Using preseqR for analysis of species accumulation curves

Chao Deng, Timothy Daley, and Andrew D Smith

preseqR is an R package that provides all necessary functions for using rational function approximations to the Good-Toulmin power series for prediction of species accumulation curves, the expected number of species observed as a function of the sample size. This prediction is based on a sampling experiment from the population of interest. In this document, we will briefly introduce the workflow of *preseqR* and show how to generate all results presented in the paper *Applications of species accumulation curves in large-scale biological data analysis.*

The input of *preseqR* is the duplicate counts histogram, a two-column matrix representing the duplicate species count frequencies of the sampled experiment. The first column is ascending sorted counts $j = 1, 2, \ldots$. The second column is n_j, the frequency of count j. One example is the word frequencies in Shakespeare's known works, where n_j is the number of words appeared exactly j times. The output is a species accumulation curve.

The primary function in the *preseqR* package is `preseqR.rfa.species.accum.curve`. This function bootstraps the observed histogram to aggregate the predicted species accumulation curves. Due to the extremely non-linear nature of rational function approximations, analytic confidence intervals are impossible. The bootstrapping allows for calculation of confidence intervals. We use log-normal confidence intervals, as suggested by Chao (1987, Formula (12)). The function `preseqR.rfa.curve` has similar functionality to `preseqR.rfa.species.accum.curve`. The primary difference is `preseqR.rfa.curve` does a single extrapolation based on the input histogram, with no bootostrapping. This allows for extremely quick predictions. Although in practice the running time of `preseqR.rfa.species.accum.curve` is within minutes.

Reproducing examples of *Applications of species accumulation curves in large-scale biological data analysis*

For all of the following, installation the preseqR package is required. The version number is 1.2.1.

```
install.packages("preseqR", repos = "http://cran.us.r-project.org", type="source")
library(preseqR)
```

Estimating the vocabulary of Shakespeare

Following the example of Efron and Thisted (1976), we investigated predicting the vocabulary of Shakespeare using the observed vocabulary contained in his known works. The word count histogram was taken from Efron & Thisted (1976, Table 1), with the tail of the histogram (counts larger than 100) set to ensure that the total number of words and distinct words agree with Efron & Thisted.

To evaluate our method we first take subsamples, taking approximately 5% and 10% of Shakespeare's words, then use these to extrapolate and compare against the curve of Shakespeare's vocabulary.

5% subsample

```
data(ShakespeareWordHist)
set.seed(123456)
ind <- 1:sum(ShakespeareWordHist[, 2])
n <- rep(ShakespeareWordHist[, 1], ShakespeareWordHist[, 2])
X <- rep(ind, n)
## sampling without replacement by built-in function sample()
T <- sample(X, 44000, replace = FALSE)
distinct <- sum(floor(ShakespeareWordHist[, 2]))
X <- vector(length=as.integer(distinct), mode="numeric")
for (i in T) {
  X[i] <- X[i] + 1
}
## construct the corresponding two-column matrix
H <- vector(length=max(X), mode="numeric")
for (i in X) {
  if (i > 0) {
    H[i] <- H[i] + 1
  }
}
ind <- which(H != 0)
Shakespeare_44000.hist <- matrix(c(ind, H[ind]), ncol = 2)
```

We compared our method with another proposed method from Anne Chao's group called iNEXT (Hsieh et al., 2015), available on github at https://github.com/JohnsonHsieh/iNEXT. The version number is 2.0.3.

```
################################################################################
### install the iNEXT package.
### see https://github.com/JohnsonHsieh/iNEXT
################################################################################
install.packages("devtools", contriburl='http://cran.cnr.Berkeley.edu/src/contrib',
                 type = "source")
library(devtools)
install_github('JohnsonHsieh/iNEXT')
library(iNEXT)
```

```
set.seed(123456)
## extrapolation based on 5% sample
## use preseqR to predict the accumulation curve
Shakespeare_44000.preseqR <- preseqR.rfa.species.accum.curve(
  Shakespeare_44000.hist, ss=1.1e4, max.extrapolation=1e7)
Shakespeare_44000.preseqR[1:20, ]
```

```
##       sample.size yield.estimate lower.0.95CI uppper.0.95CI
##  [1,]       11000       2909.365     2854.236      2965.559
##  [2,]       22000       4456.771     4343.499      4572.997
##  [3,]       33000       5701.353     5547.370      5859.611
##  [4,]       44000       6757.885     6571.799      6949.240
##  [5,]       55000       7686.345     7472.926      7905.859
##  [6,]       66000       8522.986     8285.290      8767.501
##  [7,]       77000       9281.444     9021.480      9548.899
##  [8,]       88000       9995.888     9713.161     10286.844
##  [9,]       99000      10648.405    10336.699     10969.511
```

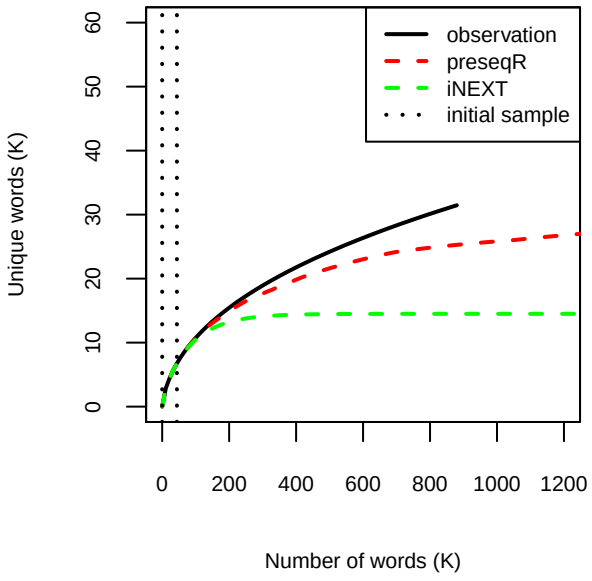
```
## [10,]      110000      11244.199    10887.442     11612.646
## [11,]      121000      11793.889    11366.179     12237.693
## [12,]      132000      12289.655    11759.320     12843.909
## [13,]      143000      12779.810    12115.352     13480.709
## [14,]      154000      13211.918    12386.847     14091.947
## [15,]      165000      13633.941    12628.364     14719.591
## [16,]      176000      14064.371    12864.483     15376.174
## [17,]      187000      14506.456    13103.189     16060.004
## [18,]      198000      14889.323    13277.414     16696.921
## [19,]      209000      15228.395    13405.027     17299.779
## [20,]      220000      15585.714    13549.247     17928.264
```

```
## convert the data format for iNEXT
S <- rep(Shakespeare_44000.hist[, 1], Shakespeare_44000.hist[, 2])
Shakespeare_44000.sample <- sort(S, decreasing=TRUE)

## use iNEXT to predict the accumulation curve
Shakespeare_44000.size <- Shakespeare_44000.preseqR[, 1]
Shakespeare_44000.iNEXT <- iNEXT(Shakespeare_44000.sample, q=0, datatype="abundance",
                                 size=Shakespeare_44000.size)

## extract useful columns (sample size, predicted values, lower bound, upper bound)
Shakespeare_44000.iNEXT <- Shakespeare_44000.iNEXT$Accumulation[, c(1, 4, 5, 6)]
Shakespeare_44000.iNEXT[1:20, ]
```

```
##         m        qD qD.95.LCL qD.95.UCL
## 1   11000  2905.827  2873.428  2938.226
## 2   22000  4449.219  4393.508  4504.930
## 3   33000  5691.989  5614.537  5769.441
## 4   44000  6746.000  6647.177  6844.823
## 5   55000  7669.233  7548.650  7789.816
## 6   66000  8482.533  8339.210  8625.855
## 7   77000  9198.989  9031.855  9366.122
## 8   88000  9830.133  9638.396 10021.870
## 9   99000 10386.124 10169.435 10602.813
## 10 110000 10875.911 10634.385 11117.436
## 11 121000 11307.376 11041.537 11573.215
## 12 132000 11687.465 11398.161 11976.769
## 13 143000 12022.295 11710.614 12333.976
## 14 154000 12317.255 11984.449 12650.060
## 15 165000 12577.093 12224.513 12929.672
## 16 176000 12805.990 12435.034 13176.947
## 17 187000 13007.632 12619.704 13395.561
## 18 198000 13185.263 12781.744 13588.783
## 19 209000 13341.743 12923.968 13759.519
## 20 220000 13479.591 13048.833 13910.348
```

```
## calculate the expected accumulation curve
data(ShakespeareWordHist)
ShakespeareWordHist.true <- preseqR.interpolate.distinct(1.1e4, ShakespeareWordHist)

## compute the scaled error
I <- 4*(1:20)
```

```
Shakespeare_44000.error.preseqR <- (
  Shakespeare_44000.preseqR[I, 2] - ShakespeareWordHist.true[I, 2]) /
  ShakespeareWordHist.true[I, 2]
Shakespeare_44000.error.iNEXT <- (
  Shakespeare_44000.iNEXT[I, 2] - ShakespeareWordHist.true[I, 2]) /
  ShakespeareWordHist.true[I, 2]

## plot the result
W = 3.5; H = 4
# pdf("Shakespeare_iNEXT44000.pdf", width=W, height=H)
x.lim = 1.2e3
y.lim = 60
CEX = 0.7
CEX.AXIS = 0.7
CEX.LAB = 0.7
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim), ylab = "Unique words (K)",
     xlab = "Number of words (K)", cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)

plot(rbind(c(0,0), ShakespeareWordHist.true / 1e3), type="l", lty=1, col="black", lwd=2,
     xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n', xaxt="n",
     yaxt="n")
lines(rbind(c(0, 0), Shakespeare_44000.preseqR[, 1:2] / 1e3), lty=2, lwd=2, col="red",
      xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n", yaxt="n")
lines(rbind(c(0, 0), Shakespeare_44000.iNEXT[, 1:2]) / 1e3, lty=2, lwd=2, col="green",
      xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n", yaxt="n")
abline(v=0, lty=3, lwd=2)
abline(v=44, lty=3, lwd=2)
legend("topright", c("observation","preseqR", "iNEXT", "initial sample"),
       lty=c(1, 2, 2, 3), lwd=c(2, 2, 2, 2), col=c("black", "red", "green", "black"),
       cex=CEX)
```

