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

Mikhail Shugay

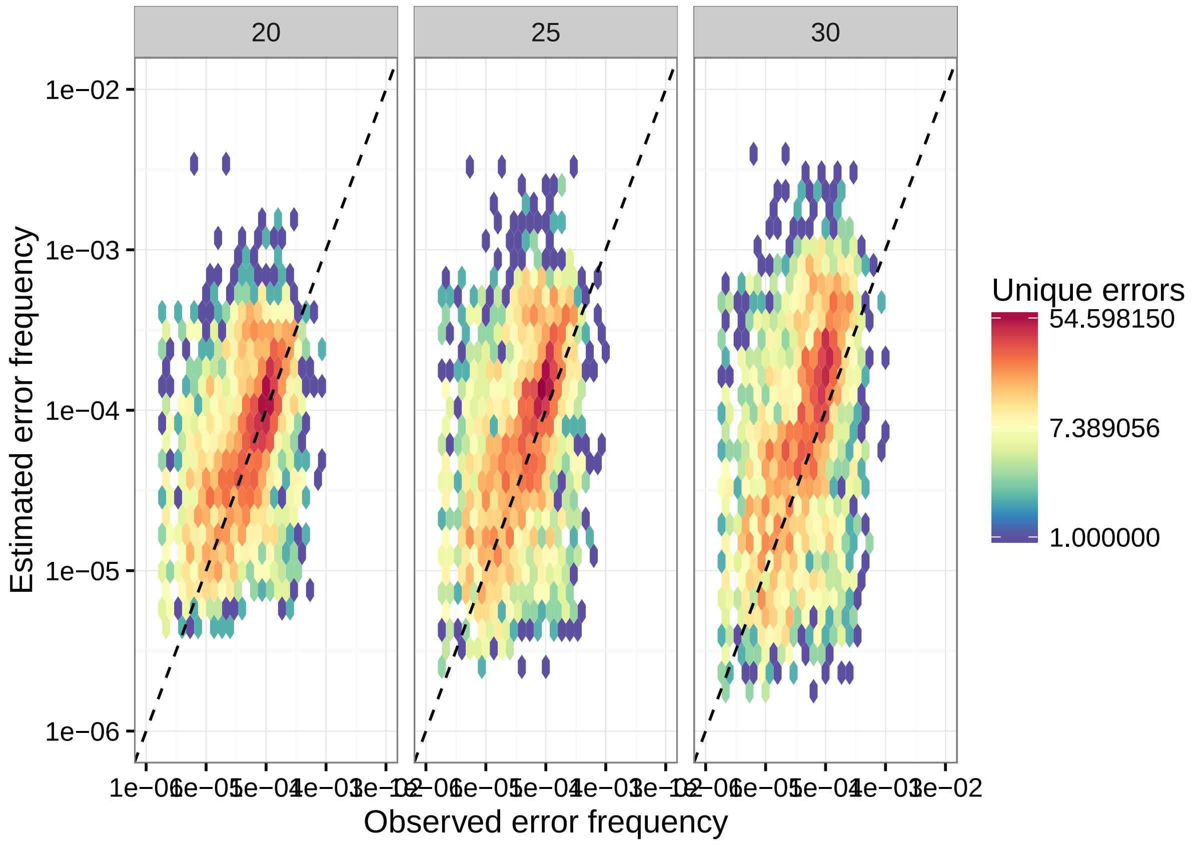
November 6, 2016

Load data from 2 independent experiments with 10 different PCR assays.

```r
df <- data.frame()

for (q in c(20,25,30)){
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")) {

    .df <- read.table(
        paste(
          paste(q, "-1", paste("polerr", proj, sep=""), sep="_"),
          paste("polerr", proj, ".", sample, ".variant.caller.txt", sep=""), sep ="/"),
                               header=T, sep="\t", stringsAsFactors = F)
    .df$q <- q
    .df$proj <- proj
    .df$sample <- sample
    df <- rbind(df, .df)
  }
}
}

df <- subset(df, freq > 0 & freq < 0.001 &
               !grepl("D", mutation) & !grepl("I", mutation) & global.est == 0 &
               coverage > 0)

df$mut.split <- sapply(df$mutation, function(x) strsplit(as.character(x),"[S:>]"))
df$mutation.pos <- as.integer(sapply(df$mut.split, function(x) x[2]))
df$mutation.from <- sapply(df$mut.split, function(x) x[3])
df$mutation.to <- sapply(df$mut.split, function(x) x[4])
df$mut.split  <- NULL
```

