Working with genomic range data in R

Biodiversity Informatics (BIOL 475/575)

February 22, 2022

Programmer: AI

In this program, xxx

Header

```r
# Load Libraries
library(ezknitr)
library(ggplot2)

# Clear Environment & Set Seed
remove(list=ls())
set.seed(71587)
```

1. Load libraries from new sources

Use biocManager to get package GenomicRanges

```r
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
```

```
## Bioconductor version 3.14 (BiocManager 1.30.16), R 4.1.2 (2021-11-01)
```

```r
BiocManager::install(version = "3.14")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## '?repositories' for details
##
## replacement repositories:
##     CRAN: https://cran.rstudio.com/
```

```
## Bioconductor version 3.14 (BiocManager 1.30.16), R 4.1.2 (2021-11-01)
```

Now install the package

```r
BiocManager::install("GenomicRanges")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## '?repositories' for details
##
## replacement repositories:
##     CRAN: https://cran.rstudio.com/
```

```
## Bioconductor version 3.14 (BiocManager 1.30.16), R 4.1.2 (2021-11-01)
```

```
## Warning: package(s) not installed when version(s) same as current; use `force =
##   TRUE` to re-install: 'GenomicRanges'
```

```r
library(IRanges)
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter,
##     Find, get, grep, grepl, intersect, is.unsorted, lapply, Map,
##     mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##     pmin.int, Position, rank, rbind, Reduce, rownames, sapply,
##     setdiff, sort, table, tapply, union, unique, unsplit,
##     which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

Source the script from the textbook to make figures

```r
source("plot-ranges.R")
```

2. Introduction to range data

IRanges are an object that R recognizes as being a genomic range.

Remember that ranges would normally have metadata such as

1. Chromosome name, such as “chr17” or “chrY” or “contig184”
2. Range that demonstrates the specific sequence on the chromosome
3. Strand, which is either forward (positive) or backward (negative)

```r
# Ranges can be made by designating start and end
(rng <- IRanges(start = 4, end = 13))
```

```
## IRanges object with 1 range and 0 metadata columns:
##           start       end     width
##       <integer> <integer> <integer>
##   [1]         4        13        10
```

```r
# Ranges can be made by designating start or end AND width
(rng <- IRanges(start = 4, width = 3))
```

```
## IRanges object with 1 range and 0 metadata columns:
##           start       end     width
##       <integer> <integer> <integer>
##   [1]         4         6         3
```

IRanges objects can also be created to contain many ranges

```r
(x <- IRanges(start = c(4, 7, 2, 20),
              end = c(13, 7, 5, 23)))
```

```
## IRanges object with 4 ranges and 0 metadata columns:
##           start       end     width
##       <integer> <integer> <integer>
##   [1]         4        13        10
##   [2]         7         7         1
##   [3]         2         5         4
##   [4]        20        23         4
```

And each range within the IRanges object can be named:

```r
names(x) <- letters[1:length(x)]
x
```

```
## IRanges object with 4 ranges and 0 metadata columns:
##         start       end     width
##     <integer> <integer> <integer>
##   a         4        13        10
##   b         7         7         1
##   c         2         5         4
##   d        20        23         4
```

Let's plot the ranges

