Haplotype-based Pangenome Analysis in Wheat

As the writer of this script I would like to express appreciation and give all credit to the authors of the following article for creating original raw data and scripts, which are required for this R script. Brinton, J., Ramirez-Gonzalez, R. H., Simmonds, J., Wingen, L., Orford, S., Griffiths, S., 10 Wheat Genome Project, Haberer, G., Spannagl, M., Walkowiak, S., Pozniak, C., & Uauy, C. (2020). A haplotype-led approach to increase the precision of wheat breeding. *Communications biology*, 3(1), 712. https://doi.org/10.1038/s42003-020-01413-2 (https://doi.org/10.1038/s42003-020-01413-2) This script was written in R 4.1.0 by Jose Antonio Montero Tena as part of his master thesis in 2021. Free use and modification of this script for personal interest is encouraged.

Introduction

SNP-haplotypes discovered from marker arrays can be redefined as IBS-haplotypes, which are identical-by-state sequences sharing 100 percentage of identity between the genomes and discovered by sequence comparison. Brinton et al developed a method in 2020 that defined haploblocks as physical regions with >= 99.99 pident across fixed-size bins of 5-, 2.5- or 1-Mbp between pairwise comparisons of cultivars in the 10+ Wheat Genome Project. This method will subsequently be referred to as the Brinton's method. In these IBS-haploblocks, IBS-haplotypes can be either shared haplotypes, when pairs of assemblies share one identical-by-state sequence in this region, or unique haplotypes, when no other cultivar presents the same sequence in the haploblock. Additionally, Brinton and colleagues published crop-haplotypes.com, a website that provides an interactive graphic visualization of the shared haplotypes between the wheat genome assemblies. These assemblies were chromosome-level genome assemblies of nine wheat cultivars (ArinaLrFor, Jagger, Julius, Lancer, Landmark, Mace, Norin61, Stanley, SY-Mattis) and the Chinese Spring RefSev1.0 assembly alongside scaffold-level assemblies of five additional cultivars (Cadenza, Claire, Paragon, Robigus and Weebill).

Aim

The aim of this script is to use Brinton's method and resources to conduct both a chromosome-scale and small-scale analysis of the pairwise comparisons between cultivars. The small-scale analysis is an original method to this script and allows the analysis of target regions within chromosomes. The aim of the small-scale analysis is to map physical start and end coordinates of the predicted haploblock both in the reference and query genomes by considering the region coverage with alignments, in other words, what is the number of alignments the haplotype prediction is based on and if there is any discontinuity that could cause false positive predictions. Also, this script aims to identify the genes contained in mapped haploblock, both in the haplotype-carrying genomes and in the IWGSC Chinese Spring Ref v1.1. Brinton et al called haplotypes both from mummer and gene-based BLAST pairwise alignments and selected for matching positions the longer predictions. Both types of analysis are provided in this script, although the results from these methods do not always match.

Background

The SNP-haplotype Hap-5B-RDMa-h2, associated with increased root dry mass (RDM), is taken as an example. This haplotype was first found by GWAS upon 9 SNP marker alleles located in the wheat chromosome 5B and matched with the assemblies of the cultivars LongReach Lancer and Paragon by BLAST. SNP-RDMa-h2 BLAST position in Lancer's chromosome 5B (655760000-656600000 bp) was observed in crop-haplotypes.com in order to tell if the SNP-haploblock region matched with the region of any IBS-haploblock, discovered with Brinton's method, in this area. The following blocks were observed in the pairwise

comparison Lancer-Paragon under each bin size (bin size - start - end - length): 5-Mbp - 660.000.000 - 665.000.000 - 5.000.000 // 2.5-Mbp - 657.500.000 - 662.500.000 - 5.000.000 // 1-Mbp - 656.000.000 - 663.000.000 - 7.000.000 (start and end according to Lancer's coordinate system as Lancer acts as the reference in this comparison). Considering that blocks and their positions can vary between bin sizes, this observation was sufficient evidence to confirm the match between the SNP- and a potential IBS-haplotype. Despite of this, Brinton's method must be applied on this redefined region to understand if the prediction made on crop-haplotypes.com is reliable. Additionally, other research questions as the exact start and end position of the IBS-haploblock or its genes can be answered in this script.

Running the script

The script is designed so that you only have to edit only the upcoming parameters. You can run the script by lines (recommended for first time) or simply pressing 'Source' in the upper right corner of this window. The running time is expected to be between 5 and 7 minutes. Please, be patient and only look for solutions if error messages interrupt the pipeline. If errors do not stop coming up, you can contact the script editor at the e-mail adress jmonterotena@gmail.com (mailto:jmonterotena@gmail.com).

Defining parameters

```
chromosome <- "5B" # Copy the same format
reference_cultivar <- "Lancer" # Alignment coordinates will apply for this cultivar's chromos
ome. Can NEVER be a scaffold-level assembly (Cadenza, Claire, Paragon, Robigus, Weebil). Foll
ow the next naming guidelines: "arinalrfor", "jagger", "julius", "Lancer", "landmark", "mac
e", "norin61", "stanley", "sy_mattis", "chinese" (works with capital letters)
query_cultivar <- "Paragon" # This genome is aligned against the reference chromosome. Can be
a scaffold-level assembly. Follow the next naming guidelines: "cadenza", "claire", "paragon",
"robigus", "weebil" (works with capital letters)
zoom_start <- 655000000 # Start of the region highlighted in the small-scale analysis, in whi
ch haploblocks are sought for. ADVICE: use multiples of 5 so that the bins match with those o
n crop-haplotypes
zoom_end <- 665000000 # End of the region highlighted in the small-scale analysis, in which h
aploblocks are sought for. ADVICE: use multiples of 5 so that the bins match with those on cr
op-haplotypes
target_start <- 655700000 # If you have evidence of a haplotype within your zoom region that
 you want to compare with the IBS-haplotypes obtained in this script, write its start coordin
ate in the reference chromosome. If not, simply write NA. The target region must be within th
e zoom region
target_end <- 656600000 # If you have evidence of a haplotype within your zoom region that yo
u want to compare with the IBS-haplotypes obtained in this script, write its end coordinate i
n the reference chromosome. If not, simply write NA. The target region must be within the zoo
m region

selected_start <- 655760000 # You do not need to wrorry about this yet. Complete section one
 and then come back to define the new coordinates of your haploblocks. Genes will be extracte
d from this position. If you have no interest in redefining your region, simply write 'target
_start' or 'zoom_start', to keep with the previous coordinates
selected_end <- 662740000 # You do not need to wrorry about this yet. Complete section one an
d then come back to define the new coordinates of your haploblocks. Genes will be extracted u
ntil this position. If you have no interest in redefining your region, simply write 'target_e
nd' or 'zoom_end', to keep with the previous coordinates
```

Running functions

Source the script ‘functions.hbpa.r’ to read the packages and functions required for this script.

REQUIRED PACKAGES

1. CALLING HAPLOTYPES FROM RAW DATA CONTAINING >= 20-KBP-LONG MUMMER PAIRWISE ALIGNMENTS BETWEEN LANCER AND PARAGON

Requirements:

- Raw delta files in ‘all_20_kb_filtered_delta_CHROMOSOME_tables.rds’; in this case the chromosome is 5B (downloadable from https://opendata.earlham.ac.uk/wheat/under_license/toronto/Brinton_etal_2020-05-20-Haplotypes-for-wheat-breeding/nucmer/ (https://opendata.earlham.ac.uk/wheat/under_license/toronto/Brinton_etal_2020-05-20-Haplotypes-for-wheat-breeding/nucmer/)). Brinton et al filtered out alignments under 20000Kbp of length in order to get rid of non-syntenic retrotransposons
- Script ‘functions.hbpa.r’ (downloadable from https://github.com/MonteroJLU/Haplotype-based-Pangenome-Wheat-Analysis.git (https://github.com/MonteroJLU/Haplotype-based-Pangenome-Wheat-Analysis.git))
- Text file ‘table_cultivar_chr_length.txt’ (downloadable from https://github.com/MonteroJLU/Haplotype-based-Pangenome-Wheat-Analysis.git (https://github.com/MonteroJLU/Haplotype-based-Pangenome-Wheat-Analysis.git))

1.1. CHROMOSOME-SCALE ANALYSIS

1.1.1. Download and save the required documents in the working directory

1.1.2. Run the script ‘functions_hbpa.r’

1.1.3. Read the rds file into a data frame:

```
aln_library <- readRDS(file = paste0("all_20_kb_filtered_delta_", chromosome,"_tables.rds"))
```

1.1.4. Create a subset of the data frame containing only the alignments from the pairwise comparison Lancer-Paragon, where Lancer acts as the the reference genome:

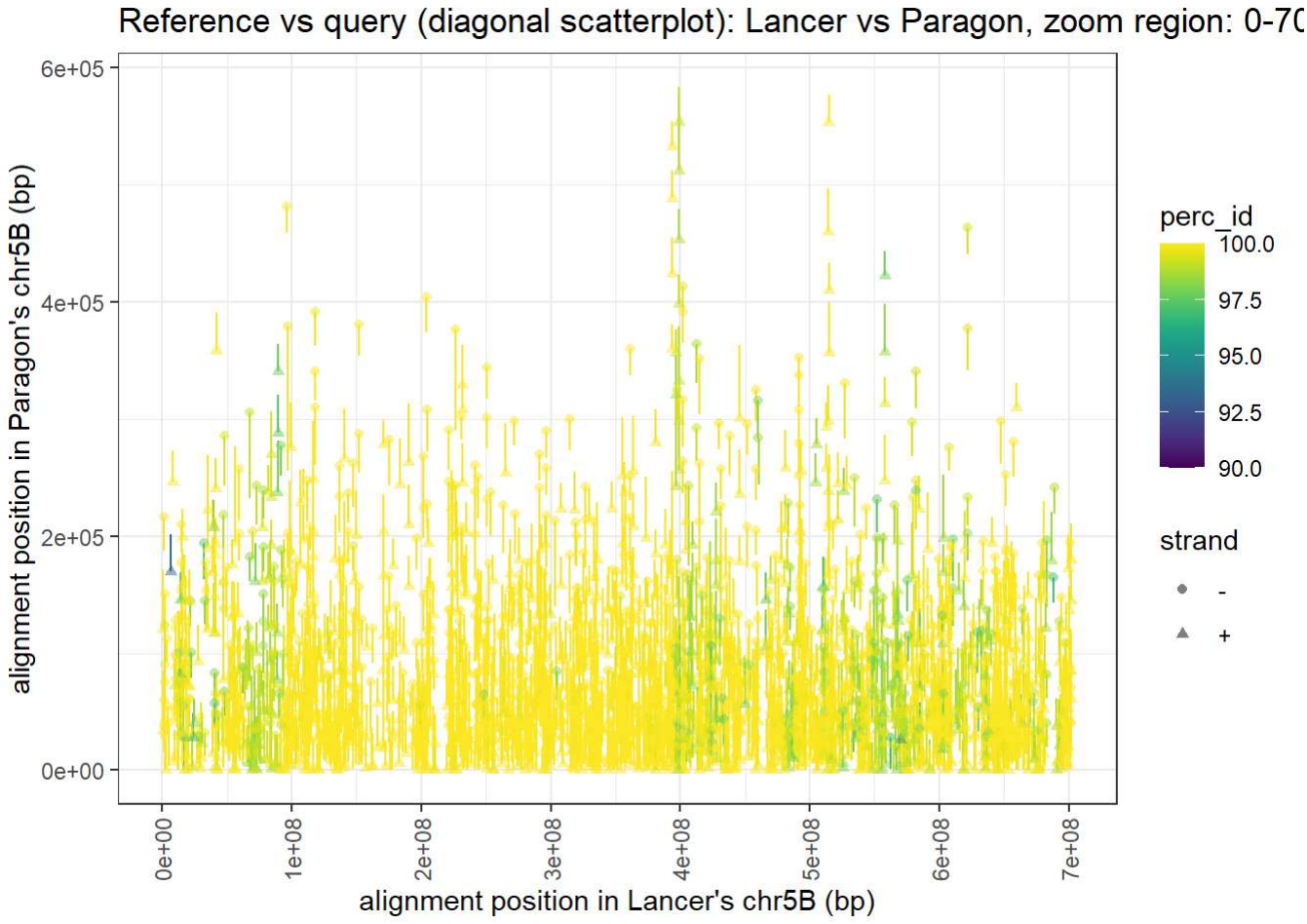
```
aln_subset <- aln_library[grepl(paste0("^", tolower(reference_cultivar), sep = ""), aln_libra
ry$comparison) & grepl(tolower(query_cultivar), aln_library$comparison),]
```

Description of the column headers: rs: reference start, re: reference end, qs: query start, qe: query end, error: number of unmatches, qid: query identification, rid: reference identification, strand: forward or reverse strand, r_length: length of the alignment in the reference, perc_id: percentage of identity, perc_id_factor: factor that

summarises perc_id, r_mid: midpoint the alignment in reference ((r_end-r_start)/2), q_mid: query midpoint, comparison: cultivars compared, chrom: chromosome.

1.1.5. Scatter-plot the alignment midpoints (X: r_mid, Y: q_mid)

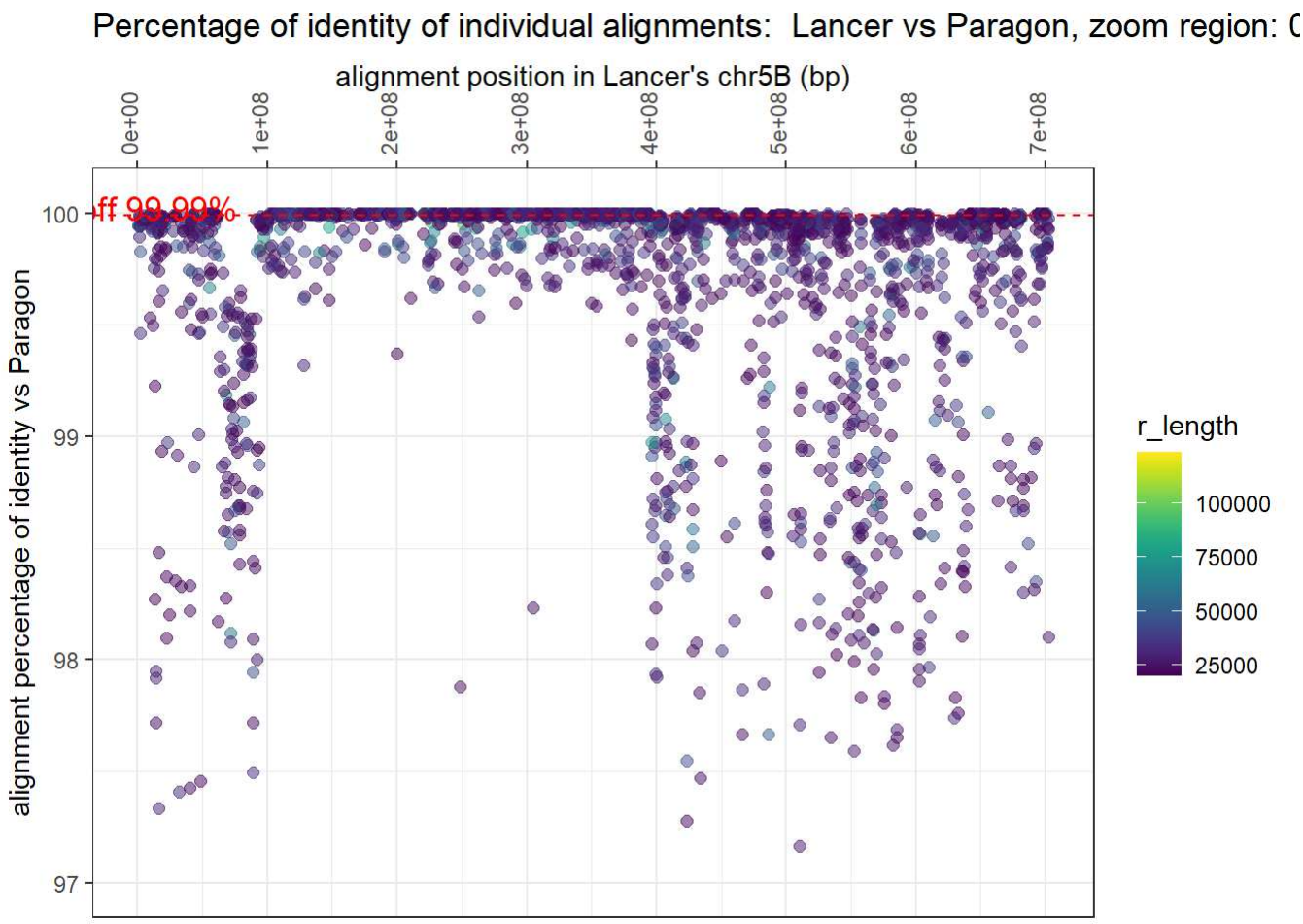
```
plot_diagonal_scatterplot(aln_subset, cap_lower = 90.00, cap_upper = 100, reference_name = re
ference_cultivar, query_name = query_cultivar , x_label_gap = 100000000)
```



Notice unexpected vertical lines in the scatter-plot due to the use of a scaffold-level assembly as query (CLA-SLA comparison). However, this function works well for CLA-CLA comparisons.

