Multivariate plots for bioinformatics data using ggplot2 and GGally

Di Cook & Hadley Wickham
@visnut & @hadleywickham

July 2015

Outline

- Types of plots: Scatterplot matrix, parallel coordinate plot

- Designed experiment: Interaction plots, porcupine plots

- Reducing dimensionality: Multidimensional scaling, projection pursuit, tours

- Inference

RNA-Seq Analysis

- Anders, S., McCarthy, D. J., Chen, Y., Okoniewski, M., Smyth, G. K., Huber, W. and Robinson, M. D. (2013) "Count-based differential expression analysis of RNA sequencing data using R and Bioconductor", Nature Protocols, 8(9):1765-1786, doi:10.1038/nprot.2013.099.

- Protocol for conducting differential expression analysis

- Download the data from the supplemental material

Example data

- Brooks, A.N. et al. Conservation of an RNA regulatory map between Drosophila and mammals. Genome Res. 21, 193–202 (2011)
- This data set consists of seven RNA-seq samples, each a cell culture of *Drosophila melanogaster* S2 cells. Three samples were treated with siRNA targeting the splicing factor *pasilla* (CG1844) (‘knockdown’) and four samples are untreated (‘control’). Our aim is to identify genes that change in expression between knockdown and control. Brooks *et al.*have sequenced some of their libraries in single-end mode and others in paired-end mode.

Setting up

```
> # Load the data
> library(edgeR)
> counts = readDGE(samples$countf)$counts
> dim(counts)
[1] 15686     7
> # Filter weakly expressed and noninformative (e.g., non-aligned)
features
> noint = rownames(counts) %in%
+     c("no_feature","ambiguous","too_low_aQual",
+       "not_aligned","alignment_not_unique")
> cpms = cpm(counts)
> counts.all <- counts[,order(samples$condition)]
> colnames(counts.all) = samples$shortname[order(samples$condition)]
> head(counts.all)
            CT.PA.1 CT.PA.2 CT.SI.5 CT.SI.7 KD.PA.3 KD.PA.4 KD.SI.6
FBgn0000008      76      71     137      82      87      68     115
FBgn0000014       2       4       1       1       9       1       0
FBgn0000015       1       2       0       0       3       1       2
FBgn0000017    3498    3087    7014    3926    3029    3264    4322
FBgn0000018     240     306     613     485     288     307     528
FBgn0000022       0       0       1       0       0       0       0
```

Filtering

```
> # Keep only genes with at least 1 read per million, for the
> # smallest number of replicates in  treatment
> keep = rowSums(cpms > 1) >= 3 & !noint
> counts = counts.all[keep,]
> head(counts, 5)
            CT.PA.1 CT.PA.2 CT.SI.5 CT.SI.7 KD.PA.3 KD.PA.4 KD.SI.6
FBgn0000008      76      71     137      82      87      68     115
FBgn0000017    3498    3087    7014    3926    3029    3264    4322
FBgn0000018     240     306     613     485     288     307     528
FBgn0000032     611     672    1479    1351     694     757    1361
FBgn0000042   40048   49144   97565   99372   70574   72850   95760
```

Library size & normalization

```
> d = DGEList(counts = counts, group = samples$condition[order(samples
$condition)])
> str(d)
Formal class 'DGEList' [package "edgeR"] with 1 slot
  ..@ .Data:List of 2
  .. ..$ : num [1:7196, 1:7] 76 3498 240 611 40048 ...
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:7196] "FBgn0000008" "FBgn0000017" "FBgn0000018"
"FBgn0000032" ...
  .. .. .. ..$ : chr [1:7] "CT.PA.1" "CT.PA.2" "CT.SI.5" "CT.SI.7" ...
  .. ..$ :'data.frame': 7 obs. of  3 variables:
  .. .. ..$ group        : Factor w/ 2 levels "CTL","KD": 1 1 1 1 2 2 2
  .. .. ..$ lib.size     : num [1:7] 8397136 9909691 19087995 12812818
9664838 ...
  .. .. ..$ norm.factors: num [1:7] 1 1 1 1 1 1 1
```

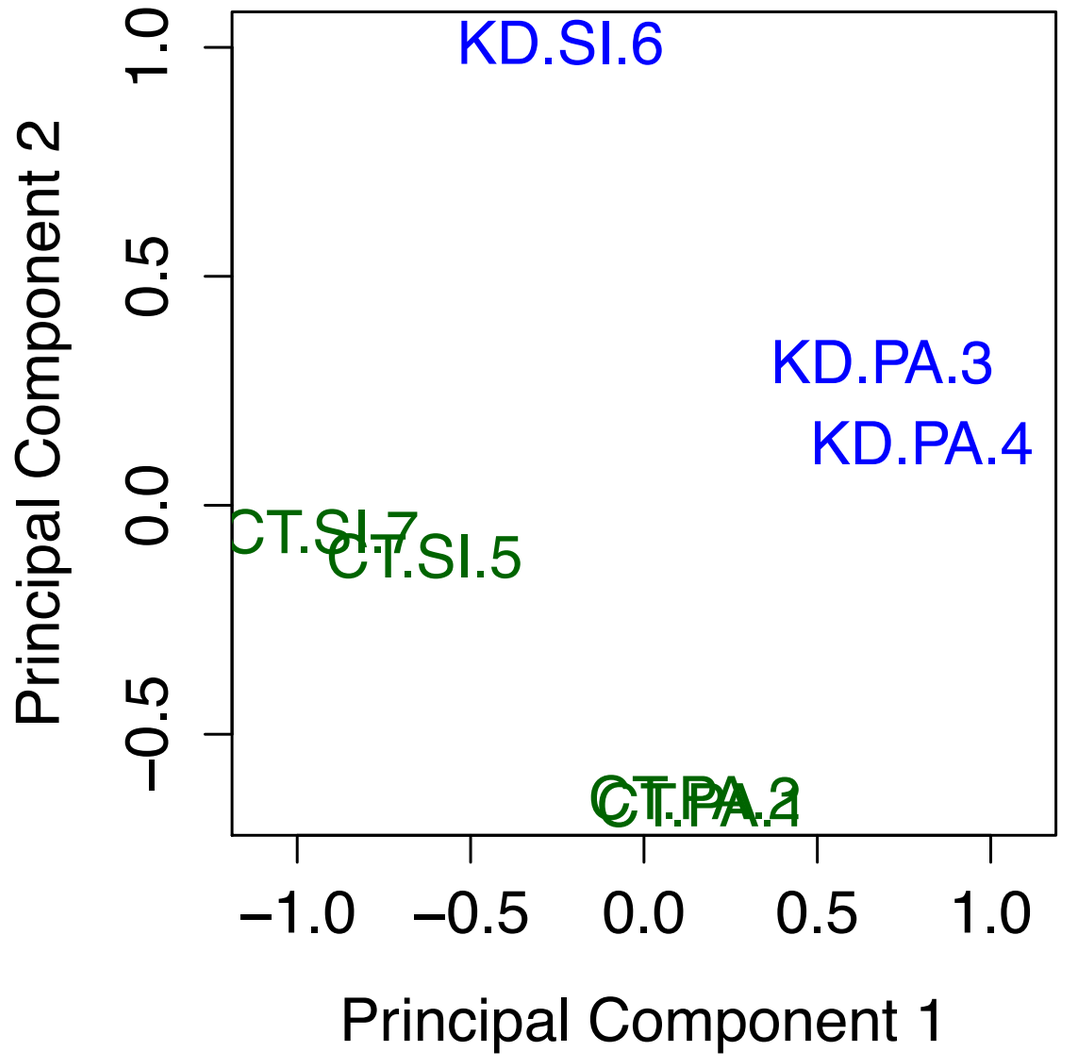
Library size & normalization

```
> d = calcNormFactors(d)
> str(d)
Formal class 'DGEList' [package "edgeR"] with 1 slot
  ..@ .Data:List of 2
  .. ..$ : num [1:7196, 1:7] 76 3498 240 611 40048 ...
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:7196] "FBgn0000008" "FBgn0000017" "FBgn0000018"
"FBgn0000032" ...
  .. .. .. ..$ : chr [1:7] "CT.PA.1" "CT.PA.2" "CT.SI.5" "CT.SI.7" ...
  .. ..$ :'data.frame': 7 obs. of  3 variables:
  .. .. ..$ group       : Factor w/ 2 levels "CTL","KD": 1 1 1 1 2 2 2
  .. .. ..$ lib.size    : num [1:7] 8397136 9909691 19087995 12812818
9664838 ...
  .. .. ..$ norm.factors: num [1:7] 0.97 0.965 1.001 1.015 0.997 ...
```

Checking samples

```
> d$samples
         group lib.size norm.factors
CT.PA.1    CTL  8397136    0.9702373
CT.PA.2    CTL  9909691    0.9652457
CT.SI.5    CTL 19087995    1.0009795
CT.SI.7    CTL 12812818    1.0145053
KD.PA.3     KD  9664838    0.9973330
KD.PA.4     KD 10325828    1.0146062
KD.SI.6     KD 15324886    1.0391230
> par(pty="s")
> d.mds <- plotMDS(d, labels = samples$shortname[order(samples$condition)],
        col = c(“darkgreen","blue")[factor(samples$condition[
                order(samples$condition)])],
        gene.selection = "common")
```

- Which are similar?
- Does this match the experimental design?



- Looks good
- Can you trust it?