```r
plotIRanges(x)
```

What values start each range?

What values end each range?

What is the width of each range?

What is the total range of the IRanges object?

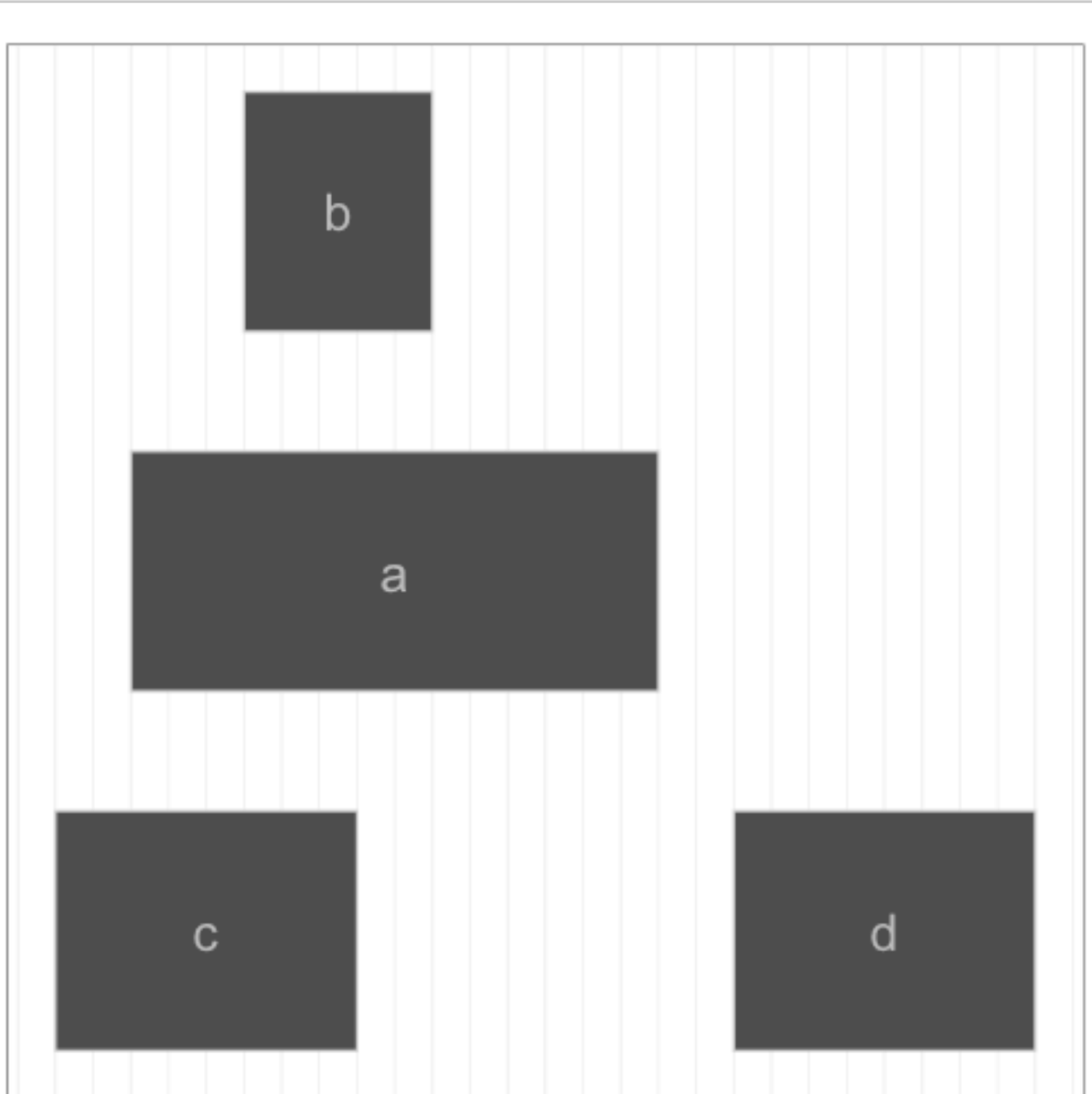
What is the difference between range(x) and width(x)??

We can manipulate the ranges with standard arithmetic:

```r
end(x) <- end(x) + 4
x
```

```
## IRanges object with 4 ranges and 0 metadata columns:
##         start       end     width
##     <integer> <integer> <integer>
##   a         4        17        14
##   b         7        11         5
##   c         2         9         8
##   d        20        27         8
```

```r
plotIRanges(x)
```



We can also use many of the other R functions to manipulate IRanges:

Use subsetting to look at just second and third row of x

Use subsetting to look at just ranges named “a” and “c”

Display logical answer for when start of x is less than 5

Display ranges that are greater than 8 in width

We can also merge ranges together with c()

```r
a <- IRanges(start = 7, width = 4)
b <- IRanges(start = 2, end = 5)
c(a, b)
```

```
## IRanges object with 2 ranges and 0 metadata columns:
##           start       end     width
##       <integer> <integer> <integer>
##   [1]         7        10         4
##   [2]         2         5         4
```

3. More advanced range operations

