Methylation Analysis of Type2 Diabetics in Mature Muscle Cells

Gayatri Samant

2025-11-28

Introduction

This project analyzes DNA methylation data from human muscle cells to compare **Type 2 Diabetes (T2D)** samples with **Normal Glucose Tolerance (NGT)** samples. DNA methylation is an epigenetic mark that can help us understand how two groups differ at the molecular level.

I have used the publicly available dataset **GSE166787**, which was generated using the **Illumina HumanMethylation450K array**. This array measures methylation levels at around **485,000 CpG sites** across the genome.

The full dataset contains:

- **14 T2D donors × 2 cell types (myoblasts + myotubes)** → 28 samples
- **14 NGT donors × 2 cell types** → 28 samples
- **Total:** 56 samples

For this analysis, we focus only on:

- **10 NGT Myotubes**
- **10 T2D Myotubes**

Myotubes were selected because they represent the mature muscle cell stage and show clearer methylation differences.

This documentation describes a complete methylation-analysis pipeline that includes:

- Loading and preparing IDAT files
- Creating the SampleSheet automatically
- Performing quality control

- Normalizing the data
- Filtering unwanted probes
- Running PCA to check sample patterns
- Identifying differentially methylated CpGs
- Finding methylation differences (ΔBeta values)
- Detecting region-level methylation changes (DMRs)
- Performing functional enrichment using GREAT

The workflow uses R and Bioconductor packages such as **minfi**, **limma**, **DMRcate**, and **rGREAT**.

Script and the outputs:

```
R.version.string

## [1] "R version 4.3.3 (2024-02-29 ucrt)"

BiocManager::version()

## [1] '3.17'
```

This R script will Unzip .idat.gz Files and automatically create SampleSheet.csv for our 20 samples (10 NGT + 10 T2D)

```
# -----------------------
# Create SampleSheet.csv for GSE166787
# -----------------------

library(dplyr)

library(stringr)

# Path to your raw IDAT files
idat_dir <- "E:/Diabetic_Data/GSE166787_RAW"


gz_files <- list.files(idat_dir, pattern = "\\.idat\\.gz$", full.names =
TRUE)
```

```r
if (length(gz_files) > 0) {
  message("Unzipping IDAT files...")
  sapply(gz_files, function(f) {
    out_f <- sub("\\.gz$", "", f)
    if (!file.exists(out_f)) {
      gunzip(f, remove = FALSE)
    }
  })
}


# List all IDAT files (already unzipped)
files <- list.files(idat_dir, pattern = "idat$", full.names = FALSE)

# Extract basenames (remove _Grn.idat or _Red.idat)
basename <- str_replace(files, "_(Grn|Red)\\.idat$", "")
basename <- unique(basename)

# Extract GSM ID (before first "_")
Sample_Name <- str_extract(basename, "^GSM\\d+")

# Extract last 3 digits to classify samples
Ending <- str_sub(Sample_Name, -3)

# Define sample groups
NGT_ids <- c("486","488","490","492","494","496","498","500","502","504")
T2D_ids <- c("514","516","518","520","522","524","526","528","530","532")

Group <- ifelse(Ending %in% NGT_ids, "NGT",
         ifelse(Ending %in% T2D_ids, "T2D", NA))

# Build final SampleSheet (only valid 20 samples)
df <- data.frame(
  Sample_Name = Sample_Name,
  Group = Group,
  Basename = basename,
  stringsAsFactors = FALSE
) %>%
  filter(!is.na(Group)) %>%
  arrange(Group)

# Save SampleSheet
write.csv(df, file = file.path(idat_dir, "SampleSheet.csv"), row.names =
FALSE)

print("SampleSheet.csv created successfully!")
```

```
## [1] "SampleSheet.csv created successfully!"
```

```r
print(df)
```

```
##    Sample_Name Group                       Basename
## 1   GSM5082486   NGT GSM5082486_9020331173_R02C02
## 2   GSM5082488   NGT GSM5082488_9020331170_R04C02
## 3   GSM5082490   NGT GSM5082490_9020331033_R04C01
## 4   GSM5082492   NGT GSM5082492_9020331080_R02C01
## 5   GSM5082494   NGT GSM5082494_9020331011_R06C01
## 6   GSM5082496   NGT GSM5082496_8963266097_R02C02
## 7   GSM5082498   NGT GSM5082498_9020331170_R02C01
## 8   GSM5082500   NGT GSM5082500_8963266081_R04C02
## 9   GSM5082502   NGT GSM5082502_9020331168_R04C01
## 10  GSM5082504   NGT GSM5082504_9020331166_R06C01
## 11  GSM5082514   T2D GSM5082514_9020331166_R02C01
## 12  GSM5082516   T2D GSM5082516_9020331080_R06C01
## 13  GSM5082518   T2D GSM5082518_9020331173_R04C01
## 14  GSM5082520   T2D GSM5082520_9020331143_R04C01
## 15  GSM5082522   T2D GSM5082522_9020331033_R02C02
## 16  GSM5082524   T2D GSM5082524_9020331170_R06C01
## 17  GSM5082526   T2D GSM5082526_9020331011_R04C02
## 18  GSM5082528   T2D GSM5082528_9020331168_R02C02
## 19  GSM5082530   T2D GSM5082530_9020331168_R04C02
## 20  GSM5082532   T2D GSM5082532_9020331143_R06C02
```

==========

1. Install Required Packages

==========

```r
# # If missing — install BiocManager
# # install.packages("BiocManager")
# # BiocManager::install(version = "3.17", ask = FALSE)
#
# # Bioconductor core packages
# # BiocManager::install(c(
# #   "minfi",
# #   "limma",
# #   "GenomicRanges",
# #   "GenomicFeatures",
# #   "rGREAT",
# #   "DMRcate"
# # ))
#
# # CRAN packages
# # install.packages(c("tidyverse", "R.utils"))
#
# # 450k annotation & manifest (manual install due to older versions)
# if (!requireNamespace("remotes", quietly = TRUE))
install.packages("remotes")
#
```

```
# remotes::install_url(
#
"https://bioconductor.org/packages/release/data/annotation/src/contrib/IlluminaHumanMethylation450kmanifest_0.4.0.tar.gz"
# )
#
# remotes::install_url(
#
"https://bioconductor.org/packages/release/data/annotation/src/contrib/IlluminaHumanMethylation450kanno.ilmn12.hg19_0.6.0.tar.gz"
# )
#
# BiocManager::install("IlluminaHumanMethylation450kanno.ilmn12.hg19")
#
# # DMRcate dependency
# BiocManager::install("IlluminaHumanMethylationEPICanno.ilm10b4.hg19")
#
# # rGREAT dependencies
# BiocManager::install(c(
#   "TxDb.Hsapiens.UCSC.hg19.knownGene",
#   "TxDb.Hsapiens.UCSC.hg38.knownGene"
# ))
```

