Supplemental code: Performance of methods to detect genetic variants from bisulfite sequencing data in a non-model species

M Lindner

August 2021

Contents

1 Set up R environment

```
# load r packages
library(dplyr)
library(tidyr)
library(stringr)
library(ggplot2)
library(cowplot)
library(RColorBrewer)
```

2 SNPs from whole genome bisulfite sequencing data

2.1 Bioinformatics pipeline

Because WGBS data showed high duplication rates, library preparation and sequencing were performed twice for all samples and data from both runs were merged.

2.1.1 Prepare raw data

```
# make links to raw data for the 2020 run
for f in *.gz; do ln -s /mnt/nfs/bioinfdata/primary/AnE/Parus_major/raw_data/
    whole_genome_sequencing/bisulfite_sequencing/novoseq/
    selection_lines2018_redo18062020/data/$f /home/NIOO.INT/melaniel/projects/
    WGBS_Snakemake_Bismark/raw_data/reseq2020/$f; done

# change file names for better processing within the pipeline
rename 's/_........_L00[0-9]//' *.gz #remove adapter sequences and lane
rename 's/_001//' *.gz # remove _001
```

```
# make links to raw data for the 2018 run
for f in *fastq; do ln -s /mnt/nfs/bioinfdata/primary/AnE/Parus_major/raw_data/
    whole_genome_sequencing/bisulfite_sequencing/novoseq/selection_lines2018/$f /
    home/NIOO.INT/melaniel/projects/WGBS_Snakemake_Bismark/raw_data/seq2018/$f; done

# change file names for better processing within the pipeline
rename 's/_........_L00[0-9]//' *.fastq #remove adapter sequences and lane
rename 's/_001//' *.fastq # remove _001
```

2.1.2 Quality control, data trimming, and alignments

Quality control and data trimming were part of a snakemake pipeline used for another project. The pipeline also included alignment and methylation calling using Bismark (new flag values) for all samples. The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs_new_flag_vals).

Bismark alignments with old flag values were part of a snakemake pipeline written for this project which can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs). The pipeline includes the Bismark alignments with old flag values, deduplication, addition of sample-specific read groups, and merging of alignments of the two sequencing runs for each sample for the Bismark alignments with the old flag values. It also includes the assessment of the number of mapped reads, average coverage depth, and breadth of coverage as well as the removal of reads mapping to the Z chromosome or mitochondrial DNA of Bismark alignments with both new and old flag values.

The alignments with biscuit and gemBS were part of the tool-specific pipelines (see below).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 32 --use-conda
```

2.1.3 SNP calling

Here, we tested 7 tools for SNP calling from WGBS data: * Bis-SNP (https://github.com/dnaase/Bis-tools/tree/master/Bis-SNP) * BS-SNPer (https://github.com/hellbelly/BS-Snper) * CGmapTools (https://github.com/guoweilong/cgmaptools) * EpiDiverse-SNP pipeline (https://github.com/EpiDiverse/snp) * MethylExtract (https://github.com/bioinfoUGR/methylextract) * biscuit (https://github.com/zhou-lab/biscuit) * gemBS (https://github.com/heathsc/gemBS)

2.1.3.1 Bis-SNP

Prior to SNP calling, Bis-SNP involved a BQ score recalibration of the alignments. The BQ recalibration requires a list of known SNPs. We here used SNPs of (mostly female) great tits genotyped on a high density SNP chip (https://onlinelibrary.wiley.com/doi/abs/10.1111/1755-0998.12778).

```
# Get fam IDs for samples
for i in `cat SampleIDs`; do echo "awk '{if(\$2==$i){print \$1,\$2}}' Sept2018/
    gt_NLallSept2018_nobadsnps2020_update.fam";done | bash > BirdsPlink.out

# filter plink files and get Snp quality:
plink --bfile Sept2018/gt_NLallSept2018_nobadsnps2020_update --allow-extra-chr --
    chr-set 32 --het --hardy --keep BirdsPlink.out --out plink_SNP_Quality/
    gt_ERC_WGBS_ExampleBirds

# Chr names in plink files differ from Chr names in WGBS data;
# update Chr position; https://zzz.bwh.harvard.edu/plink/dataman.shtml#recode
plink --bfile Sept2018/gt_NLallSept2018_nobadsnps2020_update --allow-extra-chr --
    chr-set 32 --update-map plink_updatedChrNames/new_chr_names --update-chr --make-
    bed --out plink_updatedChrNames/gt_NLallSept2018_nobadsnps2020_updatedChrNames

# get .vcf file
```

```
plink --bfile plink_updatedChrNames/gt_NLallSept2018_nobadsnps2020_updatedChrNames
    --allow-extra-chr --chr-set 32 --recode vcf --keep BirdsPlink.out --out
    plink_vcf/gt_ERC_WGBS_ExampleBirds

# prepare .vcf file
grep -vP '^#' gt_ERC_WGBS_ExampleBirds.vcf | sort -k1n > ~/gt_ERC_WGBS_ExampleBirds
    .vcf.body.sorted
grep -P '^#' gt_ERC_WGBS_ExampleBirds.vcf | sort -k1n > ~/gt_ERC_WGBS_ExampleBirds.
    vcf.header.sorted
cat ~/gt_ERC_WGBS_ExampleBirds.vcf.header.sorted | ~/gt_ERC_WGBS_ExampleBirds.vcf.
    body.sorted > ~/gt_ERC_WGBS_ExampleBirds_sorted.vcf
# manually chnage the header if needed !

# sort positions within chr of .vcf file
bcftools sort -o known_SNPs_sorted.vcf gt_ERC_WGBS_ExampleBirds_sorted.vcf
```

The pipeline includes the BQ recalibration of alignments, the removal of reads mapping to Z chromosome or mitochondrial DNA (for the recalibrated alignments), SNP calling from the recalibrated alignments as well as non-recalibrated alignments, SNP filtering (including diagnostics plots, **Figure 3** and **Figure 4**), and the evaluation of SNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/Bis-snp).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline:
snakemake -j 8 --use-conda
```

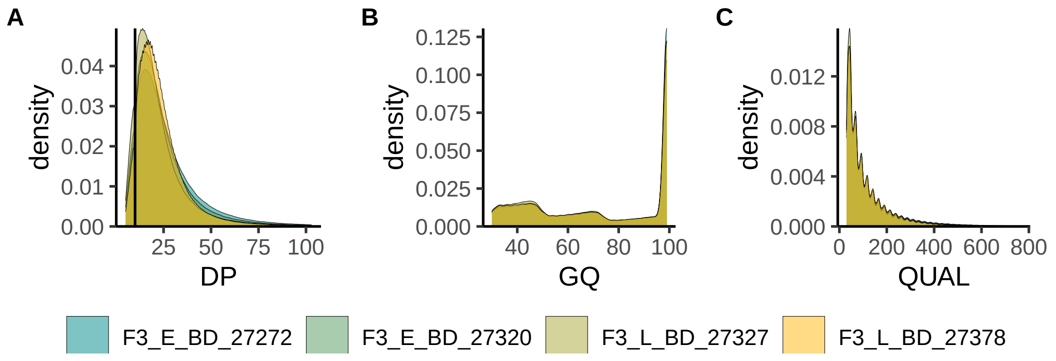

Figure 1: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with Bis-SNP from recalibrated alignments. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP and QUAL we removed values outside the 99th percentile (104 and 768.61, respectively) for better visualization. Black vertical line corresponds to the filter cutoff for minimal DP.

2.1.3.2 BS-SNPer The pipeline includes SNP calling, SNP filtering (including a diagnostics plot, **Figure 5**), and the evaluation of SNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/Bis-snp).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 4 --use-conda
```

2.1.3.3 CGmapTools The pipeline includes SNP calling using the Bayesian and binomial strategy, SNP filtering (including a diagnostics plot, **Figure 6** and **Figure 7**), and the evaluation of SNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole

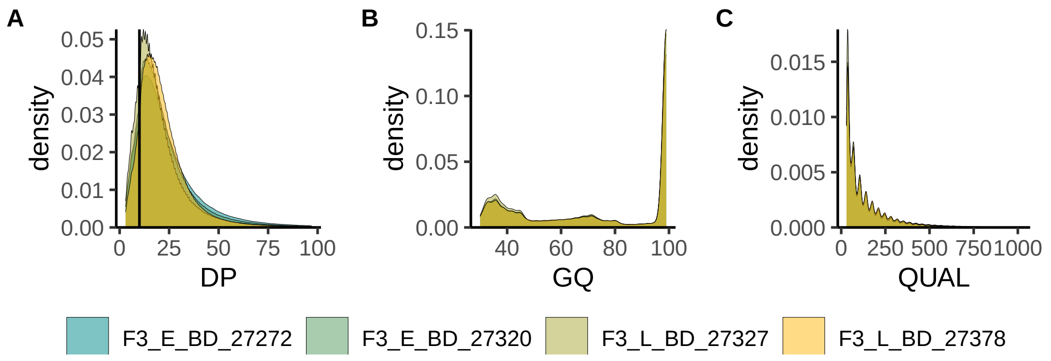

Figure 2: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with Bis-SNP from non-recalibrated alignments. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP and QUAL we removed values outside the 99th percentile (98 and 1,020.40, respectively) for better visualization. Black vertical line corresponds to the filter cutoff for minimal DP.

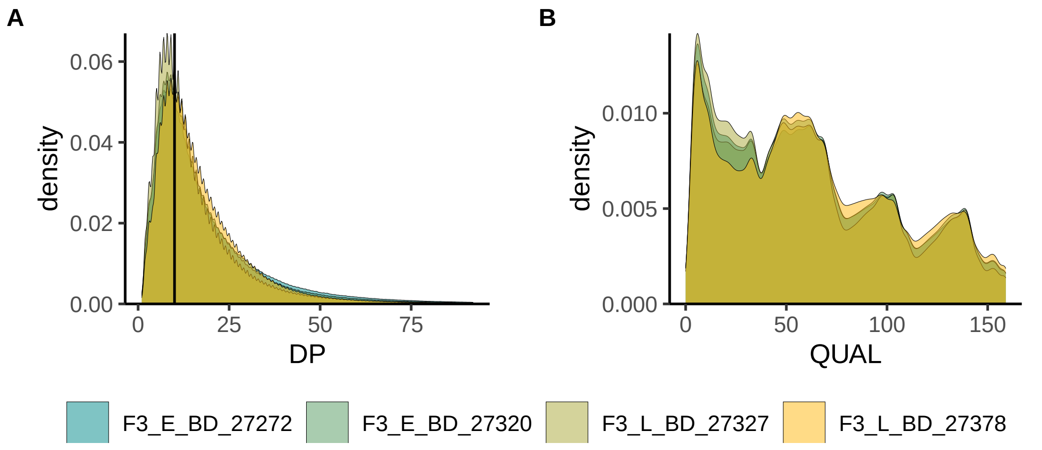

Figure 3: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with BS-SNPer. **A** Depth per SNP (DP), and **B** quality of variant. For DP and QUAL we removed values outside the 99th percentile (93 and 1,000 respectively) for better visualization. Please note that the next highest QUAL value after 1000 is 159. Black vertical line corresponds to the filter cutoff for minimal DP.

genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/CGmaptools).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 8 --use-conda
```

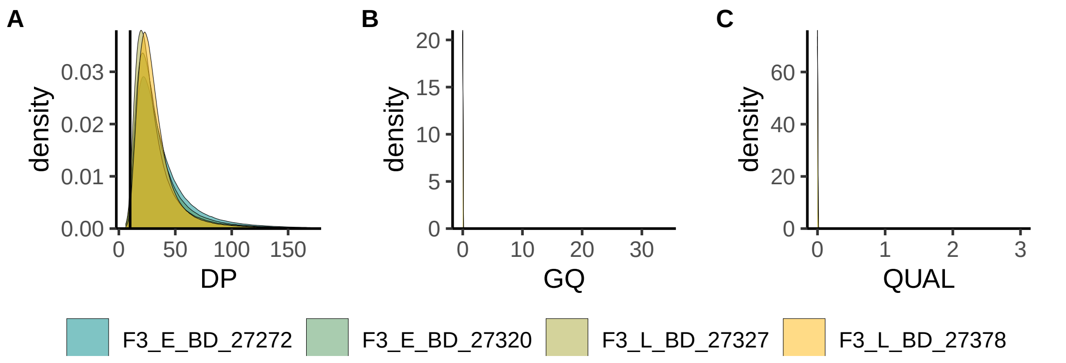

Figure 4: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with CGmapTools' Bayesian strategy. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP we removed values outside the 99th percentile (172) for better visualization. Black vertical line corresponds to the filter cutoff for minimal DP.

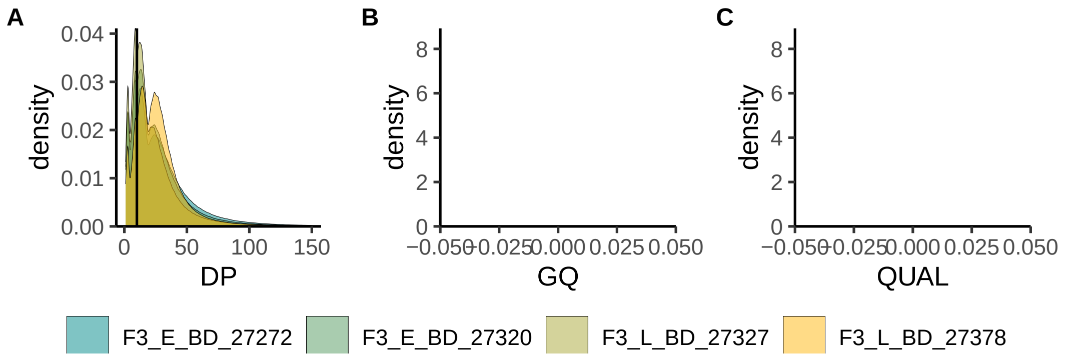

Figure 5: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with CGmapTools' binomial strategy. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP we removed values outside the 99th percentile (151) for better visualization. Black vertical line corresponds to the filter cutoff for minimal DP.

2.1.3.4 EpiDiverse-SNP pipeline

In contrast to the other tools, the EpiDiverse-SNP pipeline does not constitute a bisulfite sequencing data specific SNP caller, but rather involves a tool for double-masking of the alignments to make them compatible with conventional SNP caller. The EpiDiverse-SNP pipeline is executed using next flow. Subsequently, a snakemake pipeline is used to filter SNPs (including a diagnostics plot, **Figure 8**) and evaluate the SSNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/epidiverse).

```
# run EpiDiverse-SNP pipeline using nextflow
NXF_VER=20.07.1 nextflow run epidiverse/snp -profile conda
NXF_VER=20.07.1 nextflow run epidiverse/snp -profile conda --input alignments --
    reference ../../wgbs_snakemake_reseq/genome/reference.fa --output SNP_calls --
    variants

# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
```

```
# run pipeline
snakemake -j 4 --use-conda
```

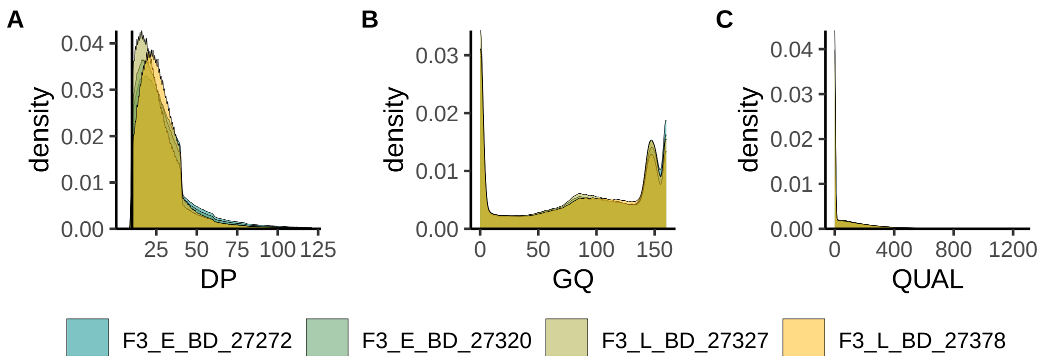

Figure 6: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with EpiDiverse-SNP pipeline. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP and QUAL we removed values outside the 99th percentile (122 and 1,0252.06 respectively) for better visualization. Black vertical line corresponds to the filter cutoff for minimal DP.

