Normalization Analysis

Nate Olson

2016-11-21

```
library(metagenomeSeq)
library(tidyverse)
```

Objective

- Assessment of count table values for different pipelines.
- Impact of normlization methods on count table values.

Loading Pipeline Data

```
mrexp_files <- list(
      dada2  = "../data/mrexp_dada2.RDS",
      mothur = "../data/mrexp_mothur.RDS",
      qiime  = "../data/mrexp_qiime_refclus_nochimera.RDS"
)
mrexp <- mrexp_files %>% map(readRDS)
```

Extracting metadata

```
meta_dat <- mrexp$mothur %>% pData()

##labeling PCR replicates
half1 <- paste(rep(c("A","B","C","D","E","F","G","H"), each = 6),1:6, sep = "_")
sam_dat <- meta_dat %>%
      mutate(pcr_half = if_else(pos %in% half1, "1","2"),
             pcr_rep = paste0(pcr_16S_plate,":",pcr_half)) %>%
      select(sampleID, dilution,sam_names, pcr_rep) %>%
      rename(samID = sam_names)
```

Subsetting data to focus on one biological replicate

Only looking at biological replicate E01JH0004

```
E01JH004_sams <- meta_dat %>%
      filter(sampleID == "E01JH0004") %>% .$sam_names
mrexp_004 <- mrexp %>%
      map(~.[,which(colnames(.) %in% E01JH004_sams)]) %>%
      map(~.[which(rowSums(MRcounts(.)) > 0), ])
```

Raw, Normalized, and Transformed Count Data

Question - does the order of the normalization and log transformation matter?

```
calc_raw_counts <- function(mrexp){
      mrexp@assayData$counts %>% as_tibble() %>%
            rownames_to_column(var = "otuID") %>%
            gather("samID","count",-otuID) %>%
            left_join(sam_dat)
}

calc_css_counts <- function(mrexp, norm = TRUE,log = TRUE,sl = 1000, p = 0.75){
      ## col_id is the name of the col in the output
      ## dataframe with the css normalized counts
      #mrexp %>% cumNormMat(p = p) %>% as_tibble() %>%
      mrexp %>% cumNorm(p = p) %>%
            MRcounts(norm, log, sl) %>% as_tibble() %>%
            rownames_to_column(var = "otuID") %>%
            gather("samID","count",-otuID) %>%
            left_join(sam_dat)
}

# TSS from http://mixomics.org/mixmc/normalisation/
calc_tss_counts <- function(mrexp){
      mrexp@assayData$counts %>% {apply(., 2, function(x){ x/sum(x) })} %>%
            as_tibble() %>% rownames_to_column(var = "otuID") %>%
            gather("samID","count",-otuID) %>%
            left_join(sam_dat)
}

calc_tsslog_counts <- function(mrexp){
      mrexp@assayData$counts %>%
            {apply(., 2, function(x){ x/sum(x) })} %>% {log2(. + 1)} %>%
            as_tibble() %>% rownames_to_column(var = "otuID") %>%
            gather("samID","count",-otuID) %>%
            left_join(sam_dat)
}
```

```
raw_counts <- mrexp_004 %>% map_df(calc_raw_counts, .id = "pipe")
rawlog_counts <- mrexp_004 %>% map_df(calc_raw_counts, .id = "pipe") %>%
      mutate(count = log2(count + 1))
```

```
css_counts <- mrexp_004 %>% map_df(calc_css_counts, log = FALSE, .id = "pipe")
csslog_counts <- mrexp_004 %>% map_df(calc_css_counts, .id = "pipe")
```

```
tss_counts <- mrexp_004 %>% map_df(calc_tss_counts, .id = "pipe")
tsslog_counts <- mrexp_004 %>% map_df(calc_tsslog_counts, .id = "pipe")
```

Combine into a single data frame

```
count_df <- list(raw = raw_counts, rawlog = rawlog_counts,
                 css = css_counts, csslog = csslog_counts,
                 tss = tss_counts, tsslog = tsslog_counts) %>%
      bind_rows(.id = "norm_method")
```

