Tutorial Human Genomeme Annotation

1. Introduction

1.1 What is gene annotation?

Over the past years, we have learnt that there are a number of chromosomes and genes in our genome. Counting the number of chromosomes is fairly easy but students might find difficult to say how many genes we have in our genome. If you can get an answer for this, could you tell how many genes encode protein and how many do not?

To answer this question, we need to access the database for gene annotation. Gene annotation is the process of making nucleotide sequence meaningful - where genes are located? whether it is protein-coding or noncoding. If you would like to get an overview of gene annotation, please find this link.

One of well-known collaborative efforts in gene annotation is the GENCODE consortium. It is a part of the Encyclopedia of DNA Elements (The ENCODE project consortium) and aims to identify all gene features in the human genome using a combination of computational analysis, manual annotation, and experimental validation (Harrow et al. 2012). You might find another database for gene annotation, like RefSeq, CCDS, and need to understand differences between them.

In this tutorial, we will access to gene annotation from the GENCODE consortium and explore genes and functional elements in our genome.

1.2 Aims

What we will do with this dataset: - Be familiar with gene annotation modality. - Tidy data and create a table for your analysis. - Apply tidyverse functions for data munging.

Please note that there is better solution for getting gene annotation in R if you use a biomart. Our tutorial is only designed to have a practice on tidyverse exercise.

2. Explore your data

2.1 Unboxing your dataset

This tutorial will use a gene annotation file from the GENCODE. You will need to download the file from the GENCODE. If you are using terminal, please download file using wget: Once you download the file, you can print out the first few lines using the following bash command (we will learn UNIX commands later):

The file is the GFT file format, which you will find most commonly in gene annotation. Please read the file format thoroughly in the link above.

For the tutorial, we need to load two packages. If the package is not installed in your system, please install it.

- tidyverse, a package you have learnt from the chapter 5.

- readr, a package provides a fast and friendly way to read. Since the file gencode.v31.basic.annotation.gtf.gz is pretty large, you will need some function to load data quickly into your workspace. readr in a part of tidyverse, so you can just load tidyverse to use readr functions.

Let's load the GTF file into your workspace. We will use read_delim function from the readr package. This is much faster loading than read.delim or read.csv from R base. However, please keep in mind that some parameters and output class for read_delim are slightly different from them.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
d = read_delim("gencode.v31.basic.annotation.gtf.gz",
               delim = '\t', skip = 5, progress = F,
               col_names = F)
```

```
## Rows: 1756502 Columns: 9
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (7): X1, X2, X3, X6, X7, X8, X9
## dbl (2): X4, X5
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Can you find out what the parameters mean? Few things to note are: - The GTF file contains the first few lines for comments (#). In general, the file contains description, provider, date, format. - The GTF file does not have column names so you will need to assign 'FALSE for col_names.

This is sort of canonical way to load your dataset into R. However, we are using a GTF format, which is specific to gene annotation so we can use a package to specifically handle a GTF file.

Here I introduce the package rtracklayer. Let's install the package first.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("rtracklayer")
```

```
## Bioconductor version 3.13 (BiocManager 1.30.16), R 4.1.0 (2021-05-18)
## Warning: package(s) not installed when version(s) same as current; use `force = TRUE` to
##   re-install: 'rtracklayer'
```

Then, now you can read the GTF file using this package. Then, you can check the class of the object d.

```
d = rtracklayer::import('gencode.v31.basic.annotation.gtf.gz')
class(d)
```

```
## [1] "GRanges"
## attr(,"package")
## [1] "GenomicRanges"
```

You will find out that this is GRanges class. This is from the package Genomic Range, specifically dealing with genomic datasets but we are not heading into this in this tutorial. So please find this information if you are serious on this.

We are converting d into a data frame as following:

```
d = d %>% as.data.frame()
```

Let's overview few lines from the data frame, and explore what you get in this object.