2.1.3.5 MethylExtract The pipeline includes SNP calling, SNP filtering (including a diagnostics plot, **Figure 9**), and the evaluation of SNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/MethylExtract).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 32 --use-conda
```

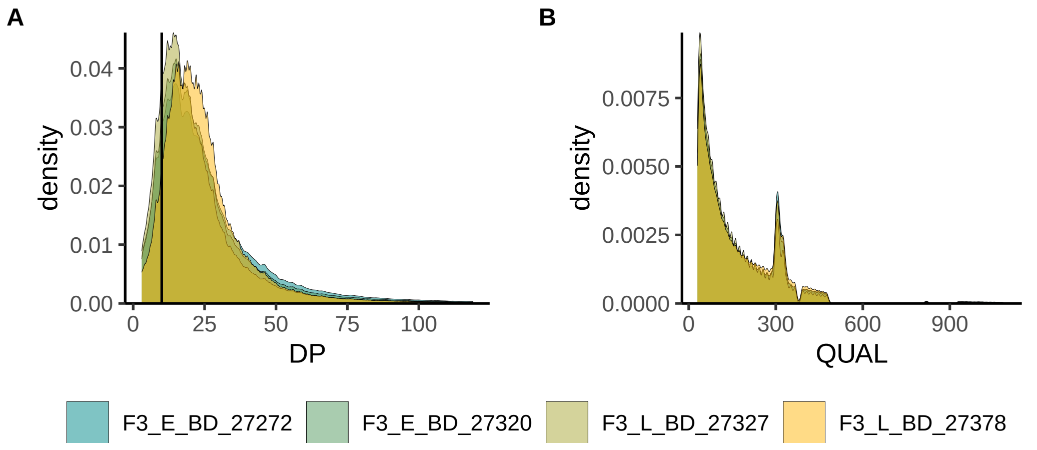

Figure 7: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with MethylExtract. **A** Depth per SNP (DP), and **B** quality of variant. For DP and QUAL we removed values outside the 99th percentile (93 and 1,000 respectively) for better visualization. Black vertical line corresponds to the filter cutoff for minimal DP.

2.1.3.6 biscuit In contrast to most other tools, biscuit (as well as gemBS, see below) is a 'whole-pipeline' tool that includes the alignment, deduplication, addition of sample-specific read groups, merging of alignments of the two sequencing runs for each sample, assessment of the number of mapped reads, average coverage depth, and breadth of coverage, the removal of reads mapping to the Z chromosome or mitochondrial DNA, SNP calling, SNP filtering (including a diagnostics plot, **Figure 10**), and the evaluation of SNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/biscuit).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 32 --use-conda
```

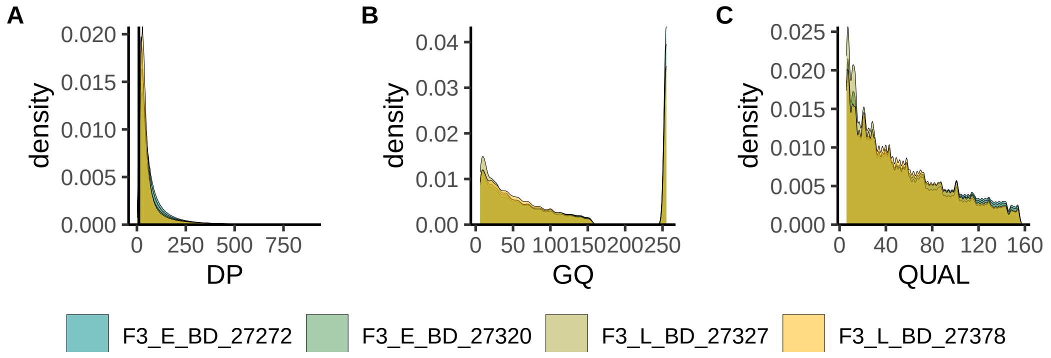

Figure 8: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with the biscuit pipeline. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP and QUAL we removed values outside the 99th percentile (898 and 255, respectively) for better visualization. Please note that the next highest QUAL value after 255 is 157. Black vertical line corresponds to the filter cutoff for minimal DP.

2.1.3.7 gemBS Like biscuit, gemBS is a 'whole-pipeline' tool. GemBS requires a metadata file and a config file that together provide information on the samples and the gemBS pipeline parameters. First, we created new links to the trimmed seq data files and prepared the metadata file by running a custom R script (which can be found on gitHub; https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/gemBS. Creating the links in this way helps with merging the samples. Then, we wrote the config file (which can be found on gitHub; same link) to set the parameters for the gemBS pipeline.

```
# make new links to trimmed data
for f in *.gz; do ln -s /home/NIOO.INT/melaniel/projects/WGBS_Snakemake_SnpCalling/
    trimmed_data/seq2018/$f /home/NIOO.INT/melaniel/projects/
    WGBS_Snakemake_SnpCalling/trimmed_data/gemBS/seq2018_$f; done
for f in *.gz; do ln -s /home/NIOO.INT/melaniel/projects/WGBS_Snakemake_SnpCalling/
    trimmed_data/reseq2020/$f /home/NIOO.INT/melaniel/projects/
    WGBS_Snakemake_SnpCalling/trimmed_data/gemBS/reseq2020_$f; done

# prepare metadata-file
R -e Make_Metadata_gemBS.R
```

The snakemake pipeline includes the writing of a .json file (based on metadata and .config file), indexing of the reference genome, alignments, assessment of the number of mapped reads, average coverage depth, and breadth of coverage, SNP calling (inlcuding deduplication), removal of SNPs located within the Z chromosome or mitochondrial DNA, SNP filtering (including a diagnostics plot, **Figure 11**), and the evaluation of SNP calls using the baseline list of known SNPs derived from the whole-genome resequencing data (but see 'Evaluation of SNPs called from whole genome bisulfite sequencing data' below). The pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/wgbs/gemBS).

```
# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 20 --use-conda
```

3 SNPs from whole-genome resequencing data as baseline SNP list

3.1 Bioinformatics pipeline

3.1.1 Prepare raw data

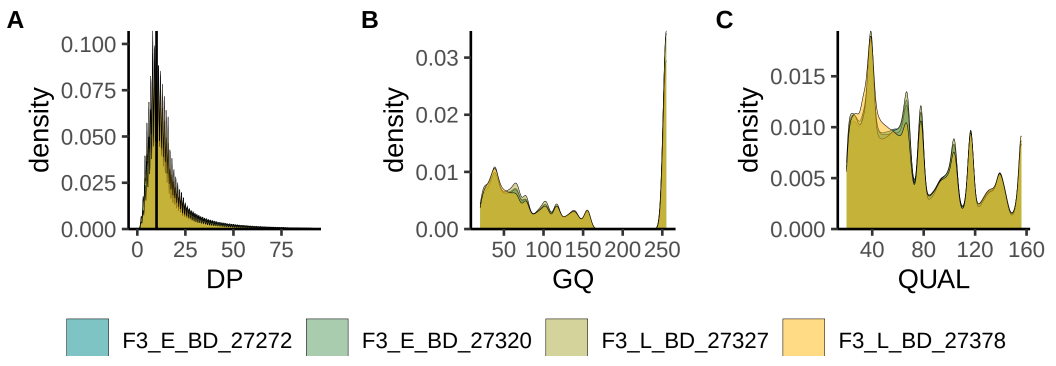

Figure 9: Diagnostics plots to determine filter thresholds for genotypes of SNPs called with the gemBS pipeline. **A** Depth per SNP (DP), **B** quality of genotypes (GQ), and **C** quality of variant. For DP and QUAL we removed values outside the 99th percentile (92 and 255 respectively) for better visualization. Please note that the next highest QUAL value after 255 is 156. Black vertical line corresponds to the filter cutoff for minimal DP.

```
# make links to raw data for four samples (R1 and R2 per sample)
ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27272/BD_27272_FDSW210077207-1
    r_HWGV3DSXY_L4_1.fq.gz raw_data/F3_E_BD_27272.R1.fq.gz
ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27272/BD_27272_FDSW210077207-1
    r_HWGV3DSXY_L4_2.fq.gz raw_data/F3_E_BD_27272.R2.fq.gz

ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27320/BD_27320_FDSW210077208-1
    r_HWGV3DSXY_L4_1.fq.gz raw_data/F3_E_BD_27320.R1.fq.gz
ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27320/BD_27320_FDSW210077208-1
    r_HWGV3DSXY_L4_2.fq.gz raw_data/F3_E_BD_27320.R2.fq.gz

ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27327/BD_27327_FDSW210077209-1
    r_HWGV3DSXY_L4_1.fq.gz raw_data/F3_L_BD_27327.R1.fq.gz
ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27327/BD_27327_FDSW210077209-1
    r_HWGV3DSXY_L4_2.fq.gz raw_data/F3_L_BD_27327.R2.fq.gz

ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27378/BD_27378_FDSW210077210-1
    r_HWGV3DSXY_L4_1.fq.gz raw_data/F3_L_BD_27378.R1.fq.gz
ln -s /data/primary/AnE/Parus_major/raw_data/whole_genome_sequencing/DNA_sequencing
    /illumina_novaseq/2021_ERCLayDate_WGBSBirds/BD_27378/BD_27378_FDSW210077210-1
    r_HWGV3DSXY_L4_2.fq.gz raw_data/F3_L_BD_27378.R2.fq.gz

# prepare list of known SNPs
cd get-known-snps
grep -vP '^#' known_SNPs_sorted.vcf  > vcf.body
grep -P '^#' known_SNPs_sorted.vcf | grep -vP '^##contig' > new.header.vcf

cat vcf.body >> new.header.vcf
gatk UpdateVCFSequenceDictionary -V new.header.vcf --source-dictionary ../genome/
    reference.dict --output known_snps.vcf --replace true
```

3.1.2 Quality control 1

```
# link to raw files in QC directory
```

```
for f in raw_data/*.gz; do ln -s /home/NIOO.INT/melaniel/projects/
    WGBS_Snakemake_reseq/$f /home/NIOO.INT/melaniel/projects/WGBS_Snakemake_reseq/QC
    -NGS/data/raw/ ; done

# run fastqc
fastqc -o ./reports/raw/ -t 8 ./data/raw/*.fq.gz

# run fastq_screen
cd ./fastq_screen_v0.11.1
./fastq_screen --outdir ../reports/raw-screen/ ../data/raw/*.fq.gz
cd ..

# run multiqc
multiqc -d -n report_all ./reports
```

3.1.3 Run the SNP calling pipeline

Our pipeline follows the GATK best practice for model (https://currentprotocols.onlinelibrary.wiley.com/doi/10.1002/0471250953.bi1110s43) and non-model organisms (https://evodify.com/gatk-in-non-model-organism/) and included quality control, data trimming, alignment, recalibration of base quality (BQ) scores, variant calling, and variant filtering (including diagnostics plots, **Figure 1** and **Figure 2**) and was executed such that samples were processed in parallel where applicable ('-j 4' option in 'snakemake' call).

The Snakefile and other files needed to run the pipeline can be found on gitHub (https://github.com/MLindner0/lindner_et_al-2021-mer-snps_from_bs_data/tree/master/reseq).

```
# link to genome:
ln -s /home/nioo/melaniel/projects/GenomePM/GCF_001522545.3/GCF_001522545.3
    _Parus_major1.1_genomic.fa genome/reference.fa

# check whether pipeline is ok by doing a dry run
snakemake -n --use-conda
# run pipeline
snakemake -j 4 --use-conda

# quality control after trimming (fastqc output produced while trimming as part of
    the pipeline)
# move files to right location
for f in trimmed_data/*.zip; do ln -s /home/NIOO.INT/melaniel/projects/
    WGBS_Snakemake_reseq/$f /home/NIOO.INT/melaniel/projects/WGBS_Snakemake_reseq/QC
    -NGS/reports/clean/ ; done
for f in trimmed_data/*.html; do ln -s /home/NIOO.INT/melaniel/projects/
    WGBS_Snakemake_reseq/$f /home/NIOO.INT/melaniel/projects/WGBS_Snakemake_reseq/QC
    -NGS/reports/clean/ ; done

# run multiqc
multiqc -d -n report_all_clean ./reports/clean
```

4 Evaluation of SNPs called from whole genome bisulfite sequencing data

For the evaluate the tools for SNP calling from bisulfite sequencing data, we compared the SNPs called with those tools to a baseline lists of known SNPs (i.e SNPs called from whole-genome resequencing data) using rtgTools (https://github.com/RealTimeGenomics/rtg-tools). RtgTools provides standard accuracy metrics such as the number of false positives (SNPs called that are not in the baseline list of true SNPs), false negatives (SNPs in the baseline list of true SNPs that are not called), true positives (SNPs called that are in the baseline list of true SNPs), precision, sensitivity, and the f-measure (i.e. the harmonic mean of precision and sensitivity). While precision is calculated as the number of SNPs called divided by the sum of the number of SNPs called and the number of

