Seminar III: R/Bioconductor Splicegear

Melvin Jesus Noe Gonzalez
Ricardo Romero Moreno

October 30, 2009

Introduction I

- With the advances in numerical processing of the data, the lowering of the costs and well defined experimental protocols, the reliability of data analysis has increased.
- However little has been done to quantify how well the technique performs, how the existing oligonucleotide data could have been in uenced by alternative phenomenon and how it could contribute to discover novel splice variants.

Introduction I

- This package defines classes to handle probe expression values in an alternative splicing context.

Installing and loading the package I

```
> library(splicegear)
```

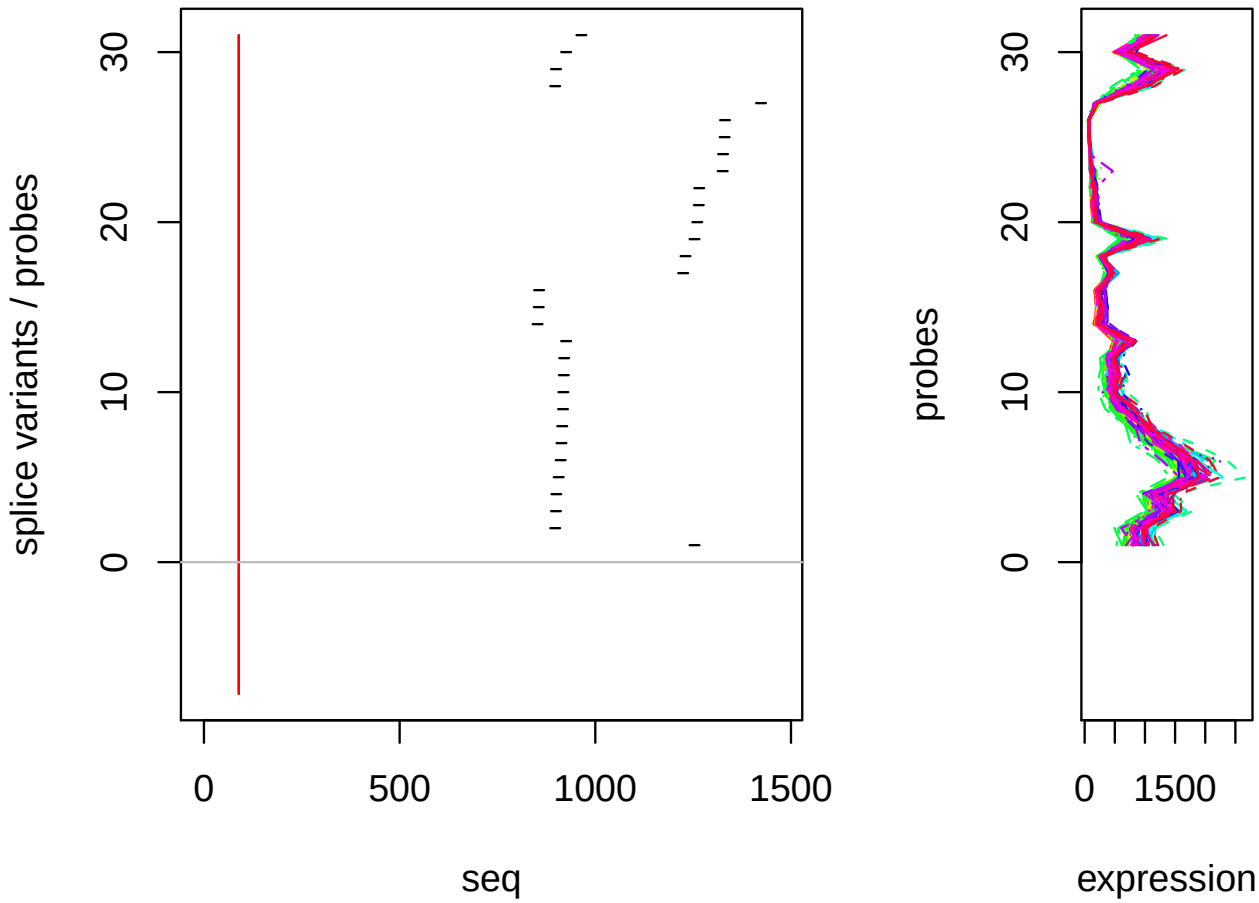
Classes and plotting

The plots are defined for the classes SPliceSites and SpliceExprSet.

