[Tutorial] Human Genome Annotation

Contents

1 1. Introduction

1.1 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.

Figure 1. Comparison of GENCODE and RefSeq gene annotation and the impact of reference geneset on variant effect prediction (Frankish et al. 2015). A) Mean number of alternatively spliced transcripts per multi-exon protein-coding locus B) Mean number of unique CDS per multi-exon protein-coding locus C) Mean number of unique (non-redundant) exons per multi-exon protein-coding locus D) Percentage genomic coverage of unique (non-redundant) exons at multi-exon protein-coding loci.

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

1.2 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 2. Explore your data

2.1 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:

```
# Run from your terminal, not R console
# wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_31/gencode.v31.basic.annotation.

# Once you downloaded the file, you won't need to download it again. So please comment out the command
```

Once you download the file, you can print out the first few lines using the following bash command (we will learn UNIX commands later):

```
# Run from your terminal, not R console
# gzcat gencode.v31.basic.annotation.gtf.gz | head -7
```

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  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     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")
```

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

```r
library(tidyverse)
```

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

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:

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

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

```r
head(d)
```

```
## # A tibble: 6 x 9
##   X1    X2     X3             X4    X5 X6    X7    X8    X9
##   <chr> <chr>  <chr>       <dbl> <dbl> <chr> <chr> <chr> <chr>
## 1 chr1  HAVANA gene        11869 14409 .     +     .     "gene_id \"ENSG00000223~
## 2 chr1  HAVANA transcript 11869 14409 .     +     .     "gene_id \"ENSG00000223~
## 3 chr1  HAVANA exon        11869 12227 .     +     .     "gene_id \"ENSG00000223~
## 4 chr1  HAVANA exon        12613 12721 .     +     .     "gene_id \"ENSG00000223~
## 5 chr1  HAVANA exon        13221 14409 .     +     .     "gene_id \"ENSG00000223~
## 6 chr1  HAVANA transcript 12010 13670 .     +     .     "gene_id \"ENSG00000223~
```

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.

```
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 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()
```

```
## # 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$type)
```

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).

3 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")
# d1 = d[d$feature_type == "gene", ]
```

3.1 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
```

3.2 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]]
```

```
## [1] "chr1    HAVANA    transcript    11869    14409    .    +    .    gene_id \"ENSG00000223972.5\";
## [2] "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_supp

d2$transcript_support_level <- as.character(do.call(rbind.data.frame, strsplit(d2$transcript_support_lev
```

Now you can check the strsplit works.

```
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.

4 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)

```
d = rtracklayer::import('gencode.v31.basic.annotation.gtf.gz')
d = d %>%  as.data.frame()
```

4.1 3.1. Annotation of transcripts in our genome

1. 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?

```
# Computes the number of transcripts per gene
d %>% group_by(gene_id, type) %>%
  filter(type == "transcript") %>%
  count()
```

```
## # A tibble: 60,603 x 3
## # Groups:   gene_id, type [60,603]
##    gene_id            type           n
##    <chr>              <fct>      <int>
##  1 ENSG00000000003.14 transcript     3
##  2 ENSG00000000005.6  transcript     1
##  3 ENSG00000000419.12 transcript     2
##  4 ENSG00000000457.14 transcript     3
##  5 ENSG00000000460.17 transcript     5
##  6 ENSG00000000938.13 transcript     3
##  7 ENSG00000000971.15 transcript     3
##  8 ENSG00000001036.13 transcript     1
##  9 ENSG00000001084.13 transcript     5
## 10 ENSG00000001167.14 transcript     2
## # ... with 60,593 more rows
```

```
# The mean number of transcripts per gene
d %>% group_by(gene_id, type) %>%
  filter(type == "transcript") %>%
  count() %>%
  ungroup() %>%
  summarize(mean = mean(n))
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  1.79
```

```
# The quantile (25%, 50%, 75%) for these numbers
d %>% group_by(gene_id, type) %>%
  filter(type == "transcript") %>%
  count() %>%
  pull(n) %>%
  quantile(seq(0.25, 0.75, 0.25))
```

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

```
# Gene has the greatest number of transcript
d %>% group_by(gene_id, type) %>%
  filter(type == "transcript") %>%
  count() %>%
  ungroup() %>%
  top_n(1, n)
```

```
## # A tibble: 1 x 3
##   gene_id            type           n
##   <chr>              <fct>      <int>
## 1 ENSG00000109339.22 transcript    87
```

2. 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.

