Re-Rewritten2017 Analysis of RNAseq at different RpoS levels

Madison Hobbs

5/25/2017

Introductory Remarks

This analysis is rewrite of a rewritten RNAseq analysis by Dr. Dan Stoebel and Garrett Wong. The methods they employed are reproduced using DESeq2 and the code now implements the tidyverse.

This analysis uses three RpoS levels, 0% (knockout), 26% (low), and 100% (wild type). The 26% is the estimate from experiments that had 5 biological replicates, with one technical replicate per sample.

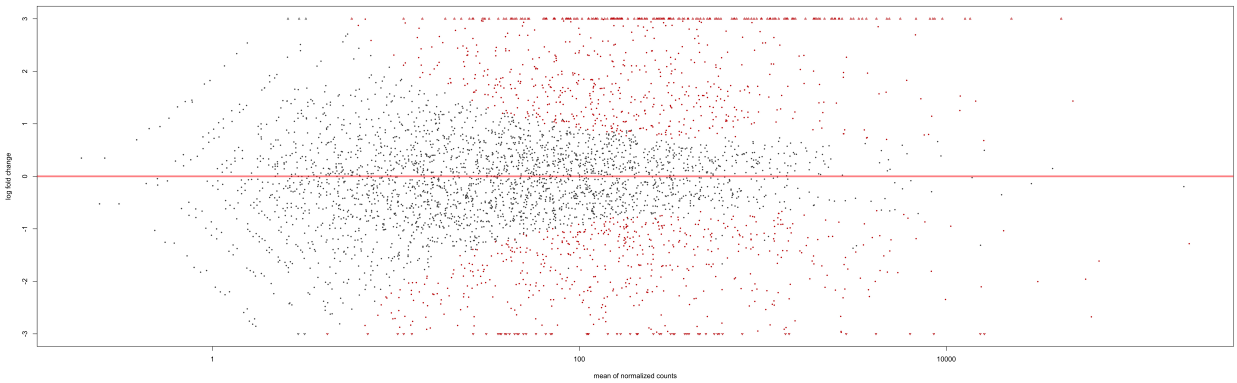
The original data actually had four levels, but Dr. Stoebel says, "I don't trust the fourth level which was greater than 100% of wild type. I don't trust it because the RNAseq samples were pooled across two different runs, which makes me nervous. Therefore, I'm going to drop those samples." Thus, although there exists counts for 4 conditions: ko ("knockout," 0% RpoS), lo ("low," 26% RpoS), wt ("wildtype," 100% RpoS), and hi (>100% Rpos), we will use only ko, lo, and wt in the analysis.

We start by making the count tables for the analysis. We make four count tables, one with all four levels (countsTable.4Cond), one with only knockout and wildtype (countsTable.0.100), one with only the low and wildtype conditions (countsTable.26.100), and one with only the knockout and low conditions (countsTable.0.26). This will allow us to compare conditions pairwise.

We now conduct the analysis on three pairwise comparisons: 0 vs. 100 (knockout vs. wildtype), 26 vs. 100 (low vs. wildtype), and 0 vs. 26 (knockout vs. low). This involves creating a count data set for just those samples, estimating size factors and dispersions, and then performing the negative binomal test. We then save all of the p-values into one large table.

Knockout (0%) vs. WildType (100%) Analysis

```
##
## out of 4343 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)     : 748, 17%
## LFC < 0 (down)   : 602, 14%
## outliers [1]     : 0, 0%
## low counts [2]   : 589, 14%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```




```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MAP): condition.0.100 wt vs ko"
## [3] "standard error: condition.0.100 wt vs ko"
## [4] "Wald statistic: condition.0.100 wt vs ko"
## [5] "Wald test p-value: condition.0.100 wt vs ko"
## [6] "BH adjusted p-values"
```

```
## [1] "Intercept"         "condition.0.100ko" "condition.0.100wt"
```

```
##       ko1       ko2       wt1       wt2
## 1.0298836 0.8064243 1.2926184 0.9646548
```

```
## [1] 0.4861941
```

```
## [1] 0.2025257
```

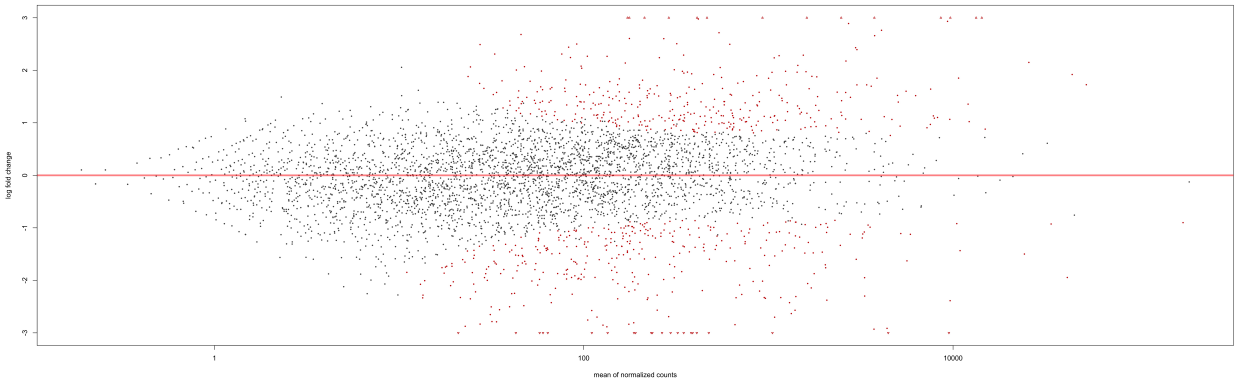
The plot shows the log2 fold changes 0% versus 100% and the mean of the normalized counts across both samples. Red points indicate an FDR-adjusted p-value of less than 0.05, and points falling outside the window get an open triange pointing up or down depending where that point is located. These log fold changes are already shrunk so that genes with low counts and high dispersion are shrunk more, and the effects of the low count and high dispersion is mitigated.

Of the 4343 (out of 4497) genes with nonzero total read count, 1350 are differentially expressed between kncokout and wildtype conditions. Of these, 748 (17% of total) are upregulated and 602 (14% of total). We identify no outliers, but we do identify many low mean counts (mean count < 4 for 589 genes, 14% of total). Low mean counts can give us higher dispersion, so this is something to watch out for, but we will use LFC shrinkage to mitigate this.

The size factors do not appear to vary wildly (range of 0.49 and standard deviation of 0.20). This hints that the sequencing depths across samples do not vary widely. ##

Low (26%) vs. WildType (100%) Analysis #

```
##
## out of 4349 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)     : 364, 8.4%
## LFC < 0 (down)   : 366, 8.4%
## outliers [1]     : 0, 0%
## low counts [2]   : 1094, 25%
## (mean count < 11)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```




```
##       lo1       lo2       wt1       wt2
## 1.1008943 0.7377879 1.3130997 0.9735136
```

