FAQ: Frequently Asked Questions

Lobry, J.R.

July 12, 2016

1 How can I extract the mitochondrial D-loop/Control Region?

This question from Sam Borstein was raised by e-mail on 2016-07-10 with an example based on the following genBank entry:

```
LOCUS       JF979198               16569 bp    DNA     circular PRI 10-MAY-2012
DEFINITION  Homo sapiens isolate 02_C4c1a(S635848) mitochondrion, complete
            genome.
ACCESSION   JF979198
[...]
FEATURES             Location/Qualifiers
     source          1..16569
                     /organism="Homo sapiens"
                     /organelle="mitochondrion"
                     /mol_type="genomic DNA"
                     /isolate="02_C4c1a(S635848)"
                     /db_xref="taxon:9606"
                     /haplotype="C4c1a"
                     /country="USA: Oklahoma, Flint District"
                     /note="Cherokee"
     D-loop          complement(join(16104..16569,1..191))
     tRNA            577..647
                     /product="tRNA-Phe"
     rRNA            648..1601
                     /product="12S ribosomal RNA"
[...]
```

The D-loop control region is listed under the `FEATURES` but is not turned out automatically into subsequences:

```
choosebank("genbank")
getType()
```

```
       sname                                                  libel
5962     CDS                              .PE protein coding region
5963   LOCUS                                sequenced DNA fragment
5964 MISC_RNA                .RN other structural RNA coding region
5965   NCRNA .NC non-protein-coding gene other than rRNA and tRNA
5966    RRNA                                .RR mature ribosomal RNA
5967   TMRNA                              .TM transfer messenger RNA
5968    TRNA                                 .TR mature transfer RNA
```

The more general question is then: can I extract the sequence of something which is documented in the `FEATURES` but not directly available as a subsequence? The answer is yes: the function `extractseqs()` can do the job, dealing

with all the tedious `join` or `complement` operations. As from **seqinR** 3.2-1 the function has been changed not to use zlib compressed sockets by default so that it works on all platforms:

```
mito <- query("mito","AC=JF979198")
myseq <- extractseqs("mito", operation = "feature", feature = "D-loop")
myseq
```

```
 [1] ">JF979198.F1             657 residues"
 [2] "TGTTCGCCTGTAATATTGAACGTAGGTGCGATAAATAATAGGATGAGGCAGGAATCAAAG"
 [3] "ACAGATACTGCGACATAGGGTGCTCCGGCTCCAGCGTCTCGCAATGCTATCGCGTGCACA"
 [4] "CCCCCCAGACGAAAATACCAAATGCATGGAGAGCTCCCGTGAGTGGTTAATAGGGTGATA"
 [5] "GACCTGTGATCCATCGTGATGTCTTATTTAAGGGGAACGTGTGGGCTATTTAGGCTTTAT"
 [6] "GGCCCTGAAGTAGGAACCAGATGTCGGATACAGTTCACTTTAGCTACCCCCAAGTGTTAT"
 [7] "GGGCCCGGAGCGAGGAGAGTAGCACTCTTGTGCGGGATATTGATTTCACGGAGGATGGTG"
 [8] "GTCAAGGGACCCCTATCTGAGGGGGGTCATCCATGGGGACGAGAAGGGATTTGACTGTAA"
 [9] "TGTGCTATGTACGATAAATGGCTTTATGTACTATGTACTGTTGAGGGTGGGTAGGTTTGT"
[10] "TGGTATCCTAGTGGGTGAGGGGTGGCTTTGGAGTTGCAGCTGATGTGTGATAGTTGAAGG"
[11] "TTGATTGCTGTACTTGCTTGTAAGCATGGGGAGGGGGTTTTGATGTGGATTGGGTTTTTA"
[12] "TGTACTACAGGTGGTCAAGTATTTATGGTACCGTACAATATTCATGGTGGCTGGCAG"
```

You can then read this just as a FASTA file with:

```
myfasta <- read.fasta(textConnection(myseq))
summary(myfasta)
```

```
            Length Class       Mode
JF979198.F1 657    SeqFastadna character
```

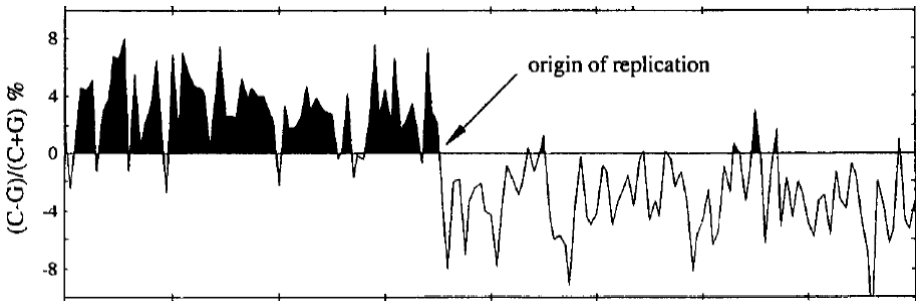


Figure 1: Screenshot of a part of figure 1 from [6]. The GC-skew is computed in non-overlapping windows of 10 Kb along a 1.6 Mb fragment of the *Escherichia coli* chromosome. The sequence is available with `data(m16j)`.

2 How can I compute a score over a moving window?

As an illustration, suppose that we want to reproduce a part of figure 1 from [6] whose screenshot is given is given in figure 1.

