# One-to-N sequence matches in coronavirus genomes

Jacques van Helden

2020-05-19

## Contents

Parameters . . . . . 5
Genome statistics . . . . . 6
One-to-N alignemnts of selected features . . . . . 7
Output files . . . . . 8
Memory image . . . . . 19
Session info . . . . . 19

```
#### General parameters for the analysis ####

reloadImage <- TRUE

#### Sequence collection ####

## Supported:
collections <- c(
  "around-CoV-2",
  "selected",
  "around-CoV-2-plus-GISAID",
  "selected-plus-GISAID"
  )



# collection <- "around-CoV-2" # 14 genomes
# collection <- "around-CoV-2-plus-GISAID" # 16 genomes
# collection <- "selected" # ~60 genomes
collection <- "selected-plus-GISAID" # ~60 genomes

## Note about GIDAID sequences.
##
## A few genomes were not available in NCBI Genbank at the time of
## this analysis, and had to be downloaded from GISAID. These sequences
##  can however not be redistributed, they should thus be downloaded
##  manually to reproduce the full trees.


## Exclude incomplete genomes (i.e. those containing a lof of Ns) to avoid biases in the distance compu
excludeIncomplete <- TRUE
```

```r
#### Define directories and files ####
dir <- list(main = '..')
dir$R <- file.path(dir$main, "scripts/R")

#### Create output directory for sequences ####
dir$seqdata <- file.path(dir$main, "data")
dir.create(dir$seqdata, showWarnings = FALSE, recursive = TRUE)

## Instantiate a list for output files
outfiles <- vector()

## Memory image
dir$images <- file.path(dir$main, "memory_images")
dir.create(dir$images, recursive = TRUE, showWarnings = FALSE)
outfiles["Memory image"] <- file.path(dir$images, "one-to-n_matches.Rdata")

## Input files
infiles <- list()

## Output tables
# di$output <- file.path(dir.main, "")
# dir$tables <-

## Load custom functions
source(file.path(dir$R, "align_n_to_one.R"))
source(file.path(dir$R, "plot_pip_profiles.R"))

## A unequivocal pattern to identify the reference genome in the sequence names of the input file
refPattern <- "HuCoV2_WH01_2019"

## Exclude some genomes with a lot of Ns, because they bias the PIP profiles and alignments and trees
excludePatterns <- c("PnMP789", "PnGu-P2S_2019")

#### Features of interest in the reference genome ####

features <- list()


## Specific features


## S1: pre-cleavage part of the spike protein
features[["S1"]] <- c(start = 21599, end =  23617)
## S2: post-cleavage part of the spike protein
features[["S2"]] <- c(start = 23618, end = 25381)
## Receptor Binding Domain (RBD)
features[["RBD"]] <- c(start = 22517, end = 23185)
## Potential Pangolin origin after Xiao (https://doi.org/10.1101/2020.02.17.951335)
features[['Recomb-Xiao']] <- c(start = 22871, end = 23092)
## Recombinant region 1 seen on the PIP profiles
features[['Recomb-reg-1']] <- c(start = 21500, end = 22800)
## Recombinant region 2 seen on the PIP profiles
features[['Recomb-reg-2']] <- c(start = 22800, end = 24000)
```

```r
## Recombinant region 3 seen on the PIP profiles
features[['Recomb-reg-3']] <- c(start = 27800, end = 28350)

## Annotated coding sequences
features[['CDS-S']] <- c(start = 21563, end = 25384) ## Spike gene
features[['CDS-ORF3a']] <- c(start = 25393, end = 26220)
features[['CDS-E']] <- c(start = 26245, end = 26472)
features[['CDS-M']] <- c(start = 26523, end = 27191)
features[['CDS-ORF6']] <- c(start = 27202, end = 27387)
features[['CDS-ORF7a']] <- c(start = 27394, end = 27759)
features[['CDS-ORF8']] <- c(start = 27894, end = 28259)
features[['CDS-N']] <- c(start = 28274, end = 29533)
features[['CDS-ORF10']] <- c(start = 29558, end = 29674)
features[['CDS-ORF1ab']] <- c(start = 266, end = 21555)

## All the sequences after the bif ORF coding for the polyproteinn 1ab
features[['After-ORF1ab']] <- c(start = 21556, end = 29899)



## Report the parameters
message("\tReference genomes: ", refPattern)
```

```r
## Genome dir and files
if (length(grep(pattern = "GISAID", x = collection)) > 0) {
  useGISAID <- TRUE
  dir$genomes <- file.path(dir$main, "data", "GISAID_genomes")
  # collections <-  paste0(collections, "-plus-GISAID")
  # collection <-  paste0(collection, "-plus-GISAID")
} else {
  dir$genomes <- file.path(dir$main, "data", "genomes")
}


## Define the input genome
infiles$genomes <- file.path(
  dir$genomes,
  paste0("genomes_", collection, ".fasta"))

## Genome sequences
if (!file.exists(infiles$genomes)) {
  stop("Genome sequence file is missing", "\n", infiles$genomes)
}

#### Load genome sequences ####
genomes <- readDNAStringSet(filepath = infiles$genome, format = "fasta")

## Shorten sequence names by suppressing the fasta comment (after the space)
names(genomes) <- sub(pattern = " .*", replacement = "", x = names(genomes), perl = TRUE)

## Exclude genomes
if (excludeIncomplete) {
  excludePattern = paste0("(", paste(collapse = ")|(", excludePatterns), ")")
  excludedGenomeNames <- grep(pattern = excludePattern, x = names(genomes),
                              value = TRUE, invert = FALSE)
  filteredGenomeIndices <- grep(pattern = excludePattern, x = names(genomes),
                                value = FALSE, invert = TRUE)
  message("\tExcluded ", length(excludedGenomeNames)," genomes: ", paste(collapse = ", ", excludedGenome
  message("\tRemaining genomes: ", length(filteredGenomeIndices))
  genomes <- genomes[filteredGenomeIndices]
# names(genomes)
}

## Report the number of genoomes
genomeNames <- names(genomes)
nbGenomes <- length(genomeNames)
message("\tLoaded ", nbGenomes, " genomes from file ", infiles$genomes)
# View(genomes)


#### Define reference and query genomes ####
refGenomeName <- grep(pattern = refPattern, x = names(genomes),
                      ignore.case = TRUE, value = TRUE)
if (is.null(refGenomeName)) {
  stop("Could not identify reference genome with pattern ", refPattern)
}
message("\tReference genome name: ", refGenomeName)

## Compute some statistics about genome sizes
genomeStat <- data.frame(
  n = 1:length(genomes),
  row.names = names(genomes),
  status = rep("Query", length.out = length(genomeNames))
)
genomeStat[,"status"] <- as.vector(genomeStat[,"status"])
genomeStat[refGenomeName,"status"] <- "Reference"
g <- 1
for (g in genomeNames) {
  genomeStat[g, "length"] <- length(genomes[[g]])
}
```

```
#### Define the color associated to each sequence ####

## Color palette per species
speciesPalette <- list(
  Human = "#880000",
  Bat = "#888888",
  Pangolin = "#448800",
  Camel = "#BB8800",
  Pig = "#FFBBBB",
  Civet = "#00BBFF"
)

## Species prefix in the tip labels
speciesPrefix <- c("Hu" = "Human",
                   "Bt" = "Bat",
                   "Pn" = "Pangolin",
                   "Cm" = "Camel",
```

```r
                   "Pi" = "Pig",
                   "Cv" = "Civet")

## Strain-specific colors
strainColor <- c(
  "HuCoV2_WH01_2019" = "red",
  "HuSARS-Frankfurt-1_2003" = "#0044BB",
  "PnGu1_2019" = "#00BB00",
  "BtRaTG13_" = "#FF6600",
  "BtYu-RmYN" = "#FFBB22",
  "BtZXC21" = "black",
  "BtZC45" = "black")

## Identify species per tip
for (prefix in names(speciesPrefix)) {
  genomeStat[grep(pattern = paste0("^", prefix), x = row.names(genomeStat), perl = TRUE), "species"] <-

}

## Assign acolor to each species
genomeStat$color <- "grey" # default
genomeStat$color <- speciesPalette[as.vector(genomeStat$species)]

for (strain in names(strainColor)) {
  genomeStat[grep(pattern = paste0("^", strain),
                  x = row.names(genomeStat), perl = TRUE), "color"] <- strainColor[strain]

}


## Assign specific color to some nodes

## Define a color for each genome
genomeColors <- (unlist(genomeStat$color))
names(genomeColors) <- row.names(genomeStat)
```

## Parameters

```r
## Define a list of parameters
parameters <- list()
parameters$collection <- collection
parameters$refGenomeName <- refGenomeName
parameters$nbGenomes <- nbGenomes
parameters$genomeDir <- dir$genomes
parameters$genomeFile <- infiles$genomes
kable(t(as.data.frame.list(parameters)), caption = "Parameters of the analysis",
      col.names = "Parameter")
```

Table 1: Parameters of the analysis

| | Parameter |
|---|---|
| collection | selected-plus-GISAID |
| refGenomeName | HuCoV2_WH01_2019 |

|  | Parameter |
|---|---|
| nbGenomes | 38 |
| genomeDir | ../data/GISAID__genomes |
| genomeFile | ../data/GISAID__genomes/genomes__selected-plus-GISAID.fasta |

