Genomic composition: mining the GTF files

Ugo Borello, Eleonora Sara Bacci

2025-09-14

This tutorial shows how to extract genomic data from a genomic annotation file.

The file format used for the script is the `GTF`. A `GTF` file (General Feature Format) is an annotation file used to store gene models. A GTF file is a tabular format that has a header, rows started with "#", following by one line per genome feature, each one containing 9 columns of data. Fields must be tab-separated.

First of all let's load some libraries.

```
library(rtracklayer)
library(plyranges)
library(ggplot2)
library(grid)
```

Now we need to download the GTF file. In this tutorial we will focus on the *Homo sapiens* genome.

N.B. GFT files from different repository are structured differently. For example, the column 'gene_type' from Genecode repository is called 'gene_biotype' in the Ensembl repository.

Let's start downloading the *Homo sapiens* annotation file from the Ensembl Database. It will probably take a while, since it's a big file.

Once it's downloaded, we need to import it inside R, in a special object called GRanges. This object is a container for the gene models and their associated annotation.

```
my_url <- "https://ftp.ensembl.org/pub/release-114/gtf/homo_sapiens/Homo_sapiens.GRCh38.114.gtf.gz"
my_gtf <- basename(my_url)
download.file(url = my_url, destfile = paste0("./", my_gtf))


my_obj <- import(paste0("./", my_gtf))

head(my_obj)
```

```
## GRanges object with 6 ranges and 22 metadata columns:
##       seqnames          ranges strand |         source        type     score
##          <Rle>       <IRanges>  <Rle> |       <factor>    <factor> <numeric>
##   [1]        1 3069168-3438621      + | ensembl_havana gene               NA
##   [2]        1 3069168-3434342      + | havana         transcript         NA
##   [3]        1 3069168-3069296      + | havana         exon               NA
##   [4]        1 3069260-3069296      + | havana         CDS                NA
##   [5]        1 3069260-3069262      + | havana         start_codon        NA
##   [6]        1 3186125-3186474      + | havana         exon               NA
##           phase         gene_id gene_version   gene_name    gene_source
##       <integer>     <character>  <character> <character>    <character>
##   [1]      <NA> ENSG00000142611           17      PRDM16 ensembl_havana
##   [2]      <NA> ENSG00000142611           17      PRDM16 ensembl_havana
##   [3]      <NA> ENSG00000142611           17      PRDM16 ensembl_havana
```

```
##   [4]         0 ENSG00000142611           17      PRDM16 ensembl_havana
##   [5]         0 ENSG00000142611           17      PRDM16 ensembl_havana
##   [6]      <NA> ENSG00000142611           17      PRDM16 ensembl_havana
##         gene_biotype   transcript_id transcript_version transcript_name
##          <character>     <character>        <character>     <character>
##   [1] protein_coding            <NA>               <NA>            <NA>
##   [2] protein_coding ENST00000511072                  5      PRDM16-206
##   [3] protein_coding ENST00000511072                  5      PRDM16-206
##   [4] protein_coding ENST00000511072                  5      PRDM16-206
##   [5] protein_coding ENST00000511072                  5      PRDM16-206
##   [6] protein_coding ENST00000511072                  5      PRDM16-206
##       transcript_source transcript_biotype             tag
##             <character>        <character>     <character>
##   [1]              <NA>               <NA>            <NA>
##   [2]           havana     protein_coding gencode_primary
##   [3]           havana     protein_coding gencode_primary
##   [4]           havana     protein_coding gencode_primary
##   [5]           havana     protein_coding gencode_primary
##   [6]           havana     protein_coding gencode_primary
##       transcript_support_level exon_number         exon_id exon_version
##                    <character> <character>     <character>  <character>
##   [1]                     <NA>        <NA>            <NA>         <NA>
##   [2]                        5        <NA>            <NA>         <NA>
##   [3]                        5           1 ENSE00002048533            1
##   [4]                        5           1            <NA>         <NA>
##   [5]                        5           1            <NA>         <NA>
##   [6]                        5           2 ENSE00001754112            1
##           protein_id protein_version     ccds_id
##          <character>     <character> <character>
##   [1]            <NA>            <NA>        <NA>
##   [2]            <NA>            <NA>        <NA>
##   [3]            <NA>            <NA>        <NA>
##   [4] ENSP00000426975               1        <NA>
##   [5]            <NA>            <NA>        <NA>
##   [6]            <NA>            <NA>        <NA>
##   -------
##   seqinfo: 70 sequences from an unspecified genome; no seqlengths
```

the file contains basic information of a GRanges object such as `seqnames`, `ranges` and `strand`, as well as other metadata column that can be use to extract some specific data of our interest.

