# Paint-Chromosome

CR Cardenas

2023-02

## Paint UCEs on chromosomes

Tasks:

1) modfiy final figure to include multiple chromosomes (e.g., chr 1 has 2 examples, one with UCE painting and one with all gene painting)
2) check outputs from previous rmd work
3) check column classes, was having issues with that previously
4) determine best course of action for scripts, Jeremy recomended thinking about the distribution of UCEs.
    - something like the number of UCEs per Mbp along the genome?
    - are the genes randomly distributed across the genome
    - are UCEs randomly distributed across the genome?
    - compare the distribution of genes and UCEs. This is probably the biggest challenge because genes aren't normally distributed across the genome.
5) Think about why you want to do these statistics, how informative is it that these genes are doing what they are doing? Is it better to just know gene identities?

*R dependency hell on linux* search for r-cran dependencies on apt (`sudo apt search r-cran-...`)

```
library(readr)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v purrr   1.0.1     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(RIdeogram)
```

In particular, I am using this software to image the UCE loci positions: make sure you reference them: http://doi.org/10.7717/peerj-cs.251

please see the RIdeogram vignette for data format: https://cran.r-project.org/web/packages/RIdeogram/vignettes/RIdeogram.html

I will be reformatting the output from my intersect R data wrangling to work with this. *Note*, because I don't have centromere information for chromosomes we do not ned to include that here.

```
karyotype <- read_tsv("./pterMadi2.genomefile", col_names = c("chr","end")) %>%
  mutate(start = c("0")) %>%
  select(chr, start, end) %>%
  rename(Chr = chr) %>%
  rename(Start = start) %>%
  rename(End = end) %>%
  as.data.frame()
```

```
## Rows: 26 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): chr
## dbl (1): end
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df.karyotype <- karyotype %>% .[-c(19:25),] # no UCEs mapped to non-chromosomal scaffolds
df.karyotype
```

```
##    Chr Start      End
## 1    1     0 47997105
## 2    2     0 46201899
## 3    3     0 45368881
## 4    4     0 44840273
## 5    5     0 42010453
## 6    6     0 41309295
## 7    7     0 40830839
## 8    8     0 38739167
## 9    9     0 37879541
## 10  10     0 37211885
## 11  11     0 35335624
## 12  12     0 35328604
## 13  13     0 33816892
## 14  14     0 32211457
## 15  15     0 29678484
## 16  16     0 28091952
## 17  17     0 27886808
## 18  18     0 23989934
## 26   X     0 36190912
```

