD4+CD62L-Subset"     "1698.84" "1458.60" "574.41"
>
```

Figura 10.- La matriz de las intensidades observadas

Uno de los problemas de tener una sola muestra de control es que la población de células utilizadas para el experimento puede ser heterogénea y puede sesgar la estimación de la mediana. Dividiendo los controles single-stain en grupos de células con expresión alta y baja mediante un *kmeansFilter* se puede mejorar la estimación de los valores observados (se adjunta el código como anexo).

Como paso final, la Figura 11 muestra como se obtiene y aplica con código R la matriz de compensación.

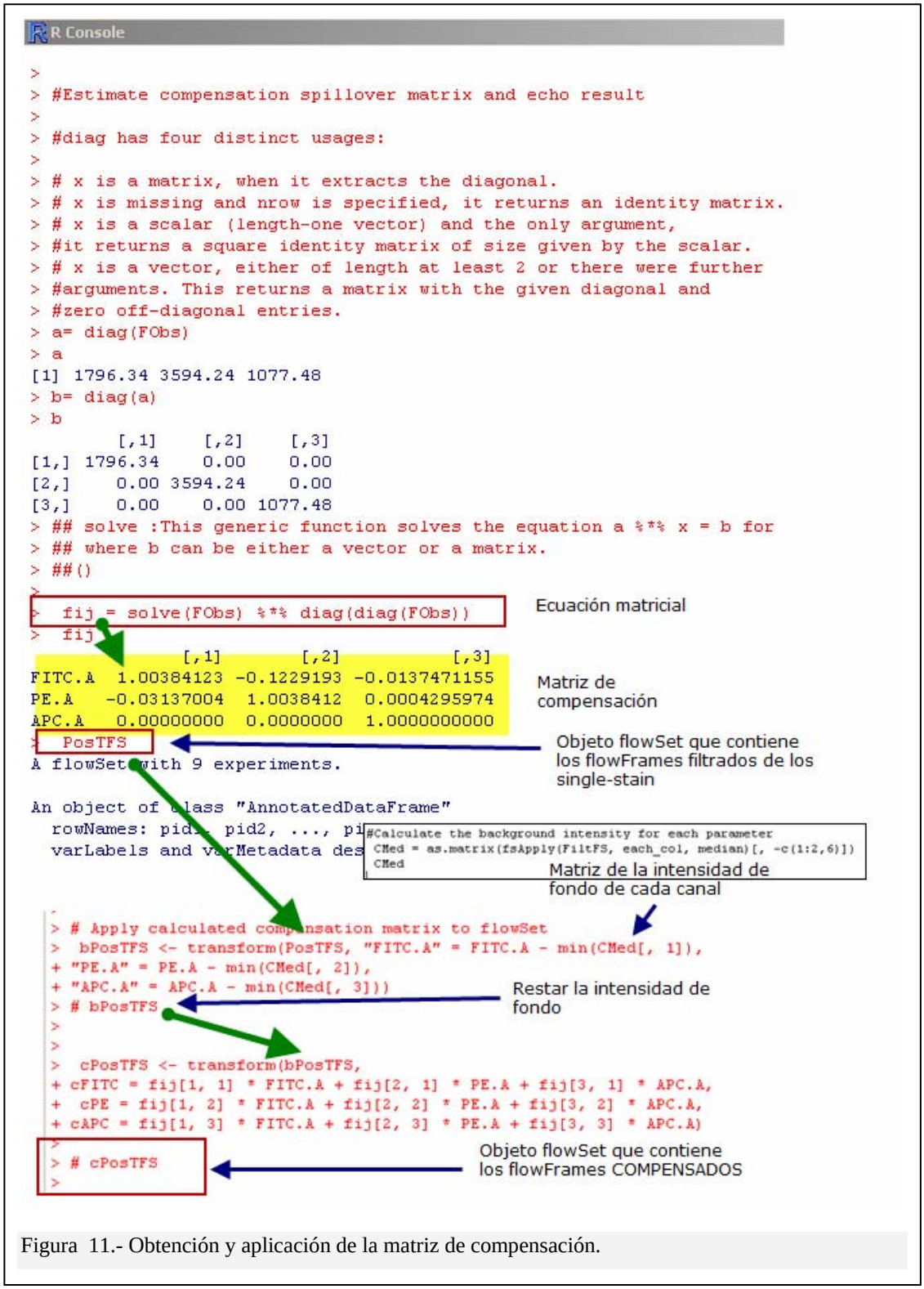


Figura 11.- Obtención y aplicación de la matriz de compensación.

2.8 Escalado y transformación

La transformación y escalado es esencial tanto para la visualización como para posterior tratamiento estadístico de los datos. Las transformaciones que se usan rutinariamente en el análisis de FCM definidas en el estándar "Transformation-ML"[14], se han implementado en flowCore (Figura 12). Además, el diseño de R hace que sea fácil definir nuevas funciones arbitrarias aplicables tanto los flowFrame como a los flowSet para incorporarlas en un protocolo habitual de flowCore.

Data transformations implemented in flowCore.

	Data Transformations
linear	$ax + b$
quadratic	$ax^2 + bx + c$
natural logarithm	$log_e(x)(r/d)$
logarithm	$log_b(x)(r/d)$
biexponential	$ae^{(b*x)} - ce^{(-d+x)} + f$
logicle	$Te^{-(m-w)}(e^{(x-w)}-p^2e^{-(x-w)/p} + p^2-1)$
truncate	$x_{x\leq a} = a$
scale	$(x-a)/(b-a)$
arcsinh	$arcsinh(a + bx) + c$

Within these formulas, x is the variable corresponding to value being transformed, a, b, c, d, f, p, m, T, and w, are constants affecting the transformation function, e is the base of the natural logarithm (see [13] for details on the logicle transformation). Other transformations can easily be implemented in R.

Figura 12.- Transformaciones de datos implementadas en

La transformación logarítmica es el método comúnmente utilizado para hacer frente a la amplia gama dinámica de las medidas de fluorescencia. Sin embargo, la compensación y la sustracción del fondo de fluorescencia crean valores negativos. La representación gráfica de los datos en los ejes logarítmicos truncara los valores negativos. Una alternativa es utilizar una transformación que es lineal en torno a cero y no lineal en otras regiones.

En nuestro ejemplo se aplica de la siguiente forma.

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

```
####################################################
### chunk number 10: Linear-Log Data Transformation
####################################################
                    definición de la función de transformación
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)

cPosTFS <- transform(cPosTFS, CD4 = lnlgT(cFITC), CD44 = lnlgT(cPE),
      CD62L = lnlgT(cAPC))
                                  Aplicación a cada canal ..... No se
                                  aplica a FSC y SSC
```

Figura 13 .- Transformación de los datos mediante una función lineal-logarítmica condicional.

Finalmente mediante observación de los dot-plots comprobamos que los datos quedan uniformemente distribuidos en el grafico. De esta forma se consigue visualizar y separar las poblaciones de forma homogénea.

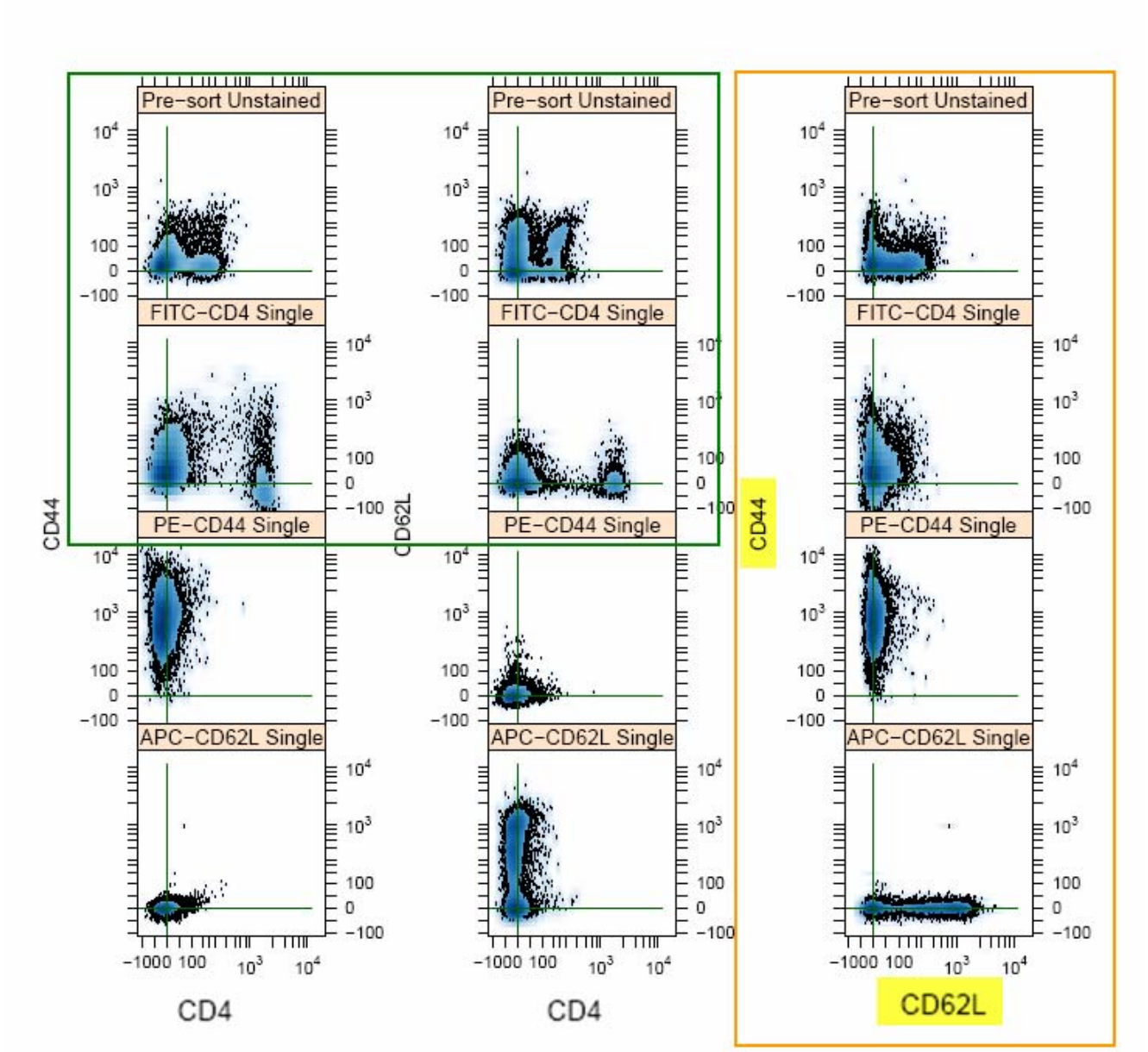


Figura 14.- Dot-plots de todas las combinaciones de fluorocromos para las muestras unstained y las single-stain (antes de la separación). En el recuadro verde es fácil visualizar dos poblaciones en el dotplot, mientras que en el recuadro amarillo, no parecen distinguirse poblaciones a simple vista.

2.9 Análisis de los datos.

El posterior análisis de los datos en R es posible realizarlo desde la perspectiva clásica o empleando técnicas de minería de datos. La técnica clásica consiste en ir visualizando las poblaciones y separarlas en sub-poblaciones positivas/negativas mediante filtros.

Para demostrar la efectiva separación de los linfocitos CD4+CD62L+ hay que comparar las poblaciones relativas de estos en cada una de las cinco alicotas que se han separado en la extracción. Para calcular las estadísticas de cada muestra, nos basamos en un límite de detección calculado a partir de las distribuciones estadísticas de los datos. El límite lo definimos como el nivel de expresión para el cual el 95% de la población de las células no marcadas exhiben un nivel de expresión bajo.

Por ejemplo en la alícuota CD4+, el límite para el marcador CD44 lo calculamos a partir de las distribuciones de población de su canal en las alícuotas single-stain de CD4 y CD62L, eligiendo la mayor de las dos. El resto de límites se calculan de igual forma. El código añadido al original facilitado por la "vignette" facilita la visualización del concepto de límite de detección estadístico.

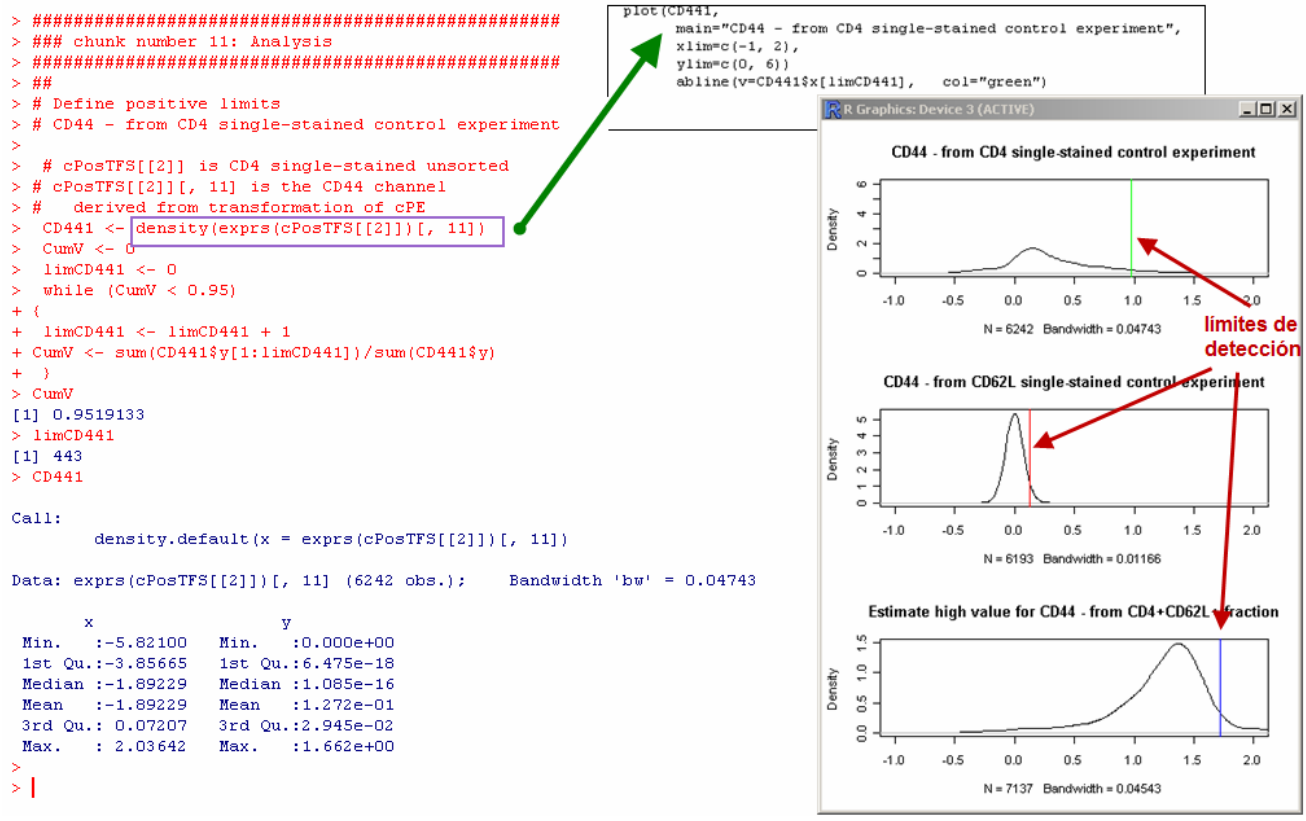


Figura 15.- Cálculo de los límites de detección, a partir de los controles single-stain.

Como hemos comprobado en la sección anterior (ver Figura 14), existe cierta dificultad en estimar un límite de detección para CD44 a partir del cruce con muestras single-stain frente a CD62L. Sin embargo, si comparamos representaciones más avanzadas ("grafico de contorno") de CD44 vs. CD62L, si que encontramos con claridad la existencia de dos poblaciones diferenciadas. Entonces, el límite de detección de CD44 se puede estimar a partir de la desaparición de la población de la población señalada en el círculo rojo de la Figura 16 y Figura 17 (Se corresponde con el grafico de densidad "Estimate High value for CD44 from CD4+CD62L+", línea Azul). (El código R modificado del original se adjunta en el anexo).

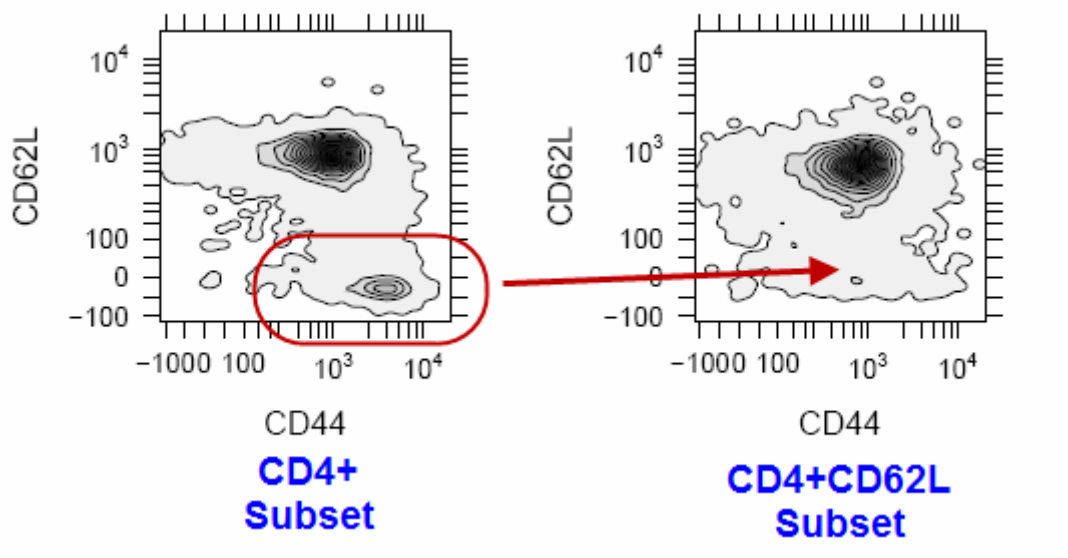


Figura 16.- Dot.plots de contorno de CD44 vs CD62L, en las muestras CD4+ y CD4+CD62L+.

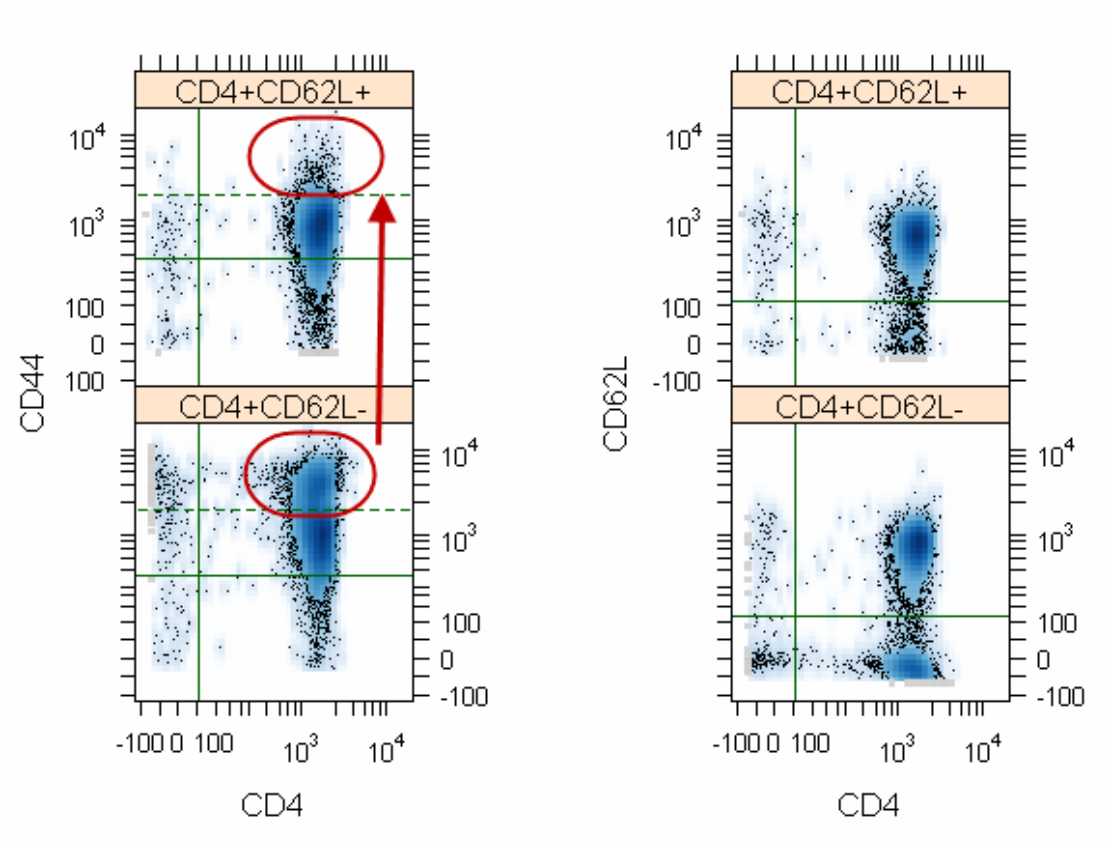


Figura 17.- Dot plots de las muestras finales CD4+CD62+ y CD4+CD62-. Se aprecia como desaparece la población "alta" de CD44.

La población de CD44 que no podemos denominar "positiva", simplemente se denomina "alta", o "baja" (High – Low, tal y como se denomina en la bibliografía) respecto al 95% de la población de CD4+CD62L+.

Esta forma de trabajo, que en principio puede parecer esotérica, complicada y costosa, seguramente sería resuelta con varios “clics” de ratón por un experto en citometría. Pero tiene un beneficio oculto, es reproducible. Tanto es así que simplemente con ejecutar el código del ejemplo (modificado convenientemente) es posible reproducir los resultados **cuantitativos** del experimento de Klinke. De forma manual hubiera sido prácticamente imposible, ya que el margen de error de un “gating” manual prácticamente ocultaría las poblaciones de CD4+CD62L+CD44^{Low} y CD4+CD62L+CD44^{high}. **En resumen, es posible reproducir cuantitativamente (con márgenes de error únicamente debidos a la configuración numérica del procesador) los resultados de Citometría de flujo de un analista a otro**.

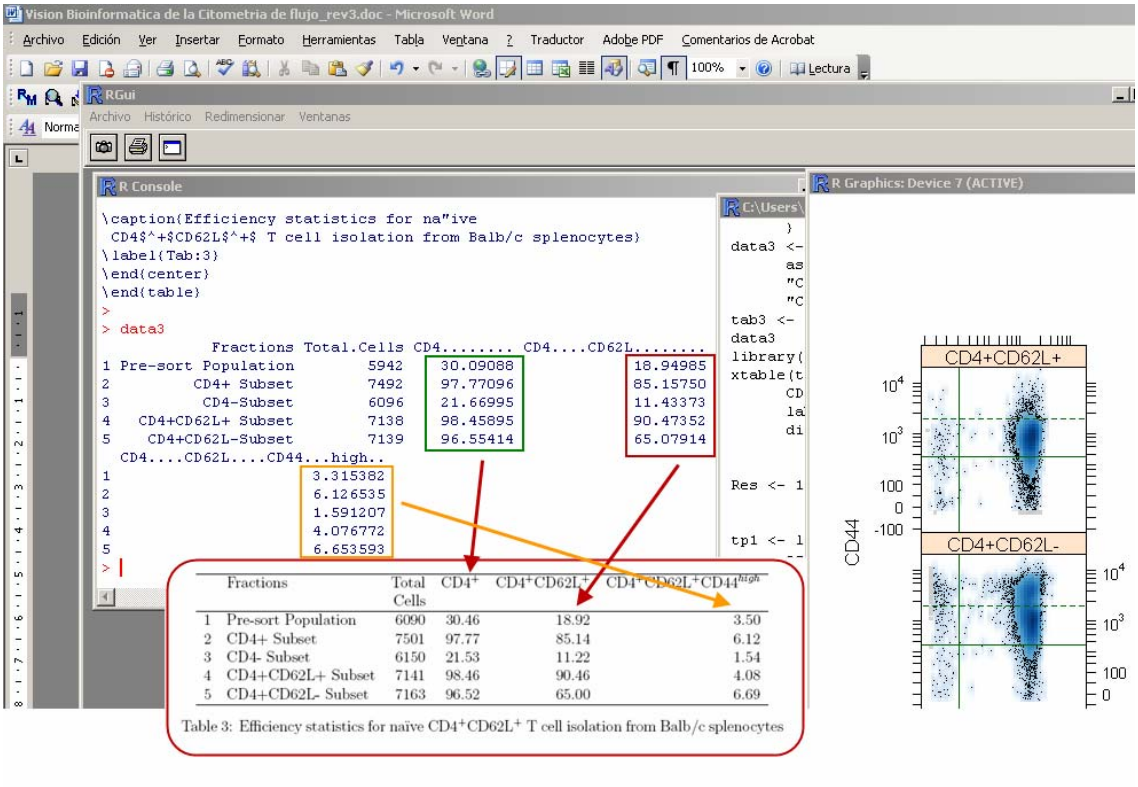


Figura 18.- Comparación de la tabla de datos de estadísticas finales del articulo del ejemplo con la obtenida en el PC local.

2.10 Análisis avanzado de los datos.

Adicionalmente, es posible ilustrar la potencia de R-bioconductor con este ejemplo. Tal y como hemos comprobado (Figura 15) es posible “ajustar” la distribución de población mediante las funcionalidades del paquete flowClust, pero es posible ir mas allá normalizando las funciones de densidad de forma que expresen porcentajes, respecto al número total de observaciones, este tipo de funciones de denominan PDF (“probability distribution function”), y se obtienen mediante estimación de núcleos y ajuste a distribuciones gausianas. Mediante esta técnica es posible estudiar el ciclo celular con un marcador de ADN[23]. En nuestro caso, mediante el PDF se visualizan perfectamente los cambios en la densidad de los picos correspondientes a CD4+ y a CD4+CD62+.

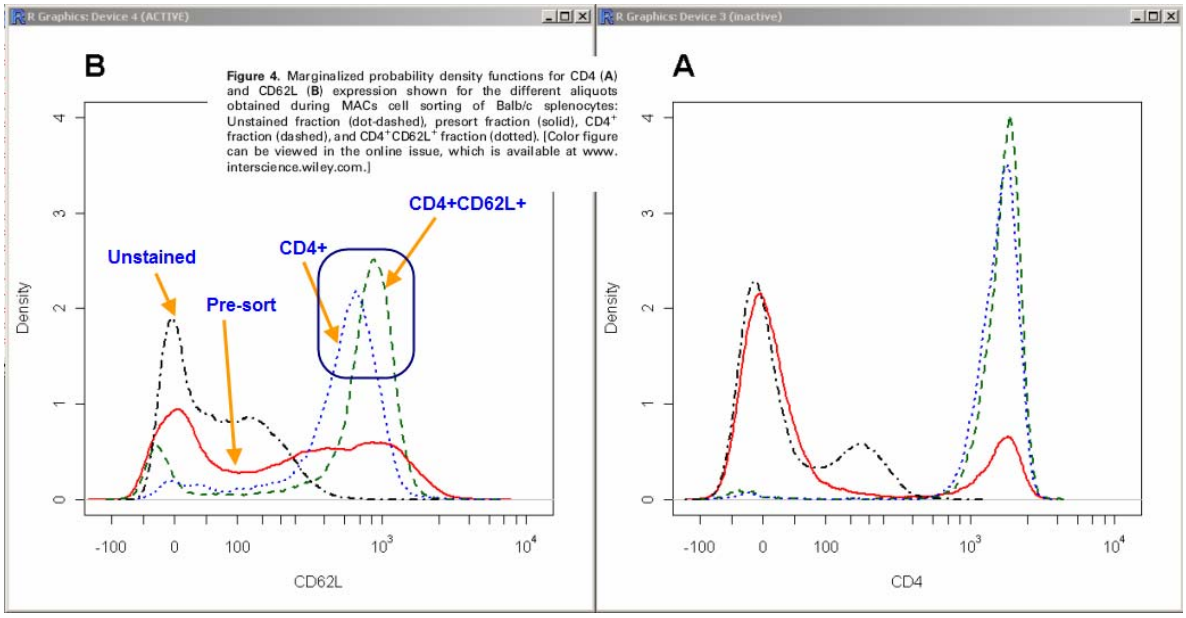


Figura 19.- Funciones de densidad normalizadas.

Las poblaciones de CD4+ y CD4+CD62+ se pueden detectar mediante análisis de componentes principales "PCA ".En estadística, el **análisis de componentes principales** (en español **ACP**, en inglés, **PCA**) es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos. Intuitivamente la técnica sirve para determinar el número de factores subyacentes explicativos tras un conjunto de datos que expliquen la variabilidad de dichos datos. La PCA se emplea sobre todo en análisis exploratorio de datos y para construir modelos predictivos. Técnicamente, el PCA busca la proyección según la cual los datos queden mejor representados en términos de mínimos cuadrados (minimizar la matriz de covarianza). La covarianza es una medida de dispersión conjunta de dos variables estadísticas. Para calcular la PCA hay que descomponer la matriz de covarianza en sus valores propios, tras centrar los datos en la media de cada atributo.

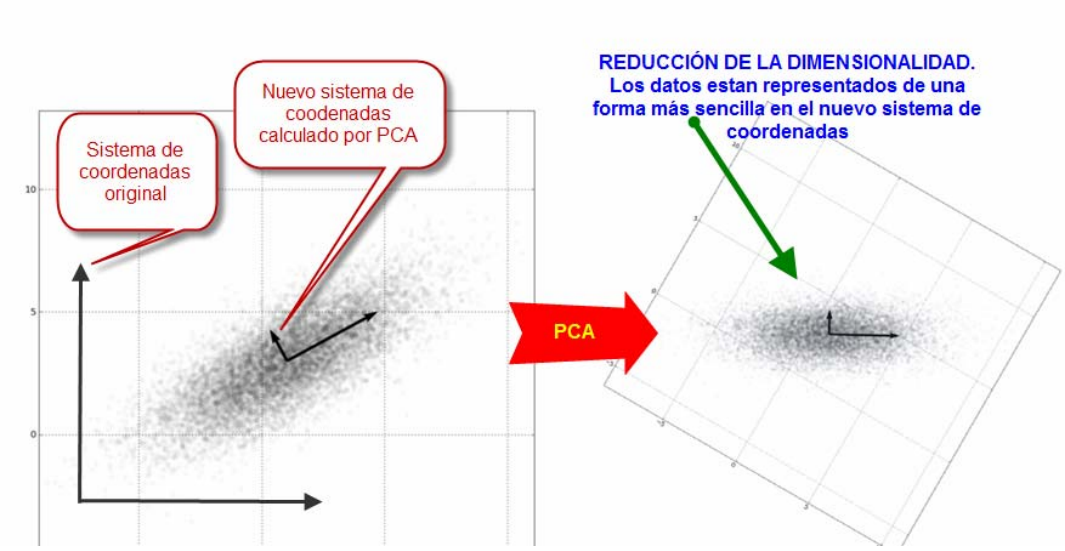


Figura 20.- Reducción de la dimensionalidad y cambio del sistema de coordenadas mediante PCA

Para hacer un análisis de componentes principales hay que elegir el numero de componentes que queremos obtener (normalmente tres) y facilitar las variables observables de nuestro sistema según, que en nuestro caso son los tres canales CD4, CD44 y CD62L. Mediante el análisis de PCA obtenemos una nueva representación de los datos en base al nuevo sistema de coordenadas. Dado que el número de variables originales coincide con el número de componentes principales, únicamente obtenemos un giro en el sistema de coordenadas (Figura 20). Si nuestro sistema hubiera tenido más de tres canales, hubiéramos obtenido un nuevo sistema de coordenadas con una complejidad igual al número de componentes principales elegido. Igualmente, el método nos permite visualizar aquellas componentes que añaden “poca variabilidad” a los datos y eliminarlas. Por ejemplo en la Figura 20 comprobamos que una de las componentes (la vertical) añade poca variabilidad (el vector es pequeño respecto al otro) y por tanto podemos eliminarla del análisis.

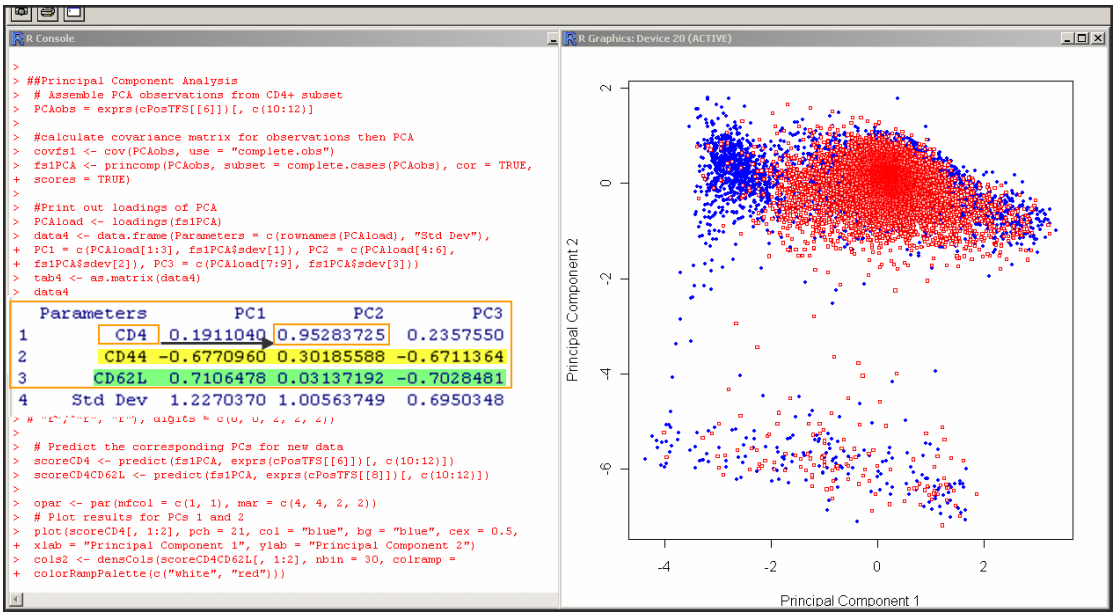


Figura 21.- Analisis PCA del ejemplo de Kirkle. Los círculos azules corresponden al PCA calculados con los datos de CD4+ y los cuadrados rojos a los estimados para CD4+CD62+.

En el ejemplo que nos ocupa, es inmediato observar en los gráficos de componentes principales las diferencias que en los dot-plots no eran tan obvias (recuadro en naranja de la Figura 21 inferior). De

los valores de los nuevos parámetros se extraen las siguientes conclusiones: (i) la expresión de CD4 es directamente proporcional al parámetro P2, (ii) CD44 y CD62L se diferencian en su respuesta inversa (uno es positivo y el otro negativo) respecto al parámetro PC1.

3 Conclusiones.

- El tratamiento clásico de los datos de Citometría de flujo mediante los paquetes de R-bioconductor está resuelto mediante los paquetes publicados actualmente.
- Aunque en el artículo de Florian et al. [24] se indica que la interface gráfica es accesible desde el repositorio de Bioconductor, actualmente no aparece como paquete publicado.
- Mediante los filtros *norm2Filter* y *kmeansFilter*, es posible extraer poblaciones sin la intervención del usuario, de esta forma se evitan los errores experimentales debidos a la selección manual.
- El cálculo de la matriz de compensación es posible evaluarlo estadísticamente a partir de las muestras control y “single-stain”.
- El modelo de datos permite aplicar, además de las transformaciones clásicas, otras posibles trasformaciones combinadas según los requerimientos de los datos.
- Las funciones de escalado y normalización eliminan la variación técnica entre experimentos, de esta forma es posible tratar conjuntamente grandes grupos.
- A diferencia del software comercial, mediante R-bioconductor, es posible aplicar técnicas estadísticas particularizas al problema sin necesidad transformaciones adicionales de formato.

Referencias

1. Gomase,V.S., Tagore,S. (2008) Cytomics. Curr.Drug Metab., 9,263-266.

2. Tarnok,A., Valet,G.K., and Emmrich,F. (2006) Systems biology and clinical cytomics: The 10th Leipziger Workshop and the 3rd International Workshop on Slide-Based Cytometry, Leipzig, Germany, April 2005. Cytometry A., 69,36-40.

3. Bocsi,J., Mittag,A., Sack,U., Gerstner,A.O., Barten,M.J., and Tarnok,A. (2006) Novel aspects of systems biology and clinical cytomics. Cytometry A., 69,105-108.

4. Bernas,T., Gregori,G., Asem,E.K., and Robinson,J.P. (2006) Integrating cytomics and proteomics. Mol.Cell Proteomics., 5,2-13.

5. Gong,J.P. (2003) [From genomics, proteomics to cytomics, or from cytometry to cytomics]. Ai.Zheng., 22,449-451.

6. Valet,G. (2005) Cytomics, the human cytome project and systems biology: top-down resolution of the molecular biocomplexity of organisms by single cell analysis. Cell Prolif., 38,171-174.

7. Kriete,A. (2005) Cytomics in the realm of systems biology. Cytometry A., 68,19-20.

8. Herrera,G., Diaz,L., Martinez-Romero,A., Gomes,A., Villamon,E., Callaghan,R.C., and O'Connor,J.E. (2007) Cytomics: A multiparametric, dynamic approach to cell research. Toxicol.In Vitro., 21,176-182.

9. Valet,G. (2005) Human cytome project, cytomics, and systems biology: the incentive for new horizons in cytometry. Cytometry A., 64,1-2.

10. Spidlen,J., Gentleman,R.C., Haaland,P.D., Langille,M., Le,M.N., Ochs,M.F., Schmitt,C., Smith,C.A., Treister,A.S., and Brinkman,R.R. (2006) Data standards for flow cytometry. OMICS., 10,209-214.

11. Klinke,D.J., Brundage,K.M. (2009) Scalable analysis of flow cytometry data using R/Bioconductor. Cytometry A., 75,699-706.

12. Qian,Y., Tchuvatkina,O., Spidlen,J., Wilkinson,P., Gasparetto,M., Jones,A.R., Manion,F.J., Scheuermann,R.H., Sekaly,R.P., and Brinkman,R.R. (2009) FuGEFlow: data model and markup language for flow cytometry. BMC.Bioinformatics., 10:184.,184.

13. Hammer,M.M., Kotecha,N., Irish,J.M., Nolan,G.P., and Krutzik,P.O. (2009) WebFlow: a software package for high-throughput analysis of flow cytometry data. Assay.Drug Dev.Technol., 7,44-55.

14. Spidlen,J., Leif,R.C., Moore,W., Roederer,M., and Brinkman,R.R. (2008) Gating-ML: XML-based gating descriptions in flow cytometry. Cytometry A., 73A,1151-1157.

15. Lee,J.A., Spidlen,J., Boyce,K., Cai,J., Crosbie,N., Dalphin,M., Furlong,J., Gasparetto,M., Goldberg,M., Goralczyk,E.M., Hyun,B., Jansen,K., Kollmann,T., Kong,M., Leif,R., McWeeney,S., Moloshok,T.D., Moore,W., Nolan,G., Nolan,J., Nikolich-Zugich,J., Parrish,D., Purcell,B., Qian,Y., Selvaraj,B., Smith,C., Tchuvatkina,O., Wertheimer,A., Wilkinson,P., Wilson,C., Wood,J., Zigon,R., Scheuermann,R.H., and Brinkman,R.R. (2008) MIFlowCyt: the minimum information about a Flow Cytometry Experiment. Cytometry A., 73,926-930.

16. Ali Bashashat, Ryan R.Brinkman (2009) Advances in Bioinformatics In Press.

17. Hahne,F., LeMeur,N., Brinkman,R.R., Ellis,B., Haaland,P., Sarkar,D., Spidlen,J., Strain,E., and Gentleman,R. (2009) flowCore: a Bioconductor package for high throughput flow cytometry. BMC.Bioinformatics., 10:106.,106.

18. Klinke,D.J., Brundage,K.M. (2009) Scalable analysis of flow cytometry data using R/Bioconductor. Cytometry A., 75,699-706.

19. Lo,K., Hahne,F., Brinkman,R.R., and Gottardo,R. (2009) flowClust: a Bioconductor package for automated gating of flow cytometry data. BMC.Bioinformatics., 10:145.,145.

20. Sarkar,D., Le,M.N., and Gentleman,R. (2008) Using flowViz to visualize flow cytometry data. Bioinformatics., 24,878-879.

21. Lee,J.A., Spidlen,J., Boyce,K., Cai,J., Crosbie,N., Dalphin,M., Furlong,J., Gasparetto,M., Goldberg,M., Goralczyk,E.M., Hyun,B., Jansen,K., Kollmann,T., Kong,M., Leif,R., McWeeney,S., Moloshok,T.D., Moore,W., Nolan,G., Nolan,J., Nikolich-Zugich,J., Parrish,D., Purcell,B., Qian,Y., Selvaraj,B., Smith,C., Tchuvatkina,O., Wertheimer,A., Wilkinson,P., Wilson,C., Wood,J., Zigon,R., Scheuermann,R.H., and Brinkman,R.R. (2008) MIFlowCyt: the minimum information about a Flow Cytometry Experiment. Cytometry A., 73,926-930.

22. Gosink,J.J., Means,G.D., Rees,W.A., Su,C., and Rand,H.A. (2009) Bridging the Divide between Manual Gating and Bioinformatics with the Bioconductor Package flowFlowJo. Adv.Bioinformatics, 809469.

23. Wang,H., Huang,S. (2007) Mixture-model classification in DNA content analysis. Cytometry A, 71,716-723.

24. Lee,K., Hahne,F., Sarkar,D., and Gentleman,R. (2009) iFlow: A Graphical User Interface for Flow Cytometry Tools in Bioconductor. Adv.Bioinformatics, 103839.

4 Anexo. Código R modificado.

