It seems that most people think Ensembl's GTF file and cDNA fasta file mean the same transcripts:

> Watch out! @ensembl's Fasta and GTF annotation files available via https://t.co/2AhCSnL7py do not match (there are transcripts in the GTF not found in the Fasta file. Anyone else expected them to match?
>
> — K. Vitting-Seerup (@KVittingSeerup) August 13, 2018

However, my colleagues Joseph Min and Sina Booeshaghi found that for several species, Ensembl's GTF file and cDNA fasta file do not have the same set of transcripts, so it would not be the same using the cDNA file as opposed to extracting the transcript sequences from the genome with the GTF file for a reference to pseudoalign RNA-seq reads. But how exactly does the GTF annotation differ from cDNA? This isn't very clear on the Ensembl website. In this blog post, I'll answer the following questions:

- What kind of genes do those non-overlapping transcripts belong to?
- For the transcripts present in both, do the GTF annotation and the cDNA fasta file mean the same sequences?

For now, I will analyze Ensembl's human genome annotations; I suspect that the same rule applies to other species as well, especially vertebrates.

```
library(tidyverse)
library(VennDiagram)
library(biomartr)
library(ggpubr)
library(BSgenome.Hsapiens.UCSC.hg38)
library(Biostrings)
library(plyranges)
library(GenomeInfoDb)
library(GenomicFeatures)
library(BUSpaRse)
library(here)
library(scales)
source(here("code", "plotting.R")) # See GitHub repo of this blog

# Download cDNA fasta file
if (!file.exists(here("reference", "hs_cdna99.fa.gz"))) {
  download.file("ftp://ftp.ensembl.org/pub/release-99/fasta/homo_sapiens/cdna/Homo_
sapiens.GRCh38.cdna.all.fa.gz",
                destfile = here("reference", "hs_cdna99.fa.gz"))
}

# Download GTF file
gtf_fn <- getGTF(db = "ensembl", organism = "Homo sapiens", path =
here("reference"))

#> Starting gtf retrieval of 'Homo sapiens' from ensembl ...

#>

#> File /Users/lambda/Documents/fs2s/reference/Homo_sapiens.GRCh38.
99_ensembl.gtf.gz exists already. Thus, download has been skipped.

#> The *.gtf annotation file of 'Homo sapiens' has been downloaded to '/Users
/lambda/Documents/fs2s/reference/Homo_sapiens.GRCh38.99_ensembl.gtf.gz' and has
been named 'Homo_sapiens.GRCh38.99_ensembl.gtf.gz'.

cdna <- readDNAStringSet(here("reference", "hs_cdna99.fa.gz"))
gtf <- read_gff(gtf_fn)
```

The sequence names in the Ensembl GTF file contain genome annotation information, which I'll compare to the corresponding GTF annotation.

```
head(names(cdna))
```

```
#> [1] "ENST00000434970.2 cdna chromosome:GRCh38:14:22439007:22439015:1
gene:ENSG00000237235.2 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene
gene_symbol:TRDD2 description:T cell receptor delta diversity 2 [Source:HGNC
Symbol;Acc:HGNC:12255]"
#> [2] "ENST00000415118.1 cdna chromosome:GRCh38:14:22438547:22438554:1
gene:ENSG00000223997.1 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene
gene_symbol:TRDD1 description:T cell receptor delta diversity 1 [Source:HGNC
Symbol;Acc:HGNC:12254]"
#> [3] "ENST00000448914.1 cdna chromosome:GRCh38:14:22449113:22449125:1
gene:ENSG00000228985.1 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene
gene_symbol:TRDD3 description:T cell receptor delta diversity 3 [Source:HGNC
Symbol;Acc:HGNC:12256]"
#> [4] "ENST00000631435.1 cdna chromosome:GRCh38:CHR_HSCHR7_
2_CTG6:142847306:142847317:1 gene:ENSG00000282253.1 gene_biotype:TR_D_gene
transcript_biotype:TR_D_gene gene_symbol:TRBD1 description:T cell receptor beta
diversity 1 [Source:HGNC Symbol;Acc:HGNC:12158]"
#> [5] "ENST00000632684.1 cdna chromosome:GRCh38:7:142786213:142786224:1
gene:ENSG00000282431.1 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene
gene_symbol:TRBD1 description:T cell receptor beta diversity 1 [Source:HGNC
Symbol;Acc:HGNC:12158]"
#> [6] "ENST00000390583.1 cdna chromosome:GRCh38:14:105904497:105904527:-1
gene:ENSG00000211923.1 gene_biotype:IG_D_gene transcript_biotype:IG_D_gene
gene_symbol:IGHD3-10 description:immunoglobulin heavy diversity 3-10
[Source:HGNC Symbol;Acc:HGNC:5495]"
```

```
head(gtf)
```

```
#> GRanges object with 6 ranges and 22 metadata columns:
#>       seqnames      ranges strand |   source       type     score     phase
#>                  |
#>   [1]        1 11869-14409      + |   havana       gene
#>   [2]        1 11869-14409      + |   havana transcript
#>   [3]        1 11869-12227      + |   havana       exon
#>   [4]        1 12613-12721      + |   havana       exon
#>   [5]        1 13221-14409      + |   havana       exon
#>   [6]        1 12010-13670      + |   havana transcript
#>                gene_id gene_version   gene_name gene_source
#>
#>   [1] ENSG00000223972            5     DDX11L1      havana
#>   [2] ENSG00000223972            5     DDX11L1      havana
#>   [3] ENSG00000223972            5     DDX11L1      havana
#>   [4] ENSG00000223972            5     DDX11L1      havana
#>   [5] ENSG00000223972            5     DDX11L1      havana
#>   [6] ENSG00000223972            5     DDX11L1      havana
#>                             gene_biotype   transcript_id transcript_version
#>
#>   [1] transcribed_unprocessed_pseudogene
#>   [2] transcribed_unprocessed_pseudogene ENST00000456328                 2
#>   [3] transcribed_unprocessed_pseudogene ENST00000456328                 2
#>   [4] transcribed_unprocessed_pseudogene ENST00000456328                 2
#>   [5] transcribed_unprocessed_pseudogene ENST00000456328                 2
#>   [6] transcribed_unprocessed_pseudogene ENST00000450305                 2
#>       transcript_name transcript_source               transcript_biotype
#>
```

```
#>   [1]
#>   [2]     DDX11L1-202            havana               processed_transcript
#>   [3]     DDX11L1-202            havana               processed_transcript
#>   [4]     DDX11L1-202            havana               processed_transcript
#>   [5]     DDX11L1-202            havana               processed_transcript
#>   [6]     DDX11L1-201            havana transcribed_unprocessed_pseudogene
#>               tag transcript_support_level exon_number         exon_id
#>
#>   [1]
#>   [2]       basic                        1
#>   [3]       basic                        1           1 ENSE00002234944
#>   [4]       basic                        1           2 ENSE00003582793
#>   [5]       basic                        1           3 ENSE00002312635
#>   [6]       basic                       NA
#>       exon_version  protein_id protein_version     ccds_id
#>
#>   [1]
#>   [2]
#>   [3]            1
#>   [4]            1
#>   [5]            1
#>   [6]
#>   -------
#>   seqinfo: 47 sequences from an unspecified genome; no seqlengths
```

```
# Extract transcript ID from fasta sequence name
cdna_tx <- str_extract(names(cdna), "^ENST\\d+")
# Transcript IDs from GTF
gtf_tx <- unique(gtf$transcript_id)
gtf_tx <- gtf_tx[!is.na(gtf_tx)]
length(cdna_tx)
```

