Proyecto final

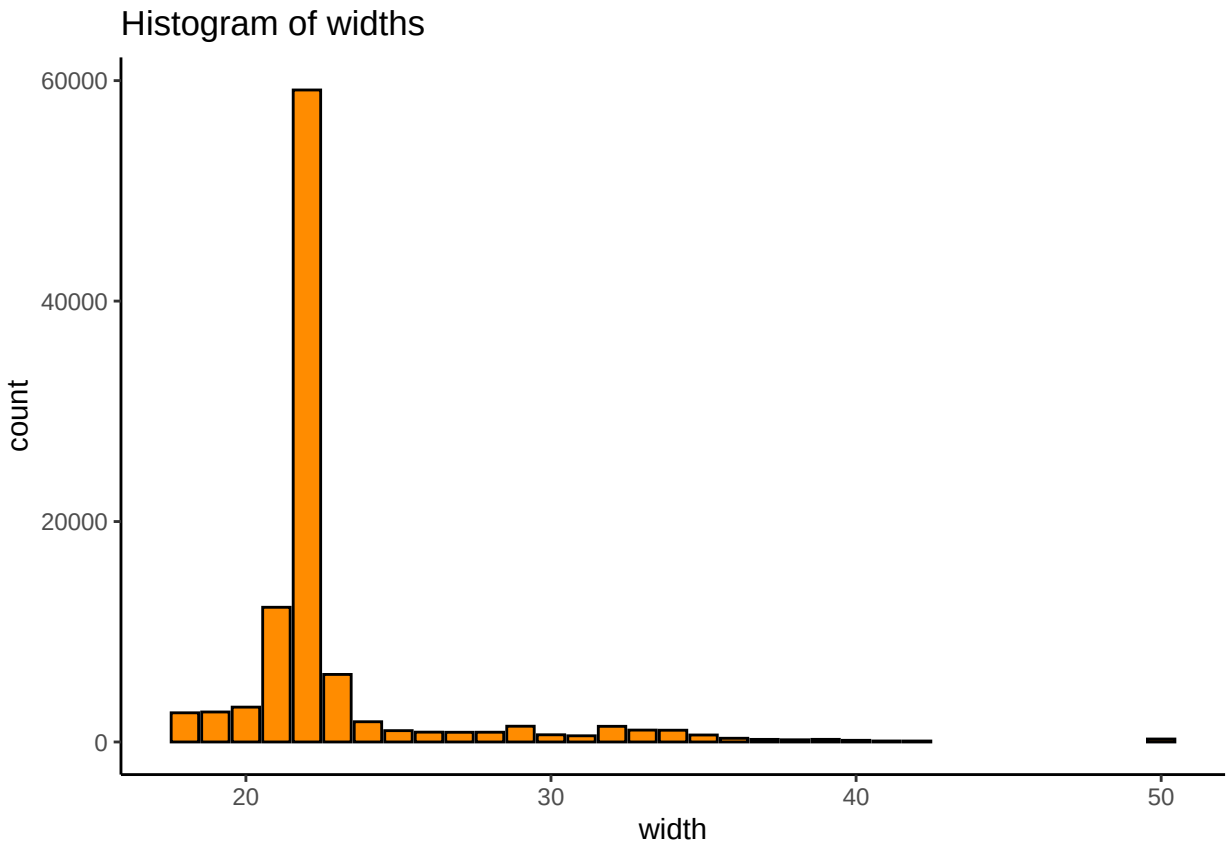
Equipo Doctorado

29/1/2020

Primera actividad para generar histogramas

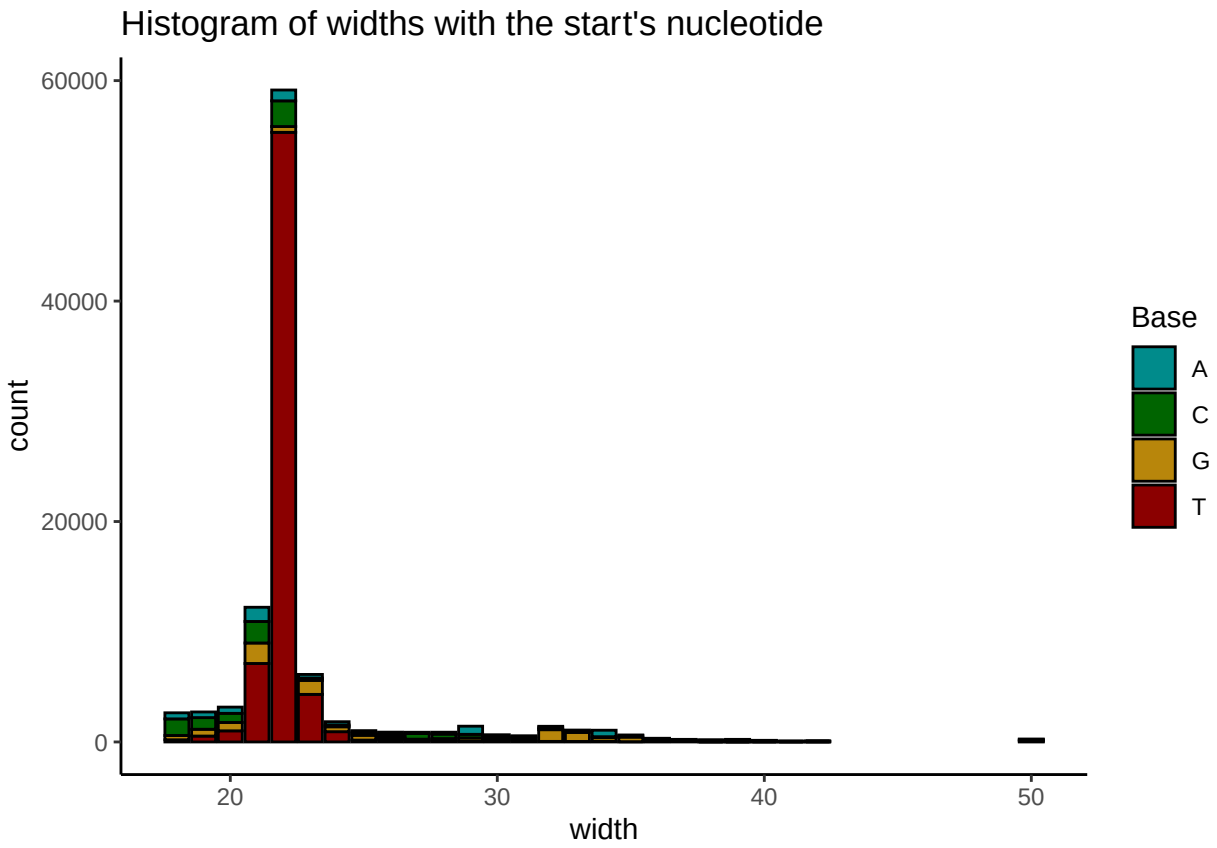
Primer histograma con la cantidad de secuencias de distintas longitudes

```
reaper_biostring <- Biostrings::readDNAStringSet("MODEK_100k.lane.clean.gz", format = "fastq")

width_nucleotideAnalysis <- function(reaper_biostring){

  a <- Biostrings::alphabetFrequency(reaper_biostring)
  a <- a[,1:4]
  b <- Biostrings::as.data.frame(reaper_biostring)
  first_nu <- substring(b$x,first = 1, last = 1)
  width <- reaper_biostring@ranges@width
  width <- data.frame(width)
  all <- cbind(width, a, first_nu)

  return(all)
}

analysis_1 <- width_nucleotideAnalysis(reaper_biostring)

ggplot(analysis_1, aes(width)) +
  geom_bar(fill="darkorange", color = "black") + theme_classic() +
  ggtitle("Histogram of widths")
```



Para conocer la cantidad de veces que aparece cada nucleotido en la primera posicion de las secuencias

```
ggplot(analysis_1, aes(width, fill=first_nu)) +
  geom_bar(color = "black") + theme_classic() +
  ggtitle("Histogram of widths with the start's nucleotide") +
  scale_fill_manual(values = c("darkcyan", "darkgreen", "darkgoldenrod", "darkred")) +
  labs(fill = "Base")
```