1.1.6. Dot-plot the alignments to show percentage of identity and alignment length (X: r_mid, Y: perc_id)

```
plot_aln_pid_and_length(aln_subset, reference_name = reference_cultivar, query_name = query_c
ultivar , x_label_gap = 100000000, dot_size = 2)
```



1.1.7. Check for alignment properties in the data frame (percentage of alignment coverage of reference chromosome, average aln length and av aln number)

```
table_chrlength_v_cultivar <- read.table(file = "table_cultivar_chr_length.txt", sep = "\t",
 header = TRUE)
chr_length <- table_chrlength_v_cultivar$sequence_length[grepl(tolower(reference_cultivar), t
able_chrlength_v_cultivar$cultivar_name) & grepl(paste0("chr", unique(aln_subset$chrom)), tab
le_chrlength_v_cultivar$sequence_name)]
print(paste0("chr", unique(aln_subset$chrom), " is ", chr_length, " bp-long in ", reference_c
ultivar))
```

```
## [1] "chr5B is 702438406 bp-long in Lancer"
```

```
print(paste0(round(mean(aln_subset$r_length), 0), " is the average alignment length in ", ref
erence_cultivar, "-", query_cultivar, paste0(" chr", unique(aln_subset$chrom)), " comparison"
))
```

```
## [1] "30858 is the average alignment length in Lancer-Paragon chr5B comparison"
```

```
print(paste0(nrow(aln_subset), " is the number of alignments in ", reference_cultivar, "-", q
uery_cultivar, paste0(" chr", unique(aln_subset$chrom)), " comparison"))
```

```
## [1] "2650 is the number of alignments in Lancer-Paragon chr5B comparison"
```

```
print(paste0(round((sum(aln_subset$r_length)/chr_length*100), 0), "% is the alignment coverag
e in ", reference_cultivar, "-", query_cultivar, paste0(" chr", unique(aln_subset$chrom)), "
 comparison"))
```

```
## [1] "12% is the alignment coverage in Lancer-Paragon chr5B comparison"
```

```
coverage <- COV(aln_library, chr_length = table_chrlength_v_cultivar)
print(coverage)
```

```
## $`Average % coverage with CHROMOSOME-LEVEL ASSEMBLIES as query`
## [1] 47.74529
##
## $`(By cultivar)`
## arinalrfor     jagger     julius     lancer   landmark       mace    norin61
##   28.44218   50.59484   51.33529   55.81078   50.99109   55.28362   53.99118
##    stanley  sy_mattis    chinese
##   51.89199   28.44447   50.66749
##
## $`Average % coverage with SCAFFOLD-LEVEL ASSEMBLIES as query`
## [1] 9.390724
##
## $`(By cultivar)`
## arinalrfor     jagger     julius     lancer   landmark       mace    norin61
##   4.216677  10.113839  10.478023  12.165450   9.566494  11.940812  11.252819
##    stanley  sy_mattis    chinese
##  10.103706   4.302350   9.767073
```

```
length <- LENGTH(aln_library)
print(length)
```

```
## $`Average aln length with CHROMOSOME-LEVEL ASSEMBLIES as query`
## [1] 45635.03
##
## $`(By cultivar)`
## arinalrfor     jagger     julius     lancer   landmark       mace    norin61
##   50573.98   42245.55   43894.80   46985.47   42621.59   46726.19   45787.59
##    stanley  sy_mattis    chinese
##   44081.69   50841.05   42592.41
##
## $`Average aln length with SCAFFOLD-LEVEL ASSEMBLIES as query`
## [1] 30425.71
##
## $`(By cultivar)`
## arinalrfor     jagger     julius     lancer   landmark       mace    norin61
##   31092.67   29791.85   30193.23   31045.12   29593.04   30915.88   30557.44
##    stanley  sy_mattis    chinese
##   29983.10   31462.44   29622.29
```

```
number <- NUMBER(aln_library)
print(number)
```

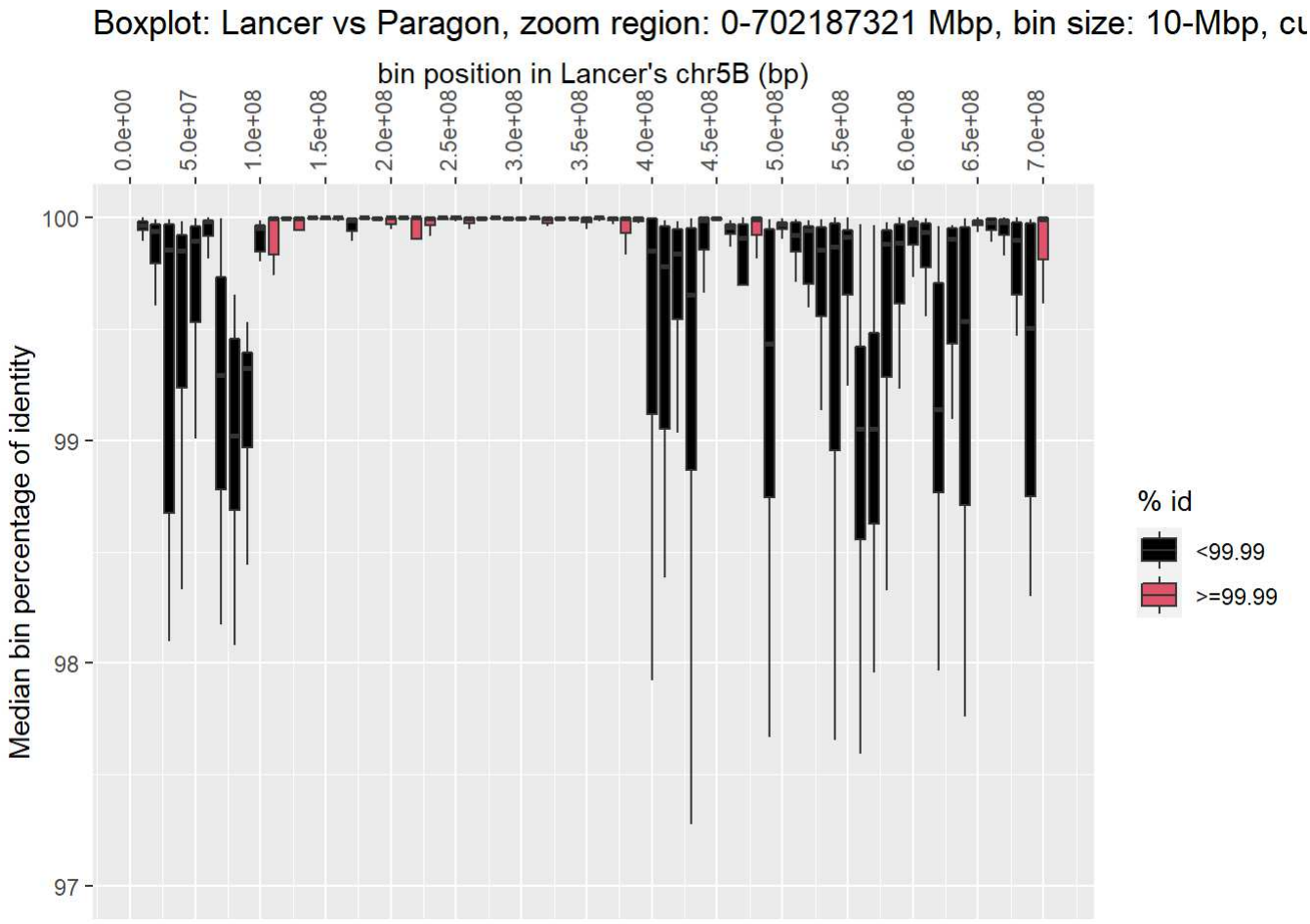
```
## $`Average aln number with CHROMOSOME-LEVEL ASSEMBLIES as query`
## [1] 7253.244
##
## $`(By cultivar)`
## arinalrfor     jagger     julius     lancer   landmark       mace    norin61
##   2703.778   8426.556   8480.333   8343.778   8500.111   8172.778   8399.667
##    stanley  sy_mattis    chinese
##   8404.222   2617.667   8483.556
##
## $`Average aln number with SCAFFOLD-LEVEL ASSEMBLIES as query`
## [1] 2129.46
##
## $`(By cultivar)`
## arinalrfor     jagger     julius     lancer   landmark       mace    norin61
##      652.0     2388.6     2516.4     2752.6     2296.8     2668.0     2623.2
##    stanley  sy_mattis    chinese
##     2405.8      639.8     2351.4
```

```
bin_size <- c(5000000, 2500000, 1000000)
names(bin_size) <- c("bin size: 5-Mbp", "bin size: 2.5-Mbp", "bin size: 1-Mbp")
for (i in 1:3){
  print("average expected number of alignments per bin across the chromosome")
  print(round(nrow(aln_subset)/(chr_length/bin_size[i]), 1))
}
```

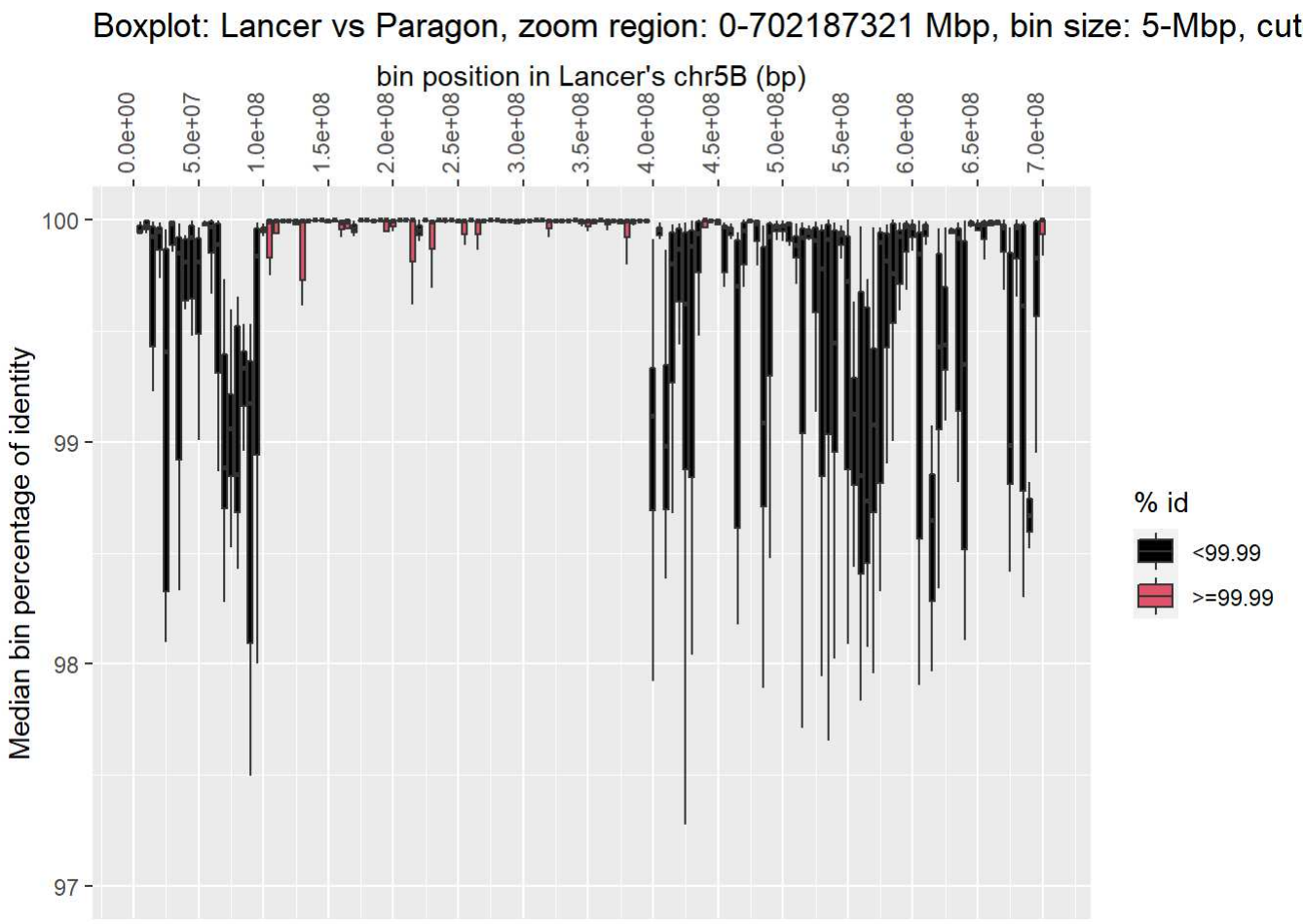
```
## [1] "average expected number of alignments per bin across the chromosome"
## bin size: 5-Mbp
##            18.9
## [1] "average expected number of alignments per bin across the chromosome"
## bin size: 2.5-Mbp
##               9.4
## [1] "average expected number of alignments per bin across the chromosome"
## bin size: 1-Mbp
##             3.8
```

1.1.8. Boxplot the alignments (X: bin, Y: perc_id_median)

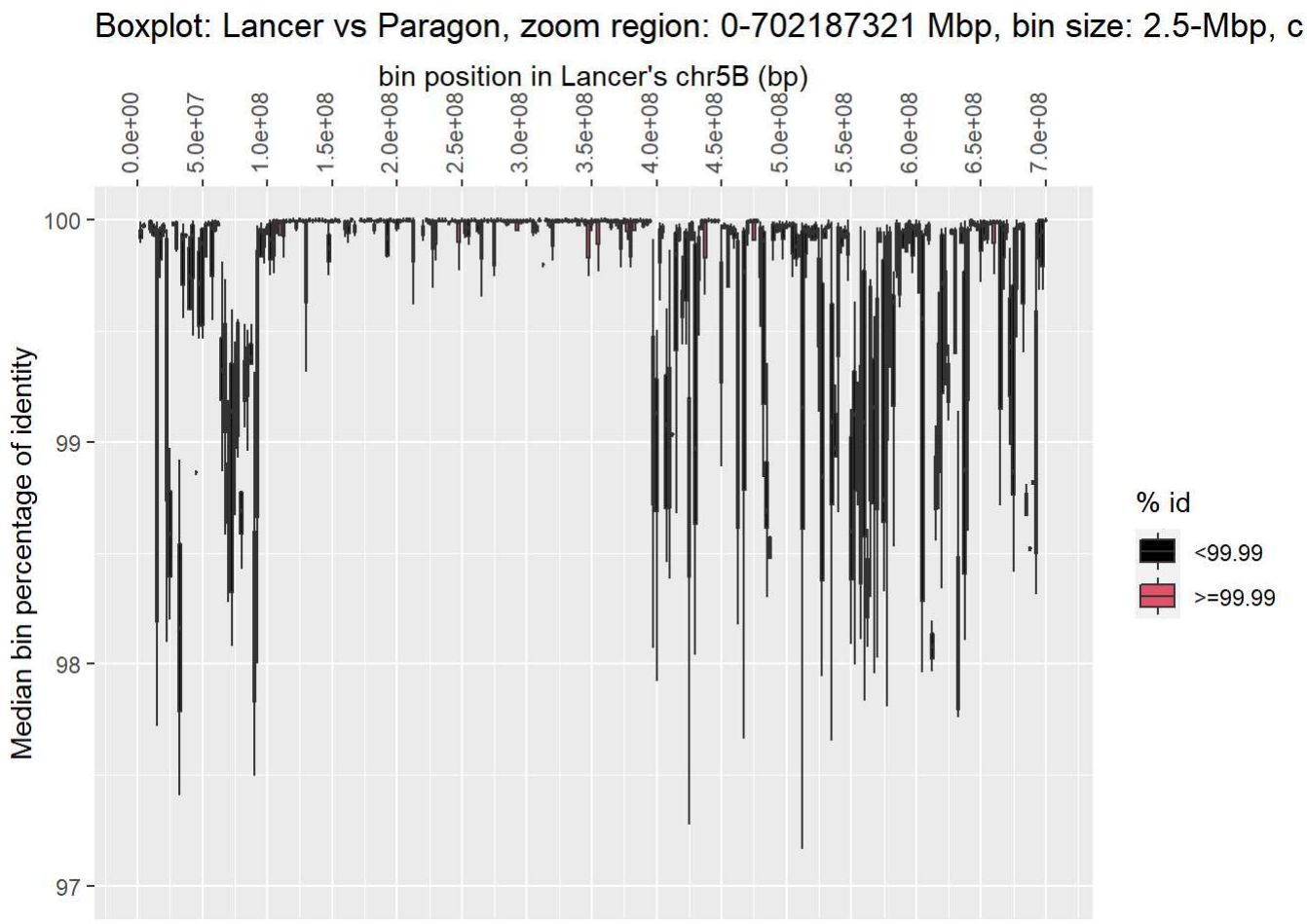
```
plot_boxplots_bin_median(aln_subset, bin_size = 10000000, bin_start = 0, bin_end = max(aln_su
bset$re), cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar ,
x_label_gap = 50000000, show_outliers = FALSE)
```