Normalization

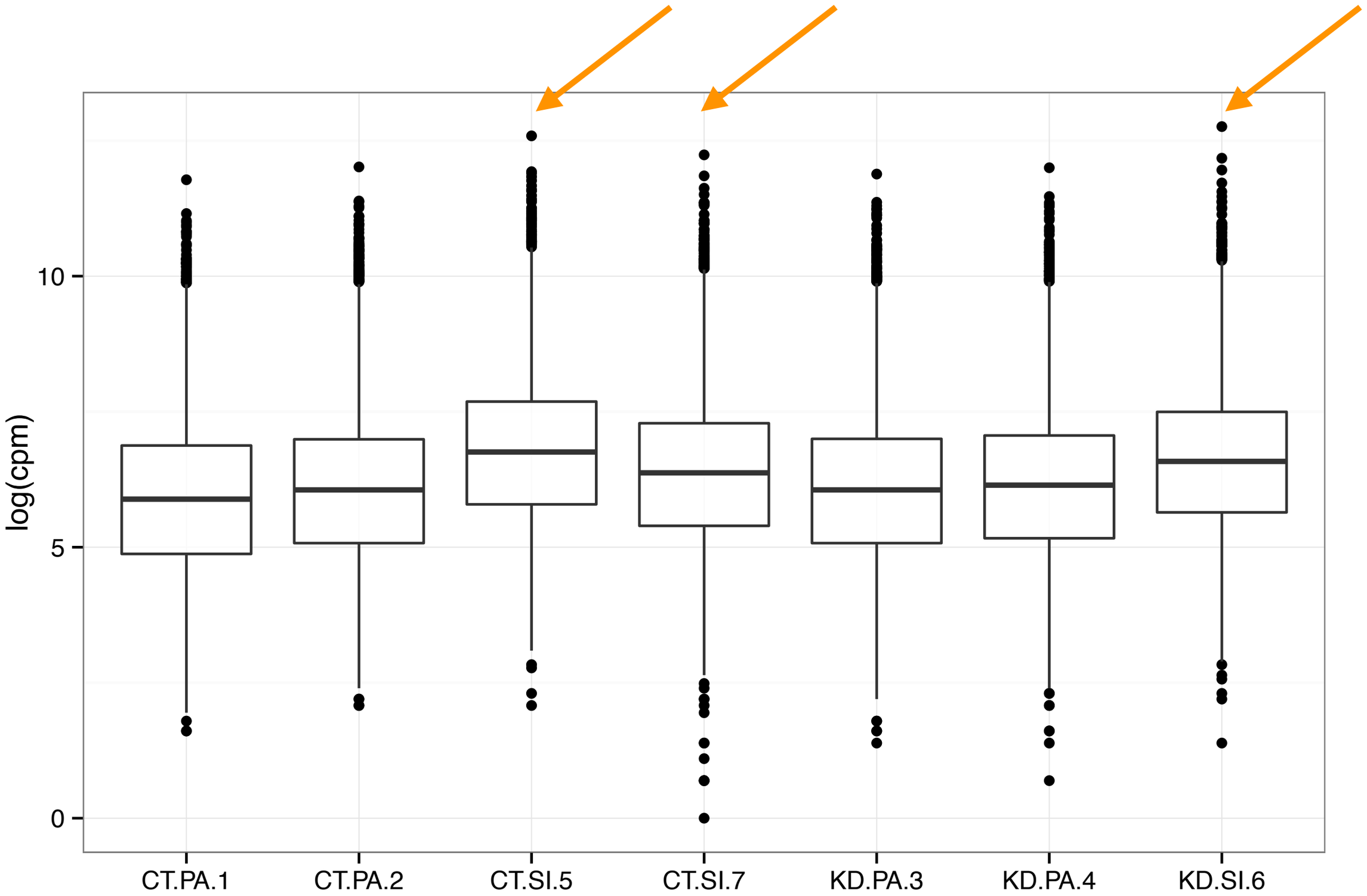
- It is important for the overall distribution from one sample to another to have the same distribution
- Why?
- To achieve this we need to normalize the distributions of each sample so that they are very similar

Raw data

- Side-by-side boxplots are a simple way to check the distribution of the raw data

- Needs to be on a log scale

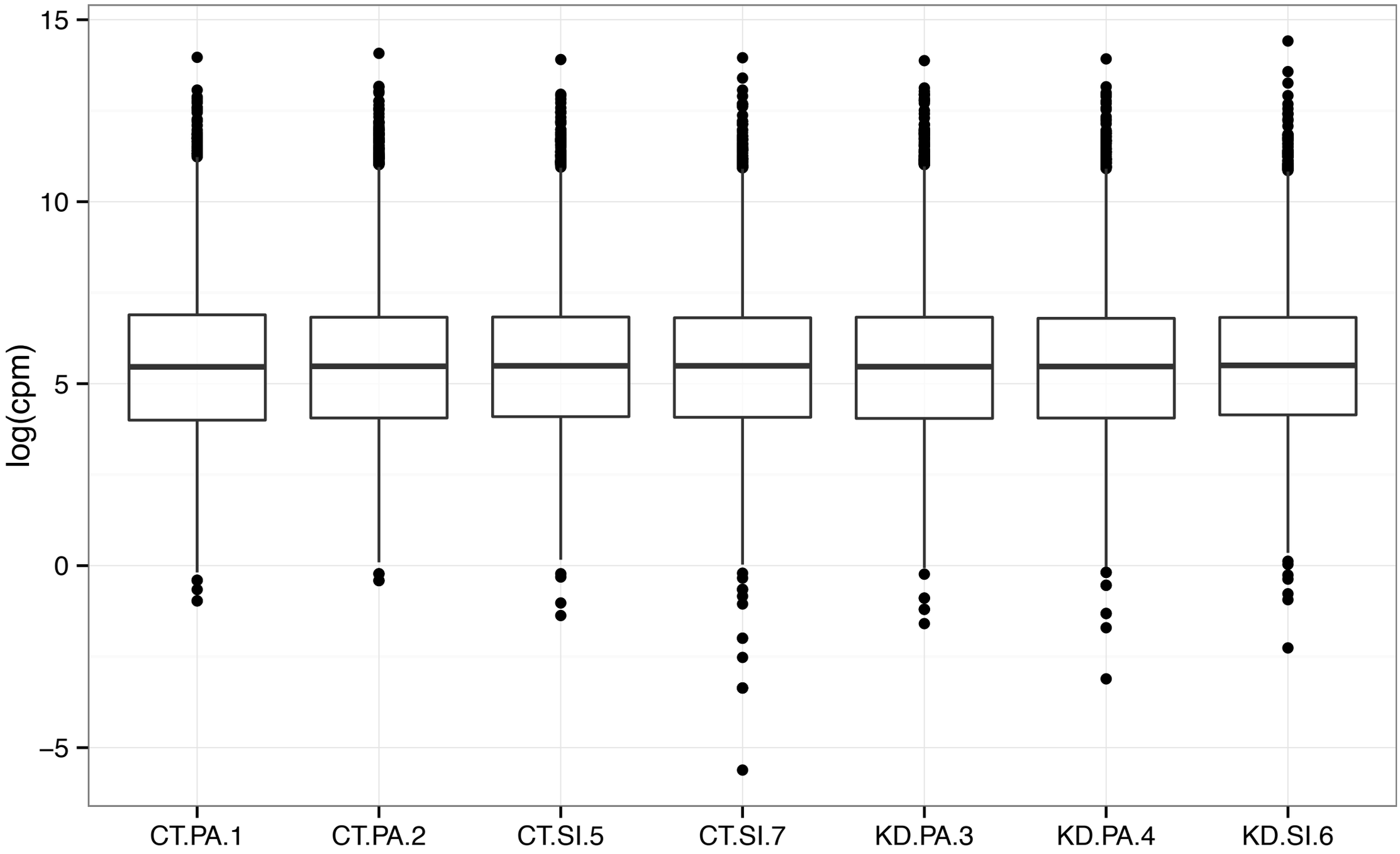
```
> library(ggplot2)
> library(GGally)
> d$counts <- data.frame(d$counts) # Needs to be a data frame
> d$lcounts <- log(d$counts+1)
> ggparcoord(d$lcounts, columns=1:7, boxplot=TRUE,
scale="globalminmax", showPoints=FALSE,
+             alphaLines=0) +
+   xlab("") + ylab("log(cpm)") +
+   theme_bw()
```



- Single-end reads have different distribution to paired end reads

Normalized data

```
> library(GGally)
> d = calcNormFactors(d, method="none")
> d$counts <- data.frame(d$counts) # Needs to be a data frame
> d$lcounts <- log(d$counts+1)
> ggparcoord(d$lcounts, columns=1:7, boxplot=TRUE,
    scale="globalminmax", alphaLines=0)
```



