Estimating Abundance Using the SonicLength Package

Charles C. Berry

March 4, 2019

Abstract

This document reviews some key functions in the `sonicLength` package. Section 1 shows how to set up data from an experiment in which sonication is used to retrieve sites. Sections 2 and 3 show how to call the `estAbund` function. Section 4 demonstrates how the `jackknife` option can be used to bias correct estimates or estimate standard errors. Simulation is helpful in assessing the properties of estimators and the `simSonic` function provides a means for simulating data; it is described in Section 5

1 Data Structures

The experimental setup for retrieving retroviral insertion sites from data via sonication was laid out by Gillet et al [1].

The data from a collection of integration sites obtained by sonication can be represented by an R `data.frame` with columns for the *Chromosome*, the *Position* on the Chromosome, and the *Strand* at which the integration site was found and a column for the *Length* of the fragment recovered. The `data.frame` would have one row for each unique combination of the above variables observed.

The following code block generates a `data.frame` that simulates such data, and displays the first and last 6 rows of it. The first 100 locations have one length each; the next 100 locations have 2,…, 100 length sampled, but duplicates of some of these were removed. The function `rfrag` generates samples from a geometric distribution with probability 0.02 that has been stochastically truncated according to a logistic law with location 45 and scale 2.5 — roughly corresponding to what is seen in actual data.

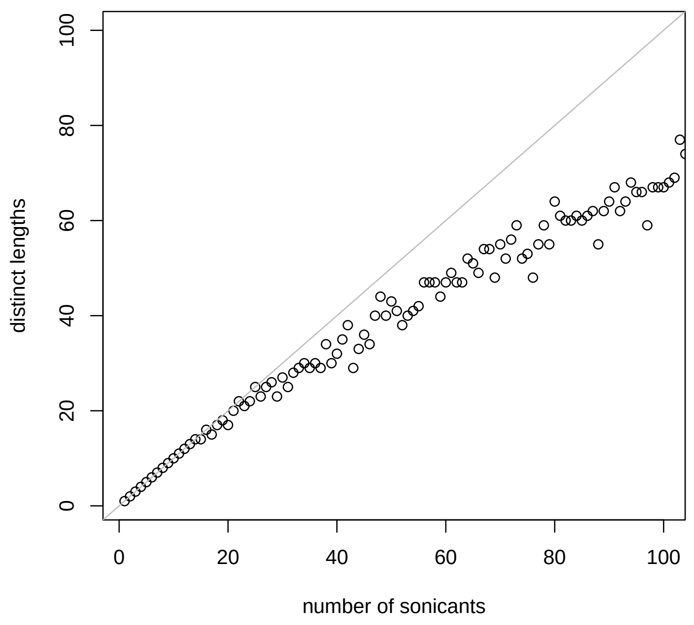
```
> require(sonicLength)
> set.seed(123)
> chr <- sample(c(1:23,"X","Y"), 200, repl=TRUE )
> pos <- sample(100000L,200)
> strand <- sample(c("+","-"),200,repl=TRUE)
> lens <- lapply(1:200, rfrag )
> nlens <- sapply( lens, length )
> loc.dframe <- data.frame( Chromosome=chr, Position=pos, Ort=strand )
> len.dframe <- unique(cbind( loc.dframe[ rep( 1:200, nlens ) , ],
+                            length=unlist(lens) ))
> rbind( head( len.dframe ), tail( len.dframe ) )

        Chromosome Position Ort length
1                8    23873   -     58
2               20    96235   +    102
2.1             20    96235   +     57
3               11    60136   -     64
3.1             11    60136   -     54
3.2             11    60136   -    125
200.163         15    98927   +     99
200.167         15    98927   +    134
200.171         15    98927   +     80
200.188         15    98927   +    109
200.189         15    98927   +    124
200.195         15    98927   +    104

>
```

