MAGERI benchmark using reference standard DNA library

Mikhail Shugay

December 1, 2016

Load metadata

```
df.vmeta <- read.table("hd734_variant_metadata.txt", sep="\t", header=T) # variants observed in HD734 a
df.smeta <- read.table("sample_metadata.txt", sep="\t", header=T) # metadata for amplicon sequencing sa
```

VCF parsing function

```
library(stringr)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)
library(ggbeeswarm)
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
```

```
read_vcf <- function(file_name) {
  .vcf <- read.table(file_name, header = F, sep = "\t", stringsAsFactors = F)
  colnames(.vcf) <- c("chromosome", "position", "skip1", "from", "to", "qual", "skip2", "info", "skip3"
  .vcf$skip1 <- NULL
  .vcf$skip2 <- NULL
  .vcf$skip3 <- NULL
  .vcf$skip4 <- NULL

  .vcf$qual <- as.integer(.vcf$qual)
  .vcf <- subset(.vcf, nchar(from) == 1 &
                   nchar(to) == 1 & !is.na(qual)) # no indels

  .infosplit <- str_split_fixed(.vcf$info, regex("[=;]"), 14)[,c(2, 4, 12, 14)]

  .vcf$coverage <- as.numeric(.infosplit[,1])
```

```
  .vcf$frequency <- as.numeric(.infosplit[,2])
  .vcf$bb.a <- as.numeric(.infosplit[,3])
  .vcf$bb.b <- as.numeric(.infosplit[,4])
  .vcf$info <- NULL

  .vcf$count <- as.integer(.vcf$coverage * .vcf$frequency)
  .vcf$qual <- as.integer(.vcf$qual)

  subset(.vcf, nchar(from) == 1 & nchar(to) == 1 & !is.na(qual))
}

head(read_vcf("p126.h4_2_ballast_m1.vcf"))
```

```
## Warning in read_vcf("p126.h4_2_ballast_m1.vcf"): NAs introduced by coercion
```

```
##   chromosome  position from to qual coverage    frequency      bb.a
## 1       chr2 212295704    C  A    8     2265 0.0004415011 0.9957058
## 2       chr2 212295705    C  A    8     2265 0.0004415011 0.9957058
## 3       chr2 212295713    A  G    7     2265 0.0004415011 1.0295689
## 4       chr2 212295718    G  A    6     2265 0.0004415011 2.1314828
## 5       chr2 212295725    C  A    8     2264 0.0004416961 0.9957058
## 6       chr2 212295732    C  T   24     2264 0.0013250883 2.1314828
##       bb.b count
## 1 21862.23     0
## 2 21862.23     0
## 3 16394.12     0
## 4 24344.82     0
## 5 21862.23     1
## 6 24344.82     2
```

Read samples with HD734 standard DNA and control human DNA, append metadata

```
library(TailRank) # For betabinom.
```

```
## Loading required package: oompaBase
```

```
# installing:
# source("https://bioconductor.org/biocLite.R")
# biocLite("Biobase")
# install.packages("TailRank", repos="http://R-Forge.R-project.org")


df <- data.frame()

read_vcf_with_metadata <- function(file_name, primer_set, replica, ratio, type) {
  .vcf <- read_vcf(file_name)
  .vcf <- merge(.vcf, df.vmeta, all.x = type != "standard", all.y = F)
  .vcf$known.frequency <- .vcf$known.frequency * ratio
  .vcf$known.frequency[is.na(.vcf$known.frequency)] <- 0
  .vcf$primer_set <- primer_set
  .vcf$replica <- primer_set
  .vcf$type <- type

  .vcf <- subset(.vcf, frequency < 0.4 & count > 0) # remove alleles in control

  .vcf
```

```
}

for (i in 1:nrow(df.smeta)) {
  df <- with(df.smeta, rbind(df,
                             read_vcf_with_metadata(paste(prefix[i], "vcf", sep="."),
                                                    primer_set[i],
                                                    replica[i],
                                                    ratio[i],
                                                    type[i])))
}
```

```
## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion

## Warning in read_vcf(file_name): NAs introduced by coercion
```

```
# Compute scores

#df$qual <- with(df,
#  mapply(function(x,y,a,b) 1 - pbb(x,y,a,b) + 0.5 * dbb(x,y,a,b),
#         count, coverage, bb.a, bb.b)
```

```
#  )

#df$qual <- ifelse(df$qual <= 1e-100, 999, -10*log10(df$qual))
```

Group observed variants into tiers

```
df$tier <- cut(df$known.frequency, c(-1, 0, 0.009, 0.02, 1))
levels(df$tier) <- c("error", "0.1%", "1%", "5%")
summary(df$tier)
```

```
## error  0.1%    1%    5%
##  1467    42    46    12
```