Count Value Variance

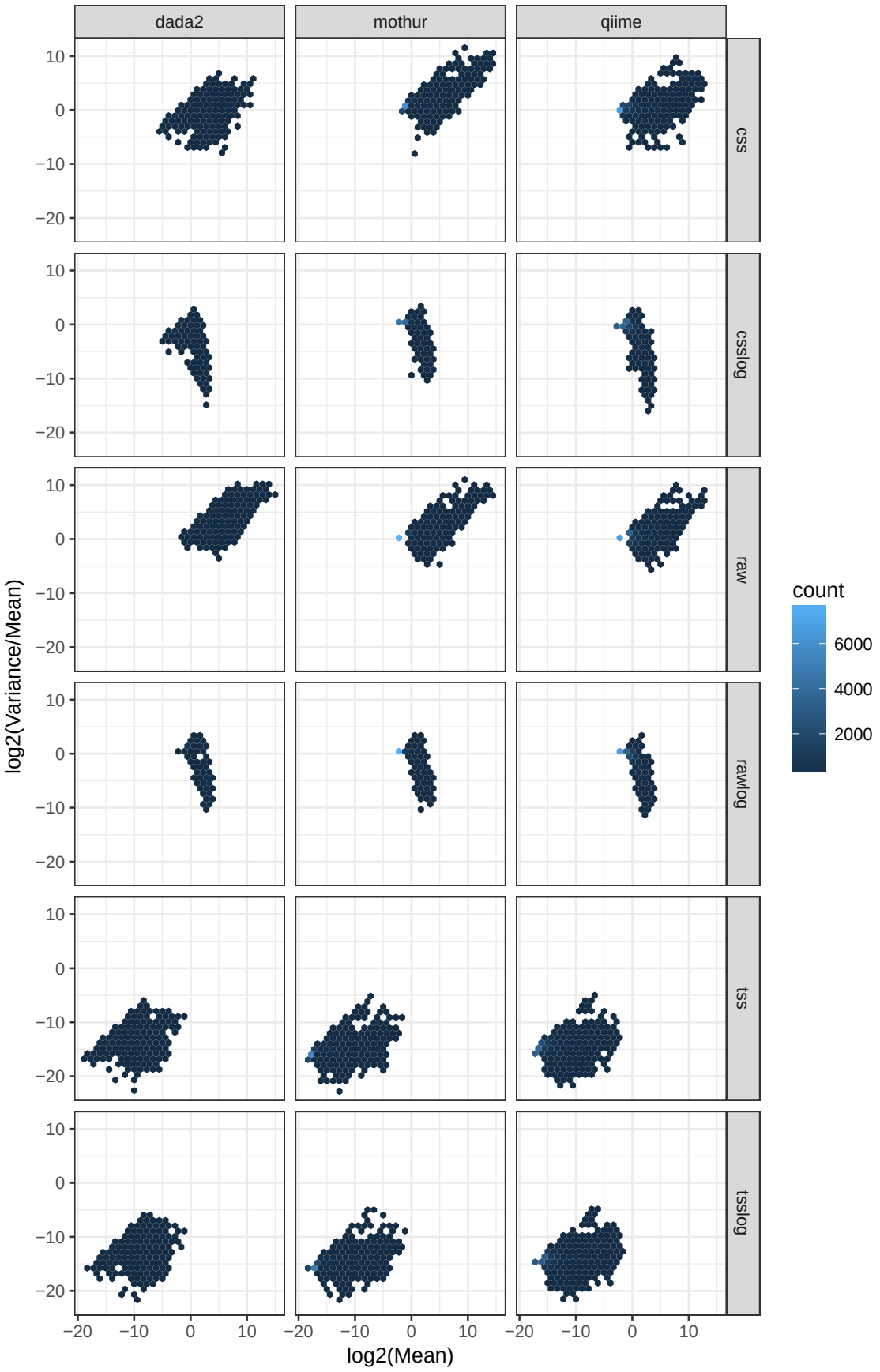
Relationship between the mean count and variance for the four PCR replicates. Note the heteroscedasicity, increase in variance with mean counts.

```
count_var_df <- count_df %>% group_by(pipe, norm_method, dilution, otuID) %>%
      summarise(mean_count  = mean(count),
                var_count   = var(count),
                lmean_count = log2(mean_count),
                lvar_count = log2(var_count),
                diff_lmean_lvar = log2((var_count)/(mean_count)))
```

Interesting distribution for dada2 mean variance relationship. Higher variance for lower count values expected.

```
# log mean x, log variance - log mean y : expectation of 1
ggplot(count_var_df) +
      # geom_point(aes(x = mean_count, y = var_count), alpha = 0.5) +
      # geom_hex(aes(x = mean_count, y = var_count)) +
      geom_hex(aes(x = lmean_count, y = diff_lmean_lvar)) +
      facet_grid(norm_method~pipe) +
      theme_bw() + labs(x = "log2(Mean)", y = "log2(Variance/Mean)")
```

```
## Warning: Removed 609366 rows containing non-finite values (stat_binhex).
```


dada2
mothur
qiime
css
csslog
raw
rawlog
tss
tsslog
log2(Variance/Mean)
log2(Mean)
count
6000
4000
2000
10
0
−10
−20