```
head(d)
```

```
##   seqnames start   end width strand source       type score phase
## 1     chr1 11869 14409  2541      + HAVANA       gene    NA    NA
## 2     chr1 11869 14409  2541      + HAVANA transcript    NA    NA
## 3     chr1 11869 12227   359      + HAVANA       exon    NA    NA
## 4     chr1 12613 12721   109      + HAVANA       exon    NA    NA
## 5     chr1 13221 14409  1189      + HAVANA       exon    NA    NA
## 6     chr1 12010 13670  1661      + HAVANA transcript    NA    NA
##             gene_id                          gene_type gene_name level
## 1 ENSG00000223972.5 transcribed_unprocessed_pseudogene   DDX11L1     2
## 2 ENSG00000223972.5 transcribed_unprocessed_pseudogene   DDX11L1     2
## 3 ENSG00000223972.5 transcribed_unprocessed_pseudogene   DDX11L1     2
## 4 ENSG00000223972.5 transcribed_unprocessed_pseudogene   DDX11L1     2
## 5 ENSG00000223972.5 transcribed_unprocessed_pseudogene   DDX11L1     2
## 6 ENSG00000223972.5 transcribed_unprocessed_pseudogene   DDX11L1     2
##      hgnc_id          havana_gene     transcript_id
## 1 HGNC:37102 OTTHUMG00000000961.2              <NA>
## 2 HGNC:37102 OTTHUMG00000000961.2 ENST00000456328.2
## 3 HGNC:37102 OTTHUMG00000000961.2 ENST00000456328.2
## 4 HGNC:37102 OTTHUMG00000000961.2 ENST00000456328.2
## 5 HGNC:37102 OTTHUMG00000000961.2 ENST00000456328.2
## 6 HGNC:37102 OTTHUMG00000000961.2 ENST00000450305.2
##                      transcript_type transcript_name transcript_support_level
## 1                               <NA>            <NA>                     <NA>
## 2                             lncRNA     DDX11L1-202                        1
## 3                             lncRNA     DDX11L1-202                        1
## 4                             lncRNA     DDX11L1-202                        1
## 5                             lncRNA     DDX11L1-202                        1
## 6 transcribed_unprocessed_pseudogene     DDX11L1-201                       NA
##    tag    havana_transcript exon_number           exon_id         ont
## 1  <NA>                 <NA>        <NA>              <NA>        <NA>
## 2 basic OTTHUMT00000362751.1        <NA>              <NA>        <NA>
## 3 basic OTTHUMT00000362751.1           1 ENSE00002234944.1        <NA>
## 4 basic OTTHUMT00000362751.1           2 ENSE00003582793.1        <NA>
## 5 basic OTTHUMT00000362751.1           3 ENSE00002312635.1        <NA>
## 6 basic OTTHUMT00000002844.2        <NA>              <NA> PGO:0000019
##   protein_id ccdsid
## 1       <NA>   <NA>
## 2       <NA>   <NA>
## 3       <NA>   <NA>
## 4       <NA>   <NA>
## 5       <NA>   <NA>
## 6       <NA>   <NA>
```

One thing you can find is that there is no columns in the data frame. Let's match which information is provided in columns. You can find the instruction page in the website (link).

Based on this, you can assign a name for 9 columns. One thing to remember is you should not use space for the column name. Spacing in the column name is actually working but not a good habit for your code. So please replace a space with underscore in the column name.

```r
# Assign column names according to the GENCODE instruction.
cols = c('chrom', 'source', 'feature_type', 'start', 'end', 'score', 'strand', 'phase', 'info')
```

Now you can set up the column names into the col_names parameter, and load the file into a data frame.