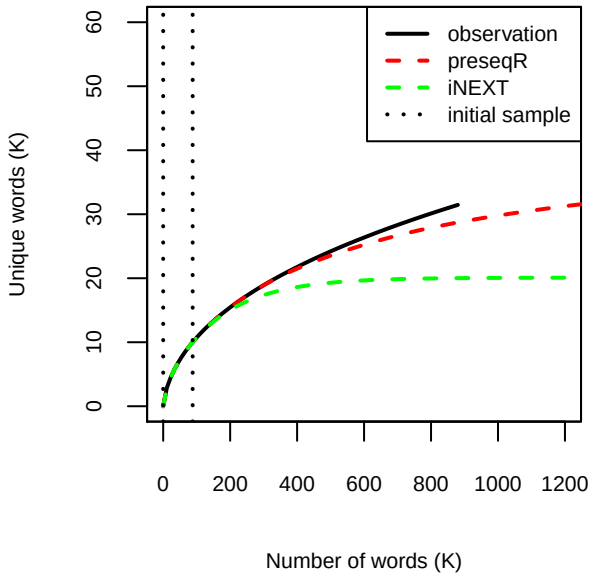



```
#dev.off()
```

10% **subsample**

```
################################################
### extrapolation based on 10% sample

#############################################################################
## generate a 10% subsample for Shakespeare's word counts
#############################################################################
set.seed(123456)
ind <- 1:sum(ShakespeareWordHist[, 2])
n <- rep(ShakespeareWordHist[, 1], ShakespeareWordHist[, 2])
X <- rep(ind, n)
T <- sample(X, 88000, replace = FALSE)
distinct <- sum(floor(ShakespeareWordHist[, 2]))
X <- vector(length=as.integer(distinct), mode="numeric")
for (i in T) {
  X[i] <- X[i] + 1
}

H <- vector(length=max(X), mode="numeric")
for (i in X) {
  if (i > 0) {
    H[i] <- H[i] + 1
  }
}
ind <- which(H != 0)
Shakespeare_88000.hist <- matrix(c(ind, H[ind]), ncol = 2)
S <- rep(Shakespeare_88000.hist[, 1], Shakespeare_88000.hist[, 2])
Shakespeare_88000.sample <- sort(S, decreasing=TRUE)

Shakespeare_88000.preseqR <- preseqR.rfa.species.accum.curve(
  Shakespeare_88000.hist, ss=1.1e4, max.extrapolation=1e7)

Shakespeare_88000.iNEXT <- iNEXT(Shakespeare_88000.sample, q=0, datatype="abundance",
                                 size=Shakespeare_88000.preseqR[, 1])

Shakespeare_88000.iNEXT <- Shakespeare_88000.iNEXT$Accumulation[, c(1, 4, 5, 6)]

data(ShakespeareWordHist)
ShakespeareWordHist.true <- preseqR.interpolate.distinct(1.1e4, ShakespeareWordHist)

I <- 8*(1:10)
Shakespeare_88000.error.preseqR <- (
  Shakespeare_88000.preseqR[I, 2] - ShakespeareWordHist.true[I, 2]) /
  ShakespeareWordHist.true[I, 2]
Shakespeare_88000.error.iNEXT <- (
  Shakespeare_88000.iNEXT[I, 2] - ShakespeareWordHist.true[I, 2]) /
  ShakespeareWordHist.true[I, 2]

## plot the figure
#W = 3.5; H = 4
#pdf("Shakespeare_88000_iNEXT.pdf", width=W, height=H)
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim), ylab = "Unique words (K)",
```

```
     xlab = "Number of words (K)", cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)

plot(rbind(c(0,0), ShakespeareWordHist.true / 1e3), type="l", lty=1, col="black", lwd=2,
     xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n', xaxt="n",
     yaxt="n")
lines(rbind(c(0, 0), Shakespeare_88000.preseqR[, 1:2] / 1e3), lty=2, lwd=2, col="red",
      xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n", yaxt="n")
lines(rbind(c(0, 0), Shakespeare_88000.iNEXT[, 1:2] / 1e3), lty=2, lwd=2, col="green",
      xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n", yaxt="n")
abline(v=0, lty=3, lwd=2)
abline(v=88, lty=3, lwd=2)
legend("topright", c("observation","preseqR", "iNEXT", "initial sample"), lty=c(1,2,2,3),
       lwd=c(2,2,2, 2), col=c("black","red","green","black"), cex=CEX)
```




```
#dev.off()
```

Average performance comparison

We evaluated the performance of these estimators on average by taking 100 independent random samples of 44,000 and 88,000 words and compared the estimated number of distinct words against the expected number of distinct words when a total of 880,000 of Shakespeares words were observed. For comparison we use the mean absolute deviation (MAD) and root mean square error (RMSE) as metrics (see Walther and Moore, 2005).

```
data(ShakespeareWordHist)
curves005.preseqR <- NULL
curves005.iNEXT <- NULL
for (count in 1:100) {
  ## set the random seed for being reproducible
  set.seed(count)
  ## downsample
  ind <- 1:sum(ShakespeareWordHist[, 2])
```

```
  n <- rep(ShakespeareWordHist[, 1], ShakespeareWordHist[, 2])
  X <- rep(ind, n)

  T <- sample(X, 44000, replace = FALSE)
  distinct <- sum(floor(ShakespeareWordHist[, 2]))
  X <- vector(length=as.integer(distinct), mode="numeric")
  for (i in T) {
    X[i] <- X[i] + 1
  }
  ## construct the corresponding two-column matrix
  H <- vector(length=max(X), mode="numeric")
  for (i in X) {
    if (i > 0) {
      H[i] <- H[i] + 1
    }
  }
  ind <- which(H != 0)
  Shakespeare_44000.hist <- matrix(c(ind, H[ind]), ncol = 2)
  ## use preseqR to predict the species accumulatio curve
  Shakespeare_44000.preseqR <- preseqR.rfa.species.accum.curve(
    Shakespeare_44000.hist, ss=1.1e4, max.extrapolation=1e7)
  ## combine all results into a matrix
  curves005.preseqR <- cbind(curves005.preseqR, Shakespeare_44000.preseqR[, 2])

  ## convert the input format for iNEXT
  S <- rep(Shakespeare_44000.hist[, 1], Shakespeare_44000.hist[, 2])
  Shakespeare_44000.sample <- sort(S, decreasing=TRUE)

  ## use iNEXT to predict the species accumulation curve
  Shakespeare_44000.size <- Shakespeare_44000.preseqR[, 1]
  Shakespeare_44000.iNEXT <- iNEXT(Shakespeare_44000.sample, q=0, datatype="abundance",
                                   size= Shakespeare_44000.size)

  ## extract useful columns (sample size, predicted values, lower bound, upper bound)
  Shakespeare_44000.iNEXT <- Shakespeare_44000.iNEXT$Accumulation[, c(1, 4, 5, 6)]
  ## combine all results into a matrix
  curves005.iNEXT <- cbind(curves005.iNEXT, Shakespeare_44000.iNEXT[, 2])
}
## the true accumulation curve
ShakespeareWordHist.true <- preseqR.interpolate.distinct(1.1e4, ShakespeareWordHist)
## MAD and RMSE of the expected words when extrapolate to 880,000 words
## MAD: mean absolute deviation; RMSE: root mean square error
MAD.preseqR <- mean( abs(curves005.preseqR[80, ] - ShakespeareWordHist.true[80, 2]) )
RMSE.preseqR <- sqrt(mean( (curves005.preseqR[80, ]-ShakespeareWordHist.true[80, 2])^2))
MAD.iNEXT <- mean(abs(curves005.iNEXT[80, ] - ShakespeareWordHist.true[80, 2]))
RMSE.iNEXT <- sqrt( mean( (curves005.iNEXT[80, ] - ShakespeareWordHist.true[80, 2])^2 ) )
## output MAD and RMSE
MAD.preseqR
```

```
## [1] 4939.906
```

```
RMSE.preseqR
```

```
## [1] 5710.927
```

```
MAD.iNEXT
```

```
## [1] 16916.19
```

