Normalization Analysis

Nate Olson

2016-11-20

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

Objective

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

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

Keeping biological replicate E01JH0004

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

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, 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() %>%
            rownames_to_column(var = "otuID") %>%
            gather("samID","count",-otuID) %>%
            left_join(sam_dat)
}

# TSS from http://mixomics.org/mixmc/normalisation/
TSS.divide = function(x){ x/sum(x) }

calc_tss_counts <- function(mrexp){
      mrexp@assayData$counts %>% {apply(., 2, TSS.divide)} %>%
            as_tibble() %>% rownames_to_column(var = "otuID") %>%
            gather("samID","count",-otuID) %>%
            left_join(sam_dat)
}

calc_tsslog_counts <- function(mrexp){
      mrexp@assayData$counts %>%
            {log2(. + 1)} %>% {apply(., 2, TSS.divide)} %>%
            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, .id = "pipe")
csslog_counts <- mrexp_004 %>% map_df(calc_css_counts, .id = "pipe") %>%
      mutate(count = log2(count + 1))
```

```
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

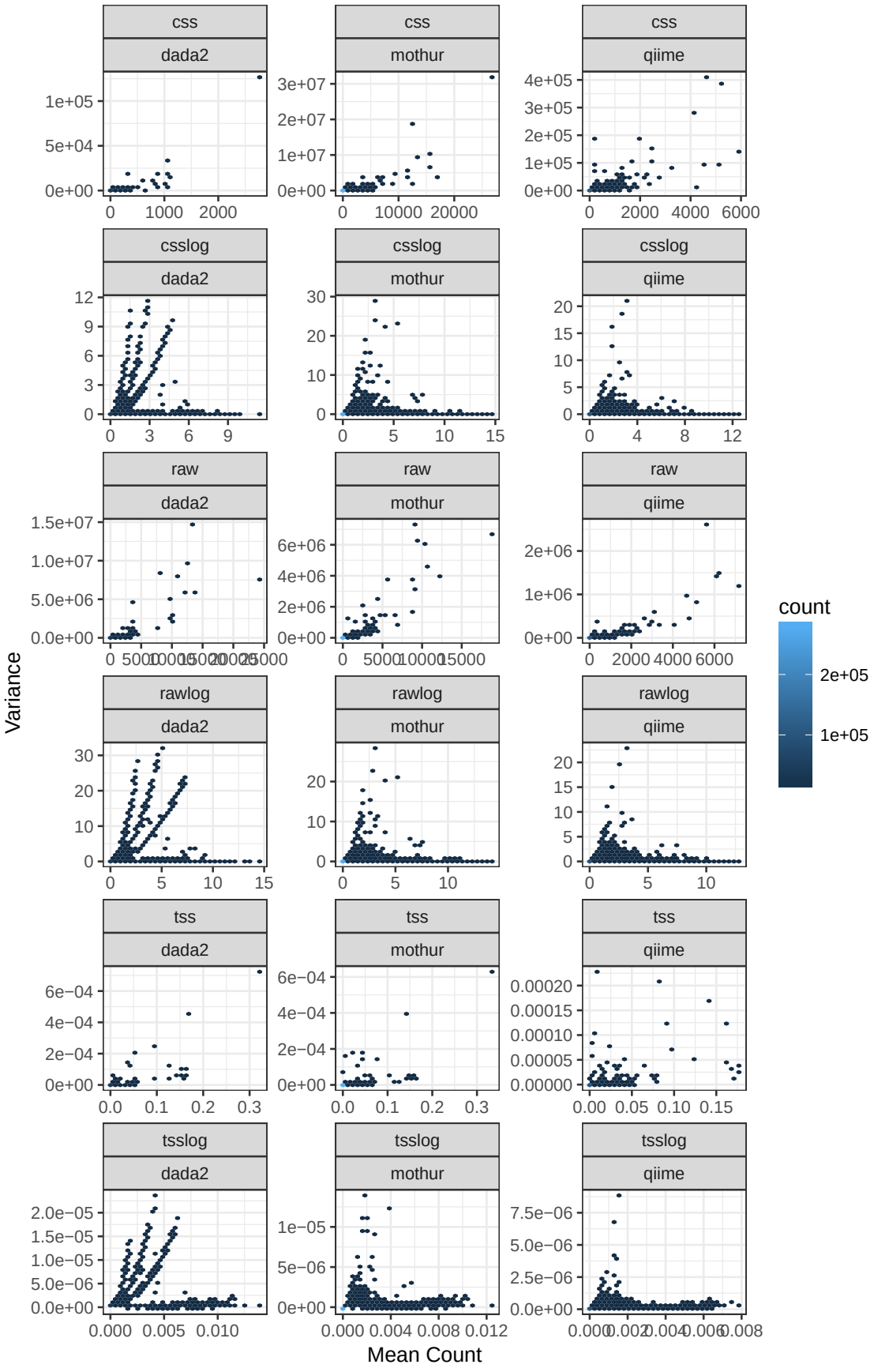
```
count_df <- list(raw = raw_counts, rawlog = rawlog_counts,
                 css = css_counts, csslog = csslog_counts,
                 tss = tss_counts, tsslog = tsslog_counts) %>%
      bind_rows(.id = "norm_method")
```