Classes and Plotting I

```
> data(spliceset)
> plot(spliceset)
```

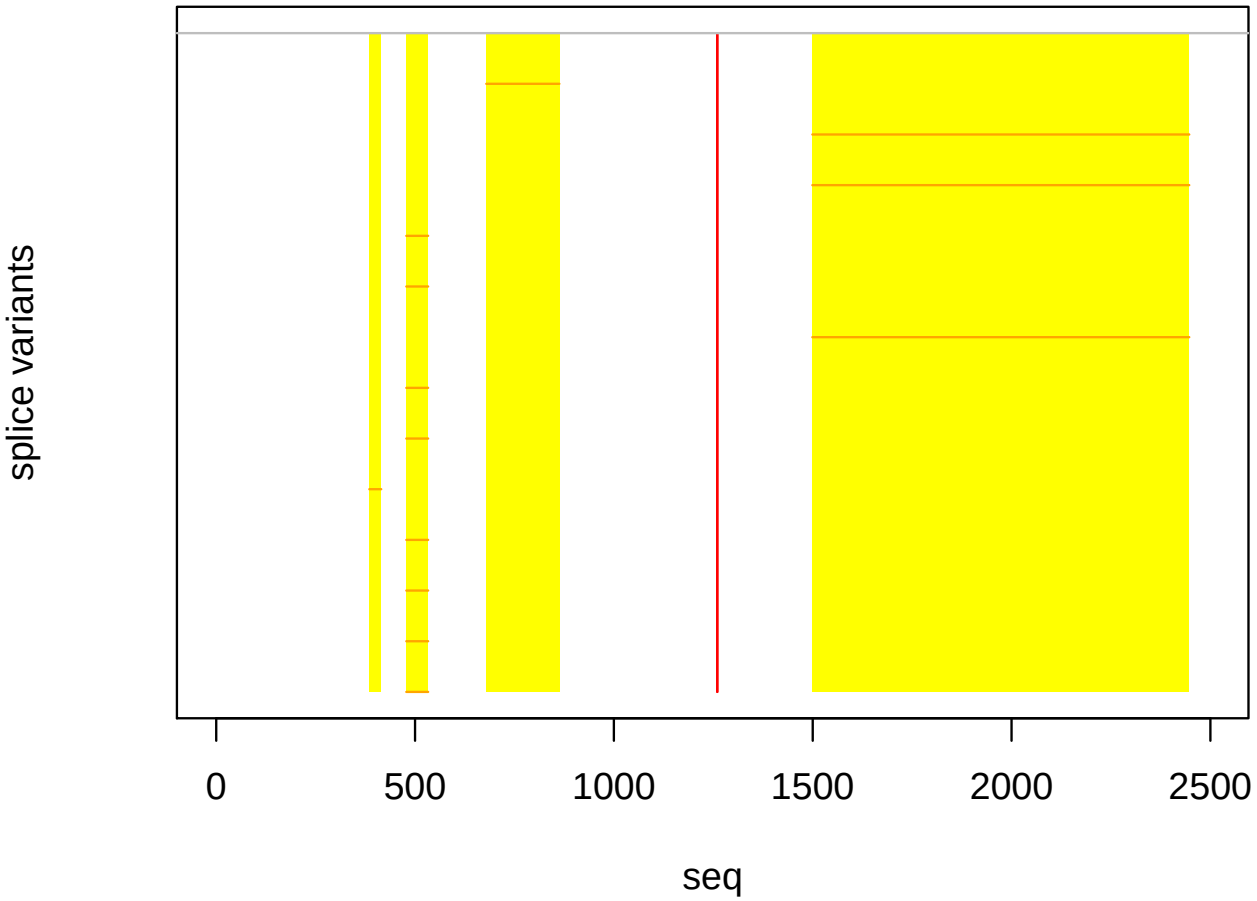
Classes and Plotting II



Classes and Plotting I

```
> data(spsites)
> plot(spsites)
> data(probes)
> plot(probes)
```