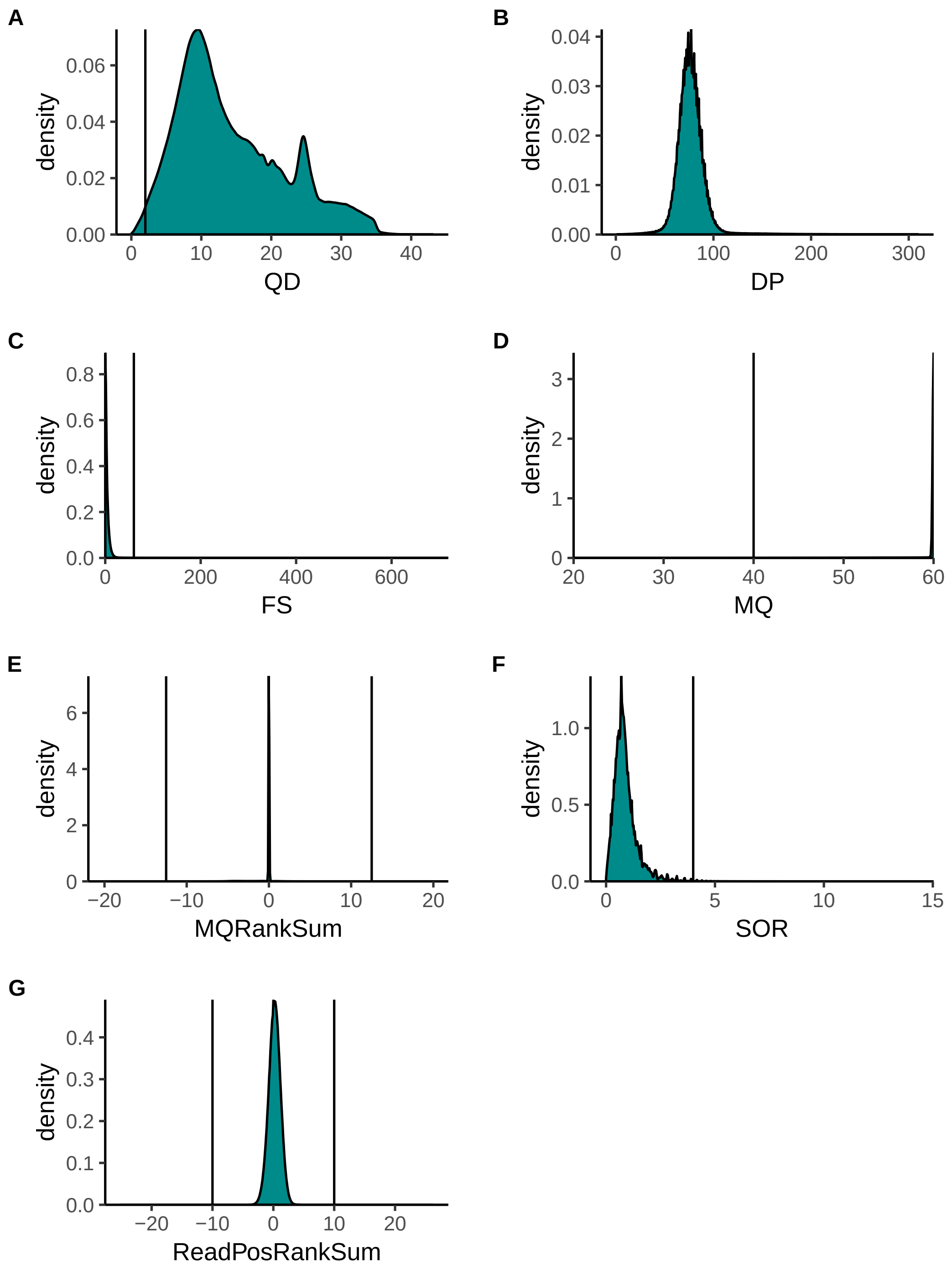

Figure 10: Diagnostics plots to determine filter thresholds for SNPs. **A** Variant confidence standardized by depth (QD), **B** Combined depth per SNP across samples (4 samples, DP), **C** Strand bias in support for REF vs ALT allele calls (FS), **D** Mapping quality of a SNP, **E** Rank sum test for mapping qualities of REF vs. ALT reads (MQRankSum), **F** sequencing bias in which one DNA strand is favored over the other (SOR), and **G** rank sum of read position (i.e. are all SNPs located near the end of SNPs, ReadPosRankSum)

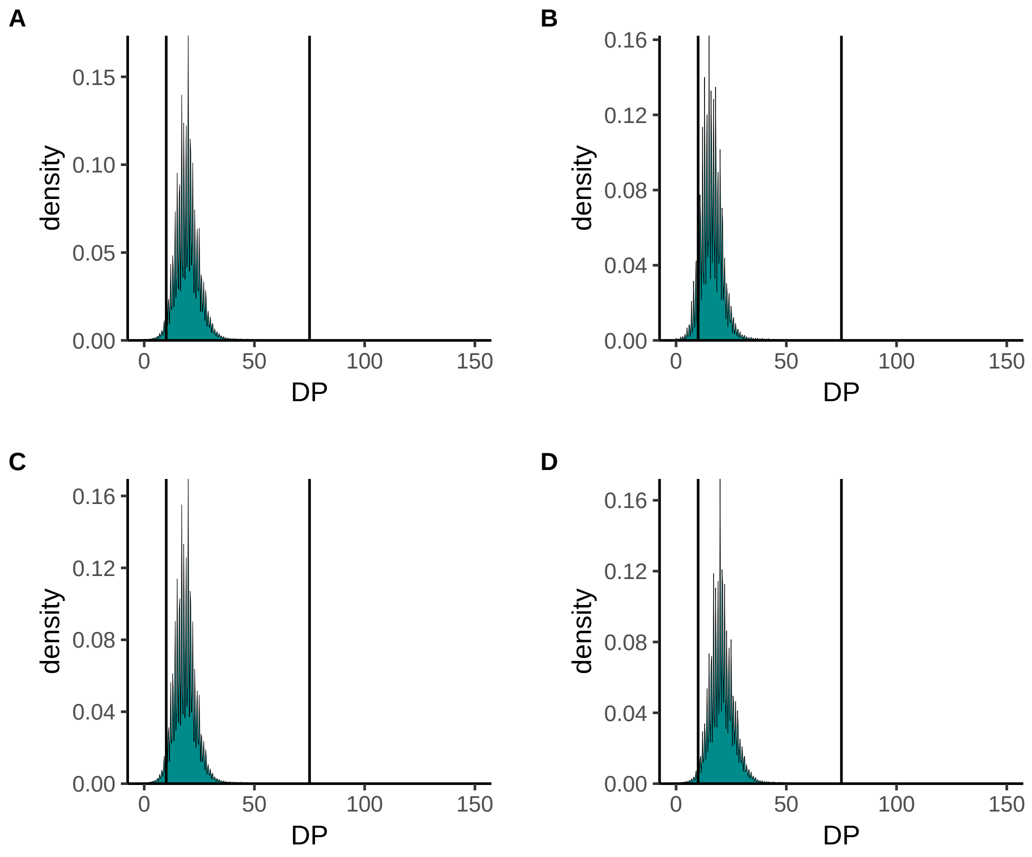

Figure 11: Diagnostics plots to determine filter thresholds for genotypes of SNPs. Depth per SNP (DP) within samples (**A**-**D**). Note, that we filtered for the quality of genotypes (GQ) during the genotyping by setting a minimum phred-scaled confidence threshold of 30 for genotyping of variants.

false-positive SNPs, sensitivity is calculated as the number of SNPs in the baseline lists of true SNPs divided by the sum of the number of SNPs in the baseline lists of true SNPs and the number of false-negative SNPs called. The execution of rtgTools is inlcuded in the tool-specific snakemake pipelines.

We evaluated and visualized the output of rtgTools, using R. This includes assessing the relationship of precision and sensitivity across the parameter value range of QUAL and GQ (**Figure 12** and **Figure 13**), number of false negative, false positive, and true positive SNPs called when f-measure (**Figure 14**), sensitivity (**Figure 15**), or precision (**Figure 16**) is maximized, and comparing the distribution of false negatives (**Figure 17**) and false positives SNPs (**Figure 18**) over substitution context in comparison to the distribution of baseline SNPs over substitution context. We furthermore, calculated the percentage of false positive SNPs called as heterozygous SNPs (**Figure 19**, **Figure 20**, **Figure 21** and **Figure 22**).

4.1 Relationship between precision and sensitivity

```
## Load and format data get file name and path of all output
## files:
Tools <- c("Bis-snp", "biscuit", "BS-snper", "CGmaptools", "epidiverse",
    "gemBS", "MethylExtract")
FileNames <- NULL
for (f in 1:length(Tools)) {
    Tool <- Tools[f]
    FileLocation <- paste(Tool, list.files(path = Tool, pattern = "RTG*"),
        sep = "/")
    FileName <- paste(FileLocation, list.files(path = FileLocation,
        pattern = "weighted_roc.tsv"), sep = "/")
    FileNames <- c(FileNames, FileName)
}

FileList <- lapply(FileNames, function(x) read.csv(x, header = F,
    sep = "\t", comment.char = "#"))

# data read ok?
lapply(FileList, function(x) dim(x))
lapply(FileList, function(x) head(x))

## format data:

# make helper for good tool names
Tools_good_name <- c("Bis-SNP", "biscuit", "BS-SNPer", "CGmapTools",
    "EpiDiverse", "gemBS", "MethylExtract")
ToolNamesHelper <- data.frame(Tools1 = Tools, Tools2 = Tools_good_name)

RTG_out_short <- NULL
RTG_F_max <- NULL
RTG_S_max <- NULL
RTG_P_max <- NULL

for (i in 1:length(FileList)) {
    helper <- str_split_fixed(FileNames[i], "/", 3)
    Tool <- helper[, 1]
    ToolGood <- ToolNamesHelper[ToolNamesHelper$Tools1 == Tool,
        2]
    helper2 <- helper[, 2]
    if (Tool == "Bis-snp" | Tool == "CGmaptools") {
        ToolGood <- paste(ToolGood, str_split_fixed(helper[,
            2], "\\.", 2)[, 2], sep = "_")
        helper2 <- str_split_fixed(helper[, 2], "\\.", 2)[, 1]
    }
    Parameter <- str_split_fixed(helper2, "_", 6)[, 2]
    helper3 <- str_split_fixed(helper2, "_", 6)
    Sample <- paste(helper3[, 3], helper3[, 4], helper3[, 5],
        helper3[, 6], sep = "_")

    # Add info on Sample, Tool, and Score parameter to data
    Data <- as.data.frame(FileList[i])
    names(Data) <- c("score", "true_positives_baseline", "false_positives",
        "true_positives_call", "false_negatives", "precision",
        "sensitivity", "f_measure")
    Data$Sample <- Sample
    Data$Tool <- ToolGood
    Data$P_Score <- Parameter
    Data$Tool <- sub("_newV", "", Data$Tool)
    Data$Tool <- sub("_NoRecal", "␣NoRecal", Data$Tool)
    Data$Tool <- sub("_bayes.dynamicP", "␣Bayesian", Data$Tool)
    Data$Tool <- sub("_binom", "␣binomial", Data$Tool)

    # reduce data when too small paramter intervals
    helper4 <- round(seq(1, nrow(Data), length.out = 10))
    Data_short <- Data[helper4, ]
    if (nrow(Data) < 10)
        Data_short <- Data
    RTG_out_short <- rbind(RTG_out_short, Data_short)

    # get maximum F statistic:
    Max <- Data[Data$f_measure == max(Data$f_measure), ]
    if (nrow(Max) > 1)
        Max <- Max[which.max(Max$true_positives_baseline), ]
    if (nrow(Data) == 1)
        Max <- Data
    RTG_F_max <- rbind(RTG_F_max, Max)

    # get maximum sensitivity:
    Max <- Data[Data$sensitivity == max(Data$sensitivity), ]
    if (nrow(Max) > 1)
        Max <- Max[which.max(Max$true_positives_baseline), ]
    if (nrow(Data) == 1)
        Max <- Data
    RTG_S_max <- rbind(RTG_S_max, Max)

    # get maximum precision:
    helper_P <- Data[Data$true_positives_call > 1e+06, ]
    Max <- helper_P[helper_P$precision == max(helper_P$precision),
        ]
    if (nrow(Max) > 1)
        Max <- Max[which.max(Max$true_positives_baseline), ]
    if (nrow(Data) == 1)
        Max <- Data
    RTG_P_max <- rbind(RTG_P_max, Max)
}

# Prepare data for Supp Mat:
RTG_short_out <- RTG_out_short[, c(9:11, 1, 8, 6, 7, 4, 2, 3,
    5)] %>%
    arrange(Tool, Sample, P_Score) %>%
    na_if("None")
RTG_F_max_out <- RTG_F_max[, c(9:11, 1, 8, 6, 7, 4, 2, 3, 5)] %>%
    arrange(Tool, Sample, P_Score) %>%
    na_if("None")
RTG_S_max_out <- RTG_S_max[, c(9:11, 1, 8, 6, 7, 4, 2, 3, 5)] %>%
    arrange(Tool, Sample, P_Score) %>%
    na_if("None")
RTG_P_max_out <- RTG_P_max[, c(9:11, 1, 8, 6, 7, 4, 2, 3, 5)] %>%
    arrange(Tool, Sample, P_Score) %>%
    na_if("None")

write.table(RTG_short_out, "out/RTG_short_out", quote = F, sep = "\t",
    col.names = TRUE, row.names = FALSE)
write.table(RTG_F_max_out, "out/RTG_F_max_out", quote = F, sep = "\t",
    col.names = TRUE, row.names = FALSE)
write.table(RTG_S_max_out, "out/RTG_S_max_out", quote = F, sep = "\t",
    col.names = TRUE, row.names = FALSE)
write.table(RTG_P_max_out, "out/RTG_P_max_out", quote = F, sep = "\t",
    col.names = TRUE, row.names = FALSE)

## Plot precision vs. sensitivity across parameter value range
## of QUAL and GQ A add more colors to help differentiate
## tools
colfunc <- colorRampPalette(c("gray40", "darkcyan", "mediumseagreen",
    "darkgoldenrod1", "khaki"))
col <- colfunc(9)

# prepare legend
data_temp_GQ <- RTG_out_short[RTG_out_short$P_Score == "GQ",
    ]
ToolNames <- unique(data_temp_GQ$Tool)
shape <- c(21, 22, 23, 24, 25, 21, 22, 23, 24)

Plot <- data_temp_GQ %>%
    filter(Sample == "F3_E_BD_27272") %>%
    mutate(Tool = factor(Tool, levels = ToolNames)) %>%
    ggplot() + geom_point(aes(x = sensitivity, y = precision,
    fill = Tool, shape = Tool), col = "black", size = 3) + scale_fill_manual(name =
        "",
    values = col, labels = ToolNames) + scale_color_manual(name = "",
    values = col, labels = ToolNames) + scale_shape_manual(name = "",
    values = shape, labels = ToolNames) + scale_y_continuous(expand = c(0,
    0)) + theme_classic() + theme(axis.text.x = element_text(size = 14),
    axis.text.y = element_text(size = 14), axis.title.x = element_text(size = 16,
        margin = margin(t = 5, r = 0, b = 0, l = 0)), axis.title.y = element_text(
            size = 16,
        margin = margin(t = 0, r = 10, b = 0, l = 0)), legend.position = "right",
    legend.text = element_text(size = 12), legend.key.size = unit(0.8,
        "cm"))
legend <- get_legend(Plot)

# Make GQ plots for all samples
Samples <- unique(RTG_out_short$Sample)
Plots <- NULL

for (i in 1:length(Samples)) {
    Plot <- data_temp_GQ %>%
        filter(Sample == Samples[i]) %>%
        mutate(Tool = factor(Tool, levels = ToolNames)) %>%
        ggplot() + geom_point(aes(x = sensitivity, y = precision,
        fill = Tool, shape = Tool), col = "black", size = 3) +
        scale_fill_manual(name = "", values = col, labels = ToolNames) +
        scale_color_manual(name = "", values = col, labels = ToolNames) +
        scale_shape_manual(name = "", values = shape, labels = ToolNames) +
        scale_x_continuous(limits = c(0, 1), breaks = seq(0.1,
            0.9, length.out = 5)) + scale_y_continuous(limits = c(0,
        1), breaks = seq(0.1, 0.9, length.out = 5)) + theme_classic() +
        theme(axis.text.x = element_text(size = 14), axis.text.y = element_text(
            size = 14),
            axis.title.x = element_text(size = 16, margin = margin(t = 5,
                r = 0, b = 0, l = 0)), axis.title.y = element_text(size = 16,
                margin = margin(t = 0, r = 10, b = 0, l = 0)),
            legend.position = "none")
    Name <- paste("plot", Samples[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}

# make figure for Supp Mat (Figure S13)
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), labels = "AUTO", label_size = 14, scale = 0.9,
    ncol = 2)
Plot_legend <- plot_grid(Plot, legend, rel_widths = c(7, 3),
    nrow = 1)
save_plot("Plots/SensPrec_GQ.pdf", Plot_legend, base_height = 6,
    base_width = 9)

# Keep GQ plot of first samples for main figure
keep <- get(Plots[[1]])

# Make QUAL plots for all samples
data_temp_QUAL <- RTG_out_short[RTG_out_short$P_Score == "QUAL",
    ]
Plots <- NULL

for (i in 1:length(Samples)) {
    Plot <- data_temp_QUAL %>%
        filter(Sample == Samples[i]) %>%
        mutate(Tool = factor(Tool, levels = ToolNames)) %>%
        ggplot() + geom_point(aes(x = sensitivity, y = precision,
        fill = Tool, shape = Tool), col = "black", size = 3) +
        scale_fill_manual(name = "", values = col, labels = ToolNames) +
        scale_color_manual(name = "", values = col, labels = ToolNames) +
        scale_shape_manual(name = "", values = shape, labels = ToolNames) +
        scale_x_continuous(limits = c(0, 1), breaks = seq(0.1,
        0.9, length.out = 5)) + scale_y_continuous(limits = c(0,
        1), breaks = seq(0.1, 0.9, length.out = 5)) + theme_classic() +
        theme(axis.text.x = element_text(size = 14), axis.text.y = element_text(
            size = 14),
            axis.title.x = element_text(size = 16, margin = margin(t = 5,
            r = 0, b = 0, l = 0)), axis.title.y = element_text(size = 16,
            margin = margin(t = 0, r = 10, b = 0, l = 0)),
            legend.position = "none")
    Name <- paste("plot", Samples[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}

# make figure for Supp Mat (Figure S12)
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), labels = "AUTO", label_size = 14, scale = 0.9,
    ncol = 2)
Plot_legend <- plot_grid(Plot, legend, rel_widths = c(7, 3),
    nrow = 1)
save_plot("Plots/SensPrec_QUAL.pdf", Plot_legend, base_height = 6,
    base_width = 9)
```