The following code creates a unique identifier for each location and plots the actual number of lengths simulated versus the number of unique lengths for each location. The graph illustrates the essential difficulty with estimating the abundance of integration sites from the lengths of sonicated fragments: the more abundant sites tend to have multiple fragments of the same length. Since the fragments are subjected to PCR amplification, it is impossible to distinguish duplicates that result from amplification from those that result from unique sonicated fragments.

```
> id <- with( len.dframe , paste(Chromosome, Position, Ort ) )
> id.counts <- table(factor(id,unique(id)))
> plot( as.vector(nlens), as.vector(id.counts),
+      xlab='number of sonicants',ylab='distinct lengths',
+      xlim=c(1,100), ylim=c(1,100))
> abline(a=0,b=1,col='gray')
```



2 Using `estAbund`

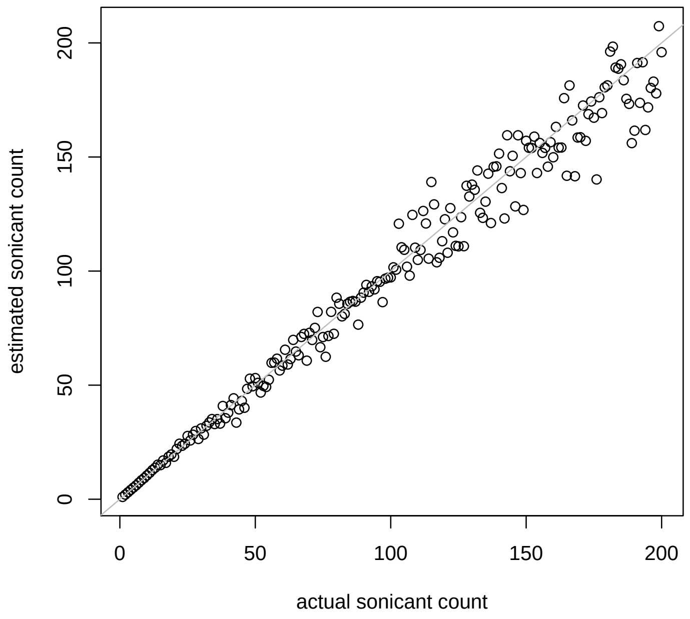
The function `estAbund` uses a maximum likelihood approach to finding the number of sonicants that underlie the number of distinct lengths seen. The following code shows how to use it. The call to `str(fit)` shows the structure of the resulting object, which has the estimated values (as `theta`), the estimated probability of recovering a sonicant of a given length (as `phi`), the number of iterations of the algorithm before convergence is achieved (as `iter`), the observed number of distinct lengths for each integration site (as `obs`), the call that generated the object, and optionally the estimated variance of `theta` (as `var.theta`). The figure shows how the estimates compare to the actual values that generated the data.

```
> fit <- estAbund(id, len.dframe$length)
> str( fit )
```

```
List of 7
$ theta : Named num [1:200] 66.6 33 159.5 110.8 175.8 ...
..- attr(*, "names")= chr [1:200] "1 16410 -" "1 26077 +" "1 52529 -" "10 10368
   -" ...
$ phi : Named num [1:503] 8.88e-10 5.78e-09 3.24e-08 1.58e-07 6.72e-07 ...
..- attr(*, "names")= chr [1:503] "25" "26" "27" "28" ...
```

```
$ var.theta: NULL
$ iter : num 7
$ call : language estAbund(locations = id, lengths = len.dframe$length)
$ obs : 'xtabs' int [1:200(1d)] 52 29 91 70 98 92 33 47 98 47 ...
..- attr(*, "dimnames")=List of 1
.. ..$ factor(locations): chr [1:200] "1 16410 -" "1 26077 +" "1 52529 -" "10
   10368 -" ...
..- attr(*, "call")= language xtabs(formula = ~factor(locations))
$ data :'data.frame': 12464 obs. of 2 variables:
..$ locations: Factor w/ 200 levels "1 16410 -","1 26077 +",..: 173 102 102 21
   21 21 123 123 123 123 ...
..$ lengths : num [1:12464] 58 102 57 64 54 125 53 130 65 68 ...
```

```
> plot(nlens, fit$theta[unique(id)],
+      xlab="actual sonicant count",
+      ylab="estimated sonicant count"
+      )
> abline(a=0,b=1,col='gray')
>
```