```r
d = read_delim('gencode.v31.basic.annotation.gtf.gz',
               delim='\t', skip = 5,
               progress = F,
               col_names = cols)
```

```
## Rows: 1756502 Columns: 9

## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (7): chrom, source, feature_type, score, strand, phase, info
## dbl (2): start, end

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

You can find the column names are now all set.

```r
head(d)
```

```
## # A tibble: 6 x 9
##   chrom source feature_type start   end score strand phase info
##   <chr> <chr>  <chr>        <dbl> <dbl> <chr> <chr>  <chr> <chr>
## 1 chr1  HAVANA gene         11869 14409 .     +      .     "gene_id \"ENSG00000~
## 2 chr1  HAVANA transcript   11869 14409 .     +      .     "gene_id \"ENSG00000~
## 3 chr1  HAVANA exon         11869 12227 .     +      .     "gene_id \"ENSG00000~
## 4 chr1  HAVANA exon         12613 12721 .     +      .     "gene_id \"ENSG00000~
## 5 chr1  HAVANA exon         13221 14409 .     +      .     "gene_id \"ENSG00000~
## 6 chr1  HAVANA transcript   12010 13670 .     +      .     "gene_id \"ENSG00000~
```

When you loaded the file, you see the message about the data class. You might want to overview this data.

```
summary(d)
```

```
##     chrom              source          feature_type           start
##  Length:1756502     Length:1756502     Length:1756502     Min.   :      577
##  Class :character   Class :character   Class :character   1st Qu.: 32101517
##  Mode  :character   Mode  :character   Mode  :character   Median : 61732754
##                                                           Mean   : 75288563
##                                                           3rd Qu.:111760181
##                                                           Max.   :248936581
##       end               score              strand              phase
##  Min.   :      647   Length:1756502     Length:1756502     Length:1756502
##  1st Qu.: 32107331   Class :character   Class :character   Class :character
##  Median : 61738373   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 75292632
##  3rd Qu.:111763007
##  Max.   :248937043
##      info
##  Length:1756502
##  Class :character
##  Mode  :character
##
##
##
```

2.2 How many feature types in the GENCODE dataset?

As instructed in the GENCODE website, the GENCODE dataset provides a range of annotations for the feature type. You can check feature types using ______ function.

```
d %>% group_by(feature_type) %>% count(feature_type)
```

```
## # A tibble: 8 x 2
## # Groups:   feature_type [8]
##   feature_type        n
##   <chr>           <int>
## 1 CDS            567862
## 2 exon           744835
## 3 gene            60603
## 4 Selenocysteine     96
## 5 start_codon     57886
## 6 stop_codon      57775
## 7 transcript     108243
## 8 UTR            159202
```

```
table(d$feature_type)
```

```
##
##            CDS           exon           gene Selenocysteine    start_codon
##         567862         744835          60603             96          57886
##     stop_codon     transcript            UTR
##          57775         108243         159202
```

How many feature types provided in the GENCODE? And how many items stored for each feature type? Please write down the number of feature types from the dataset. Also, if you are not familiar with these types, it would be good to put one or two sentences that can describe each type).

```
# 8 feature types
```

2.3 How many genes we have?

Let's count the number of genes in our genome. Since we know that the column feature_type contains rows with gene, which contains obviously annotations for genes. We might want to subset those rows from the data frame.

```
d1 <- filter(d, feature_type == "gene")
nrow(d1)
```

```
## [1] 60603
```

2.4 Ensembl, Havana and CCDS.

Gene annotation for the human genome is provided by multiple organizations with different gene annotation methods and strategy. This means that information can be varying by resources, and users need to understand heterogeniety inherent in annotation databases.

The GENCODE project utlizes two sources of gene annotation. 1. Havana: Manual gene annotation (detailed strategy in here) 2. Ensembl: Automatic gene annotation (detailed strategy in here)

It provides the combination of Ensembl/HAVANA gene set as the default gene annotation for the human genome. In addition, they also guarantee that all transcripts from the Consensus Coding Sequence (CCDS) set are present in the GENCODE gene set. The CCDS project is a collaborative effort to identify a core set of protein coding regions that are consistently annotated and of high quality. Initial results from the Consensus CDS (CCDS) project are now available through the appropriate Ensembl gene pages and from the CCDS project page at NCBI. The CCDS set is built by consensus among Ensembl, the National Center for Biotechnology Information (NCBI), and the HUGO Gene Nomenclature Committee (HGNC) for human (link).

