PloidyAssignR

Exploiting Mitotic Chromatid Segregation Patterns for Systematic Inference of Ploidy in Single Cells

PloidyAssignR is a computational framework that allows automated ploidy assignment from Strand-seq libraries. The tool observes ploidy-dependent strand state distribution patterns and optimizes kmeans clustering to assign the respective ploidy state. This vignette provides a comprehensive guide to using PloidyAssignR. We will introduce the key functionalities of PloidyAssignR and demonstrate its application using sample data from the cell line K562, including first steps of data visualization and contextualization.

```
library(PloidyAssignR)
library(ggplot2)
```

Quick Run

Customize the following function to your needs to run the analysis in one go. We recommend using the individual functions for analysis (see example workflow below) for first time users.

```
PloidyAssignR_run_all(
  input_data = "path/to/strandseq/count",
  dt_input_ROC = "path/to/ROC", # optional
  max_ploidy_force = NULL, #optional
  input_chrom = NULL, # default: all chromosomes present in data set
  input_window = 1e+07, # default: window width 10 Mbp
  input_step = 5e+06, # default: step length 5 Mbp
  export_path = NULL, # default: does not save data to file.
  export_file_name = "PloidyAssignR"
)
```

Example Workflow: Ploidy Assignment in K562 Strand-seq libraries

Input Data Set

PloidyAssignR requires binned strand-specific count data such as obtained from Strand-seq libraries. Such count data sets can be generated using the "MosaiCatcher Pipeline" (https://github.com/friendsofstrandseq/mosaicatcher-pipeline). We also recommend selecting Strand-seq libraries for analysis using ASHLEYS quality control pipeline (https://github.com/friendsofstrandseq/ashleys-qc-pipeline).

```
data_K562_example_input <- input_prep(input_data = data_K562_strand_seq_count)

data_K562_example_input
#>          chrom    start      end      sample   cell c w class
#>       1:  chr1        0   100000 K562_select 01/301 0 0    WW
#>       2:  chr1        0   100000 K562_select 01/304 2 0    WC
#>       3:  chr1        0   100000 K562_select 01/306 2 1    WC
#>       4:  chr1        0   100000 K562_select 01/312 1 1    WC
#>       5:  chr1        0   100000 K562_select 01/316 1 0    WC
#>      ---
#> 2502410:  chrY 57200000 57227415 K562_select 03/390 0 0  None
```

```
#> 2502411:  chrY 57200000 57227415 K562_select 03|392 0 0  None
#> 2502412:  chrY 57200000 57227415 K562_select 03|393 0 0  None
#> 2502413:  chrY 57200000 57227415 K562_select 03|395 0 0  None
#> 2502414:  chrY 57200000 57227415 K562_select 03|396 0 0  None
```

Calculating Relative W-strand State Frequency Fraction_w

The relative W-strand state frequency fraction_w represents the proportion of W-oriented strands. PloidyAssignR applies a sliding window across the length of a chromosome and calculates fraction_w from the Strand-seq count data. Choose parameters to specify window width and step length as well as the chromosomes included in analysis.

```
data_example <- calWatsonFractions(data_K562_example_input,
                                   input_chrom = NULL,
                                   # default: all chromosomes present in data set
                                   input_window = 1000000,
                                   # default: window width 10 Mbp
                                   input_step = 500000
                                   # default: step length 5 Mbp
                                   )
data_example
#>         chrom    start      end      sample   cell total_count fraction_w
#>      1:  chr1        0  1000000 K562_select 01|301          53  0.9622642
#>      2:  chr1        0  1000000 K562_select 01|304          58  0.5517241
#>      3:  chr1        0  1000000 K562_select 01|306          78  0.4102564
#>      4:  chr1        0  1000000 K562_select 01|312          68  0.5294118
#>      5:  chr1        0  1000000 K562_select 01|316          23  0.3043478
#>     ---
#> 473859:  chrY 56500000 57227415 K562_select 03|390         114  0.5087719
#> 473860:  chrY 56500000 57227415 K562_select 03|392          86  0.4302326
#> 473861:  chrY 56500000 57227415 K562_select 03|393         201  0.4577114
#> 473862:  chrY 56500000 57227415 K562_select 03|395         226  0.4778761
#> 473863:  chrY 56500000 57227415 K562_select 03|396         225  0.5155556
```

Assigning Ploidy States

The relative W-strand state frequency forms ploidy-dependent distribution patterns (modeled by a binomial distribution). PloidyAssignR counts the number of clusters in these distribution patterns for each sliding window. The number of clusters k is directly connected to the ploidy state n: n=k-1 The maximum ploidy for a data set is limited by the number of cells and automatically determined. The user can force a maximum ploidy by setting max_ploidy_force to a specific value (max_ploidy = 1...n)

```
data_example <- assignConsensusPloidy(data_example)
print(data_example)
#>         chrom    start      end      sample   cell total_count fraction_w sil_k
#>      1:  chr1        0  1000000 K562_select 01|301          53  0.9622642     6
#>      2:  chr1        0  1000000 K562_select 01|304          58  0.5517241     6
#>      3:  chr1        0  1000000 K562_select 01|306          78  0.4102564     6
#>      4:  chr1        0  1000000 K562_select 01|312          68  0.5294118     6
#>      5:  chr1        0  1000000 K562_select 01|316          23  0.3043478     6
#>     ---
#> 473859:  chrY 56500000 57227415 K562_select 03|390         114  0.5087719     6
#> 473860:  chrY 56500000 57227415 K562_select 03|392          86  0.4302326     6
#> 473861:  chrY 56500000 57227415 K562_select 03|393         201  0.4577114     6
```

```
#> 473862:  chrY 56500000 57227415 K562_select 03|395         226  0.4778761     6
#> 473863:  chrY 56500000 57227415 K562_select 03|396         225  0.5155556     6
#>         cons_ploidy
#>      1:           5
#>      2:           5
#>      3:           5
#>      4:           5
#>      5:           5
#>     ---
#> 473859:           5
#> 473860:           5
#> 473861:           5
#> 473862:           5
#> 473863:           5
```