==========

2. Load All Required Libraries

==========

```
library(minfi)
library(limma)
library(GenomicRanges)
library(GenomicFeatures)
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(DMRcate)
library(rGREAT)
library(tidyverse)
library(R.utils)
```

==========

3. Set Project Paths

==========

```
base_dir <- "E:/Diabetic_Data"
idat_dir <- file.path(base_dir, "GSE166787_RAW")
sample_sheet_path <- file.path(idat_dir, "SampleSheet.csv")

list.files(idat_dir)[1:20]
```

```
##  [1] "GSM5082486_9020331173_R02C02_Grn.idat"
##  [2] "GSM5082486_9020331173_R02C02_Red.idat"
##  [3] "GSM5082488_9020331170_R04C02_Grn.idat"
##  [4] "GSM5082488_9020331170_R04C02_Red.idat"
##  [5] "GSM5082490_9020331033_R04C01_Grn.idat"
##  [6] "GSM5082490_9020331033_R04C01_Red.idat"
##  [7] "GSM5082492_9020331080_R02C01_Grn.idat"
##  [8] "GSM5082492_9020331080_R02C01_Red.idat"
##  [9] "GSM5082494_9020331011_R06C01_Grn.idat"
## [10] "GSM5082494_9020331011_R06C01_Red.idat"
## [11] "GSM5082496_8963266097_R02C02_Grn.idat"
## [12] "GSM5082496_8963266097_R02C02_Red.idat"
## [13] "GSM5082498_9020331170_R02C01_Grn.idat"
## [14] "GSM5082498_9020331170_R02C01_Red.idat"
## [15] "GSM5082500_8963266081_R04C02_Grn.idat"
## [16] "GSM5082500_8963266081_R04C02_Red.idat"
## [17] "GSM5082502_9020331168_R04C01_Grn.idat"
## [18] "GSM5082502_9020331168_R04C01_Red.idat"
## [19] "GSM5082504_9020331166_R06C01_Grn.idat"
## [20] "GSM5082504_9020331166_R06C01_Red.idat"

# If it shows 40 files (20 IDAT pairs), then the command ran currectly.
```

==========

5. Load Sample Sheet

==========

```
sample_sheet <- read.csv(sample_sheet_path, stringsAsFactors = FALSE)
head(sample_sheet)

##   Sample_Name Group                      Basename
## 1  GSM5082486   NGT GSM5082486_9020331173_R02C02
## 2  GSM5082488   NGT GSM5082488_9020331170_R04C02
## 3  GSM5082490   NGT GSM5082490_9020331033_R04C01
## 4  GSM5082492   NGT GSM5082492_9020331080_R02C01
## 5  GSM5082494   NGT GSM5082494_9020331011_R06C01
## 6  GSM5082496   NGT GSM5082496_8963266097_R02C02
```

==========

6. Load Raw IDAT Files

==========

Expected: ~622,000 probes × 20 samples.

```
rgSet <- read.metharray.exp(
  targets = sample_sheet,
```

```
  extended = TRUE,
  base = idat_dir
)

rgSet
```

```
## class: RGChannelSetExtended
## dim: 622399 20
## metadata(0):
## assays(5): Green Red GreenSD RedSD NBeads
## rownames(622399): 10600313 10600322 ... 74810490 74810492
## rowData names(0):
## colnames(20): GSM5082486_9020331173_R02C02 GSM5082488_9020331170_R04C02
##   ... GSM5082530_9020331168_R04C02 GSM5082532_9020331143_R06C02
## colData names(4): Sample_Name Group Basename filenames
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
```

==========

7. Detection P-Values (QC)

==========

```
detP <- detectionP(rgSet)
```

```
## Loading required package: IlluminaHumanMethylation450kmanifest
```

```
summary(as.vector(detP))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000000 0.000000 0.001093 0.000000 1.000000
```

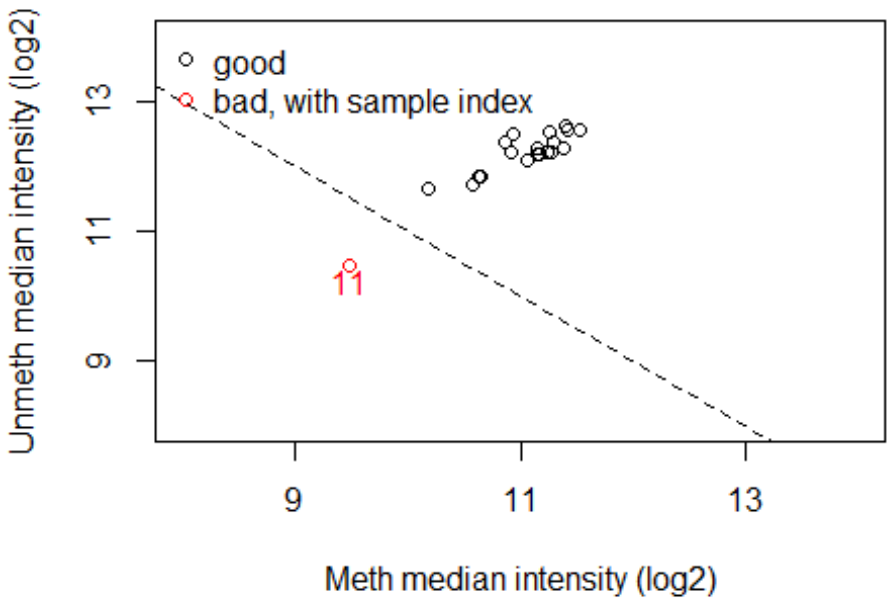
```
# Remove poor-quality samples
failed_samples <- colMeans(detP > 0.01) > 0.05
sample_sheet <- sample_sheet[!failed_samples, ]
rgSet <- rgSet[, !failed_samples]

# Remove poor-quality probes
failed_probes <- rowMeans(detP > 0.01) > 0.05
rgSet <- rgSet[!failed_probes, ]
```