The score here is the GC-skew computed in non-overlapping windows of 10 Kb for a 1.6 Mb sequence. We need a fragment of *Escherchia coli* K12 chromosome from 67.4 min to 4.1 min on the genetic map. The sequence is directly available with `data(m16j)`. Let's put this fragment into the string `myseq`:

```
data(m16j)
myseq <- m16j
```

This is not exactly the same sequence that was used in [6] but very close to[1]. We define a function called `gcskew()` that computes our score for a given string `x`:

```
gcskew <- function(x){
  if( !is.character(x) || length(x) > 1 ) stop("single string expected")
  tmp <- tolower(s2c(x))
  nC <- sum(tmp == "c")
  nG <- sum(tmp == "g")
  if( nC + nG == 0 ) return(NA)
  return(100*(nC - nG)/(nC + nG))
}
gcskew("GCCC")
```

```
[1] 50
```

```
gcskew("GCCCNNNNNN")
```

```
[1] 50
```

[1]The sequence used in [6] was a 1,616,174 bp fragment obtained from the concatenation of nine overlapping sequences (U18997, U00039, L10328, M87049, L19201, U00006, U14003, D10483, D26562 [10, 3, 4, 9, 1, 13]). Ambiguities have been resolved since then and its was a chimeric sequence from K-12 strains MG1655 and W3110 [5], the sequence used here is from strain MG1655 only [2].

Note some defensive programming tricks used here:

- We check that the argument `x` is a single string.
- We expand it as vector of single chars with `s2c()` only within the function to avoid big objects in the workspace.
- We force to lower case letters with `tolower()` so that we can use upper case letters too.
- We avoid division by zero and return `NA` in this case.
- We do not divide by the length of `x` but by the actual number of C and G so that ambiguous bases such as N do not introduce biases.

We move now along the sequence to compute the GC-skew:

```
step <- 10000
wsize <- 10000
starts <- seq(from = 1, to = nchar(myseq), by = step)
starts <- starts[-length(starts)] # remove last one
n <- length(starts)
result <- numeric(n)
for(i in seq_len(n)){
        result[i] <- gcskew(substr(myseq, starts[i], starts[i] + wsize - 1))
}
```

The following code[2] was used to produce figure 2.

```
xx <- starts/1000
yy <- result
n <- length(result)
hline <- 0
plot (yy ~ xx, type="n", axes=FALSE, ann=FALSE, ylim = c(-10, 10))
polygon(c(xx[1], xx, xx[n]), c(min(yy), yy, min(yy)), col = "black", border=NA)
usr <- par("usr")
rect(usr[1], usr[3], usr[2], hline, col="white", border=NA)
lines(xx, yy)
abline (h=hline)
box()
axis(1, at = seq(0,1600, by = 200))
axis(2, las = 1)
title(xlab = "position (Kbp)", ylab = "(C-G)/(C+G) %",
main = expression(paste("GC skew in ", italic(Escherichia~~coli))))
arrows(860, 5.5, 720, 0.5, length = 0.1, lwd = 2)
text(860, 5.5, "origin of replication", pos = 4)
```

You can now play with the `wsize` and `step` parameters to explore the signal (but note that with overlapping windows your points are no more independent) or use all the smoothing tools available under R. Figure 3 shows for instance what can be obtained with the `lowess()` function with two values for the smoothing parameter `f`. The corresponding code is as follows:

[2] This code is adapted from the code at `http://www.stat.auckland.ac.nz/~paul/RGraphics/chapter3.html` for figure 3.25 in Paul Murrell's book [7]. This book is a must read if you are interested by R's *force de frappe* in the graphic domain.