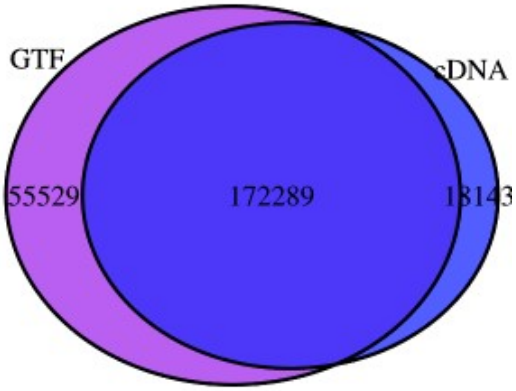
```
#> [1] 190432
```

```
length(gtf_tx)
```

```
#> [1] 227818
```

In total, there are 190432 transcripts in the fasta file, and 227818 in the GTF file.

```
v <- draw.pairwise.venn(length(gtf_tx), length(cdna_tx),
                        length(intersect(cdna_tx, gtf_tx)),
                        category = c("GTF", "cDNA"),
                        fill = c("purple", "blue"),
                        alpha = c(0.5, 0.5))
grid.draw(v)
```



```
grid.newpage()
```

While most transcripts overlap, a sizable minority don't.

It would not be so terrible if the transcripts that don't overlap between the GTF file and cDNA fasta file are all from genes most people don't care about, such as pseudogenes. Or would those genes be haplotype variants? Is this the case? Here I'll use Ensembl version 99, which is the most recent as of writing.

The Ensembl's FTP site has README files for each directory. For GTF files, the README file says

> GTF provides access to all annotated transcripts which make
> up an Ensembl gene set. Annotation is based on alignments of
> biological evidence (eg. proteins, cDNAs, RNA-seq) to a genome assembly.
> The annotation dumped here is transcribed and translated from the
> genome assembly and is not the original input sequence data that
> we used for alignment. Therefore, the sequences provided by Ensembl
> may differ from the original input sequence data where the genome
> assembly is different to the aligned sequence.

For cDNA files, the README says:

> These files hold the cDNA sequences corresponding to Ensembl gene
> predictions. cDNA consists of transcript sequences for actual and possible
> genes, including pseudogenes, NMD and the like. See the file names
> explanation below for different subsets of both known and predicted
> transcripts.
>
> FILE NAMES
>
> The files are consistently named following this pattern:
> ....fa.gz
>
> : The systematic name of the species.
>
> : The assembly build name.
>
> : cdna for cDNA sequences
>
> :
>
> - 'cdna.all' – the super-set of all transcripts resulting from
>   Ensembl gene predictions (see more below).
> - 'cdna.abinitio' – transcripts resulting from 'ab initio' gene prediction
>   algorithms such as SNAP and GENSCAN. In general all 'ab initio'
>   predictions are solely based on the genomic sequence and do not
>   use other experimental evidence. Therefore, not all GENSCAN or SNAP
>   cDNA predictions represent biologically real cDNAs.
>   Consequently, these predictions should be used with care.

The one I used is `Homo_sapiens.GRCh38.cdna.all.fa.gz`, not the `abinitio` one. However, the README doesn't seem to be clear about how the GTF annotation differs from that in the cDNA fasta file. Here I'll find out about such differences.