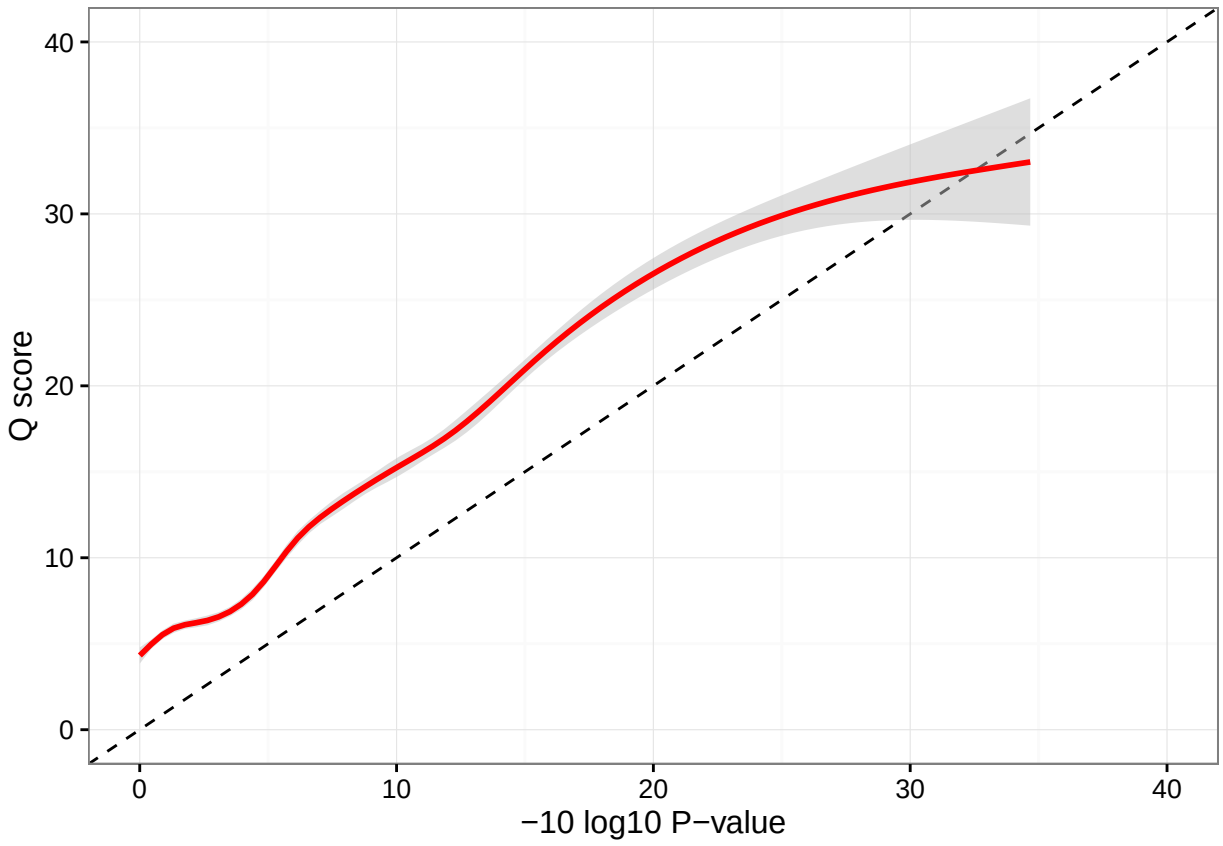
Quality score and error P-values

```
df.e <- df %>%
  filter(type == "blank") %>%
  mutate(true.p.value = -10 * log10(1 - (rank(frequency) + 0.5) / n()))
```

```
## Warning in eval(substitute(expr), envir, enclos): NaNs produced
```