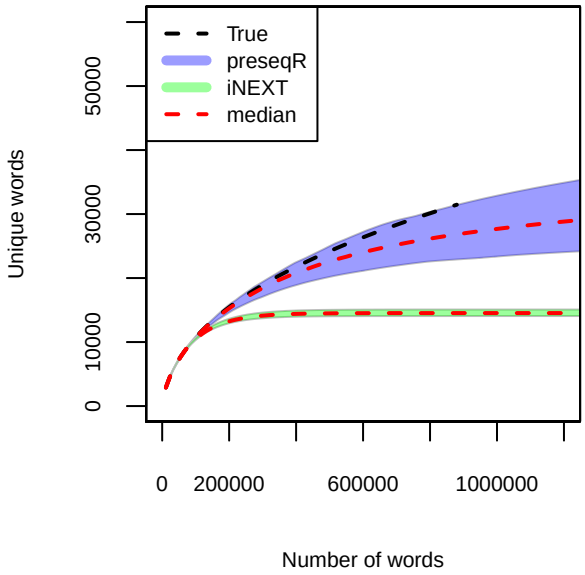
```
RMSE.iNEXT
```

```
## [1] 16920.28
```

We visualized estimated species accumulation curves by plotting the decile and median curve basis on results from 100 random samples.

```
## xy range
x.lim <- 1.2e6
y.lim <- 60000
## median curve
median.preseqR <- apply(curves005.preseqR, 1, median)
median.iNEXT <- apply(curves005.iNEXT, 1, median)
## 10% and 90% quantile
interval.preseqR <- apply(curves005.preseqR, 1, function(x) {
  c(quantile(x, 0.1)[[1]], quantile(x, 0.9)[[1]]) })
interval.iNEXT <- apply(curves005.iNEXT, 1, function(x) {
  c(quantile(x, 0.1)[[1]], quantile(x, 0.9)[[1]]) })

sample.size <- seq(1.1e4, 1e7, by=1.1e4)
## plot the xy-coordinate
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim),
     ylab = "Unique words", xlab = "Number of words",
     cex.lab=0.7, cex.axis=0.7)
## plot preseqR decile
par(new=TRUE)
plot(sample.size, interval.preseqR[2, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
plot(sample.size, interval.preseqR[1, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
polygon(c(sample.size, rev(sample.size)),
        c(interval.preseqR[2, ], rev(interval.preseqR[1, ])),
        col=rgb(0,0,255,100, maxColorValue = 255), border = NA,
        xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
## plot iNEXT decile
par(new=TRUE)
plot(sample.size, interval.iNEXT[2, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
plot(sample.size, interval.iNEXT[1, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
```

```
polygon(c(sample.size, rev(sample.size)),
        c(interval.iNEXT[2, ], rev(interval.iNEXT[1, ])),
        col=rgb(0,255,0,100, maxColorValue = 255), border = NA,
        xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
### plot true curve
par(new=TRUE)
plot(ShakespeareWordHist.true, type="l", lty=2, col="black", lwd=2, xlim=c(0, x.lim),
     ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
## plot median curve
par(new=TRUE)
plot(sample.size, median.preseqR, type="l", lty=2, col="red", lwd=2, xlim=c(0, x.lim),
     ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
plot(sample.size, median.iNEXT, type="l", lty=2, col="red", lwd=2, xlim=c(0, x.lim),
     ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
legend("topleft", c("True", "preseqR", "iNEXT", "median"), lty=c(2,1,1,2), lwd=c(2,5,5,2),
       col=c("black", rgb(0,0,255,100, maxColorValue = 255),
             rgb(0,255,0,100, maxColorValue = 255),"red"), cex=0.7)
```



We then calculated the MAD and RMSE of results basis on 10% initial experiments.

```
data(ShakespeareWordHist)
curves01.preseqR <- NULL
curves01.iNEXT <- NULL
for (count in 1:100) {
  set.seed(count)
  ind <- 1:sum(ShakespeareWordHist[, 2])
  n <- rep(ShakespeareWordHist[, 1], ShakespeareWordHist[, 2])
  X <- rep(ind, n)
  T <- sample(X, 88000, replace = FALSE)
  distinct <- sum(floor(ShakespeareWordHist[, 2]))
  X <- vector(length=as.integer(distinct), mode="numeric")
  for (i in T) {
    X[i] <- X[i] + 1
  }
```

```
  H <- vector(length=max(X), mode="numeric")
  for (i in X) {
    if (i > 0) {
      H[i] <- H[i] + 1
    }
  }
  ind <- which(H != 0)
  Shakespeare_88000.hist <- matrix(c(ind, H[ind]), ncol = 2)
  Shakespeare_88000.preseqR <- preseqR.rfa.species.accum.curve(
    Shakespeare_88000.hist, ss=1.1e4, max.extrapolation=1e7)
  curves01.preseqR <- cbind(curves01.preseqR, Shakespeare_88000.preseqR[, 2])

  S <- rep(Shakespeare_88000.hist[, 1], Shakespeare_88000.hist[, 2])
  Shakespeare_88000.sample <- sort(S, decreasing=TRUE)

  Shakespeare_88000.size <- Shakespeare_88000.preseqR[, 1]
  Shakespeare_88000.iNEXT <- iNEXT(Shakespeare_88000.sample, q=0, datatype="abundance",
                                   size= Shakespeare_88000.size)

  Shakespeare_88000.iNEXT <- Shakespeare_88000.iNEXT$Accumulation[, c(1, 4, 5, 6)]
  curves01.iNEXT <- cbind(curves01.iNEXT, Shakespeare_88000.iNEXT[, 2])
}

ShakespeareWordHist.true <- preseqR.interpolate.distinct(1.1e4, ShakespeareWordHist)

MAD.preseqR <- mean( abs(curves01.preseqR[80, ] - ShakespeareWordHist.true[80, 2]) )
RMSE.preseqR <- sqrt(mean((curves01.preseqR[80, ]-ShakespeareWordHist.true[80, 2])^2))
MAD.iNEXT <- mean(abs(curves01.iNEXT[80, ] - ShakespeareWordHist.true[80, 2]))
RMSE.iNEXT <- sqrt( mean( (curves01.iNEXT[80, ] - ShakespeareWordHist.true[80, 2])^2 ) )

MAD.preseqR
```

```
## [1] 3367.88
```

```
RMSE.preseqR
```

```
## [1] 3679.094
```

```
MAD.iNEXT
```

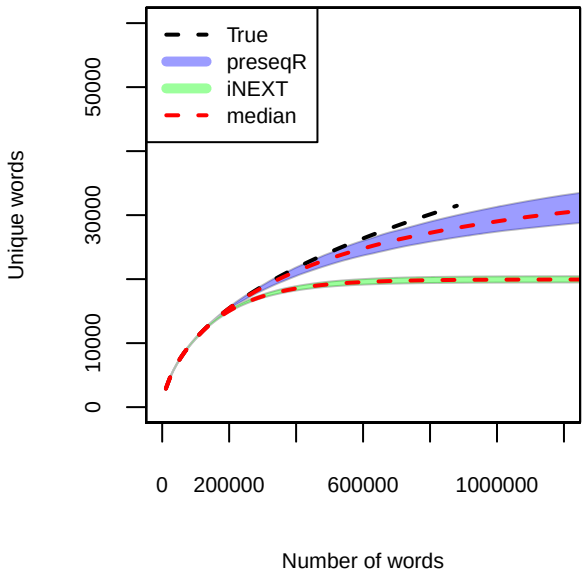
```
## [1] 11565.56
```

```
RMSE.iNEXT
```

```
## [1] 11571.02
```

```
x.lim = 1.2e6
y.lim = 60000
median.preseqR <- apply(curves01.preseqR, 1, median)
median.iNEXT <- apply(curves01.iNEXT, 1, median)
```

```
interval.preseqR <- apply(curves01.preseqR, 1, function(x) {
  c(quantile(x, 0.1)[[1]], quantile(x, 0.9)[[1]]) })
interval.iNEXT <- apply(curves01.iNEXT, 1, function(x) {
  c(quantile(x, 0.1)[[1]], quantile(x, 0.9)[[1]]) })

sample.size <- seq(1.1e4, 1e7, by=1.1e4)
## plot the xy-coordinate
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim), ylab = "Unique words",
     xlab = "Number of words", cex.lab=0.7, cex.axis=0.7)
## plot preseqR decile
par(new=TRUE)
plot(sample.size, interval.preseqR[2, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
plot(sample.size, interval.preseqR[1, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
polygon(c(sample.size, rev(sample.size)),
        c(interval.preseqR[2, ], rev(interval.preseqR[1, ])),
        col=rgb(0,0,255,100, maxColorValue = 255), border = NA,
        xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')

par(new=TRUE)
plot(sample.size, interval.iNEXT[2, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
plot(sample.size, interval.iNEXT[1, ], type="l", lty=1, col="grey", lwd=1,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
polygon(c(sample.size, rev(sample.size)),
        c(interval.iNEXT[2, ], rev(interval.iNEXT[1, ])),
        col=rgb(0,255,0,100, maxColorValue = 255), border = NA,
        xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
### plot true curve
par(new=TRUE)
plot(ShakespeareWordHist.true, type="l", lty=2, col="black", lwd=2,
     xlim=c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
## plot median curve
par(new=TRUE)
plot(sample.size, median.preseqR, type="l", lty=2, col="red", lwd=2, xlim=c(0, x.lim),
     ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
par(new=TRUE)
plot(sample.size, median.iNEXT, type="l", lty=2, col="red", lwd=2, xlim=c(0, x.lim),
     ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
legend("topleft", c("True", "preseqR", "iNEXT", "median"), lty=c(2,1,1,2), lwd=c(2,5,5,2),
       col=c("black", rgb(0,0,255,100, maxColorValue = 255),
             rgb(0,255,0,100, maxColorValue = 255), "red"), cex=0.7)
```