## Genome statistics

```r
kable(genomeStat, caption = "Reference and query genomes")
```

Table 2: Reference and query genomes

|  | n | status | length | species | color |
|---|---|---|---|---|---|
| BtBM48-31 | 1 | Query | 29276 | Bat | #888888 |
| BtBtKY72 | 2 | Query | 29274 | Bat | #888888 |
| BtCp-Yun__2011 | 3 | Query | 29452 | Bat | #888888 |
| BtGX2013 | 4 | Query | 29161 | Bat | #888888 |
| BtHKU3-12 | 5 | Query | 29704 | Bat | #888888 |
| BtHKU5 | 6 | Query | 30482 | Bat | #888888 |
| BtHKU9-1 | 7 | Query | 29114 | Bat | #888888 |
| BtJL2012 | 8 | Query | 29037 | Bat | #888888 |
| BtLYRa11 | 9 | Query | 29805 | Bat | #888888 |
| BtRaTG13__2013__Yunnan | 10 | Query | 29855 | Bat | #FF6600 |
| Btrec-SARSg__2008 | 11 | Query | 29750 | Bat | #888888 |
| BtRm1/2004 | 12 | Query | 29749 | Bat | #888888 |
| BtRp-Shaanxi2011 | 13 | Query | 29484 | Bat | #888888 |
| BtRp3-2004 | 14 | Query | 29736 | Bat | #888888 |
| BtRs__672-2006 | 15 | Query | 29059 | Bat | #888888 |
| BtRs4874 | 16 | Query | 30311 | Bat | #888888 |
| BtSC2018 | 17 | Query | 29648 | Bat | #888888 |
| BtYN2013 | 18 | Query | 29142 | Bat | #888888 |
| BtYN2018B | 19 | Query | 30256 | Bat | #888888 |
| BtYN2018C | 20 | Query | 29689 | Bat | #888888 |
| BtYNLF__31C | 21 | Query | 29723 | Bat | #888888 |
| BtZC45 | 22 | Query | 29802 | Bat | black |
| BtZXC21 | 23 | Query | 29732 | Bat | black |
| CmMERS | 24 | Query | 29851 | Camel | #BB8800 |
| Cv007-2004 | 25 | Query | 29540 | Civet | #00BBFF |
| Hu229E | 26 | Query | 27317 | Human | #880000 |
| HuCoV2__WH01__2019 | 27 | Reference | 29899 | Human | red |
| HuMERS__172-06__2015 | 28 | Query | 30068 | Human | #880000 |
| HuNL63 | 29 | Query | 27553 | Human | #880000 |
| HuOC43 | 30 | Query | 30741 | Human | #880000 |
| HuSARS-Frankfurt-1__2003 | 31 | Query | 29727 | Human | #0044BB |
| HuTGEV | 32 | Query | 28586 | Human | #880000 |
| PiPRCV | 33 | Query | 27765 | Pig | #FFBBBB |
| PiSADS | 34 | Query | 27163 | Pig | #FFBBBB |
| PnGX-P1E__2017 | 35 | Query | 29801 | Pangolin | #448800 |
| PnGX-P2V__2018 | 36 | Query | 29795 | Pangolin | #448800 |
| BtYu-RmYN02__2019 | 37 | Query | 29671 | Bat | #FFBB22 |
| PnGu1__2019 | 38 | Query | 29825 | Pangolin | #00BB00 |

The collection selected-plus-GISAID contains 38 virus genome sequences.

## One-to-N alignemnts of selected features

We perform a global pairwise alignment (Needle-Waterman algorithm) between each feature of the reference (HuCoV2_WH01_2019) and each one of the query genomes.

```
#### One-to-N alignmemnt of user-speficied genomic features ####
# featureName <- "CDS-S" # for the test
featureName <- "S1" # for the test
# featureName <- "Recomb-reg-3" # for the test
# featureName <- "RBD" # for the test
# for (collection in (collections)) {
if (reloadImage) {
  load(outfiles["Memory image"])
  reloadImage <- TRUE

} else {
  allFeatureAlignments <- list()
}
for (featureName in names(features)) {
  featureLimits <- features[[featureName]]
  featureStart <- featureLimits[["start"]]
  featureEnd <- featureLimits[["end"]]
  featureLength <-
    featureEnd - featureStart  + 1

  message("Searching matches for feature ", featureName,
          " (", featureLimits[1], "-", featureLimits[2], ")",
          " in collection ", collection)

  #### N-to-1 alignments of spike-coding sequences ####
  featurePrefix <- paste0(
    featureName,
    "_", collection)

  dir[[featureName]] <- file.path(dir$seqdata, featureName)
  dir.create(dir[[featureName]], showWarnings = FALSE, recursive = TRUE)
  outfiles[featureName]  <- file.path(
    dir[[featureName]], paste0(featurePrefix, ".fasta"))
  message("\tOutput directory: ", dir[[featureName]] )

  ## Get sequence of the feature from the reference genome
  refSeq <- subseq(genomes[refGenomeName],
                   start = featureLimits[1],
                   end = featureLimits[2])

  ## Match the reference feature with all the genomes
  if (reloadImage) {
    featureAlignmentsNto1 <- allFeatureAlignments[[featureName]]
  } else {
    featureAlignmentsNto1 <- alignNtoOne(
      refSequence = refSeq,
      querySequences = genomes,
      type = "global-local",
      outfile = outfiles[featureName])

    allFeatureAlignments[[featureName]] <- featureAlignmentsNto1
  }

  genomeOrder <- order(featureAlignmentsNto1$stat$score, decreasing = TRUE)

  if (featureLength > 20000) {
    windowSize <- 800
  } else if (featureLength > 5000) {
    windowSize <- 500
  } else if (featureLength > 3000) {
    windowSize <- 200
  } else {
    windowSize <- max(100, 10 * round(featureLength / 100))
  }

  ## PIP profile of one-to-N alignment
  plotPIPprofiles(
    alignments = featureAlignmentsNto1$alignments[genomeOrder],
    reversePlot = TRUE,
    windowSize = windowSize,
    main = paste0(featureName, " (", featureStart, ":", featureEnd ,", ", featureLength, " bp)"),
    #    colors = NULL,
    colors = genomeColors,
    legendMargin = 0.25,
    legendCorner = "topright", lwd = 1,
    legendCex = 0.5, ylim = c(30, 100))


  kable(featureAlignmentsNto1$stats[genomeOrder,],
        caption = paste0(
          "One-to-N alignment of feature ",
          featureName))

}
```

```
# }
```

## Output files

```
kable(t(as.data.frame(dir)), col.names = "Dir", caption = "Directories")
```

Table 3: Directories

| | Dir |
|---|---|
| main | .. |
| R | ../scripts/R |
| seqdata | ../data |
| images | ../memory_images |
| genomes | ../data/GISAID_genomes |
| S1 | ../data/S1 |
| S2 | ../data/S2 |

|  | Dir |
|---|---|
| RBD | ../data/RBD |
| Recomb.Xiao | ../data/Recomb-Xiao |
| Recomb.reg.1 | ../data/Recomb-reg-1 |
| Recomb.reg.2 | ../data/Recomb-reg-2 |
| Recomb.reg.3 | ../data/Recomb-reg-3 |
| CDS.S | ../data/CDS-S |
| CDS.ORF3a | ../data/CDS-ORF3a |
| CDS.E | ../data/CDS-E |
| CDS.M | ../data/CDS-M |
| CDS.ORF6 | ../data/CDS-ORF6 |
| CDS.ORF7a | ../data/CDS-ORF7a |
| CDS.ORF8 | ../data/CDS-ORF8 |
| CDS.N | ../data/CDS-N |
| CDS.ORF10 | ../data/CDS-ORF10 |
| CDS.ORF1ab | ../data/CDS-ORF1ab |
| After.ORF1ab | ../data/After-ORF1ab |

```r
outfileTable <- data.frame(path = as.vector(outfiles))
outfileTable$basename <- basename(as.vector(outfileTable$path))
outfileTable$dir <- dirname(as.vector(outfileTable$path))
outfileTable$link <- paste0(
  "[", outfileTable$basename, "](", outfileTable$path, ")"
)

kable(outfileTable[, c("dir", "link")],
      col.names = c("dir", "file"),
      caption = "Output files")
```

Table 4: Output files

| dir | file |
|---|---|
| ../memory_images | one-to-n_matches.Rdata |
| ../data/S1 | S1_selected-plus-GISAID.fasta |
| ../data/S2 | S2_selected-plus-GISAID.fasta |
| ../data/RBD | RBD_selected-plus-GISAID.fasta |
| ../data/Recomb-Xiao | Recomb-Xiao_selected-plus-GISAID.fasta |
| ../data/Recomb-reg-1 | Recomb-reg-1_selected-plus-GISAID.fasta |
| ../data/Recomb-reg-2 | Recomb-reg-2_selected-plus-GISAID.fasta |
| ../data/Recomb-reg-3 | Recomb-reg-3_selected-plus-GISAID.fasta |
| ../data/CDS-S | CDS-S_selected-plus-GISAID.fasta |
| ../data/CDS-ORF3a | CDS-ORF3a_selected-plus-GISAID.fasta |
| ../data/CDS-E | CDS-E_selected-plus-GISAID.fasta |
| ../data/CDS-M | CDS-M_selected-plus-GISAID.fasta |
| ../data/CDS-ORF6 | CDS-ORF6_selected-plus-GISAID.fasta |
| ../data/CDS-ORF7a | CDS-ORF7a_selected-plus-GISAID.fasta |
| ../data/CDS-ORF8 | CDS-ORF8_selected-plus-GISAID.fasta |
| ../data/CDS-N | CDS-N_selected-plus-GISAID.fasta |
| ../data/CDS-ORF10 | CDS-ORF10_selected-plus-GISAID.fasta |
| ../data/CDS-ORF1ab | CDS-ORF1ab_selected-plus-GISAID.fasta |
| ../data/After-ORF1ab | After-ORF1ab_selected-plus-GISAID.fasta |

