Normalization Analysis

Nate Olson

2017-01-26

```
library(ProjectTemplate)
cwd <- getwd()
setwd("../")
load.project()
setwd(cwd)
```

Objective

- Assessment of count table value bias and variance for different pipelines and normalization methods.
- Bias was calculated as the differences between the observed and expectd count values.
- The overall pipeline and normalization method performance was compared between pipelines using Pearson's correlation coefficient.
- Variance in the count table values for an OTU was calculated as the variance between PCR replicates.

Code for analysis

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) %>%
      dplyr::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

NOTE Normalization and transformation order impacts results. What is the appropriate ordering?

TODO Move to lib

```
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 -
## %%TODO%% replace with ref based normalization Deseq - not tmm
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)
}
```

TODO move to src

```
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")
```

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),
            cv_count = sd(count)/mean(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))
```

Start of Count Table Value (Normalization Analysis)

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). Previous work has shown feature counts RNAseq technical replicates are Poisson distributions (REF), with equal mean and variance. Negative binomial is used to model the over dispersion in count values between biological replicates in differential expression methods such as DESeq. **Mean-Variance Conclusion:** Unlike RNAseq data, count data from technical replicate 16S rRNA metagenomics are not Poisson distributed.

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.

```
count_var_df %>% filter(norm_method == "raw") %>%
      group_by(norm_method, pipe) %>% mutate(max_coord = max(c(mean_count, var_count))) %>%
      ggplot() +
      geom_point(aes(x = max_coord, y = max_coord), alpha = 0) + #used to max plots square
      geom_hex(aes(x = mean_count, y = var_count )) +
      geom_smooth(aes(x = mean_count, y = var_count), color = "darkorange") +
      geom_abline(aes(intercept = 0, slope = 1), color = "grey40") +
      facet_wrap(~pipe) +
