Assignment 3 File

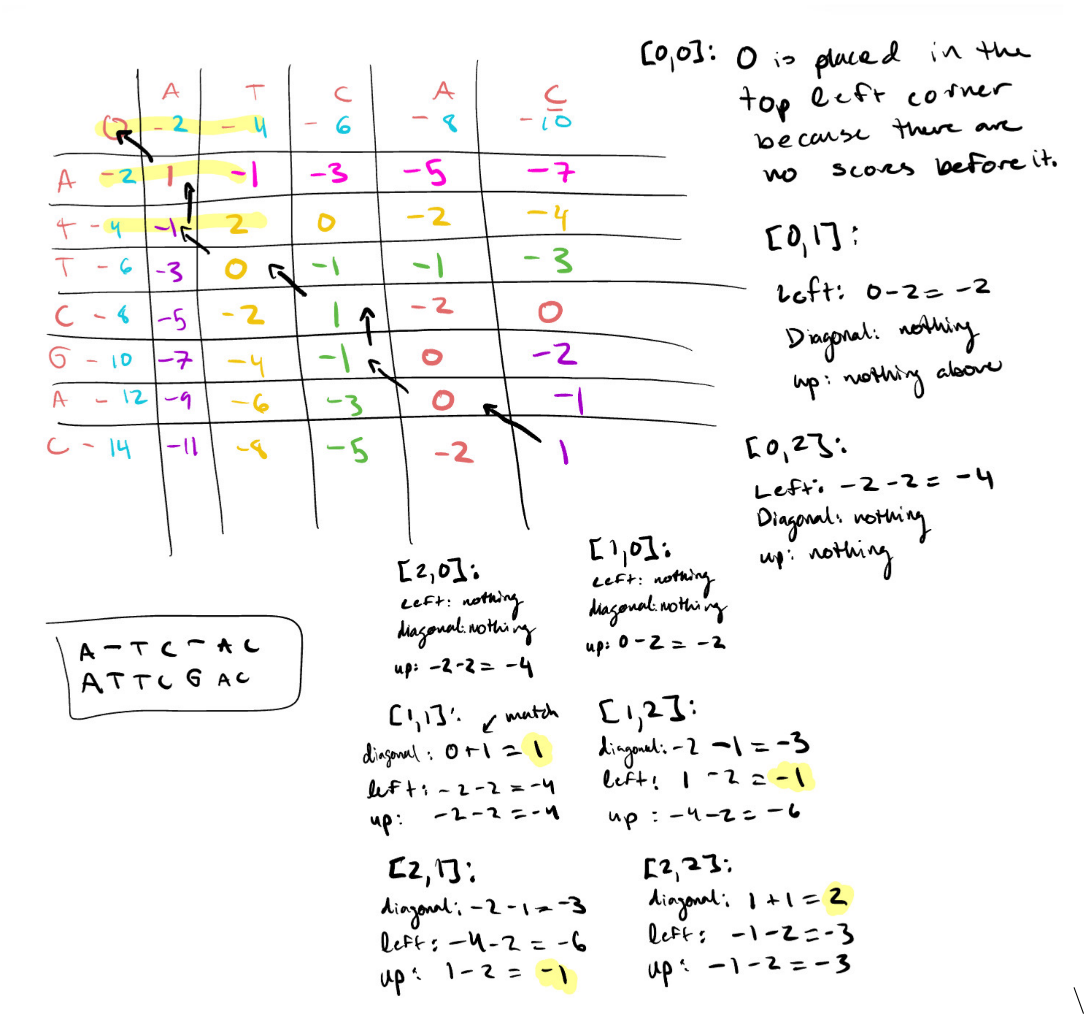
Group 15: Jordan Thompson, Arabela Tan, Zhipeng Ren

2024-11-05

Q1: Why areas of the genome with high GC content are hard to sequence?

Areas in the genome with high G-C content are challenging to sequence because they have strong hydrogen bonds. These hydrogen bonds are stable, and therefore can form secondary structures. These structures do not respond to amplification, and therefore areas with high G-C content are challenging to accurately amplify.