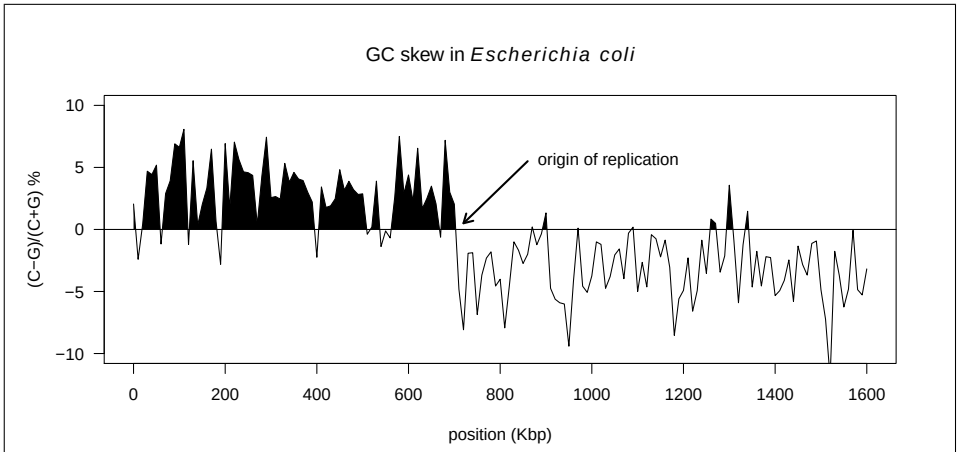


Figure 2: Re-creation of figure 1 from scratch.

```
plot(xx,yy, col = "grey", type = "b", ylim = c(-10,10), las = 1, xaxt = "n",
main = expression(paste("GC skew in ", italic(Escherichia~~coli))),
xlab = "position (Kbp)", ylab = "(C-G)/(C+G) %")
axis(1, at = seq(0,1600, by = 200))
lines(smooth <- lowess(xx,yy, f = 0.05), lwd = 1)
polycurve <- function(x, y, base.y = min(y), ...) polygon(x = c(min(x), x, max(x)), y = c(base.y, y, base.y), ..
up <- smooth$y > 0
polycurve(smooth$x[up], smooth$y[up], base.y = 0, col = rgb(0,0,1,0.5))
lines(lowess(xx,yy, f = 0.2), lwd = 2, col = "red")
legend("topright", inset = 0.01, legend = c("f = 0.05", "f = 0.20"), lwd = c(1,2), col = c("black", "red"))
abline(h=0)
arrows(860, 5.5, 720, 0.5, length = 0.1, lwd = 2)
text(860, 5.5, "origin of replication", pos = 4)
```

3 How can I extract just a fragment from my sequence?

Use the generic function `getFrag()` :

```
choosebank("emblTP")
mylist <- query("mylist", "AC=A00001")
getFrag(mylist$req[[1]], begin = 10, end = 20)
```

```
[1] "gatggagaatt"
attr(,"seqMother")
[1] "A00001"
attr(,"begin")
[1] 10
attr(,"end")
[1] 20
attr(,"class")
[1] "SeqFrag"
```

```
closebank()
```

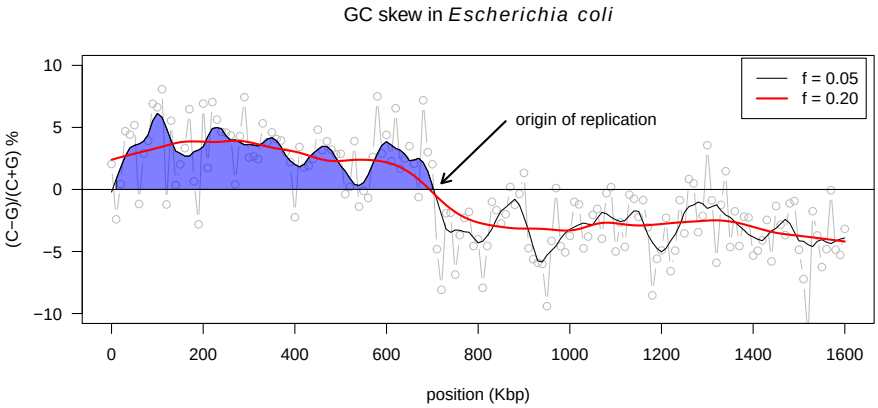


Figure 3: Playing with the smoothing parameter `f` of the `lowess()` function.

4 How do I compute a score on my sequences?

In the example below we want to compute the G+C content in third codon positions for complete ribosomal CDS from *Escherichia coli*:

```
choosebank("emblTP")
ecribo <- query("ecribo","sp=escherichia coli ET t=cds ET k=ribosom@ ET NO k=partial")
myseqs <- sapply(ecribo$req, getSequence)
(gc3 <- sapply(myseqs, GC3))
```

```
 [1] 0.4946237 0.6046512 0.5000000 0.6194030 0.5772727 0.4838710 0.5980066
 [8] 0.4974359 0.5031250 0.4324324 0.5000000 0.5113636 0.5290520 0.6142857
[15] 0.4904762 0.5714286 0.6191860 0.5906040 0.4880000 0.4880000 0.4946237
[22] 0.6046512 0.5000000 0.3522727 0.5076923 0.4343434 0.6194030 0.5522388
[29] 0.6104651 0.5661157 0.4946237 0.4946237 0.6079734 0.5000000 0.6343284
[36] 0.4659091 0.5789474 0.4946237 0.5000000 0.4974359 0.5689655 0.4611111
[43] 0.4611111 0.5303030 0.5303030 0.4482759 0.4201681 0.5915493 0.5000000
[50] 0.3829787 0.4519231 0.4302326 0.5696203 0.4285714 0.5689655 0.5000000
[57] 0.5224417 0.5661157 0.6057692 0.4444444 0.4659091 0.4130435 0.4946237
[64] 0.5661157 0.4946237 0.5680272
```

At the amino-acid level, we may get an estimate of the isoelectric point of the proteins this way:

```
sapply( sapply(myseqs, getTrans), computePI)
```

```
 [1]  6.624309  7.801329 10.864793  5.931989  7.830476  6.624309  7.801329
 [8]  9.203410  9.826485  5.674672  7.154423  6.060457  6.313741  5.571446
[15]  9.435422  4.310745  6.145496  4.876054 11.006424 10.876041  6.624309
[22]  7.801329 10.864793  9.346289  9.203410  5.877050  5.931989  9.934988
[29]  5.920490  6.612505  6.624309  6.624309  7.801329 10.864793  5.931989
[36] 11.182499  9.598944  6.624309 10.864793  9.203410 11.031938  5.858421
[43]  5.858421 11.777516 11.777511 10.619175 11.365738  9.460987 10.864793
[50] 13.002381  9.845859 10.584868 11.421257 10.248325 11.031938 10.402075
[57]  4.863862  6.612505  9.681066 11.150304 11.182505 11.043607  6.624309
[64]  6.612505  6.624309  4.310745
```

Note that some pre-defined vectors to compute linear forms on sequences are available in the `EXP` data.