Visualization of Ploidy Analysis and Preparing for SC Copy Number Assignment

The ploidy assignment results can be visualized using a karyogram style bar graph and a scatter plot of the distribution patterns of fraction_w.

Distribution Pattern Plot The distribution pattern plot is the graphic representation of the relative W-strand state frequency fraction_w across the genome. Each dot is the fraction_w value of an individual cell within a sliding window. The distribution pattern arises from the binomially distributed of mitotic strand segregation. The underlying ploidy state is determined by counting the number of clusters. The distribution pattern plot includes a colored strip at the bottom, which depicts the ploidy-state as determined by PloidyAssignR.

This plot can be used to visually evaluate the ploidy assignment results and distinguish regions with more homogeneous or heterogeneous distribution patterns.

```
fct_plot_distribution_patterns(data_example, input_chrom ="chr3", point_size = 0.2)
```

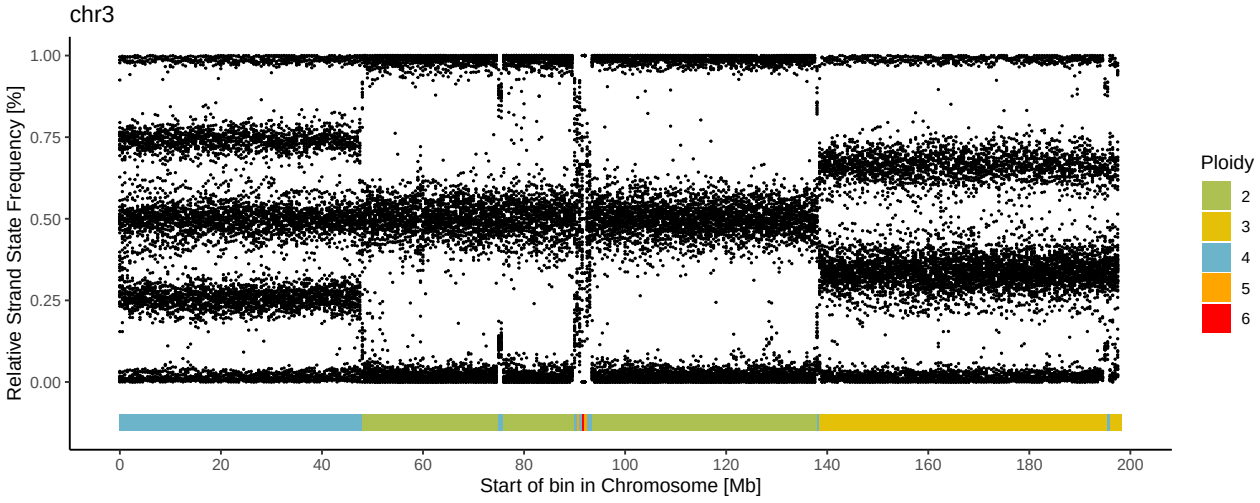

Figure 1: Figure: Distribution Patterns of Chr7 K562

Karyogram Plot The consensus ploidy state is color-coded (see graph legend) and represents the majority ploidy state of the cell population. It gives a quick visual overview of the overall ploidy and the copy number of individual chromosome segments.

```
fct_plot_karyogram(data_example)
```

