Genomic Sequence Data Analysis

MCB517A: Tools for Computational Biology - University of Washington 2020

Lavinia Carabet

12/20/2020

1: Overlaps between genomic regions and copy number alterations.

Data: **BRCA.genome__wide__snp__6__broad__Level__3__scna.seg** TCGA segment data for primary breast cancer patient samples

2: Frequency of copy number alteration events within genomic regions.

Data: **BRCA.genome__wide__snp__6__broad__Level__3__scna.seg**

3: Reading and extracting sequencing data

Data: **BRCA__IDC__cfDNA.bam and BRCA__IDC__cfDNA.bai** Generated dataset from Sequence Read Archive (SRA) https://www.ncbi.nlm.nih.gov/sra/?term=SRR2130004 and Gene Expression Omnibus (GEO) https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE71378 sequencing data from Illumina HiSeq 2000 platform, paired layout, of cell-free cfDNA libraries ("Cell-free DNA comprises an in vivo, genome-wide nucleosome footprint that informs its tissue(s)-of-origin")

4: Reading and annotating genomic variants

Data: **GIAB__highconf__v.3.3.2.vcf.gz** Publicly available dataset of genomic variant calls for the infamous individual, NA12878, a Utah woman, mormon, mother of 11 with a genetic disease, a CYP2D6 mutation. The Genome-in-a-Bottle (GIAB) Consortium has compiled consensus variant calls on this individual's genome. One of the main purposes of this data is to provide a benchmark for those who develop computational tools and analysis of human genomes. https://github.com/genome-in-a-bottle/giab_latest_release

1: Overlaps between genomic regions and copy number alterations.

Preparation

1.1 Load copy number segments in *SEG format* from BRCA.genome__wide__snp__6__broad__L

SEGment Data (http://software.broadinstitute.org/software/igv/SEG) format is tab-delimited and a flexible way to define any genomic data. The segmentation data file contains TCGA information for primary breast cancer patient samples.

Use `GenomicRanges` library for representation and manipulation of genomic intervals to create a `GRanges` object, which contains among others, an attribute called `seqnames` to represent chromosomes and `ranges` attribute to represent the `start` and `end` coordinates.

```
library(GenomicRanges)
library(tidyverse)
segs <- read.delim("BRCA.genome_wide_snp_6_broad_Level_3_scna.seg", as.is = TRUE)
head(segs, 5)
```

```
##                         Sample Chromosome     Start       End Num_Probes
## 1 TCGA-3C-AAAU-10A-01D-A41E-01          1   3218610  95674710      53225
## 2 TCGA-3C-AAAU-10A-01D-A41E-01          1  95676511  95676518          2
## 3 TCGA-3C-AAAU-10A-01D-A41E-01          1  95680124 167057183      24886
## 4 TCGA-3C-AAAU-10A-01D-A41E-01          1 167057495 167059336          3
## 5 TCGA-3C-AAAU-10A-01D-A41E-01          1 167059760 181602002       9213
##   Segment_Mean
## 1       0.0055
## 2      -1.6636
## 3       0.0053
## 4      -1.0999
## 5      -0.0008
```

```
mode(segs$Chromosome) <- "character"
segs[segs$Chromosome == 23, "Chromosome"] <- "X"
# create a GRanges object from segs
segs.gr <- as(segs, "GRanges")
segs.gr
```

```
## GRanges object with 284458 ranges and 3 metadata columns:
##             seqnames              ranges strand |                      Sample
##                <Rle>           <IRanges>  <Rle> |                 <character>
##        [1]        1    3218610-95674710      * | TCGA-3C-AAAU-10A-01D-A41E-01
##        [2]        1   95676511-95676518      * | TCGA-3C-AAAU-10A-01D-A41E-01
##        [3]        1  95680124-167057183      * | TCGA-3C-AAAU-10A-01D-A41E-01
##        [4]        1 167057495-167059336      * | TCGA-3C-AAAU-10A-01D-A41E-01
##        [5]        1 167059760-181602002      * | TCGA-3C-AAAU-10A-01D-A41E-01
##        ...      ...                 ...    ... .                          ...
##   [284454]       19     284018-58878226      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [284455]       20     455764-62219837      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [284456]       21   15347621-47678774      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [284457]       22   17423930-49331012      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [284458]        X   3157107-154905589      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##             Num_Probes Segment_Mean
##              <integer>    <numeric>
##        [1]      53225       0.0055
##        [2]          2      -1.6636
##        [3]      24886       0.0053
##        [4]          3      -1.0999
##        [5]       9213       -8e-04
##        ...        ...          ...
##   [284454]      23950       -0.117
##   [284455]      37283       0.3435
##   [284456]      20582      -0.1117
```

```
##   [284457]      16927      -0.1231
##   [284458]      63797       0.0014
##   -------
##   seqinfo: 23 sequences from an unspecified genome; no seqlengths
```

Preparation

1.2 SeqInfo for genome build and chromosome naming conventions

Get genome build information - the human genome build, hg19 - directly by installing and loading the`BSgenome.Hsapiens.UCSC.hg19` library

```
library(BSgenome.Hsapiens.UCSC.hg19)

# load the 'SeqInfo' object for human genome hg19
seqinfo2 <- seqinfo(get("BSgenome.Hsapiens.UCSC.hg19"))

# chromosome naming convention for matching and querying genomic regions.
# For the human genome reference, there are 2 conventions for hg19:

#  1. UCSC - uses "chr" string in front of the chromosome number (e.g. chr1)
#  2. NCBI - does not add a string to the front of the chromosome number (e.g. 1)

seqlevelsStyle(seqinfo2) <- "NCBI"
seqinfo2
```

```
## Seqinfo object with 93 sequences (1 circular) from hg19 genome:
##   seqnames       seqlengths isCircular genome
##   1               249250621      FALSE   hg19
##   2               243199373      FALSE   hg19
##   3               198022430      FALSE   hg19
##   4               191154276      FALSE   hg19
##   5               180915260      FALSE   hg19
##   ...                   ...        ...    ...
##   chrUn_gl000245      36651      FALSE   hg19
##   chrUn_gl000246      38154      FALSE   hg19
##   chrUn_gl000247      36422      FALSE   hg19
##   chrUn_gl000248      39786      FALSE   hg19
##   chrUn_gl000249      38502      FALSE   hg19
```

```
# work with the known autosomes and sex chromosomes in NCBI format, '1:22, "X", "Y"'.
chrs <- c(1:22, "X", "Y") # NCBI format
seqinfo3 <- keepSeqlevels(seqinfo2, value = chrs) #select the autosomes and sex chromosomes
seqinfo3
```

```
## Seqinfo object with 24 sequences from hg19 genome:
##   seqnames seqlengths isCircular genome
##   1         249250621      FALSE   hg19
##   2         243199373      FALSE   hg19
##   3         198022430      FALSE   hg19
##   4         191154276      FALSE   hg19
##   5         180915260      FALSE   hg19
```

```
##   ...             ...        ...    ...
##   20         63025520      FALSE   hg19
##   21         48129895      FALSE   hg19
##   22         51304566      FALSE   hg19
##   X         155270560      FALSE   hg19
##   Y          59373566      FALSE   hg19
```

a. Find the segments in `segs.gr` that have *any* overlap with the region `chr8:128,746,347-128,755,810`

Print out the first five unique TCGA IDs.

```
seqinfo(segs.gr) <- seqinfo3

#slen <- seqlengths(seqinfo3) # get the length of the chromosomes
tileWidth <- 500000 # tile size of 500kb
#divide the genome into tiles/windows/bins
tiles <- tileGenome(seqlengths = seqlengths(seqinfo3), tilewidth = tileWidth,
                    cut.last.tile.in.chrom = TRUE)
tiles
```

```
## GRanges object with 6206 ranges and 0 metadata columns:
##          seqnames            ranges strand
##             <Rle>         <IRanges>  <Rle>
##      [1]        1          1-500000      *
##      [2]        1     500001-1000000      *
##      [3]        1    1000001-1500000      *
##      [4]        1    1500001-2000000      *
##      [5]        1    2000001-2500000      *
##      ...      ...                ...    ...
##   [6202]        Y 57000001-57500000      *
##   [6203]        Y 57500001-58000000      *
##   [6204]        Y 58000001-58500000      *
##   [6205]        Y 58500001-59000000      *
##   [6206]        Y 59000001-59373566      *
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