Let's mine onto this file.

```
table(my_obj$type)
```

```
##
##           gene      transcript            exon             CDS     start_codon
##          78894          387954         2165096          902508           99375
##      stop_codon  five_prime_utr three_prime_utr  Selenocysteine
##          93388          176048          212655             130
```

The human genome contains 78894 genes with different features.

All of them code for a protein?

```
table(my_obj$gene_biotype)
```

```
##
```

```
##                          artifact                          IG_C_gene
##                               147                                280
##                  IG_C_pseudogene                          IG_D_gene
##                                33                                148
##                        IG_J_gene                     IG_J_pseudogene
##                                76                                  9
##                     IG_pseudogene                          IG_V_gene
##                                 3                               1140
##                  IG_V_pseudogene                            lncRNA
##                               661                             943588
##                             miRNA                           misc_RNA
##                              5637                               6648
##                            Mt_rRNA                            Mt_tRNA
##                                 6                                 66
##              processed_pseudogene                      protein_coding
##                             29951                            3085983
##                        pseudogene                            ribozyme
##                                16                                 24
##                              rRNA                     rRNA_pseudogene
##                               159                               1491
##                            scaRNA                            snoRNA
##                               147                               2826
##                             snRNA                              sRNA
##                              5730                                15
##                               TEC                          TR_C_gene
##                              3107                                70
##                         TR_D_gene                          TR_J_gene
##                                20                               316
##                  TR_J_pseudogene                          TR_V_gene
##                                12                               833
##                  TR_V_pseudogene   transcribed_processed_pseudogene
##                               124                              3744
##     transcribed_unitary_pseudogene transcribed_unprocessed_pseudogene
##                              1846                             12260
##    translated_processed_pseudogene                unitary_pseudogene
##                                 8                               384
##             unprocessed_pseudogene                         vault_RNA
##                              8528                                12
```

As we can see there are different types of genes and also of pseudogenes!

Chromosomes size and information

Let's summarize the genomic features by chromosomes ... and scaffolds.

```
my_obj %>%
    dplyr::group_by(seqnames) %>%
    summarise(total = n()) %>%
    as.data.frame()
```

```
##      seqnames  total
## 1           1 384290
## 2           2 317346
## 3           3 258971
## 4           4 173483
## 5           5 192775
## 6           6 202004
## 7           7 209206
## 8           8 155267
## 9           9 169817
## 10         10 162987
## 11         11 229995
## 12         12 220395
## 13         13  75148
## 14         14 139828
## 15         15 147492
## 16         16 174411
## 17         17 227655
## 18         18  73578
## 19         19 206625
## 20         20  95771
## 21         21  49865
## 22         22  86047
## 23          X 136382
## 24          Y  14406
## 25         MT    144
## 26 GL000008.2    631
## 27 GL000009.2    403
## 28 GL000194.1    178
## 29 GL000195.1   1111
## 30 GL000205.2    987
## 31 GL000213.1     88
## 32 GL000214.1    576
## 33 GL000216.2      3
## 34 GL000218.1    961
## 35 GL000219.1    145
## 36 GL000220.1     58
## 37 GL000221.1    860
## 38 GL000224.1     39
## 39 GL000225.1     13
## 40 KI270442.1     15
## 41 KI270706.1    331
## 42 KI270710.1      5
## 43 KI270711.1     78
## 44 KI270712.1     31
## 45 KI270713.1    118
```