Right. Then now we count how many genes annotated with HAVANA and ENSEMBL.

```
d %>% group_by(source) %>% count(source)
```

```
## # A tibble: 2 x 2
## # Groups:   source [2]
##   source        n
##   <chr>     <int>
## 1 ENSEMBL  245185
## 2 HAVANA  1511317
```

2.5 do.call

Since the last column info contains a long string for multiple annotations, we will need to split it to extract each annotation. For example, the first line for transcript annotation looks like this:

```
# chr1    HAVANA    transcript    11869    14409    .    +    .    gene_id "ENSG00000223972.5"; transcr
```

If you would like to split transcript_support_level and create a new column, you can use strsplit function.

```
a = 'chr1    HAVANA    transcript    11869    14409    .    +    .    gene_id "ENSG00000223972.5"; trans

strsplit(a, 'transcript_support_level\\s+"')
```

```
## [[1]]
## [1] "chr1    HAVANA    transcript    11869    14409    .    +    .    gene_id \"ENSG00000223972.5\";
## [2] "1\"; hgnc_id \"HGNC:37102\"; tag \"basic\"; havana_gene \"OTTHUMG00000000961.2\"; havana_transc
```

After split the string, you can select the second item in the list ([[1]][2]).

```
strsplit(a, 'transcript_support_level\\s+"')[[1]][2]
```

```
## [1] "1\"; hgnc_id \"HGNC:37102\"; tag \"basic\"; havana_gene \"OTTHUMG00000000961.2\"; havana_transc
```

You can find the 1 in the first position, which you will need to split again.

```
b = strsplit(a, 'transcript_support_level\\s+"')[[1]][2]
strsplit(b, '\\"')
```

```
## [[1]]
##  [1] "1"                    "; hgnc_id "           "HGNC:37102"
##  [4] "; tag "               "basic"                "; havana_gene "
##  [7] "OTTHUMG00000000961.2" "; havana_transcript " "OTTHUMT00000362751.1"
## [10] ";"
```

From this, you will get the first item in the list ([[1]][1]).

Now you would like to apply strsplit function across vectors. For this, do.call function can be easily implemented to strsplit over the vectors from one column. Let's try this.

```
head(do.call(rbind.data.frame, strsplit(a, 'transcript_support_level\\s+"'))[[2]])
```

```
## [1] "1\"; hgnc_id \"HGNC:37102\"; tag \"basic\"; havana_gene \"OTTHUMG00000000961.2\"; havana_transc
```

Now you can write two lines of codes to process two steps we discussed above.

```
# First filter transcripts and create a data frame.
d2 <- d %>% filter(feature_type == 'transcript')

# Now apply the functions.
d2$transcript_support_level <- as.character(do.call(rbind.data.frame,
                                                    strsplit(d2$info, 'transcript_support_level\\s+"'))

d2$transcript_support_level <- as.character(do.call(rbind.data.frame,
                                                    strsplit( d2$transcript_support_level, '\\"'))[[1]]
```

Now you can check the strsplit works.

```r
head(d2$transcript_support_level)
```

```
## [1] "1"  "NA" "NA" "NA" "5"  "5"
```

You can use the same method to extract other annotations, like gene_id, gene_name etc.

3. Exercises

Here I list the questions for your activity. Please note that it is an exercise for tidyverse functions, which you will need to use in your code. In addition, you will need to write an one-line code for each question using pipe %>%.