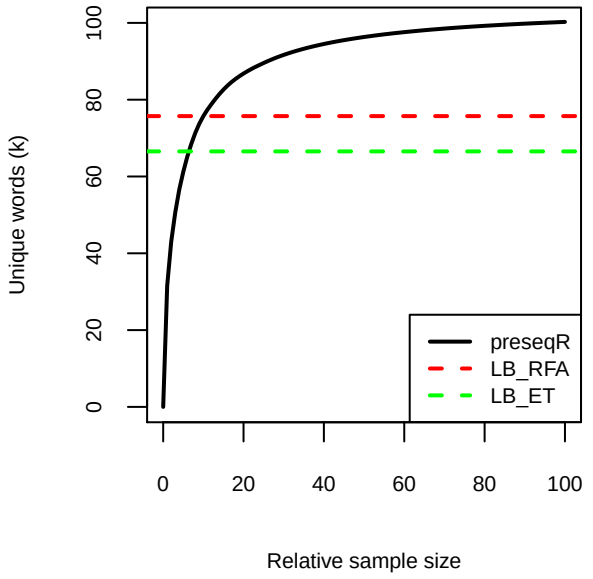
Extrapolating from all of Shakespeare's observed vocabulary

We used the totality of Shakespeare's known vocabulary to estimate how many words he knew.

```
set.seed(123456)

## use preseqR to predict the accumulation curve
ShakespeareWordHist.preseqR <- preseqR.rfa.species.accum.curve(ShakespeareWordHist)
## print out the first twenty rows
ShakespeareWordHist.preseqR[1:20, ]
```

```
##       sample.size yield.estimate lower.0.95CI uppper.0.95CI
##  [1,]      884647       31565.27     30924.63      32219.18
##  [2,]     1769294       43037.89     42198.94      43893.51
##  [3,]     2653941       50619.31     49141.06      52142.03
##  [4,]     3538588       56556.90     53622.41      59651.99
##  [5,]     4423235       61204.64     56614.18      66167.30
##  [6,]     5307882       65176.26     58886.65      72137.65
##  [7,]     6192529       68488.74     60495.47      77538.15
##  [8,]     7077176       71366.31     61688.33      82562.63
##  [9,]     7961823       73763.43     62435.92      87146.04
## [10,]     8846470       75722.53     62790.27      91318.30
## [11,]     9731117       77379.28     62890.84      95205.50
## [12,]    10615764       78832.26     62836.99      98899.16
## [13,]    11500411       80191.69     62736.91     102502.78
## [14,]    12385058       81473.60     62605.22     106028.66
## [15,]    13269705       82622.11     62386.59     109421.16
## [16,]    14154352       83649.87     62092.36     112691.80
## [17,]    15038999       84586.20     61749.66     115868.26
## [18,]    15923646       85395.54     61323.39     118917.09
## [19,]    16808293       86133.88     60866.02     121891.43
## [20,]    17692940       86812.65     60387.12     124802.04
```

```
## plot the curve
#W = 3.5; H = 4
#pdf("AllShakespeareWords.pdf", width=W, height=H)
## The magic number 884647 is the total number of words in Shakespeare's known works
t <- ShakespeareWordHist.preseqR[1:100, 1] / 884647
Y <- ShakespeareWordHist.preseqR[1:100, 2] / 1000
t = c(0, t)
Y = c(0, Y)
## label the xy-coordinate
plot(t, Y, type="l", lty=1, col=1, lwd=2, ylim=c(0, 100), xlim=c(0, 100),
     xlab="Relative sample size", ylab="Unique words (k)",
     cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
## preseqR lower bound
abline(a = ShakespeareWordHist.preseqR[10, 2]/1000, b=0, lty=2, lwd=2, col="red")
## Efron & Thisted lower bound
abline(a=(35000 + 31534)/1000, b=0, lty=2, lwd =2, col="green")
legend("bottomright", c("preseqR", "LB_RFA", "LB_ET"), lty=c(1,2,2), lwd=c(2,2,2),
       col=c("black", "red", "green"), cex=CEX)
```




```
#dev.off()

## predict the accumulation curve by running the quick mode
ShakespeareWordHist.preseqR.quick <- preseqR.rfa.curve(ShakespeareWordHist)
## print out the first twenty rows
ShakespeareWordHist.preseqR.quick$estimates[1:20,]
```

```
##       sample.size yield.estimate
##  [1,]      884647       31534.00
##  [2,]     1769294       42957.39
##  [3,]     2653941       50304.05
##  [4,]     3538588       55120.43
##  [5,]     4423235       58895.62
##  [6,]     5307882       61961.06
##  [7,]     6192529       64484.77
```

```
##  [8,]     7077176       66592.67
##  [9,]     7961823       68377.48
## [10,]     8846470       69907.29
## [11,]     9731117       71232.72
## [12,]    10615764       72391.99
## [13,]    11500411       73414.40
## [14,]    12385058       74322.79
## [15,]    13269705       75135.21
## [16,]    14154352       75866.08
## [17,]    15038999       76527.08
## [18,]    15923646       77127.78
## [19,]    16808293       77676.07
## [20,]    17692940       78178.51
```

```
## print out the constructed continued fraction
ShakespeareWordHist.preseqR.quick$continued.fraction
```

```
## CONTINUED FRACTION APPROXIMATION :
##
## DEGREE   10
##
## COEFFICIENTS:
##
## a_0 = 14376.0000
## a_1 = 0.3021
## a_2 = 0.2256
## a_3 = 0.2586
## a_4 = 0.2827
## a_5 = 0.7614
## a_6 = -2.1916
## a_7 = 0.5241
## a_8 = -0.4320
## a_9 = 1.5379
```

```
## calculate the function value of the continued fraction when t=10,
## Number of new words we expect to observe if 9 additional volumes
## of Shakespeare's work are discovered
preseqR.rfa.estimate(ShakespeareWordHist.preseqR.quick$continued.fraction, 10)
```

```
## [1] 38373.29
```

Example 2: Annotated species in metagenomics

We applied preseqR to the problem of estimating species accumulation curves in a metagenomic sequencing experiment. We examined species abundance data collected and calculated by Yatsunenko et al. (2012), downloaded from MG-RAST with ID 4461119.3 (Meyer et al., 2008). The data contains 1712 unique annotated species with a total observed abundance of 156608.

To evaluate our method we first downsampled approximately 5% of the species abundance data, then used this as an initial experiment to extrapolate and compared against the curve of the whole data.

First we generated the subsample.

```
library(preseqR)
set.seed(123456)
species_abundance.hist <-
read.table(
"http://smithlabresearch.org/downloads/preseqR_examples/4461119.3_species_abundance.hist")
ind <- 1:sum(species_abundance.hist[, 2])
n <- rep(species_abundance.hist[, 1], species_abundance.hist[, 2])
X <- rep(ind, n)
T <- sample(X, 7830, replace = FALSE)
species_abundance.hist[,2] = floor(species_abundance.hist[,2])
distinct <- sum(species_abundance.hist[, 2])
X <- vector(length=as.integer(distinct), mode="numeric")
for (i in T) {
  X[i] <- X[i] + 1
}

H <- vector(length=max(X), mode="numeric")
for (i in X) {
  if (i > 0) {
    H[i] <- H[i] + 1
  }
}
ind <- which(H != 0)
species_abundance_sub.hist <- matrix(c(ind, H[ind]), ncol = 2)
```

We used this subsample to predict the accumulation curve based on 5% sample.

