Generate GRCh38 refcoding bed

Nate Olson

2020-02-20

1 Summary

Generating a functional GRCh38 region stratification bed files for use in variant benchmarking.

2 Loading packages and defining variables

```r
library(tidyverse)

ftbl_path <- paste0("ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq",
                    "/vertebrate_mammalian/Homo_sapiens/",
                    "all_assembly_versions/GCF_000001405.39_GRCh38.p13/",
                    "GCF_000001405.39_GRCh38.p13_feature_table.txt.gz")
gff_path <- paste0("ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq",
                  "/vertebrate_mammalian/Homo_sapiens/",
                  "all_assembly_versions/GCF_000001405.39_GRCh38.p13/",
                  "GCF_000001405.39_GRCh38.p13_genomic.gff.gz")
faidx_path <- paste0("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405",
                      "/GCA_000001405.15_GRCh38/",
                      "seqs_for_alignment_pipelines.ucsc_ids/", "GCA_000001405.15_GRCh38_no_alt_analysi
```

3 Extracting CDS from RefSeq GFF

3.1 Getting Chromosome Assessions

Using feature table to extract chromosome assessions. Feature table downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/vertebrate_mammalian/Homo_sapiens/all_assembly_versions/GCF_000001405.39_GRCh38.p13/GCF_000001405.39_GRCh38.p13_feature_table.txt.gz.

```r
ftbl_md5 <- "c0587443810e3d62da505fb653c28dd3"

ftbl_file <- "GCF_000001405.39_GRCh38.p13_feature_table.txt.gz"
if (!file.exists(ftbl_file)) {
  download.file(url = ftbl_path, destfile = ftbl_file)

  ## MD5 check
  dwn_md5 <- tools::md5sum(ftbl_file)

  if(ftbl_md5 != dwn_md5){
    warning("MD5 for downloaded feature table does not match expected MD5")
  }
}
```

```r
## defining column types and names
ftbl_col_names <- read_lines(ftbl_file, n_max = 1) %>%
  str_split(pattern = "\t") %>% unlist()
ftbl_col_types <- "ccccccccddcclcccdlddc"

feature_table <- read_tsv(ftbl_file, comment = "#",
                          col_names = ftbl_col_names,
                          col_types = ftbl_col_types)

chrom_accn_df <- feature_table %>%
  dplyr::select(chromosome, genomic_accession, assembly_unit, seq_type) %>%
  distinct()
```

3.2 Downloading and Parsing GFF File

Extracting RefSeq CDS from GFF file. GFF file downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/vertebrate_mammalian/Homo_sapiens/all_assembly_versions/GCF_000001405.39_GRCh38.p13/GCF_000001405.39_GRCh38.p13_genomic.gff.gz.

```r
gff_md5 <- "e84f4e0102b6c4f4cc0035b8abf1038f"

gff_file <- "GCF_000001405.39_GRCh38.p13_genomic.gff.gz"
if (!file.exists(gff_file)) {
  download.file(url = gff_path, destfile = gff_file)

  ## MD5 check
  dwn_md5 <- tools::md5sum(gff_file)

  if(gff_md5 != dwn_md5){
    warning("MD5 for downloaded feature table does not match expected MD5")
  }
}

## defining column types and names
gff_table <- read_tsv(gff_file, col_names = c("seqid", "source", "type", "start", "end",
                              "score", "strand", "phase", "attributes"),
                comment = "#")
```

```
## Parsed with column specification:
## cols(
##   seqid = col_character(),
##   source = col_character(),
##   type = col_character(),
##   start = col_double(),
##   end = col_double(),
##   score = col_character(),
##   strand = col_character(),
##   phase = col_character(),
##   attributes = col_character()
## )
```

```r
## Exon table with only RefSeq entries
exon_table <- gff_table %>%
  filter(type == "CDS",
         grepl("RefSeq", source)) %>%
  rename(genomic_accession = seqid) %>%
  left_join(chrom_accn_df)
```

```
## Joining, by = "genomic_accession"
```

```
## Extracting CDS for chromosomes and converting to 3 column table.
exon_table_3col <- exon_table %>%
    filter(chromosome %in% c(1:22, "X", "Y", "MT"),
           seq_type %in% c("chromosome","mitochondrion"),
           assembly_unit %in% c("Primary Assembly", "non-nuclear"))%>%
    select(chromosome, start, end) %>%
    ## Sorting table by chromosome and feature start position
    arrange(chromosome, start) %>%
    mutate(chrom = paste0("chr", chromosome)) %>%
    select(chrom, start, end) %>%
  distinct()

## Write to table as a bed file
tmp_bed <- tempfile(fileext = ".bed")
write_tsv(exon_table_3col, tmp_bed, col_names = FALSE)
```