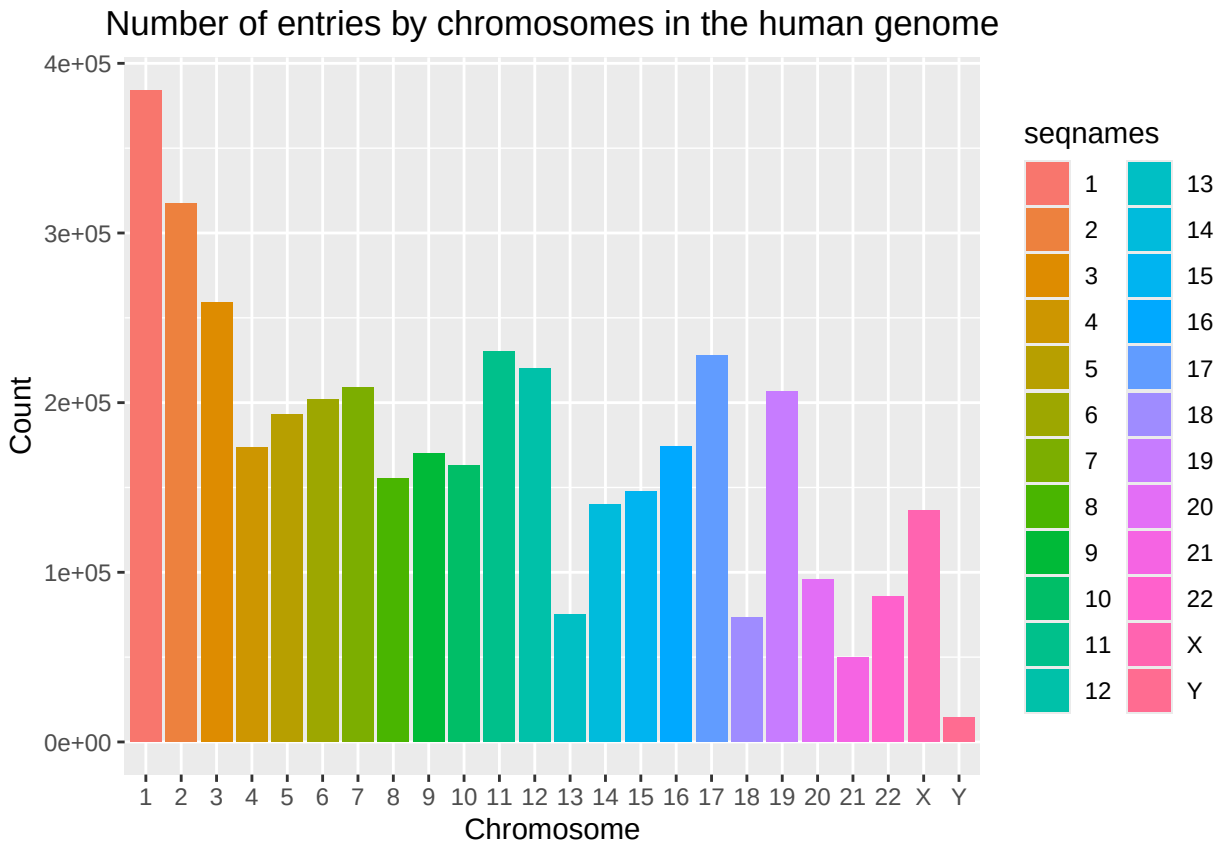
```
## 46 KI270714.1     14
## 47 KI270717.1      8
## 48 KI270718.1     24
## 49 KI270719.1     51
## 50 KI270720.1     18
## 51 KI270721.1    274
## 52 KI270722.1     19
## 53 KI270726.1     27
## 54 KI270727.1    345
## 55 KI270728.1    772
## 56 KI270731.1    220
## 57 KI270733.1     58
## 58 KI270734.1    149
## 59 KI270741.1     81
## 60 KI270742.1   2663
## 61 KI270743.1    143
## 62 KI270744.1    124
## 63 KI270745.1     29
## 64 KI270746.1     27
## 65 KI270748.1     21
## 66 KI270749.1     11
## 67 KI270750.1     63
## 68 KI270751.1    380
## 69 KI270753.1      4
## 70 KI270755.1      4
```

and here we plot those numbers

```
my_chr <- c(as.character(1:22), "X", "Y", "M")

# Convert to data frame first
df_chrs <- as.data.frame(my_obj) %>%
          dplyr::filter(seqnames %in% my_chr)

# Plot
ggplot(df_chrs, aes(x = seqnames, fill = seqnames)) +
  geom_bar() +
  labs(title = "Number of entries by chromosomes in the human genome",
       x = "Chromosome",
       y = "Count") +
 theme(plot.title = element_text(hjust = 0.5))
```


Number of entries by chromosomes in the human genome
Count
4e+05
3e+05
2e+05
1e+05
0e+00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y
Chromosome
seqnames
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
X
Y