Count Value Bias

Relationship between the observed and expected count values. Expected count (C_{exp})values calculated using the unmixed sample count values (unmixed pre - C_{pre} and unmixed post - C_{post}) and proportion of unmixed pre in the titration p. Proportion is defined as $p = 2^{-t}$, and t is the titration factor.

$$C_{exp} = [C_{pre} \times p] + [C_{post} \times (1 - p)]$$

The expected values are calculated based on pre and post unmixed samples by replicate (defined as half of PCR plate).

```
pre_count <- count_df %>% filter(dilution == -1) %>%
      rename(pre = count) %>% select(-dilution, -samID)
post_count <- count_df %>% filter(dilution == 0) %>%
      rename(post = count) %>% select(-dilution, -samID)
pre_post_count <- left_join(pre_count, post_count)
```

```
## Joining, by = c("norm_method", "pipe", "otuID", "sampleID", "pcr_rep")
```

```
rm(pre_count, post_count)

count_exp_obs <- count_df %>%
      filter(!(dilution %in% c(0,-1))) %>%
      left_join(pre_post_count) %>%
      mutate(p = 2^(-dilution), exp_count = post * (1-p) + pre * p)
```

```
## Joining, by = c("norm_method", "pipe", "otuID", "sampleID", "pcr_rep")
```

Metrics for evaluating count values

The overall pipeline and normalization method performance was evaluated using root mean squared error (RMSE) and the normalized RMSE (NRMSE). Normalizing RMSE allow for the comparison of metric value across pipeline and normalization methods. Overall the count table generated using the DADA2 sequence inference based method with CSS normalization and log2 transformation had the lowest NRMSE. The NRMSE for the QIIME pipeline, open reference clustering, was comparable for CSS and TSS normalization method.
Log2 transformation lowered that NRMSE for all three pipelines more than either TSS or CSS normalization.

```
count_rmse <- count_exp_obs %>% mutate(residual = (exp_count - count)^2) %>%
      group_by(pipe, norm_method) %>%
      summarise(mse = mean(residual),
                rmse = sqrt(mse),
                nrmse = rmse/mean(exp_count))
```

RMSE - pipeline and normalization method

```
count_rmse %>% select(-mse, -nrmse) %>%
      spread(norm_method, rmse) %>% knitr::kable()
```

pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	20.96231	0.7206376	241.12672	1.5192967	0.0026461	0.0034466
mothur	104.11251	0.4472386	74.30441	0.3803504	0.0010022	0.0012967
qiime	25.79730	0.5534182	35.53387	0.5984425	0.0006685	0.0009356