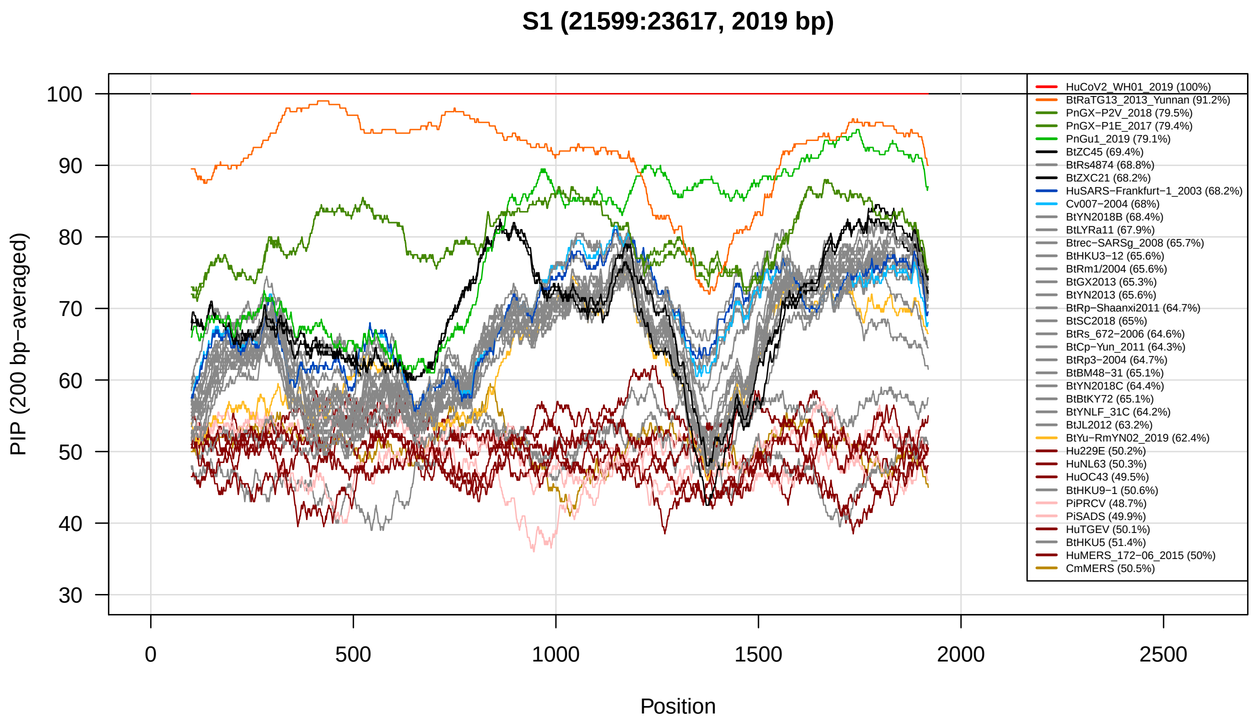

Figure 1: Feature-specific Percent Identical Positions (PIP) profiles.

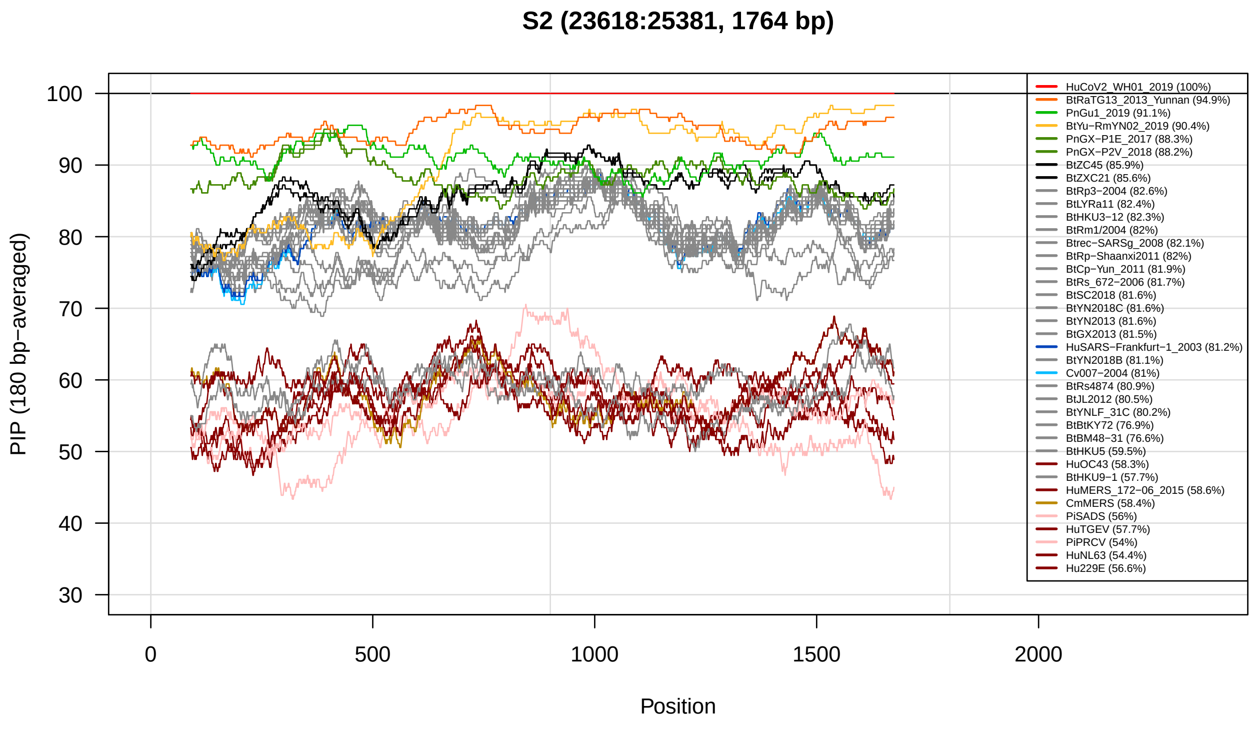

Figure 2: Feature-specific Percent Identical Positions (PIP) profiles.

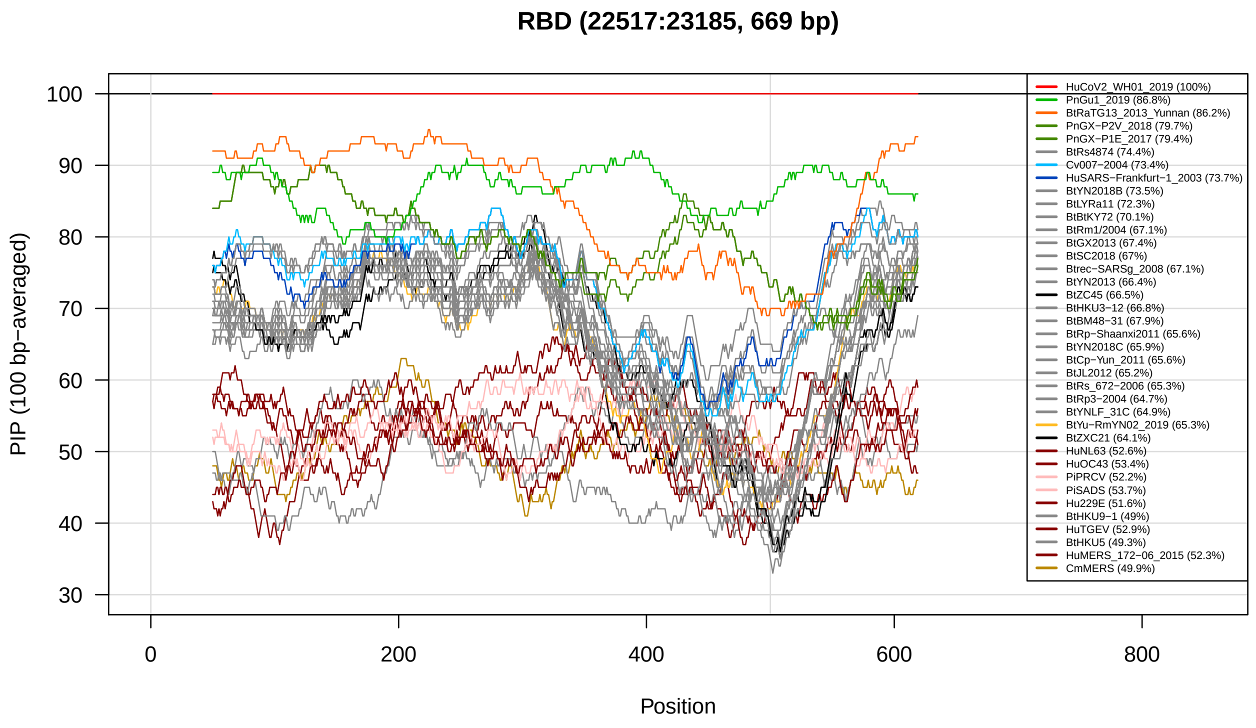

Figure 3: Feature-specific Percent Identical Positions (PIP) profiles.