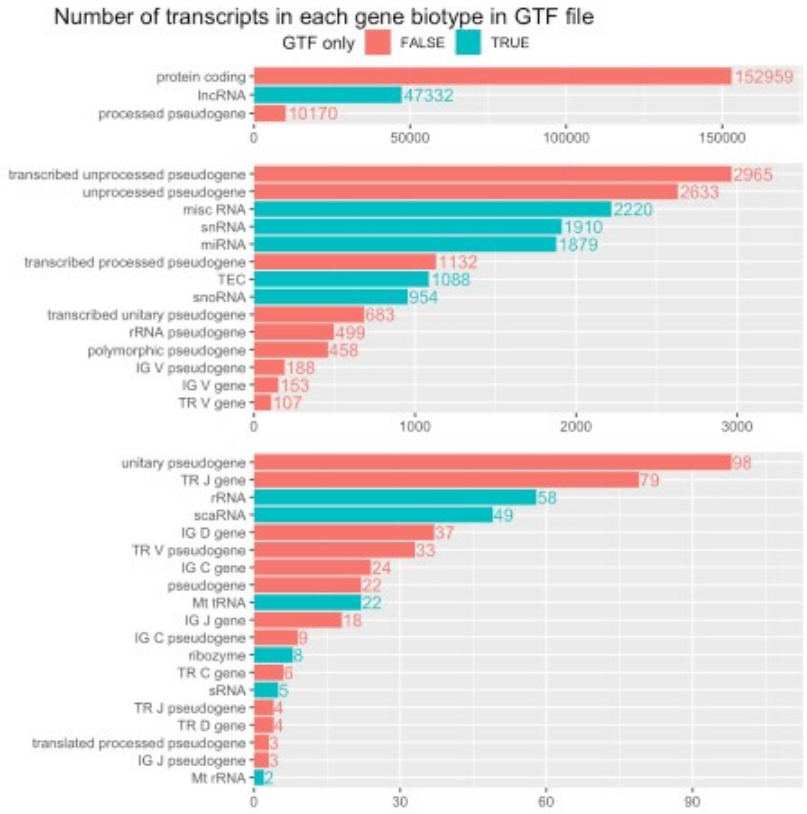
GTF only

```
gtf_meta <- as.data.frame(gtf[gtf$type == "transcript"])
gtf_meta <- gtf_meta %>%
  mutate(gtf_only = !transcript_id %in% cdna_tx,
         gene_biotype = str_replace_all(gene_biotype, "_", " "))

n_txs <- gtf_meta %>%
```

```
  count(gtf_only, gene_biotype)
```

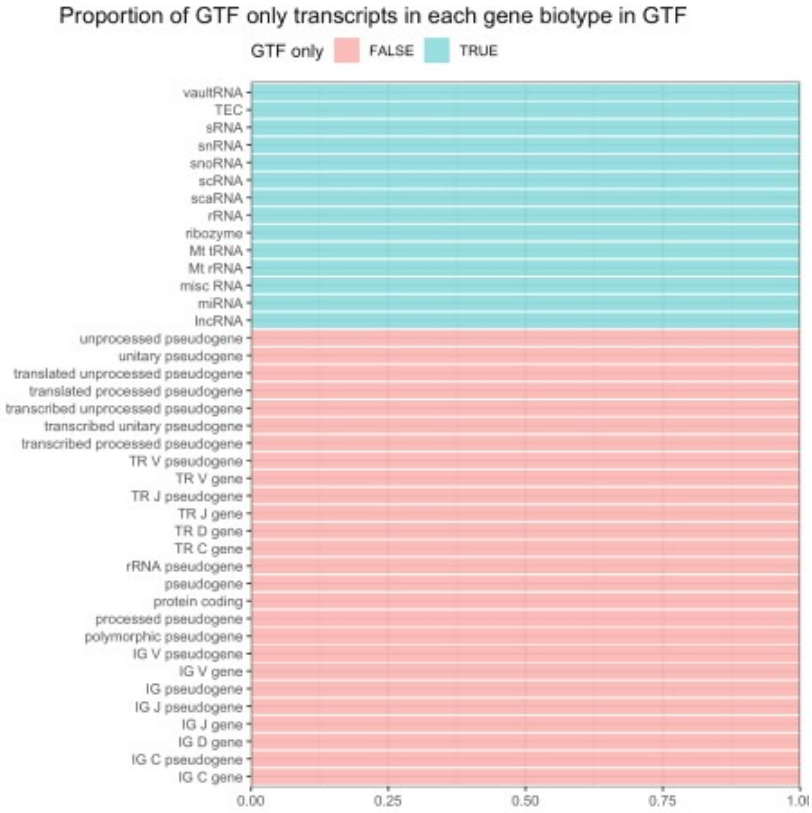
How many transcripts are there in each gene biotype, and how many transcripts in each biotype are only in the GTF file? For a description of Ensembl gene biotypes, see this page.

```
plot_bar_patch(n_txs, 3, "gtf_only", "GTF only",
               "Number of transcripts in each gene biotype in GTF file")
```



Proportion of GTF only transcripts in each biotype

```
p <- ggplot(gtf_meta, aes(fct_reorder(gene_biotype, gtf_only, .fun = mean), fill
= gtf_only)) +
  geom_bar(position = "fill", alpha = 0.5) +
  scale_y_continuous(expand = expand_scale(mult = c(0, 0))) +
  scale_fill_discrete(name = "GTF only") +
  coord_flip() + theme_bw() +
  theme(legend.position = "top", legend.justification = c(0,0.5),
        legend.margin = margin(t = 14), axis.title = element_blank())
# To place title further to the left; will be fixed in ggplot2 devel
annotate_figure(p, fig.lab = "Proportion of GTF only transcripts in each gene
biotype in GTF",
                fig.lab.pos = "top.left", fig.lab.size = 14)
```



It's now apparent that some transcripts are only present in the GTF file because their biotypes are excluded from the cDNA file. These GTF only biotypes are non-coding RNAs, except TEC, which stands for *To be Experimentally Confirmed*. However, Ensembl has a separately fasta file for lncRNA. Some non-coding RNAs are not polyadenylated (e.g. mature miRNAs), which means they are omitted by polyA selection prior to RNA-seq. However, some lncRNAs are polyadenylated, and Cell Ranger's reference does include lincRNA (long intergenic non-coding RNA).

cDNA fasta only

What about cDNA only transcripts? Are they also from specific gene biotypes?