Classes and Plotting II



Classes and Plotting I

- In these data sets you can find information used in the database of putative splice sites against the probes of Aymetrix U95A chips where you can find information about the levels of expression in cells from the liver and central nervous system.

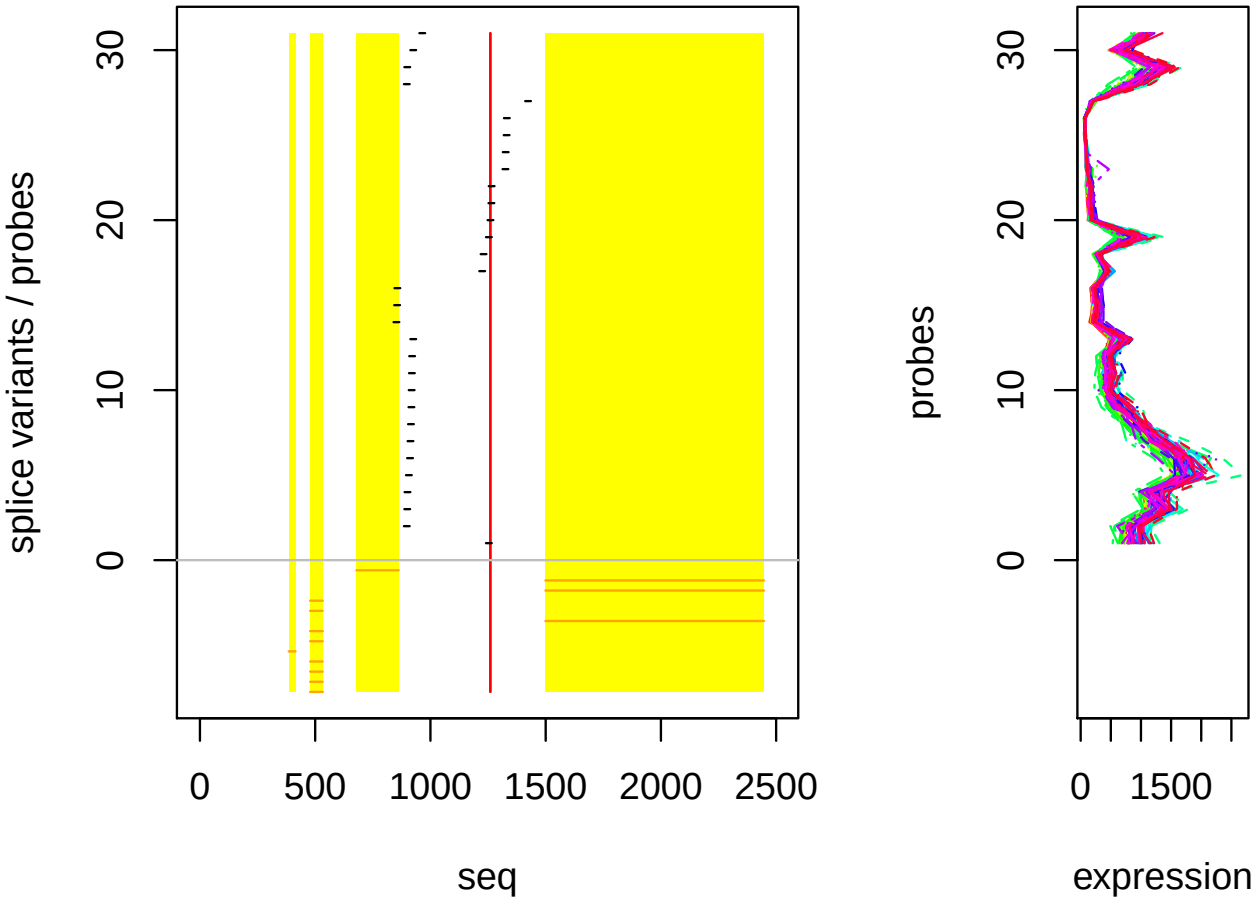
Using the classes I

- in the firs part we described how to plot individual datasets, now lets try one plot a little bit more complicated in which we create a new object of the class SpliceExprSet as a result of the fussion of two datasets.

Using the classes I

```