Mitochondrial genome

Since this file contains also the mitochondrial genome we can filter it out to see which genes are mitochondrial genes.

```
my_mit <- my_obj %>%
    dplyr::filter(seqnames == "MT")

head(my_mit, 3)
```

```
## GRanges object with 3 ranges and 22 metadata columns:
##       seqnames    ranges strand |   source       type     score     phase
##          <Rle> <IRanges>  <Rle> | <factor>   <factor> <numeric> <integer>
##   [1]       MT   577-647      + |    insdc gene              NA      <NA>
##   [2]       MT   577-647      + |    insdc transcript        NA      <NA>
##   [3]       MT   577-647      + |    insdc exon              NA      <NA>
##               gene_id gene_version   gene_name gene_source gene_biotype
##           <character>  <character> <character> <character>  <character>
##   [1] ENSG00000210049            1       MT-TF       insdc      Mt_tRNA
##   [2] ENSG00000210049            1       MT-TF       insdc      Mt_tRNA
##   [3] ENSG00000210049            1       MT-TF       insdc      Mt_tRNA
##         transcript_id transcript_version transcript_name transcript_source
##           <character>        <character>     <character>       <character>
##   [1]            <NA>               <NA>            <NA>              <NA>
##   [2] ENST00000387314                  1       MT-TF-201             insdc
##   [3] ENST00000387314                  1       MT-TF-201             insdc
##       transcript_biotype              tag transcript_support_level exon_number
##              <character>      <character>              <character> <character>
##   [1]              <NA>             <NA>                      <NA>        <NA>
##   [2]           Mt_tRNA Ensembl_canonical                        NA        <NA>
##   [3]           Mt_tRNA Ensembl_canonical                        NA           1
##              exon_id exon_version  protein_id protein_version     ccds_id
##           <character>  <character> <character>     <character> <character>
##   [1]            <NA>         <NA>        <NA>            <NA>        <NA>
##   [2]            <NA>         <NA>        <NA>            <NA>        <NA>
##   [3] ENSE00001544501            1        <NA>            <NA>        <NA>
##   -------
##   seqinfo: 70 sequences from an unspecified genome; no seqlengths
```

```
length(my_mit)
```

```
## [1] 144
```

The mitochondrial genome contains at least 144 entries (protein coding genes most of all)! Fun fact, the human mitochondrial DNA was the first significant part of the human genome to be sequenced, and its genetic code differs slightly from nuclear DNA!

Also, since animal mitochondrial DNA evolves faster than nuclear genetic markers, it represents a mainstay of phylogenetic and evolutionary biology. It also permits tracing the relationships among populations, and so has become important reference in anthropology and biogeography.

