Allele specific analysis

The six R scripts are part of a comprehensive analysis pipeline designed to study allelic imbalances and gene expression changes in T-cell Acute Lymphoblastic Leukemia (T-ALL). The pipeline integrates various data sources, performs statistical analysis, and visualizes the results to identify key genetic features associated with the disease.

Overview of Scripts and Their Interrelation:

1. **Script 1: add_expression_data_VST.R**
    - **Purpose**: Normalizes log2-fold changes in gene expression using variance stabilizing transformation (VST) and integrates gene identifiers.
    - **Input**: Reads per gene data from RNA-Seq experiments.
    - **Output**: Writes normalized expression data, VST transformed data, and a table of normalized counts to files.
    - **Key Libraries**: **DESeq2** for normalization.
2. **Script 2: ase_nomatch_hg38_functions_refractoryALL.R**
    - **Purpose**: Collects and processes allelic counts, filters them based on depth, and combines loci information.
    - **Input**: Allele count files and reference allele indices.
    - **Output**: Combined information on heterozygous SNPs and corresponding VCF files.
3. **Script 3: ase_nomatch_hg38_refractoryALL.R**
    - **Purpose**: Integrates whole exome sequencing (WES) and RNA-Seq data to assess allele-specific expression.
    - **Input**: BAM files for RNA and DNA, reference genome.
    - **Output**: Allele-specific read counts and quality control plots.
    - **Key Libraries**: **VariantAnnotation**, **BSgenome**.
4. **Script 4: combine_allelic_imbalance_results.R**
    - **Purpose**: Aggregates results of allelic imbalances across samples and identifies genes with frequent imbalances.
    - **Input**: Individual files detailing allelic imbalances per sample.
    - **Output**: Compiled list of genes showing significant allelic imbalances.
5. **Script 5: getrecurrencesummary.R**
    - **Purpose**: Checks genes for recurrence of allelic imbalances and potential nonsense mutations.

    - **Input**: Compiled allelic imbalance data.
    - **Output**: Summary of recurrently imbalanced genes, prepared to link with mutation analysis.

6. **Script 6: getrecurrencesummary_plots.R**
    - **Purpose**: Visualizes the recurrence of allelic imbalances and their relationship to log2-fold changes in expression.
    - **Input**: Data on gene expression and allelic imbalances.
    - **Output**: Plots showing gene expression vs. allelic imbalances and detailed data tables.

Common Themes and Data Flow:

- **Data Preparation**: Scripts prepare and process genetic data from raw sequencing results to more refined allele-specific counts.
- **Integration and Analysis**: The data is integrated across different genetic layers (e.g., RNA and DNA) to understand how variations affect gene expression.
- **Visualization and Reporting**: The results are visualized to identify patterns and exported in formats suitable for further analysis or publication.

External Dependencies and Files:

- **Genomic References**: Scripts rely on reference genomes (e.g., GRCh38) and gene annotations to align and interpret sequencing data.
- **Software Tools**: Uses bioinformatics tools like GATK for read counting and other R libraries for statistical analysis.

Output Files:

- **Normalized expression tables**, **lists of genes with allelic imbalances**, and **visual plots** help in interpreting the complex interactions between genetic variations and their phenotypic expressions in T-ALL.

These scripts collectively enable a detailed genetic study of T-ALL, focusing on how specific allelic variations influence gene expression and potentially contribute to the disease pathology. They are designed for scalability and integration, allowing for comprehensive genomic analysis across multiple samples.

add_expression_data_VST.R

This R script is designed to process and normalize RNA sequencing data for T-ALL (T-cell Acute Lymphoblastic Leukemia) samples. It utilizes the DESeq2 package to compute variance stabilized transformation (VST) values and combines these values with p-values of powered SNP loci for genes. The script aims to analyze gene expression data and allelic imbalances to explore genetic variations and their expression consequences in cancer samples. Here's a detailed breakdown of what each part of the script does:

Preprocessing and Data Preparation:

1. **Library Loading**: The script begins by loading **DESeq2**, a package used for differential gene expression analysis based on count data. Other necessary libraries might be loaded later as needed.
2. **Data Directory and File Reading**: It sets a directory path where the RNA-seq data files are stored and lists files matching a specific pattern (**"ReadsPerGene.out.tab"**), which likely contains the raw count data.
3. **Sample Information**: Extracts sample names from the file paths and creates a sample table specifying sample names, file names, and condition labels (all set to "T-ALL").