```
d %>% group_by(gene_type, type) %>%
  filter(type == "transcript") %>%
  count()
```

```
## # A tibble: 40 x 3
## # Groups:   gene_type, type [40]
##    gene_type       type           n
##    <chr>           <fct>      <int>
##  1 IG_C_gene       transcript    14
##  2 IG_C_pseudogene transcript     9
##  3 IG_D_gene       transcript    37
##  4 IG_J_gene       transcript    18
##  5 IG_J_pseudogene transcript     3
##  6 IG_pseudogene   transcript     1
##  7 IG_V_gene       transcript   144
##  8 IG_V_pseudogene transcript   188
##  9 lncRNA          transcript 24993
## 10 miRNA           transcript  1881
## # ... with 30 more rows
```

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

```
d %>% group_by(seqnames, gene_id, type) %>%
  filter(type == "transcript") %>%
  count()
```

```
## # A tibble: 60,603 x 4
## # Groups:   seqnames, gene_id, type [60,603]
##    seqnames gene_id            type           n
##    <fct>    <chr>              <fct>      <int>
##  1 chr1     ENSG00000000457.14 transcript     3
##  2 chr1     ENSG00000000460.17 transcript     5
##  3 chr1     ENSG00000000938.13 transcript     3
##  4 chr1     ENSG00000000971.15 transcript     3
##  5 chr1     ENSG00000001460.18 transcript     3
##  6 chr1     ENSG00000001461.17 transcript     4
##  7 chr1     ENSG00000004455.16 transcript     8
##  8 chr1     ENSG00000004487.16 transcript     3
##  9 chr1     ENSG00000006555.11 transcript     2
## 10 chr1     ENSG00000007341.19 transcript     6
## # ... with 60,593 more rows
```

4.2 3.2. Gene length in the GENCODE

1. What is the average length of human genes?

```
d %>% filter(type == "gene") %>%
  summarize(mean = mean(width))
```

```
##       mean
## 1 32629.02
```

2. 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.

```
q = c(0, 0.25, 0.50, 0.75, 1)

d %>%
  filter(type == "gene") %>%
  mutate(seqnames = ifelse(seqnames %in% c("chrX", "chrY"), "Sex chromosome", "Autosomal chromosome"))
  group_by(seqnames, type) %>%
  summarize(quant0 = quantile(width, q[1]),
            quant25 = quantile(width, q[2]),
            quant50 = quantile(width, q[3]),
            quant75 = quantile(width, q[4]),
            quant100 = quantile(width, q[5]))
```

```
## `summarise()` has grouped output by 'seqnames'. You can override using the `.groups` argument.
```

```
## # A tibble: 2 x 7
## # Groups:   seqnames [2]
##   seqnames             type  quant0 quant25 quant50 quant75 quant100
##   <chr>                <fct>  <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
## 1 Autosomal chromosome gene       8     564   3768.  25788.  2473537
## 2 Sex chromosome       gene      48     473   1912   13502   2241765
```

3. 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.

```
q <- c(0, 0.25, 0.5, 0.75, 1)
d %>% group_by(gene_type)%>%
  summarize(quantile0 = quantile(width, q[1]),
            quantile25 = quantile(width, q[2]),
            quantile50 = quantile(width, q[3]),
            quantile75 = quantile(width, q[4]),
            quantile100 = quantile(width, q[5]))
```

```
## # A tibble: 40 x 6
##    gene_type       quantile0 quantile25 quantile50 quantile75 quantile100
##    <chr>               <dbl>      <dbl>      <dbl>      <dbl>       <dbl>
##  1 IG_C_gene               3         92       312.        336        8914
##  2 IG_C_pseudogene        34        293       316         424        5211
##  3 IG_D_gene               3         17        20          31          37
##  4 IG_J_gene               3         38        49          70         176
##  5 IG_J_pseudogene        50         50        55          60          60
##  6 IG_pseudogene         306        306       306         306         306
##  7 IG_V_gene               3         46       296         436      176628
##  8 IG_V_pseudogene        28        235       291         437         792
##  9 lncRNA                  1        126       339        2658     1375317
## 10 miRNA                  41         70        80          91         180
## # ... with 30 more rows
```

4.3 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.

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

```
d %>% group_by(type, transcript_support_level) %>%
  filter(type == "transcript") %>%
  count()
```

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

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

```
d %>% group_by(gene_type, transcript_support_level, type) %>%
  filter(type == "transcript") %>%
  count()
```

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

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

```
d %>% group_by(source, transcript_support_level, type) %>%
  filter(type == "transcript") %>%
  count()
```

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

4.4 3.4. CCDS in the GENCODE

1. 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.