Count Value Variance

Relationship between the mean count and variance for the four PCR replicates. Note the heteroscedasicity, increase in variance with mean counts. Will want to look into log2 transforming count values. For tss normalization does order of operations impact the values.

```r
count_var_df <- count_df %>% group_by(pipe, norm_method, dilution, otuID) %>%
      summarise(mean_count  = mean(count),var_count   = var(count))
```

```r
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)) +
      facet_wrap(norm_method~pipe, nrow = 6, scales = "free") +
      theme_bw() + labs(x = "Mean Count", y = "Variance")
```


css
dada2
mothur
qiime
csslog
raw
rawlog
tss
tsslog
Variance
Mean Count
count
2e+05
1e+05

Count Value Bias

Relationship between the expected count values calculated using the unmixed sample count values and proportions.

Calculating expected values 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

```
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()
```

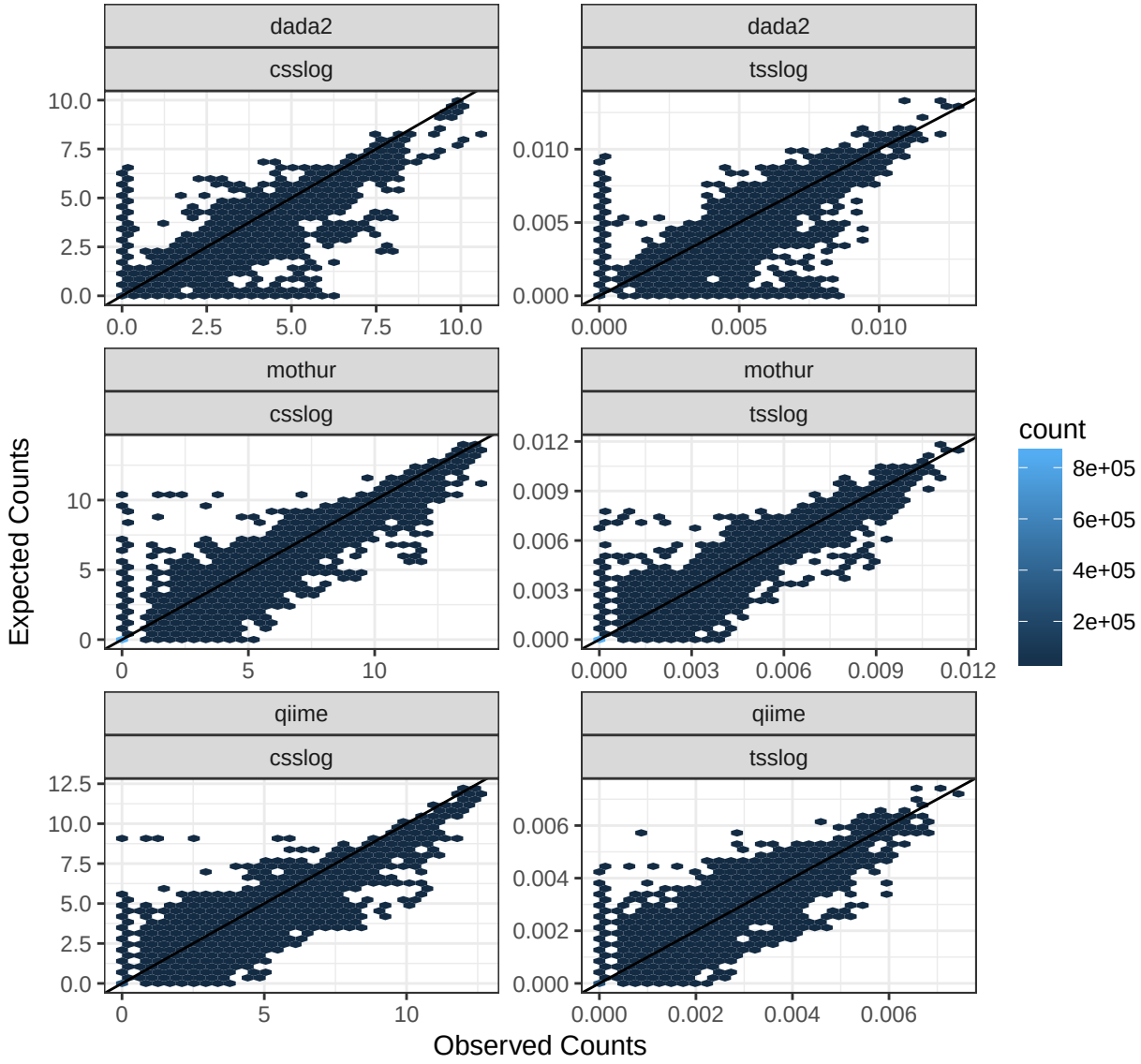
pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	11.94749	0.4107282	137.43043	0.8659248	0.0015081	0.0006963
mothur	51.66372	0.2219331	36.87206	0.1887412	0.0004973	0.0001438
qiime	18.31263	0.3928529	25.22429	0.4248141	0.0004745	0.0002183

NRMSE - pipeline and normalization mehod

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

pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	6.976880	2.439084	7.538342	2.707173	5.566477	2.570212
mothur	20.076203	4.963514	20.758012	4.802844	15.887853	4.594221
qiime	8.305818	2.496649	10.318790	2.578011	5.400798	2.484070

```
count_exp_obs %>% filter(norm_method %in% c("csslog","tsslog")) %>%
      ggplot() +
      geom_hex(aes(x = count, y = exp_count)) +
      geom_abline(aes(intercept = 0, slope = 1)) +
      facet_wrap(pipe~norm_method, ncol = 2, scales = "free") +
      theme_bw() + labs(x = "Observed Counts", y = "Expected Counts")
```



Metrics for evaluating count values excluding 0

```
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()
```