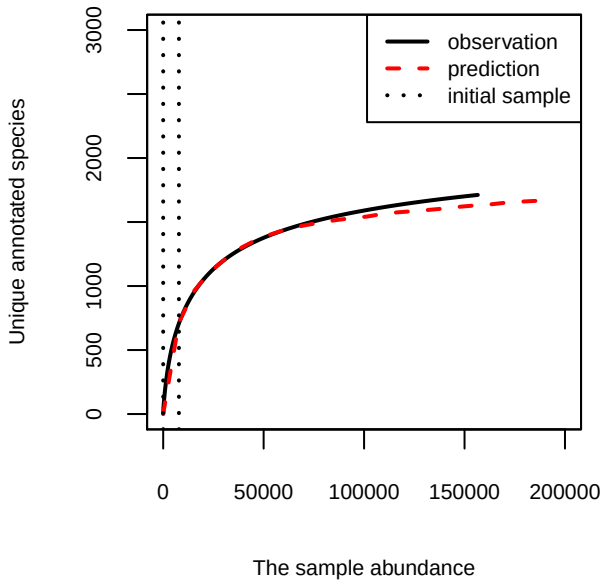
```
set.seed(123456)
## predict the accumulation curve based on the sub sample
species_abundance_sub.preseqR <- preseqR.rfa.species.accum.curve(
  species_abundance_sub.hist, ss=7800, max.extrapolation=2e5)
species_abundance_sub.preseqR
```

```
##       sample.size yield.estimate lower.0.95CI uppper.0.95CI
##  [1,]        7800       714.9249     508.2864      1005.570
##  [2,]       15600       966.8628     731.1550      1278.557
##  [3,]       23400      1112.2853     864.5165      1431.064
##  [4,]       31200      1214.1528     955.4168      1542.957
##  [5,]       39000      1300.5590    1028.3483      1644.826
##  [6,]       46800      1364.1280    1076.0837      1729.275
##  [7,]       54600      1408.9453    1103.6362      1798.715
##  [8,]       62400      1448.8226    1125.3176      1865.329
##  [9,]       70200      1473.4592    1131.7370      1918.363
## [10,]       78000      1495.3580    1135.4660      1969.320
## [11,]       85800      1514.6196    1136.8663      2017.891
## [12,]       93600      1529.7672    1134.6875      2062.407
## [13,]      101400      1542.8082    1130.9698      2104.616
## [14,]      109200      1563.3333    1134.9317      2153.443
## [15,]      117000      1576.2514    1132.1502      2194.557
## [16,]      124800      1585.2391    1126.1764      2231.429
## [17,]      132600      1595.4619    1121.9200      2268.877
## [18,]      140400      1606.2360    1118.7043      2306.234
## [19,]      148200      1620.1345    1118.9244      2345.856
```

```
## [20,]      156000      1630.6923    1116.4603      2381.775
## [21,]      163800      1639.3742    1112.6841      2415.374
## [22,]      171600      1653.2988    1114.2363      2453.157
## [23,]      179400      1660.3875    1109.7978      2484.134
## [24,]      187200      1666.6152    1104.9498      2513.785
## [25,]      195000      1672.4027    1100.0665      2542.511
```

```
## load the true accumulation curve provided by MG-RAST
species_abundance.true <- read.table(
"http://smithlabresearch.org/downloads/preseqR_examples/4461119.3_rarefaction_curve.tsv",
sep="\t")

## calculate the total abundance
species_abundance.total.abundance <- species_abundance.hist[, 1] %*%
                                     species_abundance.hist[, 2]
## calculate the total number of sequences
species_abundance.total.sequences <-
  species_abundance.true[length(species_abundance.true[,1]), 1]
## we assume the abundance is linear with the number of sequences
## the coefficient is the slope
species_abundance.slope <- species_abundance.total.abundance /
                           species_abundance.total.sequences

## convert the true curve as a function of the abundance by multiplying a constant
species_abundance.true[, 1] <- species_abundance.true[, 1] * species_abundance.slope

## compare the true curve with our predictions
## and calculate the scale error
I <- 50 * (1:20) + 1
error <- (species_abundance_sub.preseqR[1:20, 2] - species_abundance.true[I, 2]) /
  species_abundance.true[I, 2]

## plot the true and predicated curves
# W = 3.5; H = 4
# pdf("metaSpecies.pdf", width=W, height=H)
## plot
x.lim= 2e5
y.lim= 3000
CEX = 0.7
CEX.AXIS = 0.7
CEX.LAB = 0.7
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim),
     xlab = "The sample abundance", ylab = "Unique annotated species",
     cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)
plot(species_abundance.true, type="l", lty=1, col="black", lwd=2, ann=FALSE,
     xaxt = 'n', yaxt = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim))
## add a zero point
species_abundance_sub.preseqR = rbind(c(0,0,0,0), species_abundance_sub.preseqR)
lines(species_abundance_sub.preseqR[, 1:2], lty=2, lwd=2, col="red", xlim = c(0, x.lim),
ylim=c(0, y.lim),ann=FALSE, xaxt="n", yaxt="n")
abline(v=0, lty=3, lwd=2)
abline(v= species_abundance_sub.hist[,1] %*% species_abundance_sub.hist[,2], lty=3, lwd=2)
legend("topright", c("observation","prediction", "initial sample"), lty=c(1,2,3),
```

```
       lwd=c(2,2,2), col=c("black","red","black"), cex=CEX)
```




```
#dev.off()
```

Example 3: Age-related decrease in TCR repertoire

We applied preseqR to investigate age-related decreases in T cell receptor (TCR) repertoire diversity. The data sets are profiles of TCR β repertoires in 39 healthy donors aged 6-90 years of age (y) from Britanova et al. (2014). For each donor, the data is summarized as frequencies of TCR β CDR3 clonotypes. The accumulation curve in this study is the expected number of unique TCR β CDR3 clonotypes as a function of TCR β cDNA molecules sequenced.

We downsampled one million cDNA samples from the whole data, which combined data sets from all 39 donors. Later we used these data set to eveluate the performance of preseqR on estimating the TCR repertoire size.

```
## downsample 1 million molecules
set.seed(123456)
overallTCR.hist <- read.table(
  "http://smithlabresearch.org/downloads/preseqR_examples/overallTCR.hist", header=TRUE)
ind <- 1:sum(overallTCR.hist[, 2])
n <- rep(overallTCR.hist[, 1], overallTCR.hist[, 2])
X <- rep(ind, n)
T <- sample(X, 1e6, replace = FALSE)
overallTCR.hist[, 2] <- floor(overallTCR.hist[, 2])
distinct <- sum(floor(overallTCR.hist[, 2]))
X <- vector(length=as.integer(distinct), mode="numeric")
for (i in T) {
  X[i] <- X[i] + 1
}

H <- vector(length=max(X), mode="numeric")
for (i in X) {
```

```
  if (i > 0) {
    H[i] <- H[i] + 1
  }
}
ind <- which(H != 0)
overallTCRsubsample.hist <- matrix(c(ind, H[ind]), ncol = 2)
```

For each donor, we predicted an accumulation curve.

```
## load the package
library(preseqR)
## set a random seed
set.seed(123456)

## histogram for donors in group 1
group1.files = c("L1_9_M11.hist", "L1_10_M9.hist", "L1_11_F6.hist", "L1_12_M16.hist",
                 "L1_13_M16.hist", "L1_14_M10.hist", "L3_36_M25.hist", "L3_37_F22.hist",
                 "L3_38_F24.hist", "L3_39_M20.hist", "L3_40_F21.hist")
group1.curves = list()
count = 1
for (f in group1.files) {
  ## load the data
  group1.hist = read.table(
    paste("http://smithlabresearch.org/downloads/preseqR_examples/", f, sep=""),
    header=TRUE, sep="\t")
  ## predict the accumulation curve based on the sample
  res.rfa = preseqR.rfa.species.accum.curve(group1.hist, ss=1e5, max.extrapolation=2e7)
  group1.curves[[count]] = res.rfa
  count = count + 1
}
# donors of group 2
group2.files = c("L2_1_F36.hist", "L2_2_F43.hist", "L2_3_F43.hist", "L2_4_F39.hist",
                 "L2_5_F34.hist", "L3_19_M36.hist", "L3_20_M37.hist", "L3_21_M43.hist",
                 "L3_22_M39.hist", "L3_23_F43.hist")
group2.curves = list()
count = 1
for (f in group2.files) {
  group2.hist = read.table(
    paste("http://smithlabresearch.org/downloads/preseqR_examples/", f, sep=""),
    header=TRUE, sep="\t")
  res.rfa = preseqR.rfa.species.accum.curve(group2.hist, ss=1e5, max.extrapolation=2e7)
  group2.curves[[count]] = res.rfa
  count = count + 1
}
# donors of group 3
group3.files = c("L3_26_M66.hist", "L3_27_M64.hist", "L3_29_F62.hist", "L3_32_M61.hist",
                 "L3_33_F61.hist", "L3_34_F61.hist", "L3_35_F61.hist")
group3.curves = list()
count = 1
for (f in group3.files) {
  group3.hist = read.table(
    paste("http://smithlabresearch.org/downloads/preseqR_examples/", f, sep=""),
    header=TRUE, sep="\t")
```

```
  res.rfa = preseqR.rfa.species.accum.curve(group3.hist, ss=1e5, max.extrapolation=2e7)
  group3.curves[[count]] = res.rfa
  count = count + 1
}
# donors of group 4
group4.files = c("L1_15_F74.hist", "L1_16_M75.hist", "L1_17_F73.hist", "L1_18_M71.hist",
                 "L2_6_F86.hist", "L2_7_F87.hist", "L3_24_F89.hist", "L3_25_M85.hist",
                 "L3_28_M90.hist", "L3_30_M87.hist", "L3_31_M87.hist")
group4.curves = list()
count = 1
for (f in group4.files) {
  group4.hist = read.table(
    paste("http://smithlabresearch.org/downloads/preseqR_examples/", f, sep=""),
    header=TRUE, sep="\t")
  res.rfa = preseqR.rfa.species.accum.curve(group4.hist, ss=1e5, max.extrapolation=2e7)
  group4.curves[[count]] = res.rfa
  count = count + 1
}
```

We created the functions to construct accumulation regions, that were constructed by using a set of accumulation curves.