> data(eset, package = "splicegear")
> data(probes, package = "splicegear")
> data(spsites, package = "splicegear")
> spliset <- new("SpliceExprSet",
+     eset = eset, probes = probes,
+     spliceSites = spsites)
> plot(spliset)
```

Using the classes II



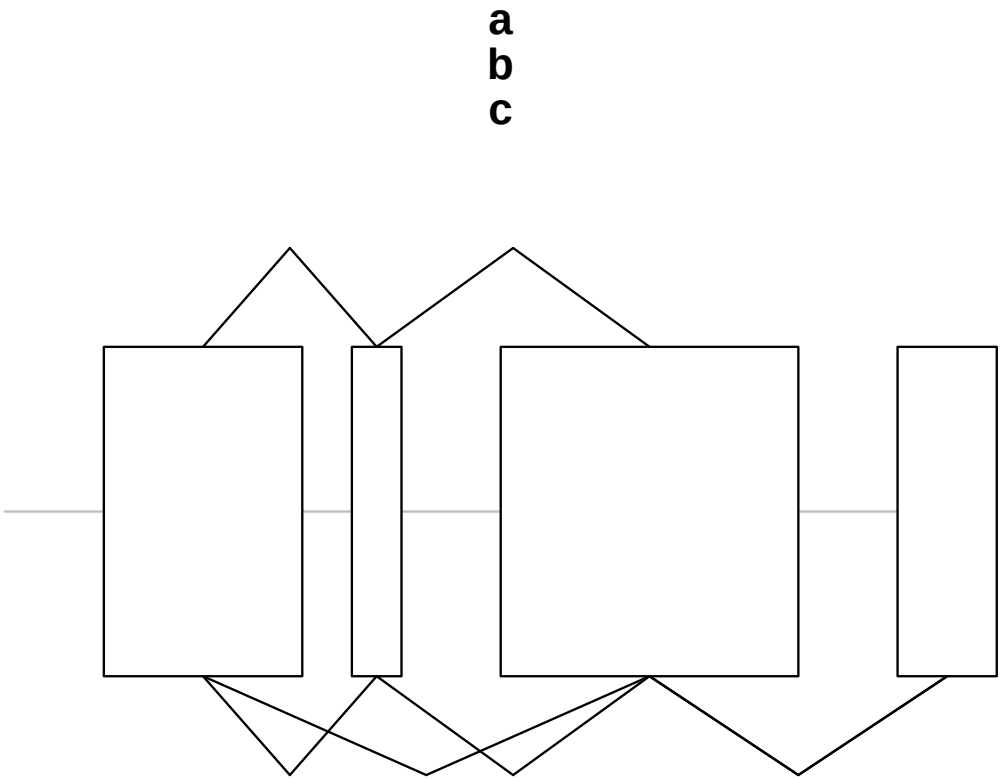
Different splice orders I

- Now lets create a new sequence in which you can find different splice orders
- We have to create an object of class SpliceSitesGenomic
- To make it possible we have to define the possible splice variants, the length of the sequence and the beggining positions of each possible splice site.