```r
x <- IRanges(start = c(40, 80),
             end = c(67, 114))
names(x) <- c("a", "b")
```

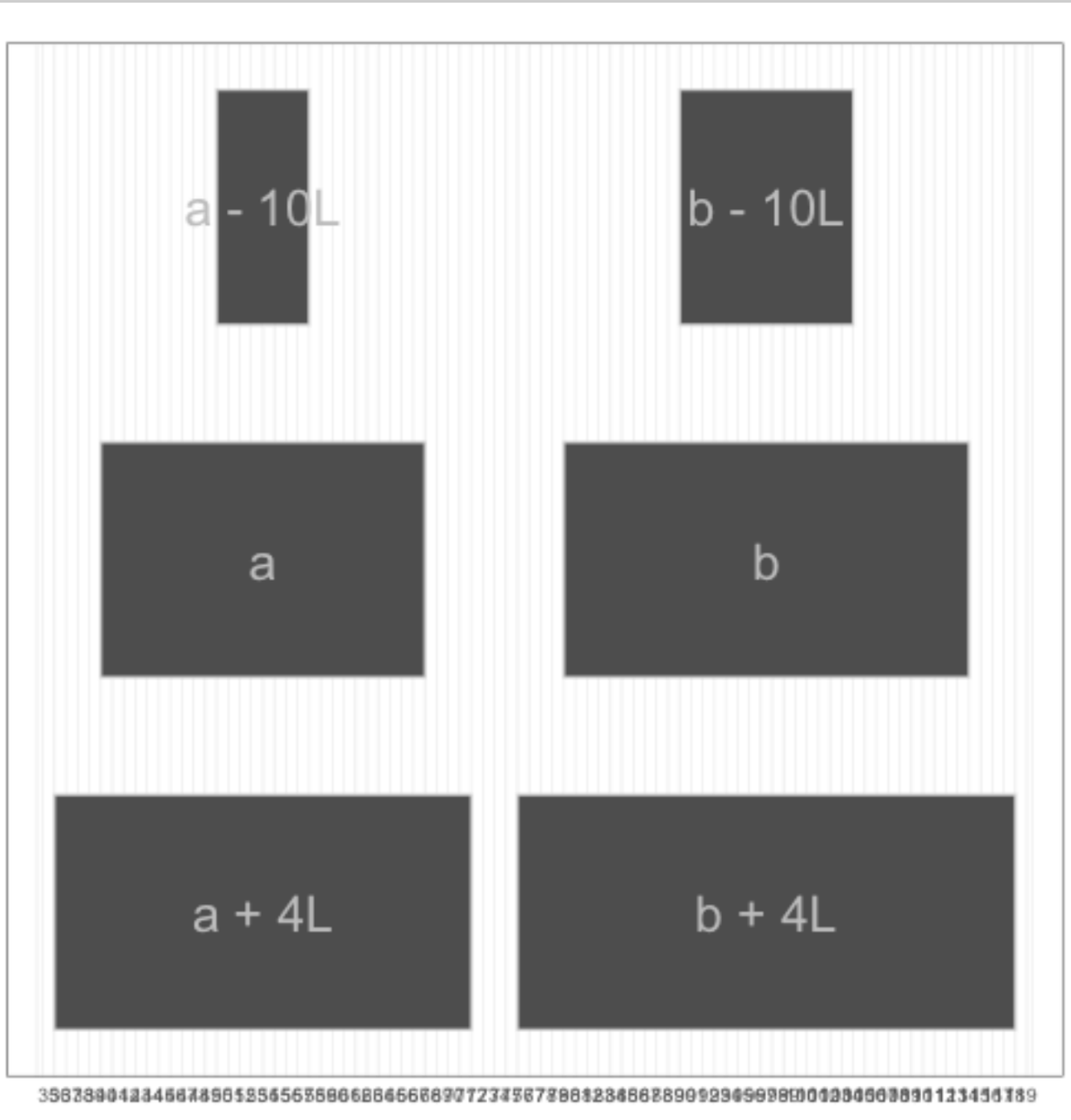
By adding 4L, this grows the sequence symmetrically by 4 on each side