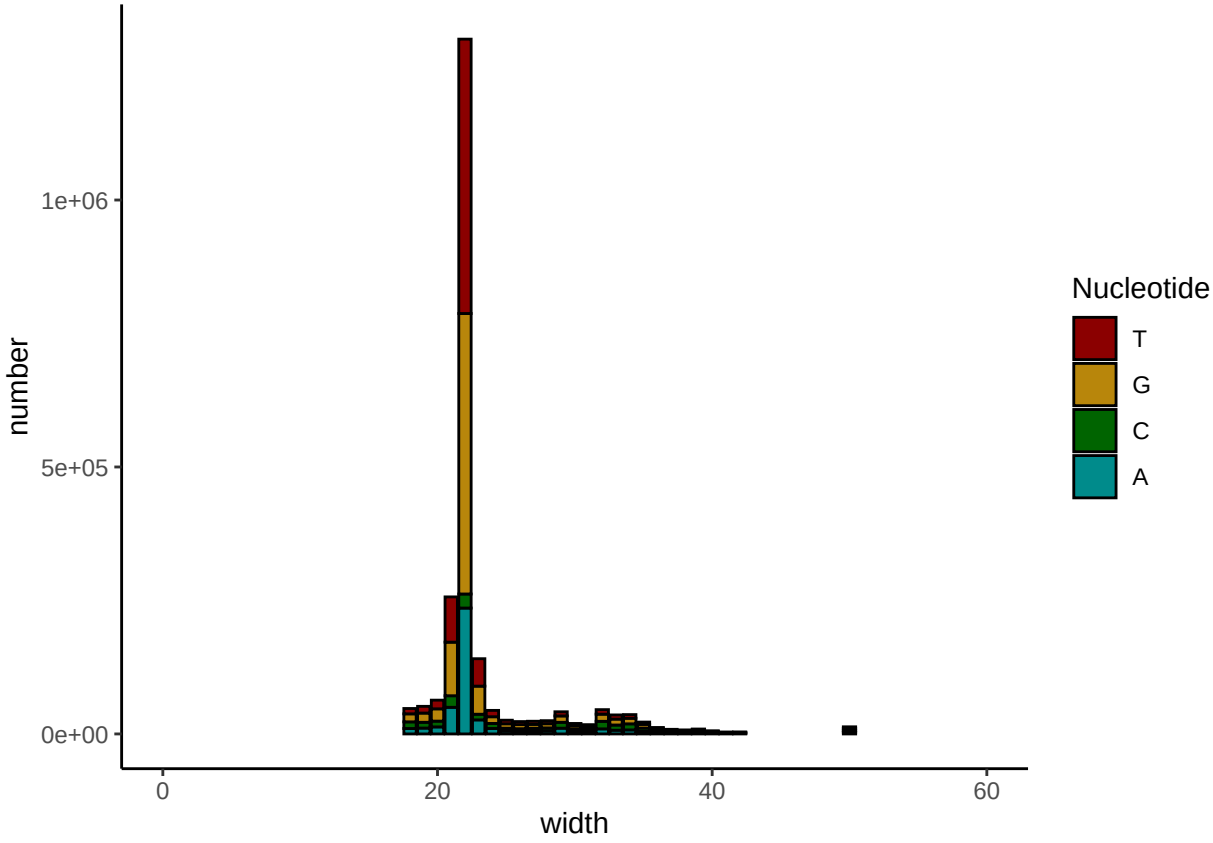
Tambien podemos conocer la cantidad de nucleotidos distintos por longitud de secuencia, para ello se realizo la siguiente funcion.

```
count_NucleotidePerWidth <- function(df){

  df$width <- as.factor(df$width)

  good_one <- data.frame()
  for (i in levels(df$width)) {

    a <- df[which(df$width==i),]

    A_lv <- data.frame(width = i, Nucleotide = "T", number=sum(a$T))
    G_lv <- data.frame(width = i, Nucleotide = "G", number=sum(a$G))
    C_lv <- data.frame(width = i, Nucleotide = "C", number=sum(a$C))
    T_lv <- data.frame(width = i ,Nucleotide = "A", number=sum(a$A))

    DF_NEW <- rbind(A_lv, G_lv, C_lv, T_lv)

    good_one <- rbind(good_one, DF_NEW)

  }

  easytable <- data.frame(table(df$width))

  prueba_listp <- merge.data.frame(good_one, easytable,
                                   by.x = "width", by.y = "Var1")
```

```
  prueba_listp$width <- as.character(prueba_listp$width)
  prueba_listp$width <- as.numeric(prueba_listp$width)

  return(prueba_listp)
}
```

Usando la funcion generada `count_NucleotidePerWidth()`

```
many_nu <- count_NucleotidePerWidth(analysis_1)


ggplot(many_nu, aes(x = width, y = number, fill=Nucleotide)) +
  geom_col(color = "black")  + theme_classic() +
  scale_fill_manual(values = c("darkred", "darkgoldenrod", "darkgreen", "darkcyan")) +
  xlim(0,60)
```