Model estimates for error rate:

```r
library(ggplot2)
library(RColorBrewer)
rf <- colorRampPalette(rev(brewer.pal(11,'Spectral')))
r <- rf(32)

ggplot(df, aes(x=freq, y=error.rate)) +
  stat_binhex() +
  geom_abline(intercept = 0, slope=1, linetype="dashed") +
  scale_x_log10("Observed error frequency", limits=c(1e-6, 1e-2),
                breaks=c(1e-6,1e-5,1e-4,1e-3,1e-2)) +
  scale_y_log10("Estimated error frequency", limits=c(1e-6, 1e-2),
                breaks=c(1e-6,1e-5,1e-4,1e-3,1e-2)) +
  facet_wrap(~q) +
  scale_fill_gradientn("Unique errors", colors=r, trans="log") +
  theme_bw()
```



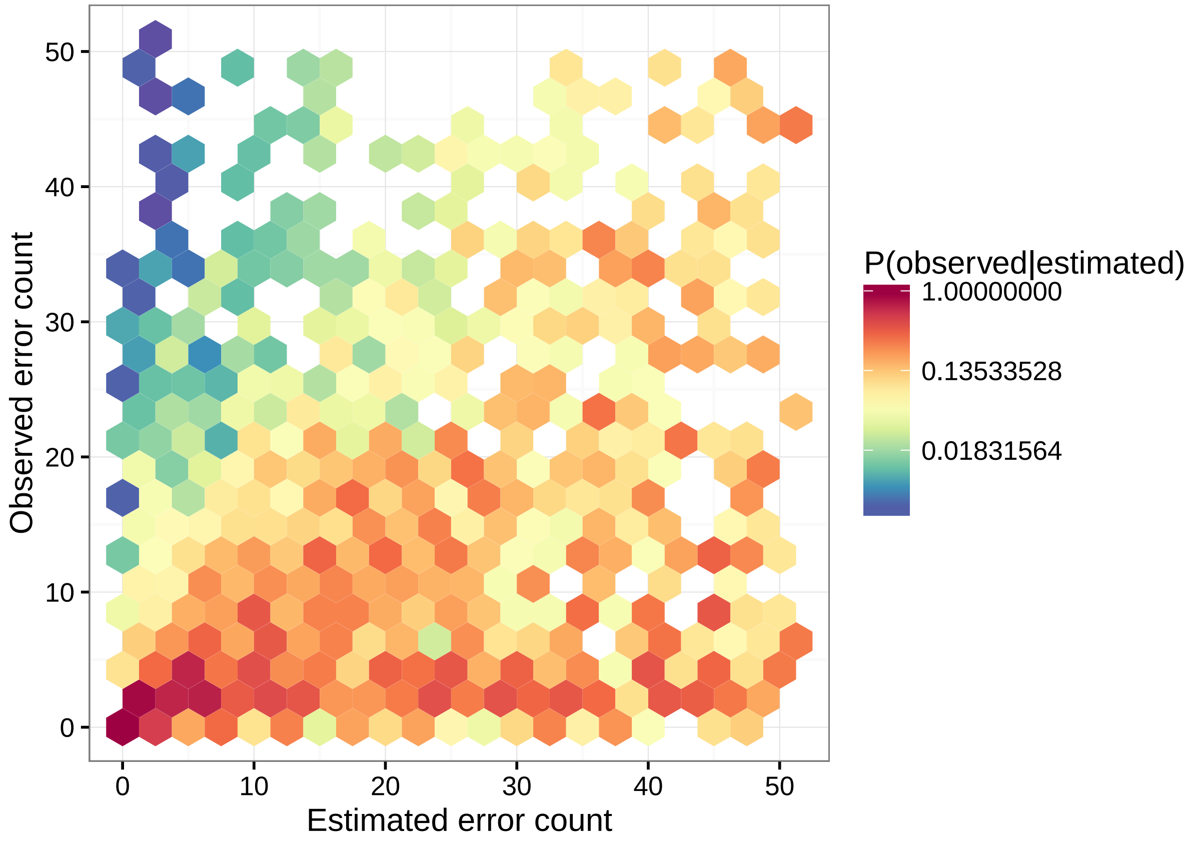
```
for (qq in c(20, 25, 30)) {
  print(with(subset(df, q == qq), cor(freq, error.rate, method = "spearman")))
}
```

```
## [1] 0.4343872
## [1] 0.4344433
## [1] 0.4356582
```