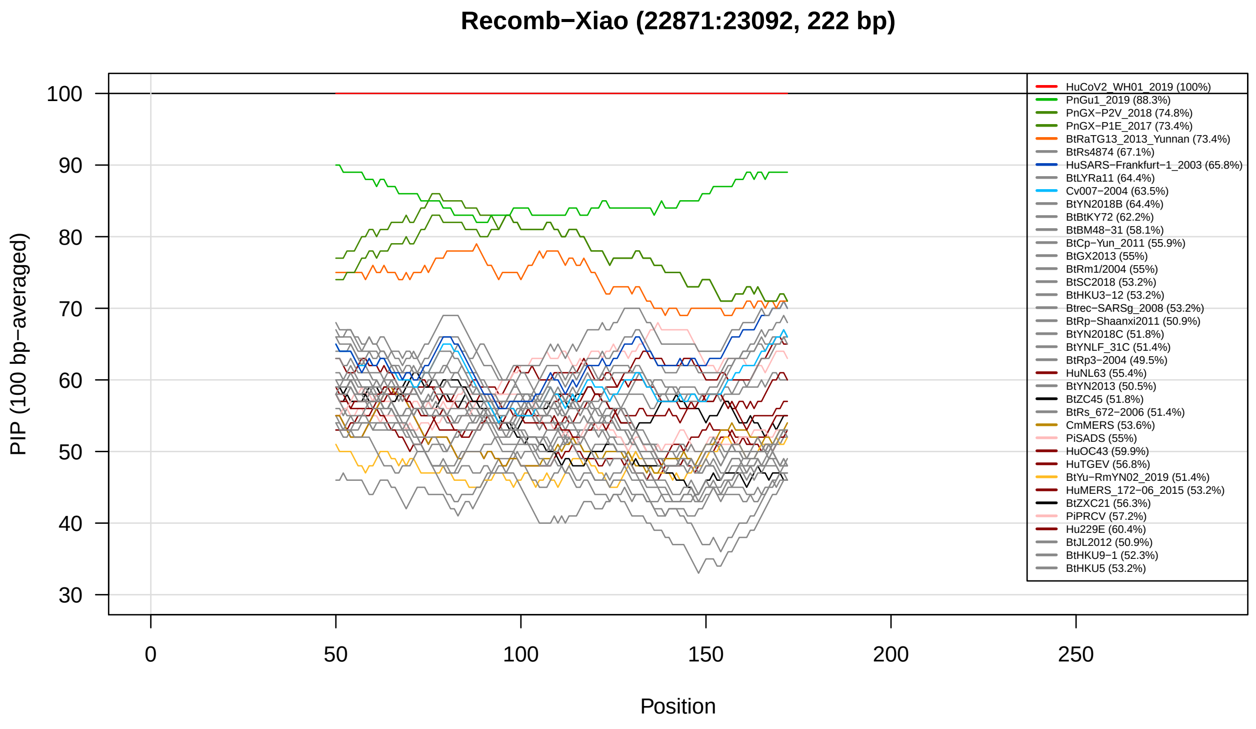

Figure 4: Feature-specific Percent Identical Positions (PIP) profiles.

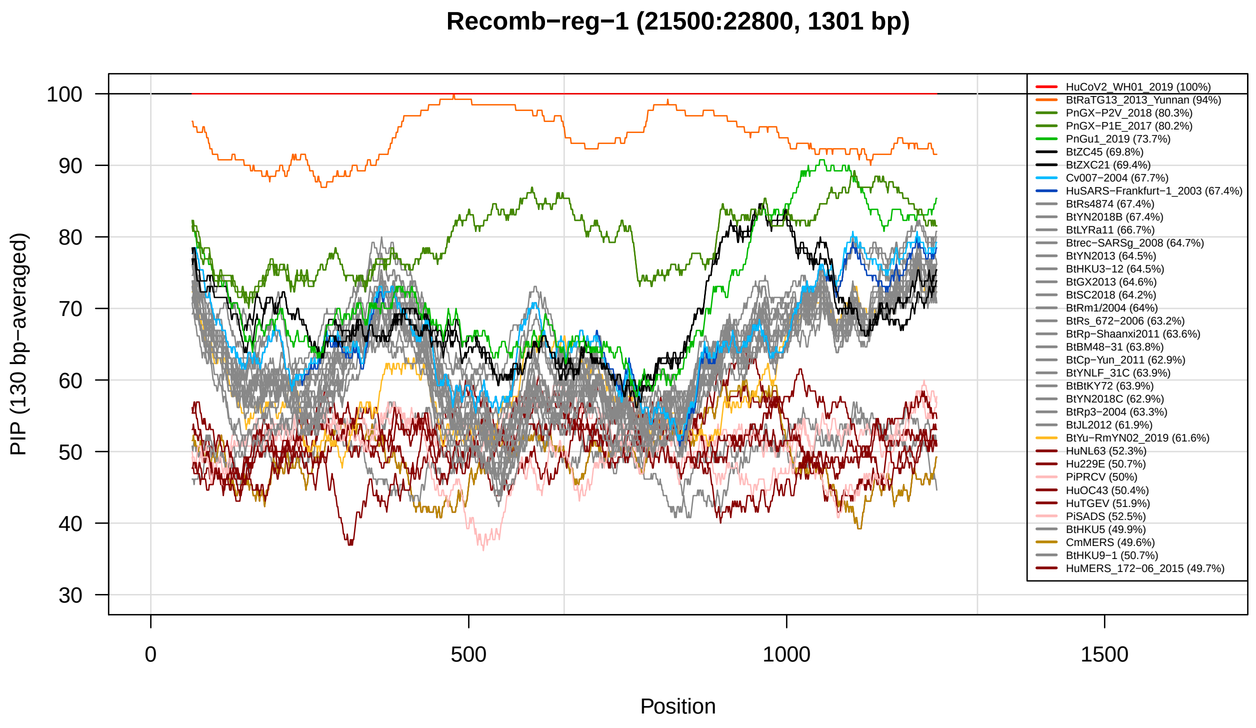

Figure 5: Feature-specific Percent Identical Positions (PIP) profiles.

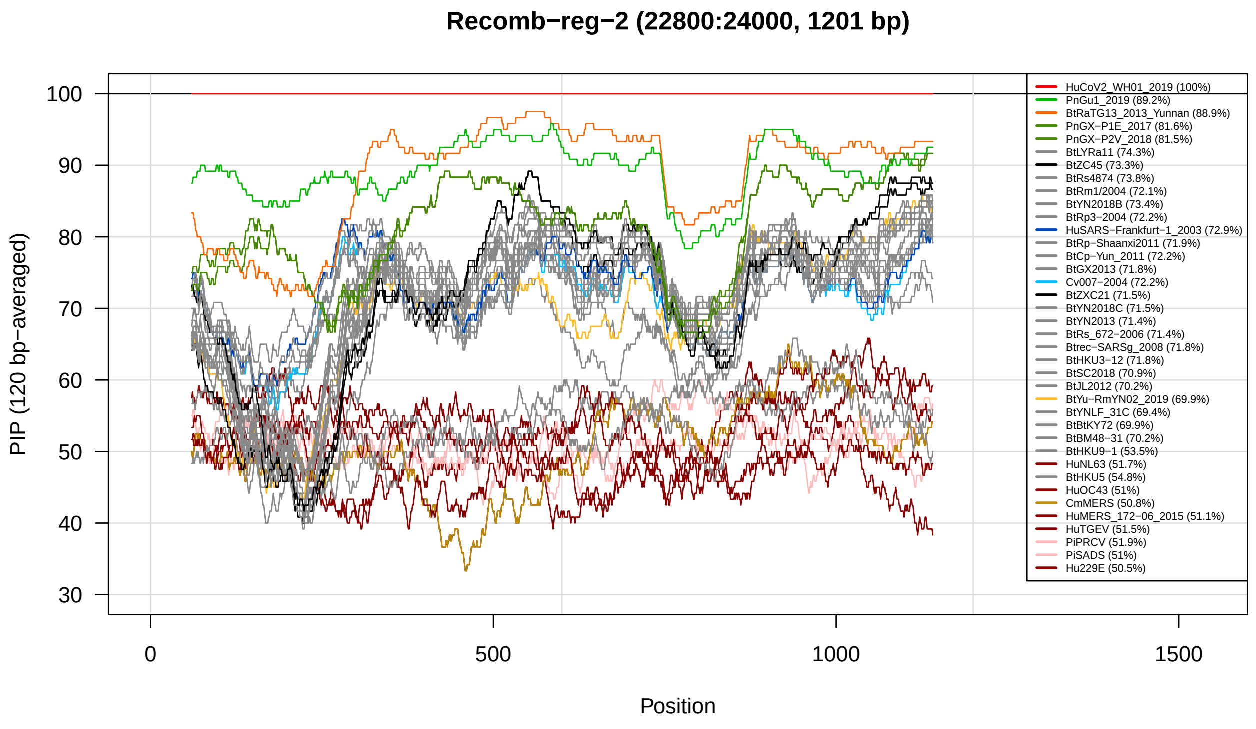

Figure 6: Feature-specific Percent Identical Positions (PIP) profiles.

