Chi-Squared-Metric

Nate Olson

2017-04-13

Objective

- Use negative binomial to calculated expected count values instead of basing expected count on CSS normalized count values. - Negative binomial accounts for differences in sampling depths among PCR replicates.
- Observed count values have different variance between technical replicates due to differenes in sampling depths.

- Implement Chi-Square for Feature Assessment
- Previous metric implentations, specifically R^2, did not appropriately account for differences between titrations.
- Using the χ^2 metric account for differences between titrations.

Negative Binomal for Weighted Count Estimates

Calculating proportion of pre and post counts using negative binomial.

- $q_{i,j,k}$ is the proportion of feature i in PCR k of sample j where a sample is defined as an individual unmixed or mixed samples for a biological replicate.

- $p_{j,k}$ is the total feature abundance for sample j, sum of all feature counts not the number of sequences generated for the sample.

- $v_{i,j,k}$ is the variance of feature i in PCR replicate j of sample k.

$$v_{i,j,k} = \frac{q_{i,j,k}(1 - q_{i,j,k})}{p_j, k}$$

- $w_{i,j,k}$ is the weight function

$$w_{i,j,k} = \frac{v_{i,j,k}^{-1}}{\sum_{k \in j} v_{i,j,k}^{-1}}$$

- $q_{i,j}$ - the weighted count estimate for feature i, k

$$q_{i,j} = \sum_{k \in j} w_{i,j,k} q_{i,j,k}$$

Loading Data and Calculating Expected Values

```
## Extracting a tidy dataframe with count values from MRexpiment objects
get_count_df <- function(mrobj, agg_genus = FALSE, css = TRUE){
      if(agg_genus){
            mrobj <- aggregateByTaxonomy(mrobj, lvl = "Rank6",
                                         norm = FALSE, log = FALSE, sl = 1)
      }

      if(css == TRUE){
            mrobj <- cumNorm(mrobj, p = 0.75)
            count_mat <- MRcounts(mrobj, norm = TRUE, log = FALSE, sl = 1000)
      }else{
            count_mat <- MRcounts(mrobj, norm = FALSE, log = FALSE, sl = 1)
      }
       count_mat %>%
            as.data.frame() %>%
            rownames_to_column(var = "feature_id") %>%
            gather("id","count", -feature_id)
}

count_df <- mrexp %>% map_df(get_count_df, css = FALSE, .id = "pipe") %>%
      left_join(pData(mrexp$dada2)) %>%
      filter(biosample_id != "NTC", id != "1-F9") %>%
      select(pipe, biosample_id, id, pcr_rep, feature_id, t_fctr, count)

count_df <- count_df %>% group_by(id) %>% mutate(total_abu = sum(count))
```

Subsetting `count_df`

```
count_df <- count_df %>% filter(feature_id %in% c(paste0("SV", 1:3), paste0("Otu0000",1:3)))
```

Estimating q_i for pre and post

```
nb_est <- count_df %>% filter(t_fctr %in% c(0, 20)) %>%
      mutate(prop = count/total_abu,
             prop_var = (prop * (1 - prop))/total_abu,
             inv_var = 1/prop_var) %>%
      group_by(pipe, biosample_id, t_fctr, feature_id) %>%
      mutate(weight = inv_var / sum(inv_var)) %>%
      summarise(prop_est = sum(weight*prop))

# Reformatting data
pre_post_prop <- nb_est %>% ungroup() %>%
      mutate(treat = if_else(t_fctr == "20", "pre","post")) %>%
      select(-t_fctr) %>%
      mutate(prop_est = if_else(is.nan(prop_est), 0, prop_est)) %>%
      spread(treat, prop_est)
```

Calculating expected counts using proportion estimates

```
calc_expected_prop <- function(pre_post_prop){
      titration_list <- data_frame(titration = c(1:5,10,15)) %>%
            mutate(post_prop = 2^-titration) %>%
            list() %>% rep(nrow(pre_post_prop))

      pre_post_prop %>% ungroup() %>%
            add_column(titration = titration_list) %>% unnest() %>%
            mutate(exp_prop = post * post_prop + pre * (1-post_prop)) %>%
            mutate(t_fctr = factor(titration)) %>%
            select(-post_prop)
}
```

```r
exp_prop_df <- calc_expected_prop(pre_post_prop)
```

```r
exp_count_df <- count_df %>%
      filter(t_fctr %in% c(1:5, 10, 15)) %>%
      left_join(exp_prop_df) %>%
      mutate(exp_count = total_abu * exp_prop)
```