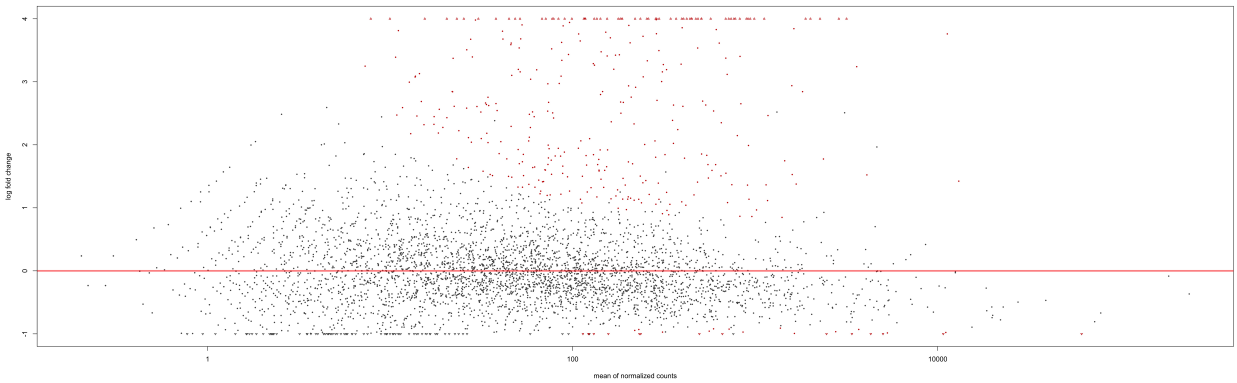
```
## [1] 0.240654
```

Of the 4349 (out of 4497) genes with nonzero total read count across the 26% and 100% conditions, 730 are differentially expressed between kncokout and wildtype conditions. Of these, 364 (8.4% of total) are upregulated and 366 (8.4% of total) are downregulated. We identify no outliers, but we do identify many low mean counts (mean count < 11 for 1094 genes, 25% of total). Low mean counts can give us higher dispersion, so this is something to watch out for, but we will use LFC shrinkage to mitigate this.

The size factors again do not vary by that much with a range of 0.575 and standard deviation of 0.241.

Knockout (0%) vs. Low (26%) Analysis

```
##
## out of 4348 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)     : 327, 7.5%
## LFC < 0 (down)   : 28, 0.64%
## outliers [1]     : 0, 0%
## low counts [2]   : 504, 12%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```




```
## [1] 0.4076419
```

```
## [1] 0.1897355
```

Of the 4348 (out of 4497) genes with nonzero total read count across the 0% and 26% conditions, 355 are differentially expressed between kncokout and wildtype conditions. Of these, 327 (7.5% of total) are upregulated and 28 (0.64% of total) are downregulated. We identify no outliers, but we do identify many low mean counts (mean count < 3 for 504 genes, 12% of total).

The size factors do not vary by much once again with a range of 0.408 and standard deviation of 0.190.

Now, we combine all of the FDR-adjusted p-values for each pairwise comparison into one table. Note that we use FDR-adjusted p-values to account for the additional likelihood of making a Type I error for each hypothesis test we conducted (and in differential expression analysis, there were a LOT of hypothesis tests!). We use FDR as opposed to Bonferroni because we have conducted so many hypothesis tests in the process of DESeq2 that using Bonferroni would be far too conservative, yielding almost no significant results.

In the previous code by Dr. Stoebel and Garrett, a Bonferoni adjustment was used to account for the three pairwise comparisons. However, the only one of those three comparisons with which we will move forward is the 0% vs 100%, used to determine our regulon. Therefore, Dr. Hardin, Dr. Stoebel and I have determined that no Bonferroni adjustment is necessary.

```
##  pvalFDR.Knockout.WildType pvalFDR.Low.WildType pvalFDR.Knockout.Low
##  Mode :logical             Mode :logical        Mode :logical
##  FALSE:2404                FALSE:2525           FALSE:3489
##  TRUE :1350                TRUE :730            TRUE :355
##  NA's :743                 NA's :1242           NA's :653
```

1350 genes were differentially expressed between Knockout and Wild-type condiitons (compare to 1044 previously), 570 genes were differentially expressed between Low and Wild-type conditions, and 299 genes were differentially expressed between Knocout and Low conditions.

Now, we determine into which of the three possibile significance categories each gene falls. The three possible significance categories are: gene is significant for one pairwise comparison, gene is significant for two pairwise comparisons, or gene is significant for three pairwise comparisons. We create a Venn Diagram reflecting these categories. Here, we will reintroduce the Bonferroni adjustment because we are making multiple comparisions.

```
##                   none             DE_ko_lo              DE_lo_wt
##                   1885                    9                    39
## DE_lo_wt and DE_ko_lo            DE_ko_wt DE_ko_wt and DE_ko_lo
##                     10                  461                   169
## DE_ko_wt and DE_lo_wt                  all                  NA's
##                    514                  167                  1243
```

The majority of genes are not differentially expressed across any comparison, which is expected (RpoS does

not regulate all genes in the E. coli genome). Of the genes that are differentially expressed across at least one comparison, the largest group is the genes which are differentially expressed across knockout and wild type which makes sense if, as we expect, regulation is more or less monotonic.

```
##   numGenesDEAcrossAll propGenesDEAcrossAll
## 1                 121           0.02690683
```

```
##   num.genes.DE.across.any prop.genes.DE.across.any
## 1                    1408                0.3130976
```

We find that about 31.3% of genes (1408 genes) were differentially expressed across at least one of the three pairwise comparisons of RpoS levels.

After exploring, we now wish to know which genes are regulated by RpoS and the direction of their regulation. The logic in carrying out this procedure relies on the assumption that regulation is more or less monotonic. We will consider the genes which showed differential expression between knockout and wildtype to be regulated by RpoS (known as the "RpoS-regulon"" or heretofore "regulon").

```
##   upregulated downreguated
## 1         748          602
```

Of the 1350 genes found to be differentially expressed across the knockout and wildtype conditions, we find that 748 genes are upregulated, while 602 genes are downregulated as the RpoS level changes from 0% (knockout) to 100% (wild-type).

Creating data frames with the gene names included, and comparing to Garrett's results (which genes were DE across the different pairwise comparisons).

```
##   FDR-adj-pval.KO.WT isDE.KO.WT   geneid genename bnum isNonUnique
## 1       1.899290e-07       TRUE  gene670     <NA> <NA>          NA
## 2       2.705308e-02       TRUE gene4359     <NA> <NA>          NA
```

Comparison with prior findings

```
## [1] 1041
```

I found all all but three of the 1044 genes Garrett had found in the regulon.

```
## [1] 309
```

There is one gene, gene670, which seems to not have a name match in nameTable. There are also 46 genes in the allCounts_new which are not accounted for in the nameMapping. This is something we could investigate further in the NC_000913.gff.

Garrett found 3 genes DE that I did not find, and I found 309 genes DE that he did not find. However, 1041 out of the 1350 genes (77.1%) which I found to be DE, Garrett also found. This could be due to the new technology of DESeq2 and the fact that I am not Bonferroni adjusting where he did. Below are the p-values for the three genes Garrett found DE that I did not. They are not low, but they are not extremely high either.