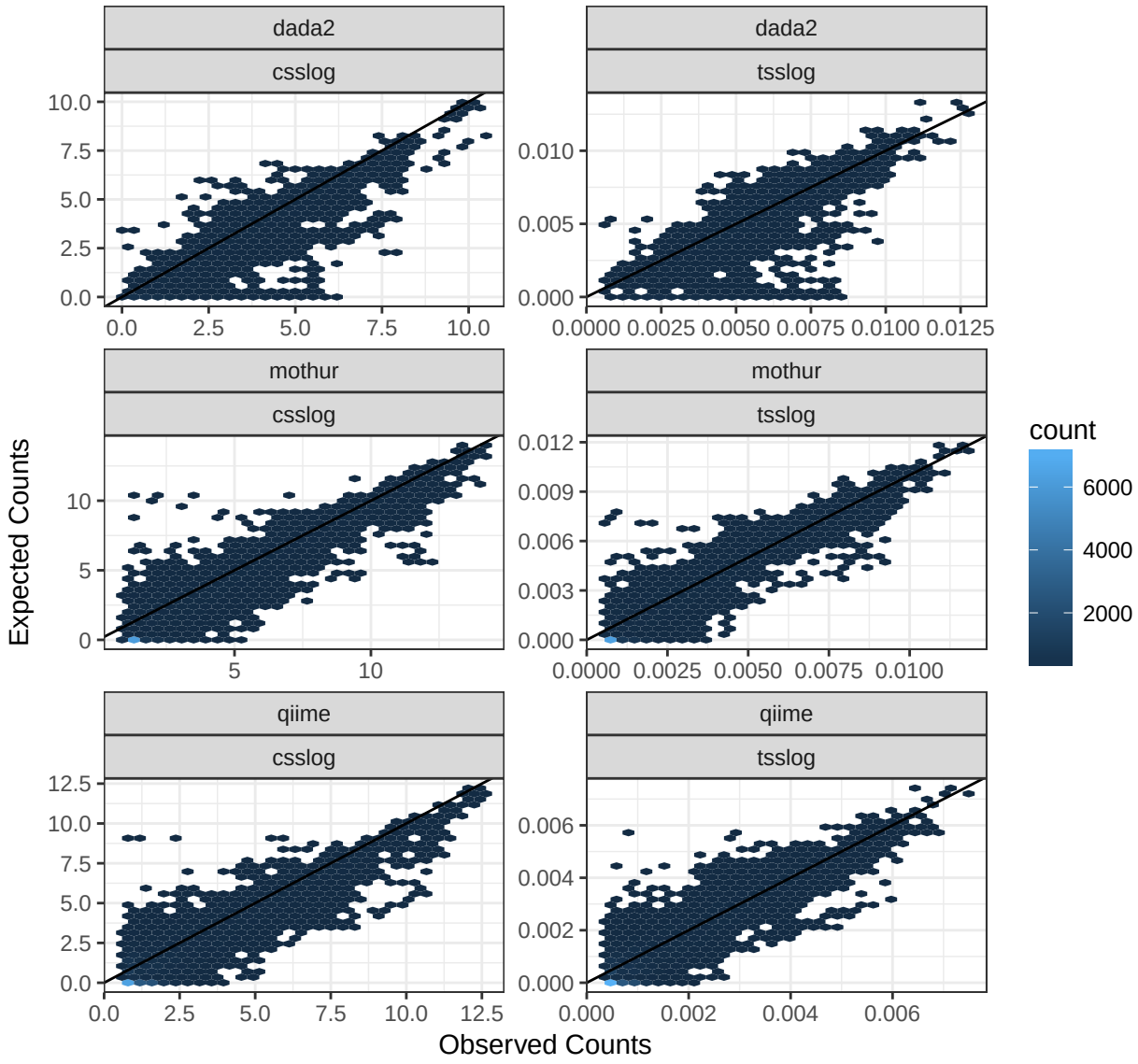
pipe	css	csslog	raw	rawlog	tss	tsslog
dada2	50.03406	1.355571	582.3248	2.896177	0.0063181	0.0022428
mothur	415.86404	1.376001	296.8210	1.194341	0.0039958	0.0008856
qiime	63.80504	1.087125	88.6479	1.188001	0.0015984	0.0005929

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	0.5811746
mothur	2.522655	0.6618411	2.609391	0.6306685	1.993238	0.5869594
qiime	2.415935	0.7998824	3.015831	0.8405510	1.523363	0.7903155

```
count_exp_obs %>% filter(norm_method %in% c("csslog","tsslog"), count != 0) %>%
      ggplot() +
      geom_hex(aes(x = count, y = exp_count)) +
      geom_abline(aes(intercept = 0, slope = 1)) +
      facet_wrap(pipe~norm_method, ncol = 2, scales = "free") +
      theme_bw() + labs(x = "Observed Counts", y = "Expected Counts")
```


dada2
csslog
dada2
tsslog
mothur
csslog
mothur
tsslog
qiime
csslog
qiime
tsslog
Expected Counts
Observed Counts
count
6000
4000
2000