Gates/filters in Flow Cytometry Data Visualization

June 3, 2008

Abstract

The flowViz package provides tools for visualization of flow cytometry data. This document describes the support for visualizing gates (a.k.a. filters).

1 Introduction

Tools to read in and otherwise manipulate flow cytometry data are implemented in the flowCore package. The flowViz package provides visualization support for such data. In this document we give examples dealing with gated or filtered data.

1.1 Filters and filter results

The flowCore package defines the concepts of filters and filter results. Filters are abstract entities defined in terms of marker (?) channels, whereas a filter result is the result of applying a filter to one or more flow frames (data objects representing individual FCM experiments). Some filters are data driven, while some some are not.

1.2 Visualization

It only makes sense to visualize a filter if it has a (two dimensional) geometric representation. This is true for (2D) rectangle gates and polygon gates, which are both frequently used. It is also true for some data driven gates, e.g. *norm2Filter* gates, which have a (data dependent) spherical representation. Also, it makes sense to draw gate boundaries only what plotting the channels that define the gate.

Visualizing filter results is a more general. Specifically, the result of applying a filter is usually a logical (`TRUE` or `FALSE`) vector for each cell, or more generally, a *factor* (as long as we restrict ourselves to non-fuzzy filters). This can be used a grouping variable within a display, and can be used with channels other than those defining the gate.

1.3 Examples

We use the GvHD data set to provide some examples (what's the equivalent of `subset(GvHD, Patient == "6"`?).

```
> data(GvHD)
```

We define 2 filters and apply one of them to the full data set.

```
> n2gate <- norm2Filter("SSC-H", "FSC-H")
> rgate <- rectangleGate("FSC-H"=c(5.5, 7),"SSC-H"=c(4, 6.5))
> n2gate.results <- filter(GvHD, n2gate)
```

2 Filters in scatter plots

The `xyplot` method for *flowSet* objects support filters through three arguments:

- `filter`: A *filter* object

- `filterResults`: A *filterResult* object that has been obtained by applying a filter (not necessarily `filter`) on the *flowSet* object that is the `data` argument in the call. This is essentially the result of applying the filter on each frame in `data`. If this is `NULL` but `filter` is not, it will be created on the fly by applying `filter` to each frame.

- `displayFilter`: logical or list. If `FALSE`, no attempt is made to render `filter`. If `TRUE`, an attempt is made to display the geometric boundaries of `filter` (if specified) in the plot. The same holds if `displayFilter` is a list, but in that case the components are interpreted as graphical parameters to be passed on to the polygon drawing routines.

The key concepts here are that

- `filter` and `filterResults` can be unrelated

- `filter` is visualized geometrically, which makes sense only under certain circumstances (e.g. display axes matching filter parameters), while `filterResults` is visualized through grouping, which makes sense more generally (display variables need not match filter parameters).

- `displayFilter` is concerned with `filter`, not `filterResults`

- Visualization of `filterResults` depends on the type of rendering. For example, with `smooth = TRUE`, `filterResults` will be ignored because the panel function used to perform the rendering cannot handle a grouping variable. Grouping is done when `smooth = FALSE`, but this has the drawback of overplotting. The effect of overplotting can be reduced somewhat using transparency, but scalability issues remain. We may try dealing with this at some point if it becomes enough of a hassle.