Q2



		A	T	C	A	C
	0	-2	-4	-6	-8	-10
A	-2	1	-1	-3	-5	-7
T	-4	-1	2	0	-2	-4
T	-6	-3	0	-1	-1	-3
C	-8	-5	-2	1	-2	0
G	-10	-7	-4	-1	0	-2
A	-12	-9	-6	-3	0	-1
C	-14	-11	-8	-5	-2	1

[0,0]: 0 is placed in the top left corner because there are no scores before it.

[0,1]:
left: 0-2= -2
Diagonal: nothing
up: nothing above

[0,2]:
Left: -2-2= -4
Diagonal: nothing
up: nothing

[1,0]:
left: nothing
Diagonal: nothing
up: 0-2 = -2

[2,0]:
left: nothing
diagonal: nothing
up: -2-2= -4

```
A - T C - A C
A T T C G A C
```

[1,1]: match
diagonal: 0+1 = 1
left: -2-2 = -4
up: -2-2 = -4

[1,2]:
diagonal: -2 -1 = -3
left: 1 -2 = -1
up: -4-2 = -6

[2,1]:
diagonal: -2-1 = -3
left: -4-2 = -6
up: 1-2 = -1

[2,2]:
diagonal: 1+1 = 2
left: -1-2 = -3
up: -1-2 = -3

Q3. Looking at the Metadata of an alignment (SAM) file

Q3.1.

```
single_cell_RNA_seq <- read.csv("single_cell_RNA_seq_bam.sam", nrows=73, sep="\t", header=FALSE,
fill=TRUE)
```

SN: Reference Sequence Name LN: Reference Sequence Length

Q3.2.

```
print(paste("Length of X Chromosome:",single_cell_RNA_seq[single_cell_RNA_seq[,2]=="SN:X",3]))
```

```
## [1] "Length of X Chromosome: LN:171031299"
```

Q4. Looking at the Reads of an alignment (SAM) file

Q4.1.

```
sam <- read.csv("single_cell_RNA_seq_bam.sam", sep="\t", header=FALSE,
comment.char="@", col.names = paste0("V",seq_len(30)), fill=TRUE)
sam <- sam[paste0("V",seq_len(11))]

print(paste("Number of Reads in the BAM file:", nrow(sam)))
```

```
## [1] "Number of Reads in the BAM file: 146346"
```

Q4.2.

```
print(sam[10,])
```

```
##                                           V1 V2 V3      V4  V5  V6 V7 V8 V9
## 10 NS500668:199:HV73CBGX2:1:11203:20546:3351 16  1 3365976 255 58M  *  0  0
##                                                           V10
## 10 AATCAAAAAGGGGGCTGTCAGTAGGATGATATAAGATATAGATGTAGTTTATCTCCTA
##                                                           V11
## 10 EEEEEEEEEEA//AAAAEEEEEE/AEEAEAEEEEEEEEEEEEEEAAEE///EEEAAA6A
```

```
print("To find the chromosome to which the read was aligned, we should look at the 3rd column")
```

```
## [1] "To find the chromosome to which the read was aligned, we should look at the 3rd column"
```

```r
print("V11 corresponds to the the ASCII of the base quality plus 33")
```

```
## [1] "V11 corresponds to the the ASCII of the base quality plus 33"
```

Q4.3.

```r
number_of_reads <- nrow(sam[sam[,3]=="X",])

print(paste("Number of reads in the file that align to chromosome X:",number_of_reads))
```

```
## [1] "Number of reads in the file that align to chromosome X: 5999"
```

Q4.4.

```r
#Isolating the Base Quality Reads of X-Chromosome
base_quality_read <- sam[sam[,3]=="X",]

#Creating a Data Frame to store the converted values
mean_bq <- data.frame("Read_Name:"=base_quality_read$V1)

#Going through each read
for (i in 1: number_of_reads){

  #Converting the ASCII Value into Base Quality
  bq_indiv_read <- as.numeric(charToRaw(base_quality_read[i, 11])) - 33

  #Adds the converted values in the dataframe
  mean_bq$Bq[i] <- I(list(bq_indiv_read))

  #Adds the mean base quality into the dataframe
  mean_bq$Mean_Base_Quality[i] <- mean(bq_indiv_read)
}

 print(paste("Mean of the Reads:", mean(mean_bq$Mean_Base_Quality)))
```

```
## [1] "Mean of the Reads: 32.7234912715338"
```

Q4.5.

```r
#library(ggplot2)

#base_quality_read <- sam[sam[,3]=="X",]

#for(i in 1:100){
#boxplot(unlist(mean_bq$Bq[i]))
#}
```

```r
#mean_bq$position <-sam$V4
#sam$V4
#order(mean_bq$Bq[1:100])
#data<-boxplot(mean_bq$Bq[1:100])
```

Q4.6.