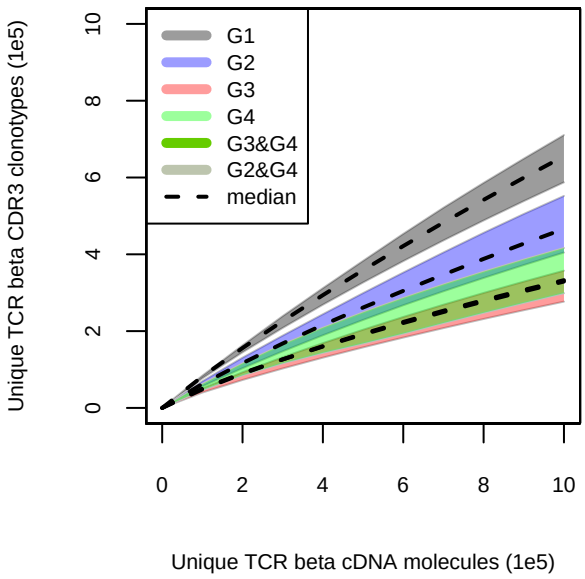
```
### function to plot the accumulation region for a group of donors
colorIntervalRegion <- function(r, g, b, alpha, x.lim, y.lim, prob = 0.3, curves) {
  X = 1:x.lim
  Y = matrix(data = 0, nrow = x.lim, ncol = length(curves))
  t = 0
  ## save all accumulation curves into a matrix Y
  for (res in curves) {
    t = t + 1
    Y[, t] = res[, 2][X]
  }
  ## add zero as an initial point; convenience for the plot
  X = c(0,X)
  ## getting the quantiles for each given sample size
  interval = apply(Y, 1, function(x) {
    c(quantile(x, prob)[[1]], quantile(x, 1 - prob)[[1]]) })
  ## scale the number
  interval = interval / 1e5
  lower.interval = c(0, interval[1, ])
  upper.interval = c(0, interval[2, ])
  ## plot the region
  plot(X, lower.interval, type="l", lty=1, col="grey", lwd=1, xlim=c(0, x.lim),
       ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
  par(new=TRUE)
  plot(X, upper.interval, type="l", lty=1, col="grey", lwd=1, xlim=c(0, x.lim),
       ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n')
  par(new=TRUE)
  polygon(c(X, rev(X)), c(upper.interval, rev(lower.interval)),
          col = rgb(r, g, b, alpha, maxColorValue = 255), border = NA)
  par(new=TRUE)
}
```

```
### function to plot the medion for a group of donors
getMedianPlot <- function(col, x.lim, y.lim, curves) {
  X = 1:x.lim
  ## save all accumulation curves into a matrix Y
  Y = matrix(data = 0, nrow = x.lim, ncol = length(curves))
  t = 0
  for (res in curves) {
    t = t + 1
    Y[, t] = res[, 2][X]
  }
  ## getting the median for each given sample size
  Z = apply(Y, 1, median)
  Z = Z / 1e5
  ## add zero as the begining point
  X = c(0, X)
  Z = c(0, Z)
  ## plot the curve
  plot(X, Z, type="l", lty=2, col=col, lwd=2, xlim=c(0, x.lim), ylim=c(0, y.lim),
       ann=FALSE, xaxt = 'n', yaxt = 'n')
  par(new=TRUE)
}
```

Accumulation regions through interpolation

```
#W = 3.5; H = 4
#pdf("repertoireInter.pdf", width=W, height=H)
## magic number 10 is the relative size of full experiment to the initial experiment
x.lim = 10
## scaled upperbound for y-axis
y.lim = 10
## font size
CEX = 0.7
CEX.AXIS = 0.7
CEX.LAB = 0.7

par(new = FALSE)
## set the xlab and ylab
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0,y.lim),
     ylab = "Unique TCR beta CDR3 clonotypes (1e5)",
     xlab = "Unique TCR beta cDNA molecules (1e5)",
     cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)

## plot accumulation regions for each group
colorIntervalRegion(0, 0, 0, 100, x.lim, y.lim, curves=group1.curves)
colorIntervalRegion(0,0 ,255, 100, x.lim, y.lim, curves=group2.curves)
colorIntervalRegion(255, 0, 0, 100, x.lim, y.lim, curves=group3.curves)
colorIntervalRegion(0,255,0, 100, x.lim, y.lim, curves=group4.curves)

## the median curve
getMedianPlot("black", x.lim, y.lim, group1.curves)
getMedianPlot("black", x.lim, y.lim, group2.curves)
```

```
getMedianPlot("black", x.lim, y.lim, group3.curves)
getMedianPlot("black", x.lim, y.lim, group4.curves)
legend("topleft", c("G1","G2","G3","G4", "G3&G4", "G2&G4", "median"),
       lty=c(1,1,1,1,1,1,2), lwd=c(5,5,5,5,5,5,2),
       col=c(rgb(0,0,0,100, maxColorValue = 255), rgb(0,0,255,100, maxColorValue = 255),
             rgb(255,0,0,100, maxColorValue = 255), rgb(0,255,0,100, maxColorValue = 255),
             rgb(102,204,0,255, maxColorValue = 255),
             rgb(85,107,47,100,maxColorValue = 255), "black"), cex=CEX)
```

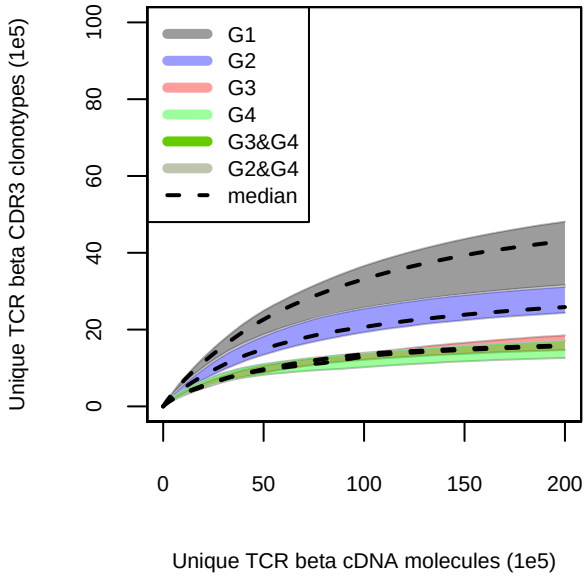



```
# dev.off()
```

Accumulation regions using extrapolation

```
### plot accumulation regions for each group
### the region from interpolation and extrapolation
## pdf("repertoireExtra.pdf", width=W, height=H)
## extrapolating to 20 times;
x.lim = 200
y.lim = 100

par(new = FALSE)
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0,y.lim),
     ylab = "Unique TCR beta CDR3 clonotypes (1e5)",
     xlab = "Unique TCR beta cDNA molecules (1e5)", cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)

colorIntervalRegion(0, 0, 0, 100, x.lim, y.lim, curves=group1.curves)
colorIntervalRegion(0,0 ,255, 100, x.lim, y.lim, curves=group2.curves)
colorIntervalRegion(255, 0, 0, 100, x.lim, y.lim, curves=group3.curves)
colorIntervalRegion(0,255,0, 100, x.lim, y.lim, curves=group4.curves)

## the median curve
```

```
getMedianPlot("black", x.lim, y.lim, group1.curves)
getMedianPlot("black", x.lim, y.lim, group2.curves)
getMedianPlot("black", x.lim, y.lim, group3.curves)
getMedianPlot("black", x.lim, y.lim, group4.curves)

legend("topleft", c("G1","G2","G3","G4", "G3&G4", "G2&G4", "median"),
       lty=c(1,1,1,1,1,1,2), lwd=c(5,5,5,5,5,5,2),
       col=c(rgb(0,0,0,100, maxColorValue = 255), rgb(0,0,255,100, maxColorValue = 255),
             rgb(255,0,0,100, maxColorValue = 255), rgb(0,255,0,100, maxColorValue = 255),
             rgb(102,204,0,255, maxColorValue = 255),
             rgb(85,107,47,100,maxColorValue = 255), "black"),
       cex=CEX)
```




```
# dev.off()
```

