```
## Quality Control base on sequence length
## Author: Ciera Martinez
## Date: February 16, 2018 - March 6, 2018

## About
## This is an exploration of using sequence length as a qualifier for
## quality control of sequences. Input is a list a dataframe describing a
## bunch of fasta files. Including

## Information from input file includes:
## 1. Genomic region
## 2. Function vs non
## 3. Species
## 4. number of sequences in fasta file
## 5. length of each sequence in fasta file

## Output
## - [x] List of sequences that need to be removed in each file.
## - [ ] Identify how to achieve this in shell

## Goals
# [x] Just get rid of too small outliers
# [x] Mark if all under 1,0000
```

```
############################
## Required Libraries
############################

library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.3.2
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.3.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(outliers)
```

```
################################
### All sequence and file names
### Get data ready
################################
```

```
## Input data came from shell command on directory that contains all fasta files lifted

## for filename in *.fa; do
## cat $filename |
## seqkit fx2tab --length |
## awk -F "\t" '{print $1"|"$4"\t"$2}' |
## seqkit tab2fx > with_length_$filename
## done

### Read in data
all_length_info <- read.table("../data_summary/seqLengths2.txt")

## One problem is that each sequence needs to have a unique identifier.
## - [ ] We can easily add the length to the end of each fasta header.

split <- as.data.frame(str_split_fixed(all_length_info$V1, "\\|", 4))
all_length_info <- cbind(split[1:4], all_length_info[2])

head(all_length_info, 100)
```

```
##         V1 V2       V3 V4   V2
## 1   VT9999  0     dkik  -  496
## 2   VT9999  0 MEMB002B  -  422
## 3   VT9999  0 MEMB002C  -  486
## 4   VT9999  0 MEMB002D  -  490
## 5   VT9999  0 MEMB002E  -  499
## 6   VT9999  0 MEMB003A  -  492
## 7   VT9999  0 MEMB003B  -  551
## 8   VT9999  0 MEMB004E  -  514
## 9   VT9999  0 MEMB005B  -  495
## 10  VT9999  0 MEMB006A  -  505
## 11  VT9999  0 MEMB007D  -  594
## 12  VT9999  0 MEMB002A  +  481
## 13  VT9999  0 MEMB002F  +  493
## 14  VT9999  0 MEMB003C  +  481
## 15  VT9999  0 MEMB003D  +  492
## 16  VT9999  0 MEMB003F  +  501
## 17  VT9999  0 MEMB004A  +  422
## 18  VT9999  0 MEMB004B  +  512
## 19  VT9999  0 MEMB005D  +  489
## 20  VT9999  0 MEMB006B  +  490
## 21  VT9999  0 MEMB006C  +  504
## 22  VT9999  0 MEMB007B  +  478
## 23  VT9999  0 MEMB007C  +  558
## 24  VT9999  0 MEMB008C  +  505
## 25  VT0809  1     dkik  - 2537
## 26  VT0809  1 MEMB002A  - 2381
## 27  VT0809  1 MEMB002B  -  766
## 28  VT0809  1 MEMB002B  - 1369
## 29  VT0809  1 MEMB002C  - 2410
## 30  VT0809  1 MEMB003A  - 1373
## 31  VT0809  1 MEMB003D  - 2451
## 32  VT0809  1 MEMB003F  -  626
## 33  VT0809  1 MEMB003F  - 1439
```

```
## 34  VT0809  1 MEMB004A  - 2425
## 35  VT0809  1 MEMB005D  - 2314
## 36  VT0809  1 MEMB006A  - 2603
## 37  VT0809  1 MEMB007B  -  158
## 38  VT0809  1 MEMB007B  - 2240
## 39  VT0809  1 MEMB007C  - 2433
## 40  VT0809  1 MEMB008C  - 2691
## 41  VT0809  1 MEMB002D  + 2550
## 42  VT0809  1 MEMB002E  + 2723
## 43  VT0809  1 MEMB002F  + 2568
## 44  VT0809  1 MEMB003A  +  773
## 45  VT0809  1 MEMB003B  + 2565
## 46  VT0809  1 MEMB003C  + 2403
## 47  VT0809  1 MEMB004B  + 2568
## 48  VT0809  1 MEMB004E  +  117
## 49  VT0809  1 MEMB004E  + 2468
## 50  VT0809  1 MEMB005B  + 2600
## 51  VT0809  1 MEMB006B  + 2534
## 52  VT0809  1 MEMB006C  + 2652
## 53  VT0809  1 MEMB007D  + 2525
## 54  VT0845  1     dkik  - 2841
## 55  VT0845  1 MEMB002A  - 3078
## 56  VT0845  1 MEMB002B  - 2484
## 57  VT0845  1 MEMB002B  -  281
## 58  VT0845  1 MEMB002C  - 3044
## 59  VT0845  1 MEMB003A  -  386
## 60  VT0845  1 MEMB003A  - 2352
## 61  VT0845  1 MEMB003A  -  217
## 62  VT0845  1 MEMB003A  -  371
## 63  VT0845  1 MEMB003C  - 3110
## 64  VT0845  1 MEMB003D  - 3125
## 65  VT0845  1 MEMB004A  - 3059
## 66  VT0845  1 MEMB004B  - 3518
## 67  VT0845  1 MEMB005B  - 2950
## 68  VT0845  1 MEMB007C  - 2788
## 69  VT0845  1 MEMB002B  +  312
## 70  VT0845  1 MEMB002D  + 3303
## 71  VT0845  1 MEMB002E  + 3796
## 72  VT0845  1 MEMB002F  + 2996
## 73  VT0845  1 MEMB003B  + 3466
## 74  VT0845  1 