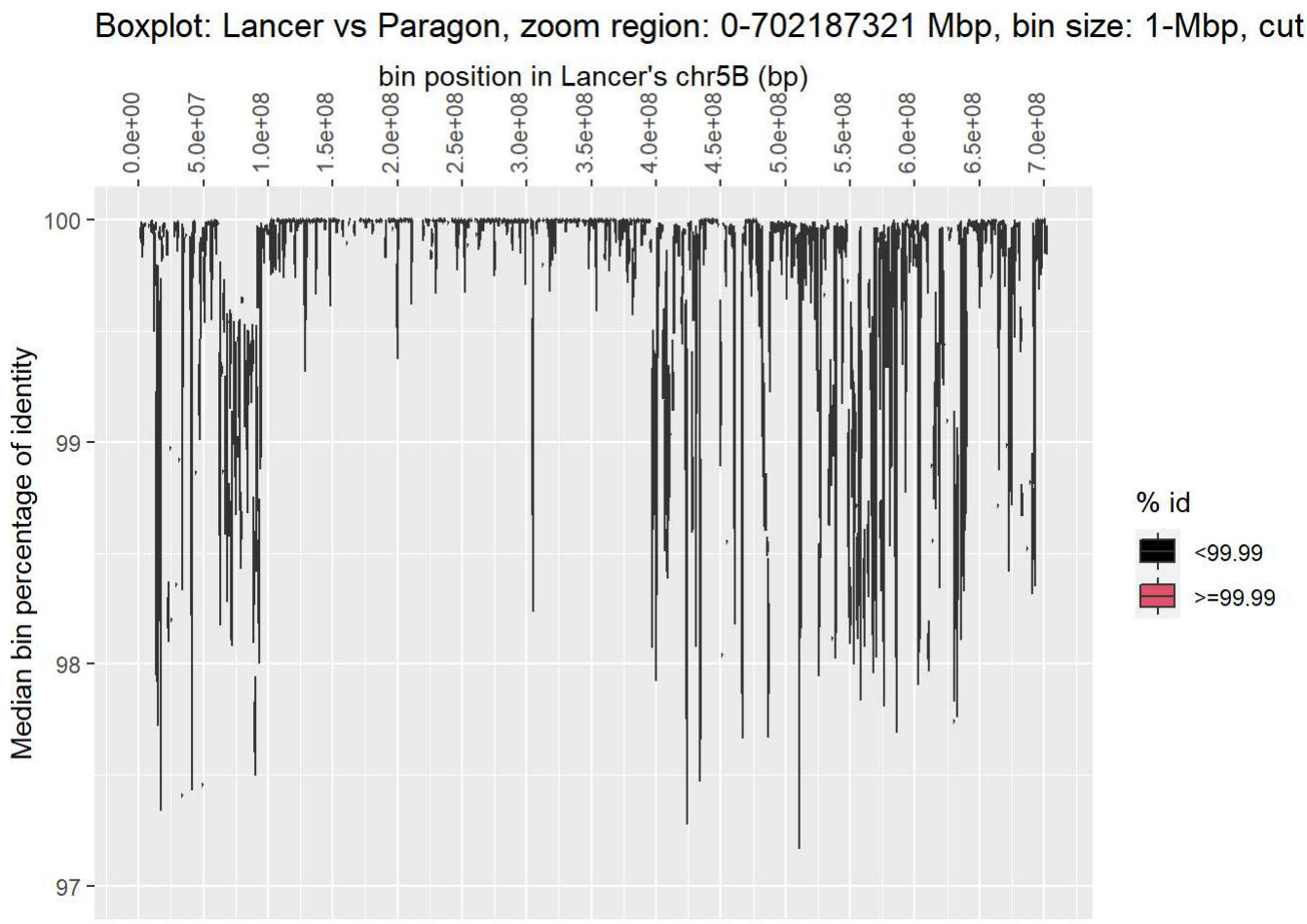

```
plot_boxplots_bin_median(aln_subset, bin_size = 5000000, bin_start = 0, bin_end = max(aln_sub
set$re), cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar ,
 x_label_gap = 50000000, show_outliers = FALSE)
```




```
plot_boxplots_bin_median(aln_subset, bin_size = 2500000, bin_start = 0, bin_end = max(aln_sub
set$re), cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar ,
 x_label_gap = 50000000, show_outliers = FALSE)
```

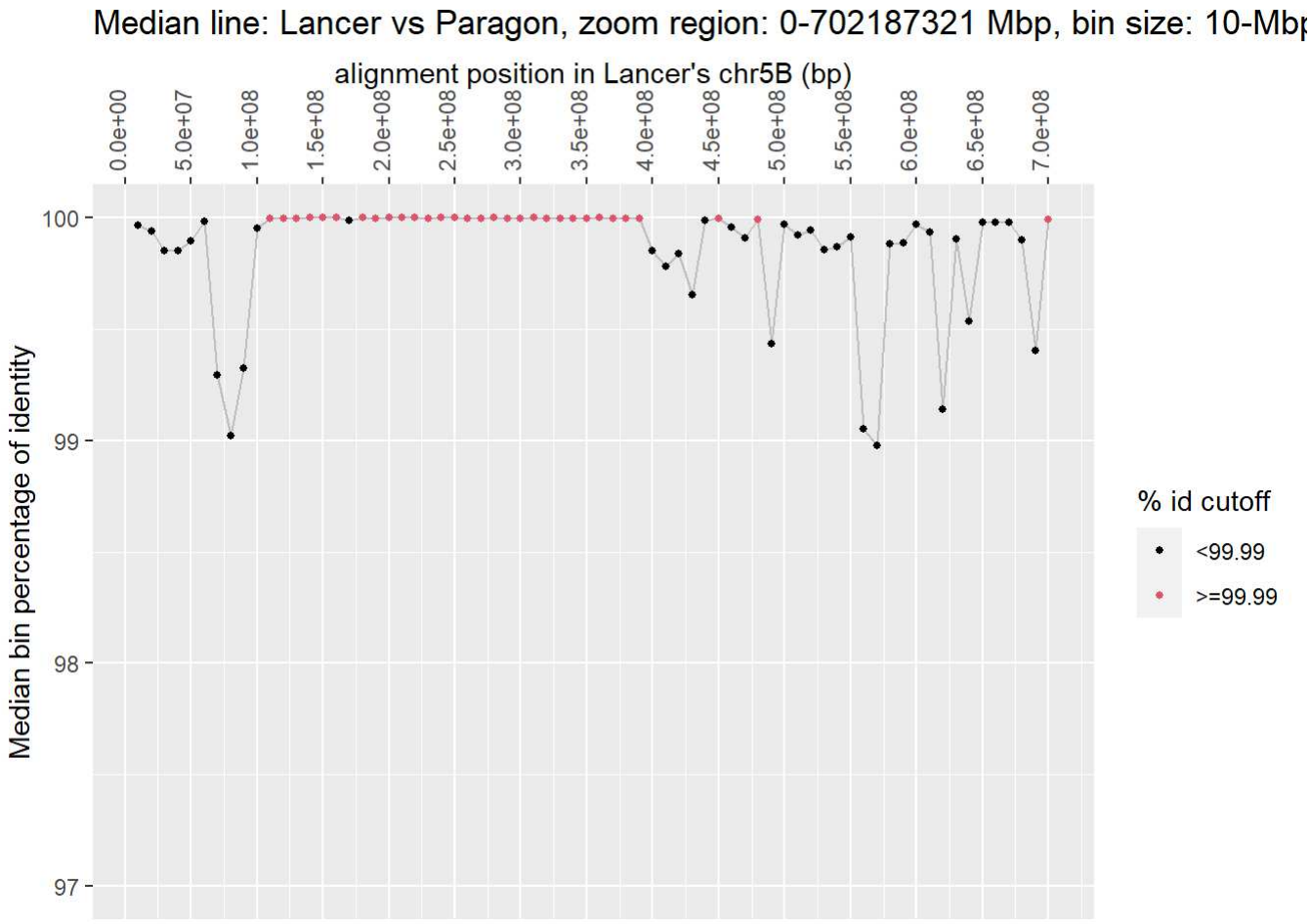



```
plot_boxplots_bin_median(aln_subset, bin_size = 1000000, bin_start = 0, bin_end = max(aln_sub
set$re), cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar ,
 x_label_gap = 50000000, show_outliers = FALSE)
```

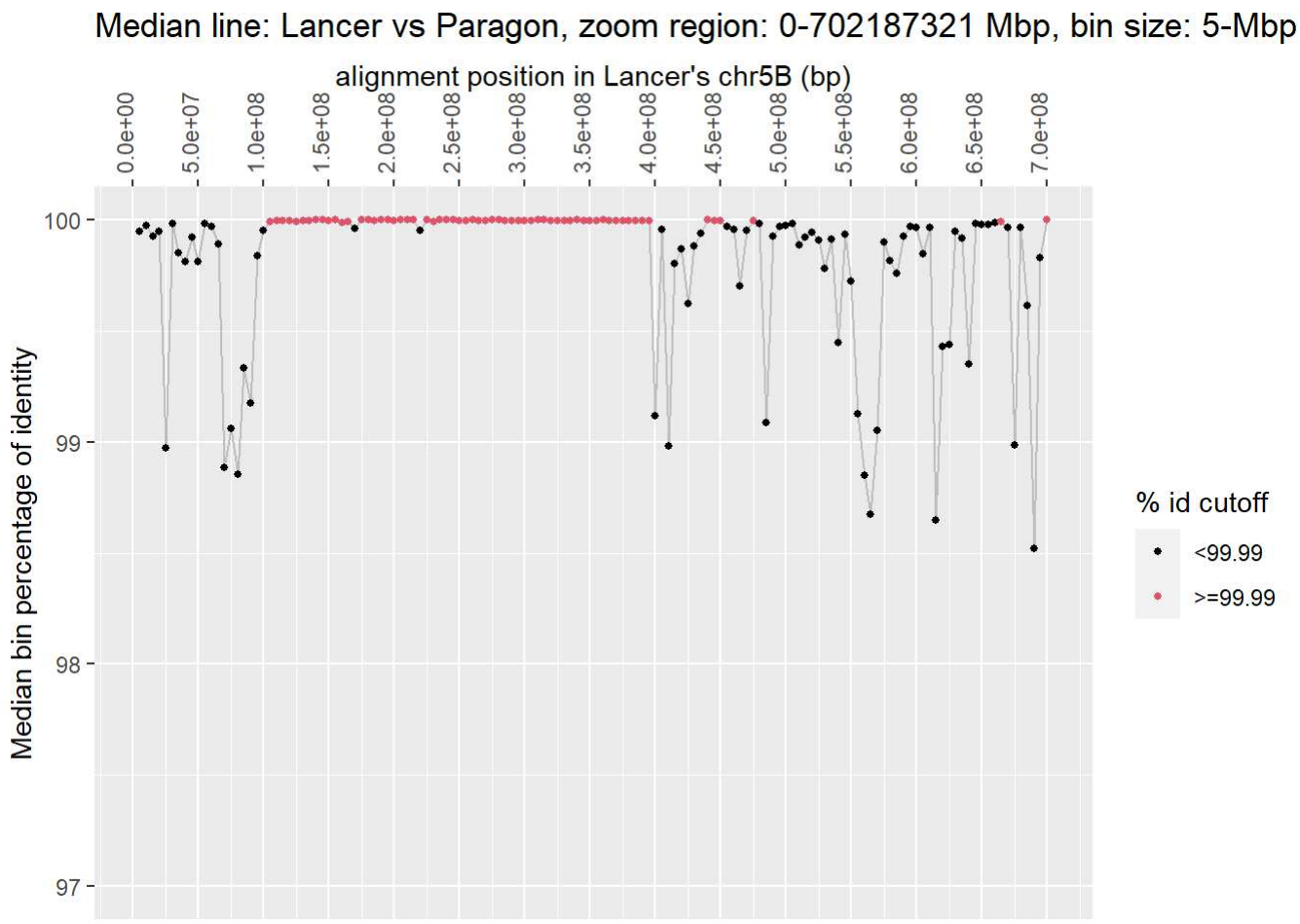


1.1.9. Plot the median percentage of identity across the chromosome or median line (X: r_mid, Y: perc_id_median)

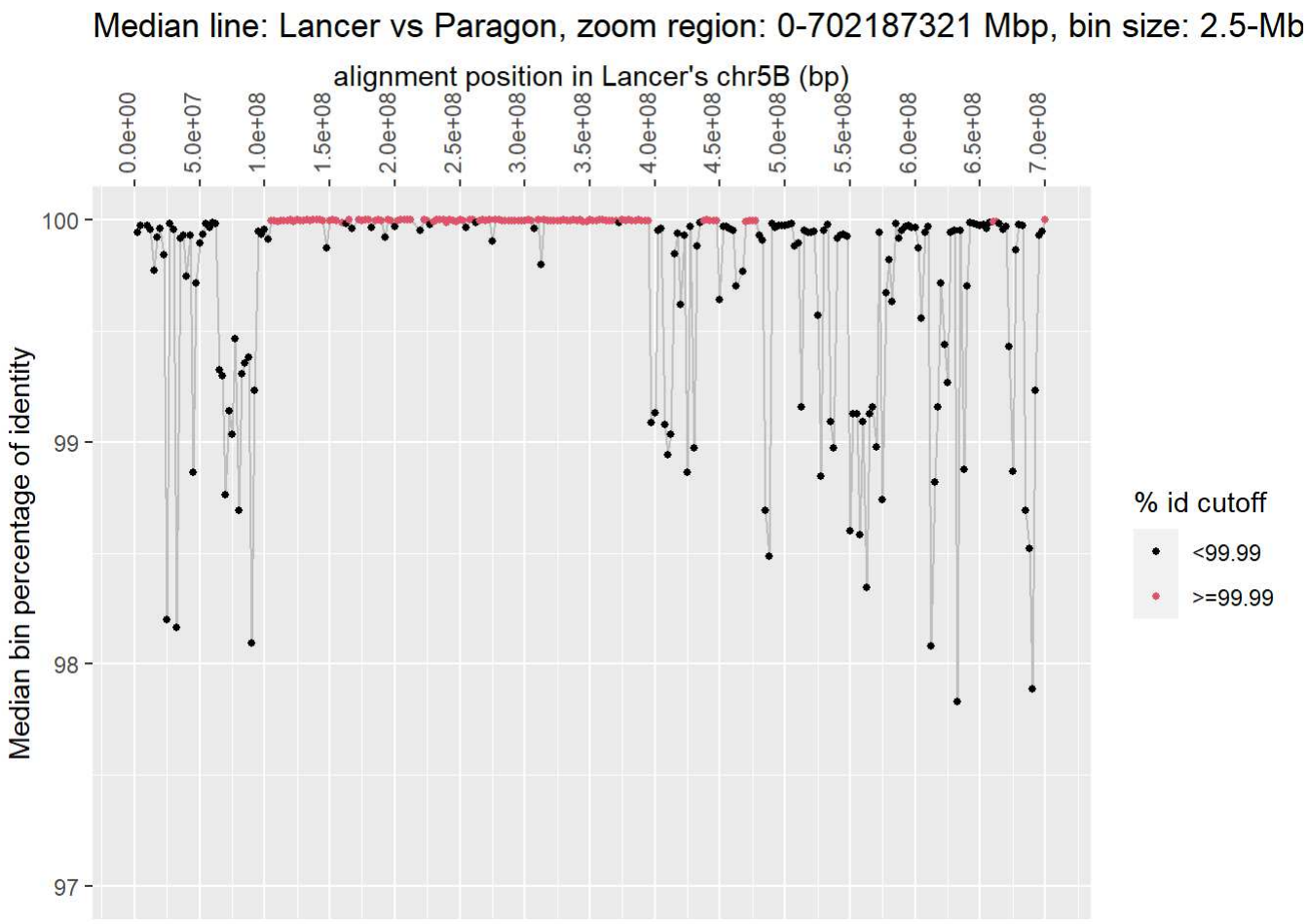
```
medians_aln_subset_10Mbp <- plot_line_bin_median(aln_subset, bin_size = 10000000, bin_start =
0, bin_end = max(aln_subset$re), cut_off = 99.99, reference_name = reference_cultivar, query_
name = query_cultivar , x_label_gap = 50000000)
medians_aln_subset_10Mbp
```



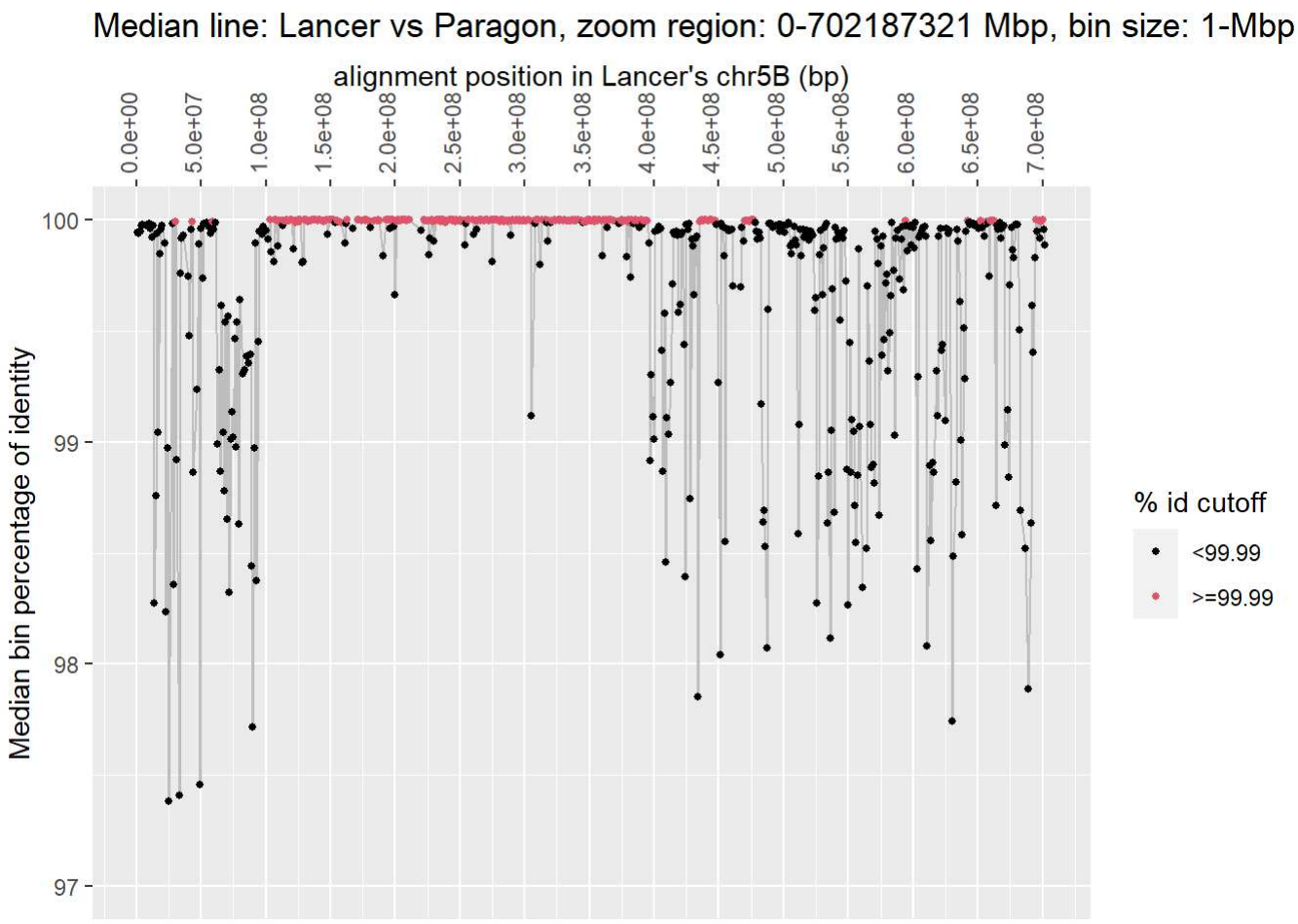

```
medians_aln_subset_5Mbp <- plot_line_bin_median(aln_subset, bin_size = 5000000, bin_start = 0
, bin_end = max(aln_subset$re), cut_off = 99.99, reference_name = reference_cultivar, query_n
ame = query_cultivar , x_label_gap = 50000000)
medians_aln_subset_5Mbp
```




```
medians_aln_subset_2.5Mbp <- plot_line_bin_median(aln_subset, bin_size = 2500000, bin_start =
0, bin_end = max(aln_subset$re), cut_off = 99.99, reference_name = reference_cultivar, query_
name = query_cultivar , x_label_gap = 50000000)
medians_aln_subset_2.5Mbp
```




```
medians_aln_subset_1Mbp <- plot_line_bin_median(aln_subset, bin_size = 1000000, bin_start = 0
, bin_end = max(aln_subset$re), cut_off = 99.99, reference_name = reference_cultivar, query_n
ame = query_cultivar , x_label_gap = 50000000)
medians_aln_subset_1Mbp
```