```
##   pvalFDR.Knockout.WildType   geneid
## 1                 0.2671826  gene199
## 2                 0.2229423 gene3177
## 3                 0.2575188 gene3933
```

Low versus WildType

```
##   pvalFDR.Low.WildType isDE.LO.WT  geneid genename bnum isNonUnique
## 1         0.0001839027       TRUE gene670     <NA> <NA>          NA
```

Again, gene670 is missing from the nameTable.

Now, we repeat the comparison to Garrett's findings with the 0% vs 26% comparison.

```
## [1] 368
```

```
## [1] 730
```

```
## [1] 368
```

```
## [1] 0
```

```
## [1] 362
```

```
## [1] pvalFDR.Low.WildType geneid
## <0 rows> (or 0-length row.names)
```

There are no genes which Garrett found significant 0% vs 26% that I did not (all 368 of his genes match mine). I found an additional 362 significant, however.

Finally, we compare my and Garrett's results for the 0% vs. 26% comparison.

```
## [1] 355
```

```
## [1] 355
```

```
## [1] 355
```

We (DESeq2) find 355 genes are differentially expressed between the Knockout and Low conditions. At this time, I do not have data on what Garrett found to be differentially expressed previously with DESeq. All of the genes found to be significant across these two categories are successfully found in the nameTable.

GENE EXPRESSION SHAPES & SENSITIVITY

The goal of this section is to identify whether each differentially expressed gene is "sensitive," "insensitive," or "neither."

In the previous analysis, we investigated if there any intermediate optima (where genes are maximally or minimally expressed at the Low RpoS concentration?); Dr. Hardin and I decided to leave this part out for the moment because even if there are such genes (though recall, Garrett only found two such genes), we can still categorize them as sensitive, insensitive, or neither.

```
##   numLowIsMin
## 1          44
```

```
##   numLowIsMax
## 1         120
```

Sensitivity: building the model

Null hypothesis: Expression is linear with increasing levels of RpoS.

Our expected mean count ("mu") for the "low" level of RpoS is interpolated from a line drawn between the means of the Knockout and Wildtype conditions after normalizing by the size factor (note: this is why, in the test itself, we multiply "mu" by the size factor for the intermediate condition generated above).

Sensitivity Testing

We now test whether the actual observed counts for each gene significantly indicate that the gene has a relationship with RpoS which may be described as sensitive, insensitive, or neither. We implement the negative binomial probability density function for the hypothesis tests.

Contingency Tables for Sensitivity and Positive/Negative Regulation by RpoS

We will divide the genes into the following six groups: 1) insensitive, positively regulated (isInsensPos) 2) insensitive, negatively regulated (isInsensNeg) 3) Neither, positively regulated (isNeitherPos) 4) Neither, negatively regulated (isNeitherNeg) 5) sensitive, positively regulated (isSensPos) 6) sensitive, negatively regulated (isSensNeg)

Groups 1 and 2 : Insensitive

```
## # A tibble: 2 x 3
##   positiveSlope numGenes   percent
##           <lgl>    <int>     <dbl>
## 1         FALSE        5 0.1470588
## 2          TRUE       29 0.8529412
```

My results suggest that 29 genes (85.3%) are insensitive and upregulated by RpoS while the other 5 (14.7%) are downregulated, totalling 34.

For comparison, Garrett identified 32 insensitive genes. I do not access to any more specific data at this time.

Groups 3 and 4 : Neither

```
## # A tibble: 2 x 3
##   positiveSlope numGenes  percent
##           <lgl>    <int>    <dbl>
## 1         FALSE      589 0.492887
## 2          TRUE      606 0.507113
```

```
anti_join(old.linear.pos, isNeitherPos) %>% select(geneID, geneName, bNum) # which genes did Garrett fi
```

```
## Joining, by = "genename"
```

```
## Warning: Column `genename` joining factors with different levels, coercing
## to character vector
```

```
##      geneID geneName  bNum
## 1  gene3595     bcsC b3530
## 2  gene3569      slp b3506
## 3  gene3375     rpsC b3314
## 4  gene3678     mtlR b3601
## 5  gene3082     ygiW b3024
## 6  gene2720     gabD b2661
## 7  gene1999     yodD b1953
## 8  gene1790     astC b1748
## 9  gene1462     ydcK b1428
## 10 gene1065     msyB b1051
## 11  gene868     potF b0854
## 12 gene3587     yhjD b3522
## 13 gene3622     yiaG b3555
## 14  gene343     prpD b0334
## 15 gene1021     wrbA b1004
```

```
length(old.linear.pos$geneID)+length(old.linear.neg$geneID)
```

```
## [1] 910
```

My results suggest suggest that out of the 1195 genes which can neither be categorized as sensitive nor insensitive, 606 genes (50.7%) are upregulated by RpoS while 589 (49.3%) are downregulated by RpoS.

For comparison, Garrett identified 910 "neither" genes, 53% of which were postively regulated and 47% of which were negatively regulated; our results are very close. He found

Groups 5 and 6 : Sensitive

```
## # A tibble: 2 x 3
##   positiveSlope numGenes   percent
##           <lgl>    <int>     <dbl>
## 1         FALSE        8 0.0661157
## 2          TRUE      113 0.9338843
```

My results suggest that 113 genes (93.3%) are sensitive and upregulated by RpoS while 8 (6.6%) are downregulated, totalling 121.

Compare to Garrett's results: Sensitive Genes.

```
## # A tibble: 2 x 3
##   positiveSlope numGenes percent
##           <lgl>    <int>   <dbl>
## 1         FALSE        3   3.125
## 2          TRUE       93  96.875
```

For the genes we each deemed "sensitive," 96/121 (79.3%) of the genes I found match up with Garrett's (96/102 = 94.1% of the genes Garrett found match up with mine). More specifically, Garrett found 93 of the 113 (78.8%) sensitive genes which I found to be upregulated and 3/8 (37.5%) which I found to be downregulated.

```
## # A tibble: 2 x 3
##   positiveSlope numGenes   percent
##           <lgl>    <int>     <dbl>
## 1         FALSE        4  3.921569
## 2          TRUE       98 96.078431
```

For comparison, we note that Garrett found 98/102 (96.1%) sensitive genes upregulated and 4/102 (3.9%) sensitive genes downregulated. Recall that my results suggest that 113 genes (93.3%) are sensitive and upregulated by RpoS while 8 (6.6%) are downregulated.

GROUP GENES BY POSITIVE/NEGATIVE REGULATION

```
## # A tibble: 3 x 2
##   Sensitivity `n()`
##         <chr> <int>
## 1 Insensitive    34
## 2     Neither  1195
## 3   Sensitive   121
```