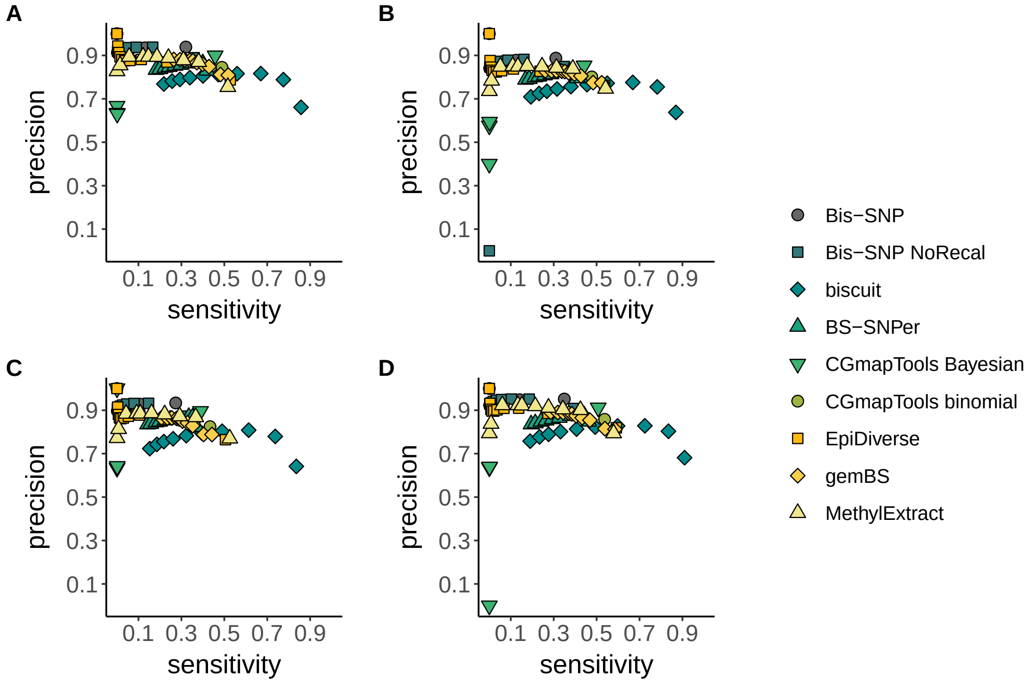

Figure 12: Relationship between precision and sensitivity for SNPs called from whole genome bisulfite sequencing data of four samples (**A-D**) relative to a list of known SNPs derived from whole-genome resequencing data of the respective same sample. Precision and sensitivity were calculated using rtgTools with QUAL as score fields which means that the accuracy metrics (here precision and sensitivity) were calculated across the full range of a parameter values for QUAL. Thus, the number of data points per tool, varies with the tool-specific and parameter-specific range of parameter values. If the parameter value is not given, the performance metrics are calculated for the full SNP list (resulting in one data point) and if the full range of a parameter value is longer than 20 values, we reduced the length of a parameter range to 20 equally spaced values across the full range of parameter values.

4.2 Number of SNPs when performance metrics are maximized

```
## Number of SNPs (true positives/false positives/false
## negatives) at maximal f-measure
colfunc <- colorRampPalette(c("darkcyan", "khaki"))
col <- colfunc(3)

# prepare labels
RTG_F_max_QUAL <- RTG_F_max %>%
    filter(P_Score == "QUAL")
ToolNames <- unique(RTG_F_max_QUAL$Tool)
Names_t <- sub("CGmapTools", "CG", ToolNames)
Names <- sub("NoRecal", "NR", Names_t)

Type <- names(RTG_F_max_QUAL[3:5])
Type_order <- Type[c(3, 1, 2)]
TypeNames_help <- sub("_call", "", Type_order)
TypeNames <- sub("_", "␣", TypeNames_help)

# make legend
ForLegend_help <- RTG_F_max_QUAL %>%
    filter(Sample == "F3_E_BD_27272")
```