MEMB003F  + 3904
## 75  VT0845  1 MEMB004E  + 2894
## 76  VT0845  1 MEMB005D  + 3590
## 77  VT0845  1 MEMB006A  +  171
## 78  VT0845  1 MEMB006A  + 3080
## 79  VT0845  1 MEMB006A  +  135
## 80  VT0845  1 MEMB006B  + 3214
## 81  VT0845  1 MEMB006C  + 3809
## 82  VT0845  1 MEMB007B  + 2946
## 83  VT0845  1 MEMB007D  + 2685
## 84  VT0845  1 MEMB008C  + 3409
## 85  VT0847  1     dkik  - 2607
## 86  VT0847  1 MEMB002A  - 2822
## 87  VT0847  1 MEMB002B  - 2117
```

```
## 88  VT0847  1 MEMB002C  - 2817
## 89  VT0847  1 MEMB003A  - 1317
## 90  VT0847  1 MEMB003A  -  720
## 91  VT0847  1 MEMB003C  - 2738
## 92  VT0847  1 MEMB003D  - 2339
## 93  VT0847  1 MEMB004A  - 2778
## 94  VT0847  1 MEMB004B  - 2800
## 95  VT0847  1 MEMB005B  - 2780
## 96  VT0847  1 MEMB007C  - 2319
## 97  VT0847  1 MEMB002B  +  678
## 98  VT0847  1 MEMB002D  + 3616
## 99  VT0847  1 MEMB002E  + 3545
## 100 VT0847  1 MEMB002F  + 2579
```

```
summary(all_length_info)
```

```
##        V1         V2               V3         V4               V2
##  VT64581:   220   0:109554   MEMB005D:  9649   -:101689   Min.   :     1
##  VT43148:   176   1: 97322   MEMB002B:  9087   +:105187   1st Qu.:  1399
##  VT27194:   162              MEMB004B:  9050              Median :  2166
##  VT64579:   152              MEMB005B:  9002              Mean   :  1986
##  VT57074:   138              MEMB003A:  8927              3rd Qu.:  2527
##  VT57072:   136              MEMB006A:  8910              Max.   :708431
##  (Other):205892              (Other) :152251
```

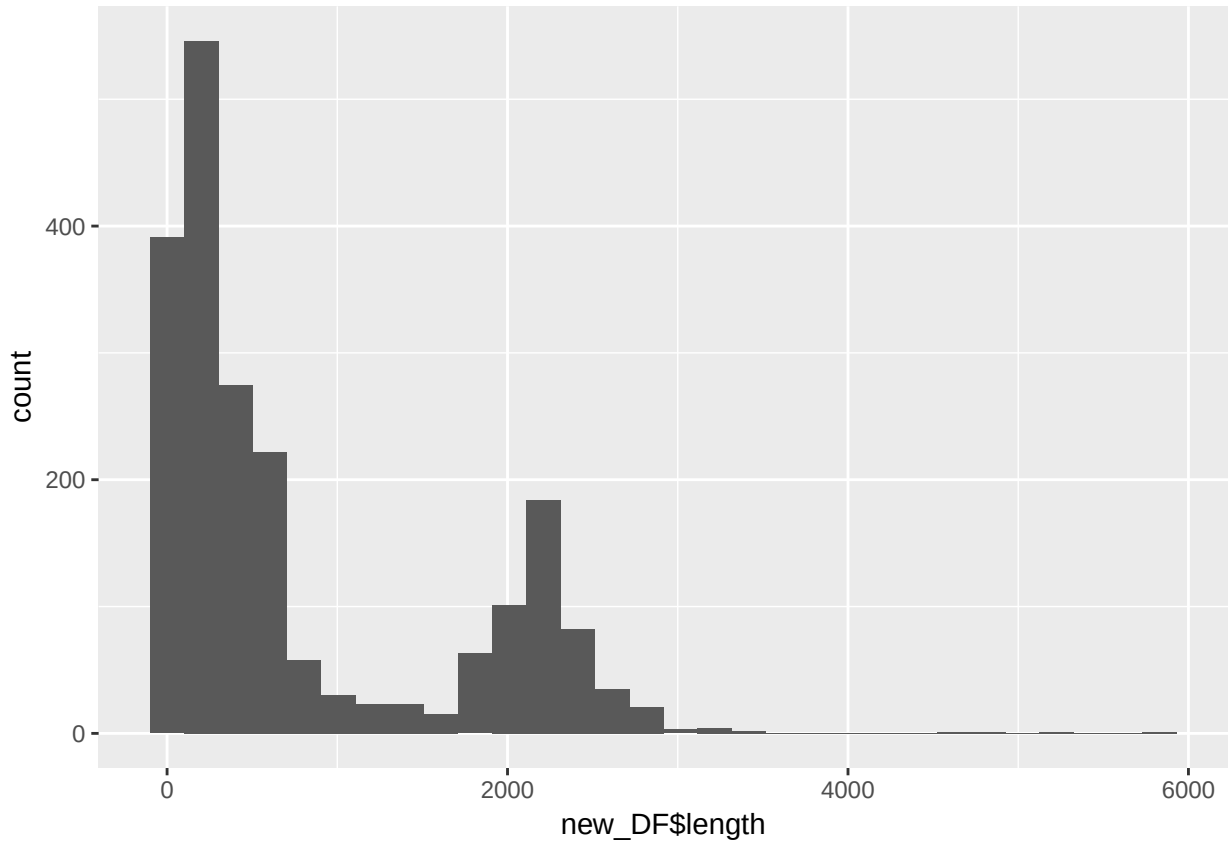
```
colnames(all_length_info) <- c("region", "expr", "species", "strand", "length")
```

```
######################
## Checking output
######################

###### Eeeeek some NAs introduced
warnings()
```

```
## NULL
```

```
## Checking where the NAs were introduced.
## Why? Do I care?
new_DF <- out[rowSums(is.na(out)) > 0,]
qplot(new_DF$length)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## Same length! That's good.
dim(out)
```

```
## [1] 206876      8
```

```
dim(all_length_info)
```

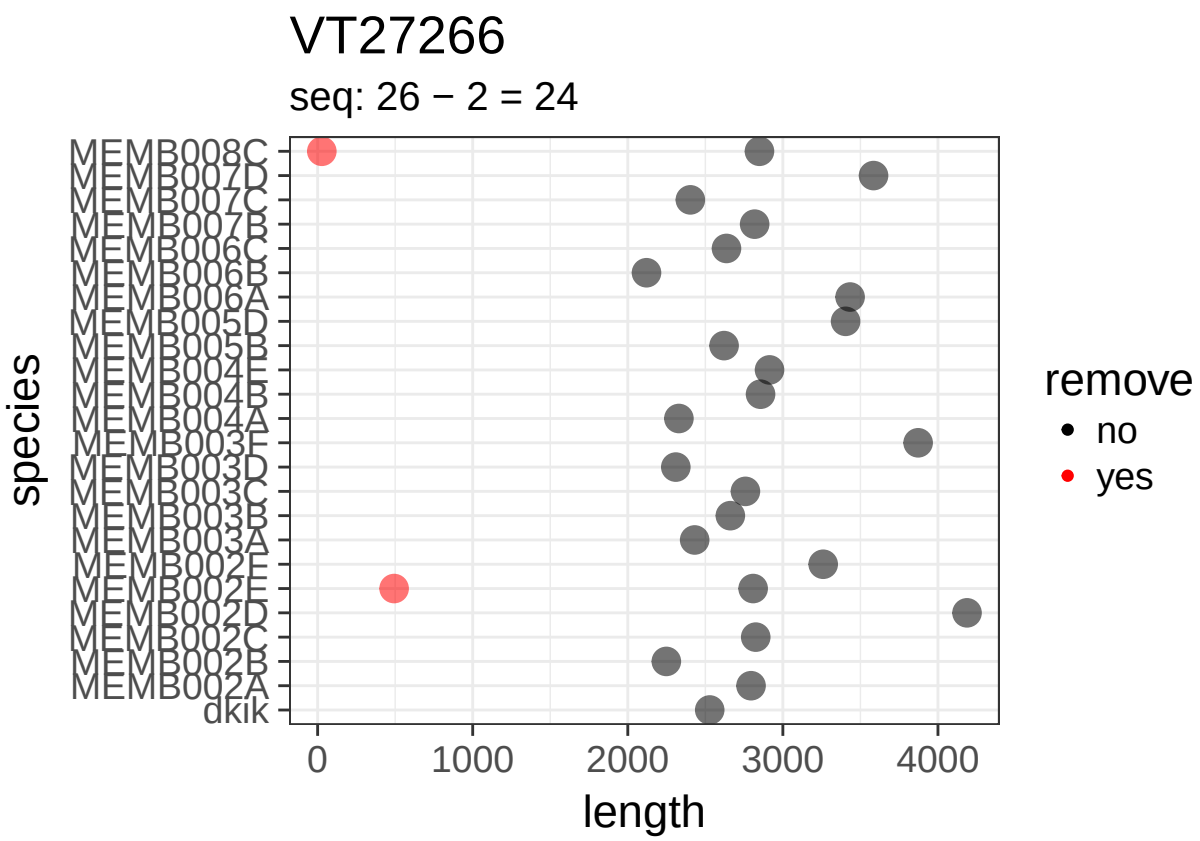
```
## [1] 206876      5
```