```
####################################################
### chunk number 0: Set working dir
####################################################
###workingDir <- getwd()
### Depends on user ........... ¡¡¡¡
dataDir <- "C:\\Users\\ramon\\Documents\\trabajos\\Cytomica\\RTutorial"

setwd(dataDir)



####################################################
### chunk number 1: Load Libraries
####################################################
 library(flowCore)
 library(flowQ)
 library(flowViz)
 library(flowStats)
 library(flowUtils)
 library(geneplotter)
 library(colorspace)
 library(grid)
 library(MASS)

####################################################
### chunk number 2: ReadFlowSet
####################################################

 flowData <- read.flowSet(path = ".", phenoData = "Annotation.txt", transformation = FALSE,
alter.names = TRUE)
 sampleNames(flowData) <- as.character(pData(flowData)[, "PatientID"])



####################################################
### chunk number 3: Quality control
####################################################
 library(flowQ)
 qaReport(flowData , c("qaProcess.timeline",
                       "qaProcess.timeflow",
                       "qaProcess.cellnumber"))

####################################################
### chunk number 4: ShowName/description fields
####################################################
 pData(parameters(flowData[[1]]))


####################################################
### chunk number 5: Save data to matrix
####################################################

 matrixData1<-(exprs(flowData[[1]]))
 write.table(matrixData1,"matrixData1.csv",sep='\t')
 write.matrix(matrixData1, file = "matrixData1.txt", sep ='\t')

 matrixData2<-(exprs(flowData[[2]]))

 matrixData3<-(exprs(flowData[[3]]))

 matrixData4<-(exprs(flowData[[4]]))

 matrixData5<-(exprs(flowData[[5]]))

 matrixData6<-(exprs(flowData[[6]]))
```

```
 matrixData7<-(exprs(flowData[[7]]))

 matrixData8<-(exprs(flowData[[8]]))

 matrixData9<-(exprs(flowData[[9]]))



 summary(matrixData)

#####################################################
### chunk number 6: UpdateDescriptionFields
#####################################################
 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")


#####################################################
### chunk number 7: Plot Forward/Side scatter
#####################################################
print(xyplot(`SSC.A` ~ `FSC.A`,data=flowData))
##print(splom(flowData[[4]],gridsize=1000))
x11()
print(xyplot(`SSC.A` ~ `FSC.A` , data=flowData ,smooth=FALSE, outline=TRUE) )

# tData <- transform(flowData, transformList(colnames(flowData[1,4]), asinh))



#####################################################
### chunk number 8: Filters
#####################################################

 #Make a copy

 fs<-flowData

 rectGate <-rectangleGate(filterId = "FSC+", "FSC.A" = c(50000,Inf))
 morphGate <-norm2Filter(filterId = "MorphologyGate", "FSC.A",  "SSC.A", scale = 2)

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

x11()
 trellis.focus()
 do.call("my.panel.xyplot", trellis.panelArgs())
 xyplot(`SSC.A` ~ `FSC.A` , data=fs, filter=RejectGate1 )

 cf <- curv2Filter("SSC.A", "FSC.A")
 fres <- filter(fs , cf)
 objects(fs)
 summary(fs)
 xyplot('SSC.A' ~ 'FSC.A'  ,
        data = fs$pid5 , filter = fres)

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

 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data=PosTFS ))
 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data=RejTFS1))
 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data=RejTFS2))


### The statistics associated with gating were calculated to determine the number of cells
### retained for subsequent analysis.
 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)
 tabS1


### Grafico conjunto
# opar <-par(mfrow = c(2, 2), mar = c(4, 4, 2, 2))
# Ptxt = c("A", "B", "C", "D")
# for (i in 5:8) {

#      print(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"))
#
#                   }

##################################################
### chunk number 9: Compensation
##################################################

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

 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)
 tabS2

# 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
#####################################################################
### This code don´t work #### rtamarit
###  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)
#####################################################################

# replaced by rtamarit new code below


 kmfilt1 <-kmeansFilter("FITC.A" = c("Low", "High"))
 filtrados1<-filter(PosTFS[2],kmfilt1)
 summary(filtrados1)
 names(filtrados1)
 summary(filtrados1$pid2)
 ## We can limit the splitting to one or several sub-populations
 FITC.high<-split(PosTFS[2], filtrados1, population="High")
 FITC.high
 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data=FITC.high$High ))



 kmfilt2 <-kmeansFilter("PE.A" = c("Low", "High"))
 filtrados2<-filter(PosTFS[3],kmfilt2)
```

```
 summary(filtrados2)
 names(filtrados2)
 summary(filtrados2$pid3)
 ## We can limit the splitting to one or several sub-populations
 PE.high<-split(PosTFS[3], filtrados2, population="High")
 PE.high
 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data= PE.high$High))



 kmfilt3 <-kmeansFilter("APC.A" = c("Low", "High"))
 filtrados3<-filter(PosTFS[4],kmfilt3)
 summary(filtrados3)
 names(filtrados3)
 summary(filtrados3$pid3)
 ## We can limit the splitting to one or several sub-populations
 APC.high<-split(PosTFS[4], filtrados3, population="High")
 APC.high

 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data= APC.high$High))

 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data= PE.high$High))

 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data= APC.high$High))

       FITC.high
       PE.high
       APC.high


#Combine resulting flowFrames into a flowSet
## This code don´t works fine. rtamarit
## FiltFS = flowSet(PosTFS[[1]], FITC.high$High[[1]], PE.high$High[[1]],
APC.high$High[[1]])
# replaced by rtamarit
 FiltFS = flowSet(PosTFS[[1]], FITC.high$High[[1]], PE.high$High[[1]], APC.high$High[[1]])
 FiltFS
 summary(FiltFS)
 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data=FiltFS))

#Calculate the background intensity for each parameter
 CMed = as.matrix(fsApply(FiltFS, each_col, median)[, -c(1:2,6)])
 CMed

#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]))
 bFiltFS
 summary(bFiltFS)
 x11()
 print(xyplot(`SSC.A` ~ `FSC.A`,data=bFiltFS))
 print(plot(bFiltFS, "FITC.A", breaks = 256))
xyplot(`SSC.A` ~ `FSC.A`,data=bFiltFS)
x11
xyplot(bFiltFS$V3)
splom(bFiltFS,`FSC.A`)

#Capture medians from single-stained flowFrames
 FObs = as.matrix(fsApply(bFiltFS[c(2:4)], each_col, median)[,-c(1:2,6)])
 FObs

#Estimate compensation spillover matrix and echo result

#diag has four distinct usages:

# x is a matrix, when it extracts the diagonal.
```

```
# x is missing and nrow is specified, it returns an identity matrix.
# x is a scalar (length-one vector) and the only argument,
#      it returns a square identity matrix of size given by the scalar.
# x is a vector, either of length at least 2 or there were further
#      arguments. This returns a matrix with the given diagonal and
#      zero off-diagonal entries.
       a= diag(FObs)
       a
       b= diag(a)
       b
## solve :This generic function solves the equation a %*% x = b for x,
## where b can be either a vector or a matrix.
##     ()

 fij = solve(FObs) %*% diag(diag(FObs))
 fij
 PosTFS


# 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]))
# bPosTFS


 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)

# cPosTFS


#####################################################
### chunk number 10: Linear-Log Data Transformation
#####################################################

 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))

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

 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))
       })
 x11()
 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)

########################################################
### chunk number 11: Analysis
########################################################
##
# Define positive limits
# CD44 - from CD4 single-stained control experiment

 # cPosTFS[[2]] is CD4 single-stained unsorted
       # cPosTFS[[2]][, 11] is the CD44 channel
       #   derived from transformation of cPE
 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)
             }
       CumV
       limCD441
       CD441


# CD44 - from CD62L single-stained control experiment

cPosTFS[[4]]
```

```
 # cPosTFS[[4]]is CD62L single-stained unsorted
       # cPosTFS[[4]][, 11] is the CD44 channel
 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)
        }
       limCD442
       CD442
       CD441$x[limCD441]
       CD442$x[limCD442]

# Max value of expression in the single-stain unsorted
 ValCD44 <- max(c(CD441$x[limCD441], CD442$x[limCD442]))
       ValCD44

 #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$x[limCD443]
 HiValCD44
       x11()
       par(mfrow=c(3,1))
       plot(CD441,
       main="CD44 - from CD4 single-stained control experiment",
              xlim=c(-1, 2),
              ylim=c(0, 6))
              abline(v=CD441$x[limCD441],      col="green")

   #   lines(CD442, col="red")

       plot(CD442,
              main="CD44 - from CD62L single-stained control experiment",
              xlim=c(-1, 2))
              abline(v=CD442$x[limCD442],      col="red")

       plot(CD443,
              main="Estimate high value for CD44 - from CD4+CD62L+ fraction",
              xlim=c(-1, 2))
              abline(v=CD443$x[limCD443],      col="blue")



# 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))
        })
 x11()
 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)),
        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))
        })


 x11()
 plot(tp1, position = c(0, 0, 0.5, 1), more = TRUE)
 plot(tp2, position = c(0.5, 0, 1, 1), more = FALSE)
```

```
 # 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)
 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))


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


 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)


##Marginalized Probability Density Functions
 # 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
x11()
 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
x11()
 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,
        kernel = "rect"), col = PCols[i], lwd = 2, lty = Plty[i])
        }


##Principal Component Analysis
 # 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)
 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))

 # 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

       x11()
       par(mfrow=c(1,3))
scoreCD4
head(scoreCD4[, 1:2])
head(scoreCD4[, 2:3])


