calculating nucleotide frequency per column

Asked 4 years ago Active 1 year, 4 months ago Viewed 2k times

5

1

I have some sequences shown below

```
CAGGTAGCC
CCGGTCAGA
AGGGTTTGA
TTGGTGAGG
CAAGTATGA
ACTGTATGC
CTGGTAACC
TATGTACTG
GCTGTGAGA
CAGGTGGGC
TCAGTGAGA
GGGGTGAGT
TGGGTATGT
GAGGTGAGA
CAGGTGGAG
```

Each line has 9 nucleotides. Consider it to be 9 columns.I want to calculate the nucleotide frequency of each nucleotide for each of the 9 columns. For example 1st column will have these bases C,C,A,T,C,A etc Out put should be something like this

```
A   0.25    0.34    0.56    0.43    0.00    0.90    0.45    0.34    0.31
C   0.45    0.40    0.90    0.00    0.40    0.90    0.30    0.25    0.2
G   0.00    0.00    0.34    1.00    0.30    0.30    0.35    0.90    0.1
T   0.24    0.56    0.00    0.00    1.00    0.34    0.45    0.35    0.36
```

Note, I just made up the numbers to show you the output file format

Share Edit Follow Flag

edited Jul 22 '20 at 15:55

M__
5,713 2 17 38

asked Dec 14 '17 at 16:40

user3138373
400 1 3 12

I think a package in Bioconductor can do this, I don't recall exactly one. Has you searched there? Or what have you tried to calculate this? The original file is a plain text or a fasta file (I don't recall if it is only for fasta or for other file types)? – llrs Dec 14 '17 at 16:57

Original file is a text file only. I am thinking of using awk(associative arrays) to calculate it, still writing the script though – user3138373 Dec 14 '17 at 17:02

7 Answers

Active | Oldest | Votes

3

For anyone interested in doing this from any sort of alignment file, I've implemented a position frequency matrix function in AlignBuddy.

input:

```
 15 9
seq_1      CAGGTAGCC
seq_2      CCGGTCAGA
seq_3      AGGGTTTGA
seq_4      TTGGTGAGG
seq_5      CAAGTATGA
seq_6      ACTGTATGC
seq_7      CTGGTAACC
seq_8      TATGTACTG
seq_9      GCTGTGAGA
seq_10     CAGGTGGGC
seq_11     TCAGTGAGA
seq_12     GGGGTGAGT
seq_13     TGGGTATGT
seq_14     GAGGTGAGA
seq_15     CAGGTGGAG
```

```
:$ alignbuddy input.phy --pos_freq_mat
```

Output:

```
### Alignment 1 ###
A       0.133   0.400   0.133   0.000   0.000   0.400   0.467   0.067   0.400
C       0.400   0.267   0.000   0.000   0.000   0.067   0.067   0.133   0.267
G       0.200   0.200   0.667   1.000   0.000   0.467   0.200   0.733   0.200
T       0.267   0.133   0.200   0.000   1.000   0.067   0.267   0.067   0.133
```

Share Edit Follow Flag

edited Dec 21 '17 at 15:15

answered Dec 16 '17 at 17:08

Steve Bond
176 1 6

5

awk

```
awk '{L=length($1);for(i=1;i<=L;i++) {B=substr($1,i,1);T[i][B]++;}} END{for(BI=0;BI<4;BI++) {B=(BI==0?"A":(BI==1?"C":
(BI==2?"G":"T")));printf("%s",B); for(i in T) {tot=0.0;for(B2 in T[i]){tot+=T[i][B2];}printf("\t%0.2f",(T[i][B]/tot));}
printf("\n");}}' input.txt
```

```
A   0.13    0.40    0.13    0.00    0.00    0.40    0.47    0.07    0.40
C   0.40    0.27    0.00    0.00    0.00    0.07    0.07    0.13    0.27
G   0.20    0.20    0.67    1.00    0.00    0.47    0.20    0.73    0.20
T   0.27    0.13    0.20    0.00    1.00    0.07    0.27    0.07    0.13
```

Or, expanded for clarity:

```
awk '{
        L=length($1);
        for(i=1;i<=L;i++) {
            B=substr($1,i,1);
```

```
                B=(BI==0?"A":(BI==1?"C":(BI==2?"G":"T")));
                printf("%s",B);
                for(i in T) {
                    tot=0.0;
                    for(B2 in T[i]){
                        tot+=T[i][B2];
                    }
                printf("\t%0.2f",(T[i][B]/tot));
                }
            printf("\n");
            }
}' input.txt
```

Share Edit Follow Flag

edited Dec 16 '17 at 16:34

terdon
7,802 3 14 40

answered Dec 15 '17 at 11:21

Pierre
1,448 5 11

Can we do same for taking dinucleotide at time (eg; AA, TT, CG etc); the output matrix would reduce to half? – kashiff007 Aug 4 at 7:32