Different splice orders I

```
> seq.length <- as.integer(10)
> spsiteIpos <- matrix(c(1, 3.5,
+     5, 9, 3, 4, 8, 10), nc = 2)
> variants <- list(a = c(1, 2, 3,
+     4), b = c(1, 2, 3), c = c(1,
+     3, 4))
> spvar <- new("SpliceSitesGenomic",
+     spsiteIpos = spsiteIpos, variants = variants,
+     seq.length = seq.length)
> plot(spvar)
```

Different splice orders II

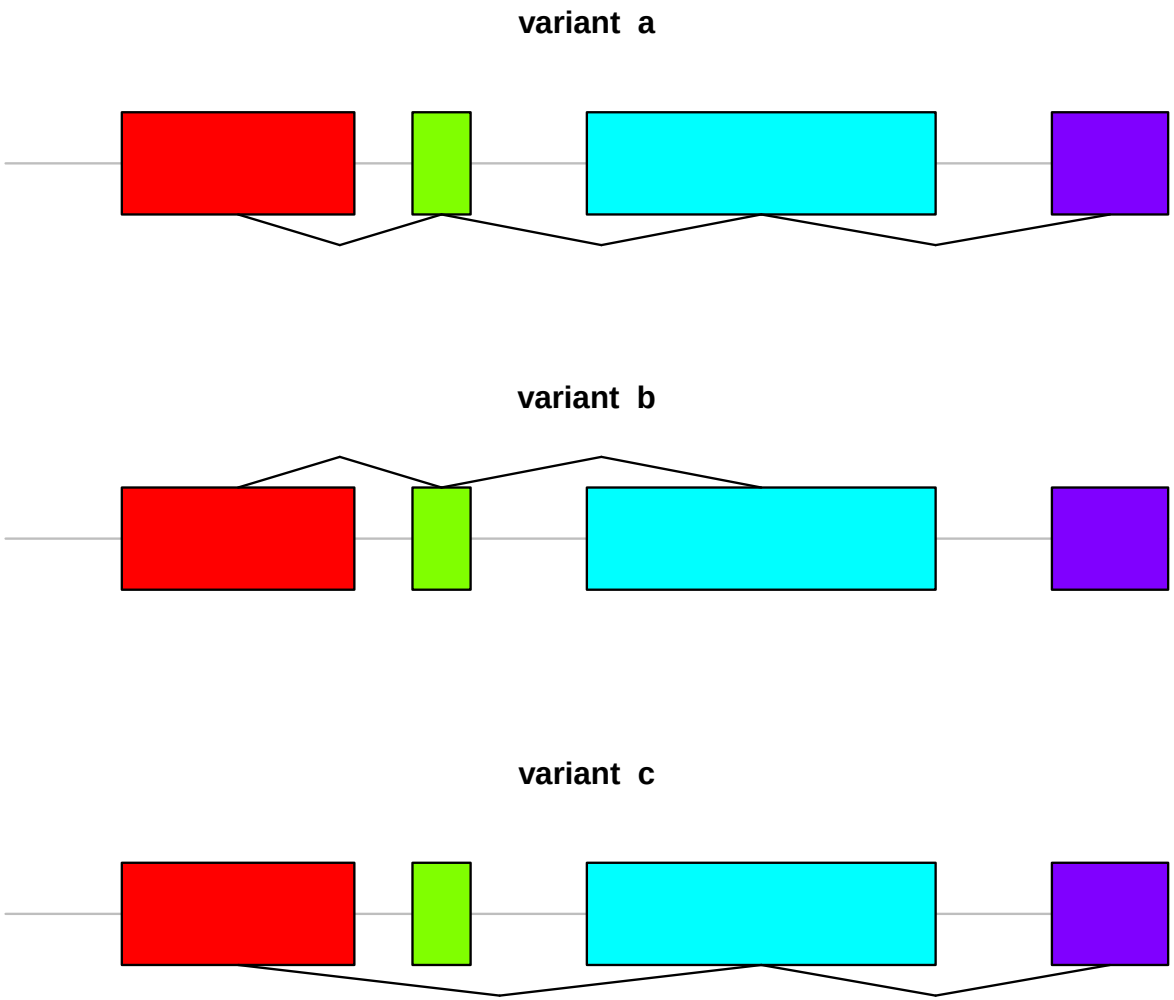


Different splice orders I

- Now to do this a bit more gentle to our eyes lets split de variants apart so you can see each one of them as if there where a plot for each one.

```
> n.exons <- nrow(spsiteIpos)
> par(mfrow = c(3, 1), mar = c(3.1,
+     2.1, 2.1, 1.1))
> plot(spvar, split = TRUE, col.exon = rainbow(n.exons)
```

Different splice orders II



Using an xml view I

- Now lets import a dataset and predict all its splice sites.

```
> library(XML)
> filename <- system.file("data",
+     "example.xml", package = "splicegear")
> xml <- xmlTreeParse(filename, asTree = TRUE)
> spsites <- buildSpliceSites(xml)

7  entrie(s) in the set.
 entrie  1  has  23  element(s).
  sub-entrie  1  has  1  element(s).
  sub-entrie  2  has  2  element(s).
  sub-entrie  3  has  3  element(s).
  sub-entrie  4  has  1  element(s).
  sub-entrie  5  has  3  element(s).
```

Using an xml view II

```
 sub-entrie  6  has  2  element(s).
 sub-entrie  7  has  7  element(s).
 sub-entrie  8  has  2  element(s).
 sub-entrie  9  has  1  element(s).
 sub-entrie  10  has  1  element(s).
entrie  2  has  52  element(s).
 sub-entrie  1  has  1  element(s).