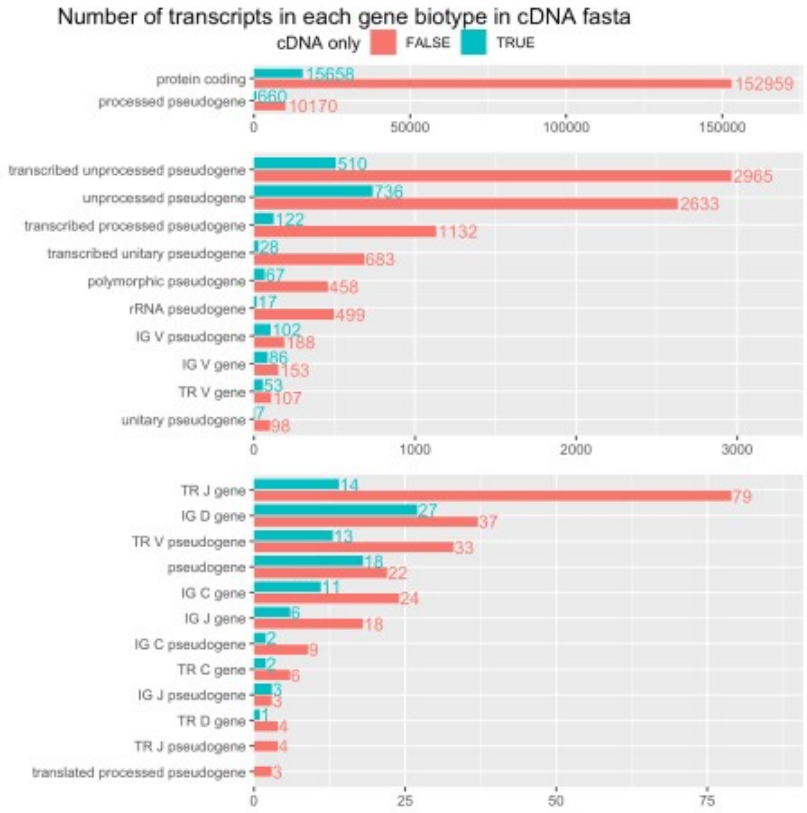
```
# Extract annotation from fasta sequence names
cdna_meta <- tibble(transcript_id = cdna_tx,
                    cr = str_extract(names(cdna),
                                     "(?<=((chromosome)|(scaffold)):GRCh38:).*?
(?=\\s)"),
                    gene_biotype = str_extract(names(cdna),
"(?<=gene_biotype:).*?(?=\\s)"),
                    gene_id = str_extract(names(cdna), "(?<=gene:).*?(?=\\.)"),
                    gene_symbol = str_extract(names(cdna), "(?<=gene_symbol:).*?
(?=\\s)"),
                    cdna_only = !transcript_id %in% gtf_tx) %>%
  separate(cr, into = c("seqnames", "start", "end", "strand"), sep = ":") %>%
  mutate(start = as.integer(start),
         end = as.integer(end),
         strand = case_when(
           strand == "1" ~ "+",
           strand == "-1" ~ "-",
           TRUE ~ "*"
         ),
         gene_biotype = str_replace_all(gene_biotype, "_", " "))

head(cdna_meta)

#> # A tibble: 6 x 9
```

```
#>   transcript_id seqnames  start    end strand gene_biotype gene_id
gene_symbol
#>
#> 1 ENST00000434… 14       2.24e7 2.24e7 +      TR D gene    ENSG00… TRDD2
#> 2 ENST00000415… 14       2.24e7 2.24e7 +      TR D gene    ENSG00… TRDD1
#> 3 ENST00000448… 14       2.24e7 2.24e7 +      TR D gene    ENSG00… TRDD3
#> 4 ENST00000631… CHR_HSC… 1.43e8 1.43e8 +      TR D gene    ENSG00… TRBD1
#> 5 ENST00000632… 7        1.43e8 1.43e8 +      TR D gene    ENSG00… TRBD1
#> 6 ENST00000390… 14       1.06e8 1.06e8 -      IG D gene    ENSG00… IGHD3-10
#> # … with 1 more variable: cdna_only
```

```
n_txs_cdna <- cdna_meta %>%
  count(cdna_only, gene_biotype)
```