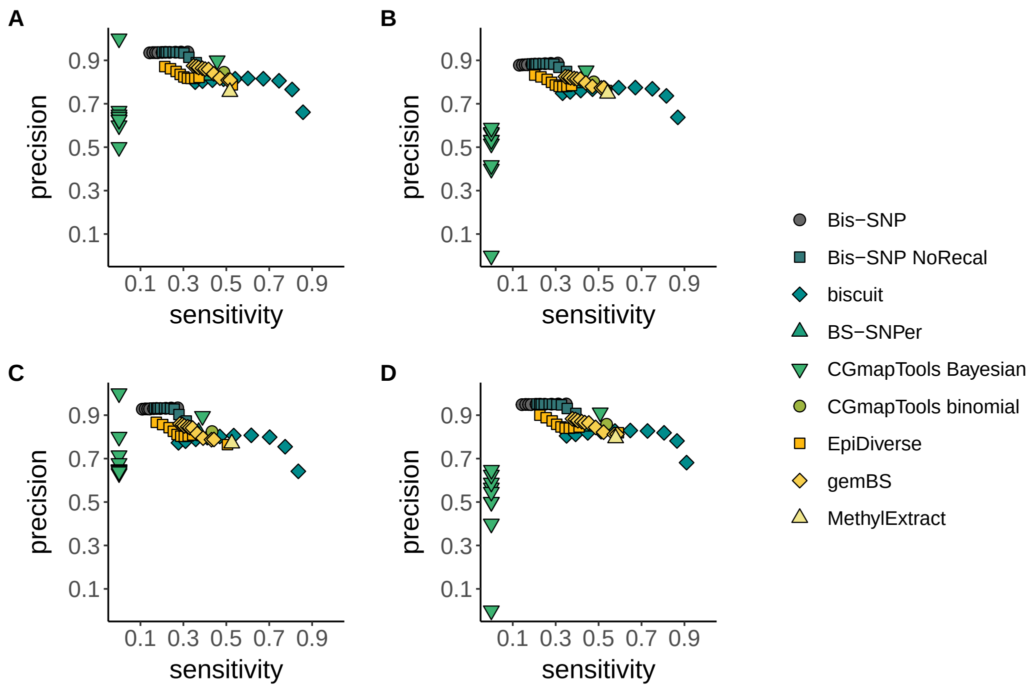

Figure 13: Relationship between precision and sensitivity for SNPs called from whole genome bisulfite sequencing data of four samples (**A**-**D**) relative to a list of known SNPs derived from whole-genome resequencing data of the respective same sample. Precision and sensitivity were calculated using rtgTools with GQ as score fields which means that the accuracy metrics (here precision and sensitivity) were calculated across the full range of a parameter values for GQ. Thus, the number of data points per tool, varies with the tool-specific and parameter-specific range of parameter values. If the parameter value is not given, the performance metrics are calculated for the full SNP list (resulting in one data point) and if the full range of a parameter value is longer than 20 values, we reduced the length of a parameter range to 20 equally spaced values across the full range of parameter values.

```
ForLegend <- gather(ForLegend_help[, c(3:5, 10)], names(ForLegend_help[3:5]),
    key = "Type", value = "Number")

Plot <- ForLegend %>%
    mutate(Tool = factor(Tool, levels = ToolNames)) %>%
    mutate(Type = factor(Type, levels = Type_order)) %>%
    ggplot() + geom_bar(aes(x = Tool, y = Number, fill = Type,
    color = Type), stat = "identity", alpha = 1, width = 0.8) +
    ylab("number␣of␣SNPs") + scale_fill_manual(name = "", values = col,
    labels = TypeNames) + scale_color_manual(name = "", values = col,
    labels = TypeNames) + scale_x_discrete(labels = Names) +
    # scale_y_continuous(breaks=seq(5,20,5), limits=c(0,23),
# expand=c(0,0), labels=paste(seq(5,20,5), 'M', sep=' ')) +
theme_classic() + theme(axis.text.x = element_text(size = 14,
    angle = 90), axis.text.y = element_text(size = 14), axis.title.x = element_
        blank(),
    axis.title.y = element_text(size = 16, margin = margin(t = 0,
        r = 10, b = 0, l = 0)), axis.line = element_line(colour = "grey50"),
    legend.text = element_text(size = 14), legend.key.size = unit(0.8,
        "cm"), legend.position = "bottom")
legend <- get_legend(Plot)

# get samples
Samples <- unique(RTG_out_short$Sample)

# prepare scaling of 2nd axis
ylim_1 <- c(0, 6)
ylim_2 <- c(0, 1)
b <- diff(ylim_1)/diff(ylim_2)
a <- b * (ylim_1[1] - ylim_2[1])

# make plot for each sample
Plots <- NULL
for (i in 1:length(Samples)) {
    ForPlot_help <- RTG_F_max_QUAL %>%
        filter(Sample == Samples[i])
    ForPlot <- gather(ForPlot_help[, c(3:5, 10)], names(ForPlot_help[3:5]),
        key = "Type", value = "Number")
    ForPlot$NumberMil <- ForPlot$Number/1e+06
    ForPlotb <- ForPlot_help[, c(8, 10)]

    # x labs only for bottom plots
    if (i == 1 | i == 2) {
        Plot <- ForPlot %>%
            mutate(Tool = factor(Tool, levels = ToolNames)) %>%
            mutate(Type = factor(Type, levels = Type_order)) %>%
            ggplot() + geom_bar(aes(x = Tool, y = NumberMil,
            fill = Type, color = Type), stat = "identity", alpha = 1,
            width = 0.8) + geom_point(data = ForPlotb, aes(x = Tool,
            y = f_measure * b), col = "black", fill = "white",
            shape = 23, size = 3) + ylab("number␣of␣SNPs") +
            scale_fill_manual(name = "", values = col, labels = TypeNames) +
            scale_color_manual(name = "", values = col, labels = TypeNames) +
            scale_x_discrete(labels = Names) + scale_y_continuous(breaks = seq(1,
            5, 1), limits = c(0, 6), expand = c(0, 0), labels = paste(seq(1,
            5, 1), "M", sep = "␣"), sec.axis = sec_axis(~./b,
            name = "f-measure", breaks = seq(0.2, 0.8, 0.2))) +
            theme_classic() + theme(axis.text.x = element_blank(),
            axis.text.y = element_text(size = 14), axis.title.x = element_blank(),
            axis.title.y = element_text(size = 16, margin = margin(t = 0,
                r = 10, b = 0, l = 0)), axis.title.y.right = element_text(size =
                    16,
                margin = margin(t = 0, r = , b = 0, l = 10),
                angle = 90), legend.position = "none")
    } else {
        Plot <- ForPlot %>%
            mutate(Tool = factor(Tool, levels = ToolNames)) %>%
            mutate(Type = factor(Type, levels = Type_order)) %>%
            ggplot() + geom_bar(aes(x = Tool, y = NumberMil,
            fill = Type, color = Type), stat = "identity", alpha = 1,
            width = 0.8) + geom_point(data = ForPlotb, aes(x = Tool,
            y = f_measure * b), col = "black", fill = "white",
            shape = 23, size = 3) + ylab("number␣of␣SNPs") +
            scale_fill_manual(name = "", values = col, labels = TypeNames) +
            scale_color_manual(name = "", values = col, labels = TypeNames) +
            scale_x_discrete(labels = Names) + scale_y_continuous(breaks = seq(1,
            5, 1), limits = c(0, 6), expand = c(0, 0), labels = paste(seq(1,
            5, 1), "M", sep = "␣"), sec.axis = sec_axis(~./b,
            name = "f-measure", breaks = seq(0.2, 0.8, 0.2))) +
            theme_classic() + theme(axis.text.x = element_text(size = 14,
            angle = 90, vjust = 0.5, hjust = 0, color = "black"),
            axis.text.y = element_text(size = 14), axis.title.x = element_blank(),
            axis.title.y = element_text(size = 16, margin = margin(t = 0,
                r = 10, b = 0, l = 0)), axis.title.y.right = element_text(size =
                    16,
                margin = margin(t = 0, r = , b = 0, l = 10),
                angle = 90), legend.position = "none")
    }
    Name <- paste("plot", Samples[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}
# make figure for Supp Mat (Figure S14)
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), rel_heights = c(3, 4.5), labels = "AUTO",
    label_size = 10, scale = 0.9, ncol = 2)
Plot_legend <- plot_grid(Plot, legend, rel_heights = c(10, 1),
    nrow = 2)
save_plot("Plots/Number_SNPs_MaxF.pdf", Plot_legend, base_height = 6,
    base_width = 10)

# Keep GQ plot of first samples for main figure
kepp_F <- get(Plots[[1]])


## Number of SNPs (true positives/false positives/false
## negatives) at maximal sensivity
colfunc <- colorRampPalette(c("darkcyan", "khaki"))
col <- colfunc(3)

RTG_S_max_QUAL <- RTG_S_max %>%
    filter(P_Score == "QUAL")
Samples <- unique(RTG_out_short$Sample)

# prepare scaling of 2nd axis
ylim_1 <- c(0, 7)
ylim_2 <- c(0, 1)
b <- diff(ylim_1)/diff(ylim_2)
```

```
a <- b * (ylim_1[1] - ylim_2[1])

# make plot for each sample
Plots <- NULL
for (i in 1:length(Samples)) {
    ForPlot_help <- RTG_S_max_QUAL %>%
        filter(Sample == Samples[i])
    ForPlot <- gather(ForPlot_help[, c(3:5, 10)], names(ForPlot_help[3:5]),
        key = "Type", value = "Number")
    ForPlot$NumberMil <- ForPlot$Number/1e+06

    ForPlotb <- ForPlot_help[, c(7, 10)]

    # x labs only for bottom plots
    if (i == 1 | i == 2) {
        Plot <- ForPlot %>%
            mutate(Tool = factor(Tool, levels = ToolNames)) %>%
            mutate(Type = factor(Type, levels = Type_order)) %>%
            ggplot() + geom_bar(aes(x = Tool, y = NumberMil,
            fill = Type, color = Type), stat = "identity", alpha = 1,
            width = 0.8) + geom_point(data = ForPlotb, aes(x = Tool,
            y = sensitivity * b), col = "black", fill = "white",
            shape = 23, size = 3) + ylab("number␣of␣SNPs") +
            scale_fill_manual(name = "", values = col, labels = TypeNames) +
            scale_color_manual(name = "", values = col, labels = TypeNames) +
            scale_x_discrete(labels = Names) + scale_y_continuous(breaks = seq(1,
            6, 1), limits = c(0, 7), expand = c(0, 0), labels = paste(seq(1,
            6, 1), "M", sep = "␣"), sec.axis = sec_axis(~./b,
            name = "sensitivity", breaks = seq(0.2, 0.8, 0.2))) +
            theme_classic() + theme(axis.text.x = element_blank(),
            axis.text.y = element_text(size = 14), axis.title.x = element_blank(),
            axis.title.y = element_text(size = 16, margin = margin(t = 0,
                r = 10, b = 0, l = 0)), axis.title.y.right = element_text(size =
                    16,
                margin = margin(t = 0, r = , b = 0, l = 10),
                angle = 90), legend.position = "none")
    } else {
        Plot <- ForPlot %>%
            mutate(Tool = factor(Tool, levels = ToolNames)) %>%
            mutate(Type = factor(Type, levels = Type_order)) %>%
            ggplot() + geom_bar(aes(x = Tool, y = NumberMil,
            fill = Type, color = Type), stat = "identity", alpha = 1,
            width = 0.8) + geom_point(data = ForPlotb, aes(x = Tool,
            y = sensitivity * b), col = "black", fill = "white",
            shape = 23, size = 3) + ylab("number␣of␣SNPs") +
            scale_fill_manual(name = "", values = col, labels = TypeNames) +
            scale_color_manual(name = "", values = col, labels = TypeNames) +
            scale_x_discrete(labels = Names) + scale_y_continuous(breaks = seq(1,
            6, 1), limits = c(0, 7), expand = c(0, 0), labels = paste(seq(1,
            6, 1), "M", sep = "␣"), sec.axis = sec_axis(~./b,
            name = "sensitivity", breaks = seq(0.2, 0.8, 0.2))) +
            theme_classic() + theme(axis.text.x = element_text(size = 14,
            angle = 90, vjust = 0.5, hjust = 0, color = "black"),
            axis.text.y = element_text(size = 14), axis.title.x = element_blank(),
            axis.title.y = element_text(size = 16, margin = margin(t = 0,
                r = 10, b = 0, l = 0)), axis.title.y.right = element_text(size =
                    16,
                margin = margin(t = 0, r = , b = 0, l = 10),
                angle = 90), legend.position = "none")
    }
    Name <- paste("plot", Samples[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}
# make figure for Supp Mat (Figure S15)
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), rel_heights = c(3, 4.5), labels = "AUTO",
    label_size = 10, scale = 0.9, ncol = 2)
Plot_legend <- plot_grid(Plot, legend, rel_heights = c(10, 1),
    nrow = 2)
save_plot("Plots/Number_SNPs_MaxSens.pdf", Plot_legend, base_height = 6,
    base_width = 10)

# Keep GQ plot of first samples for main figure
kepp_S <- get(Plots[[1]])


## Number of SNPs (true positives/false positives/false
## negatives) at maximal precision
colfunc <- colorRampPalette(c("darkcyan", "khaki"))
col <- colfunc(3)

RTG_P_max_QUAL <- RTG_P_max %>%
    filter(P_Score == "QUAL")
Samples <- unique(RTG_out_short$Sample)

# prepare scaling of 2nd axis
ylim_1 <- c(0, 6)
ylim_2 <- c(0, 1)
b <- diff(ylim_1)/diff(ylim_2)
a <- b * (ylim_1[1] - ylim_2[1])

# make plot for each sample
Plots <- NULL
for (i in 1:length(Samples)) {
    ForPlot_help <- RTG_P_max_QUAL %>%
        filter(Sample == Samples[i])
    ForPlot <- gather(ForPlot_help[, c(3:5, 10)], names(ForPlot_help[3:5]),
        key = "Type", value = "Number")
    ForPlot$NumberMil <- ForPlot$Number/1e+06

    ForPlotb <- ForPlot_help[, c(6, 10)]

    # x labs only for bottom plots
    if (i == 1 | i == 2) {
        Plot <- ForPlot %>%
            mutate(Tool = factor(Tool, levels = ToolNames)) %>%
            mutate(Type = factor(Type, levels = Type_order)) %>%
            ggplot() + geom_bar(aes(x = Tool, y = NumberMil,
            fill = Type, color = Type), stat = "identity", alpha = 1,
            width = 0.8) + geom_point(data = ForPlotb, aes(x = Tool,
            y = precision * b), col = "black", fill = "white",
            shape = 23, size = 3) + ylab("number␣of␣SNPs") +
            scale_fill_manual(name = "", values = col, labels = TypeNames) +
            scale_color_manual(name = "", values = col, labels = TypeNames) +
            scale_x_discrete(labels = Names) + scale_y_continuous(breaks = seq(1,
            5, 1), limits = c(0, 6), expand = c(0, 0), labels = paste(seq(1,
            5, 1), "M", sep = "␣"), sec.axis = sec_axis(~./b,
            name = "precision", breaks = seq(0.2, 0.8, 0.2))) +
            theme_classic() + theme(axis.text.x = element_blank(),
            axis.text.y = element_text(size = 14), axis.title.x = element_blank(),
            axis.title.y = element_text(size = 16, margin = margin(t = 0,
                r = 10, b = 0, l = 0)), axis.title.y.right = element_text(size =
                    16,
                margin = margin(t = 0, r = , b = 0, l = 10),
                angle = 90), legend.position = "none")
    } else {
        Plot <- ForPlot %>%
            mutate(Tool = factor(Tool, levels = ToolNames)) %>%
            mutate(Type = factor(Type, levels = Type_order)) %>%
            ggplot() + geom_bar(aes(x = Tool, y = NumberMil,
            fill = Type, color = Type), stat = "identity", alpha = 1,
            width = 0.8) + geom_point(data = ForPlotb, aes(x = Tool,
            y = precision * b), col = "black", fill = "white",
            shape = 23, size = 3) + ylab("number␣of␣SNPs") +
            scale_fill_manual(name = "", values = col, labels = TypeNames) +
            scale_color_manual(name = "", values = col, labels = TypeNames) +
            scale_x_discrete(labels = Names) + scale_y_continuous(breaks = seq(1,
            6, 1), limits = c(0, 7), expand = c(0, 0), labels = paste(seq(1,
            6, 1), "M", sep = "␣"), sec.axis = sec_axis(~./b,
            name = "precision", breaks = seq(0.2, 0.8, 0.2))) +
            theme_classic() + theme(axis.text.x = element_text(size = 14,
            angle = 90, vjust = 0.5, hjust = 0, color = "black"),
            axis.text.y = element_text(size = 14), axis.title.x = element_blank(),
            axis.title.y = element_text(size = 16, margin = margin(t = 0,
                r = 10, b = 0, l = 0)), axis.title.y.right = element_text(size =
                    16,
                margin = margin(t = 0, r = , b = 0, l = 10),
                angle = 90), legend.position = "none")
    }
    Name <- paste("plot", Samples[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}
# make figure for Supp Mat (Figure S16)
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), rel_heights = c(3, 4.5), labels = "AUTO",
    label_size = 10, scale = 0.9, ncol = 2)
Plot_legend <- plot_grid(Plot, legend, rel_heights = c(10, 1),
    nrow = 2)
save_plot("Plots/Number_SNPs_MaxPrec.pdf", Plot_legend, base_height = 6,
    base_width = 10)

# Keep GQ plot of first samples for main figure
kepp_P <- get(Plots[[1]]) + theme(axis.text.x = element_text(size = 14,
    angle = 90, vjust = 0.5, hjust = 0, color = "black"))

# make figure for main (Figure 2; F-measure, sensitivity,
# precision of first sample)
Plot <- plot_grid(kepp_F, kepp_S, kepp_P, labels = "AUTO", label_size = 10,
    rel_heights = c(3, 3, 5), scale = 0.9, ncol = 1)
Plot_legend <- plot_grid(Plot, legend, rel_heights = c(10, 1),
    nrow = 2)
save_plot("Plots/Number_SNPs_1Sample.pdf", Plot_legend, base_height = 9,
    base_width = 6)
```

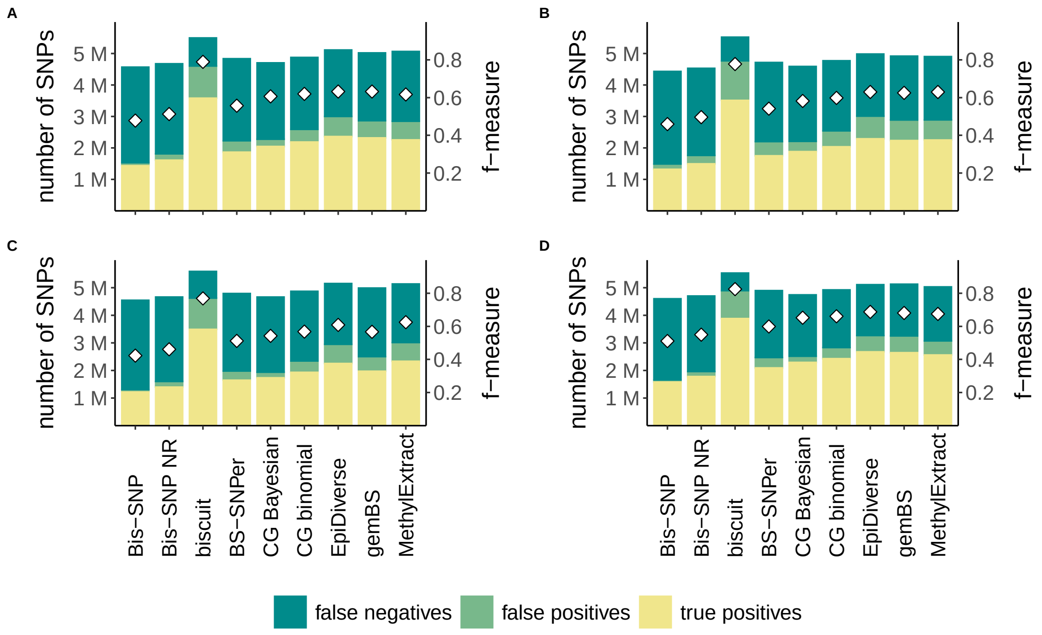

Figure 14: Number of false negative (teal), false positive (green), and true positive (yellow) SNPs called (bars and left y-axis) with the different tools tested for SNP calling from bisulfite sequencing data when the f-measure is maximized for the four samples (**A**-**D**, bars and left y-axis). Accuracy metrics are based on the evaluation with rtgTools and we here show the accuracy metrics for which the f-measure is maximized when using QUAL as score field (white diamonds and right y-axis). Note that the QUAL score values for which the f-measure is maximized differs between tools.

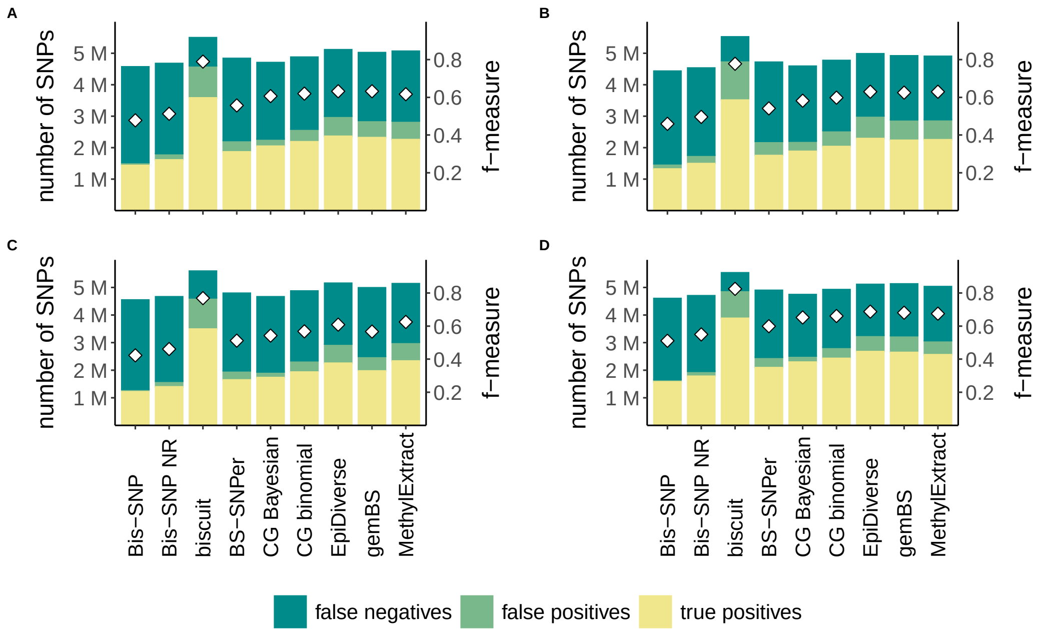

Figure 15: Number of false negative (teal), false positive (green), and true positive (yellow) SNPs called (bars and left y-axis) with the different tools tested for SNP calling from bisulfite sequencing data when the sensitivity is maximized for the four samples (**A**-**D**, bars and left y-axis). Accuracy metrics are based on the evaluation with rtgTools and we here show the accuracy metrics for which the sensitivity is maximized when using QUAL as score field (white diamonds and right y-axis). Note that the QUAL score values for which the sensitivity is maximized differs between tools.

