Hoang Hai Nam
BINF-F410

Methylation data analysis of the Cancer Genome Atlas's Lung Adenocarcinoma data collection

1. Introduction

DNA methylation alterations including hypermethylation of tumour suppressing genes or hypomethylation of oncogenes are widely recognised as important events in cancer tumour developments. Therefore, analysis of tumour methylation array data can provide great insight into the development of tumour cells and core differences to normal cells. However, due to either the drawbacks of the array technology used, or simply purity of samples, the rate of differentially expressed signals between tumour and normal cells may not always be detectable or evident. This project aims to analyse data of the Cancer Genome Atlas (TCGA)'s Lung Adenocarcinoma (LUAD) dataset, as well as explore ways to improve detection of differentially expressed signals between tumour and normal cells using deconvolution of normal contaminant signals.

2. Methods

2.1. Pre-requisites

This analysis was done in R(ver.4.0.5) and uses the following R packages:

ASCAT.sc, minfi, conumee, GenomicRanges, TCGAbiolinks, TCGAutils, GenomicDataCommons, DNAcopy.

The TCGA-LUAD dataset was downloaded using TCGAbiolinks, and 200 samples were taken randomly based on the set seed "hainamhoang", in total 400 idat files were loaded for analysis.

2.2. Data pre-processing

Using functions from the GenomicDataCommons and TCGAutils packages, the selected samples' idat files' barcodes were identified. Using these TCGA barcodes annotations, normal and tumour samples were identified and tagged. Normal samples were considered to have the sample number from 10 to 19. Total methylation intensities were extracted, and probes were ordered by chromosomal positions. Logarithmic Panel-of-Normal fitted intensity values of probes were derived based on a linear combination of intensities values of the normal samples. All probes were binned using pre-computed bins obtained from "DNA methylation-based classification of central nervous system tumours" repository by Capper et al. and logR tracks were created. Annotations from all probes were obtained using the function getAnnotation.

2.3. Copy-number profiles

ASCAT.sc was used to obtain copy number profiles, the gamma parameter was set to 0.55, this was based on experimental testing detailed in the methylation vignette included in the ASCAT's package.

The gamma parameter, or "platform-dependent technology parameter" is related to the following equation(van Loo et al., 2010):

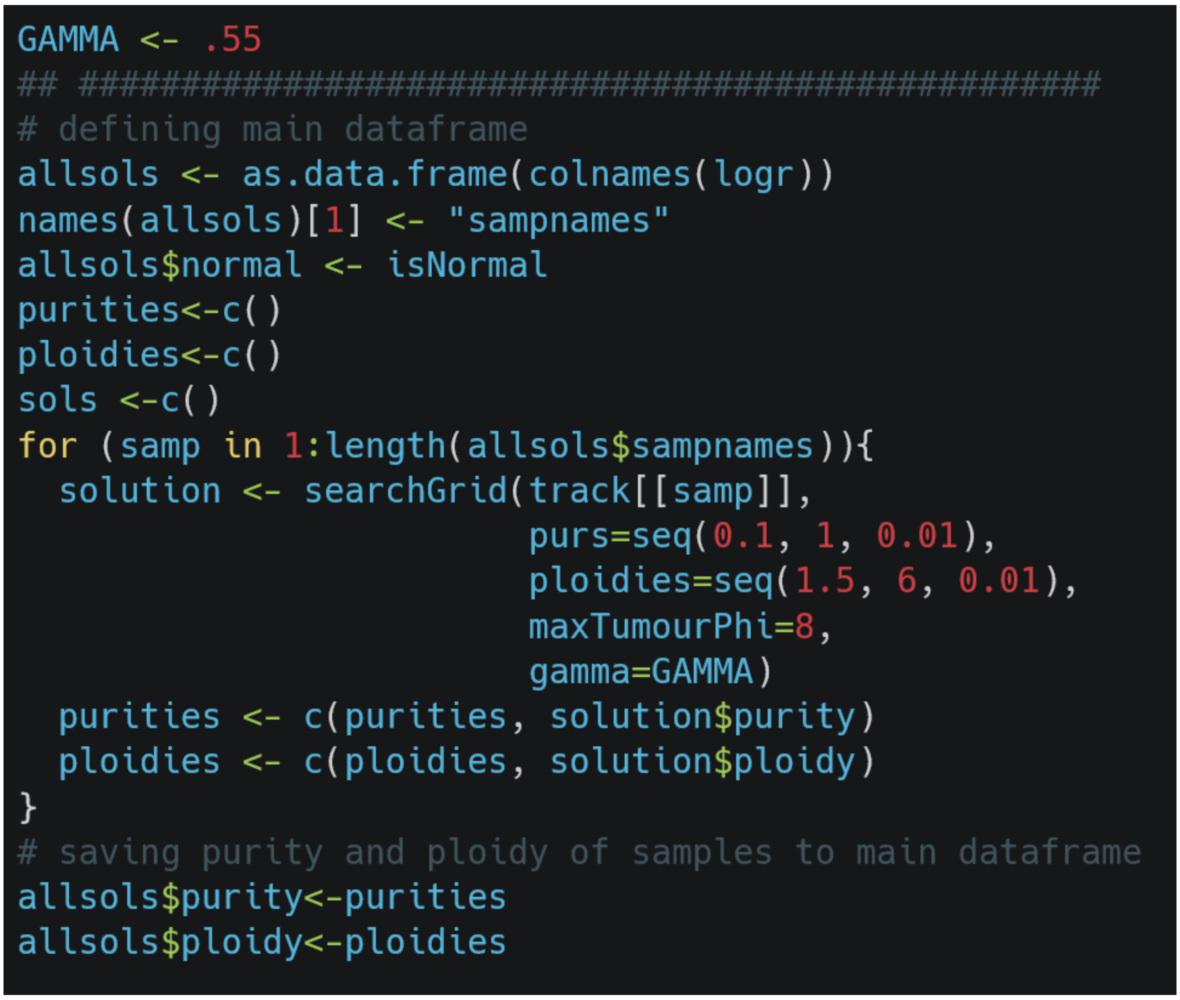
$$logr = \boldsymbol{\gamma} \log_2\left(\frac{2(1-\rho) + \rho n_{tot}}{\Psi}\right)$$

Where ρ is the purity and Ψ the ploidy. Due to complications relating to the array technology, hybridisation-based derived logr is non-linearly proportional to the DNA content. This gamma parameter differs depending on which array technology that is used and is experimentally calibrated.

First through using the searchGrid function with purities' hard limits set between 0.1 and 1, and ploidies' set between the range of 1.5 and 6, as with ploidies outside this range, biologically the cell cannot survive. With the purity and ploidy values of each sample, a copy-number profile would be constructed, using the function fitProfile.

No actual refitting was done other than the hard limits set for purity and ploidies, as I could not find a good consensus on average or supposed "correct" or "average" range of values within available papers, at least not specifically for the TCGA-LUAD dataset. In addition, refitting an entire dataset to a biased purity or ploidy values without being able to figure out if their methods of obtaining those values is better or worse, arbitrarily refitting might not be the optimal step.

The above steps were applied to each sample, purities and ploidies values were stored in a summary data frame called allsols.