Transcripts by type

We plot here the numbers of entries for each biotype

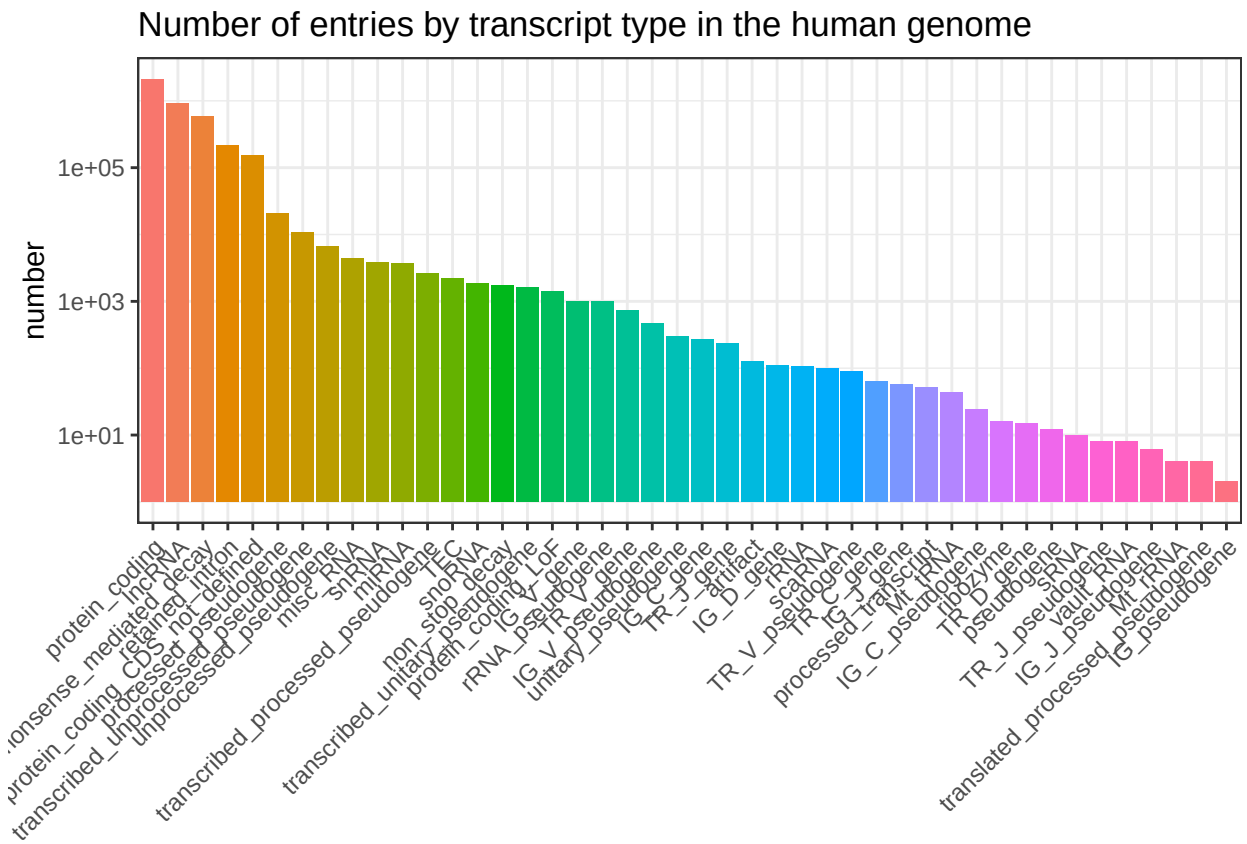
```
my_obj %>%
    group_by(transcript_biotype) %>%
    summarise(number = n()) %>%
    as.data.frame() -> my_biotypes

my_biotypes %>%
    dplyr::filter(!is.na(transcript_biotype)) -> my_biotypes

my_biotypes %>%
    arrange(desc(number)) %>%
    dplyr::pull(transcript_biotype) -> my_order

my_biotypes$transcript_biotype <- factor(my_biotypes$transcript_biotype, levels = my_order)

ggplot(my_biotypes, aes(x = transcript_biotype, y = number, fill = transcript_biotype)) +
  geom_col() +
  labs(title = "Number of entries by transcript type in the human genome") +
  scale_y_log10() +
  theme_bw() +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1),
    legend.position = "none"
  )
```



Working with genes, exons and CDS

Since the GTF file contains also the number of exons of each gene, we can count them.

```
my_obj %>%
    dplyr::filter(type == 'exon') %>%
    dplyr::filter(gene_biotype == 'protein_coding') %>%
    group_by(gene_name) %>%
    summarise(total = n()) %>%
    as.data.frame() -> my_counts
head(my_counts)
```

```
##   gene_name total
## 1      A1BG    40
## 2      A1CF   104
## 3       A2M    87
## 4     A2ML1   112
## 5   A3GALT2     5
## 6    A4GALT    17
```

curiosity kills the cat...

Which is the gene with the highest number of exon?

```
my_counts[my_counts$total==max(my_counts$total), ]
```

```
##       gene_name total
## 19450      <NA> 11614
```

Not always a gene name is present!

Let's group by gene id

```
my_obj %>%
    dplyr::filter(type == 'exon') %>%
    dplyr::filter(gene_biotype == 'protein_coding') %>%
    group_by(gene_id) %>%
    summarise(total = n()) %>%
    as.data.frame() -> my_counts
my_counts[my_counts$total==max(my_counts$total), ]
```

```
##              gene_id total
## 8569 ENSG00000145362  3886
```

Lets' use the gene id and the gene names

```
my_obj %>%
    dplyr::filter(type == 'exon') %>%
    dplyr::filter(gene_biotype == 'protein_coding') %>%
    group_by(gene_name, gene_id) %>%
    summarise(total = n()) %>%
    as.data.frame() -> my_counts
my_counts[my_counts$total==max(my_counts$total), ]
```

```
##     gene_name         gene_id total
## 677      ANK2 ENSG00000145362  3886
```

What is going on?