==========

8. Normalization (NOOB)

==========

```r
mSet.noob <- preprocessNoob(rgSet)

qc <- getQC(mSet.noob)
plotQC(qc)
```



==========

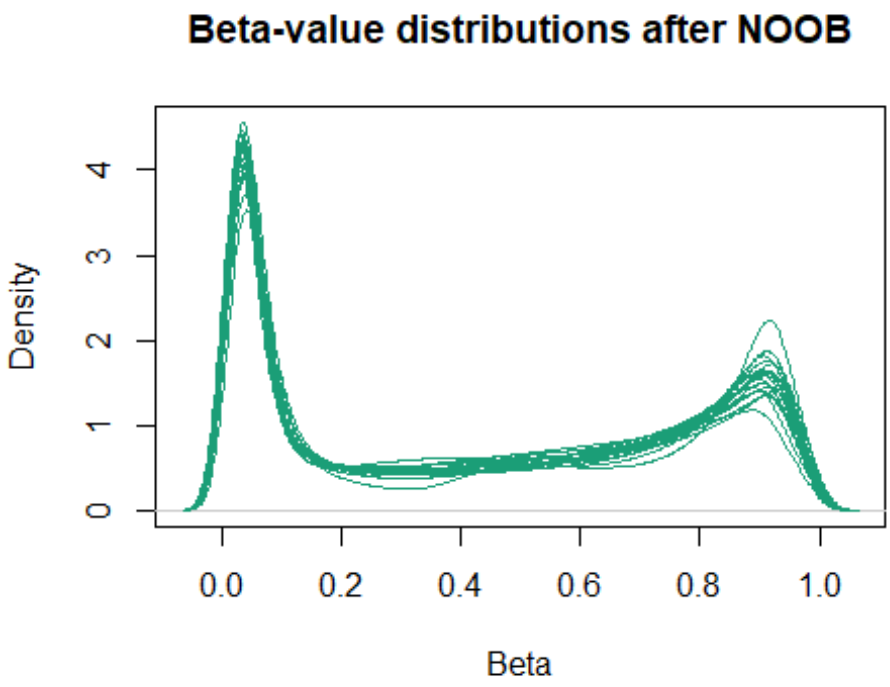
9. Extract Beta & M-values

==========

```r
# Methylation metrix
beta <- getBeta(mSet.noob)
mval <- getM(mSet.noob)
dim(beta)
```

```
## [1] 476241     20
```

```r
densityPlot(beta, main = "Beta-value distributions after NOOB", legend =
FALSE)
```



==========

10. Probe Filtering

==========

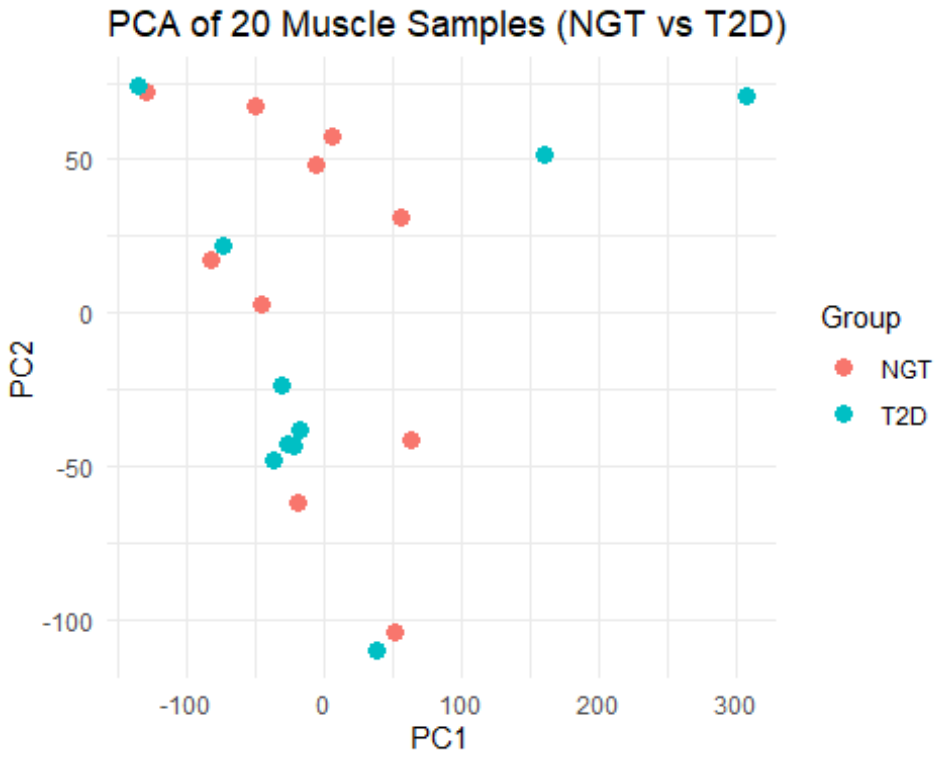
```r
# A) Map to Genome
grSet <- mapToGenome(mSet.noob)

# B) Remove SNP-overlapping CpGs
mSet.clean <- dropLociWithSnps(grSet)
beta <- getBeta(mSet.clean)
mval <- getM(mSet.clean)

# C) Remove sex-chromosome CpGs (muscle cells → safe)
anno <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)
anno <- anno[rownames(beta), ]

sex_probes <- anno$chr %in% c("chrX", "chrY")
beta <- beta[!sex_probes, ]
mval <- mval[!sex_probes, ]
```

==========

11. PCA (Unsupervised QC)

==========

```r
topVar <- order(rowVars(mval), decreasing = TRUE)[1:5000]
pca <- prcomp(t(mval[topVar, ]))

pca_df <- data.frame(
  PC1 = pca$x[,1],
  PC2 = pca$x[,2],
  Group = sample_sheet$Group
)

ggplot(pca_df, aes(PC1, PC2, color = Group)) +
  geom_point(size = 3) +
  theme_minimal() +
  labs(title = "PCA of 20 Muscle Samples (NGT vs T2D)")
```