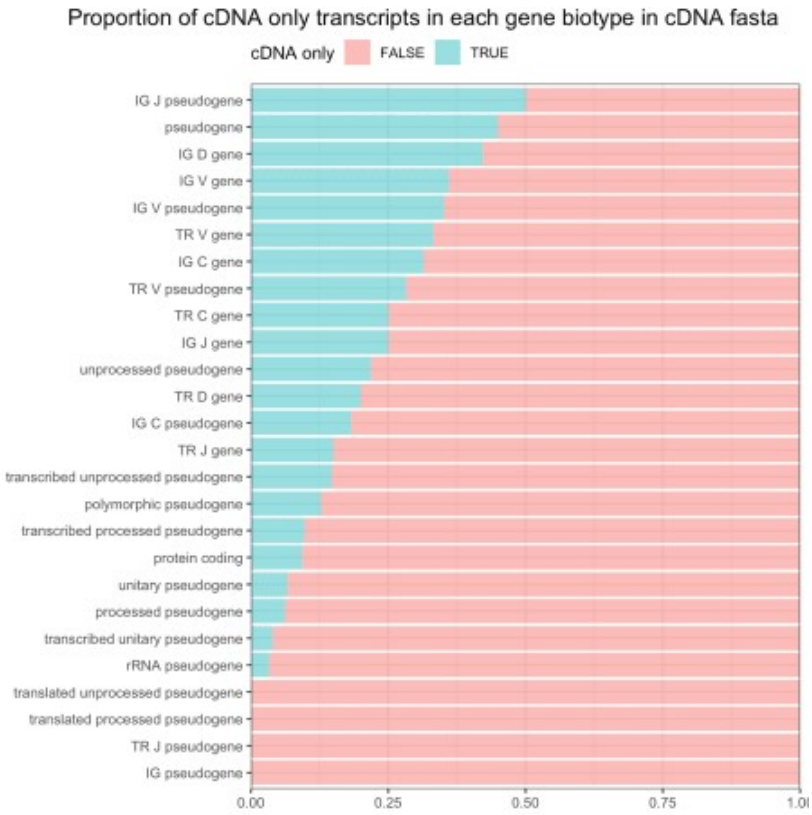
Number of transcripts in each biotype and number within each biotype that is only in the fasta file

```
plot_bar_patch(n_txs_cdna, 3, col_fill = "cdna_only", name = "cDNA only",
               title = "Number of transcripts in each gene biotype in cDNA
fasta")
```



Proportion of transcripts that are only in the fasta file in each biotype

```
p <- ggplot(cdna_meta, aes(fct_reorder(gene_biotype, cdna_only, .fun = mean),
fill = cdna_only)) +
  geom_bar(position = "fill", alpha = 0.5) +
  scale_y_continuous(expand = expand_scale(mult = c(0, 0))) +
  scale_fill_discrete(name = "cDNA only") +
  coord_flip() + theme_bw() +
  theme(legend.position = "top", legend.justification = c(0,0.5),
        legend.margin = margin(t = 14), axis.title = element_blank())
annotate_figure(p,
                fig.lab = "Proportion of cDNA only transcripts in each gene
biotype in cDNA fasta",
                fig.lab.pos = "top.left", fig.lab.size = 14)
```



Apparently, cDNA fasta only transcripts are not specific to a particular biotype.

Chromosomes

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

Gene annotations often contain information of not only the chromosomes, but also scaffolds.

```
seqlevels(gtf)
```

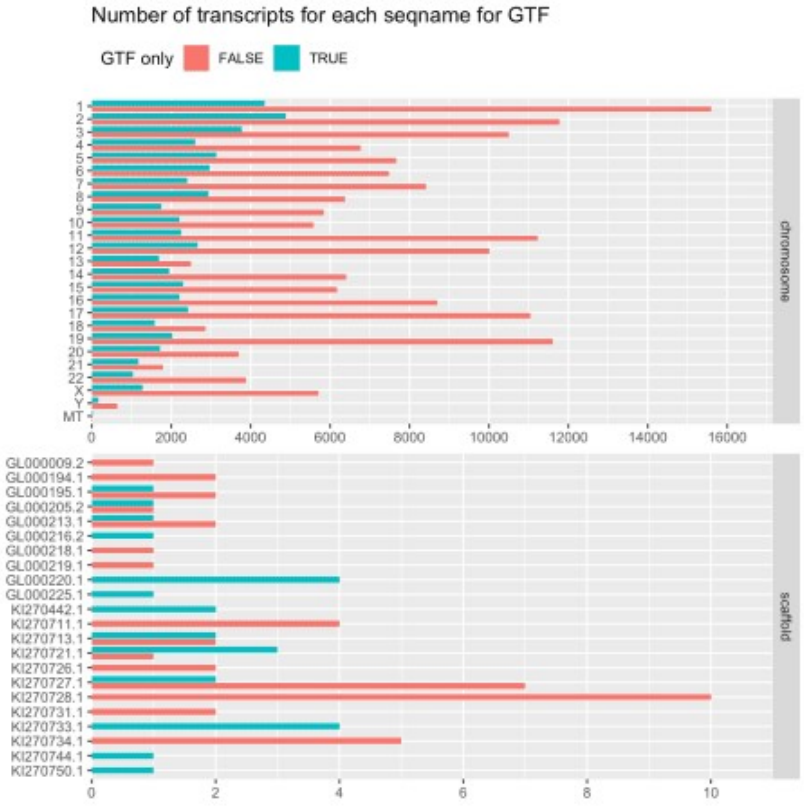
```
#>  [1] "1"          "2"          "3"          "4"          "5"
#>  [6] "6"          "7"          "8"          "9"          "10"
#> [11] "11"         "12"         "13"         "14"         "15"
#> [16] "16"         "17"         "18"         "19"         "20"
#> [21] "21"         "22"         "X"          "Y"          "MT"
#> [26] "GL000009.2" "GL000194.1" "GL000195.1" "GL000205.2" "GL000213.1"
#> [31] "GL000216.2" "GL000218.1" "GL000219.1" "GL000220.1" "GL000225.1"
#> [36] "KI270442.1" "KI270711.1" "KI270713.1" "KI270721.1" "KI270726.1"
#> [41] "KI270727.1" "KI270728.1" "KI270731.1" "KI270733.1" "KI270734.1"
#> [46] "KI270744.1" "KI270750.1"
```

The GL* and KI* things are scaffolds, which are regions not assembled into chromosomes. Genomes, such as `BSgenome.Hsapiens.UCSC.hg38` and Ensembl's top level genome (`Homo_sapiens.GRCh38.dna.toplevel.fa.gz`, downloaded by `biomartr::getGenome`), also contain haplotype information. Sometimes multiple Ensembl IDs correspond to the same gene symbol, as those Ensembl IDs correspond to different haplotypes. In contrast, `Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz` does not have the scaffolds and haplotypes.