check previous output formats *OR* read_tsv formats. Had to tr and sed inorder to clean the output file from other R project. `tr -s " " \\t < just_pterostichus_probes_intergenic-genetic.tsv | sed 's/"//g' | > pterMadi2.intergeneic-genetic.tsv`

```
d.pterMadi2.genes <- read_tsv("./just_pterostichus_probes_intergenic.tsv", col_names = T) %>% na.omit()
```

```
## Rows: 4913 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (8): scaffold, query, uce, probe, type, seqname, biotype, gene_id
## dbl (4): qstart, qend, seqstart, seqend
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pterMadi2.genes <- read_tsv("./just_pterostichus_probes_intergenic.tsv", col_names = T) %>%
  distinct(uce, .keep_all = T) %>%
  rename(Chr = scaffold) %>%
  rename(Start = qstart) %>%
  rename(End = qend) %>%
  mutate(Value = case_when(biotype == "protein_coding" ~ "100",
                           biotype != "protein_coding" ~ "1",
                           type == "intergenic" ~ "50")) %>%
  select(Chr, Start, End, Value) %>%
  as.data.frame()
```

```
## Rows: 4913 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (8): scaffold, query, uce, probe, type, seqname, biotype, gene_id
## dbl (4): qstart, qend, seqstart, seqend
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

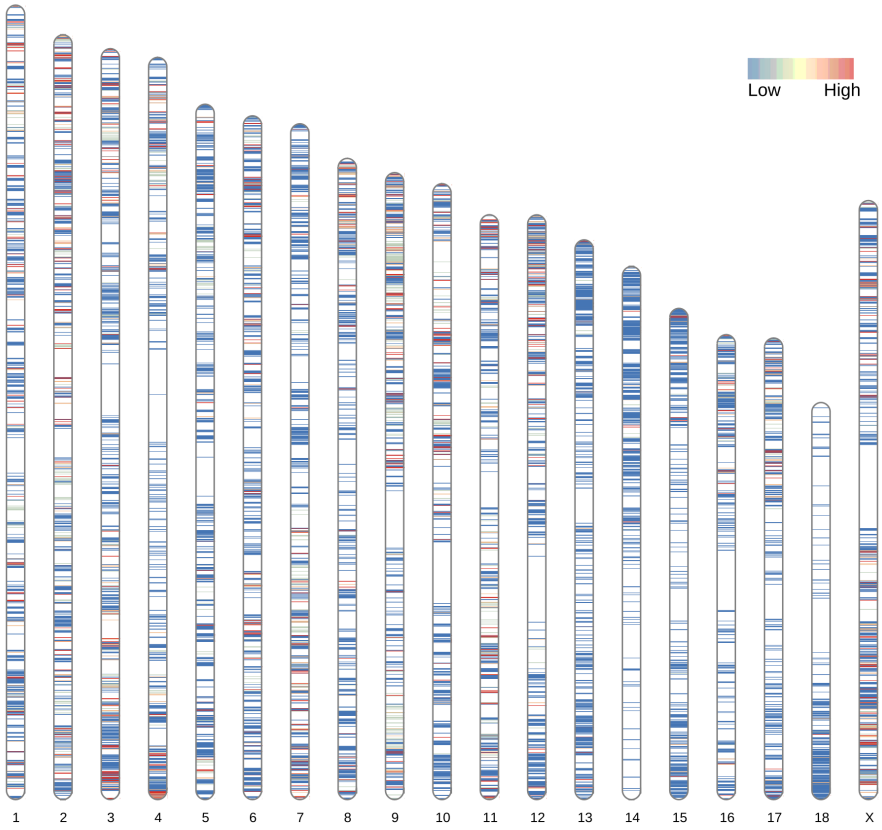
```
pterMadi2.genes$Value <- as.numeric(pterMadi2.genes$Value)
pterMadi2.genes %>% head()
```

```
##   Chr   Start     End Value
## 1   1   39310   39419     1
## 2   1   59597   59717   100
## 3   1  108107  108227    50
## 4   1 1011172 1011292   100
## 5   1 1018435 1018555   100
## 6   1 1035423 1035525   100
```

```
# turning these off for knitting HTML
# ideogram(karyotype = df.karyotype, overlaid = pterMadi2.genes, output = "temp.svg")
# convertSVG("temp.svg", device = "png")
```

Rideogram doesnt save to a plot in R studio, just the directory. This is fine, it needs to be edited in InkScape to present the data in a non "heatmap" format. We need to integrate the non-UCE gene regions as well.

E.g., known coding regions == 100, unknown coding regions == 66, intergenic == 33, and non-uce genes ==




0.

## Get genes that dont have UCEs

This will require me to go back into the bash commandline and make sure I include a file that has the *non* mapped regions.

With the original gff file I'll extract all genes, and then exclude them if they are already present in the data. There is probably a bedtools function that can let me extract gene features *not* mapped. For now I can do that using awk and the GFF file.

```
$ awk '$3 == "gene" {print $1 "\t" $4-1 "\t" $5-1 "\t" $3 "\t" $9}' pterMadi2.sorted.gff
> pterMadi2.sorted.gene.gff
```

Using the previous formatting we can get the bare minimum data we need to generate a list of gene features that did not map. We need the *.sorted.gene.gff & the intersect file

```
d.genes <- read_tsv(file="./pterMadi2.sorted.gene.gff", col_names = F, na = c("","NA"))
```

```
## Rows: 11637 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (3): X1, X4, X5
## dbl (2): X2, X3
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
colnames(d.genes) <- c("Chr","Start","End","Type","attribute")
d.genes
```

