distribution of genomic features and their correlation with crossover count

S. Chmielewski

2024-04-03

The aim of this Rmd is to analyze the distribution of genomic features in 50 kb windows (genes, repeats and GC content) along chromosomes and to check the correlation between these features and recombination rate, expressed as number of crossovers within 2 Mb windows.

Correlation between genomic features and crossover count in 2 Mb windows

```
### load 2Mb windows:
mb2_windows <- read.table("~/LM_new_bams/data/genome_res/genome_2Mb_windows_chr.txt", h = T)
mb2_windows <- mb2_windows %>%
  mutate(LG = as.integer(gsub("chr", "", LG)))

### load chromosome sizes:
chromosome_physical_length <- read.table("~/LM_new_bams/data/genome_res/chromosome_lengths.txt", h = T)
chromosome_physical_length <- chromosome_physical_length %>%
  mutate(LG = as.integer(gsub("chr", "", LG)))
```

load genes

```
gff <- read.table("~/liftover/AMU_Rhrob_2024.gff3")

gff <- gff %>%
 filter(V3 == "gene" & str_detect(V1, "chr")) %>% # keep only genes and reject genes not included in ch
 mutate(V1 = as.integer(gsub("chr", "", V1))) %>%
 select(1, 4, 5, 9)
colnames(gff) <- c("LG", "gene_start", "gene_end", "gene_id")


#### Correct the genes: some genes overlap the window boundaries. When gene density is calulated in 50

window_boundaries = data.frame(pos = seq(from = 1e6, to = 5.1e7, by = 2e6)) # by = window size

# select genes which coordinates overlap window coordinates (these will be removed)
genes_overlapping_windows <- gff %>%
  expand_grid(window_boundaries) %>%
  filter(pos > gene_start & pos < gene_end)

# function to split genes in several parts:
split_genes_at_windows <- function(df) {
  result <- data.frame(LG = integer(), gene_start = integer(), gene_end = integer(), gene_id = characte

  for (i in 1:nrow(df)) {
    row <- df[i, ]
    overlapping_windows <- sort(unique(c(df$pos[df$pos >= row$gene_start & df$pos <= row$gene_end], row

    # create the first segment
    if (!is.null(overlapping_windows) && length(overlapping_windows) > 0) {
      result <- rbind(result, c(row$LG, row$gene_start, overlapping_windows[1], row$gene_id, overlappin

      # create additional segments if there are more widnows
      if (length(overlapping_windows) > 1) {
        for (j in 1:(length(overlapping_windows) - 1)) {
          start_pos = overlapping_windows[j] + 1
          end_pos = overlapping_windows[j + 1]
          if (start_pos <= end_pos) {
            new_row <- c(row$LG, start_pos, end_pos, row$gene_id, overlapping_windows[j + 1])
            result <- rbind(result, new_row)
          }
        }
      }
    } else {
      # if no overlapping window, keep the gene as is
      result <- rbind(result, row)
    }
  }

  colnames(result) <- c("LG", "gene_start", "gene_end", "gene_id", "pos")
  return(result)
}

split_genes_df <- split_genes_at_windows(genes_overlapping_windows)

split_genes_df <- split_genes_df %>%
  distinct() %>%
  mutate(LG = as.numeric(LG),
         gene_start = as.integer(gene_start),
         gene_end = as.integer(gene_end))

# remove genes overlapping window coordinates:
gff <- gff %>%
  anti_join(genes_overlapping_windows, by = c("LG", "gene_start", "gene_end", "gene_id")) %>%
  bind_rows(split_genes_df) %>%
  select(-pos)

### end of gene correction ###

#### keep only genes expressed in adults (mean FPKM > 1)
genes_expressed <- read.table("~/liftover/Mites_RNAseq_fpkms.txt", h = T)
```

```r
genes_expressed <- genes_expressed %>%
  filter(fpkmMEAN > 1)

# modify gff to get the gene name
gff <- gff %>%
  separate(gene_id, sep = ";", into = c("id", "Gene", "length")) %>%
   mutate(Gene = gsub("Name=", "", Gene)) %>%
    filter(Gene %in% genes_expressed$Gene)
```