Are the non-overlapping transcripts only on haplotypes or scaffolds?

```
gtf_meta %>%
  mutate(seqname_type = case_when(
    seqnames %in% chrs ~ "chromosome",
    str_detect(seqnames, "^CHR_") ~ "haplotype",
    TRUE ~ "scaffold"
```

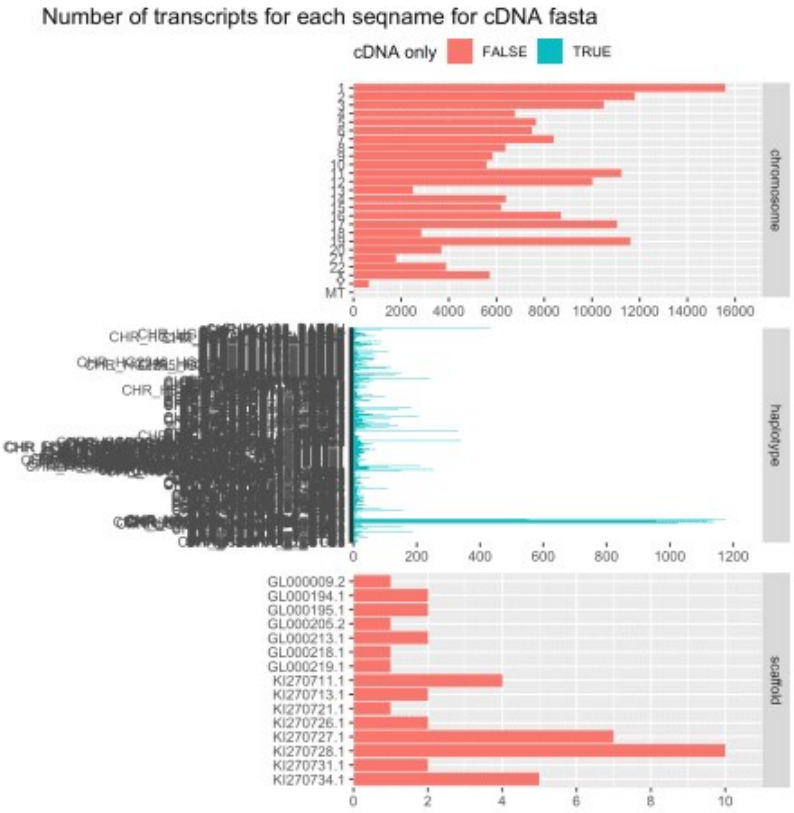
```
  )) %>%
  ggplot(aes(fct_rev(seqnames), fill = gtf_only)) +
  geom_bar(position = position_dodge2(width = 0.9, preserve = "single")) +
  scale_fill_discrete(name = "GTF only") +
  coord_flip() +
  facet_wrap(~ seqname_type, scales = "free", ncol = 1, strip.position =
"right") +
  scale_y_continuous(expand = expand_scale(mult = c(0, 0.1)),
                     breaks = pretty_breaks(n = 7)) +
  labs(title = "Number of transcripts for each seqname for GTF") +
  theme(legend.position = "top", legend.justification = c(0,0.5),
        axis.title = element_blank())
```



Apparently GTF only transcripts are not specific to scaffolds or chromosomes, though some scaffolds have a small number of genes, all of which are GTF only. What about in the cDNA file?

```
cdna_meta <- cdna_meta %>%
  mutate(
    seqname_type = case_when(
      seqnames %in% chrs ~ "chromosome",
      str_detect(seqnames, "^CHR") ~ "haplotype",
      TRUE ~ "scaffold"
    ),
    seqnames = fct_relevel(seqnames, c(chrs, setdiff(unique(seqnames), chrs) %>%
sort()))
  )

p <- ggplot(cdna_meta, aes(fct_rev(seqnames), fill = cdna_only)) +
  geom_bar(position = position_dodge2(width = 0.9, preserve = "single")) +
  coord_flip() +
  scale_fill_discrete(name = "cDNA only") +
  facet_wrap(~ seqname_type, scales = "free", ncol = 1, strip.position =
"right") +
  scale_y_continuous(expand = expand_scale(mult = c(0, 0.1)),
```

```
                    breaks = pretty_breaks(n = 7)) +
  theme(legend.position = "top", legend.justification = c(0,0.5),
        legend.margin = margin(t = 14), axis.title = element_blank())
annotate_figure(p, fig.lab = "Number of transcripts for each seqname for cDNA
fasta",
                fig.lab.pos = "top.left", fig.lab.size = 14)
```




```
cdna_meta %>%
  count(cdna_only, seqname_type) %>%
  arrange(desc(cdna_only), desc(n)) %>% knitr::kable()
```

cdna_only	seqname_type	n
TRUE	haplotype	18143
FALSE	chromosome	172246
FALSE	scaffold	43

There’re hundreds of haplotypes here. All the cDNA only transcripts are on haplotypes. As haplotypes can confuse alignment, for the purpose of aligning RNA-seq reads to the genome, haplotypes should better be excluded.