Chi-Squared Metric

Use expected count estimates based on negative binomial $\chi^2 = \sum \frac{(observed - expected)^2}{expected}$

First looked at unsquared χ^2 $\frac{(observed - expected)}{expected}$
Properties - value -1 when observed counts are 0 - large value when observed count >> expected counts - Undefined when expected and observed count are 0 - Inf when expected count is 0 but non-zero observed count

Issues - not symmetic where max negative value is -1 and max positive value is Inf - Not sure how to combine values

Calculating χ^2

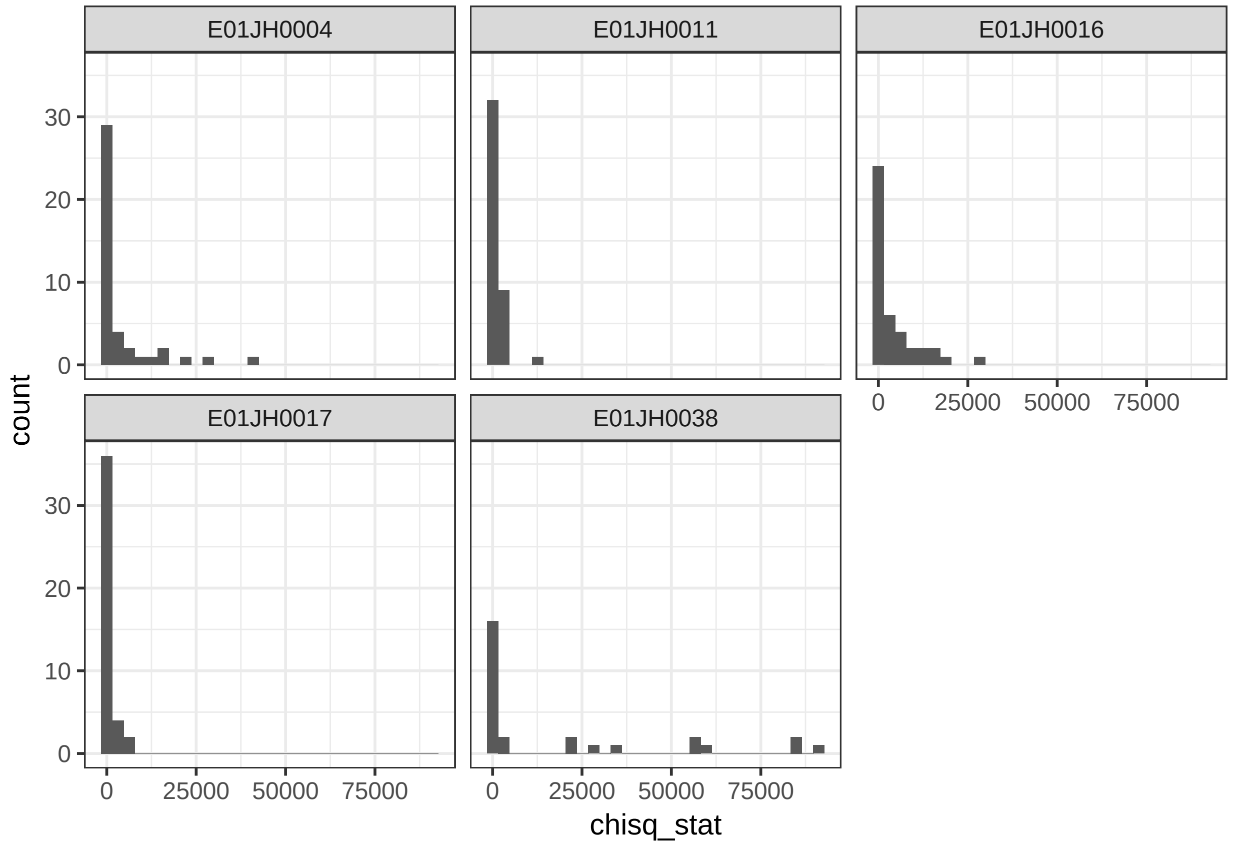
```r
chisq_df <- exp_count_df %>%
      filter(!(pre == 0 & post == 0)) %>%
      group_by(pipe, biosample_id, t_fctr, feature_id, pre, post) %>%
      summarise(chisq_stat = sum((count - exp_count)^2/exp_count))
```

```r
chisq_df$chisq_stat %>% summary()
```

