GC Content Analysis I

Gerald McCollam

2017-07-05

GC Content Analysis I

NOTE: You must first set your working directory for this to work.

```
# Set the working directory to your repository location.
# setwd('/path/to/repository/')
setwd("/Users/geraldmccollam/2017Code/atallah/transposable_elements/R-Notebooks")

# Source the full genome sequence for Homo sapiens as provided by UCSC (hg38, Dec. 2013)
 and store it in a Biostrings object.
source("http://bioconductor.org/biocLite.R")
```

```
## Bioconductor version 3.5 (BiocInstaller 1.26.0), ?biocLite for help
```

```
# Download the full hg38 genome. This may take a while...
biocLite("BSgenome.Hsapiens.UCSC.hg38")
```

```
## BioC_mirror: https://bioconductor.org
```

```
## Using Bioconductor 3.5 (BiocInstaller 1.26.0), R 3.4.0 (2017-04-21).
```

```
# Require hg38, in order to extract the data.
require(BSgenome.Hsapiens.UCSC.hg38)
```

```
# Load only the sequence for chromosome 17 into memory.
c17um = unmasked(Hsapiens$chr17)
c17um
```

```
##   83257441-letter "DNAString" instance
## seq: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

1. The first step is to load in all required packages in a form that's convenient. We need the