```
######################################################################
#################Supplementary_code_to_raw_data ###################
######################################################################

###########################
#####Moving the files######
###########################

#in a folder 'methylation_analysis/':
#download the untrimmed, but demultiplexed file from Belnet
      (Nucleomics core): expXXX.tar.gz
#move the tar.gz-file with WinSCP to the unix environment
#unzip and untar the tar.gz-file
#tar -zxvf expXXX.tar.gz
tar -zxvf exp2802-NextSeq_High_output_24libs.tar.gz

#make directories per library
      (date_method_patient_DNA_repeat_i(indexnr)) and copy the
      paired-end reads from all lanes to the directories
mkdir 20171012_cfRRBS_uzg013_cfDNA_1_i2
mv RRBS02_S2_L001_R1_001.fastq.gz RRBS02_S2_L002_R1_001.fastq.gz
RRBS02_S2_L003_R1_001.fastq.gz RRBS02_S2_L004_R1_001.fastq.gz
      RRBS02_S2_L001_R2_001.fastq.gz RRBS02_S2_L002_R2_001.fastq.gz
      RRBS02_S2_L003_R2_001.fastq.gz RRBS02_S2_L004_R2_001.fastq.gz
      20171012_cfRRBS_uzg013_cfDNA_1_i2

#go to directory
cd 20171012_cfRRBS_uzg013_cfDNA_1_i2

#concatenate the reads from all lanes
cat RRBS02_S2_L001_R1_001.fastq.gz RRBS02_S2_L002_R1_001.fastq.gz
      RRBS02_S2_L003_R1_001.fastq.gz RRBS02_S2_L004_R1_001.fastq.gz >
      20171012_i2_R1.fastq.gz
cat RRBS02_S2_L001_R2_001.fastq.gz RRBS02_S2_L002_R2_001.fastq.gz
      RRBS02_S2_L003_R2_001.fastq.gz RRBS02_S2_L004_R2_001.fastq.gz >
      20171012_i2_R2.fastq.gz

#remove the reads-per-lane files
rm RRBS02_S2_L001_R1_001.fastq.gz RRBS02_S2_L002_R1_001.fastq.gz
      RRBS02_S2_L003_R1_001.fastq.gz RRBS02_S2_L004_R1_001.fastq.gz
rm RRBS02_S2_L001_R2_001.fastq.gz RRBS02_S2_L002_R2_001.fastq.gz
      RRBS02_S2_L003_R2_001.fastq.gz RRBS02_S2_L004_R2_001.fastq.gz


#####################################################
#####extra code: Crop and/or downsample raw reads#####
#####################################################

#Environment:
#Seqtk v1.2-r101

#to compare performance between 2 methods, sometimes the amount of
      reads had to be downsampled. S2 = random seed 2
seqtk sample -s2 ../20171012_i2_R1.fastq.gz 1000000 >
      20171012_i2_R1_1M.fastq
seqtk sample -s2 ../20171012_i2_R2.fastq.gz 1000000 >
      20171012_i2_R2_1M.fastq

#SeqCapEpi samples were sequenced on a HiSeq4000 PE150, but
      optimally they should have been sequenced PE100, so the raw
      (downsampled) raw reads are cropped to 100bp
seqtk trimfq -L 100 20180615_i2_R1_1M.fastq >
      20180615_i2_R1_1M_crop100.fastq
```

```
seqtk trimfq -L 100 20180615_i2_R2_1M.fastq >
     20180615_i2_R2_1M_crop100.fastq

#seqtk gunzips the files, so they have to be gzipped again
gzip 20180615_i2_R1_1M_crop100.fastq 20180615_i2_R2_1M_crop100.fastq


############################
#####Trimming the reads#####
############################

#Environment:
#trim_galore v0.4.3 script in folder 'methylation_analysis'
#    with dependencies installed (Cutadapt v1.13, FastQC v0.11.5)

#perl ../trim_galore [options] date_i(indexnr)_R1.fastq.gz
     date_i(indexnr)_R2.fastq.gz

#options used:
#    --paired                    when trimming PE-libraries
#    --rrbs                      when trimming cfRRBS and RRBS data
#    no --rrbs                   when trimming RRBS on cfDNA data,
                                      WGBS data and SeqCapEpi data
#    non_directional             when trimming epiGBS data (chapt. 4)
#    --clip_R1 10                when trimming SeqCapEpi data with
                                      SWIFT Accel-NGS methyl-kit
#    --clip_R2 10                when trimming SeqCapEpi data with
                                      SWIFT Accel-NGS methyl-kit
#    --three_prime_clip_R1 10    when trimming SeqCapEpi data with
                                      SWIFT Accel-NGS methyl-kit
#    --three_prime_clip_R2 10    when trimming SeqCapEpi data with
                                      SWIFT Accel-NGS methyl-kit
#    --clip_R1 18                when trimming WGBS data with SWIFT
                                      Accel-NGS methyl-kit, but seen
                                      in fastqc
#    --clip_R2 18                when trimming WGBS data with SWIFT
                                      Accel-NGS methyl-kit, but seen
                                      in fastqc
#    --three_prime_clip_R1 2     when trimming WGBS data with SWIFT
                                      Accel-NGS methyl-kit, but seen
                                      in fastqc
#    --three_prime_clip_R2 2     when trimming WGBS data with SWIFT
                                      Accel-NGS methyl-kit, but seen
                                      in fastqc

