به نام خدا

تحلیل داده‌های حجیم زیستی، تمرین شماره ۲
مهدی کافی ۹۹۲۱۰۷۵۳

بخش صفر)
در اولین هدف از این سوال می‌خواهیم که یک ژنوم تصادفی تولید کنیم که برای این کار به صورت زیر عملی می‌کنیم.

```r
library("BSgenome")
library("Biostrings")
library("GenomicRanges")

#PART0: basics
#GOAL1: write a function to generate a random genome with a given length
ref_length <- 1000000
generate_genome <- function(genome_length)
{
  genome <- DNAString(paste0(sample(c("A", "C", "G", "T"), size = genome_length, replace = TRUE), collapse = ""))
  return(genome)
}
ref <- generate_genome(ref_length)
print(ref)
```

و نتیجه به صورت زیر خواهدبود.

```
> print(ref)
1000000-letter DNAString object
seq: GTCTTTGCAGTGTTCTGCTGTAAGTCTCAGTCACACTTTATTCTTCTGGAAGACCGG...AAGTTGGCTCGAGATGGTACTGCGCCTCACTACATATACCTGTCTTGGTAATGACG
>
```

سپس می‌خواهیم تابعی بنویسیم که با تعداد و طول مشخص از یک ژنوم، خوانش تولید کند که با استفاده از GRanges و getSeq این کار را انجام می‌دهیم.

```r
#GOAL2: write a function to generate random reads with given length and numbers
read_length <- 150
read_num <- 10000
generate_reads <- function(ref, read_length, read_num)
{
  starts <- sample(1:(ref_length - read_length + 1), size = read_num, replace = TRUE)
  REF <- DNAStringSet(ref)
  names(REF) <- "ref"
  gr <- GRanges(seqnames = "ref", ranges = IRanges(start = starts, width = read_length), strand = rep("+", read_num))
  reads <- getSeq(REF, gr)
  names(reads) <- paste0("Read", 1:read_num)
  return(reads)
}
naive_reads <- generate_reads(ref, read_length, read_num)
print(naive_reads)
```

و نتیجه به صورت زیر است.

```
DNAStringSet object of length 10000:
        width seq                                                                                 names
    [1]   150 CTAATACGACTCTACCCTAATCTCATTTCTCGTCTTTAAGTA...GCTAGTCCCGGGTTGCCATACAGTGTGAAAGAGATGCCACG Read1
    [2]   150 CGAAATAATTCTAAGATCAAGATCCATTGTGAGGAGCTACTT...ACTAAACTAGAGTATCTCCGCAAGACTGTAGGCAAGAACTT Read2
    [3]   150 GTTAAGTTTGGATTGTCTCAGCGAGAACCAGCACAATATCTT...TGTTCTATATGAGCTTGCGCTCACCACCAGCCTATAACACT Read3
    [4]   150 CAGATCTCCCGCCCATAGTGGATAGACGTGGGCCCATTGGGG...AGAAAGGTCTATGCTGAGCTCCAGATCGGTGCTGCATTTGC Read4
    [5]   150 GGTCCAGATTGTTGATCTATCCTGGGCCCTTCCCTCCGTTGG...AGTTTAAGTCGTCAAGTCCGACCGCGAGATAAGTTGAAAGT Read5
    ...   ... ...
 [9996]   150 TACGGGAAAGACGGGTTTGTTCAGAACGGTAGGATTCAGCTA...AGCAGTTACCCCCACGTTGTTAGGCCAGTTCGAACTGGGCC Read9996
 [9997]   150 AATCCACAATTGTGCGCTGGAATTGTTGTCCGATGCCCCGCC...GCGCGTCGGCGGTGAAGACAGATTCGACTATTTAGCGTCTC Read9997
 [9998]   150 CCTCTGTTCAACTATTATTTTACGAACAGGGCTGGTCTGAAT...GTTCCATGCATGTGGAAGCTGATACGGTTAACCGGTGACCT Read9998
 [9999]   150 GTAGTTCGTCTACCGACTGGTTTAGGGGCGTACAATGTCTGC...TCGGCAATTCAGACTTCCCTGGTAGGACACCGGGGAAGGCT Read9999
[10000]   150 GTTCCTTGGTTCTTGAGTAGTTTGCAAGCAGTAGGGGTCCAT...AGGCCTGTCGCTTTACTGATACATCCCCCCATGTAGAGCGA Read10000
>
```