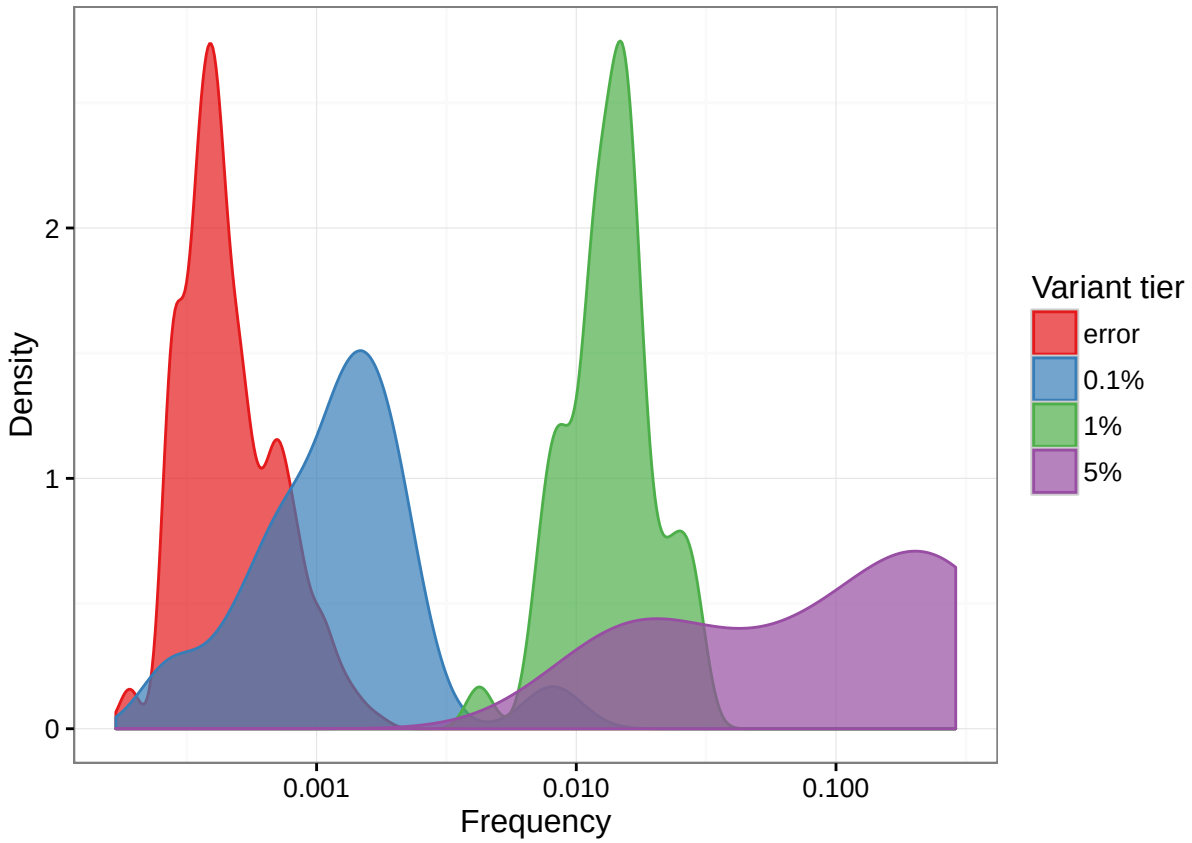
```
ggplot(df.e, aes(true.p.value, qual)) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  geom_smooth(color="red", fill="grey", alpha=0.5) +
  scale_x_continuous("-10 log10 P-value", limits = c(0, 40)) +
  scale_y_continuous("Q score", limits = c(0, 40)) +
  theme_bw()
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```



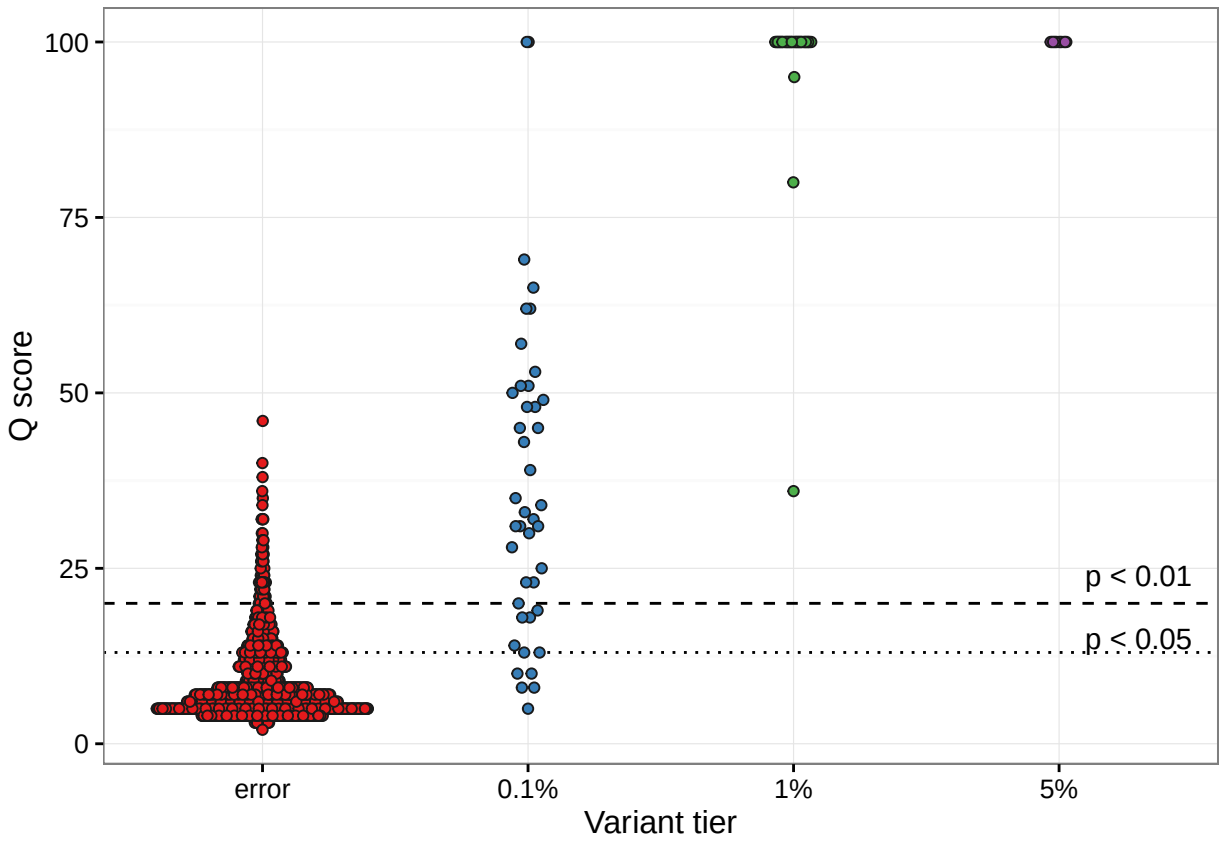
Frequency of variants and errors