1.1.10. Print information about the haploblock predictions in the pairwise comparison across the reference chromosome

```
medians_aln_subset_10Mbp_bin_info <- assign_blocks_mummer(median_cutoffs = medians_aln_subset
_10Mbp[["data"]], original_file = aln_subset)
medians_aln_subset_5Mbp_bin_info <- assign_blocks_mummer(median_cutoffs = medians_aln_subset_
5Mbp[["data"]], original_file = aln_subset)
medians_aln_subset_2.5Mbp_bin_info <- assign_blocks_mummer(median_cutoffs = medians_aln_subse
t_2.5Mbp[["data"]], original_file = aln_subset)
medians_aln_subset_1Mbp_bin_info <- assign_blocks_mummer(median_cutoffs = medians_aln_subset_
1Mbp[["data"]], original_file = aln_subset)
medians_aln_subset_10Mbp_block_summary <- block_summary(medians_aln_subset_10Mbp_bin_info, bi
n_size = 10000000, reference_name = reference_cultivar, query_name = query_cultivar )
medians_aln_subset_5Mbp_block_summary <- block_summary(medians_aln_subset_5Mbp_bin_info, bin_
size = 5000000, reference_name = reference_cultivar, query_name = query_cultivar )
medians_aln_subset_2.5Mbp_block_summary  <-  block_summary(medians_aln_subset_2.5Mbp_bin_info,
 bin_size = 2500000, reference_name = reference_cultivar, query_name = query_cultivar )
medians_aln_subset_1Mbp_block_summary <- block_summary(medians_aln_subset_1Mbp_bin_info, bin_
size = 1000000, reference_name = reference_cultivar, query_name = query_cultivar )
print(medians_aln_subset_10Mbp_block_summary)
```

```
##   bin_size      comparison block_no block_start block_end
## 1   10-Mbp Lancer->Paragon        1     1.0e+08   3.9e+08
## 2   10-Mbp Lancer->Paragon        2     4.4e+08   4.8e+08
## 3   10-Mbp Lancer->Paragon        3     6.9e+08   7.0e+08
```

```
print(medians_aln_subset_5Mbp_block_summary)
```

```
##   bin_size      comparison block_no block_start block_end
## 1    5-Mbp Lancer->Paragon        1    1.00e+08  3.95e+08
## 2    5-Mbp Lancer->Paragon        2    4.35e+08  4.50e+08
## 3    5-Mbp Lancer->Paragon        3    4.70e+08  4.75e+08
## 4    5-Mbp Lancer->Paragon        4    6.60e+08  6.65e+08
## 5    5-Mbp Lancer->Paragon        5    6.95e+08  7.00e+08
```

```
print(medians_aln_subset_2.5Mbp_block_summary)
```

```
##   bin_size      comparison block_no block_start block_end
## 1  2.5-Mbp Lancer->Paragon        1   102500000 395000000
## 2  2.5-Mbp Lancer->Paragon        2   435000000 447500000
## 3  2.5-Mbp Lancer->Paragon        3   467500000 477500000
## 4  2.5-Mbp Lancer->Paragon        4   657500000 662500000
## 5  2.5-Mbp Lancer->Paragon        5   697500000 700000000
```

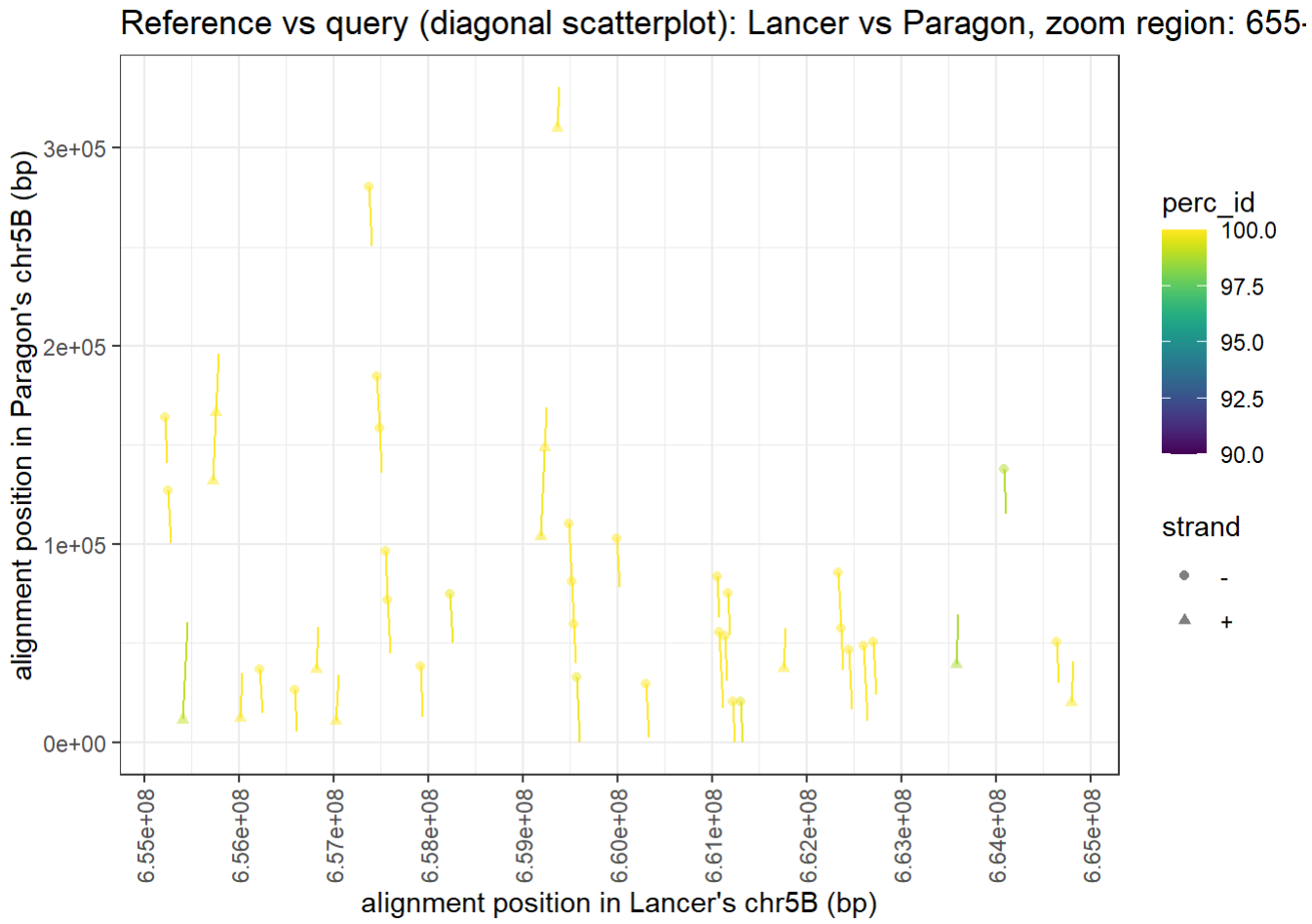
```
print(medians_aln_subset_1Mbp_block_summary)
```

```
##    bin_size      comparison block_no block_start block_end
## 1     1-Mbp Lancer->Paragon        1    2.90e+07  3.00e+07
## 2     1-Mbp Lancer->Paragon        2    4.20e+07  4.30e+07
## 3     1-Mbp Lancer->Paragon        3    5.80e+07  5.90e+07
## 4     1-Mbp Lancer->Paragon        4    1.02e+08  3.95e+08
## 5     1-Mbp Lancer->Paragon        5    4.34e+08  4.47e+08
## 6     1-Mbp Lancer->Paragon        6    4.69e+08  4.77e+08
## 7     1-Mbp Lancer->Paragon        7    5.93e+08  5.94e+08
## 8     1-Mbp Lancer->Paragon        8    6.41e+08  6.42e+08
## 9     1-Mbp Lancer->Paragon        9    6.51e+08  6.53e+08
## 10    1-Mbp Lancer->Paragon       10    6.56e+08  6.63e+08
## 11    1-Mbp Lancer->Paragon       11    6.94e+08  7.00e+08
```

1.2. SMALL-SCALE ANALYSIS

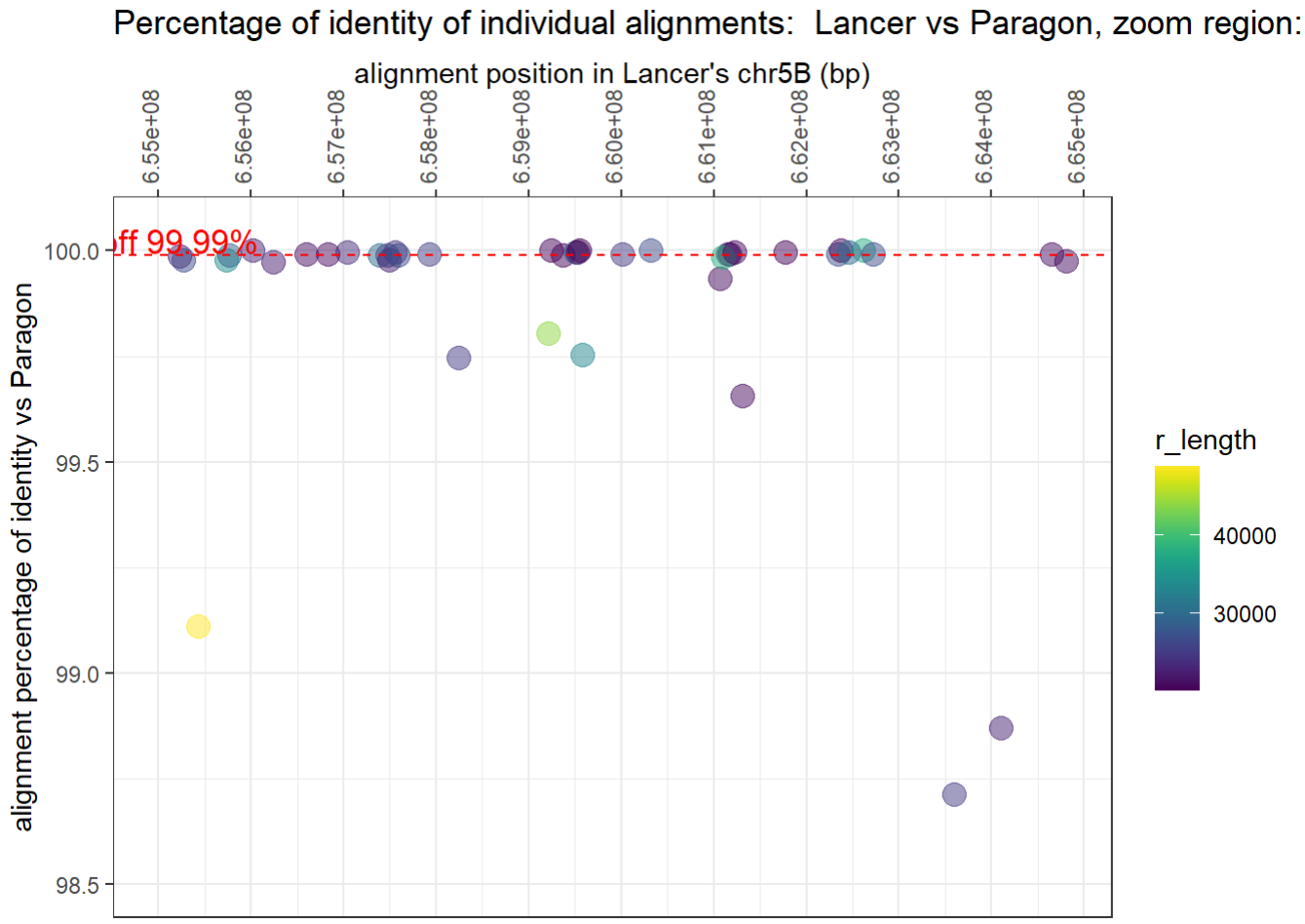
1.2.1. Scatter-plot the alignment midpoints across the zoom region (X: r_mid, Y: q_mid)

```
plot_diagonal_scatterplot(aln_subset, xmin = zoom_start, xmax = zoom_end, cap_lower = 90.00,
 cap_upper = 100, reference_name = reference_cultivar, query_name = query_cultivar , x_label_
gap = 1000000)
```



1.2.2. Dot-plot the alignments to show percentage of identity and alignment length in the zoom region (X: r_mid, Y: perc_id)

```
graph <- plot_aln_pid_and_length(data = aln_subset, xmin = zoom_start, xmax = zoom_end, ymin = 98.5, reference_name = reference_cultivar, query_name = query_cultivar , x_label_gap = 1000000, dot_size = 4)
graph
```



1.2.3. Check for alignment properties in the zoom region

```
aln_target <- aln_subset[(aln_subset$r_mid >= target_start) & (aln_subset$r_mid <= target_en
d),]
print(paste0(round(mean(aln_target$r_length), 0), " is the average alignment length for the t
arget region between ", target_start/1e06, " and ", target_end/1e06, " Mbp in ", reference_cu
ltivar, "-", query_cultivar, paste0(" chr", unique(aln_target$chrom)), " comparison"))
```

```
## [1] "27206 is the average alignment length for the target region between 655.7 and 656.6 M
bp in Lancer-Paragon chr5B comparison"
```

```
print(paste0(nrow(aln_target), " is the number of alignments for the target region between ",
target_start/1e06, " and ", target_end/1e06, " Mbp in ", reference_cultivar, "-", query_culti
var, paste0(" chr", unique(aln_target$chrom)), " comparison"))
```

```
## [1] "4 is the number of alignments for the target region between 655.7 and 656.6 Mbp in La
ncer-Paragon chr5B comparison"
```

```
print(paste0(round((sum(aln_target$r_length)/(target_end-target_start)*100), 0), "% is the al
ignment coverage for the target region between ", target_start/1e06, " and ", target_end/1e06
, " Mbp in ", reference_cultivar, "-", query_cultivar, paste0(" chr", unique(aln_target$chro
m)), " comparison") )
```

```
## [1] "12% is the alignment coverage for the target region between 655.7 and 656.6 Mbp in La
ncer-Paragon chr5B comparison"
```

```
aln_zoom <- aln_subset[(aln_subset$r_mid >= zoom_start) & (aln_subset$r_mid <= zoom_end),]
print(paste0(round(mean(aln_zoom$r_length), 0), " is the average alignment length for the zoo
m region between ", zoom_start/1e06, " and ", zoom_end/1e06, " Mbp in ", reference_cultivar,
"-", query_cultivar, paste0(" chr", unique(aln_subset$chrom)), " comparison"))
```

```
## [1] "25963 is the average alignment length for the zoom region between 655 and 665 Mbp in
Lancer-Paragon chr5B comparison"
```

```
print(paste0(nrow(aln_zoom), " is the number of alignments for the zoom region between ", zoo
m_start/1e06, " and ", zoom_end/1e06, " Mbp in ", reference_cultivar, "-", query_cultivar, pa
ste0(" chr", unique(aln_subset$chrom)), " comparison"))
```

```
## [1] "42 is the number of alignments for the zoom region between 655 and 665 Mbp in Lancer-
Paragon chr5B comparison"
```

```
print(paste0(round((sum(aln_zoom$r_length)/(zoom_end-zoom_start)*100), 0), "% is the alignmen
t coverage for the zoom region between ", zoom_start/1e06, " and ", zoom_end/1e06, " Mbp in "
, reference_cultivar, "-", query_cultivar, paste0(" chr", unique(aln_subset$chrom)), " compar
ison"))
```

