Supplemental code for ‘A rapid, sensitive, scalable method for Precision Run-On sequencing (PRO-seq)’

Julius Judd*

*Department of Molecular Biology and Genetics, Cornell University, Ithaca, New York 14835, USA

Table of Contents

This file contains code used to analyze PRO-seq data for the manuscript 'A rapid, sensitive, scalable method for Precision Run-On'. This document is intended to allow any user to download our raw data (GSEXXXXXX) and exactly reproduce the analyses presented in the manuscript. Please direct questions regarding this code to jaj256@cornell.edu. For convenience, raw and normalized bigWig signal tracks and dREG peak calls are also included in the accompanying gitHub repository: http://github.com/jaj256/qPRO

1. PRO-seq alignment metrics

Lib. Prep.	# Cells	Rep.	# Reads	% A/D	% rRNA	Map %	# Uniq Non-Dup
qPRO-seq	1M	1	40,408,336	16.19%	6.68%	87.31%	15,904,565
qPRO-seq	1M	2	37,056,631	16.19%	6.38%	84.49%	15,066,657
PRO-seq	1M	1	14,556,851	25.26%	5.37%	73.91%	5,796,951
PRO-seq	1M	2	16,284,988	22.45%	5.31%	80.39%	7,051,978
qPRO-seq	250k	1	89,407,788	47.11%	5.87%	37.64%	1,435,034
qPRO-seq	250k	2	50,898,735	42.02%	5.46%	84.87%	5,456,625

2. PRO-seq alignment pipeline

The alignment pipeline used for PRO-seq data can be found here:

http://github.com/jaj256/PROseq_alignment.sh

3. Data Analysis (R scripts)

Loading Packages

```r
library(ggsci)
library(DESeq2)
library(ggpubr)
library(viridis)
library(scales)
library(rtracklayer)
library(GenomicRanges)
library(BiocParallel)
library(tiff)
library(RColorBrewer)
library(ComplexHeatmap)
library(circlize)
library(BRGenomics)
library(extrafont)
loadfonts()
library(FSA)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(tidyverse)
```

Functions

Custom R functions used throughout the remainder of the code chunks

```r
export.bw.stranded <- function(name, path, GRanges.lst) {
  # This function takes the name (chr) of a single entry of a GRanges list,
  # That contains both + and - strand data. It writes fwd and rev bw files
  # after negating the - strand data. The file will be written at "path"
  # (chr), with the name being "name"_[fwd/rev].bw

    # Writing + strand file
  export.bw(
    GRanges.lst[[name]][strand(GRanges.lst[[name]]) == "+"],
    paste0(path, name, "_fwd.bw")
  )

  # Generating new object for - strand
  minus.gr <- GRanges.lst[[name]][strand(GRanges.lst[[name]]) == "-"]

  # Negating abs of - strand scores (so always ends up negative)
  minus.gr$score <- abs(minus.gr$score) * -1

  # Writing - strand file
  export.bw(
    minus.gr,
    paste0(path, name, "_rev.bw")
  )
}
```

```r
RPMnorm <- function(gr){
    gr$score <- (1e6 / sum(gr$score)) * gr$score
    return(gr)
}

ggtheme.jj <- function() {
    # Custom theme options for ggplot2 graphics
    theme_classic(base_size=10, base_family="Helvetica") %+replace%
        theme(
            axis.text = element_text(size = 8),
            axis.ticks = element_line(colour = "black"),
            legend.key = element_blank(),
            panel.background = element_rect(fill = "white", colour = NA),
            panel.border = element_blank(),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            strip.background = element_blank(),
            strip.text = element_text(size=10),
            plot.title = element_text(hjust = 0.5, size = 10),
            axis.title = element_text(size = 8, face = "bold")
        )
}
```