```r
y <- x + 4L
names(y) <- c("a + 4L", "b + 4L")
```

By subtracting, we symmetrically cut off each end of the sequence

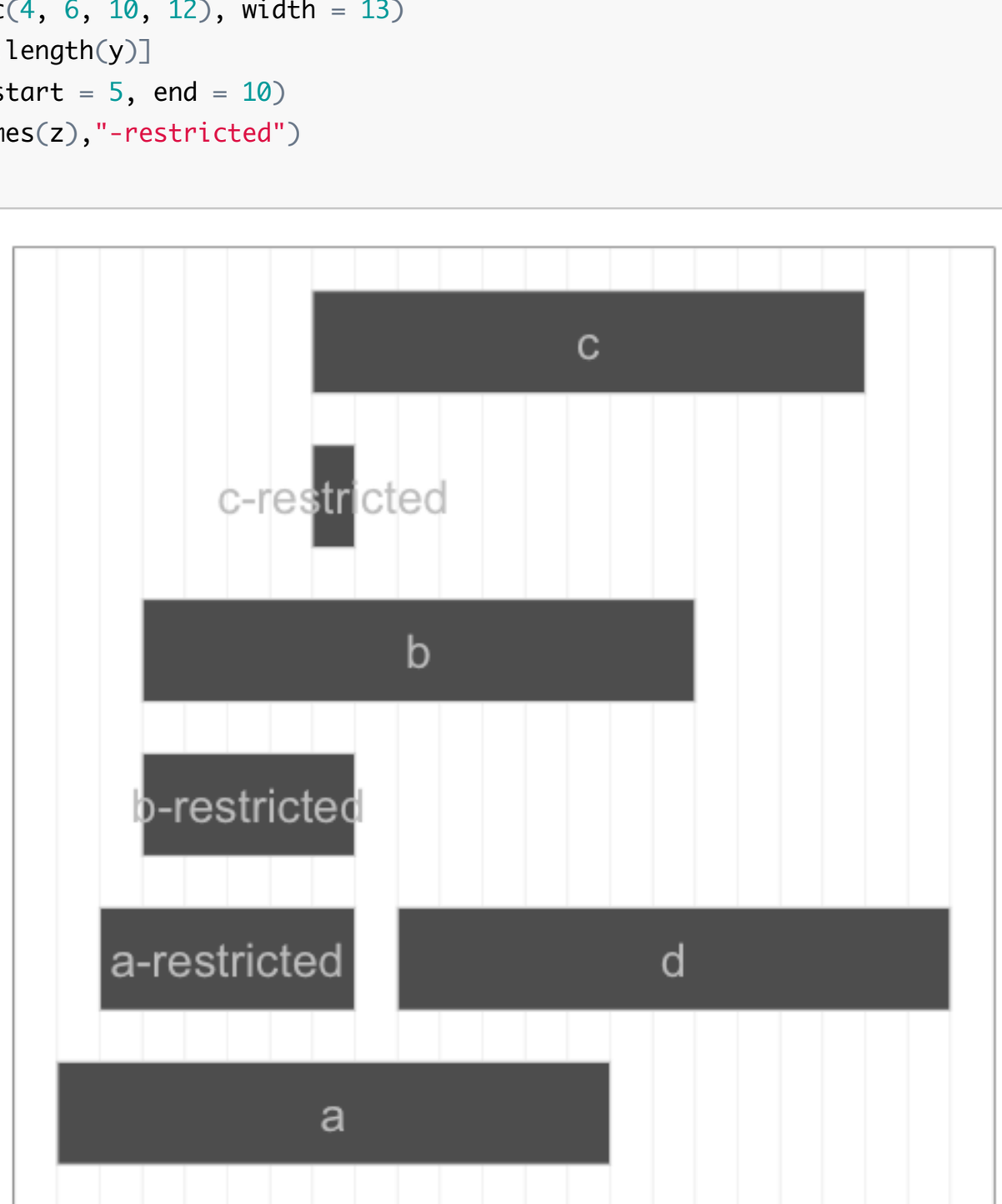
```r
z <- x - 10L
names(z) <- c("a - 10L", "b - 10L")

plotIRanges(c(x,y,z))
```



By restricting the ranges, we cut them off to fit in a specific range

```r
y <- IRanges(start = c(4, 6, 10, 12), width = 13)
names(y) <- letters[1:length(y)]
z <- restrict(x = y, start = 5, end = 10)
names(z) <- paste0(names(z),"-restricted")
plotIRanges(c(y,z))
```



We can also flank the ranges to create downstream or upstream sequences that contain promoter sequences