```
q <- GRanges(seqnames = "8", ranges = IRanges(start = 128746347, end = 128755810))
tiles.subset <- subsetByOverlaps(x = tiles, ranges = q)
tiles.subset
```

```
## GRanges object with 1 range and 0 metadata columns:
##       seqnames              ranges strand
##          <Rle>           <IRanges>  <Rle>
##   [1]        8 128500001-129000000      *
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

```
# find the indices of the elements that overlap between two `GRanges`
hits1 <- findOverlaps(query = tiles.subset, subject = segs.gr, type = "any")
class(hits1)
```

```
## [1] "SortedByQueryHits"
## attr(,"package")
## [1] "S4Vectors"
```

```
hits1
```

```
## Hits object with 2308 hits and 0 metadata columns:
##          queryHits subjectHits
##          <integer>   <integer>
##      [1]         1          34
##      [2]         1         205
##      [3]         1         426
##      [4]         1         569
##      [5]         1         837
##      ...       ...         ...
##   [2304]         1      283588
##   [2305]         1      283688
##   [2306]         1      283750
##   [2307]         1      284038
##   [2308]         1      284345
##   -------
##   queryLength: 1 / subjectLength: 284458
```

```
# create 'DataFrame' object that contains the columns in both 'query' and 'subject'
# for overlapping ranges
df7 <- mergeByOverlaps(query = tiles.subset, subject = segs.gr, type = "any")
df7
```

```
## DataFrame with 2308 rows and 5 columns
##               tiles.subset              segs.gr                      Sample
##                  <GRanges>           <GRanges>                 <character>
## 1    8:128500001-129000000  8:40440599-145232496 TCGA-3C-AAAU-10A-01D-A41E-01
## 2    8:128500001-129000000  8:87789846-145232496 TCGA-3C-AAAU-01A-11D-A41E-01
## 3    8:128500001-129000000    8:617667-145232496 TCGA-3C-AALI-10A-01D-A41E-01
## 4    8:128500001-129000000  8:62096568-145232496 TCGA-3C-AALI-01A-11D-A41E-01
## 5    8:128500001-129000000  8:82383848-145232496 TCGA-3C-AALJ-10A-01D-A41E-01
## ...                    ...                 ...                        ...
## 2304 8:128500001-129000000  8:55674507-145232496 TCGA-XX-A89A-01A-11D-A36I-01
## 2305 8:128500001-129000000    8:617667-145232496 TCGA-Z7-A8R5-10A-01D-A41E-01
## 2306 8:128500001-129000000    8:617667-145232496 TCGA-Z7-A8R5-01A-42D-A41E-01
## 2307 8:128500001-129000000 8:125418855-145232496 TCGA-Z7-A8R6-10A-01D-A41E-01
## 2308 8:128500001-129000000 8:114820195-145232496 TCGA-Z7-A8R6-01A-11D-A41E-01
##      Num_Probes Segment_Mean
##       <integer>    <numeric>
## 1         58252      -0.0018
## 2         34156       0.2819
## 3         81529        0.001
## 4         48514       0.3373
## 5         36769      -0.0023
## ...         ...          ...
## 2304      52447        0.108
## 2305      81542        1e-04
## 2306      81542       0.0554
```

```
## 2307      12351       0.0322
## 2308      19014       0.6804
```

```
df7$Sample[1:5]
```

```
## [1] "TCGA-3C-AAAU-10A-01D-A41E-01" "TCGA-3C-AAAU-01A-11D-A41E-01"
## [3] "TCGA-3C-AALI-10A-01D-A41E-01" "TCGA-3C-AALI-01A-11D-A41E-01"
## [5] "TCGA-3C-AALJ-10A-01D-A41E-01"
```

```
unique(df7$Sample)[1:5]
```

```
## [1] "TCGA-3C-AAAU-10A-01D-A41E-01" "TCGA-3C-AAAU-01A-11D-A41E-01"
## [3] "TCGA-3C-AALI-10A-01D-A41E-01" "TCGA-3C-AALI-01A-11D-A41E-01"
## [5] "TCGA-3C-AALJ-10A-01D-A41E-01"
```

b. Find the mean of the `Segment_Mean` values for copy number segments that have *any* overlap with the region chr17:37,842,337-37,886,915.

```
q17 <- GRanges(seqnames = "17", ranges = IRanges(start = 37842337, end = 37886915))
tiles.subset <- subsetByOverlaps(x = tiles, ranges = q17)
tiles.subset
```

```
## GRanges object with 1 range and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]       17 37500001-38000000      *
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

```
hits17 <- findOverlaps(query = tiles.subset, subject = segs.gr, type = "any")
class(hits17)
```

```
## [1] "SortedByQueryHits"
## attr(,"package")
## [1] "S4Vectors"
```

```
hits17
```

```
## Hits object with 2397 hits and 0 metadata columns:
##          queryHits subjectHits
##          <integer>   <integer>
##      [1]         1          57
##      [2]         1         315
##      [3]         1         453
##      [4]         1         672
##      [5]         1         864
##      ...       ...         ...
##   [2393]         1      283635
##   [2394]         1      283699
```

```
##   [2395]         1      283764
##   [2396]         1      284195
##   [2397]         1      284446
##   -------
##   queryLength: 1 / subjectLength: 284458
```

```
df17 <- mergeByOverlaps(query = tiles.subset, subject = segs.gr, type = "any")
df17
```

```
## DataFrame with 2397 rows and 5 columns
##              tiles.subset             segs.gr                      Sample
##                 <GRanges>           <GRanges>                 <character>
## 1    17:37500001-38000000   17:987221-73296953 TCGA-3C-AAAU-10A-01D-A41E-01
## 2    17:37500001-38000000 17:25270517-73296953 TCGA-3C-AAAU-01A-11D-A41E-01
## 3    17:37500001-38000000   17:987221-80917016 TCGA-3C-AALI-10A-01D-A41E-01
## 4    17:37500001-38000000 17:37341549-38148341 TCGA-3C-AALI-01A-11D-A41E-01
## 5    17:37500001-38000000   17:987221-65395076 TCGA-3C-AALJ-10A-01D-A41E-01
## ...                   ...                 ...                        ...
## 2393 17:37500001-38000000   17:987221-55644206 TCGA-XX-A89A-01A-11D-A36I-01
## 2394 17:37500001-38000000   17:987221-69039331 TCGA-Z7-A8R5-10A-01D-A41E-01
## 2395 17:37500001-38000000   17:987221-80917016 TCGA-Z7-A8R5-01A-42D-A41E-01
## 2396 17:37500001-38000000 17:37107219-45033976 TCGA-Z7-A8R6-10A-01D-A41E-01
## 2397 17:37500001-38000000 17:31804984-55880090 TCGA-Z7-A8R6-01A-11D-A41E-01
##      Num_Probes Segment_Mean
##       <integer>    <numeric>
## 1         33859       0.0088
## 2         24226       0.1856
## 3         36977       0.0057
## 4           369       2.4574
## 5         29192       0.0067
## ...         ...          ...
## 2393      24174       -0.319
## 2394      31486       -7e-04
## 2395      36979       0.0018
## 2396       3118       -0.018
## 2397      11551      -0.0508
```

```
ind.segs.overlap.tile <- subjectHits(hits17)
segs.gr[ind.segs.overlap.tile]
```

```
## GRanges object with 2397 ranges and 3 metadata columns:
##          seqnames            ranges strand |                      Sample
##             <Rle>         <IRanges>  <Rle> |                 <character>
##      [1]       17   987221-73296953      * | TCGA-3C-AAAU-10A-01D-A41E-01
##      [2]       17 25270517-73296953      * | TCGA-3C-AAAU-01A-11D-A41E-01
##      [3]       17   987221-80917016      * | TCGA-3C-AALI-10A-01D-A41E-01
##      [4]       17 37341549-38148341      * | TCGA-3C-AALI-01A-11D-A41E-01
##      [5]       17   987221-65395076      * | TCGA-3C-AALJ-10A-01D-A41E-01
##      ...      ...               ...    ... .                        ...
##   [2393]       17   987221-55644206      * | TCGA-XX-A89A-01A-11D-A36I-01
##   [2394]       17   987221-69039331      * | TCGA-Z7-A8R5-10A-01D-A41E-01
##   [2395]       17   987221-80917016      * | TCGA-Z7-A8R5-01A-42D-A41E-01
##   [2396]       17 37107219-45033976      * | TCGA-Z7-A8R6-10A-01D-A41E-01
```

