For Research Use Only. Not for use in diagnostic procedures.

Limited License for SINGuLAR Analysis Toolset

The SINGuLAR Analysis Toolset is a shared-source, proprietary data analysis resource for Fluidigm customers interested in analyzing or developing software for single-cell gene expression data generated on Fluidigm technology. It is comprised of unsupported software development resources, including R scripts, documentation, and reference data.

Registered users of the SINGuLAR Analysis Toolset may use the code contained in this file in accordance with the terms set forth in sections 1 through 8 below. You may register to use the toolset at the following address: http://http://www.fluidigm.com/singular-analysis-toolset-request.html Unregistered users or users whose registration has not been confirmed with a receipt at the aforementioned website have no rights or permission to use this code.

1. Use of the code in source and binary forms, with or without modification is permitted solely in accordance with section 3 below.

2. Redistribution of the code in source and binary forms, with or without modification is permitted only to employees and agents of entities named as registered users of the SINGuLAR Analysis Toolset. Redistribution, whether in source or binary form must include this license statement.

3. Any use must be in conjunction with a Fluidigm product. Any use with a Fluidigm product may also be in conjunction with data from any source, including products from other vendors. In any case, the code may not be used in conjunction with any product similar to the Fluidigm BioMark™ Real-Time System that is made by another entity.

4. Any redistribution and use shall be in accordance with the laws and export regulations of the United States of America. Under no circumstances shall code be distributed to or used by persons listed on the Denied Persons List maintained by the United States Department of Commerce, or be distributed to or used or executed in a country listed on the Export Control List, List of Extensively Embargoed Countries, or List of Targeted Sanctions Countries Territories maintained by the United States Department of Commerce; appropriate measures shall be taken to ensure that recipients will also refrain from distribution to such parties.

5. Fluidigm will not provide, and is not responsible for providing any end-user support.

6. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7. Fluidigm Corporation grants a limited license to the sample data set(s) provided herein for the purposes of evaluation of the SINGuLAR Analysis Toolset and for training on its use. Other uses not expressly permitted are prohibited.

8. This license, and all matters relating to use of the SINGuLAR Analysis Toolset shall be governed by and interpreted in accordance with the law of the State of California except for its choice of law rules. For any disputes arising out of this Agreement, the parties consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within San Mateo County, California.

9. This license constitutes the entire agreement between you and Fluidigm Corporation. This license may only be amended or supplemented by a writing that refers explicitly to this Agreement and that is signed by duly authorized representatives of both parties.
Contents

Chapter 1: The SINGuLAR™ Analysis Toolset ................................................. 9
  Gene Expression ............................................................................................. 9
  Variant and Mutation Analysis ....................................................................... 9
  The Key Features of the SINGuLAR Analysis Toolset ................................... 10
  Where to Get the Information You Need ....................................................... 11
  Fluidigm Related Products .......................................................................... 11

Chapter 2: Installing R and the SINGuLAR Analysis Toolset ...................... 13
  Install R 3.0.x for Windows ......................................................................... 13
  Install R 3.0.x for Mac ................................................................................. 14
  Install the SINGuLAR Analysis Toolset ....................................................... 15
    Updating on the Mac .................................................................................... 17
  Create the Directory for the SINGuLAR Analysis Toolset .......................... 18
  Rules and Guidelines for Naming Genes and Samples ................................. 18
    The Terminology in Your Dataset ............................................................. 18
    Naming Conventions .................................................................................. 18
    Restrictions .............................................................................................. 18
  Dialogs Unrelated to the R Application ...................................................... 19
  Useful R Function Assistance ....................................................................... 19
  Basic Concepts in R ..................................................................................... 19
    Example 1: Creating an Object from a File ................................................. 20
    Example 2: Creating an Object from a Calculation .................................... 20
    Overwriting Objects ................................................................................... 20
  Useful Functions for Viewing Files and Objects .......................................... 20
    To get a list of the objects created in the current working session ............. 20
    To display the R actions taken for a function ............................................ 20
    To copy and paste within R ....................................................................... 21
    To view the set of data frames for an exp or VC object ......................... 21
    To see a only a specific column within an list object ............................... 21
  Get R Help .................................................................................................. 21

Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression ............... 24
  Single-Cell Studies: Gene Expression ......................................................... 24
  Prepare and Process mRNA Seq Data .......................................................... 25
  Pre-Processing and Initial Assessment ......................................................... 26
Contents

Mapping ................................................................................................... 26
  Prepare Reference Sequences ............................................................... 26
  Map RNA Seq Reads to Reference Genome or Reference Sequences ......... 26
  Calculate Expression Values of Annotated Genes or Transcripts ............. 27
  Example of an mRNA Seq Expression File ............................................ 27

Prepare and Process qPCR Data ........................................................... 28
  Some Guidelines for Data Preparation .................................................... 28

Prepare Sample/Gene Group Annotation Files ....................................... 28

Prepare the Input File ............................................................................. 29

Workflow for Gene Expression with the SINGuLAR Analysis Toolset .......... 30

Log2EX and Limit of Detection (LoD) ....................................................... 31
  Log2EX and LoD in mRNA Seq Experiments ......................................... 31
  Log2EX and LoD in qPCR Experiments .................................................. 31

How to Handle Missing Data .................................................................. 32
  Example 1: Data for an LOD of 24 for qPCR ........................................... 33
  Example 2: Data for mRNA Seq data LOD =1 ....................................... 33

Mathematical and Plotting Functions ..................................................... 33
  Principal Component Analysis (PCA) ..................................................... 33
  ANOVA ............................................................................................... 37
  Hierarchical Clustering (HC) Analysis ................................................... 37
  Correlation Analysis ............................................................................ 38
  Visualization of Expression Profiles ..................................................... 38

The Outlier Identification Method ......................................................... 39

The Automatic Analysis Method ........................................................... 40
  The autoAnalysis Workflow ................................................................... 41

Perform Outlier Identification ............................................................... 42

Perform Automatic Analysis ................................................................. 44
  A Note About Using Functions Without Automatic Analysis ................. 44

Run the Automatic Analysis ................................................................. 45

Examine the Results ............................................................................. 46
  Violin Plot ........................................................................................... 46
  Hierarchical Clustering (HC) Plot .......................................................... 47
  PCA Plots ........................................................................................... 48

The Selected Gene List Output File ....................................................... 50

Advanced Functions ............................................................................. 51

Reading Experimental Data ................................................................... 51
Contents

To read experimental objects ................................................................. 51
To read mRNA Seq data while changing the LoD value to 6 ....................... 51
Annotating Samples ........................................................................... 51
To update a sample list (sample name plus group ID) with a file .................... 51
Annotating Genes ............................................................................. 52
To update genes with group information .................................................. 52
Setting Custom Colors and Symbols ..................................................... 52
To customize the colors and symbols for sample groups of given Exp object ........................ 52
Saving Data .................................................................................... 52
To save the EXP object to files for future use ........................................... 53
To save a gene or sample list to a *.txt file ............................................. 53
To save a plot as an image file .......................................................... 53
Working with the Analysis Output in Excel .............................................. 53
Selecting and Trimming Genes ............................................................. 53
To display the number of detected genes for samples above a threshold of 1 .............. 54
To remove all genes with expression values below twice the LoD ....................... 54
Advanced Plotting Functions ............................................................... 54
Examples of Violin Plots and Pairwise Scatter Plots .................................... 54
To generate a violin plot for each gene from a gene list ............................... 54
To display pairwise Scatter Plots between sample groups ............................ 55
Examples of Advanced PCA Functions ................................................... 55
To analyze with PCA using a previously generated expression object ............. 55
To highlight and label individual points on PCA plots .................................. 56
To apply a previously calculated PCA model to expression data of new samples .......... 57
To call the new exp for the purpose of applying the PCA model to it ................. 57
To apply the previously calculated PCA model to the now active exp object ........ 57
To display a 3D PCA Score Plot ................................................................ 57
Examples of Advanced ANOVA Functions ................................---------- 58
To generate an ANOVA pairwise summary plot from an existing *.fso file ........ 58
To only display the number of significantly expressed genes per sample-group pair .... 58
To find the top genes (most significantly expressed in the sample data) by ANOVA .......... 58
Example: Get the top 200 genes with p-values less than 0.05 ......................... 59
To sort the most significant genes by p-value for a pair of sample groups ........ 59
To display sorted p-values for all sample groups or a pair of sample groups ........... 59
To sort genes by overall p-value of ANOVA ........................................ 60
To sort genes by p-value of T Test in two sample groups, CT1 and CT2 ............. 60
To identify differentially expressed genes via a threshold value in a Volcano plot .... 60
Examples of Correlation Functions ...................................................... 61
To find co-profiling genes (correlated or anti-correlated) with defined correlation threshold .. 62
To find other genes co-expressed with target genes .................................... 62
Examples of Advanced HC Functions .................................................. 63
Contents

Cluster Identification ........................................................................................................ 91
Summary Output Files .................................................................................................... 92
Saving the VC Object as an *.fso Output File .............................................................. 93
Advanced Functions for Targeted Resequencing or DNA Seq Analysis ....................... 93
Reading Experimental Data .......................................................................................... 93
To open a VC object file with a *.fso filename extension ............................................. 93
To open a VCF file with a *.vcf filename extension ..................................................... 93
Annotating Samples ...................................................................................................... 93
To annotate Sample groups from a file ........................................................................ 94
Annotating Variants ...................................................................................................... 94
To annotate variants by CHROM .................................................................................. 94
To annotate variants by GENE SYMBOL ................................................................. 94
To annotate variants by VARIANT_TYPE ..................................................................... 94
To view the annotation ................................................................................................ 94
Setting Custom Colors and Symbols ........................................................................... 95
To customize the colors and symbols for sample groups of a given VC object ............ 95
Saving Data .................................................................................................................. 95
To write the VC dataset out as VCF file ...................................................................... 96
To save the VC object to files for future use .............................................................. 96
To save a variant or sample list to a *.txt file ............................................................. 96
To save a plot as an image file ................................................................................... 96
HC Analysis on Filtered Variant Calls ......................................................................... 96
To conduct HC analysis on a VC object and omit specific data .................................... 96
To change plot colors and omit specific data ............................................................... 97
To display an existing HC analysis ............................................................................. 97
Identifying sample and variant clusters ....................................................................... 97
To identify sample clusters of interest ........................................................................ 97
To identify variant clusters of interest ......................................................................... 97
To update the variant list and create a new VC object with a subset of variants .......... 98
Conducting a Fisher's Exact Test for Case/Control Association Analysis ...................... 98
To perform the Fisher's Exact test on VC data and create a list containing the calculation .. 98
To get the significant variants in the Fisher Exact test from the previous calculation ...... 98
To get the significant variants and display a Manhattan plot ........................................ 99
To select variants of interest ..................................................................................... 99
Updating Variant Quality, Processing a Variant List, and Updating the VC Object .... 100
To update variant quality metrics for all samples ....................................................... 100
To update the VC object to remove low-quality variants from variant list ................... 100
To get a list of the high-quality variants .................................................................... 100
To use this list to update variant list by creating a new VC object without changing the original ................................................................. 101
Getting Variant Calls with Genotyping Information from a Sample ........................... 101
Contents

To retrieve variant list with all calls from a sample or sample group for a previously defined VC object ................................................................. 101
To return a variant list with variant calls only from a sample .............................................. 101
To return a variant list with the most common genotype called above a user defined concordance threshold for all samples in the defined group ................................. 102

Evaluating Variant Call Performance by Comparing Variant Calls of Samples with Common calls ........................................................................ 102
To compare variant calls of individual samples with common calls ..................................... 103
To compare variant calls of sample groups with common calls ............................................. 103

Advanced Functions for WES or Whole Genome DNA Seq Analysis .................................. 103
Identifying Significant Variants and Mutations at Whole Exome Scale .................................. 104

Examining the Results ........................................................................................................... 107
Example: Report for Mutation Analysis ............................................................................. 108
Example: Heatmap for Mutation Analysis ........................................................................... 109
Example: Report for Single-Cell Heterogeneity ................................................................... 109

Saving the VC Object as an *.fso Output File ..................................................................... 110

Appendix A: The List of Functions for Gene Expression ................................................. 111
Appendix B: The List of Functions for Variant and Mutation Analysis ......................... 141
Appendix C: Contents of the EXP Object ................................................................. 157
Appendix D: Contents of the VC Object ......................................................................... 163
Appendix E: References for Gene Expression ............................................................... 171
Appendix F: References for Variant and Mutation Analysis ............................................ 173
Chapter 1: The SINGuLAR™ Analysis Toolset

The SINGuLAR™ Analysis Toolset is designed specifically for single-cell studies of gene expression profiles and variant and mutation analysis using targeted resequencing and whole exome sequencing (WES) data. Built on R, the robust open-source solution for statistical computing, the SINGuLAR Analysis Toolset leverages R’s capacity to streamline data preparation and analysis. The R framework was selected for its flexibility and its ability to support a variety of statistical techniques and modeling. In addition, R permits excellent visualization of multivariate data. Superior plotting capabilities and a wide range of plotting options make R an ideal backbone for the SINGuLAR Analysis Toolset.

Gene Expression

The SINGuLAR Analysis Toolset enables single-cell researchers to perform a series of differential gene expression analysis, such as ANOVA, PCA, unsupervised hierarchical clustering analysis, and gene co-profiling analysis to find genes of biological interest. It also provides a data frame for easy operation and annotation of samples and genes in expression data and for visualizing analysis results by Violin and Box-plots.

This release of the SINGuLAR Analysis Toolset supports:

- **Targeted single-cell gene expression using qPCR on the BioMark System.** In a qPCR experiment, the genes of interest are pre-determined.
- **Single-cell mRNA sequencing.** An RNA Seq experiment contains all the known genes (or isoforms).

By enabling identification of genes that are either differentially expressed or co-expressed, the SINGuLAR Analysis Toolset addresses a key challenge in single-cell gene expression experiments.