در بخش بعدی می‌خواهیم که ژنوم و خوانش‌های تولید شده را در فایل‌های ref.fasta و reads.fastq بنویسیم.

```r
#GOAL3: write simulated genome and reads in a file
REF <- DNAStringSet(ref)
names(REF) <- "ref"
writeXStringSet(x=REF, filepath = "ref.fasta")
writeXStringSet(x=naive_reads, filepath = "reads.fastq", format = "fastq")
```

و فایل‌های زیر تولید می‌شوند.

```
>ref
GAACACGATCCTGGAGGAGTTACACGGAGCGATCCAGTTTCGCATAGATGGTCTTAACATAAGGTATGCTCACATTCCAC
TCTTACTGCGTTCTTGCGGAGCTGAAATCACTATCGTACGTGCAGAGCAACGGAAGGCCACTAGGGACGACCCAAGGTAA
TTAGGGCAGTAGTCTATCGTGACCGCCCTAATGTGCGATATTTGTATACTGCCGTTTAGAGACAGGGCAGCAAATAATGT
GCTAGAACGAGCAAGAGGTCCCACAATTCGCTTTCCGGCCGCGATCTGGAACCGCCCACTTTAGAGTTCTTCTATGTGGT
CTAAACCTGAGAAGTTGCACATTTCGAGATATCTCCGACCCATCGCACCAATCCAAGAAGAGGTTCTGCCACCCGAGAAA
TGTTCAAGAGTTCGCGTGGGGTCTCTCCGTTTGACCGGTGATTCCCCGTGCGCCAGTCGGGACCCTTCACGTGAGGTGTT
TACCAACAGGTAAAAATGACTGCATTGCAACCCGTAGTGGTGGAAGATCTGCTTCGAGCGACCGTATTACACTCCGCGCA
GCCTACATCAGATCCACAAACGGACCTTGCGGAGACCACCTGTGTGGTTCTCGAATTAATAACTTCATTTATATCTGACG
TTCCCGGTCCACTTATGAGGCGCTCTTTGCGCAATTCCCGATATATAGGATGGACCGATCATATGGCGTGCAGATTGCTA
TCATCCAATGCATATTCCGGCAGCATTTTCTGGAGCAAACACGGAACAGCGTAGGTCATGGTATCTTTAAGCTTAATATA
GAAGTTATCGGCTTTTGACGTGAAGTCCGCCCGTTTGGGCATCCCCTATATAGCAATATCTGGTGTAGCAGCTTAACACC
ACCGTCCCACCGCTCGATTCTGCCTGTATCGGCCCGCTCCTGCCTCGTTAGACGGCCTTTTGTGTTGTACGGGGTGAACT
GATCGATATAAGTTCCCCATTAGGCATACCTGGCGCCCCGAACCGCCAACACCTTTCCAGAGGAAAAGTCATCAGTCCGG
TCACGATAGTGATGTGTTCCAAACGCCTATTGTATGGGCAGTCGATTGAAAGCTTGCTCATTCGTAGACACACGACCCGT
CGTCTTGATTTGCCCCCTCTTTTCAAGAAAATTTGGAAGACTGCCAGCAGTAGCTGTATGTGCTCACTTTGACTGCTCAG
TCTGGGGGAGGCATAGGAGTCTTACGCACGGCGATAGACGGTGGGGATTAGGTACGCGACGGAGCGGGGGATGCAGCTTG
GGCCCTAATCTCACTAGTGGGCCTCAGCATGTTTACGTGGCGAGTTCAGTCCATGCACGCAGACAGCAATGTCGCCTGTG
GGCACTTGCGTACTTATGACTTGAGCTAAAATTGGCTTCCGACCGTGCCCTTGGGTGCCAAGTCCATGGTGGTGCTCCCC
GTGCAGAGACAGTGTAACTCTACCTACAGAGCGACCTCCCTACGTAGGCTCTTTAATCTTCGGCGCTTAAGTTTACTCGT
ATAGACCTGGTAAACAACGTCGTTGAGCCCATACCCATAGGAGATATATCTGGGCTCGAAAAGTGTGTAAGGGCTAAACA
TCAAATGGACGGCCCCAGTATCCCGCCTGCGAAACGCGCCCGTAAAGTGTCCGGAAATCCACTCTACTTGTAGCGTCAGC
GAAGAATAAAGATACGTCCAGTATGATACCGGTCGTCTACGAAATCGTATTTTTTCGATCCCGCTGAATTTGCAAGCTCA
GGCCGCCATGTGGAGATTAACGCGGGTCGCACGAGTGTGTGTCATTGTCGCGGGCTCCATACAGATAAATTCATCACAAC
AGGTAAATGAACTGGTACCCACTCGAACCGAGCATCATGGGCAGCGGTCCTCCGACGATCCACCTTCCATCGGCTCAAGT
GGTTCGTACGTAAGTACGGCTACATAAACAGGACTGCCAAGCAACATTATGATTTGTCCCAGGCTGTTGGAGACGTTTAG
TCCTTACAGCTAGATACTTTTTTGTTACGCATTAAAAAGAAGGCACTACAATGATTCGGCGCTGACCGACTCTGCCCCCT
GTATGTACCCAGTAAGCGCCGGGATAGGTTCACGGAAGTTCCGGCAGAAGTCGATTTCGCGAGGGAGTGACTGGCTGCCG
