Benchmark of error rate inference from UMI-tagged data and PCR error model

Mikhail Shugay

November 6, 2016

Training the model

Some auxiliary functions

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(RColorBrewer)
library(stringr)
library(nloptr)
library(scales)

select <- dplyr::select

mtypes <- data.frame(mutation.fromto = c("A>C","A>G","A>T","C>A","C>G","C>T","G>A",
                                         "G>C","G>T","T>A","T>C","T>G"),
                     mutation.type = c("A>C,T>G","A>G,T>C","A>T,T>A","C>A,G>T",
                                       "C>G,G>C","C>T,G>A","C>T,G>A","C>G,G>C",
                                       "C>A,G>T","A>T,T>A","A>G,T>C","A>C,T>G"))

read_variant_table <- function(file_name, pos_filter = function(x) T, count_filter = function(x) T) {
  .df <- read.table(file_name, header=T, sep="\t", stringsAsFactors = F)
  .df <- subset(.df, !grepl("^[DI]", mutation) & coverage > 100 & freq < 0.45)

  .df$mutation.fromto <- unlist(lapply(str_split(.df$mutation, ":"), function(x) x[2]))
  .df$mutation.pos <- as.integer(unlist(lapply(str_split(.df$mutation, ":"), function(x) str_sub(x[1],
  .df <- merge(.df, mtypes)
  .df %>% filter(pos_filter(mutation.pos) & count_filter(count)) %>% select(mutation.pos, mutation.type
}
```

Load polymerase data. At this stage a single event is a combination of substituted nucleotide, its substitution, position in template, sample (polymerase) and project (replica). Samples are obtained by grouping all events by substitution type which is one of 6 from and to nucleotide combinations that can be observed when not knowing the exact strand at which a given error has happened. Note that the latter is done because in most practical applications this information is hard to obtain.

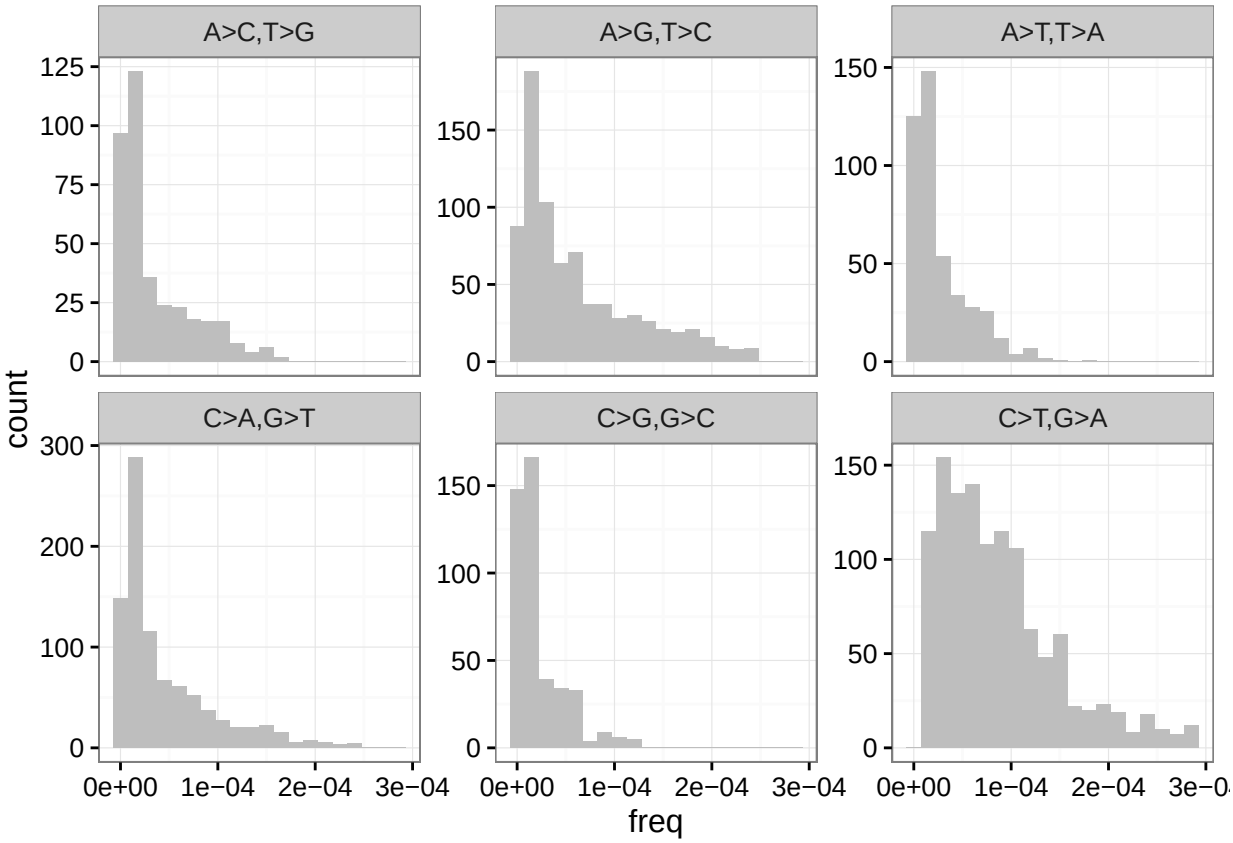
```
df <- data.frame()