The column that contains the leftmost mapping position of the reads is the POS column, found in the fourth column.

Q4.7.

```r
sam_9 <- sam[which(sam$V3 == 9 & sam$V4 >= 40801273 & sam$V4 <= 40805199), ]
length(sam_9$V4)
```

```
## [1] 119
```

Q4.8.

```r
#Number of reads that have mapping quality less than 50
mq_less50 <-nrow(sam[sam[5]<50,])

print(paste("Number of reads that have mapping quality of less than 50:", mq_less50 ))
```

```
## [1] "Number of reads that have mapping quality of less than 50: 61527"
```

Q4.9.

```r
#Isolates the reads with mapping quality of less than 50
mapping_quality_data <- sam[sam$V5 < 50, ]

# Calculate the mean mapping quality for this subset
mean_mapping_quality <- mean(mapping_quality_data$V5)

#Prints the result
print(paste("Mean Mapping Quality of Reads:", mean_mapping_quality))
```

```
## [1] "Mean Mapping Quality of Reads: 0.241812537585125"
```

Q4.10.

```r
tdTomato_reads <- sam[sam$V3 == "tdTomato", ]
num_tdTomato_reads <- nrow(tdTomato_reads)
print(paste("Number of reads aligning to tdTomato sequence:", num_tdTomato_reads))
```

```
## [1] "Number of reads aligning to tdTomato sequence: 63"
```

Yes, because tdTomato is a fluorophore, it would emit fluorescence under the appropriate excitation light.

Adding a fluorophore like tdTomato allows researchers to visualize and track specific cells or gene expression under a microscope.