perl ../trim_galore --paired --rrbs 20171012_i2_R1.fastq.gz
     20171012_i2_R2.fastq.gz

#output:
20171012_i2_R1_val_1.fq.gz
20171012_i2_R2_val_2.fq.gz
20171012_i2_R1.fastq.gz_trimming_report.txt
20171012_i2_R2.fastq.gz_trimming_report.txt

#rename files
mv 20171012_i2_R1_val_1.fq.gz 20171012_i2_R1_trimmed.fastq.gz
mv 20171012_i2_R2_val_2.fq.gz 20171012_i2_R2_trimmed.fastq.gz
mv 20171012_i2_R1.fastq.gz_trimming_report.txt
     20171012_i2_R1_trimming_report.txt
mv 20171012_i2_R2.fastq.gz_trimming_report.txt
     20171012_i2_R2_trimming_report.txt
```

```
##################################################
#####Bisulfite converting the reference genome#####
##################################################

#Environment:
#bismark v0.17.0 folder in 'methylation_analysis/'
#download 'hg19.fa' and put in folder 'hg19/'
#download 'lambda.fa' and put in folder 'lambda/'
#bowtie2 v2.3.2

#only need to do this once
bismark_v0.17.0/bismark_genome_preparation --verbose --bowtie2 hg19/

#or the lambda/ genome
bismark_v0.17.0/bismark_genome_preparation --verbose --bowtie2
     lambda/


######################
#####Read mapping#####
######################

#Environment:
#bismark v0.17.0 folder in 'methylation_analysis/'


#../bismark_v0.17.0/bismark [options] --genome_folder ../hg19/ (or
     lambda/)   -1 20171012_i2_R1_trimmed.fastq.gz -2
     20171012_i2_R2_trimmed.fastq.gz

#options used:
#    --un                      to keep the unmapped reads (only
                                    needed when mapping to hg19)
#    --ambiguous               to keep the ambiguous mapped reads
                                    (only needed when mapping to
                                    hg19)
#    --ambig_bam               to map the amiguous mapped reads,
                                    random assignment to one of the
                                    possible mapping sites (only
                                    needed when mapping to hg19)
#    --non_directional         when working with epiGBS reads
                                    (chapt.4)

../bismark_v0.17.0/bismark --genome_folder ../hg19/ --un --ambiguous
     --ambig_bam -1 20171012_i2_R1_trimmed.fastq.gz -2
     20171012_i2_R2_trimmed.fastq.gz

#output:
20171012_i2_R1_trimmed_bismark_bt2_pe.bam
20171012_i2_R1_trimmed_bismark_bt2_pe.ambig.bam
20171012_i2_R1_trimmed_bismark_bt2_PE_report.txt
20171012_i2_R1_trimmed.fastq.gz_ambiguous_reads_1.fq.gz
20171012_i2_R2_trimmed.fastq.gz_ambiguous_reads_2.fq.gz
20171012_i2_R1_trimmed.fastq.gz_unmapped_reads_1.fq.gz
20171012_i2_R2_trimmed.fastq.gz_unmapped_reads_2.fq.gz

#rename the files, because when mapping to lambda next, the files
     get the same name
mv 20171012_i2_R1_trimmed_bismark_bt2_pe.bam
     20171012_i2_mapping_hg19.bam
mv 20171012_i2_R1_trimmed_bismark_bt2_pe.ambig.bam
     20171012_i2_mapping_hg19_ambig.bam
```

```
mv 20171012_i2_R1_trimmed_bismark_bt2_PE_report.txt
      20171012_i2_mapping_hg19_report.txt
mv 20171012_i2_R1_trimmed.fastq.gz_ambiguous_reads_1.fq.gz
      20171012_i2_R1_ambiguous_reads.fastq.gz
mv 20171012_i2_R2_trimmed.fastq.gz_ambiguous_reads_2.fq.gz
      20171012_i2_R2_ambiguous_reads.fastq.gz
mv 20171012_i2_R1_trimmed.fastq.gz_unmapped_reads_1.fq.gz
      20171012_i2_R1_unmapped_reads.fastq.gz
mv 20171012_i2_R2_trimmed.fastq.gz_unmapped_reads_2.fq.gz
      20171012_i2_R2_unmapped_reads.fastq.gz


##################################
#####Extract methylation data#####
##################################

#Environment:
#bismark v0.17.0 folder in 'methylation_analysis/'

#../bismark_v0.17.0/bismark_methylation_extractor [options]
      20171012_i2_mapping_hg19.bam

#options used:
#     -p                      when working with paired-end data
#     -s                      when working with single-end data
#     --ignore_r2 2           rrbs paired end
#     --comprehensive         Specifying this option will merge all
                                    four possible strand-specific
                                    methylation info into context-
                                    dependent output files. The default
                                    contexts are: CpG context, CHG
                                    context, CHH context
#     --merge_non_CpG         This will produce two output files (in --
                                    comprehensive mode) or eight strand-
                                    specific output files (default) for
                                    Cs in CpG context and non-CpG
                                    context
#     --bedGraph              needed for option --cytosine_report
#     --cytosine_report       outputs percent methylation, methylated
                                    count and unmethylated count per