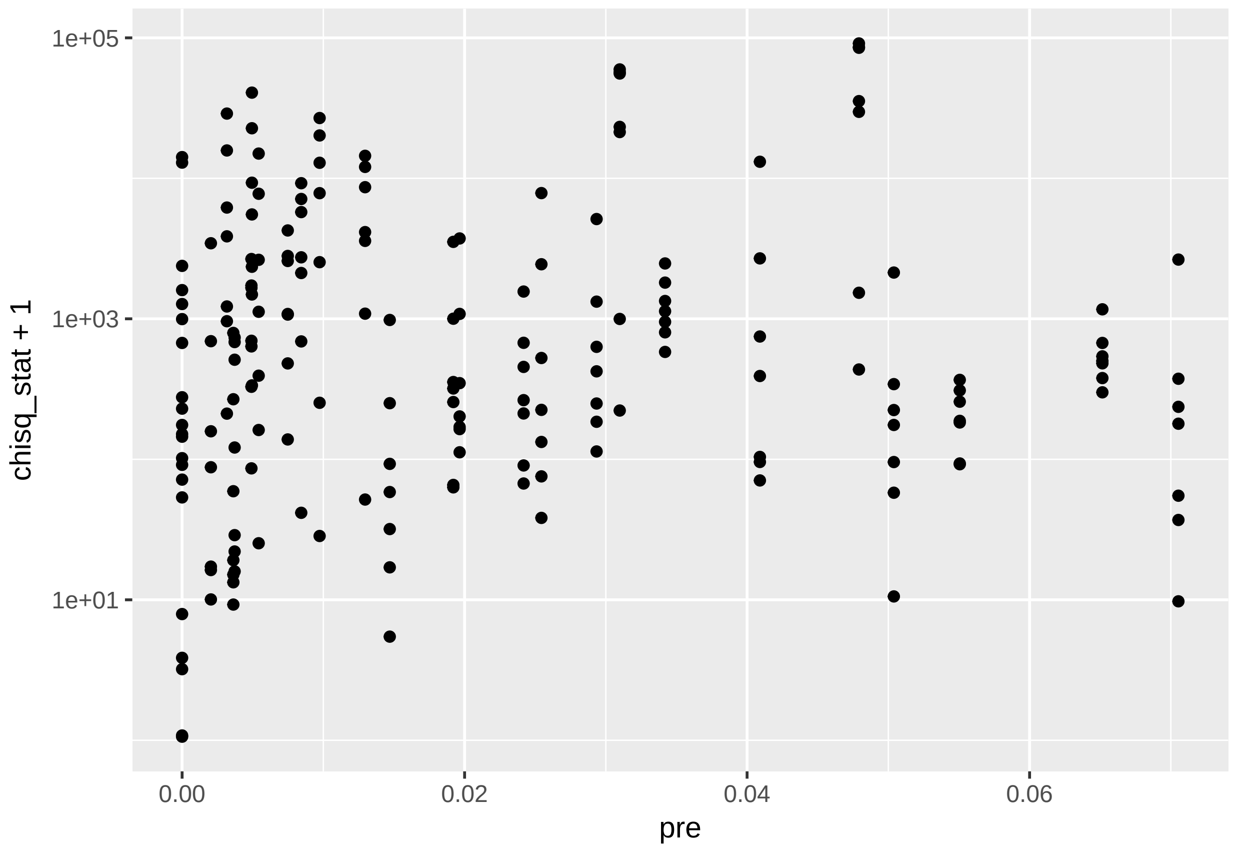
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.06   112.30   480.80  5068.00  2490.00 91220.00
```

Summary Plots