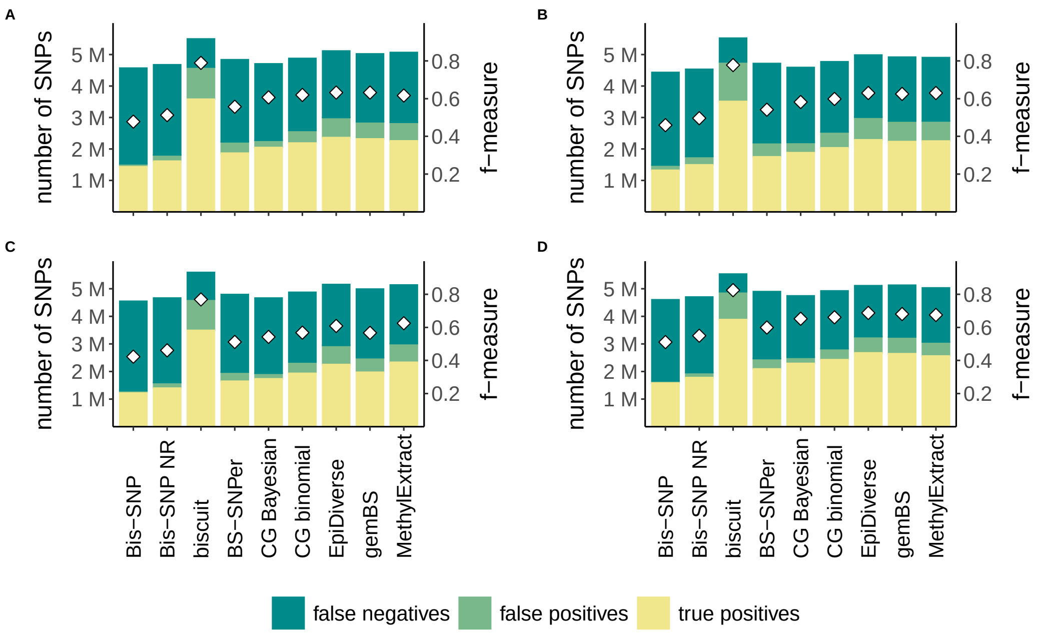

Figure 16: Number of false negative (teal), false positive (green), and true positive (yellow) SNPs called (bars and left y-axis) with the different tools tested for SNP calling from bisulfite sequencing data when the precision is maximized for the four samples (**A-D**, bars and left y-axis). Accuracy metrics are based on the evaluation with rtgTools and we here show the accuracy metrics for which the precision is maximized when using QUAL as score field (white diamonds and right y-axis). The QUAL score values for which the precision is maximized differs between tools. Note that precision is maximized on the condition that at least 1,000,000 SNPs were called.

4.3 Distribution of SNPs over substitution context

```
## Compare distribution of REF/ALT alleles called as false
## negatives and false positives in comparison to baseline
## SNPs Get baseline (whole−genome reseq) SNPs
FileName <− paste("../wgbs_snakemake_reseq/variants", list.files(path = "../wgbs_
    snakemake_reseq/variants",
     pattern = "*BaselinesAlleles"), sep = "/")
FileList <− lapply(FileName, function(x) read.csv(x, header = F,
     sep = " "))

# read ok?
lapply(FileList , function(x) dim(x))
lapply(FileList , function(x) head(x))

# prepare data for plotting
Samples_help <− str_split_fixed(FileName, "_", 8)
Samples <− paste(Samples_help[, 4], Samples_help[, 5], Samples_help[,
     6], Samples_help[, 7], sep = "_")
Data_Baseline <− NULL
for (i in 1:length(FileList)) {
     Data <− FileList[[i]]
     Data$Sample <− rep(Samples[i], nrow(Data))
     Data_Baseline <− rbind(Data_Baseline , Data)
}

Data_Baseline$Alleles <− paste(Data_Baseline$V3, Data_Baseline$V4,
     sep = "−")
Type_temp <− sort(unique(Data_Baseline$Alleles))
Type <− gsub("−", "−>", Type_temp)
```

```
Data_Baseline_Plot <- Data_Baseline %>%
    group_by(Sample, Alleles) %>%
    summarise(count = n())
Data_Baseline_Plot$count_k <- Data_Baseline_Plot$count/1000

# get legend
colfunc <- colorRampPalette(c("darkcyan", "khaki"))
col <- colfunc(4)

Plot <- Data_Baseline_Plot %>%
    mutate(Sample = factor(Sample, levels = Samples), Alleles = factor(Alleles,
        levels = Type_temp)) %>%
    ggplot() + geom_bar(aes(y = count, x = Alleles, fill = Sample),
    stat = "identity", position = position_dodge(), color = "white") +
    scale_fill_manual(name = "", values = col, labels = Samples) +
    scale_y_continuous(expand = c(0, 0)) + theme_classic() +
    theme(axis.text.x = element_text(size = 14), axis.text.y = element_blank(),
        axis.title.x = element_blank(), axis.title.y = element_blank(),
        axis.ticks.y = element_blank(), legend.text = element_text(size = 14),
        legend.key.size = unit(0.8, "cm"), legend.position = "top")
Legend <- get_legend(Plot)

# plot baseline distribution
Plot_Base <- Data_Baseline_Plot %>%
    mutate(Sample = factor(Sample, levels = Samples), Alleles = factor(Alleles,
        levels = Type_temp)) %>%
    ggplot() + geom_bar(aes(y = count_k, x = Alleles, fill = Sample),
    stat = "identity", position = position_dodge(), color = "white") +
    ggtitle("Baseline") + ylab("Number␣of␣thousand␣SNPs") + scale_fill_manual(name
       = "",
    values = col, labels = Samples) + scale_x_discrete(breaks = Type_temp,
    labels = Type) + scale_y_continuous(expand = c(0, 0)) + theme_classic() +
    theme(plot.title = element_text(size = 18, hjust = 0.5, margin = margin(t = 0,
        r = 0, b = 10, l = 0)), axis.text.x = element_text(size = 14,
        angle = 90, vjust = 0.5, color = "black"), axis.text.y = element_text(size
           = 14),
        axis.title.x = element_blank(), axis.title.y = element_text(size = 16),
        legend.position = "none")

## Load data called as false negatives: get file name and path
## of all output files:
FileLocation <- c("Bis-snp/SNP_calls/newV", "Bis-snp/SNP_calls/newV_NoRecal",
    "biscuit/pileup", "BS-snper/SNP_calls", "CGmaptools/SNP_calls",
    "epidiverse/SNP_calls/vcf", "gemBS/bcf", "MethylExtract/SNP_calls")
ToolNames <- c("Bis-SNP", "Bis-SNP␣NR", "biscuit", "BS-SNPer",
    "CG␣Bayesian", "CG␣binomial", "EpiDiverse", "gemBS", "MethylExtract")
FileNames <- NULL
for (f in 1:length(FileLocation)) {
    FileName <- paste(FileLocation[f], list.files(path = FileLocation[f],
        pattern = "*FN"), sep = "/")
    FileNames <- c(FileNames, FileName)
}
FileList <- lapply(FileNames, function(x) read.csv(x, header = F,
    sep = "␣"))

lapply(FileList, function(x) dim(x))
lapply(FileList, function(x) head(x))

# prepare data for plotting
```

```
Tools <- c(rep(ToolNames[1:4], each = 4), rep(ToolNames[5:6],
    4), rep(ToolNames[7:9], each = 4))
Samples_help <- str_split_fixed(FileNames, "/", 4)
Samples_help2 <- c(Samples_help[1:8, 4], Samples_help[9:24, 3],
    Samples_help[25:28, 4], Samples_help[29:36, 3])
Samples_help3 <- sub(".bayes.dynamicP", "", Samples_help2)
Samples_help4 <- sub(".binom", "", Samples_help3)
Samples <- sub("_SNPs_FN", "", Samples_help4)

Data_FN <- NULL
for (i in 1:length(FileList)) {
    Data <- FileList[[i]]
    Data$Sample <- rep(Samples[i], nrow(Data))
    Data$Tool <- rep(Tools[i], nrow(Data))
    Data_FN <- rbind(Data_FN, Data)
}

Data_FN$Alleles <- paste(Data_FN$V3, Data_FN$V4, sep = "-")
Type_temp <- sort(unique(Data_FN$Alleles))
Type <- gsub("-", "->", Type_temp)
Data_FN_Plot <- Data_FN %>%
    group_by(Tool, Sample, Alleles) %>%
    summarise(count = n())
Data_FN_Plot$count_k <- Data_FN_Plot$count/1000

# make plots (for each tool)
Plots <- NULL
for (i in 1:length(ToolNames)) {

    PlotDat <- Data_FN_Plot[Data_FN_Plot$Tool == ToolNames[i],
        ]

    Plot <- PlotDat %>%
        mutate(Sample = factor(Sample, levels = Samples[1:4]),
            Alleles = factor(Alleles, levels = Type_temp)) %>%
        ggplot() + geom_bar(aes(y = count_k, x = Alleles, fill = Sample),
        stat = "identity", position = position_dodge(), color = "white") +
        ylab("Number␣of␣thousand␣SNPs") + xlab("") + ggtitle(ToolNames[i]) +
        scale_fill_manual(name = "", values = col, labels = Samples) +
        scale_x_discrete(breaks = Type_temp, labels = Type) +
        scale_y_continuous(expand = c(0, 0)) + theme_classic() +
        theme(plot.title = element_text(size = 18, hjust = 0.5,
            margin = margin(t = 0, r = 0, b = 10, l = 0)), axis.text.x = element_
                text(size = 14,
            angle = 90, vjust = 0.5, color = "black"), axis.text.y = element_text(
                size = 14),
            axis.title.x = element_blank(), axis.title.y = element_text(size = 16),
            legend.position = "none")
    Name <- paste("plot", ToolNames[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}

# combine all plots and save for Supp Mat
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), get(Plots[[5]]), get(Plots[[6]]), get(Plots[[7]]),
    get(Plots[[8]]), get(Plots[[9]]), labels = "", scale = 0.9,
    ncol = 3)
Plot_help <- plot_grid(NULL, Plot_Base, NULL, labels = "", scale = 0.9,
    ncol = 3)
Plot_comb <- plot_grid(Plot_help, Plot, labels = "AUTO", label_size = 14,
    scale = 1, ncol = 1, rel_heights = c(3, 9))
Plot_legend <- plot_grid(Plot_comb, Legend, rel_heights = c(15,
    1), nrow = 2)
save_plot("Plots/FN_SNPs.pdf", Plot_legend, base_height = 15,
    base_width = 12)

## Load data called as false positives: get file name and path
## of all output files:
FileLocation <- c("Bis-snp/SNP_calls/newV", "Bis-snp/SNP_calls/newV_NoRecal",
    "biscuit/pileup", "BS-snper/SNP_calls", "CGmaptools/SNP_calls",
    "epidiverse/SNP_calls/vcf", "gemBS/bcf", "MethylExtract/SNP_calls")
ToolNames <- c("Bis-SNP", "Bis-SNP␣NR", "biscuit", "BS-SNPer",
    "CG␣Bayesian", "CG␣binomial", "EpiDiverse", "gemBS", "MethylExtract")

FileNames <- NULL
for (f in 1:length(FileLocation)) {
    FileName <- paste(FileLocation[f], list.files(path = FileLocation[f],
        pattern = "*FP"), sep = "/")
    FileNames <- c(FileNames, FileName)
}
FileList <- lapply(FileNames, function(x) read.csv(x, header = F,
    sep = "␣"))

lapply(FileList, function(x) dim(x))
lapply(FileList, function(x) head(x))

# prepare data for plotting
Tools <- c(rep(ToolNames[1:4], each = 4), rep(ToolNames[5:6],
    4), rep(ToolNames[7:9], each = 4))
Samples_help <- str_split_fixed(FileNames, "/", 4)
Samples_help2 <- c(Samples_help[1:8, 4], Samples_help[9:24, 3],
    Samples_help[25:28, 4], Samples_help[29:36, 3])
Samples_help3 <- sub(".bayes.dynamicP", "", Samples_help2)
Samples_help4 <- sub(".binom", "", Samples_help3)
Samples <- sub("_SNPs_FN", "", Samples_help4)

Data_FP <- NULL
for (i in 1:length(FileList)) {
    Data <- FileList[[i]]
    Data$Sample <- rep(Samples[i], nrow(Data))
    Data$Tool <- rep(Tools[i], nrow(Data))
    Data_FP <- rbind(Data_FP, Data)
}

Data_FP$Alleles <- paste(Data_FP$V3, Data_FP$V4, sep = "-")
Type_temp <- sort(unique(Data_FP$Alleles))
Type <- gsub("-", "->", Type_temp)
Data_FP_Plot <- Data_FP %>%
    group_by(Tool, Sample, Alleles) %>%
    summarise(count = n())
Data_FP_Plot$count_k <- Data_FP_Plot$count/1000

# make plots (for each tool)
Plots <- NULL
for (i in 1:length(ToolNames)) {

    PlotDat <- Data_FP_Plot[Data_FP_Plot$Tool == ToolNames[i],
        ]

    Plot <- PlotDat %>%
        mutate(Sample = factor(Sample, levels = Samples[1:4]),
            Alleles = factor(Alleles, levels = Type_temp)) %>%
        ggplot() + geom_bar(aes(y = count_k, x = Alleles, fill = Sample),
        stat = "identity", position = position_dodge(), color = "white") +
        ylab("Number␣of␣thousand␣SNPs") + xlab("") + ggtitle(ToolNames[i]) +
        scale_fill_manual(name = "", values = col, labels = Samples) +
        scale_x_discrete(breaks = Type_temp, labels = Type) +
        scale_y_continuous(expand = c(0, 0)) + theme_classic() +
        theme(plot.title = element_text(size = 18, hjust = 0.5,
            margin = margin(t = 0, r = 0, b = 10, l = 0)), axis.text.x = element_
                text(size = 14,
            angle = 90, vjust = 0.5, color = "black"), axis.text.y = element_text(
                size = 14),
            axis.title.x = element_blank(), axis.title.y = element_text(size = 16),
            legend.position = "none")
    Name <- paste("plot", ToolNames[i], sep = "_")
    assign(Name, Plot)
    Plots <- append(Plots, Name, length(Plots))
}

# combine all plots and save for Supp Mat
Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
    get(Plots[[4]]), get(Plots[[5]]), get(Plots[[6]]), get(Plots[[7]]),
    get(Plots[[8]]), get(Plots[[9]]), labels = "", scale = 0.9,
    ncol = 3)
Plot_help <- plot_grid(NULL, Plot_Base, NULL, labels = "", scale = 0.9,
    ncol = 3)
Plot_comb <- plot_grid(Plot_help, Plot, labels = "AUTO", label_size = 14,
    scale = 1, ncol = 1, rel_heights = c(3, 9))
Plot_legend <- plot_grid(Plot_comb, Legend, rel_heights = c(15,
    1), nrow = 2)
save_plot("Plots/FP_SNPs.pdf", Plot_legend, base_height = 15,
    base_width = 12)
```

4.4 Distribution of herozygous and homozygous false positive SNPs over substitution context

```
FileLocation <- c("Bis-snp/SNP_calls/newV", "Bis-snp/SNP_calls/newV_NoRecal",
    "biscuit/pileup", "BS-snper/SNP_calls", "CGmaptools/SNP_calls",
    "epidiverse/SNP_calls/vcf", "gemBS/bcf", "MethylExtract/SNP_calls")
ToolNames <- c("Bis-SNP", "Bis-SNP␣NR", "biscuit", "BS-SNPer",
    "CG␣Bayesian", "CG␣binomial", "EpiDiverse", "gemBS", "MethylExtract")

FileNames <- NULL
for (f in 1:length(FileLocation)) {
    FileName <- paste(FileLocation[f], list.files(path = FileLocation[f],
        pattern = "*FP_GT$"), sep = "/")
    FileNames <- c(FileNames, FileName)
}
FileList <- lapply(FileNames, function(x) read.csv(x, header = F,
    sep = "␣"))

lapply(FileList, function(x) dim(x))
lapply(FileList, function(x) head(x))

# prepare data for plotting
```