                                    covered CpG
#     --genome_folder hg19\ (or lambda\) needed for option --
                                    cytosine_report


../bismark_v0.17.0/bismark_methylation_extractor -p --ignore_r2 2 --
      comprehensive --merge_non_CpG --bedGraph --cytosine_report --
      genome_folder ../hg19/ 20171012_i2_mapping_hg19.bam

#output:
20171012_i2_mapping_hg19.bedGraph.gz
20171012_i2_mapping_hg19.bismark.cov.gz
20171012_i2_mapping_hg19.CpG_report.txt
20171012_i2_mapping_hg19.M-bias.txt
20171012_i2_mapping_hg19_splitting_report.txt
CpG_context_20171012_i2_mapping_hg19.txt
Non_CpG_context_20171012_i2_mapping_hg19.txt

#remove the files we do not need
rm 20171012_i2_mapping_hg19.bedGraph.gz
rm CpG_context_20171012_i2_mapping_hg19.txt
rm Non_CpG_context_20171012_i2_mapping_hg19.txt
rm 20171012_i2_mapping_hg19.CpG_report.txt
```

```
#keep and rename the ones needed (not necessary but when testing
     other parameters the original files would be overwritten when
     name was not changed)
mv 20171012_i2_mapping_hg19.bismark.cov.gz 20171012_i2_hg19.cov.gz
mv 20171012_i2_mapping_hg19.M-bias.txt 20171012_i2_hg19_M-bias.txt
mv 20171012_i2_mapping_hg19_splitting_report.txt
     20171012_i2_hg19_splitting_report.txt

#QC
../bismark_v0.17.0/bismark2report --alignment_report
     20171012_i2_mapping_hg19_report.txt --splitting_report
     20171012_i2_hg19_splitting_report.txt --mbias_report
     20171012_i2_hg19_M-bias.txt

#output:
20171012_i2_mapping_hg19_report.html

#once html-report is generated, M-bias file and splitting_report can
     be removed
rm 20171012_i2_hg19_splitting_report.txt
rm 20171012_i2_hg19_M-bias.txt
```

```
####################################################################
##########Supplementary_code_to_generate_genome_tracks ############
####################################################################

##################################
#####Generation of CPG tracks#####
##################################

#Environment:
#R v3.4.1
#Bioconductor v3.6
#BiocInstaller v1.28.0
#BSgenome.Hsapiens.UCSC.hg19 v1.4.0
#Biostrings v2.47.0
#rtracklayer v1.38.2
#HiTC v1.22.0
#Bedtools v2.25.0
#Awk v4.0.2


R

source("http://bioconductor.org/biocLite.R")
biocLite("BSgenome.Hsapiens.UCSC.hg19")
library(BSgenome.Hsapiens.UCSC.hg19)
library(Biostrings)

Chrs<-paste("chr",c(1:22,"X","Y","M"),sep="")
CpGpos<-lapply(Chrs,function(x) {
      RES<-matchPattern("CG",Hsapiens[[x]])
      RES<-cbind(rep(x,length(RES)),RES@ranges@start)
      return(RES)})
CpGpos<-do.call("rbind",CpGpos)
CpGstartend<-cbind(CpGpos[,1],as.numeric(CpGpos[,2])-
      1,as.numeric(CpGpos[,2]))
CpGstartendas.data.frame(CpGstartend,stringsAsFactors=F)
CpGstartend[,2]<-as.numeric(CpGstartend[,2])
CpGstartend[,3]<-as.numeric(CpGstartend[,3])
write.table(CpGstartend,file="CpGhg19.bed",sep="\t",quote=F,
      col.names=F,row.names=F)

q()


###################################
#####Generation of MspI tracks#####
###################################

R

source ("http://bioconductor.org/biocLite.R")
biocLite ("BSgenome.Hsapiens.UCSC.hg19")
biocLite("HiTC")
library (BSgenome.Hsapiens.UCSC.hg19)
library(Biostrings)
require(HiTC)
require(rtracklayer)

human_chr <-seqlevels(BSgenome.Hsapiens.UCSC.hg19)[1:25]
resFrag <- getRestrictionFragmentsPerChromosome(resSite="CCGG",
      chromosomes = human_chr, overhangs5 = 2, genomePack =
      "BSgenome.Hsapiens.UCSC.hg19")
allRF <- do.call ("c", resFrag)
```

```
names(allRF)<-unlist(sapply(resFrag, function(x)
     {paste0("HIC_",seqlevels(x),"_",1:length(x))}))
export(allRF, format="bed", con="MspI_resFrag_hg19.bed")

q()

awk '{$2=$2-1;$3=$3+1;print}' MspI_resfrag_hg19.bed >
     MspI_hg19_all.bed
sed -i 's/-1/0/g' MspI_hg19_all.bed
perl -p -i -e 's/ /\t/g' MspI_hg19_all.bed
rm MspI-resfrag_hg19.bed

###############
#####Extra#####
###############

#Generation of the 20-165bp MspI track
#22-167 because a MspI-track is defined as CGGNNN...NNNCCG, 20-165
     would be needed if MspI-track was defined as GGNNN...NNNCC
awk '$3-$2 >= 22 && $3-$2 <= 167' MspI_hg19_all.bed >
     MspI_hg19_22_167.bed

#Generation of the CpG's in the 20-165bp MspI track
bedtools intersect -u -a CpGhg19.bed -b MspI_hg19_22_167.bed >
     CpG_in_MspI_22_167


#CpG's in the SeqCapEpi track (downloaded from Roche website)
130912_HG19_CpGiant_4M_EPI_CpG.bed


#CancerLocator track (downloaded from paper and keep column 2,3,4)
awk '{print($2"\t"$3"\t"$4)}' cpg_clusters_boundaries.tsv >
     cpg_clusters_boundaries.bed
```

```
#####################################################################
#################Supplementary_code_MspI_enrichment###############
#####################################################################

