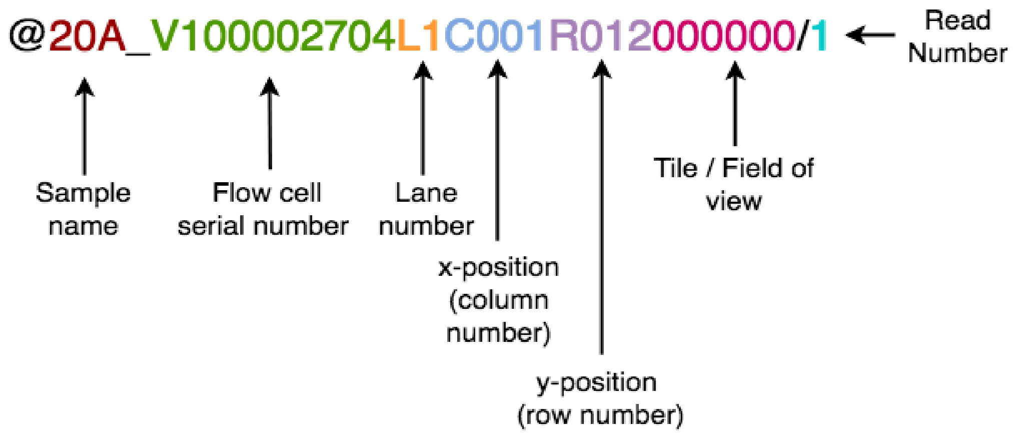
03.QDNA-seq

Overview of Pipeline

BGI header conversion



Implementations

```
library(QDNAseq)
library(dplyr)
library(tidyr)
library(magrittr)
library(QDNAseq.hg38)

# set working directory
setwd("~/Desktop/QDNAseq/data/02_bam/calibrated/")

# sample name?
samples <- lapply(strsplit(list.files(), "\\."), function(x){x[[1]]
[[1]]}) %>% unlist() %>% unique()

# get hg38 ref
bins <- getBinAnnotations(binSize=1000, genome="hg38")

for(SAMPLE in samples){
# set working directory to analysis
ANALYSIS=file.path("~/Desktop/QDNAseq/data/03_analysis/", SAMPLE)
if(!dir.exists(ANALYSIS)){
dir.create(ANALYSIS)
setwd(ANALYSIS)
} else {
```

```
setwd(ANALYSIS)
}
# load bam files
readCounts <- binReadCounts(bins, bamfiles =
paste0("~/Desktop/QDNAseq/data/02_bam/calibrated/",
SAMPLE, ".sorted.marked.calibrated.bam"))
# visualise raw copy numbers
png("01_raw_copy_numbers.png", width = 5, height = 5, units = "in",
res = 600)
plot(readCounts, logTransform=FALSE, ylim=c(-10, 410))
highlightFilters(readCounts, logTransform=FALSE, residual=TRUE,
blacklist=TRUE)
dev.off()
# filtering
readCountsFiltered <- applyFilters(readCounts, residual=TRUE,
blacklist=TRUE)
png("02_median_read_counts_per_bin.png", width = 5, height = 5, units
= "in", res = 600)
isobarPlot(readCountsFiltered)
dev.off()
readCountsFiltered <- estimateCorrection(readCountsFiltered)
png("03_sequence_depth_and_noise.png", width = 5, height = 5, units =
"in", res = 600)
noisePlot(readCountsFiltered)
dev.off()
# correction for GC content
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
png("04_CN_profile_after_GC_correction.png", width = 5, height = 5,
units = "in", res = 600)
plot(copyNumbersSmooth)
dev.off()
# export 1
exportBins(copyNumbersSmooth, file = paste0(SAMPLE, ".txt"))
exportBins(copyNumbersSmooth, file = paste0(SAMPLE, ".igv"), format =
"igv")
exportBins(copyNumbersSmooth, file = paste0(SAMPLE, ".bed"), format =
"bed")
# segmentation
copyNumbersSegmented <- segmentBins(copyNumbersSmooth, transformFun =
"sqrt")
png("05_CN_profile_after_segmentation.png", width = 5, height = 5,
units = "in", res = 600)
```

```
plot(copyNumbersSegmented)
dev.off()
# call copy number
copyNumbersCalled <- callBins(copyNumbersSegmented)
png("06_CN_profile_after_calling_gains_and_losses.png", width = 5,
height = 5, units = "in", res = 600)
plot(copyNumbersCalled)
dev.off()
# export 2
exportBins(copyNumbersCalled, file = paste0(SAMPLE, ".vcf"), format =
"vcf")
exportBins(copyNumbersCalled, file = paste0(SAMPLE, ".seg"), format =
"seg")
save.image("image.RData")
}
```

```
# combine all segmentation files
# seg_files <- list.files("~/Desktop/QDNAseq/data/03_analysis",
pattern = ".seg$",
# recursive = TRUE, full.names = TRUE)
# seg_list <- list()
# for(i in seg_files){
# temp <- read.table(i, header = TRUE)
# seg_list[[i]] <- temp
# }
# seg_all <- do.call(rbind, seg_list)
# rownames(seg_all) <- NULL
# seg_all$SAMPLE_NAME <- gsub(".seg", "", seg_all$SAMPLE_NAME)
# colnames(seg_all) <- c("Sample", "Chromosome", "Start Position",
# "End Position", "Num Markers", "Seg.CN")
# write.table(seg_all,
"~/Desktop/QDNAseq/data/04_gistic2/2025_Jan_20_QDNAseq.seg",
# sep = "\t", quote = F, col.names = T, row.names = F)
```

