R Writing Sample

Efficient data processing and visualization for cancer genomics research

```
library(dplyr)
library(ggplot2)
library(data.table)
library(FactoMineR)
library(factoextra)

# ============================================================================
# 1. DATA LOADING & PREPROCESSING
# ============================================================================

load_genomic_data <- function() {
  # Load reference genome and filter to autosomes (chr 1-22)
  cn_reference <- fread("cn_reference.csv")[chr %in% 1:22][order(chr, start)]

  # Load sample metadata with quality filters
  icgc_samples <- fread("ICGC_sample_information.csv",
                        select = c("aliquot_id", "wgd_status", "wgd_uncertain", "histology_abbreviation")
                        )[wgd_status == "no_wgd" & wgd_uncertain == FALSE]

  return(list(reference = cn_reference, samples = icgc_samples))
}

# ============================================================================
# 2. EFFICIENT COPY NUMBER PROCESSING
# ============================================================================
```

```
process_copy_number_files <- function(sample_ids, folder_path, cn_reference) {

 # Vectorized gap-filling algorithm
 fill_genomic_gaps <- function(cn_data, chr_lengths) {
  cn_data <- cn_data[!is.na(total_cn) & total_cn <= 20][order(chromosome, start)]

  # Calculate weighted ploidy for missing regions
  total_length <- sum(cn_data$end - cn_data$start)
  ploidy <- ifelse(total_length > 0,
           round(sum(cn_data$total_cn * (cn_data$end - cn_data$start)) / total_length),
           2)

  # Efficient gap detection and filling by chromosome
  filled_data <- rbindlist(lapply(1:22, function(chr) {
   chr_data <- cn_data[chromosome == chr]

   if (nrow(chr_data) == 0) {
    # No data: use overall ploidy
    return(data.table(chromosome = chr, start = 1,
              end = chr_lengths[chr], total_cn = ploidy))
   }

   # Fill gaps at boundaries and between segments
   gaps <- detect_gaps(chr_data, chr_lengths[chr])
   return(rbind(chr_data, gaps))
  }))

  return(filled_data[order(chromosome, start)])
 }
```

```
 # Parallel processing with error handling
 results <- mclapply(sample_ids, function(id) {
  tryCatch({
   file_path <- file.path(folder_path, paste0(id, ".consensus.20170119.somatic.cna.annotated.txt"))

   if (!file.exists(file_path)) return(NULL)

   # Efficient reading with column selection
   cn_data <- fread(file_path, select = c("chromosome", "start", "end", "total_cn"))
   filled_data <- fill_genomic_gaps(cn_data, chr_lengths)

   # Map to reference grid
   return(map_to_reference(filled_data, cn_reference, id))

  }, error = function(e) {
   warning(paste("Failed processing:", id))
   return(NULL)
  })
 }, mc.cores = detectCores())

 # Combine results efficiently
 successful_results <- results[!sapply(results, is.null)]
 combined_data <- do.call(cbind, successful_results)

 return(cbind(cn_reference[, .(chr, start, end)], combined_data))
}

# ==============================================================================
# 3. EXPLORATORY DATA ANALYSIS
# ==============================================================================
```

```
perform_eda <- function(bulk_data, metadata) {

 # Sample-level statistics
 sample_stats <- bulk_data[, lapply(.SD, function(x) {
  list(mean_cn = mean(x, na.rm = TRUE),
      median_cn = median(x, na.rm = TRUE),
      max_cn = max(x, na.rm = TRUE),
      extreme_regions = sum(x > 10, na.rm = TRUE))
 }), .SDcols = 4:ncol(bulk_data)]

 # Outlier detection
 outliers <- sample_stats[order(-max_cn)][1:10]

 # Chromosome-level analysis for top outlier
 max_sample_id <- names(outliers)[1]
 chr_analysis <- analyze_chromosome_patterns(bulk_data, max_sample_id)

 return(list(stats = sample_stats, outliers = outliers, chr_patterns = chr_analysis))
}