```
##   [2397]       17 31804984-55880090      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##          Num_Probes Segment_Mean
##           <integer>    <numeric>
##      [1]      33859       0.0088
##      [2]      24226       0.1856
##      [3]      36977       0.0057
##      [4]        369       2.4574
##      [5]      29192       0.0067
##      ...        ...          ...
##   [2393]      24174       -0.319
##   [2394]      31486       -7e-04
##   [2395]      36979       0.0018
##   [2396]       3118       -0.018
##   [2397]      11551      -0.0508
##   -------
##   seqinfo: 24 sequences from hg19 genome
```

```
segs.tile.means <- segs.gr[ind.segs.overlap.tile]$Segment_Mean
mean(segs.tile.means)
```

```
## [1] 0.1747237
```

c. Find the patient sample distribution of copy number for PIK3CA (hg19).

Find the counts of samples with deletion (D; `Segment_Mean < -0.3`), neutral (N; `Segment_Mean >= -0.3 & Segment_Mean <= 0.3`), gain (G; `Segment_Mean > 0.3`) segments that have `any` overlap with `PIK3CA` gene coordinates.

```
q3 <- GRanges(seqnames = "3", ranges = IRanges(start = 178866311, end = 178952497))
tiles.subset <- subsetByOverlaps(x = tiles, ranges = q3)
tiles.subset
```

```
## GRanges object with 1 range and 0 metadata columns:
##       seqnames              ranges strand
##          <Rle>           <IRanges>  <Rle>
##   [1]        3 178500001-179000000      *
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

```
hits3 <- findOverlaps(query = tiles.subset, subject = segs.gr, type = "any")
class(hits3)
```

```
## [1] "SortedByQueryHits"
## attr(,"package")
## [1] "S4Vectors"
```

```
hits3
```

```
## Hits object with 2258 hits and 0 metadata columns:
##          queryHits subjectHits
##          <integer>   <integer>
```

```
##      [1]         1          19
##      [2]         1         116
##      [3]         1         395
##      [4]         1         511
##      [5]         1         800
##      ...       ...         ...
##   [2254]         1      283568
##   [2255]         1      283679
##   [2256]         1      283735
##   [2257]         1      283862
##   [2258]         1      284320
##   -------
##   queryLength: 1 / subjectLength: 284458
```

```
df3 <- mergeByOverlaps(query = tiles.subset, subject = segs.gr, type = "any")
df3
```

```
## DataFrame with 2258 rows and 5 columns
##               tiles.subset               segs.gr                      Sample
##                  <GRanges>             <GRanges>                 <character>
## 1    3:178500001-179000000  3:89269219-197538677 TCGA-3C-AAAU-10A-01D-A41E-01
## 2    3:178500001-179000000  3:98972306-197538677 TCGA-3C-AAAU-01A-11D-A41E-01
## 3    3:178500001-179000000  3:88436220-197538677 TCGA-3C-AALI-10A-01D-A41E-01
## 4    3:178500001-179000000  3:93734671-197538677 TCGA-3C-AALI-01A-11D-A41E-01
## 5    3:178500001-179000000 3:161841018-181064228 TCGA-3C-AALJ-10A-01D-A41E-01
## ...                    ...                   ...                         ...
## 2254 3:178500001-179000000  3:71281058-197538677 TCGA-XX-A89A-01A-11D-A36I-01
## 2255 3:178500001-179000000   3:2212571-197538677 TCGA-Z7-A8R5-10A-01D-A41E-01
## 2256 3:178500001-179000000   3:2212571-197538677 TCGA-Z7-A8R5-01A-42D-A41E-01
## 2257 3:178500001-179000000 3:177896846-197538677 TCGA-Z7-A8R6-10A-01D-A41E-01
## 2258 3:178500001-179000000  3:88139747-197538677 TCGA-Z7-A8R6-01A-11D-A41E-01
##      Num_Probes Segment_Mean
##       <integer>    <numeric>
## 1         55921      -0.0034
## 2         52806       0.3057
## 3         56401       -0.003
## 4         55460        0.344
## 5          9398       0.0054
## ...         ...          ...
## 2254      65482      -0.0974
## 2255     106379       0.0012
## 2256     106367      -0.1237
## 2257      10203       0.0179
## 2258      56566      -0.0856
```

```
D <- df3[df3$Segment_Mean < -0.3, ]
D
```

```
## DataFrame with 23 rows and 5 columns
##               tiles.subset               segs.gr                      Sample
##                  <GRanges>             <GRanges>                 <character>
## 1   3:178500001-179000000 3:178929543-178929557 TCGA-A2-A0D0-01A-11D-A011-01
## 2   3:178500001-179000000 3:178502029-178876477 TCGA-A2-A0D1-01A-11D-A036-01
```

```
## 3   3:178500001-179000000 3:122835101-197538677 TCGA-A7-A4SF-01A-11D-A25N-01
## 4   3:178500001-179000000 3:178805167-178842228 TCGA-A7-A5ZV-01A-11D-A28A-01
## 5   3:178500001-179000000 3:176408675-178615602 TCGA-A8-A06U-01A-11D-A011-01
## ...                   ...                   ...                           ...
## 19  3:178500001-179000000 3:113316863-197538677 TCGA-E2-A576-01A-11D-A31T-01
## 20  3:178500001-179000000 3:175760560-181246458 TCGA-E9-A1RE-01A-11D-A160-01
## 21  3:178500001-179000000 3:102036048-197538677 TCGA-E9-A247-01A-11D-A166-01
## 22  3:178500001-179000000 3:178748480-178748514 TCGA-EW-A1P4-01A-21D-A141-01
## 23  3:178500001-179000000 3:178928076-178928091 TCGA-EW-A1P5-01A-11D-A141-01
##     Num_Probes Segment_Mean
##      <integer>    <numeric>
## 1            2      -0.5721
## 2          168       -0.605
## 3        38915      -0.3288
## 4           16      -0.4366
## 5         1108      -0.6375
## ...        ...          ...
## 19       44507      -0.3346
## 20        2937      -0.4681
## 21       50929      -0.3507
## 22           2      -1.2249
## 23           2      -1.4355
```

```
N <- df3[df3$Segment_Mean >= -0.3 & df3$Segment_Mean <= 0.3,]
N
```

```
## DataFrame with 2039 rows and 5 columns
##               tiles.subset               segs.gr                      Sample
##                  <GRanges>             <GRanges>                 <character>
## 1    3:178500001-179000000  3:89269219-197538677 TCGA-3C-AAAU-10A-01D-A41E-01
## 2    3:178500001-179000000  3:88436220-197538677 TCGA-3C-AALI-10A-01D-A41E-01
## 3    3:178500001-179000000 3:161841018-181064228 TCGA-3C-AALJ-10A-01D-A41E-01
## 4    3:178500001-179000000 3:155525798-195032856 TCGA-3C-AALJ-01A-31D-A41E-01
## 5    3:178500001-179000000  3:30071445-197538677 TCGA-3C-AALK-10A-01D-A41E-01
## ...                    ...                   ...                          ...
## 2035 3:178500001-179000000  3:71281058-197538677 TCGA-XX-A89A-01A-11D-A36I-01
## 2036 3:178500001-179000000   3:2212571-197538677 TCGA-Z7-A8R5-10A-01D-A41E-01
## 2037 3:178500001-179000000   3:2212571-197538677 TCGA-Z7-A8R5-01A-42D-A41E-01
## 2038 3:178500001-179000000 3:177896846-197538677 TCGA-Z7-A8R6-10A-01D-A41E-01
## 2039 3:178500001-179000000  3:88139747-197538677 TCGA-Z7-A8R6-01A-11D-A41E-01
##      Num_Probes Segment_Mean
##       <integer>    <numeric>
## 1         55921      -0.0034
## 2         56401       -0.003
## 3          9398       0.0054
## 4         20796      -0.0621
## 5         89938       0.0022
## ...         ...          ...
## 2035      65482      -0.0974
## 2036     106379       0.0012
## 2037     106367      -0.1237
## 2038      10203       0.0179
## 2039      56566      -0.0856
```

```
G <- df3[df3$Segment_Mean > 0.3, ]
G
```

```
## DataFrame with 196 rows and 5 columns
##              tiles.subset              segs.gr                      Sample
##                 <GRanges>            <GRanges>                 <character>
## 1   3:178500001-179000000  3:98972306-197538677 TCGA-3C-AAAU-01A-11D-A41E-01
## 2   3:178500001-179000000  3:93734671-197538677 TCGA-3C-AALI-01A-11D-A41E-01
## 3   3:178500001-179000000 3:166723780-197538677 TCGA-A1-A0SH-01A-11D-A087-01
## 4   3:178500001-179000000  3:93736226-197538677 TCGA-A1-A0SI-01A-11D-A141-01
## 5   3:178500001-179000000 3:174719405-197538677 TCGA-A1-A0SO-01A-22D-A087-01
## ...                   ...                  ...                         ...
## 192 3:178500001-179000000 3:157967774-178781257 TCGA-OL-A5S0-01A-11D-A28A-01
## 193 3:178500001-179000000 3:176404198-189090451 TCGA-PE-A5DD-01A-12D-A27O-01
## 194 3:178500001-179000000 3:177058147-194563572 TCGA-PL-A8LZ-01A-31D-A36I-01
## 195 3:178500001-179000000 3:128433396-187925892 TCGA-S3-AA0Z-01A-11D-A41E-01
## 196 3:178500001-179000000 3:153473686-197538677 TCGA-WT-AB41-01A-11D-A41E-01
##     Num_Probes Segment_Mean
##      <integer>    <numeric>
## 1        52806       0.3057
## 2        55460        0.344
## 3        15936       0.5227
## 4        55440       0.3967
## 5        11588       0.4276
## ...        ...          ...
## 192      10099       0.3819
## 193       6905       0.4713
## 194      10030       0.6824
## 195      31022       0.6937
## 196      22274       0.3783
```

```
cat("Number of patient samples with deletions:", nrow(D), "\n")
```

```
## Number of patient samples with deletions: 23
```

```
cat("Number of patient samples with neutral CN:", nrow(N), "\n")
```

```
## Number of patient samples with neutral CN: 2039
```

```
cat("Number of patient samples with gains:", nrow(G))
```

```
## Number of patient samples with gains: 196
```