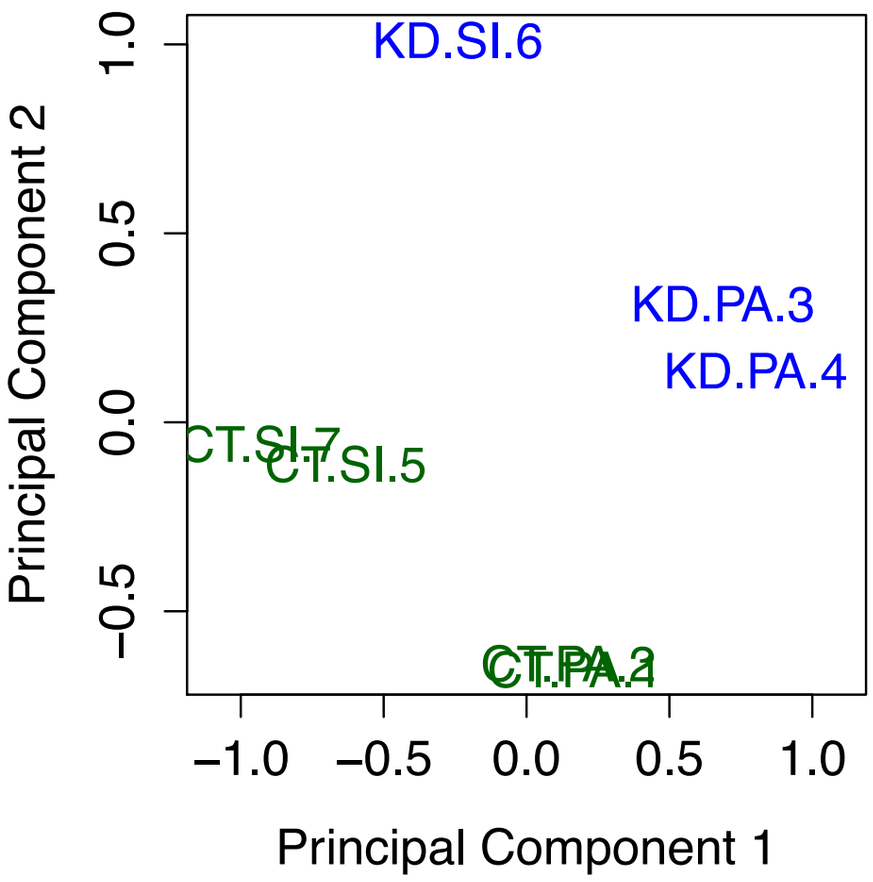
- Distributions now look very similar

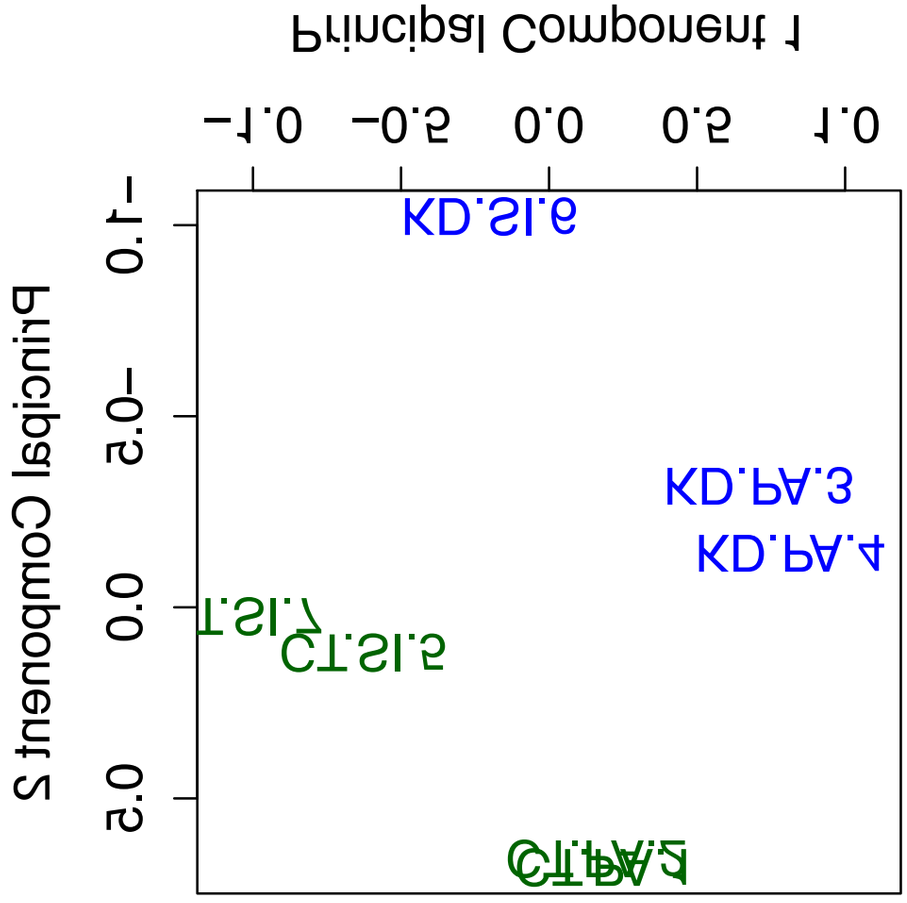
Your Turn

- Read in the katz mouse data: Katz, Wang, Eiroldi, Burge "Analysis and design of RNA sequencing experiments for identifying isoform regulation" Nat Methods. Dec 2010; 7(12): 1009–1015.
- Two classes: control mouse myoblasts, and myoblasts depleted of the splicing factor CUGBP1
- Check the normalization results.

```
# Load the data
> load("katz_mouse_eset.RData")
> str(katz.mouse.eset)
> library(Biobase)
> katz.count.table <- exprs(katz.mouse.eset)
> dim(katz.count.table)
> head(katz.count.table)
> colnames(katz.count.table) <- c("CUGBP1.1", "CONTROL.1",
  "CUGBP1.2", "CONTROL.2")
> counts.df <- data.frame(katz.count.table[,c(1,3,2,4)])
```

```
> d.mds2 <- plotMDS(nc, labels = samples$shortname[
    order(samples$condition)],
    col = c("darkgreen","blue")[
    factor(samples$condition[
     order(samples$condition)])],
    gene.selection = "common", xlim=c(-1.1, 1.1))
```





Why are they similar?

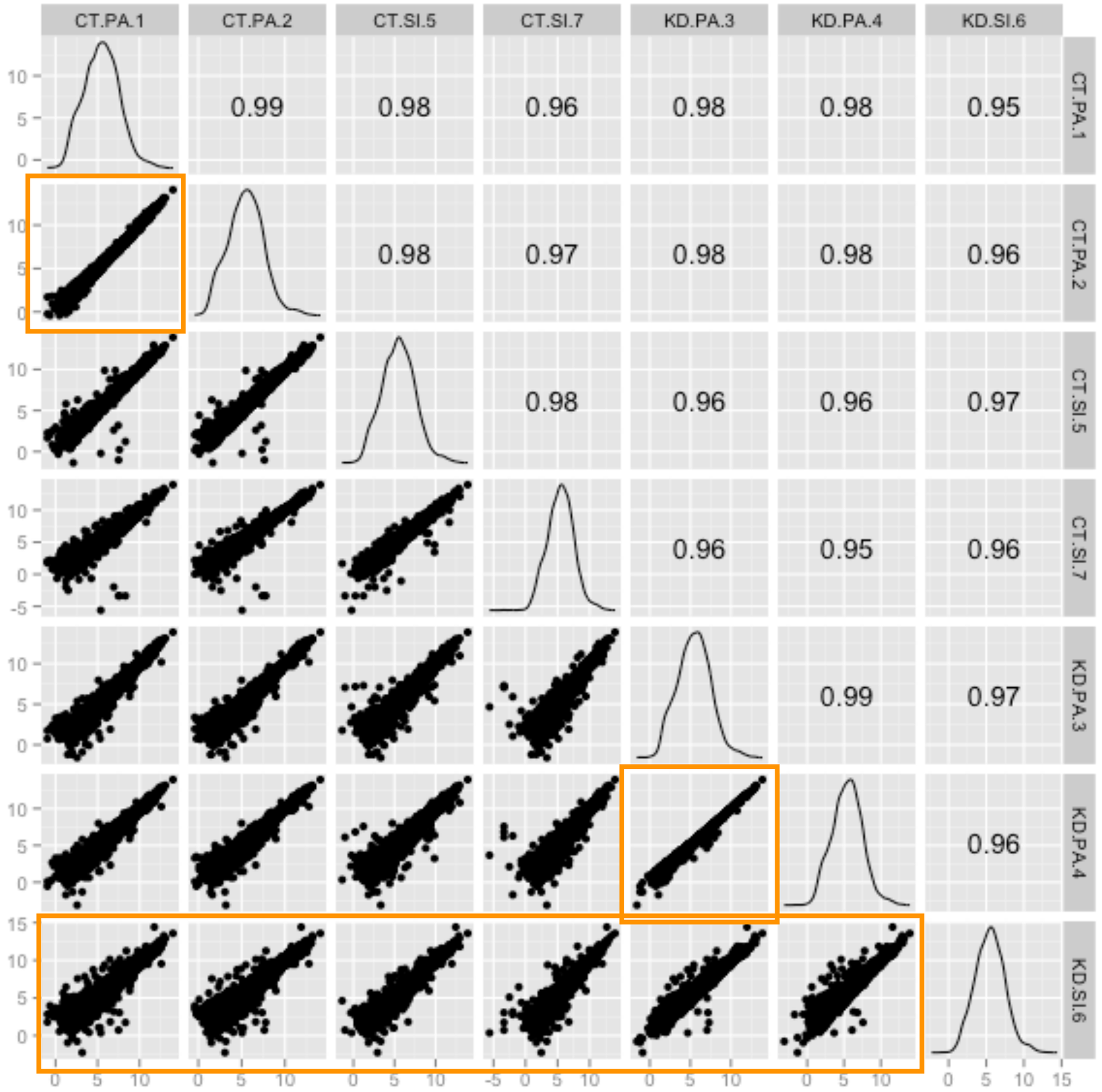
- Need to compare the values of the genes on on sample against another.
- Best way to do this is to make a scatterplot matrix of all pairs of samples

```
> ggscatmat(nc)
```

CT.PA.1, CT.PA.2
very similar

KD.PA.3, KD.PA.4
very similar

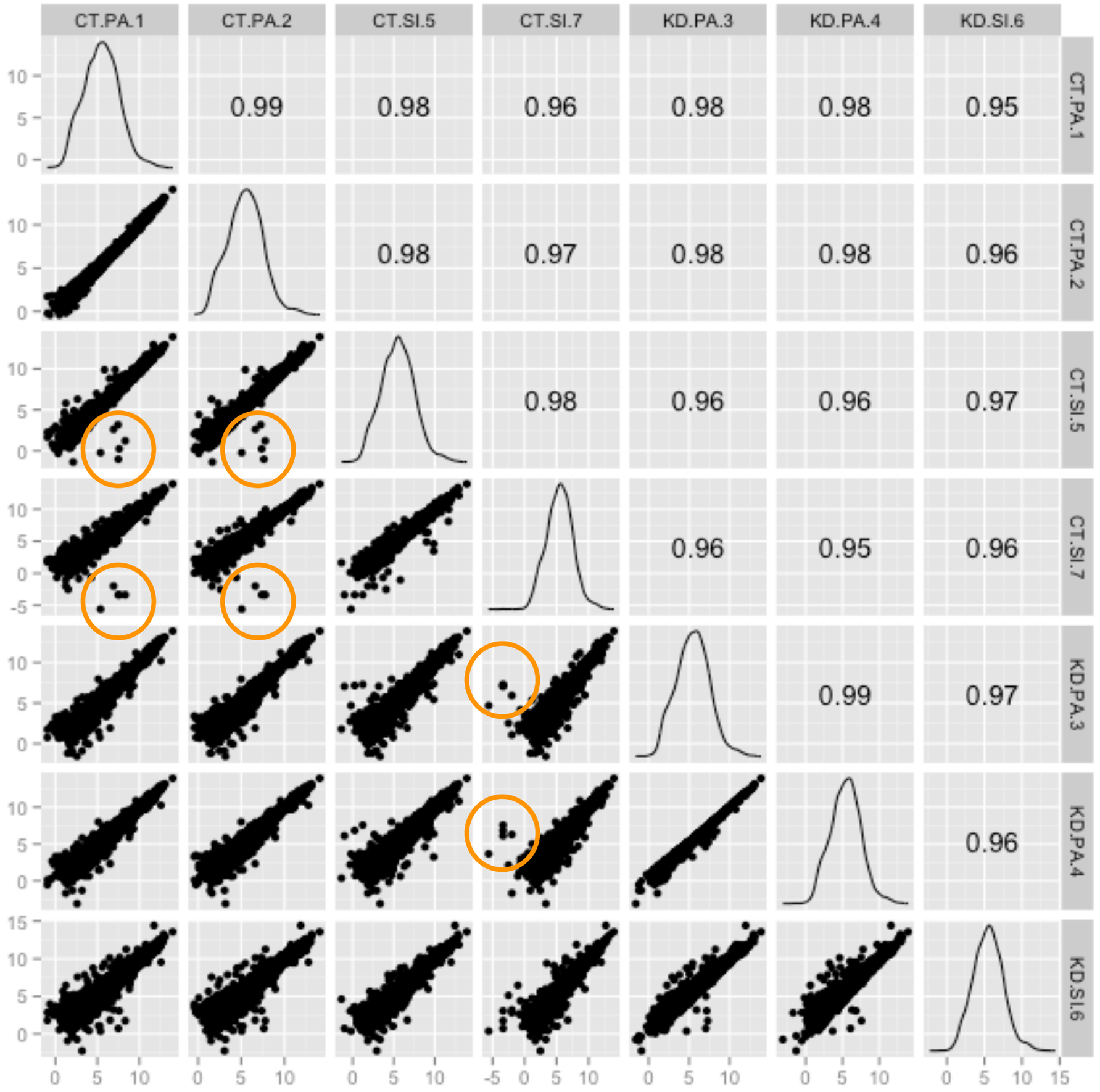
KD.SI.6 not as
similar to any



What else do we learn?