How many coding genes have only one exon!

```
my_counts %>% dplyr::filter(total == '1') %>% dim() %>% .[1]
```

```
## [1] 982
```

We can also obtain the length of the coding genes, and from this new list we can find the gene with the longest sequence.

```
my_obj %>%
    dplyr::filter(type == 'gene') %>%
    dplyr::filter(gene_biotype == 'protein_coding') -> my_lenghts

width_df <- as.data.frame(list(width=width(my_lenghts), gene_names=my_lenghts$gene_name))
```

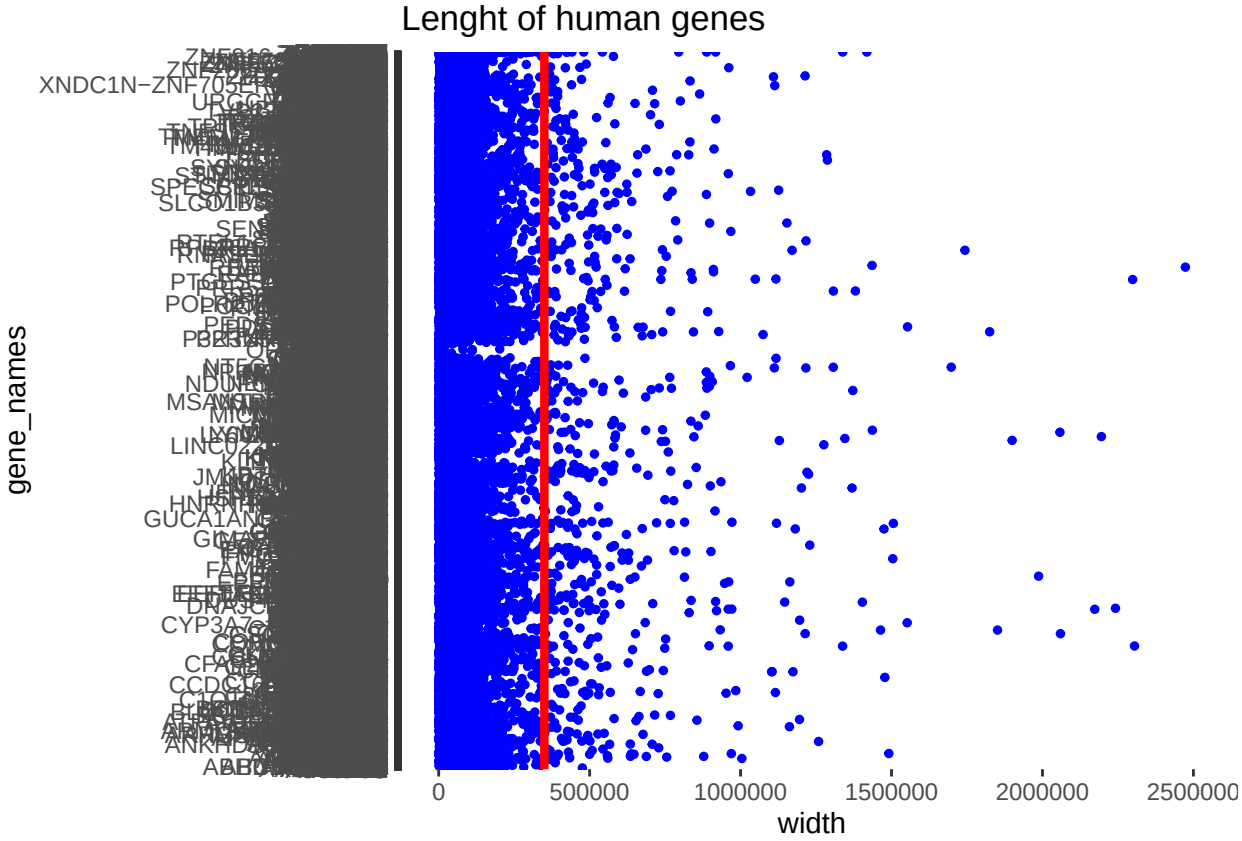
and the longest protein coding gene is:

```
width_df[width_df$width==max(width_df$width), ]
```

```
##         width gene_names
## 15699 2473539     RBFOX1
```

We can plot the length of coding genes (every blue dot is a gene).

```
ggplot(width_df, aes(x=width, y=gene_names)) +
    geom_point(colour = "blue", size = 1) +
    geom_vline(xintercept = 350000,
               color = "red", linewidth=1.5)+
  labs(title = "Lenght of human genes")
```




```
pl <- ggplot(width_df, aes(x = width)) +
      geom_freqpoly(bins=100) +
      geom_vline(xintercept = 350000,
                 color = "red", linewidth=0.8) +
      geom_vline(aes(xintercept = median(width)), color = "green")
```