Reading in Data

```r
# Reading in list of GRanges objects of bw signal tracks of
# PRO-seq generated for this manuscript
PROseq.lst <- list(
 "K562_1M_PRO_rep1" = import_bigWig(
    "bw/K562_1M_sPRO_Rep1_deDuped_fwd.bw",
    "bw/K562_1M_sPRO_Rep1_deDuped_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
    ),
 "K562_1M_PRO_rep2" = import_bigWig(
    "bw/K562_1M_sPRO_Rep2_deDuped_fwd.bw",
    "bw/K562_1M_sPRO_Rep2_deDuped_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
    ),
 "K562_1M_qPRO_rep1" = import_bigWig(
    "bw/K562_1M_qPRO_Rep1_deDuped_fwd.bw",
    "bw/K562_1M_qPRO_Rep1_deDuped_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
    ),
 "K562_1M_qPRO_rep2" = import_bigWig(
    "bw/K562_1M_qPRO_Rep2_deDuped_fwd.bw",
    "bw/K562_1M_qPRO_Rep2_deDuped_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
    ),
 "K562_250k_qPRO_rep1" = import_bigWig(
    "bw/K562_250k_qPRO_Rep1_deDuped_fwd.bw",
    "bw/K562_250k_qPRO_Rep1_deDuped_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
    ),
  "K562_250k_qPRO_rep2" = import_bigWig(
    "bw/K562_250k_qPRO_Rep2_deDuped_fwd.bw",
    "bw/K562_250k_qPRO_Rep2_deDuped_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
    )
)

# Reading in bw files for deeply sequenced K562 PRO-seq
# from Core & Martins 2014 (PMID: 25383968)
# Signal tracks were downloaded from GEO (GSE60456) and
# converted to hg38 using liftOver.
# Note: these files are not available in the gitHub
# repository because the file size is too large.
# If you need these files, please request them from the
# author directly.
K562_CoreAndMartins <- import_bigWig(
    "K562_Core2014_PROseq_hg38_fwd.bw",
    "K562_Core2014_PROseq_hg38_rev.bw",
    genome = "hg38", keep.X = F, keep.Y = F
)

# Reading in dREG peak files
# generated using the dREG gateway:
# https://django.dreg.scigap.org/)
K562_1M_sPRO_dREG <-
  read_tsv(
    "dREGpeaks/sPRO_1M_dREGpeaks.bed",
    col_names = c("chr",
                  "start",
                  "stop",
                  "maxScore",
                  "p",
                  "center")
  )
K562_1M_qPRO_dREG <-
  read_tsv(
    "dREGpeaks/qPRO_1M_dREGpeaks.bed",
    col_names = c("chr",
                  "start",
                  "stop",
                  "maxScore",
                  "p",
                  "center")
  )
K562_250k_qPRO_dREG <-
  read_tsv(
    "dREGpeaks/qPRO_250k_dREGpeaks.bed",
    col_names = c("chr",
                  "start",
                  "stop",
                  "maxScore",
                  "p",
                  "center")
  )

# Making dREG files into GRanges
K562_1M_sPRO_dREG.gr <-
  tidyChromosomes(
    makeGRangesFromDataFrame(
      K562_1M_sPRO_dREG,
      keep.extra.columns = T
      )
    )
K562_1M_qPRO_dREG.gr <-
  tidyChromosomes(
    makeGRangesFromDataFrame(
      K562_1M_qPRO_dREG,
      keep.extra.columns = T
      )
    )
K562_250k_qPRO_dREG.gr <-
  tidyChromosomes(
    makeGRangesFromDataFrame(
      K562_250k_qPRO_dREG,
      keep.extra.columns = T
      )
    )
```

Normalizing, merging replicates, saving bigWigs

```
# Merging replicates
PROseq_merged.lst <- list(
  "K562_1M_PRO" = mergeGRangesData(
    PROseq.lst$K562_1M_PRO_rep1,
    PROseq.lst$K562_1M_PRO_rep2
  ),
  "K562_1M_qPRO" = mergeGRangesData(
    PROseq.lst$K562_1M_qPRO_rep1,
    PROseq.lst$K562_1M_qPRO_rep2
  ),
  "K562_250k_qPRO" = mergeGRangesData(
    PROseq.lst$K562_250k_qPRO_rep1,
    PROseq.lst$K562_250k_qPRO_rep2
  )
)

# Normlizing PRO-seq data using Reads Per Million
PROseq_merged_normed.lst <-
  lapply(PROseq_merged.lst, RPMnorm)

PROseq_normed.lst <- lapply(PROseq.lst, RPMnorm)

# Writing merged normalized PROseq bigWig files
```

```r
dir.create("bw/merged_normed/")

lapply(X = names(PROseq_merged_normed.lst),
       FUN = export.bw.stranded,
       path = "bw/merged_normed/",
       GRanges.lst = PROseq_merged_normed.lst)
```