2: Frequency of copy number alteration events within genomic regions.

Use the copy number data stored in `segs.gr`.

a. Create a genome-wide tile of 1Mb windows for the human genome (hg19).

```
tileWidth <- 1000000 # tile size of 1Mb
#divide the genome into tiles/windows/bins
tiles <- tileGenome(seqlengths = seqlengths(seqinfo3), tilewidth = tileWidth,
                    cut.last.tile.in.chrom = TRUE)
tiles
```

```
## GRanges object with 3113 ranges and 0 metadata columns:
##          seqnames            ranges strand
##             <Rle>         <IRanges>  <Rle>
##      [1]        1         1-1000000      *
##      [2]        1   1000001-2000000      *
##      [3]        1   2000001-3000000      *
##      [4]        1   3000001-4000000      *
##      [5]        1   4000001-5000000      *
##      ...      ...               ...    ...
##   [3109]        Y 55000001-56000000      *
##   [3110]        Y 56000001-57000000      *
##   [3111]        Y 57000001-58000000      *
##   [3112]        Y 58000001-59000000      *
##   [3113]        Y 59000001-59373566      *
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

b. Find the 1Mb window with the most frequent overlapping deletions.

Find the 1Mb windows with `any` overlap with deletion copy number segments. Assume a deletion segment is defined as a segment in `segs.gr` having `Segment_Mean < -0.3`.

Return one of the 1Mb window `Granges` entry with the highest frequency (count) of deletion segments.

```
# deletion copy number segments
segs.gr.del <- segs.gr[segs.gr$Segment_Mean < -0.3, ]
segs.gr.del
```

```
## GRanges object with 78923 ranges and 3 metadata columns:
##           seqnames              ranges strand |                      Sample
##              <Rle>           <IRanges>  <Rle> |                 <character>
##       [1]        1   95676511-95676518      * | TCGA-3C-AAAU-10A-01D-A41E-01
##       [2]        1 167057495-167059336      * | TCGA-3C-AAAU-10A-01D-A41E-01
##       [3]        1 181603120-181609567      * | TCGA-3C-AAAU-10A-01D-A41E-01
##       [4]        1 201474400-201474544      * | TCGA-3C-AAAU-10A-01D-A41E-01
##       [5]        2   51517041-51524666      * | TCGA-3C-AAAU-10A-01D-A41E-01
##       ...      ...                 ...    ... .                         ...
##   [78919]       17   17970035-18860763      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [78920]       17   31234127-31804569      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [78921]       17   55882611-56164541      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [78922]       18   61809405-61809922      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##   [78923]       18   68337362-68343590      * | TCGA-Z7-A8R6-01A-11D-A41E-01
##           Num_Probes Segment_Mean
##            <integer>    <numeric>
```

```
##       [1]          2      -1.6636
##       [2]          3      -1.0999
##       [3]          6      -1.2009
##       [4]          2      -1.4235
##       [5]         11      -1.1753
##       ...        ...          ...
##   [78919]        300      -0.7827
##   [78920]        565      -0.8084
##   [78921]        161      -0.7452
##   [78922]          2      -2.0272
##   [78923]          9      -0.7669
##   -------
##   seqinfo: 24 sequences from hg19 genome
```

```
#how many segments overlap each 1Mb genomic tile
counts <- countOverlaps(query = tiles, subject = segs.gr.del, type = "any")
tiles$counts.overlapAny <- counts
tiles
```