As a matter of convenience, you may encapsulate the computation of your favorite score within a function this way:

```
GC3m <- function(list, ind = 1:list$nelem) sapply(sapply(list$req[ind], getSequence), GC3)
GC3m(ecribo)
```

```
 [1] 0.4946237 0.6046512 0.5000000 0.6194030 0.5772727 0.4838710 0.5980066
 [8] 0.4974359 0.5031250 0.4324324 0.5000000 0.5113636 0.5290520 0.6142857
[15] 0.4904762 0.5714286 0.6191860 0.5906040 0.4880000 0.4880000 0.4946237
[22] 0.6046512 0.5000000 0.3522727 0.5076923 0.4343434 0.6194030 0.5522388
[29] 0.6104651 0.5661157 0.4946237 0.4946237 0.6079734 0.5000000 0.6343284
[36] 0.4659091 0.5789474 0.4946237 0.5000000 0.4974359 0.5689655 0.4611111
[43] 0.4611111 0.5303030 0.5303030 0.4482759 0.4201681 0.5915493 0.5000000
[50] 0.3829787 0.4519231 0.4302326 0.5696203 0.4285714 0.5689655 0.5000000
[57] 0.5224417 0.5661157 0.6057692 0.4444444 0.4659091 0.4130435 0.4946237
[64] 0.5661157 0.4946237 0.5680272
```

```
GC3m(ecribo, 1:10)
```

```
 [1] 0.4946237 0.6046512 0.5000000 0.6194030 0.5772727 0.4838710 0.5980066
 [8] 0.4974359 0.5031250 0.4324324
```

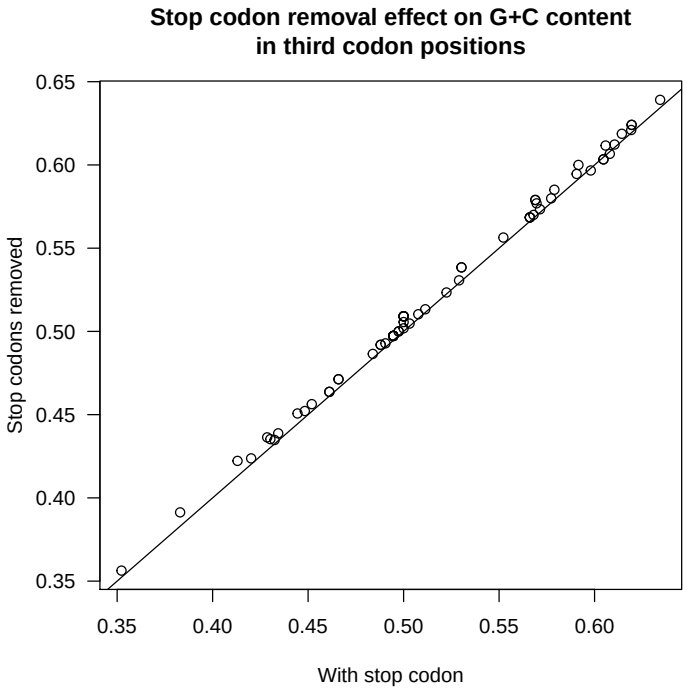
5 Why do I have not exactly the same G+C content as in codonW?

This question was raised (and solved) by Oliver Clay in an e-mail (23-AUG-2006). The program `codonW` was written in C as part of John Peden's PhD thesis on Codon Usage [8] and is available at `http://codonw.sourceforge.net/`. The reason for the small differences in G+C content between the two programs is that the default behavior in `codonW` is to remove the stop codon before computations. Here is one way of removing the stop codon under R:

```
gc3nos <- sapply(myseqs, function(s) GC3(s[1:(length(s) - 3)]))
```

As compared with the previous result, the difference is small but visible:

```
plot(x = gc3, y = gc3nos, las =1, main="Stop codon removal effect on G+C content
in third codon positions", xlab = "With stop codon", ylab ="Stop codons removed")
abline(c(0,1))
```



`CodonW` was released with a test file called `input.dat`, here are the first 10 lines of the file copied from `CodonWSourceCode_1_4_4`:

```
inputdatfile <- system.file("sequences/input.dat", package = "seqinr")
cat(readLines(inputdatfile,n=10), sep = "\n")
```

```
>YCG9 Probable          1377 residues Pha 0 Code 0
ATGAATATGCTCATTGTCGGTAGAGTTGTTGCTAGTGTTGGGGGAAGCGGACTTCAAACG
CTTTGCTTTGTTATTGGTTGTACGATGGTTGGTGAAAGGTCACGTCCATTGGTGATTTCC
ATCCTAAGTTGTGCATTTGCTGTAGCTGCTATCGTTGGTCCTATAATCGGAGGTGCCTTT
ACAACCCATGTTACCTGGAGGTGGTGCTTCTATATCAATCTTCCTATCGGTGGTCTTGCC
ATTATTATGTTTTTACTCACATATAAGGCCGAGAATAAGGGTATACTTCAACAAATTAAA
GATGCTATAGGAACAATCTCGAGCTTTACTTTTAGTAAGTTCAGACACCAAGTTAATTTT
AAAAGACTTATGAATGGCATAATCTTCAAGTTTGACTTCTTTGGTTTTGCCCTCTGCTCT
GCAGGGCTGGTCCTTTTCCTACTGGGGCTAACCTTTGGTGGTAATAAATATAGTTGGAAC
TCTGGCCAAGTCATCGCATATTTGGTTTTGGGTGTCTTACTTTTTATTTTTTCATTGGTG
```