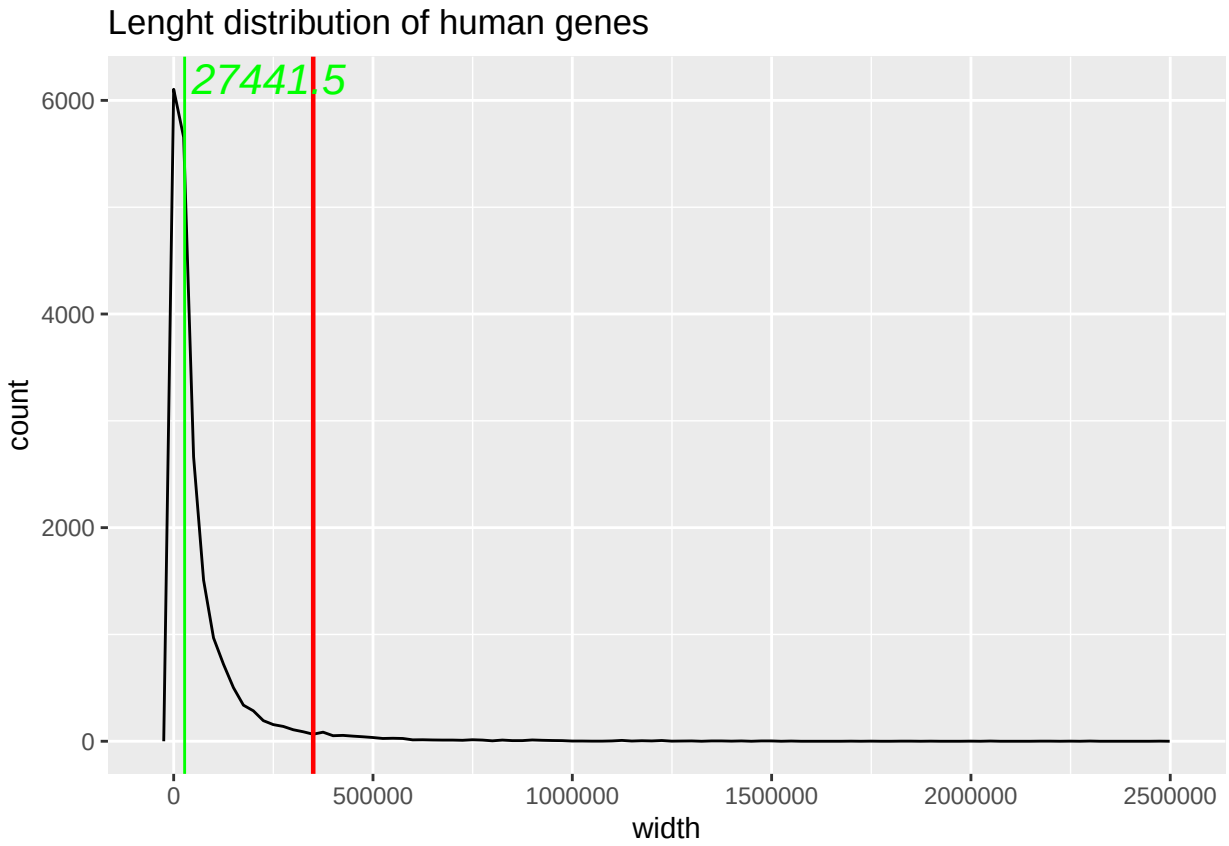
```
#median(my_lenghts$width) # 27475
```

The plot above is quite crowded because it shows the length of every single gene. Better to plot the gene length distribution.

```
pl <- ggplot(width_df, aes(x = width)) +
     geom_freqpoly(bins=100) +
     geom_vline(xintercept = 350000,
          color = "red", linewidth=0.8) +
      geom_vline(aes(xintercept = median(width)), color = "green")+
  labs(title = "Lenght distribution of human genes")

grob <- grobTree(textGrob((median(width(my_lenghts))), x=0.075,  y=0.97, hjust=0,
                          gp=gpar(col="green", fontsize=16, fontface="italic")))

pl + annotation_custom(grob)
```



The graph shows the distribution of the gene length; the green line is the median. As we can see, most genes are on the left of the red line, that indicate 350000 bp: most of the human coding genes are not very long.

The same workflow used for genes can be used for CDS lenght.