load repeats

```r
repeats <- read.table("~/repeatMasking/liftovered_repeatModeller/AMU_Rhrob_2024_RepeatModeller.gff")
repeats <- repeats %>%
  filter(str_detect(V1, "chr")) %>%
  mutate(V1 = as.integer(gsub("chr", "", V1))) %>%
    select(V1, V4, V5)

colnames(repeats) <- c("LG", "repeat_start", "repeat_end")

#### Correct the repeats: some repeats overlap the window boundaries. When repeat density is calulated

# select repeats which corrdinates overlap window coordinates (these will be removed)
repeats_overlapping_windows <- repeats %>%
  expand_grid(window_boundaries) %>%
  filter(pos > repeat_start & pos < repeat_end)

split_repeats_at_windows <- function(df) {
  result <- data.frame(LG = integer(), repeat_start = integer(), repeat_end = integer(), pos = integer(

  for (i in 1:nrow(df)) {
    row <- df[i, ]
    overlapping_windows <- sort(unique(c(df$pos[df$pos >= row$repeat_start & df$pos <= row$repeat_end],

    # Create the first segment
    if (!is.null(overlapping_windows) && length(overlapping_windows) > 0) {
      result <- rbind(result, c(row$LG, row$repeat_start, overlapping_windows[1], overlapping_windows[1

      # create additional segments if there are more widnows
      if (length(overlapping_windows) > 1) {
        for (j in 1:(length(overlapping_windows) - 1)) {
          start_pos = overlapping_windows[j] + 1
          end_pos = overlapping_windows[j + 1]
          if (start_pos <= end_pos) {
            new_row <- c(row$LG, start_pos, end_pos, overlapping_windows[j + 1])
            result <- rbind(result, new_row)
          }
        }
      }
    } else {
      # If no overlapping window, keep the repeat as is
      result <- rbind(result, row)
    }
  }

  colnames(result) <- c("LG", "repeat_start", "repeat_end", "pos")
  return(result)
}

split_repeats_df <- split_repeats_at_windows(repeats_overlapping_windows)

split_repeats_df <- split_repeats_df %>%
  distinct() %>%
  mutate(LG = as.numeric(LG),
         repeat_start = as.integer(repeat_start),
         repeat_end = as.integer(repeat_end)) %>%
  select(-pos)

# remove repeats overlapping window coordinates:
repeats <- repeats %>%
  anti_join(repeats_overlapping_windows, by = c("LG", "repeat_start", "repeat_end")) %>%
  bind_rows(split_repeats_df)

##### end repeat correction ######
```

load GC content

```
GC_2mb_windows <- read.table("~/LM_new_bams/data/genome_res/GC_2Mb_windows_renamed_sorted.gff")[,c(1:3,
colnames(GC_2mb_windows) <- c("LG", "window_start", "window_end", "GC")
GC_2mb_windows$LG <- as.integer(gsub("chr", "", GC_2mb_windows$LG))
```

count the proportion of genomic factors in 2-mb windows

```
### 1) gene proportion:
prop_CDS_length_2mb_windows <- gff %>%
  mutate(gene_length = abs(gene_end - gene_start)) %>%
  left_join(mb2_windows, by = "LG", relationship = "many-to-many") %>%
  filter((gene_start + gene_end)/2 > window_start & (gene_start + gene_end)/2 <= window_end) %>%
  group_by(LG, window_start, window_end) %>%
  summarise(gene_length_sum = sum(gene_length)) %>%
  mutate(gene_prop = gene_length_sum/(window_end - window_start + 1))
```

```
## `summarise()` has grouped output by 'LG', 'window_start'. You can override
## using the `.groups` argument.
```

```
### 2) repeat proportion:
prop_repeat_length_2mb_windows <- repeats %>%
    mutate(repeat_length = repeat_end - repeat_start) %>%
  left_join(mb2_windows, by = "LG", relationship = "many-to-many") %>%
  filter((repeat_start + repeat_end)/2 > window_start & (repeat_start + repeat_end)/2 <= window_end) %>
  group_by(LG, window_start, window_end) %>%
  summarise(repeat_length_sum = sum(repeat_length)) %>%
  mutate(repeat_proportion = repeat_length_sum/(window_end - window_start + 1))
```

```
## ‘summarise()‘ has grouped output by ’LG’, ’window_start’. You can override
## using the ‘.groups‘ argument.
```

```
### 3) distance to chromosome end:
distance_chr_end_2mb_windows <- mb2_windows %>%
  left_join(chromosome_physical_length, by = c("LG"), relationship = "many-to-many") %>%
  group_by(LG, window_start, window_end) %>%
  mutate(closer_chr_end = ifelse((window_end+window_start)/2 > physical_length/2, physical_length, 0),
         distance_chr_end = abs(closer_chr_end-(window_end+window_start)/2))
```