DESeq2 Workflow for Normalization:

4. **DESeq DataSet Creation**: Constructs a **DESeqDataSet** object from HTSeq count files using the sample table, aimed for normalization and analysis. This is a common approach in RNA-seq data workflows to manage count data for differential expression analysis.
5. **Size Factor Estimation and VST**: Estimates size factors and performs variance stabilizing transformation (VST) on the DESeq dataset. The VST is used to stabilize the variance across the range of mean values, which helps in downstream analyses by making the data more homoscedastic (constant variance).
6. **Log2 Fold Change and Mean Calculations**: Calculates the mean of VST values across samples and computes log2 fold changes by subtracting the VST means from the VST values. These steps are important for identifying genes with significant changes in expression across conditions or samples.
7. **Gene Information Integration**: Links gene IDs to gene names using an external gene annotation file and computes mean expression values, augmenting the dataset with biologically informative labels.

Plotting and Output:

8. **Data Plotting**: Plots the relationship between VST mean values and log2 fold changes for a visual overview of expression normalization. Such plots help in assessing the quality of the normalization and the overall expression dynamics across samples.
9. **Saving Results**: Writes various output files including normalized counts, VST values, and log2 fold changes into specified directories. These files are useful for downstream analyses or for audit and verification purposes.

Allelic Imbalance Analysis:

10. **Combine P-values Function**: Implements a method to combine p-values for SNP loci per gene, using statistical techniques like Fisher's method. This could be used to strengthen the evidence of allelic imbalance across multiple loci within a gene.
11. **Plotting Imbalances**: Plots the allelic imbalances with respect to expression, aiding in the visual exploration of genes with significant imbalance effects which might be biologically relevant in the context of T-ALL.

Summary:

This script is a comprehensive tool for analyzing RNA-seq data to explore gene expression and allelic imbalances in a cancer genomics context. It combines rigorous statistical analysis with practical output and visualization strategies to provide a deep insight into the molecular biology of T-ALL. The use of VST and integration of genomic annotations enhances the biological relevance of the analysis, making it a valuable component of genomic research workflows.

ase_nomatch_hg38_functions_refractoryALL.R

The R script you provided is involved in several complex operations related to genomic data analysis. It seems designed to process allele counts, analyze heterozygous SNPs, assess allelic imbalances, and visualize allele-specific expression (ASE) results in the context of genomic studies. Here's a breakdown of what each main function in the script does:

get_alleles_chr_nomatch function:

- **Purpose**: This function processes allele counts from files for a specific chromosome. It filters these counts based on a minimum depth requirement and maps reference and alternative alleles to their respective counts.
- **Processes**:
    - Reads allele index information and allele counts from text files.
    - Maps nucleotide codes to actual bases (A, C, G, T).
    - Filters alleles based on minimum depth requirements to ensure data quality.
    - Saves the filtered allele data to an output file.

combine_loci_nomatch function:

- **Purpose**: This function aggregates information from multiple chromosomes into a single dataset and creates a VCF file, which is a standard format in genomics used for describing variant calls.
- **Processes**:
    - Combines allele count files across all chromosomes.
    - Writes combined data to a tab-separated file and converts it into a VCF file.
    - The VCF file includes genomic locations, reference and alternate alleles, and genotype information.

ASEReadCount function:

- **Purpose**: Constructs a command to run the ASEReadCounter tool from GATK, which is used to quantify allele-specific expression from sequencing data.
- **Processes**:
    - Builds a command string with specified parameters for running the tool.
    - Includes input BAM file, VCF file, reference genome, and other filtering parameters like minimum base and mapping quality.
    - This function appears to be setup for testing or demonstration as it only returns the command instead of executing it.

compute_pvals_nomatch function:

- **Purpose**: This function is designed to perform statistical analysis on allele counts to assess allelic imbalance, possibly to identify ASE.