```

```
@Read1
TCGCATTATCATGCCGTGCCAGTCATTCTAAGAGCGGCTCGAGGGTTGCTCGCTTCAAGCATCTTTATTGCCAGCCAAATATGCACACCTGAAAGTACGCATCGACGATATGTCGGCTTTGCACGGAATCACAATTTGTCACCATGCGCT
+Read1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read2
TACTCAATTATTATCGGTTTTGGGAGAACCAATTAAGCGCCATTTCAAAGATGACCAACAATGAGTTGTAATAGTTGCCTAGTTTGAGCGGGATGCACAGTATAAGACACAGCAGAATAACTAGTGAACCATGCCGAGCCAGCGCGTGCG
+Read2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read3
ACGGTTTCAGAAAGAGCGGTTGCACGTGGTTTAAAGCTCGGAATGTGATTTGACTGAGCATCATAGCGTAACCAGTTTTCAAACCTCGTTTCGTCTGCAGCTTCCGATGAAAGCTAGAGAGATAACCCTACTCTATGTAAGACCGCTGAA
+Read3
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read4
CATGGGTCACCTAGGCTGTTGACATTGGGATAGTACCATGAGTCCCGAAGGTAAGTTGGAGAAGCTTTGATCGGGCCAAAACTGAGGTGTTCGCTGGATGGGAAGCATCCTGTCTATCATTTGATTTAGCCTCACTGCGGTGCCTGAATA
+Read4
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read5
TGTCGCAGCTTGAGGTCCAGATAAACCGGCAAATCGCTCTGTAAAGCCAAGTCTGGGGCTGGCATGATATTTCTATGTATACTGTTGATTCATATGTTGGATCTCGGCGCCTTTCGTACAGTCTGACTGGCGATCCAAACTTACACTATC
+Read5
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read6
AGGTAAAAGCGTCTTGCGATAGCGTCATGCAACTCTACTTTAGCGTCTCGTATTATAGCTGGCGGTAGCGAGCCCATGAGGGGAATTTGGTGTGTGACGTAGAGGCGACTGGATGGTGCTCTCGGTTCGCCATGTTCCAGCCCTGCAGGG
+Read6
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read7
CACTTAGGACTCGATCGTCGCGCTTGATATGTACACGAAACAATCGAAAAAGCAGCCCGATACCAAGCGCGCAGGCCACCCTCCTCTTCAAAAAAGCGAAGTTTCACGATATCCGCGATGTGCGAAGTACGGTAAGATAATGTTTGTTCA
+Read7
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read8
GATTCCAAGGGGTGAAAGAAGAAAGTTAAATTTCACATAGTATGACCGCCATGCACAGCTGCGCGGATGTTTTCCTTCCAAGGCGCTCGGGCAGTGAACTTATCGCTATCTGGTCTTAGGTGCTGTTAGAAACAACTGTTCTTGCCTACC
+Read8
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read9
ACTAAGTCGGCGGAAAGACCGTGGGGCGCATGGCAGTCCGGAGTCACTAGAGAAGGTCGCCTAAGTCTATCCGATCCACAAGGGCCGAACGATACTCATTACGGCTGACGGACTAATGCTCAGGTGGTGCACCACGCTCCTAATCGTAAC
+Read9
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
@Read10
CAAACGCCGAGATGAACCCTTGTAGTATTAACATTCAATCAGCAGTCCCAACCATCACGCGGTACCGAGAACAGCGCTCCTTCCTTTGAGGATTTGGTTCAAGTGCAATACACGTCTTCGCGGGCGCCCTTTTACGCAACCAGACTTGAT
+Read10
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

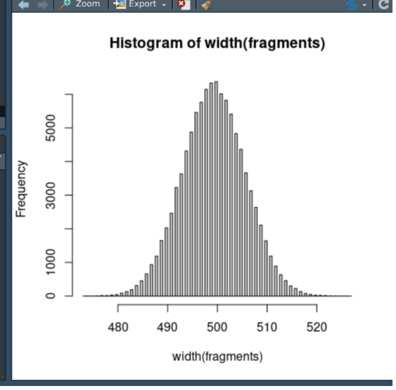
بخش ۱)

در این بخش می‌خوایم که عملیات شکستن DNA و تولید فرگمنت‌ها توسط Sonicator را شبیه‌سازی کنیم و برای این کار کافیست مانند زیر طول فرگمنت‌ها را از توزیع نرمال با میانگین و انحراف معیار داده‌شده محاسبه کنیم و باز هم با استفاده از GRanges و getSeq فرگمنت‌ها را تولید کنیم. در نهایت هیستوگرام طول فرگمنت‌ها را رسم می‌کنیم.