```
## # A tibble: 2 x 3
##   positiveSlope numGenes   percent
##           <lgl>    <int>     <dbl>
## 1         FALSE      602 0.4459259
## 2          TRUE      748 0.5540741
```

We find that among the 1350 genes regulated by RpoS (the genes we found to be differentially expressed between the Knockout and Wildtype conditions), 748 (55.4%) are upregulated and 602 (44.6% are downregulated).

Comparing this to Garrett's findings, of the 1044 genes regulated by RpoS that he found, 605 (58%) were upregulated and 439 (42%) were downregulated.

A Difference in Upregulated and Downregulated Genes in Each of the Sensitive, Neither, and Insensitive Groups

And yet, the vast majority of both sensitive and insensitive genes are positively regulated by RpoS. Is this tendency for positive regulation greater than what we would expect by chance?

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  table(regulatedGenes$positiveSlope, regulatedGenes$Sensitivity)
## X-squared = 93.604, df = NA, p-value = 0.0004998
```

Yes! There is a significant difference in positively regulated genes between the three groups ($P < 0.0005$). This is consistent with previous findings (significant at $P < 0.001$).

```
## # A tibble: 6 x 2
##   DirectionalSensitivity numGenes
##                    <chr>    <int>
## 1            isInsensNeg        5
## 2            isInsensPos       29
## 3           isNeitherNeg      589
## 4           isNeitherPos      606
## 5              isSensNeg        8
## 6              isSensPos      113
```

Of the RpoS-regulated genes (DE between Knockout and Wildtype conditions), the above table breaks down how many of these were differentially expressed between the Knockout and Low conditions and how their directional sensitivity may be categorized.

Plots

Plotting one Sensitive Gene, 3-way normalized count against RpoS level

First get the data we use DESeq to normalize across all conditions.

```
cds.3Cond <- DESeq(cds.3Cond)
```

```
## using pre-existing size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
results.3Cond <- results(cds.3Cond, alpha = 0.05)
# note: the only p-values results.3Cond condtains are the contrasts between KO and WT. I think DESeq2's
# however, one could break up the results into different pairwise comparisons.
# break cds.3Cond into the 3 pairwise comparisons to get the p-values for differential expression for e
```

```
# I decided not to do this up top for the three pairwise comparisions because I didn't want the normali
res.KO.WT <- results(cds.3Cond, contrast = c("condition.3Cond", "ko", "wt"), alpha = 0.05)
mcols(res.KO.WT)$description
```

```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MAP): condition.3Cond ko vs wt"
## [3] "standard error: condition.3Cond ko vs wt"
## [4] "Wald statistic: condition.3Cond ko vs wt"
## [5] "Wald test p-value: condition.3Cond ko vs wt"
## [6] "BH adjusted p-values"
```

```
res.LO.WT <- results(cds.3Cond, contrast = c("condition.3Cond", "lo", "wt"), alpha = 0.05)
mcols(res.LO.WT)$description
```

```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MAP): condition.3Cond lo vs wt"
## [3] "standard error: condition.3Cond lo vs wt"
## [4] "Wald statistic: condition.3Cond lo vs wt"
## [5] "Wald test p-value: condition.3Cond lo vs wt"
## [6] "BH adjusted p-values"
```

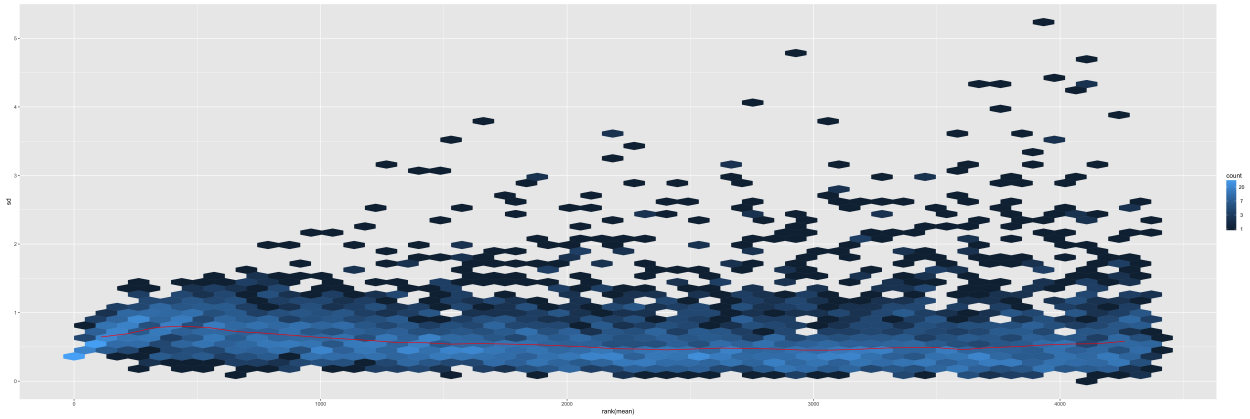
```
res.KO.LO <- results(cds.3Cond, contrast = c("condition.3Cond", "ko", "lo"), alpha = 0.05)
mcols(res.KO.LO)$description
```

```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MAP): condition.3Cond ko vs lo"
## [3] "standard error: condition.3Cond ko vs lo"
## [4] "Wald statistic: condition.3Cond ko vs lo"
## [5] "Wald test p-value: condition.3Cond ko vs lo"
## [6] "BH adjusted p-values"
```

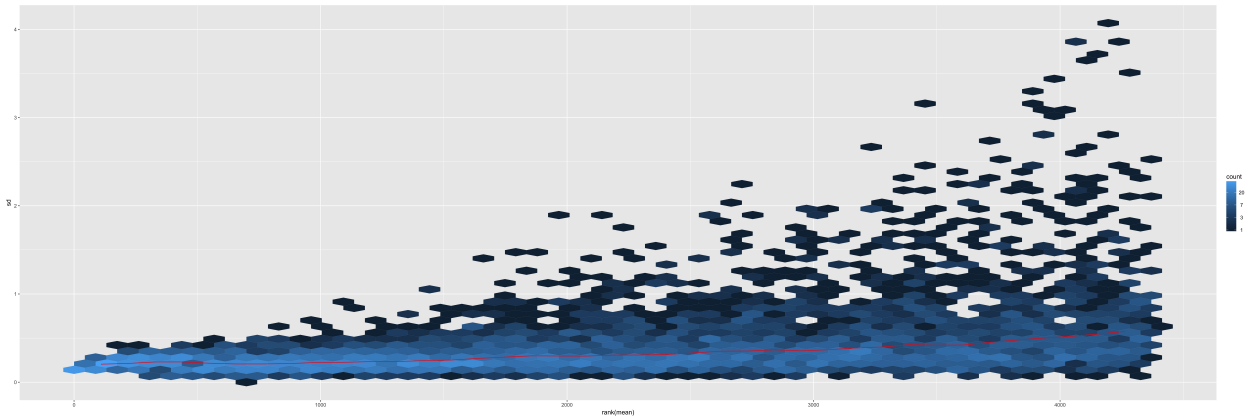
```
summary(res.KO.LO)
```

```
##
## out of 4371 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)     : 36, 0.82%
## LFC < 0 (down)   : 356, 8.1%
## outliers [1]     : 0, 0%
## low counts [2]   : 1100, 25%
## (mean count < 11)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# TRANSFORM THE COUNTS DATA
# Explore some transformations, and read up on transformations in the DESeq2 Vignette
# see below for log2 pseudo count
# rlog
rlog.cds.3Cond <- rlog(cds.3Cond, blind = FALSE)
# variance stabilising transformation
vst.cds.3Cond <- varianceStabilizingTransformation(cds.3Cond, blind=FALSE)

# Compare rank(mean) to standard deviation at different counts
notAllZero <- (rowSums(counts(cds.3Cond))>0)
# pseudocount transformation
meanSdPlot(log2(counts(cds.3Cond,normalized=TRUE)[notAllZero,] + 1))
```



```
# variance stabilising transformation
meanSdPlot(assay(vst.cds.3Cond[notAllZero,]))
```



```
# now, you could try the results with either of these transformations and see if we get different plots

# by how much do our size factors vary?
max(sizeFactors(cds.3Cond)) - min(sizeFactors(cds.3Cond))
```

```
## [1] 0.568937
```

```
sd(sizeFactors(cds.3Cond))
```

```
## [1] 0.2123739
```

```
# not by much :)
```