Aplicación de `fastq` a todos los archivos

```
cd secuencias_fastqc/

for file in *.gz
do
fastqc $file
done
```

Aplicacion de `reaper` a todos los archivos

```
cd secuencias_reaper/

for file in *.gz
do
./reaper-16-098/src/reaper -geom no-bc -3pa TGGAATTCTCGGGTGCCAAGG -i $file -basename $file -format-clea

done
```

Analisis `multiQC` en los resultados de `reaper` usando posteriormente `fastqc`

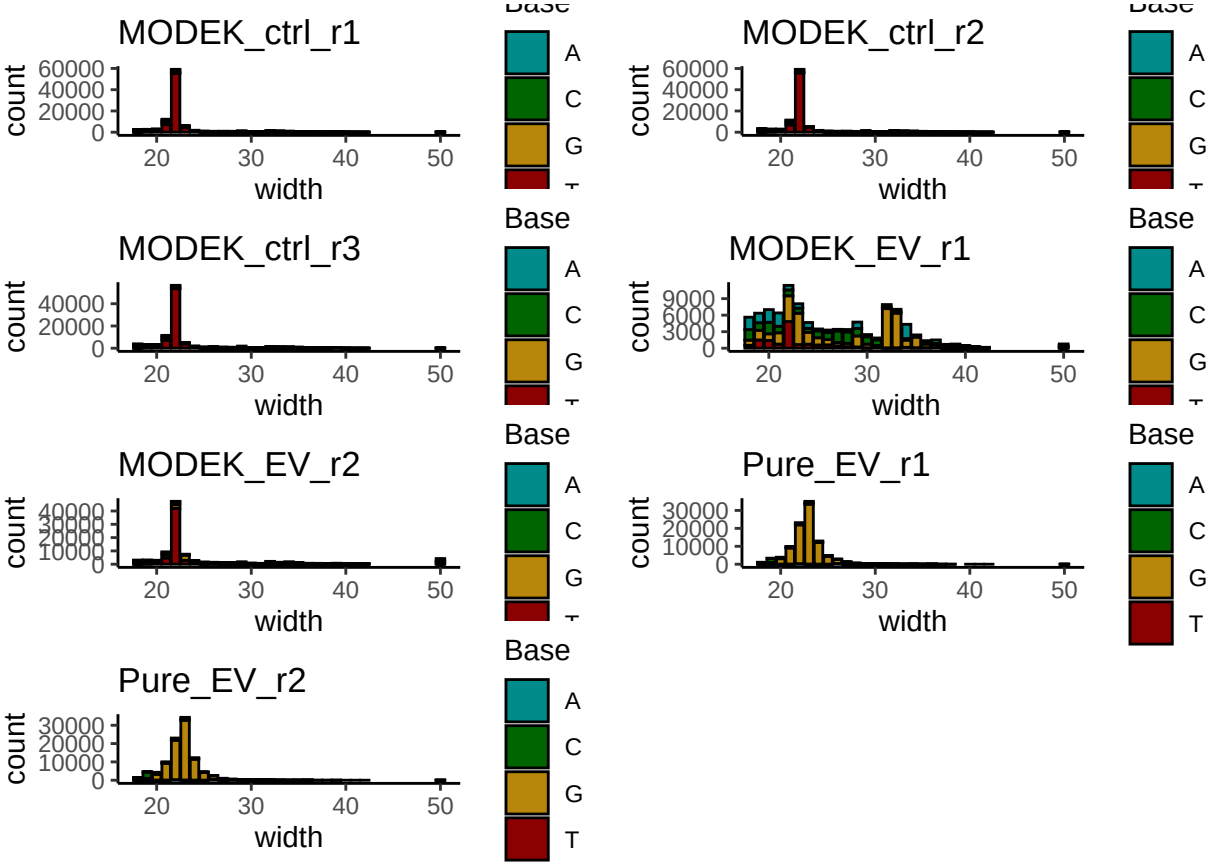
Para ello, en la carpeta `zip_reaper`se encuentran los resultados en `.zip` generados de la carpeta `reaper_fastqc`

```
cd secuecias_reaper/reaper_fastqc/
for file in *.gz
do
fastqc $file
done
```