This is a FASTA file that we import under R with:

```
input <- read.fasta(file = inputdatfile)
names(input)
```

```
 [1] "YCG9"      "YCG8"      "ALPHA2"    "ALPHA1"    "CHA1"      "KRR1"
 [7] "PRD1"      "KAR4"      "PBN1"      "LRE1"      "APA1"      "YCE9"
[13] "YCE8"      "YCE7"      "YCE5"      "YCE6"      "YCE4"      "PDI1"
[19] "GLK1"      "YCD8"      "SRO9"      "YCD6"      "YCD5"      "YCD3"
[25] "STE50"     "HIS4"      "BIK1"      "FUS1"      "YCO8"      "AGP1"
[31] "LEU2"      "NFS1"      "BUD3"      "GBP2"      "ILV6"      "CWH36"
[37] "PEL1"      "RER1"      "CDC10"     "MRPL32"    "YCP4"      "CIT2"
[43] "YCP7"      "SAT4"      "RVS161"    "YCQ0"      "ADP1"      "PGK1"
[49] "POL4"      "YCQ7"      "SRD1"      "MAK32"     "PET18"     "MAK31"
[55] "HSP30"     "YCR3"      "SYN"       "YCR6"      "GNS1"      "FEN2"
[61] "RIM1"      "CRY1"      "YCS2"      "YCS3"      "GNS1"      "RBK1"
[67] "PHO87"     "BUD5"      "MATALPHA2" "MATALPHA1" "TSM1"      "YCT5"
[73] "PETCR46"   "YCT7"      "YCT9"      "ARE1"      "RSC6"      "THR4"
```

```
 [79] "CTR86"     "PWP2"      "YCU9"      "YCV1"      "G10"       "HCM1"
 [85] "RAD18"     "CYPR"      "YCW1"      "YCW2"      "SSK22"     "SOL2"
 [91] "ERS1"      "PAT1"      "SRB8"      "YCX3"      "TUP1"      "YC16"
 [97] "ABP1"      "KIN82"     "MSH3"      "CDC39"     "YCY4"      "A2"
[103] "GIT1"      "YCZ0"      "YCZ1"      "YCZ2"      "YCZ3"      "PAU3"
[109] "YCZ5"      "YCZ6"      "YCZ7"
```

The file `input.out` contains the values obtained with `codonW` for the GC content and GC3s content:

```
inputoutfile <- system.file("sequences/input.out", package = "seqinr")
cat(readLines(inputoutfile, n=10), sep = "\n")
```

```
title                          GC3s        GC
YCG9_Probable_____________13   0.335        0.394
YCG8________573_residues_      0.439        0.446
ALPHA2________633_residue      0.328        0.351
ALPHA1________528_residue      0.345        0.379
CHA1_________1083_residue      0.328        0.394
KRR1__________951_residue      0.364        0.384
PRD1_________2139_residue      0.430        0.397
KAR4_________1008_residue      0.354        0.383
PBN1_________1251_residue      0.330        0.386
```

```
input.res <- read.table(inputoutfile, header = TRUE)
head(input.res)
```

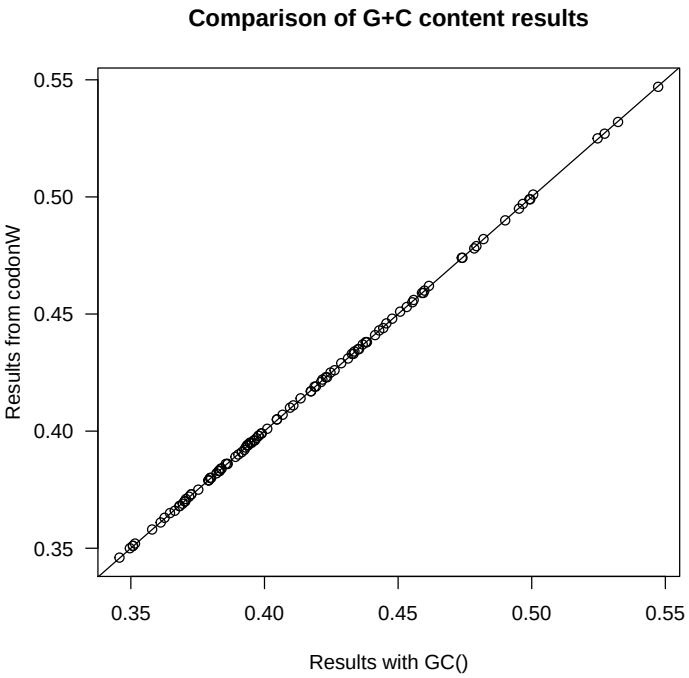
```
                      title  GC3s    GC
1 YCG9_Probable_____________13 0.335 0.394
2 YCG8________573_residues_ 0.439 0.446
3 ALPHA2________633_residue 0.328 0.351
4 ALPHA1________528_residue 0.345 0.379
5 CHA1_________1083_residue 0.328 0.394
6 KRR1__________951_residue 0.364 0.384
```