latest release of R installed first, then we'll get the latest version of Bioconductor (by starting R and entering the commands to source biocLite.R). Functionality provided by biocLite will install or update all Bioconductor and CRAN packages for the latest Bioconductor release. We then use 'biocLite' to download the complete human genome. In the next step we 'require' use of hg38, a recent version of the human genome. This step does not load the sequence into memory! Finally, we reference only our section of interest (ch17) which does load the 'unmasked' section, a 83257441-letter "DNAString" instance, into memory. "Unmasked" in general means that repetitive parts of the reference genome are made available.

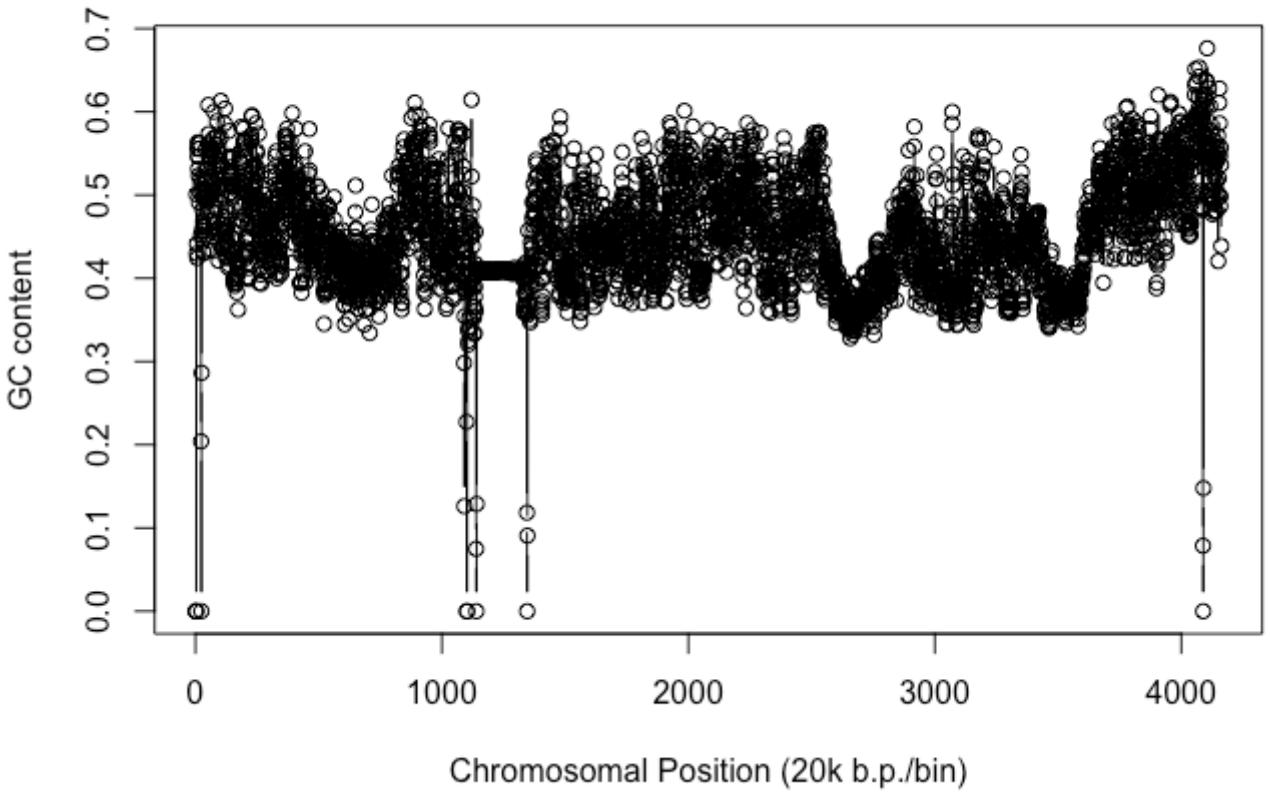
```
gcContent <-
  function(x)
  {
    alf <- alphabetFrequency(x, as.prob=TRUE)
    sum(alf[c("G", "C")])
  }
```

2. Lines 31-37 contain a function to calculate the GC content of any sequence. GC-content (or guanine-cytosine content) is the percentage of bases of a DNA or RNA sequence that are either guanine or cytosine (also including adenine and thymine in DNA and adenine and uracil in RNA as possible entries). We'll use this in code that follows.