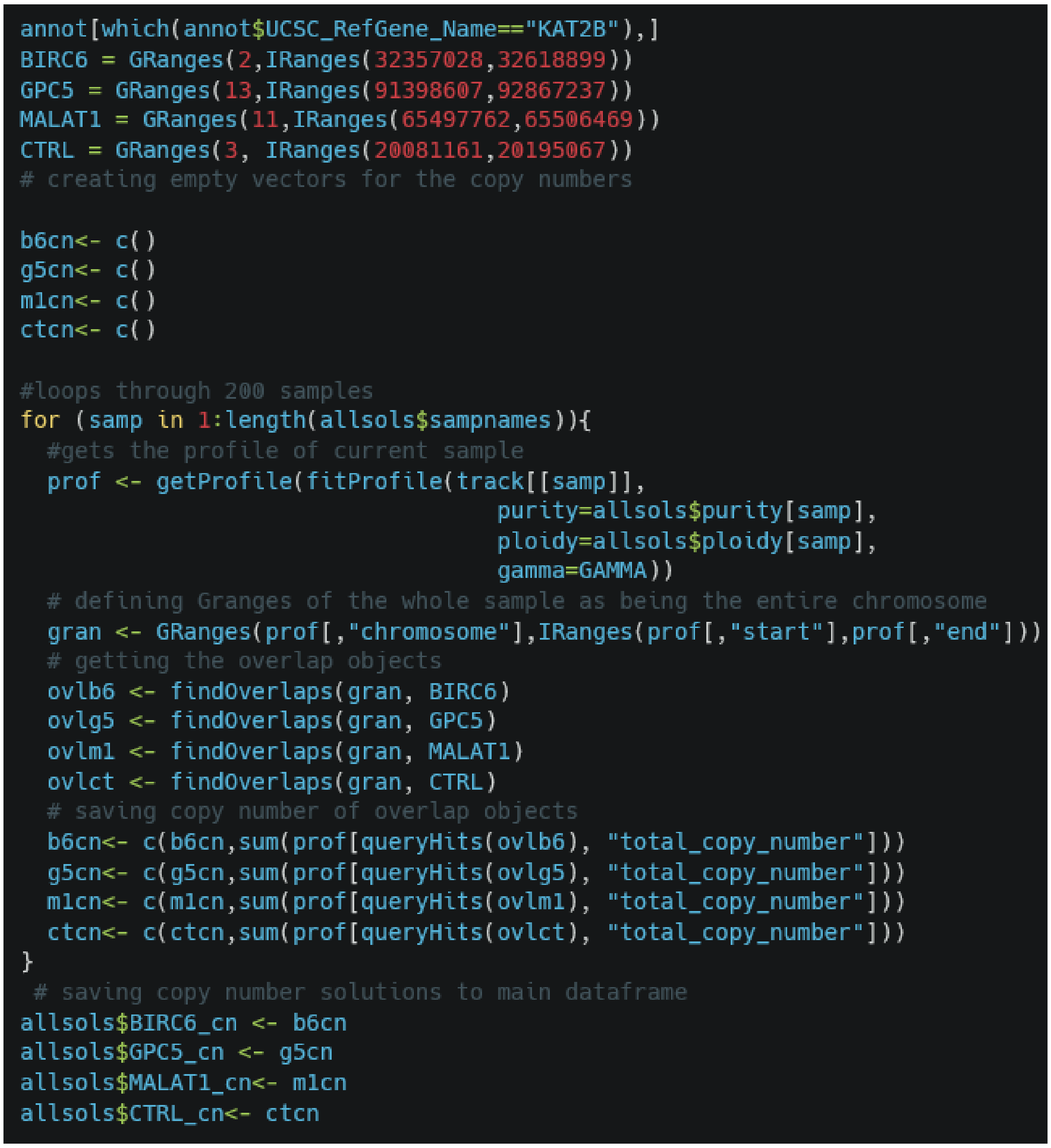
```r
GAMMA <- .55
## ###################################################
# defining main dataframe
allsols <- as.data.frame(colnames(logr))
names(allsols)[1] <- "sampnames"
allsols$normal <- isNormal
purities<-c()
ploidies<-c()
sols <-c()
for (samp in 1:length(allsols$sampnames)){
  solution <- searchGrid(track[[samp]],
                         purs=seq(0.1, 1, 0.01),
                         ploidies=seq(1.5, 6, 0.01),
                         maxTumourPhi=8,
                         gamma=GAMMA)
  purities <- c(purities, solution$purity)
  ploidies <- c(ploidies, solution$ploidy)
}
# saving purity and ploidy of samples to main dataframe
allsols$purity<-purities
allsols$ploidy<-ploidies
```

Figure 1: Code snippet for obtaining purity and ploidy values

2.4. Analysis of genes of interest

Through the use of the COSMIC Cancer Gene Census(Sondka et al., 2018), 3 genes were picked as genes of interest that has been found to play a role in lung adenocarcinoma tumours, to analyse the copy number profiles at these genes. They include BIRC6(Dong et al., 2013), GPC5(Yuan et al., 2016) and MALAT1(Gutschner et al., 2013). Alongside this, a randomly, somewhat arbitrarily picked gene, named KAT2B was included as somewhat of a control, COSMIC reports no association with any type of cancer.

The gene's genomic ranges were defined with the function GRanges, for each sample, the copy number profile was obtained using ASCAT's getProfile and fitProfile functions and overlaps between the profile and each gene's GRange were pulled. For each overlap, the copy number was obtained, and the total copy number sums are recorded in the main data frame.



```r
annot[which(annot$UCSC_RefGene_Name=="KAT2B"),]
BIRC6 = GRanges(2,IRanges(32357028,32618899))
GPC5 = GRanges(13,IRanges(91398607,92867237))
MALAT1 = GRanges(11,IRanges(65497762,65506469))
CTRL = GRanges(3, IRanges(20081161,20195067))
# creating empty vectors for the copy numbers

b6cn<- c()
g5cn<- c()
m1cn<- c()
ctcn<- c()

#loops through 200 samples
for (samp in 1:length(allsols$sampnames)){
  #gets the profile of current sample
  prof <- getProfile(fitProfile(track[[samp]],
                                   purity=allsols$purity[samp],
                                   ploidy=allsols$ploidy[samp],
                                   gamma=GAMMA))
  # defining Granges of the whole sample as being the entire chromosome
  gran <- GRanges(prof[,"chromosome"],IRanges(prof[,"start"],prof[,"end"]))
  # getting the overlap objects
  ovlb6 <- findOverlaps(gran, BIRC6)
  ovlg5 <- findOverlaps(gran, GPC5)
  ovlm1 <- findOverlaps(gran, MALAT1)
  ovlct <- findOverlaps(gran, CTRL)
  # saving copy number of overlap objects
  b6cn<- c(b6cn,sum(prof[queryHits(ovlb6), "total_copy_number"]))
  g5cn<- c(g5cn,sum(prof[queryHits(ovlg5), "total_copy_number"]))
  m1cn<- c(m1cn,sum(prof[queryHits(ovlm1), "total_copy_number"]))
  ctcn<- c(ctcn,sum(prof[queryHits(ovlct), "total_copy_number"]))
}
 # saving copy number solutions to main dataframe
allsols$BIRC6_cn <- b6cn
allsols$GPC5_cn <- g5cn
allsols$MALAT1_cn<- m1cn
allsols$CTRL_cn<- ctcn
```

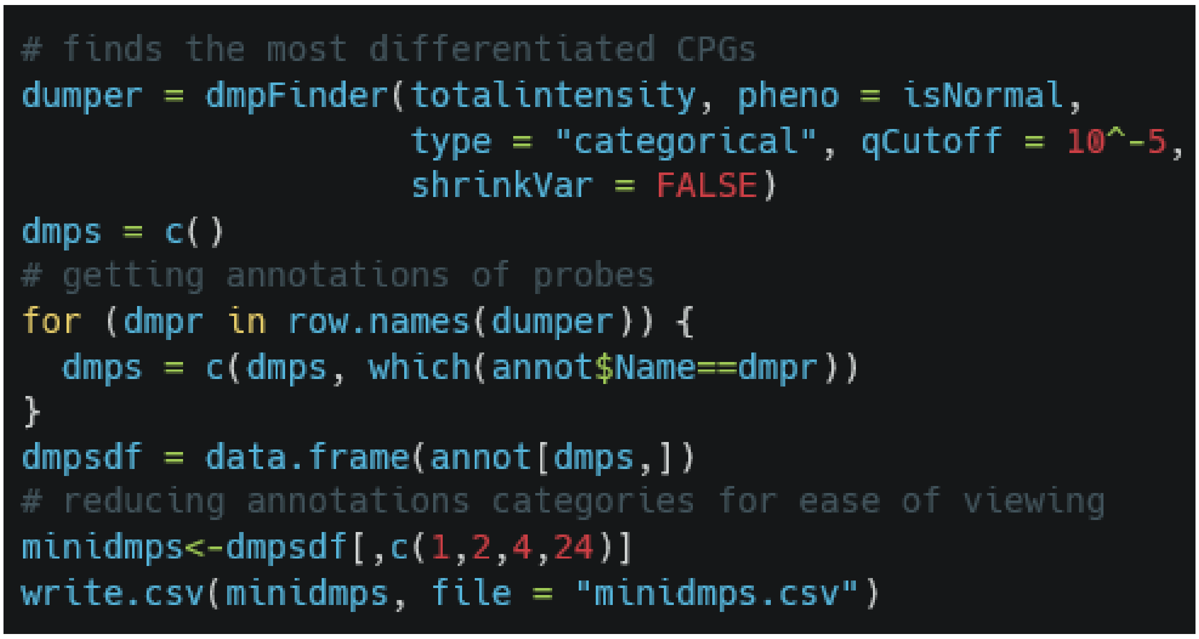
Figure 2: Code snippet for obtaining the copy number values for Genes of Interest

sampnames	normal	purity	ploidy	BIRC6_cn	GPC5_cn	MALAT1_c	CTRL_cn	CTRL_cn
6042308099_R01C01	TRUE	1	2	2	2	2	2	2
6042308099_R02C02	FALSE	1	2.01	2	2	2	2	2
6042308099_R03C01	FALSE	1	2.01	2	2	2	2	2
6042308138_R02C02	TRUE	1	2	2	2	2	2	2
6042308156_R04C01	TRUE	1	2	2	2	2	2	2
6042308156_R06C02	FALSE	0.78	3.81	4	3	5	4	4
6042316106_R03C01	TRUE	1	2	2	2	2	2	2
6264488026_R01C01	FALSE	0.86	3.67	4	4	4	4	4
6264488026_R02C01	FALSE	1	2.01	2	2	2	2	2
6264488026_R04C02	FALSE	0.61	2	2	2	2	2	2