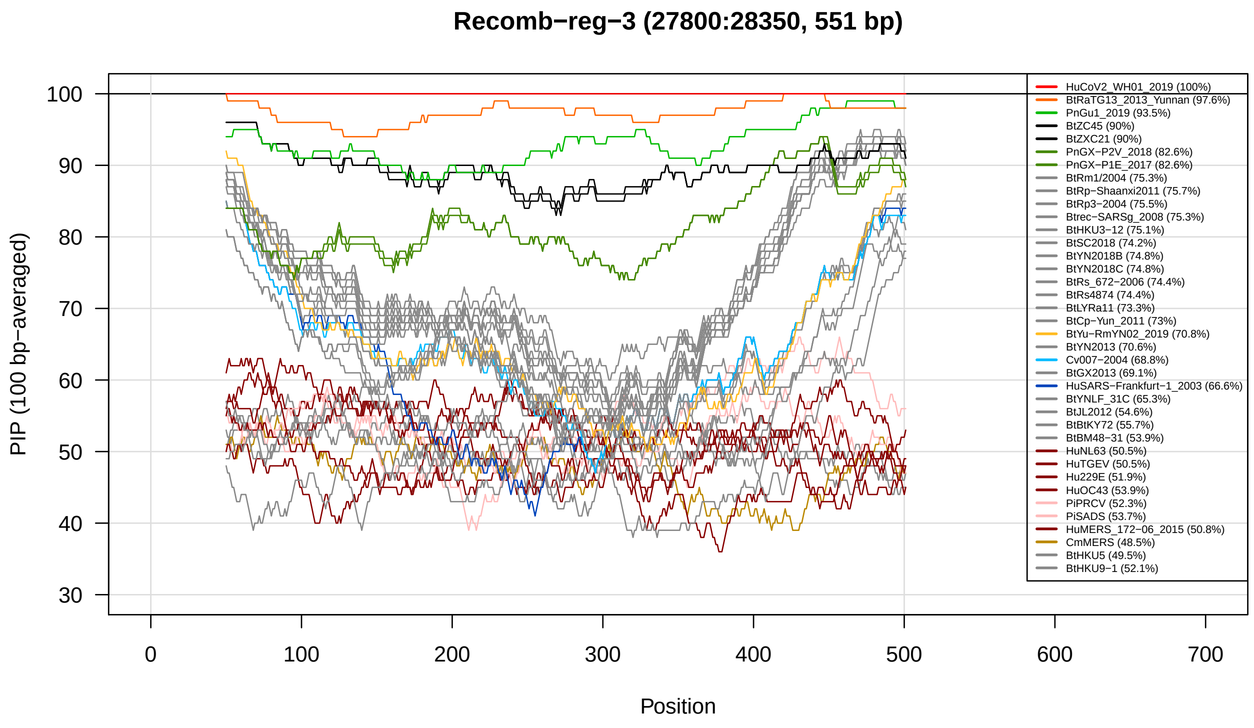

Figure 7: Feature-specific Percent Identical Positions (PIP) profiles.

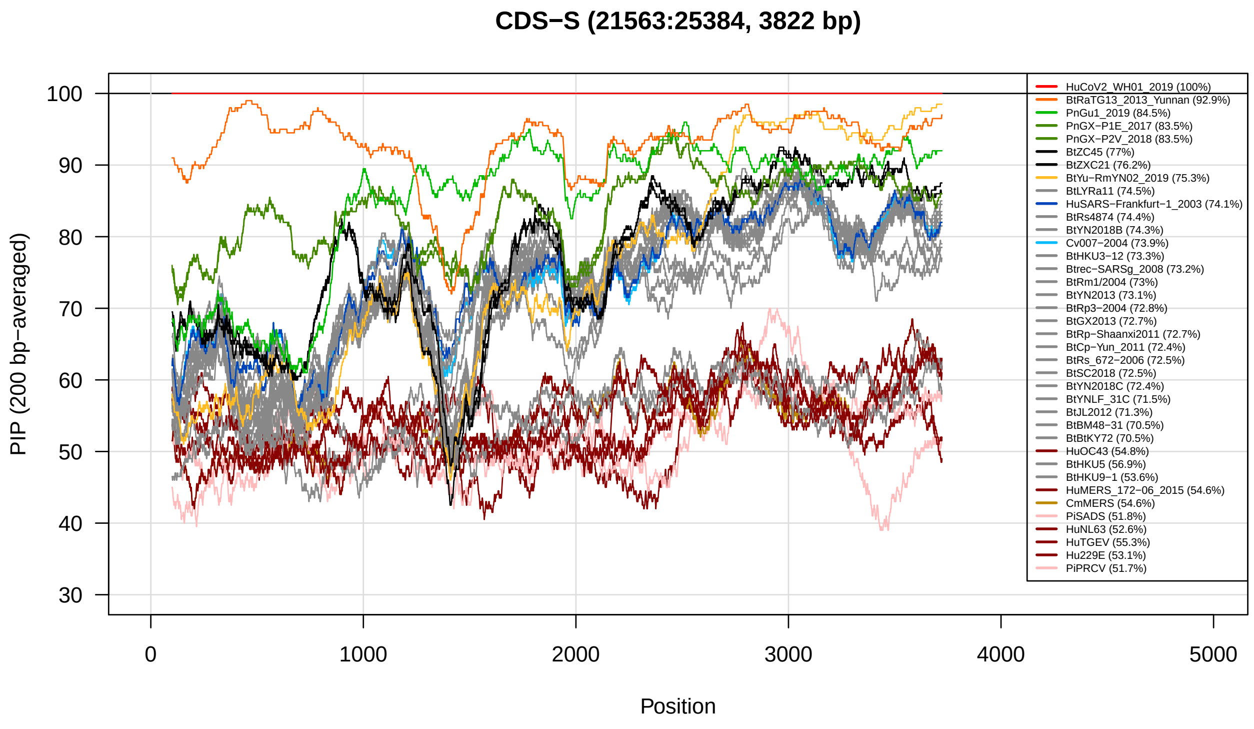

Figure 8: Feature-specific Percent Identical Positions (PIP) profiles.

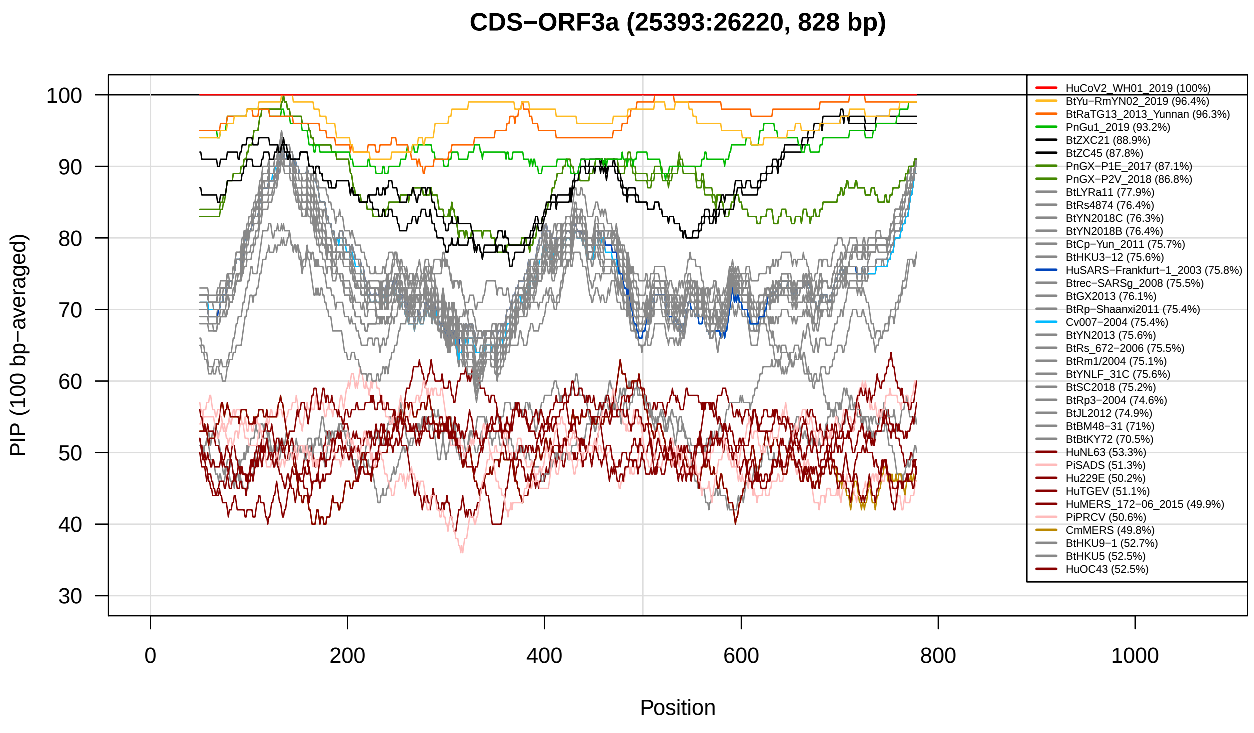

Figure 9: Feature-specific Percent Identical Positions (PIP) profiles.