#code to generate Figure 1c
#count unique mapped reads that overlap with MspI-track. Read has to
#     lie completely within the MspI-track and the MspI-fragment has
#     to be covered for 11.976%
#11.976% is the fraction of the shortest read, 20bp, divided by the
#     longest expected MspI-fragment, 165bp+2bp.
#2bp due to the fact MspI-fragments are defined as CGG---CCG
#bin MspI-fragments per 2 lengths: 20&21, 22&23, ... and sum the
#     amount of reads hitting the bins

bedtools coverage -counts -f 0.11976 -F 1 -a
      ../Tracks/MspI_hg19_all.bed -b 20170210_i2_mapping_hg19.bam >
      intermediate_result1
awk '{print $1"\t"$2"\t"$3"\t"$3-$2"\t"$4}' intermediate_result1 >
      intermediate_result2
awk '$5>=1' intermediate_result2 > intermediate_result3
awk '{
      BIN=sprintf("%d", $4*(1/BINSIZE))+0;
      DATA[BIN]=DATA[BIN]+$5;
      if((!MIN)||(MIN>BIN)) MIN=BIN;
      if((!MAX)||(MAX<BIN)) MAX=BIN;
      }
      END {
      for(BIN=MIN; BIN<=MAX; BIN++)
      print((BIN*BINSIZE)"\t"(BIN*BINSIZE)+(BINSIZE-1)"\t"DATA[BIN]);
      }' BINSIZE=2 intermediate_result3 >
      20170210_i2_hg19_reads_hitting_MspI

rm intermediate_result1 intermediate_result2 intermediate_result3


#in excel:       copy table and divide by total unique mapping reads
                 and plot

#also in excel: plot performance, copied from reports:
                 20171012_i2_R2_trimming_report.txt
                 20171012_i2_mapping_hg19_report.txt
                 20171012_i2_mapping_lambda_report.txt
```

```
##################################################################
#################Supplementary_code_CpG_coverage#################
##################################################################

#Figure 2 b
#For cf-RRBS libraries, calculate the amount of CpGs in the MspI-
     target are hit by how many reads, when starting from
     downsampled 1M raw reads
#repeat for 2M, 4M, 8M, 16M
bedtools intersect -wo -a ../Tracks/CpG_in_MspI_22_167.bed -b
     20171012_i11_R1_1M_val_1_bismark_bt2_pe.bismark.cov.gz >
     intermediate_result
bedtools groupby -i intermediate_result -g 1,2,3 -c 8,9 -o sum >
     20171012_i11_1M_CpG_in_hg19

awk '{s+=(($4+$5)>=1)}END{print s}' 20171012_i11_1M_CpG_in_hg19 >>
     sum
awk '{s+=(($4+$5)>=10)}END{print s}' 20171012_i11_1M_CpG_in_hg19 >>
     sum

#For SeqCapEpi libraries, calculate the amount of CpGs in the
     SeqCapEpi target are hit by how many reads, when starting from
     downsampled 1M raw reads
#repeat for 2M, 4M, 8M, 16M
bedtools intersect -wo -a
     ../Tracks/130912_HG19_CpGiant_4M_EPI_CpG.bed -b
     20180615_i14_R1_1M_crop100_val_1_bismark_bt2_pe.bismark.cov.gz
     > intermediate_result
bedtools groupby -i intermediate_result -g 1,2,3 -c 11,12 -o sum >
     20180615_i14_1M_crop100_CpG_in_hg19

awk '{s+=(($4+$5)>=1)}END{print s}'
     20180615_i14_1M_crop100_CpG_in_hg19 >> sum
awk '{s+=(($4+$5)>=10)}END{print s}'
     20180615_i14_1M_crop100_CpG_in_hg19 >> sum

#in excel: plot



#For cf-RRBS libraries, extract the CpG's in the MspI-target that
     are covered by >=10 sequencing reads. Starting from 8M PE75
     reads or 0.6Gbp raw data
#repeat on the 3 technical repeats
bedtools intersect -wo -a ../Tracks/CpG_in_MspI_22_167.bed -b
     20171012_i16_R1_8M_val_1_bismark_bt2_pe.bismark.cov.gz >
     intermediate_result
bedtools groupby -i intermediate_result -g 1,2,3 -c 8,9 -o sum >
     20171012_i16_8M_CpG_in_hg19
awk '($4+$5)>=10' 20171012_i16_8M_CpG_in_hg19 >
     20171012_i16_8M_CpG_in_hg19_10

#For SeqCapEpi libraries, extract the CpG's in the SeqCapEpi-target
     that are covered by >=10 sequencing reads. Starting from 6M
     PE100 reads or 0.6Gbp raw data
#repeat on the 3 technical repeats
bedtools intersect -wo -a
     ../Tracks/130912_HG19_CpGiant_4M_EPI_CpG.bed -b
     20180615_i13_R1_6M_crop100_val_1_bismark_bt2_pe.bismark.cov.gz
     > intermediate_result
bedtools groupby -i intermediate_result -g 1,2,3 -c 11,12 -o sum >
     20180615_i13_6M_crop100_CpG_in_hg19
```

```
awk '($4+$5)>=10' 20180615_i13_6M_crop100_CpG_in_hg19 >
     20180615_i13_6M_crop100_CpG_in_hg19_10

#Make a three-way intersection of the three cf-RRBS technical
     repeats and for the three SeqCapEpi technical repeats
bedtools multiinter -i
     ../20170210_cfRRBS_uzg002_cfDNA_1_i6/20170210_i6_8M_CpG_
     in_hg19_10
     ../20171012_cfRRBS_uzg009_cfDNA_2_i11/20171012_i11_8M_CpG
     _in_hg19_10
     ../20171012_cfRRBS_uzg009_cfDNA_3_i16/20171012_i16_8M_CpG_
     in_hg19_10 >
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16

bedtools multiinter -i
     ../20180615_SeqCapEpi_uzg009_cfDNA_1_i12/20180615_i12_6M_
     crop100_CpG_in_hg19_10
     ../20180615_SeqCapEpi_uzg009_cfDNA_2_i13/20180615_i13_6M_
     crop100_CpG_in_hg19_10
     ../20180615_SeqCapEpi_uzg009_cfDNA_3_i14/20180615_i14_6M_