Figure 3: Partial look at the allsols data frame

2.5. Finding differentially methylated positions

In order to search for differentially methylated positions (DMPs), the function dmpFinder from the minfi package was used, with a False Discovery Rate (FDR) cut-off set to 10^{-5}(qCutoff). With this cut-off, the number of found differentiated positions were limited to 13. For each probe, annotation was looked up in order to get information on the probe.



```r
# finds the most differentiated CPGs
dumper = dmpFinder(totalintensity, pheno = isNormal,
                   type = "categorical", qCutoff = 10^-5,
                   shrinkVar = FALSE)
dmps = c()
# getting annotations of probes
for (dmpr in row.names(dumper)) {
  dmps = c(dmps, which(annot$Name==dmpr))
}
dmpsdf = data.frame(annot[dmps,])
# reducing annotations categories for ease of viewing
minidmps<-dmpsdf[,c(1,2,4,24)]
write.csv(minidmps, file = "minidmps.csv")
```

Figure 4: Code snippet for finding differentiated cPGs

2.6. Deconvolution of normal contaminant signals

This step was done only to selected probes, specifically probes within genes of interest. As well as being done on beta values instead of methylation intensities. Only the tumorous samples had their beta signal deconvoluted.

Based on the following equation from the publication of CAMDAC(Cadieux et al., 2020), a not yet published deconvolution analysis tool:

$$m_b = \frac{\rho n_t m_t + n_n m_n (1-\rho)}{\rho n_t + n_n (1-\rho)}$$

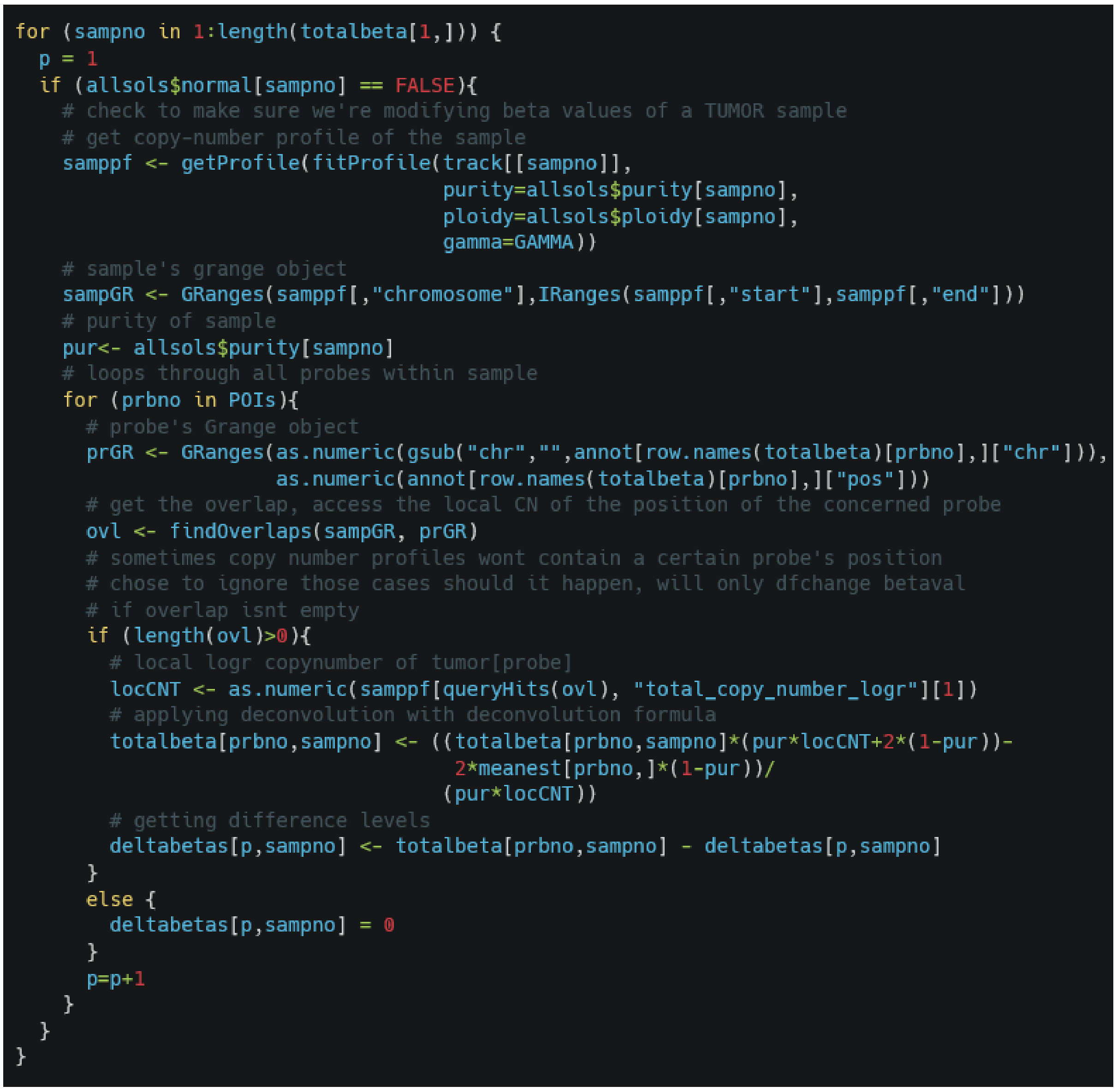
In which the bulk tumour methylation signal $\boldsymbol{m_b}$ is given by different components including the normal and tumour cells methylation signals $\boldsymbol{m_n}$ and $\boldsymbol{m_t}$, the purity $\boldsymbol{\rho}$, the local copy number of the normal and tumour cells $\boldsymbol{n_n}$ and $\boldsymbol{n_t}$. The "pure" tumour cells methylation signal $\boldsymbol{m_t}$ can be derived as:

$$m_t = \frac{m_b\big(\rho n_t + n_n(1-\rho)\big) - n_n m_n (1-\rho)}{\rho n_t}$$

This is done using a nested, double loop, one through all samples, and one through the selected probes. For each sample, a copy-number profile is generated using purity and ploidy numbers saved in the allsols data frame. With this copy-number profile, we can infer local copy-number to the probe being analysed and obtain $\boldsymbol{m_n}$ and $\boldsymbol{m_t}$.

As for the values of the normal cells including $\boldsymbol{n_n}$ and $\boldsymbol{m_n}$, local copy number of the normal cells are set to 2, while the methylation signals are derived from the arithmetic mean of the methylation signal at each probe across the 14 normal samples.

The deconvolution was done on defined list of probe indexes, for this analysis only selected probes were deconvoluted, specifically probes that were marked by annot as being within the gene BIRC6. This could technically be expanded to be done on all probes; and is coded to be able to receive any gene names and deconvolute probes associated with them; however, deconvolution of all probes of 200 samples estimated to take 200 hours of computation, partially due to the hard calculations of the copy numbers.



```r
for (sampno in 1:length(totalbeta[1,])) {
  p = 1
  if (allsols$normal[sampno] == FALSE){
    # check to make sure we're modifying beta values of a TUMOR sample
    # get copy-number profile of the sample
    samppf <- getProfile(fitProfile(track[[sampno]],
                                    purity=allsols$purity[sampno],
                                    ploidy=allsols$ploidy[sampno],
                                    gamma=GAMMA))
    # sample's grange object
    sampGR <- GRanges(samppf[,"chromosome"],IRanges(samppf[,"start"],samppf[,"end"]))
    # purity of sample
    pur<- allsols$purity[sampno]
    # loops through all probes within sample
    for (prbno in POIs){
      # probe's Grange object
      prGR <- GRanges(as.numeric(gsub("chr","",annot[row.names(totalbeta)[prbno],]["chr"])),
                      as.numeric(annot[row.names(totalbeta)[prbno],]["pos"]))
      # get the overlap, access the local CN of the position of the concerned probe
      ovl <- findOverlaps(sampGR, prGR)
      # sometimes copy number profiles wont contain a certain probe's position
      # chose to ignore those cases should it happen, will only dfchange betaval
      # if overlap isnt empty
      if (length(ovl)>0){
        # local logr copynumber of tumor[probe]
        locCNT <- as.numeric(samppf[queryHits(ovl), "total_copy_number_logr"][1])
        # applying deconvolution with deconvolution formula
        totalbeta[prbno,sampno] <- ((totalbeta[prbno,sampno]*(pur*locCNT+2*(1-pur))-
                                     2*meanest[prbno,]*(1-pur))/
                                    (pur*locCNT))
        # getting difference levels
        deltabetas[p,sampno] <- totalbeta[prbno,sampno] - deltabetas[p,sampno]
      }
      else {
        deltabetas[p,sampno] = 0
      }
      p=p+1
    }
  }
}
```

Figure 5: Code snippet of loop used to calculate and apply deconvoluted tumour signals.