Variant and Mutation Analysis

For variant and mutation analysis of targeted resequencing and WES data, the SINGuLAR™ system enables researchers to remove low-quality variants according to user-defined thresholds via a graphical user-interface (GUI). Like gene expression analysis, researchers can easily add annotations to samples and variants.

In single-cell variant analysis, besides using general QC metrics (such as DP, GQ, and so forth) to evaluate the quality of each variant call, researchers can use “variant event occurring in multiple numbers of single cells” to significantly eliminate false variant calls caused by random errors from whole genome amplification and sequencing. The application of false discovery rate (FDR), calculated by single-cell genotyping based on high confidence homozygous sites in bulk
genomic DNA to a binomial test (a cumulative distribution function), can determine the probability of observing a variant in a given number of cells among the total number of cells tested. We have observed that the FDR value is approximately 4.61E-06 in the C1™ system, and most variants in homozygous sites are only found in one cell.

After low-quality variants are removed, researchers can use the SINGuLAR Analysis Toolset to perform unsupervised hierarchical clustering analysis to determine the extent of single-cell heterogeneity. Researchers can further perform a standard case/control association analysis using Fisher’s Exact test to identify significant mutations associated with defined conditions, such as disease traits. All these analyses are accompanied by top-quality visualizations to optimize the insights that can be obtained from single-cell data.

This release of the SINGuLAR Analysis Toolset supports VCF data generated from diverse variant callers, such as GATK, MuTect, and VarScan. The supported variant annotation package is SnpEff.

The Key Features of the SINGuLAR Analysis Toolset

<table>
<thead>
<tr>
<th>Identify Significant Variants / Mutations at Whole Exome Scale</th>
<th>With a minimum of clicks in the graphical user interface (GUI), remove low-quality variants using the set of filters and rational default thresholds based on the analysis or specify your own combination of filters and their threshold values.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In mutation detection at single cell resolution, besides using general QC metrics (like DP and GQ) to evaluate the quality of each variant call, remove low-quality variant calls caused by random error from DNA amplification and sequencing using single-cell related filters, such as (1) “variant found in multiple single cells” (at least 2-3) and (2) “variant allele frequency is not rare” (&gt;10%).</td>
</tr>
<tr>
<td></td>
<td>Apply the false discovery rate (FDR) calculation in determining the probability of observing a variant in a given number of cells among the total number of cells tested.</td>
</tr>
<tr>
<td></td>
<td>After low-quality variants are removed, perform unsupervised hierarchical clustering analysis to determine the extent of single-cell heterogeneity.</td>
</tr>
<tr>
<td></td>
<td>Further perform a standard case/control association analysis using Fisher’s Exact Test to identify significant mutations associated with defined conditions, such as disease traits.</td>
</tr>
</tbody>
</table>
Where to Get the Information You Need

To find the information you need, click any of the links in the diagram. To Install R and the SINGuLAR Analysis Toolset, see Chapter 2: Installing R and SINGuLAR Analysis Toolset.

Fluidigm Related Products

- Access Array™ System
- Access Array™ Target-Specific Primers
- BioMark™ HD System
- BioMark™ HD Data Collection Software
- C1™ Single-Cell Auto Prep System
- C1™ Single-Cell Auto Prep IFC & Consumables
- D3™ Assay Design
- DELTAGene™ Assays
- Real-Time PCR Analysis Software
- 48.48, 96.96, and 192.24 Dynamic Array™ IFCs for Gene Expression & Variant Analysis
Chapter 2: Installing R and the SINGuLAR Analysis Toolset

Install R 3.0.x for Windows

1. Ensure that you are installing the software on a computer running on the Windows operating system, (XP or higher).
3. Click the CRAN link in the left menu bar.
4. Scroll down to the closest mirror (replicate server) to your region, then download R for Windows.
5. Download and run the *.exe file for your environment. An install wizard walks you through the process. Install the base subdirectories only.
6. Install R version 3.0.3 or later.
Install R 3.0.x for Mac

1. Ensure that you are installing the software on a computer running the Mac operating system (OS X 10.6 Snow Leopard, or higher). Fluidigm requires the X11 server to run R 3.0.x on the Mac. The X11 server is no longer included in OS 10.6 or later, so you need to download and install it.


3. Click the CRAN (Comprehensive R Archive Network) link in the left menu bar.

4. Scroll down to the closest mirror (replicate server) to your region, then download R for Windows.

5. Download and run the *.exe file for your environment. An install wizard walks you through the process.

6. Install R version 3.0.3 or later.
Install the SINGuLAR Analysis Toolset

The SINGuLAR Analysis Toolset is compatible with R for the Microsoft Windows® and Mac operating systems.

2. Launch R. The R graphical user interface (GUI) displays.
3. Do either of the following:
   - If you are using a Windows computer, click Packages > Install package(s) from local zip file:
   - If you are using a Mac computer, click Packages & Data, and then click Package Installer:

   ![Package Installer](image)

   - If you are using a Mac computer, click Packages & Data, and then click Package Installer:
Chapter 2: Installing R and the SINGuLAR Analysis Toolset

In the Packages Repository, click Local Binary Package:

4 Although no files are selected, click Install in the lower right corner to navigate to the *.zip file, and then open that file.
5 Click the file fluidigmSC_v 3.x.zip.

Note: On the Mac, you need to additionally close the Packages Repository dialog to get back to the R command line.

6 At the R command prompt, type in library(fluidigmSC) and press Enter.
Install the SINGuLAR Analysis Toolset

7 Type `firstRun()`, which will install the following dependent packages:

- **lattice.** A high-level data visualization system that produces trellis graphics with an emphasis on multivariate data for typical graphics needs. Trellis graphics display multivariate data by plotting subsets of the data on adjacent panels. Often each panel is a plot of the data for a given level of a categorical variable. This system is also flexible enough to handle most non-standard requirements.

- **tcltk.** The Tool Command Language (tcl) scripting language that includes the Tool Kit (tk) library of GUI elements, such as listboxes, scrollbars, and sliders.

- **SDMTools.** A set of Species Distribution Modelling Tools for post-processing the outcomes of species distribution modeling exercises, and it includes novel methods for comparing models and tracking changes in distributions through time and methods for visualizing outcomes, selecting thresholds, calculating measures of accuracy and landscape fragmentation statistics, and so forth.

- **rgl.** A visualization device system that provides medium to high level functions for 3D interactive graphics, including functions modelled on base graphics, such as `plot3d()`, as well as functions for constructing representations of geometric objects, such as `cube3d()`. Output may be on screen using OpenGL or to various standard 3D file formats (such as WebGL, PLY, OBJ, STL) and 2D image formats (such as PNG, Postscript, SVG, PGF).

8 Select the nearest mirror and press Enter. You will need to set the CRAN mirror for the session.

**Note:** For efficient download, select a local download site. For example, to download from UCLA, choose the USA (CA2) mirror. This ensures that you receive R updates and can access online help.

For Windows: If the mirror is too slow or does not function, select the next closest mirror by navigating to Packages > Set CRAN mirror, then entering `firstRun()` again.

For the Mac: If the mirror is too slow or does not function, select the closest mirror by entering:

```r
chooseCRANmirror()
```

Then enter `firstRun()` again.

The R GUI will display a series of messages. Proceed to set the directory for the SINGuLAR Analysis Toolset for data analysis.

**Updating on the Mac**

To update your fluidigmSC package, choose Packages & Data > Package Manager, and then check the box Loaded in the Status column for the fluidigmSC package labeled Fluidigm SINGuLAR Analysis Toolset. Be certain that there is only one fluidigmSC package listed.

If more than one is listed, (1) close all R consoles, (2) start a new R console, (3) run local installation as new installation, (4) enter `library(fluidigmSC)`, and then (5) enter `firstRun()`.
Create the Directory for the SINGuLAR Analysis Toolset

In Windows, navigate to File > Change dir to set the working directory for this session. On the Mac navigate to Workspace > Save Default Workspace to set the working directory for this session.

**Note:** To facilitate data handling, set the data directory to the folder containing the data being analyzed.

Rules and Guidelines for Naming Genes and Samples

The Terminology in Your Dataset

- **Gene.** Either a gene (in mRNA Seq) or an assay (in qPCR data).
- **Sample.** A cell, a set of cells, or a tissue.
- **Variant.** A variation in the genome at a specific location, specified by location information.
- **Gene Group.** A set of genes on a chromosome that tend to be transmitted together.
- **Sample Group.** A cell type, treatment, or any given condition that groups samples together. Ideally, the samples in the same group should have the "same" gene expression profiling.
- **Variant Group.** A type of call (SNP, DEL, INS), location of calls (chromosome), or gene ID of calls (GAPDH).

Naming Conventions

- Sample and gene names can include any combination of alphanumeric (digits and letters) characters.
- **All characters are case-insensitive.** For example, A is identical to a.
- To improve the display of sample and gene names in HC plots, use brief names and these naming conventions (with either a hyphen or an underscore between the group and the name):
  - For Samples: Group_SampleName
  - For Genes: Group_GeneName
  - There is no need to enter a prefix here if you are using the Cell Attribution worksheet rev (current) or newer and you have entered the group information as the "sample type" in the worksheet. In this case, sample annotation is done automatically, and GroupID is separated by an underscore "_".

Restrictions

Samples and gene names have non-alphanumeric character restrictions:
Dialogs Unrelated to the R Application

Some functions that are not linked to the R application display a dialog that allows you to select a file. It is important not to click anywhere outside this dialog before completing the file selection. If you click outside this dialog, it will be hidden behind the R application window.

To find the dialog, minimize the R application window or press the keys Alt+Esc to toggle through the open windows until the dialog is in front of the R application window.

Useful R Function Assistance

Useful R functions are available on the internet. For example:

http://sites.tufts.edu/cbi/files/2013/02/Key-R-commands_10_14_2010.doc
http://www.calvin.edu/~scofield/courses/m143/materials/RcmdsFromClass.pdf

Basic Concepts in R

R is a programming language that was created by Robert Gentleman and Ross Ihaka at the University of Auckland as a testbed for trying out ideas in statistical computing. It is based on S, a language and computational environment developed at Bell Laboratories for carrying out "statistical" computations.

The R project is now an international collaboration of researchers in statistical computing. Releases of the R environment are made through the CRAN (comprehensive R archive network) twice per year.

In R, values or files are stored by assigning them a name for the current session. A named file in R is called an "object". The formula for creating an object is as follows:

```r
object <- function
```

Thus:

```r
object <- file/data calculation to use that object
```
Example 1: Creating an Object from a File

```r
exp <- readExpObject()
```

In this example, an expression object named "exp" is created for this session and will be called upon for various tasks in subsequent functions. The function in this example opens a file dialog that allows you to read in an existing file.

Example 2: Creating an Object from a Calculation

```r
hc <- HC(exp)
```

In this example, an expression object that was created in Example 1 is used by the HC function to calculate hierarchical clustering. These calculations are stored as an "hc" object that can be manipulated during further analysis in this session. Examples of manipulations of this "hc" object are changing the colors of genes and samples in the HC plot or selecting subsets of gene clusters.

Overwriting Objects

At any point in the session, if another function is performed that creates an already-defined object, the newly created object will overwrite the initial object. For example, if you identified a subset of samples based on hc called hc_sample_list and update your exp file to contain only those samples, the original exp file will no longer exist in its original form.

To avoid replacing your original exp object:

- Never manipulate the original exp after removing outliers.
- Always provide a unique object name within the session. For example, if you named an object hc_sample_list, name a newly generated subset of samples hc_sample_list_exp.

Useful Functions for Viewing Files and Objects

If the R session is terminated these objects are no longer active. Objects can be saved to the computer as files and brought into a new session. For more information on saving objects, see the “Saving Data” sections for gene expression or variant analysis.

To get a list of the objects created in the current working session

```r
objects()
```

To display the R actions taken for a function

The SINGuLAR Analysis Toolset package contains a library of R-derived actions that perform specific functions or calculations and display plots. The standard R function for displaying the R code for a particular function is:

```r
print(enter your function here)
```
For example, to display the R code used for calculating and displaying an HC plot, enter:

```r
print(HC)
```

Alternatively, you can type the following:

```r
HC
```

To copy and paste within R

Select text, copy and paste it into the R console, and then click Ctrl X.

To view the set of data frames for an exp or VC object

```r
names(exp) or names(vc)
```

This function returns the set of data frame names. In the fluidigmSC library, the returned names are as follows:

```r
> names(exp)
[1] "obj_type"   "data_type"   "lcm"         "gene_lods"
[5] "org_data"   "gene_list"   "sample_list" "outlier_list"
[9] "log2ex_data" "log2ex_avg_data" "summary"
```

To see a only a specific column within an list object

```r
object$column_header
```

For example: `vc$variant_list`

Get R Help

On Windows: To launch help from a Windows computer:

1. Type `??fluidigmSC` at the R command line. Your internet browser appears.
2. At the upper left of your internet browser, click the link `fluidigmSC::fluidigmSC_<version number>`.
3. At the bottom right, click the link `[Package fluidigmSC version <version number> Index]`.

On the Mac: To launch help from a Mac computer, select `Packages & Data > Package Manager`, and then select `Fluidigm SINGuLAR Analysis`.

On Windows or the Mac: To get specific help:

- For a function, type `?functionName` (for example, `?autoAnalysis`).
- For a list of categories and functions in each category, type `scExpFunctions()` Or `scVarFunctions`.
- For a list of variables and their associated data types, type `scVariables()`.
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

This chapter provides a high-level view of preparation and processing of data and the types of analysis and visualization that the SINGuLAR Analysis Toolset can perform on your gene expression data.

Single-Cell Studies: Gene Expression

The aim of this chapter is to focus on the capabilities of the SINGuLAR Analysis Toolset package rather than to explore all available methods of analysis or visualization. To provide additional guidance, Table 1 lists examples of published research where the Fluidigm® BioMark™ HD System was used to study single-cell gene expression for biological insight. You can find additional publications in Appendix E: References for Gene Expression.

