Titration Abundance Expectation Calculation

Nate Olson

2017-04-04

TODO Add sanity checks for expected value calculations

Objective

Generate a data_frame with observed and predicted count values for individual PCR replicates. Expected values are calculated using unmixed PCR count table values from the same set of PCR replicates, same half of one of the replicate 96 well plates.

Functions for Processing Data

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

      mrobj <- cumNorm(mrobj, p = 0.75)
      mrobj %>%
            # not sure whether or not to normalize counts prior to analysis
            MRcounts(norm = TRUE, log = FALSE, sl = 1000) %>%
            #MRcounts(sl = 1) %>%
            as.data.frame() %>%
            rownames_to_column(var = "feature_id") %>%
            gather("id","count", -feature_id)
}
```

Extracting observed count values for unmixed pre and post samples

```
get_pre_post_counts <- function(count_df){
      pre_post_df <- count_df %>%
            filter(t_fctr %in% c(0, 20)) %>%
            mutate(pre_post = if_else(t_fctr == 0, "post_count", "pre_count"))

      ## sanity checks
      post_check <- pre_post_df %>% filter(pre_post == "post_count") %>% .$t_fctr %>% unique()
      if(length(post_check) != 1){
            print("post t_fctr not unique")
            paste0("Post: ", post_check) %>% print()
      }

      if(post_check != 0){
            print("post t_fctr not 20")
            paste0("Post t_fctr: ", post_check) %>% print()
      }
      pre_check <- pre_post_df %>% filter(pre_post == "pre_count") %>% .$t_fctr %>% unique()
      if(length(pre_check) != 1){
            print("pre t_fctr not unique")
            paste0("Pre: ", pre_check) %>% print()
      }

      if(pre_check != 20){
            print("pre t_fctr not 20")
            paste0("Pre t_fctr: ", pre_check) %>% print()
      }
      pre_post_df %>%
            select(pipe, biosample_id, feature_id, pcr_rep, count, pre_post) %>%
            spread(pre_post, count)

}
```

Calculating expected counts

```
calc_expected <- function(count_df, pre_post_counts){

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

      pre_post_counts %>% ungroup() %>%
            add_column(titration = titration_list) %>% unnest() %>%
            mutate(exp_count = post_count * post_prop + pre_count * (1-post_prop))
}
```

Generating Expected Count Data Frames

One data frame for feature level analysis and one when aggregating at the genus level.

Feature Level

```

count_df <- mrexp %>% map_df(get_count_df, .id = "pipe")

## Annotate with metadata and exclude NTCs
count_df <- pData(mrexp$dada2) %>% right_join(count_df) %>%
      filter(biosample_id != "NTC")
```

Adding expected count values to observed count value data frame and calculating residuals.

```
pre_post_counts <- get_pre_post_counts(count_df)
exp_count <- calc_expected(count_df, pre_post_counts)



## annotating data frame
count_exp_df <- left_join(exp_count, count_df) %>% dplyr::rename(obs_count = count)
```

```
count_exp_df %>% saveRDS("../data/expected_count_values_feature_df.rds")
```

Genus Level

```
count_df <- mrexp %>% map_df(get_count_df, agg_genus = TRUE, .id = "pipe")

## Annotate with metadata and exclude NTCs
count_df <- pData(mrexp$dada2) %>% right_join(count_df) %>%
      filter(biosample_id != "NTC")
```

Adding expected count values to observed count value data frame and calculating residuals.

```
pre_post_counts <- get_pre_post_counts(count_df)
exp_count <- calc_expected(count_df, pre_post_counts)

## annotating data frame
count_exp_df <- left_join(exp_count, count_df) %>% dplyr::rename(obs_count = count)
```

```
glimpse(count_exp_df)
```

```
## Observations: 71,540
## Variables: 15
## $ pipe          <chr> "dada2", "dada2", "dada2", "dada2", "dada2", "da...
## $ biosample_id  <chr> "E01JH0004", "E01JH0004", "E01JH0004", "E01JH000...
## $ feature_id    <chr> "[Eubacterium]_brachy_group", "[Eubacterium]_bra...
## $ pcr_rep       <chr> "1:1", "1:1", "1:1", "1:1", "1:1", "1:1", "1:1",...
## $ post_count    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ pre_count     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ titration     <dbl> 1, 2, 3, 4, 5, 10, 15, 1, 2, 3, 4, 5, 10, 15, 1,...
## $ post_prop     <dbl> 5.000000e-01, 2.500000e-01, 1.250000e-01, 6.2500...
## $ exp_count     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ t_fctr        <fctr> 1, 2, 3, 4, 5, 10, 15, 1, 2, 3, 4, 5, 10, 15, 1...
## $ id            <chr> "1-B1", "1-C1", "1-D1", "1-E1", "1-F1", "1-G1", ...
## $ pcr_16S_plate <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, ...
## $ pos           <chr> "B1", "C1", "D1", "E1", "F1", "G1", "H1", "B7", ...
## $ pcr_half      <chr> "1", "1", "1", "1", "1", "1", "1", "2", "2", "2"...
## $ obs_count     <dbl> 0.6935500, 0.0000000, 0.0000000, 0.0000000, 0.55...
```