3. Results and discussion

3.1. Analysis of genes of interest

This analysis will focus mostly on the main data frame built, as expected, all copy number of all genes within the 14 normal samples have a value of 2, suggesting that the inferred copy number is not likely erroneous.

The results for copy numbers of the 186 tumour samples are summarised in the following table:

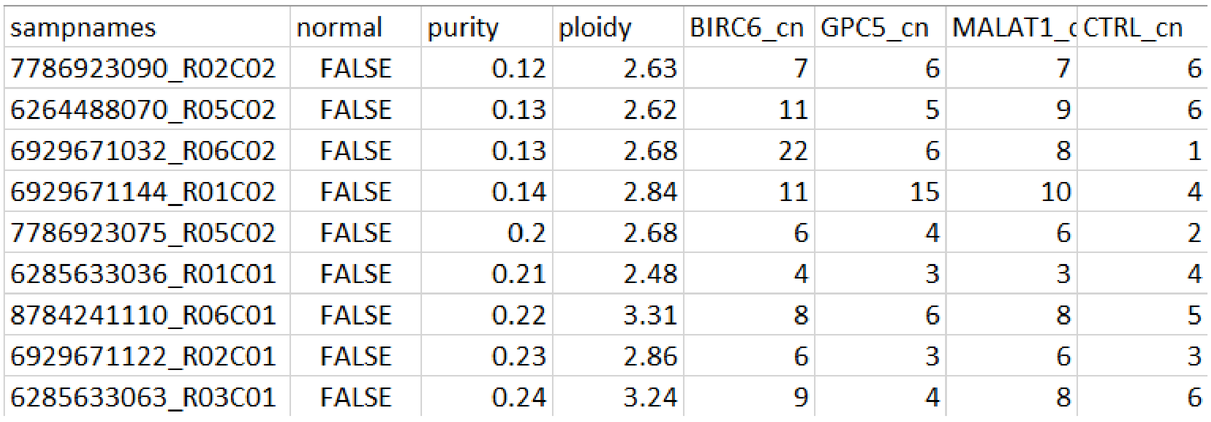
Gene	Amplified (>)	Supressed (<2)	No change (=2)
BIRC6	50.54%	1.08%	48.39%
GPC5	48.39%	5.38%	46.24%
MALAT1	51.08%	0%	48.92%
KAT2B	45.70%	2.69%	51.61%

Table 1: Percentages of Amplified, Supressed or No change for copy-number of each gene across 186 tumour samples

Across all tumour samples, a small but significant percentage had deletion of the GPC5 gene, consistent with the literature(Yuan et al., 2016), this gene has been observed to be under expressed in lung adenocarcinoma tissues, and its low expression was linked with its hypermethylation. As for BIRC6 and MALAT1, there seems to be a slightly higher number of amplified samples, somewhat consistent with previous findings, BIRC6's elevated expression had been found to relate to the chemoresistance of non-small-cell lung cancer(Dong et al., 2013), and MALAT1 is a known prognostic marker for lung cancer metastasis(Gutschner et al., 2013). None of the tumour samples shows deletion of the MALAT1 gene, suggesting its importance in tumour cell viability.

As for the randomly picked gene, KAT2B, it can be used to somewhat visualise the "normal" gene that might not be involved in cancer cell development but is still modified as cells metastasise. While these results look strange, as one might expect a gene that plays little or no known roles should have consistent, no change in copy number. However, the differences between them are consistent with previous research, BIRC6 and MALAT1 have been shown as being overexpressed in cancer tissue, and here, they are the most frequently amplified, meanwhile GPC5, known to be under expressed in cancer tissue, is the most frequently supressed, albeit slightly more amplified than our control. It is then, notable that most if not all DNA content within a tumorous cell is impacted in a somewhat consistent way, as shown by our control gene, and that the differentiation is difficult to reveal.

Another interesting piece of data is the distribution of the copy numbers, analysing this also gives us some insight into the samples. For BIRC6, GPC5, MALAT1, the highest derived copy numbers were 22, 15, 10, greatly higher than the averages across tumour samples of 3.69, 3.20, and 3.59 respectively. These top copy numbers are only from 4 samples, 2 of which are from MALAT1. Out of these 4 samples, 3 had an inferred purity below 0.14. These results could be due to aberrant data or is correlated with the low purity. Pearson tests were done, and there is a somewhat negative correlation between purity and all the genes at an average of -0.58.



sampnames	normal	purity	ploidy	BIRC6_cn	GPC5_cn	MALAT1_c	CTRL_cn
7786923090_R02C02	FALSE	0.12	2.63	7	6	7	6
6264488070_R05C02	FALSE	0.13	2.62	11	5	9	6
6929671032_R06C02	FALSE	0.13	2.68	22	6	8	1
6929671144_R01C02	FALSE	0.14	2.84	11	15	10	4
7786923075_R05C02	FALSE	0.2	2.68	6	4	6	2
6285633036_R01C01	FALSE	0.21	2.48	4	3	3	4
8784241110_R06C01	FALSE	0.22	3.31	8	6	8	5
6929671122_R02C01	FALSE	0.23	2.86	6	3	6	3
6285633063_R03C01	FALSE	0.24	3.24	9	4	8	6

Figure 6: 10 samples with the lowest purity

This is rather surprising, as inferred copy-numbers from a low purity sample logically should fall towards being diploid, CN of 2. As in low purity samples, the assumption is that there're more normal samples, and so the bulk signal should resemble that of a normal cell more than a tumour cell.

3.2. Searching for differentiated cPGs

Using dmpFinder with the parameters mentioned above, 13 cPGs were recorded to be differentiated cPGs that had the lowest FDR. Their shortened annotations are in the following table:

	chr	pos	Name	UCSC_RefGene_Name
cg04864807	chr2	121412139	cg04864807	
cg05973398	chr21	36420164	cg05973398	RUNX1
cg14754787	chr17	35299600	cg14754787	LHX1
cg05563515	chr6	30039027	cg05563515	RNF39;RNF39
cg08566455	chr2	130971164	cg08566455	
cg14789818	chr1	227748712	cg14789818	
cg25774643	chr11	627175	cg25774643	SCT
cg26521404	chr7	27204981	cg26521404	HOXA9
cg20300343	chr1	149719461	cg20300343	

Table 2: Differentiated cPG with FDR<10^(-5)

Only some of these cPGs are associated with a known, named gene, and none had any literature on them relating to tumorous cells. All these cPGs were found to be hypermethylated within tumorous cells, it is notable that none are hypomethylated.

3.3. Deconvolution of normal contaminant signals

For the deconvolution, the chosen value for the normal methylation signal is an average of the signal at each probe across 14 normal samples, this makes the most sense as for a normal methylation signal value.

Before any deconvolution is done, we can visualise methylation beta values at some CPGs using plotCpg. The following plot is of the probe cg20300343 within the gene of interest BIRC6.