```
###################################
## Make a column that for removal.
## If sequence has length under mean and is an outlier mark TRUE
## If else mark FALSE
###################################

out$outlier <- as.factor(out$outlier)
out$remove <- with(out, ifelse(out$length < out$mean &
                     out$outlier == 1, "yes", "no"))

## Visualize
## random region
random_region <- sample(all_regions, 1)
out_sub <- subset(out, region == random_region)
removed_num <- nrow(filter(out_sub, remove == "yes"))
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.2
```

```
total_num <- nrow(out_sub) - removed_num
```

```
ggplot(out_sub, aes(length, species, color = remove, size = 2, alpha = .5)) +
  theme(panel.grid.major.y = element_line(colour = "grey70")) +
  theme_bw(base_size = 17) +
  geom_jitter(width = 15, height = 0) +
  scale_color_manual(values = c("black", "red")) +
  labs(title = random_region, subtitle = paste("seq:", nrow(out_sub), "-", removed_num, "=", total_num
  guides(size = FALSE, alpha = FALSE)
```




```
#######################################
### Check how many duplicate sequences per species and region
### before and after
#######################################

## How many sequences will be removed?
## 206,876 - 36,675 = 170,201
nrow(out) - out %>% filter(remove == "yes") %>% nrow()
```

```
## [1] 170201
```

```
## How many should there be
#6994 * 24 = 167,856
length(levels(out$region)) * 24
```

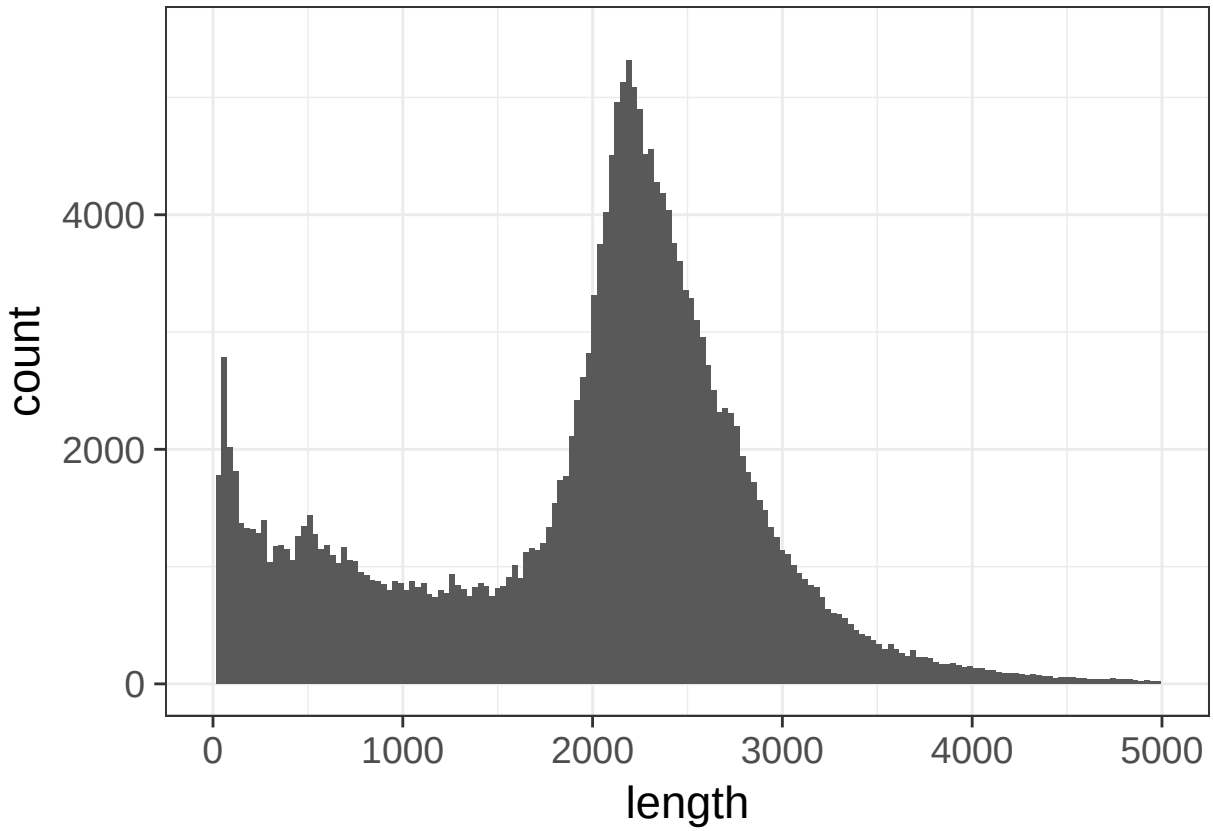
```
## [1] 167856
```

```
## Before outlier removal
ggplot(out, aes(length)) +
  geom_histogram(binwidth = 30) +
  scale_fill_manual(values = c("grey26", "red3", "grey")) +
```

```
  xlim(0, 5000) +
  ylim(0, 5500) +
  theme_bw(base_size = 17)
```

```
## Warning: Removed 1353 rows containing non-finite values (stat_bin).
```