Getting gene list

```r
# Getting list of all hg38 transcripts
genes.gr <-
  tidyChromosomes(
    transcripts(
      TxDb.Hsapiens.UCSC.hg38.knownGene
      ),
    keep.X = F,
    keep.Y = F)

# Removing transcripts smaller than 5kb
genes.gr <-
  genes.gr[which((end(genes.gr) - start(genes.gr)) > 5000)]

# Gene Body (TSS+300 to TES-300)
GB.gr <- genebodies(genes.gr, 300, -300)

# Pause Region (TSS-200 to TSS+300)
PR.gr <- genebodies(genes.gr, -200, 300, fix.end = "start")

# Upstream Region (TSS-700 to TSS-200)
US.gr <- genebodies(genes.gr, -700, -200, fix.end = "start")
```

Getting count matrices

```r
# Getting count matrix for GB (of RPM)
GB_counts.mat <- getCountsByRegions(
    PROseq_normed.lst,
    GB.gr,
    melt = TRUE,
    region_names = GB.gr$tx_name
    ) %>%
    spread(sample, signal)

# Getting count matrix for PR (of RPM)
PR_counts.mat <- getCountsByRegions(
    PROseq_normed.lst,
    PR.gr,
    melt = TRUE,
    region_names = PR.gr$tx_name
    ) %>%
    spread(sample, signal)

# Getting count matrix for US (of RPM)
US_counts.mat <- getCountsByRegions(
    PROseq_normed.lst,
    US.gr,
    melt = TRUE,
    region_names = US.gr$tx_name
    ) %>%
    spread(sample, signal)
```

Plotting correlation

```r
# Generating GB counts matrix in long format pivoted on Replicate
GB_counts_long.mat <-
  GB_counts.mat[,-c(6:7)] %>%
    gather("sample", "count", -region) %>%
    separate(sample, into = c(NA, NA, "method", "Rep")) %>%
    spread(Rep, count)

# Generating GB counts matrix in long format pivoted on method
GB_counts_long_byMethod.mat <-
  GB_counts.mat[,-c(6:7)] %>%
    gather( "sample", "count", - region) %>%
    separate(sample, into = c(NA, NA, "method", "Rep")) %>%
    spread(method, count)

# Plotting correlation scatterplot by replicate
p1 <- ggplot(
  GB_counts_long.mat, aes(x = rep1, y = rep2)) +
    geom_abline(
      intercept = 0,
      slope = 1,
      linetype = "dashed",
      size = 0.5) +
    geom_hex(bins = c(50, 50)) +
    stat_cor(
            method = "spearman",
            label.x.npc = c("left"),
            label.y.npc = c("top"),
            output.type = "text",
            hjust = -0.1
        ) +
    scale_fill_viridis_c(name = "Density") +
    facet_grid(.~method)+
    annotation_logticks(size = 0.25) +
    xlab("Gene Body Rep. 1")+
    ylab("Gene Body Rep. 2")+
    scale_x_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    scale_y_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    ggtheme.jj()

# Plotting correlation scatterplot by method
p2 <- ggplot(
  GB_counts_long_byMethod.mat, aes(x = qPRO, y = PRO)) +
    geom_abline(
      intercept = 0,