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")

plot(scoreCD4[, 2:3], pch = 21, col = "blue", bg = "blue", cex = 0.5,
        xlab = "Principal Component 2", ylab = "Principal Component 3")
 cols2 <- densCols(scoreCD4CD62L[, 2:3], nbin = 30, colramp =
        colorRampPalette(c("white", "red")))
 points(scoreCD4CD62L[, 2:3], pch = 22, cex = 0.5, lwd = 0.25,
        bg = cols2, col = "red")

plot(scoreCD4[,1],scoreCD4[,3], pch = 21, col = "blue", bg = "blue", cex = 0.5,
        xlab = "Principal Component 1", ylab = "Principal Component 3")
 cols2 <- densCols(scoreCD4CD62L[,1], scoreCD4CD62L[,3], nbin = 30, colramp =
        colorRampPalette(c("white", "red")))
 points(scoreCD4CD62L[,1], scoreCD4CD62L[,3], pch = 22, cex = 0.5, lwd = 0.25,
        bg = cols2, col = "red")
```

ANÁLISIS DE DATOS DE CITOMETRIA DE FLUJO:

APLICACIONES DE R-BIOCONDUCTOR

Febrero 2010

Ramón Tamarit

Índice

¿BIOINFORMÁTICA?

- La bioinformática es la aplicación de tecnología informática a la gestión y análisis de datos biológicos.

- La bioinformática es una "*ciencia*" interdisciplinar, que requiere el uso o el desarrollo de diferentes técnicas para solucionar problemas, analizar datos, o simular sistemas o mecanismos, todos ellos de índole biológica y médica, y normalmente (pero no siempre) a nivel molecular

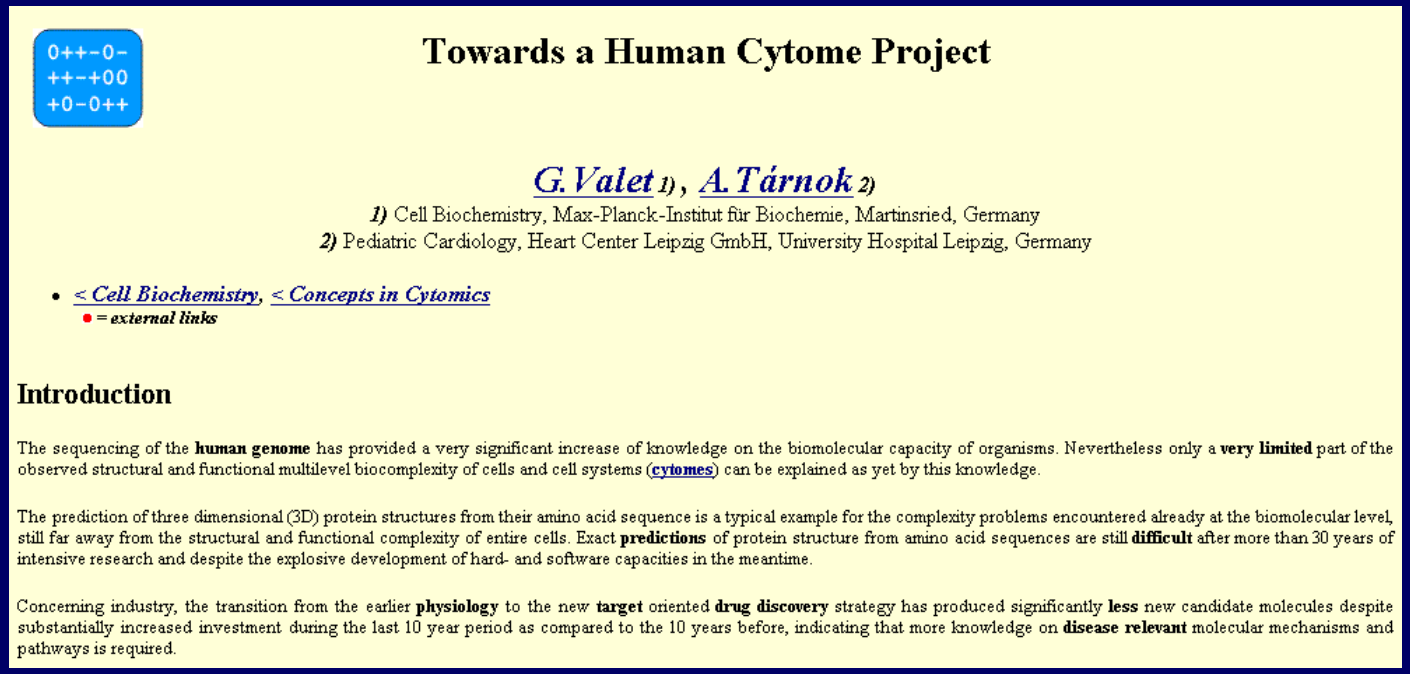
  - Técnicas empleadas:
  - Informática y ciencias de la computación
  - Matemática aplicada y estadística,
  - Inteligencia artificial,
  - Biología, bioquímica, química, física ….

- El ámbito de aplicación de la bioinformática se centra en solucionar o investigar problemas sobre escalas de tal magnitud que sobrepasan el discernimiento humano, haciéndose necesaria la utilización de recursos computacionales.

- Los principales esfuerzos de investigación en bioinformática se han centrado principalmente en aplicaciones genómicas, proteonómicas, evolutivas y metabolómicas.

- ¿CITÓMICA?
  - La citómica es el estudio de los fenotipos moleculares de las células individuales en combinación con una exhaustiva extracción bioinformática del conocimiento
  - Los citomas pueden ser definidos como los sistemas y subsistemas celulares y los componentes funcionales del organismo.
  - **La citómica** estudia la heterogeneidad de los citomas, resultante de
    - la expresión del genoma
    - y de la exposición de las células a factores externos

- ¿BIOINFORMÁTICA Y CITOMETRÍA DE FLUJO?
  - La creciente automatización y sofisticación de los clitómetros de flujo ha resultado en la potencialidad de generar de una cantidad de datos similar a la Genómica o Proteónomica, de hecho ya se considera a la citómica como una "ómica" más
  - En la publicación de Valet en 2005 ("*Human cytome project, cytomics, and systems biology: the incentive for new horizons in cytometry*") se plantean las necesidades tecnológicas bioinformáticas del **Proyecto del Citoma Humano**



0++-0-
++-+00
+0-0++

Towards a Human Cytome Project

G. Valet 1), *A. Tárnok* 2)

1) Cell Biochemistry, Max-Planck-Institut für Biochemie, Martinsried, Germany

2) Pediatric Cardiology, Heart Center Leipzig GmbH, University Hospital Leipzig, Germany

- *< Cell Biochemistry*, *< Concepts in Cytomics*
  - = *external links*

Introduction

The sequencing of the **human genome** has provided a very significant increase of knowledge on the biomolecular capacity of organisms. Nevertheless only a **very limited** part of the observed structural and functional multilevel biocomplexity of cells and cell systems (**cytomes**) can be explained as yet by this knowledge.

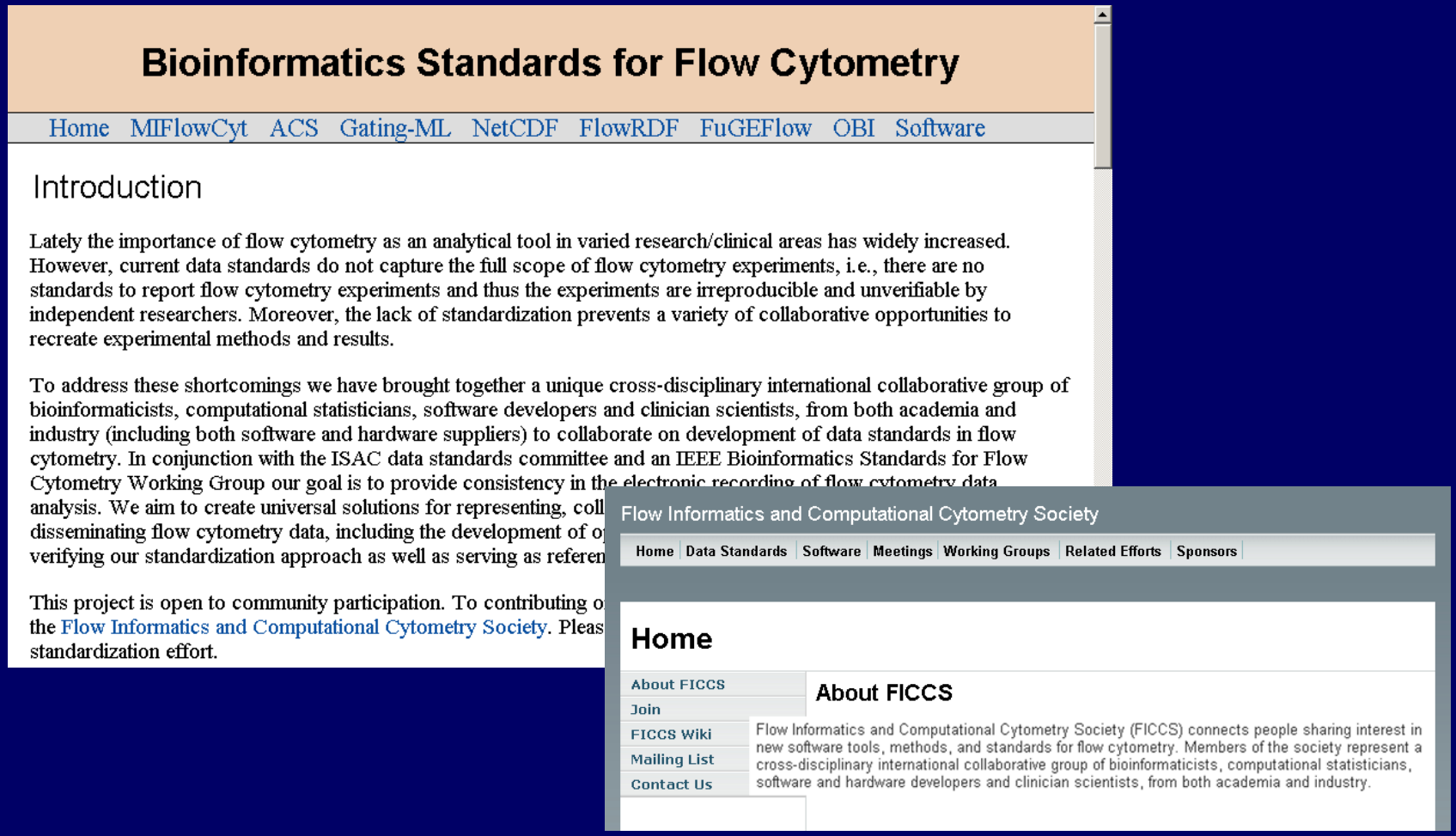
The prediction of three dimensional (3D) protein structures from their amino acid sequence is a typical example for the complexity problems encountered already at the biomolecular level, still far away from the structural and functional complexity of entire cells. Exact **predictions** of protein structure from amino acid sequences are still **difficult** after more than 30 years of intensive research and despite the explosive development of hard- and software capacities in the meantime.

Concerning industry, the transition from the earlier **physiology** to the new **target** oriented **drug discovery** strategy has produced significantly **less** new candidate molecules despite substantially increased investment during the last 10 year period as compared to the 10 years before, indicating that more knowledge on **disease relevant** molecular mechanisms and pathways is required.

CITÓMICA Y CITOMETRÍA DE FLUJO. IMPLICACIONES BIOINFORMÁTICAS

- ¿BIOINFORMÁTICA Y CITOMETRÍA DE FLUJO?
  - La creciente automatización y sofisticación de los clitómetros de flujo ha resultado en la potencialidad de generar de una cantidad de datos similar a la Genómica o Proteónomica, de hecho ya se considera a la citómica como una “ómica” más
  - En la publicación de Valet en 2005 (“*Human cytome project, cytomics, and systems biology: the incentive for new horizons in cytometry*”) se plantean las necesidades tecnológicas bioinformáticas del **Proyecto del Citoma Humano**
  - Hasta la fecha, los avances bioinformáticos han sido relevantes, pero aún existen diferencias sustanciales con el resto de “ómicas”. Entre otras se encuentran las siguientes:
    - No existe un repositorio público de “datos” de citometría
    - El software existente esta principalmente orientado a la “visualización de los datos”
    - las implementaciones orientadas al análisis estadístico y minería de datos son escasas

- El proyecto “Bioinformatics Standards for Flow Cytometry” auspiciado por la FICCS (Flow Informatics and Computational Cytometry Society) es la principal fuente de recursos y avances en Bioinformática aplicada a la citómica y citometría de Flujo.



CITÓMICA Y CITOMETRÍA DE FLUJO. IMPLICACIONES BIOINFORMÁTICAS

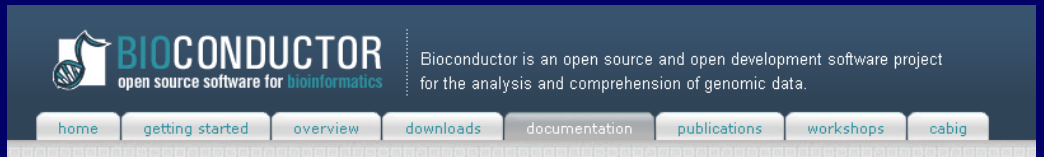
- El proyecto “Bioinformatics Standards for Flow Cytometry” se centra en dos elementos clave:
  - Diseño de bases de datos relacionales y estructuras de datos: La información obtenida de los experimentos de citometría tiene que indexarse en bases de datos integradas con el resto de “ómicas”
    - Para ello es necesario desarrollar las ontologías (OBI),
    - los estándares de almacenamiento y transmisión de datos junto con sus metadatos (XML-based standards),
    - Los modelos de objetos y los esquemas de base de datos
  - Desarrollo de herramientas de software: El software de tratamiento de datos debe ser desarrollado en base a los requerimientos de la capa inferior de diseño.
  - Desde el punto de vista bioinformático debe cumplir dos funciones esenciales:
    - Estandarizar los protocolos de análisis de datos
    - Encapsular las implementaciones estadísticas, facilitando el desarrollo de pasarelas de minería de datos

CITÓMICA Y CITOMETRÍA DE FLUJO. IMPLICACIONES BIOINFORMÁTICAS

- El software actual de análisis de datos de Citometría de flujo
- Lo podemos clasificar según la procedencia como:
  - **Software propietario del instrumento**:
    - normalmente esta orientado a la adquisición y visualización “tradicional” de los datos en tiempo real. véase por ejemplo el desarrollado por BD Biosciencies, o Amnis.
  - **Software de terceros en forma de “programa” ejecutable**.
    - Permite representar y analizar a posteriori los ficheros FCS obtenidos en el instrumento. Como ejemplos de este tipo de software tenemos FlowJo, WinMDI, FCS Express
    - El principal inconveniente de este tipo es que no permite introducir modificaciones metodológicas en el flujo de trabajo
    - Su principal ventaja es la facilidad de uso y productividad