3 Replicated Data

When there are replicates available, likelihood methods provide a means for combining the data from all of them by maximizing the likelihood using data from all the replicates simultaneously. Here more data is simulated for the same set of locations as were used above. An additional column called `repl` is added to keep track of the separate replicates. Also, the simulations use different sonication methods resulting in a different distribution of lengths for each of the three replicates. There are some differences in the object returned by `estAbund`; there is now a `data.frame` in the component called `lframe`, and it keeps track of the lengths (in column `x`) and replicates (in column `strata`). Also, the `obs` component is now a matrix with one row for each integration site loction and one column for the count for each replicate.

```
> len.dframe$repl <- 1
> lens2 <- lapply(1:200,rfrag, rate=0.03 )
> nlens2 <- sapply( lens2, length )
> len.dframe2 <- unique(cbind( loc.dframe[ rep( 1:200, nlens2 ) , ],
+                              length=unlist(lens2), repl=2 ))
> lens3 <- lapply(1:200,rfrag, rate=0.04 )
> nlens3 <- sapply( lens, length )
> len.dframe3 <- unique(cbind( loc.dframe[ rep( 1:200, nlens3 ) , ],
+                              length=unlist(lens3), repl=3 ))
> len.dframe <- rbind(len.dframe,len.dframe2,len.dframe3)
> fit2 <- with(len.dframe, estAbund( paste(Chromosome, Position, Ort ),
+                                     length, repl ))
>
```

Here is the structure of the resulting object.

```
> str( fit2 )
```

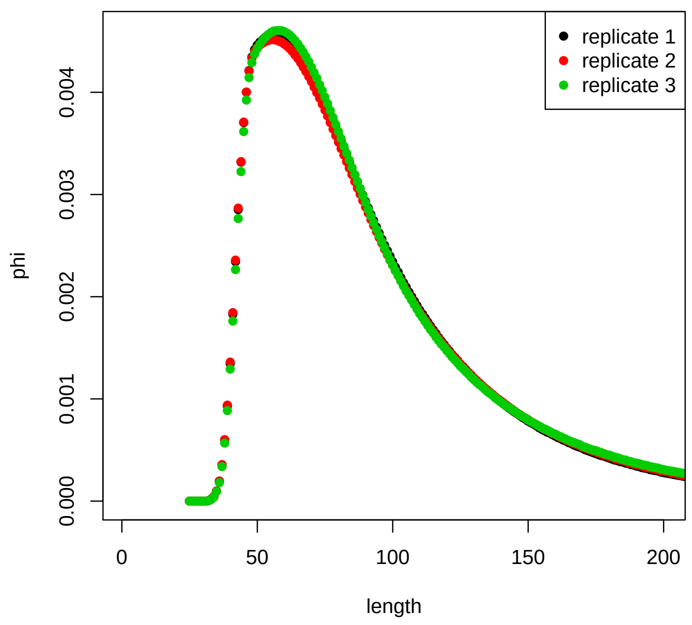
```
List of 8
$ theta : Named num [1:200] 3.02 6.07 9.08 12.21 15.28 ...
..- attr(*, "names")= chr [1:200] "8 23873 -" "20 96235 +" "11 60136 -" "23
   51502 -" ...
$ phi : Named num [1:1509] 1.61e-10 1.13e-09 6.77e-09 3.51e-08 1.58e-07 ...
..- attr(*, "names")= chr [1:1509] "1 25" "1 26" "1 27" "1 28" ...
$ var.theta: NULL
$ iter : num 5
$ call : language estAbund(locations = paste(Chromosome, Position, Ort),
   lengths = length, replicates = repl)
$ lframe :'data.frame': 1611 obs. of 4 variables:
..$ y : num [1:1611] 0 0 0 0 0 0 0 0 0 0 ...
..$ x : int [1:1611] 1 2 3 4 5 6 7 8 9 10 ...
..$ orig : logi [1:1611] FALSE FALSE FALSE FALSE FALSE FALSE ...
..$ strata: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ obs : 'xtabs' int [1:200, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
..- attr(*, "dimnames")=List of 2
.. ..$ factor(locations, sl.start) : chr [1:200] "8 23873 -" "20 96235 +" "11
   60136 -" "23 51502 -" ...
.. ..$ factor(replicates, sort(unique(replicates))): chr [1:3] "1" "2" "3"
..- attr(*, "call")= language xtabs(formula = ~factor(locations, sl.start) +
   factor(replicates, sort(unique(replicates))))
$ data :'data.frame': 37496 obs. of 3 variables:
..$ locations : Factor w/ 200 levels "1 16410 -","1 26077 +",..: 173 102 102 21
   21 21 123 123 123 123 ...
..$ lengths : num [1:37496] 58 102 57 64 54 125 53 130 65 68 ...
..$ replicates: num [1:37496] 1 1 1 1 1 1 1 1 1 1 ...

>
```