```
# Get a list of chunks, one chunk equals 20k base pairs.
chunks <- seq(1, length(c17um)-20000, by=20000)

n <- length(chunks) # the length of vector 'chunks'
GCchunks <- numeric(n) # Initialize a vector of length 'n', containing zeroes.

for (i in 1:n) {
  chunk <- c17um[chunks[i]:(chunks[i]+19999)]
  GCchunk <- gcContent(chunk)
  GCchunks[i] <- GCchunk
  }
```

3. In line 26 we loaded only chromosome 17 from the complete hg38 object. In line 45 we set up a list of indices of length 20,000 (b.p.). This will allow us to cut the entire chromosome up into roughly 4160 'chunks' of equal length. In line 48 we initialize a numeric vector to hold the GC content value of each of chunk.

```
plot(GCchunks,type="b",xlab="Chromosomal Position (20k b.p./bin)", ylab="GC content")
```

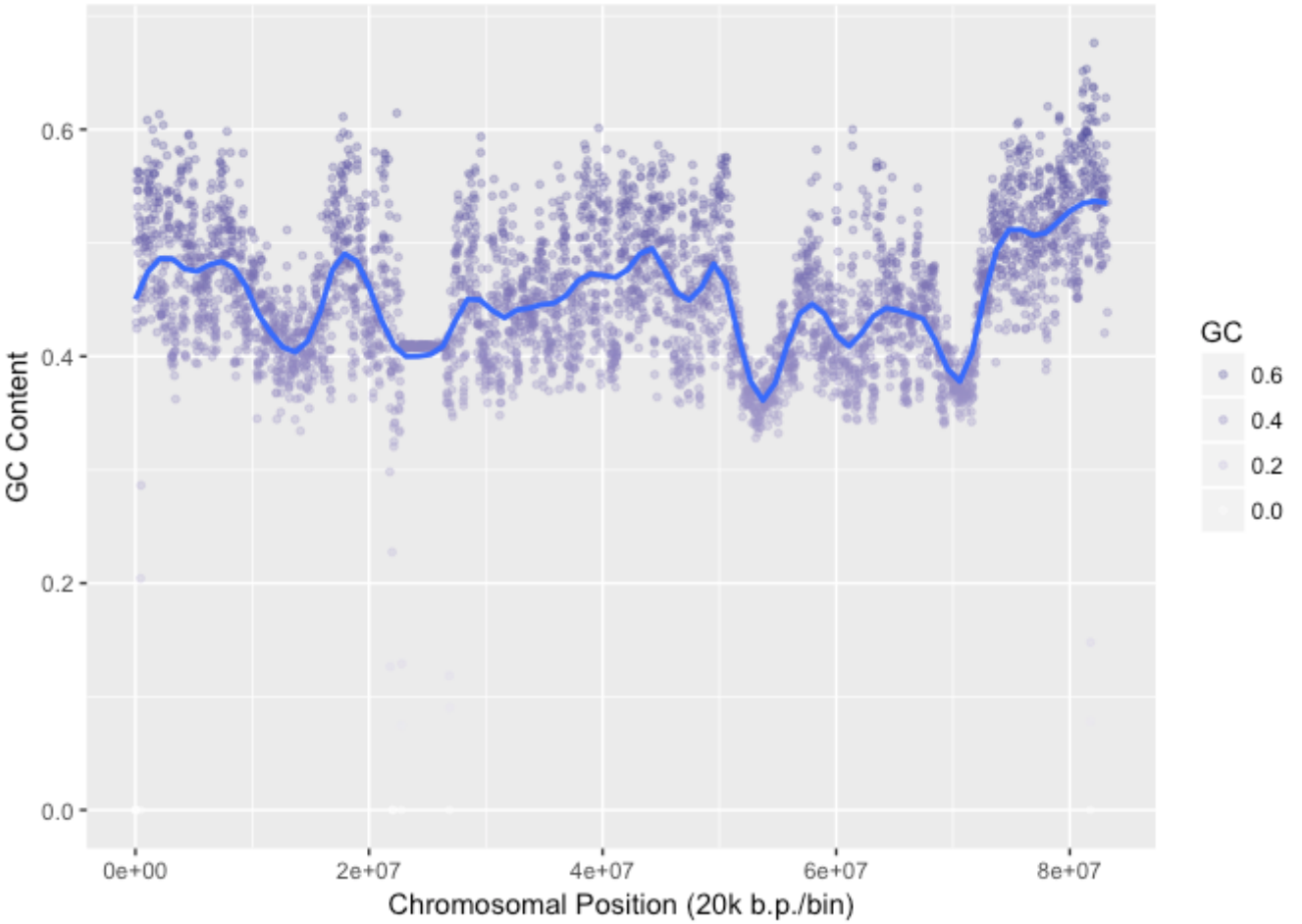


4.

Line 59 provides a simple scatter plot of GC content in terms of position along the entire chromosome sequence - from b.p. 1 to 83,257,441. Each point represents 20,000 base pairs which is a fairly course representation. The plot is useful but we can do better.

```
library(ggplot2)