```r
x <- IRanges(start = c(40, 80),
             width = c(28, 35))
names(x) <- letters[1:length(x)]

y <- flank(x, width = 7, start = TRUE)
names(y) <- paste0(names(y), "-upstrm")

z <- flank(x, width = 7, start = FALSE)
names(z) <- paste0(names(z), "-dwnstrm")

c(x,y,z)
```

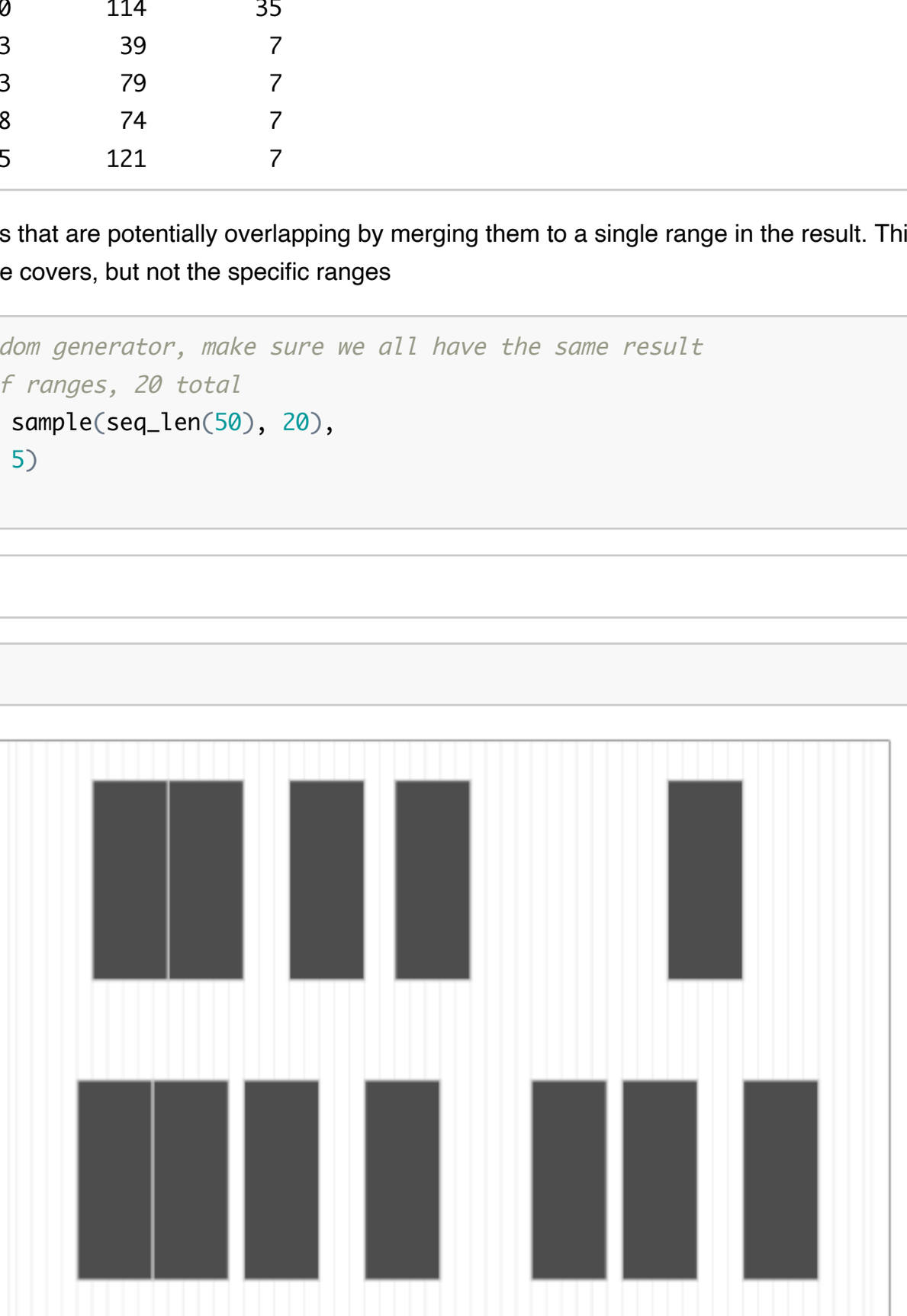
```
## IRanges object with 6 ranges and 0 metadata columns:
##                 start       end     width
##             <integer> <integer> <integer>
##           a        40        67        28
##           b        80       114        35
##    a-upstrm        33        39         7
##    b-upstrm        73        79         7
##   a-dwnstrm        68        74         7
##   b-dwnstrm       115       121         7
```

We can also reduce the ranges that are potentially overlapping by merging them to a single range in the result. This is useful if we care about what regions a sequence covers, but not the specific ranges

```r
set.seed(0) # reset random generator, make sure we all have the same result
# Create a longer set of ranges, 20 total
alns <- IRanges(start = sample(seq_len(50), 20),
                width = 5)
length(alns)
```