     crop100_CpG_in_hg19_10 >
     Multiinter_6M_crop100_CpG_20180615_i12_20180615_i13_20180615_
     i14

#Calculate all intersections of a Venn diagram for cf-RRBS and
     SeqCapEpi and do this for the CpG's and the CancerLocator-
     clusters
awk '{s+=$5=="1"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output
awk '{s+=$5=="2"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output
awk '{s+=$5=="1,2"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output
awk '{s+=$5=="3"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output
awk '{s+=$5=="1,3"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output
awk '{s+=$5=="2,3"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output
awk '{s+=$5=="1,2,3"}END{print $5,s}'
     Multiinter_8M_CpG_20170210_i6_20171012_i11_20171012_i16 >>
     Venn_cfRRBS_output

#Make Venn-diagrams in Python for cf-RRBS and SeqCapEpi
#open python
python3.6

#import needed packages
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np
from matplotlib_venn import venn3, venn3_circles

#make weighted venn diagram
plt.figure(figsize=(4,4))
```

```
#subsets=(Abc, aBc, ABc, abC, AbC, aBC, ABC)
v = venn3(subsets=(78370, 54815, 44122, 71567, 45900, 69627,
      298432), set_labels = ('cfRRBS1', 'cfRRBS2', 'cfRRBS3'))
c = venn3_circles(subsets=(78370, 54815, 44122, 71567, 45900, 69627,
      298432))

plt.savefig('Venn_SeqCapEpi_CpG.svg')

quit()

######################################################################

#Figure 2b/c
#Make correlation plots and calculate pearson r-value

#navigate to the mother folder and intersect both files you just
      generated to obtain the CpG's covered by both repeats/methods
bedtools intersect -wo -a
      ../20170210_cfRRBS_uzg009_cfDNA_1_i6/20170210_i6_8M_CpG
      _in_hg19_10 -b
      ../20171012_cfRRBS_uzg009_cfDNA_2_i11/20171012_i11_8M_CpG_
      in_hg19_10 > intermediate1
bedtools intersect -wo -a  intermediate1 -b
      ../20171012_cfRRBS_uzg009_cfDNA_3_i16/20171012_i16_8M_CpG_
      in_hg19_10 > intermediate2
awk '{
      print($1"\t"$2"\t"$3"\t"$4"\t"$5"\t"$4/($4+$5)"\t"$9"\t"$10"\t"
      $9/($9+$10)"\t"$15"\t"$16"\t"$15/($15+$16))} intermediate2 >
      intersect_20170210_i6_20171012_i11_20171012_i16
awk '{print($1"\t"$2"\t"$3"\t"$6"\t"$9"\t"$12)}'
      intersect_20170210_i6_20171012_i11_20171012_i16 >
      intersect_20170210_i6_20171012_i11_20171012_i16_1

rm intermediate1

#Generate correlation plots in python
#open python
python3.6

#load needed packages
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
import pandas as pd
import seaborn as sns
from scipy.stats import pearsonr
from matplotlib.gridspec import GridSpec
import matplotlib.ticker as ticker

#read the intersection file, make a header
df =
      pd.read_csv(‘intersect_20170210_i6_20171012_i11_20171012_i16_1'
      , sep="\t", header=None)
df.columns = ["chr", "start", "stop", "prct_meth_i6",
      "prct_meth_i11", "prct_meth_i16"]
X = df["prct_meth_i11"]
Y = df["prct_meth_i16"]

#make a correlation plot
g = sns.JointGrid(X, Y, xlim=[0,1], ylim=[0,1])
g.plot_joint(plt.scatter, color="#43b7ba", s=5)
```

```
g.plot_marginals(sns.distplot, bins = 50, kde=False,
color="#ec672a", hist_kws={'log':True})

plt.setp(g.ax_joint.set_xlabel('rpt2', fontsize = 20),
     visible=True)
plt.setp(g.ax_joint.set_ylabel('rpt3', fontsize = 20),
     visible=True)

plt.setp(g.ax_joint.get_xticklabels(), fontsize = 15, visible=True)
plt.setp(g.ax_joint.get_yticklabels(), fontsize = 15, visible=True)

plt.setp(g.ax_marg_x.set_ylabel('#CanLoc', fontsize = 20),
     visible=True)
plt.setp(g.ax_marg_y.set_xlabel('#CanLoc', fontsize = 20),
     visible=True)

plt.setp(g.ax_marg_x.set_yticks([1.e+01, 1.e+03]), visible=True)
plt.setp(g.ax_marg_y.set_xticks([1.e+01, 1.e+03]), visible=True)

plt.setp(g.ax_marg_x.get_yticklabels(), visible=True, fontsize = 15)
plt.setp(g.ax_marg_y.get_xticklabels(), visible=True, fontsize = 15)

plt.setp(g.ax_marg_x.yaxis.get_majorticklines(), visible=True)
plt.setp(g.ax_marg_y.xaxis.get_majorticklines(), visible=True)

g.annotate(pearsonr, template = "r = {val:.2g}", frameon=False,
     loc = (-0.2,0.9), fontsize = 20)

plt.tight_layout()
plt.savefig('intersect_cfRRBS_WGBS_sNBLO5.png')

quit()

#similar for SeqCap Epi
#intersect the cf-RRBS intersection file with the SeqCap Epi
intersection file to compare a cf-RRBS repeat with a SeqCap Epi
repeat (fig2c top)
#average the cf-RRBS values and the SeqCap Epi values of the
intersection file of fig2c top before generating the correlation
plot of fig 2c bottom
```

```
## Import libraries
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
plt.switch_backend('Agg')

import glob
import os
import re
import sys, getopt
import numpy as np
import pandas as pd
import seaborn as sns
import subprocess