GC <- as.data.frame(GCchunks) # ggplot2 requires a data.frame

ggplot(GC, aes(x = chunks, y = GC)) + geom_point(aes(colour = GC), alpha = 0.3, size =
1) + geom_smooth(method = "loess", se = FALSE, span = 1/10) + xlab("Chromosomal Position
 (20k b.p./bin)") + ylab("GC Content") + scale_colour_gradient2(guide=guide_legend(rever
se = TRUE))
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting
 to continuous.
```



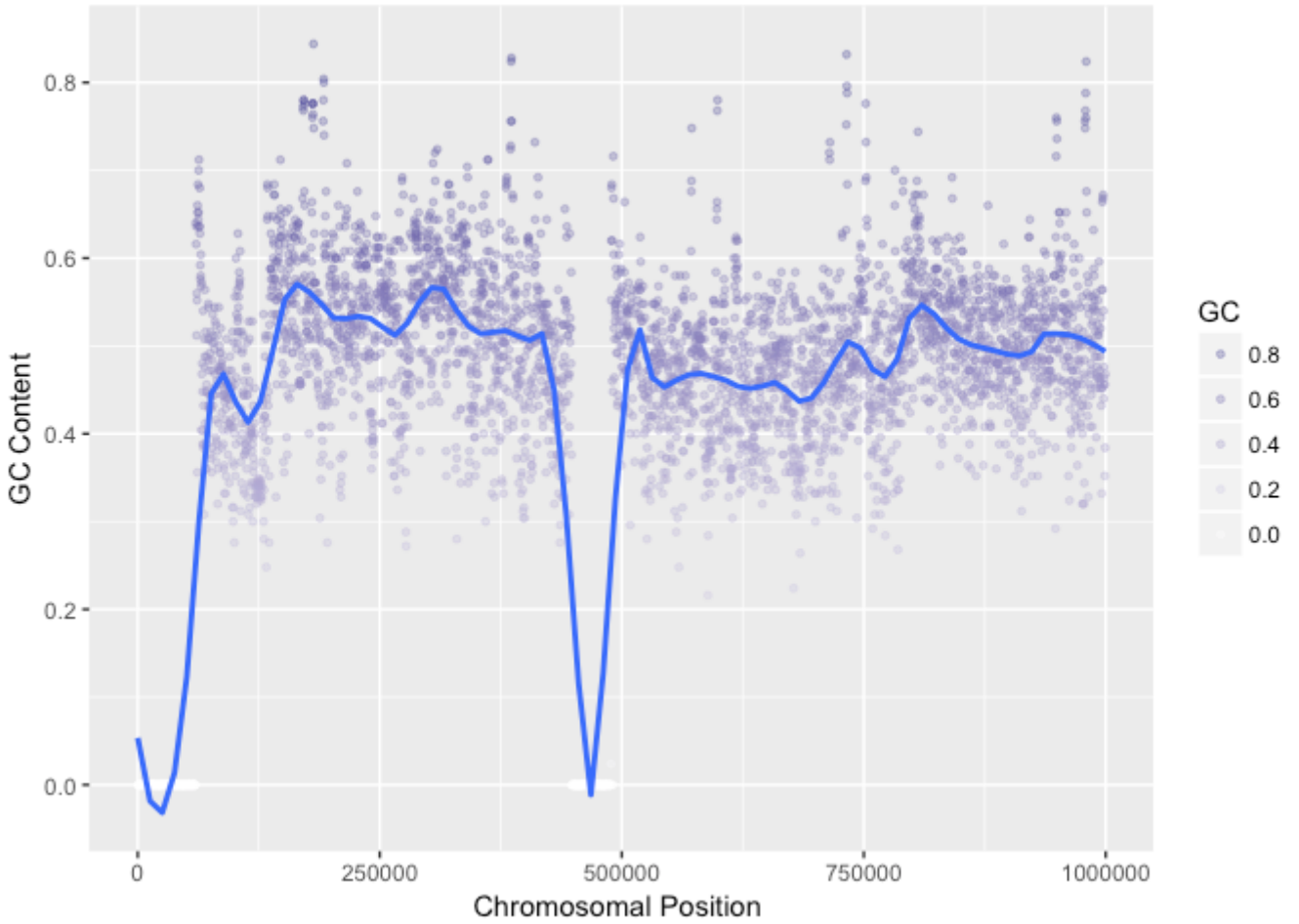
5. Here we use Hadley Wickam's ggplot2 package to create a more meaningful view of GC content across chromosome 17. ggplot2 is a plotting system for R, based on the grammar of graphics. It takes care of many details that make plotting a chore (like drawing legends) as well as providing a model of graphics that makes it easier to produce complex multi-layered graphics. Before making use of ggplot we first convert an numeric vector object into a data frame (line 68). This step is mandatory!