for (proj in c("73", "82")) {
  for (sample in c("encyclo", "kappa-hf-taq", "phusion", "sd-hs", "snp-detect",
                   "taq-hs", "tersus", "tersus-snp-buff", "truseq", "velox")) {
    file_name <- paste("data/25_8_polerr", proj, ".", sample, ".variant.caller.txt", sep = "")
    .df <- read_variant_table(file_name,
                              count_filter=function(x) x>0)
    .df$proj <- proj
    .df$sample <- sample
    df <- rbind(df, .df)
  }
}
```

Plot error frequencies grouped by substitution type

```
df.1 <- df %>%
  select(mutation.type, count, coverage) %>%
  mutate(freq = count / coverage) %>%
  # winsorize data
  group_by(mutation.type) %>%
  mutate(q5 = quantile(freq, 0.05), q95 = quantile(freq, 0.95)) %>%
  filter(freq >= q5 & freq <= q95) %>%
  select(mutation.type, freq)

ggplot(df.1, aes(x=freq)) + geom_histogram(fill="grey", bins=20) +
  facet_wrap(~mutation.type, scales="free_y") + theme_bw()
```



Fit frequency distributions with a Gamma model. We are fitting frequencies as otherwise we'll run in the need to simultaneously handle the coverage parameter.

```
gammanll <- function(pars, data){
  alpha <- pars[[1]]
  beta <- pars[[2]]
  return (-sum(dgamma(data, shape = alpha, rate = beta, log = T)))
}

param_guess = function(data) {
  m = mean(data)
  v = sd(data) ^ 2
  c(m * m / v, m / v)
}

fit_gamma <- function(data) {
  list(nloptr(x0 = param_guess(data), eval_f = gammanll, lb = c(0,0), data = data,
         opts = list(algorithm = "NLOPT_LN_SBPLX", maxeval = 1e5))$solution)
}

fit.params <- df.1 %>%
  group_by(mutation.type) %>%
  summarize(fit = fit_gamma(freq))

# theta parametrization more intuitive

fit.params$alpha <- unlist(lapply(fit.params$fit, function(x) x[1]))
fit.params$theta <- unlist(lapply(fit.params$fit, function(x) 1/ x[2]))
fit.params$fit <- NULL
```

Gamma distribution parameter estimates:

```
print(fit.params)
```

```
## # A tibble: 6 × 3
##   mutation.type     alpha        theta
##           <fctr>     <dbl>        <dbl>
## 1       A>C,T>G 0.9279161 3.727861e-05
## 2       A>G,T>C 1.0296172 6.099085e-05
## 3       A>T,T>A 1.0838314 2.565636e-05
## 4       C>A,G>T 0.9957468 4.573685e-05
## 5       C>G,G>C 1.0584827 2.081932e-05
## 6       C>T,G>A 2.1316472 4.106974e-05
```

Checking goodness of fit. Note that the missing values near 0 can be attributed to sampling effect under finite coverage.

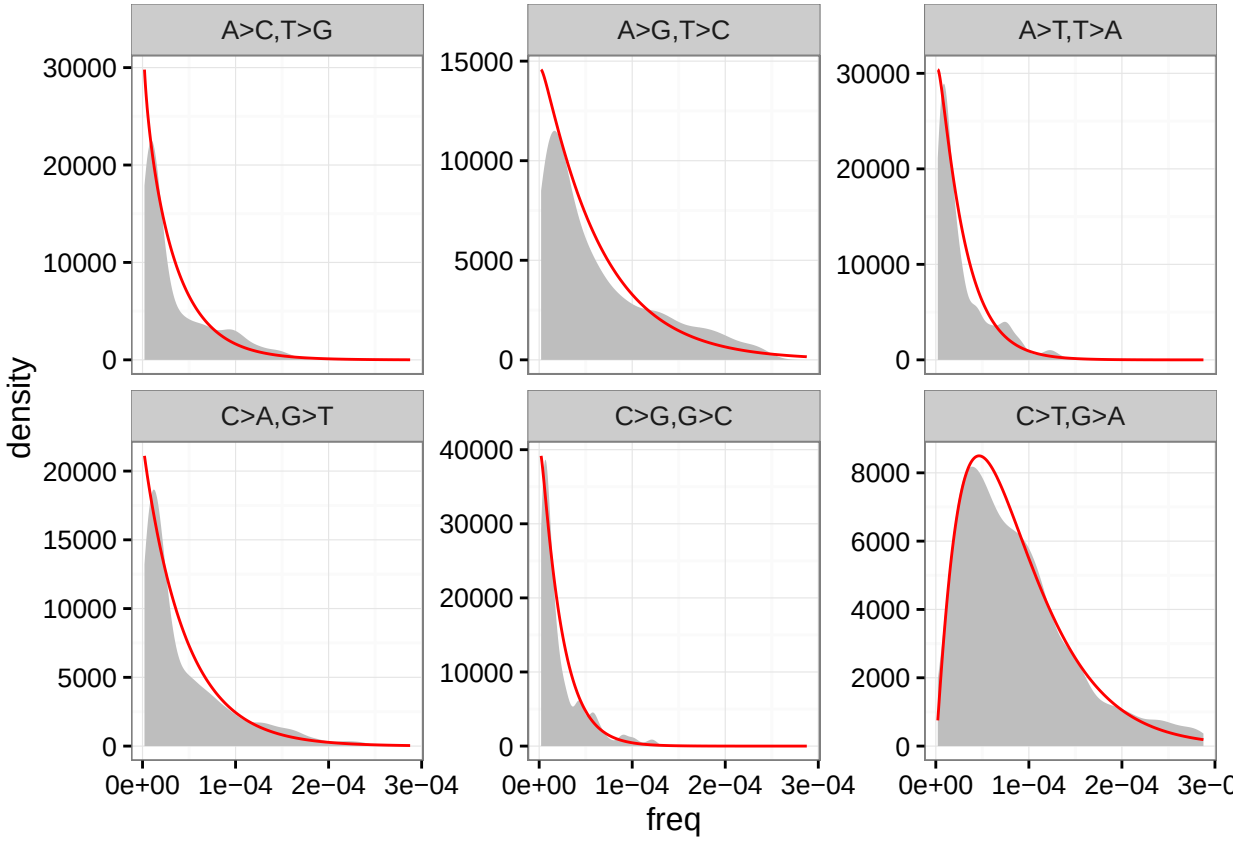
```
df.1.fit <- expand.grid(mutation.type = unique(df.1$mutation.type),
                        freq = seq(min(df.1$freq), max(df.1$freq), length.out = 1000))

df.1.fit <- merge(df.1.fit, fit.params)

df.1.fit$density <- with(df.1.fit, dgamma(freq, shape = alpha, scale = theta))


ggplot(data=df.1, aes(x=freq)) +
```

```
  geom_density(fill="grey", color=NA) +
  geom_line(data=df.1.fit, aes(y=density), color="red") +
  facet_wrap(~mutation.type, scales="free_y") +
  theme_bw()
```



The frequency of errors f for each subsitution type group T is fitted with $Gamma$ distribution:

$$f \sim Gamma(\alpha, \theta|T)$$

Thus the expected number of mutations λ at a coverage N is distributed as

$$\lambda = Nf \sim Gamma(\alpha, \theta N|T)$$

And the final model

$$n \sim NegBinom(\alpha, \frac{\theta N}{1+\theta N}|T)$$

Benchmark using control sample

Load control data from amplicon sequencing of control donor DNA

```
df.control <- data.frame()
for (r in 1:2) {
  for (m in 1:4) {
    file_name <- paste("data/p127.ballast", r, "_m", m, ".variant.caller.txt", sep = "")
```