We will next give several examples using the filters defined above.

```
> xyplot(`SSC-H` ~ `FSC-H` | Visit, data = GvHD,
+        filter = n2gate, displayFilter = TRUE,
+        subset = Patient == "6")
```
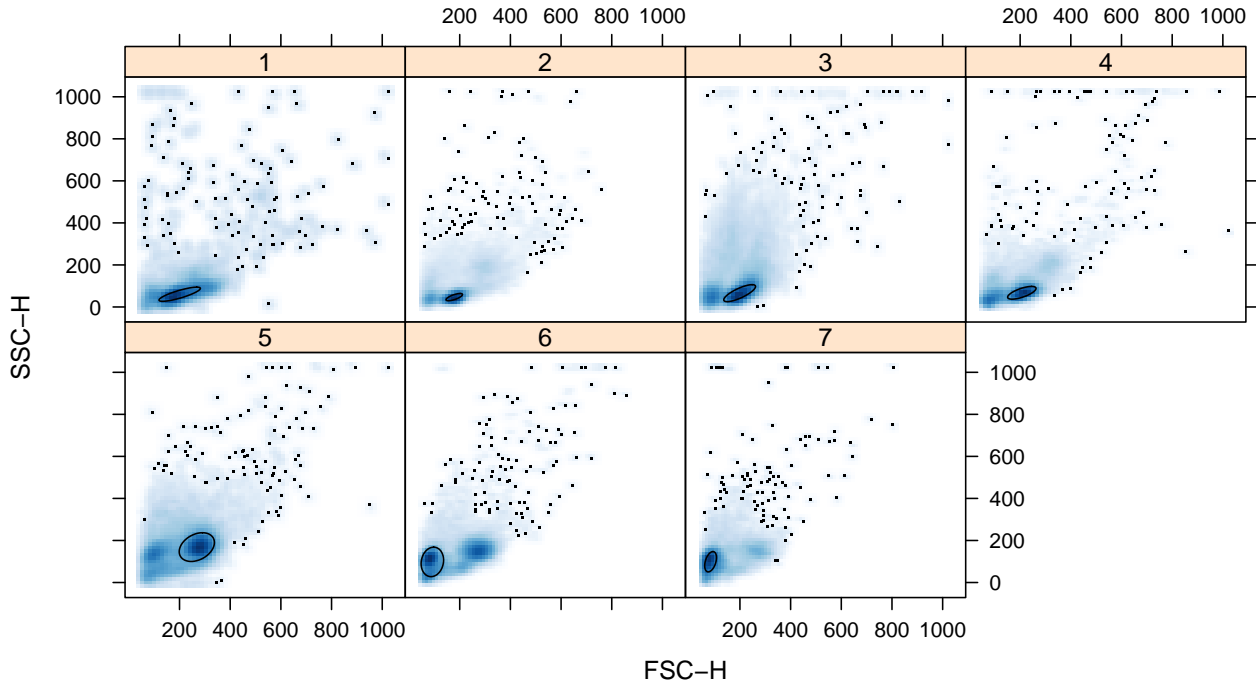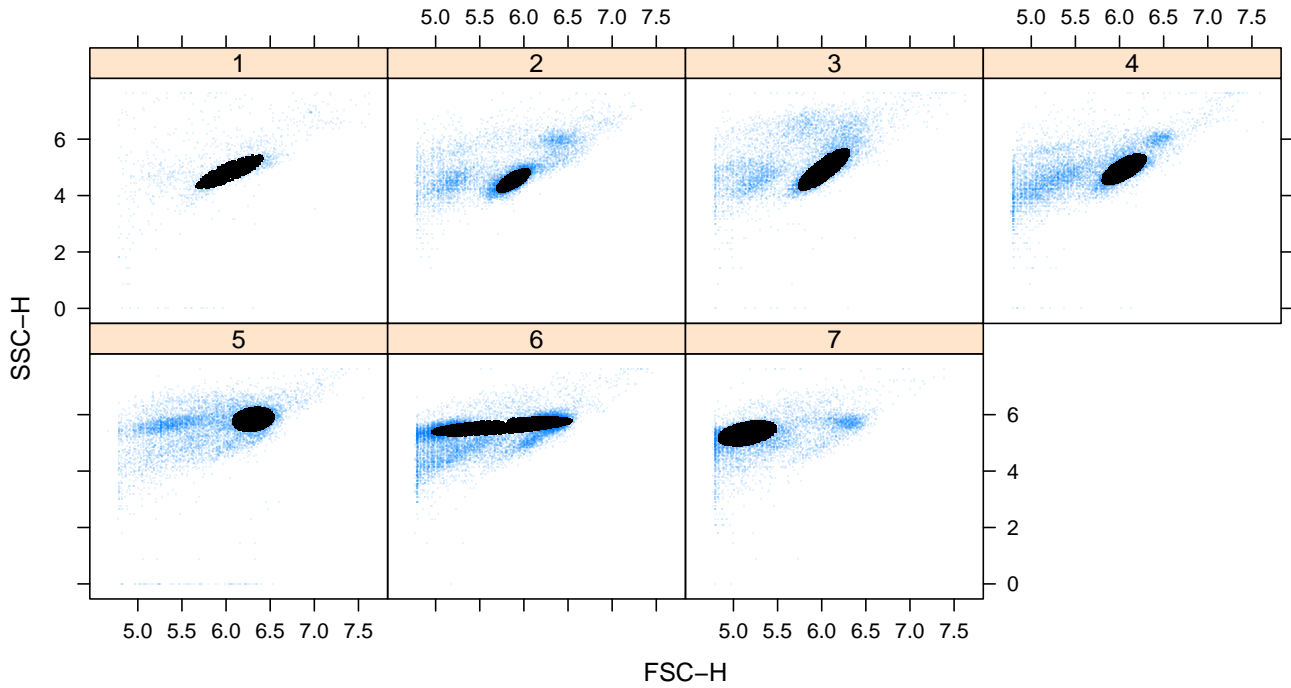