==========

12. Design Matrix (NGT = Control)

==========

This models: T2D effect = T2D – NGT

```r
sample_sheet$Group <- factor(sample_sheet$Group, levels = c("NGT", "T2D"))
design <- model.matrix(~ Group, data = sample_sheet)
design
```

```
##    (Intercept) GroupT2D
## 1            1        0
## 2            1        0
## 3            1        0
## 4            1        0
## 5            1        0
## 6            1        0
## 7            1        0
## 8            1        0
## 9            1        0
## 10           1        0
## 11           1        1
## 12           1        1
## 13           1        1
## 14           1        1
## 15           1        1
## 16           1        1
## 17           1        1
## 18           1        1
## 19           1        1
## 20           1        1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$Group
## [1] "contr.treatment"
```

==========

13. Differential Methylation (limma)

==========

In muscle-derived methylation datasets, T2D effects are subtle, so PCA may not show separation, but limma will still detect probe-level statistical differences.

```r
# Fit linear model and compute moderated statistics
fit <- lmFit(mval, design)
fit <- eBayes(fit)

# Extract ALL CpGs ranked by significance
topCpGs <- topTable(fit, coef = "GroupT2D", number = Inf)

# Preview table
head(topCpGs)
```

```
##                logFC     AveExpr         t      P.Value adj.P.Val
B
## cg20010506 -0.9054849 -3.31492434 -5.470675 2.115821e-05 0.9767778 -
```

```
3.410626
## cg20707907  2.0067896  0.04702131  5.205225 3.906014e-05 0.9767778 -
3.466550
## cg26102564  1.8738699  1.87982254  5.123209 4.726661e-05 0.9767778 -
3.484517
## cg26516532 -2.4718534 -3.11952963 -4.985488 6.518655e-05 0.9767778 -
3.515431
## cg05942574 -1.0583537 -2.70474309 -4.913038 7.724045e-05 0.9767778 -
3.532069
## cg16626196  0.9848310  0.69532042  4.838968 9.190545e-05 0.9767778 -
3.549347
```

==========

14 . Save CpG-Level Results

==========

Convert your topCpGs to a clean data.frame

```r
mean_T2D <- rowMeans(beta[, sample_sheet$Group == "T2D"])
mean_NGT <- rowMeans(beta[, sample_sheet$Group == "NGT"])

probe_delta <- mean_T2D - mean_NGT

# Convert limma Bioconductor object to a clean data.frame
topCpGs_df <- as.data.frame(topCpGs)

# Add CpG name
topCpGs_df$CpG <- rownames(topCpGs)

# Add gene annotation
topCpGs_df$gene <- anno[topCpGs_df$CpG, "UCSC_RefGene_Name"]

# Add deltaBeta for CpGs
topCpGs_df$deltaBeta <- probe_delta[match(topCpGs_df$CpG,
names(probe_delta))]

# Clean gene annotation
# The initial annotation may assign multiple genes, so we consider 1st gene
while plotting
topCpGs_df$gene_label <- ifelse(
  topCpGs_df$gene == "" | is.na(topCpGs_df$gene),
  NA,
  sapply(strsplit(topCpGs_df$gene, ";"), `[`, 1)  # take the first gene
)
```

Now topCpGs_df is your main table for CpG-level saving and plotting.

Save complete CpG table

```
out_dir <- "E:/Diabetic_Data/"

write.csv(topCpGs_df,
          file.path(out_dir, "DMP_all_CpGs.csv"),
          row.names = FALSE)
```

Save Top 10 Hyper / Hypo CpGs

```
# Top 10 hyper
top10_hyper <- topCpGs_df[order(-topCpGs_df$deltaBeta), ][1:10, ]
write.csv(top10_hyper, file.path(out_dir, "Top10_hyper_CpGs.csv"), row.names
= FALSE)

# Top 10 hypo
top10_hypo <- topCpGs_df[order(topCpGs_df$deltaBeta), ][1:10, ]
write.csv(top10_hypo, file.path(out_dir, "Top10_hypo_CpGs.csv"), row.names =
FALSE)
```

Volcano plot showing Hyper and Hypo methylated Genes

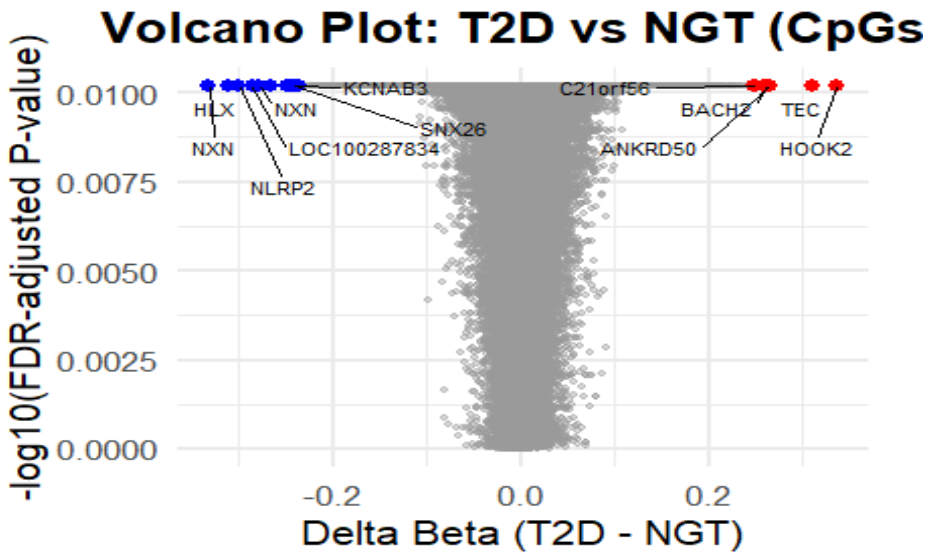
```
library(ggplot2)
library(ggrepel)

# Add neglogFDR to main table
volc_df <- topCpGs_df
volc_df$neglogFDR <- -log10(volc_df$adj.P.Val)

# Add neglogFDR to top lists
top10_hyper$neglogFDR <- -log10(top10_hyper$adj.P.Val)
top10_hypo$neglogFDR  <- -log10(top10_hypo$adj.P.Val)

gg_volcano <- ggplot(volc_df, aes(deltaBeta, neglogFDR)) +
  geom_point(alpha = 0.4, size = 0.8, color = "grey60") +

  geom_point(data = top10_hyper,
             aes(deltaBeta, neglogFDR),
             color = "red", size = 2.5) +

  geom_point(data = top10_hypo,
             aes(deltaBeta, neglogFDR),
             color = "blue", size = 2.5) +

  geom_text_repel(
    data = rbind(top10_hyper, top10_hypo),
    aes(deltaBeta, neglogFDR, label = gene_label),
    size = 3,
```

```
    box.padding = 0.4,
    max.overlaps = 100,
    segment.color = "black"
  ) +

  theme_minimal(base_size = 16) +
  labs(
    title = "Volcano Plot: T2D vs NGT (CpGs)",
    x = "Delta Beta (T2D - NGT)",
    y = "-log10(FDR-adjusted P-value)"
  ) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    plot.margin = margin(20, 20, 20, 20)
  )

# Save
ggsave(file.path(out_dir, "Volcano_T2D_vs_NGT.png"),
       plot = gg_volcano,
       width = 12, height = 8, dpi = 300)
```

```
## Warning: Removed 8 rows containing missing values or values outside the
scale range
## (`geom_text_repel()`).
```

```
print(gg_volcano)
```

```
## Warning: Removed 8 rows containing missing values or values outside the
scale range
## (`geom_text_repel()`).
```