```
    df.control <- rbind(df.control, read_variant_table(file_name))
  }
}
```

Observed and fitted histogram of error frequencies

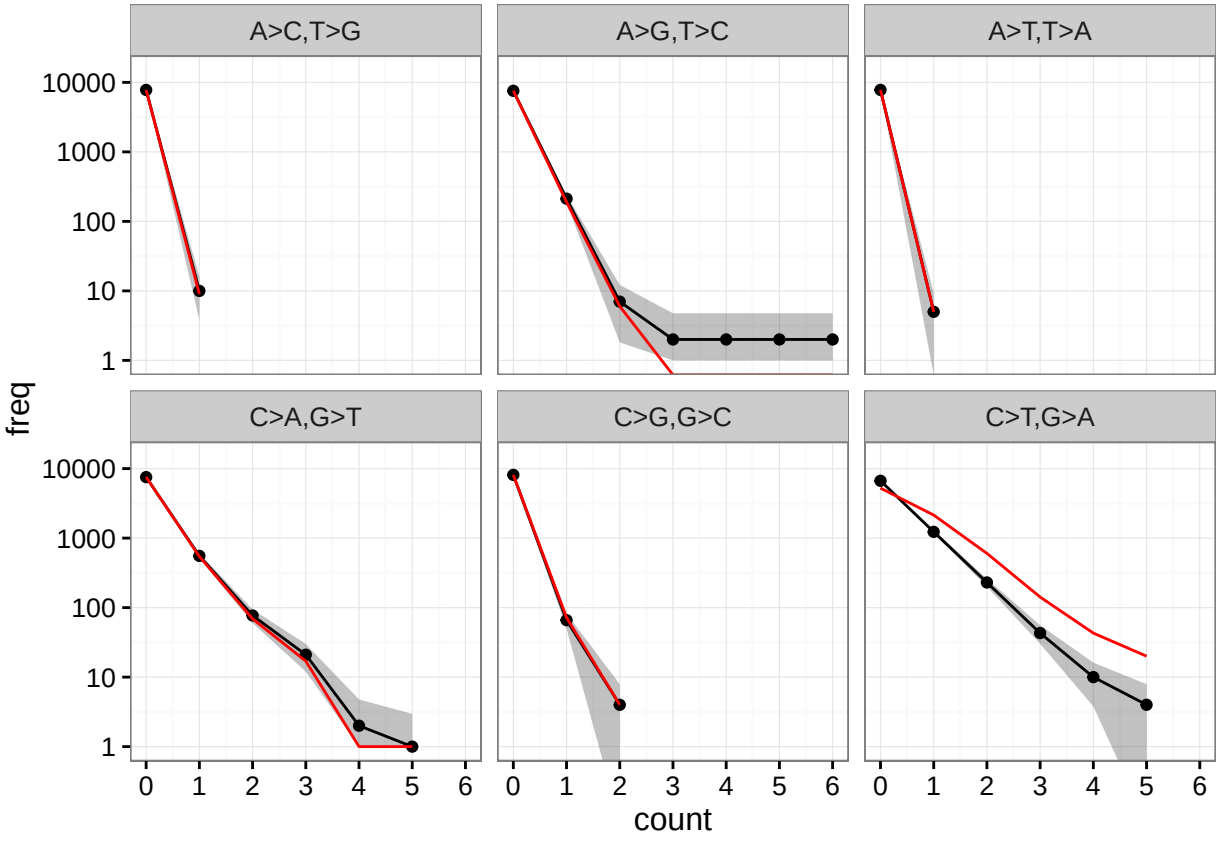
```
alpha <- fit.params$alpha
names(alpha) <- fit.params$mutation.type
theta <- fit.params$theta
names(theta) <- names(alpha)

compute_density <- function(count, coverage, mutation.type) {
  mapply(function(x,y,z) dnbinom(x, size = y, prob = z / (1 + z)),
         count, alpha[mutation.type], theta[mutation.type] * coverage)
}

df.control.1 <- df.control %>%
  mutate(fit = compute_density(count, coverage, mutation.type)) %>%
  group_by(count, mutation.type) %>%
  summarise(freq = length(count), freq.fit = sum(fit)) %>%
  group_by(mutation.type) %>%
  mutate(freq.fit = freq.fit / sum(freq.fit) * sum(freq))

ggplot(df.control.1, aes(x = count)) +
  geom_ribbon(aes(ymin=ifelse(freq - 1.96*sqrt(freq) <= 0, 1, freq - 1.96*sqrt(freq)),
                  ymax=freq + 1.96*sqrt(freq)), alpha=0.3) +
  geom_line(aes(y=freq))  +
  geom_point(aes(y=freq))  +
  geom_line(aes(y=round(freq.fit)), color="red") +
  scale_x_continuous(limits=c(0,6), breaks = 0:6) +
  scale_y_log10(limits=c(1, 15000), breaks=10^(0:4), oob = rescale_none) +
  facet_wrap(~mutation.type) +
  theme_bw()
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```


A>C,T>G
A>G,T>C
A>T,T>A
C>A,G>T
C>G,G>C
C>T,G>A
freq
10000
1000
100
10
1
0
1
2
3
4
5
6
count