```
GCPlot <- function(windowsize, inputseq)
{
   chunks <- seq(1, length(inputseq)-windowsize, by = windowsize)
   n <- length(chunks) # the length of vector 'chunks'
   GCchunks <- numeric(n) # Initialize a vector of length 'n', containing zeroes.
   for (i in 1:n) {
        chunk <- inputseq[chunks[i]:(chunks[i]+windowsize-1)]
        GCchunk <- gcContent(chunk)
        GCchunks[i] <- GCchunk
   }
GC <- as.data.frame(GCchunks) # ggplot2 requires a data.frame
ggplot(GC, aes(x = chunks, y = GC)) + geom_point(aes(colour = GC), alpha = 0.3, size =
1) + geom_smooth(method = "loess", se = FALSE, span = 1/10) + xlab("Chromosomal Position
 ") + ylab("GC Content") + scale_colour_gradient2(guide=guide_legend(reverse = TRUE))
}
```

6. Being able to do a quick plot of an entire chromosome is really helpful but we would like to dig deeper and visualize the GC content of any sub-sequence of the entire chromosome. Lines 76-88 provide this ability with GCPlot. The function takes two arguments: an integer representing a sequence length and the sequence to be queried. It returns a ggplot object.

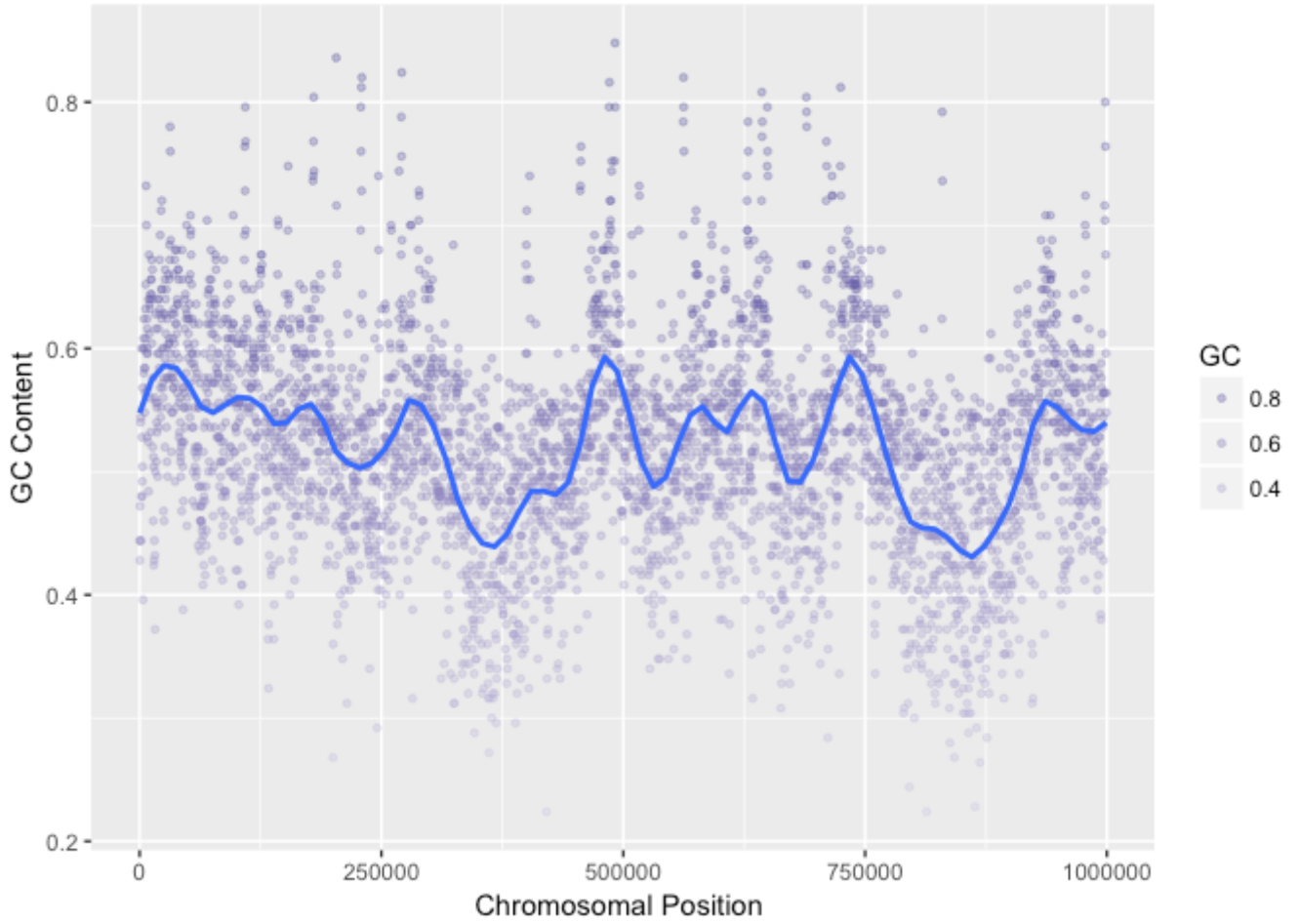
```
c17um_firstmil <- c17um[1:1000000]
c17um_secondmil <- c17um[1000001:2000000]
c17um_thirdmil <- c17um[2000001:3000000]
c17um_fourthmil <- c17um[3000001:4000000]
c17um_fifthmil <- c17um[4000001:5000000]

