Normalization Analysis

Nate Olson

2016-11-29

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

Objective

- Assessment of count table values for different pipelines.
- Impact of normlization methods on count table values.
- Evaluate the bias-variance relationship for single summary metrics - will initially used normalized RMSE.
- Desired summary metric characteristics.
  - able to obtain a distribution of the values
  - scaled to facilitate comparison across datasets

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, to avoid overfitting the data.

```
E01JH004_sams <- meta_dat %>%
      filter(sampleID == "E01JH0004") %>% .$sam_names
mrexp_004 <- mrexp %>%
```

```
      map(~.[,which(colnames(.) %in% E01JH004_sams)]) %>%
      map(~.[which(rowSums(MRcounts(.)) > 0), ])
```

Extracting Raw, Normalized, and Transformed Count Data

Order of normalization and transformation impacts results. What is the appropriate ordering?

```
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){
      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)
}

## DESeq method median of ratios
calc_dsq_counts <- function(mrexp){
  mrexp@assayData$counts %>% {./estimateSizeFactorsForMatrix(.)} %>%
    as_tibble() %>% rownames_to_column(var = "otuID") %>%
    gather("samID","count",-otuID) %>%
    left_join(sam_dat)
}

calc_dsqlog_counts <- function(mrexp){
  mrexp@assayData$counts %>%
    {./estimateSizeFactorsForMatrix(.)} %>% {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))
```

```
uqs_counts <- mrexp_004 %>% map_df(calc_css_counts, p = 0.75,sl = 1, log = FALSE, .id = "pipe")
uqslog_counts <- mrexp_004 %>% map_df(calc_css_counts, sl = 1, .id = "pipe")

css_counts <- mrexp_004 %>%
  {map_df(.x=., .f=~calc_css_counts(.,log = FALSE, p = cumNormStat(.), sl = 1), .id = "pipe")}
csslog_counts <- mrexp_004 %>%
  {map_df(.x=., .f=~calc_css_counts(., p = cumNormStat(.), sl = 1),  .id = "pipe")}
```

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

```
dsq_counts <- mrexp_004 %>% map_df(calc_dsq_counts, .id = "pipe")
```

```
## Joining, by = "samID"
## Joining, by = "samID"
## Joining, by = "samID"
```

```
dsqlog_counts <- mrexp_004 %>% map_df(calc_dsqlog_counts, .id = "pipe")
```

```
## Joining, by = "samID"
## Joining, by = "samID"
## Joining, by = "samID"
```

Combine into a single data frame

```
count_df <- list(raw = raw_counts, rawlog = rawlog_counts,
                 uqs = uqs_counts, uqslog = uqslog_counts,
                 css = css_counts, csslog = csslog_counts,
                 tss = tss_counts, tsslog = tsslog_counts,
                 dsq = dsq_counts, dsqlog = dsqlog_counts) %>%
      bind_rows(.id = "norm_method")
```

ALL E01JH0004 OTUs Further subsetting dataset for faster development.

```
# count_df <- count_df %>%
#   select(norm_method, pipe, otuID) %>% unique() %>%
#   group_by(norm_method, pipe) %>% sample_n(1000) %>%
#   left_join(count_df)
```

Count Value Variance

```
# calculating mean and variance for technical count replicates
count_var_df <- count_df %>%
  group_by(pipe, norm_method, dilution, otuID) %>%
  summarise(mean_count  = mean(count),
            var_count   = var(count),
            n_unique = n_distinct(count))

# Removing features with 0 counts for all replicates
count_var_df <- count_var_df %>% filter(mean_count != 0 & n_unique != 1) %>%
  mutate(diff_var_mean = log2(var_count + 1) - log2(mean_count + 1),
```

```
         lmean = log2(mean_count + 1))
```

Count Mean-Variance Relationship

Relationship between the mean count and variance for the four PCR replicates. Differential abundance methods assume different count distributions for samples in the groups being compared (biological replicates). revious work has shown feature counts RNAseq technical replicates are Poisson distributions. Mean and variance are equal for Poisson distributed data. Negative binomial is used to model the over dispersion in count values between biological replicates in differential expression methods such as DESeq.