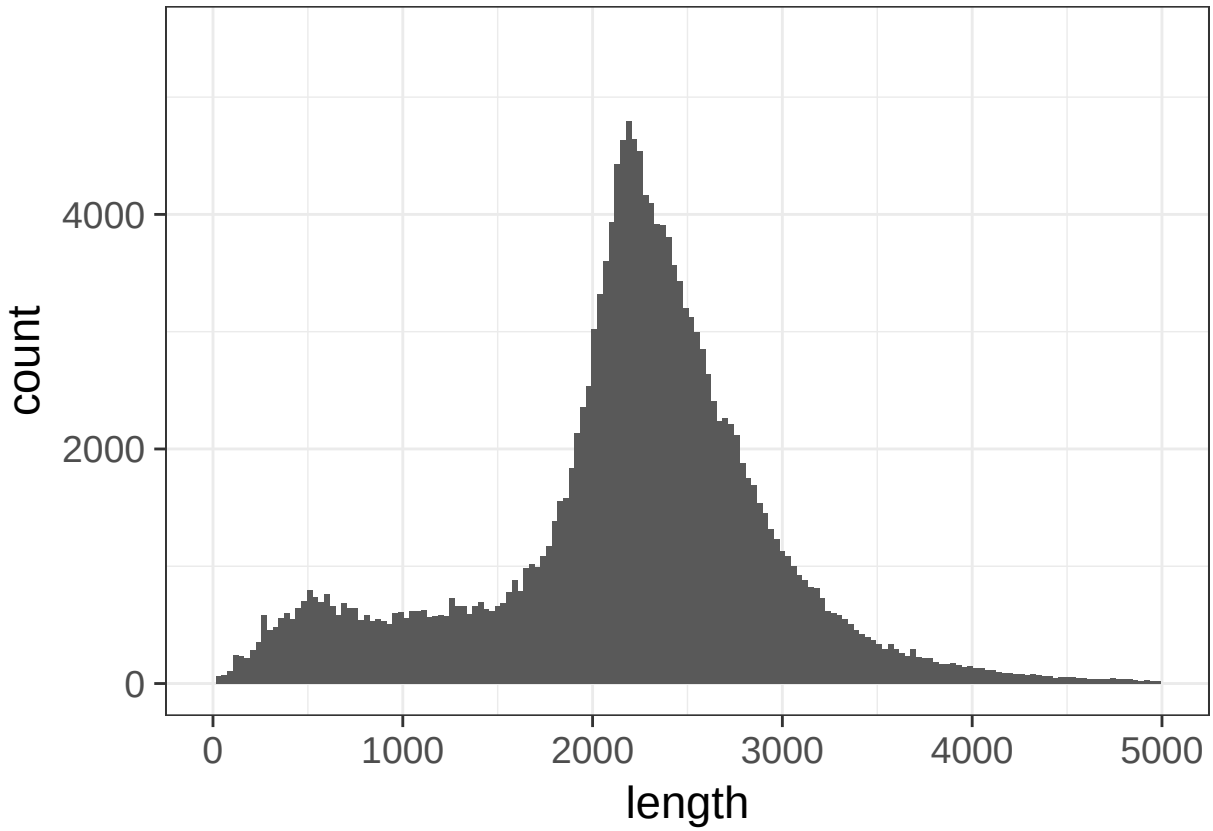
```
## Warning: Removed 1 rows containing missing values (geom_bar).
```




```
## After removal
out %>% filter(remove == "no") %>%
ggplot(., aes(length)) +
  geom_histogram(binwidth = 30) +
  xlim(0, 5000) +
  ylim(0, 5500) +
  theme_bw(base_size = 17)
```

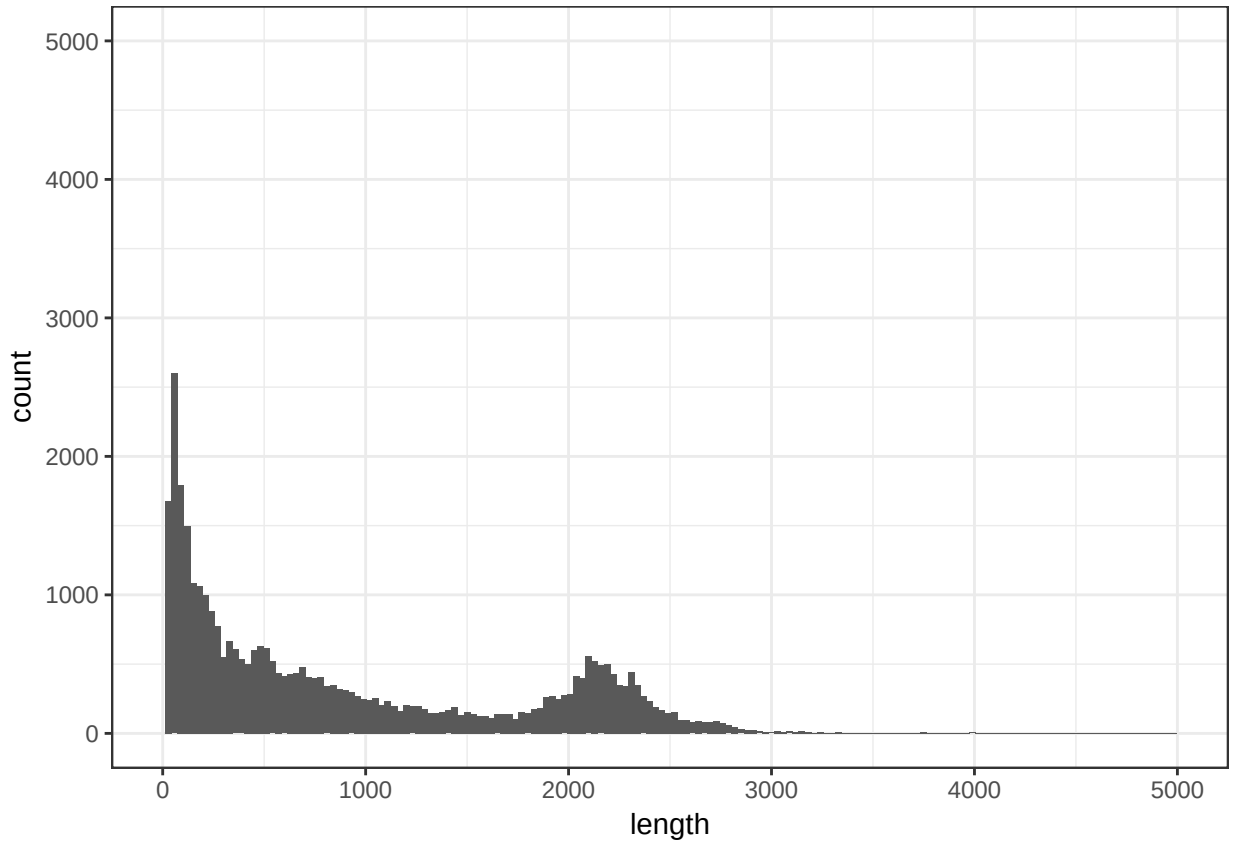
```
## Warning: Removed 1353 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



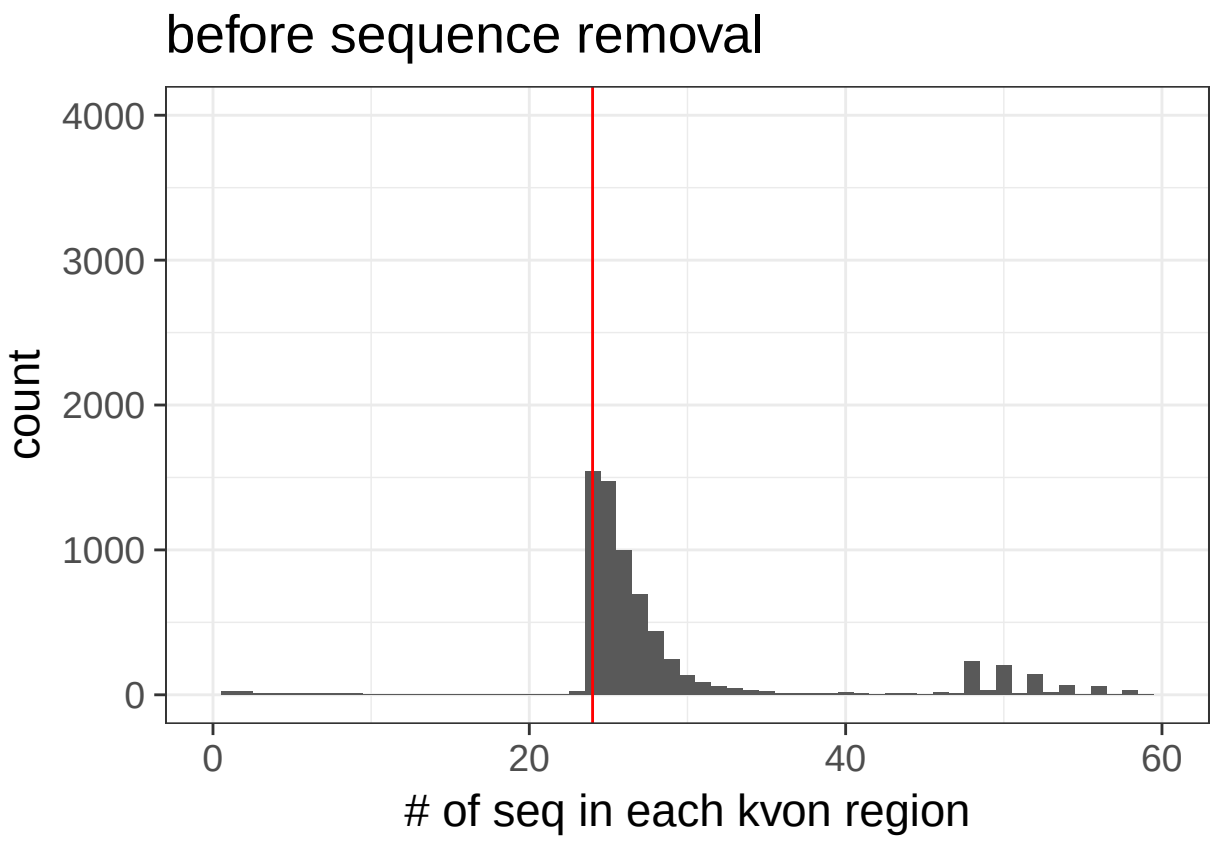

```
## Those that are removed
out %>% filter(remove == "yes") %>%
  ggplot(., aes(length)) +
  geom_histogram(binwidth = 30) +
  xlim(0, 5000) +
  ylim(0,5000) +
  theme_bw()
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



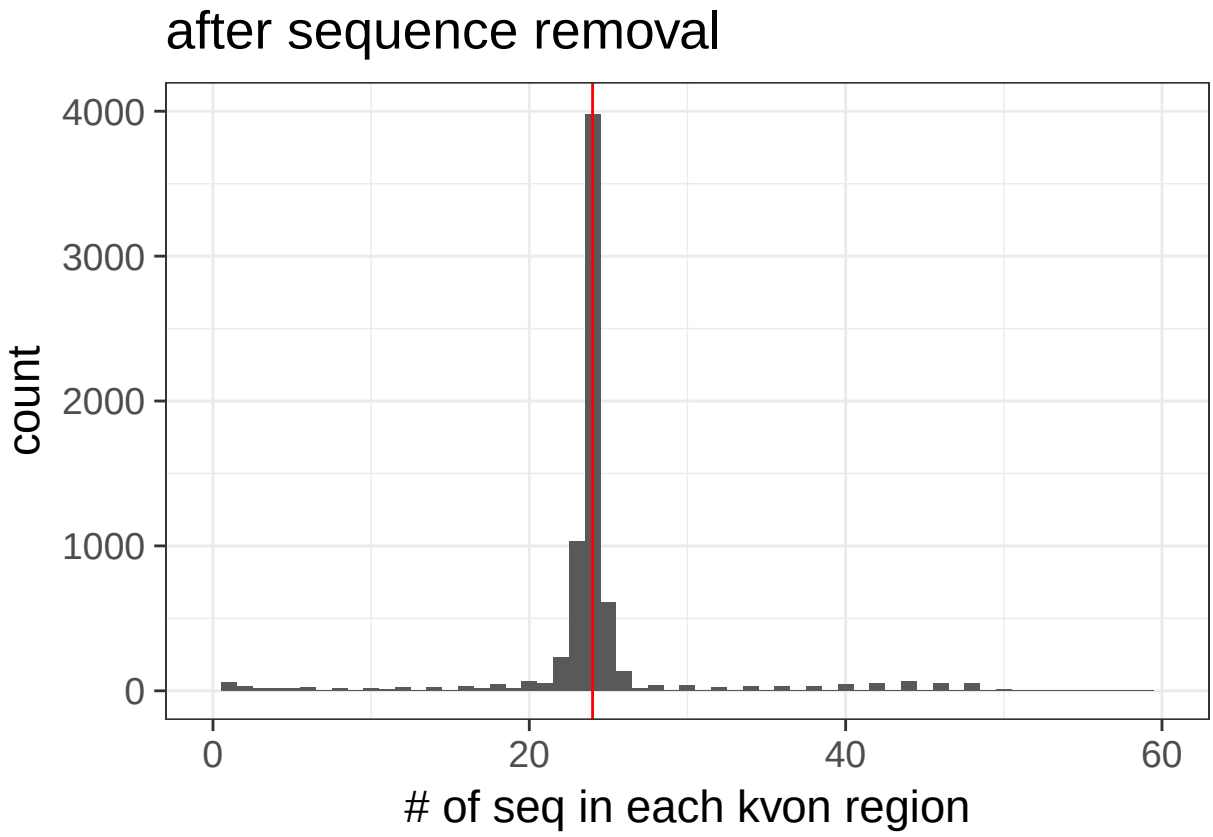

```
#######################################
## How many species?
#######################################

## before
out %>%
  group_by(region) %>%
  summarize(number_sequences = n()) %>%