Let's try to reproduce the results for the G+C content, we know that we have to remove the last stop codon:

```
input.gc <- sapply(input, function(s) GC(s[1:(length(s)-3)]))
max(abs(input.gc - input.res$GC))
```

```
[1] 0.0004946237
```

```
plot(x = input.gc, y = input.res$GC, las = 1,
xlab = "Results with GC()", ylab = "Results from codonW",
main = "Comparison of G+C content results")
abline(c(0,1))
```

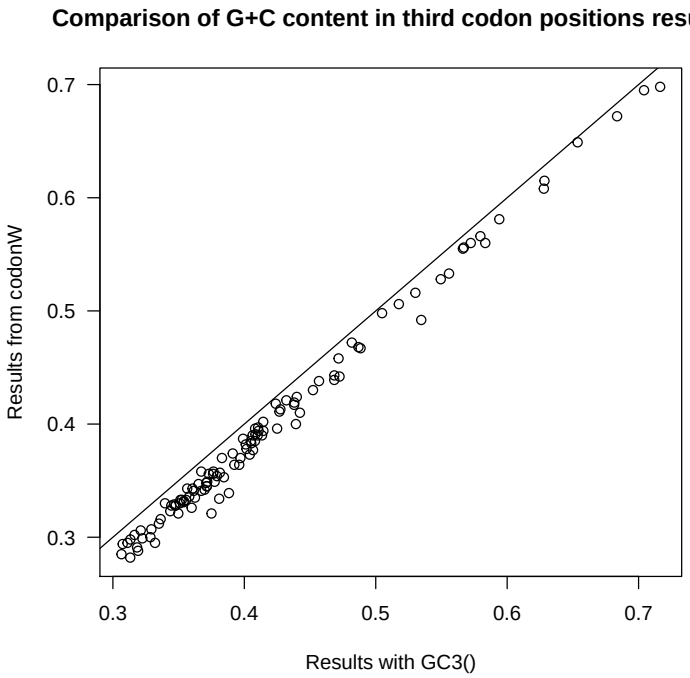


The results are consistent if we consider that we have 3 significant digits in the file `input.out`. Now, let's try to reproduce the results for G+C in third codon positions:

```
input.gc3 <- sapply(input, function(s) GC3(s[1:(length(s)-3)]))
max(abs(input.gc3 - input.res$GC3s))
```

```
[1] 0.054
```

```
plot(x = input.gc3, y = input.res$GC3s, las = 1,
xlab = "Results with GC3()", ylab = "Results from codonW",
main = "Comparison of G+C content in third codon positions results")
abline(c(0,1))
```

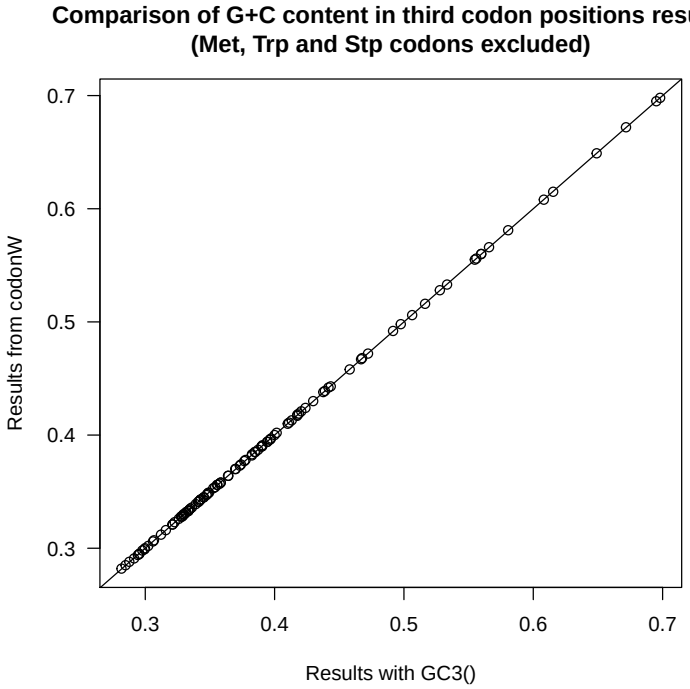


There is clearly a problem here. Looking into the documentation of `codonW`, GC3s is the G+C content in third codon position after removing non-synonymous and stop codons (those corresponding to Met, Trp, Stp). Let's remove these codons:

```
codons <- words()
names(codons) <- sapply(codons, function(c) aaa(translate(s2c(c), numcode = 1)))
okcodons <- codons[! names(codons) %in% c("Met", "Trp", "Stp")]
gc3s <- function(s){
  tmp <- splitseq(s)
  tmp <- tmp[tmp %in% okcodons]
  tmp <- s2c(paste(tmp, collapse = ""))
  GC3(tmp)
}
input.gc3s <- sapply(input, gc3s)
max(abs(input.gc3s - input.res$GC3s))
```

```
[1] 0.0004980843
```

```
plot(x = input.gc3s, y = input.res$GC3s, las = 1,
xlab = "Results with GC3()", ylab = "Results from codonW",
main = "Comparison of G+C content in third codon positions results\n(Met, Trp and Stp codons excluded)")
abline(c(0,1))
```



The results are now consistent. But thinking more about it there is still a problem with the codons for Ile:

```
 codons[names(codons) == "Ile"]
  Ile   Ile   Ile
"ata" "atc" "att"
```