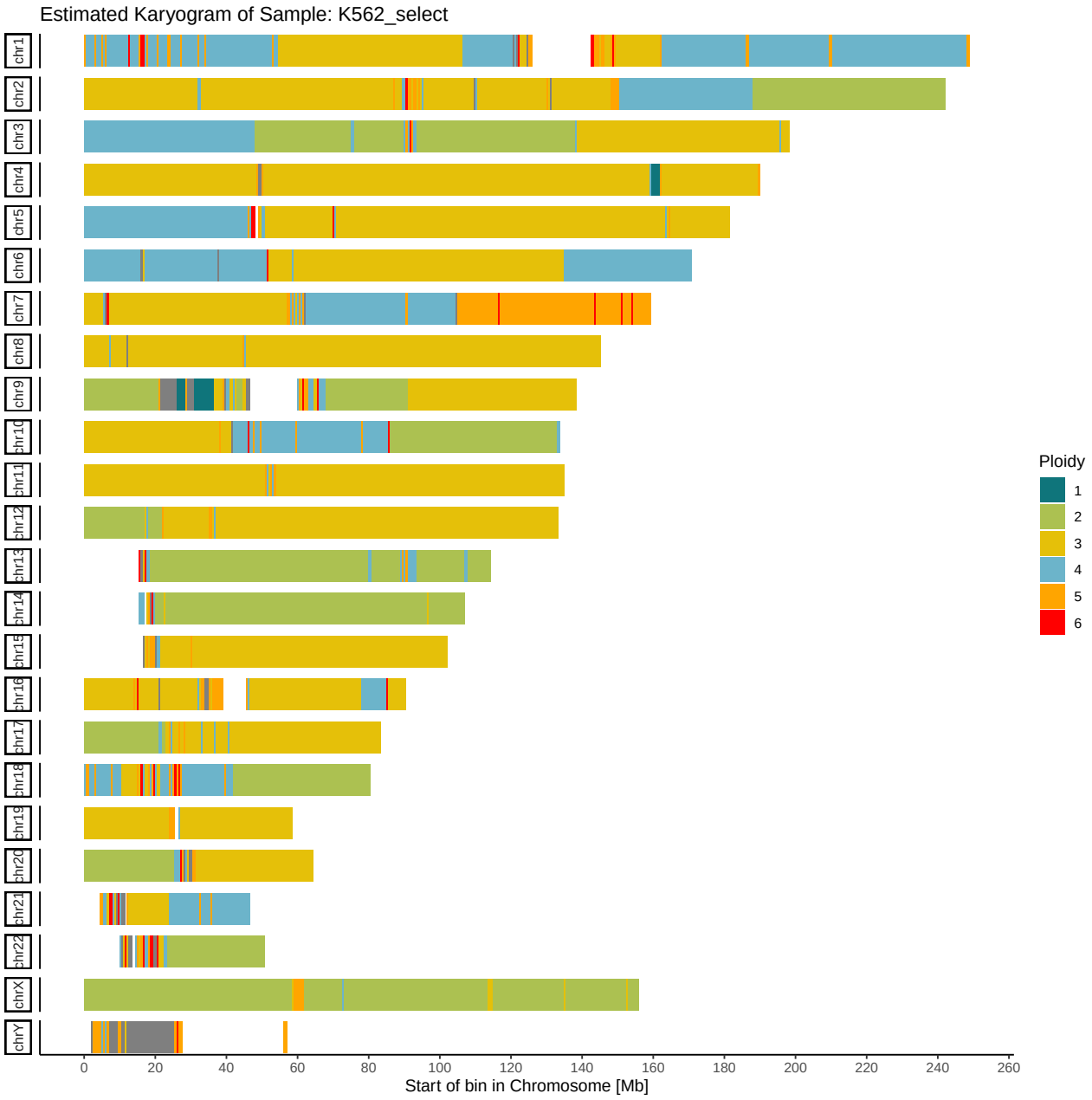

Figure 2: Figure: Ploidy Assignment for K562

To obtain a rough estimate of the ploidy states contributing to the mean ploidy of a sample, we count the number of bins for each ploidy state:

```
data_summary <- ploidy_summary(data_K562_ploidy)
print(data_summary)
#>          sample mean_ploidy cons_ploidy window percent
#> 1: K562_select        3.07           1     21     0.4
#> 2: K562_select        3.07           2   1380    23.6
#> 3: K562_select        3.07           3   3040    52.1
```

```
#> 4: K562_select        3.07           4   1086    18.6
#> 5: K562_select        3.07           5    238     4.1
#> 6: K562_select        3.07           6     38     0.7
#> 7: K562_select        3.07           7     34     0.6
```

Understanding Distribution Patterns

A significant amount information can be obtained just from inspecting the distribution pattern plot, by itself. The following steps can help to obtain a basic understanding of the sample's karyotype:

1. Flip through all chromosomes and check whether the ploidy state in the colored-strip matches the number of visually distinguishable clusters

- Ploidy n = Number of Clusters k - 1

2. If there are regions where visually the number of clusters does not match the ploidy state, consider the following:

- Could the distribution pattern be a mix of several different distribution pattern and result from subclonal copy number deviation?
- Are these regions of max_ploidy or higher? Then the number of cells might be too low for pattern-based ploidy assignment. Perform SC Copy Number Assignment and then reevaluate.
- Are these centromeric regions? These regions are highly repetitive and the quality of sequencing alignment might be too low for analysis. Consider excluding these regions from analysis.

3. Highlight individual cells and observe the fraction_w values across the chromosome. Fraction_w changes can results from:

- structural variation that impacts strand orientation (e.g. inversions)
- structural variation where chromosome segments co-segregate with other chromosomes (e.g. translocations). These translocated segments are aligned to their chromosome origin, however are physically linked to another chromosome. That way these segments have the same strand state as the recipient chromosome. As a result, the fraction_w value changes in some cells. This is exploited for translocation analysis in diploid cells (Sanders et al. 2020). In the case of unbalanced translocations, these translocated segments appear as a ploidy gain.

Single Cell Copy Number Assignment

PloidyAssignR normalizes the Strand-seq read count by determining a count baseline for Regions of Confidence (ROC) selected by the user (see above). This way we can firstly detect copy number of each individual cell and sliding window and secondly also detect regions with higher amplification levels as determined as max_ploidy.