4

Here is one example of how to do this with a bit of python. Alternatively one could create strings of each column and using `letterFrequency()` from the `Biostrings` package.

```
#Make a list of hashes
hl = []
for i in range(9):
    hl.append({'A': 0, 'C': 0, 'G': 0, 'T': 0})

f = open("foo.txt")  # CHANGE ME
nLines = 0
for line in f:
    for idx, c in enumerate(line.strip()):
        hl[idx][c] += 1
```

```
    print("{}\t{}".format(char, "\t".join(["{:0.2f}".format(x[char]/nLines) for
x in hl])))
```

The output of your example is then:

```
A   0.13    0.40    0.13    0.00    0.00    0.40    0.47    0.07    0.40
C   0.40    0.27    0.00    0.00    0.00    0.07    0.07    0.13    0.27
G   0.20    0.20    0.67    1.00    0.00    0.47    0.20    0.73    0.20
T   0.27    0.13    0.20    0.00    1.00    0.07    0.27    0.07    0.13
```

Share Edit Follow Flag

edited Dec 15 '17 at 10:22

terdon
7,802 3 14 40

answered Dec 14 '17 at 21:01

Devon Ryan ♦
18.9k 2 23 50

how to change enumerate if I want to take dinucleotides at time (eg; AA, TT, CG etc); the output matrix would reduce to half? – kashiff007 Aug 4 at 7:37

Here's a Perl approach:

3

```
#!/usr/bin/env perl
use strict;
my %counts;

## Read the input file line by line
while (my $line = <>) {
  print;
  ## remove trailing '\n' characters
  chomp $line;
  ## split the line into an array at every character
  my @columns=split(/./, $line);
  ## iterate over the array from the first to the last position
  for my $i (0..$#columns){
```

```
    $counts{$nt}[$i]++;
  }
}
## Iterate over the counts hash
for my $nt (sort keys(%counts)){
  print "$nt\t";
  ## dereference the array stored in the hash
  my @countsForThisNt = @{$counts{$nt}};
  ## iterate over the counts for each position for this nt
  for (my $l=0;$l<=$#countsForThisNt; $l++) {#
    ## If the value for this position isn't defined,
    ## set it to 0.
    $countsForThisNt[$l]||=0;
    ## Print all the things
    printf "%.2f\t", $countsForThisNt[$l]/$., $l;
  }
  print "\n";
}
```

Save the script somewhere in your PATH, make it executable and run:

```
$ foo.pl file
A   0.13    0.40    0.13    0.00    0.00    0.40    0.47    0.07    0.40
C   0.40    0.27    0.00    0.00    0.00    0.07    0.07    0.13    0.27
G   0.20    0.20    0.67    1.00    0.00    0.47    0.20    0.73    0.20
T   0.27    0.13    0.20    0.00    1.00    0.07    0.27    0.07    0.13
```

Alternatively, if you're into the whole brevity thing, and enjoy some golfing, here's the same thing as a one-liner:

```
 perl -ne 'chomp;@F=split(//);for$i(0..$#F){$k{$F[$i]}[$i]++}}{for $nt(sort keys(%k)){print"$nt\t";for$i(0..$#{$k{$nt}})
{$g=$k{$nt}[$i]||0; printf"%.2f\t",$g/$.}print"\n";}' file
```

Share Edit Follow Flag

edited Dec 15 '17 at 10:12

answered Dec 15 '17 at 10:01

terdon

2

Ben's answer works but is very un-R-like: you wouldn't normally iteratively assign to items in a matrix. R has powerful vectorised operations which make this manual work unnecessary. Here's a more idiomatic solution:

```
sequences = readLines('sequences.txt')
bases_matrix = do.call(rbind, strsplit(sequences, ''))

apply(bases_matrix, 2L, function (col) {
    str = DNAString(paste(col, collapse = ''))
    letterFrequency(str, letters = 'ACGT', OR = 0L, as.prob = TRUE)
})
```

This uses the same Bioconductor packages. Since this is such a simple problem, it can also be written without Bioconductor:

```
bases = strsplit(sequences, '')
# Use a data.frame here so we can use factors in the next step:
# R does not support matrices of factors. Ugh.
bases_by_column = setNames(do.call(rbind.data.frame, bases),
seq_along(bases[[1L]]))
# Ensure that every column will be a complete set of ACGT frequencies
bases_by_column = lapply(bases_by_column, factor, c('A', 'C', 'G', 'T'))
sapply(lapply(bases_by_columns, table), prop.table)
```