Make the table of normalized counts

```
rowNames <- rownames(countsTable.4Cond)
# get three-way normalized counts comparison
normCounts.3Cond <- as.data.frame(counts(cds.3Cond, normalized = TRUE)) %>%
                      mutate(geneid = rowNames)

regulatedGenes.normCount <- left_join(regulatedGenes, normCounts.3Cond, by = "geneid") %>%
                    mutate(normCountKO1 = ko1, normCountKO2 = ko2,
                           normCountLO1 = lo1, normCountLO2 = lo2,
                           normCountWT1 = wt1, normCountWT2 = wt2)
```

regulated.normCount.tidy

```
# data frame of regulated genes with the variables
# genename, Sensitivity (sensitive, insensitive, or neither), DirectionalSensitivity (specifies positiv
regulated.normCount.tidy <- regulatedGenes.normCount %>%
                select(genename, normCountKO1, normCountKO2,
                       normCountLO1, normCountLO2, normCountWT1,
                       normCountWT2, Sensitivity, DirectionalSensitivity)  %>%
                # key: , value: normCount
                gather(level, normCount, -genename, -Sensitivity, -DirectionalSensitivity) %>%
                separate(level, c("level", "sample"), 11) %>%
                arrange(genename) %>%
                mutate(level = replace(level, level == "normCountKO",
                                       "KO"),
                       level = replace(level, level == "normCountLO",
                                       "LO"),
                       level = replace(level, level == "normCountWT",
                                       "WT")) %>%
                mutate(nlevel = ifelse(level == "KO", 0,
                                       ifelse(level == "LO", 0.26, 1)),
                       nlevel = as.numeric(nlevel)) %>%
                mutate(logNormCount = log(normCount)) %>%
                group_by(genename, level) %>%
                    mutate(levelMean = mean(normCount),
                           levelMeanLog = mean(logNormCount)) %>%
                ungroup() %>%
                group_by(genename) %>%
mutate(scaler = ifelse(DirectionalSensitivity %in% c("isNeitherPos", "isSensPos", "isInsensPos") , level
      logScaler = ifelse(DirectionalSensitivity %in% c("isNeitherPos", "isSensPos", "isInsensPos") , le
      meanScaled = levelMean/scaler,
      normCountScaled = normCount/scaler,
      maxNormCount = max(normCount), # maximum normalized count
      levelPropOverMaxCount = levelMean/maxNormCount, # scale level Mean by maximum normalized count fo
      expressionProportionOverMaxCount = normCount/maxNormCount,
    ### LOGGED NORMCOUNTS ###
      meanScaledLog = levelMeanLog/logScaler,
      logNormCountScaled = logNormCount/logScaler,
      maxLevelMeanLog = max(levelMeanLog), # maximum among means of logged data across each level for e
      levelPropOverMaxMeanLog = levelMeanLog/maxLevelMeanLog, # scale levelMeanLog by maximum levelMean
      expressionProportionOverMaxMeanLog = logNormCount/maxLevelMeanLog, # scale logNormCounts by maxim
      maxLogNormCount = max(logNormCount), # maximum logged normalized count
      levelPropOverMaxCountLog = levelMeanLog/maxLogNormCount, # scale level Mean by maximum normalized
      expressionProportionOverMaxCountLog = logNormCount/maxLogNormCount) %>% # scale logged normCounts
      ungroup()
```

data frames to use going off of regulated.tidy

```
# the subset of regulated.tidy containing sensitive genes
regulated.sens.tidy <- regulated.normCount.tidy %>%
                          filter(Sensitivity == "Sensitive")
regulated.sens.pos.tidy <- regulated.normCount.tidy %>%
                            filter(DirectionalSensitivity == "isSensPos")
regulated.sens.neg.tidy <- regulated.normCount.tidy %>%
                            filter(DirectionalSensitivity == "isSensNeg")

# the subset of regulated.tidy containing insensitive genes
```

```
regulated.insens.tidy <- regulated.normCount.tidy %>%
                         filter(Sensitivity == "Insensitive")
regulated.insens.pos.tidy <- regulated.normCount.tidy %>%
                           filter(DirectionalSensitivity == "isInsensPos")
regulated.insens.neg.tidy <- regulated.normCount.tidy %>%
                           filter(DirectionalSensitivity == "isInsensNeg")

# the subset of regulated.tidy containing "neither" genes
regulated.neither.tidy <- regulated.normCount.tidy %>%
                         filter(Sensitivity == "Neither")
regulated.neither.pos.tidy <- regulated.normCount.tidy %>%
                           filter(DirectionalSensitivity == "isNeitherPos")
regulated.neither.neg.tidy <- regulated.normCount.tidy %>%
                           filter(DirectionalSensitivity == "isNeitherNeg")

# the subset of regulated.tidy where the level is KO or WT
regulated.normCount.KO_WT.tidy <- regulated.normCount.tidy %>%
                         filter(level %in% c("KO", "WT"))
###
regulated.sens.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                             filter(Sensitivity == "Sensitive")
regulated.sens.pos.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                              filter(DirectionalSensitivity == "isSensPos")
regulated.sens.neg.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                             filter(DirectionalSensitivity == "isSensNeg")
###
regulated.insens.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                               filter(Sensitivity == "Insensitive")
regulated.insens.pos.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                             filter(DirectionalSensitivity == "isInsensPos")
regulated.insens.neg.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                             filter(DirectionalSensitivity == "isInsensNeg")
###
regulated.neither.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                                 filter(Sensitivity == "Neither")
regulated.neither.pos.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                         filter(DirectionalSensitivity == "isNeitherPos")
regulated.neither.neg.KO_WT.tidy <- regulated.normCount.KO_WT.tidy %>%
                         filter(DirectionalSensitivity == "isNeitherNeg")
```

Plotting: unLogged One gene by normCount

```
# we try a loess fit to show shape. However, we later decide that loess and splines are not the best sh
ggplot(data = head(regulated.normCount.tidy), aes(x = nlevel, y = normCountScaled, col = genename, grou
```