  - **Software especifico para el análisis y minería de datos**, como por ejemplo SPSS, MatCad, R, S-Plus e incluso la hoja de cálculo Excel.
    - Este tipo de software tiene el inconveniente de su dificultad de uso ya que en muchos casos hay que programar los protocolos de trabajo.
    - Sin embargo todas las aplicaciones avanzadas de minería de datos de Citometría de flujo se están desarrollando bajo este tipo de plataformas, ya que ofrecen la posibilidad de implementar técnicas avanzadas

- Proyecto R-Bioconductor
- La implementación de la capa de análisis y minería de datos impulsada por la FICCS se esta realizando dentro del proyecto Bioconductor, empleando R como herramienta de programación.

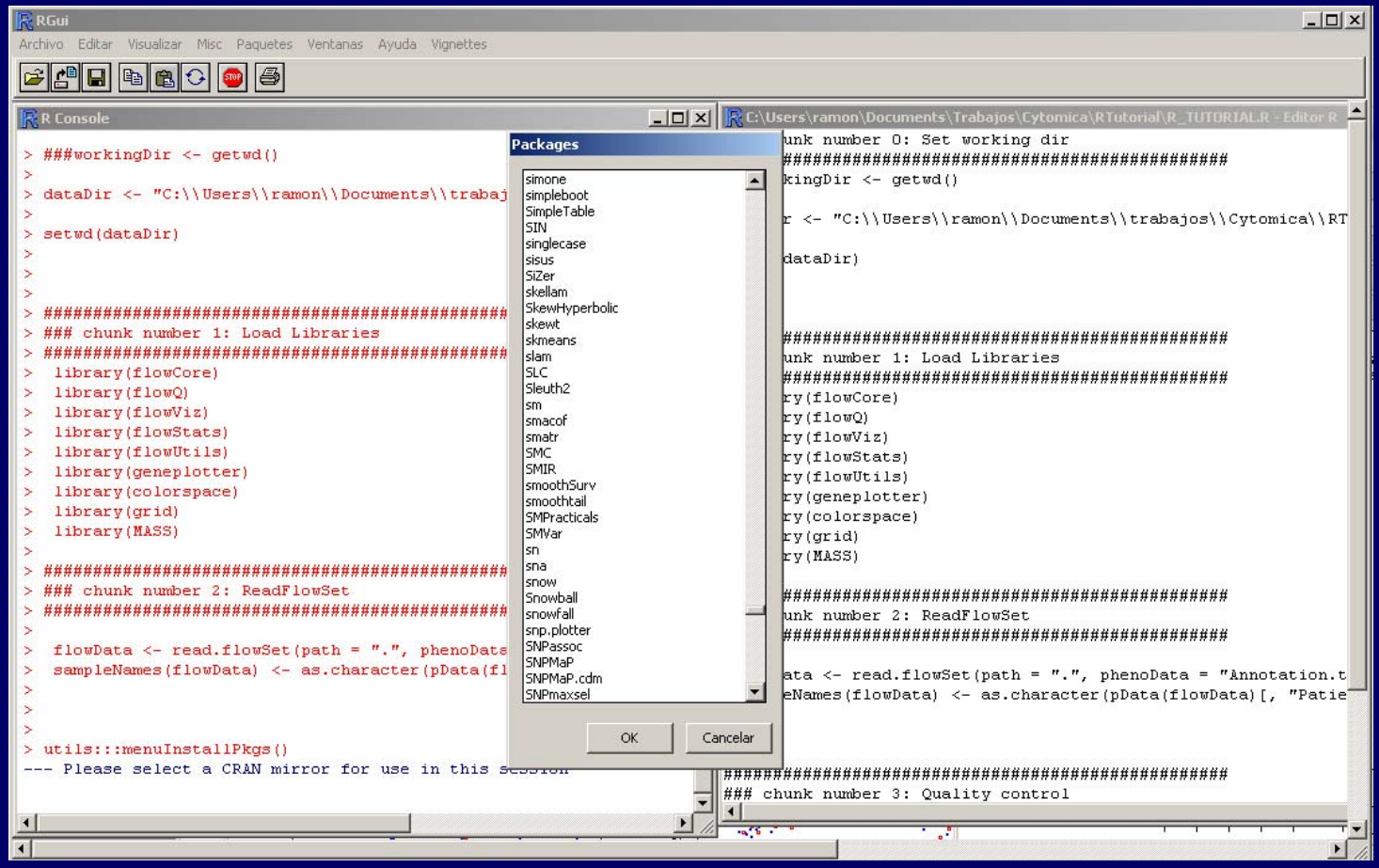
- El resultado es un conjunto de paquetes que resuelven las necesidades básicas de análisis estadístico de cualquier estudio de Citometría de flujo ya sea los tradicionales o los de alto rendimiento.



Módulos bajo flowCore	
flowCore	**Es modulo principal encargado de importar y preprocesar los datos. Los objetos generados por este modulo se pueden analizar mediante las implementaciones del resto de módulos.**
flowViz	Métodos gráficos para la visualización grafica
flowQ	Control de calidad de los datos
flowStats	Métodos estadísticos adicionales a flowCore
flowUtils	Utilidades para integrar modelos de datos de otro mediante XML.
flowClust	Clustering mediante "t mixture models with Box-Cox transformation"
flowMerge	Herramientas para automatizar el modelo de clustering de flowClust, creando filtros automáticos.
flowFP	Creación de huellas dactilares a partir de datos de Citometría de flujo.
flowFlowjo	Importación de espacios de trabajo de FlowJo.
Módulos independientes de flowCore	
prada	Conjunto de herramientas para fenotipado con cellHTS2
cellHTS2	Conjunto de herramientas para análisis de datos de FCHS (flow cytometry high-content screening)
plateCore	Conjunto de herramientas para análisis de datos de FCHS
rflowcyt	Métodos estadísticos adicionales a flowCore

Tabla 1 .- Módulos para el análisis de datos de Citometría de Flujo con R-Bioconductor.

- Proyecto R-Bioconductor. Descripción de R
- R es un programa basado en scripts y módulos de código (biocondutor)
- Funciona en modo consola en diversos sistemas operativos
- No es un programa tipo “excel”, los comandos hay que escribirlos ….. O ejecutarlos desde un script o programa …



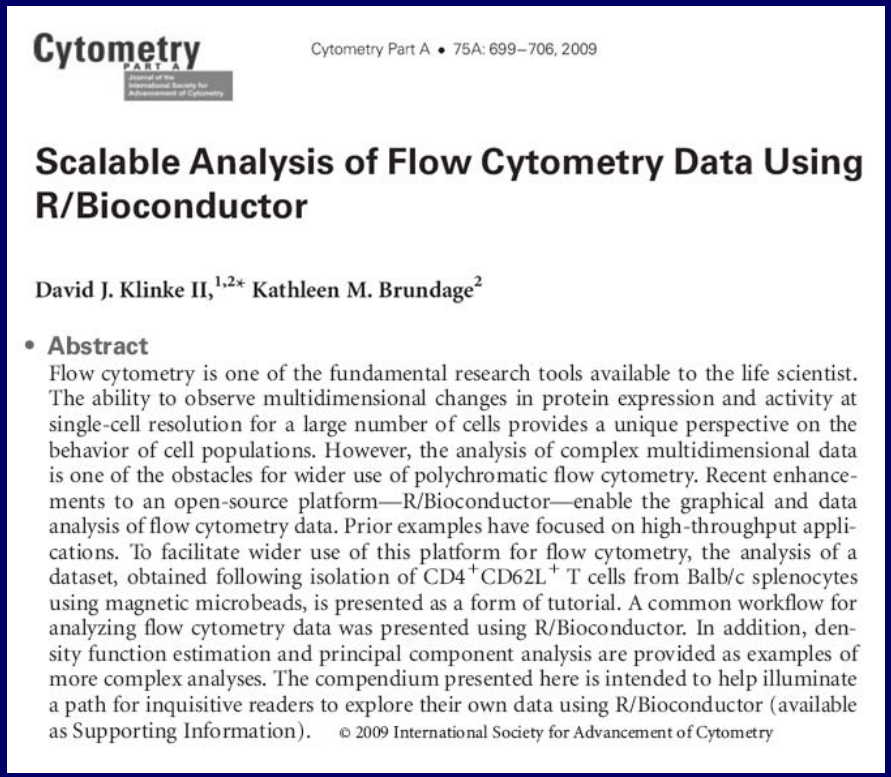
- Las ventajas de R-Bioconductor sobre otras plataformas son :
  - Se posee toda la experiencia de las implementaciones de microarrays,
  - es software libre
  - es multiplataforma (Unix, Linux, windows, Mac)
  - dispone prácticamente de todas las técnicas estadísticas implementadas a bajo nivel

- Cumple todos los requisitos necesarios para integrarse como herramienta bioinformática para la investigación básica

- Los inconvenientes de R quedan relegados a un segundo término debido a reciente incorporación del paquete iFlow como interface grafica, y a la posibilidad de importar y exportar espacios de trabajo de flowCore a FlowJo

ANÁLISIS DE DATOS A PEQUEÑA ESCALA CON BIOCONDUCTOR

- En este apartados veremos las estructuras y protocolos propuestos por R-Bioconductor, para manejar los datos de citometría de flujo a través de las principales etapas del pre-procesamiento: la compensación, transformación, filtrado, y el posterior análisis de datos.
- Como ejemplo, revisaremos el código y resultados de un experimento publicado recientemente por David J. Klinke , cuyos datos son públicos, y esta distribuido dentro del proyecto bioconductor en forma de vignette



Cytometry PART A
Cytometry Part A • 75A: 699–706, 2009

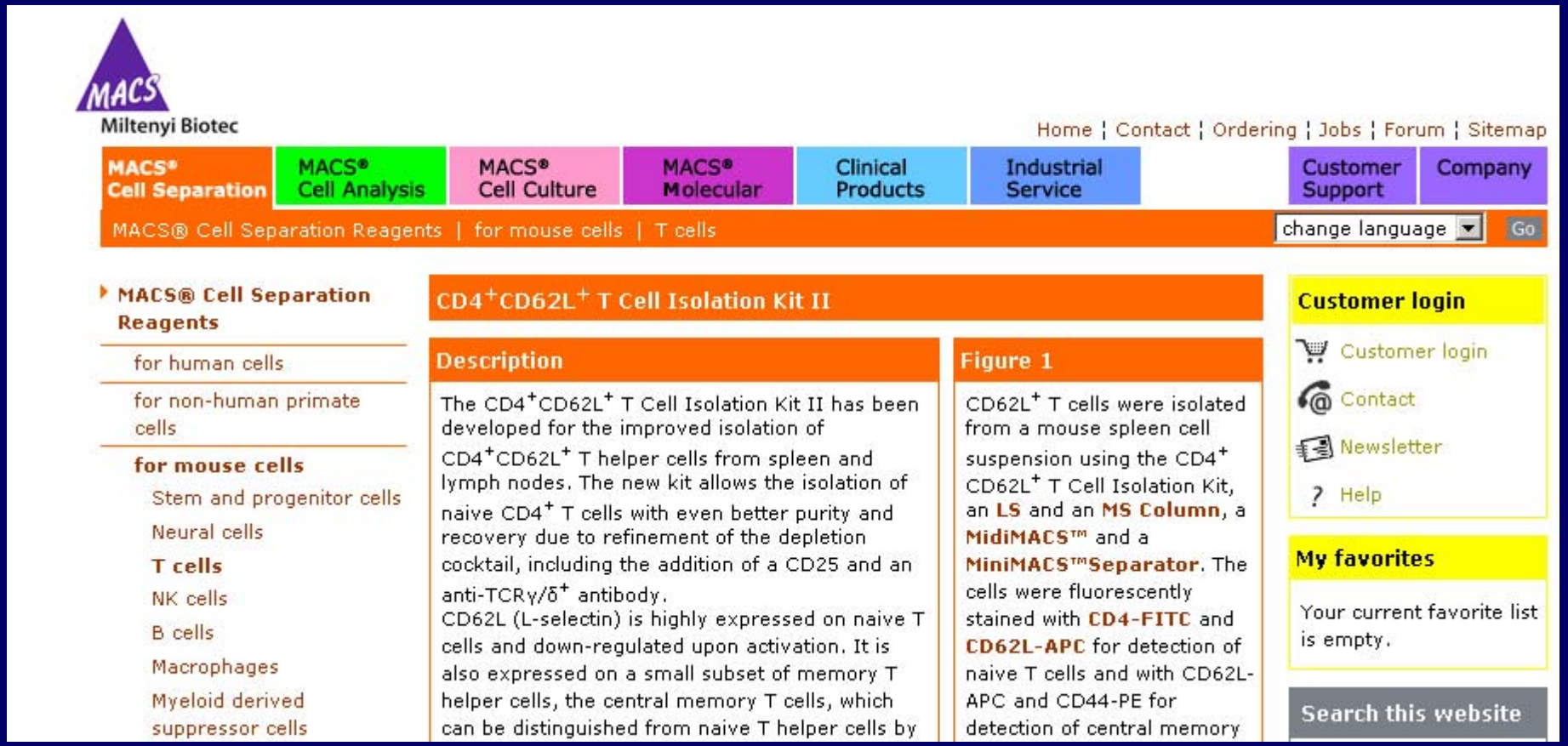
Scalable Analysis of Flow Cytometry Data Using R/Bioconductor

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

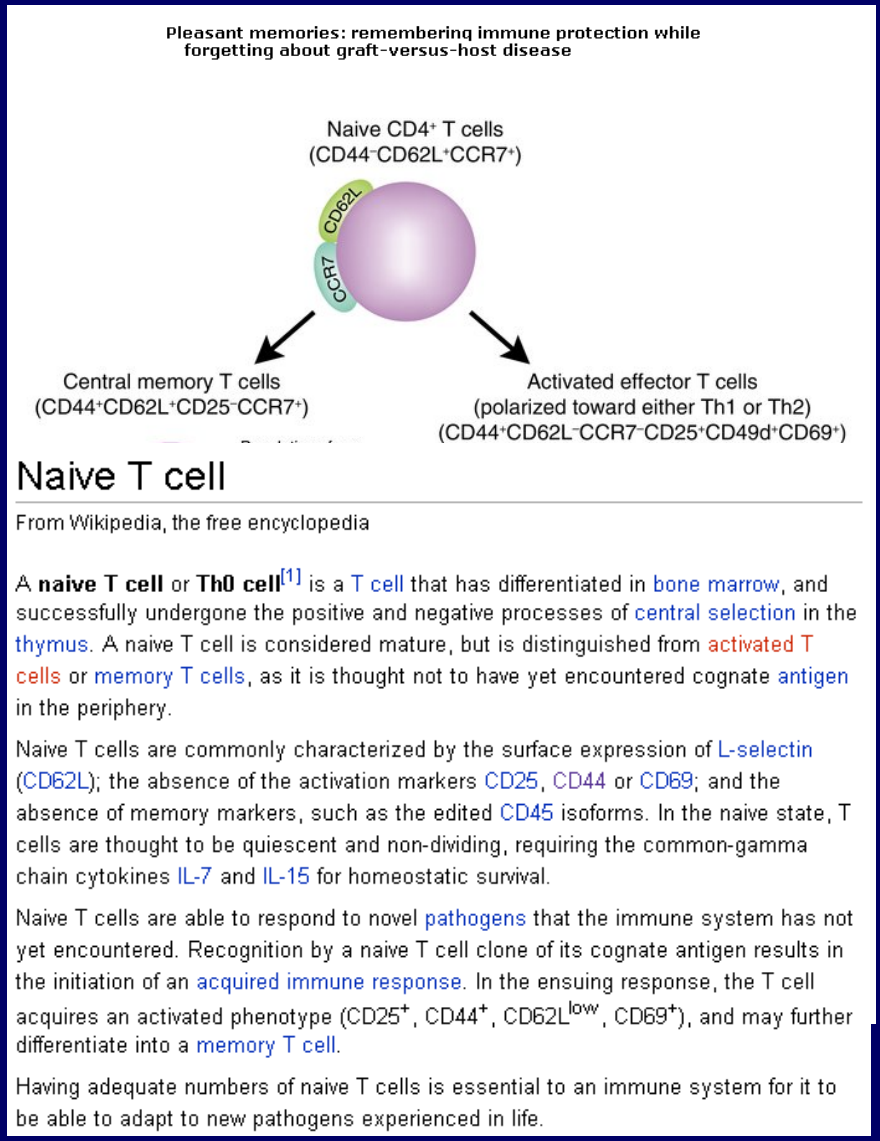
• **Abstract**

Flow cytometry is one of the fundamental research tools available to the life scientist. The ability to observe multidimensional changes in protein expression and activity at single-cell resolution for a large number of cells provides a unique perspective on the behavior of cell populations. However, the analysis of complex multidimensional data is one of the obstacles for wider use of polychromatic flow cytometry. Recent enhancements to an open-source platform—R/Bioconductor—enable the graphical and data analysis of flow cytometry data. Prior examples have focused on high-throughput applications. To facilitate wider use of this platform for flow cytometry, the analysis of a dataset, obtained following isolation of $CD4^+CD62L^+$ T cells from Balb/c splenocytes using magnetic microbeads, is presented as a form of tutorial. A common workflow for analyzing flow cytometry data was presented using R/Bioconductor. In addition, density function estimation and principal component analysis are provided as examples of more complex analyses. The compendium presented here is intended to help illuminate a path for inquisitive readers to explore their own data using R/Bioconductor (available as Supporting Information).

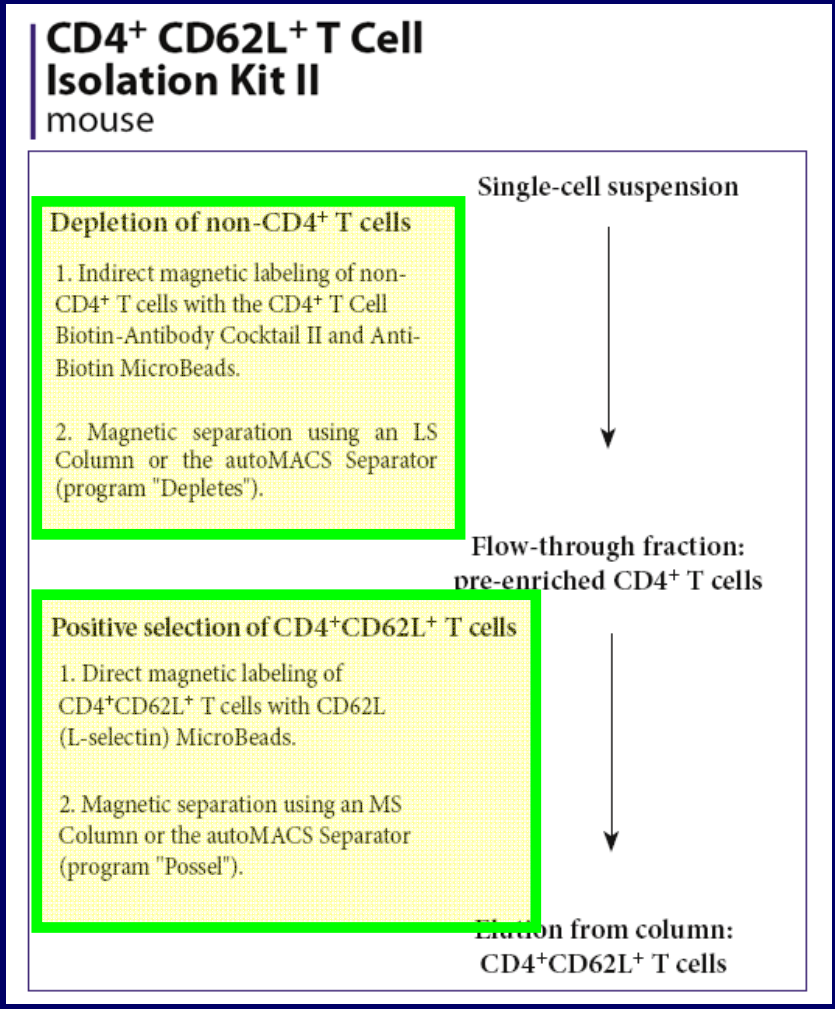
- Descripción del experimento:
- El objetivo final es comprobar la eficacia de un kit de separación de Linfocitos CD4+/CD62L+ mediante la técnica de micro-esferas magnéticas.



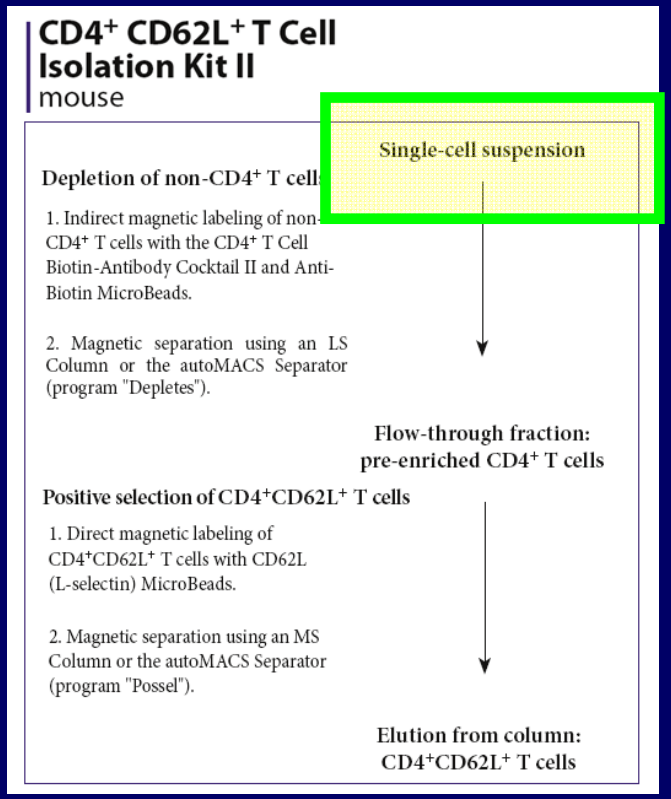
- Descripción del experimento:
  - Los linfocitos se extraen del bazo de ratones balb/c
  - Para comprobar los resultados se emplean tres marcadores: CD4, CD44 y CD62L.
    - El CD4 confiere a la célula papel de “Helper T-Cell”.
    - El CD44 es un marcador de “Effector-memory T-cells”.
    - El marcador CD62L, permite diferenciar los linfocitos que aún no responden a ningún patógeno (Naive T-Cell) de los que si (Memory T-Cell).

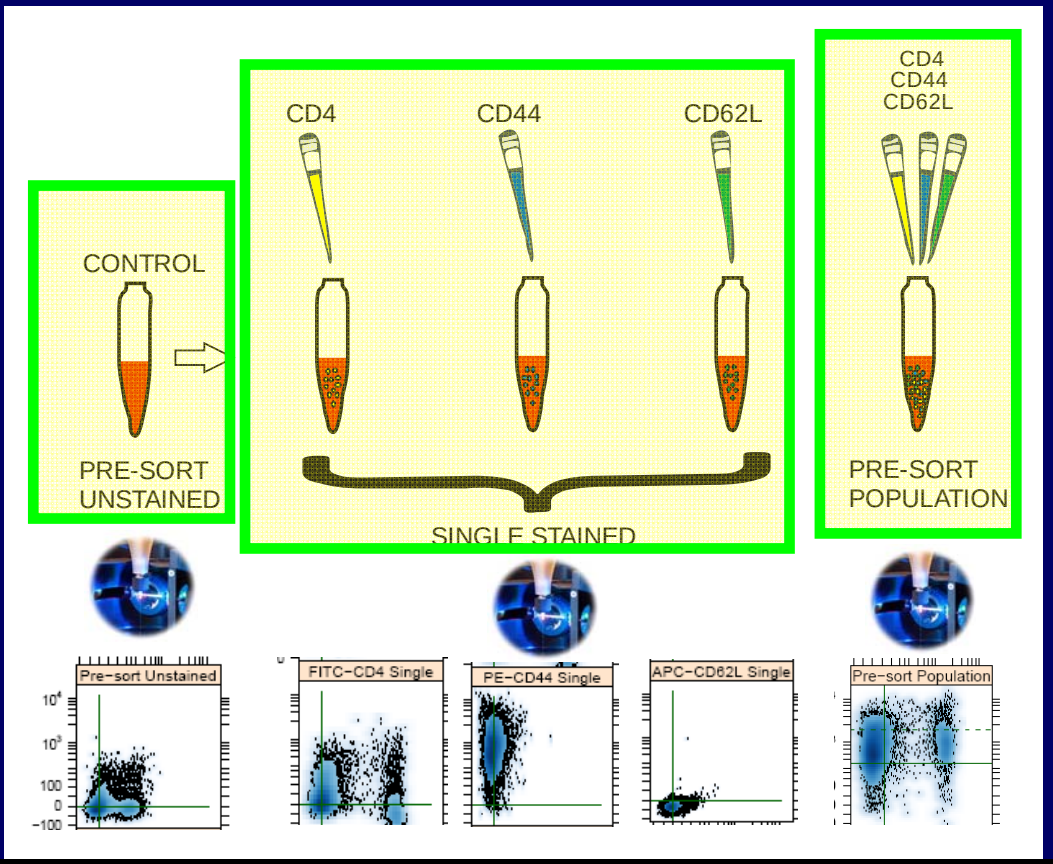


- Descripción del experimento:
  - El proceso de separación consta de dos etapas.
    - Primero se separan los CD4+ de los CD4-.
    - Segundo, la alícuota de CD4+ se trata para separar los CD62L- de los CD62L+.

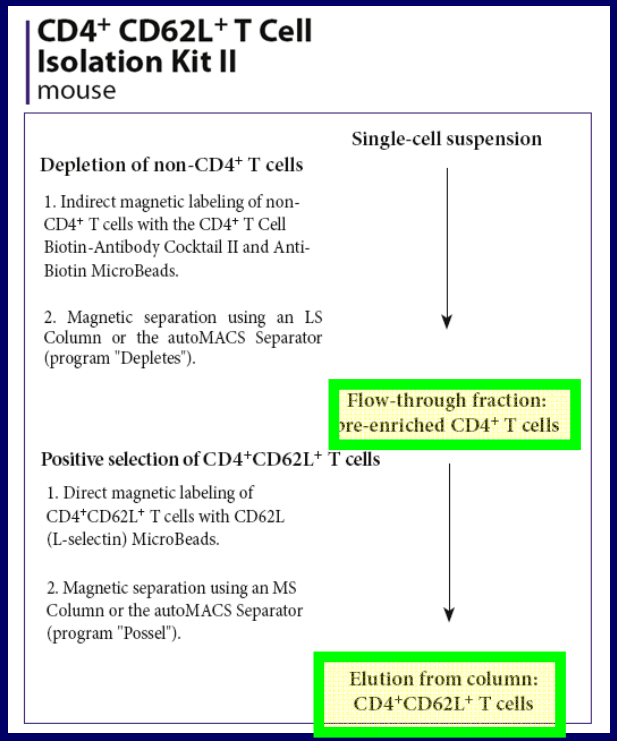


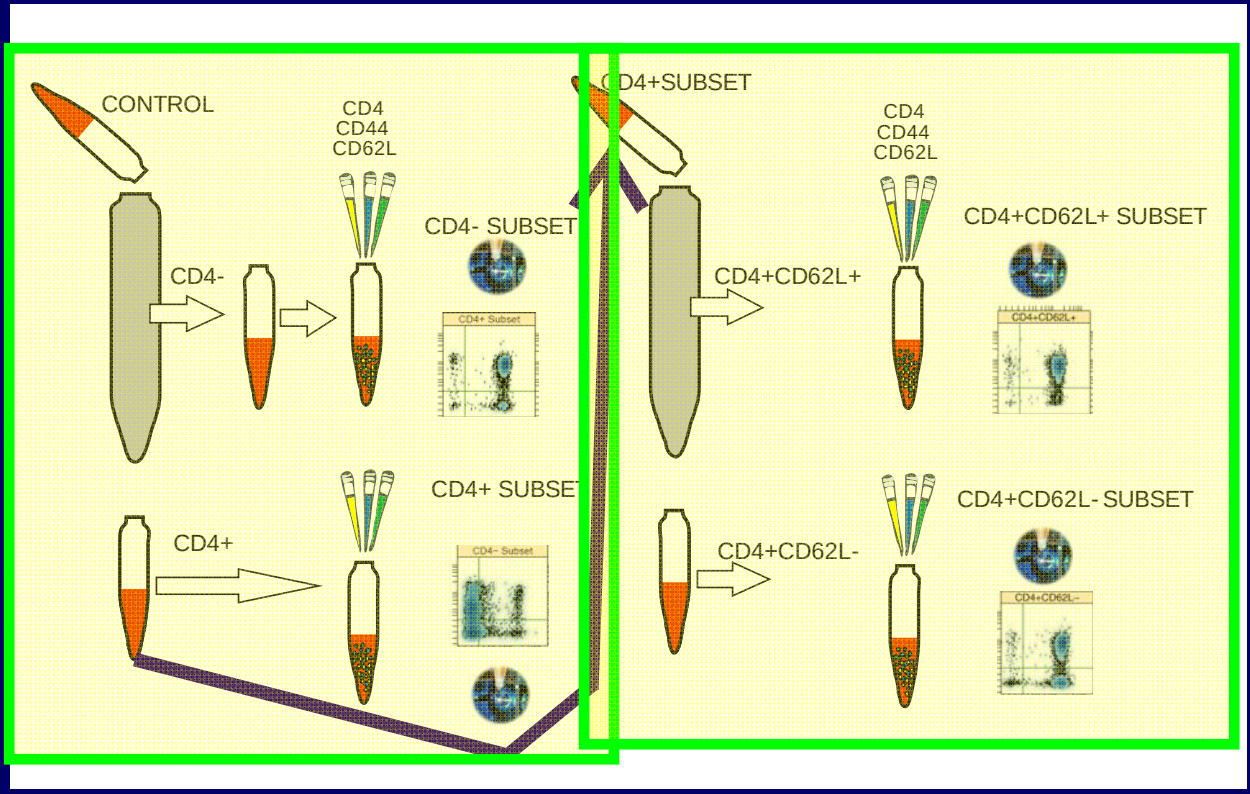
- Descripción del experimento.
- De la muestra control se realizan citometrías de flujo de:
  - la muestra control (“pre-sort unstained”),
  - de los tres marcadores por separado (“single-stained”),
  - y de todos los marcadores juntos (“pre-sort population”).



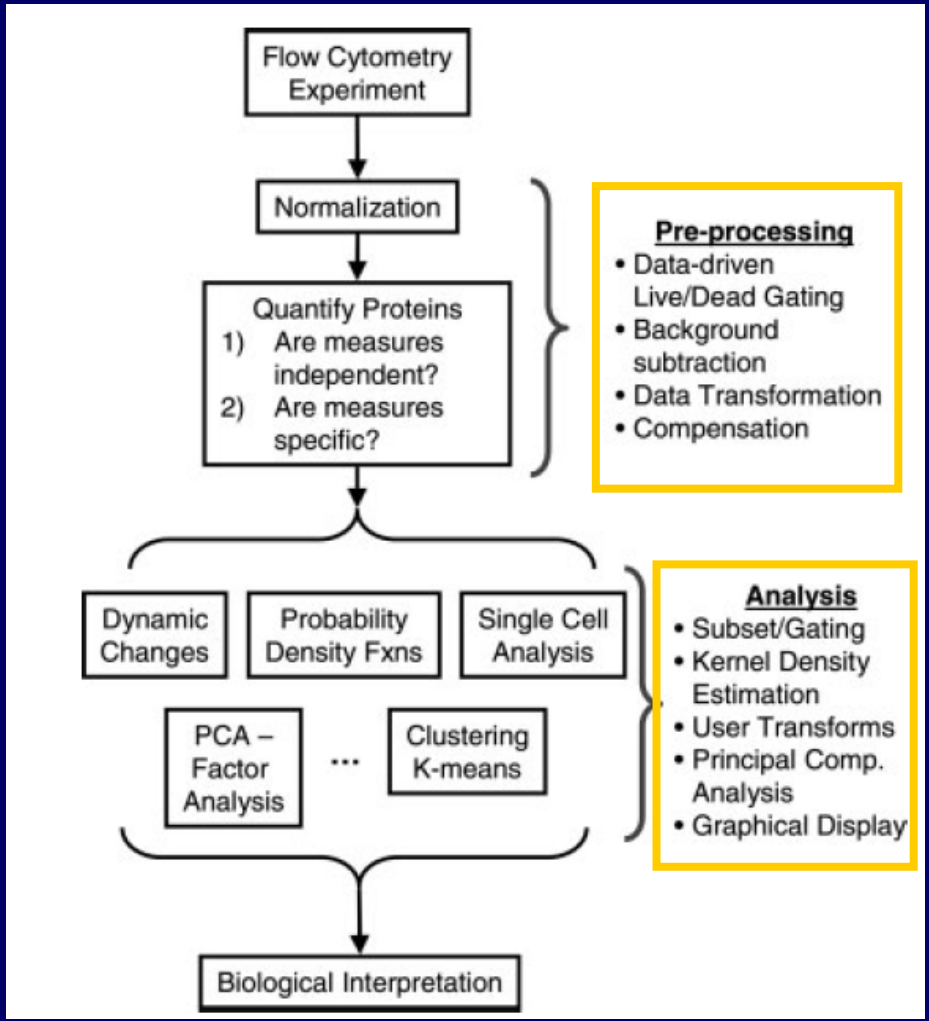


- Descripción del experimento.
- De cada una de las alícuotas obtenidas en las etapas de separación se realiza una Citometría de flujo de:
  - Primera etapa: “CD4- subset” y “CD4+ subset”,
  - Segunda etapa: “CD4+CD62L+ subset” y “CD4+CD62L- subset”





- El Flujo de trabajo se divide en dos etapas
  - **Preprocesado**:
    - Normalización
    - Control de calidad
    - Limpieza: Vivos/Muertos
    - Compensación
    - Transformación
  - **Análisis**:
    - Análisis clásico:
      - Particionamiento por positivos/negativos
      - Recuento de estadísticas
    - Minería de datos: Búsqueda de grupos en los datos o fenotipado
      - Clustering
      - PCA
      - Kernels de densidad



- Formato de los datos. Ficheros FCS.
  - Los datos generados por la mayoría de los citómetros de flujo comercial se almacenan en el formato Flow Cytometry Standard (FCS).
  - Los ficheros FCS son binarios y no son tratables directamente.
  - Se necesita un estándar para poder manipularlos

ENERO 2010

Cytometry PART A

Cytometry Part A • 77A: 97–100, 2010

Data File Standard for Flow Cytometry, Version FCS 3.1