In this figure, `n2gate` is applied to each frame and the results rendered as an ellipse. Since the gate is data driven, the ellipse is different for each panel. Since `smooth=TRUE`, the subset of points in the gate are not indicated separately.

```
> xyplot(`SSC-H` ~ `FSC-H` | Visit,
+        data = transform("SSC-H"=asinh,"FSC-H"=asinh) %on% GvHD,
+        smooth = FALSE, alpha = 0.1,
+        filter = n2gate,
+        displayFilter = list(lwd = 2),
+        subset = Patient == "6")
```


In this figure, n2gate is again applied to each frame, but this time to the inline-transformed *flowSet* object suppiled as the data argument. Since smooth=FALSE, the subset of points in the gate are shown in a different color (filterResults is computed by applying filter to each frame). Making the points translucent allows us to get a feel of the local density of points.

```
> xyplot(`SSC-H` ~ `FSC-H` | Patient:Visit,
+         data = transform("SSC-H"=asinh,"FSC-H"=asinh) %on% GvHD,
+         subset = Patient == "6",
+         filter = rgate)
```
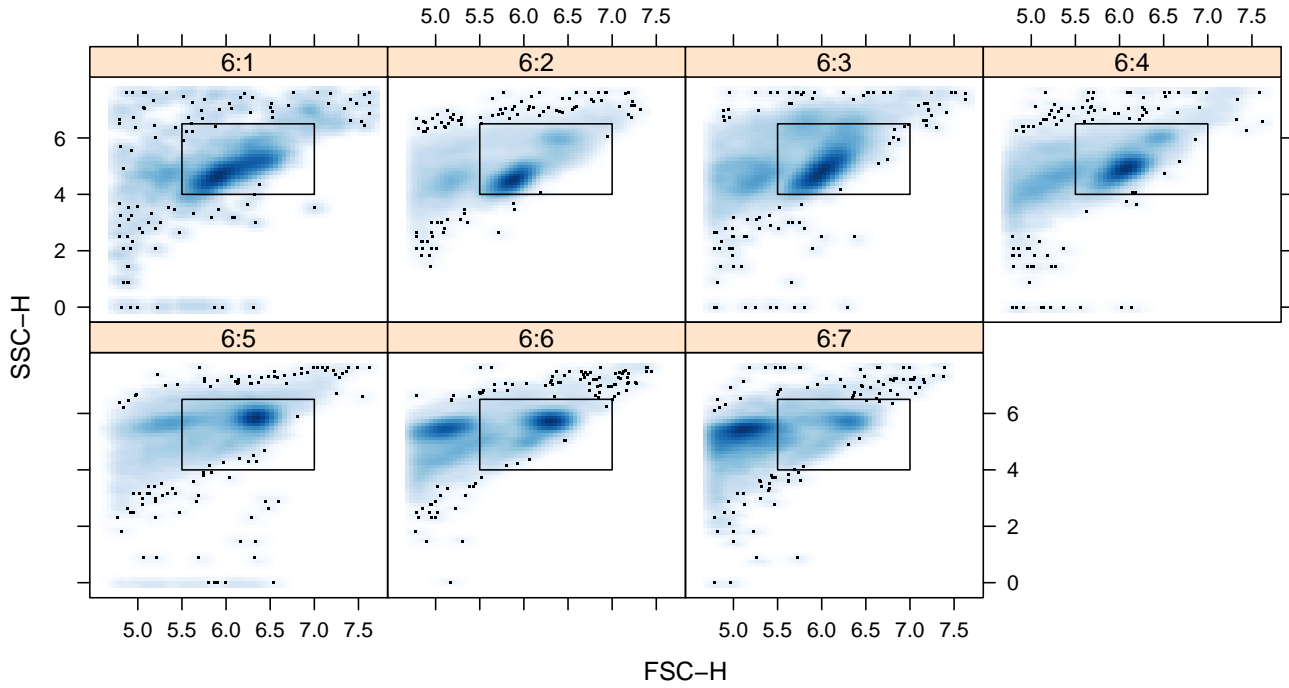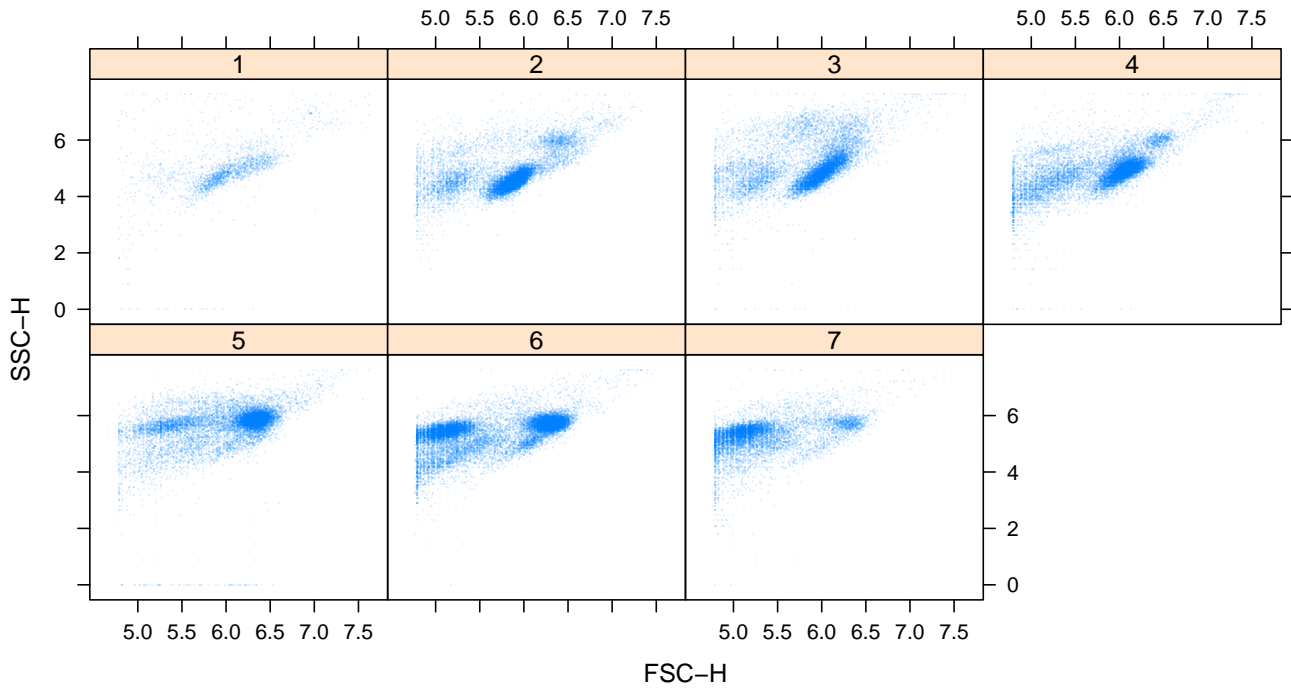