For questions, you should read some information thoroughly, including:

- Gene biotype.
- 0 or 1 based annotation in GTF, BED format
- Why some features have 1 bp length?
- What is the meaning of zero-length exons in GENCODE? Also fun to have a review for microexons
- Transcript support level (TSL)

```r
d_ <- rtracklayer::import('gencode.v31.basic.annotation.gtf.gz')
d_ <- as.data.frame(d_)
d_ <- d_ %>% rename(chromosome = seqnames)
```

3.1 Annotation of transcripts in our genome

Q1. Computes the number of transcripts per gene. What is the mean number of transcripts per gene? What is the quantile (25%, 50%, 75%) for these numbers? Which gene has the greatest number of transcript?

```r
# <Mean number of transcripts per gene>
d_2 <- d_ %>%
  filter(type == "transcript")
tperg <- d_2 %>%
  count(gene_id)
tperg %>%
  summarize(mean = mean(n))
```

```
##     mean
## 1 1.7861
```

```r
# <Quantile>
p <- seq(0.25, 0.75, 0.25)
quantile(tperg$n, p)
```

```
## 25% 50% 75%
##   1   1   2
```

```
# <Max>
max(tperg$n)
```

```
## [1] 87
```

Q2. Compute the number of transcripts per gene among gene biotypes. For example, compare the number of transcript per gene between protein-coding genes, long noncoding genes, pseudogenes.

```
# <Mean number of transcripts per gene among gene biotypes>
# Count the number of transcripts per gene among gene biotypes
tperg_type <- d_2 %>% group_by(gene_type) %>% summarize(gene_n = n_distinct(gene_id), transcripts_n=n()
tperg_type
```

```
## # A tibble: 40 x 4
##    gene_type       gene_n transcripts_n t_per_g
##    <chr>            <int>         <int>   <dbl>
##  1 IG_C_gene           14            14    1
##  2 IG_C_pseudogene      9             9    1
##  3 IG_D_gene           37            37    1
##  4 IG_J_gene           18            18    1
##  5 IG_J_pseudogene      3             3    1
##  6 IG_pseudogene        1             1    1
##  7 IG_V_gene          144           144    1
##  8 IG_V_pseudogene    188           188    1
##  9 lncRNA           16840         24993    1.48
## 10 miRNA             1881          1881    1
## # ... with 30 more rows
```

```
# Protein-coding genes, long noncoding genes, pseudogenes
d_2 <- d_2 %>%
  mutate(group = case_when(
    gene_type %in% c("IG_C_gene","IG_D_gene","IG_J_gene","IG_LV_gene", "IG_V_gene", "TR_C_gene","TR_J_g
    gene_type %in% c("IG_pseudogene","IG_C_pseudogene","IG_J_pseudogene","IG_V_pseudogene","TR_V_pseudo
    gene_type == "lncRNA" ~ "long noncoding gene",
    TRUE ~ "Others"
  ))

d_2 %>% group_by(group) %>% summarize(gene_n = n_distinct(gene_id), transcripts_n=n(), t_per_g = transc
```

```
## # A tibble: 4 x 4
##   group                gene_n transcripts_n t_per_g
##   <chr>                 <int>         <int>   <dbl>
## 1 long noncoding gene   16840         24993    1.48
## 2 Others                 8140          8140    1
## 3 Protein-coding genes  20383         58254    2.86
## 4 pseudogenes           15240         16856    1.11
```

Q3. Final task is to compute the number of transcripts per gene per chromosome.

```
# Count the number of transcripts per gene per chromosome
tperg_chrm <- d_2 %>% group_by(chromosome) %>% summarize(transcripts_n=n(), gene_n = n_distinct(gene_id
tperg_chrm
```

```
## # A tibble: 25 x 4
##    chromosome transcripts_n gene_n t_per_g
##    <fct>              <int>  <int>   <dbl>
##  1 chr1                9827   5471    1.80
##  2 chr2                7432   4196    1.77
##  3 chr3                6157   3185    1.93
##  4 chr4                4662   2651    1.76
##  5 chr5                5203   2983    1.74
##  6 chr6                5455   3059    1.78
##  7 chr7                5292   3014    1.76
##  8 chr8                4350   2482    1.75
##  9 chr9                3949   2327    1.70
## 10 chr10               4157   2332    1.78
## # ... with 15 more rows
```

3.2 Gene length in the GENCODE

Q1. What is the average length of human genes?

```
d_1 <- d_ %>%
  filter(type == "gene")
mean(d_1$width)
```

```
## [1] 32629.02
```