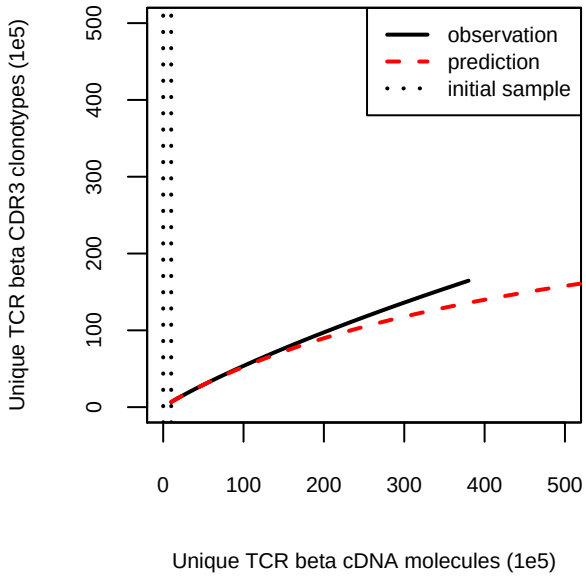
Predicting the full TCR diversity

```
library(preseqR)
set.seed(123456)

## predict the accumulation curve based on the sub sample
overallTCR.preseqR <- preseqR.rfa.species.accum.curve(overallTCRsubsample.hist, ss=1e6,
                                                      max.extrapolation=1e8)
## display the first 20 rows
overallTCR.preseqR[1:20, ]
```

```
##       sample.size yield.estimate lower.0.95CI uppper.0.95CI
##  [1,]     1.0e+06       668519.6     647275.9      690460.6
##  [2,]     2.0e+06      1270014.4    1230193.4     1311124.4
##  [3,]     3.0e+06      1839839.5    1783069.6     1898416.9
##  [4,]     4.0e+06      2382196.6    2306289.4     2460602.2
##  [5,]     5.0e+06      2902256.8    2793526.2     3015219.5
```

```
##  [6,]     6.0e+06      3404168.2    3246034.6     3570005.3
##  [7,]     7.0e+06      3882130.1    3664046.9     4113193.7
##  [8,]     8.0e+06      4346420.1    4059092.0     4654087.0
##  [9,]     9.0e+06      4796815.7    4431026.9     5192800.9
## [10,]     1.0e+07      5231973.3    4779874.6     5726833.2
## [11,]     1.1e+07      5665399.8    5121119.6     6267526.8
## [12,]     1.2e+07      6071764.4    5431463.4     6787548.9
## [13,]     1.3e+07      6468383.3    5729620.9     7302399.9
## [14,]     1.4e+07      6849399.2    6010804.8     7804989.5
## [15,]     1.5e+07      7226621.6    6287419.2     8306120.1
## [16,]     1.6e+07      7599429.0    6559283.2     8804517.2
## [17,]     1.7e+07      7965677.3    6824565.8     9297590.1
## [18,]     1.8e+07      8309724.9    7068020.3     9769571.2
## [19,]     1.9e+07      8644423.9    7302522.0    10232911.9
## [20,]     2.0e+07      8971099.0    7529493.9    10688715.5
```

```
## calculate the true accumulation curve
overallTCR.true <- preseqR.interpolate.distinct(overallTCR.hist, ss=1e6)
## scale the result
overallTCR.preseqR <- overallTCR.preseqR / 1e5
overallTCR.true <- overallTCR.true / 1e5
l <- length(overallTCR.true[, 1])
## calculate the scaled error
error <- (overallTCR.preseqR[1:l, 2] - overallTCR.true[, 2]) / overallTCR.true[, 2]

## plot
# W = 3.5; H = 4
# pdf("overallTCR.pdf", width=W, height=H)
CEX = 0.7
CEX.AXIS = 0.7
CEX.LAB = 0.7
plot(0, 0, type = 'n', xlim = c(0, 500), ylim=c(0,500),
     ylab = "Unique TCR beta CDR3 clonotypes (1e5)",
     xlab = "Unique TCR beta cDNA molecules (1e5)", cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)
plot(overallTCR.true, type="l", lty=1, col="black", lwd=2, xlim=c(0, 500), ylim=c(0, 500),
     ann=FALSE, xaxt = 'n', yaxt = 'n')
lines(overallTCR.preseqR[, 1:2], lty=2, lwd=2, col="red", xlim=c(0, 500), ylim=c(0, 500),
      ann=FALSE, xaxt="n", yaxt="n")
abline(v=0, lty=3, lwd=2)
abline(v=10, lty=3, lwd=2)
legend("topright", c("observation","prediction", "initial sample"), lty=c(1,2,3),
       lwd=c(2,2,2), col=c("black","red","black"), cex=CEX)
```




```
# dev.off()
```

Example 4: Extrapolating distinct k-mers for assembly with NGS data

We investigated the diversity of k-mers as a function of total bases sequenced. A data set is selected from the Assemblathon 2 (Bradnam et al., 2013), whole genome sequencing of a male budgerigar. It is downloaded from the Sequenced Read Archive in NCBI (accession number ERX218679). This experiment contains approximately 161 million 150 base pair length read pairs for a total of approximately 48 billion total bases.

We subsampled two experiments from the sequenced reads, randomly downsampling 1% and 10% from the original sequenced reads using a custom python script We counted the k-mers using Jellyfish (Marçais and Kingsford, 2011) and chose k = 31 since it was the default setting for the widely used assembly algorithm Velvet (Zerbino and Birney, 2008).

Predicting distinct K-mers based on subsamples

```
library(preseqR)
set.seed(123456)

## k = 31
## load the data
n <- read.table(
  "http://smithlabresearch.org/downloads/preseqR_examples/ERR244145_k31.hist")

## total sample size
n[, 1] %*% n[, 2]
```

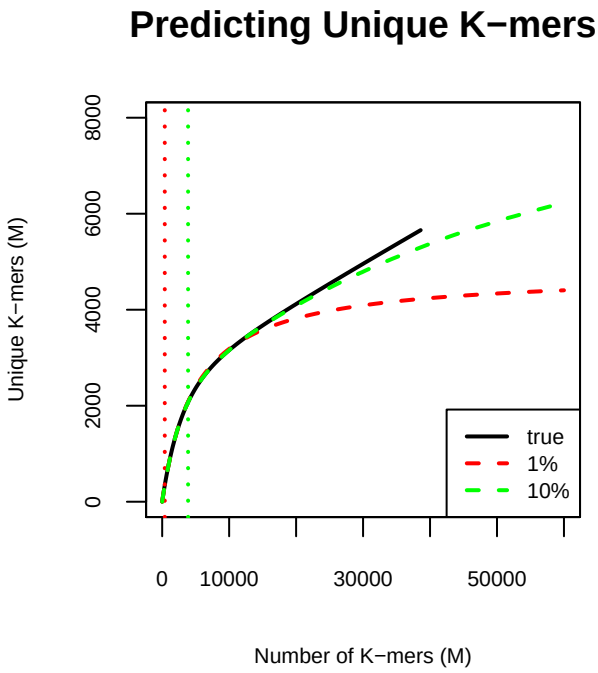
```
##              [,1]
## [1,] 38668122906
```

```
## subsampling the data as a golden stardard for the accumulation curve
ERR244145_k31.true_curve <- preseqR.interpolate.distinct(n, ss=1e8)

## predicting the accumulation curve using 1% random sample
ERR244145_k31_0.01downsampled_hist <- read.table(
  "http://smithlabresearch.org/downloads/preseqR_examples/ERR244145_001_k31.hist")
ERR244145_k31_0.01downsampled.preseqR <- preseqR.rfa.species.accum.curve(
  ERR244145_k31_0.01downsampled_hist, ss=1e8, max.extrapolation=6e10)
## predicting the accumulation curve using 10% random sample
ERR244145_k31_0.1downsampled_hist <- read.table(
  "http://smithlabresearch.org/downloads/preseqR_examples/ERR244145_01_k31.hist")
ERR244145_k31_0.1downsampled.preseqR <-
  preseqR.rfa.species.accum.curve(ERR244145_k31_0.1downsampled_hist, ss=1e8,
                                  max.extrapolation=6e10)

## calculate the initial sample sizes
total.sample1 <- ERR244145_k31_0.01downsampled_hist[, 1] %*%
                 ERR244145_k31_0.01downsampled_hist[, 2]
total.sample2 <- ERR244145_k31_0.1downsampled_hist[, 1] %*%
                 ERR244145_k31_0.1downsampled_hist[, 2]

l = length(ERR244145_k31.true_curve[, 2])
## scaled errors for predictions
ERR244145_k31_0.01downsampled.preseqR.rel_error <-
  (ERR244145_k31_0.01downsampled.preseqR[1:l, 2] - ERR244145_k31.true_curve[, 2]) /
  ERR244145_k31.true_curve[, 2]
ERR244145_k31_0.1downsampled.preseqR.rel_error <-
  (ERR244145_k31_0.1downsampled.preseqR[1:l, 2] - ERR244145_k31.true_curve[, 2]) /
  ERR244145_k31.true_curve[, 2]
```