We'll go Bayesian way to compute P-values (and Q-scores) of erroneous variants accounting for the indirect minor-based error model (MBEM) we apply to infer the error rate at a given position. We'll assume that observed erroneous variant counts are distributed according to Poisson distribution with `lambda` parameter that is estimated as `MBEM error rate * coverage`, modelling the uncertainity in `lambda` using Gamma distribution. Thus, we'll arrive to a Negative Binomial distribution for our final P-values estimates. First lets plot the conditional probability `P(lambda|MBEM error rate * coverage)`:

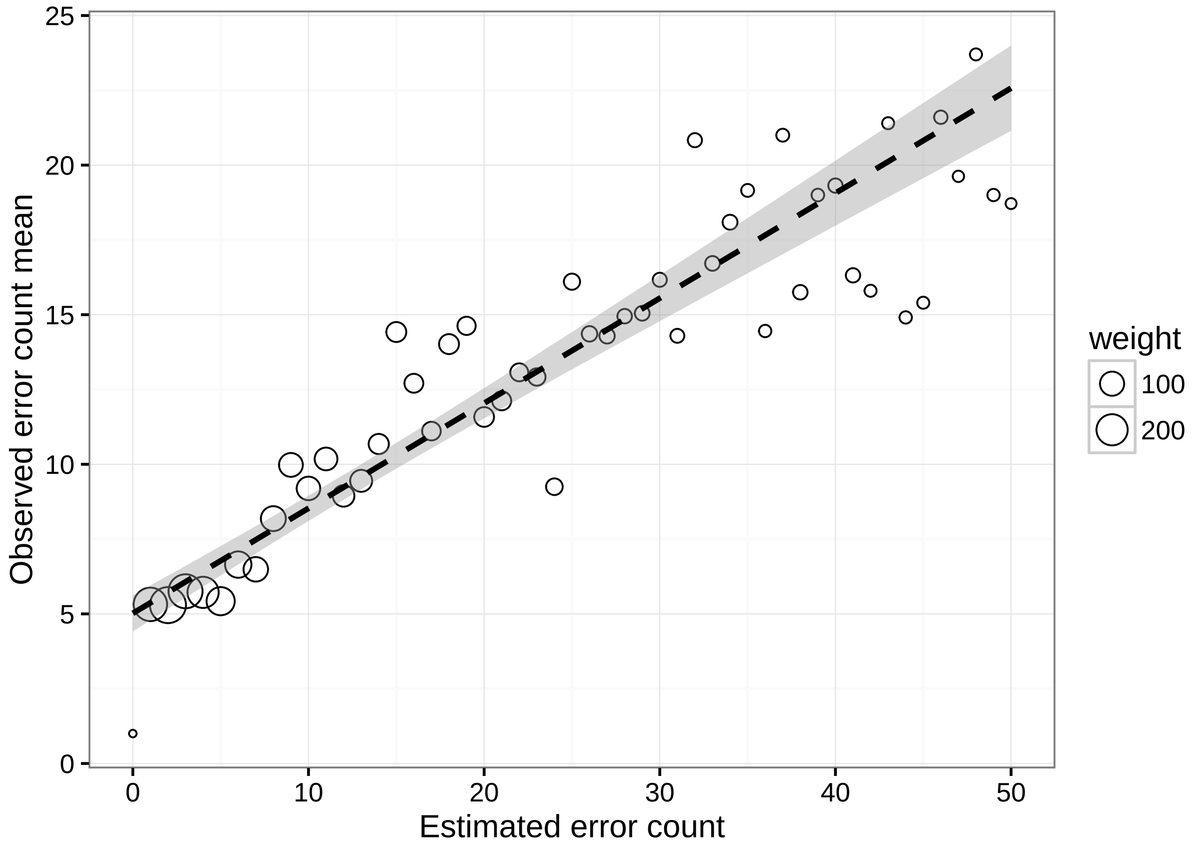
```
library(plyr)

df.1 <- subset(df, q == 20)

df.1$count.est <- round(df.1$coverage * df.1$error.rate)
df.1$count.obs <- round(df.1$coverage * df.1$freq)
df.1 <- subset(df.1, count.est <= 50 & count.obs <= 50)

# Compute mean and variance for lambda, we'll need it later
```

```
df.1 <- ddply(df.1, .(count.est), transform, f.mean = mean(count))
df.1 <- ddply(df.1, .(count.est), transform, f.var = var(count))

df.1.s <- ddply(df.1, .(count.est, count.obs),
                summarize,
                weight = length(count.est), f.mean = f.mean[1], f.var = f.var[1])
df.1.m <- ddply(df.1.s, .(count.est), transform, weight.norm = weight / sum(weight))

# requires install.packages("hexbin")

ggplot(df.1.m, aes(x=count.est, y=count.obs, weight=weight.norm)) +
  geom_hex(bins = 20) +
  xlab("Estimated error count") +
  ylab("Observed error count") +
  scale_fill_gradientn("P(observed|estimated)", colors=r, trans="log") +
  theme_bw()
```