```
## GRanges object with 3113 ranges and 1 metadata column:
##          seqnames            ranges strand | counts.overlapAny
##             <Rle>         <IRanges>  <Rle> |         <integer>
##      [1]        1         1-1000000      * |                 0
##      [2]        1   1000001-2000000      * |                 0
##      [3]        1   2000001-3000000      * |                 0
##      [4]        1   3000001-4000000      * |               239
##      [5]        1   4000001-5000000      * |               274
##      ...      ...               ...    ... .               ...
##   [3109]        Y 55000001-56000000      * |                 0
##   [3110]        Y 56000001-57000000      * |                 0
##   [3111]        Y 57000001-58000000      * |                 0
##   [3112]        Y 58000001-59000000      * |                 0
##   [3113]        Y 59000001-59373566      * |                 0
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

```
# genomic tiles with highest overlap frequency (count) with deletion segments
tiles[tiles$counts.overlapAny == max(tiles$counts.overlapAny), ]
```

```
## GRanges object with 2 ranges and 1 metadata column:
##       seqnames            ranges strand | counts.overlapAny
##          <Rle>         <IRanges>  <Rle> |         <integer>
##   [1]       16 79000001-80000000      * |               620
##   [2]       16 83000001-84000000      * |               620
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

c. Visually inspect the deletion overlap result from part (b) using the Integrative Genome Viewer (IGV).

Screen shot of IGV at the 1Mb window with the most frequent overlap with deletion segments. The image includes the segments from `BRCA.genome_wide_snp_6_broad_Level_3_scna.seg` loaded.

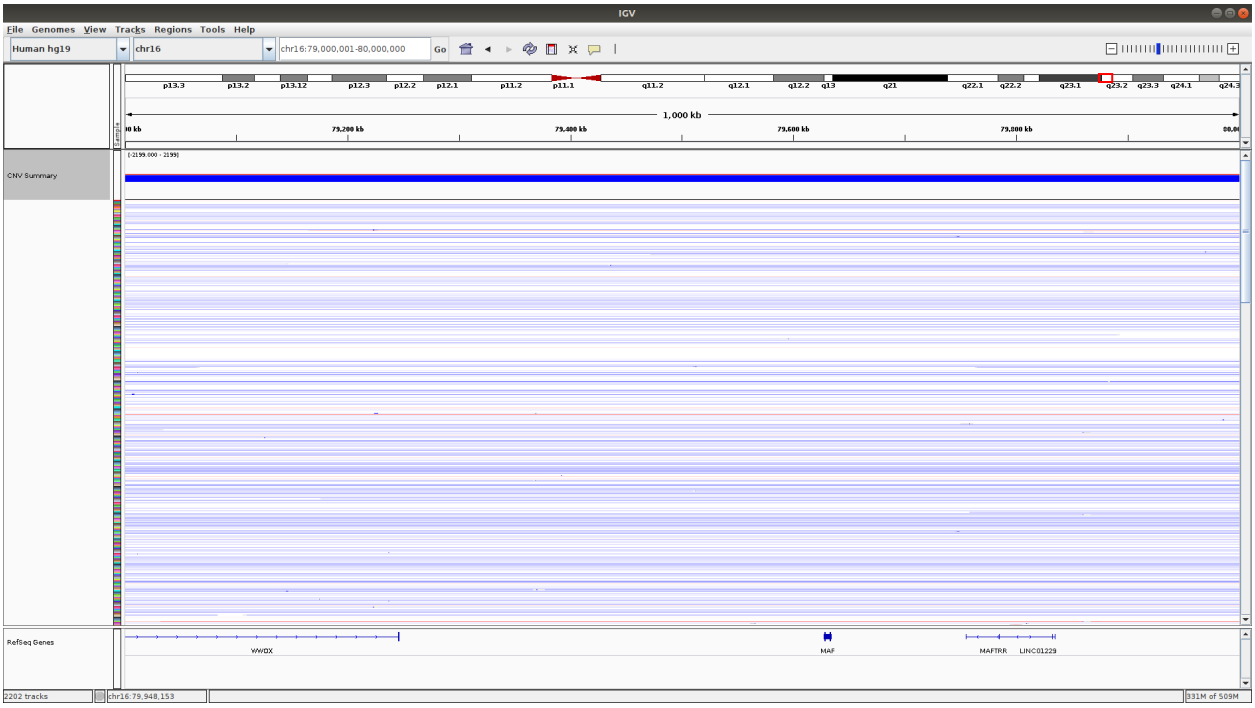

Figure 1: IGV snapshot of the first 1 MB window (chr16:79,000,001-80,000,000) with most frequent overlap with deletion segments

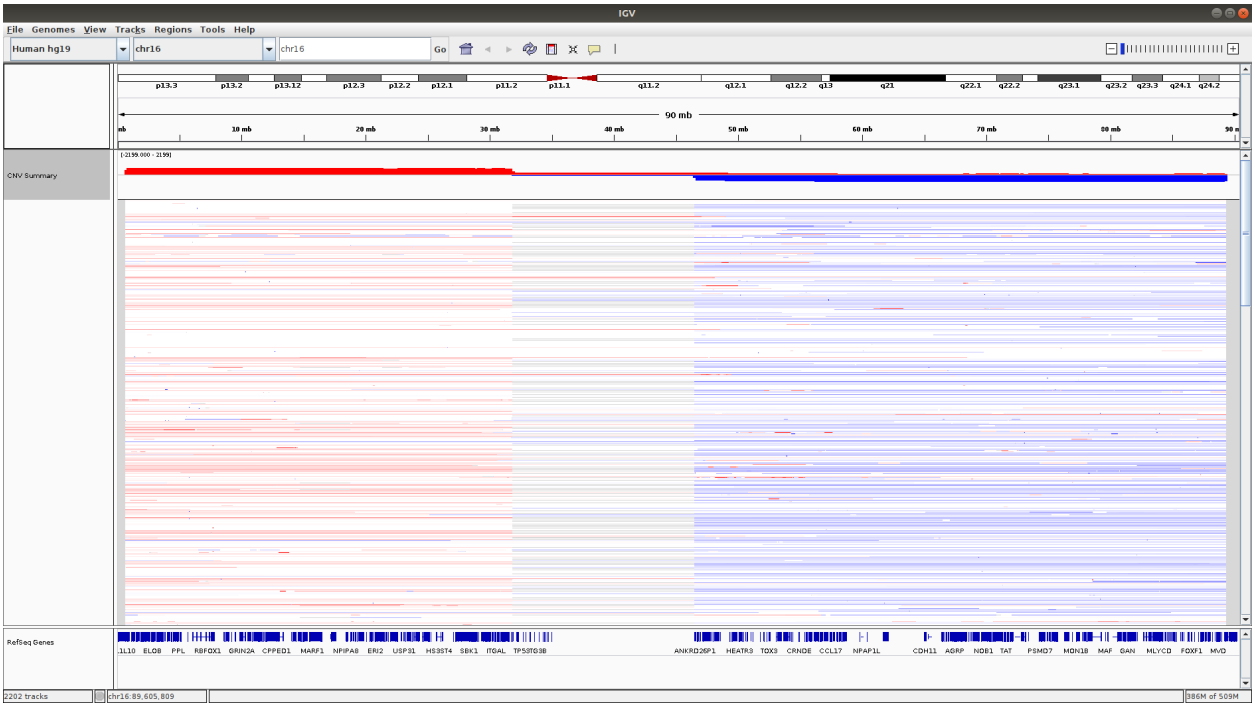

Figure 2: IGV snapshot of the chromosome 16 containing the two 1 MB windows (chr16:79,000,001-80,000,000 and chr16:83,000,001-84,000,000) where most frequent overlaps with deletion segments events occur

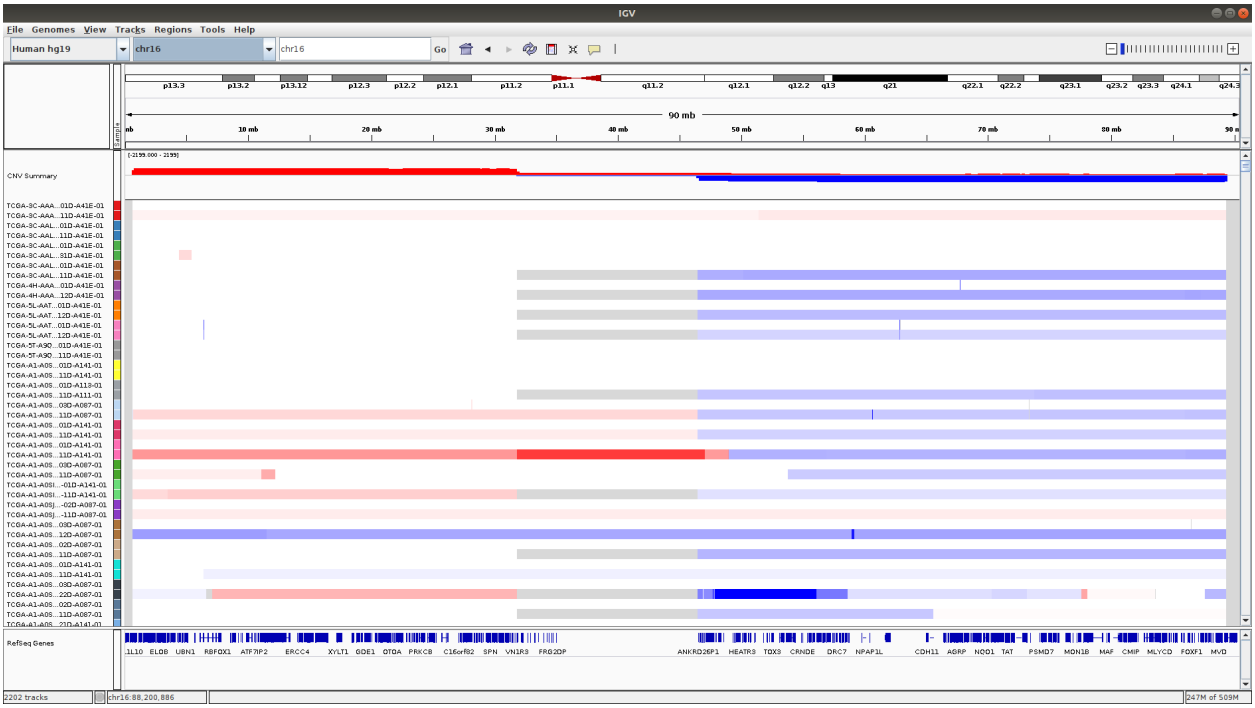

Figure 3: IGV snapshot of the chromosome 16 containing the two 1 MB windows (chr16:79,000,001-80,000,000 and chr16:83,000,001-84,000,000) where most frequent overlaps with deletion segments events occur - zoomed in, showing the sample tracks

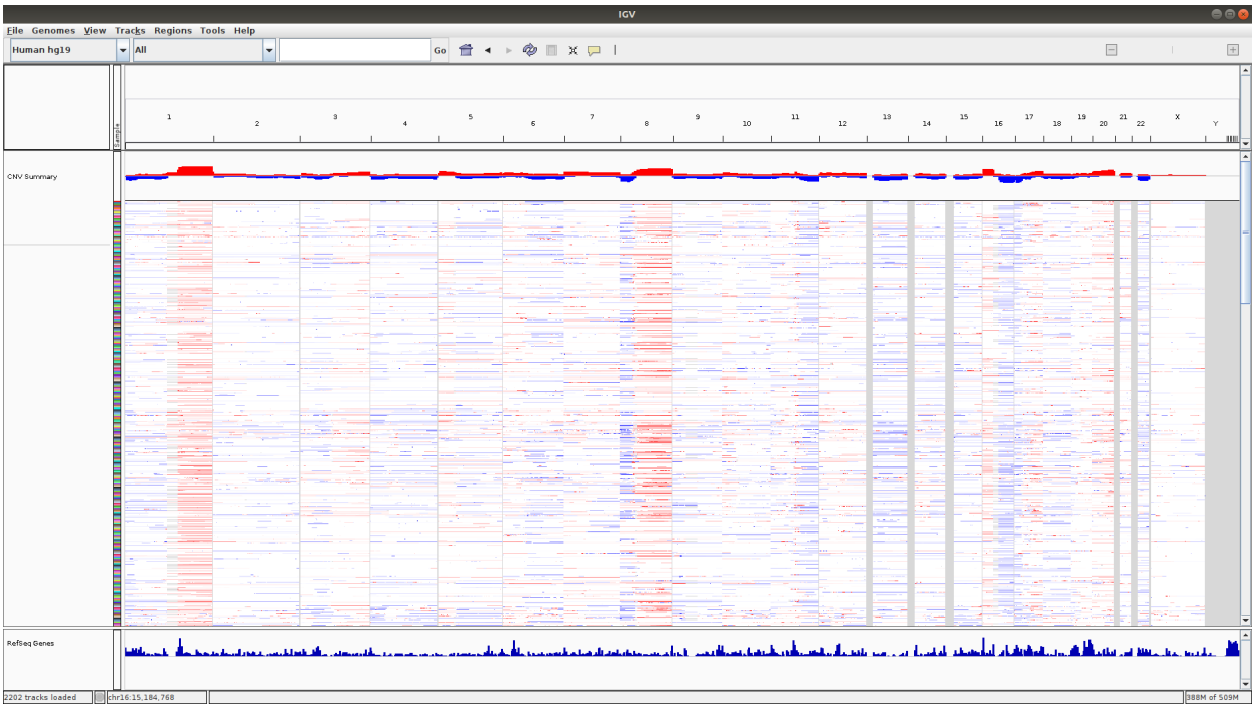

Figure 4: IGV snapshot whole genome

3: Reading and extracting sequencing data

Binary Alignment/Map Format (BAM) http://samtools.github.io/hts-specs/SAMv1.pdf specification

Preparation