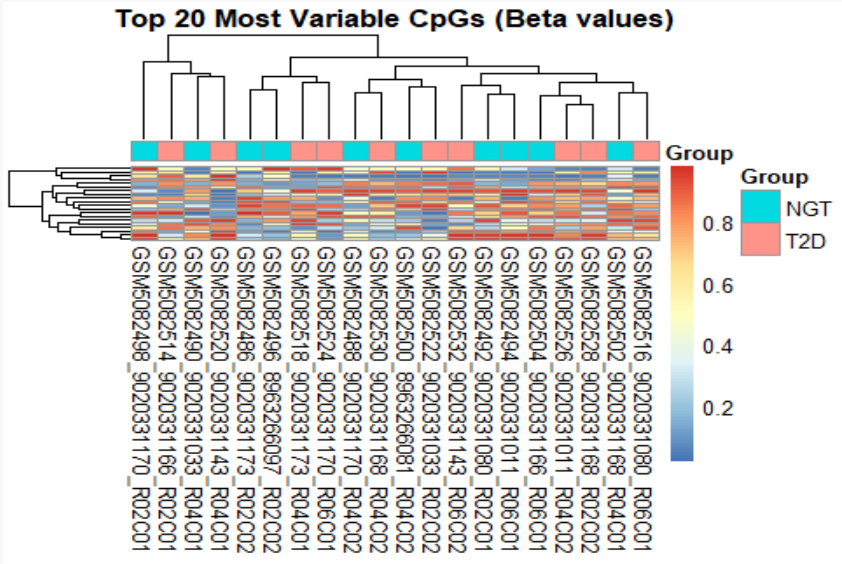
Grey cloud = all CpGs

Red = top 10 hyper (T2D hypermethylated)

Blue = top 10 hypo (T2D hypomethylated)

Heatmap of Top 20 CpGs

```r
# install.packages("pheatmap")

library(matrixStats)
library(pheatmap)

# Select top 20 variable CpGs
top20 <- order(rowVars(beta), decreasing = TRUE)[1:20]
beta_top20 <- beta[top20, ]

# Annotation
annotation_col <- data.frame(Group = sample_sheet$Group)
rownames(annotation_col) <- colnames(beta_top20)

# Plot heatmap
pheatmap(
  beta_top20,
  annotation_col = annotation_col,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  show_rownames = FALSE,
  main = "Top 20 Most Variable CpGs (Beta values)"
)
```



==========

14. DMRcate — Region-Based Differential Methylation
==========

This step matters because,

- In tissues like skeletal muscle:
- Individual CpG effects are small
- Few CpGs survive FDR at probe-level
- BUT CpGs cluster into regions of consistent hyper/hypomethylation

So DMRcate detects regional methylation shifts that are more biologically meaningful.

14.1 Attempt probe-level annotation (will likely return warning)

⚠ EXPECTED WARNING: Your contrast returned no individually significant probes.

```
myAnnotation <- cpg.annotate(
  datatype = "array",
  what = "M",
  object = mval,
  analysis.type = "differential",
  design = design,
  coef = "GroupT2D"
)

## Loading required package: IlluminaHumanMethylationEPICanno.ilm10b4.hg19

##
## Attaching package: 'IlluminaHumanMethylationEPICanno.ilm10b4.hg19'

## The following objects are masked from
'package:IlluminaHumanMethylation450kanno.ilmn12.hg19':
##
##     Islands.UCSC, Locations, Manifest, Other, SNPs.132CommonSingle,
##     SNPs.135CommonSingle, SNPs.137CommonSingle, SNPs.138CommonSingle,
##     SNPs.141CommonSingle, SNPs.142CommonSingle, SNPs.144CommonSingle,
##     SNPs.146CommonSingle, SNPs.147CommonSingle, SNPs.Illumina

## Your contrast returned no individually significant probes. Try increasing
the fdr. Alternatively, set pcutoff manually in dmrcate() to return DMRs, but
be warned there is an increased risk of Type I errors.
```

This is normal for muscle T2D datasets → proceed with relaxed thresholds.

==========

15. DMRcate (Relaxed Thresholds to Detect Regions)
========== -

We set FDR = 1 in cpg.annotate → keep ALL CpGs

- We control significance at the region level using pcutoff

- This is standard in low-effect tissues

```
myAnnotation <- cpg.annotate(
  datatype = "array",
  what = "M",
  object = mval,
  analysis.type = "differential",
  design = design,
  coef = "GroupT2D",
  fdr = 1            # keep all probes (no FDR cut at probe level)
)

## Your contrast returned 417641 individually significant probes. We
recommend the default setting of pcutoff in dmrcate().

dmr_results <- dmrcate(
  myAnnotation,
  lambda = 1000,
  C = 2,
  pcutoff = 0.05     # region seeding threshold
)



## Demarcating regions...

## Done!

print(dmr_results)

## DMResults object with 34 DMRs.
## Use extractRanges() to produce a GRanges object of these.
```

==========

16. Extract and Save DMR Regions

==========

```
slotNames(dmr_results)
```

```
## [1] "coord"            "no.cpgs"          "min_smoothed_fdr" "Stouffer"
## [5] "HMFDR"            "Fisher"           "maxdiff"          "meandiff"

# All DMR information is stored across these slots.
```

Understanding what each slot contains

- coord –> Genomic coordinates in the form “chr:start-end”
- no.cpgs –> Number of CpGs in each DMR
- min_smoothed_fdr –> Minimum smoothed FDR across region
- Stouffer–> Stouffer region statistic
- HMFDR–> Region-level FDR
- Fisher–> Fisher combined p-value
- maxdiff–> Maximum Δβ within region
- meandiff –> Mean Δβ within region