```
gene <- data.frame(gene_id = d$gene_id, gene_name = d$gene_name, hgnc_id = d$hgnc_id, gene_type =d$gene_
```

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

```
gene %>% group_by(gene_type, isHgnc) %>%
  count()
```

```
## # A tibble: 60 x 3
## # Groups:   gene_type, isHgnc [60]
##    gene_type       isHgnc     n
##    <chr>            <dbl> <int>
##  1 IG_C_gene            1   176
##  2 IG_C_pseudogene      1    33
##  3 IG_D_gene            1   152
##  4 IG_J_gene            1    76
##  5 IG_J_pseudogene      1     9
##  6 IG_pseudogene        0     3
##  7 IG_V_gene            0    14
##  8 IG_V_gene            1  1101
##  9 IG_V_pseudogene      0    11
## 10 IG_V_pseudogene      1   653
## # ... with 50 more rows
```

3. 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 %>% group_by(level, hgnc_id) %>%
  count()
```

```
## # A tibble: 49,081 x 3
## # Groups:   level, hgnc_id [49,081]
##    level hgnc_id        n
##    <chr> <chr>      <int>
##  1 1     HGNC:100      26
##  2 1     HGNC:10001    16
##  3 1     HGNC:10002    37
##  4 1     HGNC:10007     1
##  5 1     HGNC:10008    30
##  6 1     HGNC:1001     22
##  7 1     HGNC:10010     3
##  8 1     HGNC:1004     18
##  9 1     HGNC:10049     3
## 10 1     HGNC:10055     1
## # ... with 49,071 more rows
```

4.5 3.5. Transcripts in the GENCODE

1. Which gene has the largest number of transcripts?

```
d %>% group_by(gene_id, type) %>%
  filter(type == "transcript") %>%
  count() %>%
  ungroup() %>%
  top_n(1, n)
```

```
## # A tibble: 1 x 3
##   gene_id            type           n
##   <chr>              <fct>      <int>
## 1 ENSG00000109339.22 transcript    87
```

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

```
q = c(0, 0.25, 0.50, 0.75, 1)
d %>% group_by(gene_type) %>%
  filter(gene_type %in% c("protein_coding", "lncRNA") & type == "gene") %>%
  summarize(quant0 = quantile(width, q[1]),
            quant25 = quantile(width, q[2]),
            quant50 = quantile(width, q[3]),
            quant75 = quantile(width, q[4]),
            quant100 = quantile(width, q[5]))
```

```
## # 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
```

3. Please count the number of transcripts by chromosomes.

```
d %>% group_by(seqnames, type) %>%
  filter(type == "transcript") %>%
  count()
```

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

4.6 3.6. Autosomal vs. Sex chromosomes.

1. Please calculate the number of genes per chromosome.

```
d %>% group_by(seqnames, type) %>%
  filter(type == "gene") %>%
  count()
```

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

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

```
d %>%
  filter(type == "gene") %>%
  group_by(seqnames, type) %>%
  count() %>%
  mutate(seqnames = ifelse(seqnames %in% c("chrX", "chrY"), "Sex chromosome", "Autosomal chrosome")) %>%
  summarize(mean = mean(n),
            median = median(n))
```

```
## `summarise()` has grouped output by 'seqnames'. You can override using the `.groups` argument.
```

```
## # A tibble: 2 x 4
## # Groups:   seqnames [2]
##   seqnames           type   mean median
##   <chr>              <fct> <dbl>  <dbl>
## 1 Autosomal chrosome gene  2505.  2556
## 2 Sex chromosome     gene  1494.  1494.
```

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

```
d %>% group_by(seqnames, gene_type, type) %>%
  filter(type == "gene" & gene_type %in% c("protein_coding", "lncRNA")) %>%
  count()
```

```
## # A tibble: 49 x 4
## # Groups:   seqnames, gene_type, type [49]
##    seqnames gene_type      type      n
##    <fct>    <chr>          <fct> <int>
##  1 chr1     lncRNA         gene   1416
##  2 chr1     protein_coding gene   2048
##  3 chr2     lncRNA         gene   1241
##  4 chr2     protein_coding gene   1247
##  5 chr3     lncRNA         gene    861
##  6 chr3     protein_coding gene   1075
##  7 chr4     lncRNA         gene    790
##  8 chr4     protein_coding gene    751
##  9 chr5     lncRNA         gene    950
## 10 chr5     protein_coding gene    886
## # ... with 39 more rows
```