Q5.1.

```
vcf_con <- file("RNA_seq_annotated_variants.vcf", open="r")
vcf_file <- readLines(vcf_con)
close(vcf_con)
vcf <- data.frame(vcf_file)
header <- vcf[grepl("##", vcf$vcf_file), ]
factor(header)
```

```
##   [1] ##fileformat=VCFv4.1
##   [2] ##fileDate=20200930
##   [3] ##source=strelka
##   [4] ##source_version=2.9.2
##   [5] ##startTime=Wed Sep 30 13:12:59 2020
##   [6] ##contig=<ID=1,length=195471971>
##   [7] ##contig=<ID=10,length=130694993>
##   [8] ##contig=<ID=11,length=122082543>
##   [9] ##contig=<ID=12,length=120129022>
##  [10] ##contig=<ID=13,length=120421639>
##  [11] ##contig=<ID=14,length=124902244>
##  [12] ##contig=<ID=15,length=104043685>
##  [13] ##contig=<ID=16,length=98207768>
##  [14] ##contig=<ID=17,length=94987271>
##  [15] ##contig=<ID=18,length=90702639>
##  [16] ##contig=<ID=19,length=61431566>
##  [17] ##contig=<ID=2,length=182113224>
##  [18] ##contig=<ID=3,length=160039680>
##  [19] ##contig=<ID=4,length=156508116>
##  [20] ##contig=<ID=5,length=151834684>
##  [21] ##contig=<ID=6,length=149736546>
##  [22] ##contig=<ID=7,length=145441459>
##  [23] ##contig=<ID=8,length=129401213>
##  [24] ##contig=<ID=9,length=124595110>
##  [25] ##contig=<ID=MT,length=16299>
##  [26] ##contig=<ID=X,length=171031299>
##  [27] ##contig=<ID=Y,length=91744698>
##  [28] ##contig=<ID=JH584299.1,length=953012>
##  [29] ##contig=<ID=GL456233.1,length=336933>
##  [30] ##contig=<ID=JH584301.1,length=259875>
##  [31] ##contig=<ID=GL456211.1,length=241735>
##  [32] ##contig=<ID=GL456350.1,length=227966>
##  [33] ##contig=<ID=JH584293.1,length=207968>
##  [34] ##contig=<ID=GL456221.1,length=206961>
##  [35] ##contig=<ID=JH584297.1,length=205776>
##  [36] ##contig=<ID=JH584296.1,length=199368>
##  [37] ##contig=<ID=GL456354.1,length=195993>
##  [38] ##contig=<ID=JH584294.1,length=191905>
##  [39] ##contig=<ID=JH584298.1,length=184189>
##  [40] ##contig=<ID=JH584300.1,length=182347>
##  [41] ##contig=<ID=GL456219.1,length=175968>
##  [42] ##contig=<ID=GL456210.1,length=169725>
##  [43] ##contig=<ID=JH584303.1,length=158099>
##  [44] ##contig=<ID=JH584302.1,length=155838>
##  [45] ##contig=<ID=GL456212.1,length=153618>
##  [46] ##contig=<ID=JH584304.1,length=114452>
##  [47] ##contig=<ID=GL456379.1,length=72385>
##  [48] ##contig=<ID=GL456216.1,length=66673>
##  [49] ##contig=<ID=GL456393.1,length=55711>
##  [50] ##contig=<ID=GL456366.1,length=47073>
##  [51] ##contig=<ID=GL456367.1,length=42057>
##  [52] ##contig=<ID=GL456239.1,length=40056>
##  [53] ##contig=<ID=GL456213.1,length=39340>
##  [54] ##contig=<ID=GL456383.1,length=38659>
##  [55] ##contig=<ID=GL456385.1,length=35240>
##  [56] ##contig=<ID=GL456360.1,length=31704>
##  [57] ##contig=<ID=GL456378.1,length=31602>
##  [58] ##contig=<ID=GL456389.1,length=28772>
##  [59] ##contig=<ID=GL456372.1,length=28664>
##  [60] ##contig=<ID=GL456370.1,length=26764>
##  [61] ##contig=<ID=GL456381.1,length=25871>
##  [62] ##contig=<ID=GL456387.1,length=24685>
##  [63] ##contig=<ID=GL456390.1,length=24668>
##  [64] ##contig=<ID=GL456394.1,length=24323>
##  [65] ##contig=<ID=GL456392.1,length=23629>
##  [66] ##contig=<ID=GL456382.1,length=23158>
##  [67] ##contig=<ID=GL456359.1,length=22974>
##  [68] ##contig=<ID=GL456396.1,length=21240>
##  [69] ##contig=<ID=GL456368.1,length=20208>
##  [70] ##contig=<ID=JH584292.1,length=14945>
##  [71] ##contig=<ID=JH584295.1,length=1976>
##  [72] ##contig=<ID=tdTomato,length=2250>
##  [73] ##contig=<ID=SSM2_GFP,length=1619>
##  [74] ##contig=<ID=CreERT2,length=1983>
##  [75] ##content=strelka germline small-variant calls
##  [76] ##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the region described in this
##  [77] ##INFO=<ID=BLOCKAVG_min30p3a,Number=0,Type=Flag,Description="Non-variant multi-site block. Non-
##  [78] ##INFO=<ID=SNVHPOL,Number=1,Type=Integer,Description="SNV contextual homopolymer length">
##  [79] ##INFO=<ID=CIGAR,Number=A,Type=String,Description="CIGAR alignment for each alternate indel al
##  [80] ##INFO=<ID=RU,Number=A,Type=String,Description="Smallest repeating sequence unit extended or co
##  [81] ##INFO=<ID=REFREP,Number=A,Type=Integer,Description="Number of times RU is repeated in referenc
##  [82] ##INFO=<ID=IDREP,Number=A,Type=Integer,Description="Number of times RU is repeated in indel all
##  [83] ##INFO=<ID=MQ,Number=1,Type=Integer,Description="RMS of mapping quality">
##  [84] ##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##  [85] ##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##  [86] ##FORMAT=<ID=GQX,Number=1,Type=Integer,Description="Empirically calibrated genotype quality sco
##  [87] ##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Filtered basecall depth used for site genoty
##  [88] ##FORMAT=<ID=DPF,Number=1,Type=Integer,Description="Basecalls filtered from input prior to site
##  [89] ##FORMAT=<ID=MIN_DP,Number=1,Type=Integer,Description="Minimum filtered basecall depth used for
##  [90] ##FORMAT=<ID=AD,Number=.,Type=Integer,Description="Allelic depths for the ref and alt alleles i
##  [91] ##FORMAT=<ID=ADF,Number=.,Type=Integer,Description="Allelic depths on the forward strand">
##  [92] ##FORMAT=<ID=ADR,Number=.,Type=Integer,Description="Allelic depths on the reverse strand">
##  [93] ##FORMAT=<ID=FT,Number=1,Type=String,Description="Sample filter, 'PASS' indicates that all fil
##  [94] ##FORMAT=<ID=DPI,Number=1,Type=Integer,Description="Read depth associated with indel, taken fro
##  [95] ##FORMAT=<ID=PL,Number=G,Type=Integer,Description="Normalized, Phred-scaled likelihoods for ger
##  [96] ##FORMAT=<ID=PS,Number=1,Type=Integer,Description="Phase set identifier">
##  [97] ##FORMAT=<ID=SB,Number=1,Type=Float,Description="Sample site strand bias">
##  [98] ##FILTER=<ID=IndelConflict,Description="Indel genotypes from two or more loci conflict in at le
##  [99] ##FILTER=<ID=SiteConflict,Description="Site is filtered due to an overlapping indel call filter
## [100] ##FILTER=<ID=LowGQX,Description="Locus GQX is below threshold or not present">
## [101] ##FILTER=<ID=HighDPFRatio,Description="The fraction of basecalls filtered out at a site is grea
## [102] ##FILTER=<ID=HighSNVSB,Description="Sample SNV strand bias value (SB) exceeds 10">
## [103] ##FILTER=<ID=LowDepth,Description="Locus depth is below 3">
## [104] ##FILTER=<ID=NotGenotyped,Description="Locus contains forcedGT input alleles which could not be
## [105] ##FILTER=<ID=PloidyConflict,Description="Genotype call from variant caller not consistent with
## [106] ##FILTER=<ID=NoPassedVariantGTs,Description="No samples at this locus pass all sample filters a
## [107] ##SnpEffVersion="4.3t (build 2017-11-24 10:18), by Pablo Cingolani"
## [108] ##INFO=<ID=ANN,Number=.,Type=String,Description="Functional annotations: 'Allele | Annotation
## [109] ##INFO=<ID=LOF,Number=.,Type=String,Description="Predicted loss of function effects for this va
## [110] ##INFO=<ID=NMD,Number=.,Type=String,Description="Predicted nonsense mediated decay effects for
## 110 Levels: ##content=strelka germline small-variant calls ...
```

```r
variants <- read.csv("RNA_seq_annotated_variants.vcf", skip=length(header),
header=TRUE, sep="\t")
```

```r
print(paste("The reference allele: ", variants[1,"REF"]))
```

```
## [1] "The reference allele:  G"
```

```r
print(paste("The alternative allele: ", variants[1,"ALT"]))
```

```
## [1] "The alternative allele:  A"
```