count crossovers in 2Mb windows

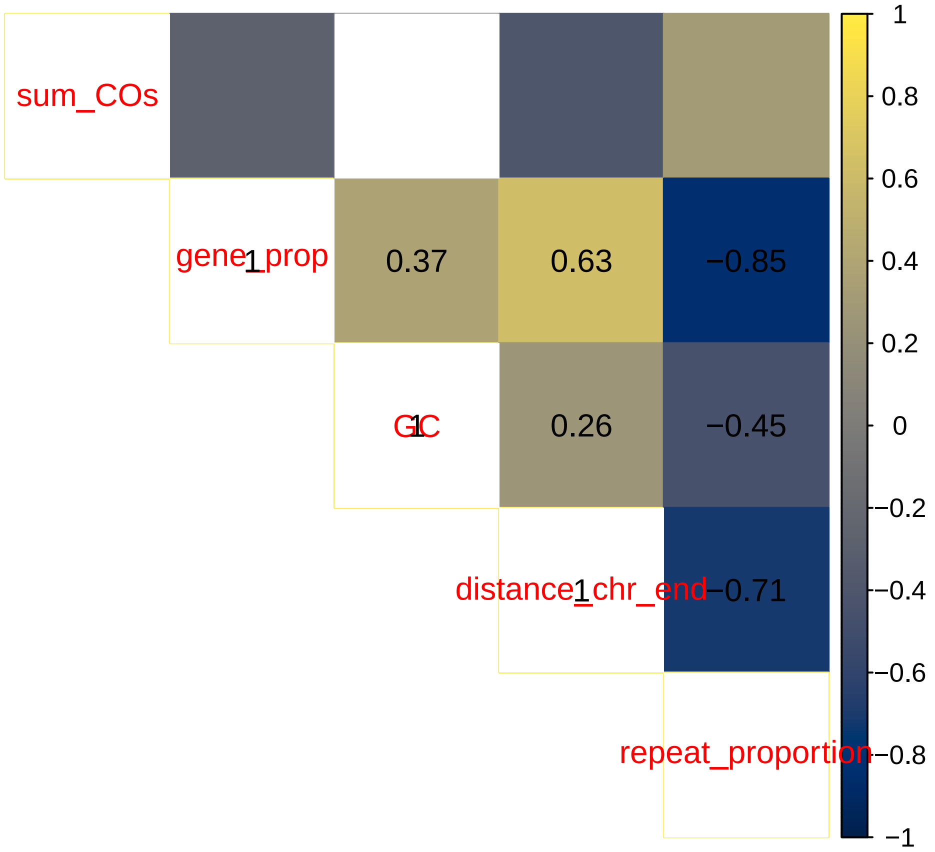
```
# output of the script CO_location_CO_count.Rmd
all_chr_CO_location <- read.table("~/LM_new_bams/results/16.ordering_after_LA/final_ordering/no_scaling

# rename linkage groups to match chromosome sizes:
all_chr_CO_location <- all_chr_CO_location %>%
     mutate(LG = recode(LG,
                          "7" = "1",
                          "5" = "2",
                          "3" = "3",
                          "1" = "4",
                          "8" = "5",
                          "4" = "6",
                          "2" = "7",
                          "6" = "8")) %>%
   mutate(LG = as.numeric(LG, levels = c("1", "2", "3", "4", "5", "6", "7", "8"))) %>%
   arrange(LG)

CO_location_2mb_windows <- all_chr_CO_location %>%
  full_join(mb2_windows, by = "LG", relationship = "many-to-many") %>%
  filter(LG_CO_pos > window_start & LG_CO_pos <= window_end) %>%
  group_by(LG, window_start, window_end) %>%
  tally(name = "sum_COs") %>%
  full_join(mb2_windows, by = c("LG", "window_start", "window_end"),relationship = "many-to-many") %>%
  replace(is.na(.), 0) %>%
  arrange(LG, window_start)
```

Calculate the correlation between genomic features

```
model_n_COs <- CO_location_2mb_windows %>%
#  ungroup() %>%
  left_join(prop_CDS_length_2mb_windows, by = c("LG", "window_start", "window_end")) %>%
  left_join(GC_2mb_windows, by = c("LG", "window_start", "window_end")) %>%
  left_join(distance_chr_end_2mb_windows, by = c("LG", "window_start", "window_end")) %>%
  left_join(prop_repeat_length_2mb_windows, by = c("LG", "window_start", "window_end")) %>%
  mutate(sum_COs = as.numeric(sum_COs))

colSums(is.na(model_n_COs))
```

```
##                LG       window_start         window_end            sum_COs
##                 0                  0                  0                  0
##   gene_length_sum          gene_prop                 GC    physical_length
##                 0                  0                  0                  0
##    closer_chr_end  distance_chr_end repeat_length_sum repeat_proportion
##                 0                  0                  0                  0
```

```
# select variables for the correlation
data_cor <- model_n_COs[,c(4, 6, 7, 10, 12)]

testRes <- cor.mtest(as.matrix(data_cor))
cor_matrix <- cor(data_cor, method = "pearson")

corrplot(cor_matrix, p.mat = testRes$p, method = 'color', type = 'upper', insig='blank', col = viridis(
text(p1$x, p1$y, round(p1$corr, 2))
```