In this figure, the rectagle gate **r2gate** is rendered on the transformed data.

```
> xyplot(`SSC-H` ~ `FSC-H` | Visit,
+          data = transform("SSC-H"=asinh,"FSC-H"=asinh) %on% GvHD,
+          subset = Patient == "6",
+          filterResults = n2gate.results,
+          smooth = FALSE, alpha = 0.1)
```


In this figure, there is no `filter` argument. However, there is a `filterResults` argument, which has been set to `n2gate.results`. Note that the filter was originally applied to untransformed data, but the data being displayed is transformed. Consequently, the boundaries of the points in the gate are not ellipses in the display scale.

```
> xyplot(`SSC-H` ~ `FSC-H` | Visit,
+         data = transform("SSC-H"=asinh,"FSC-H"=asinh) %on% GvHD,
+         subset = Patient == "6",
+         filterResults = n2gate.results,
+         filter = n2gate,
+         smooth = FALSE, alpha = 0.1)
```
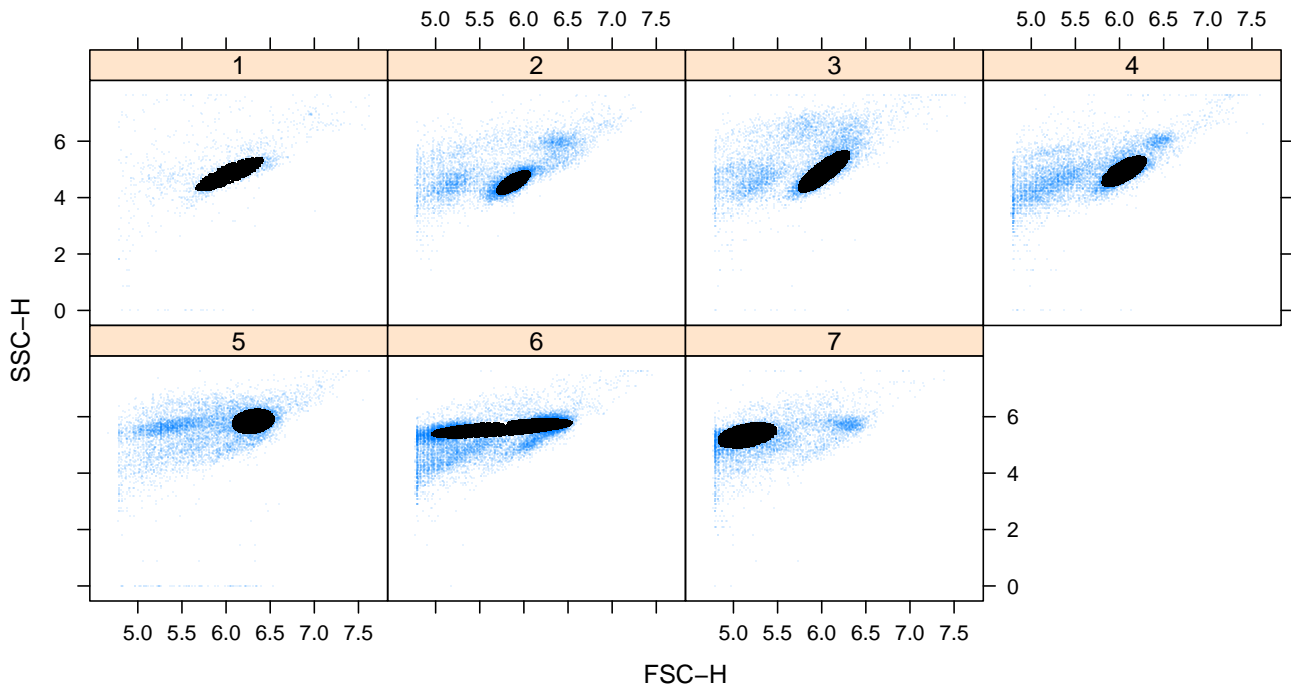