Before normalization the ratio of count variance to count mean is greater than 1. Count variance to count mean ratio Values greater than 1 indicate that the raw count data is not Poisson distributed but is overdispered. **Is this due to the increased sparseness relative to bulk RNAseq?** After normalization and transformation the ratio count variance and mean is less than 1 for most methods. As the mean and variance values are not equal for raw count or transformed count data the technical replicate count data are not Poisson distributed.

```
# log mean x, log variance - log mean y : expectation of 1
count_var_df %>% filter(!(norm_method %in% c("tss","tsslog"))) %>%
ggplot() +
      geom_hex(aes(x = lmean, y = diff_var_mean)) +
      geom_hline(aes(yintercept = 1), color = "grey") +
      facet_grid(norm_method~pipe) +
      theme_bw() + labs(x = "log2(Mean)", y = "log2(Variance/Mean)")
```

Impact of Normalization and Tranformation on Count Variance

The count variance was scaled by dividing variance by the mean variance to facilitate comparison between pipeline, normalization method, and log transformed data. The count variance values are heavy tailed based on the proportion of outliers in boxplots. The scaled variance increased after log transformation for the raw counts and DESeq normalized count data. Overall log transformation tends to increse the scaled variance values especially for the mothur and qiime pipelines.

```
count_var_df %>%
  ungroup() %>%
  mutate(log_tran = if_else(grepl("log", norm_method),"log","raw"),
         norm_method = gsub("log","", norm_method)) %>%
  group_by(pipe, norm_method, log_tran) %>%
  mutate(m = mean(var_count),
         scaled_var = var_count/m) %>%
  ggplot() + geom_boxplot(aes(y = scaled_var +1 , x = norm_method, color = log_tran), outlier.alpha = 0
  facet_grid(.~pipe) + scale_y_log10() +
  labs(x = "Normalization Method",y = "Scaled Variance [var/mean(var)]", color = "Log Transformed") +
  theme_bw() + theme(legend.position = "bottom")
```