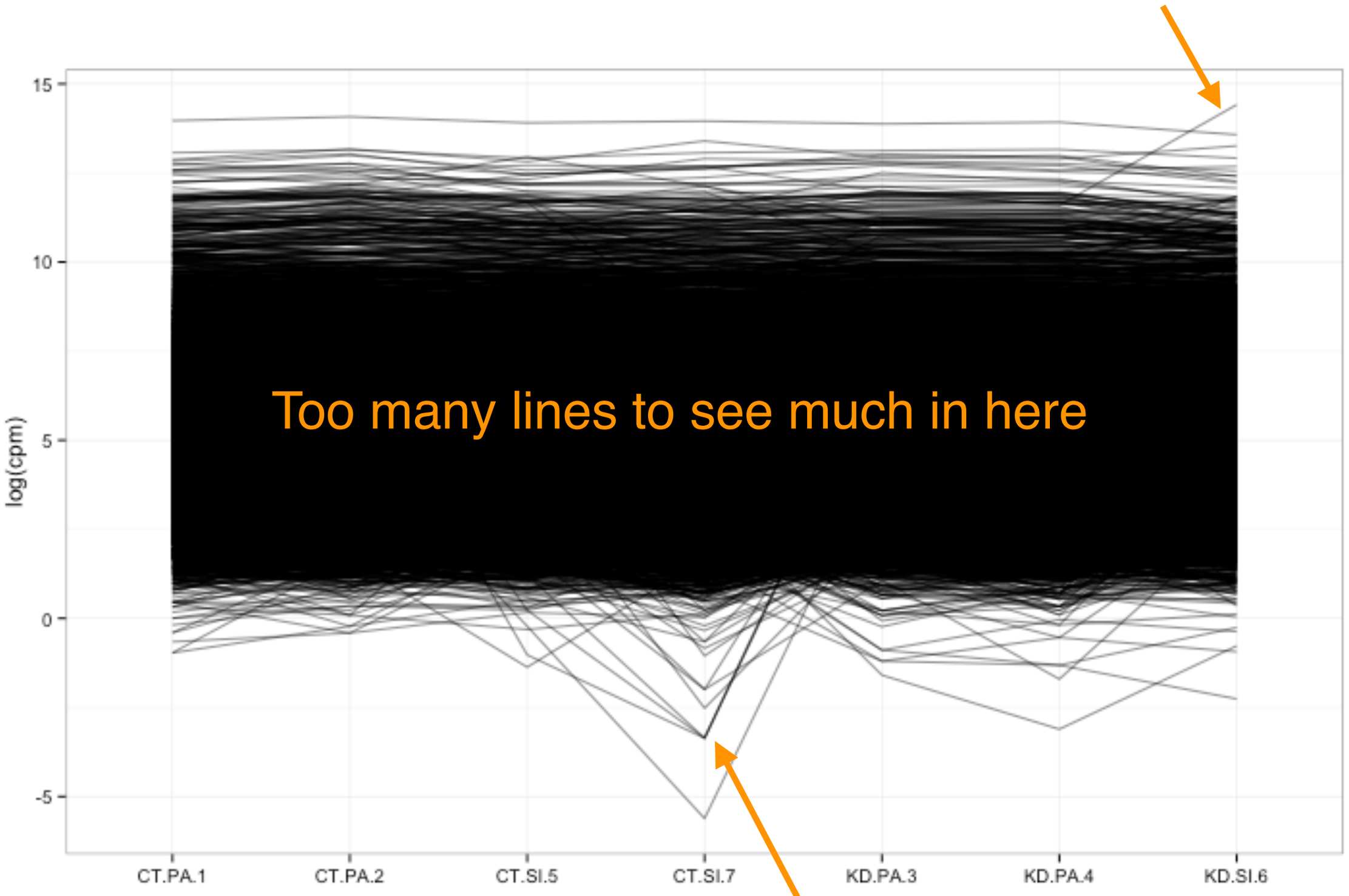
- The correlations are very high, which matched pattern in all plots, that most counts are similar from one sample to another
- There are a handful of genes that have different counts on some samples

Handful of genes behaving differently

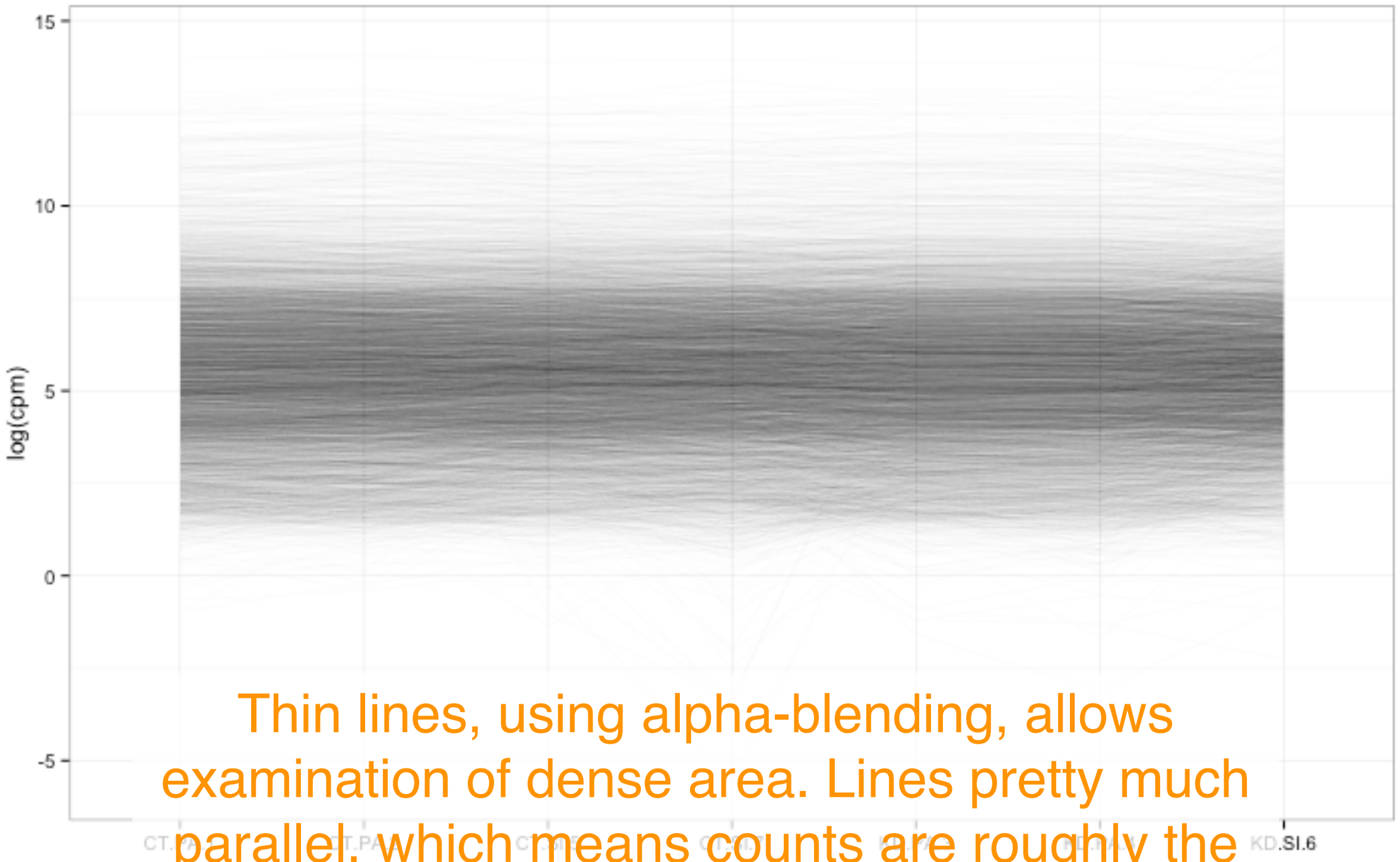


Parallel coordinate plot

- Like the side-by-side boxplot
- Lines connect counts of same gene from one sample to the next
- Another way to compare samples, particularly for different counts of genes on some samples



a few genes have low counts on CT.SI.7
relative to counts on other samples



Thin lines, using alpha-blending, allows examination of dense area. Lines pretty much parallel, which means counts are roughly the same across samples for most genes

Focus on paired-end

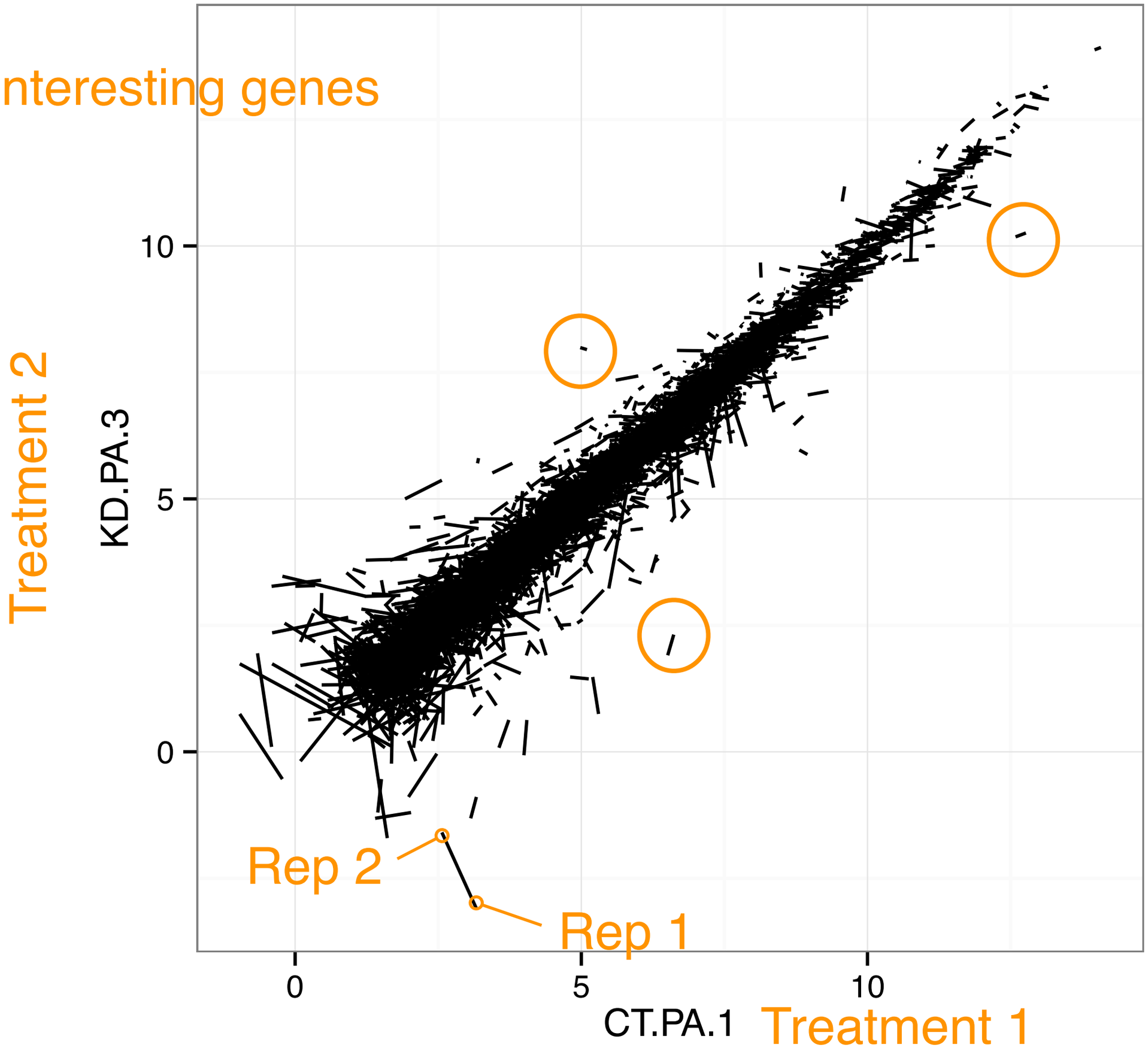
- Want to examine how the counts differ across knockdown and control relative to reps

- Only two treatments, two reps so display is simple

- Porcupine plots: : T1 vs T2, both reps shown and connected by a line

```
> ggplot(nc) + geom_segment(aes(x=CT.PA.1, xend=CT.PA.2,
    y=KD.PA.3, yend=KD.PA.4)) +
    theme_bw() + theme(aspect.ratio=1)
```