```r
simulate_sonicator <- function(ref, fragment_num, fragment_width_mean, fragment_width_var)
{
  fragment_width <- floor(rnorm(fragment_num, fragment_width_mean, sqrt(fragment_width_var)))
  fragment_starts <- sample(1:(ref_length - max(fragment_width) + 1), size = fragment_num, replace
  REF <- DNAStringSet(ref)
  names(REF) <- "ref"
  gr <- GRanges(seqnames = "ref", ranges = IRanges(start = fragment_starts, width = fragment_widt
  fragments <- getSeq(REF, gr)
  return(fragments)
}
fragment_num <- read_num*10
fragment_width_mean <- 500
fragment_width_var <- 40
fragments <- simulate_sonicator(ref, fragment_num, fragment_width_mean, fragment_width_var)
hist(width(fragments), breaks = 100)
print(fragments)
```

```
read_length  150
read_num     10000
ref_length   1e+06
Functions
generate_g…  function (genome_length)
generate_r…  function (ref, read_length, …
PCR          function (fragments, lambda_…
simulate_s…  function (ref, fragment_num,…
```

```
~/Work/code/MBDA/HW2/
      [3]   496 GTCGATCCCAGGTCCTGGCAAACCATCTCGTCTTCGTCATCG...CAAGTACCTTATCACTGGGACGCGAGTATGAGAGCATTGTC
      [4]   504 GCAGCGTCTATGAAACCCTCATAACTTGAGGCAGGCATTGCA...AGGTTTGAAGCGTTTGTTGAAAAATGCGTTGGTTTCCCCAT
      [5]   507 GATTTACATTTACGTTGTTATTGCTAACCCGTGTTAACTCGC...CGCCTTATACCATCAGGGACAATTGTTGGCTTATCGCGATT
      ...   ... ...
  [99996]   505 CCTTATAAGTCCTCGGCTAAAGCGGGGATCCCCGTGTAAGAT...CACCCAATTCATAGTGACACCTCCTGTCCCAACAGAATCGC
  [99997]   506 TCTTGTAAGCGGCGTGCGAAACAACCTCCTAACTCTTAAAGT...CAATTGCCCGTCTATAGCAGTCCCGTAGAAGAGTGCATACG
  [99998]   503 CCACGACGCCCATAAAGTATTCCTTTAATGTTGGACGAGCGG...TTTACACTGTCACAGTATTCACCGTTGCCTAACACTTACTT
  [99999]   496 TAACTAATAGGAAGTATCCAGCCTTCATTCAAGCCGAAGGGA...GTTTGCTAACATAAATCAAGGCAGATATATTACGGCCACGA
 [100000]   507 AACTCCCTTAAACTAATCTACCTCTATTGTGCATGTACACGA...TATCTGTGGACTCTGGAGCCAATCGAGGCTGTTGTCAAGCC
```



بخش ۲)

در این بخش در ابتدا می‌خواهیم که عملیات PCR را شبیه‌سازی کنیم که این تابع فقط به ازای هر یک از فرگمنت‌ها یک عدد از توزیع پوآسون برمی‌گرداند که مشخص کننده تعداد کپی‌های آن فرگمنت پس از PCR است.