Selecting Regions of Confidence Flip through the distribution pattern plots to select ROC. ROC should have a low likelihood of subclonal copy number deviation contributing to the consensus ploidy and therefore should fulfill the following criteria:

1. homogeneous strand state distribution pattern/clustering
2. narrow clustering of strand states
3. individual fraction_w values of cells should be consistent

For this analysis we have selected the following ROC:

```
#>    chrom    start       end
#> 1:  chr1 57000000  72000000
#> 2:  chr7 21500000  49000000
#> 3:  chr8 88000000 132500000
#> 4: chr13 20000000  49000000
```

```
#> 5: chr18 51500000  66000000
#> 6: chr20 17500000  22500000
```

Create a text file with the columns chrom, start, end and import as data.table:

```
your_ROC <- data.table::fread(file = "path/to/ROC")
```

Normalization of Read Count Data PloidyAssignR determines the mean read count in the ROC and divides the mean read count with the respective ploidy state of those ROC. The resulting baseline read count (that is the read count of one homolog) is then used to normalize all other regions to obtain the single cell copy number.

```
data_example <- scCoverageNorm(data_example,
                               data_K562_ROC)

print(data_example)
#>            cell chrom    start      end      sample total_count fraction_w sil_k
#>       1: 01|301  chr1        0  1000000 K562_select          53  0.9622642     6
#>       2: 01|301  chr1   500000  1500000 K562_select         114  0.9824561     5
#>       3: 01|301  chr1  1000000  2000000 K562_select         129  1.0000000     5
#>       4: 01|301  chr1  1500000  2500000 K562_select         141  0.9787234     5
#>       5: 01|301  chr1  2000000  3000000 K562_select         171  0.9707602     5
#>      ---
#> 473859: 03|396  chrY 25500000 26500000 K562_select           3  0.3333333     6
#> 473860: 03|396  chrY 26000000 27000000 K562_select          10  0.4000000     7
#> 473861: 03|396  chrY 26500000 27500000 K562_select           7  0.4285714     6
#> 473862: 03|396  chrY 56000000 57000000 K562_select         225  0.5155556     6
#> 473863: 03|396  chrY 56500000 57227415 K562_select         225  0.5155556     6
#>         cons_ploidy baseline_coverage norm_count
#>       1:           5          52.46187 1.01025764
#>       2:           4          52.46187 2.17300700
#>       3:           4          52.46187 2.45892898
#>       4:           4          52.46187 2.68766655
#>       5:           4          52.46187 3.25951050
#>      ---
#> 473859:           5         108.87763 0.02755387
#> 473860:           6         108.87763 0.09184623
#> 473861:           5         108.87763 0.06429236
#> 473862:           5         108.87763 2.06654023
#> 473863:           5         108.87763 2.06654023
```

Single Cell Heatmap

To visualize the single cell copy number within the context of the cell population we use a heatmap that uses hierarchical clustering to group cells. See an example plot of Chr2 for K562 below.

```
fct_plot_sc_heatmap(data_example, input_chrom = "chr2")
```

Example Analysis of Subclonal Copy Number Deviation in Chr20 p-arm of K562

The p-arm of Chr20 of K562 carries an example for subclonal copy number deviation that can be detected using PloidyAssignR. We can observe several signs of individual cells not following the majority ploidy state:

1. Heterogeneous distribution pattern
2. A subset of cells has a different SC copy number and is clustered together

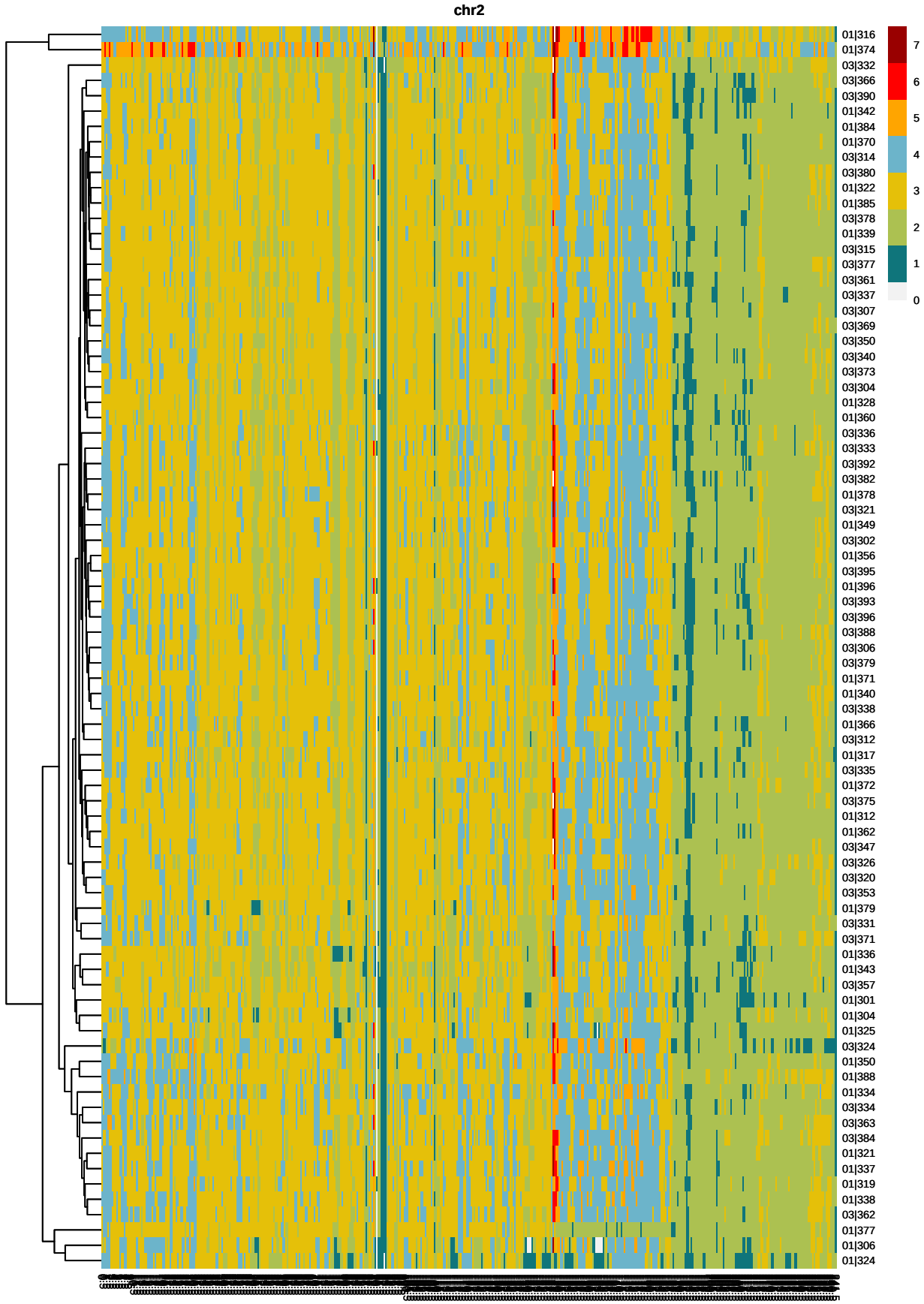

Figure 3: Figure: Example Heatmap of SC Copy Number - Chr2 K562

```
fct_plot_sc_heatmap(data_K562_ploidy, input_chrom = "chr20")
```