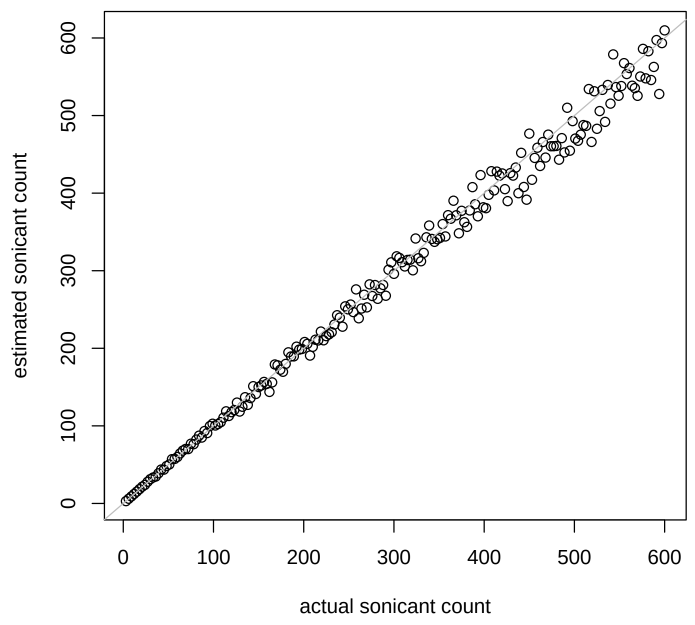
Here is a plot of the estimated values of `phi`.

```
> with( fit2,
+      plot( lframe$x[ lframe$orig ], phi, pch=16,
+           col=lframe$strata[ lframe$orig ],
+           xlab="length",
+           ylab='phi',
+           xlim=c(1,200)
+           ))
> legend("topright",col=1:3,pch=rep(16,3),legend=paste("replicate",1:3))
>
>
```



And here the estimated number of sonicants are plotted against the actual number (which now range up to 300, since there are three replicates).

```
> with( fit2,
+      plot(3*nlens , theta[unique(id)],
+      xlab="actual sonicant count",
+      ylab="estimated sonicant count"
+      )
+      )
> abline(a=0,b=1,col='gray')
>
>
```



4 Jackknife Corrections

When replicated data are available, one can use the jackknife method to correct the bias of any estimator or to estimate its standard error. See Miller [2] for a review.