Table 1: Comparison of secondary analysis methods in published research using the BioMark™ HD System for single-cell gene expression (HC = Hierarchical Clustering; PCA = Principal Component Analysis; LDA = Linear Discriminant Analysis; DTA = Decision Tree Analysis; JSD = Jensen-Shannon Divergence).

<table>
<thead>
<tr>
<th>Field</th>
<th>Violin</th>
<th>+/-</th>
<th>Pairwise</th>
<th>HC</th>
<th>PCA</th>
<th>LDA</th>
<th>DTA</th>
<th>JSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guo et al. 2010</td>
<td>Developmental Biology</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dalerba et al. 2011</td>
<td>Cancer</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flatz et al. 2011</td>
<td>Immunology</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pang et al. 2011</td>
<td>Neuroscience</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vincent et al. 2011</td>
<td>Developmental Biology</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agullo et al. 2011</td>
<td>Stem cells</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Buganim et al. 2012</td>
<td>Stem cells</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moignard et al. 2013</td>
<td>Hematopoietic stem cells</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tanabe et al. 2013</td>
<td>iPSC</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prepare and Process mRNA Seq Data

How to prepare and process data before analysis with the SINGuLAR Analysis Toolset:

- **Raw RNA Seq data (FASTQ format)**
  - **Read QC (FASTX or FastQC Toolkit)**
    - **Report QC statistics**
    - **Low quality score? Tag contamination in reads?**
      - **Yes**
        - **Trim data (remove aberrant bases)**
        - **High quality reads (FASTQ format)**
      - **No**
    - **Identification of known/novel gene expression and splicing events?**
      - **Yes**
        - **Map reads to reference genome or known RefSeq genes (TopHat/Bowtie)**
          - **Cufflinks**
        - **FPKM of genes/transcripts**
        - **Differential gene expression analysis (SINGuLAR Analysis Toolset)**
      - **No**
        - **Map reads to known RefSeq genes (RSEM/Bowtie)**
          - **TPM/FPKM of known genes/transcripts**
Pre-Processing and Initial Assessment

A quality control check on raw sequencing data produces better mapping results. It helps identify and fix potential issues in FASTQ files, including sequencing artifacts, genetic contaminants (primers, vectors, and adaptors), and low-quality reads. To perform this quality control:

1. Review quality scores across all bases using the FastQC toolkit (downloadable from http://www.bioinformatics.babraham.ac.uk/projects/download.html#fastqc) or the FASTX toolkit (downloadable from http://hannonlab.cshl.edu/fastx_toolkit/).
2. If you notice issues, trim the data by removing aberrant bases or reads to generate a high quality FASTQ file.

Mapping

Map sequences to a reference genome or known-gene reference sequences and calculate the expression values of transcripts. The recommended mapping and expression value calculation tools are RSEM (downloadable from http://deweylab.biostat.wisc.edu/rsem/) and the Tuxedo Suite which includes these components:

- Tophat (http://tophat.cbcb.umd.edu/downloads/)
- Cufflinks (http://cufflinks.cbcb.umd.edu/downloads/)

In general, the steps to map RNA Seq data and calculate expression values are:

Prepare Reference Sequences

To map reads to reference sequences, begin by indexing the reference sequences.

For example, for the RSEM mouse genome, enter:

```
rsem-prepare-reference --gtf <mm9.gtf>
--transcript-to-gene-map
<knownIsoforms.txt> --bowtie-path </sw/
bowtie> </data/mm9> </ref/mouse_125>
```

For Tophat/Bowtie, enter:

```
bowtie2-build [options]* <reference_in> <bt2_index_base>
```

Map RNA Seq Reads to Reference Genome or Reference Sequences

For example, for the RSEM mouse liver, enter:

```
rsem-calculate-expression -p <8>
--paired-end </data/mmliver_1.fq> </data/
mmliver_2.fq> </ref/mouse_125>
<mmliver_paired_end_quals>
```
RSEM generates expression values of gene and transcripts.

For Tophat/Bowtie, enter:

```
tophat -G refseq.gtf  genome-ref-index read1.fastq read2.fastq
```

**Calculate Expression Values of Annotated Genes or Transcripts**

*Note: RSEM already generated expression values of gene and transcripts in the previous step.*

For **Cufflinks**, enter:

```
cufflinks -o cufflinks_brain  –G refseq.gtf tophat_brain/accepted_hits.bam
```

After this workflow is complete, the expression data of individual samples (FPKM/TPM) should be integrated into a single tab-delimited file before secondary analysis in the SINGuLAR Analysis Toolset.

**Example of an mRNA Seq Expression File**

mRNA Seq experimental results are in a matrix format in a *.txt tab-delimited file in which the first row contains the sample names and first column contains the gene annotation (Gene ID, Isoform ID, and so forth). The file format for mRNA Seq file must be in a tab-delimited txt file. For example:

```
Sample Name
Gene ID
```

![Example of mRNA Seq Expression File](image.png)
Prepare and Process qPCR Data

Data should be analyzed and reviewed using Real-time Analysis software prior export to ensure proper pass-fail calls. If using DELTAgene assays, identify and eliminate data from non-specific amplification to improve specificity and sensitivity. The SINGuLAR Analysis Toolset package uses the “Pass” and “Fail” scores when analyzing.

The following CSV (comma-delimited) file from Fluidigm® Real-Time Analysis Software contains the table of heatmap data:

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chip Run A</td>
<td>BioMar</td>
<td>1.36E+09</td>
<td>96.96</td>
<td>(136G96</td>
<td>96</td>
<td>FROX</td>
<td>EvaGreen</td>
<td>1/9/2013</td>
<td>14.51</td>
</tr>
<tr>
<td>2</td>
<td>Applicati 4.0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Applicati 20130423</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Export Tab Table Results</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Quality Th k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Baseline C Linear (Derivative)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ct ThreshAuto (Global)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Chamber</td>
<td>Sample</td>
<td>Sample</td>
<td>Sample</td>
<td>EvaGreen</td>
<td>EvaGreen</td>
<td>EvaGreen</td>
<td>EvaGreen</td>
<td>EvaGreen</td>
<td>EvaGreen</td>
</tr>
<tr>
<td>11</td>
<td>ID</td>
<td>Name</td>
<td>Type</td>
<td>rConc</td>
<td>Name</td>
<td>Type</td>
<td>Value</td>
<td>Quality</td>
<td>Call</td>
<td>Threshold In Range</td>
</tr>
<tr>
<td>13</td>
<td>S96-A01</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G01</td>
<td>Test</td>
<td>8.268477</td>
<td>1 Pass</td>
<td>0.018799</td>
<td>84.063</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>S96-A02</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G02</td>
<td>Test</td>
<td>999</td>
<td>0 Fail</td>
<td>0.018799</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>S96-A03</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G03</td>
<td>Test</td>
<td>999</td>
<td>0 Fail</td>
<td>0.018799</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>S96-A04</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G04</td>
<td>Test</td>
<td>19.36748</td>
<td>1 Pass</td>
<td>0.018799</td>
<td>88.840</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>S96-A05</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G05</td>
<td>Test</td>
<td>999</td>
<td>0 Fail</td>
<td>0.018799</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>S96-A06</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G06</td>
<td>Test</td>
<td>12.31763</td>
<td>1 Pass</td>
<td>0.018799</td>
<td>81.647</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>S96-A07</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G07</td>
<td>Test</td>
<td>18.78365</td>
<td>1 Pass</td>
<td>0.018799</td>
<td>86.268</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>S96-A08</td>
<td>CT1_94-1</td>
<td>Unknown</td>
<td>1 G08</td>
<td>Test</td>
<td>999</td>
<td>0 Fail</td>
<td>0.018799</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
```

Some Guidelines for Data Preparation

The SINGuLAR Analysis Toolset does not normalize data, since there is no definitive way to analyze burst-like expression for single cells. Normalization of data will be a future consideration.

**Note:** Do not include delta Ct Values in R, since those values include negative numbers, which are converted to the LoD.

Prepare Sample/Gene Group Annotation Files

**Sample GroupID:** Create a sample list file that contains all samples from all expression files and their conditions as a *.txt file. This file needs to contain with a minimum of two columns with headers as SampleID and GroupID, respectively. The remaining rows are sample names (first column) and their corresponding group name (second column). For example:
Gene GroupID: The gene group file also needs to be a *.txt file. In this file, the first column contains gene or assay names with the header "GeneID," and the second column contains the header "GroupID."

IMPORTANT! When you open a tab-delimited file in Excel, Excel uses the “General” format for each cell by default. This is a problem for some gene names, which are converted to a Date. Thus, it is important to assign the Gene ID column as “Text” in the import wizard.

Prepare the Input File

Before you begin analysis:

1. Ensure that each cell line (or condition) has its own expression data file. Although this is not absolutely necessary, it is highly recommended.
2. Move all expression data files that you want to analyze into a common directory.
3. Create a sample list file containing all samples from all expression data files and their conditions as a *.txt file with a minimum of two columns with headers as SampleID and GroupID, respectively.
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

Workflow for Gene Expression with the SINGuLAR Analysis Toolset

mRNA Seq – RNA Sequencing Data
1. Get sequencing data from Illumina sequencer.
2. Map data and call expression.
3. Export file as *.txt with Sample ID and expression values.
4. Prepare sample annotation file with "Sample ID" and "Group ID".
5. Export as *.txt file.

qPCR – Target Amplification Data
1. Get qPCR results from BioMark System.
2. Process the data using Fluidigm Real-Time PCR Analysis software.
3. Export processed qPCR results to tab-delimited *.csv file.

Set Up Your R Environment
   Navigate to your output folder or create a new one. Click OK.

Basic Analysis

SINGuLAR™ Analysis Toolset
- Perform Outlier Identification: identifyOutliers()

SINGuLAR™ Analysis Toolset
- Perform Automatic Analysis: autoAnalysis()

Advanced Analysis

Principal Component Analysis (PCA)
```r
pca <- PCA(exp)
```

ANOVA
```r
anova <- ANOVA(exp)
```

Hierarchical Clustering (HC)
```r
hc <- HC(exp)
```

Correlation Analysis
- Box plots, violin plots, etc.
**Log₂EX and Limit of Detection (LoD)**

The SINGuLAR Analysis Toolset performs all statistical analyses based on the expression values in the Log₂ domain. It supports expression values (TPM, FPKM, or other values in the linear domain) in mRNA sequencing experiments and Ct values in qPCR experiments.

**Log₂EX and LoD in mRNA Seq Experiments**

Typical expression levels in RNA Seq experiments are calculated as FPKM (fragments per kilobase per million) or TPM (transcripts per million). Since the primary interest in expression analysis is the fold change, these values are converted to Log₂ values (Log₂Ex). Because small numbers (less than 1) can become very large negative values in the Log₂ domain, a LoD is required to eliminate background noise. By default, the LoD for RNA Seq experiments is set to 1 so that background expression values are zeros in the Log₂ domain.

In general, the LoD depends on read depth. If the single read is considered as background, the LoD can be defined as 1 over RPSM (reads per sample in million).

Linear expression values are converted to Log₂Ex by the following formula:

\[
\text{Log₂Ex} = \begin{cases} 
\text{Log}_2(\text{exp}) & \text{if } \text{LoD} \leq \text{exp} \\
0 & \text{if } \text{LoD} > \text{exp}
\end{cases}
\]

**Log₂EX and LoD in qPCR Experiments**

When qPCR experiments are run on bulk RNA samples, the results are typically displayed as fold-change differences between samples for each individual gene and known controls. Because of the extensive normal variation in a given gene at the single-cell level, looking at fold changes between individual cells is potentially not very informative. A better approach may be to first assess the population behavior for each gene. By assessing which genes display a lognormal, unimodal distribution within the cell population under investigation, this type of first-pass analysis can provide the first significant insight into the unique biology of the cell population and dictate further, more directed analyses. This is best done by studying histograms that bin expression levels and display the number of cells in each bin. To generate such histograms, the expression for each gene must be comparable between different single-cell samples. The researcher starts by calculating the LoD first and then computing Log₂Ex values.

As a result of the lognormal distribution described by Bengtsson et al. (2005) and others, it is useful to view single-cell data as expression level above detection limit on a log scale. For qPCR data, it is convenient and appropriate to do this in Log₂ by defining the term Log₂Ex:

\[
\text{Log₂Ex} = \text{LoD Ct} - \text{Ct [Gene]}
\]

If the value is negative, \( \text{Log₂Ex} = 0 \)
Log_{2}Ex represents the transcript level above background, expressed in log base 2. Conversion from a log scale to a linear scale can be accomplished by calculating $2^{\log_{2}Ex}$, which gives the fold change. The value of each sample is subtracted from the LoD. LoD is a gene-independent value.

LoD is set to convert C_t values to Log_{2} values (Log_{2}Ex), which represent transcript levels above background. The exact value to be used depends on the system, the assay, and the C_t calculation. It can be determined by statistical analysis (Livak et al., 2013) or simply by using the C_t value representing a single copy of a transcript, or representing the last cycle of PCR (ref Jaenisch paper). For BioMark™ and BioMark HD™ Systems, the recommended LoD value is 24, as suggested by Livak et al. 2013:

$$
\begin{align*}
\text{Log}_{2}Ex &= \text{LoD} - C_t \quad \text{if} \quad C_t \leq \text{LoD} \\
\text{Log}_{2}Ex &= 0 \quad \text{if} \quad C_t > \text{LoD}
\end{align*}
$$

The default LoD value is 24. For more information on this value, see K.J. Livak et al., Methods (2012), http://dx.doi.org/10.1016/j.ymeth.2012.10.004.

### How to Handle Missing Data

**Missing qPCR data.** For expression data retrieved from a BioMark qPCR experiment, a missing value is defined as data that does not have a C_t value of 999 and is a FAIL. This could be due to failed curves or melt curves within the BioMark™ Real-Time software. For more information, see the Fluidigm® Real Time Data Analysis User Guide.

**Note:** If the value of C_t is 999, then it is not a missing data point; and this means that there is no expression.

**Missing mRNA Seq data.** For expression data retrieved from an mRNA Seq experiment, the missing value is data point with an empty string.

**Note:** If the value of mRNAseq expression is "0", then it is not a missing data point; and this means that there is no expression.

**Conversion to the average value of the gene.** Missing data will be converted to a Log_{2}Ex value of -1. This will then be converted in subsequent calculations missing data will be given a value of the mean expression for the gene of that sample group if sample group information is provided. Thus, all values whose Log_{2}Ex is negative will be converted to the average value of that gene.
Example 1: Data for an LOD of 24 for qPCR

<table>
<thead>
<tr>
<th>CT</th>
<th>PASS/FAIL</th>
<th>Converts to (Ct))</th>
<th>Log2Ex</th>
<th>Mean value for gene of sample group</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Pass</td>
<td>20</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Fail</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>PASS</td>
<td>LOD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>999</td>
<td>Fail</td>
<td>LOD</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Example 2: Data for mRNA Seq data LOD =1

<table>
<thead>
<tr>
<th>Expression</th>
<th>Converts to (Linear)</th>
<th>Log2Ex</th>
<th>Mean value of gene of sample group</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>3.321928</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>LOD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BLANK</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>LOD</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Mathematical and Plotting Functions

The functions in this section return mathematical expressions or plots.

Principal Component Analysis (PCA)

PCA extracts meaningful information from single-cell data. The objective of running PCA is to account for the variation within a data set in as few variables as possible, while retaining most of the original variation information. This data reduction technique allows multi-dimensional data sets, such as those from qPCR and mRNA Seq experiments, to be simplified for plotting purposes and visual variance analysis.

PCA reduces the dimensionality of a data set by transforming it into a new set of uncorrelated variables, called principal components (PCs), with decreasing degrees of variability. The first PC explains most of the variation in the data set. Each successive PC in turn explains the next highest variance for the data, under the constraint that its relationship with the previous PC is zero.
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

The PCA algorithm in the SINGuLAR Analysis Toolset uses successive orthogonal transformations to convert data into a series of PCs that explain variance in the data. You can look at variance in single-cell samples and in the genes used in a single-cell data set. It further permits you to define the number of components, or axes, used to analyze the data set.

There are two options for generating PCA plots in the SINGuLAR Analysis Toolset:

- **With Automatic Analysis.** The autoAnalysis function uses two PCs by default and generates a PCA Score plot as a two-dimensional grid, with each axis representing one component and each point representing a single-cell sample. For an example, see A Study: Insights into Dimensional Data with PCA.

You can also generate a PCA Loading plot at this step.

- A PCA Loading plot in which each point represents an individual gene. For example:

![PCA Loading Plot](image)

- **Without Automatic Analysis.** You can run the PCA function manually with the PCA function instead of performing automatic analysis. In the manual scenario, in addition to automatically calculating and displaying the PCA Score plot and PCA Loading plot, you also generate a PCA Scree plot that displays the first ten PC scores with the height of each bar indicating the score. This lets you quickly determine the number of PCs to use.

For example, the relative heights of the bars indicate that the first two PCs in the PCA Scree plot contain most of the original data variance:
A scatter plot of PC scores can also provide interesting insights into data, as described in the case study here:
**A Study: Insights into Dimensional Data with PCA**

Basic statistics indicate that for any single gene a homogenous population can be characterized on the basis of 30 samples. Thus, if every subpopulation within a sample of single cells was represented by at least 30 cells, one could be reasonably confident that the experiment would robustly identify all subpopulations. The implication is that, to reliably identify a subpopulation that constitutes 10% of the total population, 300 cells must be examined.

In practice, subpopulations can be identified with fewer than 30 cells, depending on the cells and genes being analyzed. Guo et al. (2010) analyzed 159 single cells from 64-cell stage mouse embryos, assaying 48 genes in each cell. A scatter plot of PCA scores from the study is shown here.

![PCA scatter plot showing subpopulations.](image)

Guo et al. were able to clearly identify the epiblast (EPI) subpopulation, with only 17 cells in that subpopulation. They could do this because of the type of cells analyzed, the use of 48 genes, and the fact that those 48 genes revealed very distinct signatures between EPI, primitive endoderm (PE), and trophectoderm (TE) cells.
3D PCA Score Plot. The SINGuLAR Analysis Toolset also enables the display of an interactive 3D PCA scores that can be me rotated to enable the easiest views to differentiate the samples.

ANOVA

Analysis of variance (ANOVA) is predicated on the idea that variability in the quantity being measured (gene expression, for example) can be partitioned into a number of identifiable sources. ANOVA allows the single-cell researcher to examine whether the variability that is due to a single factor (1-way ANOVA) is statistically significant.

The SINGuLAR Analysis Toolset enables 1-way ANOVA on single-cell data. ANOVA can be executed by sample groups or hierarchical clusters to generate pairwise ANOVA summaries. Users can also select other ANOVA genes based on rank or p-values.

A Volcano plot helps to identify meaningful changes where there are high numbers of replicate single cells among different sample groups (or conditions). Volcano plots are visual representations of genes where there are large, statistically significant changes between two sample groups. These plots have a Log2 fold change for all genes plotted on the X-axis and the negative Log10 p-value plotted on the Y-axis. The data points for the highly significant genes (low p-values and high fold change) will appear on the top left and right corners of the plot.

Hierarchical Clustering (HC) Analysis

Clustering refers to grouping data in a way that data points in a group (or cluster) are more similar to each other than to data points in other groups (or clusters). The goal of HC is to build a binary tree or dendrogram of data in the form of a heatmap that successively merges similar groups of data points. HC algorithms connect data points to form clusters based on distance, with a cluster being described largely by the maximum distance needed to connect parts of the cluster.

Co-profiled genes are clustered together using the Pearson method, and samples are clustered together using the Euclidean method. In each case, the complete linkage method is then used to find similar clusters.

The SINGuLAR Analysis Toolset allows you to perform unbiased HC analyses on your data and visually present it as a heatmap with a dendrogram, as depicted in the figure below. For more information, see HC in the fluidigmSC Help system.
Correlation Analysis

Correlation analysis enables tracking of the linear association between two variables. Values of the correlation coefficient always lie between -1 and +1, with +1 indicating that two variables are perfectly related in a positive linear sense and 0 indicating no linear relationship. Since the correlation coefficient measures only the degree of linear association between two variables, not causality, any conclusions about a cause-and-effect relationship must be based on the judgment of the analyst.

In the SINGuLAR Analysis Toolset, new functions allow you to find co-profiling genes by providing target genes of interest. Co-profiling genes can be either correlated or anti-correlated, having correlation coefficient value greater than the given threshold. The correlation coefficient between two genes is calculated by the Pearson method.

Visualization of Expression Profiles

With the SINGuLAR Analysis Toolset, you can plot your data in multiple ways, including:

- Enhanced **scatter plots** display average expression values of pairwise sample groups.
- **Box plots** provide at-a-glance visualization of large data sets by genes or samples.
- **Density plots** allow visualization of expression across all genes for a sample or for a particular sample group. Applied to individual samples, the ensemble, or collective, expression can be plotted as well, to help compare how gene expression differs between samples and ensemble.
- **Violin plots** depict the probability density of the data at different values. The SINGuLAR Analysis Toolset permits Box and Violin plotting by samples and by genes. Violin plots are convenient to compare histograms for multiple genes.
The Outlier Identification Method

Outlier analysis is based on the assumption that samples (cells) of the same type also have a set of commonly-expressed genes. The outlier algorithm iteratively trims the low-expressing genes in an expression file until 95% of the genes that remain are expressed above the Limit of Detection (LoD) value that you set for half of the samples. The assumption is that the set of samples contains less than 50% outliers. This means that subsequent calculations will only include the half of the samples that have the highest expression for the trimmed gene list. The trimmed gene list represents genes that are present above the LoD in at least half the samples or the most evenly expressed genes—though they might not be the highest or lowest in their expression value.

A Study: Insights into Visualization with Violin Plots

Figure 3.4 shows a Violin plot from Guo et al. (2010) that compares 10 genes in 75 single cells derived from 16-cell stage mouse embryos:

The plot reveals that seven genes have unimodal distributions and three (Id2, Nanog, Sox2) have bimodal distributions. Whereas the unimodal distributions indicate no detectable variation other than intrinsic noise, bimodal distributions indicate that the Id2, Nanog, and Sox2 genes are differentially expressed in at least two subpopulations within these 75 cells.

The vertical position of each histogram indicates the relative expression level. Here, ActB has the highest expression level among these 10 genes.

It is also possible to see that transcripts can have distributions of varying widths, distribution being an indicator of variation. For example, Pou5f1 has a much narrower distribution (less variation on the Log2Ex axis) than Cdx2. This is because each gene has a characteristic transcriptional burst size, frequency, and decay rate.
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

For the 50% of the samples that remain, a distribution is calculated that represents their combined expression values for the gene list defined above. For this distribution, the median represents the 50\textsuperscript{th} percentile expression value for the set of data.