Genomic features in 50 kb windows

```
#### load 50-kb genomic windows ####
kb50_windows <- read.table("~/LM_new_bams/data/genome_res/genome_50kb_windows.txt", col.names = c("LG",

kb50_windows <- kb50_windows %>%
 mutate(LG = as.integer(gsub("chr", "", LG)))


##### 1) load gene count ####
gff <- read.table("~/liftover/AMU_Rhrob_2024.gff3")

gff <- gff %>%
 filter(V3 == "gene" & str_detect(V1, "chr")) %>% # keep only genes and reject genes not included in ch
 mutate(V1 = as.integer(gsub("chr", "", V1))) %>%
 select(1, 4, 5, 9)
colnames(gff) <- c("LG", "gene_start", "gene_end", "gene_id")

# swap the values of gene_start and gene_end, if the gene_start is larger than gene_end:
gff <- gff %>%
  mutate(
    new_gene_start = if_else(gene_start > gene_end, gene_end, gene_start),
    new_gene_end = if_else(gene_start > gene_end, gene_start, gene_end)
  ) %>%
  select(-gene_start, -gene_end) %>%
  rename(gene_start = new_gene_start, gene_end = new_gene_end)

#### Correct the genes: some genes overlap the window boundaries. When gene density is calulated in 50 

window_boundaries_50kb = data.frame(pos = seq(from = 1e6, to = max(gff$gene_end), by = 5e4)) # by = win

# select genes which coordinates overlap window coordinates (these will be removed)
genes_overlapping_windows <- gff %>%
  expand_grid(window_boundaries_50kb) %>%
  filter(pos > gene_start & pos < gene_end)

# function to split genes in several parts:
split_genes_at_windows <- function(df) {
  result <- data.frame(LG = integer(), gene_start = integer(), gene_end = integer(), gene_id = characte

  for (i in 1:nrow(df)) {
    row <- df[i, ]
    overlapping_windows <- sort(unique(c(df$pos[df$pos >= row$gene_start & df$pos <= row$gene_end], row

    # create the first segment
    if (!is.null(overlapping_windows) && length(overlapping_windows) > 0) {
      result <- rbind(result, c(row$LG, row$gene_start, overlapping_windows[1], row$gene_id, overlappin

      # create additional segments if there are more windows
      if (length(overlapping_windows) > 1) {
        for (j in 1:(length(overlapping_windows) - 1)) {
          start_pos = overlapping_windows[j] + 1
          end_pos = overlapping_windows[j + 1]
          if (start_pos <= end_pos) {
            new_row <- c(row$LG, start_pos, end_pos, row$gene_id, overlapping_windows[j + 1])
            result <- rbind(result, new_row)
          }
        }
      }
    } else {
      # if no overlapping window, keep the gene as is
      result <- rbind(result, row)
    }
  }

  colnames(result) <- c("LG", "gene_start", "gene_end", "gene_id", "pos")
  return(result)
}

split_genes_df <- split_genes_at_windows(genes_overlapping_windows)

split_genes_df <- split_genes_df %>%
  distinct() %>%
  mutate(LG = as.integer(LG),
         gene_start = as.integer(gene_start),
         gene_end = as.integer(gene_end),
         pos = as.numeric(pos))

# remove genes overlapping window coordinates:
gff <- gff %>%
  anti_join(genes_overlapping_windows, by = c("LG", "gene_start", "gene_end", "gene_id")) %>%
  bind_rows(split_genes_df)

# END OF THE GENE CORRECTION

### include only genes which are expressed:

#### keep only genes expressed in adults (mean FPKM > 1)
genes_expressed <- read.table("~/liftover/Mites_RNAseq_fpkms.txt", h = T)
genes_expressed <- genes_expressed %>%
  filter(fpkmMEAN > 1)

# modify gff to get the gene name
gff <- gff %>%
   separate(gene_id, sep = ";", into = c("id", "Gene", "length")) %>%
   mutate(Gene = gsub("Name=", "", Gene)) %>%
   filter(Gene %in% genes_expressed$Gene)

#### 2) load the repeats ####
repeats <- read.table("~/repeatMasking/liftovered_repeatModeller/AMU_Rhrob_2024_RepeatModeller.gff")
repeats <- repeats %>%
  filter(str_detect(V1, "chr")) %>%
  mutate(V1 = as.integer(gsub("chr", "", V1))) %>%
    select(V1, V4, V5)

colnames(repeats) <- c("LG", "repeat_start", "repeat_end")
```

```r
#### Correct the repeats: some repeats overlap the window boundaries. When gene density is calulated in

window_boundaries_50kb = data.frame(pos = seq(from = 1e6, to = max(repeats$repeat_end), by = 5e4)) # by

# select genes which corrdinates overlap window coordinates (these will be removed)
repeats_overlapping_windows <- repeats %>%
  expand_grid(window_boundaries_50kb) %>%
  filter(pos > repeat_start & pos < repeat_end)

split_repeats_at_windows <- function(df) {
  result <- data.frame(LG = integer(), repeat_start = integer(), repeat_end = integer(), pos = integer(

  for (i in 1:nrow(df)) {
    row <- df[i, ]
    overlapping_windows <- sort(unique(c(df$pos[df$pos >= row$repeat_start & df$pos <= row$repeat_end],

    # Create the first segment
    if (!is.null(overlapping_windows) && length(overlapping_windows) > 0) {
      result <- rbind(result, c(row$LG, row$repeat_start, overlapping_windows[1], overlapping_windows[1

      # Create additional segments if there are more windows
      if (length(overlapping_windows) > 1) {
        for (j in 1:(length(overlapping_windows) - 1)) {
          start_pos = overlapping_windows[j] + 1
          end_pos = overlapping_windows[j + 1]
          if (start_pos <= end_pos) {
            new_row <- c(row$LG, start_pos, end_pos, overlapping_windows[j + 1])
            result <- rbind(result, new_row)
          }
        }
      }
    } else {
      # If no overlapping window, keep the repeat as is
      result <- rbind(result, row)
    }
  }

  colnames(result) <- c("LG", "repeat_start", "repeat_end", "pos")
  return(result)
}

split_repeats_df <- split_repeats_at_windows(repeats_overlapping_windows)

split_repeats_df <- split_repeats_df %>%
  distinct() %>%
  mutate(LG = as.integer(LG),
         repeat_start = as.integer(repeat_start),
         repeat_end = as.integer(repeat_end))

# remove repeats overlapping window coordinates:
repeats <- repeats %>%
  anti_join(repeats_overlapping_windows, by = c("LG", "repeat_start", "repeat_end")) %>%
  bind_rows(split_repeats_df)
# END repeat CORRECTION #


#### 3) load GC ####

GC_50kb_windows <- read.table("~/LM_new_bams/data/genome_res/GC_50kb_windows.txt")[,c(1:3,5)]
colnames(GC_50kb_windows) <- c("LG", "window_start", "window_end", "GC")
GC_50kb_windows <- GC_50kb_windows %>%
  filter(str_detect(LG, "^chr"))
GC_50kb_windows$LG <- as.integer(gsub("chr", "", GC_50kb_windows$LG))

#### 5) load number of crossovers ####
CO_location_50kb_windows <- all_chr_CO_location %>%
  full_join(kb50_windows, by = "LG") %>%
  filter(LG_CO_pos > window_start & LG_CO_pos <= window_end) %>%
  group_by(LG, window_start, window_end) %>%
  tally(name = "sum_COs")
```

```
## Warning in full_join(., kb50_windows, by = "LG"): Detected an unexpected many-to-many relationship be
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

distribution of genomic features in 50 kb windows