Q2. Is the distribution of gene length differed by autosomal and sex chromosomes? Please calculate the quantiles (0%, 25%, 50%, 75%, 100%) of the gene length for each group.

```
# 1) Add "chrom_group" column
d_1 <- d_1%>%
  mutate(chrom_group = case_when(
    chromosome == "chrM" ~ "Mitrochondrial",
    chromosome %in% c("chrX", "chrY") ~ "Sex",
    TRUE ~ "Autosomal" ))

# 2) Quantiles of gene length for each group
Sex_length <- d_1 %>%
  filter(chrom_group == "Sex") %>%
  .$width

Autosome_length <- d_1 %>%
  filter(chrom_group == "Autosomal") %>%
  .$width

quantile(Sex_length,)
```

```
##      0%     25%     50%     75%    100%
##      48     473    1912   13502 2241765
```

```
quantile(Autosome_length,)
```

```
##      0%     25%     50%     75%    100%
##       8     565    3779   25813 2473537
```

Q3. Is the distribution of gene length differed by gene biotype? Please calculate the quantiles (0%, 25%, 50%, 75%, 100%) of the gene length for each group.

```
d_1 %>%
  group_by(gene_type) %>%
  summarize(quant0 = quantile(width, probs = 0),
            quant25 = quantile(width, probs = 0.25),
            quant50 = quantile(width, probs = 0.5),
            quant75 = quantile(width, probs = 0.75),
            quant100 = quantile(width, probs = 1))
```

```
## # A tibble: 40 x 6
##    gene_type       quant0 quant25 quant50 quant75 quant100
##    <chr>            <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
##  1 IG_C_gene          441    477.   4590.   5479.     8914
##  2 IG_C_pseudogene    248    313     317     734      5211
##  3 IG_D_gene           11     17      20      31        37
##  4 IG_J_gene           33     38.2    49      67.8     176
##  5 IG_J_pseudogene     50     52.5    55      57.5      60
##  6 IG_pseudogene      306    306     306     306       306
##  7 IG_V_gene          294    496.    522     572.   176628
##  8 IG_V_pseudogene     28    271     416.    458.      792
##  9 lncRNA              68   1874.   6272. 24774.   1375317
## 10 miRNA               41     70      80      91       180
## # ... with 30 more rows
```

3.3 Transcript support levels(TSL)

The GENCODE TSL provides a consistent method of evaluating the level of support that a GENCODE transcript annotation is actually expressed in humans.

Q1. With transcript, how many transcripts are categorized for each TSL?

```
d_2 %>% group_by(transcript_support_level) %>% count()
```

```
## # A tibble: 7 x 2
## # Groups:   transcript_support_level [7]
##   transcript_support_level     n
##   <chr>                    <int>
## 1 1                        31801
## 2 2                        13372
## 3 3                         7228
## 4 4                         2245
## 5 5                        13674
## 6 NA                       27843
## 7 <NA>                     12080
```

Q2. From the first question, please count the number of transcript for each TSL by gene biotype.

```
d_2 %>% group_by(gene_type, transcript_support_level) %>% count()
```

```
## # A tibble: 91 x 3
## # Groups:   gene_type, transcript_support_level [91]
##    gene_type       transcript_support_level     n
##    <chr>           <chr>                    <int>
##  1 IG_C_gene       1                            1
##  2 IG_C_gene       5                            1
##  3 IG_C_gene       NA                           7
##  4 IG_C_gene       <NA>                         5
##  5 IG_C_pseudogene NA                           9
##  6 IG_D_gene       NA                          37
##  7 IG_J_gene       NA                          18
##  8 IG_J_pseudogene NA                           3
##  9 IG_pseudogene   NA                           1
## 10 IG_V_gene       5                            3
## # ... with 81 more rows
```