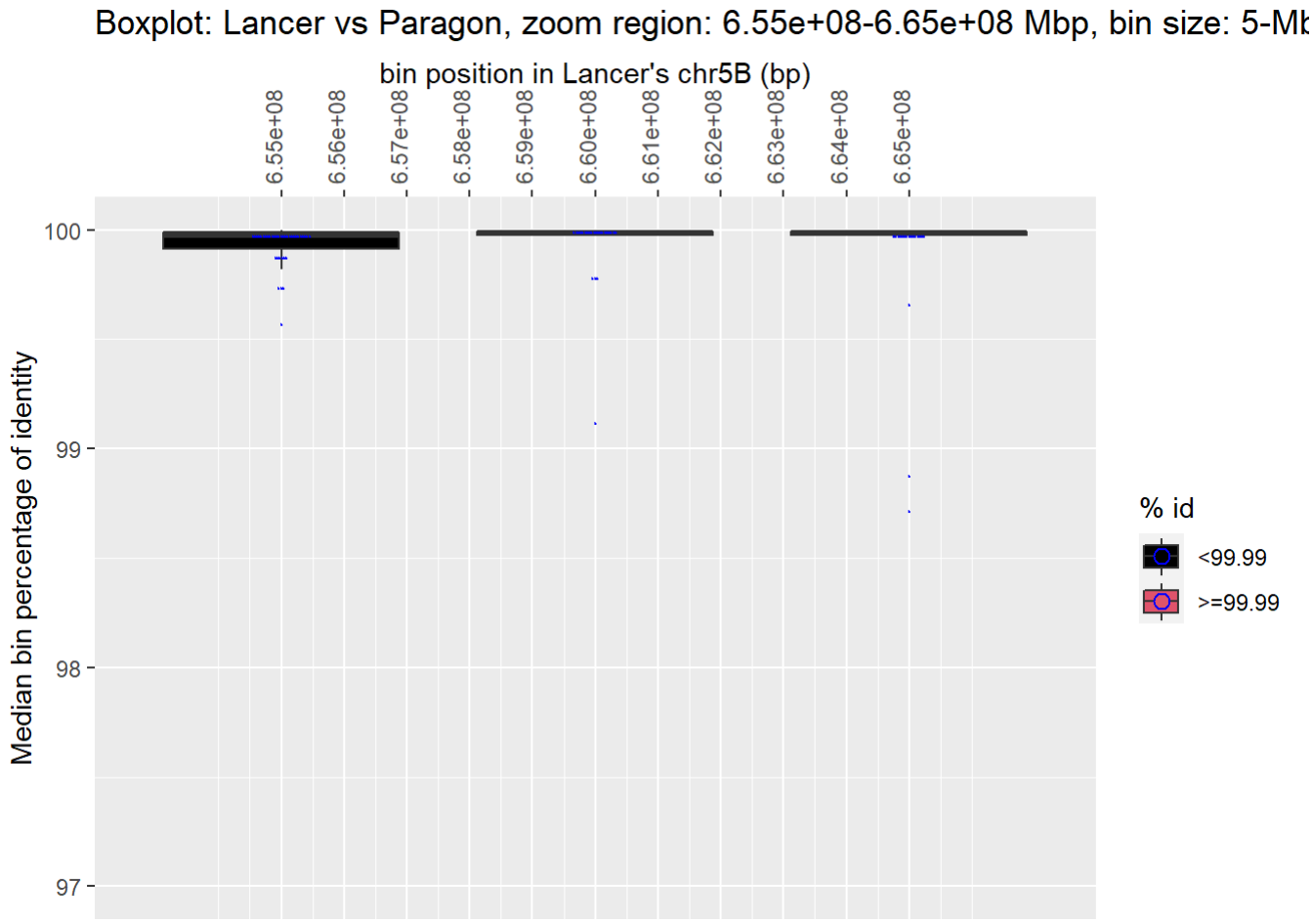
```
## [1] "11% is the alignment coverage for the zoom region between 655 and 665 Mbp in Lancer-P
aragon chr5B comparison"
```

```
bin_size <- c(5000000, 2500000, 1000000)
names(bin_size) <- c("bin size: 5-Mbp", "bin size: 2.5-Mbp", "bin size: 1-Mbp")
for (i in 1:3){
  print("average expected number of alignments per bin across zoom region")
  print(nrow(aln_zoom)/((zoom_end-zoom_start)/bin_size[i]))
}
```

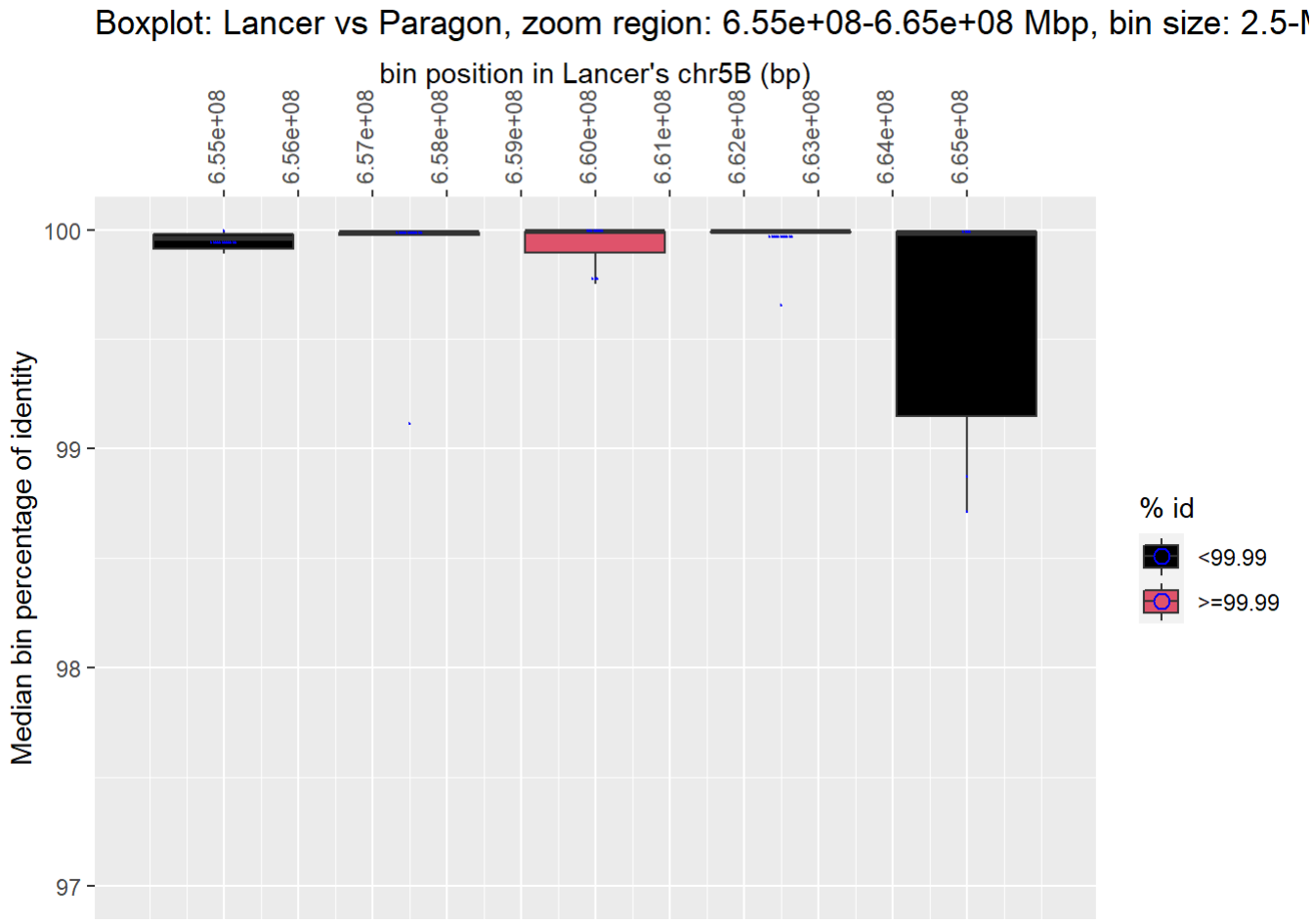
```
## [1] "average expected number of alignments per bin across zoom region"
## bin size: 5-Mbp
##              21
## [1] "average expected number of alignments per bin across zoom region"
## bin size: 2.5-Mbp
##              10.5
## [1] "average expected number of alignments per bin across zoom region"
## bin size: 1-Mbp
##             4.2
```

1.2.4. Boxplot the bin median percentage of identity in the zoom region to check for outliers

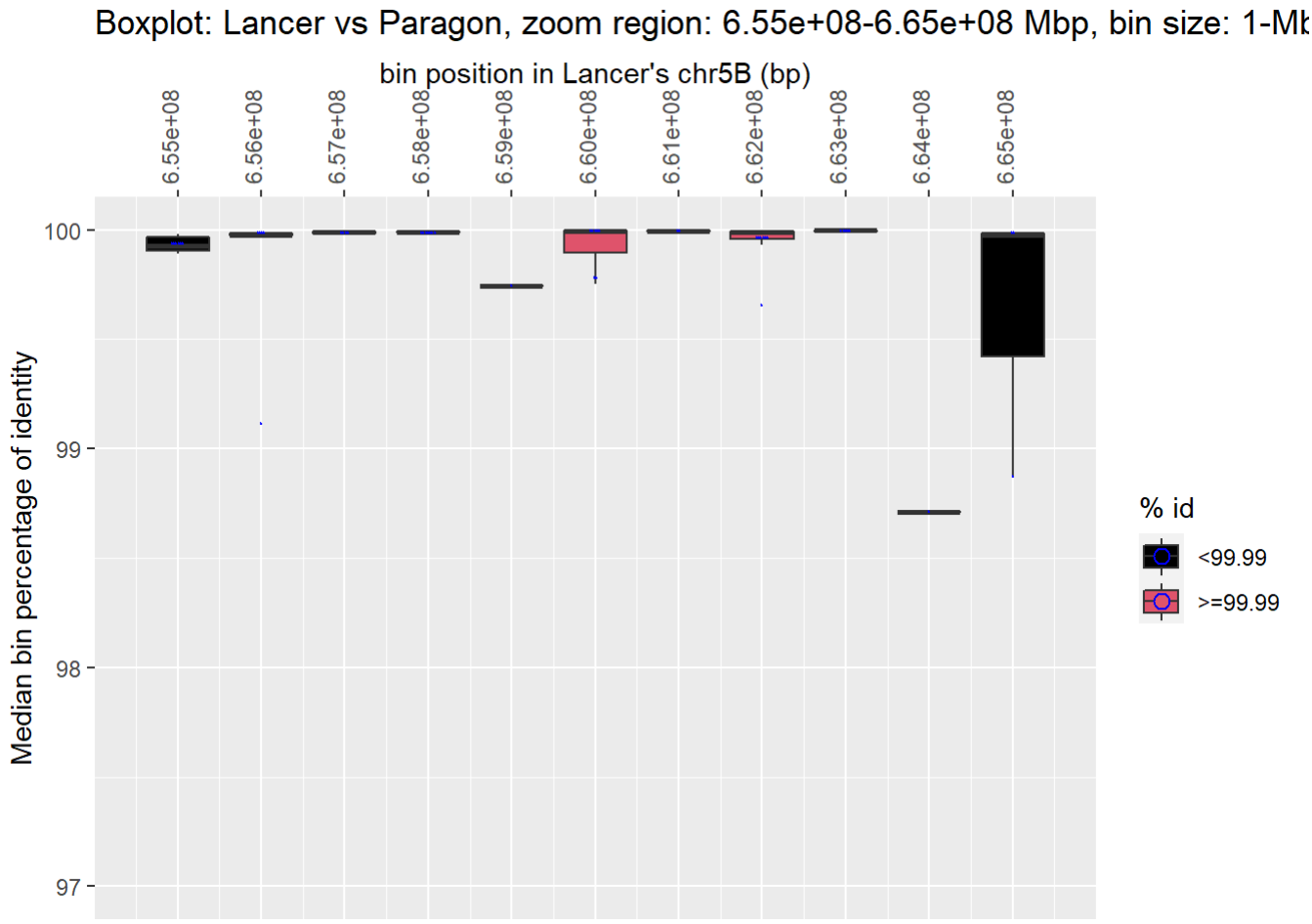
```
plot_boxplots_bin_median(aln_subset, bin_size = 5000000, bin_start = zoom_start, bin_end = zo
om_end, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar , x
_label_gap = 1000000, show_outliers = TRUE)
```




```
plot_boxplots_bin_median(aln_subset, bin_size = 2500000, bin_start = zoom_start, bin_end = zo
om_end, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar , x
_label_gap = 1000000, show_outliers = TRUE)
```

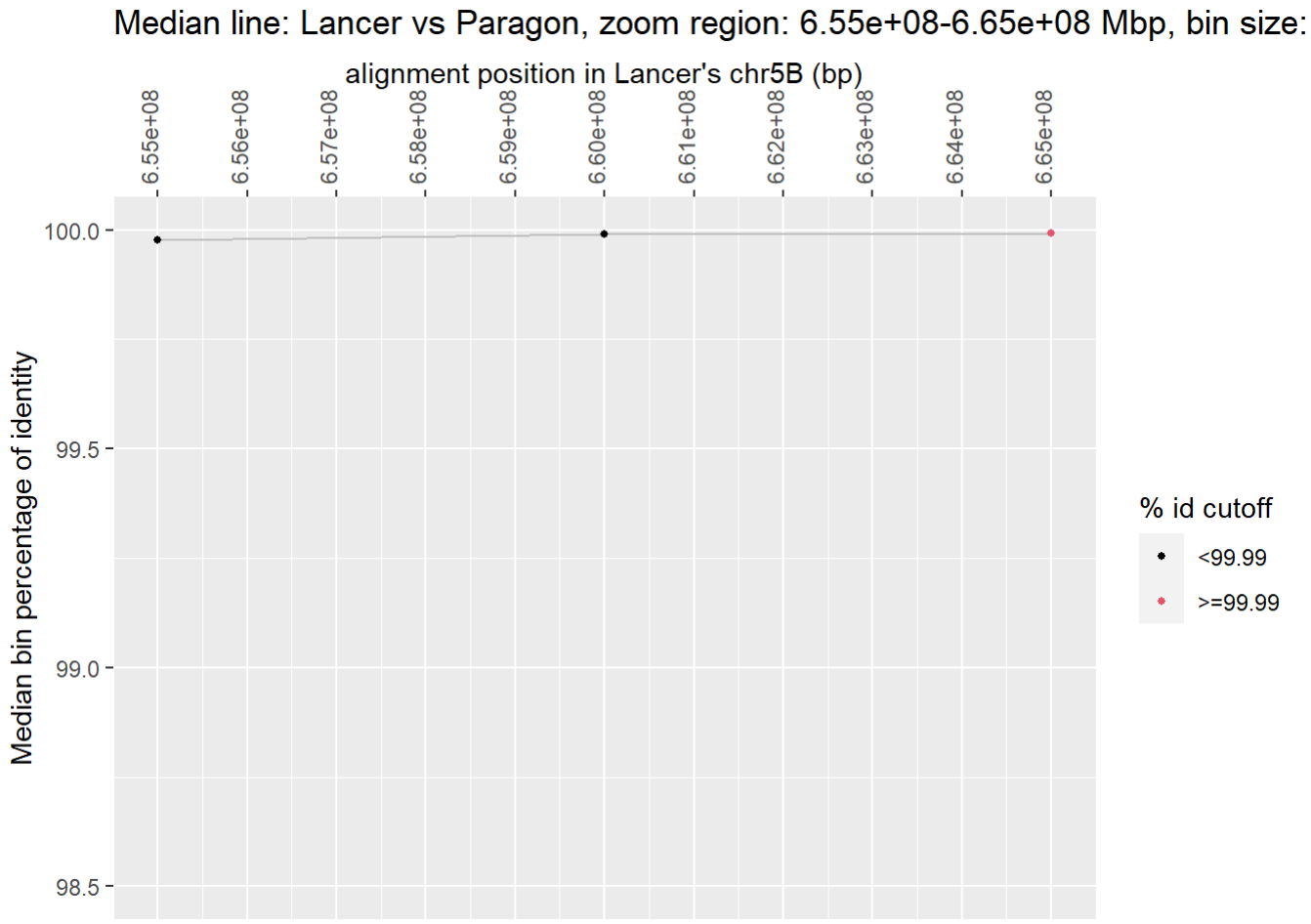



```
plot_boxplots_bin_median(aln_subset, bin_size = 1000000, bin_start = zoom_start, bin_end = zo
om_end, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar , x
_label_gap = 1000000, show_outliers = TRUE)
```