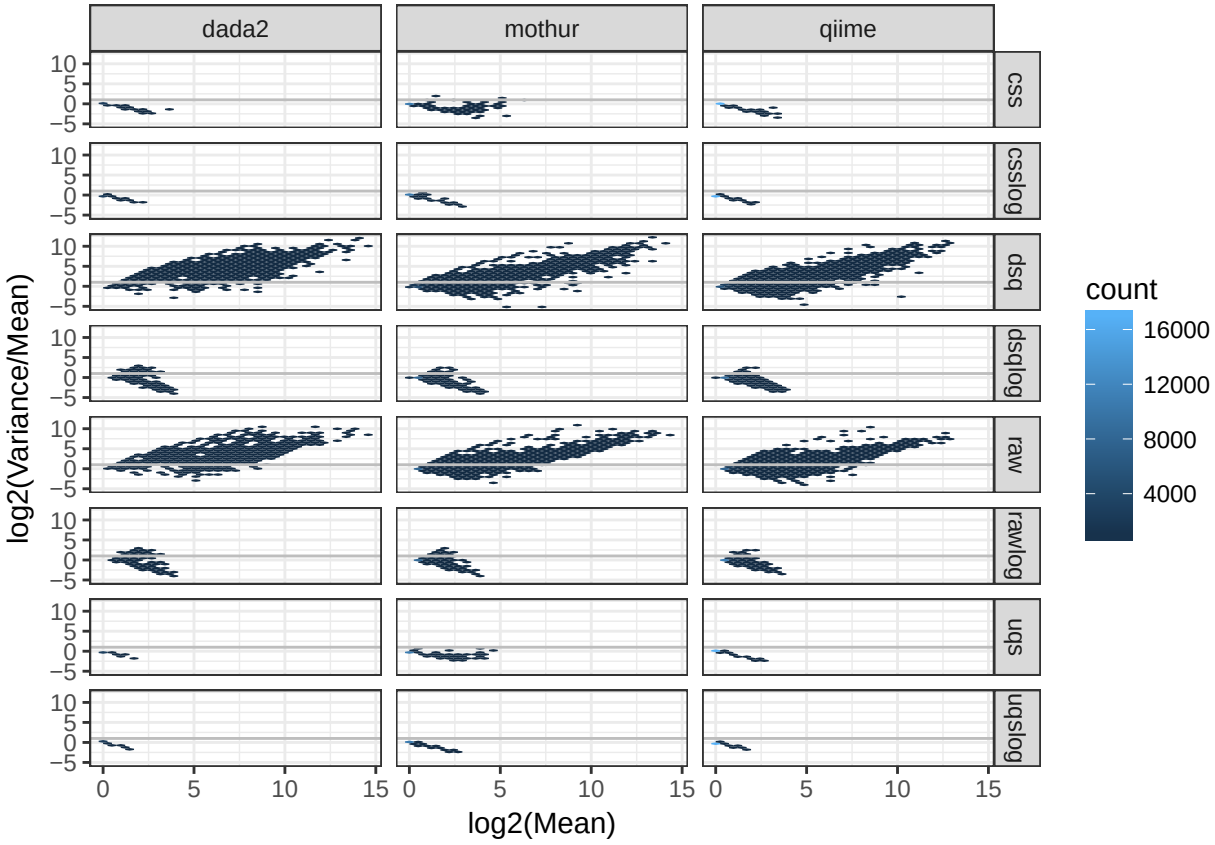
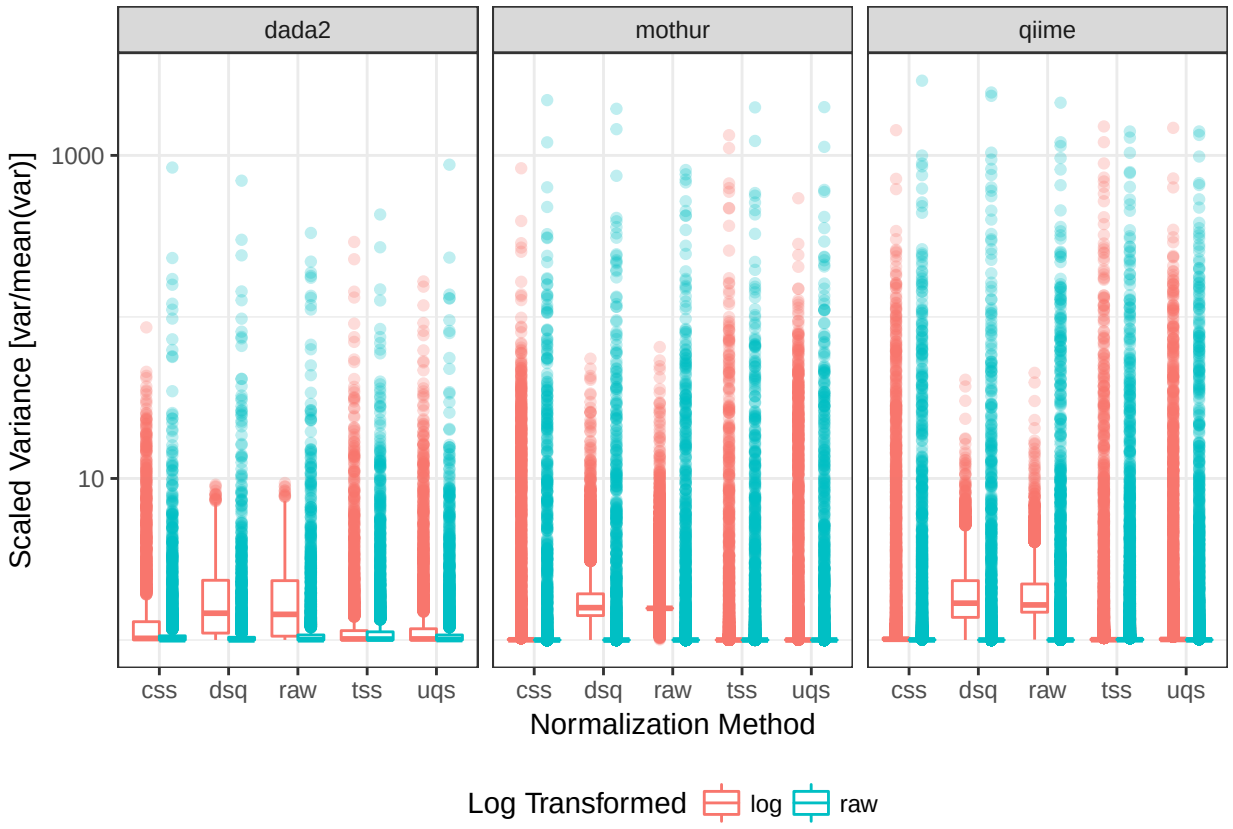