Josef Spidlen,[1] Wayne Moore,[2] David Parks,[3] Michael Goldberg,[4] Chris Bray,[5] Pierre Bierre,[6] Peter Gorombey,[7] Bill Hyun,[8] Mark Hubbard,[9] Simon Lange,[10] Ray Lefebvre,[11] Robert Leif,[12] David Novo,[13] Leo Ostruszka,[14] Adam Treister,[15] James Wood,[16] Robert F. Murphy,[17] Mario Roederer,[18] Damir Sudar,[19] Robert Zigon,[20] Ryan R. Brinkman[1*]

[1]Terry Fox Laboratory, BC Cancer Agency, Vancouver, British Columbia, Canada

[2]Department of Genetics, Stanford University School of Medicine, Stanford, California, USA

[3]Stanford Shared FACS Facility, Stanford University, Stanford, California, USA

[4]Becton Dickinson and Company, San Jose, California, USA

[5]Verity Software House, Topsham, Maine, USA

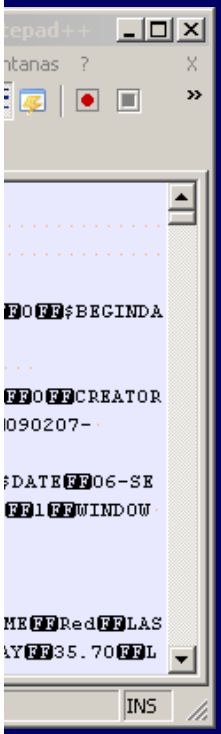
[6]Cytek Development, Fremont, California, USA

[7]Soft Flow Informatics, Debrecen, Hungary

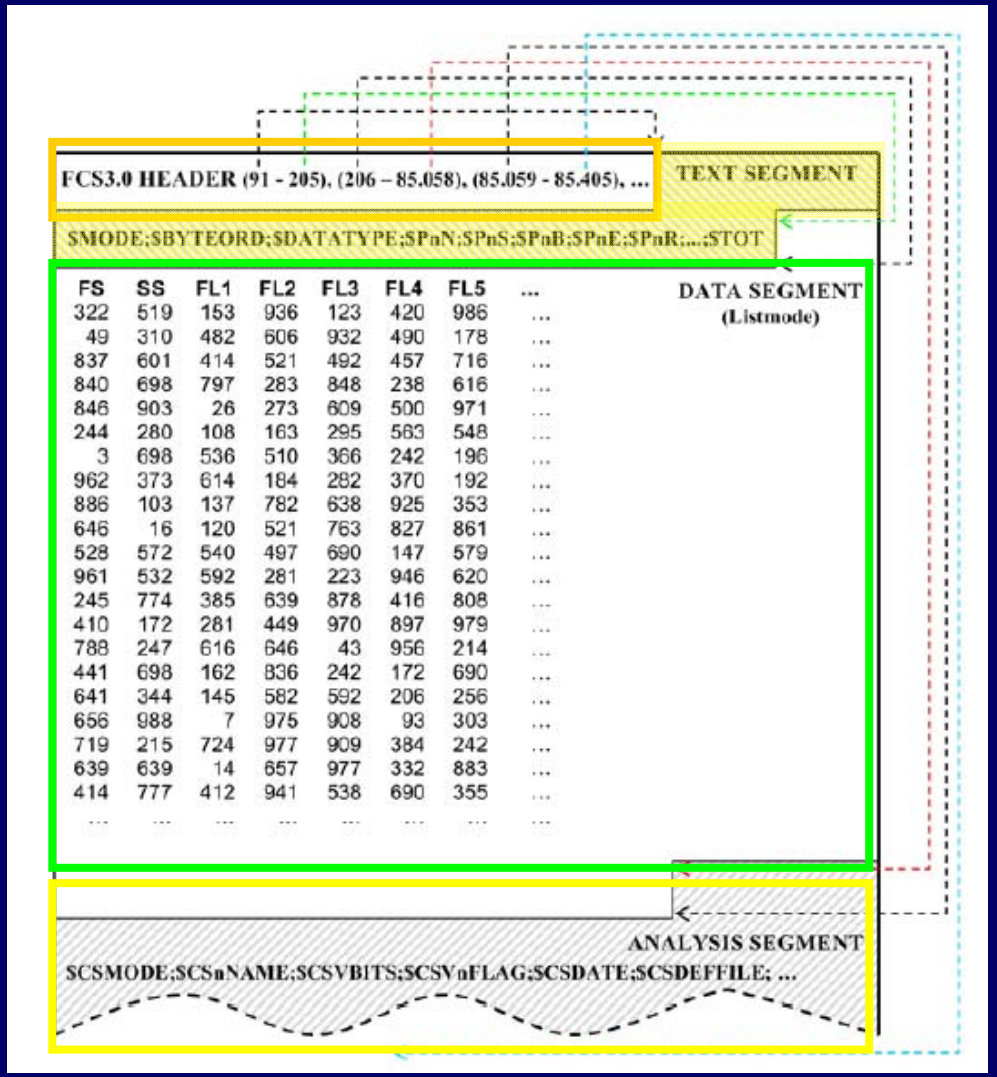
• Abstract

The flow cytometry data file standard provides the specifications needed to completely describe flow cytometry data sets within the confines of the file containing the experimental data. In 1984, the first Flow Cytometry Standard format for data files was adopted as FCS 1.0. This standard was modified in 1990 as FCS 2.0 and again in 1997 as FCS 3.0. We report here on the next generation flow cytometry standard data file format. FCS 3.1 is a minor revision based on suggested improvements from the community. The unchanged goal of the standard is to provide a uniform file format that allows files created by one type of acquisition hardware and software to be analyzed by any other type.

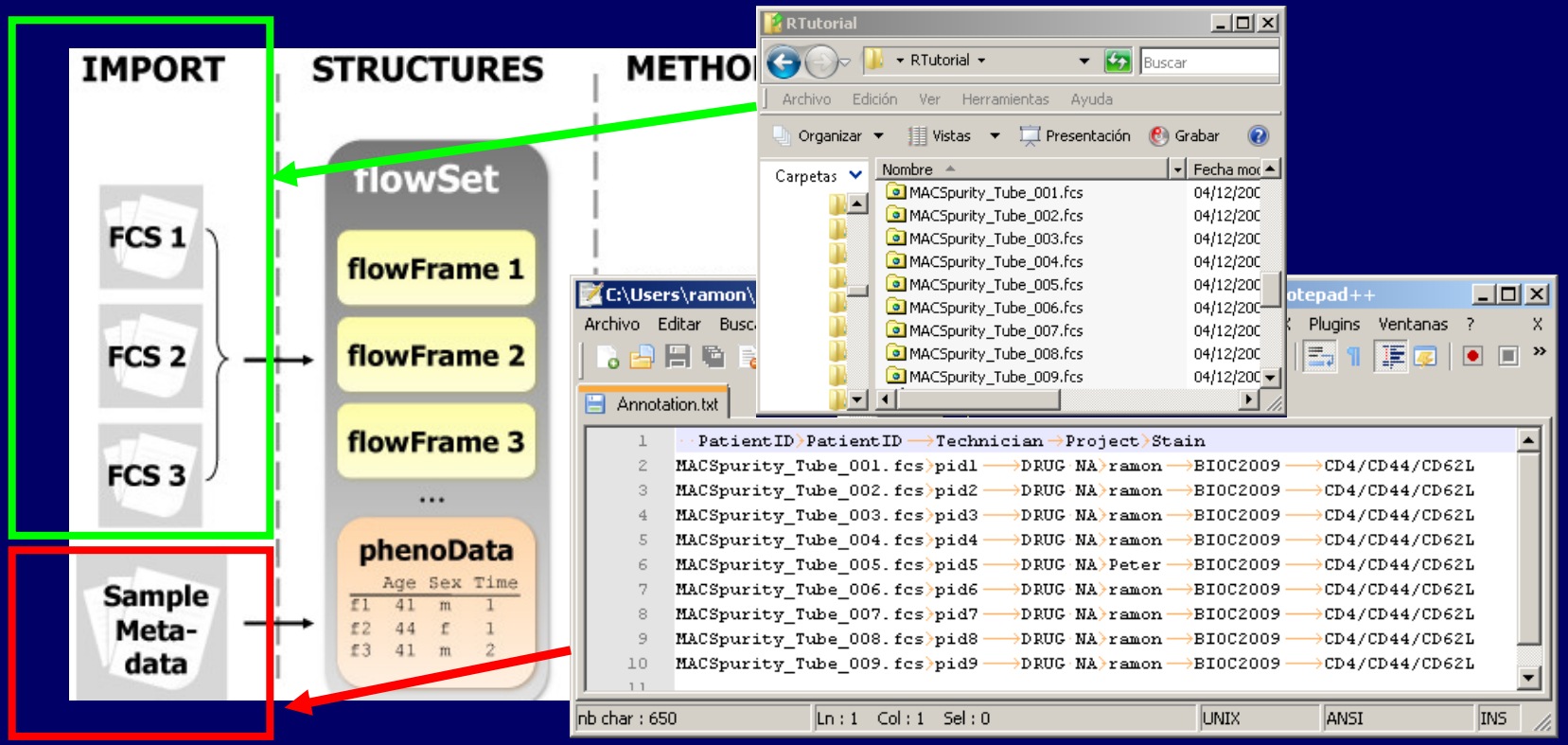
The FCS 3.1 standard retains the basic FCS file structure and most features of previous versions of the standard. Changes included in FCS 3.1 address potential ambiguities in the previous versions and provide a more robust standard. The major changes include simplified support for international characters and improved support for storing compensation. The major additions are support for preferred display scale, a standardized way of capturing the sample volume, information about originality of the data file, and support for plate and well identification in high throughput, plate based experiments. Please see the normative version of the FCS 3.1 specification in Supporting Information for this manuscript (or at http://www.isac-net.org/ in the Current standards section) for a complete list of changes.



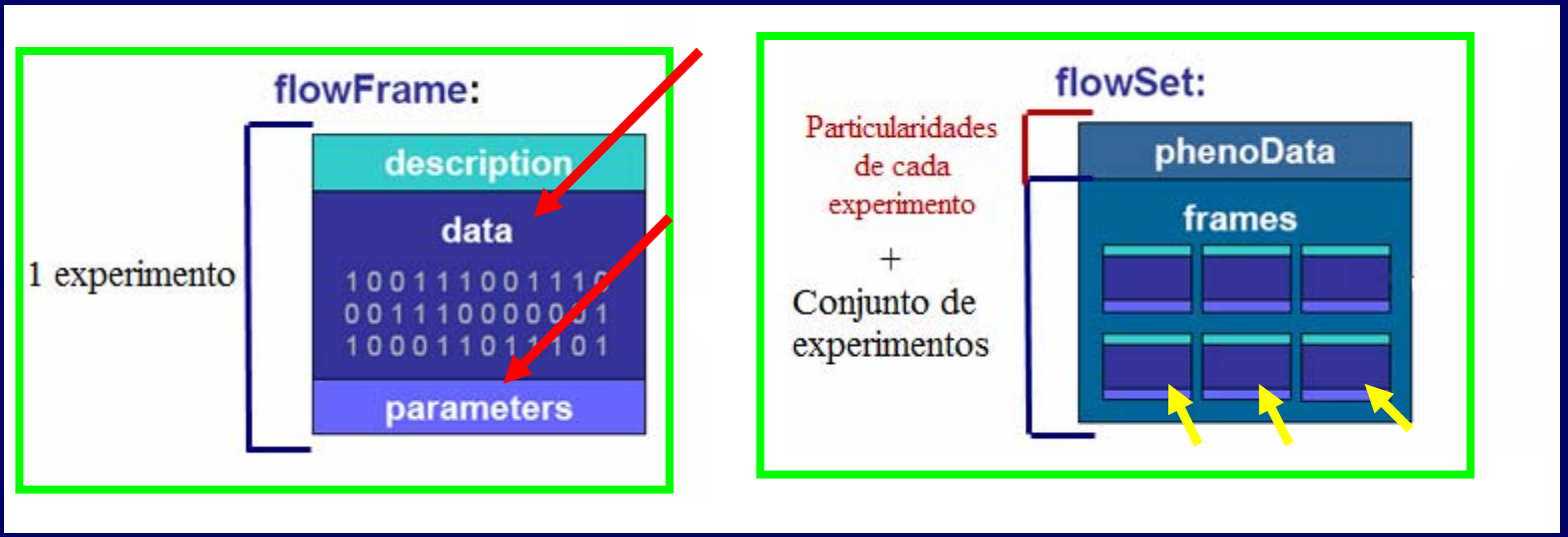
- Formato de los datos. Estructura de un fichero FCS.
  - Cabecera:
  - Segmento de Texto
  - Datos
  - Análisis



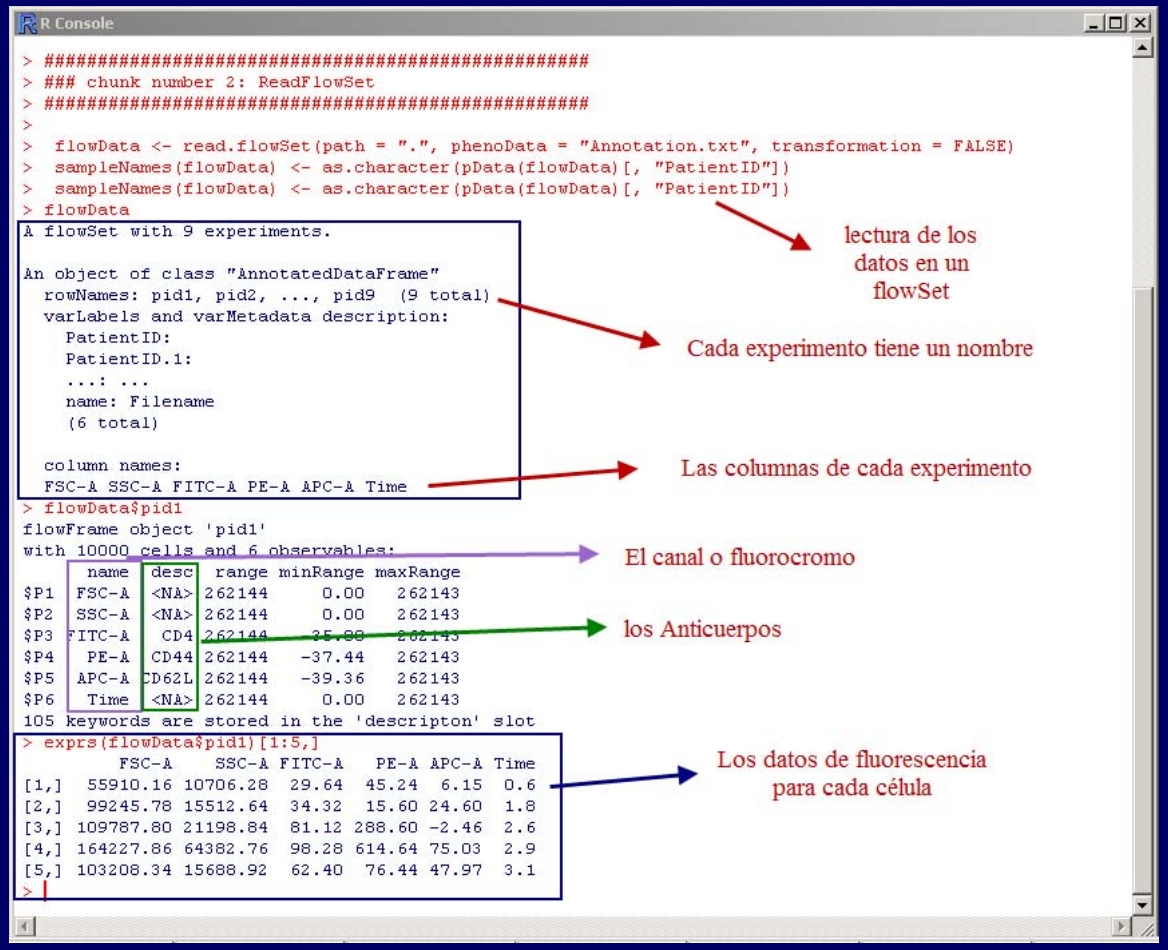
- Manipulación básica de los datos con flowCore
- La tarea principal del paquete flowCore es la adquisición, representación y manipulación básica de los datos de citometría de flujo. Esto se logra a través de un modelo de datos muy similar a la adoptada por otros paquetes de bioconductor.
- Para operar con los datos:
  - Cargaremos los ficheros FCS junto con su descripción en estructuras de datos denominadas flowFrame y flowSet



- Manipulación básica de los datos con flowCore
- La unidad básica de manipulación e información en flowCore es el flowFrame, que se corresponde con un solo archivo "FCS" (un tubo de experimento).
- Un flowFrame se compone de “slots”:
  - De “expresión” que con contienen la información a nivel de eventos (los resultados de fluorescencia de cada célula detectada), y
  - de “parámetros” que contiene los metadatos respectivamente.
- La mayoría de los experimentos (varios tubos) consisten en varios objetos flowFrame, que se organizan mediante un objeto flowSet.

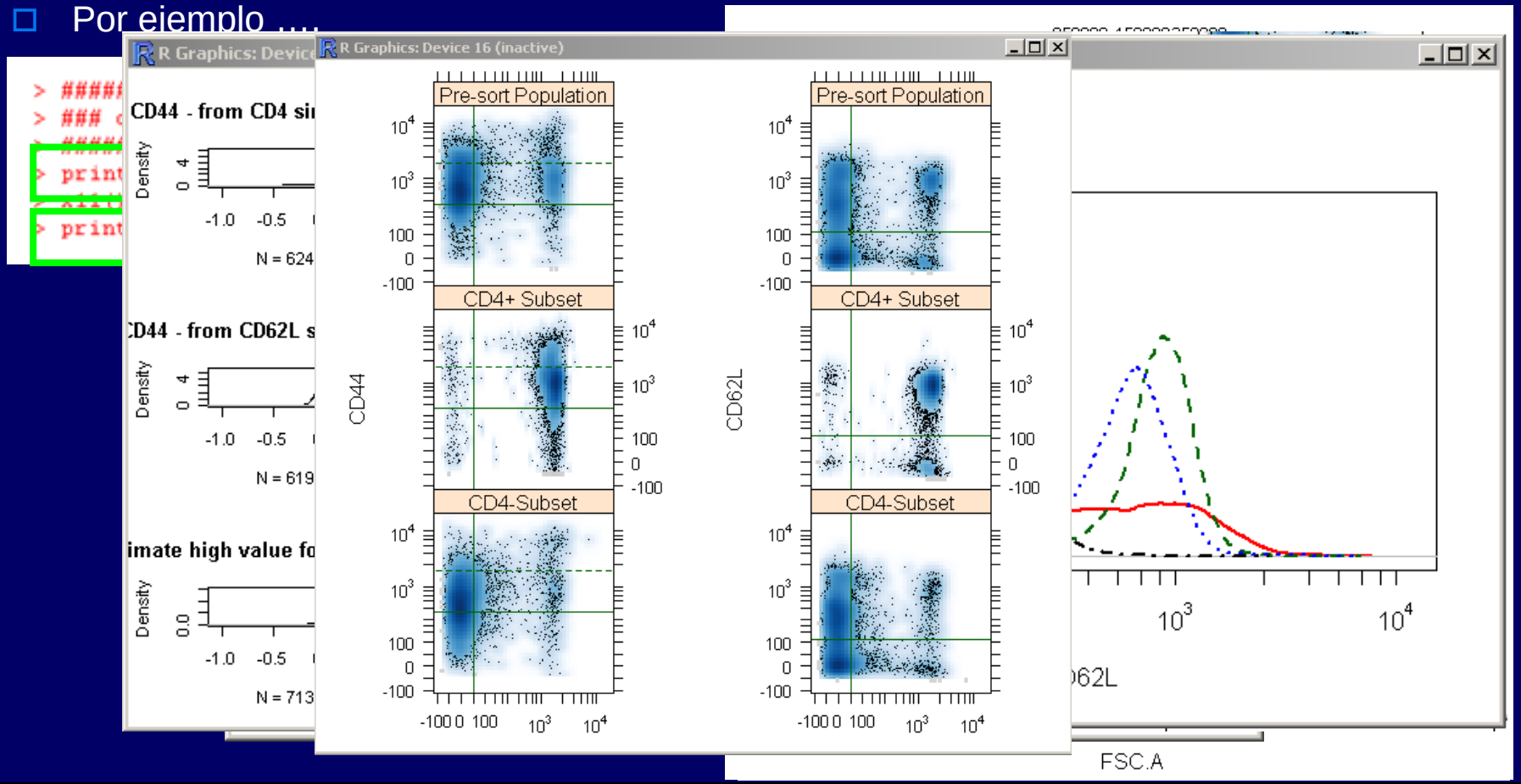


- Manipulación básica de los datos con flowCore
- Los datos de fluorescencia se almacenan como una matriz en el flowSet-flowFrame
- Los métodos de los paquetes de Bioconductor añaden diferentes funcionalidades a los flowSet o flowFrames



- Lectura de datos
- Inspección del flowSet
- Inspección de un flowframe
- Inspección de los datos de “expresión”

- Visualización de los datos
- La visualización más sofisticada de los flowFrame y objetos flowSet, se lleva a cabo por el paquete flowViz
- La lista de métodos de visualización de flowViz es muy extensa, prácticamente se pueden reproducir todos los gráficos existentes en la bibliografía.
- Por ejemplo ....



- Filtrado o gatting: Es la tarea más común en el análisis de los datos de citometría de flujo es el filtrado (o gatting), se usa para:
  - Obtener estadísticas de resumen sobre el número de eventos que cumplan determinados criterios o para
  - Realizar nuevos análisis en un subconjunto de los datos.
  - La mayoría de las operaciones de filtrado son una composición de una o más operaciones.
- La definición de los filtros ("gates") en flowCore sigue la "Gating Markup Language Candidate Recommendation" Spidlen et al. (2008), por lo que cualquier estrategia de filtrado de flowCore puede ser reproducida por cualquier otro software que siga el estándar, y viceversa, por ejemplo en flowJo.
  - Los filtros más simples, son los "gates" geométricos, que corresponden a los que se suelen encontrar en el software interactivo de la citometría de flujo como son los: filtros marginales, rectangulares, poligonales y elipsoidales.
  - Adicionalmente, se introduce el concepto filtros generados por la distribución estadística de los datos o "data-driven gates", concepto que no se encuentra bien definido en el software comercial de citometría de flujo.

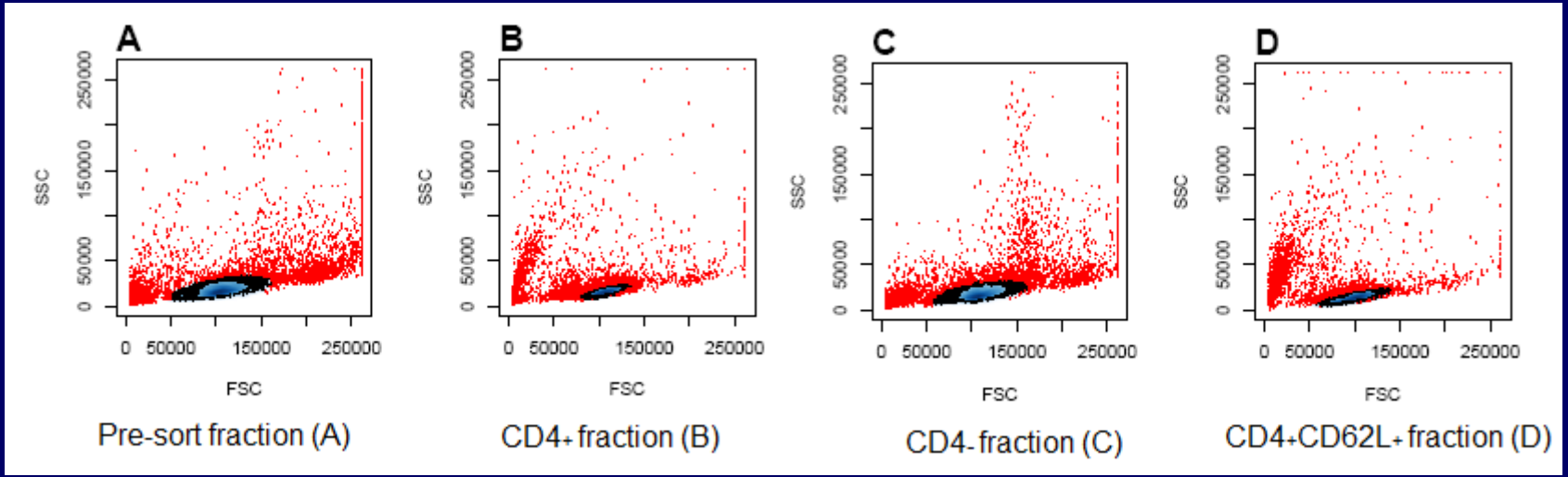
Table 2: Filter and gate classes implemented in flowCore.

	Gates
rectangleGate	n-dimensional rectangular regions
quadGate	quadrant regions in two dimensions
polygonGate	polygonal regions in two dimensions
polytopeGate	generalization of polygon in n dimensions
ellipsoidGate	n-dimensional ellipsoid region
	Filters
sampleFilter	random sub-sampling
expressionFilter	results of a boolean expression
kmeansFilter	K-means clustering
norm2Filter	bivariate normal distribution
curv1Filter	local density regions in 1D
curv2Filter	density regions in 2D
timeFilter	abnormal data acquisition over time
filterSet	gating strategies

Filters are automated, data driven procedures. Gates are static, user-defined methods.

- Filtrado o gatting . Los data-driven gates
- En el enfoque de “data-driven gates”, los parámetros necesarios se calculan sobre la base de las propiedades de los datos subyacentes, mediante un ajuste a una distribución determinada o por la estimación de la densidad de la población (gráficos PDF).
  - El filtro *norm2Filter* es un método robusto para encontrar la región que más se asemeja a una distribución normal bivariada,
  - El filtro *curv2filter*, Identifica las poblaciones sobre la base de clusters de densidad. Este último filtro permite separar múltiples poblaciones
- **Los “data-driven gates”, son independientes de la “mano” del operador y de las fluctuaciones del equipo entre experimentos**
  - **Son reproducibles entre operadores,**
  - **Son reproducibles entre experimentos**

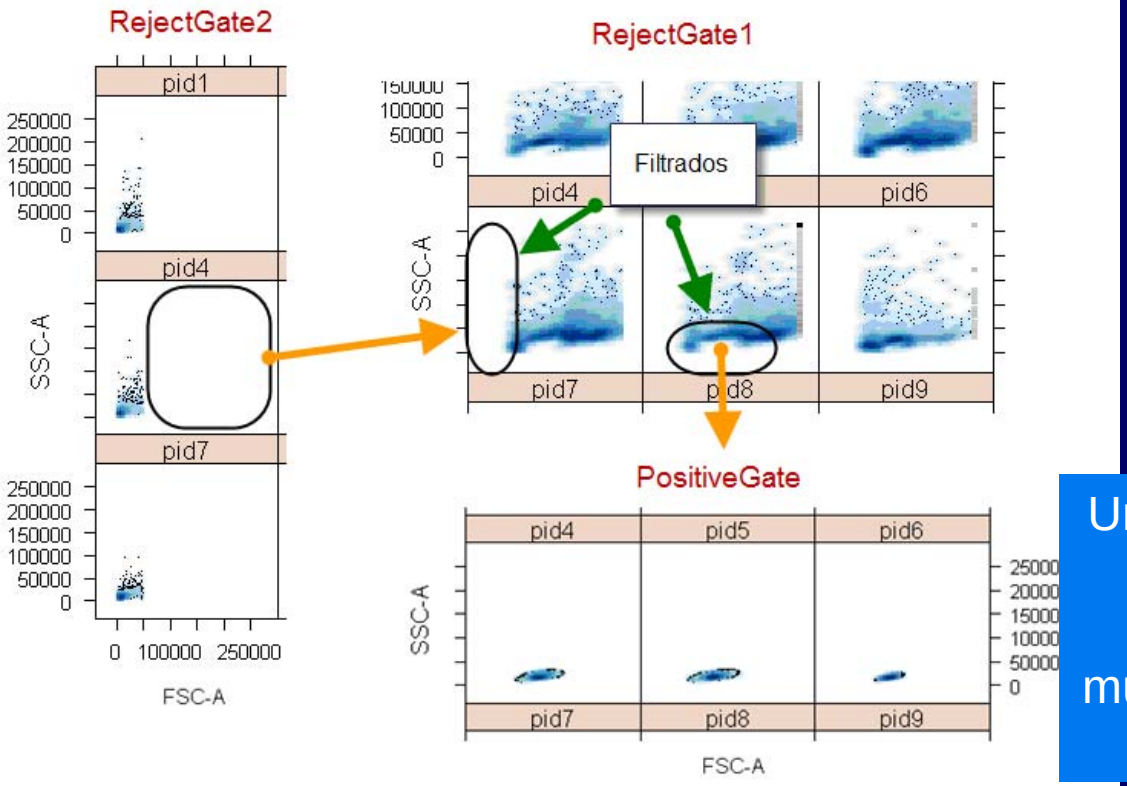
- Filtrado o gatting . Los data-driven gates
- En el ejemplo se usan para limpiar los restos no celulares y las células muertas mediante filtros en los canales FSC y SSC del experimento
- Los filtros asociados a los linfocitos vivos son:
  - Partículas con una intensidad de FSC mayor de 50000
  - Un filtro estadístico *norm2Filter* con los parámetros de dispersión frontal y lateral para crear una distribución normal (en dos dimensiones) que se centre en la mediana de la población de células y que encierre una región que incluya el 95% de la población (es decir, 2 desviaciones estándar).



- Filtrado o gatting . Como se aplican los data-driven gates