The **Outlier Threshold** is defined as the expression value that is the 15\textsuperscript{th} percentile for those samples, or the value at which 85\% of the gene expression values are above that line. Outliers are then identified as samples whose median expression values are less than the **Outlier Threshold**. Using the above method, the function automatically identifies outliers and displays a Box plot of expression values for each sample, with the outlier candidates labeled.

When you enter the function `identifyOutliers()`, the **FluidigmSC Outlier Identification** dialog is displayed. This dialog allows you to enter an LoD for the file type chosen, and select a file for upload. The results are returned as an interactive graph that allows you to identify normal and outlier candidates in an editable Box plot for a set of samples. You can then display a PCA Score plot of normal and outlier data points with sample names labeled to visualize each sample’s location in relation to each other. You have the option of saving the expression file with your outlier information to a **FluidigmSC Expression Object** file that includes all your original data.

Outlier identification can also be performed using hierarchical clustering. For examples, see Examples of Advanced HC Functions. For information on how to perform this analysis, see Performing Outlier Identification.

### The Automatic Analysis Method

When you enter the function `autoAnalysis()`, the **FluidigmSC Analysis** dialog is displayed that allows you to select one or more expression objects for automatic analysis. If you select multiple expression objects, they will be merged into one.

The `autoAnalysis` function runs PCA and displays a 2D PCA Score plot, ANOVA (if sample groups are provided), Hierarchical Clustering and with a heatmap, and generates Violin plots for the top-ranked genes. Results are expressed as Log\textsubscript{2}Ex values and exported to ".fso" files and saved in the working directory set for the session. In this file, the Log\textsubscript{2}Ex data is located in the "Log2Ex_data" frame. The gene names are displayed in Column A and sample names are displayed in Row 1 in the order in which they were exported from the Real-Time PCR Analysis software. To open this file, see Saving Data.

The `autoAnalysis` function uses two PCs by default and generates a PCA Score plot with two-dimensional grids, where each axis represents one component in which each data point represents a single-cell sample. Violin plots are also automatically displayed for the number of genes selected or up to 400, ranked by ANOVA scores (when sample groups are provided). Lastly, a Hierarchical Clustering (HC) heatmap is displayed with up to 400 genes.

This function saves the (1) expression object, (2) selected gene lists, (3) PCA Gene-Ranking Score, and the (4) ANOVA p-value (if sample group information is provided). Several objects are created that can be used for subsequent analysis: fldm_exp, fldm_pca, fldm_hc, and fldm_located_samples.
For information on how to perform this analysis, see Performing Automatic Analysis.

**IMPORTANT!**
- For mRNA Seq data, low expression genes (< 2*lod) will be removed before performing the analysis if gene_list is not provided.
- If a number of genes is defined, the dataset will be trimmed and all analyses in the session from this point forward will be performed against the expression data with the reduced gene list. In the workflow that follows, you can see how trimmed data affects the outcome of the results.

The `autoAnalysis` Workflow

For more information, see Performing Automatic Analysis.
Perform Outlier Identification

1. Launch R.
2. **Do this only one time per session:** Enter the function `library(fluidigmSC)` to load the package.
3. Enter the function `identifyOutliers()` to remove the outliers for your single-cell experiment.
4. Select a data type, and then upload one or more expression files. All selected files must be in the same folder.

   ![Expression File Selection](image)

   Your choices of data types are:

   - **fluidigmSC EXP.** Upload an existing *.fso file (created after performing the outlier identification procedure. For more information, see [Performing Outlier Identification](#). The LoD in the case will have been defined during the identifyOutliers() function and does not need to be defined again here.
   - **RNA-Seq.** (This is the default.) Upload a *.txt file that contains RNA Seq data. The default LOD value is 1.
   - **BioMark qPCR.** Upload an existing *.csv file from BioMark. The default LoD value is 24. For more information on qPCR, see [Preparing and Processing qPCR Data](#).

5. (Optional) You can change the default **LoD** value to suit your needs.

6. In the **Sample Annotation** section, select the way that you want sample group information to be annotated. You can add to the sample list with sample group annotation by sample prefix (Annotation by prefix "-" or Annotation by prefix "."). The group information must be BEFORE the sample name, and it must be separated from that sample name by either a hyphen ",-" or an underscore ",.", respectively. You can also annotate sample groups by providing an updated sample-list file with (From file) and browsing to select the file. These files are described in detail in [Preparing Sample/Gene Group Annotation Files](#).

   If you have multiple sample types in your analysis, it is important to annotate the samples so that outlier analysis is performed for each sample type.

   ![Sample Annotation](image)

7. Click **Analyze**. The SINGuLAR Analysis Toolset will determine outlier candidates and display a Box plot.

8. Click as many samples as you want in the **Box** plot to toggle between Normal (blue) and Outlier (red) for a given sample.
Perform Outlier Identification

For example, click the sample at the far right:

**Before (no outliers)**

![Before outlier identification](image)

**After (one outlier selected at the far right)**

![After outlier identification](image)

**Toggling Between Sample Types:** The data in the example above contains only one sample type (CT1). If your data contains multiple sample types, you will be able to toggle between **Box plots** if you click the sample type at the right. For example, for data not shown above, if you have sample types CT1 and CT2, click CT1 to change the display from CT2 to CT1:
9. In the Box plot, click PCA to display a PCA Score plot for normal and outlier data points for that particular sample group if provided, such as for the sample group CT1:

![PCA Score of Normal and Outlier](image)

10. When you are satisfied with the output, click OK to save to a fluidigmSC Expression Object (.fso) file. The file will include the original data as well as outlier information. You can use the *.fso file for analysis with the SINGuLAR Analysis Toolset.

11. Repeat if necessary for each dataset.

### Perform Automatic Analysis

The function autoAnalysis performs several tasks, as described in this section and in the section The Automatic Analysis Method.

### A Note About Using Functions Without Automatic Analysis

All the functions that are associated with the function AutoAnalysis can also be run by themselves. To find the functions that are available in the SINGuLAR Analysis Toolset, do any of the following:

- See Appendix A: The List of Functions for Gene Expression.
- See the section Getting R Help.

In addition, see Without Automatic Analysis in the section Principal Component Analysis (PCA).
Run the Automatic Analysis

Automated analysis in the SINGuLAR Analysis Toolset is a PCA-driven method.

1. Launch R.
2. Do this only one time per session: Enter the function `library(fluidigmSC)` to load the package.
3. Enter the function `autoAnalysis()` to display the FluidigmSC Analysis dialog that will retrieve experiment data along with the sample list and/or gene list supplied by you.
4. Select a data type, and then upload one or more expression files. All selected files must be in the same folder. To select more than one file, press the Ctrl key while clicking a file. (If multiple expression objects are selected, they will be merged into one expression object automatically.)

Your choices of data types are:
- **fluidigmSC EXP.** Upload an existing *.fso file (created after performing the outlier identification procedure. For more information, see Performing Outlier Identification.) The LoD in the case will have been defined during the identifyOutliers() function and does not need to be defined again here.
- **RNA-Seq.** Upload a *.txt file that contains RNA Seq data. The default LOD value is 1.
- **BioMark qPCR.** Upload an existing *.csv file from BioMark. The default LoD value is 24. For more information on qPCR, see Preparing and Processing qPCR Data.

5. (Optional) You can change the default LoD value to suit your needs.
6. In the Sample Annotation section, select the way that you want sample group information to be annotated. You can add to the sample list with sample group annotation by sample prefix (Annotation by prefix "-" or Annotation by prefix "."). The group information must be BEFORE the sample name, and it must be separated from that sample name by either a hyphen "-" or an underscore ".", respectively. You can also annotate sample groups by providing an updated sample-list file with (From file) and browsing to select the file. These files are described in detail in Preparing Sample/Gene Group Annotation Files:

7. In the Genes of Interest section, do any of the following:
   - Defined in the expression file(s)
   - From file (and then upload a file)
   - Find the top genes by this analysis. (The default value is 100.)
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

IMPORTANT!

- For the option Find the top genes by this analysis, autoAnalysis finds the top N PCA genes and trims expression data with those top genes. All subsequent analysis is performed using the edited gene list. If the number of genes is not specified, ANOVA is performed and the top 400 ranked genes is used for plotting.

- For mRNA Seq data, low expression genes (< 2*lod) will be removed before performing the analysis if gene_list is not provided.

8 In the Output Folder box, enter a folder that will hold the resulting EXP object file, which is now available for further analysis. For example:

   ![Output Folder](C:/myfolder/mydata)

9 Click Analyze. You see a set of plots for gene expression.

   As the analysis progresses, you see the following in the R console:

   ```
   Retrieving expression data ...
   Retrieving sample annotation file ...
   Finding differentially expressed genes by PCA Analysis ...
   Finding differentially expressed genes by ANOVA analysis ...
   Performing PCA analysis ...
   Saving analysis results to output folder ...
   Displaying analysis results ...
   ```

Note: When multiple expression files are selected, the SINGuLAR Analysis Toolset merges them into a single expression object.

Examine the Results

The results that you get with the autoAnalysis function give you a set of plots.

Violin Plot

If you supplied sample group information, the autoAnalysis function performs ANOVA and creates a Violin plot of differentially expressed genes that are ranked by p-values. Since the p-value measures the likelihood of obtaining the data if no real difference existed, a small p-value (typically <0.05) indicates the possibility of a biological process of interest happening randomly. For example:
If the gene number is not designated in Step 7, the top 400 genes according to ANOVA are plotted in 4 plots of 100 genes each. The number of a plot of four plots is in the title.

Hierarchical Clustering (HC) Plot

The autoAnalysis function performs unbiased Hierarchical Clustering for either N genes named in Step 7 or the 400 genes identified according to ANOVA p-values. For example:
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

### PCA Plots

The autoAnalysis function outputs a two-dimensional **PCA Score plot** by default. In the **PCA Score** plot, each point represents a single cell.

**Note:** Outlier identification and elimination before running PCA is highly recommended.

To assist with analysis of clusters, the SINGuLAR Analysis Toolset provides you with the ability to identify each point in a PCA plot.

You can identify individual samples in a PCA Score plot and individual genes in a PCA Loading plot. To identify samples or genes:

1. At the command line, enter one of the following:
   - To identify individual samples in a PCA Score plot, enter `displayPCAScore()`.
   - To identify individual genes in a PCA Score plot, enter `displayPCALoading()`.

   The PCA plot is displayed with several buttons along the right margin. For example:

   ![PCA Score Plot](image)

2. You can identify samples of interest in a PCA Score plot, and you can identify genes of interest in a PCA Loading plot. To identify the data points of interest:
   a. Do either of the following:
      - To display the ID of an individual point, click it with the **Point** option selected (the default). Repeat this step as needed to display other ID values one at a time.
Perform Automatic Analysis

- To display the ID for a group of points, click Circle. Use your cursor to encircle a set of points. Once a circle is formed, the plot displays each ID inside the circle.

  (To return the plot to its original state, click Clear.)

b To save the set of displayed ID values as a sample list, click Save to open a dialog that allows you to save the selected samples to a *.txt file.

c Repeat the steps above to save additional sets of sample lists as needed.

d When you are finished, click Done to return to the main R window.

The R window displays the contents of your sample list. The values are ranked by score.

3 You can save the plot, copy it to the clipboard, or print it. Do any of the following:

- To save the PCA Score plot as an image file, choose File > Save as, and then click any of the following options:
  - Metafile. Opens a dialog that allows you to save the plot as an enhanced metafile with an *.emf filename extension.
  - Postscript. Opens a dialog that allows you to save the plot as an encapsulated postscript file with an *.eps filename extension.
  - PDF. Opens a dialog that allows you to save the plot as a PDF file with a *.pdf filename extension.
  - PNG. Opens a dialog that allows you to save the plot as an image file with a *.png filename extension.
  - BMP. Opens a dialog that allows you to save the plot as an image file with a *.bmp filename extension.
  - TIFF. Opens a dialog that allows you to save the plot as an image file with a *.tiff filename extension.
  - Jpeg. After you click a quality value of 50%, 75%, or 100%, opens a dialog that allows you to save the plot as an image file with a *.jpg, or *.jpeg filename extension.

  [Alternatively, right-click the plot, and then click Save as metafile or Save as PDF. This action opens a dialog that allows you to save the plot as an enhanced metafile (*.emf) or as a PDF (*.pdf). A PDF file gives you the highest resolution.]

- To copy the PCA Score plot to the clipboard, do either of the following:
  - Choose File > Copy to the clipboard, and then click either as a Bitmap or as a Metafile. (Alternatively, right-click the plot and then click either Copy as metafile or Copy as bitmap.) For example:
The Selected Gene List Output File

The ANOVA output file from the `autoAnalysis` function is saved in the user-defined directory and is named `selected_gene_list (auto_analysis).txt` by default. This tab-delimited file includes gene names in Column 1 and p-values in Column 2 for all groups and average expression values per each sample group. The p-values for pairwise groupings, scores, and ANOVA ranks are also tabulated.

In this example of a selected gene list output file, the column heading `Score(PC1-3)` denotes values of PC1, PC2, and PC3.

<table>
<thead>
<tr>
<th>GeneID</th>
<th>GroupID</th>
<th>GroupID.x</th>
<th>GroupID.y</th>
<th>ANova.pValue</th>
<th>CT2_vs_CT1.pValue</th>
<th>CT1_Average</th>
<th>CT2_Average</th>
<th>GroupID.x</th>
<th>GroupID.y</th>
<th>Score(PC1-3)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>G05635</td>
<td>Untitled</td>
<td>3.6E-50</td>
<td>0</td>
<td>8.474375529</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.04807372</td>
<td>12</td>
</tr>
<tr>
<td>G09498</td>
<td>Untitled</td>
<td>4.2E-41</td>
<td>0</td>
<td>8.09041975</td>
<td>0.2531385963</td>
<td>0.042255688</td>
<td>31</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.036304633</td>
<td>78</td>
</tr>
<tr>
<td>G02355</td>
<td>Untitled</td>
<td>5.23E-38</td>
<td>0</td>
<td>6.575623277</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.034213418</td>
<td>100</td>
</tr>
<tr>
<td>G18081</td>
<td>Untitled</td>
<td>4.39E-35</td>
<td>0</td>
<td>6.038817499</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.038623455</td>
<td>53</td>
</tr>
<tr>
<td>G22096</td>
<td>Untitled</td>
<td>4.99E-30</td>
<td>0</td>
<td>6.92026124</td>
<td>0.09195883</td>
<td>0.046812305</td>
<td>53</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.025363614</td>
<td>4</td>
</tr>
<tr>
<td>G20354</td>
<td>Untitled</td>
<td>1.71E-27</td>
<td>0</td>
<td>9.491456385</td>
<td>0.850216314</td>
<td>0.046531589</td>
<td>16</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.052363132</td>
<td>8</td>
</tr>
<tr>
<td>G04839</td>
<td>Untitled</td>
<td>2.39E-24</td>
<td>0</td>
<td>1.078689355</td>
<td>8.294444679</td>
<td>0.046531589</td>
<td>16</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.046531589</td>
<td>16</td>
</tr>
<tr>
<td>G17357</td>
<td>Untitled</td>
<td>2.57E-20</td>
<td>0</td>
<td>8.350182257</td>
<td>0.811629315</td>
<td>0.047718274</td>
<td>58</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.037325919</td>
<td>67</td>
</tr>
<tr>
<td>G03991</td>
<td>Untitled</td>
<td>2.96E-19</td>
<td>0</td>
<td>7.251667186</td>
<td>9.597781138</td>
<td>0.037325919</td>
<td>67</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.037325919</td>
<td>67</td>
</tr>
<tr>
<td>G21774</td>
<td>Untitled</td>
<td>5.70E-18</td>
<td>0</td>
<td>8.5749347853</td>
<td>1.550554394</td>
<td>0.016358588</td>
<td>76</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.016358588</td>
<td>76</td>
</tr>
<tr>
<td>G01913</td>
<td>Untitled</td>
<td>1.15E-17</td>
<td>0</td>
<td>4.597095212</td>
<td>6.884561125</td>
<td>0.047611931</td>
<td>14</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.047611931</td>
<td>14</td>
</tr>
<tr>
<td>G11849</td>
<td>Untitled</td>
<td>3.12E-17</td>
<td>0</td>
<td>9.606908888</td>
<td>1.4852387897</td>
<td>0.040669499</td>
<td>39</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.040669499</td>
<td>39</td>
</tr>
<tr>
<td>G06824</td>
<td>Untitled</td>
<td>1.03E-16</td>
<td>0</td>
<td>5.937647583</td>
<td>0</td>
<td>0.035406582</td>
<td>87</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.035406582</td>
<td>87</td>
</tr>
<tr>
<td>G13554</td>
<td>Untitled</td>
<td>2.60E-16</td>
<td>0</td>
<td>7.936997739</td>
<td>0.242622464</td>
<td>0.04182588</td>
<td>32</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.04182588</td>
<td>32</td>
</tr>
<tr>
<td>G08381</td>
<td>Untitled</td>
<td>2.82E-11</td>
<td>2.75E-11</td>
<td>10.3751501</td>
<td>3.69265897</td>
<td>0.037288579</td>
<td>64</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.037288579</td>
<td>64</td>
</tr>
<tr>
<td>G09959</td>
<td>Untitled</td>
<td>6.06E-10</td>
<td>6.06E-10</td>
<td>0.137613626</td>
<td>5.209354465</td>
<td>0.066764286</td>
<td>2</td>
<td>Untitled</td>
<td>Untitled</td>
<td>0.018849114</td>
<td>6</td>
</tr>
</tbody>
</table>
Perform Automatic Analysis

**Note:** When this function is performed, the *selected_gene_list (auto_analysis).txt* file containing the genes and samples of interest is generated automatically. However, the file is not displayed in the R window actively after the commands are entered. You do not need to open this file, but you can open it in any Text editor or within Excel as a tab-delimited file.

## Advanced Functions

The SINGuLAR Analysis Toolset includes a variety of advanced functions. The remainder of this chapter provides you with examples on the uses of some of these. To view the complete list of functions for Gene Expression, see Appendix A: The List of Functions for Gene Expression.

### Reading Experimental Data

When analysis is initiated with identification of outliers and auto analysis, an *.fso expression object file is created and saved in the specific output directory. This file can be called for further analysis.

If you have not performed either of the `identifyOutliers` or `autoAnalysis` functions in your current session but you have created and saved an *.fso file previously, enter the object and function as follows: `exp <- readExpObject()` to read an existing *.fso expression object. This function displays a dialog prompting you to select one or more files. All selected files must be in the same folder. To select more than one file, press the Ctrl key while clicking a file.

An identical protocol applies to the function `readLinearExp` for mRNA Seq with a default LoD of 1 and `readCtExp` for qPCR with a default LoD of 5.

**To read experimental objects**