Now lets fit it with Gamma distribution estimating `alpha` and `beta` parameters using linear regression of mean and variance of `lambda` against `MBEM error rate * coverage`:

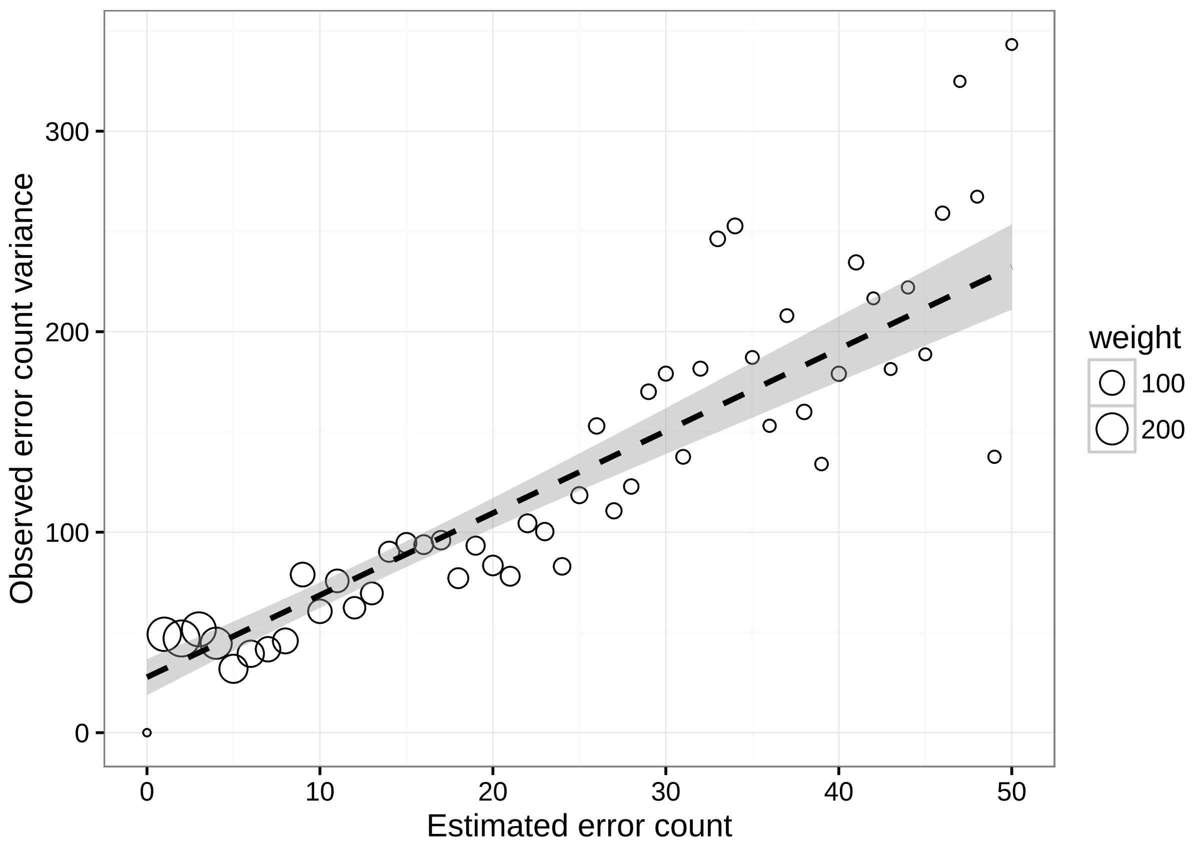
```
df.1.m.0 <- ddply(df.1.m, .(count.est, f.mean, f.var),
                  summarize,
                  weight = sum(weight))

ggplot(df.1.m.0, aes(x=count.est, y=f.mean)) + geom_point(aes(size=weight), shape=21) +
  geom_smooth(aes(weight=weight), method=lm, color="black", linetype="dashed") +
  xlab("Estimated error count") +
  ylab("Observed error count mean") +
  theme_bw()
```



```
l.mean.fit <- lm(f.mean ~ count.est,
                 df.1.m.0, weights=weight)

print(coef(l.mean.fit))
```

```
## (Intercept)   count.est
##   5.0191954   0.3510992
```

```
ggplot(df.1.m.0, aes(x=count.est, y=f.var)) + geom_point(aes(size=weight), shape=21) +
  geom_smooth(aes(weight=weight), method=lm, color="black", linetype="dashed") +
  xlab("Estimated error count") +
  ylab("Observed error count variance") +
  theme_bw()
```



