Hw3_Zheng

Zheng Cui

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

1. Removing the version number from an Ensembl ID

(50%) Write an R function cut.version.number(ensembl.id) using regular expression to extract the Ensembl ID without the attached version number. For example,given the Ensembl ID "ENST00000621592.8" of the MYC-206 transcript as input, the function must return "ENST00000621592".

The function should be vectorized, meaning that it can take a vector of Ensembl IDs as input. For example, if the input is c("ENST00000377970.6", "ENST00000259523.10"), the return of the function must be c("ENST00000377970", "ENST00000259523"). It should avoid any for-loop to maintain efficiency.

Removing the version number from an Ensembl ID

```
cut.version.number <- function(ensembl.ids) {
  # Use regular expression to remove the version number
  remove_version <- sub("\\.\\d+$", "", ensembl.ids)
  return(remove_version)
}

# Example usage:
ensembl_ids <- c("ENST00000621592.8", "ENST00000377970.6", "ENST00000259523.10")
remove_version <- cut.version.number(ensembl_ids)
print(remove_version)
```

```
[1] "ENST00000621592" "ENST00000377970" "ENST00000259523"
```

Apply to the attributer of the GFF file, filter the gene_id out, and then remove the version number, as follows

```
#  import  GFF3 table
library(stringr)
file <- "E:/Language/R/gencode.v44.primary_assembly.annotation.gff3"
GFF <- read.table(file, header = FALSE, sep = "\t")
colnames(GFF) <- c("seqid", "source", "type", "start", "end",
                   "score", "strand", "phase", "attributes")
rownames(GFF) <- 1:nrow(GFF)


pattern <- "gene_id=([^;]*)"
m <- regexec(pattern, GFF[, "attributes"])
gene_ID <- sapply(
  regmatches(GFF[, "attributes"], m),
  function(e) {return(e[2])})

# Add the extracted gene IDs to the GFF dataframe
GFF$gene_ID <- gene_ID
```

```r
# Apply the cut.version.number function to the gene_id column of GFF
GFF$trimmed_ids <- cut.version.number(GFF$gene_ID)
head(GFF)
```

```
  seqid source       type start   end score strand phase
1  chr1 HAVANA       gene 11869 14409     .      +     .
2  chr1 HAVANA transcript 11869 14409     .      +     .
3  chr1 HAVANA       exon 11869 12227     .      +     .
4  chr1 HAVANA       exon 12613 12721     .      +     .
5  chr1 HAVANA       exon 13221 14409     .      +     .
6  chr1 HAVANA       gene 12010 13670     .      +     .

1
2                                                    ID=ENST00000456328.2;Parent=ENSG00000290825
3 ID=exon:ENST00000456328.2:1;Parent=ENST00000456328.2;gene_id=ENSG00000290825.1;transcript_
4 ID=exon:ENST00000456328.2:2;Parent=ENST00000456328.2;gene_id=ENSG00000290825.1;transcript_
5 ID=exon:ENST00000456328.2:3;Parent=ENST00000456328.2;gene_id=ENSG00000290825.1;transcript_
6
            gene_ID     trimmed_ids
1 ENSG00000290825.1 ENSG00000290825
2 ENSG00000290825.1 ENSG00000290825
3 ENSG00000290825.1 ENSG00000290825
4 ENSG00000290825.1 ENSG00000290825
5 ENSG00000290825.1 ENSG00000290825
6 ENSG00000223972.6 ENSG00000223972
```

2. Reading genome annotation in GTF format

(15%) 2.1 The GTF format is as popular as the GFF3 format. Please describe three differences between the two formats.

```r
#  import  GFF3 table
library(stringr)
file <- "E:/Language/R/gencode.v44.annotation.gtf"
GTF <- read.table(file, header = FALSE, sep = "\t")
colnames(GTF) <- c("seqid", "source", "type", "start", "end",
                   "score", "strand", "phase", "attributes")
rownames(GTF) <- 1:nrow(GTF)
head(GTF)
```

```
  seqid source       type start   end score strand phase
```

```
1  chr1 HAVANA       gene 11869 14409     .      +     .
2  chr1 HAVANA transcript 11869 14409     .      +     .
3  chr1 HAVANA       exon 11869 12227     .      +     .
4  chr1 HAVANA       exon 12613 12721     .      +     .
5  chr1 HAVANA       exon 13221 14409     .      +     .
6  chr1 HAVANA       gene 12010 13670     .      +     .

1
2                                                gene_id ENSG00000290825.1; transcript_id ENST000
3 gene_id ENSG00000290825.1; transcript_id ENST00000456328.2; gene_type lncRNA; gene_name DD
4 gene_id ENSG00000290825.1; transcript_id ENST00000456328.2; gene_type lncRNA; gene_name DD
5 gene_id ENSG00000290825.1; transcript_id ENST00000456328.2; gene_type lncRNA; gene_name DD
6
```

Answer

After importing, we can see: 1, mainly the difference between attributes, GTF uses Spaces, while GFF3 equals the sign =, the attribute format in GFF3: ** Although GFF3 uses key-value pairs in attributes, the keys and values themselves can sometimes have additional structure. For example, some properties may include multiple values separated by commas, or they may use different key-value pairs to convey additional information. 2. GFF3 has an ID and parent ID in its properties, but GTF does not 3. GTF usually contains the required basic attributes (e.g., gene_id, transcript_id) as mandatory attributes, while GFF3 allows greater flexibility in attribute names. 4. Optional fields: GFF3 contains optional fields for additional information, while GTF is more concise and may lack some of these optional details. 5, the file size is roughly the same, the compressed file size is also roughly the same 6, the compressed file size is also roughly the same.

(35%) 2.2 Use regular expression to extract gene names for every feature in the file. Report the runtime of your code using the system.time() function.

```r
time.vec <- system.time({
pattern <- "gene_name ([^;]+);"

 m <- regexec(pattern, GTF[, "attributes"])
  gene_names <- sapply(
    regmatches(GTF[, "attributes"], m),
    function(e) {return(e[2])})
# Add the extracted gene names as a new column
GTF$gene_name <- gene_names
  #
```

```
  # Very slow alternative to extract gene names:
  #   d <- data.frame(regmatches(gff3[, "tag"], m))
  #   gene_names <- d[2, ]
  } )
print(time.vec["user.self"])
```

```
user.self
     8.39
```

```
head(GTF$gene_name)
```

```
[1] "DDX11L2" "DDX11L2" "DDX11L2" "DDX11L2" "DDX11L2" "DDX11L1"
```