How about the transcripts shared between GTF and cDNA? Do those two sources mean the same sequence for the same transcript?

```
inter <- gtf_meta %>%
  inner_join(cdna_meta, by = c("gene_id", "transcript_id", "seqnames"))
```

```
#> Warning: Column `seqnames` joining factors with different levels, coercing to
#> character vector
```

Do the GTF and cDNA files place the same transcripts at the same genomic ranges?

```
all.equal(inter$start.x, inter$start.y)
```

```
#> [1] TRUE
```

```
all.equal(inter$end.x, inter$end.y)
```

```
#> [1] TRUE
```

```
all.equal(as.character(inter$strand.x), as.character(inter$strand.y))
```

```
#> [1] TRUE
```

```
all.equal(inter$gene_biotype.x, inter$gene_biotype.y)
```

```
#> [1] TRUE
```

So the genomic ranges, strand, and gene biotypes do match. However, this is just for transcripts; exon annotations are absent from the sequence names of the cDNA fasta file. Are the exons also the same?

```
unique(inter$seqnames)
```

```
#>  [1] "1"          "2"          "3"          "4"          "5"
#>  [6] "6"          "7"          "X"          "8"          "9"
#> [11] "11"         "10"         "12"         "13"         "14"
#> [16] "15"         "16"         "17"         "18"         "20"
#> [21] "19"         "Y"          "22"         "21"         "MT"
#> [26] "KI270728.1" "KI270727.1" "GL000009.2" "GL000194.1" "GL000205.2"
#> [31] "GL000195.1" "GL000219.1" "KI270734.1" "GL000213.1" "GL000218.1"
#> [36] "KI270731.1" "KI270721.1" "KI270726.1" "KI270711.1" "KI270713.1"
```

Say we don't care about the scaffolds. I'll extract the transcriptome (only for genes also present in the cDNA fasta file) using the GTF file. `BSgenome.Hsapiens.UCSC.hg38` denotes chromosomes as something like `chr1`, while Ensembl just uses `1`, so I'll convert `BSgenome.Hsapiens.UCSC.hg38` to Ensembl style.

```
gn <- BSgenome.Hsapiens.UCSC.hg38
seqlevelsStyle(gn) <- "Ensembl"

# This will discard scaffolds
gl <- BUSpaRse:::subset_annot(gn, gtf)
```

```
#> 22 sequences in the annotation absent from the genome were dropped.
```

```
#> 430 sequences in the genome are absent from the annotation.
```

```
# Only keep overlapping transcripts
gl <- gl[gl$type == "exon" & gl$transcript_id %in% inter$transcript_id]
# Exons are already sorted in ascending order in the GTF file, even for minus
strand genes
# Need to sort if not already sorted
gl <- split(gl, gl$transcript_id)
# Extract transcriptome
tx_gtf <- extractTranscriptSeqs(gn, gl)

cdna_compare <- cdna
names(cdna_compare) <- cdna_meta$transcript_id
# sort transcripts from the cDNA file, discard scaffolds
cdna_compare <- cdna_compare[names(tx_gtf)]
```

From the cDNA fasta:

```
cdna_compare
```

```
#>   A DNAStringSet instance of length 172246
#>           width seq                                               names
#>      [1]   1032 CTGCTGCTGCTGCGCCCCAT...TAAATTTGCTGTGGTTTGTA ENST00000000233
#>      [2]   2450 AGAGTGGGGCACAGCGAGGC...TAAAAAACAAACAAAACATA ENST00000000412
#>      [3]   2274 GTCAGCTGGAGGAAGCGGAG...TATAATACCGAGCTCAAAAA ENST00000000442
#>      [4]   3715 CCTACCCCAGCTCTCGCGCC...GTGAGGATGTTTTGTTAAAA ENST00000001008
```

```
#>      [5]   4732 AGGCAATTTTTTTCCTCCCT...AATAAACCGTGGGGACCCGC ENST00000001146
#>      ...    ... ...
#> [172242]   4105 TAGATGTAACCCTGAGTGAA...AATCACAATTCTGCTAATGT ENST00000674151
#> [172243]   1374 AGGCTGATAAAATACCAGTA...TGAGCACGATGATGATGCAA ENST00000674152
#> [172244]   2789 CCTGCGCAGAGTCTGCGGAG...AAAATGAGCAAAAGTTGATC ENST00000674153
#> [172245]   8288 ATGGCCGAGAATGTGGTGGA...TAAACTGTGTGAGACAGACA ENST00000674155
#> [172246]    898 TCTCTGGATATGAGGCAGGA...ACTCAATTTGTTATTCAAAA ENST00000674156
```

Sequences extracted from genome with GTF file:

```
tx_gtf
```

```
#>   A DNAStringSet instance of length 172246
#>           width seq                                          names
#>      [1]   1032 CTGCTGCTGCTGCGCCCCAT...TAAATTTGCTGTGGTTTGTA ENST00000000233
#>      [2]   2450 AGAGTGGGGCACAGCGAGGC...TAAAAAACAAACAAAACATA ENST00000000412
#>      [3]   2274 GTCAGCTGGAGGAAGCGGAG...TATAATACCGAGCTCAAAAA ENST00000000442
#>      [4]   3715 CCTACCCCAGCTCTCGCGCC...GTGAGGATGTTTTGTTAAAA ENST00000001008
#>      [5]   4732 AGGCAATTTTTTTCCTCCCT...AATAAACCGTGGGGACCCGC ENST00000001146
#>      ...    ... ...
#> [172242]   4105 TAGATGTAACCCTGAGTGAA...AATCACAATTCTGCTAATGT ENST00000674151
#> [172243]   1374 AGGCTGATAAAATACCAGTA...TGAGCACGATGATGATGCAA ENST00000674152
#> [172244]   2789 CCTGCGCAGAGTCTGCGGAG...AAAATGAGCAAAAGTTGATC ENST00000674153
#> [172245]   8288 ATGGCCGAGAATGTGGTGGA...TAAACTGTGTGAGACAGACA ENST00000674155
#> [172246]    898 TCTCTGGATATGAGGCAGGA...ACTCAATTTGTTATTCAAAA ENST00000674156
```

Do the transcript sequences at least have the same lengths?

```
all.equal(width(tx_gtf), width(cdna_compare))
```

```
#> [1] TRUE
```

Are the sequences the same? Since I don't care how the sequences are different if they are different, no alignment is needed.

```
all(pcompare(tx_gtf, cdna_compare) == 0)
```

```
#> [1] TRUE
```

Yes, the sequences are the same.