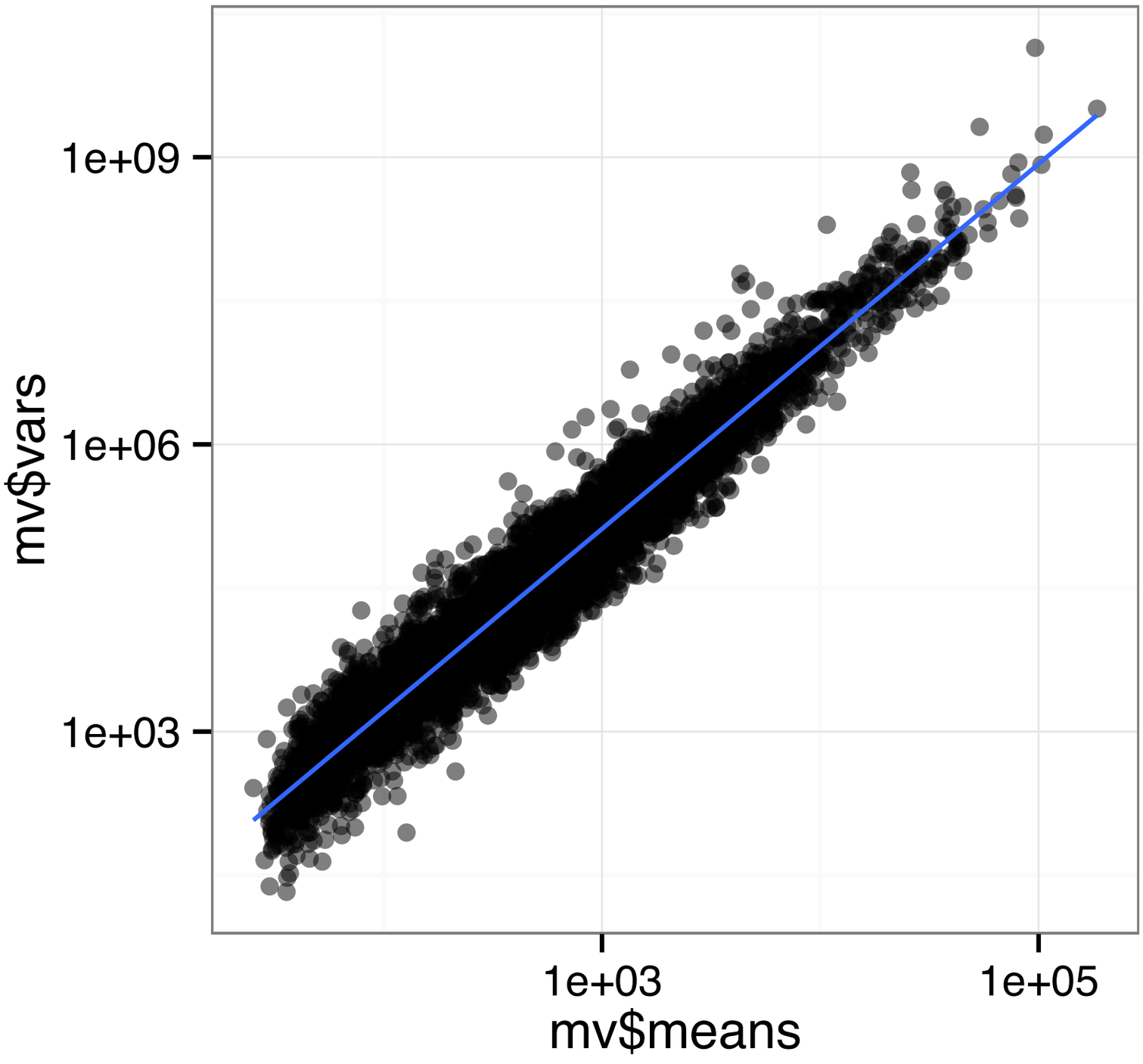
Interesting genes



Examine mean vs dispersion

- Poisson distributions for count data require that the mean equals the variance
- This needs to be checked for the significance testing
- A negative binomial model can be used when the variance is in a proportional ratio to the mean

```
> d = estimateCommonDisp(d)
> d$common.dispersion
[1] 0.03329083
> d = estimateTagwiseDisp(d)
> summary(d$tagwise.dispersion)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01647 0.02187 0.02589 0.03540 0.03599 1.67900
> d$prior.n
[1] 2
> mv <- binMeanVar(d, group = d$samples$group)
> qplot(mv$means, mv$vars, alpha=I(0.5)) +
   scale_x_log10() + scale_y_log10() +
   geom_smooth(method="lm") + theme_bw() +
   theme(aspect.ratio=1)
```



Nice linear relationship, variance is proportional to mean

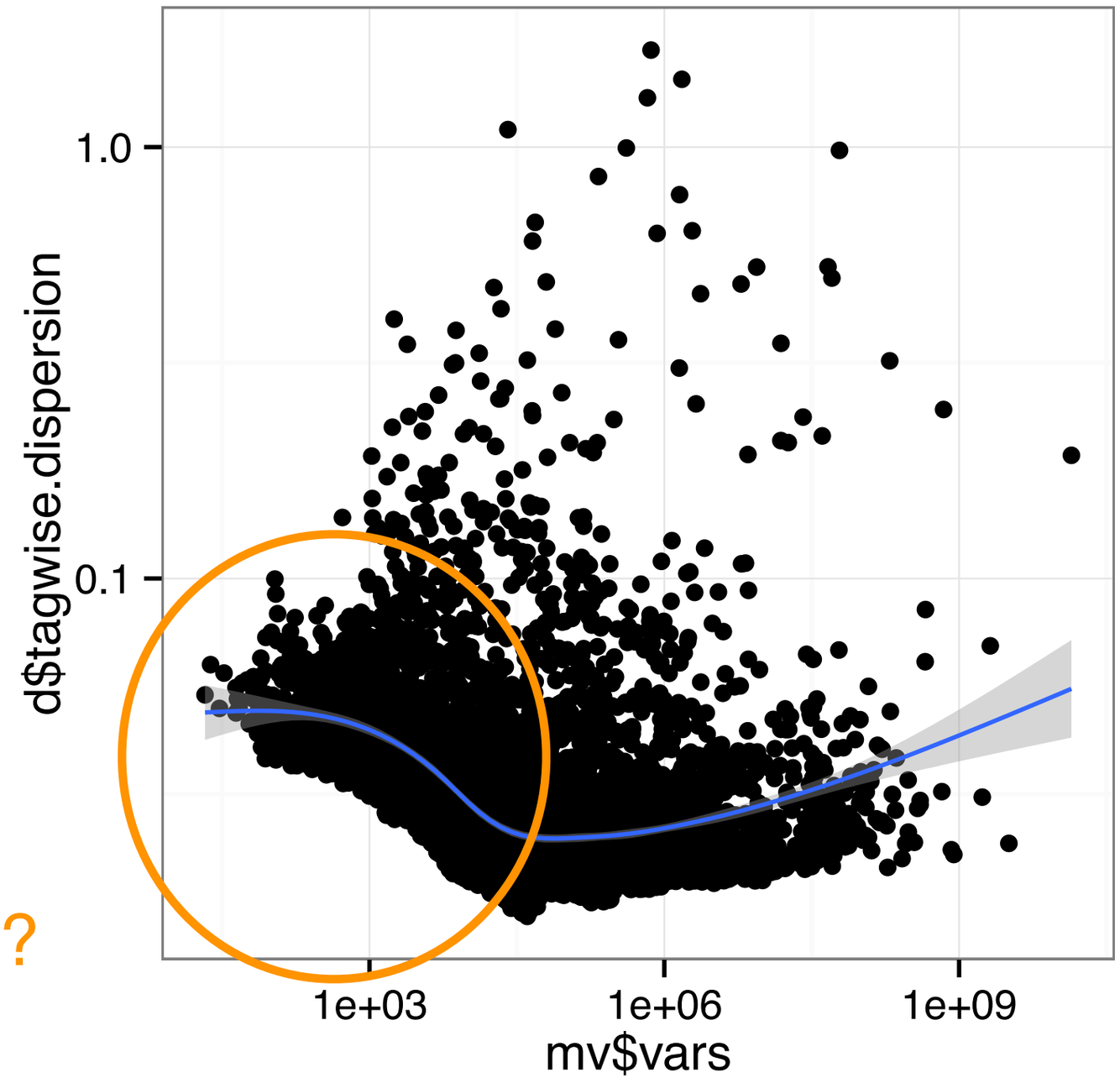
Variance vs dispersion

- In order to test for significant difference in expression, means need to be compared with variance.

- However, there is a choice here: (1) Compare individuals on their own scale, (2) Use the global scale, (3) Some combination.

- Tagwise is a weighted average of the common variance with the individual gene variance.

```
> qplot(mv$vars, d$tagwise.dispersion) +
   scale_x_log10() + scale_y_log10() +
   geom_smooth() + theme_bw() +
   theme(aspect.ratio=1)
```

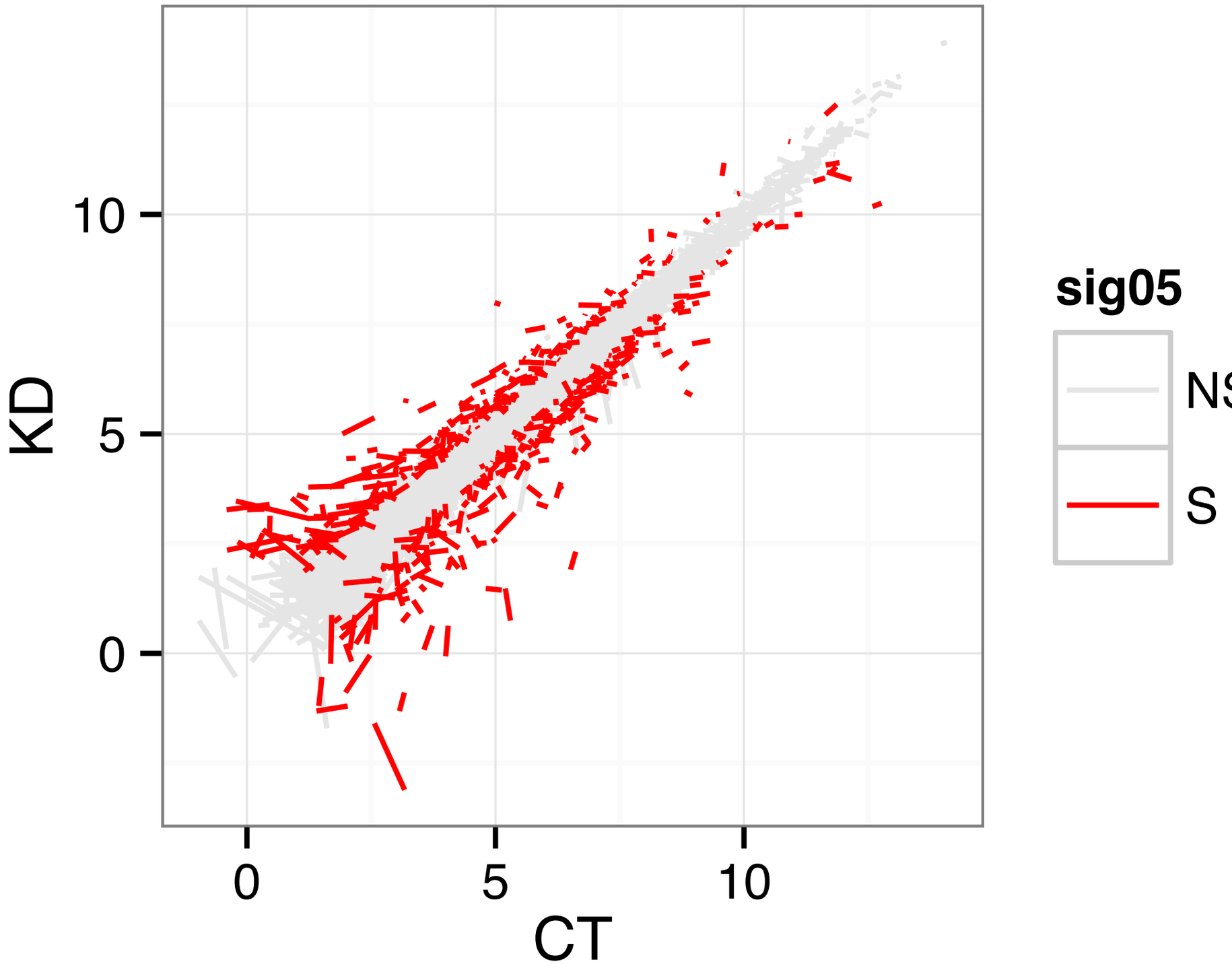


What's this??