      scale_y_log10() + scale_x_log10() +
      theme_bw() +
      labs(x = "OTU-level Mean", y = "OTU-level Variance")
```

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

Variance Analysis using MA Plots

TODO

- MA plots between representative replicates pre and post
- Looking for technical shift from 0, have strong expectations as to what is will look like

- Additionally add other mixture combinations, e.g. pre and 2^-4: **NOTE** Not sure what Hector meant by this.

Impact of Normalization and Tranformation on Count Variance

NEXT STEP Present similar summary for impact of different normalization and transformations. Challenge due to differences in scales between pipelines and normalization methods.

Options

1. Only present summary statistic for the mean and variance relationship.- covariance, Senthil's difference in log space.

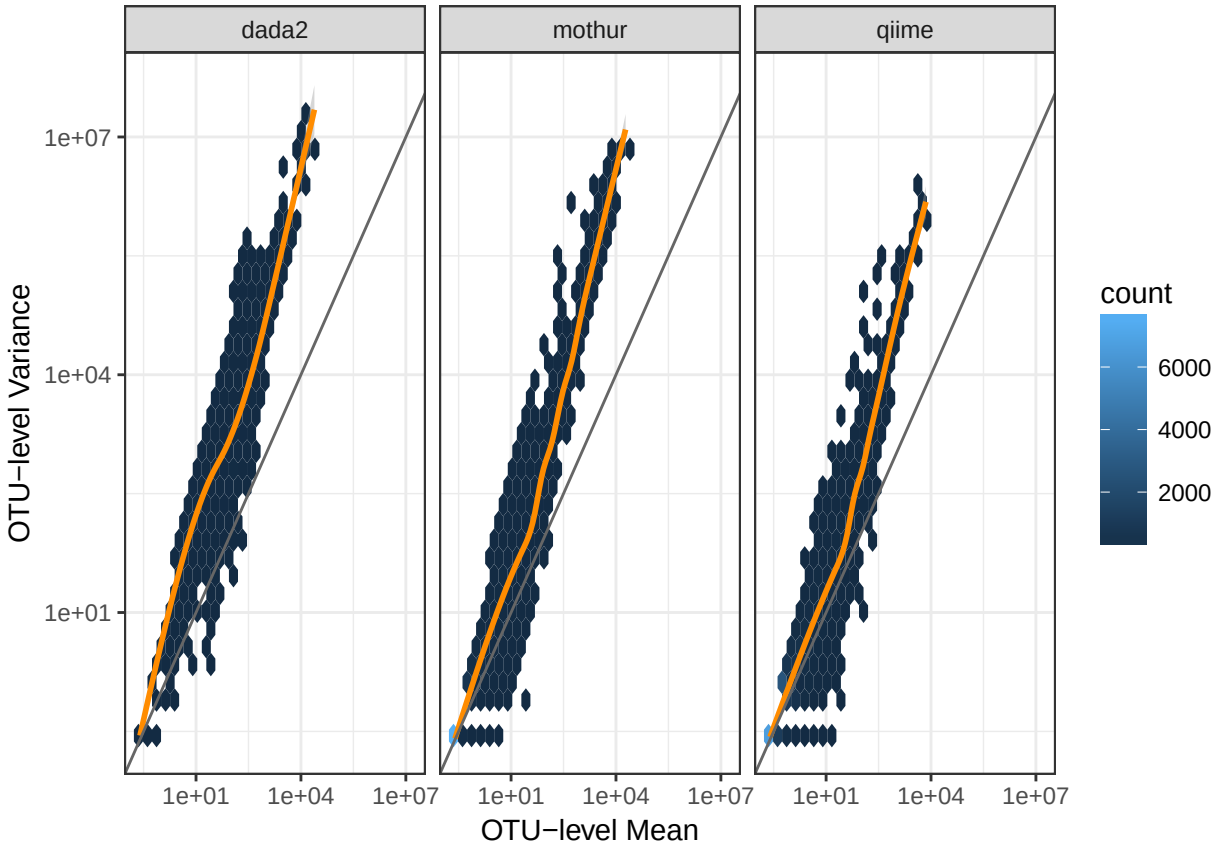


Figure 1: Comparison of different normalization methods on the relationship between OTU-level mean and variance for the four technical (PCR) replicates for the three pipeline. The grey line indicates the expected 1 to 1 mean-variance relationship for Poisson distributed data and the orange line is the ordered relationship as determined using a Generalized Additive Model smoothing spline.

2. Plot only of smoothing splines

3. regression for equality - splot 1 with intercept 0

Count Variance Summary Metric

Mean of the coefficient of variation ($CV = stdev(x)/mean(x)$) was used to compare overall count variance between pipelines and normalization methods.

```
var_cv <- count_var_df %>% ungroup() %>%
  group_by(pipe, norm_method) %>%
  summarise(mu_cv = mean(cv_count), med_cv = median(cv_count))
```

```
var_cv %>% select(-med_cv) %>% spread(pipe, mu_cv) %>% knitr::kable()
```

norm_method	dada2	mothur	qiime
css	1.191795	1.704983	1.525784
csslog	1.187224	1.700552	1.524663
dsq	1.245987	1.721058	1.558044
dsqlog	1.131914	1.672785	1.490293
raw	1.208252	1.709787	1.533664
rawlog	1.125221	1.669462	1.481008
tss	1.172724	1.698436	1.525358
tsslog	1.172589	1.698403	1.525331
uqs	1.179449	1.707211	1.525413
uqslog	1.178399	1.704297	1.524666

Median of the coefficient of variation ($CV = stdev(x)/mean(x)$) was used to compare overall count variance between pipelines and normalization methods.