Figure 1: Ratio of count variance and mean. For data following a Poisson distribution, the expected value of 1, is indicated by the grey horizontal line. Ratio values greater and less than 1 indicate the variance is greater than and less than the mean respectively.



Count Variance Summary Metric

Using RMSE with mean count values

```
var_rmse <- count_var_df %>% ungroup() %>%
  group_by(pipe, norm_method) %>%
  mutate(m = mean(var_count), scaled_var = var_count/m) %>%
  summarise(mse = mean(var_count), rmse = sqrt(mse), nrmse = rmse/mean(mean_count))
```

RMSE - pipeline and normalization method

```
var_rmse %>% select(-mse, -nrmse) %>%
      spread(pipe, rmse) %>% knitr::kable()
```

norm_method	dada2	mothur	qiime
css	0.0668902	0.2924530	0.0303492
csslog	0.0402520	0.0512820	0.0164505
dsq	347.3505547	162.1694996	79.0546261
dsqlog	1.9674496	0.6858055	0.7527170
raw	209.9279351	94.4502885	34.9250911
rawlog	1.9456992	0.6615090	0.7168311
tss	0.0012917	0.0005608	0.0004016
tsslog	0.0017242	0.0007399	0.0005506
uqs	0.0121463	0.1277481	0.0169854
uqslog	0.0113051	0.0382119	0.0118459