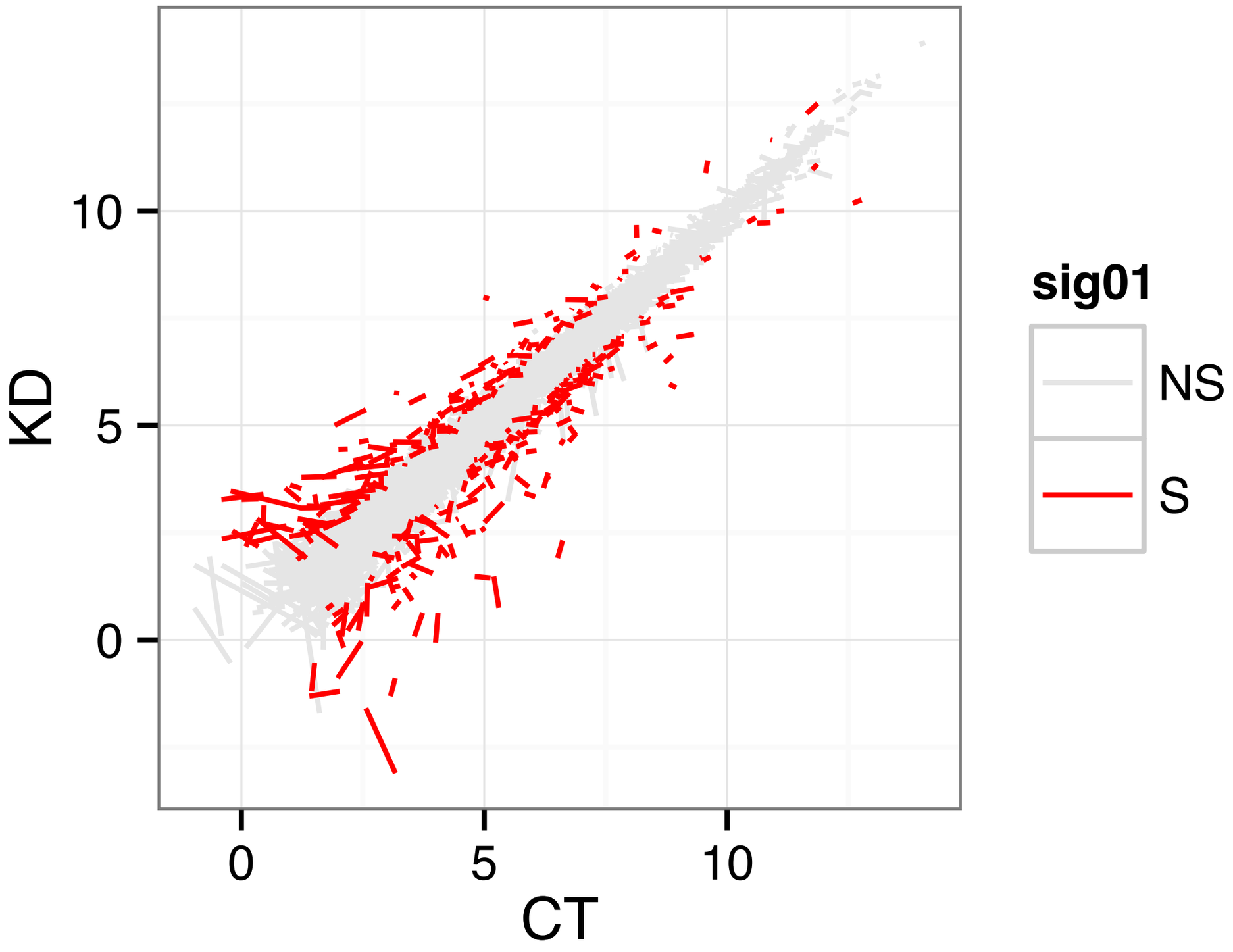
Significance testing

```
> # Test for differential expression (‘classic’ edgeR)
> de = exactTest(d, pair = c("CTL","KD"))
> tt = topTags(de, n = nrow(d), sort.by="none")
> nc.sig <- data.frame(gene=rownames(nc), nc, tt)
> nc.sig$sig05 <- ifelse(nc.sig$FDR < 0.05, "S", "NS")
> nc.sig$sig01 <- ifelse(nc.sig$FDR < 0.01, "S", “NS")
> ggplot(nc.sig) + geom_segment(aes(x=CT.PA.1,
 xend=CT.PA.2, y=KD.PA.3, yend=KD.PA.4, color=sig05)) +
 scale_color_manual(values=c("S"="red", "NS"="grey90")) +
 xlab("CT") + ylab("KD") +
 theme_bw() + theme(aspect.ratio=1)
```