- **Processes**:
    - Reads allele count data from both genomic DNA and transcriptome data.
    - Merges these datasets based on genomic positions.
    - Performs statistical tests to assess whether observed allelic counts differ significantly from expected counts based on genomic data.
    - Applies multiple testing corrections to the p-values obtained from these tests.

ase_annotate function:

- **Purpose**: Annotates allele-specific expression data with gene names, likely using overlap information between genomic coordinates of SNPs and gene annotations.
- **Processes**:
    - Identifies overlaps between genomic regions of interest (from allele-specific expression data) and annotated genomic features.
    - Annotates each relevant genomic coordinate with corresponding gene names.

plot_ase_manhattan function:

- **Purpose**: Visualizes the results of allele-specific expression analysis using a Manhattan plot, which is commonly used to display significant genomic regions (in this case, based on p-values).
- **Processes**:
    - Prepares data for plotting by calculating average positions and breaks for chromosomes.
    - Creates a Manhattan plot displaying -log10 transformed p-values to highlight regions of significant allelic imbalance.

Overall, the script is highly specialized for genomic data analysis, particularly in the context of exploring allele-specific expression and allelic imbalances in genomic datasets. It incorporates data handling, statistical analysis, and result visualization—all common tasks in computational genomics.

ase_nomatch_hg38_refractoryALL.R

This R script is designed for the analysis of allele-specific expression (ASE) in tumor samples, particularly focusing on using RNA sequencing data alongside Whole Genome Sequencing (WGS) data to assess differences in allele expression. It's part of a larger workflow, likely in a cancer research context, and includes extensive use of genomic data processing and visualization tools. Here's a step-by-step explanation of the key components of the script:

Environment Setup and Dependencies

- **Environment**: The script is intended to be run as a standalone R script directly from the command line, indicated by the shebang line (**#!/usr/bin/env Rscript**).
- **Arguments**: It accepts command line arguments to dynamically specify which tumor index it should process, allowing for batch processing or integration with high-throughput computing environments.
- **Libraries**: It loads several R packages necessary for genomic data manipulation (**VariantAnnotation**, **BSgenome.Hsapiens.NCBI.GRCh38**), statistical analysis (**VGAM**), data reading (**readr**), and parallel processing (**parallel**).

Data and File Management

- **Source Files**: The script sources additional R functions from specific paths, likely containing custom functions for handling allele counts and utility functions for the project.
- **Configuration**: Paths to genomic data files, reference genomes, and other utilities are defined, setting up a structured environment for data access.

Data Preprocessing

- **Data Reading**: Reads a sample data frame which likely contains mapping information for RNA and WGS data of different samples.
- **Reference Annotation**: Imports gene annotations using a GFF file, which will later be used for annotating the genomic locations in the ASE analysis.

Main Function: get_ase_ppaml

- **Purpose**: This function is central to the script, designed to analyze allele-specific expression for a given sample pair (tumor WGS and RNA-seq data).
- **Data Paths**: Constructs paths to the data files based on sample identifiers.
- **Directory Management**: Ensures the output directory exists; if not, it creates one.
- **Genomic Data Processing**:
    - If allele counts for the tumor exome are not already processed, it processes them using a utility presumed to be defined in one of the sourced scripts.
    - It then combines these counts across chromosomes and possibly generates a VCF file that contains heterozygous SNP information specific to the tumor.
- **Transcriptomic Data Processing**:

    - It ensures the RNA-seq BAM file has the correct header information before proceeding, possibly reformatting it with SAMtools.
    - Runs a GATK tool for allele-specific read counting, specifying the minimum quality scores for bases and mappings.
- **Post-Processing**:
    - Statistical calculations for ASE, including p-value adjustments and filtering based on statistical significance.
    - Visualization of results using ggplot2, including generating histograms and Manhattan plots to visually assess the distribution and significance of allele-specific expression across the genome.
- **Output**: Saves processed data, statistical results, and plots to specified directories.

Execution and Debugging

- The script retrieves specific sample identifiers from the data frame based on the tumor index provided via command line arguments and runs the **get_ase_ppaml** function for these samples. It also includes comments on potential debugging commands and parallel execution hints, suggesting it's used in a high-throughput or batch processing environment.