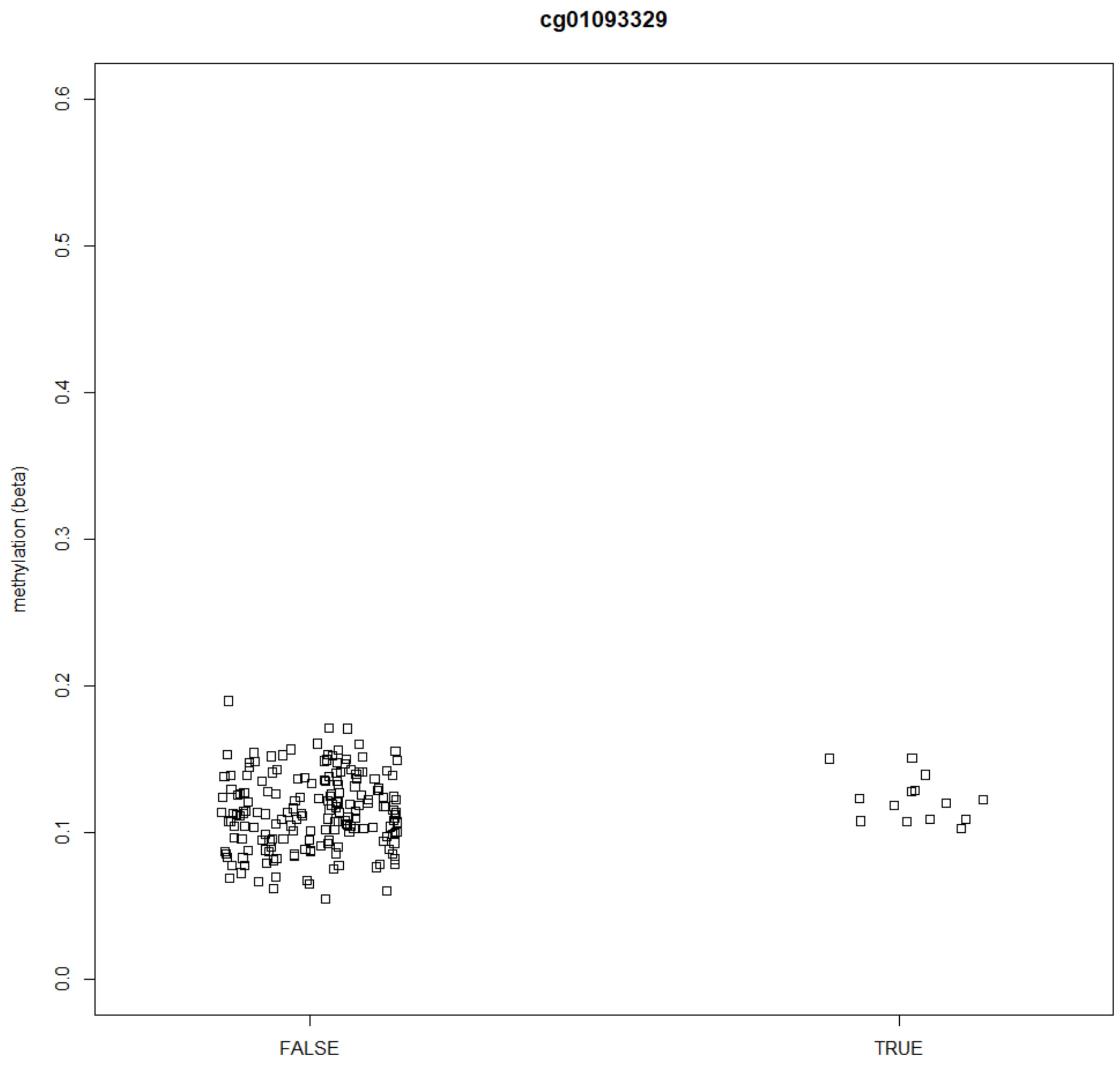


Figure 7: cPG plot of a probe within BIRC6 before deconvolution, False being tumour samples

After the deconvolution, we can observe that a part of the samples had their signals increased, and part of them decreased. This is consistent with the deconvolution formula as if the bulk signal is lower than the normal signal, and that the purity is high (more tumour cells), then the tumour signal should be lower than the bulk signal, and vice-versa.

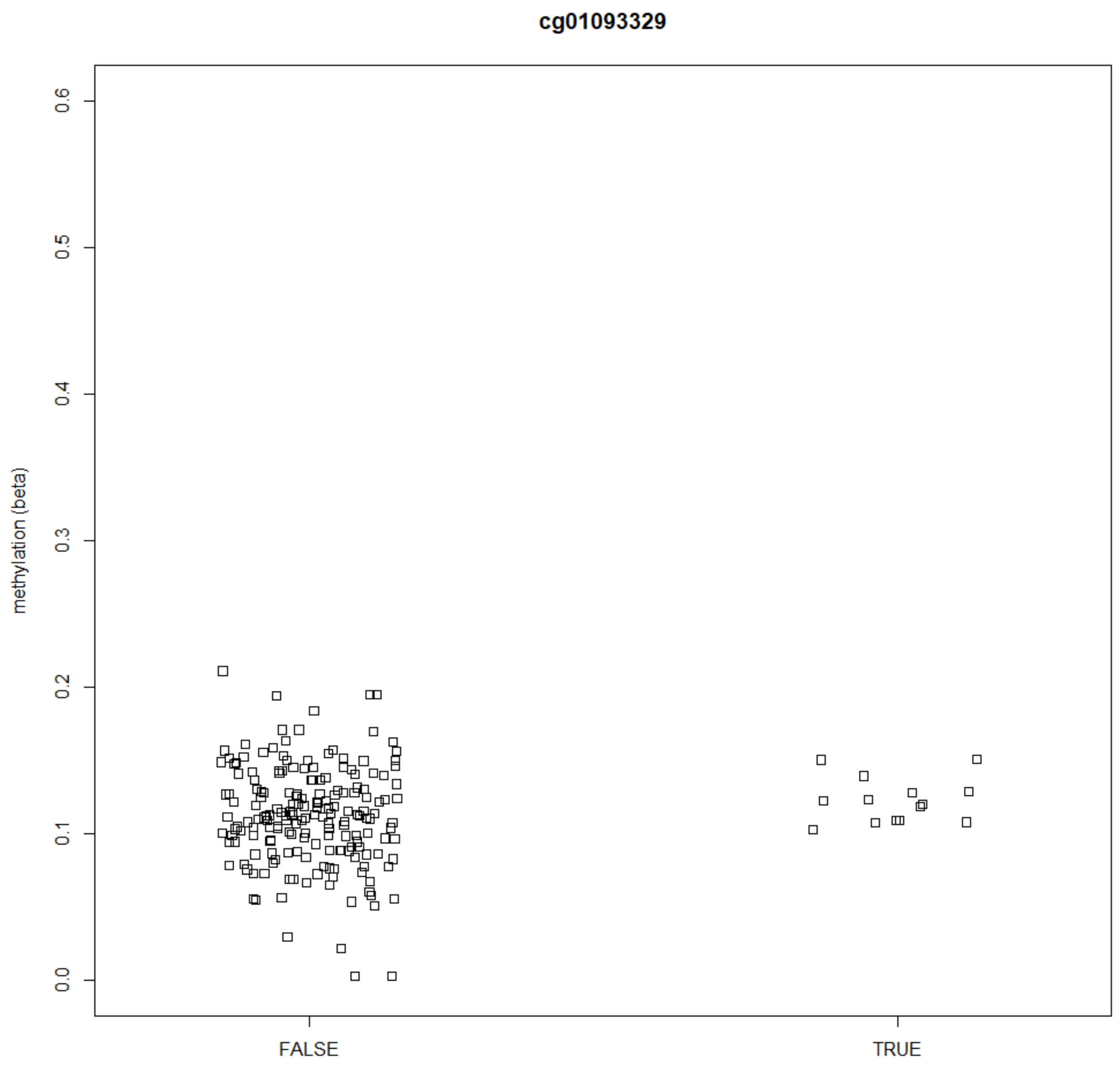


Figure 8: cPG plot of a probe within BIRC6 after deconvolution

We can observe the same deconvolution method on a different CPG, this time of a probe within the GPC5, a gene that tends to be supressed in tumour cells. Here however, the change is much more substantial, as deconvoluted tumour samples reach much lower beta values. This is consistent with prior knowledge, as GPC5 tends to be more supressed within tumour samples. The deconvolution does indeed amplify the differential expression between normal and tumour samples.