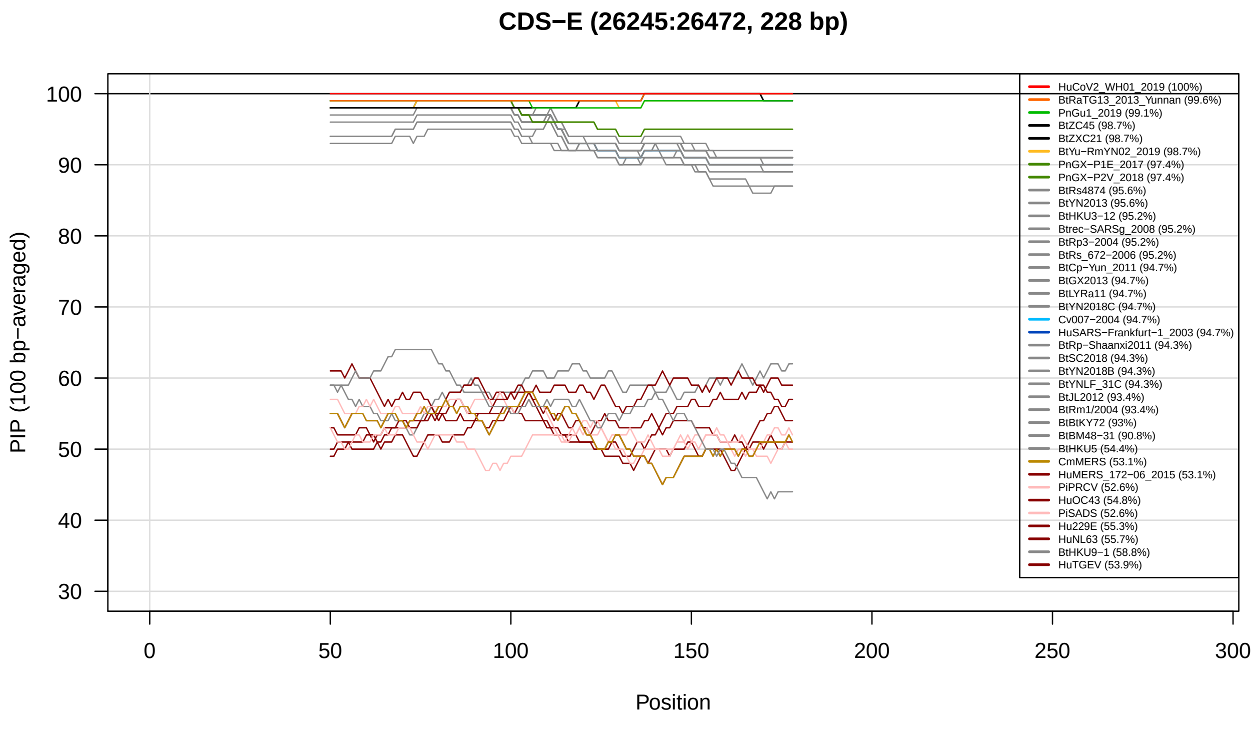

Figure 10: Feature-specific Percent Identical Positions (PIP) profiles.

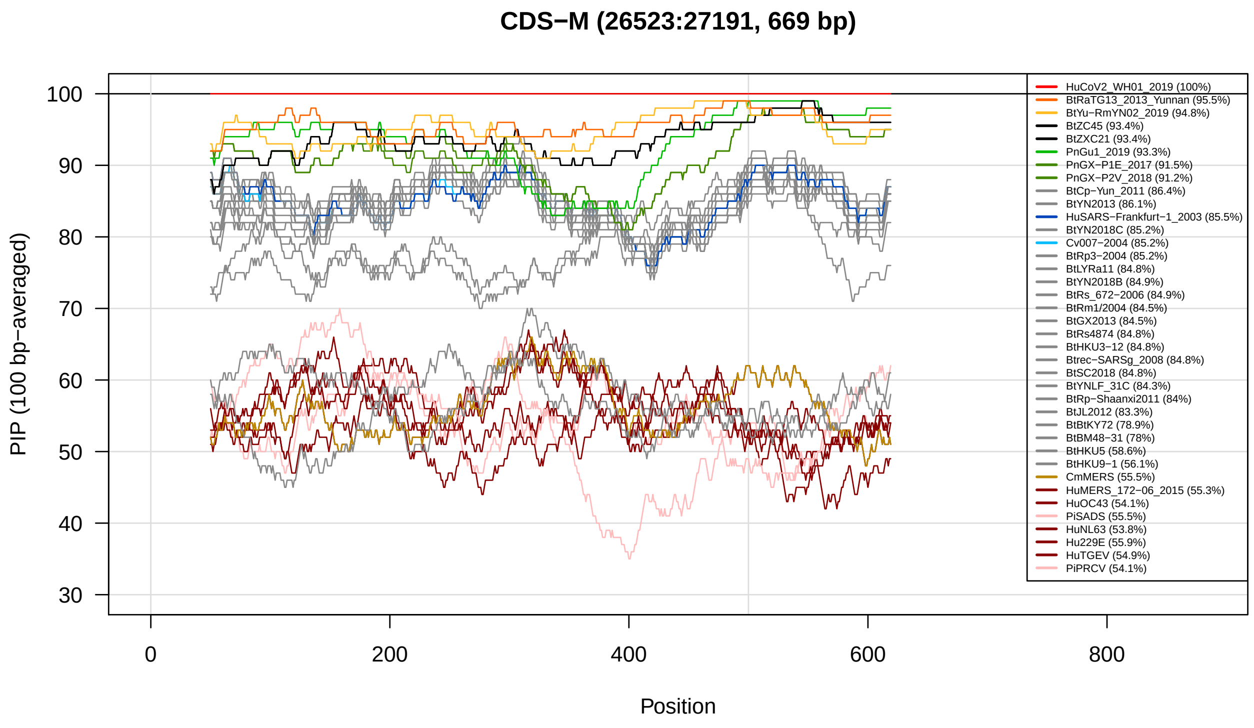

Figure 11: Feature-specific Percent Identical Positions (PIP) profiles.

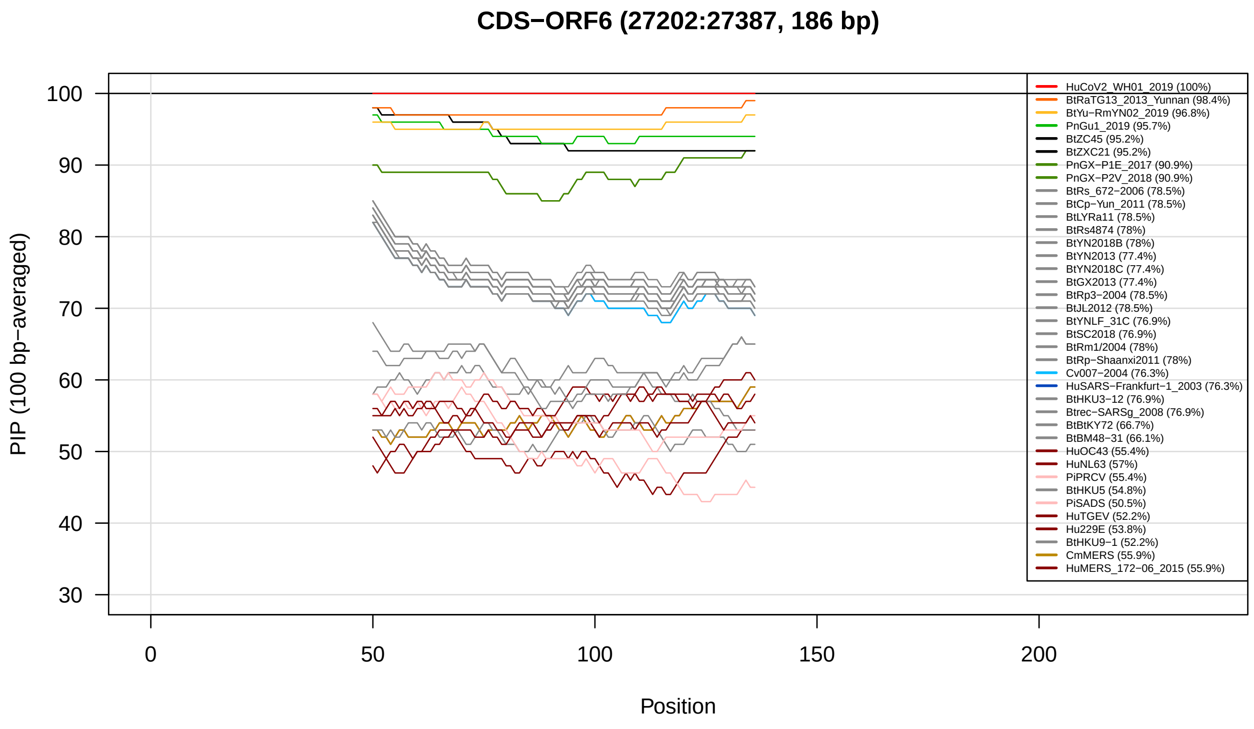

Figure 12: Feature-specific Percent Identical Positions (PIP) profiles.