So all we need to do is:

- Parse the “coord” slot into chromosome, start, end.
- Combine all slots into a table.
- Convert to GRanges.

```
# STEP 1 – Extract DMR table from slots
dmr_coord <- dmr_results@coord
dmr_no_cpgs <- dmr_results@no.cpgs
dmr_minFDR <- dmr_results@min_smoothed_fdr
dmr_Stouffer <- dmr_results@Stouffer
dmr_HMFDR <- dmr_results@HMFDR
dmr_Fisher <- dmr_results@Fisher
dmr_maxdiff <- dmr_results@maxdiff
dmr_meandiff <- dmr_results@meandiff


# STEP 2 – Parse the genomic coordinates

library(stringr)
library(dplyr)

coord_df <- data.frame(coord = dmr_coord, stringsAsFactors = FALSE) %>%
  tidyr::separate(coord, into = c("chr", "pos"), sep = ":", remove = TRUE)
%>%
  tidyr::separate(pos, into = c("start", "end"), sep = "-", remove = TRUE)
```

```r
coord_df$start <- as.numeric(coord_df$start)
coord_df$end   <- as.numeric(coord_df$end)

# STEP 3 — Combine into full DMR table
dmr_df <- data.frame(
  chr = coord_df$chr,
  start = coord_df$start,
  end = coord_df$end,
  no.cpgs = dmr_no_cpgs,
  min_smoothed_fdr = dmr_minFDR,
  Stouffer = dmr_Stouffer,
  HMFDR = dmr_HMFDR,
  Fisher = dmr_Fisher,
  maxdiff = dmr_maxdiff,
  meandiff = dmr_meandiff,
  stringsAsFactors = FALSE
)


# Check:

head(dmr_df)
```

```
##     chr     start       end no.cpgs min_smoothed_fdr  Stouffer     HMFDR
## 1  chr4   2964211   2964223       2     0.0256384174 0.9971843 0.9748631
## 2  chr5 132759700 132759792       2     0.0362257343 0.9971843 0.9748631
## 3  chr5 177753258 177753286       2     0.0065539108 0.9971843 0.9748631
## 4  chr7 102313389 102313412       2     0.0188087530 0.9971843 0.9748631
## 5 chr17    724273    724374       2     0.0001727313 0.9971843 0.9748631
## 6 chr18  30349900  30350221       2     0.0315710241 0.9971843 0.9748631
##      Fisher     maxdiff    meandiff
## 1 0.9987469 -0.08758346 -0.05323767
## 2 0.9987469 -0.05925439 -0.03341390
## 3 0.9987469 -0.20294361 -0.19329602
## 4 0.9987469  0.09087416  0.06469844
## 5 0.9987469 -0.33110462 -0.30424952
## 6 0.9987469  0.05897431  0.03989756
```

```r
# STEP 4 — Convert to GRanges

library(GenomicRanges)

dmr_ranges <- GRanges(
  seqnames = dmr_df$chr,
  ranges = IRanges(start = dmr_df$start, end = dmr_df$end),
  no.cpgs = dmr_df$no.cpgs,
  min_smoothed_fdr = dmr_df$min_smoothed_fdr,
  Stouffer = dmr_df$Stouffer,
```

```
  HMFDR = dmr_df$HMFDR,
  Fisher = dmr_df$Fisher,
  maxdiff = dmr_df$maxdiff,
  meandiff = dmr_df$meandiff
)

dmr_ranges
```

```
## GRanges object with 34 ranges and 7 metadata columns:
##        seqnames              ranges strand |   no.cpgs min_smoothed_fdr
##           <Rle>           <IRanges>  <Rle> | <integer>        <numeric>
##    [1]     chr4     2964211-2964223      * |         2      0.025638417
##    [2]     chr5 132759700-132759792      * |         2      0.036225734
##    [3]     chr5 177753258-177753286      * |         2      0.006553911
##    [4]     chr7 102313389-102313412      * |         2      0.018808753
##    [5]    chr17       724273-724374      * |         2      0.000172731
##     ...      ...                 ...    ... .       ...              ...
##   [30]     chr5 134366534-134367394      * |         6       0.00288639
##   [31]     chr1   35441734-35442839      * |         6       0.00257304
##   [32]     chr5 110062384-110062837      * |         6       0.01337047
##   [33]    chr17     7832479-7833237      * |         8       0.00217505
##   [34]    chr10   47083283-47083632      * |         8       0.00415977
##         Stouffer     HMFDR    Fisher    maxdiff   meandiff
##        <numeric> <numeric> <numeric>  <numeric>  <numeric>
##    [1]  0.997184  0.974863  0.998747 -0.0875835 -0.0532377
##    [2]  0.997184  0.974863  0.998747 -0.0592544 -0.0334139
##    [3]  0.997184  0.974863  0.998747 -0.2029436 -0.1932960
##    [4]  0.997184  0.974863  0.998747  0.0908742  0.0646984
##    [5]  0.997184  0.974863  0.998747 -0.3311046 -0.3042495
##     ...       ...       ...       ...        ...        ...
##   [30]  0.999999  0.974863         1 -0.1949652 -0.1152067
##   [31]  1.000000  0.976830         1  0.1335430  0.0672685
##   [32]  1.000000  0.977182         1 -0.1609797 -0.0889486
##   [33]  1.000000  0.976018         1 -0.2391475 -0.1219599
##   [34]  1.000000  0.976747         1 -0.0926176 -0.0494345
##   -------
##   seqinfo: 16 sequences from an unspecified genome; no seqlengths
```

```
# STEP 5 — Save
out_dir<-"E:/Diabetic_Data"
write.csv(as.data.frame(dmr_ranges),
          file.path(out_dir, "DMRs_final_manual_extraction.csv"),
          row.names = FALSE)
```