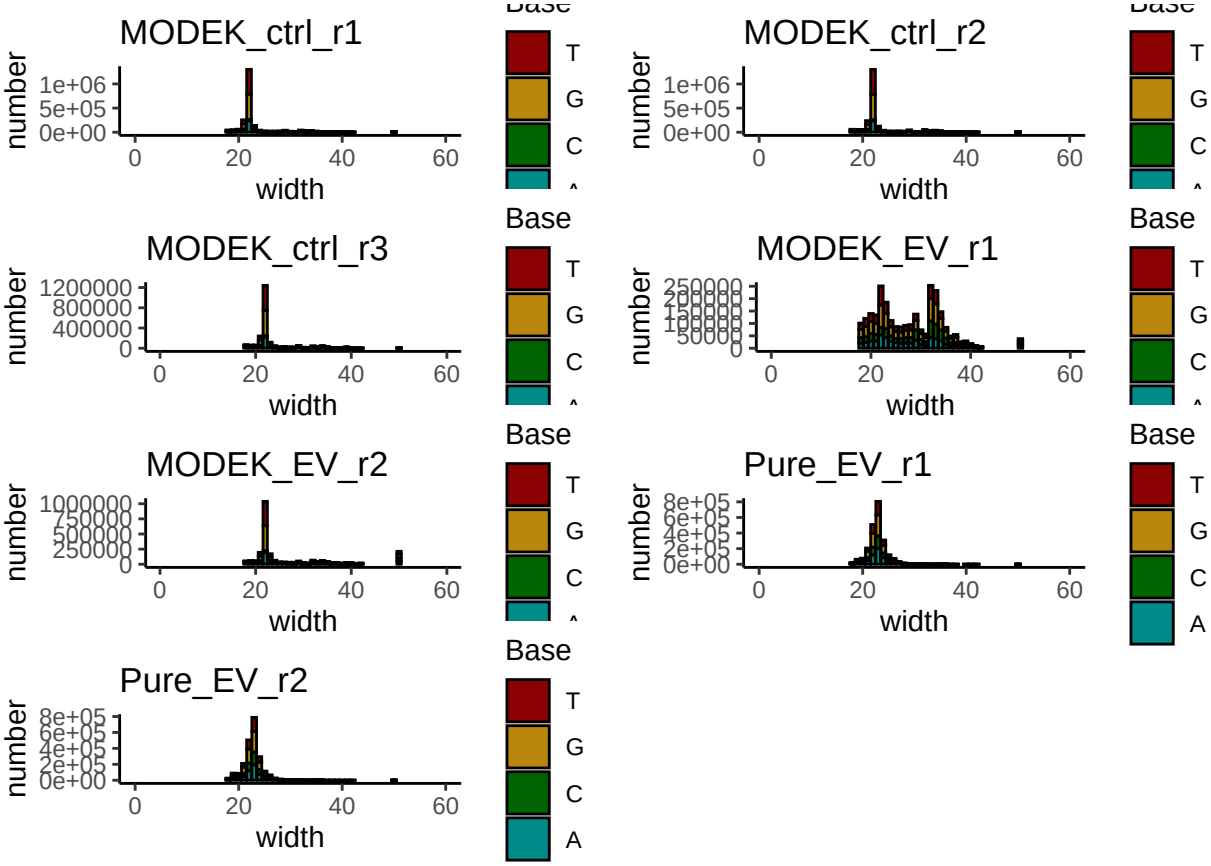
```
multiqc secuencias_reaper/reaper_fastqc_zip -o secuencias_reaper/reaper_fastqc_zip/
```

Resultados de los graficos de las secuencias

Histogramas con la cantidad de nucleotidos en la primera posición



Histogramas de la cantidad de nucleotidos distintos por longitud de secuencia



Alineamiento de genomas

Alineamiento de las secuencias con hbak (*Heligmosomoides bakeri*) Primero se hizo `bowtie-build` de hbak y posteriormente se utilizo `bowtie` para generar los formatos `.sam`

```
cd Alineamiento_hbak/

for file in *.gz
do
bowtie -S -v 1 -k 1 -f hbak $file $file.sam
done
```

```
samtools view -b Pure_EV_r1.fa.gz.sam > Pure_EV_r1.fa.gz.bam
samtools view -b MODEK_ctrl_r1_100k.fa.gz.sam > MODEK_ctrl_r1_100k.fa.gz.bam
samtools view -b MODEK_ctrl_r2_100k.fa.gz.sam > MODEK_ctrl_r2_100k.fa.gz.bam
samtools view -b MODEK_ctrl_r3_100k.fa.gz.sam > MODEK_ctrl_r3_100k.fa.gz.bam
samtools view -b MODEK_EV_r1_100k.fa.gz.sam > MODEK_EV_r1_100k.fa.gz.bam
samtools view -b MODEK_EV_r2_100k.fa.gz.sam > MODEK_EV_r2_100k.fa.gz.bam
samtools view -b Pure_EV_r2_100k.fa.gz.sam > Pure_EV_r2_100k.fa.gz.sam
```