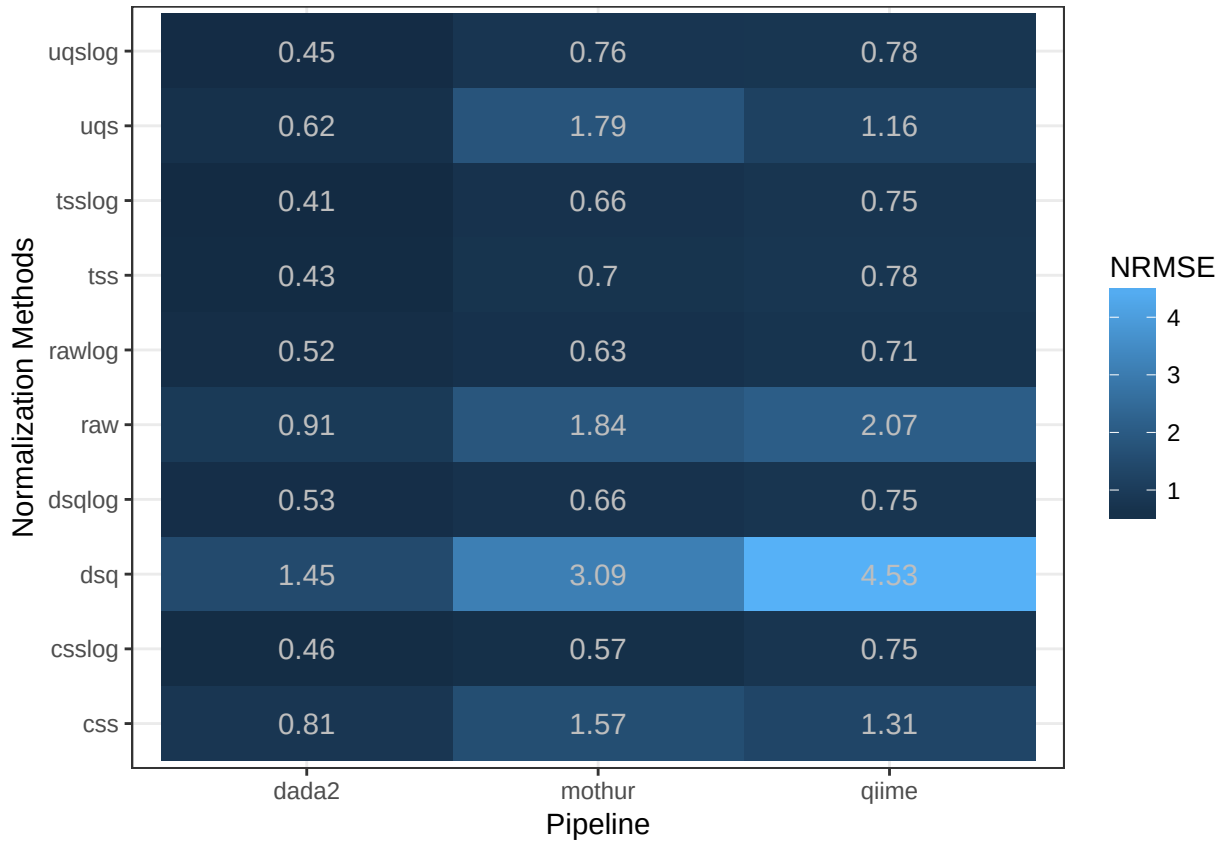


Figure 2: Normalized RMSE for pipelines and normalization methods providing an overall summary of the count variance. Lower values are better.

NRMSE - pipeline and normalization mehod

```
var_rmse %>% select(-mse, -rmse) %>%
      spread(pipe, nrmse) %>% knitr::kable()
```

norm_method	dada2	mothur	qiime
css	0.8051584	1.5708973	1.3058265
csslog	0.4607430	0.5668869	0.7467125
dsq	1.4522738	3.0872532	4.5269719
dsqlog	0.5278418	0.6585469	0.7509851
raw	0.9087793	1.8413433	2.0706431
rawlog	0.5222174	0.6333994	0.7118919
tss	0.4287050	0.7039345	0.7752539
tsslog	0.4061556	0.6613188	0.7529938
uqs	0.6170706	1.7917746	1.1627545
uqslog	0.4511300	0.7585443	0.7760601

```
var_rmse %>%
  ggplot() + geom_raster(aes(x = pipe, y = norm_method, fill = nrmse)) +
  geom_text(aes(x = pipe, y = norm_method, label = round(nrmse, 2)), color = "grey") +
  theme_bw() + labs(x = "Pipeline",y = "Normalization Methods", fill = "NRMSE")
```

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(pipe, rmse) %>% knitr::kable()
```

norm_method	dada2	mothur	qiime
css	0.0878928	0.2932243	0.0438628
csslog	0.0479714	0.0504240	0.0258951
dsq	303.3002593	91.0100487	64.1820574
dsqlog	1.5298258	0.3936908	0.6255237
raw	241.1267212	74.3044149	35.5338692

norm_method	dada2	mothur	qiime
rawlog	1.5192967	0.3803504	0.5984425
tss	0.0026461	0.0010022	0.0006685
tsslog	0.0034466	0.0012967	0.0009356
uqs	0.0209623	0.1041125	0.0257973
uqslog	0.0177906	0.0325843	0.0186266

NRMSE - pipeline and normalization mehod

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

norm_method	dada2	mothur	qiime
css	4.070635	11.413823	6.416214
csslog	2.084291	3.787616	3.842633
dsq	4.999154	12.040544	12.378209
dsqlog	1.551099	2.471712	1.919355
raw	4.296486	10.300742	7.324960
rawlog	1.542956	2.383314	1.830043
tss	3.172620	7.884024	3.833844
tsslog	2.932782	7.267840	3.800375
uqs	3.976480	9.962408	5.896019
uqslog	2.646888	4.266234	3.957800

```
count_rmse %>%
  ggplot() + geom_raster(aes(x = pipe, y = norm_method, fill = nrmse)) +
  geom_text(aes(x = pipe, y = norm_method, label = round(nrmse, 2)), color = "grey") +
    theme_bw() + labs(x = "Pipeline",y = "Normalization Methods", fill = "NRMSE")
```

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")
```