# ============================================================================
# 4. MULTI-DATASET PCA VISUALIZATION
# ============================================================================

run_comparative_pca <- function(single_cell_data, bulk_data, simulated_data = NULL, metadata) {

 # Prepare datasets for PCA (samples as rows, genomic regions as columns)
 prepare_pca_matrix <- function(data, sample_type) {
  pca_matrix <- t(as.matrix(data[, 4:ncol(data)]))
```

```
  colnames(pca_matrix) <- paste0(data$chr, ":", data$start)

  return(data.frame(pca_matrix,
                    sample_type = sample_type,
                    sample_id = rownames(pca_matrix)))
}

# Combine all datasets
datasets <- list(
  prepare_pca_matrix(single_cell_data, "Single_Cell"),
  prepare_pca_matrix(bulk_data, "Bulk")
)

if (!is.null(simulated_data)) {
  datasets[[3]] <- prepare_pca_matrix(simulated_data, "Simulated")
}

combined_data <- do.call(rbind, datasets)

# Add metadata annotations
combined_data <- merge(combined_data, metadata, by = "sample_id", all.x = TRUE)

# PCA with preprocessing
pca_matrix <- combined_data[, !names(combined_data) %in% c("sample_type", "sample_id")]
pca_matrix <- pca_matrix[, colSums(is.na(pca_matrix)) == 0]  # Remove NA columns
pca_matrix <- pca_matrix[, apply(pca_matrix, 2, var) > 0]    # Remove zero variance

pca_result <- prcomp(pca_matrix, scale. = TRUE, center = TRUE)
variance_explained <- round(pca_result$sdev^2 / sum(pca_result$sdev^2) * 100, 1)
```

```
 # Create visualization data
 plot_data <- data.frame(
  PC1 = pca_result$x[, 1],
  PC2 = pca_result$x[, 2],
  sample_type = combined_data$sample_type,
  group = create_color_groups(combined_data),
  sample_id = combined_data$sample_id
 )

 return(list(pca = pca_result, plot_data = plot_data, variance = variance_explained))
}

# ============================================================================
# 5. PUBLICATION-READY VISUALIZATION
# ============================================================================

create_pca_plot <- function(pca_results) {

 # Define color palette
 colors <- c(
  "Single_Cell" = "#E31A1C", "Bulk_Ovary" = "#FF7F00",
  "Bulk_Other" = "#1F78B4", "Simulated" = "#CCCCCC"
 )

 p <- ggplot(pca_results$plot_data,
        aes(x = PC1, y = PC2, color = group, shape = sample_type)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_manual(values = colors) +
  scale_shape_manual(values = c(16, 17, 18)) +
  labs(
```

```
    title = "Genomic Copy Number Profile Comparison",
    subtitle = "PCA of Single-Cell, Bulk, and Simulated Data",
    x = sprintf("PC1 (%s%%)", pca_results$variance[1]),
    y = sprintf("PC2 (%s%%)", pca_results$variance[2])
   ) +
   theme_minimal() +
   theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom",
    panel.grid.minor = element_blank()
   )

 return(p)
}

# ============================================================================
# 6. MAIN ANALYSIS PIPELINE
# ============================================================================

main_analysis <- function() {

 # Load and process data
 cat("Loading genomic data...\n")
 data_list <- load_genomic_data()

 cat("Processing copy number files...\n")
 bulk_cn <- process_copy_number_files(
  data_list$samples$aliquot_id,
  "consensus.20170119.somatic.cna.annotated/",
  data_list$reference
```

```
  )

  # Load additional datasets
  single_cell <- fread("all_single_cn.csv")
  simulated <- fread("simulated_CN.csv")

  # Exploratory analysis
  cat("Performing EDA...\n")
  eda_results <- perform_eda(bulk_cn, data_list$samples)

  # Comparative PCA
  cat("Running comparative PCA...\n")
  pca_results <- run_comparative_pca(single_cell, bulk_cn, simulated, data_list$samples)

  # Generate publication plot
  final_plot <- create_pca_plot(pca_results)

  # Save results
  ggsave("genomic_pca_analysis.png", final_plot, width = 12, height = 8, dpi = 300)
  fwrite(eda_results$stats, "sample_statistics.csv")

  cat("Analysis complete. Results saved.\n")
  return(list(eda = eda_results, pca = pca_results, plot = final_plot))
}

# Run analysis
# results <- main_analysis()
```