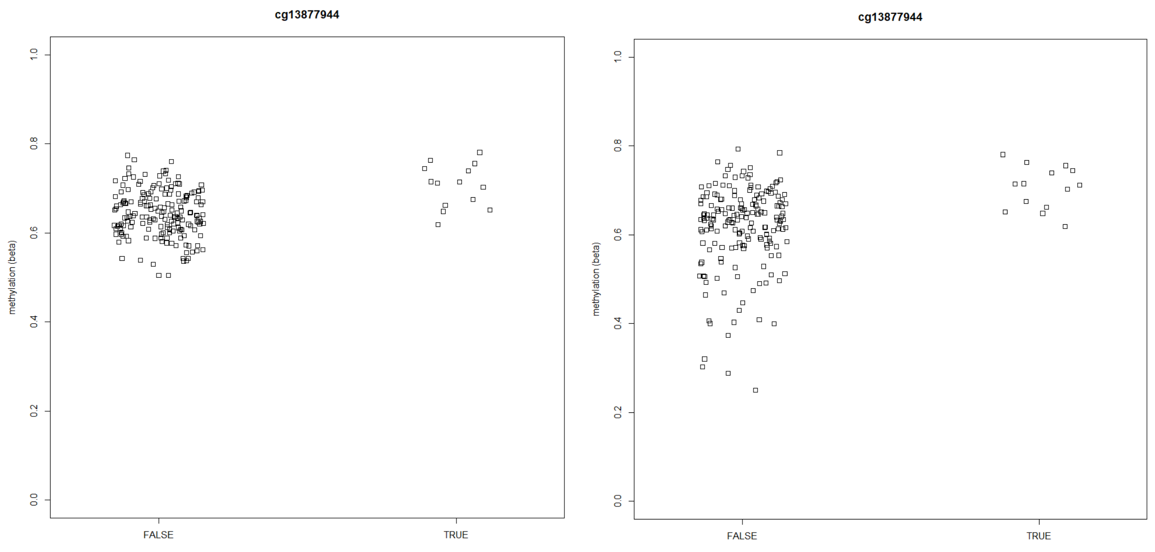


Figure 9: cPG plots of cg13877944 before(left) and after(right) deconvolution

The deconvolution model presented by the authors of CAMDAC seems to be rather effective at making probes more differentiated between normal and tumour samples. Whether this method should be applied to all methylation analysis is hard to say, as while it can reveal more phylogenetic relationship between samples as it makes differences more apparent, there's not much evidence supporting its accuracy, and proneness to false positives. In the publication of CAMDAC they evaluated the accuracy of ASCAT's inferred purity, ploidy, and logR estimates with agreements to Whole Genome Sequencing. However, evaluations of the accuracy and possible drawbacks of deconvolution was not sufficiently discussed. As it stands, the given formula decomposition of the bulk methylation signal is another model (just like ASCAT's logR) with strong biological basis and potentially a powerful tool to improve methylation data analysis, however, it should be subjected to more accuracy testing to fully realise the drawbacks and possible increase rate of false positives.

4. Conclusion

Using ASCAT.sc, copy-number of different genes were analysed and found to be somewhat consistent with the literature. Across the dataset of 200 samples from TCGA-LUAD, differentiated cPGs were identified, however they were not found to be associated with any previously identified oncogenes.

Through a model of bulk methylated signal by the authors of CAMDAC, deconvolution was tested and visualised. This was found to be effective at amplifying differential expression, however, more evaluation needs to be done to prove the model's accuracy, as a means to also evaluate the accuracy of any deconvolution based upon such model.

5. Bibliography

Cadieux, E. L., Tanić, M., Wilson, G. A., Baker, T., Dietzen, M., Dhami, P., Vaikkinen, H., Watkins, T. B. K., Kanu, N., Veeriah, S., Jamal-Hanjani, M., McGranahan, N., Feber, A., Swanton, C., consortium, on behalf of the Tracer., Beck, S., Demeulemeester, J., & van Loo, P. (2020). Copy number-aware deconvolution of tumor-normal DNA methylation profiles. *BioRxiv*, 2020.11.03.366252. https://doi.org/10.1101/2020.11.03.366252

Dong, X., Lin, D., Low, C., Vucic, E. A., English, J. C., Yee, J., Murray, N., Lam, W. L., Ling, V., Lam, S., Gout, P. W., & Wang, Y. (2013). Elevated Expression of BIRC6 Protein in Non–Small-Cell Lung Cancers is Associated with Cancer Recurrence and Chemoresistance. *Journal of Thoracic Oncology*, *8*(2). https://doi.org/10.1097/JTO.0b013e31827d5237

Gutschner, T., Hämmerle, M., & Diederichs, S. (2013). MALAT1 — a paradigm for long noncoding RNA function in cancer. *Journal of Molecular Medicine*, *91*(7). https://doi.org/10.1007/s00109-013-1028-y

Sondka, Z., Bamford, S., Cole, C. G., Ward, S. A., Dunham, I., & Forbes, S. A. (2018). The COSMIC Cancer Gene Census: describing genetic dysfunction across all human cancers. *Nature Reviews Cancer*, *18*(11). https://doi.org/10.1038/s41568-018-0060-1

van Loo, P., Nordgard, S. H., Lingjaerde, O. C., Russnes, H. G., Rye, I. H., Sun, W., Weigman, V. J., Marynen, P., Zetterberg, A., Naume, B., Perou, C. M., Borresen-Dale, A.-L., & Kristensen, V. N. (2010). Allele-specific copy number analysis of tumors. *Proceedings of the National Academy of Sciences*, *107*(39), 16910–16915. https://doi.org/10.1073/pnas.1009843107

Yuan, S., Yu, Z., Liu, Q., Zhang, M., Xiang, Y., Wu, N., Wu, L., Hu, Z., Xu, B., Cai, T., Ma, X., Zhang, Y., Liao, C., Wang, L., Yang, P., Bai, L., & Li, Y. (2016). GPC5, a novel epigenetically silenced tumor suppressor, inhibits tumor growth by suppressing Wnt/β-catenin signaling in lung adenocarcinoma. *Oncogene*, *35*(47). https://doi.org/10.1038/onc.2016.149

Hoang Hai Nam
BINF-F410

6. The full R code used for analysis

```
## ####################################################
## These are the R libraries used in the code
## They are all loaded individually through a call to "library()"
library(ASCAT.sc)
library(minfi)
library(conumee)
## ####################################################
library(GenomicRanges)
library(TCGAbiolinks)
library(TCGAutils)
## ####################################################
library("GenomicDataCommons")
## ####################################################
library(DNAcopy)
## ####################################################
library(data.table)
library(kableExtra)
## ####################################################
#------------------------------------------------------
# Example to idat files from TCGA projects
#------------------------------------------------------
projects <- TCGAbiolinks:::getGDCprojects()$project_id
projects <- projects[grepl('^TCGA',projects,perl=T)]
tcgaproj <- data.frame(projects=projects)
knitr::kable(tcgaproj, format="html") %>%
  kable_styling() %>%
  scroll_box(width = "100%", height = "200px")
####################################################
match.file.cases.all <- NULL
projects.student=projects[projects%in%c("TCGA-LUAD")]
for(proj in projects.student){
  print(proj)
  query <- GDCquery(project = proj,
                    data.category = "Raw microarray data",
                    data.type = "Raw intensities",
                    experimental.strategy = "Methylation array",
                    legacy = TRUE,
                    file.type = ".idat",
                    platform = "Illumina Human Methylation 450")
  match.file.cases <- getResults(query,cols=c("cases","file_name"))
  match.file.cases$project <- proj
  match.file.cases.all <- rbind(match.file.cases.all,match.file.cases)
  tryCatch(GDCdownload(query, method = "api", files.per.chunk = 20),
           error = function(e) GDCdownload(query, method = "client"))
}
##############################################################################
###############################DONE######################################
## This will create a map between idat file name, cases (barcode) and project
readr::write_tsv(match.file.cases.all, path =  "idat_filename_case.txt")

## The following code detects all input files in the downloaded folders
allfs=paste0(getwd(),"/",dir("GDCdata",rec=T, pattern="idat",full=T))
allfsF=paste0(getwd(),"/",dir("GDCdata",rec=T, pattern="idat",full=F))
```

```
seq1=seq(1,length(allfs),2)
seq2=seq(2,length(allfs),2)

## Set seed here to get your own dataset to study
getINTEGER <- function(STRING)
  round(log(as.numeric(paste(sapply(strsplit(STRING,split="")[[1]],function(x)
which(letters==x)),collapse="")))*10000000)

nameseed = getINTEGER("hainamhoang")

set.seed(nameseed)

## Here the sample number was set to 200
NSAMPLES=200
indices=sample(1:length(seq1),min(NSAMPLES,length(seq1)),rep=F)
allindices=c(seq1[indices],seq2[indices])
allfs.=allfs[order(gsub("(.*)/(.*)","\\2",allfs))[allindices]] ## only take 100 files == 50 samples

## This is a system call to create a new directory "allidat", which will contain all the input data
shell(paste0("mkdir ","allidat"))

## Under Windows, ln -s is mklink and the syntax if reversed: first link and then target
## Also, these should be quoted and you need to make sure to allow SymLinkCreation on your Windows
system
## Or run the command or R/Rstudio as Administrator (right-click: Run As Admin).
tmp.windows=sapply(allfs.,function(x)
{
  allidat <- paste0(getwd(),"/allidat")
  filename <- gsub("(.*)/(.*)","\\2",x)
  shell(paste0('mklink "',
            paste0(allidat,"/",filename),'" "',
            x,'" '))
})
## ####################################################

filenameToBarcode  <-  function(file_names, legacy = TRUE)
{
  info  <-  files(legacy = legacy) %>%
    filter( ~ file_name %in% file_names) %>%
    select(c("cases.samples.portions.analytes.aliquots.submitter_id", "file_name")) %>%
    results_all()
  filenames <- as.character(unlist(info$file_name))
  cases <- sapply(info$cases,function(x) paste(unlist(x),collapse=","))
  df <- data.frame(filenames=filenames,
                cases=cases)
  rownames(df) <- df[,1]
  df
}


####################################################
# POINT 3: PRE-PROCESS THE DATA
## ####################################################
## This is where we load and preprocess the data
## ####################################################
## COMMENT: gc() are explicit calls to garbage collection
## to release memory that is not used anymore
```