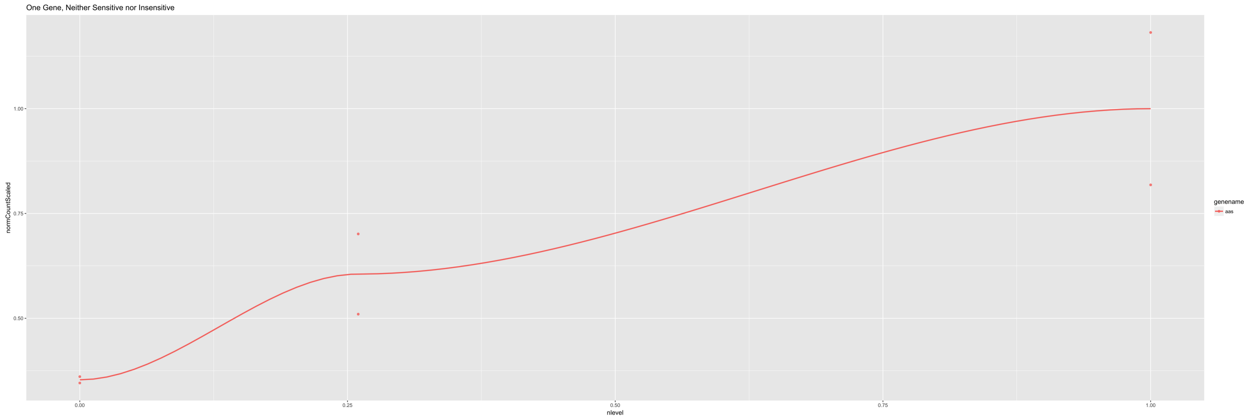
```
## `geom_smooth()` using method = 'loess'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```



4 sensitive genes : LOESS

```
ggplot(data = regulated.sens.tidy[1:24,], aes(x = nlevel, y = normCountScaled, col = genename, group = g
```

```
## `geom_smooth()` using method = 'loess'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005
```

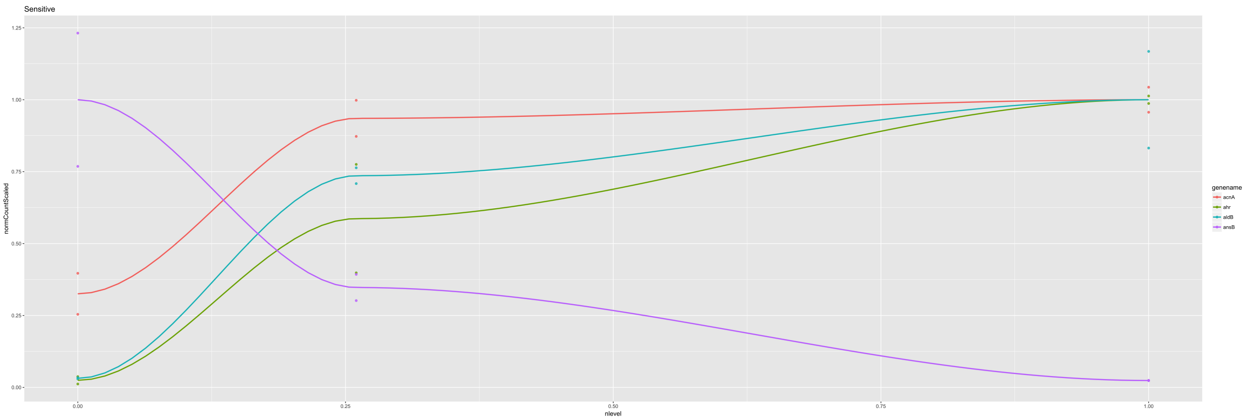
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005
```

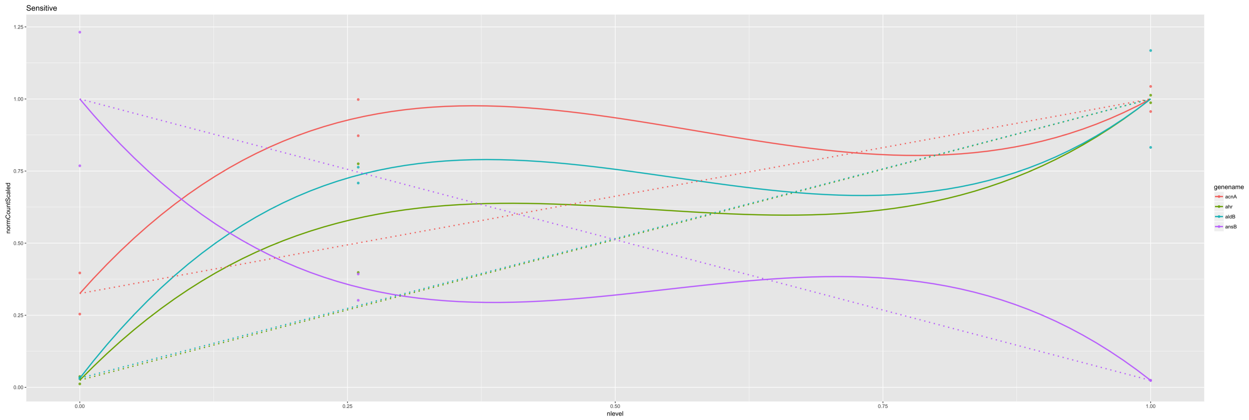
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```



4 sensitive genes : spline

```r
# the dotted lines show what we would expect gene expression to look like if the shape was linear
ggplot(data = regulated.sens.tidy[1:24,], aes(x = nlevel, y = normCountScaled, col = genename, group = g
```

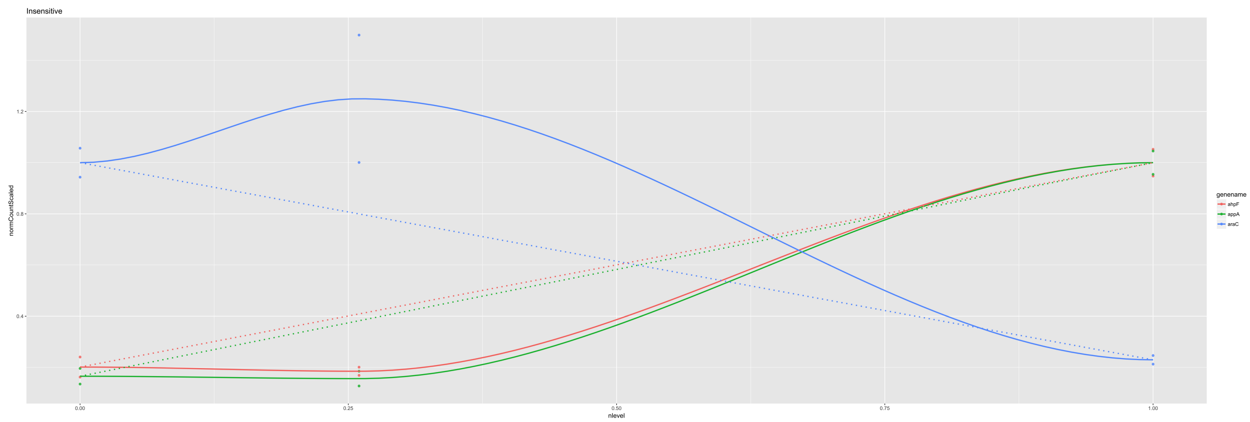
```
## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in predict.lm(model, newdata = data.frame(x = xseq), se.fit = se, :
## prediction from a rank-deficient fit may be misleading

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in predict.lm(model, newdata = data.frame(x = xseq), se.fit = se, :
## prediction from a rank-deficient fit may be misleading

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in predict.lm(model, newdata = data.frame(x = xseq), se.fit = se, :
## prediction from a rank-deficient fit may be misleading

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in splines::bs(x, 1): 'df' was too small; have used 3

## Warning in predict.lm(model, newdata = data.frame(x = xseq), se.fit = se, :
## prediction from a rank-deficient fit may be misleading
```