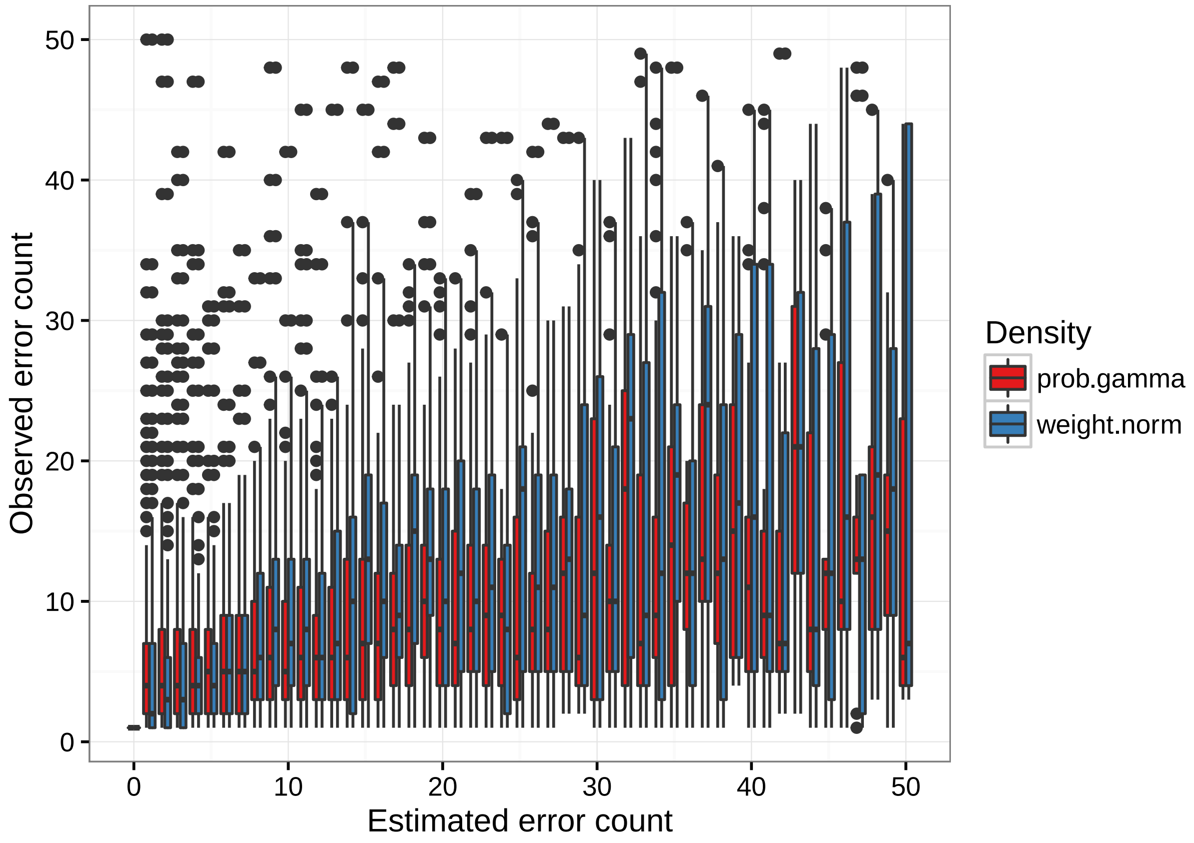
```
l.var.fit <- lm(f.var ~ count.est,
                df.1.m.0, weights=weight)

print(coef(l.var.fit))
```

```
## (Intercept)   count.est
##   27.668746    4.092389
```

Check fit by comparing probabilities computed using Gamma distribution and `P(lambda|MBEM error rate * coverage)`:

```
df.1.m <- ddply(df.1.m, .(count.est),
                transform,
              f.mean.est =
                coef(l.mean.fit)["(Intercept)"]
                + coef(l.mean.fit)["count.est"] * count.est,
              f.var.est =
                coef(l.var.fit)["(Intercept)"]
                + coef(l.var.fit)["count.est"] * count.est)

df.1.m$a <- df.1.m$f.mean.est * df.1.m$f.mean.est / df.1.m$f.var.est
df.1.m$b <- df.1.m$f.mean.est / df.1.m$f.var.est
df.1.m$prob.gamma <- dgamma(df.1.m$count.obs, rate = df.1.m$b, shape = df.1.m$a)

library(reshape2)

df.1.m.1 <- melt(df.1.m, id.vars = c("count.est", "count.obs"),
                 measure.vars = c("prob.gamma", "weight.norm"))
```