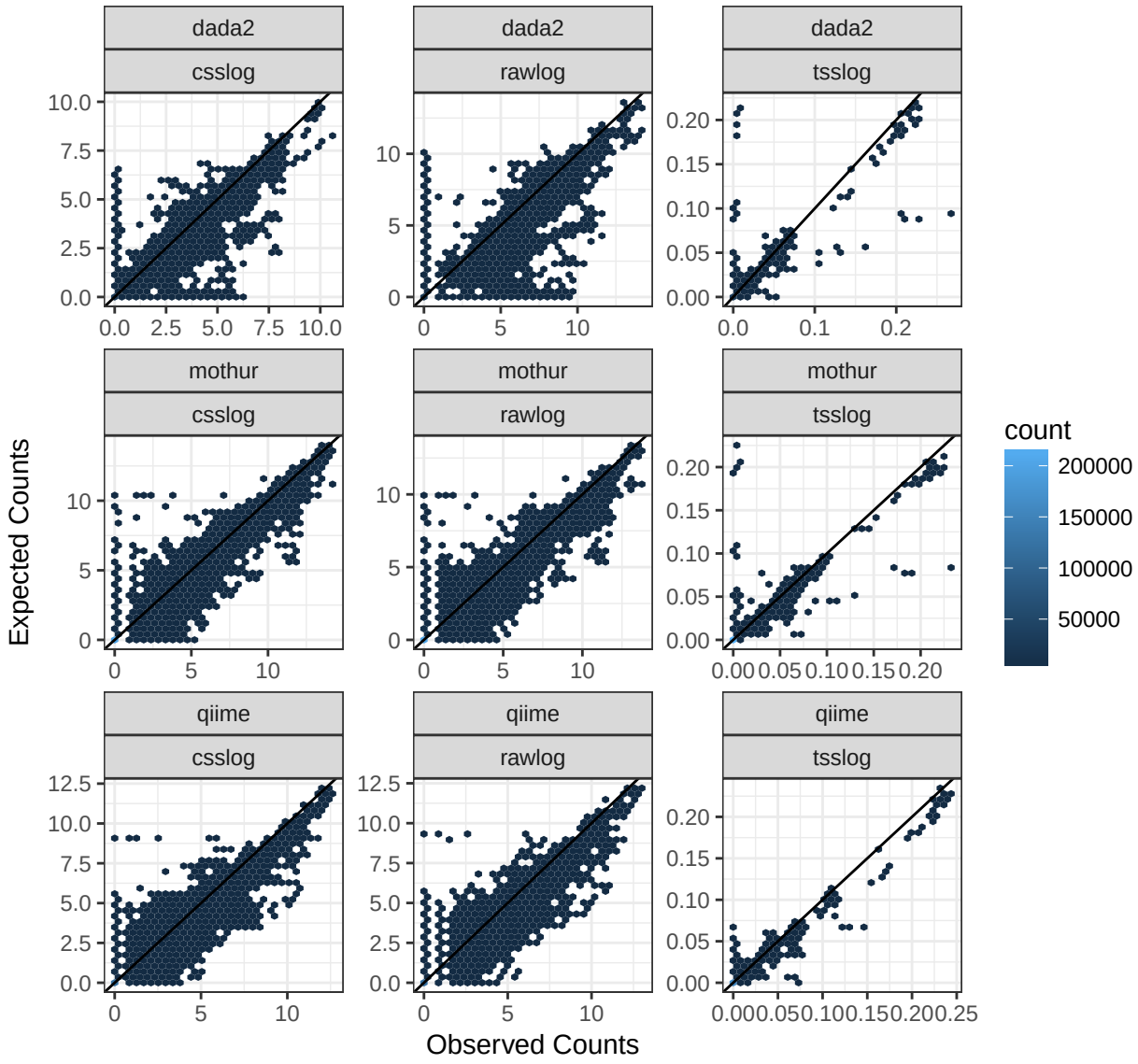
NRMSE - pipeline and normalization mehod

```
count_rmse %>% select(-mse, -rmse) %>%
      spread(norm_method, nrmse) %>% knitr::kable()
```

pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	3.976480	1.390158	4.296486	1.542956	3.172620	2.932782
mothur	9.962408	2.463043	10.300742	2.383314	7.884024	7.267840
qiime	5.896019	1.772287	7.324960	1.830043	3.833844	3.800375

Black line indicates expected 1 to 1 relationship between the expected and observed values.

```
count_exp_obs %>% filter(norm_method %in% c("csslog","tsslog", "rawlog")) %>%
      ggplot() +
      geom_hex(aes(x = count, y = exp_count)) +
      geom_abline(aes(intercept = 0, slope = 1)) +
      facet_wrap(pipe~norm_method, ncol = 3, scales = "free") +
      theme_bw() + labs(x = "Observed Counts", y = "Expected Counts")
```



Metrics for evaluating count values excluding 0

When excluding observed 0 count values the NRMSE are significantly lower for all pipelines.

```
count_rmse <- count_exp_obs %>% filter(count != 0) %>%
      mutate(residual = (exp_count - count)^2) %>%
      group_by(pipe, norm_method) %>%
      summarise(mse = mean(residual),
                rmse = sqrt(mse),
                nrmse = rmse/mean(exp_count))
```

RMSE - pipeline and normalization method

```
count_rmse %>% select(-mse, -nrmse) %>%
      spread(norm_method, rmse) %>% knitr::kable()
```

pipe	css	csslog	raw	rawlog	tss	tsslog

pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	50.03406	1.355571	582.3248	2.896177	0.0063181	0.0081753
mothur	415.86404	1.376001	296.8210	1.194341	0.0039958	0.0051646
qiime	63.80504	1.087125	88.6479	1.188001	0.0015984	0.0022349