```r
PCR <- function(fragments, lambda_min, lambda_max=NULL, mode=1){
  if (mode == 1){
    return(rpois(length(fragments), lambda = lambda_min))
  }
  if (mode == 2){
    return(rpois(length(fragments), lambda = sample(x = lambda_min:lambda_max, size = length(fragr
  }
}

cnts <- PCR(fragments, lambda_min = 10, mode = 1)
```

و نتیجه آن نیز به صورت زیر خواهدبود.

```
> cnts <- PCR(fragments, lambda_min = 10, mode = 1)
> print(cnts)
   [1] 10 10 10 15 12 13  8 14  8  8  8  7 12  7 12  9 12  8 10 12  7  8 10 10 18  6  4 13  5  8  8
  [32]  7 17  8 12  4 10  7  9 11  6  4 11  6 12  8 11 10  9 14 10 15  9  7 11  8  9  9 13  5  6 14
  [63] 17  9 10 15 13 13  4  5  7 13  5  8 13 19  6  9  6  6 13 11 12  9 11 12  3 14  6  7  8 12  8
  [94] 11 11 14 12 13  6  7 16 17  9 14 12 13  7 13 13  7  9  9 10 10  9  7  3 11  9  5 10 11 17  5
 [125] 10  9 11  7  7 12 10 19  8  9 12 17 10 11 10 15 14 13 12 14  9 13  7 10  8 11  6  7  5 11 16
 [156]  7  5 10  8  8 12 14  9  9 20  5 10 11  2  8 10 17  7 14  7 13 15  8  8 10 10  9 12 11  9  5
 [187] 10  9 13 14 10 15 10  6  7  8 12 13  7 10  7 11 17 16  6 11 11  9 10  9  4  9  4  7  6 10 16
 [218] 12  7  4 10 12  9 10  9  7  8  9 10  8  8  7  8  7  9 13  8 10 12 15 13 14  9 17  4 12  8 12
```

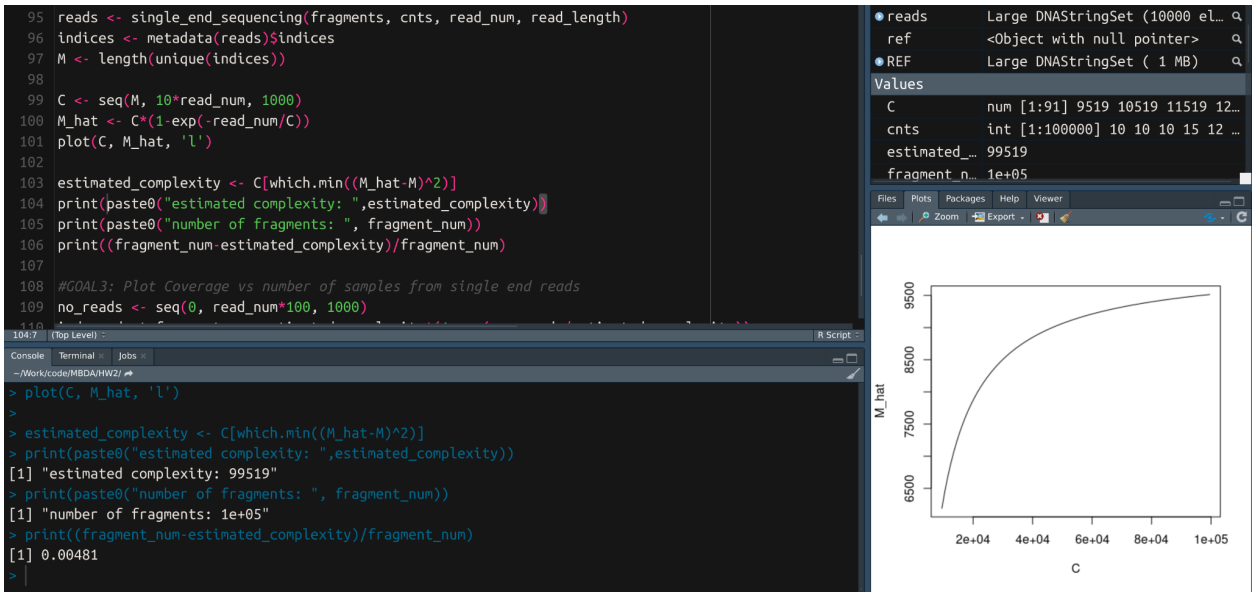
در ادامه تابعی می‌نویسیم که از فرگنت‌های تولید شده و تعداد cnts حاصل از PCR، خوانش‌هایی با طول و تعداد مشخص کند. برای این کار در ابتدا هر کدام از فرگمنت‌ها را به تعداد مرتبط با خودش که خروجی PCR است کپی می‌کنیم و سپس از تمام فرگمنت‌ها به تعداد خوانش‌ها که ورودی تابع است، فرگمنت انتخاب می‌کنیم و به طول گفته شده از ابتدای آن‌ها می‌خوانیم.