      slope = 1,
      linetype = "dashed",
      size = 0.5) +
    geom_hex(bins = c(50, 50)) +
    stat_cor(
            method = "spearman",
            label.x.npc = c("left"),
            label.y.npc = c("top"),
            output.type = "text",
            hjust = -0.1
        ) +
    scale_fill_viridis_c(name = "Density") +
    facet_grid(.~Rep)+
    annotation_logticks(size = 0.25) +
    xlab("Gene Body qPRO")+
    ylab("Gene Body PRO")+
    scale_x_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    scale_y_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    ggtheme.jj()

# Generating PR counts matrix in long format pivoted on Replicate
PR_counts_long.mat <-
  PR_counts.mat[,-c(6:7)] %>%
    gather("sample", "count", -region) %>%
    separate(sample, into = c(NA, NA, "method", "Rep")) %>%
    spread(Rep, count)

# Generating PR counts matrix in long format pivoted on method
PR_counts_long_byMethod.mat <-
  PR_counts.mat[,-c(6:7)] %>%
    gather( "sample", "count", - region) %>%
    separate(sample, into = c(NA, NA, "method", "Rep")) %>%
    spread(method, count)

# Plotting PR correlation scatterplot by replicate
p3 <- ggplot(
  PR_counts_long.mat, aes(x = rep1, y = rep2)) +
    geom_abline(
      intercept = 0,
      slope = 1,
      linetype = "dashed",
      size = 0.5) +
    geom_hex(bins = c(50, 50)) +
    stat_cor(
            method = "spearman",
            label.x.npc = c("left"),
            label.y.npc = c("top"),
            output.type = "text",
            hjust = -0.1
        ) +
    scale_fill_viridis_c(name = "Density") +
    facet_grid(.~method)+
    annotation_logticks(size = 0.25) +
    xlab("Gene Body Rep. 1")+
    ylab("Gene Body Rep. 2")+
    scale_x_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    scale_y_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    ggtheme.jj()

# Plotting PR correlation scatterplot by method
p4 <- ggplot(
  PR_counts_long_byMethod.mat, aes(x = qPRO, y = PRO)) +
    geom_abline(
      intercept = 0,
      slope = 1,
      linetype = "dashed",
      size = 0.5) +
    geom_hex(bins = c(50, 50)) +
    stat_cor(
            method = "spearman",
            label.x.npc = c("left"),
            label.y.npc = c("top"),
            output.type = "text",
            hjust = -0.1
        ) +
    scale_fill_viridis_c(name = "Density") +
    facet_grid(.~Rep)+
    annotation_logticks(size = 0.25) +
    xlab("Gene Body qPRO")+
    ylab("Gene Body PRO")+
    scale_x_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    scale_y_log10(
      labels = trans_format("log10", math_format(10 ^ .x))
      ) +
    ggtheme.jj()
```