 sub-entrie  2  has  2  element(s).
 sub-entrie  3  has  1  element(s).
 sub-entrie  4  has  1  element(s).
 sub-entrie  5  has  1  element(s).
 sub-entrie  6  has  1  element(s).
 sub-entrie  7  has  1  element(s).
 sub-entrie  8  has  31  element(s).
```

Using an xml view III

```
 sub-entrie  9  has  1  element(s).
 sub-entrie  10  has  7  element(s).
 sub-entrie  11  has  2  element(s).
 sub-entrie  12  has  1  element(s).
 sub-entrie  13  has  1  element(s).
 sub-entrie  14  has  1  element(s).
entrie  3  has  2  element(s).
 sub-entrie  1  has  2  element(s).
entrie  4  has  17  element(s).
 sub-entrie  1  has  2  element(s).
 sub-entrie  2  has  2  element(s).
 sub-entrie  3  has  8  element(s).
 sub-entrie  4  has  1  element(s).
 sub-entrie  5  has  3  element(s).
```

Using an xml view IV

```
 sub-entrie  6  has  1  element(s).
entrie  5  has  11  element(s).
 sub-entrie  1  has  1  element(s).
 sub-entrie  2  has  8  element(s).
 sub-entrie  3  has  2  element(s).
entrie  6  has  5  element(s).
 sub-entrie  1  has  4  element(s).
 sub-entrie  2  has  1  element(s).
entrie  7  has  18  element(s).
 sub-entrie  1  has  2  element(s).
 sub-entrie  2  has  9  element(s).
 sub-entrie  3  has  1  element(s).
 sub-entrie  4  has  4  element(s).
```

Using an xml view V

```
sub-entrie  5  has  1  element(s).
sub-entrie  6  has  1  element(s).
```

Using XML I

```
> data(spliceset)
> dataf <- as.data.frame(spliceset)
> lm.panel <- function(x, y, ...) {
+     points(x, y, ...)
+     p.lm <- lm(y ~ x)
+     abline(p.lm)
+ }
```