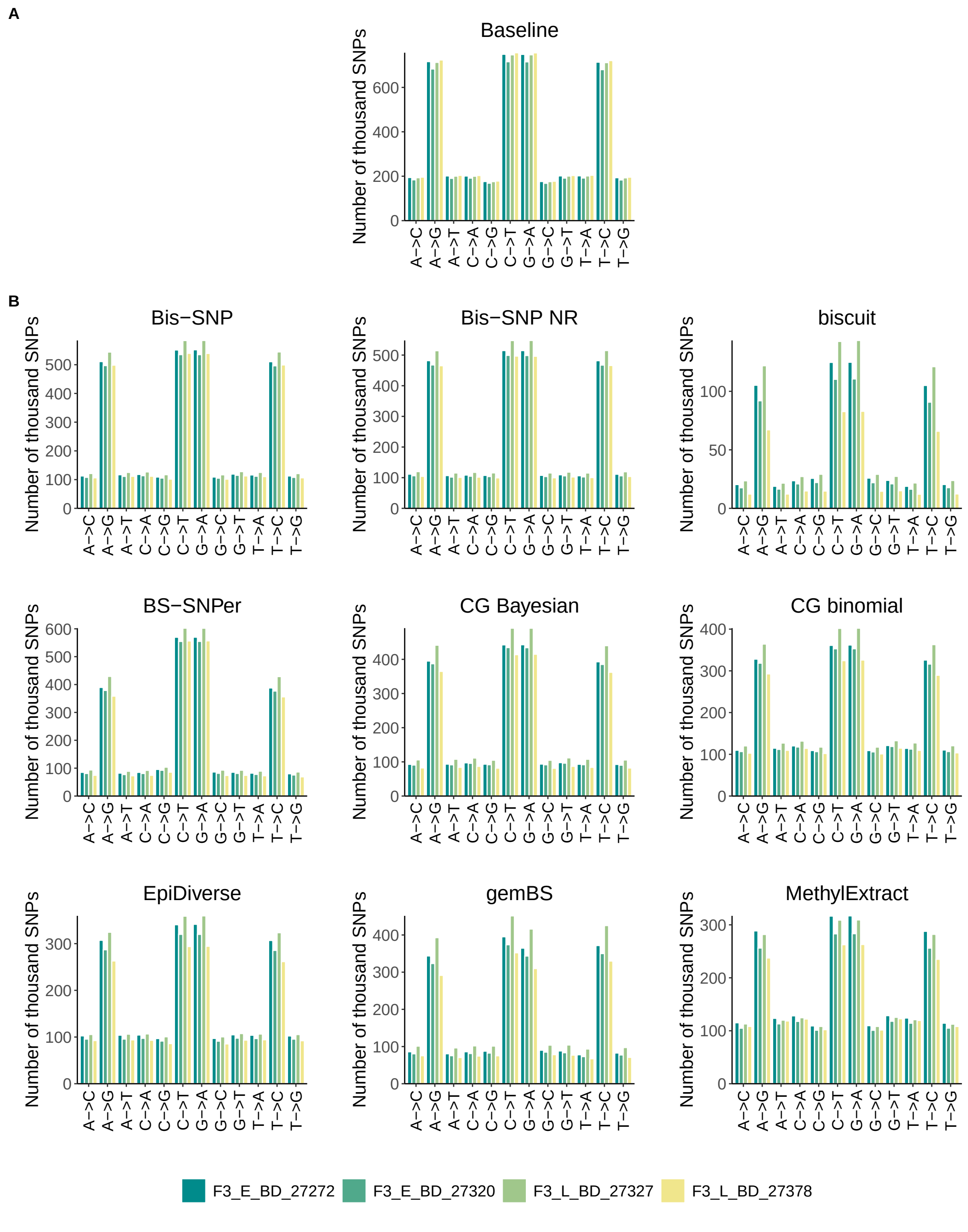

Figure 17: Distribution of SNPs over substitution contexts (alternative and reference allele) for the baseline list of true SNPs derived from whole-genome resequencing data (**A**) and the tool-specific list of false negatives SNPs (**B**). Samples are differentiated by colour (teal to yellow) and plots in **B** have tool-specific plot titles.

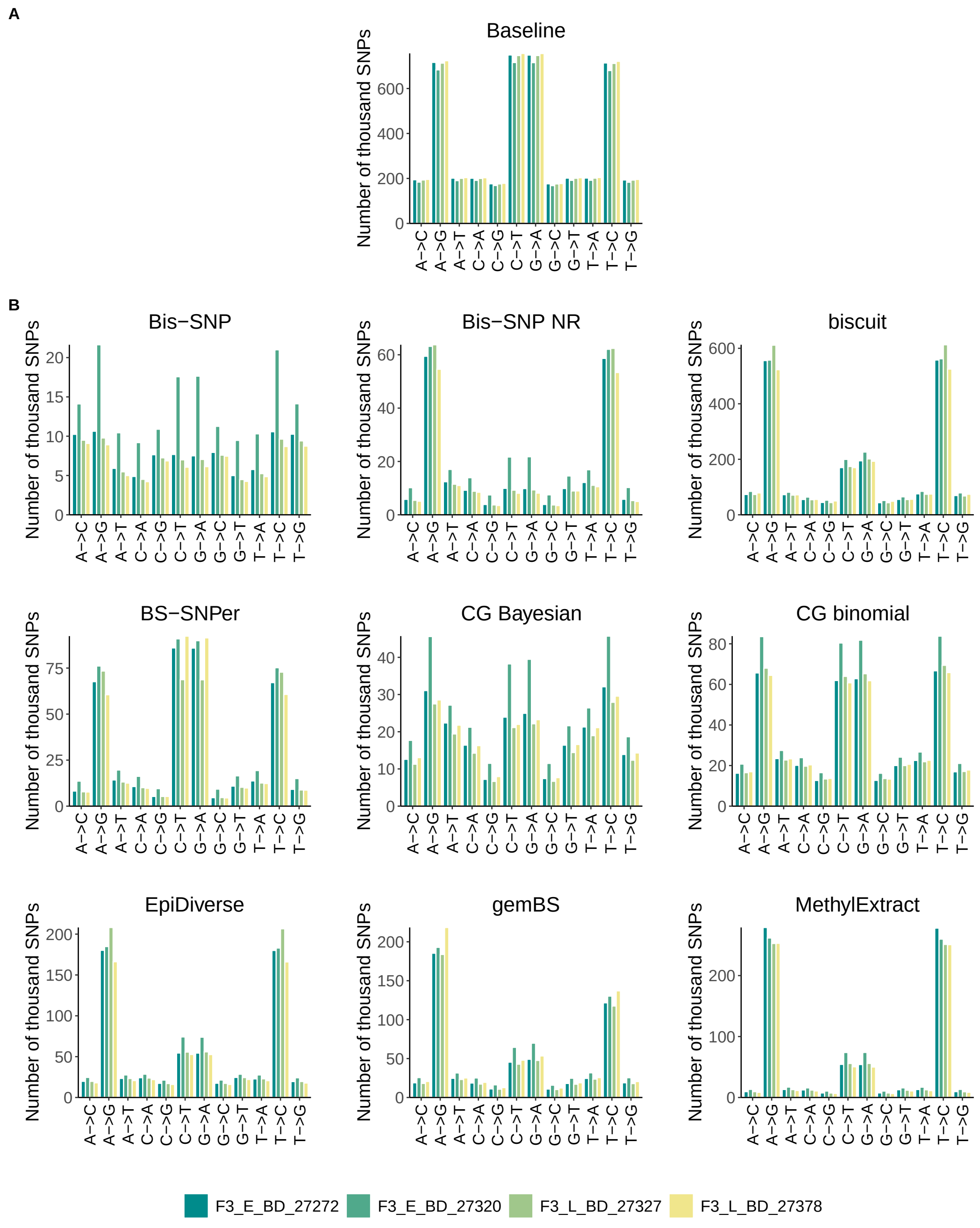

Figure 18: Distribution of SNPs over substitution contexts (alternative and reference allele) for the baseline list of true SNPs derived from whole-genome resequencing data (A) and the tool-specific lists of false positive SNPs (B). Samples are differentiated by colour (teal-yellow) and plots in **B** have tool-specific plot titles.

```
Tools <- c(rep(ToolNames[1:4], each = 4), rep(ToolNames[5:6],
     4), rep(ToolNames[7:9], each = 4))
Samples_help <- str_split_fixed(FileNames, "/", 4)
Samples_help2 <- c(Samples_help[1:8, 4], Samples_help[9:24, 3],
     Samples_help[25:28, 4], Samples_help[29:36, 3])
Samples_help3 <- sub(".bayes.dynamicP", "", Samples_help2)
Samples_help4 <- sub(".binom", "", Samples_help3)
Samples <- sub("_SNPs_FP_GT", "", Samples_help4)

Data_FP <- NULL
for (i in 1:length(FileList)) {
     Data <- FileList[[i]]
     Data$Sample <- rep(Samples[i], nrow(Data))
     Data$Tool <- rep(Tools[i], nrow(Data))
     Data_FP <- rbind(Data_FP, Data)
}

# remove 6495 SNPs from EpiDiverse-SNP pipeline that have no
# definite call (i.e. can be heterozygous or homozygous for
# ALT allele)
Data_FP_GT <- Data_FP %>%
     filter(V5 != "./1")

Data_FP_GT$Alleles <- paste(Data_FP_GT$V3, Data_FP_GT$V4, sep = "-")
Type_temp <- sort(unique(Data_FP_GT$Alleles))
Type <- gsub("-", "->", Type_temp)
Data_FP_GT$Genotype <- ifelse(Data_FP_GT$V5 == "1/1", "homozygous",
     "heterozygous")
GType <- unique(Data_FP_GT$Genotype)
Data_FP_GT_Plot <- Data_FP_GT %>%
     group_by(Tool, Sample, Alleles, Genotype) %>%
     summarise(count = n())
Data_FP_GT_Plot$count_k <- Data_FP_GT_Plot$count/1000

# make plots (for each tool)
colfunc <- colorRampPalette(c("darkcyan", "khaki"))
col <- colfunc(2)

# get legend
SampleDat <- Data_FP_GT_Plot[Data_FP_GT_Plot$Sample == "F3_E_BD_27272",
     ]
PlotDat <- SampleDat[SampleDat$Tool == ToolNames[1], ]
Plot <- PlotDat %>%
     mutate(Genotype = factor(Genotype, levels = GType), Alleles = factor(Alleles,
          levels = Type_temp)) %>%
     ggplot() + geom_bar(aes(y = count_k, x = Alleles, fill = Genotype),
     stat = "identity", color = "white") + ylab("") + xlab("") +
     scale_fill_manual(name = "", values = col, labels = GType) +
     scale_y_continuous(expand = c(0, 0)) + theme_classic() +
     theme(plot.title = element_text(size = 18, hjust = 0.5, margin = margin(t = 0,
          r = 0, b = 10, l = 0)), axis.text.x = element_text(size = 14,
          angle = 90, vjust = 0.5, color = "black"), axis.text.y = element_blank(),
          axis.title.x = element_blank(), axis.title.y = element_blank(),
          axis.ticks.y = element_blank(), legend.text = element_text(size = 14),
          legend.key.size = unit(0.8, "cm"), legend.position = "top")
Legend <- get_legend(Plot)

# loop through samples and then through tools
for (s in Samples[1:4]) {
    SampleDat <- Data_FP_GT_Plot[Data_FP_GT_Plot$Sample == Samples[s],
        ]

    Plots <- NULL
    for (i in 1:length(ToolNames)) {

        PlotDat <- SampleDat[SampleDat$Tool == ToolNames[i],
            ]
        Plot <- PlotDat %>%
            mutate(Genotype = factor(Genotype, levels = GType),
                Alleles = factor(Alleles, levels = Type_temp)) %>%
            ggplot() + geom_bar(aes(y = count_k, x = Alleles,
            fill = Genotype), stat = "identity", color = "white") +
            ylab("Number␣of␣thousand␣SNPs") + xlab("") + ggtitle(ToolNames[i]) +
            scale_fill_manual(name = "", values = col, labels = GType) +
            scale_x_discrete(breaks = Type_temp, labels = Type) +
            scale_y_continuous(expand = c(0, 0)) + theme_classic() +
            theme(plot.title = element_text(size = 18, hjust = 0.5,
                margin = margin(t = 0, r = 0, b = 10, l = 0)),
                axis.text.x = element_text(size = 14, angle = 90,
                  vjust = 0.5, color = "black"), axis.text.y = element_text(size =
                    14),
                axis.title.x = element_blank(), axis.title.y = element_text(size =
                   16),
                legend.position = "none")
        Name <- paste("plot", ToolNames[i], sep = "_")
        assign(Name, Plot)
        Plots <- append(Plots, Name, length(Plots))
    }

    # combine all plots and save for Supp Mat
    Plot <- plot_grid(get(Plots[[1]]), get(Plots[[2]]), get(Plots[[3]]),
        get(Plots[[4]]), get(Plots[[5]]), get(Plots[[6]]), get(Plots[[7]]),
        get(Plots[[8]]), get(Plots[[9]]), labels = "", scale = 0.9,
        ncol = 3)
    Plot_legend <- plot_grid(Plot, Legend, rel_heights = c(8,
        0.5), nrow = 2)
    Plot_Name <- paste("plots/FP_GT_SNPs_", Samples[s], ".pdf",
        sep = "")
    save_plot(Plot_Name, Plot_legend, base_height = 15, base_width = 12)
}

# quantify access of heterozygosity
Data_FP_GT_temp <- spread(Data_FP_GT_Plot[, 1:5], Genotype, count)
Data_FP_GT_temp$Perc_het <- Data_FP_GT_temp$heterozygous/(Data_FP_GT_temp$
   heterozygous +
    Data_FP_GT_temp$homozygous)
Data_FP_GT_het <- spread(Data_FP_GT_temp[, c(1:3, 6)], Alleles,
    Perc_het)
Data_FP_GT_het$max <- apply(Data_FP_GT_het[, 3:14], 1, max)
Data_FP_GT_het$min <- apply(Data_FP_GT_het[, 3:14], 1, min)
names(Data_FP_GT_het)[3:14] <- gsub("-", "->", names(Data_FP_GT_het)[3:14])

write.table(Data_FP_GT_het, "out/Data_FP_het", quote = F, sep = "\t",
    col.names = TRUE, row.names = FALSE)
```