In this figure, we have both a `filter` and a `filterResults` argument. The effect of `filterResults` is exactly as it was in the previous plot. However, `filter` is applied anew, and rendered as ellipses, on the *transformed* data. As we can see, both versions localize the primary cluster in roughly the same region, except in one case.

```
> xyplot(`SSC-H` ~ `FSC-H` | Visit,
+        data = transform("SSC-H"=asinh,"FSC-H"=asinh) %on% GvHD,
+        subset = Patient == "6",
+        smooth = FALSE, alpha = 0.1,
+        filter = rectangleGate("FSC-H"=c(5.5, 7),"SSC-H"=c(4, 6.5)),
+        filterResults = n2gate.results,
+        displayFilter = list(border = 'red'))
```
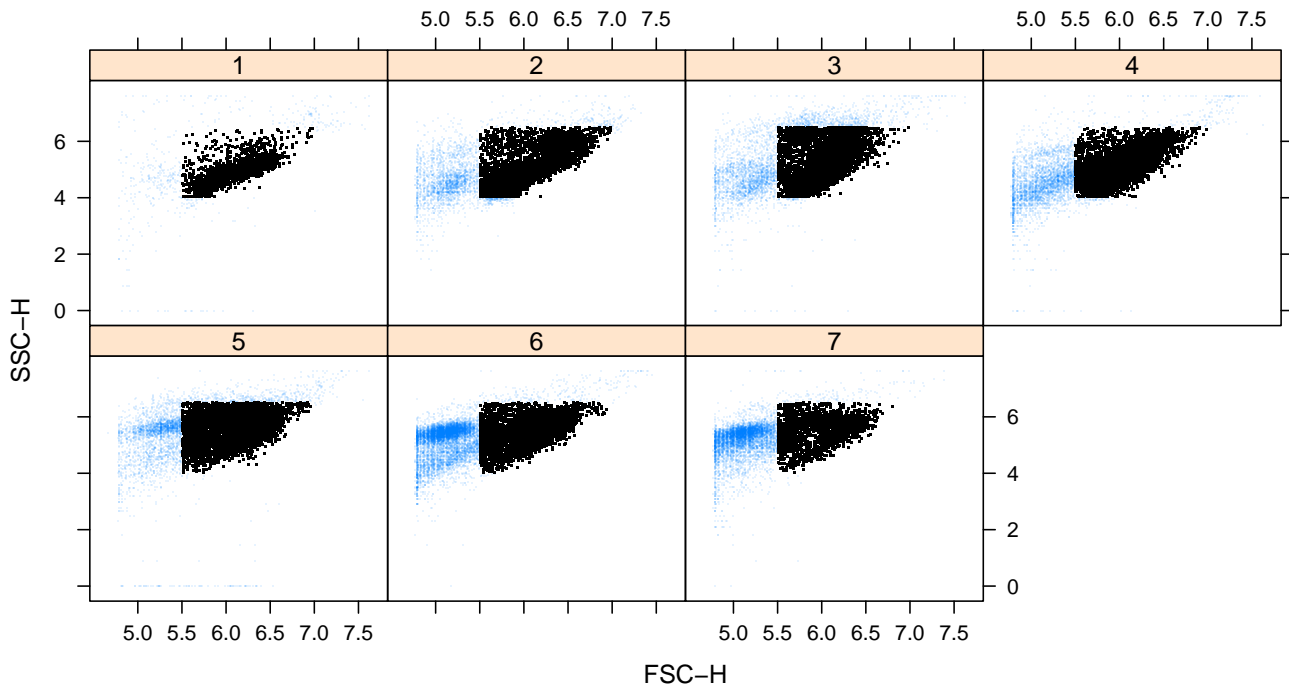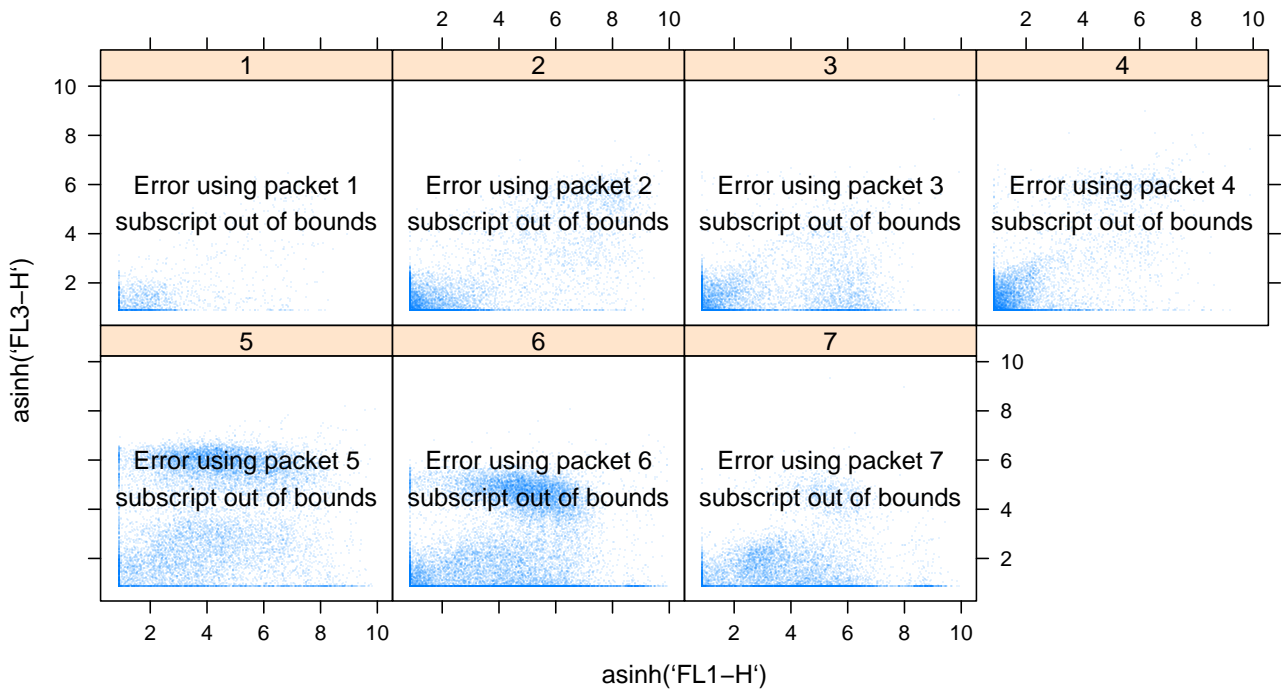