```r
exp <- readExpObject()
```

**To read mRNA Seq data while changing the LoD value to 6**

```r
exp <- readLinearExp(lod=6)
```

### Annotating Samples

If sample group or gene group information was not provided during your use of the function `autoAnalysis`—or `autoAnalysis` was not performed—you can update the sample and genes from a file to include group information. In this case, after expression data is read, annotate samples using the function `updateSampleListFromFile`. This can also be done for each individual sample by name with the function `updateSampleListFromName`.

**To update a sample list (sample name plus group ID) with a file**

The sample file must be a .txt file with a minimum of two columns with headers as SampleID and GroupID, respectively.
exp <- updateSampleListFromFile (exp)

**Annotating Genes**

The release of the SINGuLAR Analysis Toolset includes the addition of a GroupID for gene list files.

**To update genes with group information**

The sample file must be a *.txt file with a minimum of two columns with headers as GeneID and GroupID, respectively.

exp <- updateGeneListFromFile(exp)

For more information, see [Preparing Sample/Gene Group Annotation Files](#).

**Setting Custom Colors and Symbols**

The SINGuLAR Analysis Toolset assigns a default color and symbol to each sample and gene group for visualization of samples and genes in each expression (EXP) object.

**To customize the colors and symbols for sample groups of given Exp object**

exp<-setSampleGroupColorAndSymbols(exp)

You see a dialog that contains colors and symbols. These options are as follows:

- ◯ △ ⊗ ◊ ▽ ◦ ▼ ◎ ◌ ◐ △ ◆ ◇ ◈ ◉ ◊ ◌ ◉ ◆ ◎ ◇ ▽ ◥ ◗ ◘

This function displays a dialog that allows you to select a color and symbol for each sample group. This is an interactive display that allow you to click on symbol and color for each member of the group. When you are done, click OK. Subsequent plots in your current session (including PCA and HC) will contain this gene group information as well as the colors selected automatically.

For more information on the functions for customizing colors and symbols, see [Appendix A: The List of Functions for Gene Expression](#).

**Saving Data**

You can save any fluidigmSC data object (EXP, HC, ANOVA, and PCA), a gene list, and a sample list to a file for future use. For an explanation about how to create gene and sample lists, see [Select and Trim Genes](#).
To save the EXP object to files for future use

saveData (exp)

To save a gene or sample list to a *.txt file

saveData (example_gene_list)

To save a plot as an image file

Choose File > Save as, and then click the type of file you want. Your choices are: Metafile, Postscript, PDF, Png, Bmp, TIFF, and Jpeg.

Working with the Analysis Output in Excel

The Excel macro fluidigmSCObjectToExcel (100-6988) with an *.xlsm filename extension is a utility to open an *.fso object and convert each data member or frame into an individual worksheet. Each worksheet is a data frame. For more information on the contents of each data frame, see Appendix C: Contents of the EXP Object.

Selecting and Trimming Genes

A typical mRNA Seq data set contains well over 20,000 genes, many of which are either not expressed or have low estimated expression values. With the analyzeGeneDetection() function, you can check how many genes are detectable above the Limit of Detection (LoD).

Subsequently, low-expressing genes can be removed using the function removeGenesByLinearExp(). For a qPCR experiment, the equivalent function is removeGenesByCtExp().

The analyzeGeneDetection() function displays the number of genes in the sample with expression values above the threshold. For example:
To remove genes by Linear Exp you must first set a threshold below which genes will be identified and subsequently removed. The threshold can be any value that equates to the number of reads detected (mRNAseq) or to the Ct value (qPCR). The default threshold is 1.

To display the number of detected genes for samples above a threshold of 1

```r
analyzeGeneDetection(exp)
```

To remove all genes with expression values below twice the LoD

In the function `removeGenesByLinearExp`, the argument `linear_threshold` has no default value. For example, to remove all genes with expression values below twice the LoD (such as an LoD of 1):

```r
1  analyzeGeneDetection(exp, threshold)
2  exp <- removeGenesByLinearExp (exp, linear_threshold=2)
```

Advanced Plotting Functions

The functions in this section are used either after or without the autoAnalysis function, and plots are returned that can include interactivity.

Examples of Violin Plots and Pairwise Scatter Plots

To generate a violin plot for each gene from a gene list

You can generate Violin plots for each gene from a gene list or a gene list file, and you can generate them with a given number of plots per page. One example is as follows:
violinPlotByGenes (exp, gene_list=HC_gene_list)

**To display pairwise Scatter Plots between sample groups**

This function displays histograms of the average gene expression values for each sample group and a scatter plot and the Correlation Coefficient between any two sample groups.

pairwiseScatterPlotBetweenSampleGroups (exp)

For example, Log2EX values are plotted for sample groups CT1 and CT2. In the lower left quadrant, each data point is a single gene where the x-axis contains CT1 values and the y-axis contains CT2 values. The output in the upper right quadrant is the Correlation Coefficient for CT1 and CT2.

Examples of Advanced PCA Functions

The SINGuLAR Analysis Toolset performs a variety of PCA functions. For more information, see Appendix A: The List of Functions for Gene Expression.

To analyze with PCA using a previously generated expression object

```r
pca <- PCA (exp)
```

This function displays three PCA plots automatically:
Chapter 3: The SINGuLAR Analysis Toolset for Gene Expression

- **Scree Plot.** Displays the first ten PC scores with the height of each bar indicating the variance. This allows you visually determine the number of PCs to use.

- **Score Plot.** Displays individual samples, as described in "Without Automatic Analysis" in the section Principal Component Analysis (PCA) and in "PCA Plots" in the section Examine the Results. If sample groups are supplied, the chosen symbol and color will be used to represent those sample groups.

- **Loading Plot.** Displays individual genes. If gene groups are supplied, the chosen symbol and color will be used to represent those gene groups. An example of a PCA Loading plot that includes gene group colors is as follows:

![PCA Loading Plot](image)

To highlight and label individual points on PCA plots

The functions `displayPCAScore(pca)` for samples and `displayPCALoading(pca)` for genes can be used to highlight and label individual sample data points and gene data points, respectively. The default is `(fldm_pca)` instead of `(pca)`, which is the output from autoAnalysis. Thus, to use these functions on the data calculated in the previous procedure, edit the pca value in the parentheses (not shown here):

```plaintext
displayPCAScore(pca)
displayPCALoading(pca)
```
To apply a previously calculated PCA model to expression data of new samples

When a pca object is created for a given exp object (as in the previous example), this PCA model can then be applied to a completely new exp by calling up the `new exp`.

To call the new exp for the purpose of applying the PCA model to it

```r
exp <- readExpObject()
```

To apply the previously calculated PCA model to the now active exp object

```r
applyPCA(exp, pca, display_plots = TRUE, include_outliers = FALSE)
```

To display a 3D PCA Score Plot

You can display a rotatable 3D PCA score plot with an optional graphical user interface (GUI) for locating samples of interest. To display the PCA score for principal components 1, 2 and 3 with explicit definition of the PCA object:

```r
display3DPCAScore(pca = pca, x_axis=1, y_axis=2, z_axis=3, locate=TRUE)
```

The default is set to display PCA plots. When set to FALSE, only the new PCA analysis object will be returned.

To rotate the 3D PCA Score plot, click any area of the plot and drag the plot in the any direction. For example:

In the 3D PCA Score plot, you can:

- **Select.** Click-and-drag a box on the plot to select interested samples.
- **Clear.** Returns the plot to its original state.
- **Save Selection.** Save the selected samples to a file.
Examples of Advanced ANOVA Functions

The SINGuLAR Analysis Toolset performs ANOVA based on either sample group annotation from user-defined annotation or on identified sample groups. Genes can be obtained according to P-value for all groups or any two groups (if you have more than two sample groups). For more information, see Appendix A: The List of Functions for Gene Expression.

To generate an ANOVA pairwise summary plot from an existing *.fso file

This function displays the number of significant genes:

\[ \text{anova} \leftarrow \text{ANOVA} \left( \text{exp} \right) \]

For example:

![ANOVA Pairwise Summary](image)

To only display the number of significantly expressed genes per sample-group pair

After you create the anova object, you can display the number of genes that are significantly differently expressed (less than a given pvalue_threshold) between any two sample groups or with defined p-values:
Perform Automatic Analysis

Less than a given threshold (such as the p-value threshold less than 0.02):

```
pairwiseANOVASummary (anova, pvalue_threshold = 0.02)
```

To find the top genes (most significantly expressed in the sample data) by ANOVA

Use the function `getTopANOVAGenes` to perform ANOVA to get the top \( N \) genes and return a list called `anova_gene_list`, sorted by overall p-value of ANOVA. The returned list of \( N \) genes contains their names and p-values. You can also change the p-value threshold of this function.

If there are less genes in the dataset than what you specified, then all genes present with an overall p-values of ANOVA will be returned.

**Example: Get the top 200 genes with p-values less than 0.05**

```
anova_gene_list <- getTopANOVAGenes (anova, top_gene_num=200, pvalue_threshold = 0.05)
```

To save this list, see [Saving Data](#).

To sort the most significant genes by p-value for a pair of sample groups

If more than 2 sample groups are named in the exp file and you want to specify two of these groups, change the function by setting `sample_group1` and `sample_group2` equal to the names of the two sample groups you want to analyze. This will then return a list that is sorted by p-value of a T test between the two sample groups.

For example:

```
anova_gene_list <- getTopANOVAGenes (anova, top_gene_num = 100, sample_group1 = "CT1", sample_group2 = "CT2")
```

To display sorted p-values for all sample groups or a pair of sample groups

You can display a graph with p-Values on the y-axis and the number of genes with p-values less than or equal to that number (cumulative) on the x-axis—and sorted according to p-values (small to large).

To plot all genes, set the maximum number of ranked genes (`top_gene_num`) to a negative value. (The default is -1).

To enter a title for this plot, replace the term "title" with your own title (such as "ANOVA P-Values"), and include the quotes.

```
displayANOVAPValues(anova, top_gene_num = -1, pvalue_threshold = 1, sample_group1 = FALSE, sample_group2 = FALSE, "title")
```
To sort genes by overall p-value of ANOVA

```r
sample_group1 = FALSE, sample_group2 = FALSE
```

To sort genes by p-value of T Test in two sample groups, CT1 and CT2

```r
sample_group1 = "CT1", sample_group2 = "CT2"
```

To identify differentially expressed genes via a threshold value in a Volcano plot

Genes are taken to be differentially expressed if the expression under one condition is over a specific amount greater or less than that under the other condition, and hence you see a large mean expression ratio between the two. With the `foldChangeAnalysis` function, you can find statistically significant genes with mean expression ratios between two given sample groups that are equal to or greater than a given threshold.

This example performs ANOVA analysis and displays the fold-change on a Volcano plot between a pair of sample groups (CT1 and CT2) for each gene where the fold-change threshold is the default value of 2:

```r
anova <- ANOVA(exp)

volcano_gene_list <- foldChangeAnalysis(anova, "CT1", "CT2",
foldchange_threshold = 2, pvalue_threshold = 0.05, display_plot = TRUE, locate = TRUE)
```

For example:
If `locate = FALSE`, the function will return the list of genes as an object called `volcano_gene_list` that can be saved as a *.txt text file.

If `locate = TRUE`, the function will allow the user to select points from the displayed graph to save as `volcano_gene_list`.

In the Volcano plot, you can:
- **Circle.** Draw a circle by connecting dots on the plot. Once a circle is formed, the plot displays each ID inside the circle.
- **Point.** Click a data point to display its ID.
- **Clear.** Returns the plot to its original state.
- **Save.** Save selected data points to a file.
- **Done.** Indicate that you are done locating samples of interest.

### Examples of Correlation Functions

This section provides examples of correlation functions. For more information, see Appendix A: The List of Functions for Gene Expression.
To find co-profiling genes (correlated or anti-correlated) with defined correlation threshold

With an EXP object and a gene list file, you can find all genes that are co-expressed and with correlation coefficient values greater than the defined threshold. The gene list file in this case is saved as gene_list from another output. For example: the selected_gene_list (auto_analysis).txt file is automatically saved when you run the function autoAnalysis. The default threshold for this function is 0.5.

corr <- findCorrGenesFromFile(exp, gene_list_file = TRUE, sample_group = "all", corr_threshold = 0.5)

This corr object can be saved and opened according to the Saving Data section.

To find other genes co-expressed with target genes

The function displayCorrGenes allows you to view genes that are co-expressed with a given gene of interest across the defined samples. This function uses the corr object created in the previous calculation and, as such, the sample group defined above is used. You can select the gene of interest by name and also change the correlation threshold if desired. The corr_pattern is a flag to indicate if genes are co-expressed or anti-co-expressed with target genes. The default is set as "positive" but can be changed to "negative" for anti-co-expressed.

displayCorrGenes(corr, query_gene_name="G01596", corr_threshold = 0.5, corr_pattern = "positive", "title")

In this example, there is one gene correlated with Gene G01596 in the two sample groups:
Examples of Advanced HC Functions

This section provides examples of HC functions. For more information, see Appendix A: The List of Functions for Gene Expression.

To analyze with HC

If you have a previously generated expression object, read it in, and then enter:

\[
\text{hc} \leftarrow \text{HC} \left(\text{exp}\right)
\]
If gene groups have been identified they will be displayed on the plot.

**To select your own color scheme for hierarchical clustering**

Enter the longer HC function:

```r
hc <- HC (exp, color_scheme= "green_orange_black", display="global_z_score", display_sample_names=TRUE, display_gene_names=TRUE)
```

Colors can be be selected by setting `color_scheme = ""` This will list color choices. The acceptable colors are two or three of the following, separated by ",":

- red
- green
- blue
- black
- white
- pink
- gold
- yellow
- grey

**To change the type of values displayed**

When you use the display function, this action does not change the clustering results. The heatmap of "global_z_score" and "expression" best represents the sample similarity while the heatmap of "gene_z_score" best represents the gene similarity.
The “global_z_score” display option normalizes the expression value with the global mean and the global standard deviation.

The “gene_z_score” display option normalizes the expression value per gene with the mean and standard deviation for each gene.

The “expression” display option shows the expression value without normalization.

To apply an existing HC dendrogram to a new HC from the sample side

An existing dendrogram can be from either the sample side or from gene side of the given HC. This allows you to directly compare two EXP objects without changing their sample or gene order in the two datasets.

From the Sample Side

Goal. Identify a set of unknown samples from a control set.

Strategy. You have a control set of samples and you want to calculate an HC for all genes based on those samples of known identity. You then want to calculate the clustering of a set of unknown samples, based on those genes to potentially identify them as the different control subtypes.

Steps. Get an EXP file containing the group information for the known/control samples. Get a second EXP containing simply the unknown samples. If these samples are all contained within one EXP file (such as the case where you previously ran identifyOutliers), the file will need to be split.

Example. An existing EXP contains sample groups ControlA, ControlB, ControlC, and Unknown. For illustration purposes, create two new EXP objects: exp_control containing only the control samples and exp_unknown containing the unknown samples.

1 Remove the sample group Unknown from the existing EXP object:
   exp_control <-removeSampleGroup(exp, "Unknown")

2 Create a new HC from the EXP of only control samples without the Unknown:
   hc_exp_control <-HC(exp_control)

3 Define the new EXP object (which contains only sample group Unknown):
   exp_unknown <-retainSampleGroup(exp, "Unknown")

4 Apply the existing HC (hc_exp_control) to new HC (exp_unknown) such that the new HC object now contains Sample Groups ControlA, ControlB, ControlC, and Unknown:
   apply_HC_samples<-applyHC(hc_exp_control, exp_unknown)

By not modifying the original exp object, this can be called for comparison or further analysis without reloading.
This can also be done for gene groups with the functions `removeGeneGroup()` and `retainGeneGroup()`.

**To interactively identify gene/sample clusters from HC and return the identified cluster as a list**

The functions `identifySampleClusters` and `identifyGeneClusters` enable you to select clusters by clicking on the dendrogram and selecting the region of your choice. The data for each selected cluster is returned as either a sample or gene list with the cluster IDs.

For a gene list:

```r
hc_gene_list <- identifyGeneClusters (hc)
```

For a sample list:

```r
hc_sample_list <- identifySampleClusters (hc)
```

To accept your selected regions of interest, click **OK**. To return to the previous screen, click **Cancel**.

The functions `getSampleClusterByThreshold(hc)` and `getGeneClusterByThreshold(hc)` enable you to get clusters by entering a threshold value. By default, you see a dendrogram that allows you to set a threshold. This is then typed into the command line as a number. When you press the **Enter** key, you see your threshold on the graph. For example:
You also see a table displayed on the command line of the R console. For example:

```
R Console
> getSampleClusterByThreshold(hc)
Please review the Sample Cluster Plot, and enter the desired threshold:

Distance Threshold:  0.38

<table>
<thead>
<tr>
<th>SampleID</th>
<th>GroupID</th>
<th>ClusterDistance</th>
<th>SampleAnnotation</th>
<th>GroupID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT1_04-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT2_32-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_04-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_55-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_05-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_02-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_31-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_10-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_71-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_14-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2_20-1</td>
<td>SC_3_2</td>
<td>0.35135793052343</td>
<td>CT2</td>
<td></td>
</tr>
</tbody>
</table>
```

You can save this information as a list by first creating an object, and then using the function `Saving Data`. To create the object:

```
threshold_list<- getSampleClusterByThreshold(hc)
```
Outliers by HC Analysis

Aside from automatic outlier analysis using the function `identifyOutliers()`, you can determine outliers by HC analysis on an existing hc object created in an exp object. In the latter case, outliers can be selected using the function `identifySampleClusters(hc)`.

To select outliers by HC analysis

```r
outlier_list <- identifySampleClusters(hc)
```

To remove the outliers from the experiment

```r
hc_outlier_exp <- addOutlierFromList(exp, outlier_list)
```

This newly updated expression object will no longer contain the outliers that have been identified.
This chapter provides a high-level view of preparation and processing of data and the types of analysis and visualization that the SINGuLAR Analysis Toolset can perform on your variant and mutation analysis data.

The types of data that can be analyzed for variants in the SINGuLAR Analysis Toolset are singleton SNPs, insertions, and deletions. If a SNP contains more than one alternate allele or variant identified in the VC object, then these data points will not be analyzed in this current version of the SINGuLAR Analysis Toolset. Instead, these data points will be removed from the analyzed data and stored in an alternate table in the VC object labeled as mult_allele_data.

In the mult_allele_data file, you can see the data that has been excluded from variant and mutation analysis. For more information, see Data Frame: mult_allele_data in Appendix D: Contents of the VC Object.
Prepare and Process DNA Seq Data

This section shows you how to prepare and process data before analysis with the SINGuLAR Analysis Toolset.
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

The WES Workflow
Mutation Analysis Strategy

A quality control check on raw sequencing data produces better mapping results. It helps identify and fix potential issues in FASTQ files, including sequencing artifacts, genetic contaminants (primers, vectors, and adaptors), and low-quality reads. To perform this quality control:

1. Review quality scores across all bases using the FastQC toolkit (downloadable from http://www.bioinformatics.babraham.ac.uk/projects/download.html#fastqc) or FASTX toolkit (downloadable from the http://hannonlab.cshl.edu/fastx_toolkit/).
2. If you notice issues, trim the data by removing aberrant bases or reads to generate a high quality FASTQ file.

Mapping & Variant Calling

Currently, the SINGuLAR Analysis Toolset supports VCF data that is generated from diverse variant callers, such as GATK, MuTect, and VarScan. Fluidigm recommends that your data contains GT, DP, GQ, AD, and Allele Frequency in the genotype fields of VCF, but allows DP, GQ, AD, and AF to be potentially missed. The supported variant annotation package is SnpEff.

For mapping sequences to a reference genome, the recommended alignment tools are:


For variant calling, the recommended tool is:

- GATK: [http://www.broadinstitute.org/gatk/download](http://www.broadinstitute.org/gatk/download)

Other variant callers:


For somatic point mutation detection, the recommended tool is:


For variant or mutation annotation, the recommended package is:


In terms of the analysis workflow, we suggest that you follow the workflow GATK Best Practices recommended to do the analysis, which includes three main sections:

- Data pre-processing: from raw sequence reads (FASTQ files) to analysis-ready reads (BAM files)
- Variant discovery: from reads (BAM files) to variants (VCF files)
- Suggested preliminary analyses

For single-cell variant analysis, we suggest using multi-sample variant calling to simultaneously call variants across sets of single cells.

For filters used in SNP calling, as starting point we suggest using the GATK recommended parameters listed below:


```
java -jar GenomeAnalysisTK.jar \
    -T VariantFiltration \ 
    -R reference.fa \ 
    -V raw_snps.vcf \ 
    --filterExpression "QD < 2.0 || FS > 60.0 || MQ < 40.0 || HaplotypeScore > 13.5 || MappingQualityRankSum < -12.5 || ReadPosRankSum < -8.0" \ 
    --filterName "my_snp_filter" \ 
    -o filtered_snps.vcf
```

If you are conducting targeted variant calling at single-cell resolution, we suggest adding two more parameters of AF > 0.1 and DP > 200.

Explanation: For WGA-AA, the sum DP (depth) threshold of samples is set to 200 because the depth is normally quite high. For variant calls at single-cell resolution, there is no rare mutation. Its allele frequency should be greater than 10%.