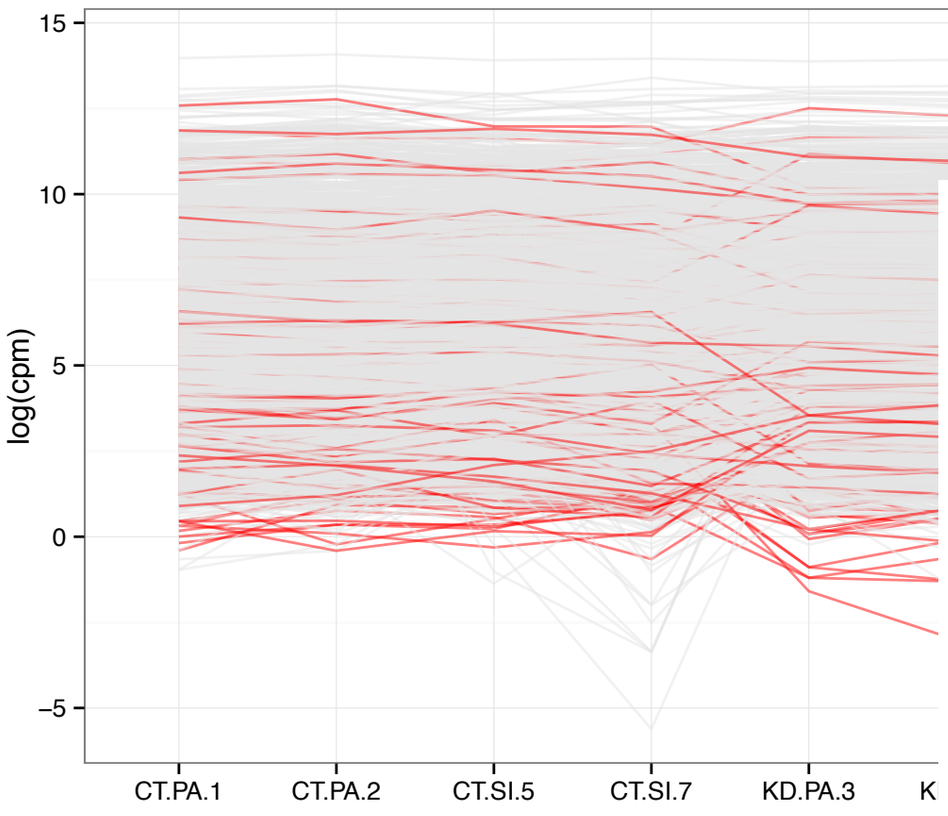
FDR 0.05

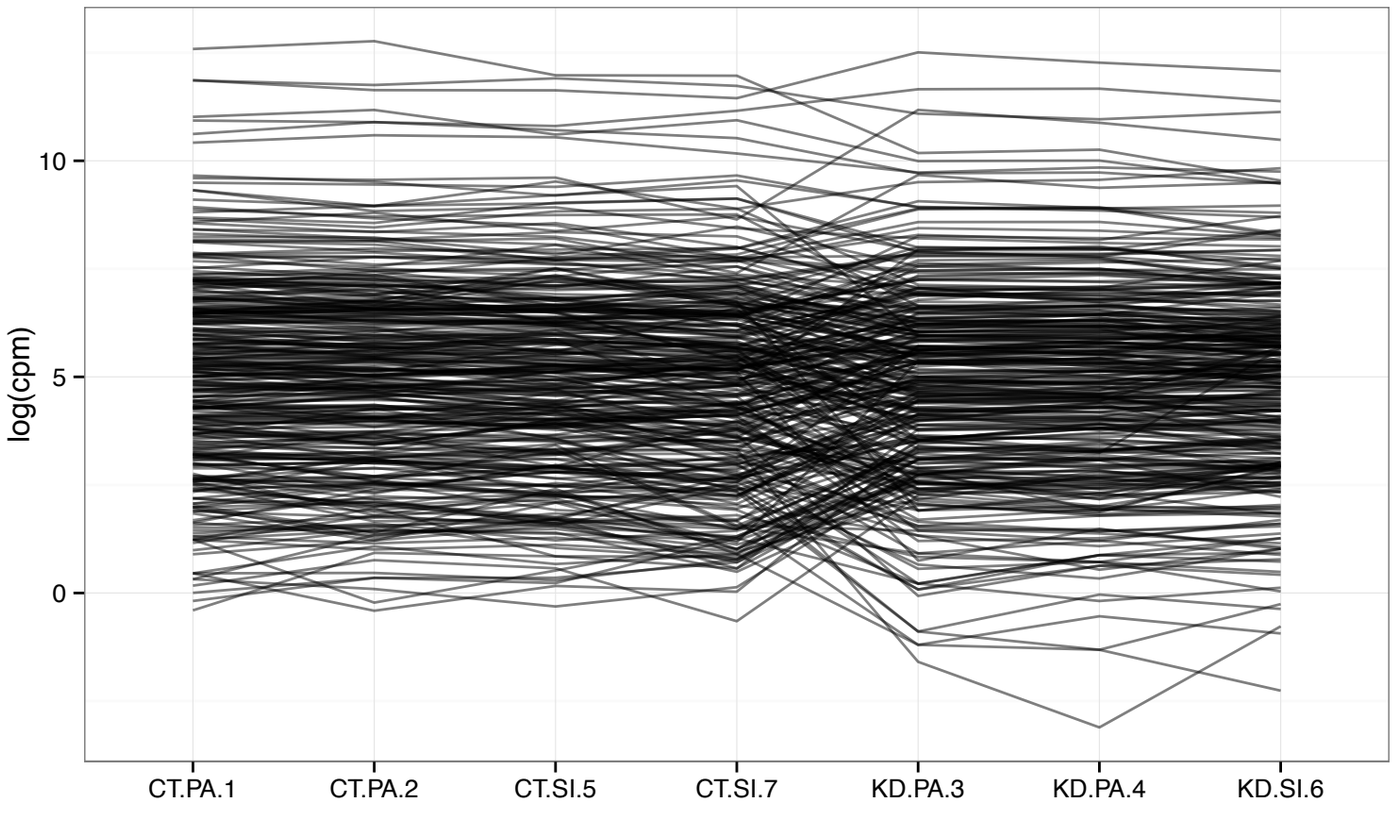


FDR 0.01

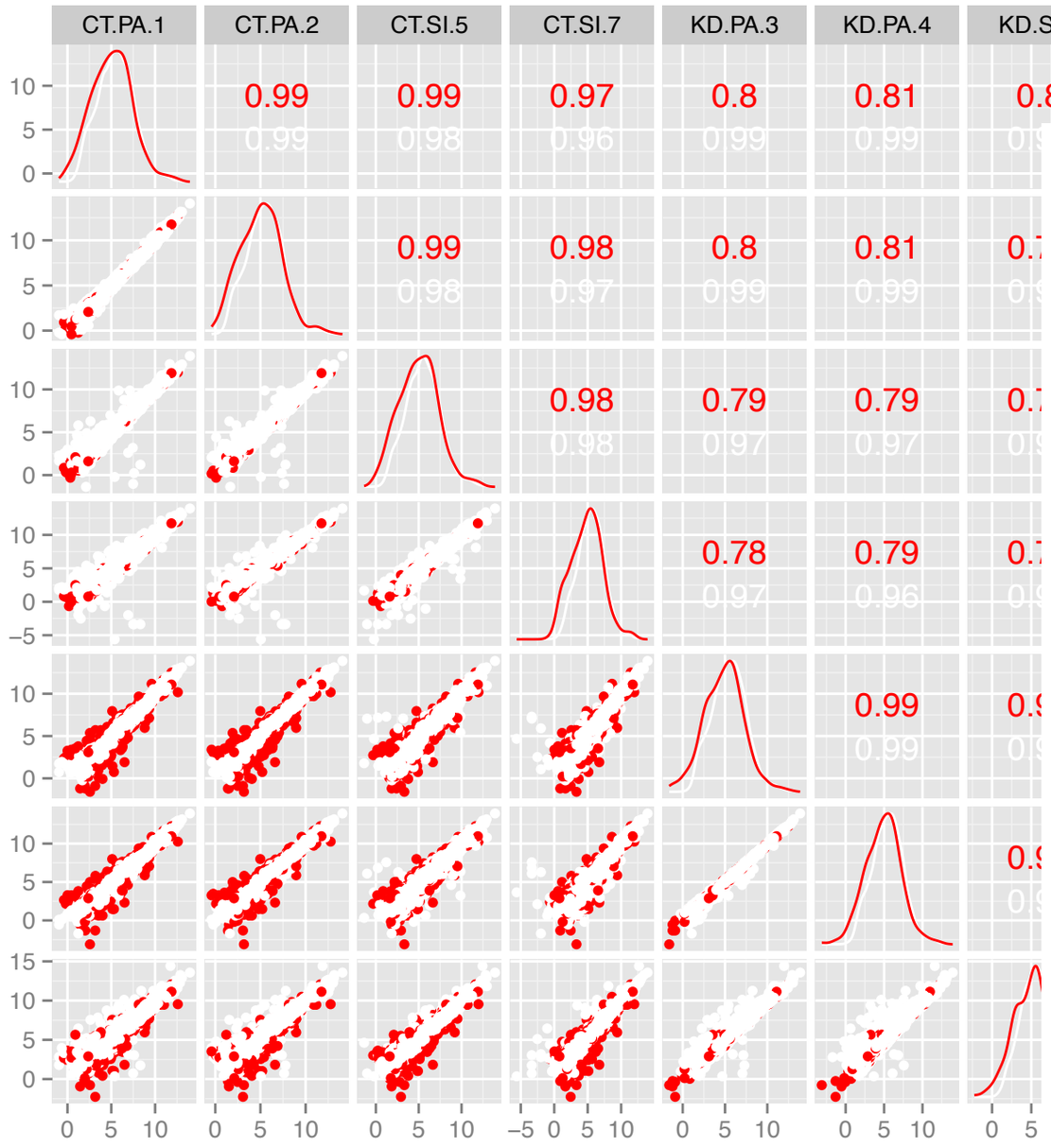


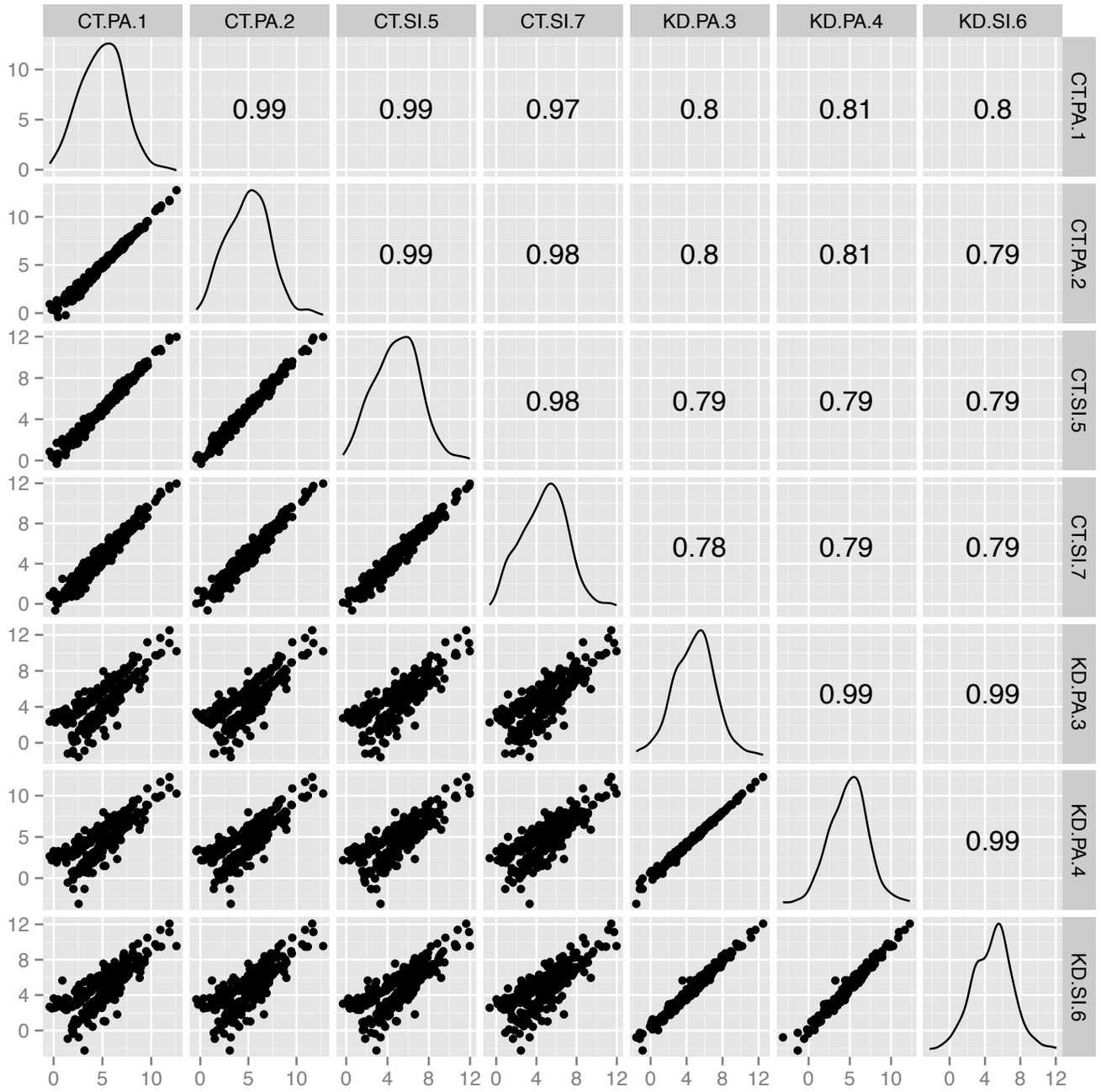
```
> ggparcoord(nc.sig, columns=2:8, scale="globalminmax",
   alphaLines=0.5, groupColumn="sig01") +
   scale_color_manual(values=c("S"="red", "NS"="grey90")) +
  xlab("") + ylab("log(cpm)") + theme_bw()
> ggparcoord(nc.sig[nc.sig$sig01=="S",], columns=2:8,
   scale="globalminmax", alphaLines=0.5) +
   xlab("") + ylab("log(cpm)") +
   theme_bw()
```





```
> ggscatmat(nc.sig, columns=2:8, color="sig01") +
+   scale_color_manual(values=c("S"="red", "NS"="white"))
> ggscatmat(nc.sig[nc.sig$sig01=="S",], columns=2:8)
```





Your Turn

- For the katz mouse data:
- Conduct the significance testing.
- Make a porcupine plot, scatterplot matrix and parallel coordinate plot, with the genes having FDR<0.01 colored differently from the rest.