```
## [1] 20
```

```r
plotIRanges(alns)
```

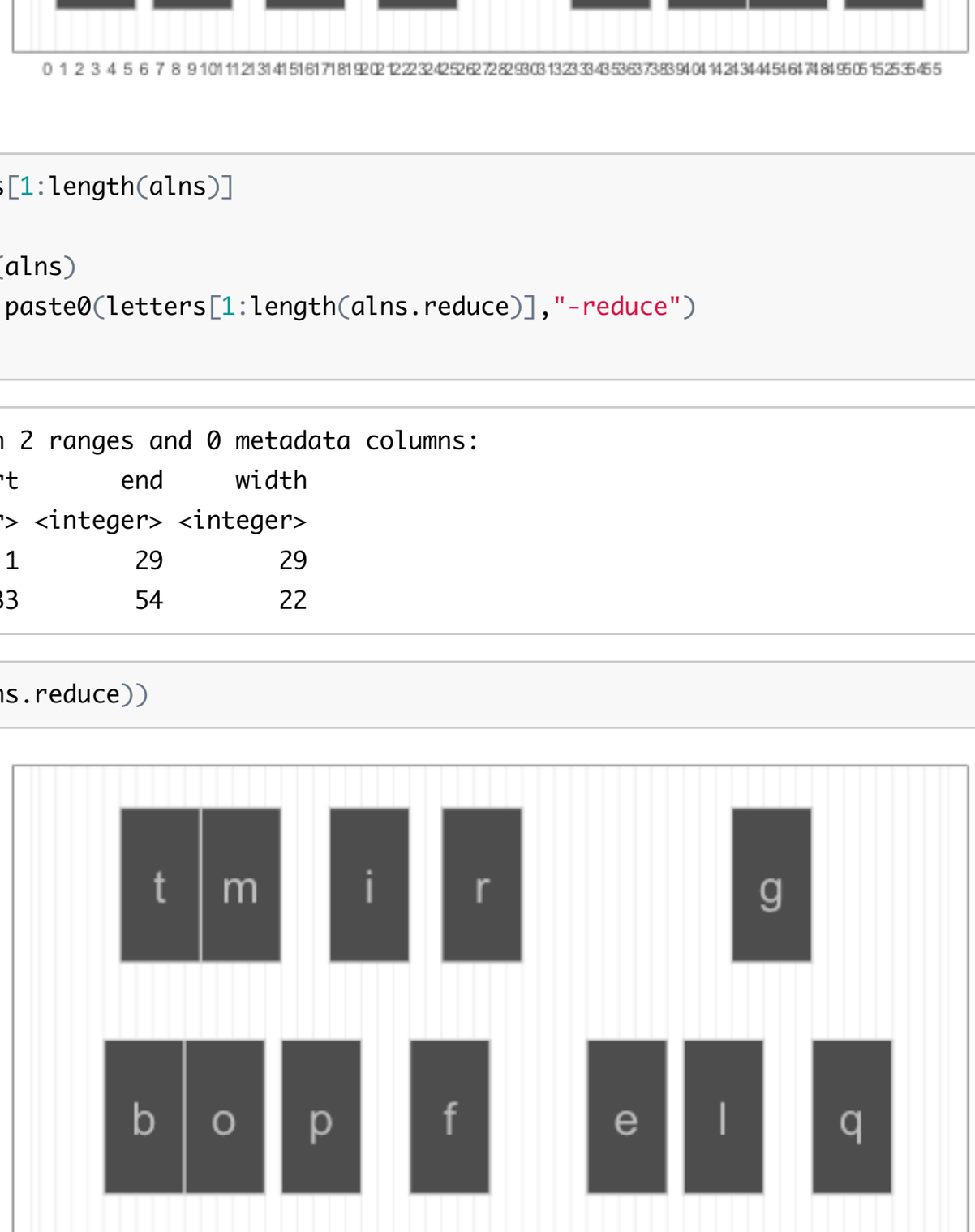


```r
names(alns) <- letters[1:length(alns)]