Conclusion

The script is well-structured for scalable genomic data analysis, specifically tailored to assess allele-specific expression by integrating genomic and transcriptomic data. It involves preprocessing, statistical analysis, and rich visualization, indicating a comprehensive approach to understanding genetic expression in tumors. This kind of analysis is crucial in cancer research, where understanding differential allele expression can contribute to insights into tumor behavior, heterogeneity, and treatment response.

combine_allelic_imbalance_results.R

The R script provided is designed to analyze and identify recurrent allelic imbalances coupled with gene expression changes (either upregulated or downregulated) across multiple samples in a cancer research study, specifically for refractory T-cell Acute Lymphoblastic Leukemia (T-ALL). Here's an overview of how each part of the script contributes to this objective:

File Reading and Data Aggregation

- **File Identification**: The script begins by searching for files within a specified directory that match a particular pattern (***imbalance_expression_vst.txt**). These files are expected to contain data about allelic imbalances and expression changes for individual samples.
- **Data Loading**: Using **lapply**, it reads each of these files into R as separate data frames. The assumption here is that each file has a similar structure, which allows them to be combined in subsequent steps.

Data Processing

- **Combining Data**: The data frames loaded from each file are combined into a single data frame (**aidf**) using **do.call** with **rbind**. This creates a pooled dataset of allelic imbalance and expression data across all samples.
- **Sample Identification**: The script extracts sample identifiers from the filenames and adds these as a new column (**sample**) to the combined data frame. This is important for tracking which results come from which samples.
- **Filtering**: The data is filtered to include only those rows where:
    - The adjusted p-value (**padj**) is less than or equal to 0.05, suggesting significant allelic imbalance.
    - The log2 fold change (**log2fc**) is either greater than or equal to 1 (upregulation) or less than or equal to -0.73 (downregulation), indicating significant changes in gene expression.
    - Additional filters remove genes located on the X chromosome and those belonging to certain gene families that are typically highly variable or less reliable (HLA, Immunoglobulin heavy chain (IG), and T-cell receptor (TR) genes).

Analysis of Recurrence

- **Identifying Recurrence**: The script calculates the frequency of each gene appearing in the filtered dataset using the **table** function, which is then sorted in descending order to identify the most recurrent genes. This step is crucial for understanding which genes are most commonly affected by allelic imbalances and expression changes across multiple samples, which can indicate genes that are critical in the disease process or potential therapeutic targets.
- **Output**: The top 25 most recurrent genes are displayed using **head**, providing a quick overview of key genes of interest.

Saving Results

- **Writing Data**: Finally, the filtered and processed data is written back to disk, saving the results in a tab-separated text file. This file includes significant, recurrent allelic imbalances and expression changes pooled across samples, ready for further analysis or reporting.

Conclusion

This script is an essential part of a larger analytical pipeline aimed at understanding genetic drivers of refractory T-ALL through the lens of allelic imbalances and differential expression. By focusing on recurrent genes across samples, the analysis can reveal patterns that might not be apparent from single-sample studies, contributing to a more comprehensive understanding of the genetic landscape of the disease.

getrecurrencesummary.R

This R script focuses on analyzing allelic imbalances in genes that are downregulated in a study of refractory T-cell Acute Lymphoblastic Leukemia (T-ALL). The script also sets up a framework to check for nonsense mutations in these genes but leaves the actual mutation analysis as commented out, indicating it may be used in a different context or requires further development. Here's a detailed explanation of the script's components:

Analyzing Allelic Imbalances and Downregulation

1. **Data Loading**: The script begins by loading a dataset from a specified file, which presumably contains information about allelic imbalances and gene expression levels (log2 fold change) across various samples from a T-ALL study.
2. **Data Transformation and Analysis**:
   - **Recurrence Calculation**: It calculates the recurrence of allelic imbalances in genes that are upregulated (log2fc > 0) using a table of gene names. The script then sorts these recurrences in descending order to identify genes most frequently experiencing allelic imbalance.
   - **Aggregating Gene Occurrences**: It further processes the entire dataset to aggregate information per gene, including the samples in which each gene is upregulated or downregulated, and the count of upregulations and downregulations. This provides a detailed view of how each gene's expression is altered across different samples.
3. **Output**: The aggregated data is saved to a text file. This file likely serves as a reference for further analysis or reporting on genes that consistently show allelic imbalances alongside significant expression changes.