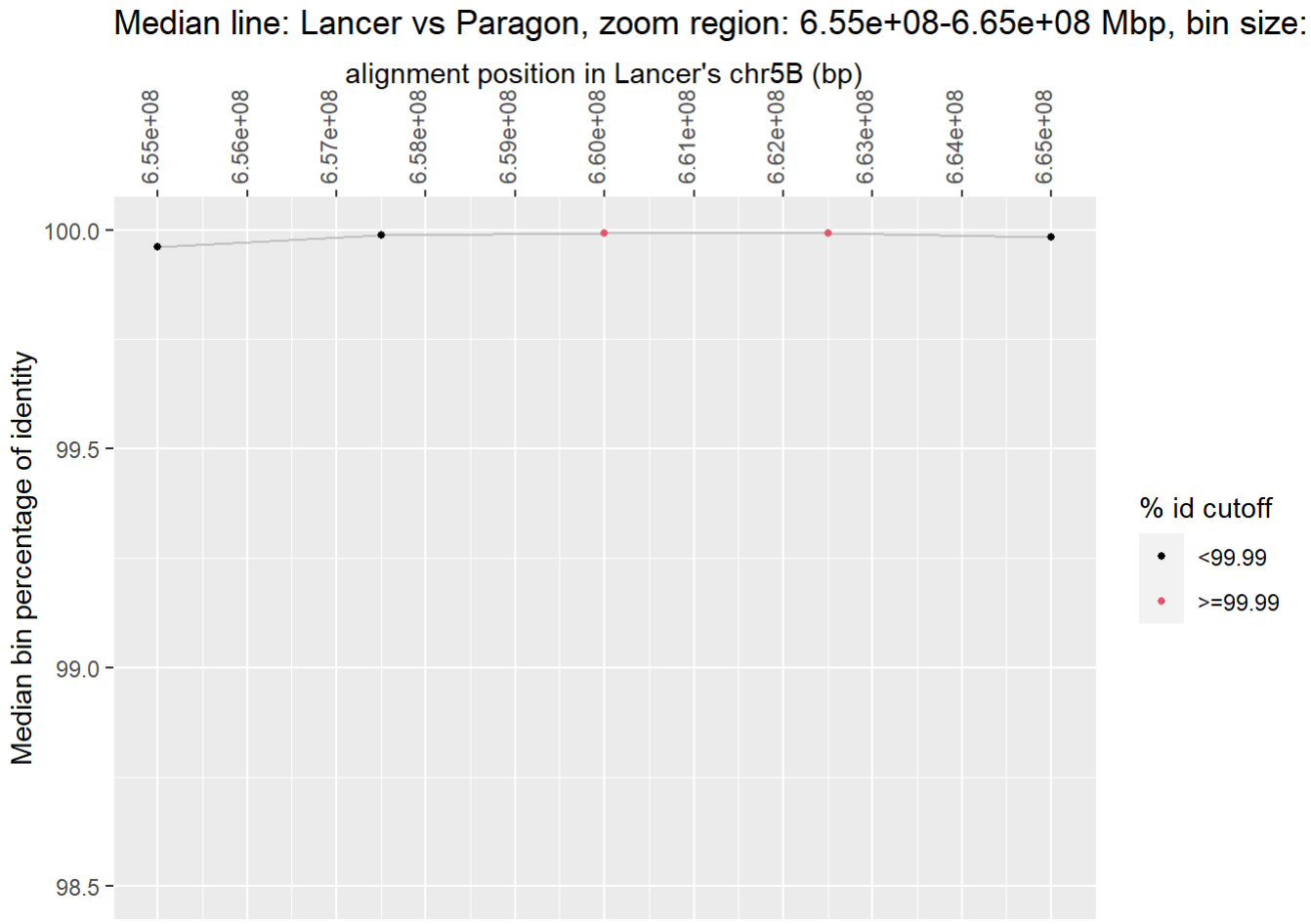


1.2.5. Plot the median line in the zoom region to see the haploblock predictions at different bin sizes

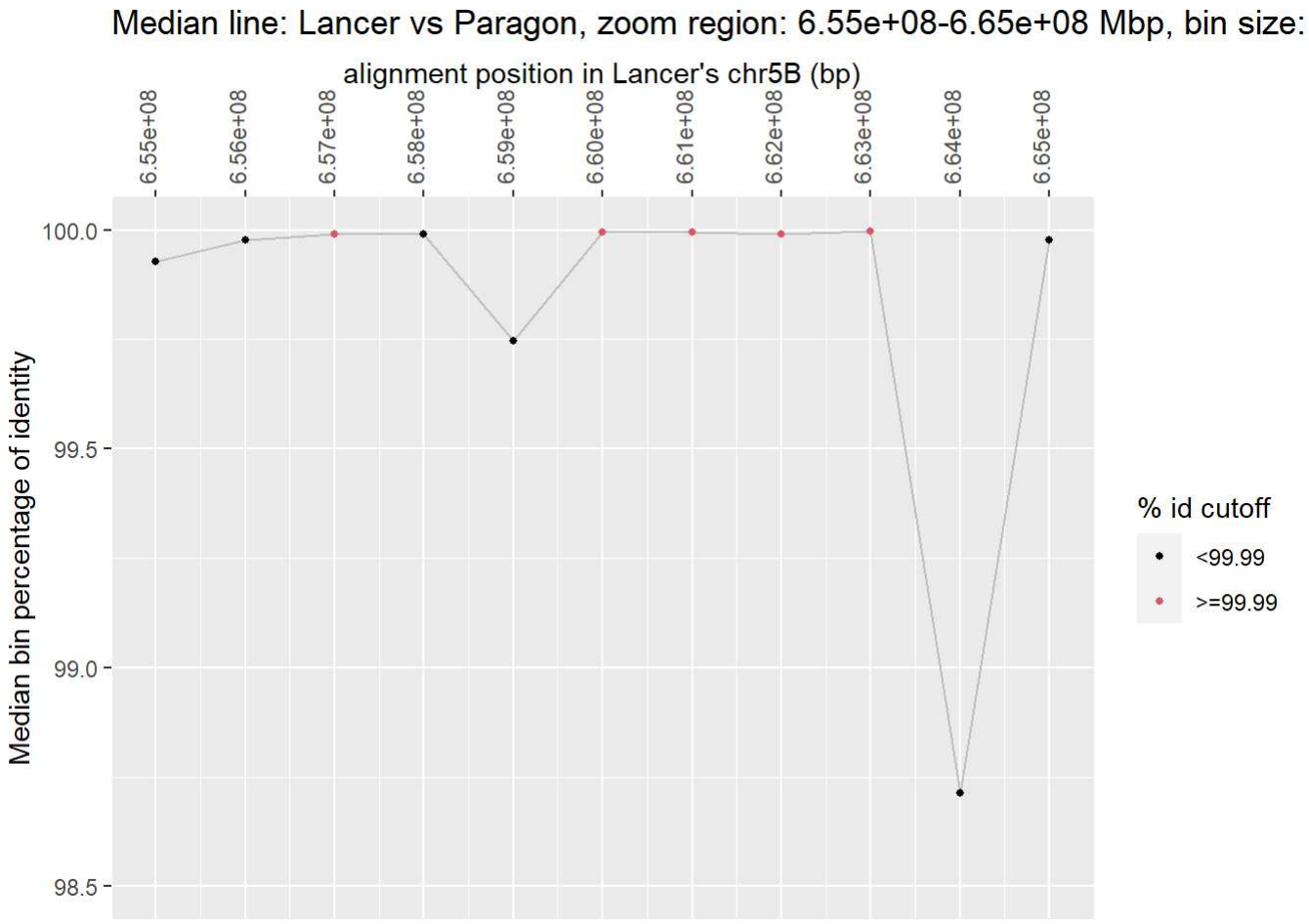
```
plot_line_bin_median(aln_subset, bin_size = 5000000, bin_start = zoom_start, bin_end = zoom_e
nd, ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cul
tivar , x_label_gap = 1000000)
```




```
plot_line_bin_median(aln_subset, bin_size = 2500000, bin_start = zoom_start, bin_end = zoom_e
nd, ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cul
tivar , x_label_gap = 1000000)
```



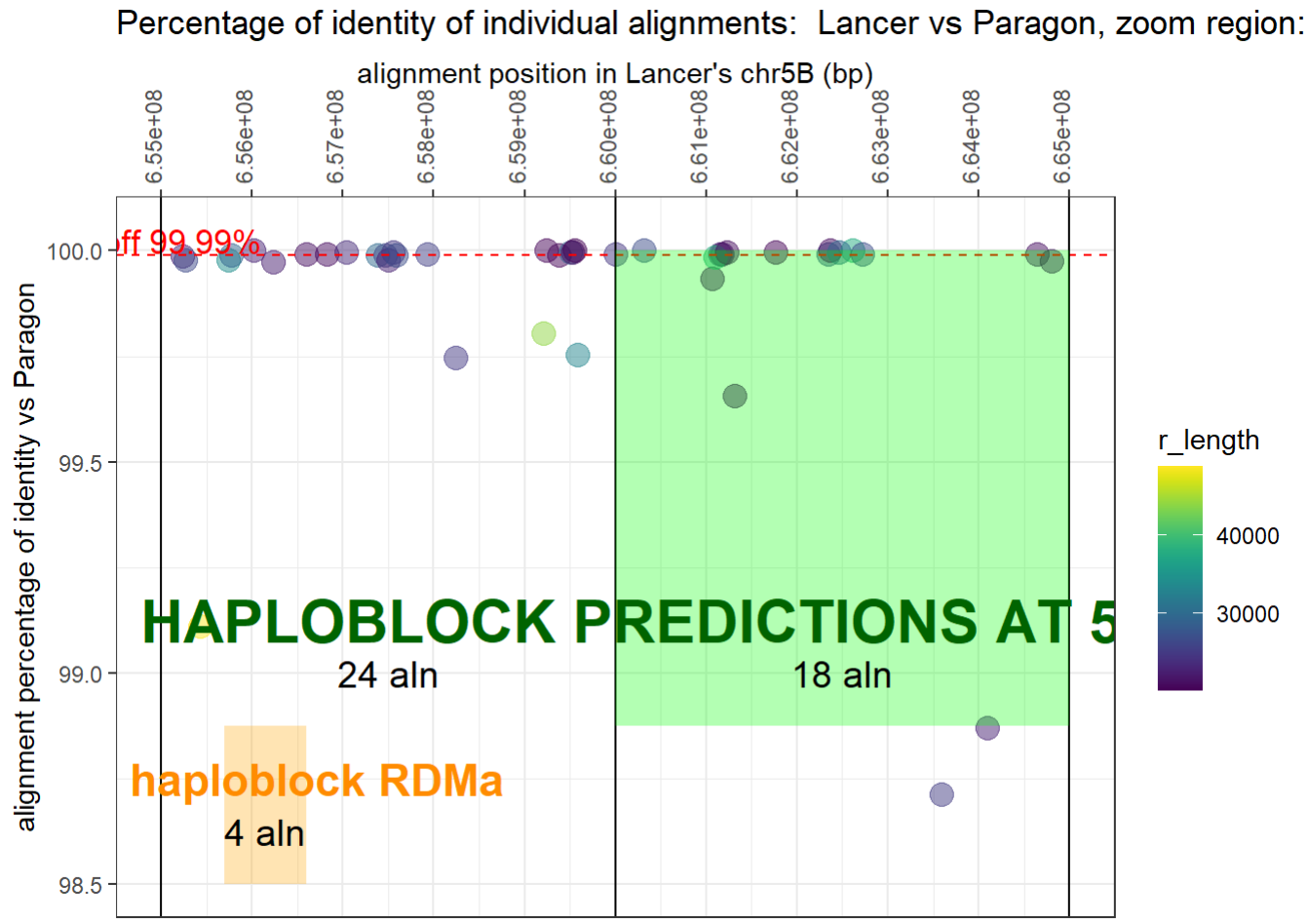

```
plot_line_bin_median(aln_subset, bin_size = 1000000, bin_start = zoom_start, bin_end = zoom_e
nd, ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cul
tivar , x_label_gap = 1000000)
```



1.2.6. Dot-plot the zoom region with the haploblock predictions and the target region and make decisions regarding the start and end limits of the IBS-haploblock

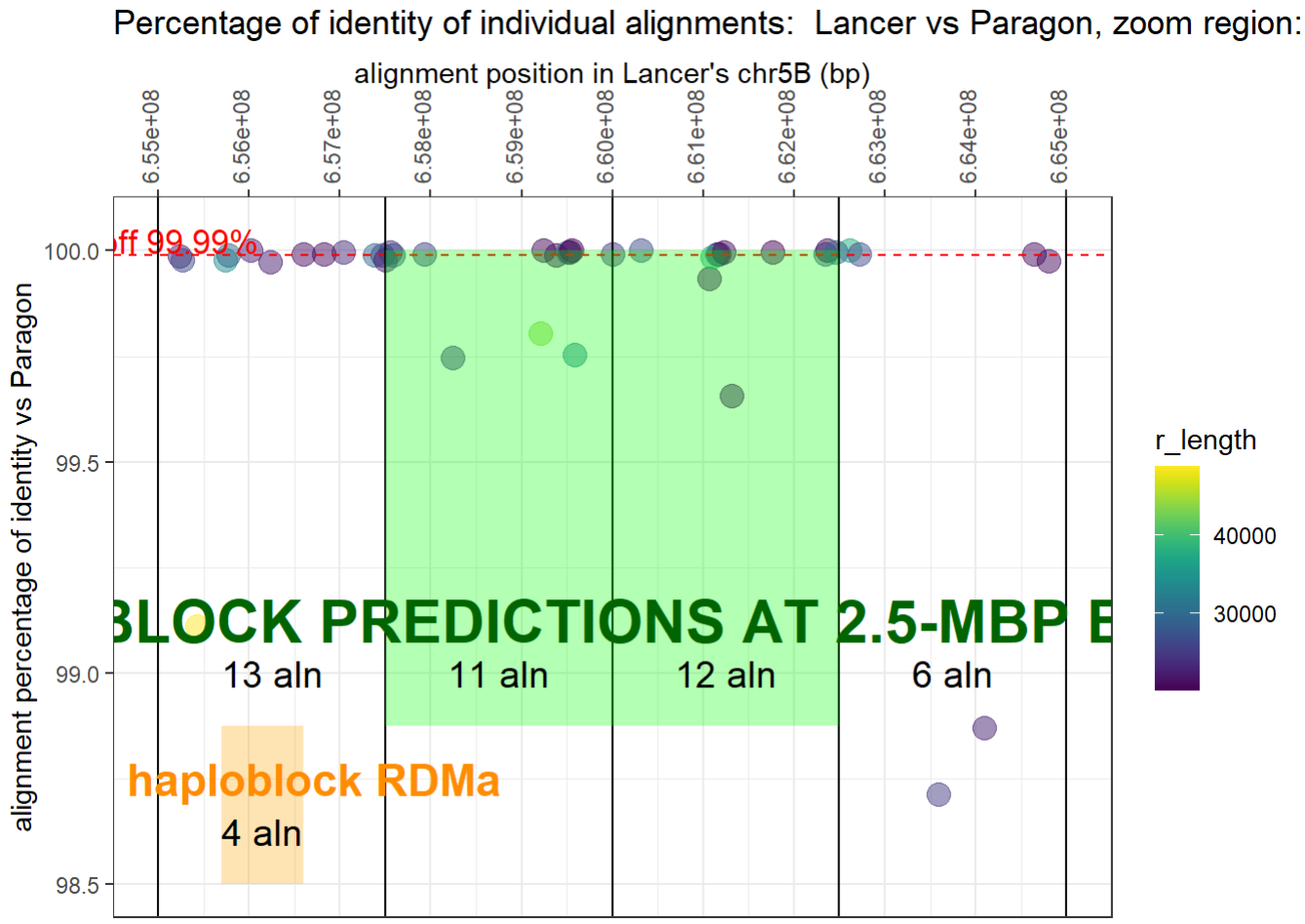
```
target <- data.frame(target_start, target_end)
plot_aln_and_bins(aln_subset = aln_subset, bin_size = 5000000, zoom_start = zoom_start, zoom_
end = zoom_end, highlighted_target = target, target_text = "SNP haploblock RDMa", fill_target
= "orange", color_target_text = "darkorange", fill_predictions = "green", color_prediction_te
xt = "darkgreen", ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_na
me = query_cultivar, dot_size = 4, x_label_gap = 1000000)
```

```
## [1] "BINS AT 5-MBP BIN SIZE"
##        bin perc_id_median cut_off block_no bin_start  bin_end aln_number
## 1 6.55e+08       99.97763  <99.99 NO_BLOCK  6.50e+08 6.55e+08         37
## 2 6.60e+08       99.98986  <99.99 NO_BLOCK  6.55e+08 6.60e+08         24
## 3 6.65e+08       99.99159 >=99.99        1  6.60e+08 6.65e+08         18
## [1] "BLOCK SUMMARY AT 5-MBP BIN SIZE"
##   bin_size      comparison block_no block_start block_end
## 1    5-Mbp Lancer->Paragon        1     6.6e+08  6.65e+08
```



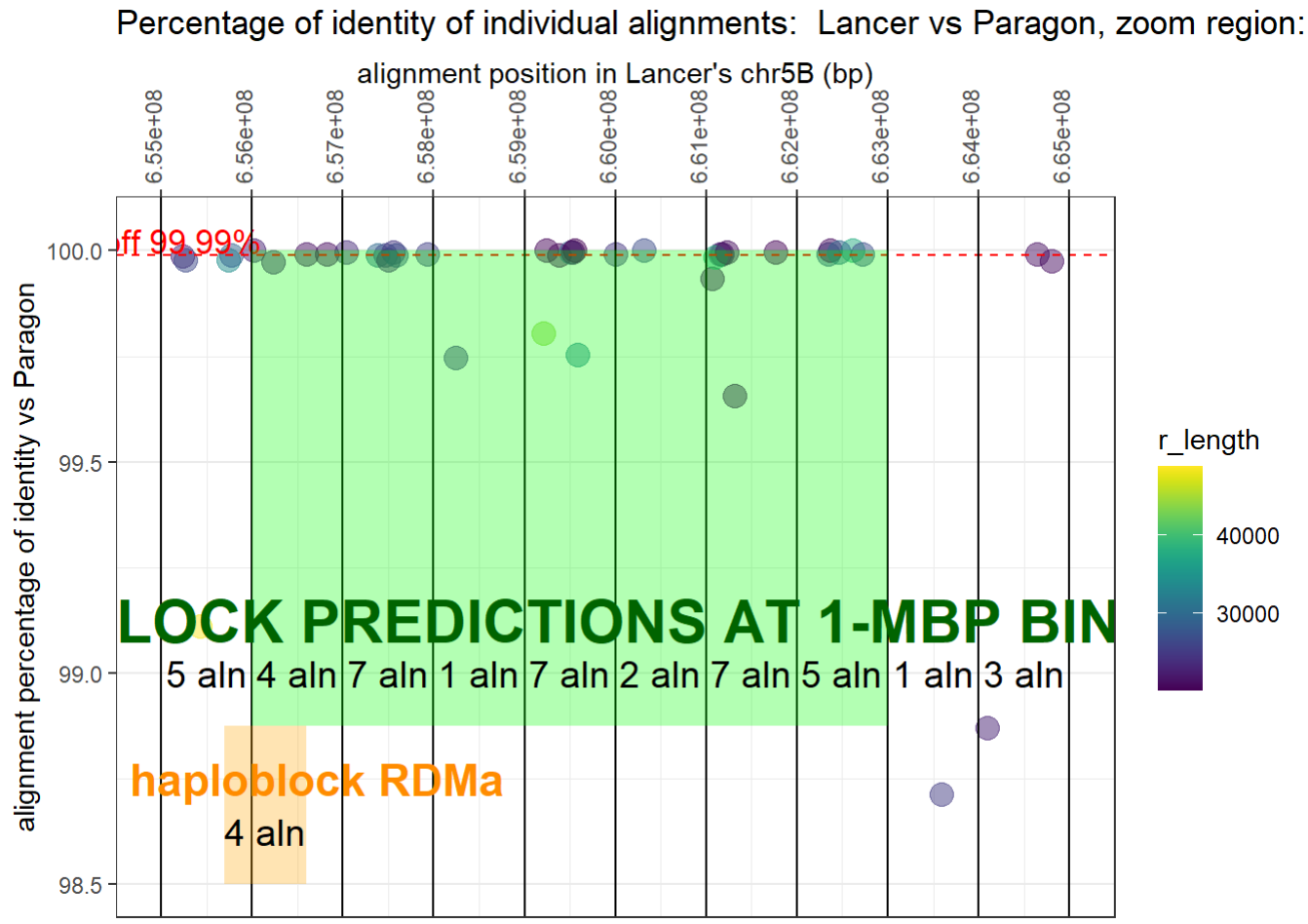

```
plot_aln_and_bins(aln_subset = aln_subset, bin_size = 2500000, zoom_start = zoom_start, zoom_
end = zoom_end, highlighted_target = target, target_text = "SNP haploblock RDMa", fill_target
= "orange", color_target_text = "darkorange", fill_predictions = "green", color_prediction_te
xt = "darkgreen", ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_na
me = query_cultivar, dot_size = 4, x_label_gap = 1000000)
```

```
## [1] "BINS AT 2.5-MBP BIN SIZE"
##         bin perc_id_median cut_off block_no bin_start   bin_end aln_number
## 1 655000000       99.96227  <99.99 NO_BLOCK 652500000 655000000         13
## 2 657500000       99.98867  <99.99 NO_BLOCK 655000000 657500000         13
## 3 660000000       99.99211 >=99.99        1 657500000 660000000         11
## 4 662500000       99.99228 >=99.99        1 660000000 662500000         12
## 5 665000000       99.98336  <99.99 NO_BLOCK 662500000 665000000          6
## [1] "BLOCK SUMMARY AT 2.5-MBP BIN SIZE"
##   bin_size      comparison block_no block_start block_end
## 1  2.5-Mbp Lancer->Paragon        1   657500000 662500000
```




```
plot_aln_and_bins(aln_subset = aln_subset, bin_size = 1000000, zoom_start = zoom_start, zoom_
end = zoom_end, highlighted_target = target, target_text = "SNP haploblock RDMa", fill_target
= "orange", color_target_text = "darkorange", fill_predictions = "green", color_prediction_te
xt = "darkgreen", ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_na
me = query_cultivar, dot_size = 4, x_label_gap = 1000000)
```

```
## [1] "BINS AT 1-MBP BIN SIZE"
##         bin perc_id_median cut_off block_no bin_start  bin_end aln_number
## 1  6.55e+08       99.92852  <99.99 NO_BLOCK  6.54e+08 6.55e+08          6
## 2  6.56e+08       99.97730  <99.99 NO_BLOCK  6.55e+08 6.56e+08          5
## 3  6.57e+08       99.99048 >=99.99        1  6.56e+08 6.57e+08          4
## 4  6.58e+08       99.98996  <99.99 NO_BLOCK  6.57e+08 6.58e+08          7
## 5  6.59e+08       99.74574  <99.99 NO_BLOCK  6.58e+08 6.59e+08          1
## 6  6.60e+08       99.99528 >=99.99        1  6.59e+08 6.60e+08          7
## 7  6.61e+08       99.99595 >=99.99        1  6.60e+08 6.61e+08          2
## 8  6.62e+08       99.99072 >=99.99        1  6.61e+08 6.62e+08          7
## 9  6.63e+08       99.99664 >=99.99        1  6.62e+08 6.63e+08          5
## 10 6.64e+08       98.71248  <99.99 NO_BLOCK  6.63e+08 6.64e+08          1
## 11 6.65e+08       99.97621  <99.99 NO_BLOCK  6.64e+08 6.65e+08          3
## [1] "BLOCK SUMMARY AT 1-MBP BIN SIZE"
##   bin_size      comparison block_no block_start block_end
## 1    1-Mbp Lancer->Paragon        1    6.56e+08  6.63e+08
```