```
# calculate the fraction of gene length in windows:
prop_CDS_length_50kb_windows <- gff %>%
  mutate(gene_length = abs(gene_end - gene_start)) %>%
  left_join(kb50_windows, by = "LG", relationship = "many-to-many") %>%
  filter((gene_start + gene_end)/2 > window_start & (gene_start + gene_end)/2 <= window_end) %>%
  group_by(LG, window_start, window_end) %>%
  summarise(gene_prop = sum(gene_length)) %>%
  mutate(gene_prop = gene_prop/(window_end - window_start + 1))
```

```
## `summarise()` has grouped output by 'LG', 'window_start'. You can override
## using the `.groups` argument.
```

```
# calculate the fraction of repeats length in windows:
prop_repeat_length_50kb_windows <- repeats %>%
    mutate(repeat_length = repeat_end - repeat_start) %>%
  left_join(kb50_windows, by = "LG") %>%
  filter((repeat_start + repeat_end)/2 > window_start & (repeat_start + repeat_end)/2 <= window_end) %>
  group_by(LG, window_start, window_end) %>%
  summarise(repeat_proportion = sum(repeat_length)) %>%
  mutate(repeat_proportion = repeat_proportion/(window_end - window_start + 1))
```

```
## Warning in left_join(., kb50_windows, by = "LG"): Detected an unexpected many-to-many relationship be
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set ‘relationship =
##   "many-to-many"‘ to silence this warning.