Making metaplots

```r
# getting matrix of mean+75%CI by subsampling for
# Promoter region, + strand
PR_sub_fwd.mat <- metaSubsample(
  PROseq_merged_normed.lst,
  promoters(genes.gr, 500, 500),
  binsize = 1,
  first.output.xval = -500
)

# Adding strand column
PR_sub_fwd.mat$strand <- "fwd"

# Creating genes.gr with inverted strands
# For subsampling minus strand at promoters
PR_rev.gr <-
  invertStrand(promoters(genes.gr, 500, 500))

# getting matrix of mean+75%CI by subsampling for
# Promoter region, - strand
PR_sub_rev.mat <- metaSubsample(
  PROseq_merged_normed.lst,
  PR_rev.gr,
  binsize = 1,
  first.output.xval = -500
)

# Adding strand column
PR_sub_rev.mat$strand <- "rev"

# Inverting x values (they are oriented backwards
# because we inverted the strand of our regions)
PR_sub_rev.mat$x <- rev(PR_sub_rev.mat$x)

# Negating values
PR_sub_rev.mat$mean = PR_sub_rev.mat$mean * -1
PR_sub_rev.mat$upper = PR_sub_rev.mat$upper * -1
PR_sub_rev.mat$lower = PR_sub_rev.mat$lower * -1

PR_sub_all.mat <- rbind(
  PR_sub_fwd.mat, PR_sub_rev.mat
)

# Plotting
p5 <- ggplot(
  PR_sub_all.mat, aes(x = x, y = mean)
  ) +
    geom_ribbon(
      aes(ymin = lower,
          ymax = upper,
          fill = strand),
      alpha = 0.5,
      show.legend = F
      ) +
    geom_line(
      aes(color = strand),
      show.legend = F,
      size = 0.1
      )+
    facet_grid(sample.name~.)+
    scale_color_manual(
      values = c(fwd = "#BB0021", rev = "#3B4992")
      )+
    scale_fill_manual(
      values = c(fwd = "#BB0021", rev = "#3B4992")
      )+
    ggtheme.jj()+
    xlab("Distance from TSS (bp)")+
    ylab("Mean + 75% CI")

# Getting GRanges of enhancers
# (dREG peaks > 10kb from a promoter)
enh.gr <- K562_1M_sPRO_dREG.gr
ranges(enh.gr) <- enh.gr$center
enh.gr <-
  enh.gr[(distanceToNearest(enh.gr, PR.gr))@from,]
enh.gr$distTosPR <-
  mcols(distanceToNearest(enh.gr, PR.gr))$distance
enh.gr <- enh.gr[which(enh.gr$distTosPR > 1e4)]

# creating stranded GRs for enhancers
enh_fwd.gr <- enh.gr
strand(enh_fwd.gr) <-  "+"
enh_rev.gr <- enh.gr
strand(enh_rev.gr) <-  "-"

# getting matrix of mean+75%CI by subsampling for
# enhancer regions, + strand

enh_sub_fwd.mat <- metaSubsample(
  PROseq_merged_normed.lst,
  promoters(enh_fwd.gr, 500, 500),
  binsize = 1,
  first.output.xval = -500
)

# Adding strand column
enh_sub_fwd.mat$strand <- "fwd"

# getting matrix of mean+75%CI by subsampling for
# Promoter region, - strand
enh_sub_rev.mat <- metaSubsample(
  PROseq_merged_normed.lst,
  promoters(enh_rev.gr, 500, 500),
  binsize = 1,
```

```r
  first.output.xval = -500
)

# Adding strand column
enh_sub_rev.mat$strand <- "rev"

# Inverting x values (they are oriented backwards
# because we inverted the strand of our regions)
enh_sub_rev.mat$x <- rev(enh_sub_rev.mat$x)

# Negating values
enh_sub_rev.mat$mean = enh_sub_rev.mat$mean * -1
enh_sub_rev.mat$upper = enh_sub_rev.mat$upper * -1
enh_sub_rev.mat$lower = enh_sub_rev.mat$lower * -1

enh_sub_all.mat <- rbind(
  enh_sub_fwd.mat, enh_sub_rev.mat
)

p6 <-
  ggplot(enh_sub_all.mat, aes(x = x, y = mean)) +
  geom_ribbon(
    aes(ymin = lower,
        ymax = upper,
        fill = strand),
    alpha = 0.5,
    show.legend = F
    ) +
    geom_line(
      aes(color = strand),
      show.legend = F,
      size = 0.1
      )+
    facet_grid(sample.name~.)+
    scale_color_manual(
      values = c(fwd = "#BB0021", rev = "#3B4992")
      )+
    scale_fill_manual(
      values = c(fwd = "#BB0021", rev = "#3B4992")
      )+
    ggtheme.jj()+
    xlab("Distance from center of dREG peak (bp)")+
    ylab("Mean + 75% CI")
```