4 Preparing Stratification Files

Generating bed file with merged overlapping CDS regions as well as notin bed. See mappability documentation for how GRCh38 genome bed file was generated.

```
merged_bed_file <- "GRCh38_refseq_cds_merged.bed"
system2("bedtools", args = c("merge", "-i", tmp_bed),
        stdout = merged_bed_file)

## Generating not-in bed
## Generating genome bed for subtractBed
faidx_file <- "GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.fai"
faidx_md5 <- "5fddbc109c82980f9436aa5c21a57c61"
genome_bed_file <- "human.GRCh38.chroms.only.genome.bed"

if (!file.exists(faidx_file)) {
  download.file(url = faidx_path, destfile = faidx_file)

  ## MD5 check
  dwn_md5 <- tools::md5sum(faidx_file)

  if(faidx_md5 != dwn_md5){
   warning("MD5 for downloaded reference index does not match expected MD5")
  }
}


faidx_df <- read_tsv(faidx_file, col_names = c("CHROM","SIZE","X1","X2","X3")) %>%
  filter(CHROM %in% paste0("chr", c(1:22,"X","Y"))) %>%
  mutate(START = 1) %>%
  select(CHROM, START, SIZE)
```

```
## Parsed with column specification:
## cols(
##   CHROM = col_character(),
##   SIZE = col_double(),
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double()
## )
```

```
write_tsv(faidx_df, path = genome_bed_file, col_names = FALSE)

notin_merged_bed_file <- "notin_GRCh38_refseq_cds_merged.bed"
system2("subtractBed", args = c("-a", genome_bed_file, "-b", merged_bed_file),
        stdout = notin_merged_bed_file)

## Compressing stratificaiton beds
system2("bgzip", args = c("-f", merged_bed_file))
system2("bgzip", args = c("-f", notin_merged_bed_file))
```

5 File Sanity Checks

```
merged_bed <- read_tsv(paste0(merged_bed_file, ".gz"),
                       col_names = c("chrom","start","end"))
```

```
## Parsed with column specification:
## cols(
##   chrom = col_character(),
##   start = col_double(),
##   end = col_double()
## )
```

```
total_cds <- sum(merged_bed$end - merged_bed$start)

notin_merged_bed <- read_tsv(paste0(notin_merged_bed_file, ".gz"),
                       col_names = c("chrom","start","end"))
```

```
## Parsed with column specification:
## cols(
##   chrom = col_character(),
##   start = col_double(),
##   end = col_double()
## )
```

```
notin_total_cds <- sum(notin_merged_bed$end - notin_merged_bed$start)
```

Number of bases in merged 34,142,545. Number of bases not-in merged 3,054,138,498. Total bases (in + not-in) 3,088,281,043.

6 System Information

```
s_info <- devtools::session_info()
print(s_info$platform)
```

```
##  setting  value
##  version  R version 3.6.0 (2019-04-26)
##  os       macOS  10.15.3
##  system   x86_64, darwin15.6.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/New_York
##  date     2020-02-20
```

```
s_info$packages %>%
  filter(attached) %>%
  dplyr::select(package, loadedversion, source) %>%
      knitr::kable()
```

package	loadedversion	source
dplyr	0.8.3	CRAN (R 3.6.0)
forcats	0.4.0	CRAN (R 3.6.0)
ggplot2	3.2.1	CRAN (R 3.6.0)
purrr	0.3.2	CRAN (R 3.6.0)
readr	1.3.1	CRAN (R 3.6.0)
stringr	1.4.0	CRAN (R 3.6.0)
tibble	2.1.3	CRAN (R 3.6.0)
tidyr	1.0.0	CRAN (R 3.6.0)
tidyverse	1.2.1	CRAN (R 3.6.0)

6.1 Software Versions

BEDtools

```
system("bedtools --version",intern = TRUE)
```

```
## [1] "bedtools v2.28.0"
```

bgzip

```
system("bgzip --version",intern = TRUE)
```

```
## [1] "bgzip (htslib) 1.9"
## [2] "Copyright (C) 2018 Genome Research Ltd."
```