Primeros ejercicios

```
hbak <- import("Alineamiento_hbak/Heligmosomoides_bakeri.gff3.gz")
mcols(hbak) <- mcols(hbak)[,c("source", "type","ID","Name","rep_name","class")]


bamFile = "Alineamiento_hbak/MODEK_ctrl_r2_100k.fa.gz.bam"
# what = c("rname", "strand", "pos", "qwidth")
# param = ScanBamParam(what=what)
# bam = scanBam(bamFile, param=param)

# mapGR = GenomicRanges::GRanges(
#   seqnames = bam[[1]]$rname,
#   ranges   = IRanges(start=bam[[1]]$pos, width=bam[[1]]$qwidth),
#   strand   = bam[[1]]$strand
# )

bamAlign <- GenomicAlignments::readGAlignments(bamFile)
mapGR <- as(bamAlign, "GRanges")

# Para obtener los miRNA o cadenas -
miRNAs <- hbak[hbak$type=="miRNA",]
minus_string <- hbak[strand(hbak)=="-"]

mcols(hbak)$counts = countOverlaps(hbak, mapGR)

typeCounts = aggregate(mcols(hbak)$counts, by=list("type"=mcols(hbak)$type), sum)

typeCounts
```

```
##               type     x
## 1             gene 47989
## 2             mRNA 47993
## 3             exon 47854
## 4              CDS 47854
## 5  three_prime_UTR     0
## 6   five_prime_UTR     0
## 7        rep_unk_S    16
## 8        rep_dna_S     1
## 9        rep_rna_S    21
## 10      rep_simple   453
## 11       rep_sat_S     0
## 12            tRNA    15
## 13           piRNA     0
## 14           miRNA 48311
## 15           snRNA     0
## 16            rRNA   206
## 17     other_ncRNA     0
## 18        mapmiRNA     0
```

Ejercicios avanzados:

-Calcular el número y porcentaje de bases que tienen alguna anotación traslapada de acuerdo a su objeto hbak

```
overlaped_hbak <- (hbak[overlapsAny(hbak, mapGR)])

base_overlaped <- data.frame('overlaped'=sum(overlapsAny(hbak, mapGR))/length(hbak) *100)

print(paste("El numero de bases traslapadas es de:",length(overlaped_hbak),
            "y el porcentaje es de:", base_overlaped[1,1], "%"))
```

```
## [1] "El numero de bases traslapadas es de: 72 y el porcentaje es de: 0.279644230395774 %"
```

-Diseñar una estrategia para evitar contar más de una vez los reads que mapean a estas regiones traslapadas.

```
new_new <- hbak[hbak$counts>0,]
ve <- vector()
for(i in 1:length(new_new)){

  if (i == length(new_new)){
    break()
  }
  add <- ranges(new_new)[i] != ranges(new_new)[i+1]
  ve[i] <- add

}

new_new <- new_new[ve]

typeCounts = aggregate(mcols(new_new)$counts, by=list("type"=mcols(new_new)$type), sum)

typeCounts$type <- as.character(typeCounts$type)

for(i in 1:nrow(typeCounts)){

  if(typeCounts[i,2] <= 711){
    typeCounts[i,1] <- "other"
  } else {
    next()
  }
}

typeCounts_other <- typeCounts[typeCounts$type=="other",]
typeCounts_other <- data.frame(type=as.character(typeCounts_other$type[1]), x = sum(typeCounts_other$x))
typecount_graph <- rbind(typeCounts[1:6,],typeCounts_other)

typecount_graph$type <- as.factor(typecount_graph$type)
typecount_graph <- typecount_graph %>% arrange(desc(x))

ggplot(typecount_graph, aes(x="", y=x, fill=sort(type)))+
  geom_bar(width = 1, stat = "identity", size = 1, color = "black",
```

```
          alpha= 0.6) +
coord_polar("y", start = 0, direction = -1) +
theme_void()
```