Let's plot:

```
my_obj %>%
    filter(type == 'CDS') %>%
    filter(gene_biotype == 'protein_coding') -> my_lenghts
```

```
width_df <- as.data.frame(list(width=width(my_lenghts), gene_names=my_lenghts$gene_name))
```

and the longest CDS is:

```
width_df[width_df$width==max(width_df$width), ]
```

```
##        width gene_names
## 863069 21693      MUC16
## 863156 21693      MUC16
## 863244 21693      MUC16
## 863330 21693      MUC16
```

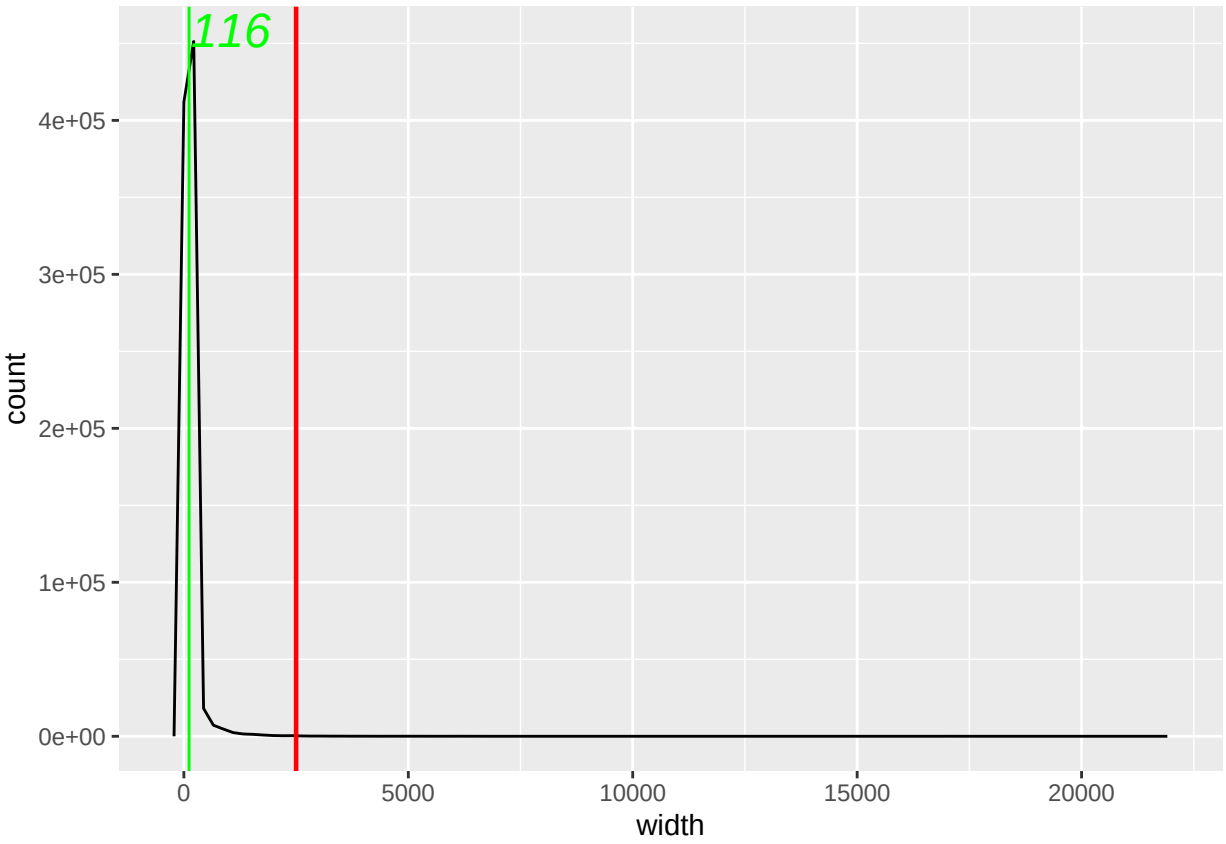
why more than one entries?

and now let's plot the CDS lenght distribution

```
pl <- ggplot(width_df, aes(x = width)) +
    geom_freqpoly(bins=100) +
    geom_vline(xintercept = 2500,
               color = "red", linewidth=0.8) +
    geom_vline(aes(xintercept = median(width)), color = "green")

grob <- grobTree(textGrob((median(width(my_lenghts))), x=0.065,  y=0.97, hjust=0,
                          gp=gpar(col="green", fontsize=18, fontface="italic")))

pl + annotation_custom(grob)
```



As expected the median of the CDS lenght is smaller than the gene lenght?

Can you explain why?