Q5.2.

```r
#Using the ANN line of the VCF file to get column names for the dataframe.
cols_divide <- strsplit(vcf[108,1], ":") #Isolate ANN line
cols_divide <- cols_divide[[1]]
cols_divide <- cols_divide[2] #Getting the string after the :
col_names <- unlist(strsplit(cols_divide, "\\|")) #Separate that string by "\" and get a vector of stri

variants$INFO <- as.character(variants$INFO)
info <- strsplit(variants[1,8],"ANN") #getting the first line of the variant after ANN
info_1  <- info[[1]]
info_ann <- info_1[2]
info_sep <- unlist(strsplit(info_ann, ",")) #Separate it by comma
info_ann_sep_1 <- unlist(strsplit(info_sep[1], "\\|")) #Then separate by "\"
info_ann_sep_1[16] <- "" #add last element, was not included because it is empty.

df <- as.data.frame(t(info_ann_sep_1)) #form df with first variant of ANN
```

```
colnames(df) <- col_names

df
```

```
##    'Allele     Annotation   Annotation_Impact   Gene_Name            Gene_ID
## 1        =A intron_variant            MODIFIER       Sulf1 ENSMUSG00000016918
##    Feature_Type           Feature_ID   Transcript_BioType   Rank
## 1     transcript ENSMUST00000088585.9       protein_coding   2/21
##           HGVS.c   HGVS.p   cDNA.pos / cDNA.length   CDS.pos / CDS.length
## 1 c.-133+17418G>A
##    AA.pos / AA.length   Distance   ERRORS / WARNINGS / INFO' ">
## 1
```

Q5.3.