alns.reduce <- reduce(alns)
names(alns.reduce) <- paste0(letters[1:length(alns.reduce)],"-reduce")
alns.reduce
```

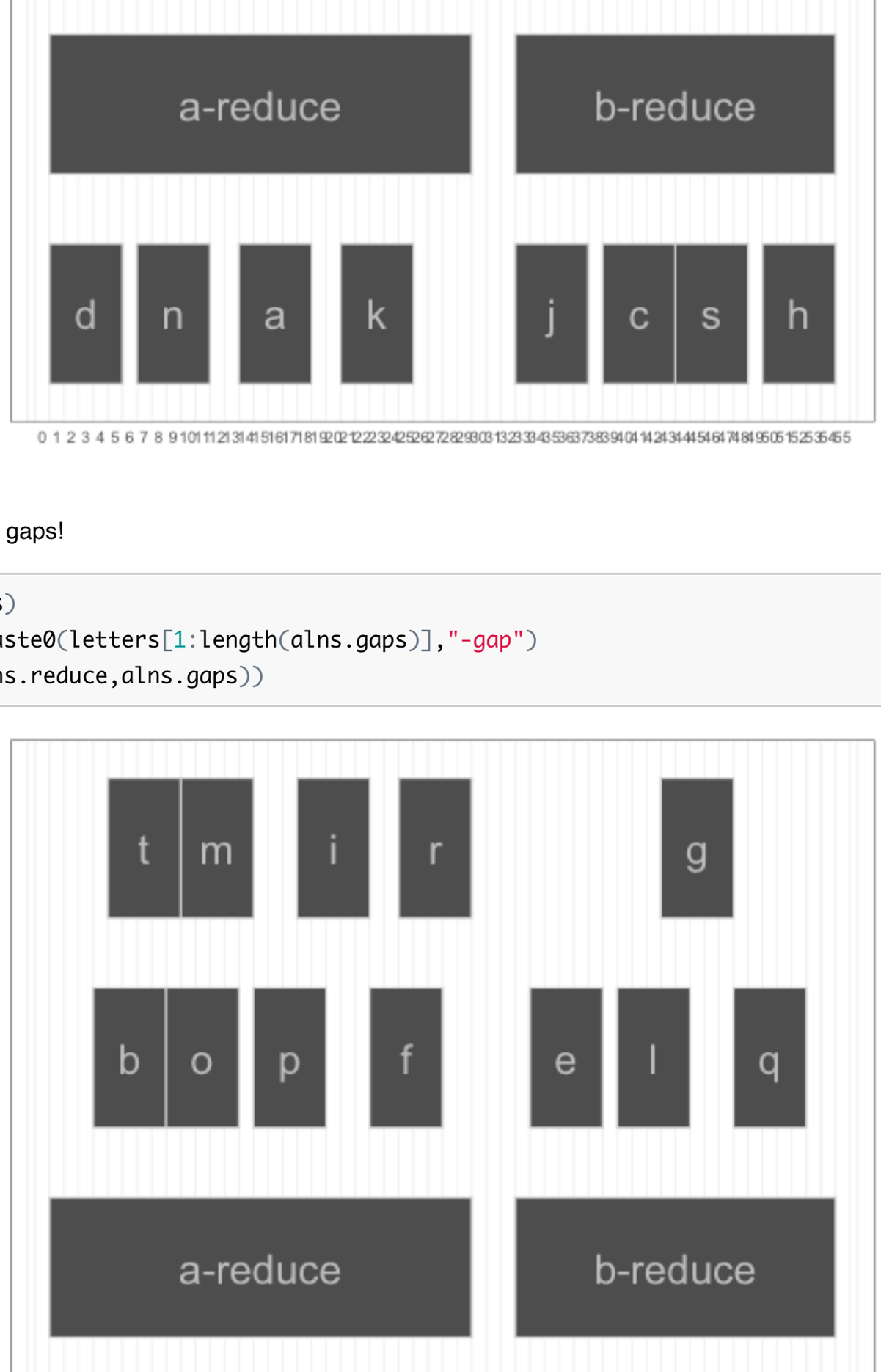
```
## IRanges object with 2 ranges and 0 metadata columns:
##                start       end     width
##            <integer> <integer> <integer>
##   a-reduce         1        29        29
##   b-reduce        33        54        22
```

```r
plotIRanges(c(alns,alns.reduce))
```



Similarly, we can identify the gaps!

```r
alns.gaps <- gaps(alns)
names(alns.gaps) <- paste0(letters[1:length(alns.gaps)],"-gap")
plotIRanges(c(alns,alns.reduce,alns.gaps))
```



Footer

spin this with: ezspin(file = “aaarcher/programs/20220223_range_data.R”,out_dir = “aaarcher/output”, fig_dir = “figures20220223”,keep_md = FALSE, keep_rmd = FALSE)