In this section we add the `jackknife=TRUE` argument and the `theta.var=TRUE` argument to the call for `estAbund`. The former argument is one of the named arguments for `estAbund`, but the latter is not. The latter argument is one of the named arguments to the fitting function `maxEM` and is passed to it with the effect that estimates of the variances of the abundance estimates and of the proportional abundance estimates are calculated. After the fitting is completed, some of the structure of the resulting object is revealing by listing the names of the components of the list returned.

```
> fit2 <- with(len.dframe,
+              estAbund( paste(Chromosome, Position, Ort ),
+                       length, repl, jackknife=TRUE,
+                       theta.var=TRUE))
> head.names <- function(x) head( names(x) )
> sapply( fit2, head.names )
```

```
$theta
```

```
[1] "8 23873 -"  "20 96235 +" "11 60136 -" "23 51502 -" "X 40256 +"
[6] "2 88021 +"

$phi
[1] "1 25" "1 26" "1 27" "1 28" "1 29" "1 30"

$var.theta
[1] "theta" "prop"

$iter
NULL

$call
[1] ""            "locations"  "lengths"    "replicates" "jackknife"
[6] "theta.var"

$lframe
[1] "y"      "x"      "orig"   "strata"

$obs
NULL

$jackknife
NULL

$data
[1] "locations"  "lengths"    "replicates"

>
```

And here some of the structure of the jackknife component is shown. Basically, that component has the same length as the number of replicates. . .

```
> length(fit2$jackknife)

[1] 3

>
```

. . . and each element has the same element names as the parent object (except for `jackknife`.

```
> str(fit2$jackknife[[1]])

List of 5
$ theta : Named num [1:200] 2.01 4.05 6.04 8.12 10.18 ...
..- attr(*, "names")= chr [1:200] "8 23873 -" "20 96235 +" "11 60136 -" "23
```

```
   51502 -" ...
$ phi : Named num [1:1006] 1.68e-10 1.23e-09 7.74e-09 4.17e-08 1.94e-07 ...
..- attr(*, "names")= chr [1:1006] "2 25" "2 26" "2 27" "2 28" ...
$ var.theta:List of 2
..$ theta: num [1:200] 2.02 4.09 6.12 8.27 10.41 ...
..$ prop : num [1:200] 1.29e-09 2.62e-09 3.92e-09 5.29e-09 6.67e-09 ...
$ iter : num 6
$ call : language maxEM(slmat = slmat3[slmat.rowuse, slmat.coluse], theta.var =
   TRUE, phi.update = phi.update.lframe, lframe =| __truncated__

>
```

And here is a demonstration of using the jackknife to correct bias in the estimate of the proportion represented by the most abundant site.

```
> nreps <- length( fit2$jackknife )
> argmax.theta <- which.max( fit2$theta )
> ## function to extract the estimate
> maxpr <- function(x) prop.table( x$theta )[ argmax.theta ]
> ## apply to full sample
> maxpr.total <- maxpr( fit2 )
> ## make pseudo observations
> pseudo.maxpr <- nreps * maxpr.total -
+   (nreps-1) * sapply( fit2$jackknife, maxpr )
> ## report results
> likeStdErr <- sqrt( fit2$var.theta$prop[ argmax.theta ] )
> jackStdErr = sd ( pseudo.maxpr ) / sqrt(nreps)
> all.res <- c( uncorrected = unname(maxpr.total),
+              corrected = mean(pseudo.maxpr),
+              likeStdErr = likeStdErr,
+              jackStdErr = jackStdErr)
> cbind(all.res)

                 all.res
uncorrected 0.0103142430
corrected   0.0103070674
likeStdErr  0.0006465330
jackStdErr  0.0004442845

>
>
```

The bias correction has scant effect here. As it turns out, the estimator is nearly unbiased, so this is expected. The jackknife standard error is within error distance of the standard error based on asymptotic theory given that there are just 3 replicates.