The GTF file contains annotations for non-coding RNAs, while the cDNA fasta file does not. The cDNA file contains haplotypes, while the GTF file does not. For pseudoalignment of RNA-seq reads from polyA selected techniques, non-coding RNAs in the GTF file probably aren't so important, unless you do care about polyadenylated lncRNAs, so it's fine to use the cDNA fasta file, but we should remove the haplotypes as they may cause confusion in alignment. However, if you are interested in non-coding RNAs, then download the ncRNA fasta file from Ensembl or extract the transcriptome with the GTF file. We've also got example R code here to filter by gene biotypes and to extract transcriptome from the genome with the GTF file.

```
sessionInfo()
```

```
#> R version 3.6.2 (2019-12-12)
#> Platform: x86_64-apple-darwin15.6.0 (64-bit)
#> Running under: macOS Catalina 10.15.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib
/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib
/libRlapack.dylib
```

```
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
#>  [1] stats4    parallel  grid      stats     graphics  grDevices utils
#>  [8] datasets  methods   base
#>
#> other attached packages:
#>  [1] rlang_0.4.4                      scales_1.1.0
#>  [3] here_0.1                         BUSpaRse_1.0.0
#>  [5] GenomicFeatures_1.38.1           AnnotationDbi_1.48.0
#>  [7] Biobase_2.46.0                   plyranges_1.6.8
#>  [9] BSgenome.Hsapiens.UCSC.hg38_1.4.1 BSgenome_1.54.0
#> [11] rtracklayer_1.46.0               Biostrings_2.54.0
#> [13] XVector_0.26.0                   GenomicRanges_1.38.0
#> [15] GenomeInfoDb_1.22.0              IRanges_2.20.2
#> [17] S4Vectors_0.24.3                 BiocGenerics_0.32.0
#> [19] ggpubr_0.2.4                     magrittr_1.5
#> [21] biomartr_0.9.2                   VennDiagram_1.6.20
#> [23] futile.logger_1.4.3              forcats_0.4.0
#> [25] stringr_1.4.0                    dplyr_0.8.4
#> [27] purrr_0.3.3                      readr_1.3.1
#> [29] tidyr_1.0.2                      tibble_2.1.3
#> [31] ggplot2_3.2.1                    tidyverse_1.3.0
#>
#> loaded via a namespace (and not attached):
#>  [1] colorspace_1.4-1            ggsignif_0.6.0
#>  [3] ellipsis_0.3.0              rprojroot_1.3-2
#>  [5] fs_1.3.1                    rstudioapi_0.10
#>  [7] farver_2.0.3                bit64_0.9-7
#>  [9] fansi_0.4.1                 lubridate_1.7.4
#> [11] xml2_1.2.2                  knitr_1.27
#> [13] zeallot_0.1.0               jsonlite_1.6.1
#> [15] Rsamtools_2.2.1             broom_0.5.4
#> [17] dbplyr_1.4.2                compiler_3.6.2
#> [19] httr_1.4.1                  backports_1.1.5
#> [21] assertthat_0.2.1            Matrix_1.2-18
#> [23] lazyeval_0.2.2              cli_2.0.1
#> [25] formatR_1.7                 htmltools_0.4.0
#> [27] prettyunits_1.1.1           tools_3.6.2
#> [29] gtable_0.3.0                glue_1.3.1
#> [31] GenomeInfoDbData_1.2.2      rappdirs_0.3.1
#> [33] Rcpp_1.0.3                  cellranger_1.1.0
#> [35] vctrs_0.2.2                 nlme_3.1-144
#> [37] blogdown_0.17               xfun_0.12
#> [39] rvest_0.3.5                 lifecycle_0.1.0
#> [41] ensembldb_2.10.2            XML_3.99-0.3
#> [43] zlibbioc_1.32.0             ProtGenerics_1.18.0
#> [45] hms_0.5.3                   SummarizedExperiment_1.16.1
#> [47] AnnotationFilter_1.10.0     lambda.r_1.2.4
#> [49] yaml_2.2.1                  curl_4.3
#> [51] gridExtra_2.3               memoise_1.1.0
#> [53] biomaRt_2.42.0              stringi_1.4.5
#> [55] RSQLite_2.2.0               highr_0.8
#> [57] BiocParallel_1.20.1         pkgconfig_2.0.3
#> [59] bitops_1.0-6                matrixStats_0.55.0
```

```
#> [61] evaluate_0.14              lattice_0.20-38
#> [63] labeling_0.3               GenomicAlignments_1.22.1
#> [65] cowplot_1.0.0              bit_1.1-15.1
#> [67] tidyselect_1.0.0           bookdown_0.17
#> [69] R6_2.4.1                   generics_0.0.2
#> [71] DelayedArray_0.12.2        DBI_1.1.0
#> [73] pillar_1.4.3               haven_2.2.0
#> [75] withr_2.1.2                RCurl_1.98-1.1
#> [77] modelr_0.1.5               crayon_1.3.4
#> [79] futile.options_1.0.1       utf8_1.1.4
#> [81] BiocFileCache_1.10.2       rmarkdown_2.1
#> [83] progress_1.2.2             readxl_1.3.1
#> [85] data.table_1.12.8          blob_1.2.1
#> [87] reprex_0.3.0               digest_0.6.23
#> [89] openssl_1.4.1              RcppParallel_4.4.4
#> [91] munsell_0.5.0              askpass_1.1
```