```
count_exp_df %>% saveRDS("../data/expected_count_values_genus_df.rds")
```

Session information

```
s_info <- devtools::session_info()
print(s_info$platform)
```

```
##  setting  value
##  version  R version 3.3.3 (2017-03-06)
##  system   x86_64, darwin15.6.0
##  ui       unknown
##  language (EN)
##  collate  en_US.UTF-8
##  tz       America/New_York
##  date     2017-04-04
```

```
s_info$packages %>% filter(`*` == "*") %>% select(-`*`) %>%
      knitr::kable()
```

package	version	date	source
bbmle	1.0.18	2016-02-11	CRAN (R 3.3.2)
Biobase	2.34.0	2016-11-07	Bioconductor
BiocGenerics	0.20.0	2016-11-07	Bioconductor
BiocParallel	1.8.1	2016-11-07	Bioconductor
Biostrings	2.42.1	2016-12-19	Bioconductor
DESeq	1.26.0	2016-11-28	Bioconductor
DESeq2	1.15.28	2017-02-02	bioc (readonly/DESeq2@125913)
dplyr	0.5.0	2016-06-24	CRAN (R 3.3.2)
edgeR	3.16.5	2017-02-02	Bioconductor
forcats	0.2.0	2017-01-23	CRAN (R 3.3.2)
foreach	1.4.3	2015-10-13	CRAN (R 3.3.1)
GenomeInfoDb	1.10.3	2017-03-28	Bioconductor
GenomicAlignments	1.10.1	2017-03-28	Bioconductor
GenomicRanges	1.26.4	2017-03-28	Bioconductor
ggplot2	2.2.1	2016-12-30	CRAN (R 3.3.2)
glmnet	2.0-5	2016-03-17	CRAN (R 3.3.1)
IRanges	2.8.2	2017-03-28	Bioconductor
knitr	1.15.1	2016-11-22	CRAN (R 3.3.2)
lattice	0.20-34	2016-09-06	CRAN (R 3.3.3)
limma	3.30.13	2017-03-28	Bioconductor
locfit	1.5-9.1	2013-04-20	CRAN (R 3.3.1)
Matrix	1.2-8	2017-01-20	CRAN (R 3.3.3)
metagenomeSeq	1.16.0	2016-11-07	Bioconductor
modelr	0.1.0	2016-08-31	cran (@0.1.0)
permute	0.9-4	2016-09-09	CRAN (R 3.3.1)
phyloseq	1.19.1	2017-01-04	Bioconductor
ProjectTemplate	0.7	2016-08-11	CRAN (R 3.3.1)
purrr	0.2.2	2016-06-18	CRAN (R 3.3.1)
RColorBrewer	1.1-2	2014-12-07	CRAN (R 3.3.1)
readr	1.1.0	2017-03-22	CRAN (R 3.3.2)
readxl	0.1.1	2016-03-28	cran (@0.1.1)
Rqc	1.8.0	2016-11-07	Bioconductor
Rsamtools	1.26.1	2016-11-07	Bioconductor
S4Vectors	0.12.2	2017-03-28	Bioconductor
sads	0.3.1	2016-05-13	CRAN (R 3.3.2)
savR	1.12.0	2016-11-07	Bioconductor
ShortRead	1.32.1	2017-03-28	Bioconductor
stringr	1.2.0	2017-02-18	CRAN (R 3.3.2)
SummarizedExperiment	1.4.0	2016-11-07	Bioconductor
tibble	1.2	2016-08-26	CRAN (R 3.3.1)
tidyr	0.6.1	2017-01-10	CRAN (R 3.3.2)
tidyverse	1.1.1	2017-01-27	CRAN (R 3.3.2)
vegan	2.4-2	2017-01-17	CRAN (R 3.3.2)
XVector	0.14.1	2017-03-28	Bioconductor