```
var_cv %>% select(-mu_cv) %>% spread(pipe, med_cv) %>% knitr::kable()
```

norm_method	dada2	mothur	qiime
css	1.192613	2	2
csslog	1.191443	2	2
dsq	1.231192	2	2
dsqlog	1.161477	2	2
raw	1.204792	2	2
rawlog	1.158635	2	2
tss	1.190441	2	2
tsslog	1.190438	2	2
uqs	1.190072	2	2
uqslog	1.190039	2	2

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

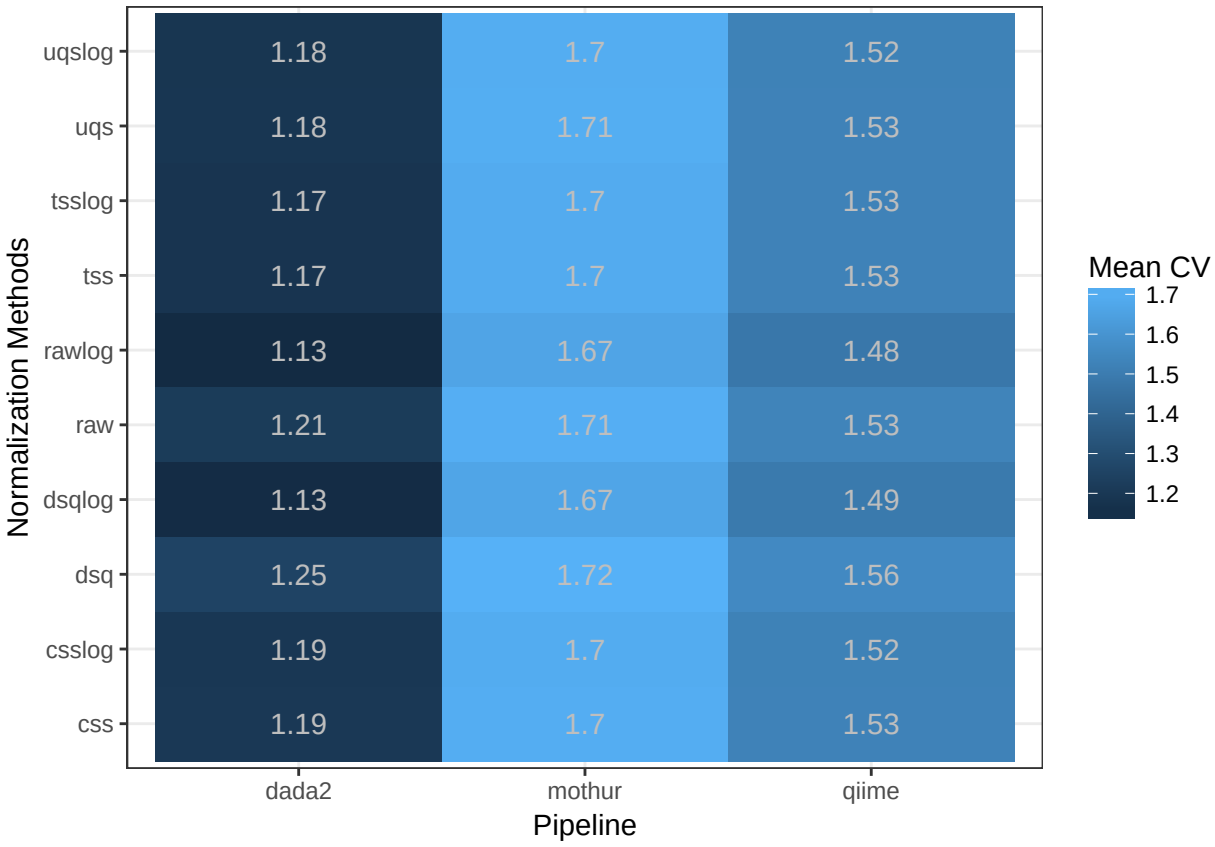


Figure 2: CV for pipelines and normalization methods providing an overall summary of the count variance. Lower values are better.

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) %>%
      dplyr::rename(pre = count) %>% select(-dilution, -samID)
post_count <- count_df %>% filter(dilution == 0) %>%
      dplyr::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")
```