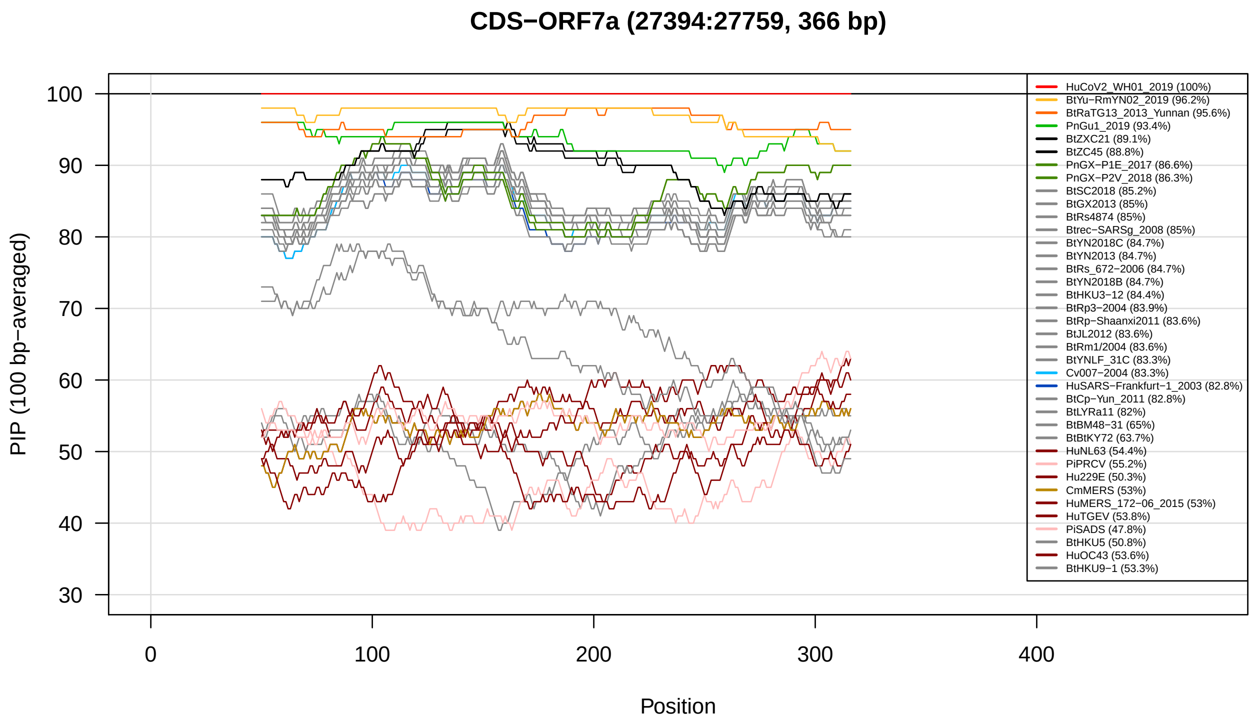

Figure 13: Feature-specific Percent Identical Positions (PIP) profiles.

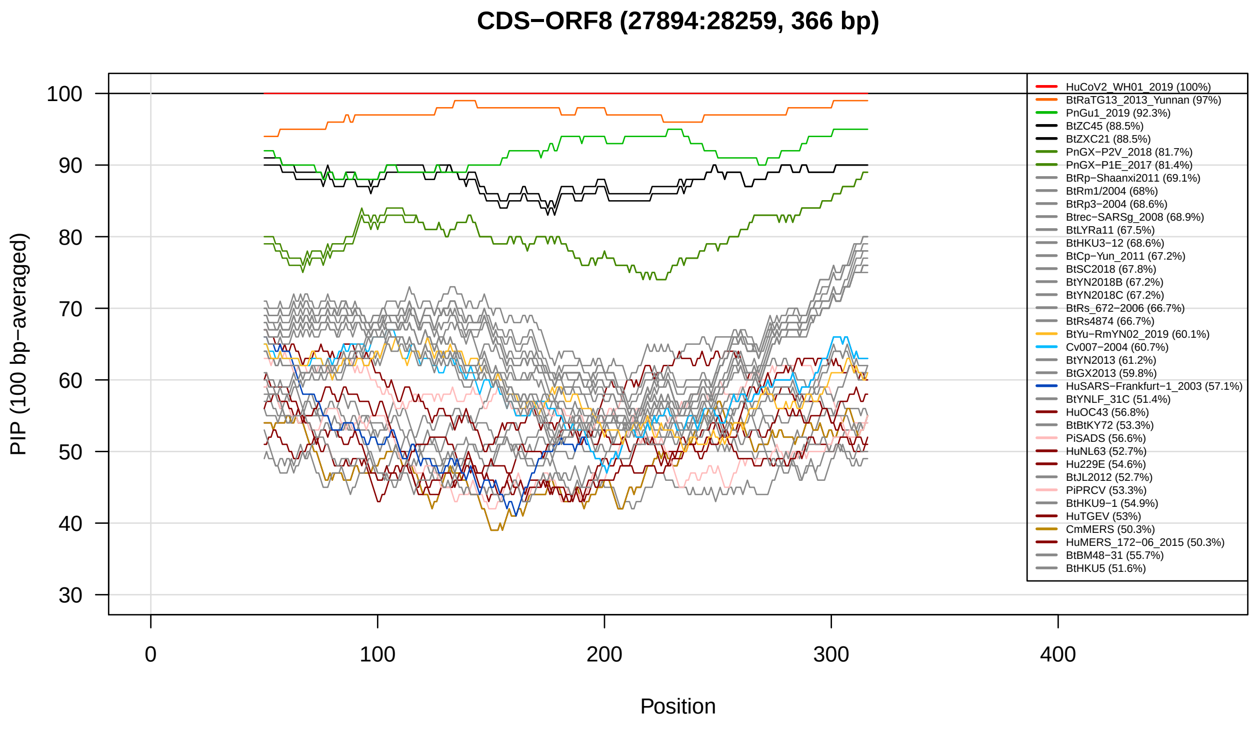

Figure 14: Feature-specific Percent Identical Positions (PIP) profiles.

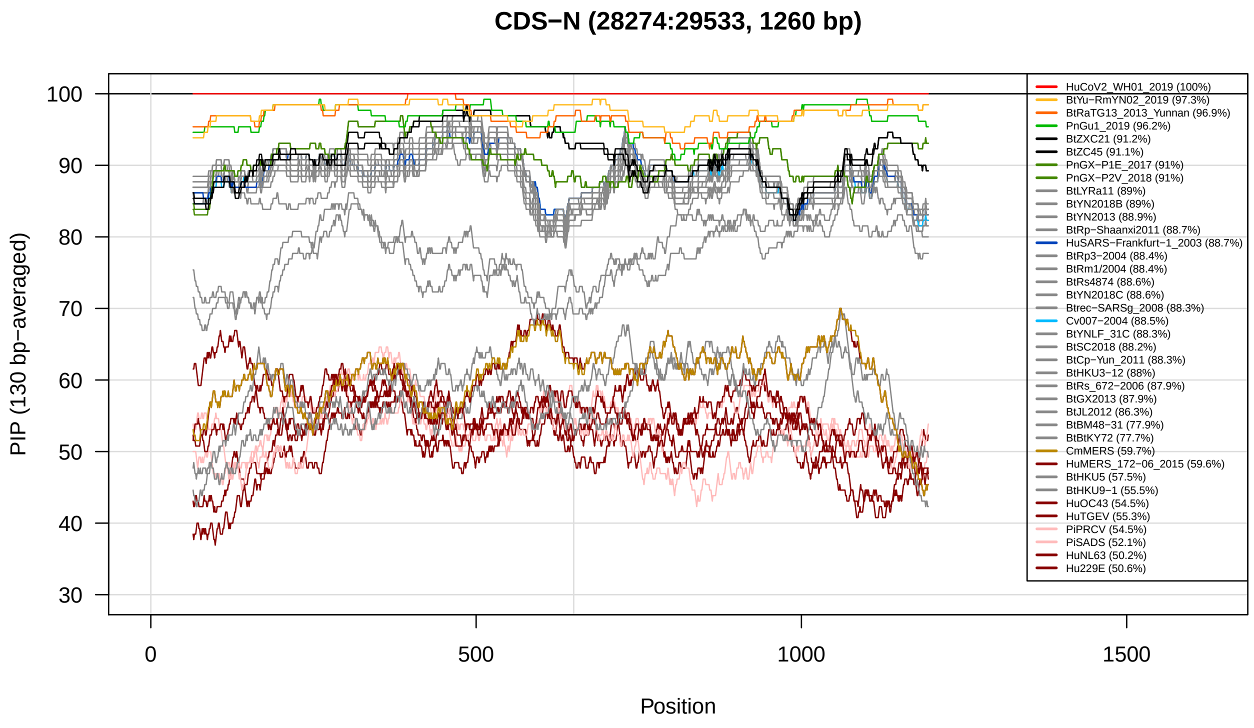

Figure 15: Feature-specific Percent Identical Positions (PIP) profiles.