Comparison of pause index

```r
# Getting df of pause indexes for PRO-seq
PRO_PI <-
  getPausingIndices(
    PROseq_merged_normed.lst$K562_1M_PRO,
    PR.gr,
    GB.gr
    )

# Getting df of pause indexes for qPRO-seq
qPRO_PI <- getPausingIndices(
  PROseq_merged_normed.lst$K562_1M_qPRO,
  PR.gr,
  GB.gr
  )

PI.df <- rbind(
    data.frame(method = "PRO", PI = PRO_PI),
    data.frame(method = "qPRO", PI = qPRO_PI)
)

p7 <-
  ggplot(
    PI.df, aes(x = method, y = PI)
    )+
    geom_violin(show.legend =F,
                color = "black")+
    geom_boxplot(
      show.legend = F,
      width = 0.1,
      fill = "white",
      outlier.shape = NA)+
    ggtheme.jj()+
    ylab("Pause Index")+
    xlab(NULL)+
    yscale("log10", .format = T)
```

Intron:Exon read ratio

```
# Getting GRs of introns and exons
exons.gr <-
  tidyChromosomes(
    tidyExons(TxDb.Hsapiens.UCSC.hg38.knownGene),
    keep.X = F, keep.Y = F
    )
introns.gr <-
  tidyChromosomes(
    tidyIntrons(TxDb.Hsapiens.UCSC.hg38.knownGene),
    keep.X = F, keep.Y = F
    )

# dropping all 1st exons (near TSS and influenced by
# Pause signal)
exons.gr <- exons.gr[which(exons.gr$exon_rank > 1)]

# dropping all exons left within 5kb of annotated TSS
exon_dist.hits <- distanceToNearest(exons.gr, PR.gr)
```

```r
exons_safe <- exon_dist.hits[which(mcols(exon_dist.hits)$distance > 5000)]
exons.gr <- exons.gr[exons_safe@from]

# Getting sum of total reads mapping in introns and exons
intronExon.df <- data.frame(
    "data" = c("PRO", "qPRO"),
    "exonCount" = c(
        sum(getCountsByRegions(
            PROseq_merged_normed.lst$K562_1M_PRO, exons.gr
            )
          ),
        sum(getCountsByRegions(
            PROseq_merged_normed.lst$K562_1M_qPRO, exons.gr
            )
          )
    ),
    "intronCount" = c(
        sum(getCountsByRegions(
            PROseq_merged_normed.lst$K562_1M_PRO, introns.gr
            )
          ),
        sum(getCountsByRegions(
            PROseq_merged_normed.lst$K562_1M_qPRO, introns.gr
            )
          )
    )
)

# Getting total bp covered by introns and exons
exonLength <- sum(lengths(exons.gr))
intronLength <- sum(lengths(introns.gr))

# Normalizing read counts per million bp
intronExon.df$exonCount <-
  intronExon.df$exonCount * (1e6 / exonLength)
intronExon.df$intronCount <-
  intronExon.df$intronCount * (1e6 / intronLength)

# Calculating ratio of reads in exons:introns
intronExon.df$ratio <-
  intronExon.df$exonCount / intronExon.df$intronCount


intronExon.df$data <- factor(intronExon.df$data, levels = c("PRO", "qPRO"))

p8 <- ggplot(intronExon.df, aes(x = data, y = ratio))+
    geom_col(fill = "grey")+
    scale_fill_npg()+
    ggtheme.jj()
```

PCA analysis

```r
# Getting count matrix for GB (raw counts for DESeq)
GB_counts_raw.mat <- getCountsByRegions(
    PROseq.lst,
    GB.gr,
    melt = TRUE,
    region_names = GB.gr$tx_name
    ) %>%
    spread(sample, signal)

# Getting count matrix in GB for Core&Martins data
GB_counts_raw_CoreAndMartins.int <-
  getCountsByRegions(
    K562_CoreAndMartins,
    GB.gr
  )

# Combining
GB_counts_raw.mat$CoreMartins <-
  GB_counts_raw_CoreAndMartins.int

# Making df for DESeq
GB_counts_raw.df <- GB_counts_raw.mat
rownames(GB_counts_raw.df) <- GB_counts_raw.mat$region
GB_counts_raw.df <- GB_counts_raw.df[,-c(1, 6,7)]

colData <- data.frame(
    row.names = colnames(GB_counts_raw.df),
    "method" = c("PRO", "PRO", "qPRO", "qPRO", "CoreAndMartins"),
    "rep" = c("1", "2", "1", "2","1"))

# Creating DESeq object
ddsGB <-
  DESeqDataSetFromMatrix(
    GB_counts_raw.df,
    colData,
    design = ~ method
    )

ddsGB <- DESeq(ddsGB)

# Regularizing log transform
rldGB <- rlog(ddsGB, blind = F)

# Getting PCA data
pcaDataGB <-
   plotPCA(rldGB,
           intgroup = c("method"),
           returnData = TRUE)
percentVarGB <- round(100 * attr(pcaDataGB, "percentVar"))

# Plotting PCA
```

```r
p9 <- ggplot(pcaDataGB, aes(PC1, PC2, color = method)) +
   geom_point(size = 1, shape = 17) +
   xlab(paste0("PC1: ", percentVarGB[1], "% variance")) +
   ylab(paste0("PC2: ", percentVarGB[2], "% variance")) +
   scale_color_npg() +
   ggtheme.jj()
```