## ‘summarise()‘ has grouped output by ’LG’, ’window_start’. You can override
## using the ‘.groups‘ argument.
```

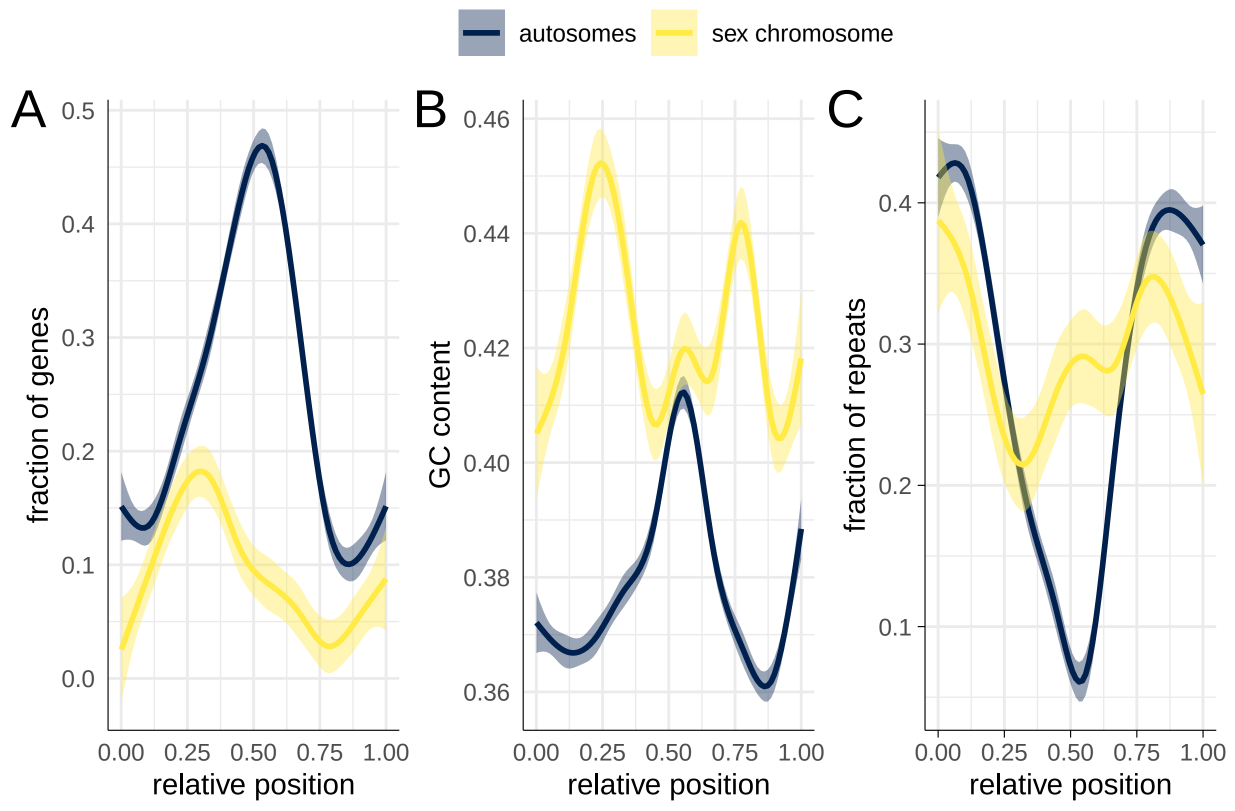
```
#### 4) calculate the distance to chromosome end ####
distance_chr_end_50kb_windows <- kb50_windows %>%
  left_join(chromosome_physical_length, by = c("LG")) %>%
  group_by(LG, window_start, window_end) %>%
  mutate(closer_chr_end = ifelse((window_end+window_start)/2 > physical_length/2, physical_length, 0),
         distance_chr_end = abs(closer_chr_end-(window_end+window_start)/2)) %>%
  dplyr::select(-c(physical_length, closer_chr_end))

#### merge the data ####
data_genomic_parameters_50kb <- prop_CDS_length_50kb_windows %>%
  full_join(prop_repeat_length_50kb_windows, by = c("LG", "window_start", "window_end")) %>%
  full_join(GC_50kb_windows, by = c("LG", "window_start", "window_end")) %>%
  full_join(distance_chr_end_50kb_windows, by = c("LG", "window_start", "window_end")) %>%
  full_join(CO_location_50kb_windows, by = c("LG", "window_start", "window_end")) %>%
  replace(is.na(.), 0) %>%
  left_join(chromosome_physical_length, by = "LG") %>%
  mutate(relative_position = (window_start + window_end)/2/physical_length)