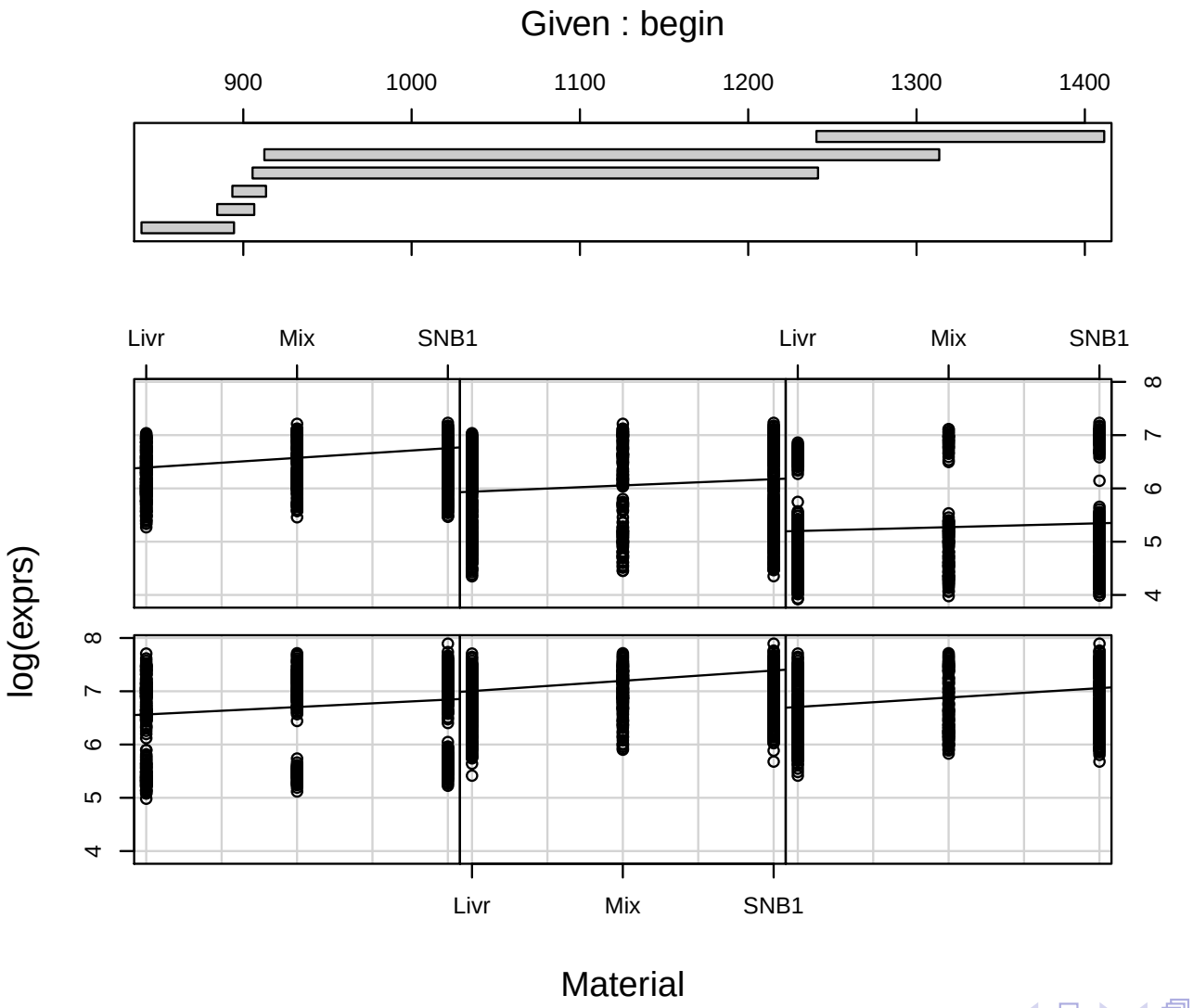
And now? I

- Probe intensity values conditioned by the position of the probes on the mRNA

One interesting plot I

```
> coplot(log(exprs) ~ Material |
+     begin, data = dataf, panel = lm.panel)
```

One interesting plot II



The end I

- Thank you very much!