==========

17. Compute Mean ΔBeta for Each DMR

==========

We compute the average methylation difference inside the region. This gives effect size at the region level.

```
# Compute deltaBeta for each probe
mean_T2D <- rowMeans(beta[, sample_sheet$Group == "T2D"])
mean_NGT <- rowMeans(beta[, sample_sheet$Group == "NGT"])
probe_delta <- mean_T2D - mean_NGT

probe_anno <- anno[rownames(beta), c("chr", "pos")]
probe_gr <- GRanges(seqnames = probe_anno$chr,
                    ranges = IRanges(start = probe_anno$pos,
                                     end = probe_anno$pos),
                    CpG = rownames(beta),
                    deltaBeta = probe_delta)

# For each DMR: extract probes + compute mean deltaBeta
dmr_means <- sapply(seq_along(dmr_ranges), function(i) {
  ol <- findOverlaps(dmr_ranges[i], probe_gr)
  mean(mcols(probe_gr[subjectHits(ol)])$deltaBeta, na.rm = TRUE)
})

dmr_df <- as.data.frame(dmr_ranges)
dmr_df$mean_deltaBeta <- dmr_means

write.csv(dmr_df,
          file.path(out_dir, "DMRs_with_mean_deltaBeta.csv"),
          row.names = FALSE)
```

==========

18. Annotate DMRs with Gene Names

==========

```
# Use UCSC_RefGene_Name column
dmr_genes <- sapply(seq_along(dmr_ranges), function(i) {
  ol <- findOverlaps(dmr_ranges[i], probe_gr)
  probes <- names(probe_gr[subjectHits(ol)])
  genes <- unique(anno[probes, "UCSC_RefGene_Name"])
  genes <- genes[genes != ""]
  if (length(genes) == 0) return(NA)
  paste(unique(unlist(strsplit(genes, ";"))), collapse = ";")
})
```

```
dmr_df$genes <- dmr_genes

write.csv(dmr_df,
          file.path(out_dir, "DMRs_annotated.csv"),
          row.names = FALSE)
```

==========

20. GREAT Enrichment

==========

SECTION 1 — GREAT Submission

```
library(rGREAT)

# Submit DMRs to GREAT (hg19 is correct for 450k annotation)
job <- submitGreatJob(dmr_ranges, species = "hg19")

# Retrieve GO tables
go_tables <- getEnrichmentTables(job)

## The default enrichment table does not contain informatin of associated
## genes for each input region. You can set `download_by = 'tsv'` to
## download the complete table, but note only the top 500 regions can be
## retreived. See the following link:
##
## https://great-
help.atlassian.net/wiki/spaces/GREAT/pages/655401/Export#Export-GlobalExport
##
## Except the additional gene-region association column if taking 'tsv' as
## the source of result, all other columns are the same if you choose
## 'json' (the default) as the source. Or you can try the local GREAT
## analysis with the function `great()`.

names(go_tables)

## [1] "GO Molecular Function" "GO Biological Process" "GO Cellular
Component"
```

SECTION 2 — Extract Each GO Category

```
go_bp <- go_tables[["GO Biological Process"]]
go_mf <- go_tables[["GO Molecular Function"]]
go_cc <- go_tables[["GO Cellular Component"]]
```

```r
# Convert GO tables to regular data.frames (removes Rle)
go_bp <- as.data.frame(lapply(go_bp, as.vector))
go_mf <- as.data.frame(lapply(go_mf, as.vector))
go_cc <- as.data.frame(lapply(go_cc, as.vector))

# fix the rownames:
rownames(go_bp) <- NULL
rownames(go_mf) <- NULL
rownames(go_cc) <- NULL

# Check structure to confirm:
str(go_bp)
```

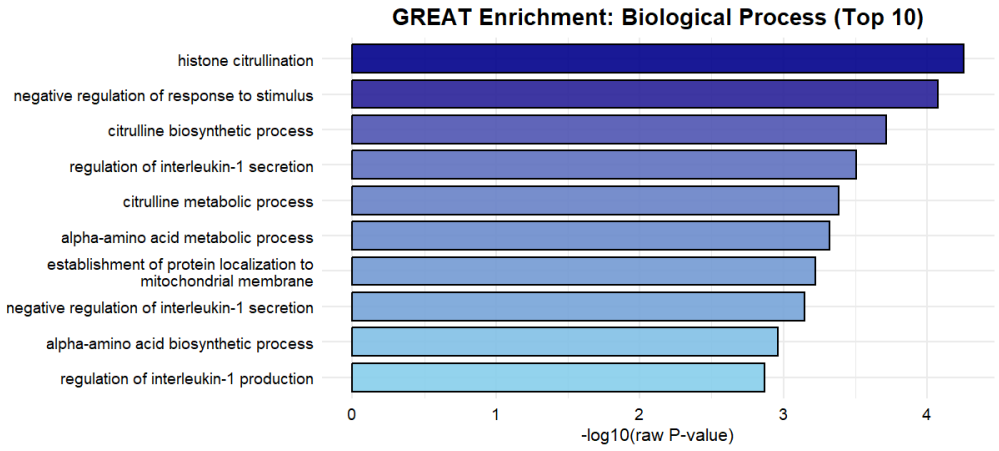
SECTION 3 — Save GO Tables

```r
out_dir <- "E:/Diabetic_Data/GREAT_results"
dir.create(out_dir, showWarnings = FALSE)

write.csv(go_bp, file.path(out_dir, "GREAT_GO_BP.csv"), row.names = FALSE)
write.csv(go_mf, file.path(out_dir, "GREAT_GO_MF.csv"), row.names = FALSE)
write.csv(go_cc, file.path(out_dir, "GREAT_GO_CC.csv"), row.names = FALSE)
```

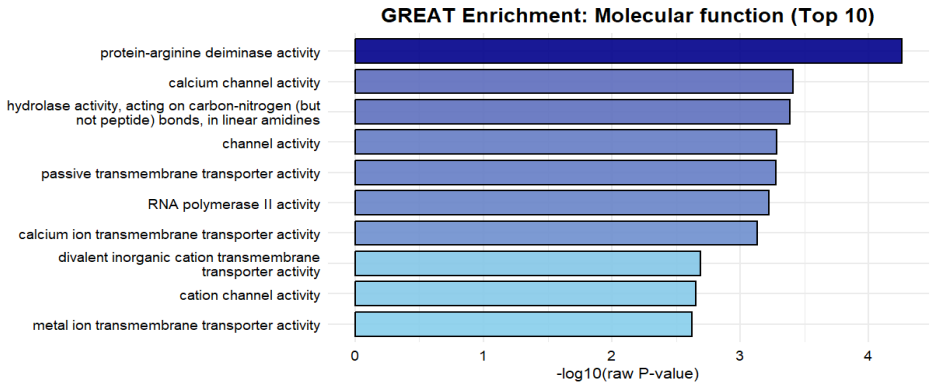
SECTION 4 — BEAUTIFUL GO PLOTS (Top 10 Terms)

```r
library(ggplot2)
library(stringr)

plot_great_go <- function(go_df, title_text) {

  # Sort by raw p-value (ascending)
  ord <- order(go_df$Hyper_Raw_PValue, na.last = NA)

  df2 <- go_df[ord[1:10], ]   # top 10

  df2$neglogP <- -log10(df2$Hyper_Raw_PValue)

  # Wrap long labels for visibility
  df2$name_wrapped <- str_wrap(df2$name, width = 50)

  # Reorder for plotting
  df2$name_wrapped <- factor(df2$name_wrapped,
                             levels = rev(df2$name_wrapped))

  ggplot(df2, aes(x = name_wrapped, y = neglogP, fill = neglogP)) +
    geom_col(width = 0.8, color = "black", alpha = 0.9) +
    coord_flip() +
```