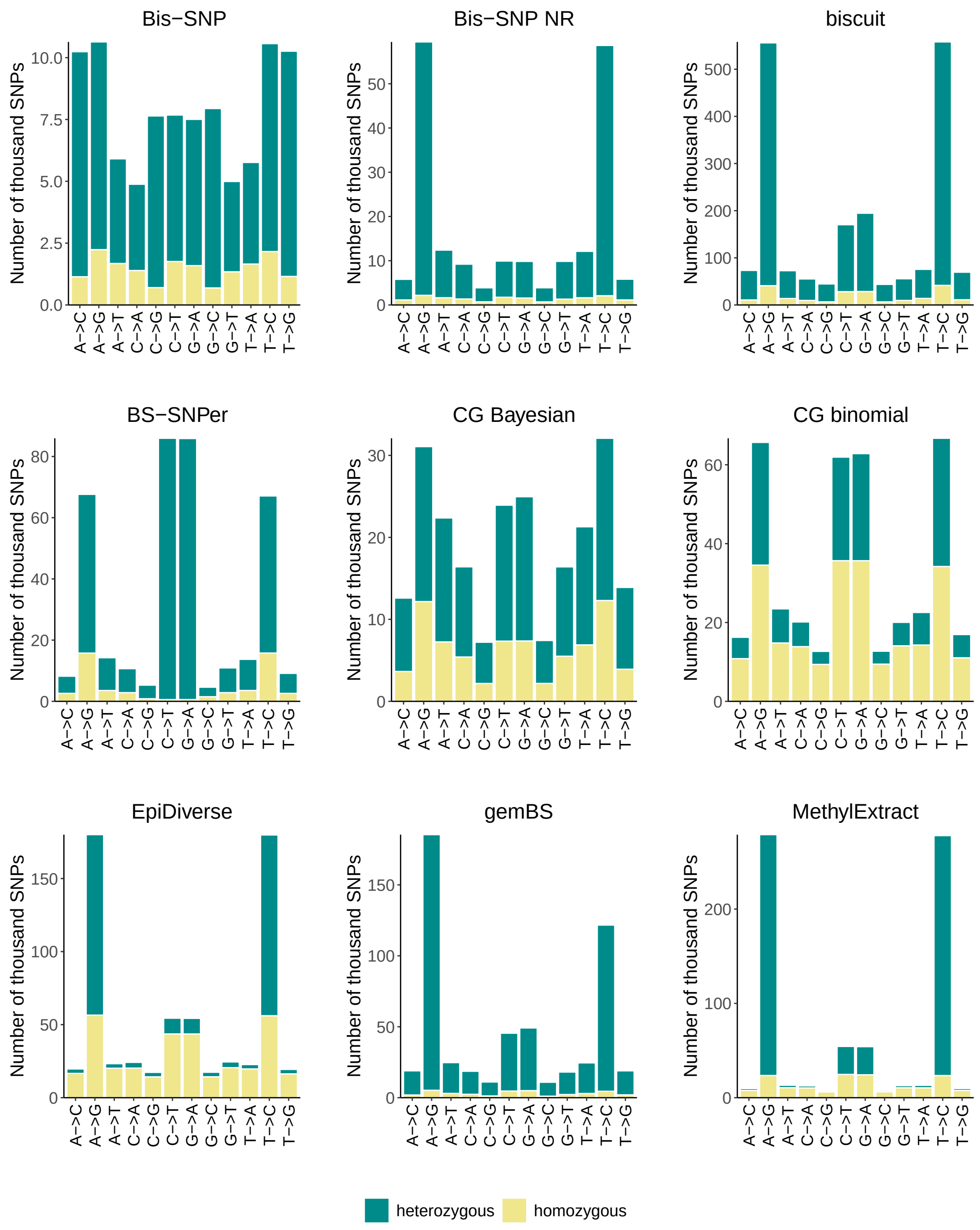

Figure 19: Distribution of SNPs over substitution contexts (alternative and reference allele) for sample F3_E_BD_27272 differentiating between heterozygous (teal) and homozygous (yellow) genotypes. Plots have tool-specific plot titles.

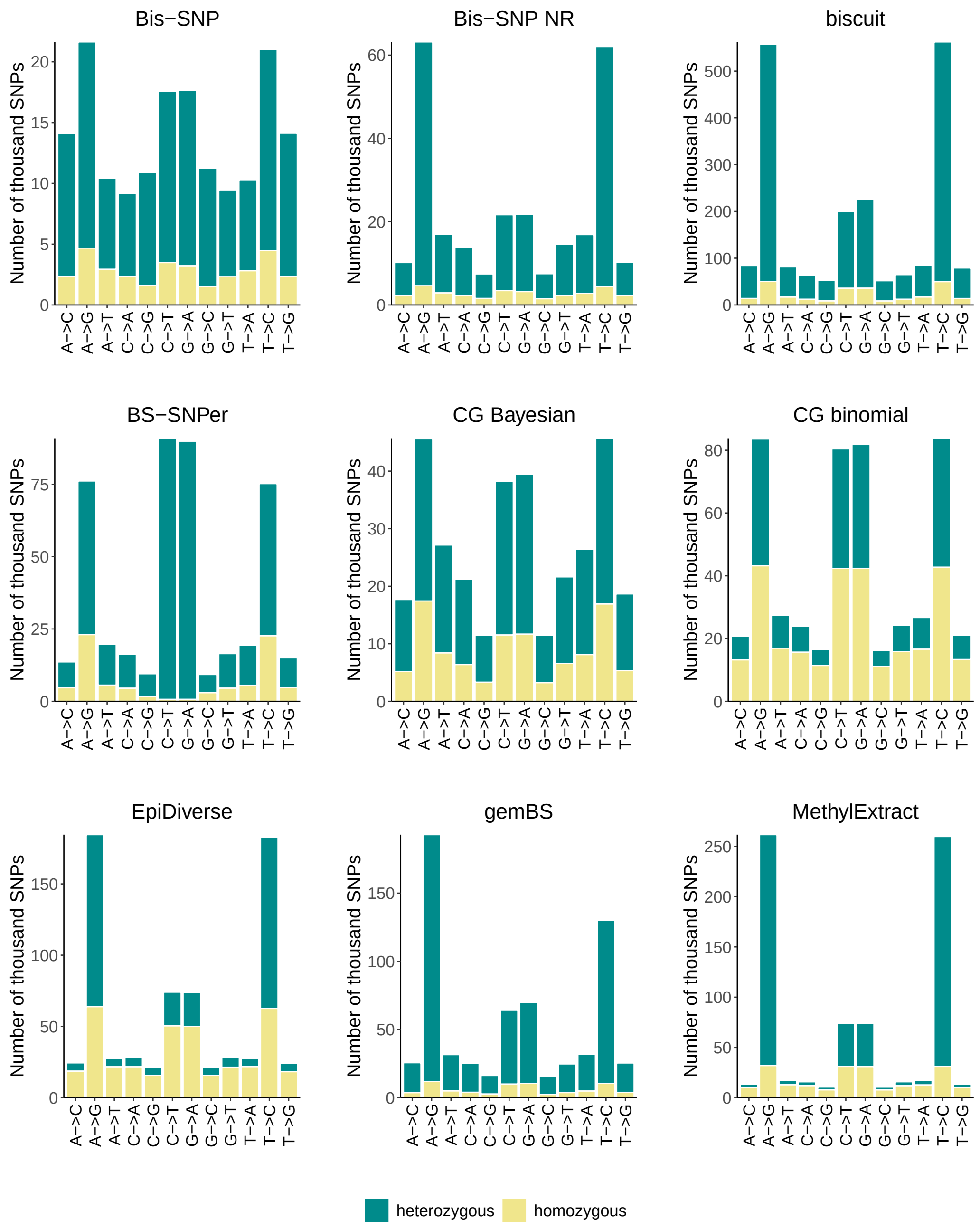

Figure 20: Distribution of SNPs over substitution contexts (alternative and reference allele) for sample F3_E_BD_27320 differentiating between heterozygous (teal) and homozygous (yellow) genotypes. Plots have tool-specific plot titles.

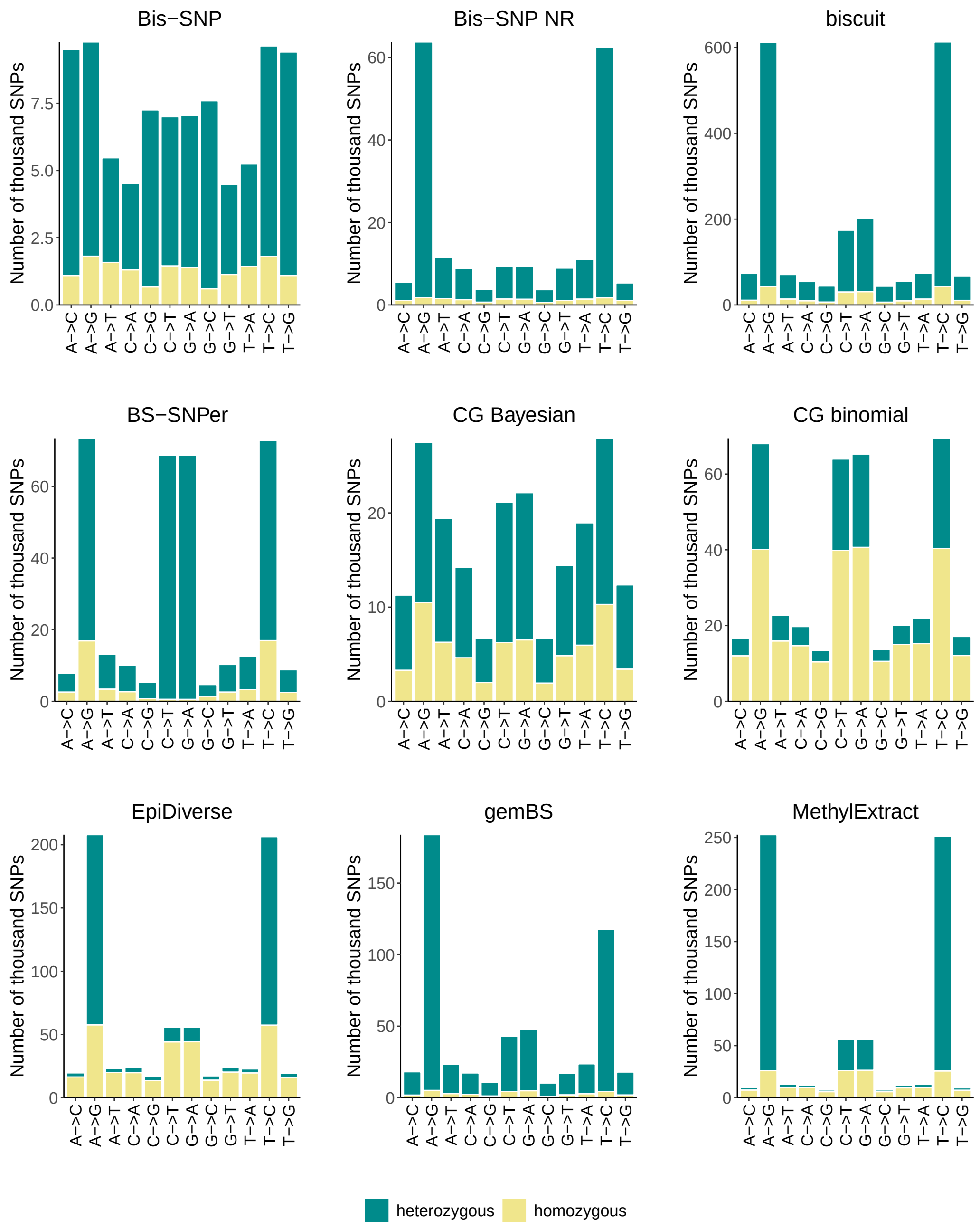

Figure 21: Distribution of SNPs over substitution contexts (alternative and reference allele) for sample F3_L_BD_27327 differentiating between heterozygous (teal) and homozygous (yellow) genotypes. Plots have tool-specific plot titles.

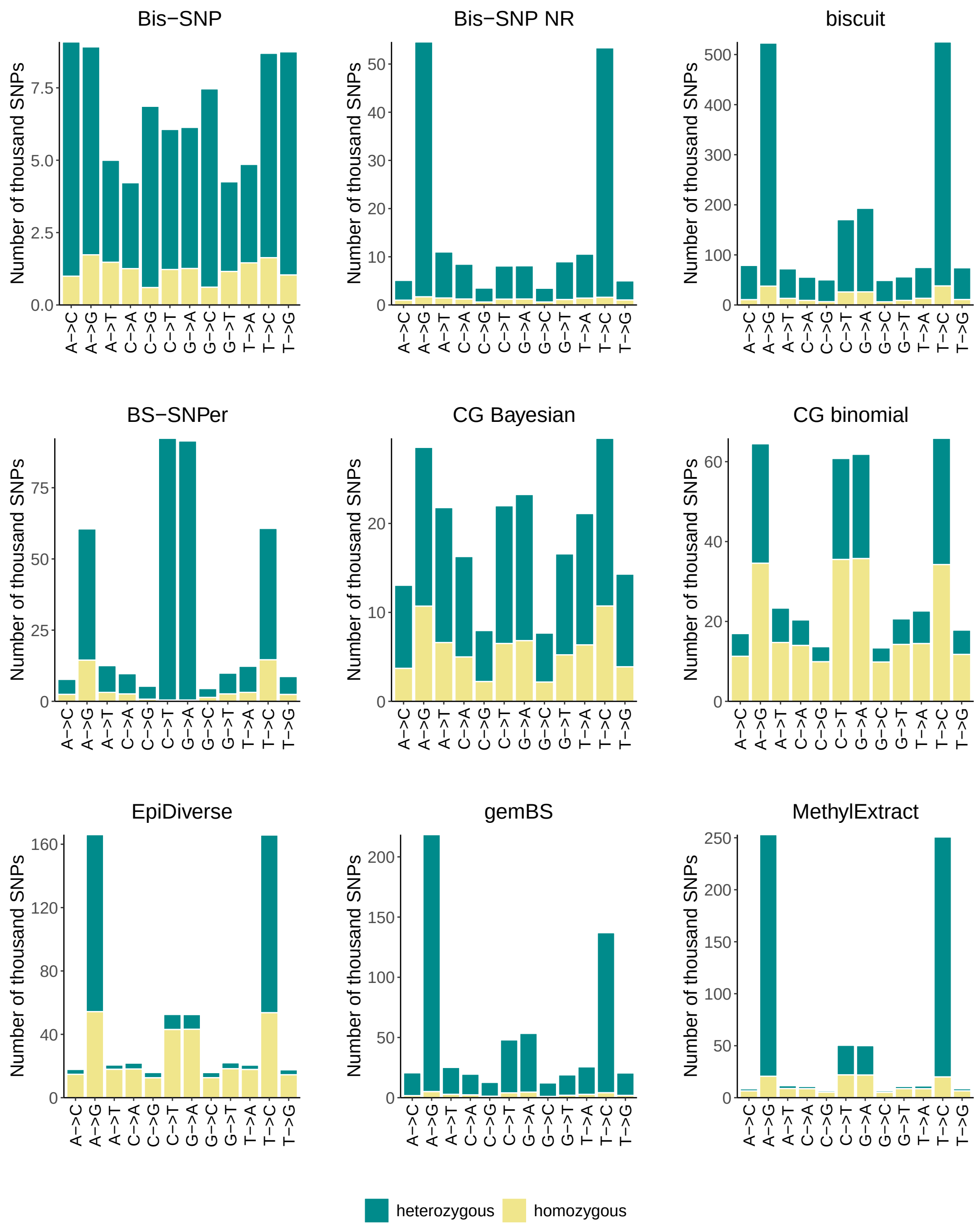

Figure 22: Distribution of SNPs over substitution contexts (alternative and reference allele) for sample F3_L_BD_27378 differentiating between heterozygous (teal) and homozygous (yellow) genotypes. Plots have tool-specific plot titles.