Preparing for Mutation Analysis (Commented Out)

The commented sections suggest that there was an intention to analyze specific variant files (VCF or similar formats) for mutations that might explain the gene expression changes observed:

- **File Listing**: Lists files containing single nucleotide variant (SNV) calls, which could be used to identify nonsense mutations in the genes of interest.
- **Variant Extraction Function (getXvariants)**: A function setup to extract variants from these files based on specific criteria (e.g., filtering for novel variants not present in major databases like ExAC or 1000 Genomes).
- **Mutation Analysis**: The approach hints at aggregating these mutation data for further analysis, perhaps to correlate specific mutations with gene expression changes.

Conclusion

This script is structured to perform a detailed analysis of genes with allelic imbalances and expression changes in a T-ALL dataset, potentially preparing for a deeper genetic analysis to understand the mutations underlying these changes. By focusing on downregulated genes with allelic imbalances, the study aims to uncover genetic alterations that might contribute to the disease pathology or response to therapy.

The commented sections imply that the analysis could extend to include genetic variants affecting these genes, although the actual execution of this part of the analysis is not activated in the script provided. This might be due to the need for additional validation, the complexity of integrating variant data, or dependencies on external data or computational resources not detailed in the script.

getrecurrencesummary_plots.R

The R script provided aims to visualize the recurrence of genes that are allelically imbalanced and significantly upregulated or downregulated. It processes genomic data to highlight patterns and associations within the gene expression dataset from a refractory T-cell Acute Lymphoblastic Leukemia (T-ALL) project. Here's a detailed breakdown of its functions:

Data Preparation

1. **Sample Data Loading**: The script starts by loading a sample data frame, which presumably contains identifiers and metadata for RNA sequencing samples. Only samples with a specific FASTQ file present are retained for further analysis.
2. **Log2 Fold Change Data**: It reads a matrix of log2 fold change values that have been variance stabilized transformed (VST) from a previous analysis. This dataset likely contains gene names and their corresponding expression levels across multiple samples.

Filtering and Data Transformation

3. **Allelic Imbalance Data**: The script reads a dataset that details allelic imbalances and filters this data to focus on cases where there is a significant upregulation or downregulation (log2fc thresholds). It further cleans up the data by excluding certain chromosomes and gene families, which are typically excluded due to their complex genetic variability.
4. **Data Combination and Melting**: It combines the allelic imbalance data with the log2fc data to provide a comprehensive view of gene expression alongside allelic imbalances. The **melt** function from the **reshape2** package is used to transform the data frame into a format suitable for plotting, where each row represents a single observation of gene expression and allelic imbalance for a sample.

Visualization

5. **Violin and Dot Plot**: The script creates a complex plot using **ggplot2**:
   - **Violin Plot**: Represents the distribution of log2 fold changes for each gene across all samples.
   - **Dot Plot**: Overlays specific samples to highlight their expression levels. Samples with significant allelic imbalances are marked distinctly to indicate their unique status.
   - The plot is designed to visually compare the spread and central tendency of gene expression while highlighting those genes with significant allelic imbalances.
6. **Saving Outputs**: The plot is saved as a PNG file, and the transformed data used for the plot is saved as a tab-delimited text file for further analysis or reporting.

Purpose and Usage

The primary goal of this script is to provide a visual summary that combines gene expression data with allelic imbalance information. By identifying and highlighting genes that not only show significant expression changes but are also allelically imbalanced, researchers can pinpoint potential biomarkers or therapeutic targets in T-ALL. This script is crucial for translational research, where understanding the genetic basis of disease can inform clinical strategies.

Overall, this script serves as a powerful tool for genomic data exploration, emphasizing the integration and visualization of complex biological datasets to derive meaningful insights in cancer genomics.