There are three codons for Ile. If the distribution of the four bases was uniform and selectively neutral in third codon position of synonymous codons, then we would expect to get a G+C of 50% in quartet and duet codons at third codons positions because they all have the same number of W (A or T)and S (C or G) bases in third position. But for Ile we have two codons ending in W versus only one in S so that we would get a G+C of $\frac{1}{3}$ instead of $\frac{1}{2}$. This point was clearly stated [11] by Sueoka in 1988:

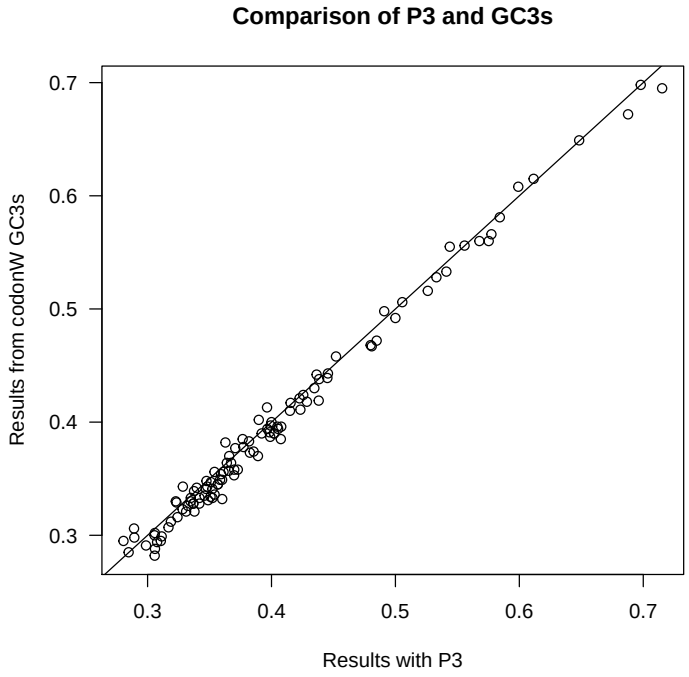
> **G + C Content of the Three Codons Positions.** In the present analysis, observed G + C contents of the first, second, and third codon positions (P_1, P_2, and P_3, respectively) are corrected average G + C contents of the three codon positions that are calculated from 56 triplets out of 64. Because of the inequality of α and γ at the third codon position, the three stop codons (TAA, TAG, and TGA) and the three codons for isoleucine (ATT, ATC, and ATA) were excluded in calculation of P_3, and two single codons for methionine (ATG) and tryptophan (TGG) were excluded in all three (P_1, P_2, and P_3)

Let's compute P_3 and compare it with GC3s:

```
P3codons <- codons[! names(codons) %in% c("Met", "Trp", "Ile", "Stp")]
P3 <- function(s){
  tmp <- splitseq(s)
  tmp <- tmp[tmp %in% P3codons]
  tmp <- s2c(paste(tmp, collapse = ""))
  GC3(tmp)
}
input.P3 <- sapply(input, P3)
max(abs(input.P3 - input.res$GC3s))
```

```
[1] 0.02821505
```

```
plot(x = input.P3, y = input.res$GC3s, las = 1,
xlab = "Results with P3", ylab = "Results from codonW GC3s",
main = "Comparison of P3 and GC3s")
abline(c(0,1))
```



This is not exactly the same, the maximum observed difference here is about 3%. In practice, P_3, GC3, and GC3s are only slightly different [12].

6 How do I get a sequence from its name?

This question is adapted from an e-mail (22 Jun 2006) by Gang Xu. I know that the UniProt (SwissProt) entry of my protein is `P08758`, if I know its name[3], how can I get the sequence?

[3] More exactly, this is the accession number. Sequence names are not stable over time, it's always better to use the accession numbers.

```
choosebank("swissprot")
myprot <- query("myprot","AC=P08758")
getSequence(myprot$req[[1]])
```

```
  [1] "M" "A" "Q" "V" "L" "R" "G" "T" "V" "T" "D" "F" "P" "G" "F" "D" "E" "R"
 [19] "A" "D" "A" "E" "T" "L" "R" "K" "A" "M" "K" "G" "L" "G" "T" "D" "E" "E"
 [37] "S" "I" "L" "T" "L" "L" "T" "S" "R" "S" "N" "A" "Q" "R" "Q" "E" "I" "S"
 [55] "A" "A" "F" "K" "T" "L" "F" "G" "R" "D" "L" "L" "D" "D" "L" "K" "S" "E"
 [73] "L" "T" "G" "K" "F" "E" "K" "L" "I" "V" "A" "L" "M" "K" "P" "S" "R" "L"
 [91] "Y" "D" "A" "Y" "E" "L" "K" "H" "A" "L" "K" "G" "A" "G" "T" "N" "E" "K"
[109] "V" "L" "T" "E" "I" "I" "A" "S" "R" "T" "P" "E" "E" "L" "R" "A" "I" "K"
[127] "Q" "V" "Y" "E" "E" "E" "Y" "G" "S" "S" "L" "E" "D" "D" "V" "V" "G" "D"
[145] "T" "S" "G" "Y" "Y" "Q" "R" "M" "L" "V" "V" "L" "L" "Q" "A" "N" "R" "D"
[163] "P" "D" "A" "G" "I" "D" "E" "A" "Q" "V" "E" "Q" "D" "A" "Q" "A" "L" "F"
[181] "Q" "A" "G" "E" "L" "K" "W" "G" "T" "D" "E" "E" "K" "F" "I" "T" "I" "F"
[199] "G" "T" "R" "S" "V" "S" "H" "L" "R" "K" "V" "F" "D" "K" "Y" "M" "T" "I"
[217] "S" "G" "F" "Q" "I" "E" "E" "T" "I" "D" "R" "E" "T" "S" "G" "N" "L" "E"
[235] "Q" "L" "L" "L" "A" "V" "V" "K" "S" "I" "R" "S" "I" "P" "A" "Y" "L" "A"
[253] "E" "T" "L" "Y" "Y" "A" "M" "K" "G" "A" "G" "T" "D" "D" "H" "T" "L" "I"
[271] "R" "V" "M" "V" "S" "R" "S" "E" "I" "D" "L" "F" "N" "I" "R" "K" "E" "F"
[289] "R" "K" "N" "F" "A" "T" "S" "L" "Y" "S" "M" "I" "K" "G" "D" "T" "S" "G"
[307] "D" "Y" "K" "K" "A" "L" "L" "L" "L" "C" "G" "E" "D" "D"
```