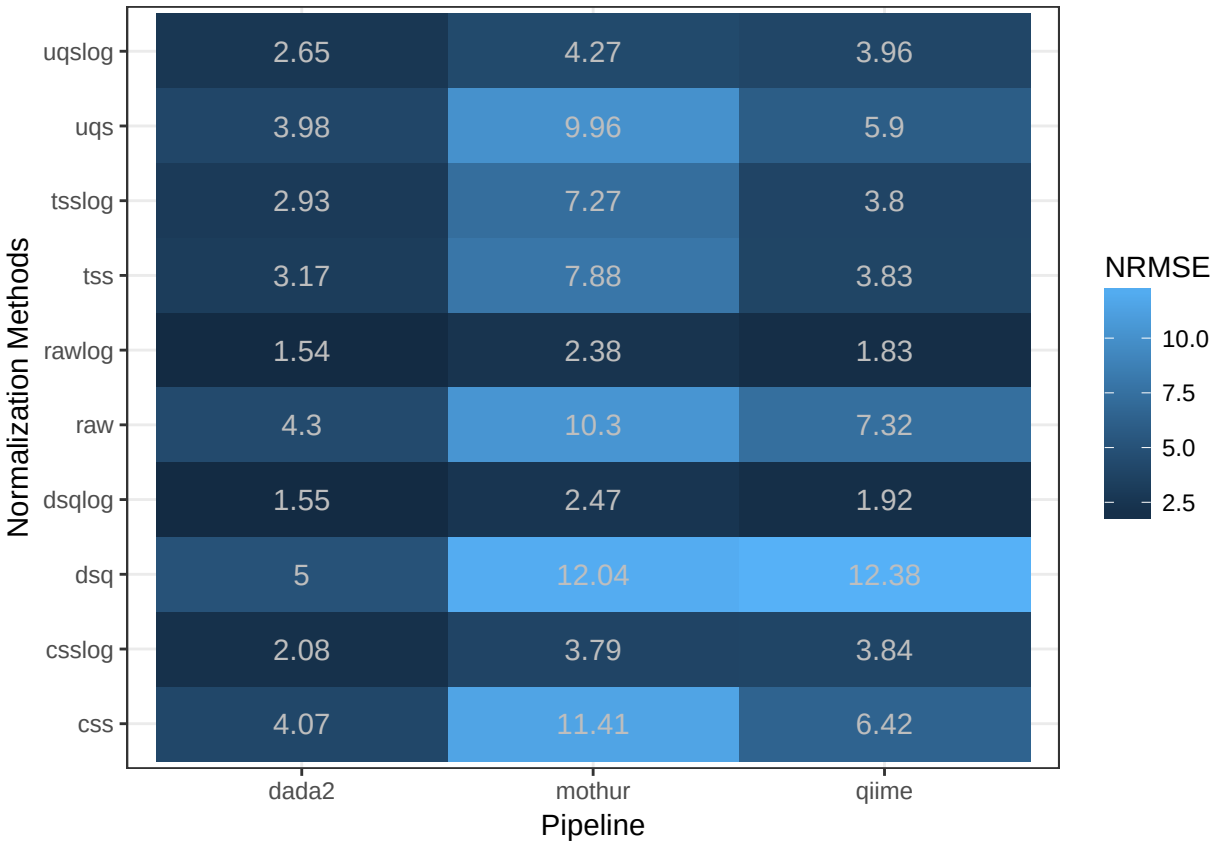
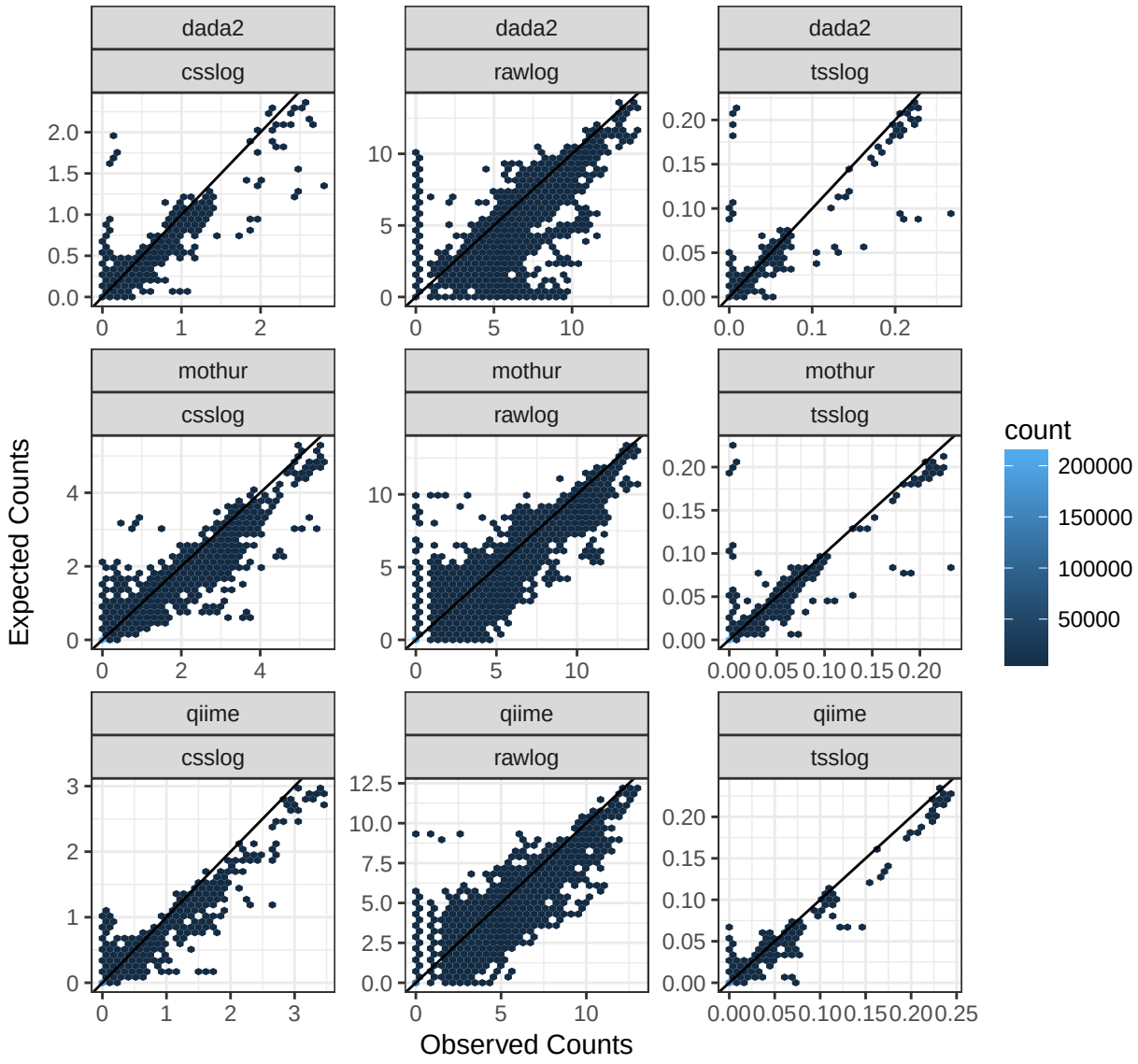


Figure 3: Normalized RMSE for pipelines and normalization methods providing an overall summary of the count bias. Lower values are better.



Proposal figure 3

```
# count_exp_obs %>% filter(norm_method %in% c("raw","csslog"), pipe == "dada2") %>%
#       mutate(norm_method = factor(norm_method, levels =  c("raw","csslog"))) %>%
#       ggplot() +
#       geom_point(aes(x = count, y = exp_count), alpha = 0.25, color = "darkblue") +
#       geom_abline(aes(intercept = 0, slope = 1),color = "darkorange") +
#       facet_wrap(~norm_method, scales = "free") +
#       theme_bw() + labs(x = "Observed Counts", y = "Expected Counts")
# ggsave("~/Projects/proposal/proposal_figure3.png",dpi = 450)
```

Variance-Bias Relationship

```
var_nrmse <- var_rmse %>% select(-mse, -rmse) %>% rename(var_nrmse= nrmse)
count_nrmse <- count_rmse %>% select(-mse, -rmse) %>% rename(count_nrmse= nrmse)
nrmse <- left_join(var_nrmse, count_nrmse)
```

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

```
ggplot(nrmse) + geom_point(aes(x = var_nrmse, y = count_nrmse, color = norm_method, shape = pipe)) + th
```