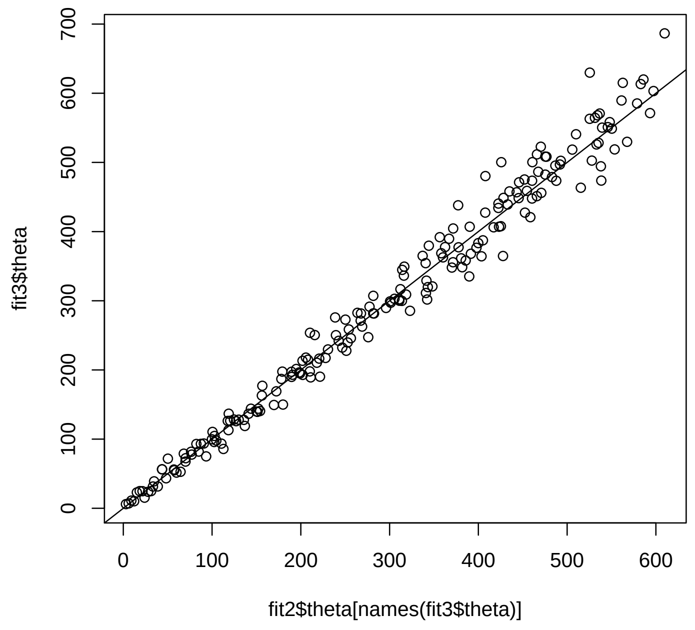
5 Simulating Data

A convenience wrapper called `simSonic` is given for simulating data. Here it is used to simulate more data according to the estimated values of `fit2[['theta']]` and `fit2[['phi']]`, and the resulting estimate of `theta` is plotted against its parent.

```
> more.data <- simSonic(fit2$theta,fit2$phi)
> fit3 <- do.call(estAbund, more.data )
> plot( fit2$theta[names(fit3$theta)], fit3$theta )
> abline(a=0,b=1)
> str(fit3)

List of 8
$ theta : Named num [1:200] 6.05 7.08 11.16 10.18 22.93 ...
..- attr(*, "names")= chr [1:200] "8 23873 -" "20 96235 +" "11 60136 -" "23
   51502 -" ...
$ phi : Named num [1:1380] 2.23e-06 6.77e-06 1.84e-05 4.47e-05 9.82e-05 ...
..- attr(*, "names")= chr [1:1380] "1 31" "1 32" "1 33" "1 34" ...
$ var.theta: NULL
$ iter : num 5
$ call : language (function (locations, lengths, replicates = NULL, jackknife =
   F, kmax = 0, min.length = 20, ...)  ...
$ lframe :'data.frame': 1500 obs. of 4 variables:
..$ y : num [1:1500] 0 0 0 0 0 0 0 0 0 0 ...
..$ x : int [1:1500] 1 2 3 4 5 6 7 8 9 10 ...
..$ orig : logi [1:1500] FALSE FALSE FALSE FALSE FALSE FALSE ...
..$ strata: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
$ obs : 'xtabs' int [1:200, 1:3] 1 2 4 4 6 9 6 4 8 9 ...
..- attr(*, "dimnames")=List of 2
.. ..$ factor(locations, sl.start) : chr [1:200] "8 23873 -" "20 96235 +" "11
   60136 -" "23 51502 -" ...
.. ..$ factor(replicates, sort(unique(replicates))): chr [1:3] "1" "2" "3"
..- attr(*, "call")= language xtabs(formula = ~factor(locations, sl.start) +
   factor(replicates, sort(unique(replicates))))
$ data :'data.frame': 37657 obs. of 3 variables:
..$ locations : Factor w/ 200 levels "1 16410 -","1 26077 +",..: 173 173 173
   173 173 173 102 102 102 102 ...
..$ lengths : num [1:37657] 248 78 69 86 102 69 85 137 42 68 ...
..$ replicates: num [1:37657] 1 3 2 2 2 3 1 2 3 1 ...

>
```



References

[1] N. A. Gillet, N. Malani, A. Melamed, N. Gormley, R. Carter, D. Bentley, C. Berry, F. D. Bushman, G. P. Taylor, and C. R. Bangham. The host genomic environment of the provirus determines the abundance of HTLV-1-infected T-cell clones. *Blood*, 117:3113–3122, Mar 2011.

[2] R. G. Miller. The jackknife-a review. *Biometrika*, 61(1):1–15, 1974.