```
## # A tibble: 11,637 x 5
##    Chr    Start    End Type  attribute
##    <chr>  <dbl>  <dbl> <chr> <chr>
##  1 1      34012  39462 gene  ID=gene:ENSMPTG00005029508;Name=GTF2H1;biotype=pro~
##  2 1      40515  41942 gene  ID=gene:ENSMPTG00005017539;biotype=protein_coding;~
##  3 1      42431  43302 gene  ID=gene:ENSMPTG00005024373;Name=EMB;biotype=protei~
##  4 1      43643  47503 gene  ID=gene:ENSMPTG00005020641;biotype=protein_coding;~
##  5 1      47051  55369 gene  ID=gene:ENSMPTG00005018223;biotype=protein_coding;~
##  6 1      58919  66729 gene  ID=gene:ENSMPTG00005026164;biotype=protein_coding;~
##  7 1      60373  62454 gene  ID=gene:ENSMPTG00005026449;Name=si:ch211-200p22.4;~
##  8 1      67940  72313 gene  ID=gene:ENSMPTG00005024697;Name=CCK;biotype=protei~
##  9 1     146673 147542 gene  ID=gene:ENSMPTG00005020824;Name=NEUROG1;biotype=pr~
## 10 1     566522 569725 gene  ID=gene:ENSMPTG00005026310;Name=NAA50;biotype=prot~
## # ... with 11,627 more rows
```

```
# now split the columns
df.genes <- d.genes %>%
  separate_wider_delim(cols = attribute,
                       delim = ";",
                       names = c("ID","Name","biotype1","descrip1")) %>%
  separate_wider_delim(cols = ID,
                       delim = ":",
                       names = c("temp1","gene_id")) %>%
  separate_wider_delim(cols = Name,
                       delim = "=",
```

```
                      names = c("descrip2","biotype")) %>%
  select(Chr, Start, End, Type, biotype, gene_id)
df.genes
```

```
## # A tibble: 11,637 x 6
##    Chr    Start    End Type  biotype           gene_id
##    <chr>  <dbl>  <dbl> <chr> <chr>             <chr>
##  1 1      34012  39462 gene  GTF2H1            ENSMPTG00005029508
##  2 1      40515  41942 gene  protein_coding    ENSMPTG00005017539
##  3 1      42431  43302 gene  EMB               ENSMPTG00005024373
##  4 1      43643  47503 gene  protein_coding    ENSMPTG00005020641
##  5 1      47051  55369 gene  protein_coding    ENSMPTG00005018223
##  6 1      58919  66729 gene  protein_coding    ENSMPTG00005026164
##  7 1      60373  62454 gene  si:ch211-200p22.4 ENSMPTG00005026449
##  8 1      67940  72313 gene  CCK               ENSMPTG00005024697
##  9 1     146673 147542 gene  NEUROG1           ENSMPTG00005020824
## 10 1     566522 569725 gene  NAA50             ENSMPTG00005026310
## # ... with 11,627 more rows
```

Next step is to exclude genes that have already been mapped. Using the `d.pterMadi2.genes` DF as a reference/lookup object, we can exclude any gene that has been mapped. Additionally, we add a new column, `Value`, that gives us something to map with later. See next section

```
# make list of Gene IDs with UCEs mapped to them
list <- d.pterMadi2.genes %>% distinct(gene_id)
unmapped <- df.genes[ !df.genes$gene_id %in% list$gene_id,]

# use the ! negation and %in% logic
pterMadi2.genes.unmapped <- unmapped %>%
  mutate(Value = 0) %>%
  select(Chr, Start, End, Value) %>%
  as.data.frame()
pterMadi2.genes.unmapped$Value <- as.numeric(pterMadi2.genes.unmapped$Value)

list
```

```
## # A tibble: 1,591 x 1
##    gene_id
##    <chr>
##  1 ENSMPTG00005029508
##  2 ENSMPTG00005026164
##  3 ENSMPTG00005021500
##  4 ENSMPTG00005026854
##  5 ENSMPTG00005031604
##  6 ENSMPTG00005016919
##  7 ENSMPTG00005021805
##  8 ENSMPTG00005025850
##  9 ENSMPTG00005018169
## 10 ENSMPTG00005029447
## # ... with 1,581 more rows
```

```
unmapped
```

```
## # A tibble: 10,046 x 6
##    Chr    Start    End Type  biotype           gene_id
##    <chr>  <dbl>  <dbl> <chr> <chr>             <chr>
##  1 1      40515  41942 gene  protein_coding    ENSMPTG00005017539
##  2 1      42431  43302 gene  EMB               ENSMPTG00005024373
##  3 1      43643  47503 gene  protein_coding    ENSMPTG00005020641
##  4 1      47051  55369 gene  protein_coding    ENSMPTG00005018223
##  5 1      60373  62454 gene  si:ch211-200p22.4 ENSMPTG00005026449
##  6 1      67940  72313 gene  CCK               ENSMPTG00005024697
##  7 1     146673 147542 gene  NEUROG1           ENSMPTG00005020824
##  8 1     566522 569725 gene  NAA50             ENSMPTG00005026310
##  9 1     636135 637114 gene  protein_coding    ENSMPTG00005022354
## 10 1     855660 856328 gene  NEUROG1           ENSMPTG00005021054
## # ... with 10,036 more rows
```