Hoang Hai Nam
BINF-F410

```
## ####################################################
## All chromosome names
ALLCHR = paste0("chr",c(1:22,"X"))[1:22] ## only take autosomes
## ####################################################
## Set input directory and read in the files - where your idat files
## are (they should all be in the same folder)
inputdir <- paste0("allidat")
system.time(rgSet <- read.metharray.exp(inputdir));
gc();
## YOU SHOULD DEFINE THE IDS OF YOUR NORMAL SAMPLES - try and find out
## which is which using TCGA annotation packages

# grab all idat files and translate them into barcodes
# barcodes annotations:
# [TCGA]-[int:TSS]-[int:participant]-[int:sample+chr:vial]-[int:portion+chr:analyte]-[int:plate]-[int:center]
# portion+analyte is what we're interested in: e.g: 01A
barcodes <- filenameToBarcode(dir(inputdir, pattern="idat$"), legacy = TRUE)
# removing duplicates, 1 case presents 2 idat files,
# removing duplicated entries with the same barcodes
barcodes <- barcodes[!duplicated(barcodes[,"cases"]),]
# removes the last 2 sections of the rownames(idat names, removed e.g: _Grn.idat)
# regex grabbing [*]_[*] pattern, and replacing the entire pattern with the first element of the pattern
rownames(barcodes)=gsub("(.*)_(.*)","\\1",rownames(barcodes))
# flagging normal samples, regex grabbing the (entire) barcode patterns, replacing entire pattern with the 4th
element
# effectively only keeping the 4th element(sample + vial), checking if the sample int starts with 1
# normal types sample int ranges from 10-19:
https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/
isNormal <- substr(gsub("(.*)-(.*)-(.*)-(.*)-(.*)-(.*)-(.*)","\\4",barcodes[,2]),1,1)=="1"
names(isNormal)=rownames(barcodes)
## ####################################################
## Get raw signals for all samples and the panel of normals
data <- preprocessRaw(rgSet); rm("rgSet"); gc();
## ####################################################


## Load precomputed bins
## You have to download the file from https://github.com/mwsill/mnp_training
## And place it in your working directory
load("CNanalysis4_conumee_ANNO.vh20150715.RData")
## ####################################################
## Extract total intensity and get annotations
totalintensity <- getMeth(data)+getUnmeth(data)+1; gc();
annot <- as.data.frame(getAnnotation(data));  gc();
annot <- annot[annot[,"chr"]%in%ALLCHR,]
## ####################################################
## order chromosome, starts and ends of probes
annot <- annot[order(annot[,"chr"],annot[,"pos"],decreasing=F),]
starts <- as.numeric(as.character(annot[,"pos"]))
ends <- as.numeric(as.character(annot[,"pos"]))
chrs <- as.character(annot[,"chr"])
## ####################################################
## Final ordered total intensities for all and normal
totalintensity <- totalintensity[rownames(annot),]
isNormal <- isNormal[colnames(totalintensity)]
# intensities of normal samples
totalintensityNormal <- totalintensity[,isNormal]; gc();
```

```
# intensities of tumor samples
#totalintensityTumor <- totalintensity[,!isNormal]; gc();
## ##################################################

#####################
## Correct for PoN ##
#####################

## ##################################################
## log PoN-fitted intensity values of all probes for all samples
logr <- log2(sapply(1:ncol(totalintensity),function(x)
{
  notalreadyinpanel <- !colnames(totalintensityNormal)%in%colnames(totalintensity)[x]
  predicted <- lm(y~.-1,
                data=data.frame(y=totalintensity[,x],
                                X=totalintensityNormal[,notalreadyinpanel]))$fitted.values
  predicted[predicted<1] <- 1
  totalintensity[,x]/predicted
}))
colnames(logr) <- colnames(totalintensity); gc();

## ##################################################
###############################
## Segmenting the logr track ##
###############################
## ##################################################
## Here we derive the logR tracks for copy-number calling
## Across all samples
## ##################################################
track <- list()
for(samp in colnames(logr))
{
  cat(".")
  ## ##################################################
  .logr <- logr[,samp]
  ## ##################################################
  input <- meth_bin(.logr,
                 starts=starts,
                 ends=ends,
                 chrs=chrs,
                 anno@bins)
  track[[samp]] <- getTrackForAll.bins(input[[1]],
                               input[[4]],
                               input[[2]],
                               input[[3]],
                               allchr=gsub("chr","",ALLCHR))
  ## ##################################################
}
## ##################################################


############################################
## Get copy-number profiles with ASCAT.sc ##
############################################

## ##################################################
GAMMA <- .55
```

Hoang Hai Nam
BINF-F410

```
## ####################################################
# defining main dataframe
allsols <- as.data.frame(colnames(logr))
names(allsols)[1] <- "sampnames"
allsols$normal <- isNormal
purities<-c()
ploidies<-c()
sols <-c()
for (samp in 1:length(allsols$sampnames)){
  solution <- searchGrid(track[[samp]],
                      purs=seq(0.1, 1, 0.01),
                      ploidies=seq(1.5, 6, 0.01),
                      maxTumourPhi=8,
                      gamma=GAMMA)
  purities <- c(purities, solution$purity)
  ploidies <- c(ploidies, solution$ploidy)
}
# saving purity and ploidy of samples to main dataframe
allsols$purity<-purities
allsols$ploidy<-ploidies


## ####################################################
samp <- 69
# toy example
## ####################################################
solution <- searchGrid(track[[samp]],
                    purs=seq(0.1, 1, 0.01),
                    ploidies=seq(1.5, 6, 0.01),
                    maxTumourPhi=8,
                    gamma=GAMMA)
#> Solution found.
str(solution)
#>List of 4
#>$ errs     : num [1:91, 1:451] 0.01094 0.02326 0.01466 0.00966 0.02411 ...
#>..- attr(*, "dimnames")=List of 2
#>.. ..$ : chr [1:91] "0.1" "0.11" "0.12" "0.13" ...
#>.. ..$ : chr [1:451] "1.5" "1.51" "1.52" "1.53" ...

##########################
##>$ purity   : num 1   ##
##>$ ploidy   : num 2.01##
##########################


#>$ ambiguous: logi FALSE
#>plotSunrise(solution)
## ####################################################
## Plotting Sunrise ##
## ####################################################
plotSunrise(solution)

## Plotting the solution
# Toy example
plotSolution(track[[samp]],
          purity=solution$purity,
          ploidy=solution$ploidy,lwdSeg=2,ylim=c(0,15),
            gamma=GAMMA)
## ###################################################
#########
## Getting the corresponding CN profile using the purity and ploidy obtained in solution
# Toy example
profile <- getProfile(fitProfile(track[[samp]],
                       purity=solution$purity,
                       ploidy=solution$ploidy,
                       gamma=GAMMA))
######################
## GENEs OF INTEREST ##
######################
# not the neatest codes i've ever made and i apologise..
# defining genes of interest's genomic ranges according to the Cancer Genome Atlas
# CONTROL gene chosen as KAT2B, a randomly picked gene.
# getting annotation of probes that is within the gene KAT2B
annot[which(annot$UCSC_RefGene_Name=="KAT2B"),]
BIRC6 = GRanges(2,IRanges(32357028,32618899))
GPC5 = GRanges(13,IRanges(91398607,92867237))
MALAT1 = GRanges(11,IRanges(65497762,65506469))
CTRL = GRanges(3, IRanges(20081161,20195067))
# creating empty vectors for the copy numbers

b6cn<- c()
g5cn<- c()
m1cn<- c()
ctcn<- c()

#loops through 200 samples
for (samp in 1:length(allsols$sampnames)){
  #gets the profile of current sample
  prof <- getProfile(fitProfile(track[[samp]],
                       purity=allsols$purity[samp],
                       ploidy=allsols$ploidy[samp],
                       gamma=GAMMA))
  # defining Granges of the whole sample as being the entire chromosome
  gran <- GRanges(prof[,"chromosome"],IRanges(prof[,"start"],prof[,"end"]))
  # getting the overlap objects
  ovlb6 <- findOverlaps(gran, BIRC6)
  ovlg5 <- findOverlaps(gran, GPC5)
  ovlm1 <- findOverlaps(gran, MALAT1)
  ovlct <- findOverlaps(gran, CTRL)
  # saving copy number of overlap objects
  b6cn<- c(b6cn,sum(prof[queryHits(ovlb6), "total_copy_number"]))
  g5cn<- c(g5cn,sum(prof[queryHits(ovlg5), "total_copy_number"]))
  m1cn<- c(m1cn,sum(prof[queryHits(ovlm1), "total_copy_number"]))
  ctcn<- c(ctcn,sum(prof[queryHits(ovlct), "total_copy_number"]))
}
 # saving copy number solutions to main dataframe
allsols$BIRC6_cn <- b6cn
allsols$GPC5_cn <- g5cn
allsols$MALAT1_cn<- m1cn
allsols$CTRL_cn<- ctcn
#################################################################################

write.csv(allsols, file = "allsols.csv")
allsolsnormal = allsols[allsols$normal==TRUE,]
```