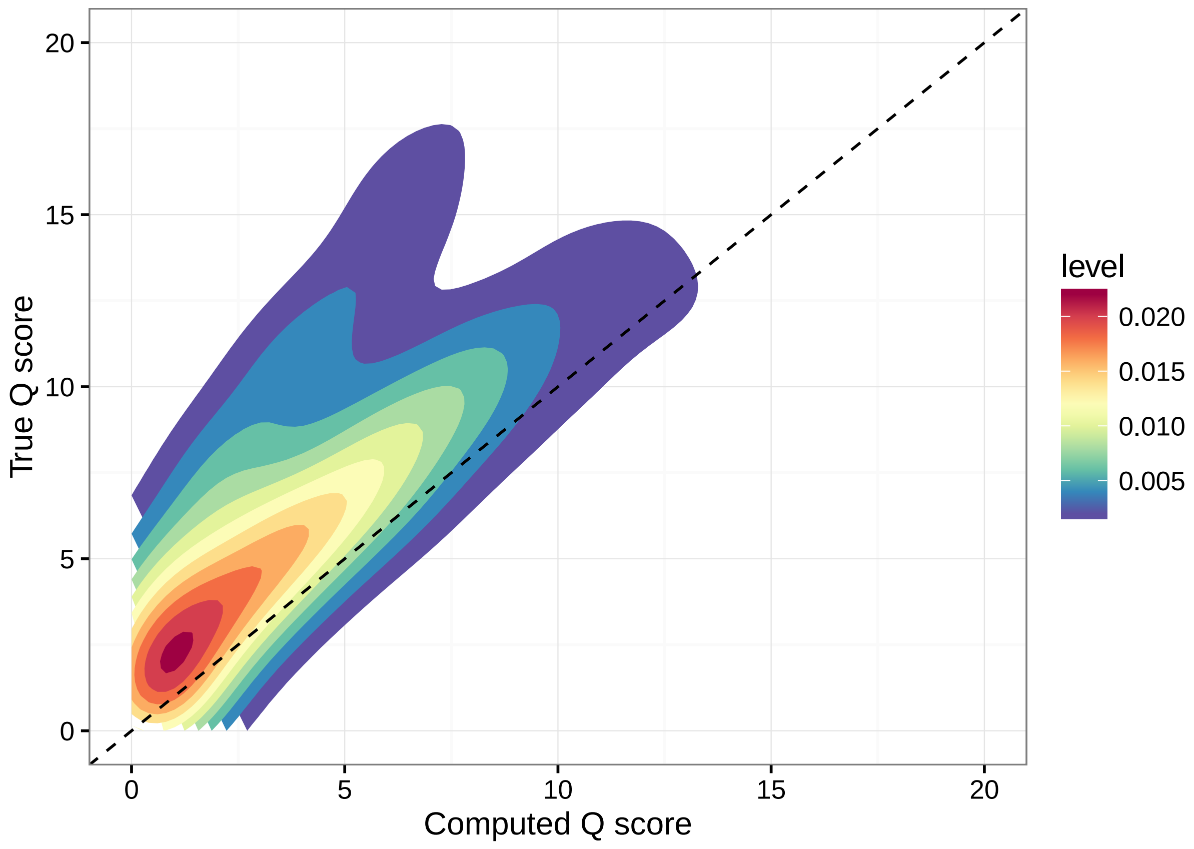
```
# requires install.packages("quantreg")
ggplot(df.1.m.1, aes(x=count.est, group = interaction(variable,count.est), fill=variable)) +
  geom_boxplot(aes(y=count.obs, weight=value)) +
  xlab("Estimated error count") +
  ylab("Observed error count") +
  scale_fill_brewer(palette = "Set1", "Density") +
  theme_bw()
```



Lets now compute P-values for errors using fitted Negative Binomial model and compare them to real P-values of error counts:

```
df.1.m$size <- df.1.m$a + df.1.m$count.obs
df.1.m$prob <- 1 / (1 + df.1.m$b)

df.1.m$pval <- with(df.1.m,
                    1.0 - pbinom(count.obs,
                                 prob = prob,
                                 size = round(size)) +
                      0.5 * dbinom(count.obs, prob = prob, size = round(size)))

df.1.m$pval.true <- sapply(df.1.m$count.obs,
                           function(x) sum(ifelse(x > df.1.m$count.obs, 0,
                                                  ifelse(x == df.1.m$count.obs,
                                                         0.5 * df.1.m$weight,
                                                         df.1.m$weight)
                                                  ))) / sum(df.1.m$weight)
```

```
ggplot(df.1.m, aes(x=-10*log10(pval), y=-10*log10(pval.true))) +
  stat_density2d(aes(fill=..level.., weight=weight), geom="polygon") +
  geom_abline(intercept = 0, slope = 1, color = "black", linetype="dashed") +
  scale_x_continuous("Computed Q score", limits=c(0,20)) +
  scale_y_continuous("True Q score", limits=c(0,20)) +
  scale_fill_gradientn(colors=r) +
  theme_bw()
```


True Q score
Computed Q score
level