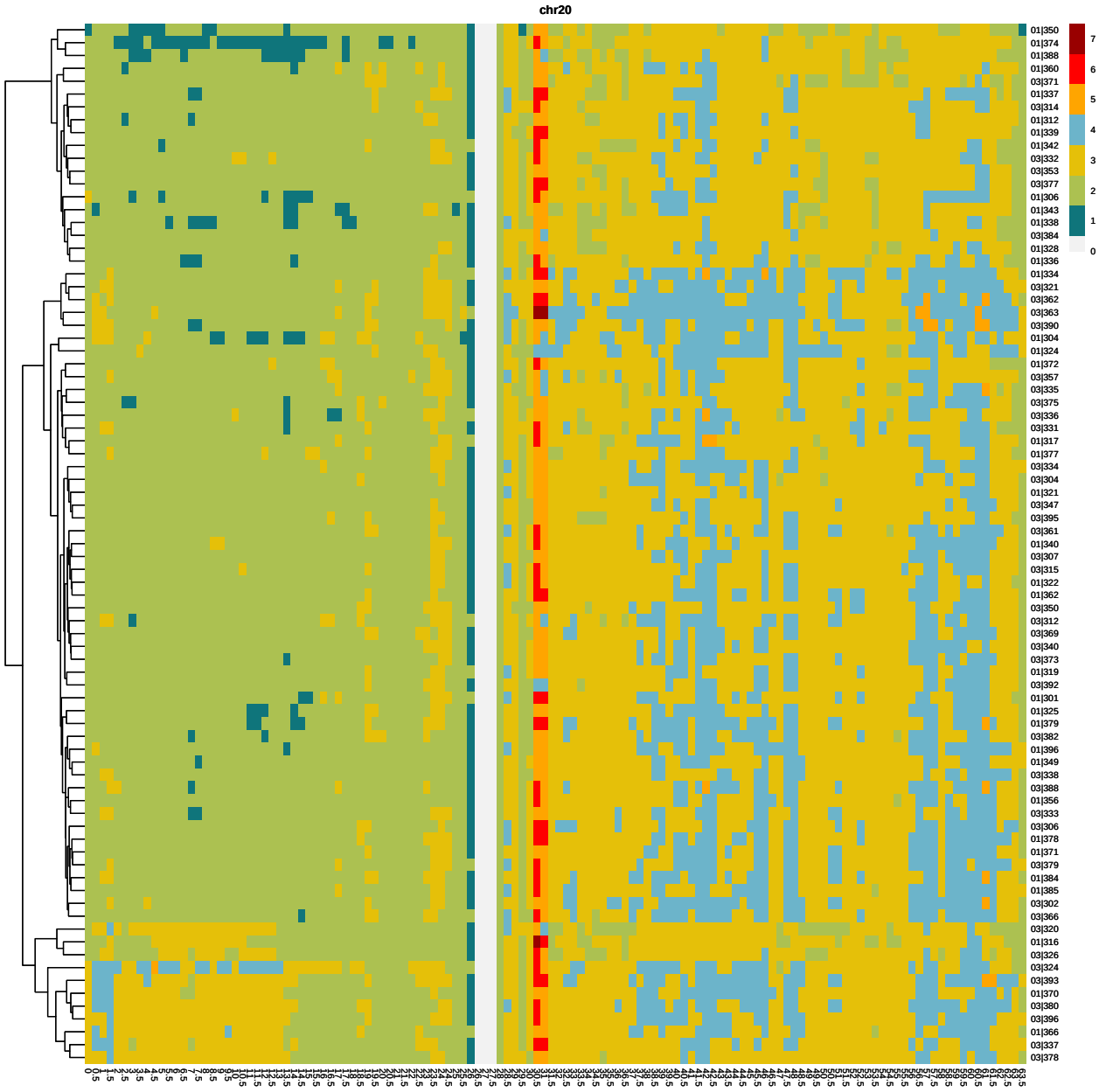

Figure 4: Figure: K562 Chr20 Subclonal Copy Number Deviation in Eight Cells (Trisomic Signal in p-Arm)

```
fct_plot_distribution_patterns(data_K562_ploidy[start<=35000000],
                               input_chrom ="chr20", point_size = 0.2,
                               input_cell = "03|380", reg_color = "grey",
                               cell_color = "black", cell_size = 0.7)
```

Strand-seq Count Plot of individual Chromosomes and selected Cells

The original Strand-seq count data can be used to validate the copy number assignment by PloidyAssignR:

- For this purpose we plot the Strand-seq count data of the cells determined to be the subclone: “03|393”,

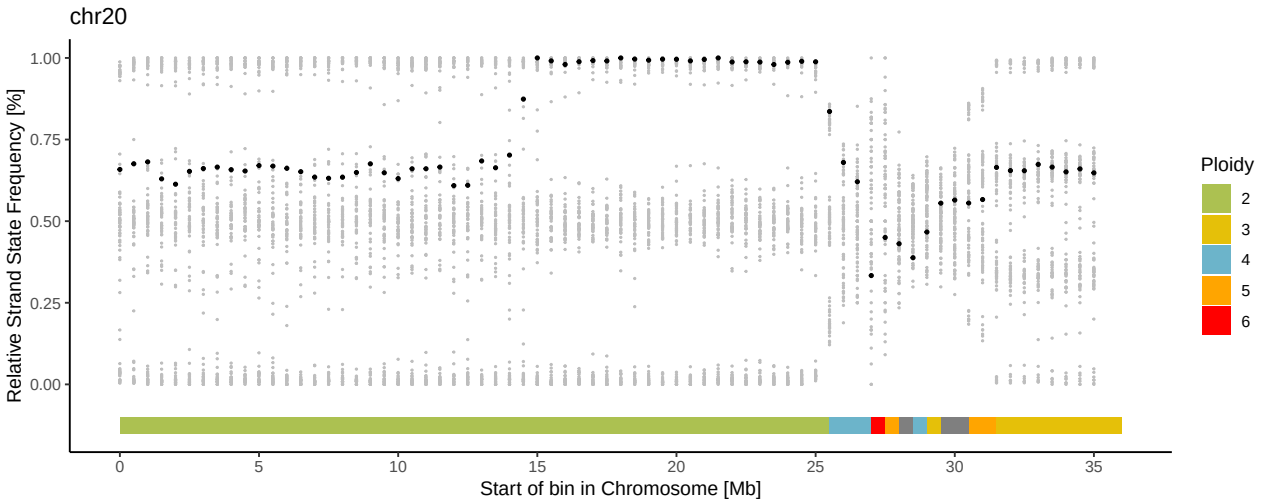

Figure 5: Figure: K562 Chr20 p-arm - Example Cell Deviating from Majority Distribution Pattern

"03|324" ,"01|366", "03|337", "03|380", "03|396", "01|370" and "03|378"

- This is then compared to randomly chosen cells of the major clone. We selected eight random cells for better visualization: "01|384", "03|340", "01|379", "01|306", "03|314", "01|342", "01|317" and "03|388"

```
subclone_cells <- c("03|393", "03|324" ,"01|366", "03|337", "03|380", "03|396", "01|370", "03|378")

plot_subclone <- fct_plot_counts_cellwise(data_K562_strand_seq_count[start<=35000000],
                                          data_SC_CN = data_K562_ploidy,
                          selected_cells = subclone_cells,
                          input_chrom = "chr20",
                          lab_title = "K562 Chr20p Subclone Candidates"
                          )
plot_subclone + ylim(-50,50)
```

```
K562_major_clone <- data_K562_strand_seq_count[chrom=="chr20"&start<=30000000]
K562_major_clone <-K562_major_clone[!(cell%in% subclone_cells)]

major_clone <- K562_major_clone[,unique(cell)]
major_clone_subset <- major_clone[1:8]
plot_majority <- fct_plot_counts_cellwise(data_K562_strand_seq_count[start<=35000000],
                                          data_SC_CN = data_K562_ploidy,
                          selected_cells = major_clone_subset,
                          input_chrom = "chr20",
                          lab_title = "K562 Chr20p Eight Cells from Major Clone"
                          )
plot_majority + ylim(-50,50)
```

The eight cells of the subclone carry a trisomic signal in Chr20p arm.

References

Sanders AD, Meiers S, Ghareghani M, Porubsky D, Jeong H, van Vliet MACC, Rausch T, Richter-Pechańska P, Kunz JB, Jenni S, Bolognini D, Longo GMC, Raeder B, Kinanen V, Zimmermann J, Benes V, Schrappe M, Mardin BR, Kulozik AE, Bornhauser B, Bourquin JP, Marschall T, Korbel JO. Single-cell analysis of structural variations and complex rearrangements with tri-channel processing. Nat Biotechnol. 2020 Mar;38(3):343-354. doi: 10.1038/s41587-019-0366-x. Epub 2019 Dec 23. PMID: 31873213; PMCID: PMC7612647.