The annotation field tells us that the variant is an intron. Therefore this part of the sequence does not code protein.

```
print(df$` Annotation `) #prints the annotation field
```

```
## [1] "intron_variant"
```

Q5.4.

```
variants$INFO <- as.character(variants$INFO)
info <- strsplit(variants[683,8],"ANN") #getting the 683 line of the variant after ANN
info_1  <- info[[1]]
info_ann <- info_1[2]
info_sep <- unlist(strsplit(info_ann, ","))
info_ann_sep_1 <- unlist(strsplit(info_sep[1], "\\|"))
info_ann_sep_1[16] <- ""

df <- as.data.frame(t(info_ann_sep_1))
colnames(df) <- col_names
print(df$` Gene_Name `)
```

```
## [1] "Rps19"
```

Q5.5.

```
variant_types<-c()

#Converting into character
INFO_field <- as.character(variants[,"INFO"])

#Isolating the variant's info field
VCF_df <- data.frame("INFO"=INFO_field)
```

```r
for (j in 1:nrow(VCF_df)){

  #Separates by | to get the ANN information
  #[[1]] to access the value in the list
  Initial_split <- strsplit(INFO_field[j], split=",")[[1]]

  #Separates by ; to get individual variables
  Second_split <- strsplit(Initial_split,"\\|")

  #Goes through each variant file and separates by & to get individual variants
  for(i in 1:length(Second_split)){

    split_variants <- unlist(strsplit(Second_split[[i]][2], "&"))

    #Places the variant types in a vector
    variant_types<-c(variant_types,split_variants)
  }
}

#Gets a summary of the variant types
data.frame(table(variant_types))
```

```
##                        variant_types Freq
## 1                 3_prime_UTR_variant  211
## 2                 5_prime_UTR_variant    8
## 3             downstream_gene_variant  944
## 4                  frameshift_variant    6
## 5                   intergenic_region  130
## 6                  intragenic_variant    2
## 7                      intron_variant 1605
## 8                    missense_variant  156
## 9  non_coding_transcript_exon_variant  281
## 10      non_coding_transcript_variant    1
## 11            splice_acceptor_variant    9
## 12              splice_region_variant   63
## 13                        stop_gained    9
## 14                 synonymous_variant   72
## 15              upstream_gene_variant  636
```

###Q5.6 A frame shift variant is a variant that shifts all codons after that mutation because it is not a multiple of 3. Therefore entirely changing the protein made from that sequence, this causes more harm than mutations that are multiples of three as they only change one amino

Q5.8

```r
high_impact_genes <- c()
moderate_impact_genes <- c()

for (i in 1:nrow(variants)) {
  info_field <- as.character(variants$INFO[i])
  info_parts <- strsplit(info_field, ";")[[1]]
  ann_field <- info_parts[grep("^ANN=", info_parts)]

  if (length(ann_field) > 0) {
    ann_value <- sub("ANN=", "", ann_field)
    annotations <- strsplit(ann_value, ",")[[1]]

    for (annotation in annotations) {
      annotation_details <- strsplit(annotation, "\\|")[[1]]
      impact <- annotation_details[3]
      gene_name <- annotation_details[4]
      feature_type <- annotation_details[8]

      if (impact == "HIGH" && feature_type == "protein_coding") {
        high_impact_genes <- c(high_impact_genes, gene_name)
      }

      if (impact == "MODERATE" && feature_type == "protein_coding") {
        moderate_impact_genes <- c(moderate_impact_genes, gene_name)
      }
    }
  }
}
unique_high_impact_genes <- unique(high_impact_genes)
unique_moderate_impact_genes <- unique(moderate_impact_genes)

cat("Unique genes affected by HIGH impact coding mutations:\n", unique_high_impact_genes, "\n\n")
```

```
## Unique genes affected by HIGH impact coding mutations:
##  Ddx1 Rps14 Rps19 Hnrnpl
```

```r
cat("Unique genes affected by MODERATE impact coding mutations:\n", unique_moderate_impact_genes, "\n")
```

```
## Unique genes affected by MODERATE impact coding mutations:
##  Dpt Dcaf8 Rps15 Hsp90b1 Dcn Os9 Naca Atp5b Myl6 Rpl41 Grb10 Mtif2 Rps27a Rpl19 Rps7 Jkamp Crip1 Nid
```