Session Informations

This part was compiled under the following R environment:

- R version 3.2.4 (2016-03-10), `x86_64-apple-darwin13.4.0`
- Locale: `fr_FR.UTF-8/fr_FR.UTF-8/fr_FR.UTF-8/C/fr_FR.UTF-8/fr_FR.UTF-8`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: ade4 1.7-4, ape 3.5, grImport 0.9-0, MASS 7.3-45, seqinr 3.2-1, tseries 0.10-35, XML 3.98-1.4, xtable 1.8-2
- Loaded via a namespace (and not attached): lattice 0.20-33, nlme 3.1-125, quadprog 1.5-5, tools 3.2.4, zoo 1.7-12

There were two compilation steps:

- R compilation time was: Tue Jul 12 21:02:14 2016
- LaTeX compilation time was: July 12, 2016

References

[1] F.R. Blattner, V. Burland, G. Plunkett, H.J. Sofia, and D.L. Daniels. Analysis of the *Escherichia coli* genome. IV. DNA sequence of the region from 89.2 to 92.8 minutes. *Nucleic Acids Research*, 21:5408–5417, 1993.

[2] F.R. Blattner, G. Plunkett III, C.A. Bloch, N.T. Perna, V. Burland, M. Riley, J. Collado-Vides, J.D. Glasner, C.K. Rode, G.F. Mayhew, J. Gregor, N.W. Davis, H.A. Kirkpatrick, M.A. Goeden, D.J. Rose, B. Mau, and Y. Shao. The complete genome sequence of *Escherichia coli* K-12. *Science*, 277:1453–1462, 1997.

[3] V. Burland, G. Plunkett, D.L. Daniels, and F.R. Blattner. DNA sequence and analysis of 136 kilobases of the *Escherichia coli* genome: organizational symmetry around the origin of replication. *Genomics*, 16:551–561, 1993.

[4] D.L. Daniels, G. Plunkett, V. Burland, and F.R. Blattner. Analysis of the *Escherichia coli* genome: DNA sequence of the region from 84.5 to 86.5 minutes. *Science*, 257:771–778, 1992.

[5] K. Hayashi, N. Morooka, Y. Yamamoto, K. Fujita, K. Isono, S. Choi, E. Ohtsubo, T. Baba, B.L. Wanner, H. Mori, and T. Horiuchi. Highly accurate genome sequences of *Escherichia coli* K-12 strains MG1655 and W3110. *Molecular Systems Biology*, 2:2006.0007, 2006.

[6] J.R. Lobry. Asymmetric substitution patterns in the two DNA strands of bacteria. *Molecular Biology and Evolution*, 13:660–665, 1996.

[7] P. Murrell. *R Graphics*. Computer Science & Data Analysis. Chapman & Hall/CRC, New York, 2005. ISBN: 9781584884866 `http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html`.

[8] J.F. Peden. *Analysis of codon usage*. PhD thesis, University of Nottingham, 1999.

[9] G. Plunkett, V. Burland, D.L. Daniels, and F.R. Blattner. Analysis of the *Escherichia coli* genome. III. DNA sequence of the region from 87.2 to 89.2 minutes. *Nucleic Acids Research*, 21:3391–3398, 1993.

[10] H.J. Sofia, V. Burland, D.L. Daniels, G. Plunkett, and F.R. Blattner. Analysis of the *Escherichia coli* genome. V. DNA sequence of the region from 76.0 to 81.5 minutes. *Nucleic Acids Research*, 22:2576–2586, 1994.

[11] N. Sueoka. Directional mutation pressure and neutral molecular evolution. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2653 –2657, 1988.

[12] N. Sueoka. Two aspects of DNA base composition: G+C content and translation-coupled deviation from intra-strand rule of $A = T$ and $G = C$. *J. Mol. Evol.*, 49:49–62, 1999.

[13] T. Yura, H. Mori, H. Nagai, T. Nagata, A. Ishihama, N. Fujita, K. Isono, K. Mizobuchi, and A. Nakata. Systematic sequencing of the *Escherichia coli* genome: analysis of the 0-2.4 min region. *Nucleic Acids Research*, 20:3305–3308, 1992.