DEFINING NEW PARAMETERS

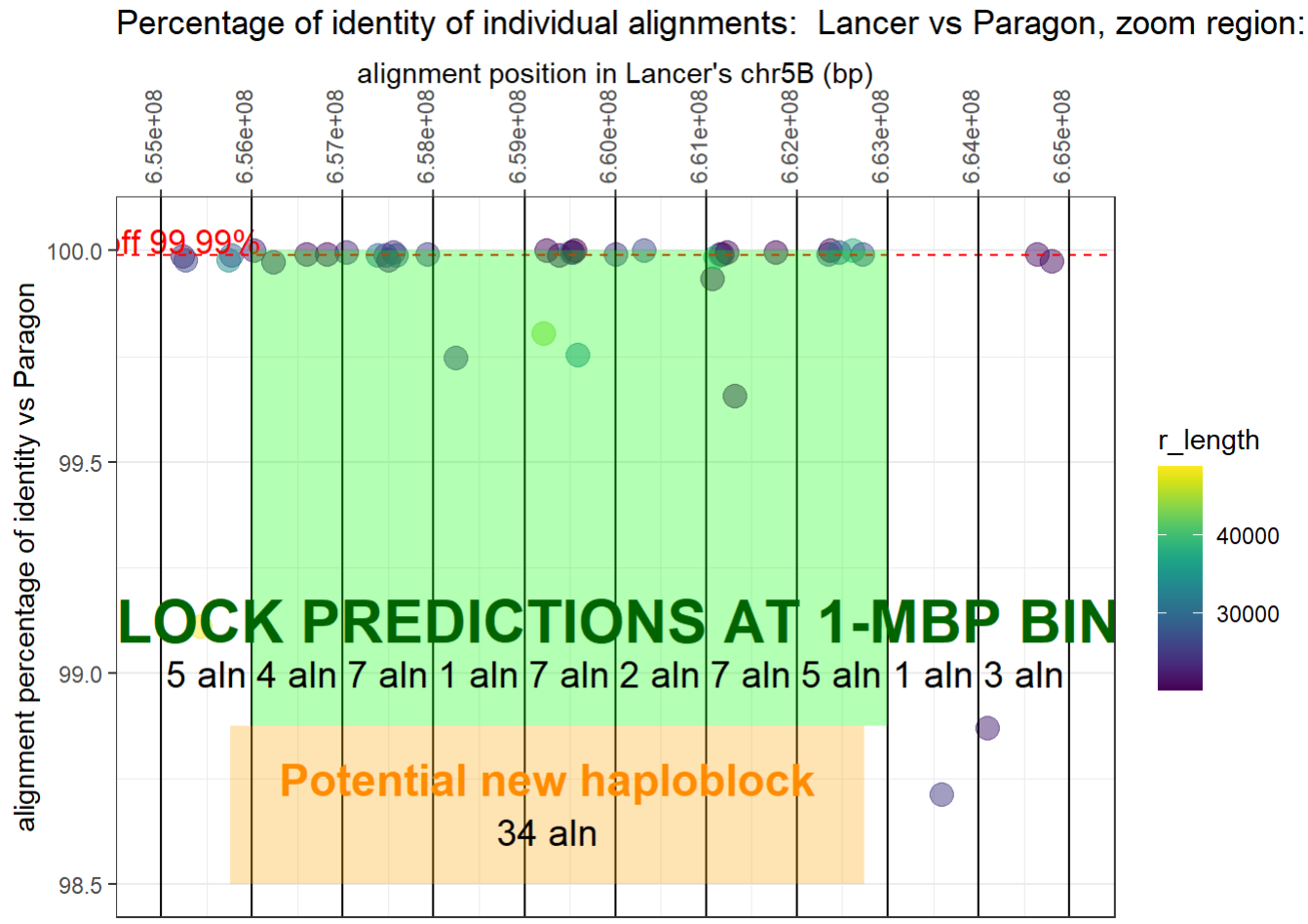
```
selected_start <- 655760000 # Genes will be extracted from this position. If you have no inte
rest in redefining your region, simply write 'target_start' or 'zoom_start', to keep with the
previous coordinates
selected_end <- 662740000 # Genes will be extracted until this position. If you have no inter
est in redefining your region, simply write 'target_end' or 'zoom_end', to keep with the prev
ious coordinates
target_text <- "Potential new haploblock" # Text to print on the selected region

selected_haploblock <- data.frame(selected_start, selected_end)
print(selected_haploblock)
```

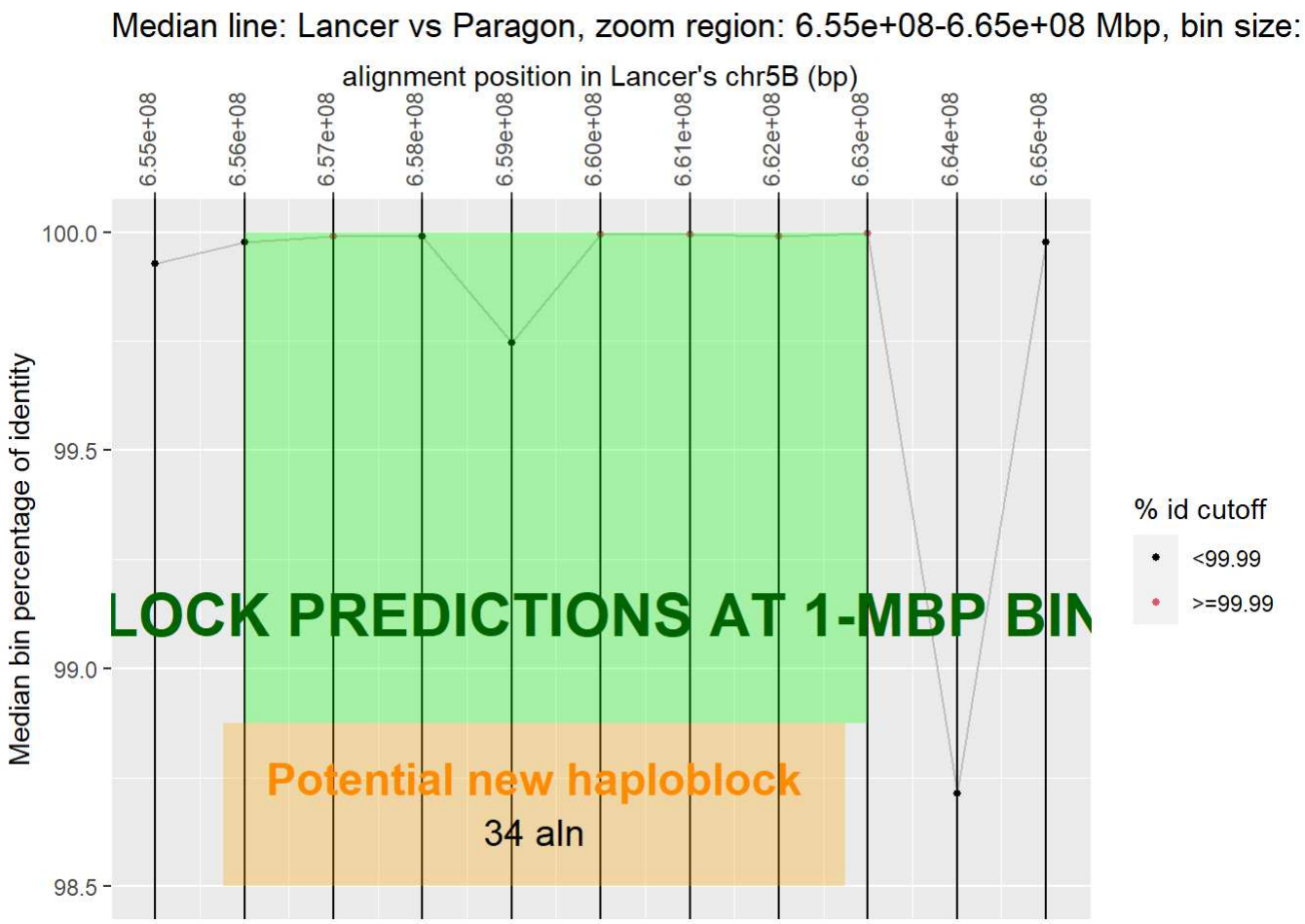
```
##   selected_start selected_end
## 1      655760000    662740000
```

1.2.7. Plot summary graphs

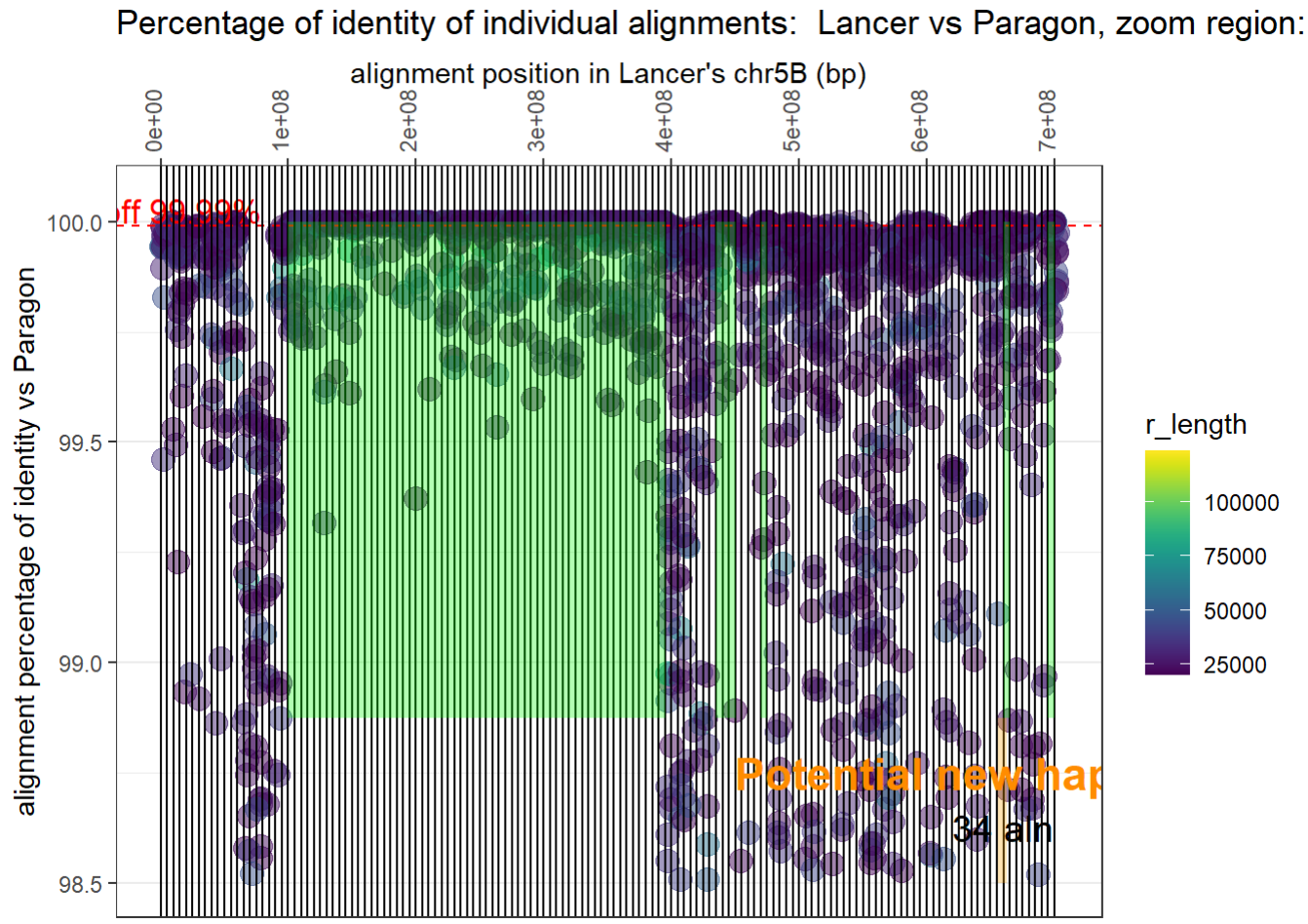
```
plot_aln_and_bins(print_tables = FALSE, aln_subset = aln_subset, bin_size = 1000000, zoom_sta
rt = zoom_start, zoom_end = zoom_end, highlighted_target = selected_haploblock, target_text =
target_text, fill_target = "orange", color_target_text = "darkorange", fill_predictions = "gr
een", color_prediction_text = "darkgreen", ymin = 98.5, cut_off = 99.99, reference_name = ref
erence_cultivar, query_name = query_cultivar, dot_size = 4, x_label_gap = 1000000)
```