```
pterMadi2.genes.unmapped %>% head()
```

```
##   Chr Start   End Value
## 1   1 40515 41942     0
## 2   1 42431 43302     0
## 3   1 43643 47503     0
## 4   1 47051 55369     0
## 5   1 60373 62454     0
## 6   1 67940 72313     0
```

UCEs in the genome represent only 13.67% of the coding genes! BUT how do they compare to the rest of the genome, we need to join these two dataframes

```
pterMadi2.genes2 <- read_tsv("./just_pterostichus_probes_intergenic.tsv", col_names = T) %>%
  distinct(uce, .keep_all = T) %>%
  rename(Chr = scaffold) %>%
  rename(Start = qstart) %>%
  rename(End = qend) %>%
  mutate(Value = case_when(biotype == "protein_coding" ~ "100",
                           biotype != "protein_coding" ~ "66",
                           type == "intergenic" ~ "33")) %>%
  select(Chr, Start, End, Value) %>%
  as.data.frame()
```

```
## Rows: 4913 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (8): scaffold, query, uce, probe, type, seqname, biotype, gene_id
## dbl (4): qstart, qend, seqstart, seqend
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pterMadi2.genes2$Value <- as.numeric(pterMadi2.genes2$Value)

pterMadi2.UCE_nonUCEs <- full_join(pterMadi2.genes.unmapped, pterMadi2.genes2)
```

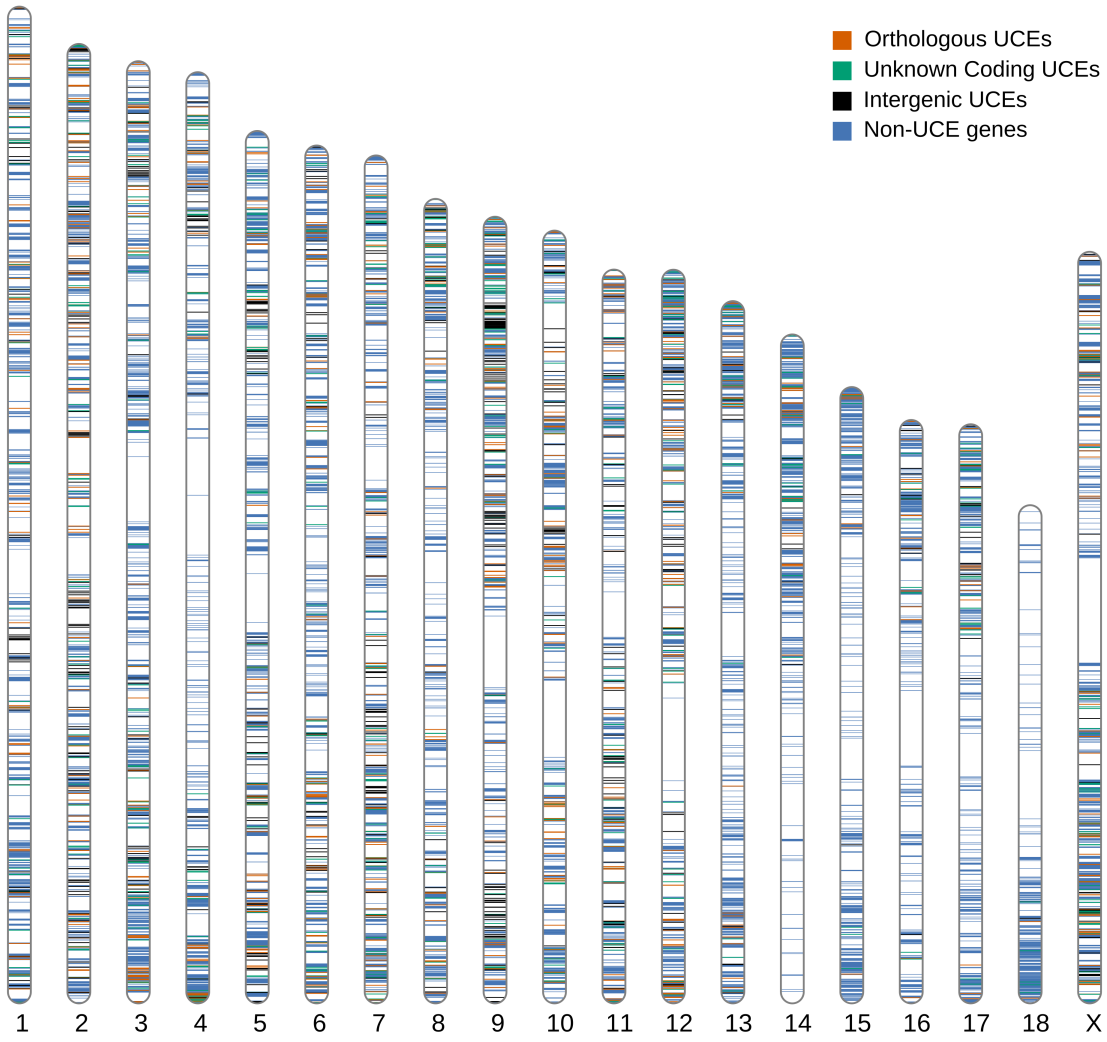
```
## Joining with `by = join_by(Chr, Start, End, Value)`
```