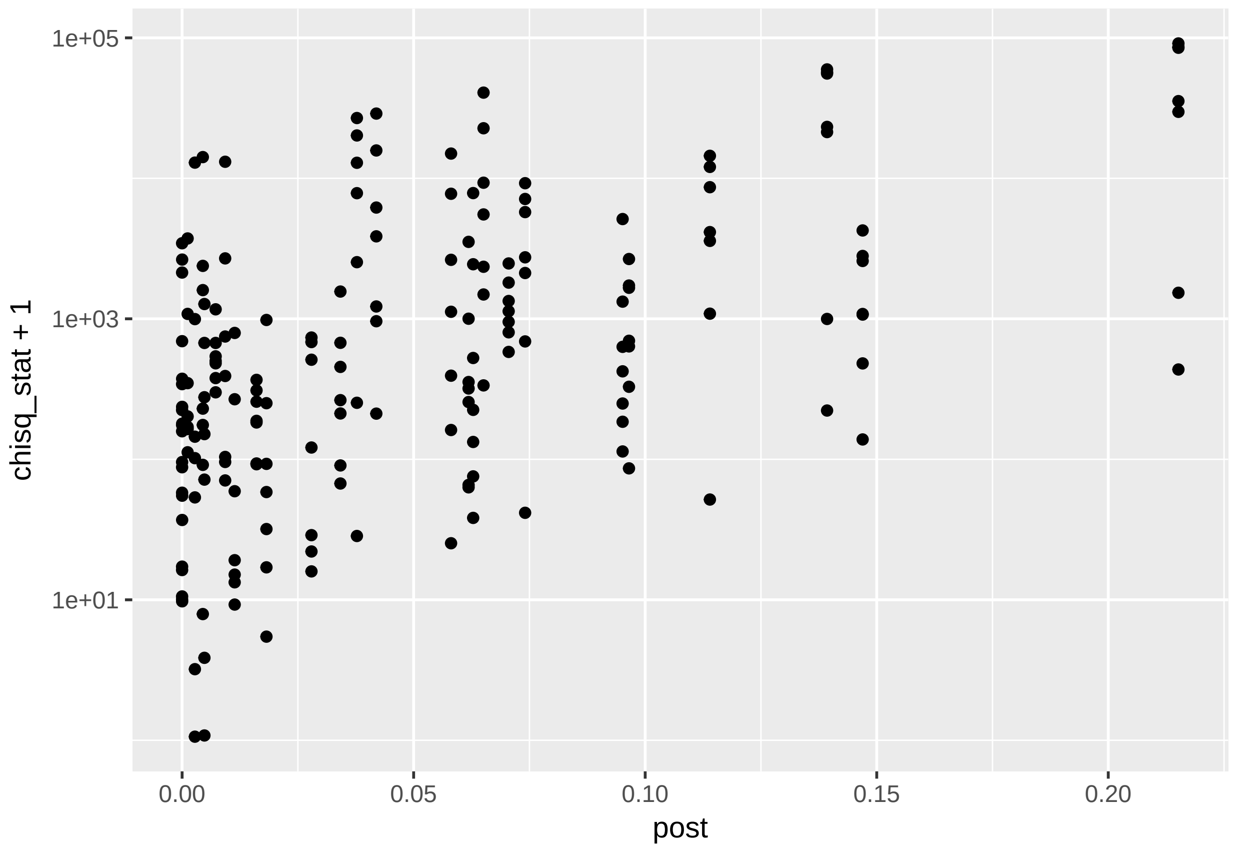
```r
chisq_df %>% ggplot() + geom_histogram(aes(x = chisq_stat)) +
      facet_wrap(~biosample_id) + theme_bw()
```