GCPlot(250, c17um_firstmil) # plots the first 1000k base pairs, in chunks of 250.
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting
 to continuous.
```



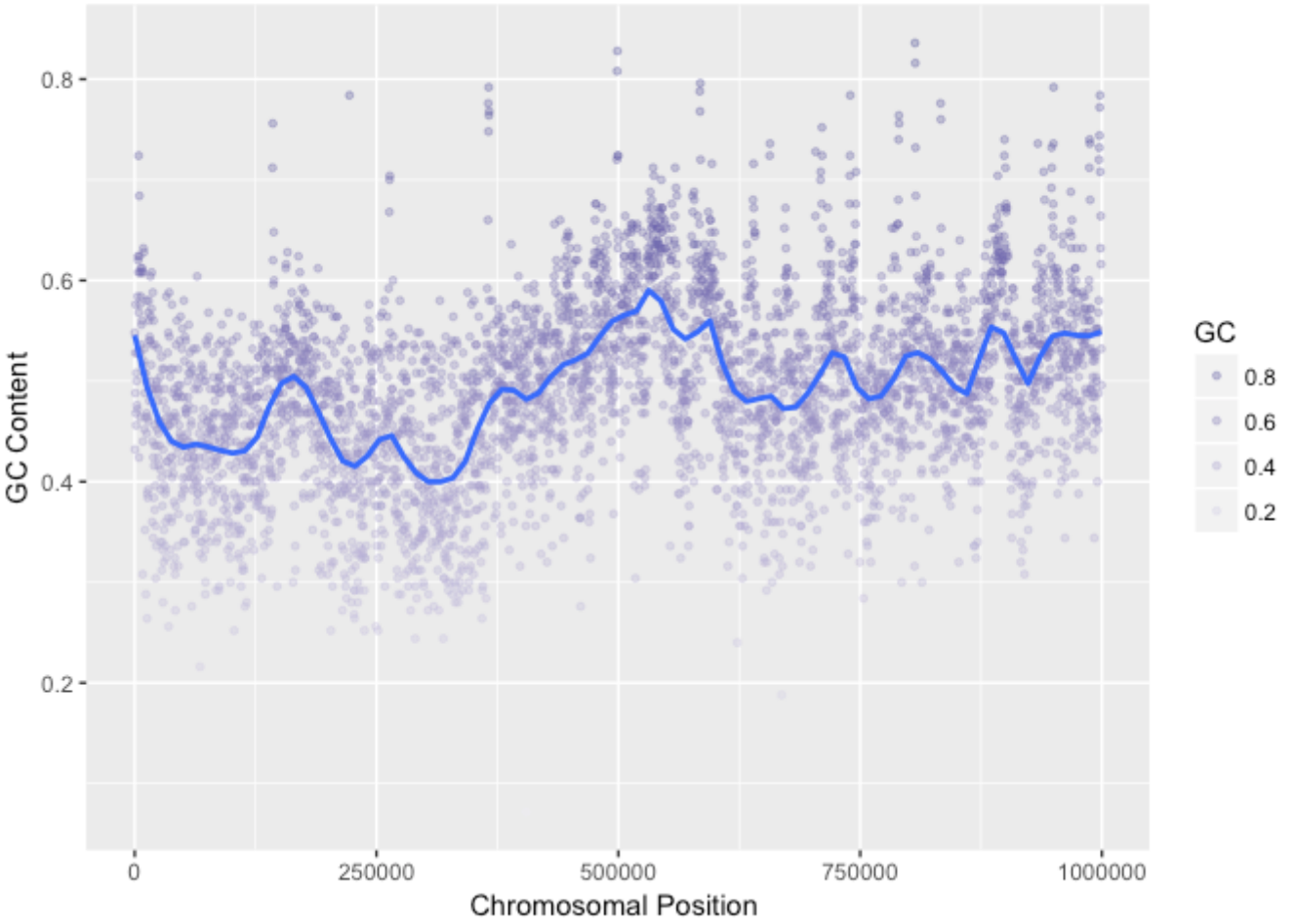

```
GCPlot(250, c17um_secondmil) # plots the second 1000k base pairs.
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting
 to continuous.
```




```
GCPlot(250, c17um_fifthmil)  # plots the fifth 1000k base pairs.
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting
 to continuous.
```


0.8
0.6
0.4
0.2
GC Content
0
250000
500000
750000
1000000
Chromosomal Position
GC
0.8
0.6
0.4
0.2