```
# turning these off for HTML output
#ideogram(karyotype = df.karyotype, overlaid = pterMadi2.UCE_nonUCEs, output = "temp.svg")
#convertSVG("temp.svg", device = "png")
```

Again, here is the logic for mapping on the chromosomes

- known coding regions == 100 (red)
- unknown coding regions == 66 (orange)
- intergenic == 33 (bluegreen)
- and non-uce genes == 0 (blue)

using this info I have modified this in an svg editor (inkscape, I wish I had illustrator!) The only change I have made includes changing size of the UCE vectors so they are easier to see. This is purely descriptive.




---

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.2 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
```

```
##  [1] LC_CTYPE=en_US.UTF-8    LC_NUMERIC=C            LC_TIME=en_GB.UTF-8
##  [4] LC_COLLATE=en_US.UTF-8  LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8    LC_NAME=C               LC_ADDRESS=C
## [10] LC_TELEPHONE=C          LC_MEASUREMENT=C        LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] RIdeogram_0.2.2 forcats_1.0.0   stringr_1.5.0   dplyr_1.1.0
##  [5] purrr_1.0.1     tidyr_1.3.0     tibble_3.1.8    tidyverse_1.3.2
##  [9] ggplot2_3.4.1   readr_2.1.4
##
## loaded via a namespace (and not attached):
##  [1] lubridate_1.8.0     png_0.1-8           rsvg_2.4.0
##  [4] assertthat_0.2.1    digest_0.6.31       utf8_1.2.3
##  [7] R6_2.5.1            cellranger_1.1.0    backports_1.4.1
## [10] reprex_2.0.2        evaluate_0.20       httr_1.4.2
## [13] pillar_1.8.1        rlang_1.0.6         googlesheets4_1.0.1
## [16] readxl_1.4.2        rstudioapi_0.14     rmarkdown_2.20
## [19] googledrive_2.0.0   bit_4.0.5           munsell_0.5.0
## [22] broom_1.0.3         compiler_4.1.2      modelr_0.1.10
## [25] xfun_0.37           pkgconfig_2.0.3     base64enc_0.1-3
## [28] htmltools_0.5.4     tidyselect_1.2.0    XML_3.99-0.9
## [31] fansi_1.0.4         crayon_1.5.2        tzdb_0.3.0
## [34] dbplyr_2.3.0        withr_2.5.0         grid_4.1.2
## [37] jsonlite_1.8.4      gtable_0.3.1        lifecycle_1.0.3
## [40] DBI_1.1.3           magrittr_2.0.3      scales_1.2.1
## [43] cli_3.6.0           stringi_1.7.12      vroom_1.6.1
## [46] grImport2_0.2-0     fs_1.6.1            xml2_1.3.3
## [49] ellipsis_0.3.2      generics_0.1.3      vctrs_0.5.2
## [52] tools_4.1.2         bit64_4.0.5         glue_1.6.2
## [55] hms_1.1.2           jpeg_0.1-10         parallel_4.1.2
## [58] fastmap_1.1.0       yaml_2.3.7          colorspace_2.1-0
## [61] gargle_1.3.0        rvest_1.0.3         knitr_1.42
## [64] haven_2.5.1
```