```r
ggplot(chisq_df) + geom_point(aes(x = pre, y = chisq_stat + 1)) + scale_y_log10()
```



```r
ggplot(chisq_df) + geom_point(aes(x = post, y = chisq_stat + 1)) + scale_y_log10()
```



Feature Level Summary

```
chi_feature <-chisq_df %>% group_by(pipe, biosample_id, feature_id) %>%
      summarise(total_chisq = sum(chisq_stat)) %>% arrange(total_chisq)
```

```
chi_feature$total_chisq %>% summary()
```

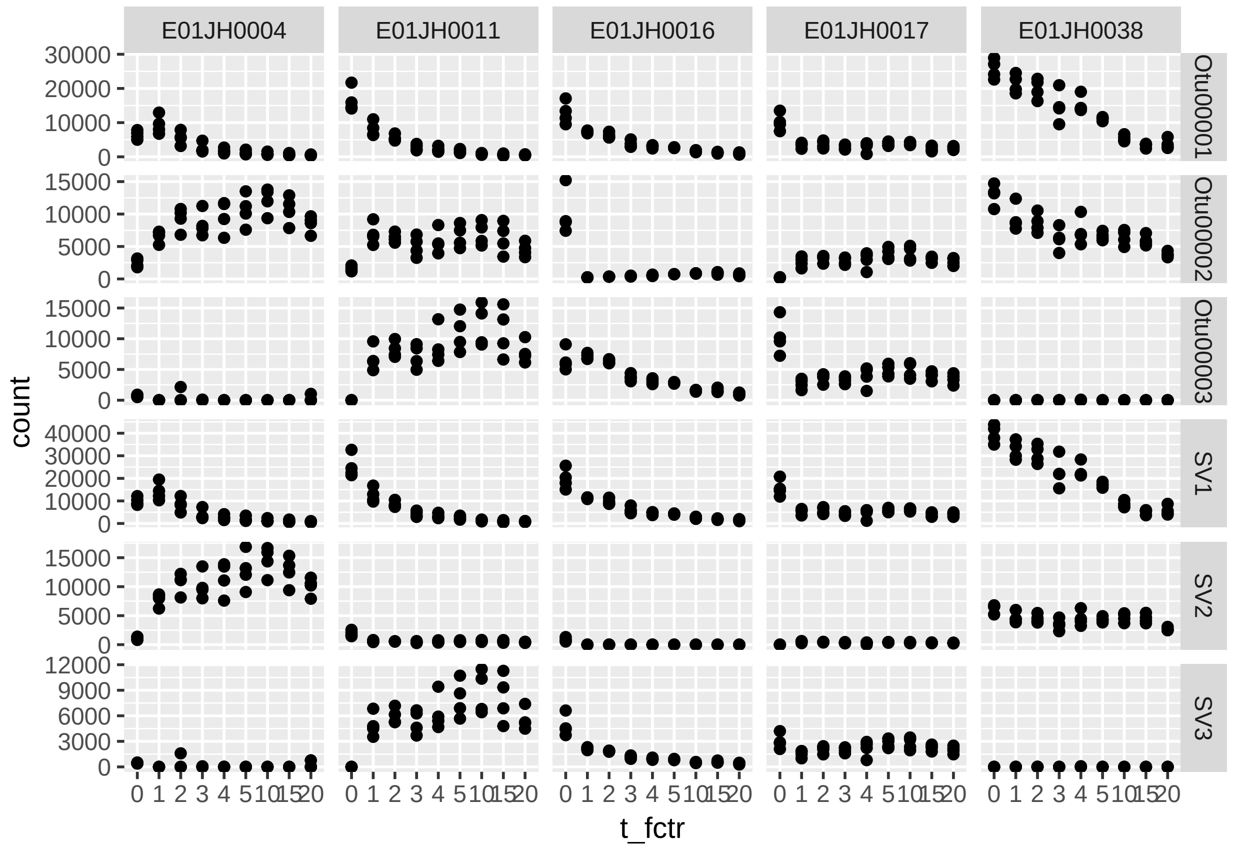
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1170    3473    8569   35480   27240  329300
```

```
chi_feature %>% ungroup() %>% select(-pipe) %>%
      mutate(total_chisq = round(total_chisq,0)) %>%
      spread(biosample_id, total_chisq) %>%
      knitr::kable()
```

feature_id	E01JH0004	E01JH0011	E01JH0016	E01JH0017	E01JH0038
Otu00001	57140	7795	27692	5582	218039
Otu00002	1483	17217	27095	5807	9088
Otu00003	18643	3525	70599	11287	NA
SV1	82456	12405	44034	8051	329287
SV2	4037	1170	2445	4426	3319
SV3	14227	3034	2112	1431	NA

```
count_df %>%
      mutate(t_fctr = factor(t_fctr, levels = c(0:5, 10, 15, 20))) %>%
```

```r
ggplot() +
geom_point(aes(x = t_fctr, y = count)) +
facet_grid(feature_id~biosample_id, scales = "free")
```


E01JH0004
E01JH0011
E01JH0016
E01JH0017
E01JH0038
Otu00001
Otu00002
Otu00003
SV1
SV2
SV3
count
t_fctr