```
rectGate <-rectangleGate(filterId = "FSC+", "FSC-A" = c(50000,Inf))
morphGate <-norm2Filter(filterId = "MorphologyGate", "FSC-A",  "SSC-A", scale = 2)

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

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

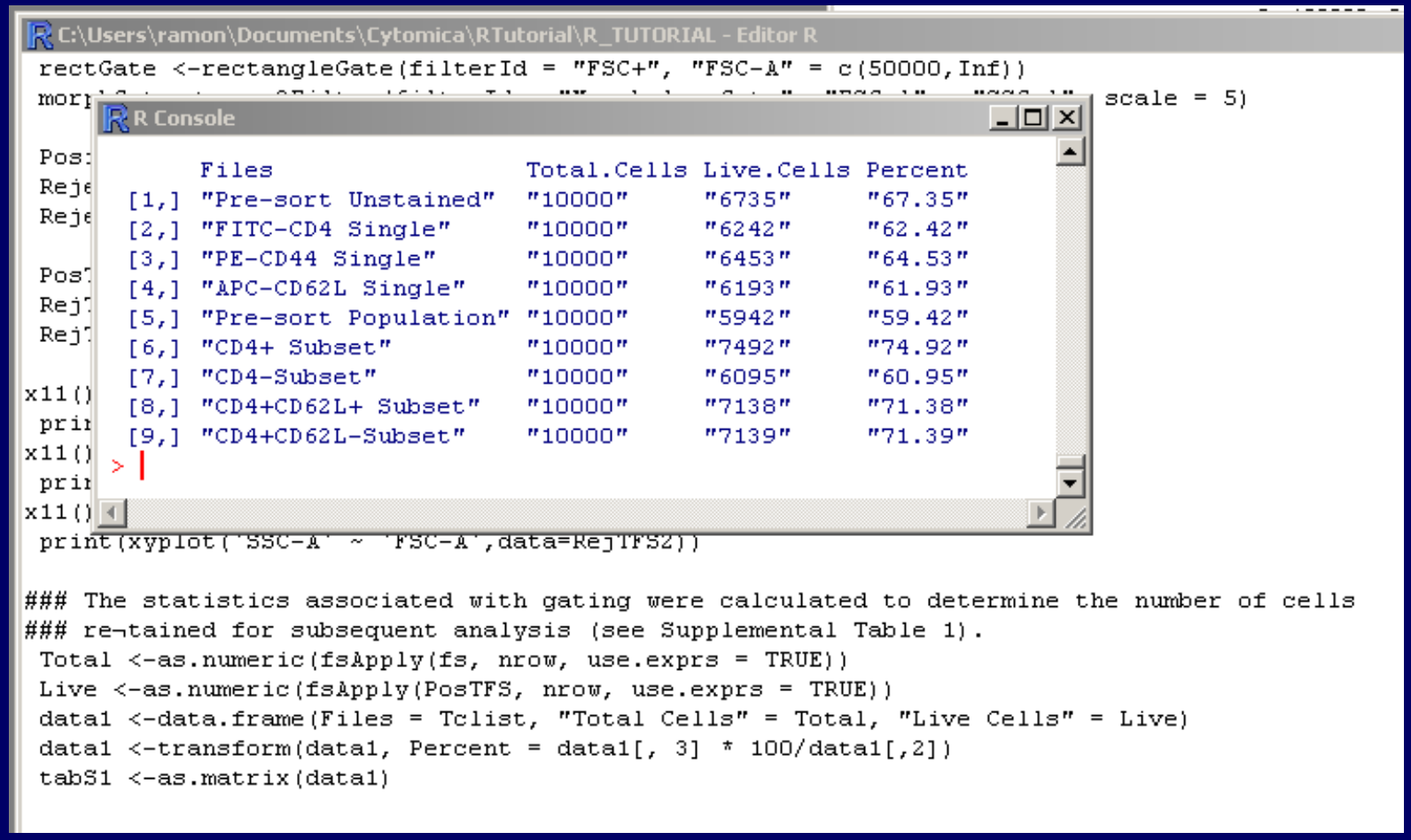
combinación de filtros



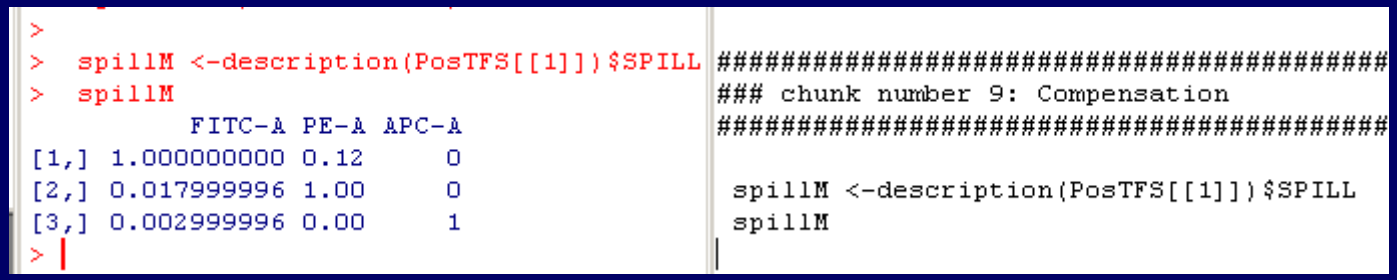
- Definir el filtro: Al filtro se le asigna un nombre
- Combinar el filtro
- Aplicar el filtro: El método Subset aplicado sobre fs (que es un flowset) genera un nuevo flowset filtrado
- Visualizar los resultados del nuevo flowset

Una de las ventajas de los filtros estadísticos de R, es que se pueden fácilmente aplicar a múltiples experimentos mediante el objeto flowSet

- Filtrado o gatting . Estadísticas
- Una vez establecidos los filtros se puede utilizar código R estándar para obtener cualquier estadística asociada a las poblaciones



- Compensación y corrección de fondo
- Dada la dificultad de determinar los valores de una compensación adecuada en tiempo real (durante la realización del experimento), los citómetros de flujo de incorporan diversos avances para el análisis de los datos de citometría de flujo.
  - Los controladores de software de los citómetros de flujo incluyen utilidades para calcular o estimar la matriz de compensación de fluorescencia
  - Los datos en bruto se almacenan en el fichero FCS sin compensación
  - La matriz de compensación empleada en la visualización en tiempo real se almacena en el fichero FCS
- En el ejemplo que nos ocupa la estimación inicial de la matriz de compensación empleada en el experimento se puede extraer de los metadatos de texto de MACSPurity_Tube_001.fcs. Esta es estimación inicial de la matriz de compensación se basa en experimentos anteriores y se utiliza para observar los datos durante la adquisición

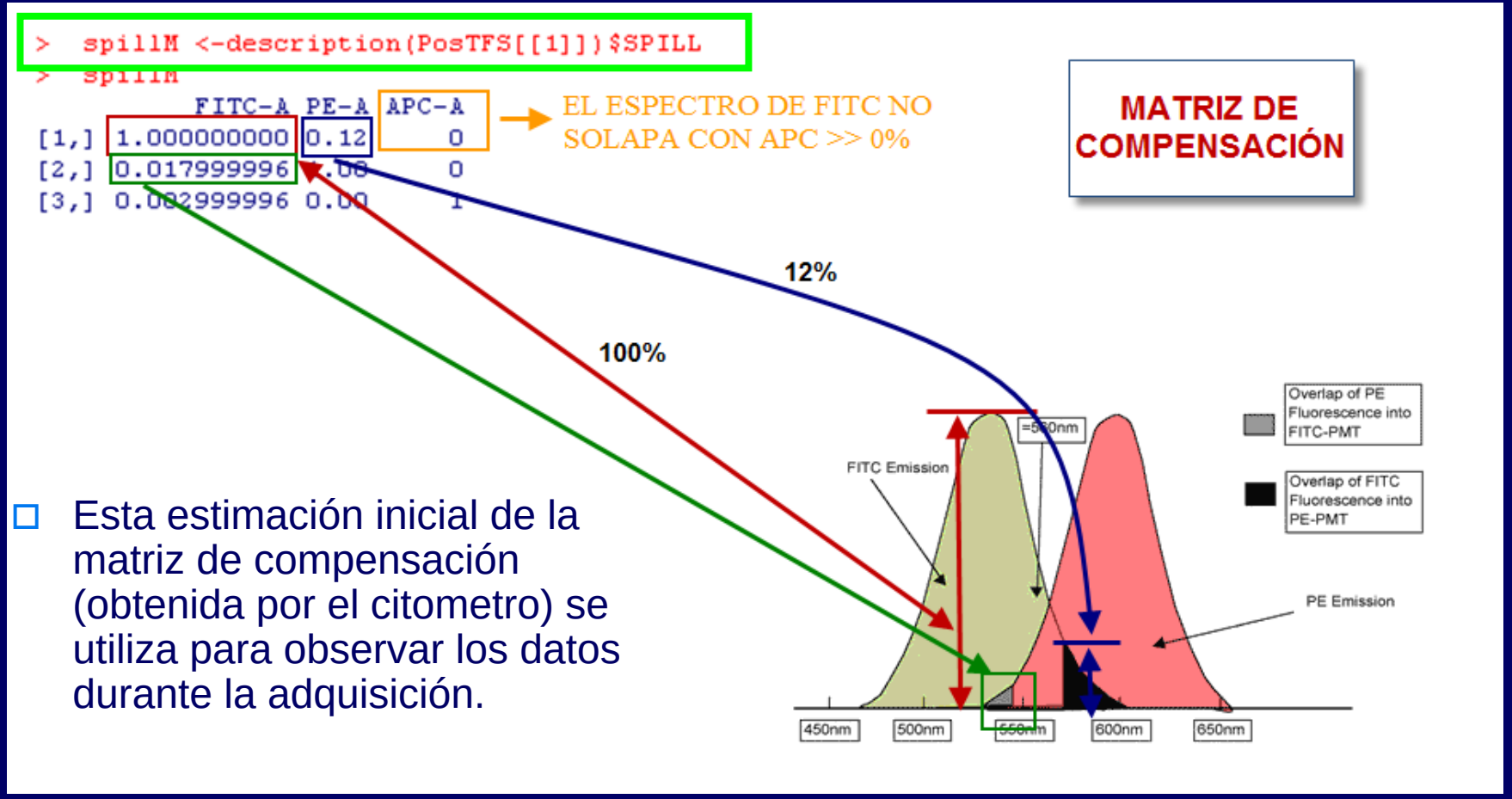


- Compensación y corrección de fondo
- Recuperación de la matriz de compensación desde los metadatos de texto de MACSPurity_Tube_001.fcs (Muestra control)

```
> spillM <-description(PosTFS[[1]])$SPILL
> spillM
          FITC-A PE-A APC-A
[1,] 1.000000000 0.12     0
[2,] 0.017999996 0.00     0
[3,] 0.002999996 0.00     1
```

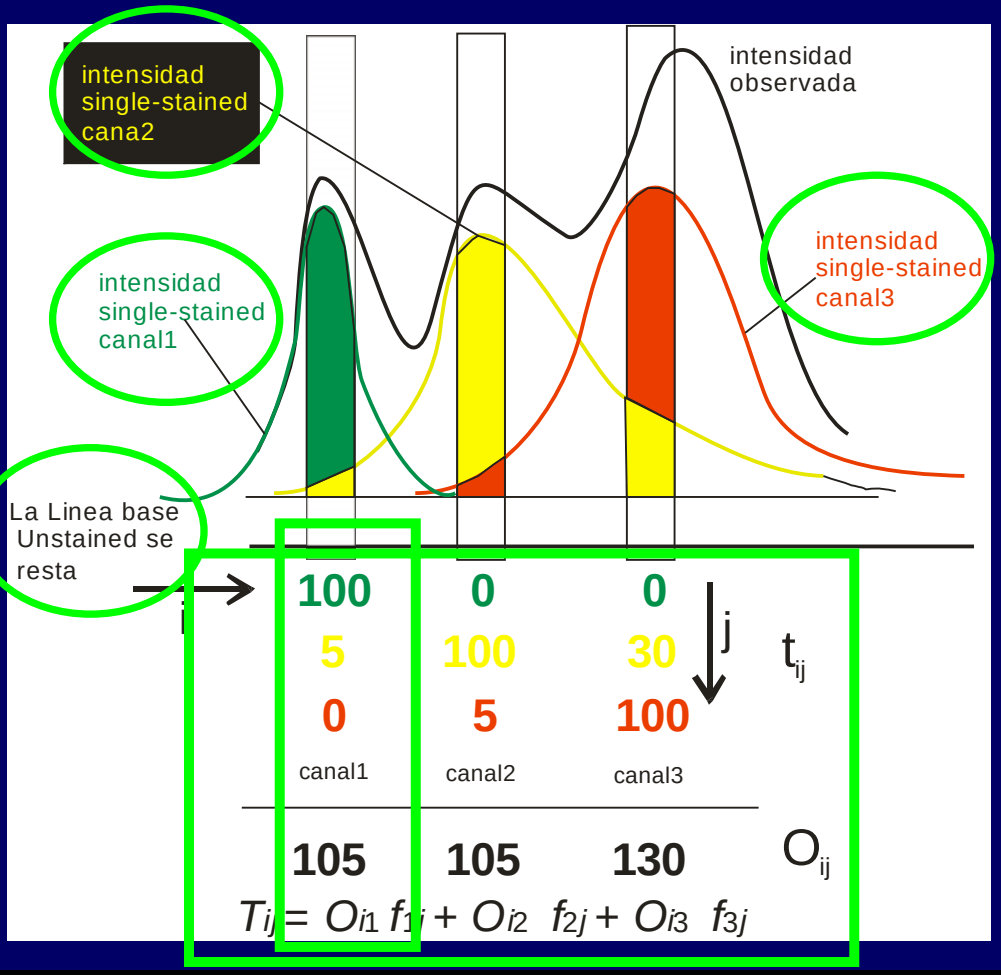
EL ESPECTRO DE FITC NO SOLAPA CON APC >> 0%

MATRIZ DE COMPENSACIÓN



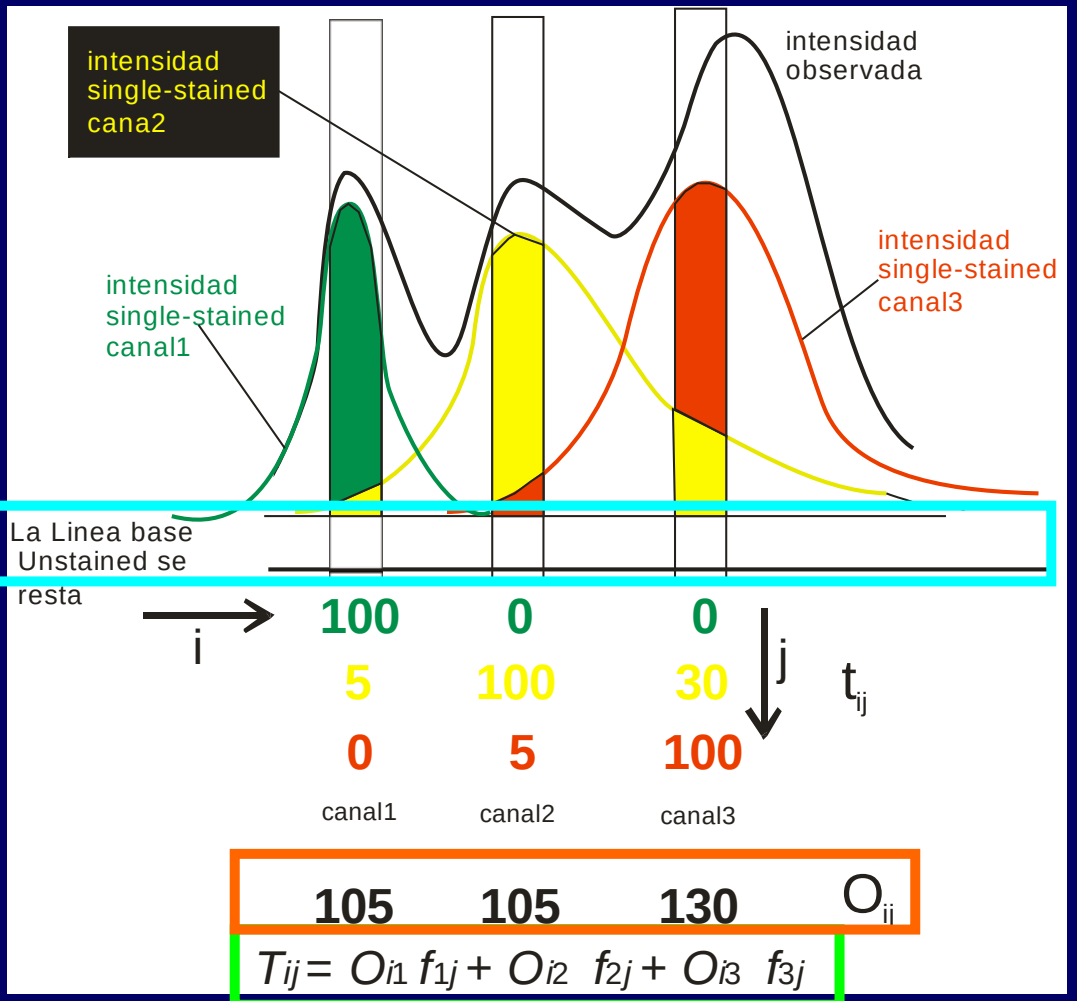
- Esta estimación inicial de la matriz de compensación (obtenida por el citometro) se utiliza para observar los datos durante la adquisición.

- Compensación y corrección de fondo
- Desde R se puede optimizar el cálculo de la matriz de compensación mediante estimación estadística.
- La matriz de compensación (fij) se puede calcular a partir de la muestra de control (unstained) y de las muestras que contienen un solo fluorocromo (single-stained).
- Esta matriz de compensación, Fij, se calcula de la siguiente manera.
  - El desdoblamiento del canal parámetro primario en los secundarios se supone que será una función lineal del parámetro primario.
  - Los valores compensados son combinación lineal de los observados (Oij) en los experimentos single-stain

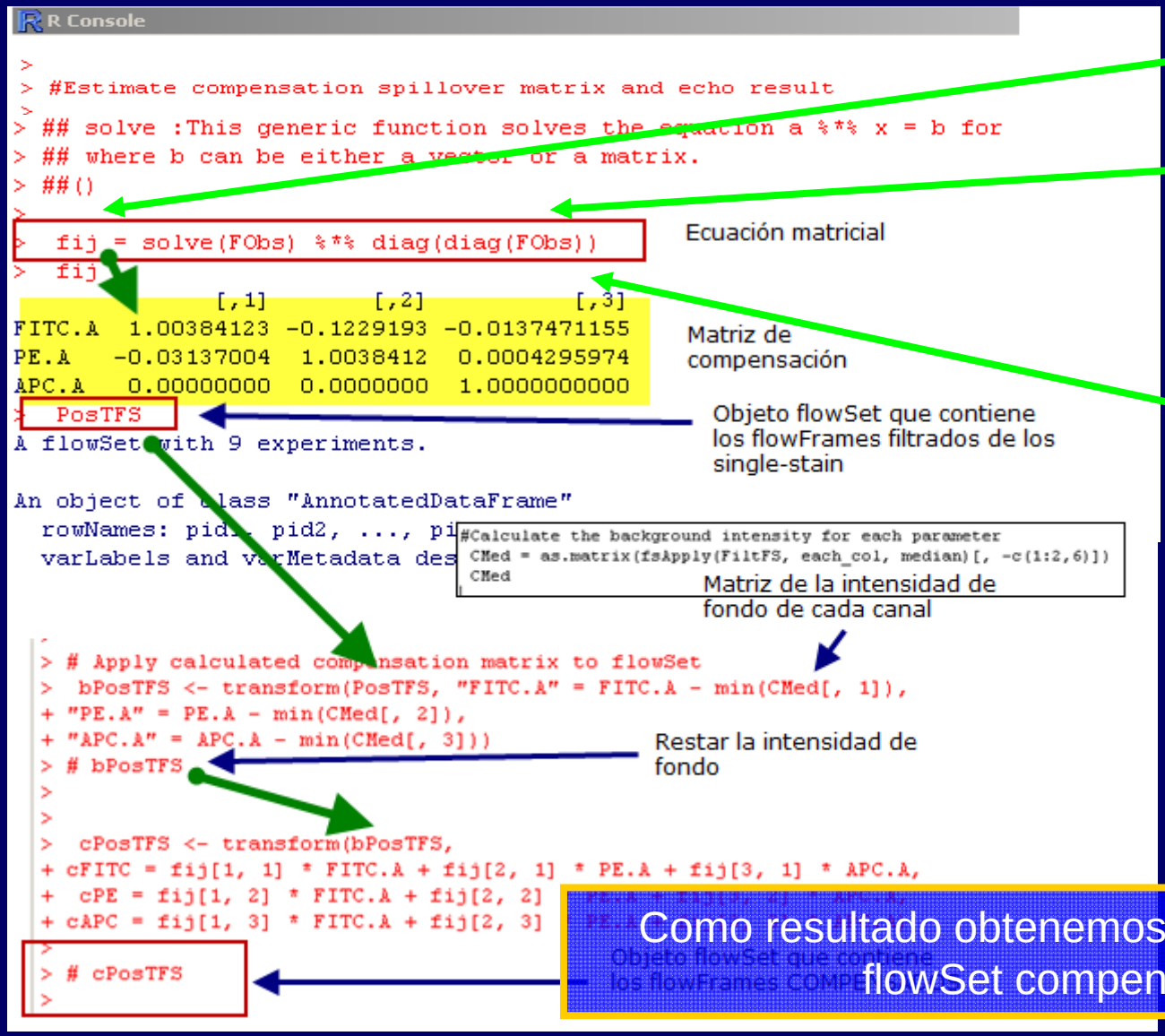


Compensación y corrección de fondo

- El problema del cálculo de la matriz de compensación se resume a resolver la ecuación (o sistema de ecuaciones) anterior.
- La matriz de las intensidades observadas (es decir, O) se estima de los valores de la mediana de cada experimento, single-stain.
- Antes de calcular los valores de la mediana, la fluorescencia de fondo se resta de los valores en bruto



Compensación y corrección de fondo. Aplicación en R



- Fij es la matriz de compensación
- FObs es la matriz de observaciones
- La ecuación matricial en R equivale a:

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

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

- Antes de calcular los valores de la mediana, la fluorescencia de fondo se resta de los valores en bruto
- La intensidad de fondo se obtiene de la muestra "pre-sort Unstained"

Como resultado obtenemos un nuevo objeto flowSet compensado

- Escalado y transformación: Consiste en aplicar una transformación a los ejes de forma que los datos cubran de forma homogenea el espacio de visualización.
- La transformación y escalado es esencial tanto para la visualización como para posterior tratamiento estadístico de los datos.
- Las transformaciones que se usan rutinariamente en el análisis de FCM definidas en el estándar “Transformation-ML”, están implementadas en flowCore.

  - La transformación logarítmica es el método comúnmente utilizado para hacer frente a la amplia gama dinámica de las medidas de fluorescencia.
  - Problemas¡¡¡,
    - La compensación y la sustracción del fondo de fluorescencia crean valores negativos.
    - La representación gráfica de los datos en los ejes logarítmicos truncara los valores negativos.
  - Una alternativa es utilizar una transformación que es lineal en torno a cero y no lineal en otras regiones

Data transformations implemented in flowCore.

	Data Transformations
linear	$ax + b$
quadratic	$ax^2 + bx + c$
logarithm	$\log_b(x)(r/d)$
logicle	$Te^{-(m-w)}(e^{(x-w)}-p^2e^{-(x-w)/p} + p^2-1)$
truncate	$x_{x\leq a} = a$
scale	$(x-a)/(b-a)$
arcsinh	$arcsinh(a + bx) + c$

Within these formulas, x is the variable corresponding to value being transformed, a, b, c, d, f, p, m, T, and w, are constants affecting the transformation function, e is the base of the natural logarithm (see [13] for details on the logicle transformation). Other transformations can easily be implemented in R.