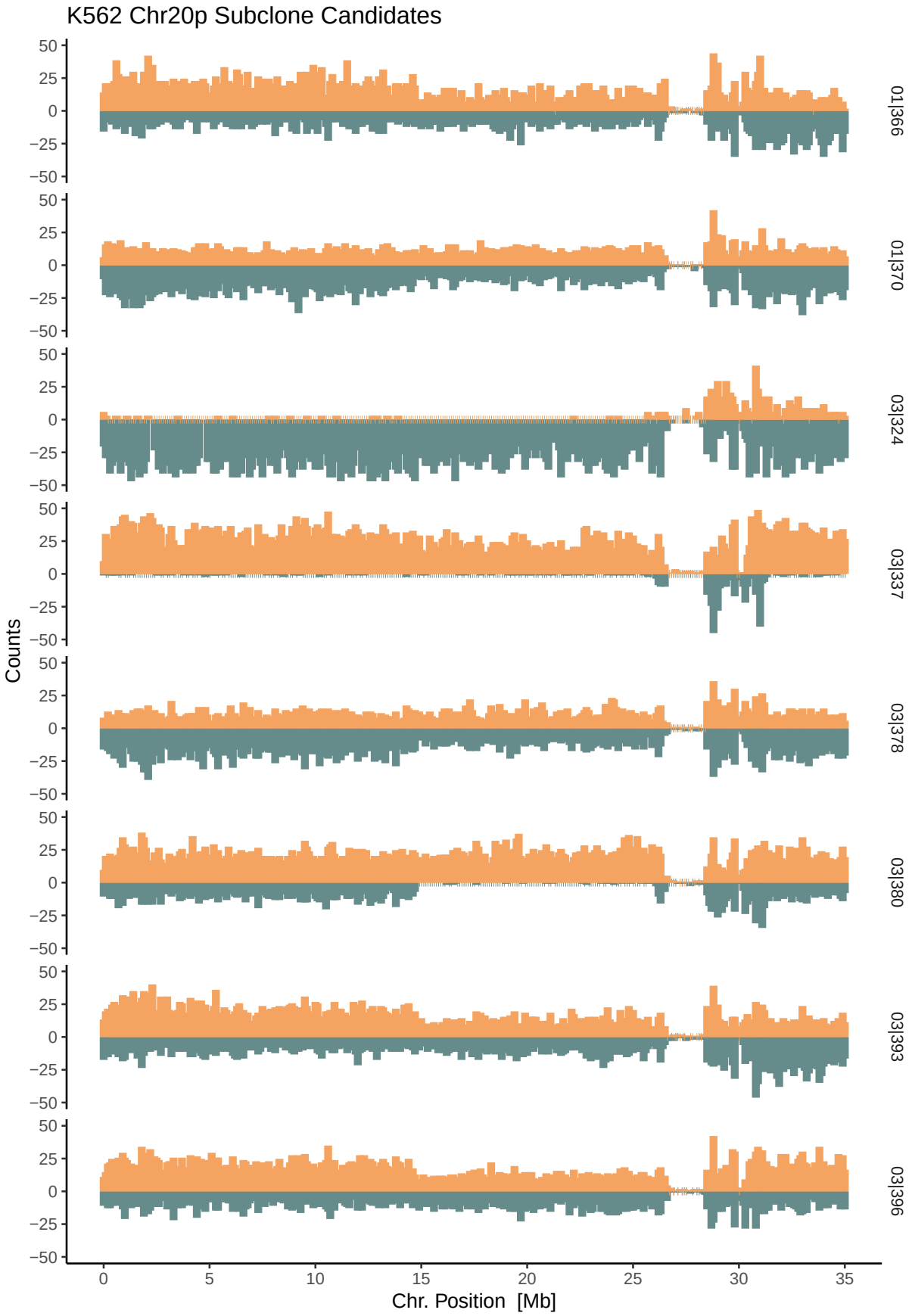

Figure 6: Figure: Scaled Strand-seq Count Plot of K562 Cells with Subclonal Trisomic Signal in Chr20p