4. Session Info

```r
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] forcats_0.5.0
##  [2] stringr_1.4.0
##  [3] dplyr_0.8.5
##  [4] purrr_0.3.3
##  [5] readr_1.3.1
##  [6] tidyr_1.0.2
##  [7] tibble_2.1.3
##  [8] tidyverse_1.3.0
##  [9] TxDb.Hsapiens.UCSC.hg38.knownGene_3.4.6
## [10] GenomicFeatures_1.36.4
## [11] AnnotationDbi_1.48.0
## [12] FSA_0.8.30
## [13] extrafont_0.17
## [14] BRGenomics_0.99.26
## [15] circlize_0.4.8
## [16] ComplexHeatmap_2.0.0
## [17] RColorBrewer_1.1-2
## [18] tiff_0.1-5
## [19] rtracklayer_1.46.0
## [20] scales_1.1.0
## [21] viridis_0.5.1
## [22] viridisLite_0.3.0
## [23] ggpubr_0.2.5
## [24] magrittr_1.5
## [25] ggplot2_3.3.0
## [26] DESeq2_1.26.0
## [27] SummarizedExperiment_1.16.1
## [28] DelayedArray_0.12.2
## [29] BiocParallel_1.20.1
## [30] matrixStats_0.56.0
## [31] Biobase_2.46.0
## [32] GenomicRanges_1.38.0
## [33] GenomeInfoDb_1.22.0
## [34] IRanges_2.20.2
## [35] S4Vectors_0.24.3
## [36] BiocGenerics_0.32.0
## [37] ggsci_2.9
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.4-1        ggsignif_0.6.0          rjson_0.2.20
##  [4] htmlTable_1.13.3        XVector_0.26.0          fs_1.3.2
##  [7] GlobalOptions_0.1.1     base64enc_0.1-3         clue_0.3-57
## [10] rstudioapi_0.11         bit64_0.9-7             fansi_0.4.1
## [13] lubridate_1.7.4         xml2_1.2.2              splines_3.6.3
## [16] geneplotter_1.64.0      knitr_1.28              jsonlite_1.6.1
## [19] Formula_1.2-3           Rsamtools_2.2.3         broom_0.5.5
## [22] Rttf2pt1_1.3.8          annotate_1.64.0         dbplyr_1.4.2
## [25] cluster_2.1.0           png_0.1-7               httr_1.4.1
## [28] compiler_3.6.3          backports_1.1.5         assertthat_0.2.1
## [31] Matrix_1.2-18           cli_2.0.2               acepack_1.4.1
## [34] htmltools_0.4.0         prettyunits_1.1.1       tools_3.6.3
## [37] gtable_0.3.0            glue_1.3.2              GenomeInfoDbData_1.2.2
## [40] Rcpp_1.0.4              cellranger_1.1.0        vctrs_0.2.4
## [43] Biostrings_2.54.0       nlme_3.1-145            extrafontdb_1.0
## [46] xfun_0.12               rvest_0.3.5             lifecycle_0.2.0
## [49] XML_3.99-0.3            zlibbioc_1.32.0         hms_0.5.3
## [52] yaml_2.2.1              memoise_1.1.0           gridExtra_2.3
## [55] biomaRt_2.40.5          rpart_4.1-15            latticeExtra_0.6-29
## [58] stringi_1.4.6           RSQLite_2.2.0           genefilter_1.68.0
## [61] checkmate_2.0.0         shape_1.4.4             rlang_0.4.5
## [64] pkgconfig_2.0.3         bitops_1.0-6            evaluate_0.14
## [67] lattice_0.20-40         GenomicAlignments_1.22.1 htmlwidgets_1.5.1
## [70] bit_1.1-15.2            tidyselect_1.0.0        plyr_1.8.6
## [73] R6_2.4.1                generics_0.0.2          Hmisc_4.4-0
## [76] DBI_1.1.0               haven_2.2.0             pillar_1.4.3
## [79] foreign_0.8-76          withr_2.1.2             survival_3.1-11
## [82] RCurl_1.98-1.1          nnet_7.3-13             modelr_0.1.6
## [85] crayon_1.3.4            rmarkdown_2.1           jpeg_0.1-8.1
## [88] GetoptLong_0.1.8        progress_1.2.2          readxl_1.3.1
## [91] locfit_1.5-9.2          data.table_1.12.8       blob_1.2.1
## [94] reprex_0.3.0            digest_0.6.25           xtable_1.8-4
## [97] munsell_0.5.0
```