  ggplot(., aes(number_sequences)) +
    geom_histogram(binwidth = 1) +
    theme_bw(base_size = 17) +
    geom_vline(xintercept = 24, color = "red") +
    xlim(0,60) +
    ylim(0,4000) +
    xlab("# of seq in each kvon region") +
    ggtitle("before sequence removal")
```

```
## Warning: Removed 99 rows containing non-finite values (stat_bin).
```




```
## after
out %>%
  filter(remove == "no") %>%
  group_by(region) %>%
  summarize(number_sequences = n()) %>%
  ggplot(., aes(number_sequences)) +
    geom_histogram(binwidth = 1) +
    theme_bw(base_size = 17) +
    geom_vline(xintercept = 24, color = "red") +
    xlim(0,60) +
    ylim(0,4000) +
    xlab("# of seq in each kvon region") +
    ggtitle("after sequence removal")
```

```
## Warning: Removed 3 rows containing non-finite values (stat_bin).
```




```
###############
## Conclusion
## At this point I am just going to output the list of fasta files that need
## to be removed and then filter for sequences that have 24 sequences with
## 24 represenative species. But I am going to do this in bash.
#####################

## Output entire dataset
## write.csv(out, "../data/output/out_all_data_from_QC_pipeline_4_kvon_outliers_1.csv")

head(out)
```

```
##   region expr  species strand length     mean outlier       score remove
## 1 VT9999    0     dkik      -    496 497.9167       1 -0.05436085    yes
## 2 VT9999    0 MEMB002B      -    422 497.9167       1 -2.15316241    yes
## 3 VT9999    0 MEMB002C      -    486 497.9167       1 -0.33798268    yes
## 4 VT9999    0 MEMB002D      -    490 497.9167       1 -0.22453395    yes
## 5 VT9999    0 MEMB002E      -    499 497.9167       1  0.03072570     no
## 6 VT9999    0 MEMB003A      -    492 497.9167       1 -0.16780958    yes
```

```
removed_seqs <- out %>%
  filter(remove == "yes")

## Check
nrow(removed_seqs)
```

```
## [1] 36675
```

```
## Now bring back into row

remove_list <- as.data.frame(paste(removed_seqs$region,
                     removed_seqs$expr,
                     removed_seqs$strand,
                     removed_seqs$length, sep = "|"  ))
colnames(remove_list)[1] <- "ID"

## Add bracket, delete unused rows
remove_list$bracket <- ">"
remove_list$fasta_headers <- paste0(remove_list$bracket, remove_list$ID)
remove_list <- as.data.frame(remove_list[,-c(1,2)])

## Check
head(remove_list)
```

```
##   remove_list[, -c(1, 2)]
## 1         >VT9999|0|-|496
## 2         >VT9999|0|-|422
## 3         >VT9999|0|-|486
## 4         >VT9999|0|-|490
## 5         >VT9999|0|-|492
## 6         >VT9999|0|-|495
```

```
## Alright, ready to export list of seq for removal
## write.table(remove_list, "../data/output/list_of_seq_for_removal_6March2018.txt",sep = "\t", col.nam
```