Deduct CNV

```
library(QDNAseq)
library(dplyr)
library(tidyr)
library(tibble)
library(magrittr)
library(ggplot2)
library(QDNAseq.hg38)

# get hg38 ref
setwd("~/Desktop/QDNAseq/data/QDNA-seq/500 bin/")
samples <- list.files()

# for(x in samples){
# print(x)
# load(file.path(x, "image.RData"))
# copyNumbersSegmented <- segmentBins(copyNumbersSmooth, transformFun
= "sqrt")
# copyNumbersCalled <- callBins(copyNumbersSegmented)
# save.image(file.path(x, "image.RData"))
# saveRDS(copyNumbersCalled, file.path(x, 'copyNumbersCalled.rds'))
# rm = setdiff(ls(), "samples")
# rm(list = rm)
# }

# Load Ref
ref <- read.table("FT190_NT/FT190_NT.txt", header=TRUE, sep = "\t")
colnames(ref)[[5]] <- "reference"
ref <- ref %>% dplyr::select("feature", "reference") %>%
column_to_rownames(var = "feature")

# Modify each
for(i in samples[2:10]){
setwd(file.path("~/Desktop/QDNAseq/data/QDNA-seq/500 bin/", i))
load("image.RData")
new_cn <- merge(copyNumbersSmooth@assayData$copynumber, ref, by =
"row.names", all = TRUE) %>%
column_to_rownames(var = "Row.names")
new_cn[, 1] <- new_cn[, 1] - new_cn[, 2]
new_cn <- new_cn[rownames(copyNumbersSmooth@assayData$copynumber), ]
new_cn$reference <- NULL
copyNumbersFitted <- copyNumbersSmooth
unlockBinding("copynumber", copyNumbersFitted@assayData)
copyNumbersFitted@assayData$copynumber <- new_cn
lockBinding("copynumber", copyNumbersFitted@assayData)
copyNumbersFittedSegmented <- segmentBins(copyNumbersFitted,
transformFun = "sqrt")
png("07_CN_calibrated_segmented_Bins.png", width = 8, height = 4,
units = "in", res = 600)
plot(copyNumbersFittedSegmented)
dev.off()
copyNumbersFittedBinned <- callBins(copyNumbersFittedSegmented,
organism = "human", method = "cutoff")
png("08_CN_calibrated_call_Bins.png", width = 8, height = 4, units =
"in", res = 600)
plot(copyNumbersFittedBinned)
dev.off()
save.image("image.RData")
}
```

Make Plots

```
library(QDNAseq)
library(dplyr)
library(tidyr)
library(magrittr)
library(tibble)
library(QDNAseq.hg38)
library(ggplot2)
library(CGHbase)

setwd("~/Desktop/QDNAseq/")
samples <- list.files("data/QDNA-seq/500 bin")


for(SAMPLE in samples){
setwd(file.path("~/Desktop/QDNAseq/data/QDNA-seq/500 bin", SAMPLE))
load("image.RData")
png("04_CN_profile_after_GC_correction.png", width = 8, height = 4,
units = "in", res = 600)
plot(copyNumbersSmooth)
dev.off()

png("05_CN_profile_after_segmentation.png", width = 8, height = 4,
units = "in", res = 600)
plot(copyNumbersSegmented)
dev.off()
png("06_CN_profile_after_calling_gains_and_losses.png", width = 8,
height = 4, units = "in", res = 600)
plot(copyNumbersCalled)
dev.off()
png("07_CN_calibrated_segmented_Bins.png", width = 8, height = 4,
units = "in", res = 600)
plot(copyNumbersFittedSegmented)
dev.off()

png("08_CN_calibrated_call_Bins.png", width = 8, height = 4, units =
"in", res = 600)
plot(copyNumbersFittedBinned)
dev.off()
}
```