3 insensitive genes

```
ggplot(data = regulated.insens.tidy[1:18,], aes(x = nlevel, y = normCountScaled, col = genename, group =
```

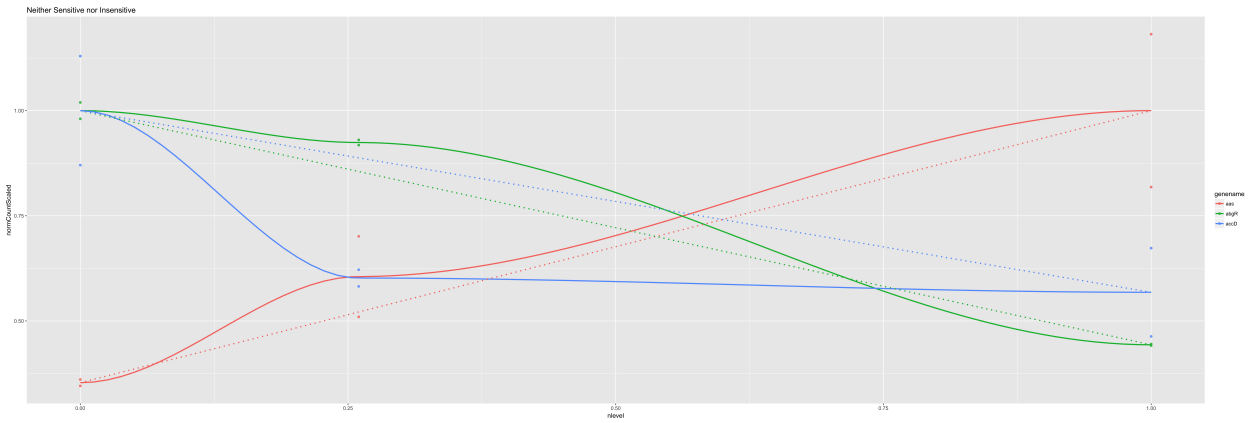
```
## `geom_smooth()` using method = 'loess'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```



3 neither genes

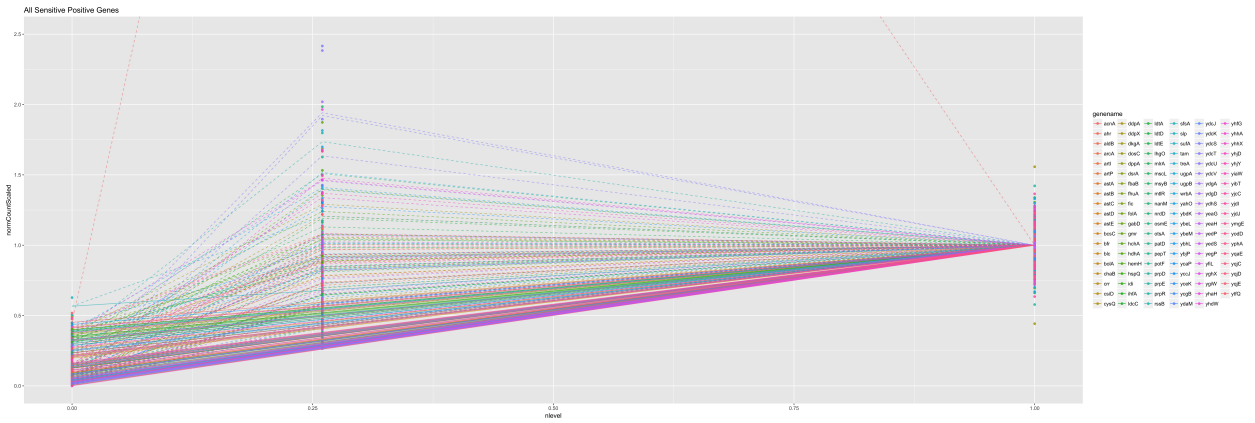
```
ggplot(data = regulated.neither.tidy[1:18,], aes(x = nlevel, y = normCountScaled, col = genename, group
```

```
## `geom_smooth()` using method = 'loess'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.265

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 0.55502
```



1. All Sensitive Positive Genes

```
ggplot(data = regulated.sens.pos.tidy, aes(x = nlevel, y = normCountScaled, col = genename, group = gene
```



Profile-gene correlation

Scaled normCounts

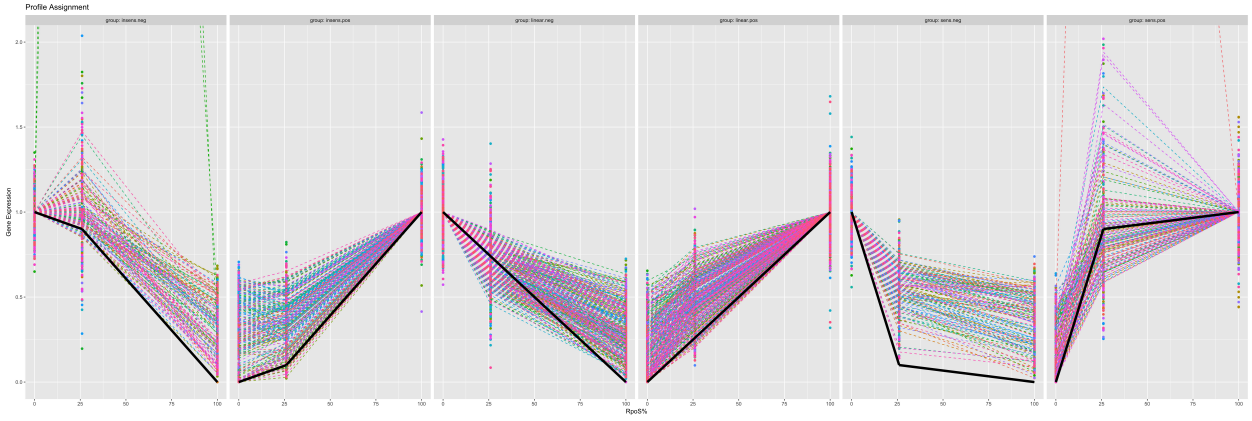
```
# Design profiles!
profile.linear.pos <- c(0, 0, .26, .26, 1, 1)
profile.linear.neg <- c(1, 1, 0.74, 0.74, 0.00, 0.00)
profile.sens.pos <- c(0, 0, 0.9, 0.9, 1, 1)
profile.sens.neg <- c(1, 1, 0.1, 0.1, 0, 0)
profile.insens.pos <- c(0, 0, 0.1, 0.1, 1, 1)
profile.insens.neg <- c(1, 1, 0.9, 0.9, 0, 0)

profile.6 <- data.frame(sens.pos = profile.sens.pos, sens.neg = profile.sens.neg, insens.pos = profile.