```
library(Rsamtools)
bamFile <- "BRCA_IDC_cfDNA.bam"
```

a. Retrieve reads in the BAM file `BRCA_IDC_cfDNA.bam` at `chr12:25,380,269-25,380,308`.

Use the same settings for `scanBamWhat()`, `scanBamFlag()`, and `ScanBamParam()` as in Section 1 of `Lecture16_Rsamtools.Rmd`.

The BAM file is the primary input for Rsamtools. There are two initial steps:

1. Define the genomic coordinates and components to query (`ScanBamParam`)
2. Scan the BAM file (`scanBam`)

1. Setup parameters for scanning BAM file

- Specify the genomic location of interest to query in the BAM file.
- Specify which fields to return in the query.
- Specify the filters to use to include or exclude reads.
- Instantiate parameter object used in scanning the BAM file.

```
# Specify the genomic location of interest to query in the BAM file.
whichRanges <- GRanges(seqnames = "12",
                       IRanges(start = 25380269, end = 25380308))
whichRanges
```

```
## GRanges object with 1 range and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]       12 25380269-25380308      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

```
# Specify which fields to return in the query.
whatFields <- scanBamWhat()
whatFields
```

```
##  [1] "qname"       "flag"        "rname"       "strand"      "pos"
##  [6] "qwidth"      "mapq"        "cigar"       "mrnm"        "mpos"
## [11] "isize"       "seq"         "qual"        "groupid"     "mate_status"
```

```
# Specify the filters to use to include or exclude reads.
# First, specify the status of the reads based on the 'FLAG'
flag <- scanBamFlag(isDuplicate = FALSE)
flag
```

```
## keep0 keep1
##  4095  3071
```

```
# Next, specify additional filters to use including `mapqFilter`, `tagFilter`
mapqFilter = 30  # specifies the minimum mapping quality to include
tagFilter = c("RG")  # A character vector naming tags to be extracted.
#A tag is an optional field, with arbitrary information, stored with each record; RG - Read Group

# Instantiate parameter object influencing what fields and which records
# are imported from a (binary alignment) BAM file
param <- ScanBamParam(which = whichRanges, what = whatFields,
            mapqFilter = mapqFilter, tag = tagFilter)
param
```

```
## class: ScanBamParam
## bamFlag (NA unless specified):
## bamSimpleCigar: FALSE
## bamReverseComplement: FALSE
## bamTag: RG
## bamTagFilter:
## bamWhich: 1 ranges
## bamWhat: qname, flag, rname, strand, pos, qwidth, mapq, cigar, mrnm,
##   mpos, isize, seq, qual, groupid, mate_status
## bamMapqFilter: 30
```

2. Query the BAM file

```
bam <- scanBam(bamFile, param = param)
bam
```

```
## $`12:25380269-25380308`
## $`12:25380269-25380308`$qname
## [1] "33451983"
##
## $`12:25380269-25380308`$flag
## [1] 83
##
## $`12:25380269-25380308`$rname
## [1] 12
## 86 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y ... hs37d5
##
## $`12:25380269-25380308`$strand
## [1] -
## Levels: + - *
##
## $`12:25380269-25380308`$pos
## [1] 25380272
##
## $`12:25380269-25380308`$qwidth
## [1] 39
##
```

```
## $`12:25380269-25380308`$mapq
## [1] 60
##
## $`12:25380269-25380308`$cigar
## [1] "39M"
##
## $`12:25380269-25380308`$mrnm
## [1] 12
## 86 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y ... hs37d5
##
## $`12:25380269-25380308`$mpos
## [1] 25380159
##
## $`12:25380269-25380308`$isize
## [1] -152
##
## $`12:25380269-25380308`$seq
##   A DNAStringSet instance of length 1
##     width seq
## [1]    39 CTCTTGACCTGCTGTGTCGAGAATATCCAAGAGACAGGT
##
## $`12:25380269-25380308`$qual
##   A PhredQuality instance of length 1
##     width seq
## [1]    39 FFFFFFFFFFFFFFF<FFFFFFFFFFFFAFAFFFFAAAA<
##
## $`12:25380269-25380308`$tag
## $`12:25380269-25380308`$tag$RG
## [1] "P12.17.7_Breast"
```