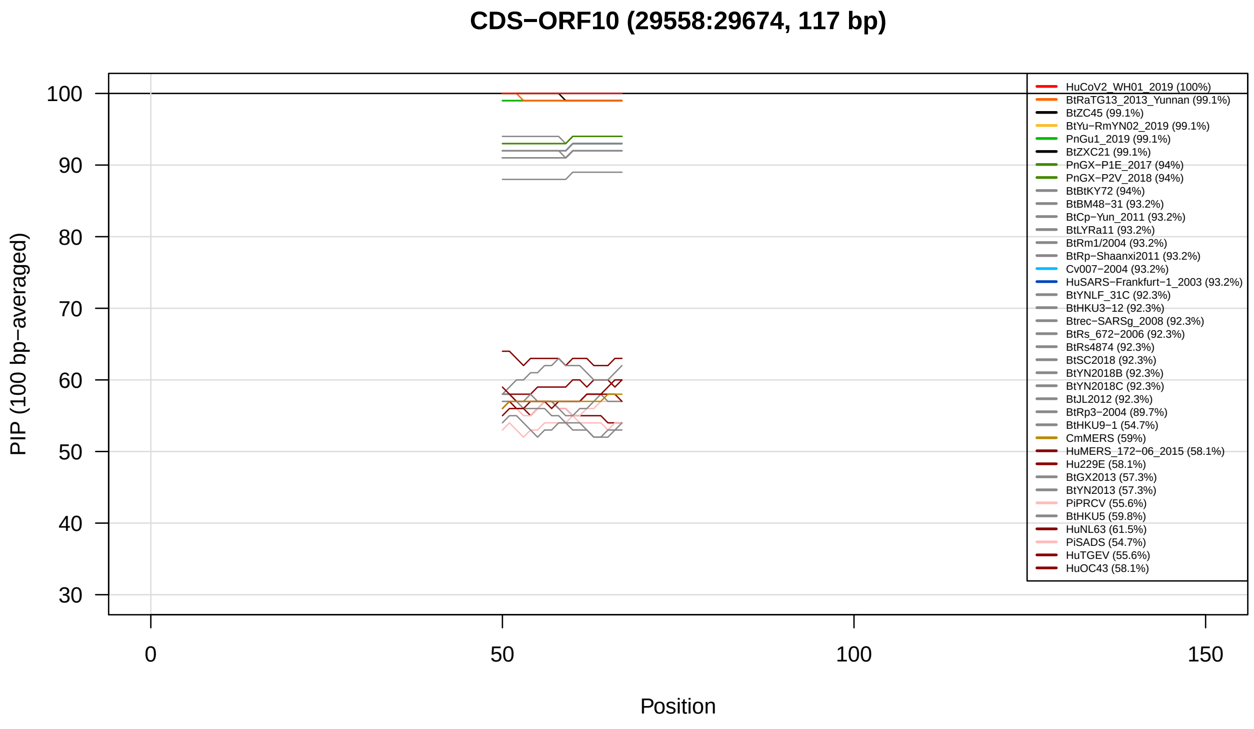

Figure 16: Feature-specific Percent Identical Positions (PIP) profiles.

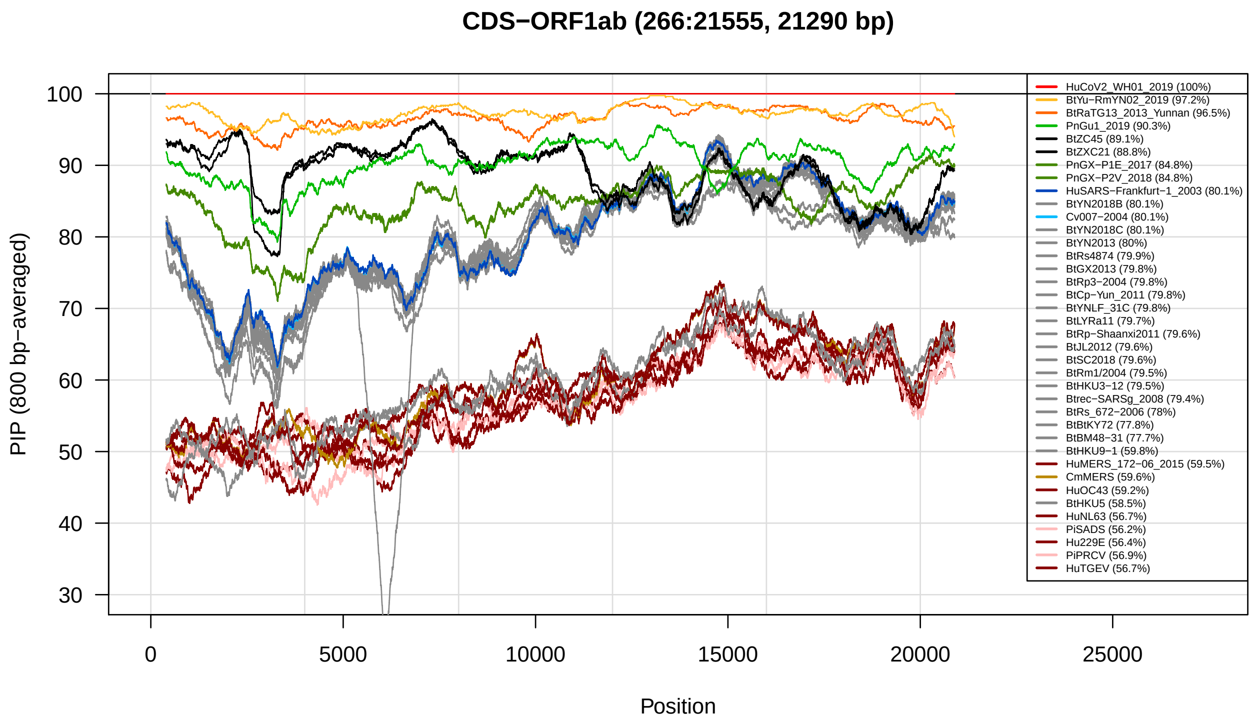

Figure 17: Feature-specific Percent Identical Positions (PIP) profiles.

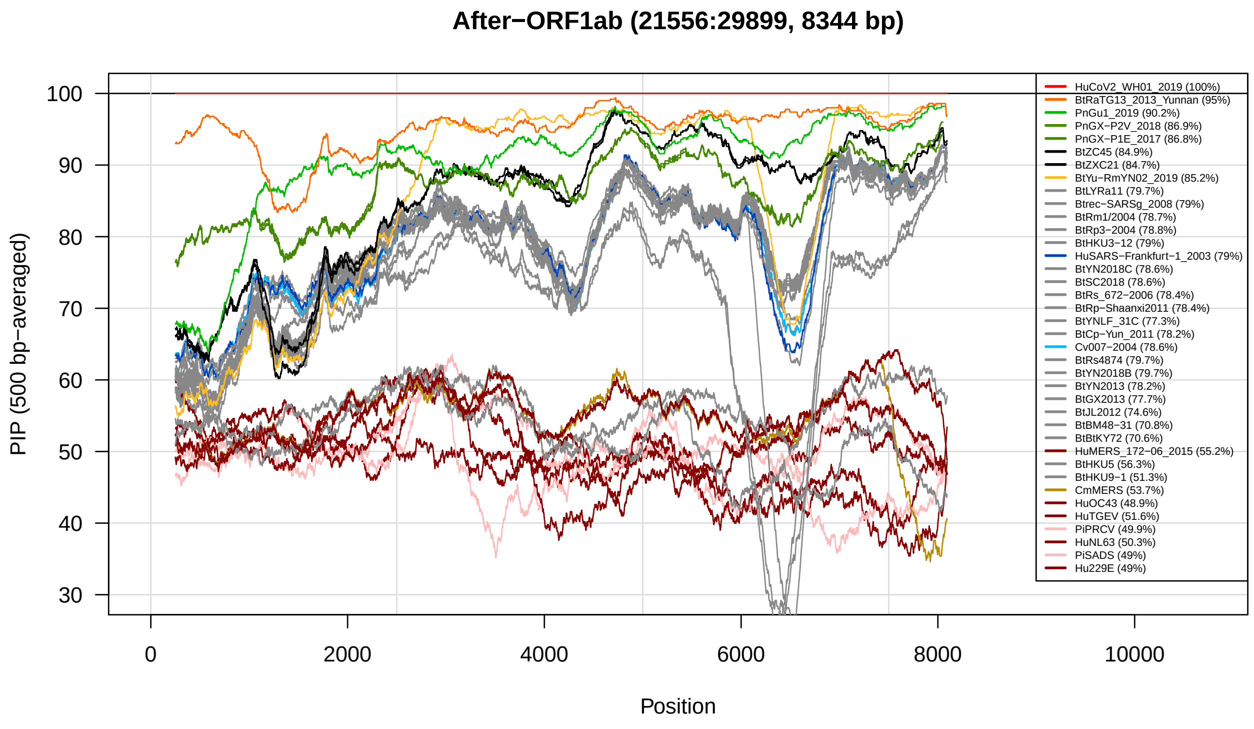

Figure 18: Feature-specific Percent Identical Positions (PIP) profiles.

## Memory image

We store the result ina memory image, in oder to be able reloading it to plot PIP profiles with different parameters.

```
save.image(file = outfiles["Memory image"])
```

## Session info

```
sessionInfo()
```

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] Biostrings_2.52.0   XVector_0.24.0      IRanges_2.18.3      S4Vectors_0.22.1    BiocGenerics_0.30.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.4          digest_0.6.25       magrittr_1.5        evaluate_0.14       highr_0.8
[15] BiocManager_1.30.10 htmltools_0.4.0
```