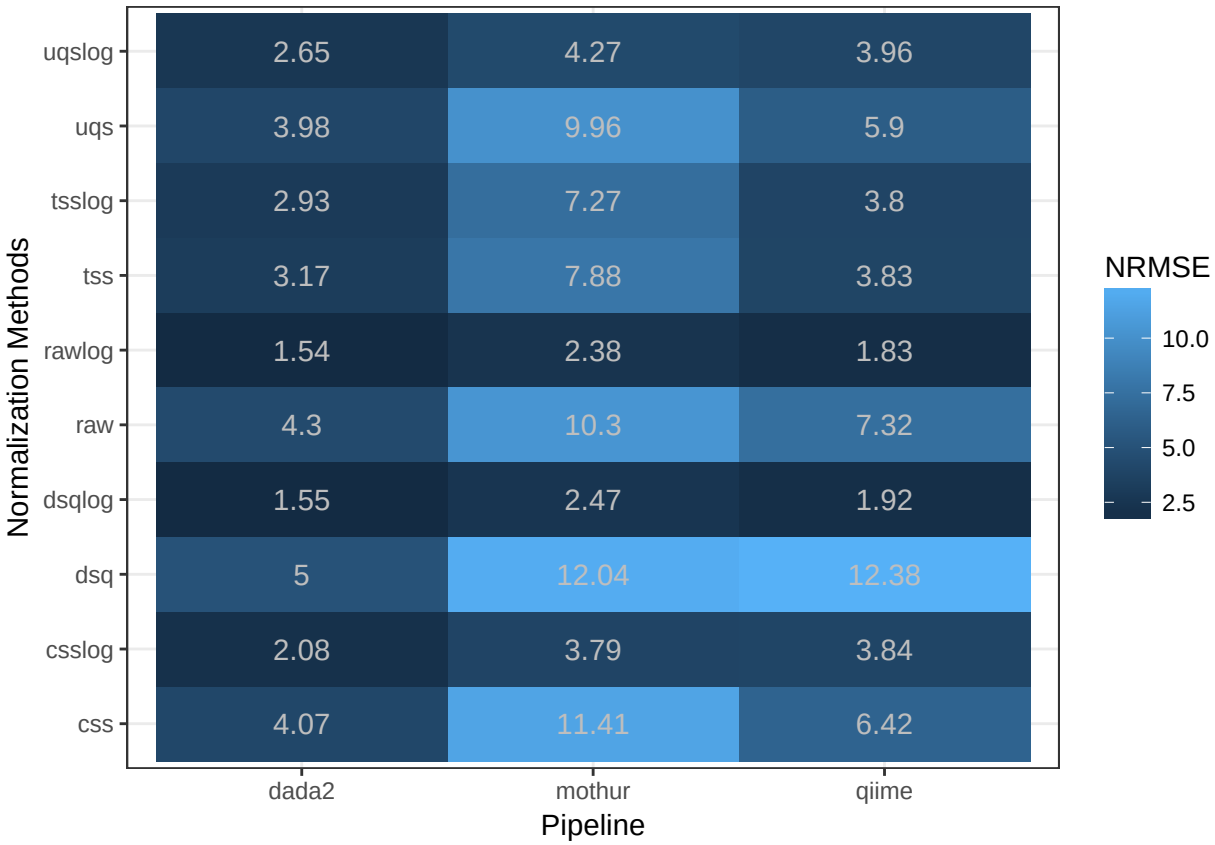
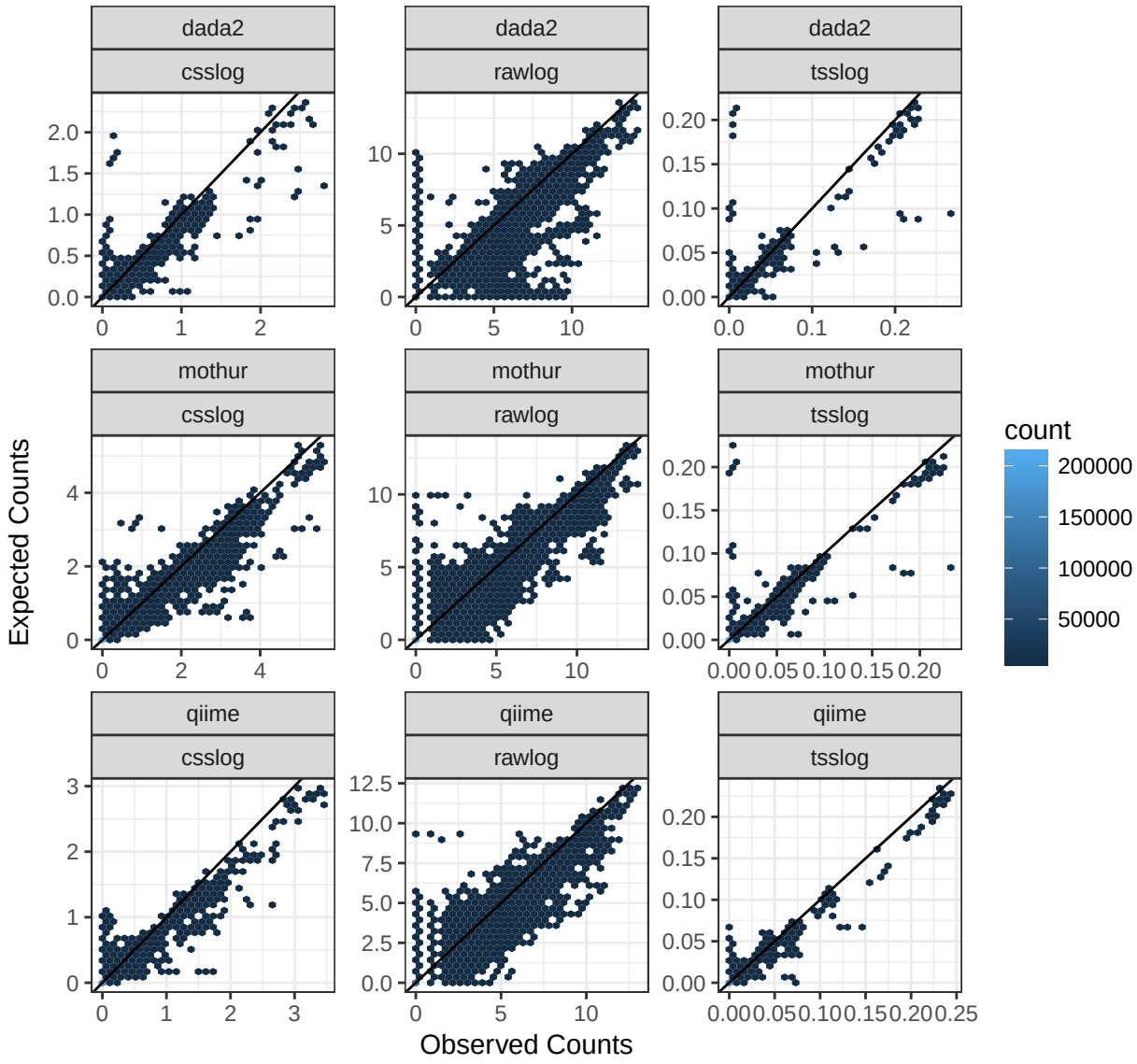


Figure 3: Normalized RMSE for pipelines and normalization methods providing an overall summary of the count bias. Lower values are better.



Variance-Bias Relationship

TODO Update with use of new metrics

```
# 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)
```

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

Feature Exploration

- Correlating factors such as well position, primer matching, and GC content with observed variance and bias.