```
ggplot(df, aes(x=frequency, color = tier)) +
  geom_density(aes(fill=tier), alpha=0.7) +
  ylab("Density") +
  scale_x_log10("Frequency") +
  scale_color_brewer("Variant tier", palette = "Set1") +
  scale_fill_brewer("Variant tier", palette = "Set1") +
  theme_bw()
```



Variant quality scores

```
ggplot(df,aes(tier, qual, fill=tier)) +
  geom_hline(yintercept = 13.0103, linetype ="dotted") +
  geom_hline(yintercept = 20.0103, linetype ="dashed") +
  geom_quasirandom(varwidth = T, shape=21, color="grey10") +
  annotate("text", label = "p < 0.05", x=4.3,y=13.0103+2) +
  annotate("text", label = "p < 0.01", x=4.3,y=20.0103+4) +
  scale_fill_brewer("", palette = "Set1", guide=F) +
  xlab("Variant tier") + ylab("Q score") +
  theme_bw()
```



ROC curve for rare variants

```
df.1 <- subset(df, tier %in% c("error", "0.1%"))
df.1$type <- ifelse(df.1$tier == "error", 0, 1)

make_roc <- function(rocobj, type) {
  .df.roc <- data.frame(spec = seq(0, 1, 0.01))
  sens.ci <- ci.se(rocobj, specificities = .df.roc$spec)

  .df.roc$sens.lo <- sens.ci[,1]
  .df.roc$sens.me <- sens.ci[,2]
  .df.roc$sens.hi <- sens.ci[,3]
  .df.roc$type <- type

  .df.roc
}

rocobj <- roc(type ~ qual, df.1, ci=T)
print(rocobj)
```

```
##
## Call:
## roc.formula(formula = type ~ qual, data = df.1, ci = T)
##
## Data: qual in 1467 controls (type 0) < 42 cases (type 1).
## Area under the curve: 0.9287
## 95% CI: 0.8827-0.9748 (DeLong)
```

```
df.roc <- make_roc(rocobj, "Q score")
rocobj <- roc(type ~ frequency, df.1, ci=T)
print(rocobj)
```

```
##
## Call:
## roc.formula(formula = type ~ frequency, data = df.1, ci = T)
##
## Data: frequency in 1467 controls (type 0) < 42 cases (type 1).
## Area under the curve: 0.8633
## 95% CI: 0.7873-0.9393 (DeLong)
```

```
df.roc <- rbind(df.roc, make_roc(rocobj, "Frequency"))

ggplot(df.roc, aes(x=spec)) +
  geom_ribbon(aes(ymin=sens.lo, ymax=sens.hi, group=type), fill="grey", alpha=0.5) +
  geom_abline(slope = 1, intercept = 1, linetype = "dashed") +
  geom_line(aes(y=sens.me, color = type)) +
  scale_x_reverse("Specificity") +
  scale_y_continuous("Sensitivity", limits=c(0,1)) +
  scale_color_brewer("Threshold", palette = "Set1") +
  theme_bw()
```