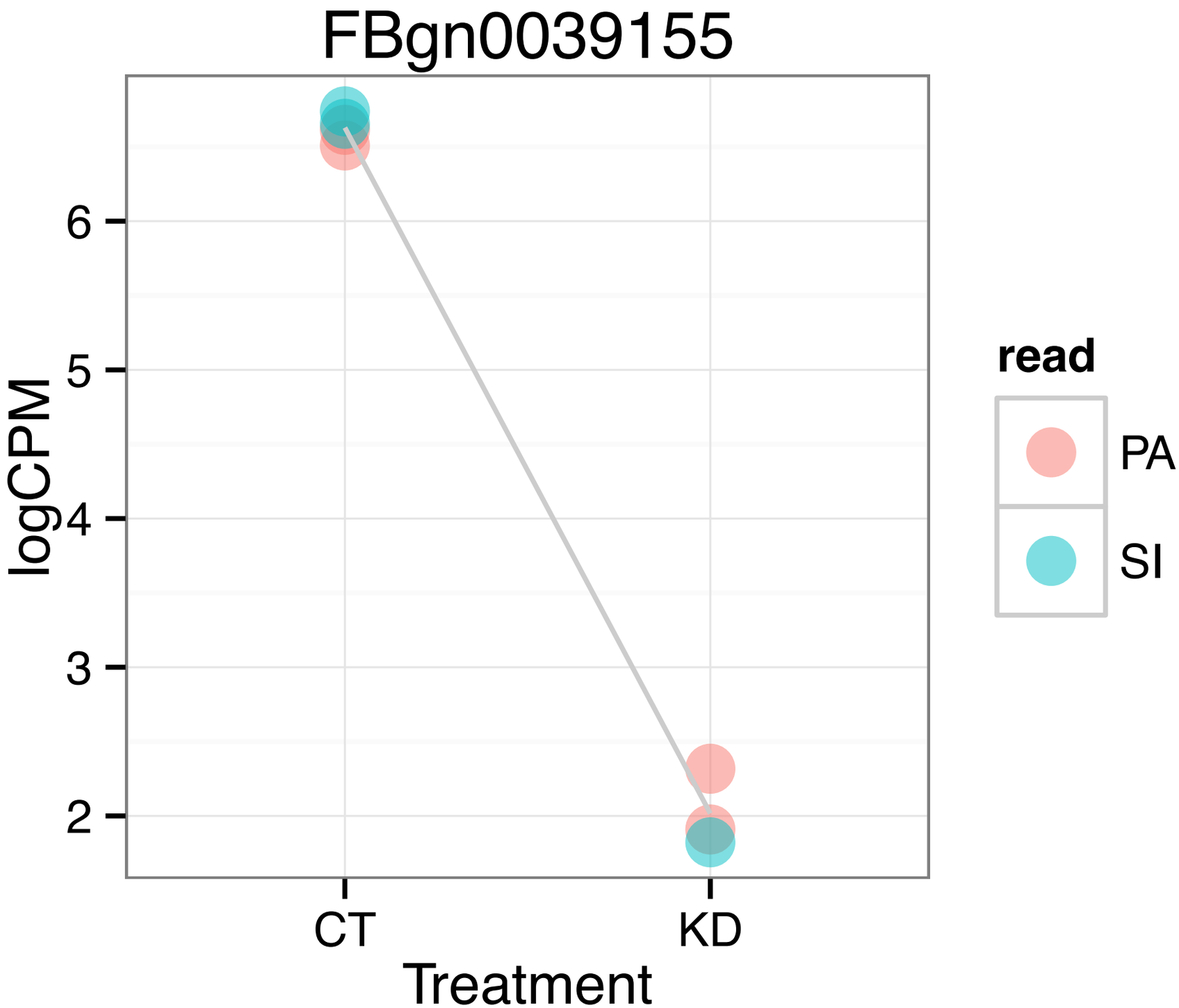
Top genes

- Classical experimental design plot is the interaction plot
- Display treatments horizontally, and response vertically
- Connect treatment means with a line

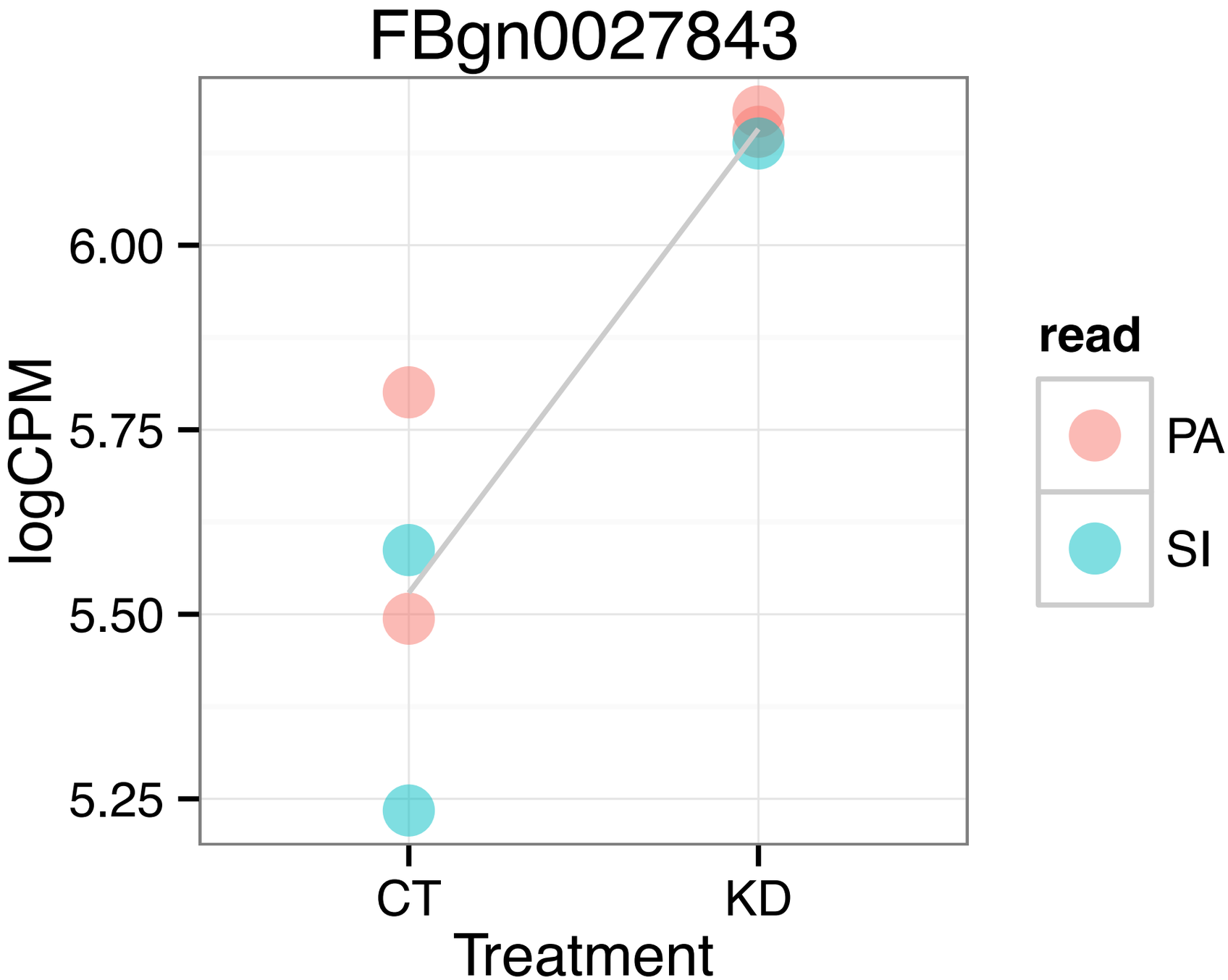
Interaction plot

```
> library(tidyr)
> library(dplyr)
> nc.sig <- arrange(nc.sig, PValue)
> g1 <- gather(nc.sig[1,2:8])
> g1$trt <- substr(g1$key, 1, 2)
> g1$read <- substr(g1$key, 4, 5)
> g1.mean <- summarise(group_by(g1, trt), m=mean(value))
> qplot(trt, value, data=g1, xlab="Treatment",
    ylab="logCPM", colour=read,
    size=I(5), alpha=I(0.5)) +
  annotate("segment", x=1, xend=2, y=g1.mean$m[1],
           yend=g1.mean$m[2], colour="grey80") +
  ggtitle(nc.sig$gene[1]) +
  theme_bw() + theme(aspect.ratio=1)
```

Most significantly expressed gene



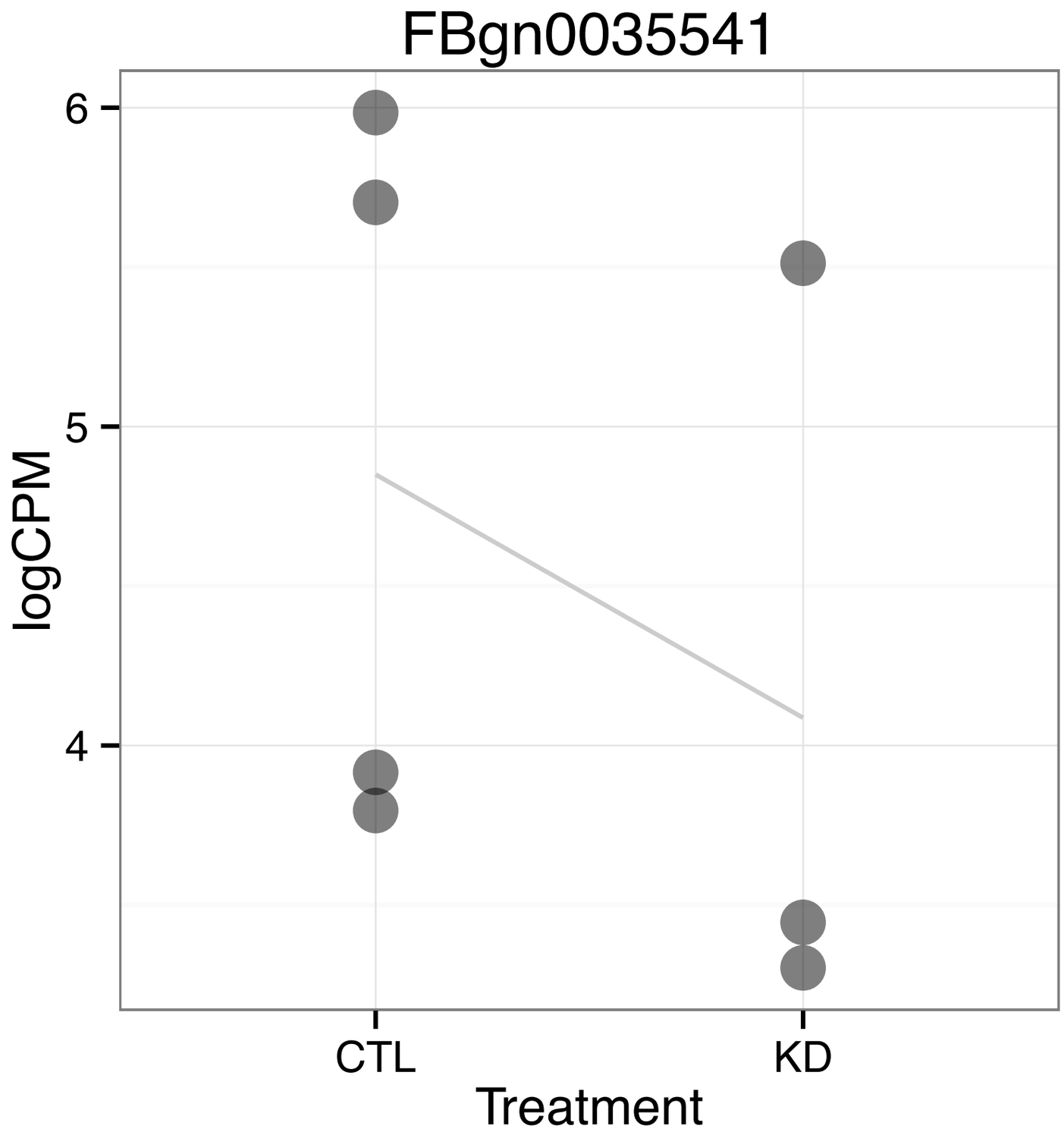
314th most significantly expressed gene (at the cutoff)



Sanity check

- Try re-running the significance testing using “scrambled” experimental design
- How does the top gene look compared to the top gene for the true design?

```
> dp <- d
> ncp <- nc
> dp$samples$group <-
c("CTL","KD","CTL","KD","CTL","KD","CTL")
> dep = exactTest(dp, pair = c("CTL","KD"))
```


FBgn0035541
6
5
4
logCPM
CTL
KD
Treatment

Your Turn

- For the katz mouse data:
- Make an interaction plot of the top gene.