This returns a `list` object with each element representing a read. For each element/read, there is another `list` with the fields in the BAM file requested with `scanBamWhat()`. Below is a breakdown of what is in the first read.

```
bam[[1]]$qname # read query template name
```

```
## [1] "33451983"
```

```
bam[[1]]$flag  # bitwise flag describing the read alignment
```

```
## [1] 83
```

```
bam[[1]]$rname # reference sequence name (i.e chr12 or 12)
```

```
## [1] 12
## 86 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y ... hs37d5
```

```
bam[[1]]$pos   # position of aligned read (leftmost coordinate)
```

```
## [1] 25380272
```

```
bam[[1]]$mapq  # mapping quality of the read alignment
```

```
## [1] 60
```

```
bam[[1]]$cigar # CIGAR string: code string to describe read alignment sequence match to reference
```

```
## [1] "39M"
```

```
bam[[1]]$mrnm  # mate read's reference sequence name
```

```
## [1] 12
## 86 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y ... hs37d5
```

```
bam[[1]]$mpos  # mate read's aligned position
```

```
## [1] 25380159
```

```
bam[[1]]$isize # insert size or template length; aka fragment size
```

```
## [1] -152
```

```
as.character(bam[[1]]$seq)   # sequence of mapped reads on forward strand
```

```
## [1] "CTCTTGACCTGCTGTGTCGAGAATATCCAAGAGACAGGT"
```

```
as.character(bam[[1]]$qual)  # base qualities of the sequence alignment
```

```
## [1] "FFFFFFFFFFFFFFF<FFFFFFFFFFFFFAFAFFFFAAAA<"
```

```
bam[[1]]$tag   # value for the tag we specified
```

```
## $RG
## [1] "P12.17.7_Breast"
```

b. What is the fraction of G+C bases in the mapped read sequence?

Count the number of G and C bases in the read sequence.

```
#mapped read sequence
dna.seq <- as.character(unlist(bam[[1]]$seq))
dna.seq
```

```
## [1] "CTCTTGACCTGCTGTGTCGAGAATATCCAAGAGACAGGT"
```

```
# GC fraction
gc.fraction <- sum(unlist(strsplit(dna.seq, split = "")) %in% c("C", "G")) /  nchar(dna.seq)
gc.fraction
```

```
## [1] 0.4871795
```

```
# GC fraction - alternative implementation
gc.fraction <- lengths(gregexpr("C|G", dna.seq)) / nchar(dna.seq)
gc.fraction
```

```
## [1] 0.4871795
```

```
# GC fraction - alternative implementation using str_count from stringr package
# library(stringr)
gc.fraction <- str_count(dna.seq, "C|G") / nchar(dna.seq)
gc.fraction
```

```
## [1] 0.4871795
```

4: Reading and annotating genomic variants

Variant Call Format (VCF) http://samtools.github.io/hts-specs/VCFv4.2.pdf specification

Preparation

```
library(VariantAnnotation)
vcfFile <- "GIAB_highconf_v.3.3.2.vcf.gz"
```

a. Load variant data from VCF file `GIAB_highconf_v.3.3.2.vcf.gz` for `chr8:128,700,000-129,000,000`.

Use genome build `hg19`.

```
vcfHead <- scanVcfHeader(vcfFile)
q <- GRanges(seqnames = "8", ranges = IRanges(start = 128700000, end = 129000000))
vcf.param <- ScanVcfParam(which = q) # single 500kb bin
vcf <- readVcf(vcfFile, genome = "hg19", param = vcf.param)
vcf
```

```
## class: CollapsedVCF
## dim: 308 1
## rowRanges(vcf):
##   GRanges with 5 metadata columns: paramRangeID, REF, ALT, QUAL, FILTER
## info(vcf):
##   DFrame with 16 columns: DPSum, platforms, platformnames, platformbias, ...
## info(header(vcf)):
```

```
##                                   Number Type    Description
##    DPSum                          1      Integer Total read depth summed ac...
##    platforms                      1      Integer Number of different platfo...
##    platformnames                  .      String  Names of platforms for whi...
##    platformbias                   .      String  Names of platforms that ha...
##    datasets                       1      Integer Number of different datase...
##    datasetnames                   .      String  Names of datasets for whic...
##    datasetsmissingcall            .      String  Names of datasets that are...
##    callsets                       1      Integer Number of different callse...
##    callsetnames                   .      String  Names of callsets that cal...
##    varType                        1      String  Type of variant
##    filt                           .      String  List of callsets that had ...
##    callable                       .      String  List of callsets that had ...
##    difficultregion                .      String  List of difficult region b...
##    arbitrated                     1      String  TRUE if callsets had disco...
##    callsetwiththisuniqgenopassing .      String  Callset that uniquely call...
##    callsetwithotheruniqgenopassing .      String  Callset that uniquely call...
## geno(vcf):
##   SimpleList of length 8: GT, DP, GQ, ADALL, AD, IGT, IPS, PS
## geno(header(vcf)):
##          Number Type    Description
##    GT    1      String  Consensus Genotype across all datasets with called g...
##    DP    1      Integer Total read depth summed across all datasets, excludi...
##    GQ    1      Integer Net Genotype quality across all datasets, calculated...
##    ADALL R      Integer Net allele depths across all datasets
##    AD    R      Integer Net allele depths across all unfiltered datasets wit...
##    IGT   1      String  Original input genotype
##    IPS   1      String  Phase set for IGT
##    PS    1      String  Phase set for GT
```

The `vcf` variable is of class `CollapsedVCF` and contains header information and data. The information that has been parsed by `readVcf` is explained below.

The `rowRanges` function returns a `GRanges` object containing the coordinates, REF/ALT bases, quality, and filtering status of the variants.

```
genoRanges <- rowRanges(vcf)
genoRanges
```

```
## GRanges object with 308 ranges and 5 metadata columns:
##                                  seqnames              ranges strand |
##                                     <Rle>           <IRanges>  <Rle> |
##                       rs6984323         8           128706908      * |
##                       rs4478537         8           128708943      * |
##                      rs34141920         8 128710237-128710239      * |
##                      rs17772814         8           128711742      * |
##                      rs77977256         8           128713029      * |
##                             ...       ...                 ...    ... .
##                      rs10808563         8           128996845      * |
##                      rs71300287         8 128997083-128997091      * |
##               8:128997155_CTT/C         8 128997155-128997157      * |
##   8:128997161_TTCTTTCTCTTTCTC/T         8 128997161-128997175      * |
##                       rs2392884         8           128999174      * |
##                                  paramRangeID             REF                 ALT
```

```
##                                    <factor>  <DNAStringSet> <DNAStringSetList>
##                        rs6984323         <NA>               C                   T
##                        rs4478537         <NA>               T                   A
##                       rs34141920         <NA>             GCA                   G
##                       rs17772814         <NA>               G                   A
##                       rs77977256         <NA>               A                   G
##                              ...          ...             ...                 ...
##                       rs10808563         <NA>               C                   T
##                       rs71300287         <NA>       TTTTCTTTC                   T
##                 8:128997155_CTT/C         <NA>             CTT                   C
##   8:128997161_TTCTTTCTCTTTCTC/T         <NA> TTCTTTCTCTTTCTC                   T
##                        rs2392884         <NA>               A                   C
##                                       QUAL      FILTER
##                                  <numeric> <character>
##                        rs6984323        50        PASS
##                        rs4478537        50        PASS
##                       rs34141920        50        PASS
##                       rs17772814        50        PASS
##                       rs77977256        50        PASS
##                              ...       ...         ...
##                       rs10808563        50        PASS
##                       rs71300287        50        PASS
##                 8:128997155_CTT/C        50        PASS
##   8:128997161_TTCTTTCTCTTTCTC/T        50        PASS
##                        rs2392884        50        PASS
##   -------
##   seqinfo: 25 sequences from hg19 genome
```