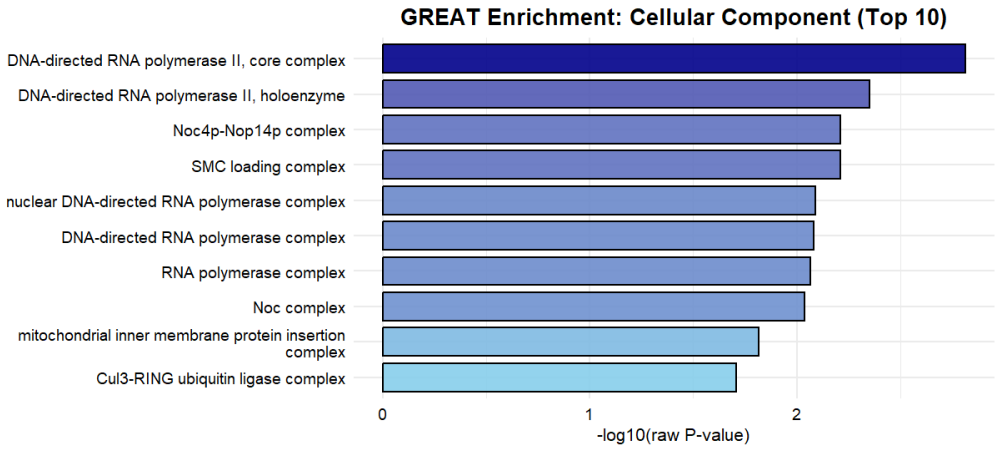
```
    scale_fill_gradient(low = "skyblue", high = "darkblue") +

    # Wide plot theme adjustments
    theme_minimal(base_size = 16) +
    labs(
      title = title_text,
      x = "",
      y = "-log10(raw P-value)"
    ) +
    theme(
      plot.title = element_text(face = "bold", size = 20, hjust = 0.5),
      axis.text = element_text(color = "black", size = 14),
      legend.position = "none",

      # Extra margin on the left to avoid label trimming
      plot.margin = margin(t = 20, r = 20, b = 20, l = 80)
    )
}

plot_great_go(go_bp, "GREAT Enrichment: Biological Process (Top 10)")
```




```
plot_great_go(go_mf, "GREAT Enrichment: Molecular function (Top 10)")
```

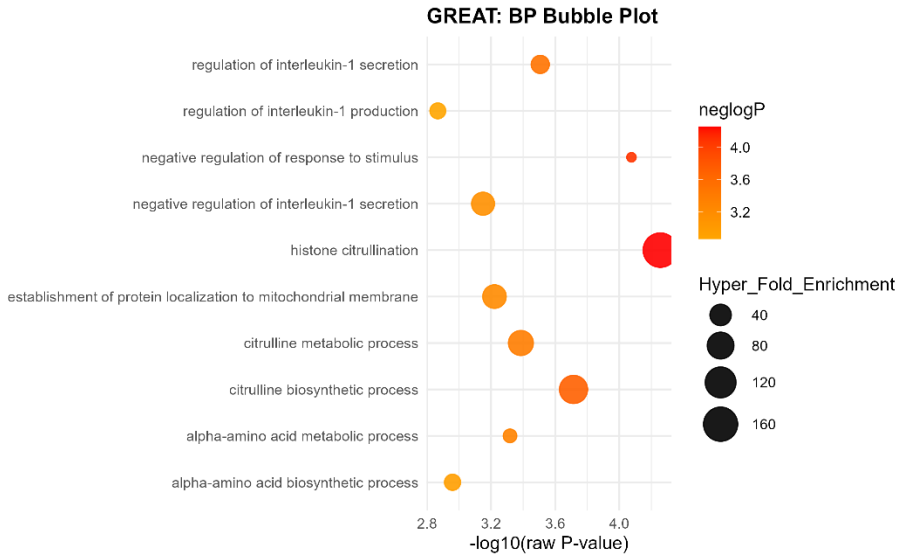


```
plot_great_go(go_cc, "GREAT Enrichment: Cellular Component (Top 10)")
```



GREAT Bubble Plot on Biological Process

```
library(ggplot2)
plot_great_bubble <- function(go_df, title) {
  ord <- order(go_df$Hyper_Raw_PValue)
  df2 <- go_df[ord[1:10], ]
  df2$neglogP <- -log10(df2$Hyper_Raw_PValue)

  ggplot(df2, aes(x = neglogP, y = name)) +
    geom_point(aes(size = Hyper_Fold_Enrichment, color = neglogP), alpha =
0.9) +
    scale_color_gradient(low = "orange", high = "red") +
    scale_size(range = c(3, 12)) +
    theme_minimal(base_size = 14) +
    labs(title = title, x = "-log10(raw P-value)", y = "") +
    theme(plot.title = element_text(face = "bold", size = 16))
}

ggsave("GO_BP_bubble.png",
       plot = plot_great_bubble(go_bp, "GREAT: BP Bubble Plot"),
       width = 10, height = 6, dpi = 300)
```



Save Session Info

```
# Capture sessionInfo() output as text
si_text <- capture.output(sessionInfo())

# Print first 10 lines
cat(si_text[1:20], sep = "\n")
```

```
## R version 4.3.3 (2024-02-29 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 26200)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_India.utf8  LC_CTYPE=English_India.utf8
## [3] LC_MONETARY=English_India.utf8 LC_NUMERIC=C
## [5] LC_TIME=English_India.utf8
##
## time zone: Asia/Calcutta
## tzcode source: internal
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
```