```r
single_end_sequencing <- function(fragments, cnts, N, L){
  library <- rep(fragments, cnts)
  lib.indices <- rep(1:length(fragments), cnts)
  read.indices<- sample(1:length(library), size = N)
  selected_fragments <- library[read.indices]
  names(selected_fragments) <- paste0("s_fragments", 1:N)
  gr <- GRanges(seqnames = paste0("s_fragments", 1:N), ranges = IRanges(start = 1, width = L), str
  reads <- getSeq(selected_fragments, gr)
  indices <- lib.indices[read.indices]
  metadata(reads)$indices <- indices
  return(reads)
}
```

نتیجه خوانش‌ها نیز به صورت زیر است. البته علاوه بر رشته خوانش‌ها، شماره فرگمنتی که خوانش از آن خوانده شده نیز در متادیتای خوانش‌ها ذخیره می‌شود تا مراحل بعدی بتوانیم که تعداد فرگمنت‌هایی که از آن‌ها خوانش مجزا داشته‌ایم را محاسبه کنیم.

```
> reads
DNAStringSet object of length 10000:
        width seq
    [1]   150 TCGCTGGCATTGCGGGTTATCGCCGCGCGATCAGGAGGAGAA...GTCGCTATGTAGAACCGCCGCTCCTCAACCTGTTAAAGGAGA
    [2]   150 GGGTCTCGGGCTACCGAGACCGCTGAACCACGATGCGGCTAT...ATCCGACTAACTCCATCCGAGATCAGAAGCTATTAACCCTGA
    [3]   150 AGGGACATACCCAAGGGACACGTGCACCTTCGCATACACAAG...CTAGTCCCACCGCCAATAAGAAGGCTCACTGCGAGACATTTC
    [4]   150 CGACGCTCACAACTATTACGATGACACTAGGCCAGCGCTACG...CCCTTCTCTATACAACTAGTCATTGGTGACGAACGCCGTTCT
    [5]   150 TATGCTCCAGAGTACAATTTCTGCCGAGTCCCGCATACCACA...CTACTTACCATGCAAACGTAGACCAAACTGAGCGTCATAGTC
    ...   ... ...
 [9996]   150 ACGAGCGCTCAGCCACACGATGGGGCGAAACCAGGGCCATTC...ATAGTCCTATCAGCGTGCGGCAGCACCGTCAGTACCCCGGCT
```

سپس با محاسبه کردن تعداد شماره فرگمنت‌های یکتا در خوانش‌ها می‌توانیم تعداد خوانش‌های مجزای واقعی را به دست آوریم. و سپس از به ازای Cهای متفاوت مقدار خوانش‌های مجزا را تخمین بزنیم و در نهایت محاسبه می‌کنیم که کدام C باعث شده‌است که تخمین بهتری بزنیم و آنرا به عنوان تخمین پیچیدگی چاپ می‌کنیم.

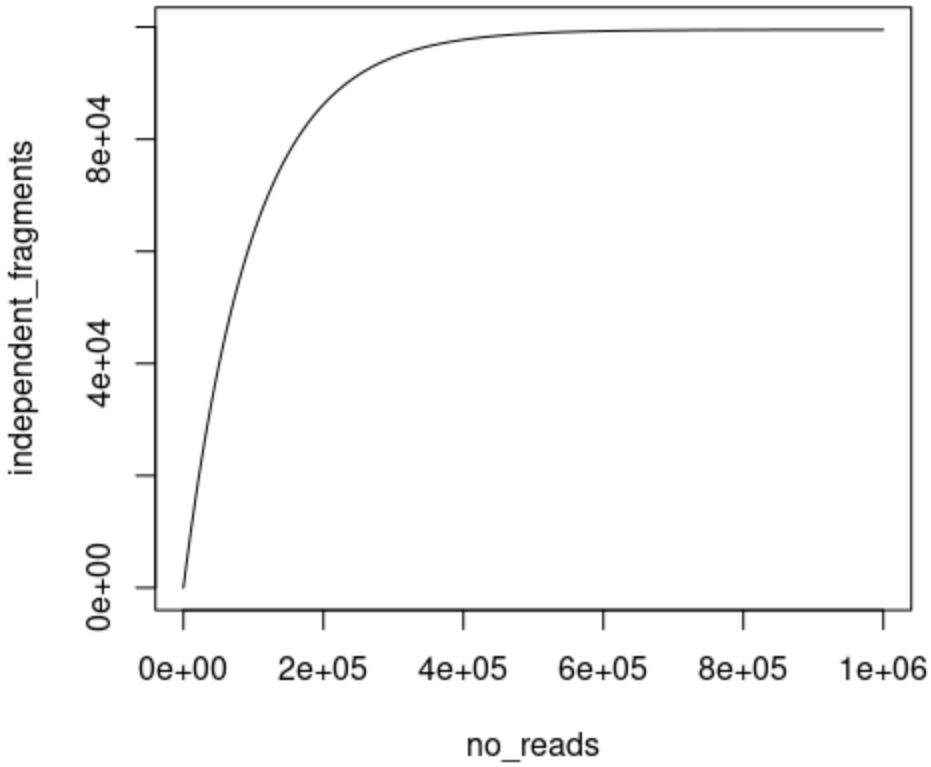