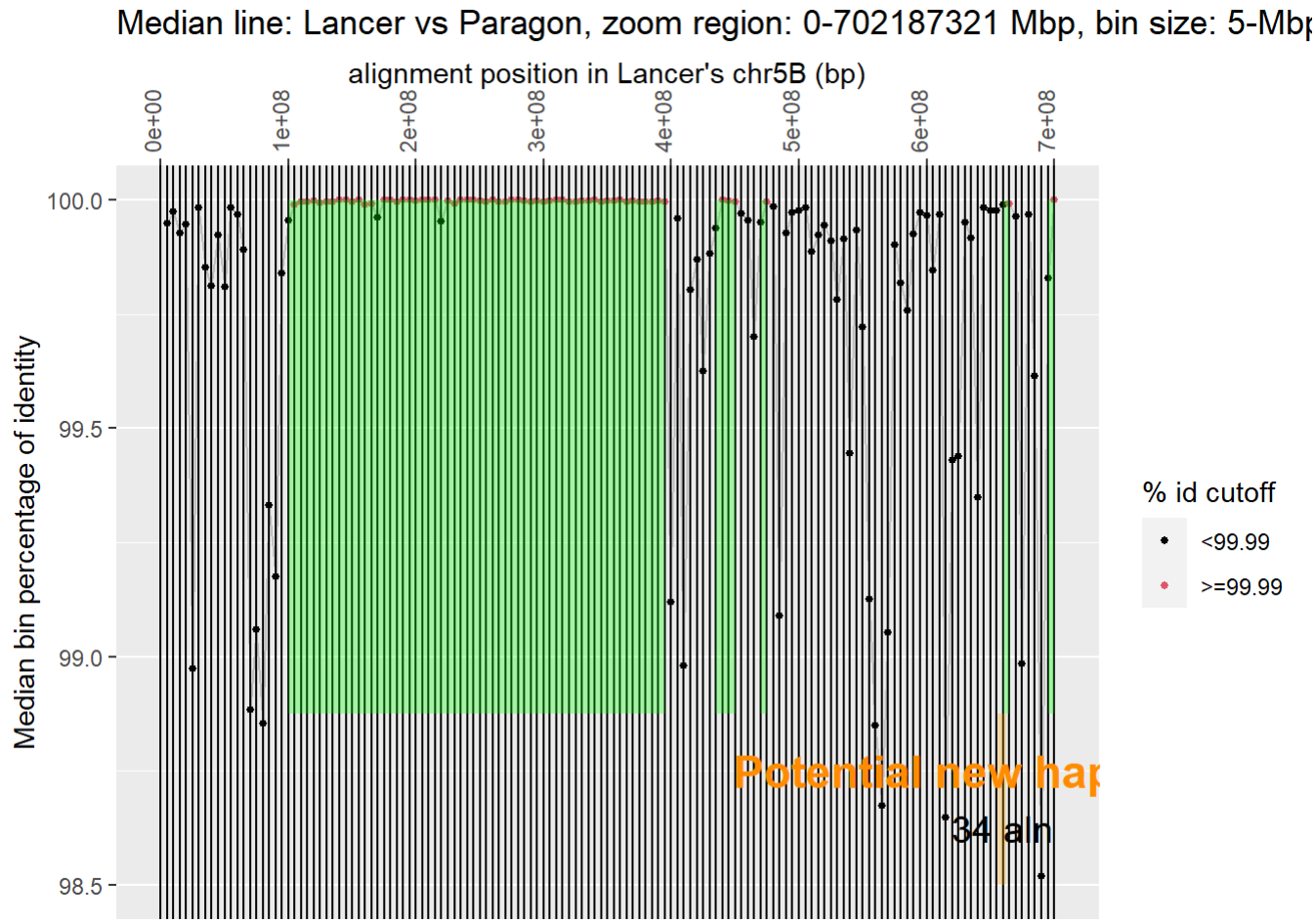

```
plot_bins_and_selected_region(print_tables = FALSE, aln_subset = aln_subset, bin_size = 10000
00, zoom_start = zoom_start, zoom_end = zoom_end, highlighted_target = selected_haploblock, t
arget_text = target_text, fill_target = "orange", color_target_text = "darkorange", fill_pred
ictions = "green", color_prediction_text = "darkgreen", ymin = 98.5, cut_off = 99.99, referen
ce_name = reference_cultivar, query_name = query_cultivar, dot_size = 4, x_label_gap = 100000
0)
```




```
plot_aln_and_bins(print_tables = FALSE, aln_subset = aln_subset, bin_size = 5000000, zoom_sta
rt = 0, zoom_end = max(aln_subset$re), highlighted_target = selected_haploblock, target_text
 = target_text, fill_target = "orange", color_target_text = "darkorange", fill_predictions =
"green", ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_name = quer
y_cultivar, dot_size = 4, x_label_gap = 100000000, prediction_text = FALSE, aln_text = F)
```




```
plot_bins_and_selected_region(print_tables = FALSE, aln_subset = aln_subset, bin_size = 5000000, zoom_start = 0, zoom_end = max(aln_subset$re), highlighted_target = selected_haploblock, target_text = target_text, fill_target = "orange", color_target_text = "darkorange", fill_predictions = "green", color_prediction_text = "darkgreen", ymin = 98.5, cut_off = 99.99, reference_name = reference_cultivar, query_name = query_cultivar, dot_size = 4, x_label_gap = 100000000, prediction_text = FALSE)
```



2. IDENTIFY GENES WITHIN THE JUST-MAPPED HAPLOBLOCK

Requirements:

- File 'projectedGenes__Triticum_aestivum_REFERENCECULTIVAR_v1.0.gff', in this case the reference is LongReach Lancer (downloadable from https://webblast.ipk-gatersleben.de/downloads/wheat/gene_projection/ (https://webblast.ipk-gatersleben.de/downloads/wheat/gene_projection/))
- File 'geneid_2_chinese.sourceid.txt' (downloadable from https://webblast.ipk-gatersleben.de/downloads/wheat/gene_projection/ (https://webblast.ipk-gatersleben.de/downloads/wheat/gene_projection/))

2.1. Download the files and save them in the working directory

2.2. Read the gff file containing the gene model projection for the reference cultivar

```
ref_gff <- read.table ( file = "projectedGenes__Triticum_aestivum_LongReach_Lancer_v1.0.gff",
sep = "\t" , header = F, stringsAsFactors = F)
```

2.3. Create subsets for gene projections only and edit the table

```
ref_gff_gene_only <- ref_gff[ref_gff$V3 == "gene",]
colnames(ref_gff_gene_only) <- c("chr", "annotation", "biotype", "start", "end", "score", "st
rand", "info", "ref_id")
ref_gff_gene_only$var_id <- gsub("ID=", "", ref_gff_gene_only$ref_id)
```

2.4. Create a subset for the haploblock

```
my_genes <- ref_gff_gene_only[grep(chromosome, ref_gff_gene_only$chr),]
my_genes <- my_genes[(my_genes$start >= selected_start) & (my_genes$end <= selected_end),]
```

2.5. Download ‘geneid_2_chinese.sourceid.txt’ from the last link and read the table

```
cs_id <- read.table(file = "geneid_2_chinese.sourceid.txt", sep="\t", header=T, stringsAsFact
ors = F)
```

2.6. Add a column to the subset with the IDs of the gene sources in Chinese Spring for each of Lancer’s genes

```
my_genes <- cs_id_filler(data = my_genes, library = cs_id, chr = chromosome, ref.var = tolowe
r(reference_cultivar))
```

2.7. Extract the names of the genes in chromosome 5B separated by commas

```
my_gene_sources <- as.character(my_genes$cs_id[!grepl("^source", my_genes$cs_id)])
my_gene_sources_text <- paste(my_gene_sources, collapse = ", ")
write.table(x = my_gene_sources_text, file = "my_gene_sources.txt", sep = "", row.names = F,
 col.names = F, quote = F)

my_variety_genes <- as.character(my_genes$var_id[!grepl("^source", my_genes$cs_id)])
my_variety_genes_text <- paste(my_variety_genes, collapse = ", ")
write.table(x = my_variety_genes_text, file = "my_variety_genes.txt", sep = "", row.names = F
, col.names = F, quote = F)

print_my_genes <- data.frame( "var_id" = my_genes$var_id, "chinese_id" = my_genes$cs_id)
write.csv2(x = print_my_genes, file = "my_genes_and_sources.csv", row.names = F, quote = F)
```

3. PROVE IF THE HAPLOBLOCK WAS CALLED BY GENE-BASED BLAST PAIRWISE ALIGNMENTS

Requirements:

- File ‘varieties_all_identities_2000bp.tar.gz’ (downloadable from https://opendata.earlham.ac.uk/wheat/under_license/toronto/Brinton_etal_2020-05-20-Haplotypes-for-wheat-breeding/pairwise_blast/

(https://opendata.earlham.ac.uk/wheat/under_license/toronto/Brinton_etal_2020-05-20-Haplotypes-for-wheat-breeding/pairwise_blast/))

- File 'iwgsc_refseq_v1.2_gene_annotation.zip' (downloadable from https://urgi.versailles.inrae.fr/download/iwgsc/IWGSC_RefSeq_Annotations/v1.2/ (https://urgi.versailles.inrae.fr/download/iwgsc/IWGSC_RefSeq_Annotations/v1.2/))

3.1. Download the zip file, decompress it in the working directory and read the tables

```
HC_gtf <- read.table("IWGSC_v1.2_HC_20200615.gff3", sep = "\t", header = FALSE, stringsAsFact
ors = FALSE)
LC_gtf <- read.table("IWGSC_v1.2_LC_20200615.gff3", sep = "\t", header = FALSE, stringsAsFact
ors = FALSE)
ALL_gtf <- rbind(HC_gtf, LC_gtf)
```

3.2. Extract the BLAST alignments and put them in table with Chinese Spring genes and their location in the IWGSC genome (long process)

```
BLAST_library <- read_pairwise_position(blast_path_gz = "varieties_all_identities_2000bp.tab.
gz", gtf = ALL_gtf, write_table = "BLAST_library.tab")
```

3.3. Extract a subset with Lancer-Paragon comparison in chr5B

```
BLAST_subset <- BLAST_library[grepl(paste0(tolower(reference_cultivar), "->", tolower(query_c
ultivar), sep = ""), BLAST_library$aln_type) & grepl(chromosome, BLAST_library$chr),]
```

3.4. Use the vector with the genes identified in the previous step to extract another subset with only the genes in our haploblock

```
BLAST_subset <- BLAST_subset[grepl(paste(my_gene_sources, collapse = "|"), BLAST_subset$trans
cript),]
write.csv2(x = BLAST_subset, file = "BLAST_subset.csv", row.names = F, quote = F)
```

Brinton et al (2020) only retained only gene projections consistent with the expected chromosome. This explains why this subset contains less genes that the total amount of annotated genes in our region.

```
colnames(BLAST_subset)[1] <- "cs_transcript"
colnames(BLAST_subset)[14] <- "cs_start"
colnames(BLAST_subset)[15] <- "cs_end"

BLAST_subset$var_transcript <- my_genes$var_id[grepl(paste(BLAST_subset$cs_transcript, collap
se = "|"), my_genes$cs_id)]
BLAST_subset$var_start <- my_genes$start[grepl(paste(BLAST_subset$var_transcript, collapse =
"|"), my_genes$var_id)]
BLAST_subset$var_end <- my_genes$end[grepl(paste(BLAST_subset$var_transcript, collapse = "|"
), my_genes$var_id)]
```

Notice that some genes were filtered out if they contained more than one projection in the expected chromosome, so the amount of genes shown in this table is expected to be smaller than those found in lancer's projected genes in the region. The genes that were excluded can be seen here:

```
my_gene_sources[my_gene_sources %in% BLAST_subset$cs_transcript == FALSE]
```

```
##  [1] "TraesCS5B02G496400" "TraesCS5B02G496700" "TraesCS5B02G497300"
##  [4] "TraesCS5B02G497500" "TraesCS5B02G500500" "TraesCS5B02G500600"
##  [7] "TraesCS5B02G500700" "TraesCS5B02G502700" "TraesCS5B02G503400"
## [10] "TraesCS5B02G503500" "TraesCS5B02G504400" "TraesCS5B02G552700"
```

```
my_genes$var_id[grepl(paste(my_gene_sources[my_gene_sources %in% BLAST_subset$cs_transcript =
= FALSE], collapse = "|"), my_genes$cs_id)]
```

```
##  [1] "TraesLAC5B01G535600" "TraesLAC5B01G535800" "TraesLAC5B01G536100"
##  [4] "TraesLAC5B01G536300" "TraesLAC5B01G539700" "TraesLAC5B01G539800"
##  [7] "TraesLAC5B01G539900" "TraesLAC5B01G542300" "TraesLAC5B01G542700"
## [10] "TraesLAC5B01G542900" "TraesLAC5B01G543700" "TraesLAC5B01G544200"
```

We can extract another list containing only the genes that were present among the BLAST alignments

```
my_gene_sources_filtered_by_Brinton <- my_gene_sources[my_gene_sources %in% BLAST_subset$cs_t
ranscript == TRUE]
my_variety_genes_filtered_by_Brinton <- my_genes$var_id[grepl(paste(my_gene_sources [my_gene_
sources %in% BLAST_subset$cs_transcript == TRUE], collapse = "|"), my_genes$cs_id)]
my_gene_sources_filtered_by_Brinton_text <- paste(my_gene_sources_filtered_by_Brinton, collap
se = ", ")
write.table(x = my_gene_sources_filtered_by_Brinton_text, file = "my_gene_sources_filtered_by
_Brinton_text.txt", sep = "", row.names = F, col.names = F, quote = F)

my_variety_genes_filtered_by_Brinton_text <- paste(my_variety_genes_filtered_by_Brinton, coll
apse = ", ")
write.table(x = my_variety_genes_filtered_by_Brinton_text, file = "my_variety_genes_filtered_
by_Brinton_text.txt", sep = "", row.names = F, col.names = F, quote = F)

print_blast <- data.frame( "var_id" = my_variety_genes_filtered_by_Brinton_text, "chinese_id"
= my_gene_sources_filtered_by_Brinton_text)
write.csv2(x = print_blast, file = "my_genes_and_sources_filtered_by_Brinton.csv", row.names
 = F, quote = F)
```

3.5. Calculate the percentage of identity in windows of 20 genes where genes containing Ns are filtered out

```
BLAST_subset <- BLAST_subset[ BLAST_subset$Ns_total == 0, ]

window_BLAST_subset <- edited_calculate_pid_windows(aln_data = BLAST_subset)
blocks_BLAST_subset <- assign_blocks(window_BLAST_subset)
```

```
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
## [1] "less than 100%"
```

```
blocks_BLAST_subset
```

```
##    start end  cs_start    cs_end var_start   var_end pident_mean
## 1      1  20 663134978 666196835 655761539 658486203    99.99828
## 2      2  21 663234734 666207172 656020702 658496928    99.99761
## 3      3  22 663796017 666209291 656553758 658499392    99.99761
## 4      4  23 664022132 666372136 656600250 658525735    99.99742
## 5      5  24 664672987 668463343 657035254 659882405    99.99742
## 6      6  25 664682920 668474547 657044824 659893484    99.99609
## 7      7  26 664726808 668519308 657104030 659938479    99.99609
## 8      8  27 664949927 668583715 657329620 660019299    99.99498
## 9      9  28 664979682 669217640 657359517 660818003    99.99587
## 10    10  29 664998106 669232964 657377616 660827247    99.99460
## 11    11  30 665016776 669292146 657396273 660884339    99.99254
## 12    12  31 665024295 669306368 657403038 660900818    99.99387
## 13    13  32 665170918 669452267 657534486 661060959    99.99280
## 14    14  33 665539582 669483645 657903967 661091774    99.99280
## 15    15  34 665560612 669898631 657925163 661273027    99.99280
## 16    16  35 665719477 669909637 658082849 661430757    99.99382
## 17    17  36 665767971 670099306 658131416 661722411    99.99146
## 18    18  37 665938590 670116476 658242152 661739162    99.99146
## 19    19  38 665939237 670118309 658242780 661741296    99.99146
## 20    20  39 666195640 670130079 658485922 661752676    99.99146
## 21    21  40 666202129 670246635 658492724 661761415    99.99146
## 22    22  41 666207065 670305619 658497166 661773746    99.99213
## 23    23  42 666371429 670694936 658525028 662378160    99.99213
## 24    24  43 668461124 670716758 659880577 662400081    99.99449
##           aln_type start_cs_transcript  end_cs_transcript start_var_transcript
## 1  lancer->paragon  TraesCS5B02G495900 TraesCS5B02G500000  TraesLAC5B01G535100
## 2  lancer->paragon  TraesCS5B02G496300 TraesCS5B02G500100  TraesLAC5B01G535500
## 3  lancer->paragon  TraesCS5B02G496600 TraesCS5B02G500200  TraesLAC5B01G535700
## 4  lancer->paragon  TraesCS5B02G497100 TraesCS5B02G500300  TraesLAC5B01G536000
## 5  lancer->paragon  TraesCS5B02G497700 TraesCS5B02G501100  TraesLAC5B01G536500
## 6  lancer->paragon  TraesCS5B02G497800 TraesCS5B02G501200  TraesLAC5B01G536600
## 7  lancer->paragon  TraesCS5B02G497900 TraesCS5B02G501300  TraesLAC5B01G536700
## 8  lancer->paragon  TraesCS5B02G498000 TraesCS5B02G501400  TraesLAC5B01G537100
## 9  lancer->paragon  TraesCS5B02G498100 TraesCS5B02G501600  TraesLAC5B01G537200
## 10 lancer->paragon  TraesCS5B02G498200 TraesCS5B02G501800  TraesLAC5B01G537300
## 11 lancer->paragon  TraesCS5B02G498300 TraesCS5B02G501900  TraesLAC5B01G537400
## 12 lancer->paragon  TraesCS5B02G498400 TraesCS5B02G502100  TraesLAC5B01G537500
## 13 lancer->paragon  TraesCS5B02G498500 TraesCS5B02G502200  TraesLAC5B01G537600
## 14 lancer->paragon  TraesCS5B02G498700 TraesCS5B02G502300  TraesLAC5B01G537800
## 15 lancer->paragon  TraesCS5B02G498800 TraesCS5B02G503200  TraesLAC5B01G537900
## 16 lancer->paragon  TraesCS5B02G498900 TraesCS5B02G503300  TraesLAC5B01G538000
## 17 lancer->paragon  TraesCS5B02G499100 TraesCS5B02G503700  TraesLAC5B01G538200
## 18 lancer->paragon  TraesCS5B02G499300 TraesCS5B02G503800  TraesLAC5B01G538400
## 19 lancer->paragon  TraesCS5B02G499400 TraesCS5B02G503900  TraesLAC5B01G538500
## 20 lancer->paragon  TraesCS5B02G500000 TraesCS5B02G504000  TraesLAC5B01G539000
## 21 lancer->paragon  TraesCS5B02G500100 TraesCS5B02G504100  TraesLAC5B01G539100
## 22 lancer->paragon  TraesCS5B02G500200 TraesCS5B02G504200  TraesLAC5B01G539200
## 23 lancer->paragon  TraesCS5B02G500300 TraesCS5B02G504600  TraesLAC5B01G539300
## 24 lancer->paragon  TraesCS5B02G501100 TraesCS5B02G504700  TraesLAC5B01G540400
##     end_var_transcript block_no
## 1  TraesLAC5B01G539000       NA
## 2  TraesLAC5B01G539100       NA
## 3  TraesLAC5B01G539200       NA
## 4  TraesLAC5B01G539300       NA
## 5  TraesLAC5B01G540400       NA
## 6  TraesLAC5B01G540500       NA
```

```
## 7  TraesLAC5B01G540600       NA
## 8  TraesLAC5B01G540700       NA
## 9  TraesLAC5B01G541300       NA
## 10 TraesLAC5B01G541500       NA
## 11 TraesLAC5B01G541600       NA
## 12 TraesLAC5B01G541800       NA
## 13 TraesLAC5B01G541900       NA
## 14 TraesLAC5B01G542000       NA
## 15 TraesLAC5B01G542400       NA
## 16 TraesLAC5B01G542600       NA
## 17 TraesLAC5B01G543100       NA
## 18 TraesLAC5B01G543200       NA
## 19 TraesLAC5B01G543300       NA
## 20 TraesLAC5B01G543400       NA
## 21 TraesLAC5B01G543500       NA
## 22 TraesLAC5B01G543600       NA
## 23 TraesLAC5B01G543800       NA
## 24 TraesLAC5B01G543900       NA
```