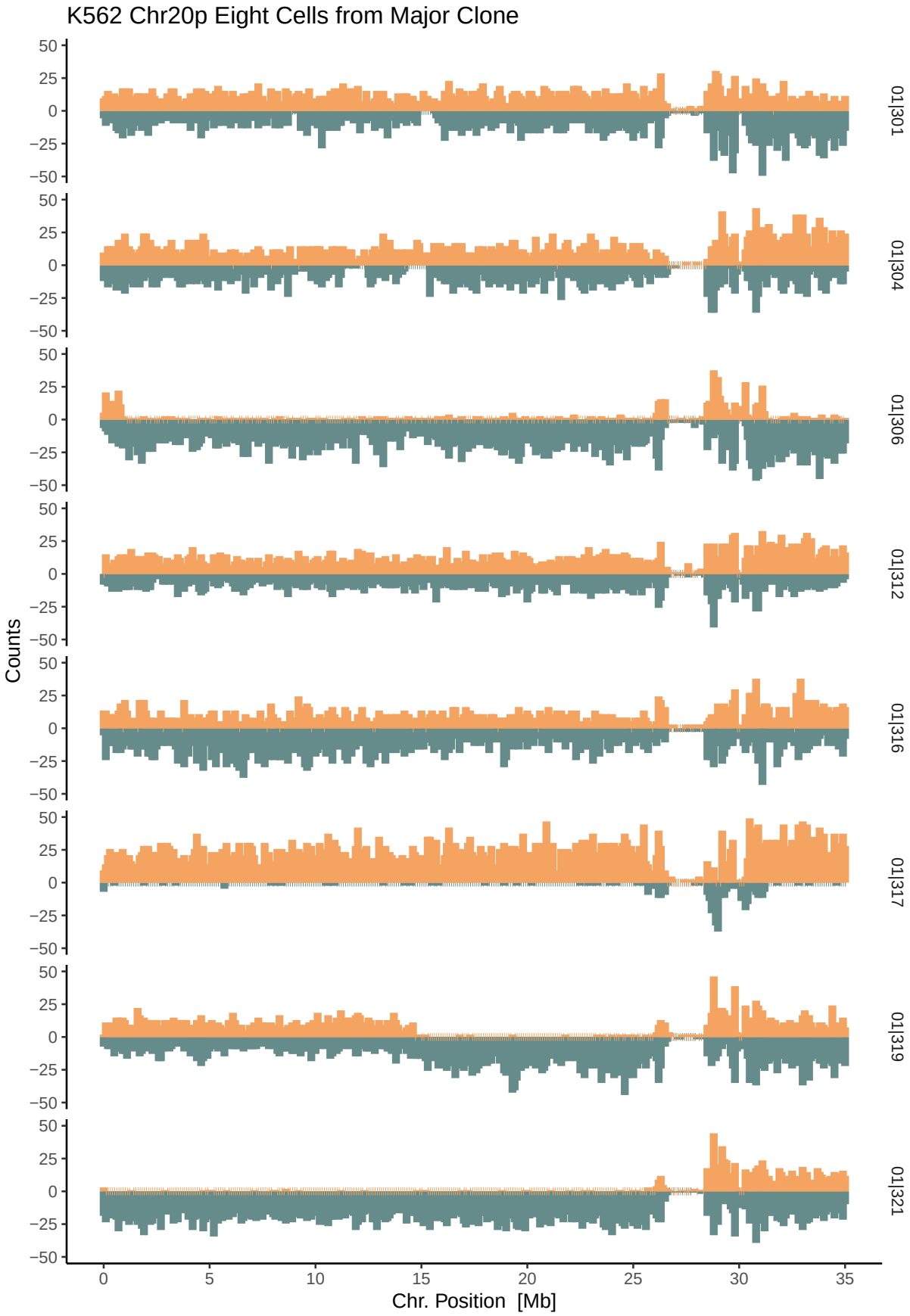

Figure 7: Figure: Scaled Strand-seq Count Plot of eight K562 Cells Chr20p as Example for the Majority Disomic Signal

```
sessionInfo()
#> R version 4.2.2 (2022-10-31)
#> Platform: x86_64-apple-darwin17.0 (64-bit)
#> Running under: macOS Big Sur ... 10.16
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.2/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] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] ggplot2_3.4.2       PloidyAssignR_1.0.0
#>
#> loaded via a namespace (and not attached):
#>  [1] tidyselect_1.2.0    xfun_0.39           bslib_0.4.2
#>  [4] purrr_1.0.1         colorspace_2.1-0    vctrs_0.6.2
#>  [7] generics_0.1.3      htmltools_0.5.5     yaml_2.3.7
#> [10] utf8_1.2.3          rlang_1.1.1         pillar_1.9.0
#> [13] later_1.3.1         jquerylib_0.1.4     withr_2.5.0
#> [16] glue_1.6.2          RColorBrewer_1.1-3  lifecycle_1.0.3
#> [19] munsell_0.5.0       gtable_0.3.3        fontawesome_0.5.1
#> [22] htmlwidgets_1.6.2   evaluate_0.21       labeling_0.4.2
#> [25] knitr_1.43          fastmap_1.1.1       golem_0.4.0
#> [28] Cairo_1.6-0         httpuv_1.6.11       fansi_1.0.4
#> [31] Rcpp_1.0.10         xtable_1.8-4        scales_1.2.1
#> [34] promises_1.2.0.1    DT_0.28             cachem_1.0.8
#> [37] jsonlite_1.8.4      config_0.3.1        farver_2.1.1
#> [40] mime_0.12           digest_0.6.31       dplyr_1.1.2
#> [43] shiny_1.7.4         grid_4.2.2          cli_3.6.1
#> [46] tools_4.2.2         magrittr_2.0.3      sass_0.4.6
#> [49] tibble_3.2.1        cluster_2.1.4       tidyr_1.3.0
#> [52] pkgconfig_2.0.3     ellipsis_0.3.2      pheatmap_1.0.12
#> [55] data.table_1.14.8   shinyFeedback_0.4.0 attempt_0.3.1
#> [58] assertthat_0.2.1    rmarkdown_2.21      rstudioapi_0.14
#> [61] R6_2.5.1            compiler_4.2.2
```