This figure is in principle similar to the previous one, except that `filter` is now the rectangle gate.

```
> xyplot(asinh(`FL3-H`) ~ asinh(`FL1-H`) | Visit, data = GvHD,
+        subset = Patient == "6",
+        smooth = FALSE, alpha = 0.1,
+        filter = rectangleGate("FSC-H"=c(5.5, 7),"SSC-H"=c(4, 6.5)),
+        filterResults = n2gate.results,
+        displayFilter = list(border = 'red'))
```


As noted earlier, rendering a filter only makes sense when variables match, whereas a filter result can be used for grouping even when this does not hold. In this figure, a `filter` argument is specified, but it is ignored because the parameters of the filter are not being plotted. However, `filterResults` is still used.

3   Filters in parallel coodinate plots

Although the `filter` argument does not make much sense except in scatter plots, `filterResults` can be used for grouping in other contexts. So far, only the `parallel` method supports this syntax.

```
> parallel(~ . | Visit,
+          data =
+          transform("SSC-H"=asinh,"FSC-H"=asinh, "FL1-H"=asinh,
+                     "FL2-H"=asinh, "FL3-H"=asinh, "FL2-A"=asinh,
+                     "FL4-H"=asinh) %on% GvHD,
+          subset = Patient == "6",
+          filterResults = n2gate.results,
+          alpha = 0.01)
```