### BEDtools > 2.27.1 is needed!!
### Tested with py >3.4
### More information about the bed file is at
https://github.com/shulik7/CancerLocator/tree/Add_instruction_and_data

# cfRRBS data
test_folder = "./testfiles/"
test_files = glob.glob(os.path.join(test_folder, "*.cov"))

# Infinium data
NBL_infinium_folder = "./train/NBL"
NBL_infinium_files = glob.glob(os.path.join(NBL_infinium_folder, "*.txt"))
WBC_infinium_folder = "./train/WBC_child/"
WBC_infinium_files = glob.glob(os.path.join(WBC_infinium_folder, "*.bed"))
WT_infinium_folder = "./train/WT/"
WT_infinium_files = glob.glob(os.path.join(WT_infinium_folder, "*.txt"))
Adrenal_infinium_folder = "./train/ADRENAL"
Adrenal_infinium_files = glob.glob(os.path.join(Adrenal_infinium_folder, "*.txt"))

NBLsubcat = "MYCN"

# Folder to store intermediate files
tmp_folder = "./processed/"

clusters = pd.read_csv("./CpG_clusters_b37.tsv", sep="\t",usecols=[1,2,3], skiprows=[0,1],
header=None)
clusters = clusters[[1,2,3]]
clusterFile = "CpG_clusters_b37"
clusters[0] = clusters.index

clusters.to_csv(tmp_folder + "%s.bed" % clusterFile, header=None, index=None, sep='\t', mode =
'w')
# Use empty index to later extract all the clusters from, so that every sample has the same number
of clusters
clusters = clusters.drop([0,1,2,3], axis = 1)

# Load Infinium reference file
array450k = pd.read_csv("./HumanMethylation450_15017482_v1-2.csv", dtype={"CHR": str}, header = 7,
usecols = (0,10,11,12), index_col="IlmnID")
array450k = array450k.dropna()
array450k[['MAPINFO', 'Genome_Build']] = array450k[['MAPINFO', 'Genome_Build']].astype(int)
## Extract locations with genome build GRCh37
array450k = array450k[array450k['Genome_Build'] == 37]
array450k = array450k.drop(['Genome_Build'], axis = 1)
array450k[['CHR', 'MAPINFO']] = array450k[['CHR', 'MAPINFO']].astype(str)
array450k.index.name = None

## Create empty lists to fill up with samples
testMethy_list = []
testDepth_list = []
testBeta_list  = []
print("Generating tesfiles")
for file in test_files:
    # Get base name from file
```

```
    file_name = os.path.splitext(os.path.basename(file))[0]
    # From methylation percentage to methylation ratio
    df = pd.read_csv(file, sep="\t",usecols=[0,1,2,3,4,5], header=None)
    df[3] = df[3]/100
    # Save bismark cov as bed file to process with bedtools
    df.to_csv(tmp_folder + "%s.bed" % file_name , header=None, index=None, sep='\t', mode = 'w')

    ## Get reads per clusters
    # Make a new file with the intersect of the clusters with the cov file
    outfile = open(tmp_folder + '%s_intersect.bed' % file_name, 'w')
    print("     Running bedtools intersect on %s.bed..." % file_name)
    arg = "bedtools intersect -wb -b %s%s.bed -a %s%s.bed" % (tmp_folder, clusterFile, tmp_folder,
file_name)
    arg = arg.split()
    proc = subprocess.Popen(args=arg, stdout=outfile, universal_newlines=True).wait()

    # The previous step shuffles the column order, so this step rearranges the column order
    df = pd.read_csv(tmp_folder + '%s_intersect.bed' % file_name, sep="\t", usecols=
[6,7,8,3,4,5,9], header=None )
    df = df[[6,7,8,3,4,5,9]] # chr, start, stop, beta value, count methylated, count unmethylated,
cluster number
    df.to_csv(tmp_folder + "%s_reordered.bed" % file_name , header=None, index=None, sep='\t',
mode = 'w')

    # Group all the rows that are within a cluster, and get the sum of the methylated and
unmethylated values. Also get the row with the CpG cluster number for future indexing.
    arg = "bedtools groupby -i %s%s_reordered.bed -g 1-3,7 -c 5,6 -o sum" % (tmp_folder,
file_name)
    arg = arg.split()
    outfile = open(tmp_folder + '%s_clustered.bed' % file_name, 'w')
    print("     Running bedtools groupby on %s.bed..." %file_name)
    proc = subprocess.Popen(args=arg, stdout=outfile, universal_newlines=True).wait()

    df = pd.read_csv(tmp_folder + '%s_clustered.bed' % file_name, sep="\t", header=None, index_col
= 3 )
    # Remove index.name for consistence
    df.index.name = None

    # Get beta value per cluster
    df[6] = df[4]/(df[4] + df[5])

    # Reorder the columns
    df = df[[0,1,2,6,4,5]] # chr, start, stop, beta value, no methylated, no unmethylated

    # Sort by chromosome
    df.sort_values(by=[0,1,2], inplace=True)

    # Get total depth (=methylated + unmethylated count)
    df[7] = df[4] + df[5]

    # Mark all clusters lower than 30 reads with NA
    df[[7,6,4,5]] = df[[7,6,4,5]].mask(df[7] < 30)

    # Replace numpy NaN with NA string
    df = df.replace(np.nan, 'NA', regex=True)

    # Make a new variable for visualisation that contains the beta values of the clusters
    testBeta_df = df
    testBeta_df.columns = [0,1,2, "%s" % file_name, 4, 5,7]
    testBeta_df = testBeta_df.drop([0,1,2,4,5,7], axis = 1).astype(str)
    testBeta_list.append(testBeta_df)

# Merge list in pandas df for clustermap
testBeta = pd.concat(testBeta_list, axis = 1)
testBeta = pd.merge(clusters, testBeta, how = "left", left_index=True, right_index=True)
testBeta = testBeta.transpose().fillna('NA')

print("Generating reference")
## Function that gets the average beta value in a cluster,
```

```
## or writes NA if more than half are not available
def getAvg(x):
    if isinstance(x, float):
        return x
    elif len(x) == 0:
        x = "NA"
    else:
        line_values = []
        countNA = 0
        line_values = x.split(',')
        line_values = [i.strip(' ') for i in line_values]
        line_values = list(filter(None, line_values))
        countTot = len(line_values)
        for value in line_values:
            if value == 'NA':
                countNA = countNA +1
        if countTot == 0:
            x = "NA"
        elif countNA/countTot >= 0.5:
            x = "NA"
        else:
            line_values_rmNA = filter(lambda a: a != 'NA', line_values)
            line_values_rmNA_list = list(line_values_rmNA)
            calcmean = np.array(line_values_rmNA_list).astype(np.float)
            x = np.mean(calcmean)
        return x

## The input for this function is ordered an infinium 450k file
## with the order chr - start - stop - beta value
def generateTrain_Infinium(label, file_name):
    # Get reads per clusters
    outfile = open(tmp_folder + "%s_intersect.bed" % file_name, 'w')
    print("     Running bedtools intersect on %s.bed..." % file_name)
    proc = subprocess.Popen(args=["bedtools", "intersect", "-b", tmp_folder + "%s.bed" %
clusterFile, "-a", tmp_folder + "%s.bed" % file_name, "-wb"], stdout=outfile,
universal_newlines=True).wait()

    df = pd.read_csv(tmp_folder + '%s_intersect.bed' % file_name, sep="\t", usecols=[6,3,4,5,7],
header=None )
    df = df[[4,5,6,3,7]]
    df[3] = df[3].replace(np.nan, 'NA', regex=True)
    df.to_csv(tmp_folder + "%s_reordered.bed" % file_name , header=None, index=None, sep='\t',
mode = 'w')

    # Group the total and methylated reads per cluster
    arg = "bedtools groupby -i %s%s_reordered.bed -g 1-3,5 -c 4 -o collapse" % (tmp_folder,
file_name)
    arg = arg.split()
    outfile = open(tmp_folder + '%s_clustered.bed' % file_name, 'w')
    print("     Running bedtools groupby on %s.bed..." %file_name)
    proc = subprocess.Popen(args=arg, stdout=outfile, universal_newlines=True).wait()

    df = pd.read_csv(tmp_folder + '%s_clustered.bed' % file_name, sep="\t", header=None,
index_col=3 )
    df = df[~df.index.duplicated(keep='first')]
    df.index.name = None
    df.sort_values(by=[0,1,2], inplace=True)
    df[4] = df[4].apply(getAvg)
    df.columns = [0,1,2,label]
    df = df.drop([0,1,2], axis = 1)
    return df

### Initialize empty list
trainFile_list = []

# ### Read in NBL files
## Specify files with clinical parameters
MNA = pd.read_table('./train/MNA.txt', sep='\n', header = None)
MA = pd.read_table('./train/MA.txt', header = None)
```

```
## Label the different clinical parameters
if NBLsubcat == "MYCN":
    for file in NBL_infinium_files:
        file_name = os.path.splitext(os.path.basename(file))[0]
        name = file_name.split('.')