Q3. From the first question, please count the number of transcript for each TSL by source.

```
d_2 %>% group_by(source, transcript_support_level) %>% count()
```

```
## # A tibble: 14 x 3
## # Groups:   source, transcript_support_level [14]
##    source  transcript_support_level     n
##    <fct>   <chr>                    <int>
##  1 HAVANA  1                        29434
##  2 HAVANA  2                        12052
##  3 HAVANA  3                         6964
##  4 HAVANA  4                         2116
##  5 HAVANA  5                        10157
##  6 HAVANA  NA                       19962
##  7 HAVANA  <NA>                     11901
##  8 ENSEMBL 1                         2367
##  9 ENSEMBL 2                         1320
## 10 ENSEMBL 3                          264
## 11 ENSEMBL 4                          129
## 12 ENSEMBL 5                         3517
## 13 ENSEMBL NA                        7881
## 14 ENSEMBL <NA>                       179
```

3.4 CCDS in the GENCODE

Q1. With gene, please create a data frame with the columns - gene_id, gene_name, hgnc_id, gene_type, chromosome, start, end, and strand. Then, please create new columns for presence of hgnc and ccds. For example, you can put 1 in the column isHgnc, if hgnc annotation is avaiable, or 0 if not. Then, you can put 1 in the column isCCDS, if ccds annotation is avaiable, or 0 if not.

```
# Create a new data frame
d_gene <- d_1 %>%
  select(gene_id, gene_name, hgnc_id, gene_type, chromosome, start, end, strand)

# column isHgnc
d_gene <- d_gene %>%
  mutate(isHgnc = case_when(
    is.na(hgnc_id) ~ 0,
    TRUE ~ 1
  ))

# column isCCDS
CCDS_gene <- d_ %>%
  filter(!is.na(ccdsid)) %>%
  count(gene_id) %>%
  .$gene_id

d_gene <- d_gene %>%
  mutate(isCCDS = case_when(
    gene_id %in% CCDS_gene ~ 1,
    TRUE ~ 0
  ))

head(d_gene)
```

```
##             gene_id   gene_name    hgnc_id                          gene_type
## 1 ENSG00000223972.5     DDX11L1 HGNC:37102 transcribed_unprocessed_pseudogene
## 2 ENSG00000227232.5      WASH7P HGNC:38034             unprocessed_pseudogene
## 3 ENSG00000278267.1   MIR6859-1 HGNC:50039                              miRNA
## 4 ENSG00000243485.5 MIR1302-2HG HGNC:52482                             lncRNA
## 5 ENSG00000284332.1   MIR1302-2 HGNC:35294                              miRNA
## 6 ENSG00000237613.2     FAM138A HGNC:32334                             lncRNA
##   chromosome start   end strand isHgnc isCCDS
## 1       chr1 11869 14409      +      1      0
## 2       chr1 14404 29570      -      1      0
## 3       chr1 17369 17436      -      1      0
## 4       chr1 29554 31109      +      1      0
## 5       chr1 30366 30503      +      1      0
## 6       chr1 34554 36081      -      1      0
```

Q2. Please count the number of hgnc by gene biotypes.

```
d_gene %>%
  group_by(gene_type) %>%
  summarize(hgnc_id_n = n_distinct(hgnc_id))
```

```
## # A tibble: 40 x 2
##    gene_type       hgnc_id_n
##    <chr>               <int>
##  1 IG_C_gene              14
##  2 IG_C_pseudogene         9
##  3 IG_D_gene              37
##  4 IG_J_gene              18
##  5 IG_J_pseudogene         3
##  6 IG_pseudogene           1
##  7 IG_V_gene             143
##  8 IG_V_pseudogene       186
##  9 lncRNA               3962
## 10 miRNA                1855
## # ... with 30 more rows
```

Q3. Please count the number of hgnc by level. Please note that level in this question is not TSL. Please find information in this link: 1 (verified loci), 2 (manually annotated loci), 3 (automatically annotated loci).

```
d_gene %>%
  mutate(level = d_1$level) %>%
  group_by(level) %>%
  summarize(hgnc_id_n = n_distinct(hgnc_id))
```

```
## # A tibble: 3 x 2
##   level hgnc_id_n
##   <chr>     <int>
## 1 1         11342
## 2 2         20475
## 3 3          5725
```