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

- Escalado y transformación
- El diseño de R hace que sea fácil definir nuevas funciones arbitrarias aplicables tanto los flowFrame como a los flowSet para incorporarlas en un protocolo habitual de flowCore.
- En el caso que nos ocupa se define una función personalizada lin-log que encapsula la nueva transformación.
- Posteriormente se aplica al flowSet compensado para obtener un nuevo flowSet con la escala transformada.

```
#####################################################
### chunk number 10: Linear-Log Data Transformation
#####################################################
```

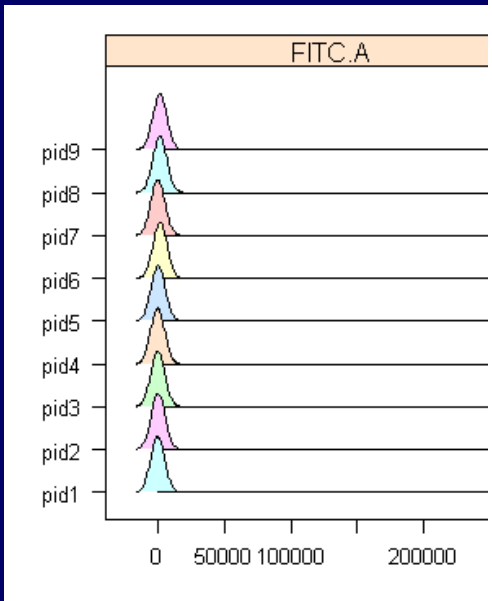
definición de la función de transformación

```
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
    }
```

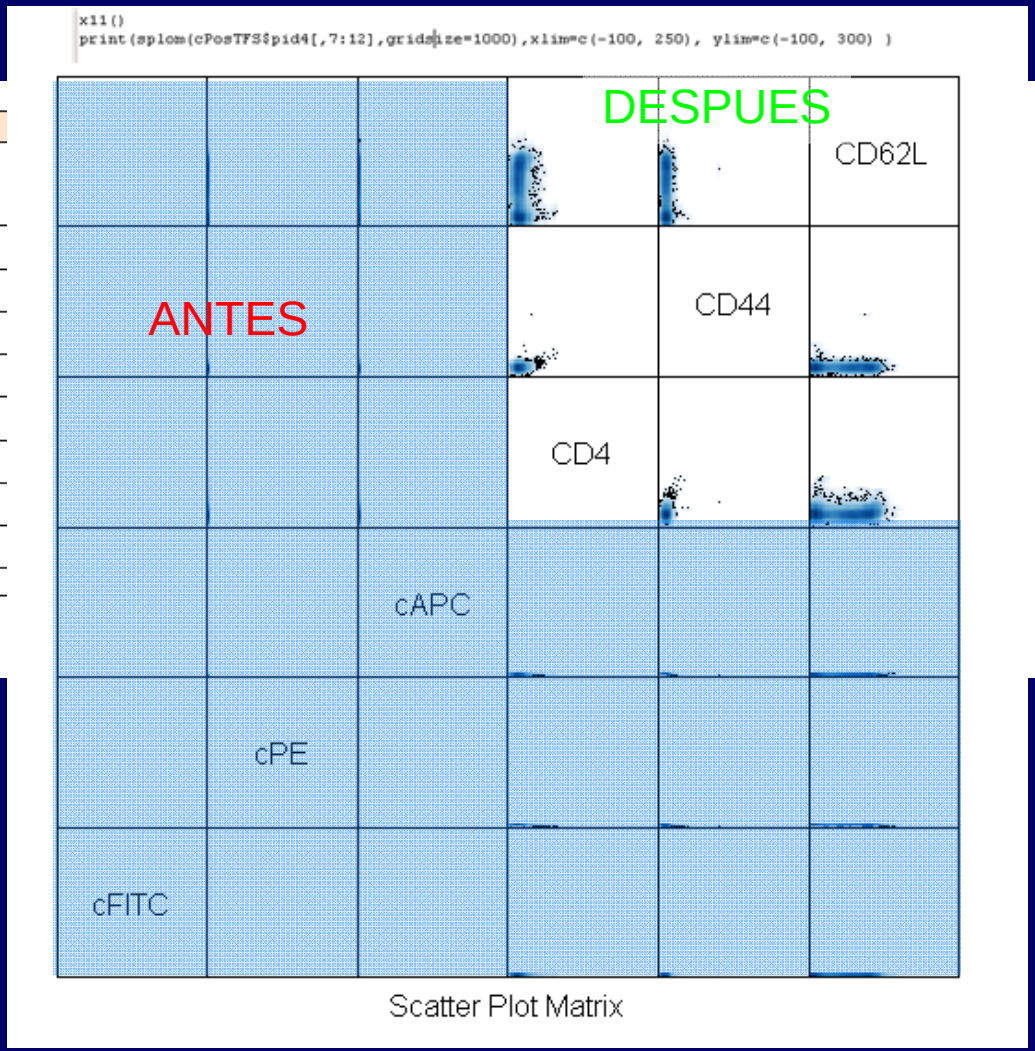
```
lnlgT <- linlogTransform(transformationId = "splitscale", median = 0, dist = 100)

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

Aplicación a cada canal ..... No se aplica a FSC y SSC

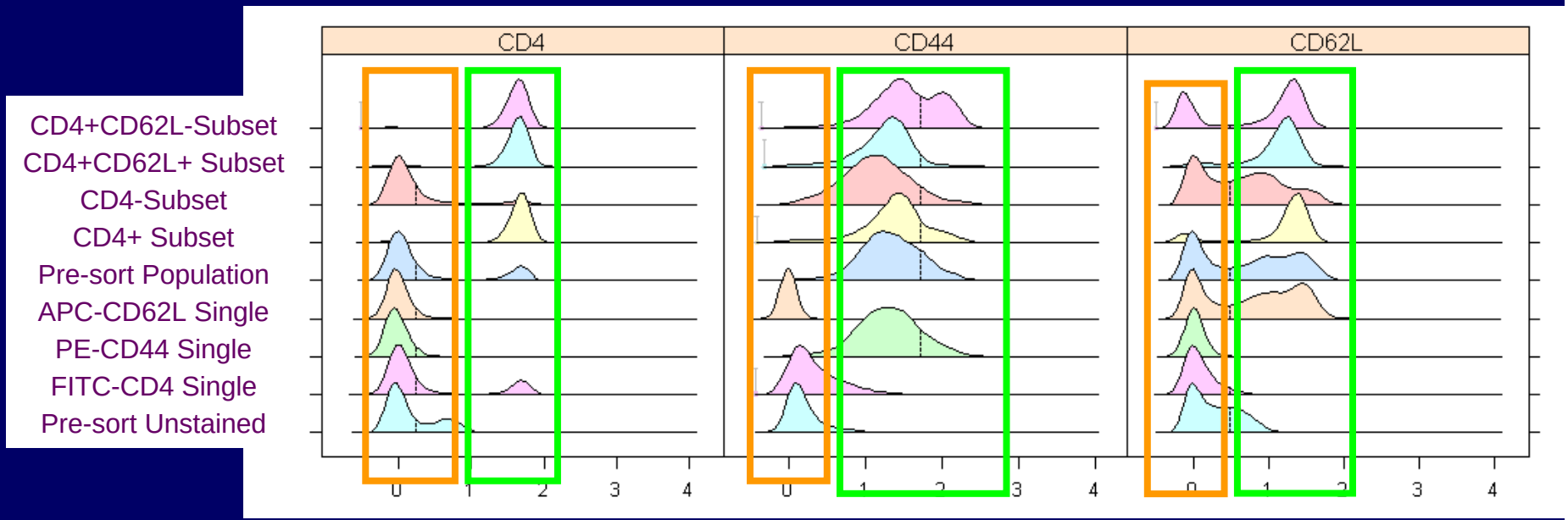
- Visualización final de los datos.
  - Antes del procesado (filtrado, compensación y transformación) es muy complicado discernir las poblaciones.




```
x11()
print(splom(cPosTFS$pid4[,7:12],gridsize=1000),xlim=c(-100, 250), ylim=c(-100, 300) )
```



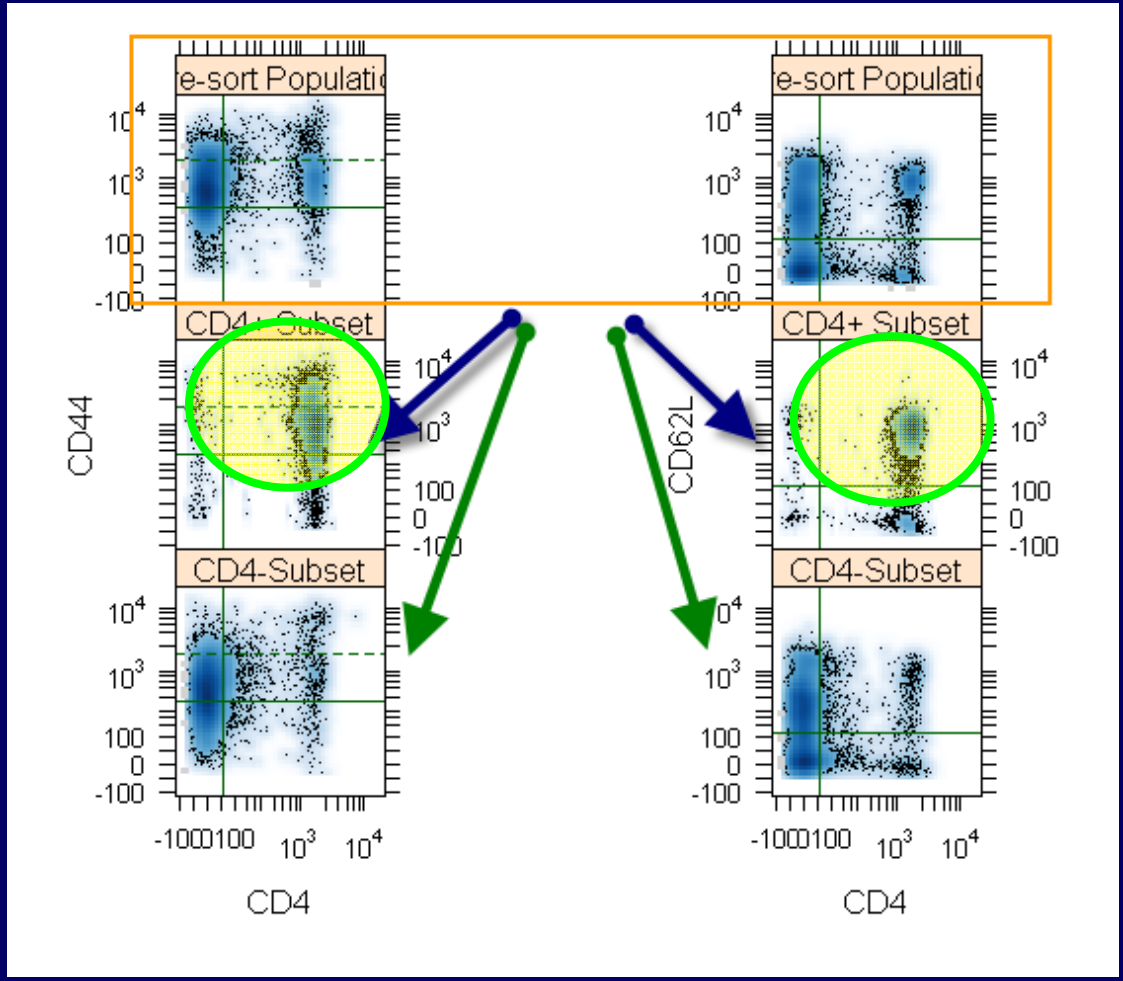
- Visualización final de los datos.
  - Después del procesado podemos identificar con claridad
    - los positivos
    - de los negativos



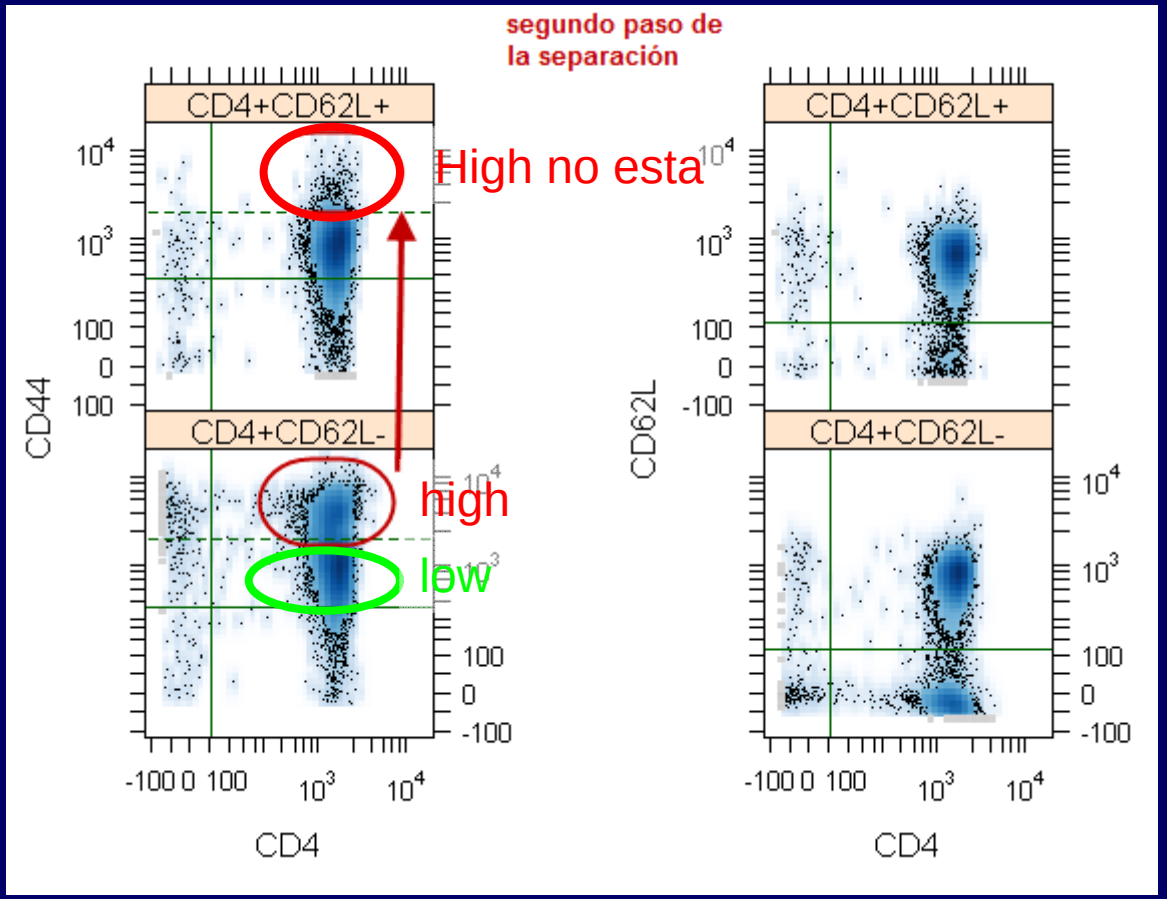
Análisis de los datos.

- Para demostrar la efectiva separación de los linfocitos CD4+CD62L+ hay que comparar las poblaciones relativas de estos en cada una de las cinco alicotas que se han separado en la extracción.

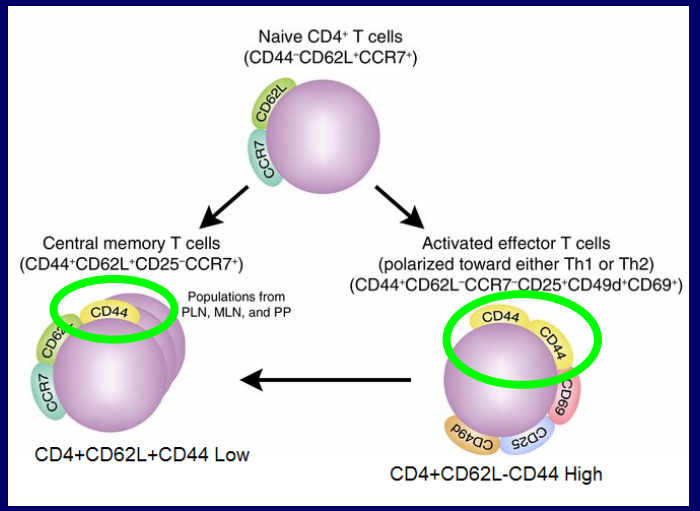
- Análisis de los datos.
- Para demostrar la efectiva separación de los linfocitos CD4+CD62L+ hay que comparar las poblaciones relativas de estos en cada una de las cinco alicotas que se han separado en la extracción.
- Primer paso de la separación
- La alícuota CD4+ se enriquece en CD4 y CD44

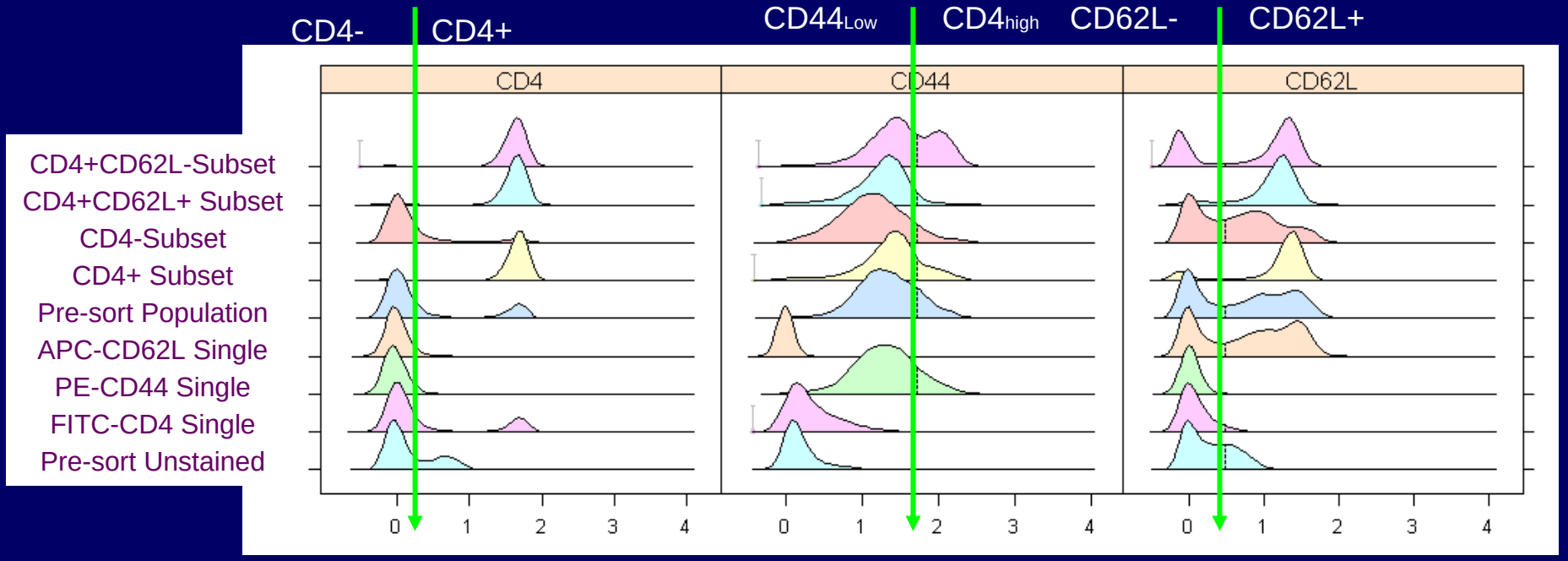


- Análisis de los datos.
  - Segundo paso de la separación
  - Hay dos poblaciones de CD44+, una denominada High y otra Low (verde)
    - La población high no está en la alicuota CD4+CD62L+



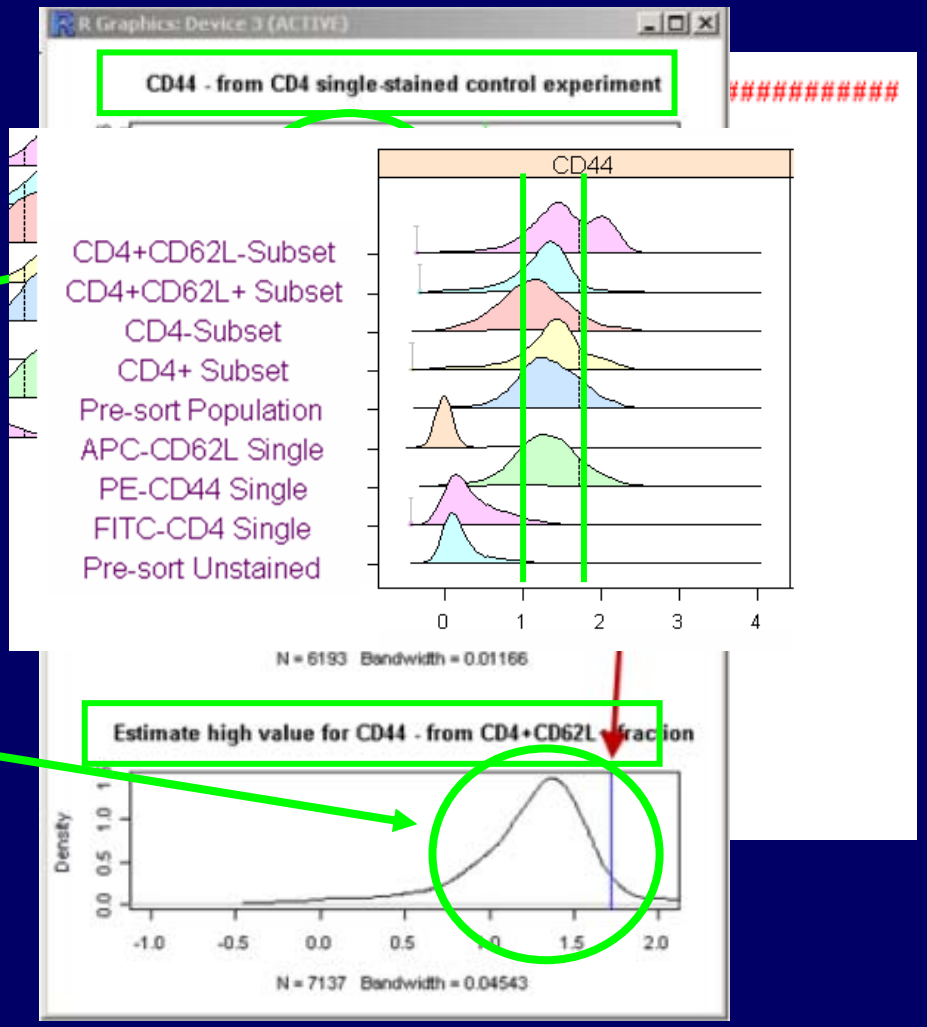
- Análisis de los datos.
- Con los gráficos PDF es más sencillo visualizar la evolución de las poblaciones
  - Los CD4+ es fácil identificarlos y discriminarlos con un límite de detección
  - La población de CD44+ vemos que tiene dos poblaciones una alta (high) y otra baja (low) compatible con las poblaciones previstas.
  - La población de CD62L es compleja pero es posible identificarla mediante un límite de detección.



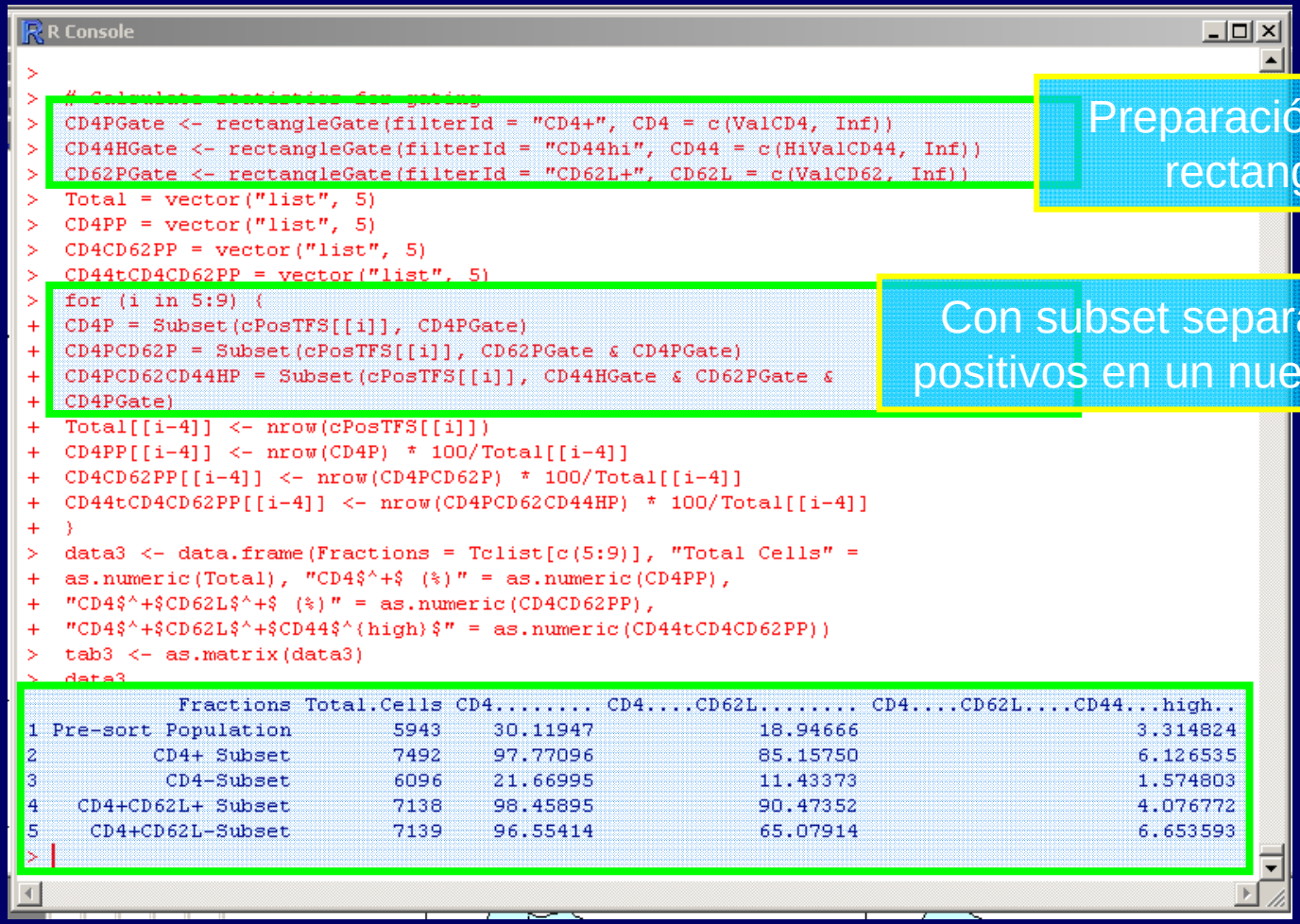


- Análisis de los datos.
- Para calcular las estadísticas de cada muestra, en el ejemplo se basan en un límite de detección estadístico **y no en una selección manual.**

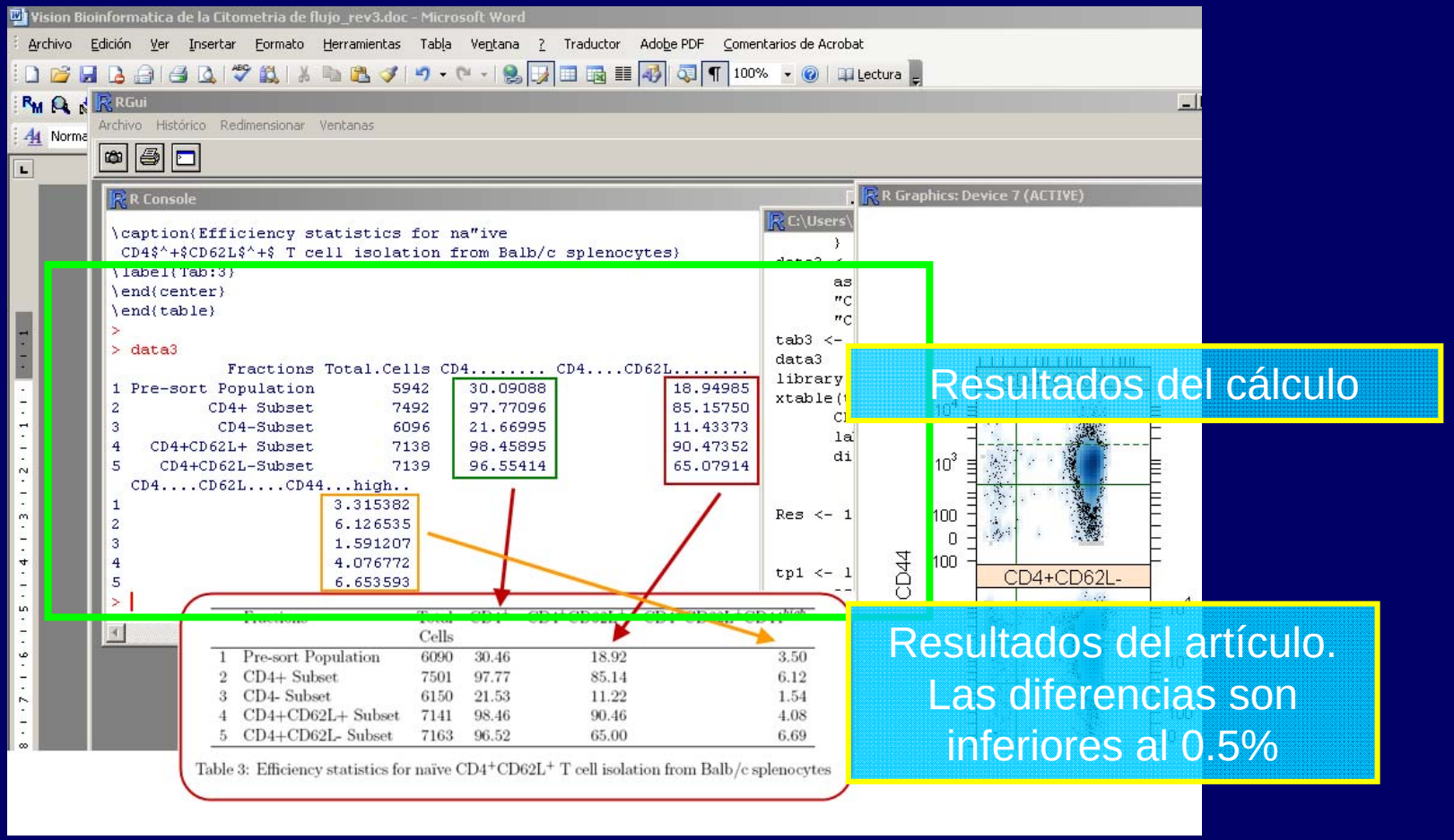
- El límite lo definimos como el nivel de expresión para el cual el 95% de la población de las células no marcadas exhiben un nivel de expresión bajo
- El límite para el marcador CD44 lo calculamos a partir de las distribuciones de población de su canal en las alícuotas single-stain de CD4 y CD62L, eligiendo la mayor de las dos.
- El límite high/low se calcula de la misma forma pero en la alícuota CD4+CD62L+
- El resto de límites se calculan de forma similar teniendo en cuenta lo observado en los gráficos PDF



- Análisis de los datos.
- Finalmente, las estadísticas se calculan aplicando los filtros al flowSet compensado y transformado.



- Análisis de los datos.
- **En resumen, es posible reproducir cuantitativamente los resultados de Citometría de flujo de un analista a otro.**
- **Los márgenes de error son únicamente debidos a la configuración numérica del procesador**



- Análisis avanzado de los datos.
- **Adicionalmente, es posible discernir las diferentes poblaciones mediante filtros de densidad como el curv2filter**

C:\Users\ramon\Documents\Trabajos\Cytomica\RTutorial\curv2filter_ej.R - Editor R