The `INFO` column in the original VCF text file contains a semi-colon delimited set of custom fields with flexible format that algorithms will output, here parsed into usable format. Fields are available from the header below

```
info(header(vcf)) # returns a DataFrame object
```

```
## DataFrame with 16 rows and 3 columns
##                                       Number        Type
##                                  <character> <character>
## DPSum                                     1     Integer
## platforms                                 1     Integer
## platformnames                             .      String
## platformbias                              .      String
## datasets                                  1     Integer
## ...                                     ...         ...
## callable                                  .      String
## difficultregion                           .      String
## arbitrated                                1      String
## callsetwiththisuniqgenopassing            .      String
## callsetwithotheruniqgenopassing           .      String
##
##
## DPSum
## platforms                                                                  Numb
## platformnames
```

```
## platformbias                    Names of platforms that have reads containing a variant at this loca
## datasets                                                                                        Numb
## ...
## callable
## difficultregion
## arbitrated
## callsetwiththisuniqgenopassing                                                            Callset
## callsetwithotheruniqgenopassing                                                  Callset that uni
```

The FORMAT column in the original VCF text file contains the format and description of the genotype fields.

```
geno(header(vcf))
```

```
## DataFrame with 8 rows and 3 columns
##            Number        Type
##       <character> <character>
## GT              1      String
## DP              1     Integer
## GQ              1     Integer
## ADALL           R     Integer
## AD              R     Integer
## IGT             1      String
## IPS             1      String
## PS              1      String
##
##
## GT                                                                                          Conse
## DP                                                                                       Total read
## GQ    Net Genotype quality across all datasets, calculated from GQ scores of callsets supporting the
## ADALL
## AD                                                                                  Net allele dep
## IGT
## IPS
## PS
```

b. Combine the fields of the VCF genotype information into a table.

```
genoData <- data.frame(geno(vcf)$GT,
                       geno(vcf)$DP,
                       geno(vcf)$GQ,
                       geno(vcf)$ADALL,
                       geno(vcf)$AD,
                       geno(vcf)$IGT,
                       geno(vcf)$IPS,
                       geno(vcf)$PS
                       )
colnames(genoData) <- rownames(geno(header(vcf)))
head(genoData, n=10)
```

```
##                   GT  DP   GQ    ADALL       AD IGT IPS     PS
## rs6984323        1|1 765  583   1, 332   0, 315 1/1   . PATMAT
```

```
## rs4478537        0|1 544  813 103, 124 135, 172 0/1   . PATMAT
## rs34141920       0|1 523  222 132, 121 132, 121 0/1   . PATMAT
## rs17772814       1|0 695 1503 143, 158 196, 199 0/1   . PATMAT
## rs77977256       1|0 642  685 154, 157 160, 166 0/1   . PATMAT
## 8:128715845_AT/A 0|1 368   99   66, 91   66, 91 0/1   . PATMAT
## rs143209301      1|0 581  595 128, 128 151, 165 0/1   . PATMAT
## rs202231913      0|1 369   99   81, 97   81, 97 0/1   . PATMAT
## rs16902340       0|1 689 1294 144, 150 184, 204 0/1   . PATMAT
## rs7841229        0|1 635 1010 180, 172 134, 130 0/1   . PATMAT
```

```
dim(genoData)
```

```
## [1] 308   8
```

c. Retrieve the following information at chr8:128747953.

Print out the SNP ID (i.e. "rs ID"), reference base (`REF`), alternate base (`ALT`), genotype (`GT`), depth (`DP`), allele depth (`ADALL`), phase set (`PS`).

```
variant <- genoRanges[ranges(genoRanges) == IRanges(start = 128747953, end=128747953)]
variant
```

```
## GRanges object with 1 range and 5 metadata columns:
##             seqnames    ranges strand | paramRangeID            REF
##                <Rle> <IRanges>  <Rle> |     <factor> <DNAStringSet>
##   rs3824120        8 128747953      * |         <NA>              G
##                            ALT      QUAL      FILTER
##             <DNAStringSetList> <numeric> <character>
##   rs3824120                  T        50        PASS
##   -------
##   seqinfo: 25 sequences from hg19 genome
```

```
snp.id <- names(variant)
ref.base <- as.character(variant$REF)
alt.base <- as.character(unlist(variant$ALT))

genoData[snp.id, ]
```

```
##            GT  DP  GQ   ADALL       AD IGT IPS     PS
## rs3824120 0|1 461 668 105, 94 128, 121 0/1   . PATMAT
```

```
genotype <- as.character(genoData[snp.id, ]$GT)
depth <- genoData[snp.id, ]$DP
allele.depth <- unlist(genoData[snp.id, ]$ADALL)
phase.set <- as.character(genoData[snp.id, ]$PS)

cat("SNP ID:", snp.id, "\n")
```

```
## SNP ID: rs3824120
```

```
cat("Reference base:", ref.base, "\n")
```

```
## Reference base: G
```

```
cat("Alternate base:", alt.base, "\n")
```

```
## Alternate base: T
```

```
cat("Genotype:", genotype, "\n")
```

```
## Genotype: 0|1
```

```
cat("Read depth:", depth, "\n")
```

```
## Read depth: 461
```

```
cat("Allele depth:", allele.depth, "\n")
```

```
## Allele depth: 105 94
```

```
cat("Phase set:", phase.set)
```

```
## Phase set: PATMAT
```

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.5 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
##  [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
##  [7] LC_PAPER=en_CA.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] VariantAnnotation_1.32.0          SummarizedExperiment_1.16.1
```

```
##  [3] DelayedArray_0.12.3              BiocParallel_1.20.1
##  [5] matrixStats_0.61.0               Biobase_2.46.0
##  [7] Rsamtools_2.2.3                  BSgenome.Hsapiens.UCSC.hg19_1.4.0
##  [9] BSgenome_1.54.0                  rtracklayer_1.46.0
## [11] Biostrings_2.54.0                XVector_0.26.0
## [13] forcats_0.5.1                    stringr_1.4.0
## [15] dplyr_1.0.8                      purrr_0.3.4
## [17] readr_2.1.2                      tidyr_1.2.0
## [19] tibble_3.1.6                     ggplot2_3.3.5
## [21] tidyverse_1.3.1                  GenomicRanges_1.38.0
## [23] GenomeInfoDb_1.22.1              IRanges_2.20.2
## [25] S4Vectors_0.24.4                 BiocGenerics_0.32.0
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-7             fs_1.5.2                 lubridate_1.8.0
##  [4] bit64_4.0.5              progress_1.2.2           httr_1.4.2
##  [7] tools_3.6.3              backports_1.4.1          utf8_1.2.2
## [10] R6_2.5.1                 DBI_1.1.2                colorspace_2.0-3
## [13] withr_2.4.3              prettyunits_1.1.1        tidyselect_1.1.2
## [16] curl_4.3.2               bit_4.0.4                compiler_3.6.3
## [19] cli_3.2.0                rvest_1.0.2              xml2_1.3.3
## [22] scales_1.1.1             askpass_1.1              rappdirs_0.3.3
## [25] digest_0.6.29            rmarkdown_2.11           pkgconfig_2.0.3
## [28] htmltools_0.5.2          dbplyr_2.1.1             fastmap_1.1.0
## [31] rlang_1.0.1              readxl_1.3.1             rstudioapi_0.13
## [34] RSQLite_2.2.10           generics_0.1.2           jsonlite_1.8.0
## [37] RCurl_1.98-1.6           magrittr_2.0.2           GenomeInfoDbData_1.2.2
## [40] Matrix_1.2-18            Rcpp_1.0.8               munsell_0.5.0
## [43] fansi_1.0.2              lifecycle_1.0.1          stringi_1.7.6
## [46] yaml_2.3.5               zlibbioc_1.32.0          BiocFileCache_1.10.2
## [49] blob_1.2.2               grid_3.6.3               crayon_1.5.0
## [52] lattice_0.20-41          haven_2.4.3              GenomicFeatures_1.38.2
## [55] hms_1.1.1                knitr_1.37               pillar_1.7.0
## [58] biomaRt_2.42.1           reprex_2.0.1             XML_3.99-0.3
## [61] glue_1.6.2               evaluate_0.15            modelr_0.1.8
## [64] vctrs_0.3.8              tzdb_0.2.0               cellranger_1.1.0
## [67] openssl_1.4.6            gtable_0.3.0             assertthat_0.2.1
## [70] cachem_1.0.6             xfun_0.29                broom_0.7.12
## [73] AnnotationDbi_1.48.0     memoise_2.0.1            GenomicAlignments_1.22.1
## [76] ellipsis_0.3.2
```