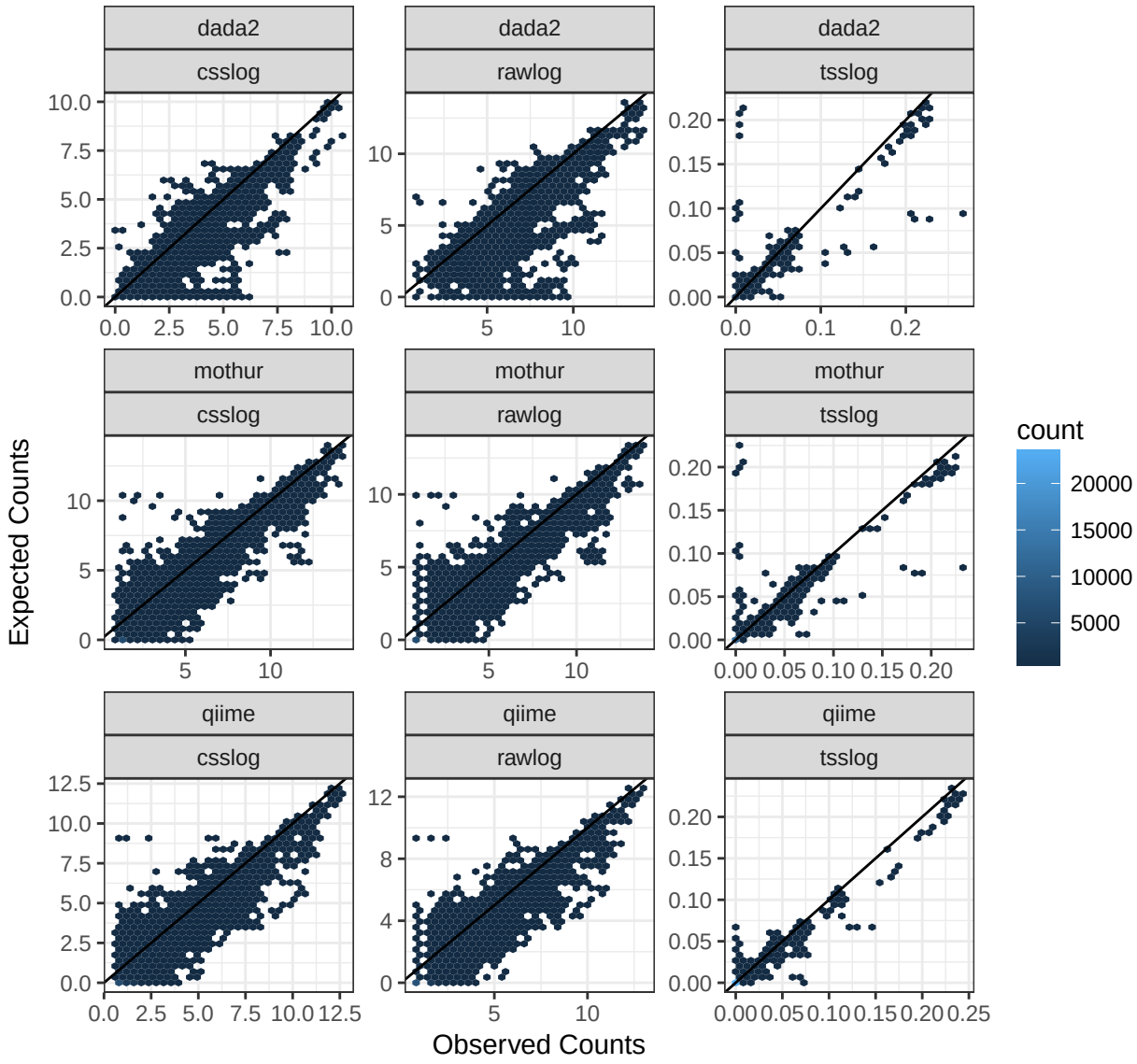
NRMSE - pipeline and normalization mehod

```
count_rmse %>% select(-mse, -rmse) %>%
      spread(norm_method, nrmse) %>% knitr::kable()
```

pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	1.724800	0.5290210	1.867534	0.6335966	1.353507	1.244655
mothur	2.522655	0.6618411	2.609391	0.6306685	1.993238	1.836242
qiime	2.415935	0.7998824	3.015831	0.8405510	1.523363	1.509793

Black line indicates expected 1 to 1 relationship between the expected and observed values.

```
count_exp_obs %>% filter(norm_method %in% c("csslog","tsslog", "rawlog"), count != 0) %>%
      ggplot() +
      geom_hex(aes(x = count, y = exp_count)) +
      geom_abline(aes(intercept = 0, slope = 1)) +
      facet_wrap(pipe~norm_method, ncol = 3, scales = "free") +
      theme_bw() + labs(x = "Observed Counts", y = "Expected Counts")
```


dada2
csslog
dada2
rawlog
dada2
tsslog
mothur
csslog
mothur
rawlog
mothur
tsslog
qiime
csslog
qiime
rawlog
qiime
tsslog
Expected Counts
Observed Counts
count
20000
15000
10000
5000