Hoang Hai Nam
BINF-F410

```
allsolstumor = allsols[allsols$normal==FALSE,]

# overview confirming the expected copy number of
# the normal samples
summary(allsolsnormal$BIRC6_cn)
summary(allsolsnormal$GPC5_cn)
summary(allsolsnormal$MALAT1_cn)
summary(allsolsnormal$CTRL_cn)
# overview of the copy number of the tumor samples
summary(allsolstumor$BIRC6_cn)
summary(allsolstumor$GPC5_cn)
summary(allsolstumor$MALAT1_cn)
summary(allsolstumor$CTRL_cn)
# counting different CN, amplified, deletions and no change
# BIRC6_cn
count(allsolstumor$BIRC6_cn>2) /186*100
count(allsolstumor$BIRC6_cn<2) /186*100
count(allsolstumor$BIRC6_cn==2) /186*100
mean(allsolstumor$BIRC6_cn)
# GPC5_cn
count(allsolstumor$GPC5_cn>2)/186*100
count(allsolstumor$GPC5_cn<2)/186*100
count(allsolstumor$GPC5_cn==2)/186*100
mean(allsolstumor$GPC5_cn)
# MALAT1_cn
count(allsolstumor$MALAT1_cn>2)/186*100
count(allsolstumor$MALAT1_cn<2)/186*100
count(allsolstumor$MALAT1_cn==2)/186*100
mean(allsolstumor$MALAT1_cn)
# KAT2B
count(allsolstumor$CTRL_cn>2)/186*100
count(allsolstumor$CTRL_cn<2)/186*100
count(allsolstumor$CTRL_cn==2)/186*100
mean(allsolstumor$CTRL_cn)

# testing pearson correlations
cor(allsolstumor$purity, allsolstumor$ploidy, method = "pearson")
cor(allsolstumor$BIRC6_cn, allsolstumor$purity, method = "pearson")
cor(allsolstumor$GPC5_cn, allsolstumor$purity, method = "pearson")
cor(allsolstumor$MALAT1_cn, allsolstumor$purity, method = "pearson")
cor(allsolstumor$CTRL_cn, allsolstumor$purity, method = "pearson")
cor(allsolstumor$BIRC6_cn, allsolstumor$GPC5_cn, method = "pearson")

##################################################################
# finds the most differentiated CPGs
dumper = dmpFinder(totalintensity, pheno = isNormal, type = "categorical", qCutoff = 10^-5,
            shrinkVar = FALSE)
dmps = c()
# getting annotations of probes
for (dmpr in row.names(dumper)) {
  dmps = c(dmps, which(annot$Name==dmpr))
}
dmpsdf = data.frame(annot[dmps,])
# reducing annotations categories for ease of viewing
minidmps<-dmpsdf[,c(1,2,4,24)]
write.csv(minidmps, file = "minidmps.csv")
##################################
```

Hoang Hai Nam
BINF-F410

```
## Part 8 Methylation Analysis ##
##################################
##################################
# Same operation done here as with totalintensity
# But for beta values
totalbeta<- getBeta(data)
# reordering totalbeta to be consistent with annot
totalbeta<- totalbeta[rownames(annot),]
totalbetaNormal <- totalbeta[,isNormal];gc()
##################################
# getting the mean beta vals of normal samples
# this is to be m_N, values of the mean beta value( of normal samples)
# at each probe
meanest <- as.data.frame(rowMeans(totalbetaNormal))
meantumor <- as.data.frame(rowMeans(totalbeta[,isNormal]))
names(meanest)<- "isNormalmean"
names(meantumor)<- "Tumormean"
#################################################################
######### CHANGE THE GENE OF INTEREST HERE ##################
# indexes of probes of interests, here set to get probes that
# are marked as being within the gene BIRC6
POIs= which(annot$UCSC_RefGene_Name=="GPC5")
# get probes of interest of specific probe name
# POIs = which(annot$Name=="cg12922032")

#setting up a table of differences
deltabetas = data.frame(totalbeta[POIs,])
colnames(deltabetas)<-colnames(totalbeta)
for (sampno in 1:length(totalbeta[1,])) {
  p = 1
  if (allsols$normal[sampno] == FALSE){
    # check to make sure we're modifying beta values of a TUMOR sample
    # get copy-number profile of the sample
    samppf <- getProfile(fitProfile(track[[sampno]],
                              purity=allsols$purity[sampno],
                              ploidy=allsols$ploidy[sampno],
                              gamma=GAMMA))
    # sample's grange object
    sampGR <- GRanges(samppf[,"chromosome"],IRanges(samppf[,"start"],samppf[,"end"]))
    # purity of sample
    pur<- allsols$purity[sampno]
    # loops through all probes within sample
    for (prbno in POIs){
      # probe's Grange object
      prGR <- GRanges(as.numeric(gsub("chr","",annot[row.names(totalbeta)[prbno],]["chr"])),
               as.numeric(annot[row.names(totalbeta)[prbno],]["pos"]))
      # get the overlap, access the local CN of the position of the concerned probe
      ovl <- findOverlaps(sampGR, prGR)
      # sometimes copy number profiles wont contain a certain probe's position
      # chose to ignore those cases should it happen, will only dfchange betaval
      # if overlap isnt empty
      if (length(ovl)>0){
        # local logr copynumber of tumor[probe]
        locCNT <- as.numeric(samppf[queryHits(ovl), "total_copy_number_logr"][1])
        # applying deconvolution with deconvolution formula
        totalbeta[prbno,sampno] <- ((totalbeta[prbno,sampno]*(pur*locCNT+2*(1-pur))-2*meanest[prbno,]*(1-
pur))/
```

```
          (pur*locCNT))
        # getting difference levels
        deltabetas[p,sampno] <- totalbeta[prbno,sampno] - deltabetas[p,sampno]
      }
      else {
        deltabetas[p,sampno] = 0
      }
      p=p+1
    }
  }
}
deltabetas = deltabetas[,not(isNormal)]
# this was supposed to put the deconvolution step in a more numerical
# analysis however i could not figure out how to do it in a logical way
# due to a lack of methods to test the accuracy of the deconvolution step
##################################################################
#Plot of a single cPG
plotCpg(totalbeta, cpg = "cg20300343", pheno = isNormal, type = "categorical",
        measure = "beta", ylab = "methylation (beta)", ylim = c(0, 1), xlab = "IsNormal")
# cpg within BIRC6
plotCpg(totalbeta, cpg = "cg01093329", pheno = isNormal, type = "categorical",
        measure = "beta", ylab = "methylation (beta)", ylim = c(0, 0.6), xlab = "IsNormal")
# cpg within GPC5
plotCpg(totalbeta, cpg = "cg13877944", pheno = isNormal, type = "categorical",
        measure = "beta", ylab = "methylation (beta)", ylim = c(0, 1), xlab = "IsNormal")
# plot all cpgs within gene of interest
for (pno in POIs){
  plotCpg(totalbeta, cpg = annot$Name[pno], pheno = isNormal, type = "categorical",
          measure = "beta", ylab = "methylation (beta)", ylim = c(0, 1), xlab = "IsNormal")
}
```