        name = str(name[4])
        name = "-".join(name.split("-", 3)[:3])
        # Extract position and beta value
        df = pd.read_csv(file, sep="\t",usecols=[0,1,2,3,4], header=1)
        df["Genomic_Coordinate_Stop"] = df["Genomic_Coordinate"]
        df = df[["Chromosome", "Genomic_Coordinate", "Genomic_Coordinate_Stop", "Beta_value"]]
        df.sort_values(by = ["Chromosome", "Genomic_Coordinate"], inplace=True)
        df.to_csv(tmp_folder + "%s.bed" % file_name , header=None, index=None, sep='\t', mode =
'w')

        if name in MNA.values:
            df = generateTrain_Infinium(label = "nbl-mna", file_name = file_name)
            trainFile_list.append(df)
        elif name in MA.values:
            df = generateTrain_Infinium(label = "nbl-ma", file_name = file_name)
            trainFile_list.append(df)
        else:
            df = generateTrain_Infinium(label = "nbl-nos", file_name = file_name)
            trainFile_list.append(df)
else:
    for file in NBL_infinium_files:
        file_name = os.path.splitext(os.path.basename(file))[0]
        # Extract position and beta value
        df = pd.read_csv(file, sep="\t",usecols=[0,1,2,3,4], header=1)
        df["Genomic_Coordinate_Stop"] = df["Genomic_Coordinate"]
        df = df[["Chromosome", "Genomic_Coordinate", "Genomic_Coordinate_Stop", "Beta_value"]]
        df.sort_values(by = ["Chromosome", "Genomic_Coordinate"], inplace=True)
        df.to_csv(tmp_folder + "%s.bed" % file_name , header=None, index=None, sep='\t', mode =
'w')

        df = generateTrain_Infinium(label = "nbl", file_name = file_name)
        trainFile_list.append(df)

### Similar to NBL, only the column headers are different.
for file in WT_infinium_files:
    file_name = os.path.splitext(os.path.basename(file))[0]
    ## Extract position and beta value
    df = pd.read_csv(file, sep="\t",usecols=[1,2,3,4,5], header=0, dtype={"Position": str,
"Chromosome": str})
    ## Add a stop and reorder the columns
    df["Position_Stop"] = df["Position"]
    df = df[["Chromosome", "Position", "Position_Stop", "AVG_Beta"]]
    df.sort_values(by = ["Chromosome", "Position"], inplace=True)
    df = df.dropna()
    df.to_csv(tmp_folder + "%s.bed" % file_name , header=None, index=None, sep='\t', mode = 'w')

    df = generateTrain_Infinium(label = "wt", file_name = file_name)
    trainFile_list.append(df)

for file in WBC_infinium_files:
    file_name = os.path.splitext(os.path.basename(file))[0]
    ## Open the file
    df = pd.read_csv(file, sep="\t", header=None, index_col=0, names = ["Beta_Value"])
    ## Add the chromosomal position to the sample
    df = pd.merge(array450k, df, how = "inner", left_index=True, right_index=True)
    ## Add a stop and reorder the columns
    df["MAPINFO_Stop"] = df["MAPINFO"]
    df = df[["CHR", "MAPINFO", "MAPINFO_Stop", "Beta_Value"]]
    df.sort_values(by = ["CHR", "MAPINFO"], inplace=True)
    df.to_csv(tmp_folder + "%s.bed" % file_name , header=None, index=None, sep='\t', mode = 'w')

    df = generateTrain_Infinium(label = "wbc", file_name = file_name)
    trainFile_list.append(df)
```

```
for file in Adrenal_infinium_files:
    file_name = os.path.splitext(os.path.basename(file))[0]
    ## Open the file
    df = pd.read_csv(file, sep="\t", header=3, index_col=0, names = ["Beta_Value",
"Pval"]).drop("Pval", axis=1)
    ## Add the chromosomal position to the sample
    df = pd.merge(array450k, df, how = "inner", left_index=True, right_index=True)
    ## Add a stop and reorder the columns
    df["MAPINFO_Stop"] = df["MAPINFO"]
    df = df[["CHR", "MAPINFO", "MAPINFO_Stop", "Beta_Value"]]
    df.sort_values(by = ["CHR", "MAPINFO"], inplace=True)
    df.to_csv(tmp_folder + "%s.bed" % file_name , header=None, index=None, sep='\t', mode = 'w')

    df = generateTrain_Infinium(label = "adrenal", file_name = file_name)
    trainFile_list.append(df)

# Generate full matrix from list
trainFile = pd.concat(trainFile_list, axis = 1)
# Make sure that the file contains all the clusters
trainFile = pd.merge(clusters, trainFile, how = "left", left_index=True, right_index=True)
trainFile = trainFile.transpose().fillna('NA')
trainFile_rmNA = trainFile.select_dtypes(include=['float64'])
print("The number of columns in the trainfile is: %i" %  trainFile.shape[1])
print("The number of columns in the trainfile after removing all NA values is: %i" %
trainFile_rmNA.shape[1])

print("Generating clustermap")

trainFile = trainFile.transpose()