```
cPosTFS
fs <- cPosTFS

##curv2Filter es un filtro de densidad
cf <- curv2Filter("CD44", "CD62L",bwFac=2.0)
fres <- filter(fs, cf)
cf
objects(fres)
summary(fres)
     gate=list(fill="blak", alpha=0.6)
     gate
flowViz.par.set(gate=list(fill="red", alpha=0.9))
trellis.par.set(theme = col.whitebg())
lw <- list(ylab.axis.padding = list(x = 0.4), left.padding = list(
      units = "inches"), right.padding = list(x = 0, units = "inch
      panel = list(x = 1.5, units = "inches"))
lh <- list(bottom.padding = list(x = 0, units = "inches"), top.pad
      list(x = 0, units = "inches"), panel = list(x = 1.5, units =

lattice.options(layout.widths = lw, layout.heights = lh)
x11()
xyplot(`CD44` ~ `CD62L`, data=fs, filter=fres )
```

Filtro de densidad sobre los positivos

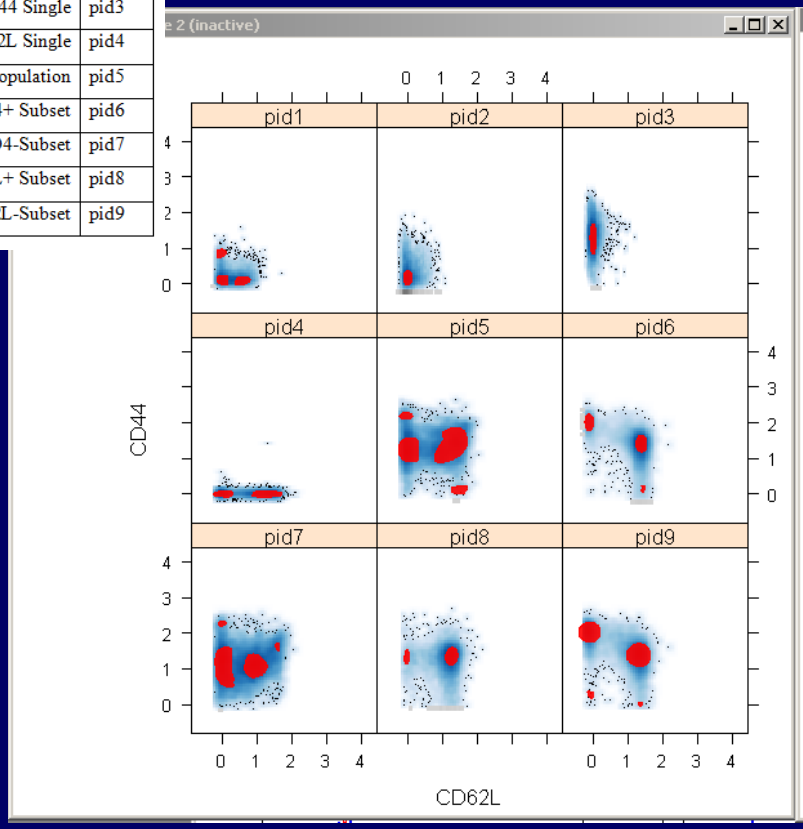
Visualización

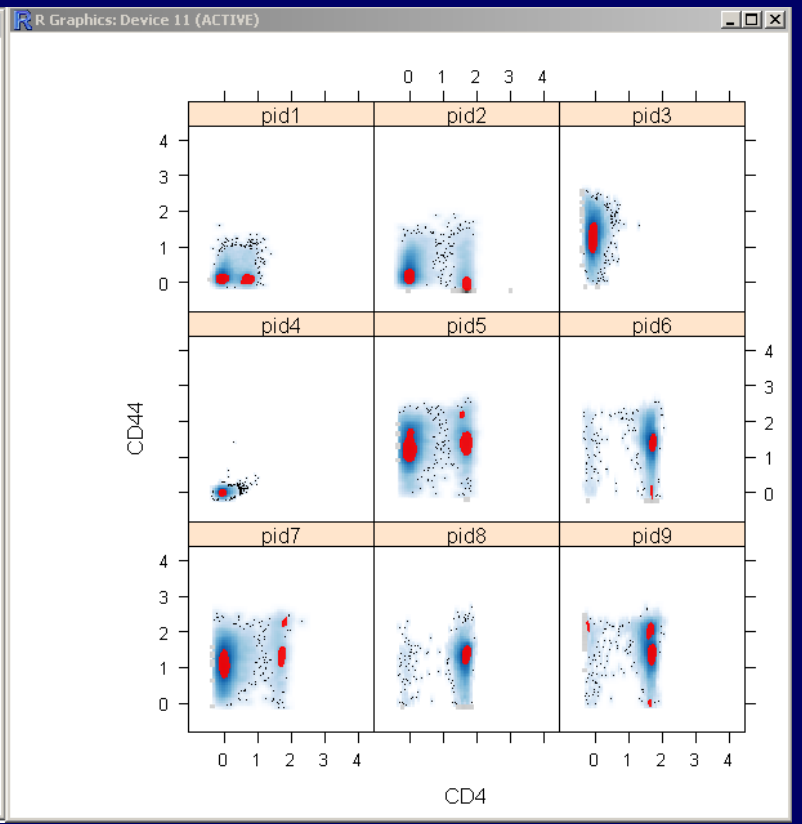
R Console

```
filter summary for frame 'pid5'
 rest: 3030 of 5943 events (50.98%)
 area 1: 2376 of 5943 events (39.98%)
 area 2: 521 of 5943 events (8.77%)
 area 3: 16 of 5943 events (0.27%)

filter summary for frame 'pid6'
 rest: 5298 of 7492 events (70.72%)
 area 1: 19 of 7492 events (0.25%)
 area 2: 2175 of 7492 events (29.03%)
 area 3: 0 of 7492 events (0.00%)

filter summary for frame 'pid7'
 rest: 3840 of 6096 events (62.99%)
 area 1: 2135 of 6096 events (35.02%)
 area 2: 101 of 6096 events (1.66%)
 area 3: 20 of 6096 events (0.33%)

filter summary for frame 'pid8'
 rest: 4553 of 7138 events (63.79%)
 area 1: 2585 of 7138 events (36.21%)

filter summary for frame 'pid9'
 rest: 4403 of 7139 events (61.68%)
 area 1: 14 of 7139 events (0.20%)
 area 2: 1919 of 7139 events (26.88%)
 area 3: 793 of 7139 events (11.11%)
 area 4: 10 of 7139 events (0.14%)
```

- Análisis avanzado de los datos.
- **Adicionalmente, es posible discernir las diferentes poblaciones mediante filtros de densidad como el curv2filter**

Pre-sort Unstained	pid1
FITC-CD4 Single	pid2
PE-CD44 Single	pid3
APC-CD62L Single	pid4
Pre-sort Population	pid5
CD4+ Subset	pid6
CD4-Subset	pid7
CD4+CD62L+ Subset	pid8
CD4+CD62L-Subset	pid9



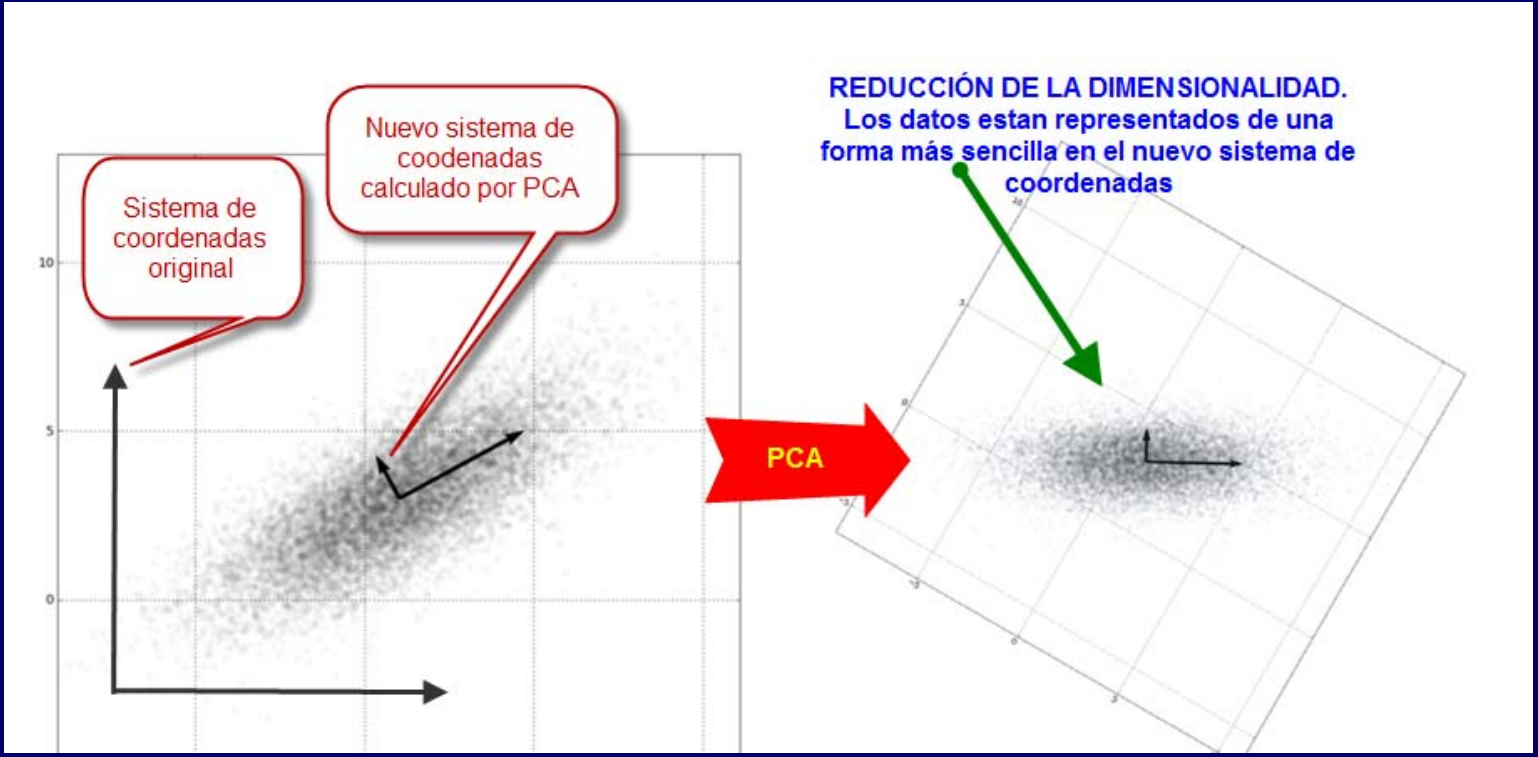


- Análisis avanzado de los datos. Análisis de componentes principales

- En estadística, el análisis de componentes principales (en español ACP, en inglés, PCA) es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos.
- Intuitivamente la técnica sirve para determinar el número de factores subyacentes explicativos tras un conjunto de datos que expliquen la variabilidad de dichos datos.
  - La PCA busca la proyección según la cual los datos queden mejor representados en términos de mínimos cuadrados (minimizar la matriz de covarianza).
  - Es equivalente a transformar los ejes de coordenadas para buscar la representación más sencilla

- Análisis avanzado de los datos. Análisis de componentes principales

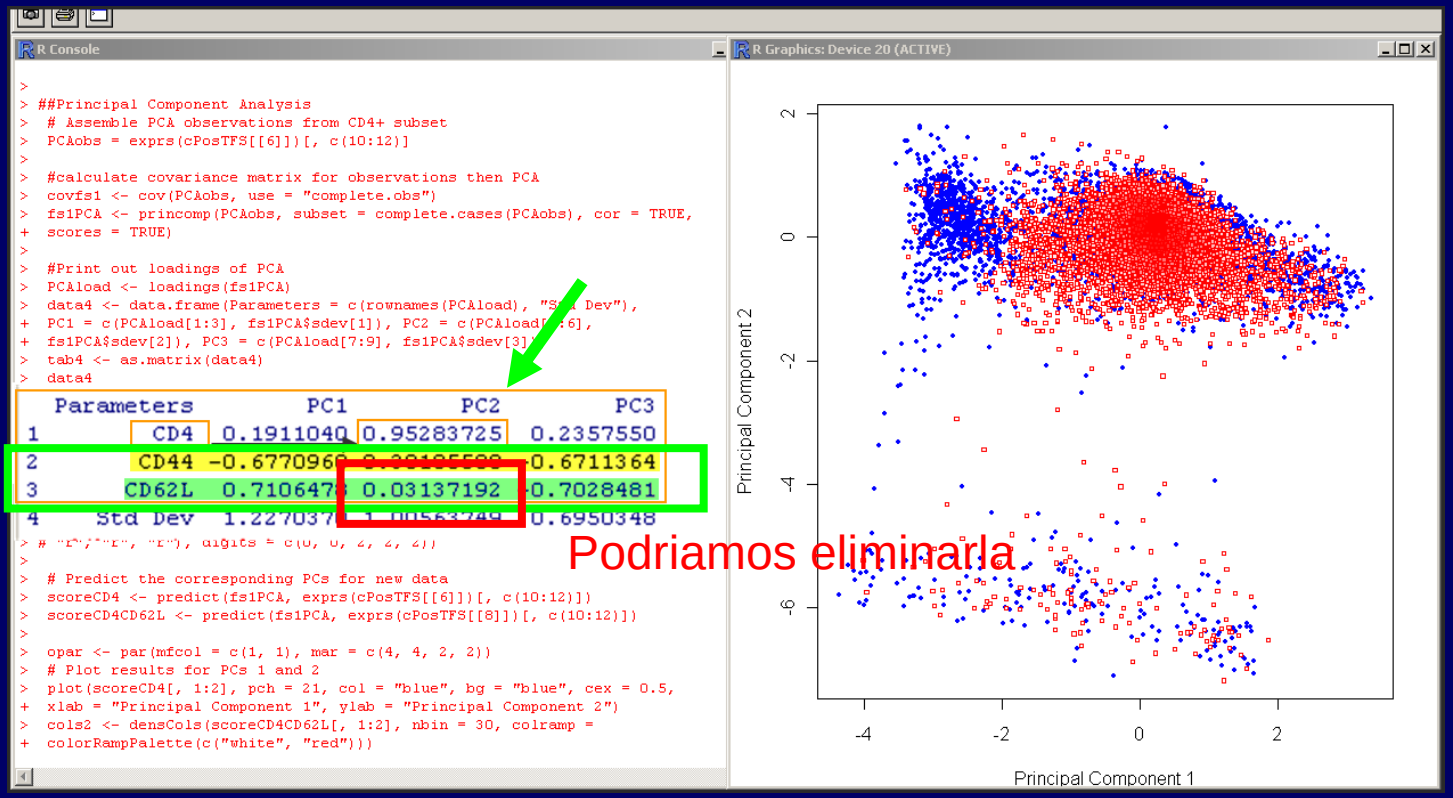
  - Mediante el análisis de PCA obtenemos una nueva representación de los datos en base al nuevo sistema de coordenadas.
  - La PCA se emplea sobre todo en análisis exploratorio de datos y para construir modelos predictivos.



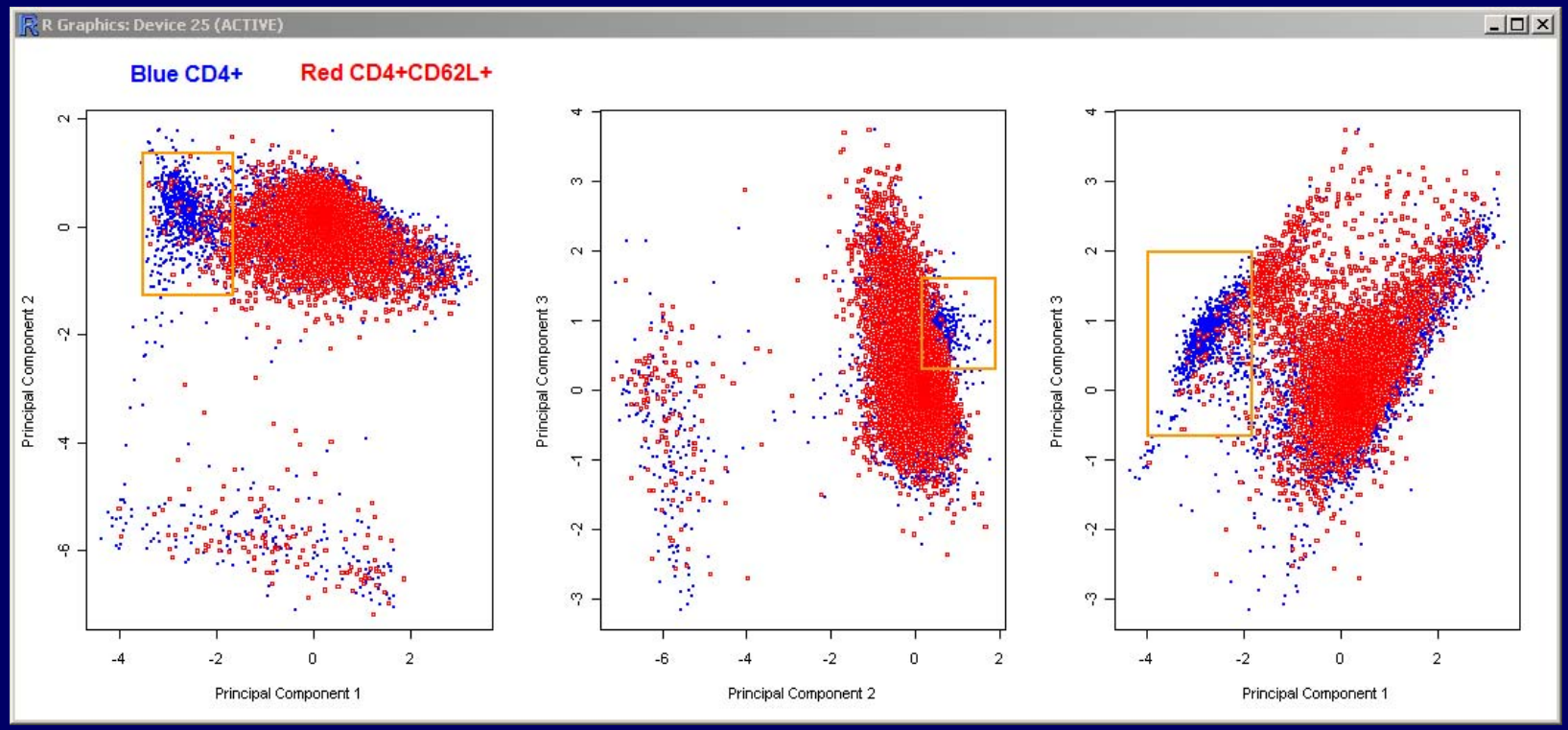
ANÁLISIS DE DATOS A PEQUEÑA ESCALA CON BIOCONDUCTOR

- Análisis avanzado de los datos. Análisis de componentes principales
- Las poblaciones de CD4+ y CD4+CD62+ se pueden detectar mediante análisis de componentes principales “PCA “.
- Para hacer un análisis de componentes principales hay que elegir el numero de componentes que queremos obtener (normalmente tres) y facilitar las variables observables de nuestro sistema según, que en nuestro caso son los tres canales CD4, CD44 y CD62L.
  - Dado que el número de variables originales coincide con el número de componentes principales, únicamente obtenemos un giro en el sistema de coordenadas.
  - Si nuestro sistema hubiera tenido más de tres canales, hubiéramos obtenido un nuevo sistema de coordenadas con una complejidad igual al número de componentes principales elegido.
  - Igualmente, el método nos permite visualizar aquellas componentes que añaden “poca variabilidad” a los datos y eliminarlas.

- Análisis avanzado de los datos. Análisis de componentes principales
- En nuestro ejemplo la PCA nos permite visualizar aquellas componentes que añaden “poca variabilidad” a los datos y eliminarlas.
- De los valores de los parámetros se extraen las siguientes conclusiones:
  - la expresión de CD4 es directamente proporcional al parámetro P2,
  - CD44 y CD62L se diferencian en su respuesta inversa (uno es positivo y el otro negativo) respecto al parámetro PC1.



- Análisis avanzado de los datos. Análisis de componentes principales
- En los gráficos de componentes principales es inmediato observar las diferencias que en los dot-plots no eran tan obvias (recuadro en naranja)



- **Conclusiones.**
- El tratamiento clásico de los datos de Citometría de flujo mediante los paquetes de R-bioconductor está resuelto mediante los paquetes publicados actualmente
- Aunque en el artículo de Florian et al. se indica que la interface gráfica iflow es accesible desde el repositorio de Bioconductor, actualmente no aparece como paquete publicado
- Mediante los filtros *norm2Filter*, *curv2filter, kmeansfilter* es posible extraer poblaciones sin la intervención del usuario, de esta forma se evitan los errores experimentales debidos a la selección manual
- El cálculo de la matriz de compensación es posible evaluarlo estadísticamente a partir de las muestras control y “single-stain”
- El modelo de datos permite aplicar, además de las transformaciones clásicas, otras posibles trasformaciones combinadas según los requerimientos de los datos
- Las funciones de escalado y normalización eliminan la variación técnica entre experimentos, de esta forma es posible tratar conjuntamente grandes grupos
- A diferencia del software comercial, mediante R-bioconductor, es posible aplicar técnicas estadísticas particularizas al problema sin necesidad transformaciones adicionales de formato

Gracias por vuestra atención