Next we plotted the estimated accumulation curves.

```
#W = 3.5; H = 4
#pdf("K31.pdf", width=W, height=H)
x.lim = 6e4
y.lim = 0.8e4
CEX = 0.7
CEX.AXIS = 0.7
CEX.LAB = 0.7
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim), ylab = "Unique K-mers (M)",
     xlab = "Number of K-mers (M)", cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)

plot(rbind(c(0,0), ERR244145_k31.true_curve / 1e6), type="l", lty=1, col="black", lwd=2,
     xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n', xaxt="n",
     yaxt="n")
lines(rbind(c(0, 0), ERR244145_k31_0.01downsampled.preseqR[, 1:2] / 1e6), lty=2, lwd=2,
      col="red", xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n", yaxt="n")
abline(v=total.sample1 / 1e6, lty=3, lwd=2, col="red")
lines(rbind(c(0, 0), ERR244145_k31_0.1downsampled.preseqR[, 1:2] / 1e6), lty=2, lwd=2,
      col="green", xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n", yaxt="n")
abline(v=total.sample2 / 1e6, lty=3, lwd=2, col="green")
title(main="Predicting Unique K-mers")
```

```
legend("bottomright", c("true","1%","10%"), lty=c(1,2,2), lwd=c(2,2,2),
       col=c("black","red","green"), cex=CEX)
```




```
#dev.off()
```

Asymptotically linear rational function approximations to the Good-Toulmin power series

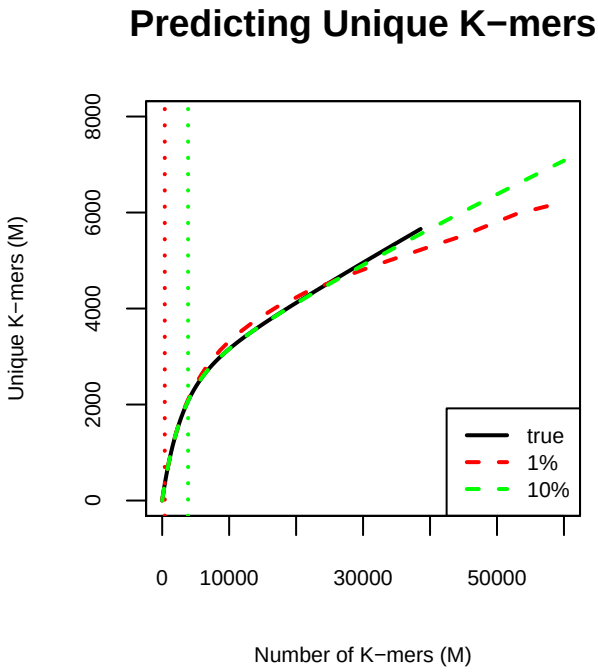
We chose rational function approximations that were asymptotically a linear function as the argument t, the relative sample size, goes to infinity.

```
## adding the asympototic linear constraint to rational function approximation
ERR244145_k31_0.01downsampled.linear_preseqR <- preseqR.rfa.species.accum.curve(
  ERR244145_k31_0.01downsampled_hist, ss=1e8, max.extrapolation=6e10, asym.linear=1)
ERR244145_k31_0.1downsampled.linear_preseqR <- preseqR.rfa.species.accum.curve(
  ERR244145_k31_0.1downsampled_hist, ss=1e8, max.extrapolation=6e10, asym.linear=1)

## scaled errors for predictions by setting asym.linear=1
ERR244145_k31_0.01downsampled.linear_preseqR.rel_error <-
  (ERR244145_k31_0.01downsampled.linear_preseqR[1:1, 2] - ERR244145_k31.true_curve[, 2]) /
  ERR244145_k31.true_curve[, 2]
ERR244145_k31_0.1downsampled.linear_preseqR.rel_error <-
  (ERR244145_k31_0.1downsampled.linear_preseqR[1:1, 2] - ERR244145_k31.true_curve[, 2]) /
  ERR244145_k31.true_curve[, 2]
```

Next we plotted estimated accumulation curves.

```
#pdf("K31_asym_linear.pdf", width=W, height=H)
plot(0, 0, type = 'n', xlim = c(0, x.lim), ylim=c(0, y.lim), ylab = "Unique K-mers (M)",
```

```
     xlab = "Number of K-mers (M)", cex.lab=CEX.LAB, cex.axis=CEX.AXIS)
par(new=TRUE)

plot(rbind(c(0,0), ERR244145_k31.true_curve / 1e6), type="l", lty=1, col="black", lwd=2,
     xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt = 'n', yaxt = 'n', xaxt="n",
     yaxt="n")
lines(rbind(c(0, 0), ERR244145_k31_0.01downsampled.linear_preseqR[, 1:2] / 1e6), lty=2,
      lwd=2, col="red", xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n",
      yaxt="n")
abline(v=total.sample1 / 1e6, lty=3, lwd=2, col="red")
lines(rbind(c(0, 0), ERR244145_k31_0.1downsampled.linear_preseqR[, 1:2] / 1e6), lty=2,
      lwd=2, col="green", xlim = c(0, x.lim), ylim=c(0, y.lim), ann=FALSE, xaxt="n",
      yaxt="n")
abline(v=total.sample2 / 1e6, lty=3, lwd=2, col="green")
title(main="Predicting Unique K-mers")

legend("bottomright", c("true", "1%", "10%"), lty=c(1,2, 2),lwd=c(2,2, 2),
       col=c("black","red", "green"), cex=CEX)
```




```
#dev.off()
```

downsample.py

Below is the code we used for subsampling reads in Example 4.

```
#!/usr/bin/python

import sys
import random
```

```
random.seed(123456)

if len(sys.argv) != 4 and len(sys.argv) != 6:
  sys.stderr.write(
    "usage: downSample.py [fastq1] [fastq2] [outputfile] [prob] for paired end\n")
  sys.stderr.write("Or downSample.py [fastq] [outputfile] [prob] for single end\n")
  sys.exit(1)

if len(sys.argv) == 4:
  f = open(sys.argv[1], "r")
  outf = open(sys.argv[2], "w")
  p = float(sys.argv[3])
  line1 = f.readline()
  line2 = f.readline()
  line3 = f.readline()
  line4 = f.readline()
  while line1:
    p1 = random.random()
    if p1 < p:
      outf.write(line1)
      outf.write(line2)
      outf.write(line3)
      outf.write(line4)

    line1 = f.readline()
    line2 = f.readline()
    line3 = f.readline()
    line4 = f.readline()

  f.close()
  outf.close()

elif len(sys.argv) == 6:
  f1 = open(sys.argv[1], "r")
  f2 = open(sys.argv[2], "r")
  outf1 = open(sys.argv[3], "w")
  outf2 = open(sys.argv[4], "w")
  p = float(sys.argv[5])
  line1 = f1.readline()
  line2 = f1.readline()
  line3 = f1.readline()
  line4 = f1.readline()
  line5 = f2.readline()
  line6 = f2.readline()
  line7 = f2.readline()
  line8 = f2.readline()
  while line1:
    p1 = random.random()
    if p1 < p:
      outf1.write(line1)
      outf1.write(line2)
      outf1.write(line3)
      outf1.write(line4)
```

```
      outf2.write(line5)
      outf2.write(line6)
      outf2.write(line7)
      outf2.write(line8)

    line1 = f1.readline()
    line2 = f1.readline()
    line3 = f1.readline()
    line4 = f1.readline()
    line5 = f2.readline()
    line6 = f2.readline()
    line7 = f2.readline()
    line8 = f2.readline()

f1.close()
f2.close()
outf1.close()
outf2.close()
```

References

Bradnam, K. R. et al. (2013). Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. GigaScience, 2(1), 1-31.

Britanova, O. V. et al. (2014). Age-related decrease in TCR repertoire diversity measured with deep and normalized sequence profiling. The Journal of Immunology, 192(6), 2689-2698.

Chao, A. (1987). Estimating the population size for capture-recapture data with unequal catchability. Biometrics, 43(4), 783-791.

Efron, B., and Thisted, R. (1976). Estimating the number of unseen species: How many words did Shakespeare know?. Biometrika, 63(3), 435-447.

Marçais, G., and Kingsford, C. (2011). A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics, 27(6), 764-770.

Meyer, F. et al. (2008). The metagenomics RAST server–a public resource for the automatic phylogenetic and functional analysis of metagenomes. BMC bioinformatics, 9(1), 386.

Hsieh, T. C., K. H. Ma, and A. Chao. (2015). iNEXT: An R package for interpolation and extrapolation of species diversity (Hill numbers). Submitted manuscript.

Walther, B. A., and Moore, J. L. (2005). The concepts of bias, precision and accuracy, and their use in testing the performance of species richness estimators, with a literature review of estimator performance. Ecography, 28(6), 815-829.

Yatsunenko, T. et al. (2012). Human gut microbiome viewed across age and geography. Nature, 486(7402), 222-227.

Zerbino, D. R., and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome research, 18(5), 821-829.