testBeta = testBeta.transpose()
#TotalMatrix = trainFile
TotalMatrix = pd.merge(trainFile, testBeta, how = "left", left_index=True, right_index=True)

outputfolder = "./plots"
TotalMatrix = TotalMatrix.apply(pd.to_numeric, errors='coerce').dropna()

print("The number of remaining rows in the clustermap is %d (after removing rows containing NA
values)" % (len(TotalMatrix)))
TotalMatrix.to_csv('%s/TotalMatrix.csv' % outputfolder, sep=',', mode='w')

## Get colors for each tumor in the plot
TotalMatrix_labels = TotalMatrix.columns.unique()
TotalMatrix_pal = sns.cubehelix_palette(TotalMatrix_labels.unique().size,
                                    light=.9, dark=.3, reverse=True,
                                    start=1, rot=-2)
TotalMatrix_lut = dict(zip(map(str, TotalMatrix_labels.unique()), TotalMatrix_pal))
TotalMatrix_colors = pd.Series(TotalMatrix_lut)

print("Generating UMAP")
TotalMatrix = TotalMatrix.transpose()
## Extract the tumor name from the indices
TotalMatrix['tumor'] = TotalMatrix.index

import umap
X_umap = TotalMatrix.drop("tumor", axis = 1)
y_umap = TotalMatrix['tumor']

print("The number of CpGs in the UMAP-plot is: %i" % len(TotalMatrix.columns))
umap_results = umap.UMAP(n_neighbors=25, min_dist=0.5).fit_transform(X_umap)

umapDf = pd.DataFrame(data = umap_results, columns = ['UMAP 1', 'UMAP 2'])
y_umap = y_umap.values
final_umap_Df = pd.concat([umapDf, pd.DataFrame(y_umap, columns=["target"])], axis = 1)
print("Generating UMAP plots...")
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('UMAP 1', fontsize = 18)
ax.set_ylabel('UMAP 1', fontsize = 18)
ax.set_title('cfRRBS generates similar clinical relevant data like WGBS and SeqCapEpi', fontsize =
```

```
20)
targets = list(TotalMatrix['tumor'].unique())
targets.sort()
targets_tumor = ["adrenal", "nbl-ma", "nbl-mna", "wbc", "wt"]
targets_tumor.sort()
colors_tumor = ["#43b7ba", "#ec672e", "#8392af", "#6f706f", "#e7025f"]
targets_blood = ["NBL-MA cfRRBS", "NBL-MNA cfRRBS"]
targets_blood.sort()
colors_blood = ["#5e2912", "#1b2944"]
targets_WGBS = ["NBL-MA WGBS", "NBL-MNA WGBS"]
targets_WGBS.sort()
colors_WGBS = ["#5e2912", "#1b2944"]
targets_SeqCapEpi = ["NBL-MA SeqCapEpi", "NBL-MNA SeqCapEpi"]
targets_SeqCapEpi.sort()
colors_SeqCapEpi = ["#5e2912", "#1b2944"]
for target, color in zip(targets_tumor,colors_tumor):
    indicesToKeep = final_umap_Df['target'] == target
    print('Number of %s on UMAP plot: %d' % (target,len(final_umap_Df.loc[indicesToKeep])))
    ax.scatter(final_umap_Df.loc[indicesToKeep, 'UMAP 1']
               , final_umap_Df.loc[indicesToKeep, 'UMAP 2']
               , c = color
               , s = 30,
               label = targets_tumor)#, shade=True, shade_lowest=False)
for target, color in zip(targets_blood,colors_blood):
    indicesToKeep = final_umap_Df['target'] == target
    print('Number of %s on UMAPplot: %d' % (target,len(final_umap_Df.loc[indicesToKeep])))
    ax.scatter(final_umap_Df.loc[indicesToKeep, 'UMAP 1']
               , final_umap_Df.loc[indicesToKeep, 'UMAP 2']
               , c = color
               , s = 30
               , marker = 'X',
               label = targets_blood)
for target, color in zip(targets_WGBS,colors_WGBS):
    indicesToKeep = final_umap_Df['target'] == target
    print('Number of %s on UMAP plot: %d' % (target,len(final_umap_Df.loc[indicesToKeep])))
    ax.scatter(final_umap_Df.loc[indicesToKeep, 'UMAP 1']
               , final_umap_Df.loc[indicesToKeep, 'UMAP 2']
               , c = color
               , s = 30
               , marker = 's',
               label = targets_WGBS)
for target, color in zip(targets_SeqCapEpi,colors_SeqCapEpi):
    indicesToKeep = final_umap_Df['target'] == target
    print('Number of %s on UMAP plot: %d' % (target,len(final_umap_Df.loc[indicesToKeep])))
    ax.scatter(final_umap_Df.loc[indicesToKeep, 'UMAP 1']
               , final_umap_Df.loc[indicesToKeep, 'UMAP 2']
               , c = color
               , marker = 'v',)
legend_labels = targets_tumor + targets_blood + targets_WGBS + targets_SeqCapEpi
legend_labels = ",".join(legend_labels)
legend_labels = legend_labels.split(",")
ax.legend(legend_labels, fontsize = 16, bbox_to_anchor=(1.04,1), loc="upper left", ncol=1,
fancybox=True)
ax.grid(False)
fig.show()
fig.savefig('%s/UMAPplot.svg' % (outputfolder), dpi = 300, bbox_inches="tight")
fig.savefig('%s/UMAPplot.png' % (outputfolder), dpi = 300, bbox_inches="tight")
print("Done")
```