#### plot Figure 3 ####
gene_distribution_along_chr <- data_genomic_parameters_50kb %>%
  mutate(sex_chromosome = ifelse(LG == "6", "sex chromosome", "autosomes")) %>%
  ggplot(aes(relative_position, gene_prop, color = sex_chromosome, fill = sex_chromosome)) +
  #geom_point(alpha = 0.1) +
  geom_smooth() +
  theme_minimal() +
  theme(legend.position = "none",
        plot.tag = element_text(size = 20),
        plot.tag.position = c(0.005, .86),
        axis.line = element_line(size = .2, colour = "black"),
        legend.box.margin = margin(t = -5, b = -5, unit = "mm"),
        legend.spacing.y = unit(0, "mm")) +
  viridis::scale_color_viridis(discrete = T, option = "E") +
  viridis::scale_fill_viridis(discrete = T, option = "E") +
  labs(tag = "A", y = "fraction of genes", x = "relative position")
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
GC_distribution_along_chr <- data_genomic_parameters_50kb %>%
  mutate(sex_chromosome = ifelse(LG == "6", "sex chromosome", "autosomes")) %>%
  ggplot(aes(relative_position, GC, color = sex_chromosome, fill = sex_chromosome)) +
  geom_smooth() +
  theme_minimal() +
```

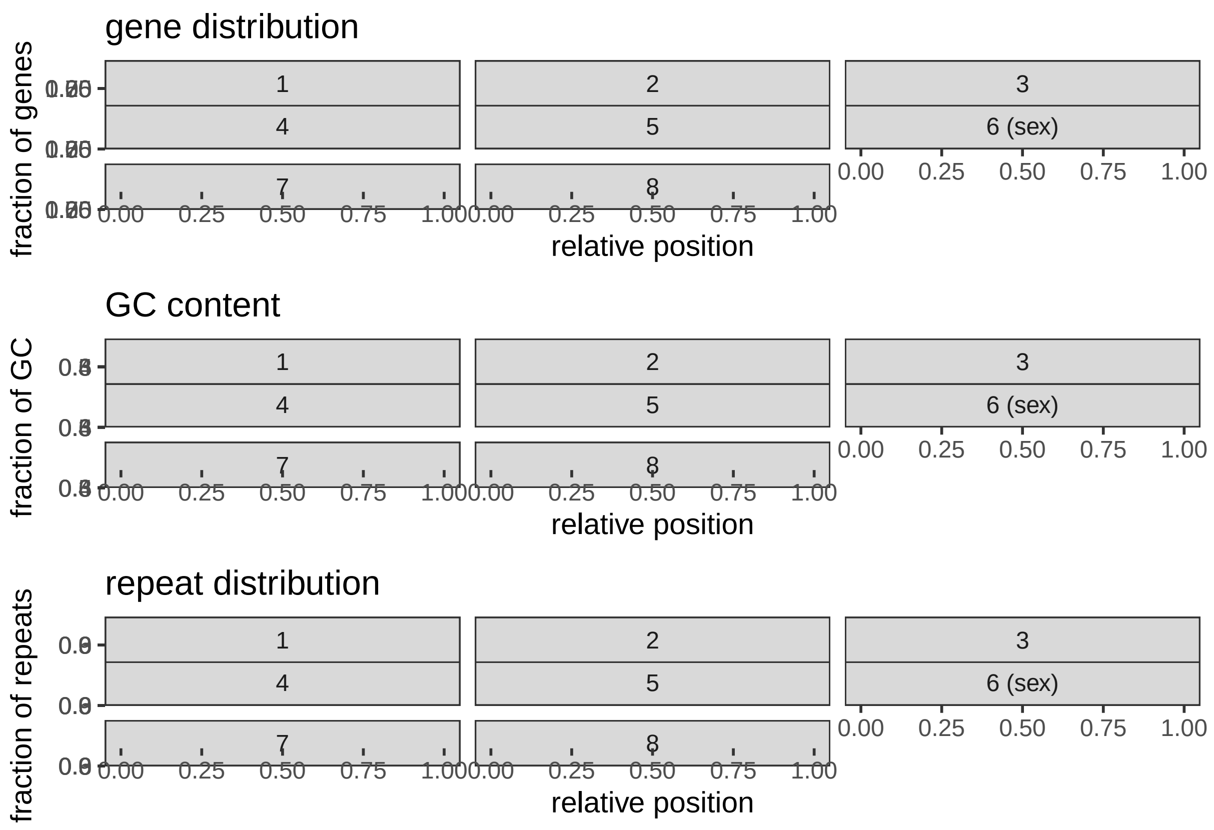
```r
  theme(legend.position = "top",
        legend.title = element_blank(),
        plot.tag = element_text(size = 20),
        plot.tag.position = c(0.005, .86),
        axis.line = element_line(size = .2, colour = "black")) +
  viridis::scale_color_viridis(discrete = T, option = "E") +
  viridis::scale_fill_viridis(discrete = T, option = "E") +
  labs(tag = "B", y = "GC content", x = "relative position")