Using modern R idioms from the 'magrittr' package, I'd write this as a pipeline; this very directly shows the sequence of transformations.

```
do.call(rbind.data.frame, bases) %>%
    setNames(seq_along(bases[[1L]])) %>%
    lapply(factor, c('A', 'C', 'G', 'T')) %>%
    lapply(table) %>%
    sapply(prop.table)
```

Share Edit Follow Flag

edited Jul 22 '20 at 13:58

answered Dec 21 '17 at 15:08

Konrad Rudolph
4,665 10 38

bioinformatics with this “it works” attitude, which leads to a proliferation of a lot of truly atrocious code. “But it works”. Yeah, just about … as long as you don’t nudge it. – Konrad Rudolph Dec 21 '17 at 16:04

I don't completely agree, no offense. With many bioinformaticians the BIO part is much more important than the best script practices. At the company where I work, they don't care about these best script practises, but do want someone that understands e.g., what a tissue resident memory cell is. That's just my point of view. – benn Dec 21 '17 at 16:09

1 @b.nota It’s akin to lab (or generally science) best practices, to be honest: Many people use the same argument to justify sloppy lab work or dodgy use of statistics (p-hacking etc). At the moment bad software engineering is still somewhat accepted but I’m confident that this acceptance will lessen as the field matures. Just to clarify: this doesn’t so much concern your code, which works fine. But there’s a lot of genuinely bad, unmaintainable code in circulation in bioinfomatics *tools* that costs money and time to fix because of bad practices. – Konrad Rudolph Dec 21 '17 at 16:13

1 Yeah I agree with that part. I am PhD in biology, and learned to code with a java course here and a course in R there. I work with R almost daily, but unfortunately I still try to avoid `apply` functions (I do try them more often, but still the old school java double for loop is more intuitive for me). – benn Dec 21 '17 at 16:19

2

Here an example in R using `Biostrings` and `letterFrequency` (as suggested by Devon Ryan).

```
library(Biostrings)

data <- read.table("DNA.txt", stringsAsFactors = F)

new <- matrix(nrow = 9, ncol = 15)

for(i in 1:9){
  for(j in 1:15){
    new[i,j] <- substring(data[j,], i, i)
  }
}

countTable <- matrix(nrow = 9, ncol = 4)
for(i in 1:9){
  columnSeq <- DNAStringSet(paste0(new[i,], collapse = ""))
  columnCounts <- letterFrequency(columnSeq, letters = "ACGT", OR = 0)
  countTable[i,] <- columnCounts
```

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
A 0.13 0.40 0.13    0    0 0.40 0.47 0.07 0.40
C 0.40 0.27 0.00    0    0 0.07 0.07 0.13 0.27
G 0.20 0.20 0.67    1    0 0.47 0.20 0.73 0.20
T 0.27 0.13 0.20    0    1 0.07 0.27 0.07 0.13
```

Share Edit Follow Flag

edited Dec 15 '17 at 19:59

answered Dec 15 '17 at 9:07

benn
3,511 6 24

The whole `new` matrix generation can be written in a single line as `do.call(rbind, strsplit(data, ''))`. The subsequent `for` loop can likewise be replaced. – Konrad Rudolph Dec 21 '17 at 15:10

Yes, there are many roads that lead to Rome. – benn Dec 21 '17 at 15:55

1

Here is how you get the distributions by column using the `TraMineR` R package.

```
library(TraMineR)

sts.data <- c(
  "CAGGTAGCC",
  "CCGGTCAGA",
  "AGGGTTTGA",
  "TTGGTGAGG",
  "CAAGTATGA",
  "ACTGTATGC",
  "CTGGTAACC",
  "TATGTACTG",
  "GCTGTGAGA",
  "CAGGTGGGC",
  "TCAGTGAGA",
  "GGGGTGAGT",
  "TGGGTATGT"
```

```
seqstatd(seq)
```

and here is the outcome of the `seqstatd` function

```
  [State frequencies]
   [1]  [2]  [3] [4] [5]   [6]   [7]   [8]  [9]
A 0.13 0.40 0.13   0   0 0.400 0.467 0.067 0.40
C 0.40 0.27 0.00   0   0 0.067 0.067 0.133 0.27
G 0.20 0.20 0.67   1   0 0.467 0.200 0.733 0.20
T 0.27 0.13 0.20   0   1 0.067 0.267 0.067 0.13

   [Valid states]
  [1] [2] [3] [4] [5] [6] [7] [8] [9]
N  15  15  15  15  15  15  15  15  15

   [Entropy index]
   [1]  [2]  [3] [4] [5]  [6]  [7]  [8]  [9]
H 0.94 0.94 0.62   0   0 0.78 0.87 0.62 0.94
```

Share Edit Follow Flag

answered Dec 21 '17 at 10:51

Gilbert
111 2