scaledNormCountsByGene.6 <- regulated.normCount.tidy %>%
  select(genename, level, sample, normCount) %>%
  unite(level.sample, level, sample) %>%
  filter(genename != "NA") %>%
  spread(genename, normCount) %>%
  select(-1)

# using normCounts (3 levels, 2 at each level)
```

```
profile.scaledNormCountsByGene.6 <- as.data.frame(t(cor(profile.6, scaledNormCountsByGene.6)))
rownames <- row.names(profile.scaledNormCountsByGene.6)
rowMax <- rowMax(as.matrix(profile.scaledNormCountsByGene.6))
profile.scaledNormCountsByGene.6 <- profile.scaledNormCountsByGene.6 %>%
  mutate(genename = rownames,
         maxCorr = rowMax, # find in which classification maximum correlation lives
         group = ifelse(maxCorr == linear.pos, "linear.pos",
                        ifelse(maxCorr == linear.neg, "linear.neg",
                               ifelse(maxCorr == insens.pos, "insens.pos",
                                      ifelse(maxCorr == insens.neg, "insens.neg",
                                             ifelse(maxCorr == sens.pos, "sens.pos", "sens.neg"))))))


regulated.normCount.tidy <- left_join(regulated.normCount.tidy, profile.scaledNormCountsByGene.6)
```

```
## Joining, by = "genename"
```

```
## Warning: Column `genename` joining factor and character vector, coercing
## into character vector
```

```
# add the profiles so that we can plot these alongside the normCounts
regulated.normCount.tidy <- regulated.normCount.tidy %>% group_by(genename) %>%
     mutate(profile.shape = ifelse(group == "linear.pos",  profile.linear.pos,
                            ifelse(group == "linear.neg", profile.linear.neg,
                            ifelse(group == "insens.pos", profile.insens.pos,
                            ifelse(group == "insens.neg", profile.insens.neg,
                            ifelse(group == "sens.pos", profile.sens.pos,
                                      profile.sens.neg))))) %>%
  ungroup()

ggplot(data = filter(regulated.normCount.tidy, group != "NA"), aes(x = nlevel*100, y = normCountScaled)
```




```
#examine how strong the correlations are:
summary(regulated.normCount.tidy$maxCorr)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.2964  0.8936  0.9378  0.9189  0.9697  0.9998      12
```

Comparing directional sensitivity classification to that created by pnbinom. SENS.POS

Looking at the above plots, we see that the majority of genes match up for pnbinom and the correlation profile assignment method. We see that the 17 genes pnbinom found sensitive positive which the correlation method did not have a shallow slope, and were therefore grouped with linear.pos. Interestingly, pnbinom

classified the 55 genes corr found sensitive positive which pnbinom did not as neither positive. Yet when we graph them, most of their shapes appears quite sensitive positive. We suspect this difference is because pnbinom will not find more variable significant and therefore place them into the "neither" category.

SENS.NEG

We identify 109 genes as sensitive negative through correlating the means of means curve with correlations ranging from 0.747 to 0.999, 0.923 median and 0.917 mean. In contrast, pnbinom found only 7 sensitive negative genes. All 7 of these are among the 109 found by correlation, with correlations ranging from 0.904 to 0.993, 0.943 median and 0.948 mean. If we were to keep only the genes whose correlations are 0.90 or above, 77 would be identified as sensitive negative by correlation. All 102 of the genes correlation found sensitive negative but pnbinom did not were identified by pnbinom as neither negative.Their correlations ranged from 0.747 to 0.999, 0.923 median and 0.915 mean. We see relatively low correlations and little differentiability between these correlations and correlations with the linear negative profile.

INSENS.POS

We see many more genes being identified by the correlation method as insensitive positive than by pnbinom. We also notice that the difference between the correlation for insens pos and linear pos is VERY small. This indistinguishability problem is a recurring theme.

INSENS NEG

LINEAR POS

Correlation found linear-looking genes which had been pnbinom-classified as sens.pos. It classified previously linear positive genes as sens.pos or insens.pos, and their shapes look confirming to this new classification style.

LINEAR NEG

Plotting: Logged

1. All Sensitive Positive Genes

It seems like logging does make this one more readable and lets us keep the outlying ytfQ in the plot view. But rows with zero and less than 1 are made to look weird. Though I'm confused why there are normalized counts less than 1.

2. All Sensitive Negative Genes

The iconic sensitive negative shape rather disappears, espcially for the scaled counts closer to 1.

3. All Insensitive Positive Genes

We have a problem with the slope looking very different for these points! They almost all look sensitive now, rather than insensitive. Logging can certainly do that, so this isn't shocking but it shows us that log transformations might not be the best way to go with our data.

As a close-up example, see the plot below as we follow one gene's change in slope between untransformed and transformed normCount.

4. All Insensitive Negative Genes

Logging does help mitigate the variability and allows us to see every gene on the same plot, but I am still concerned about the accuracy of ibpB's 26% measurement. 5. All Linear Positive Genes

These certainly don't look linear positive.

6. 40 Neither Negative Genes

In conclusion, I am not a fan of this particular log transformation on the data. Because the correlation profile assignment is done on the UNLOGGED data, logging is purely for visualization purposes. However, the slopes are very different on the logged data and this is confusing when we are classifying genes based on shapes. Furthermore, logging removes zero counts (because their output is undefined in the log function) and we're getting weird negative values.

Looking into DESeq suggestions for transofrmations, there are others I try below. They are all on the log scale however, so although the undefined/negative value problems goes away, the slope problem is not solved. In conclusion, I would recommend not using logged data for visualization.

Pseudo-Logged Plots (log2(normCount + 1)

Make the table of normalized counts

Try correlation on log2(normCount + 1) data

SENS.POS

data frames needed

1. All Sensitive Positive Genes
2. All Sens Neg
3. All Insensitive Positive - !!! Same problem! Obviously.

- just ymdF

4. All Insens Neg
5. First 40 Neither Pos
6. First 40 neither neg

rlog plots

data frames needed

1. All Sensitive Positive Genes
2. All Sens Neg
3. All Insensitive Positive - !!! Same problem again!

- just ymdF

4. All Insens Neg
5. First 40 Neither Pos
6. First 40 neither neg

vst plots

(use as long as size factors don't vary too much)

data frames needed

1. All Sensitive Positive Genes

2. All Sens Neg
3. All Insensitive Positive - !!! Same problem again!

- just ymdF

4. All Insens Neg
5. First 40 Neither Pos
6. First 40 neither neg