repeat_distribution_along_chr <- data_genomic_parameters_50kb %>%
  mutate(sex_chromosome = ifelse(LG == "6", "sex chromosome", "autosomes")) %>%
  ggplot(aes(relative_position, repeat_proportion, color = sex_chromosome, fill = sex_chromosome)) +
  geom_smooth() +
  theme_minimal() +
  theme(legend.position = "none",
        plot.tag = element_text(size = 20),
        plot.tag.position = c(0.005, .86),
        axis.line = element_line(size = .2, colour = "black"),
        axis.ticks = element_line(size = .2, color="black")) +
  viridis::scale_color_viridis(discrete = T, option = "E") +
  viridis::scale_fill_viridis(discrete = T, option = "E") +
  labs(tag = "C", y = "fraction of repeats" , x = "relative position")


Figure3 <- (gene_distribution_along_chr + GC_distribution_along_chr + repeat_distribution_along_chr)

print(Figure3)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



```r
### per chromosome:
gene_distribution_along_per_chr <- data_genomic_parameters_50kb %>%
  mutate(LG = ifelse(LG == "6", "6 (sex)", as.character(LG))) %>%
  ggplot(aes(relative_position, gene_prop)) +
  geom_point(alpha = 0.1) +
  geom_smooth() +
  theme_bw() +
  scale_color_manual(values = c("#4D9223","#C6377E")) +
  scale_fill_manual(values = c("#D8E8CE", "#F1CFE4")) +
  labs(y = "fraction of genes", x = "relative position", title = "gene distribution") +
  facet_wrap(~LG)

GC_distribution_along_per_chr <- data_genomic_parameters_50kb %>%
  mutate(LG = ifelse(LG == "6", "6 (sex)", as.character(LG))) %>%
  ggplot(aes(relative_position, GC)) +
  geom_point(alpha = 0.1) +
  geom_smooth() +
  theme_bw() +
  scale_color_manual(values = c("#4D9223","#C6377E")) +
  scale_fill_manual(values = c("#D8E8CE", "#F1CFE4")) +
  labs(y = "fraction of GC", x = "relative position", title = "GC content") +
  facet_wrap(~LG)

repeat_distribution_along_per_chr <- data_genomic_parameters_50kb %>%
  mutate(LG = ifelse(LG == "6", "6 (sex)", as.character(LG))) %>%
  ggplot(aes(relative_position, repeat_proportion)) +
  geom_point(alpha = 0.1) +
  geom_smooth() +
  theme_bw() +
  scale_color_manual(values = c("#4D9223","#C6377E")) +
  scale_fill_manual(values = c("#D8E8CE", "#F1CFE4")) +
  labs(y = "fraction of repeats", x = "relative position", title = "repeat distribution") +
  facet_wrap(~LG)

plot_layout <- (gene_distribution_along_per_chr / GC_distribution_along_per_chr / repeat_distribution_al
print(plot_layout)
```

```
## 'geom_smooth()' using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## 'geom_smooth()' using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## 'geom_smooth()' using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



```
### test if the density of repeats differ between autosomes and the sex chromosome
  # repeats: autosomes vs sex chromosome
data_genomic_parameters_50kb <- data_genomic_parameters_50kb %>%
  mutate(sex_chromosome = ifelse(LG == "6", "sex chromosome", "autosome"))

t.test(repeat_proportion ~ sex_chromosome, data = data_genomic_parameters_50kb)
```

```
##
##  Welch Two Sample t-test
```

```
##
## data:  repeat_proportion by sex_chromosome
## t = -2.6547, df = 831.09, p-value = 0.00809
## alternative hypothesis: true difference in means between group autosome and group sex chromosome is
## 95 percent confidence interval:
##  -0.035359553 -0.005298008
## sample estimates:
##      mean in group autosome mean in group sex chromosome
##                   0.2746889                    0.2950177
```

```r
t.test(gene_prop ~ sex_chromosome, data = data_genomic_parameters_50kb)
```

```
##
##  Welch Two Sample t-test
##
## data:  gene_prop by sex_chromosome
## t = 25.172, df = 1107.9, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group autosome and group sex chromosome is
## 95 percent confidence interval:
##  0.1426508 0.1667694
## sample estimates:
##      mean in group autosome mean in group sex chromosome
##                  0.24755638                   0.09284628
```