در بخش بعدی می‌خواهیم که با استفاده از خوانش به صورت paired-end پیچیدگی را تخمین بزنیم ولی به نظر می‌رسد که اگر به طور مثال با ۱۵۰ خوانش single-end پیچیدگی را تخمین بزنیم؛ با ۱۵۰ خوانش به صورت paired-end نیز همان تخمین را بزنیم. زیرا که تعداد فرگمنت‌های مجزا که شماره فرگمنت‌ها به دست می‌آید ثابت خواهدماند و فقط باعث می‌شود که از آن فرگمنت‌ها دو خوانش از دو طرف داشته‌باشیم که در مواقعی مانند alignment به کار خواهندآمند.

در بخش بعدی می‌خواهیم نمودار تعداد فرگمنت‌های مجزا را به نسبت تعداد خوانش رسم کنیم و برای این کار با پیچیدگی‌ای که تخمین زده‌ایم، مقدار فرگمنت‌ها را به ازای تعداد متفاوت خوانش تخمین زده و نمودار را رسم می‌کنیم.

```
#GOAL3: Plot Coverage vs number of samples from single end reads
no_reads <- seq(0, read_num*100, 1000)
independent_fragments <- estimated_complexity*(1-exp(-no_reads/estimated_complexity))
plot(no_reads, independent_fragments, 'l')
```

و نمودار به صورت زیر خواهدبود.



و در بخش آخر می‌خواهیم تخمین بزنیم که آیا خوانش کردن از این کتابخانه اطلاعات بیشتری به ما می‌دهد یا خیر. برای این کار از تعداد فرگمنت‌های مجزایی که تخمین زده‌ایم به ازای تعداد خوانش‌های متفاوت، بیشینه میگیریم و میبینیم که به تعداد واقعی نمیرسد ولی فاصله کمی با آن دارد.

```
113 #GOAL4: Predict if adding more reads with increase the coverage
114 max_coverage <- max(independent_fragments)
115 print(fragment_num - max_coverage)
116
```

113:44 (Top Level)

Console | Terminal | Jobs

~/Work/code/MBDA/HW2/

```
> #GOAL3: Plot Coverage vs number of samples from single end reads
> no_reads <- seq(0, read_num*100, 1000)
> independent_fragments <- estimated_complexity*(1-exp(-no_reads/estimated_complexi
> plot(no_reads, independent_fragments, 'l')
>
> #GOAL4: Predict if adding more reads with increase the coverage
> max_coverage <- max(independent_fragments)
> print(fragment_num - max_coverage)
[1] 485.305
>
```