3.5 Transcripts in the GENCODE

Q1. Which gene has the largest number of transcripts?

```
d_ %>%
  group_by(gene_id, gene_name) %>%
  count() %>%
  as.data.frame() %>%
  filter(n == max(n))
```

```
##              gene_id gene_name    n
## 1 ENSG00000155657.26       TTN 3876
```

Q2. Please calculate the quantiles (0%, 25%, 50%, 75%, 100%) of the gene length for protein coding genes and long noncoding genes.

```
d_1 %>%
  filter(gene_type %in% c("lncRNA", "protein_coding")) %>%
  group_by(gene_type) %>%
  summarize(quant0 = quantile(width, probs = 0),
            quant25 = quantile(width, probs = 0.25),
            quant50 = quantile(width, probs = 0.5),
            quant75 = quantile(width, probs = 0.75),
            quant100 = quantile(width, probs = 1))
```

```
## # A tibble: 2 x 6
##   gene_type      quant0 quant25 quant50 quant75 quant100
##   <chr>            <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
## 1 lncRNA              68   1874.   6272.  24774.  1375317
## 2 protein_coding     117   9632.  27212   70809   2473537
```

Q3. Please count the number of transcripts by chromosomes.

```
d_2 %>%
  group_by(chromosome) %>%
  count()
```

```
## # A tibble: 25 x 2
## # Groups:   chromosome [25]
##    chromosome     n
##    <fct>      <int>
##  1 chr1        9827
##  2 chr2        7432
##  3 chr3        6157
##  4 chr4        4662
##  5 chr5        5203
##  6 chr6        5455
##  7 chr7        5292
##  8 chr8        4350
##  9 chr9        3949
## 10 chr10       4157
## # ... with 15 more rows
```

3.6 Autosommal vs. Sex chromosomes

Q1. Please calculate the number of genes per chromosome.

```
d_1 %>%
  group_by(chromosome) %>%
  count()
```

```
## # A tibble: 25 x 2
## # Groups:   chromosome [25]
##    chromosome     n
##    <fct>      <int>
##  1 chr1        5471
##  2 chr2        4196
##  3 chr3        3185
##  4 chr4        2651
##  5 chr5        2983
##  6 chr6        3059
##  7 chr7        3014
##  8 chr8        2482
##  9 chr9        2327
## 10 chr10       2332
## # ... with 15 more rows
```

Q2. Please compare the number of genes between autosomal and sex chromosome (Mean, Median).

```
# chrom_group in d_1 already has been defined in previous questions
d_1 %>%
  filter(chrom_group %in% c("Autosomal", "Sex")) %>%
  group_by(chrom_group, chromosome) %>%
  count() %>%
  group_by(chrom_group) %>%
  summarize(sum = sum(n), mean = mean(n), median = median(n))
```

```
## # A tibble: 2 x 4
##   chrom_group   sum  mean median
##   <chr>       <int> <dbl>  <dbl>
## 1 Autosomal   57577 2617.  2604.
## 2 Sex          2989 1494.  1494.
```

Q3. Please divide the genes into groups ‘protein coding’ and ‘long noncoding’, and then compare the number of genes in each chromosomes within groups.

```
d_1 %>%
  filter(gene_type %in% c("protein_coding", "lncRNA")) %>%
  group_by(gene_type, chromosome) %>%
  count()
```

```
## # A tibble: 49 x 3
## # Groups:   gene_type, chromosome [49]
##    gene_type chromosome     n
##    <chr>     <fct>      <int>
##  1 lncRNA    chr1        1416
##  2 lncRNA    chr2        1241
##  3 lncRNA    chr3         861
##  4 lncRNA    chr4         790
##  5 lncRNA    chr5         950
##  6 lncRNA    chr6         826
##  7 lncRNA    chr7         720
##  8 lncRNA    chr8         831
##  9 lncRNA    chr9         555
## 10 lncRNA    chr10        695
## # ... with 39 more rows
```