```
java -jar GenomeAnalysisTK.jar \
```
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

Getting a variant input file

The VCF files created by SnpEff are the input files used for variant analysis by the SINGuLAR Analysis Toolset. In each SnpEff VCF file, there are eight fixed fields including an INFO field, which varies widely in composition. The SINGuLAR Analysis Toolset reports the recognizable information to the VC object from the SnpEff VCF file and also uses the Variant Group information.

If any other application is used to generate VCF input files for the SINGuLAR Analysis Toolset, you might see fields that contain the term "UNDEFINED" within the VC object. For more information on VCF files, see:

http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41

Getting the Sample and Variant Group Annotation Files

Sample GroupID. Before using the SINGuLAR Analysis Toolset, generate a sample list file as a tab-delimited *.txt file with the samples from all the VCF files and their conditions. The first two columns headers are labeled as SampleID (Column A) and GroupID (Column B), and additional data can be included in the columns that follow. The rows are populated with sample names (Column A) and their corresponding sample group IDs (Column B). For example:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SampleID</td>
<td>GroupID</td>
</tr>
<tr>
<td>2</td>
<td>CT1-gDNA_S01</td>
<td>CT1-gDNA</td>
</tr>
<tr>
<td>3</td>
<td>CT1-gDNA_S02</td>
<td>CT1-gDNA</td>
</tr>
<tr>
<td>4</td>
<td>CT1-gDNA_S03</td>
<td>CT1-gDNA</td>
</tr>
<tr>
<td>5</td>
<td>CT1-SC_1_S01</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>6</td>
<td>CT1-SC_1_S02</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>7</td>
<td>CT1-SC_1_S03</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>8</td>
<td>CT1-SC_1_S04</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>9</td>
<td>CT1-SC_1_S05</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>10</td>
<td>CT1-SC_1_S06</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>11</td>
<td>CT1-SC_1_S07</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>12</td>
<td>CT1-SC_1_S08</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>13</td>
<td>CT1-SC_1_S09</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>14</td>
<td>CT1-SC_1_S10</td>
<td>CT1-SC</td>
</tr>
</tbody>
</table>
Variant **GroupID**. You can group variants in advance of using the SINGuLAR Analysis Toolset, as needed. Other options for GroupID with variants are discussed in Annotate Variants. The Variant Group file is also a tab-delimited *.txt file with at least the first two columns headers labeled as **VariantID** (Column A) and **GroupID** (Column B). The rows are populated with variant names (Column A) and their corresponding variant group ID (Column B). Each variant name can contain an rs ID or an UID with the combination of chrome, pos, and ref, and alt.

For example:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VariantID</td>
<td>GroupID</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>rs2072454</td>
<td>VC_1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>rs4947986</td>
<td>VC_1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7:55240708_T_TG</td>
<td>VC_1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7:55241604_T_TC</td>
<td>VC_1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>rs2293347</td>
<td>VC_1</td>
<td></td>
</tr>
</tbody>
</table>

**IMPORTANT!** When you open a tab-delimited *.txt file from within Excel, each cell receives the “General” format, and some gene names are converted to a date. To avoid this conversion to a date, assign the Gene ID column as “Text” in the import wizard.

**Preparing the Input Files**

Before you begin analysis:

1. Ensure that each cell line (or condition) has its own variant data file. Although this is not absolutely necessary, it is highly recommended.
2. Move all variant data files that you want to analyze into a common directory.
Workflow for Variant and Mutation Analyses with the SINGuLAR Analysis Toolset

1. Read VCF data
2. Add sample annotation to sample list via sample_list file or sample name prefixes.
3. View distributions of variant call features (DP, GQ, AF, etc.) and determine feature thresholds for No-Calls.
4. Choose analysis method and their thresholds to filter variants.
5. Conduct HC analysis.
6. Conduct a Fisher’s Exact Test if there is sample group annotation.
7. Save the VC object as an *.vso file and return the updated VC object.

Advanced Variant Analysis
- No-Call analysis
- Apply custom filters to update the VC object

Write updated VCF file

Summary Report for Variant Calls

HC Analysis for Filtered Variant Calls
And find sample and variant clusters

Fisher’s Exact Test for Case/Control Association Analysis
“Between” sample groups, and display Manhattan plot

Report Evaluation Comparison of Variant Calls in Samples & Sample Groups
Provide control calls (ADO, LDO, Non-Reference Sensitivity, & Non-Reference Discrepancy)
How to Handle Missing Data and Conversion to Numerical Values

During analysis, variants can be called in different ways, depending on the application and their associated parameters.

**Options for Individual Samples.** For variants called using samples individually at any given position, sample reference (non-variant) calls might be converted into "missing data" due to cases where there is no variant at a given position for that sample. Thus the data would not be stored. In this case, there are only three types of variant calls: missing data, variant 1, and variant 2. For example if A/A is the reference, then the variant types are missing data [non-variant(A/A) and missing data], variant 1 (A/B), and variant 2 (B/B).

**Options for Sample Groups.** For variants called using samples as a group, data might be produced that has both "missing data" and "reference" (non-variant) calls. In this case, there are four options: missing data, non-variant (A/A), variant 1 (A/B), and variant 2 (B/B).

**Additional Option for QC Metrics Below Thresholds.** During data analysis for specific functions like HC, another option presents itself where data that has QC metrics below a defined threshold will be converted to "No-Call" data for that calculation.

**Categories of Variant Calls in Samples.** Each genotyping call of samples is assigned one of following categories:

- 0 - missing data
- 1 - A/A (if applicable)
- 2 - A/B
- 3 - B/B
- 4 - No-CALL

**Mathematical and Plotting Functions**

The functions in this section return mathematical expressions or plots.

**Hierarchical Clustering**

Clustering refers to grouping data in a way that data points in a group (or cluster) are more similar to each other than to data points in other groups. The goal of HC is to build a binary tree or dendrogram of data that successively merges similar groups of data points.

HC analysis clusters samples and variants using the modified Hamming distance method applied to the numerically converted variant calls as above, whereby:

- **Call Distance = 1** if two calls are different
Call Distance = 0 if two calls are the same

HC analysis then does the following:

1. Displays the results visually as a heatmap with a legend indicating variant calls. For example:

(2) Creates an hc object with a sample and variant list for all variables included in the table. This hc object can be saved. For more information, see Saving Data.

There are two options for generating HC plots in the SINGuLAR Analysis Toolset:

- With automatic analysis, as described in The Automatic Analysis Method.
- Without automatic analysis, as described in HC Analysis on Filtered Variant Calls. This also includes the ability to easily select individual clusters for trimming and eliminating data.

Fisher's Exact test

A Fisher's Exact test delivers an exact p-value between two categorical variables or variables with a finite number of outcomes. To determine whether two sample groups are associated for
variants, this test is performed for each variant in each pair of sample groups and calculates the p-value for each pairwise sample group. This can only be calculated if sample group information is provided.

The calculation is dependent on knowing the variant and non-variant calls for each identified chromosome position. This information can be calculated from the reference data used in creating the VCF or can come from a user identified control sample or control sample groups.

The SINGuLAR Analysis Toolset then automatically calculates the number of variant and non-variant samples for the different sample groups. Missing data is not included as non-variant data for these calculations. For example, for four types of variants (1) missing data, (2) non-variant (A/A), (3) variant 1 (A/B), and (4) variant 2 (B/B), the p-value is calculated as shown in Table 2 and the associated equation:

<table>
<thead>
<tr>
<th>Sample Group 1</th>
<th>Variants: A/B, B/B</th>
<th>Non-Variants: A/A</th>
<th>Row Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a + b</td>
<td></td>
</tr>
<tr>
<td>Sample Group 2</td>
<td>c</td>
<td>d</td>
<td>c + d</td>
</tr>
<tr>
<td>Column Total</td>
<td>a + c</td>
<td>b + d</td>
<td>a + b + c + d (=n)</td>
</tr>
</tbody>
</table>

Table 2: P-value calculations for four types of variants: (1) missing data, (2) non-variant (A/A), (3) variant 1 (A/B), and (4) variant 2 (B/B)

\[
p = \frac{(a + b)(c + d)!}{(a + b + c + d)!} \frac{(a + c)! (b + d)!}{a! b! c! d! n!}
\]


### Visualizing Variant Calls

The calculated p-values from the Fisher’s Exact Test p values are displayed in a Manhattan plot. This plot is a visual representation of variants between two sample groups that are statistically significant.

Significant variants are plotted along the x-axis according to genomic position, with the negative log of the p-value for each SNP displayed on the y-axis. Thus, highly significant variants will be the highest on the plot.
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

Evaluating Data

You can employ quality control metrics to assess call sensitivity and concordance between genotypes for a variant callset.

Evaluation of variant call performance compares the calls of the samples in the VC object to give the common call list and returns a tab-delimited data table. Sources can be genomic samples or the reference genome.

A list of the possible outcomes of the sample:common call comparison is displayed below. The number of variants with each of these outcomes is totaled and reported. You can compare a single sample:common call or with groups:common call comparisons.

Variant Call Definitions for a Sample Group

(1) **aa_aa.** The number of samples of a variant with GT call as AA; and its GT in common call list is AA.
(2) **aa_ab.** The number of samples of a variant with GT call as AB; and its GT in common call list is AA.
(3) **aa_bb.** The number of samples of a variant with GT call as BB; and its GT in common call list is AA.
(4) **aa_missing.** The number of samples of a variant with GT call as missing data; and its GT in common call list is AA.
(5) **ab_aa.** The number of samples of a variant with GT call as AA; and its GT in common call list is AB.
(6) **ab_ab.** The number of samples of a variant with GT call as AB; and its GT in common call list is AB.
(7) **ab_bb.** The number of samples of a variant with GT call as BB; and its GT in common call list is AB.
(8) **ab_missing.** The number of samples of a variant with GT call as missing data; and its GT in common call list is AB.
(9) **bb_aa.** The number of samples of a variant with GT call as AA; and its GT in common call list is BB.
(10) **bb_ab.** The number of samples of a variant with GT call as AB; and its GT in common call list is BB.
(11) **bb_bb.** The number of samples of a variant with GT call as BB; and its GT in common call list is BB.
(12) **bb_missing.** The number of samples of a variant with GT call as missing data; and its GT in common call list is BB.

A comparison is made between each sample or sample group and the control samples or reference data, resulting in the following calculations: Allele Dropout (ADO), Locus Dropout (LDO), overall genotype concordance, non-reference sensitivity, and non-reference discrepancy.
The following quality control metrics are from the VariantEval module in the GATK forum on the Broad Institute website: http://gatkforums.broadinstitute.org/discussion/48/using-varianteval

**Allele Dropout (ADO)**

This table shows the loss of one allele from a heterozygous site during amplification.

<table>
<thead>
<tr>
<th>Genotype of comparison calls</th>
<th>A/A</th>
<th>A/B</th>
<th>B/B</th>
<th>./</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A/B</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>B/B</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>./</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

Calculation = \((2+10) \div (2+6+10) = \frac{12}{18}\)

**Locus Dropout (LDO)**

This table shows the loss of both alleles during amplification.

<table>
<thead>
<tr>
<th>Genotype of comparison calls</th>
<th>A/A</th>
<th>A/B</th>
<th>B/B</th>
<th>./</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A/B</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>B/B</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>./</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

Calculation = \((13+14+15) / (1+2+3+5+6+7+9+10+11+13+14+15) = \frac{42}{96}\)
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

Non-Reference Sensitivity

This table shows the fraction of sites called variant that are also variant in the evaluation set.

<table>
<thead>
<tr>
<th>Genotype of comparison calls</th>
<th>A/A</th>
<th>A/B</th>
<th>B/B</th>
<th>./</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A/B</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>B/B</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>./</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

Calculation = \( \frac{6+7+10+11}{2+3+6+7+10+11+14+15} = \frac{34}{68} \)

Non-Reference Discrepancy Rate

This table shows the accuracy of genotype calls at sites called in both the comparison and evaluation set, excluding concordant homozygous reference calls.

<table>
<thead>
<tr>
<th>Genotype of comparison calls</th>
<th>A/A</th>
<th>A/B</th>
<th>B/B</th>
<th>./</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A/B</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>B/B</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>./</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

Calculation = \( \frac{2+3+5+7+9+10}{2+3+5+6+7+9+10+11} = \frac{36}{53} \)
Overall Genotype Concordance

This table shows the accuracy of genotype calls at all sites.

| Genotype of comparison calls | A/A | A/B | B/B | ./.
|------------------------------|-----|-----|-----|-----
| A/A                          | 1   | 2   | 3   | 4   |
| A/B                          | 5   | 6   | 7   | 8   |
| B/B                          | 9   | 10  | 11  | 12  |
| ./                           | 13  | 14  | 15  | 16  |

Calculation = \( \frac{1+6+11}{1+2+3+5+6+7+9+10+11} = \frac{18}{54} \)

The Automatic Analysis Method

When you enter the function `vcAutoAnalysis`, the FluidigmSC Analysis dialog is displayed. Use this graphical user interface to:

- **Read single or multiple VCF files and return the created VC object.** You can also restrict the import to specific-target regions of interest.
- **Add sample group annotation to the sample list.**
- **Choose either a control sample or a control sample group** to include only those variant calls that are in the defined control.
- **Enter user-defined threshold values for DP, GQ, AF and minimum variant number in samples.** The resulting Histogram plots show you variant-call quality and contain a red line to highlight each user-defined threshold value. Re-define threshold values as needed. The distribution of parsed Quality metrics from VCF are as follows:
  - **DP.** Read Depth, combined depth across samples
  - **GQ.** Conditional Genotype Quality, encoded as a Phred quality in the org_data data frame. (For more information on org_data, see Appendix D: Contents of the VC Object.)

  Phred quality scores are assigned to each nucleotide base call in automated sequencer traces, provide accurate and quality-based consensus sequences, and can be used to compare the efficacy of different sequencing methods.

  Phred quality scores \( Q \) are defined as a property which is logarithmically related to the base-calling error probabilities \( P \).
\[ Q = -10 \log_{10} P \]

Or

\[ P = 10^{-\frac{Q}{10}} \]

From: http://en.wikipedia.org/wiki/Phred_quality_score

For example, if Phred assigns a quality score of 30 to a base, the chances that this base is called incorrectly are 1 in 1000. The most commonly used method is to count the bases with a quality score of 20 and above.

- **AF.** Allele Frequency (for each ALT allele in the same order as listed: use this when estimated from primary data, not called genotypes)
- **VSN.** The variant call frequency of sample group

**Perform HC analysis to display variant calls in samples.** Each genotyping call on a sample is assigned as one of five categories for each sample:

- **M** - Allele data is missing for the genomic sites under consideration.
- **AA** - Sample contains only Allele A for all genomic sites under consideration.
- **AB** - Sample contains both Allele A and Allele B for all genomic sites under consideration.
- **BB** - Sample contains only Allele B for all genomic sites under consideration.
- **NC (NO-CALL)** - No variant call is made because the data falls below all user-defined threshold values.

**Analysis method to filter variants:**

- **All Variants.** Considers all variant calls in the analysis and filters any calls having low-quality in all samples.
- **Significant Variants.** Performs Fisher's Exact test for variant calls between sample group pair, and then filters variants having a p-value greater than the defined threshold (default 0.05) in all sample-pairs.
- **Control Variants.** Filters out all variant calls that are non-variant calls in the control sample or control sample group.
- **Control Calls.** Gets a common call list in control sample group with call concordance greater than the defined threshold in control samples. (For AutoAnalysis, this threshold is 0.51). Then filters out any variant calls that are not in the common call list.

**Filter sample calls.** You can choose the following analysis methods to filter samples:

- **All Samples.** All samples are used in variant analysis.
- **High Quality Samples.** Filters out all low-quality samples, which are defined as having the percentage of either NO-Call or Missing data from all variants as greater than the defined threshold. (For AutoAnalysis, this threshold is 0.50.)
- Save the VC object file.
- Write out VCF data in VCF format. The variants that failed in filters are not included.

**Perform Automatic Analysis**

The function `autoAnalysis` performs several tasks, as described in this section and in the section *The Automatic Analysis Method*.

**A Note About Using Functions Without Automatic Analysis**

All the functions that are associated with the function `vcAutoAnalysis` can also be run by themselves. To find the functions that are available in the SINGULAR Analysis Toolset, see either *Appendix B: The List of Functions for Variant Analysis* or open the fluidigmSC Help while you have the R console open.

**To browse the fluidigmSC Help from the R console**

1. Enter `??fluidigmSC`.
2. At the upper left, click the link `fluidigmSC::fluidigmSC_<version number>`.
3. At the bottom right, click the link `[Package fluidigmSC version <version number> Index]`.

**Running the Automatic Analysis**

Automated analysis of variants in the SINGULAR Analysis Toolset helps you to conduct a series of VC analyses using a graphical user interface:

1. Launch R.
2. Do this only one time per session: Enter the function `library(fluidigmSC)` to load the package.
3. Enter the function `vcAutoAnalysis()` to display the FluidigmSC Analysis dialog that will retrieve experiment data along with the sample list and/or variant list supplied by you.
4. Select a data type, and then upload one or more files. (If multiple files are selected, they will be merged into one VC object automatically.)

Your choices of data types are:

- **fluidigmSC VC**. Upload an existing *.fso file.
- **VCF**. You can restrict the import of this data to target regions by providing targets of interest. Targets can be either (a) chromosome coordinates, such as 7:123-
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

345;chr7:123-345 or chr7 ; (b) gene symbol such as TP53. The targets can have multiple (a) and (b) or the combination of (a) and (b). The delimiter is “;”.

5 In the Sample Annotation section, select the appropriate sample annotation method. The default is Defined in the variant file(s), indicating that sample group annotation was defined previously in the variant files.

Alternatively, you can add to the sample list by either providing an updated sample-list file with sample group annotation (From file) or by specifying a sample prefix (Annotation by prefix “-“ or Annotation by prefix “_“). File formats are located in Prepare the Input Files, and information about naming samples and sample groups are in Rules and Guidelines for Naming Genes and Samples.

6 If you have sample group annotation, enter a control sample or sample group name into the Control: Sample Name or Sample Group Name box. In this example, no name is entered:

The name you enter will be used in variant filter analysis, which will only include calls that contain variant calls in the defined control.

7 In the Variant Quality section:
   - You can enter defined thresholds for DP, GQ, AF, and the minimum sample number per variant (also referred to as VSN). These numbers will be used to determine variant-call quality. If a variant call of sample fails in all of thresholds, it is assigned as "No-CALL", with a flag value as "0" in the SCQC data frame of VC object. The defaults are DP = 10; GQ = 80; AF = 0.1, and VSN = 2. These values are what Fluidigm considers to be the minimum set of variant calling values to include the maximum amount of data. You can be more stringent with your data. For example, a VSN value of 2 precludes random values.

When you click Display Quality in the Variant Quality section, you see histogram plots that contain a red line to highlight the defined threshold.

For example:
You can display an HC analysis that contains variant calls in samples. Click **Display Heatmap** in the **Variant Quality** section:

- **Variant Quality:**
  - Read Depth (DP): 10
  - Genotype Quality (GQ): 80
  - Allele Frequency (AF): 0.1

- **Variant Sample Number (VSN):** 2

  ![Variant QC Plots](image)

- **Display Quality...**
- **Display Heatmap...**
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

Each genotyping call of sample is assigned as one of four categories: M, AA, AB, BB, and NC (No-CALL). For example:

8 In the **Analysis Method** section:
   a Select the method to filter variant calls:
     
     ![VCF viewer interface]

   b Specify whether the samples should be restricted to only high quality samples:

9 In the **Output Folder** box, enter a folder that will hold the resulting VC object file, which is now available for further analysis. For example:

10 Click **Analyze**.
   As the analysis progresses, you see the following in the R console:

   ```
   Retrieving variant data
   Applying variant quality filter
   Saving analysis results to output folder
   Displaying analysis results
   ```
Examining the Results

The results that you get with the `vcAutoAnalysis` function give you a display-only Heatmap plot. The results that you get without automatic analysis of variants can provide some interaction, such as the ability to identify sample or variant clusters.

Heatmap Plot

The Heatmap of Variants plot is displayed automatically upon running the function `vcAutoAnalysis`. For more information, see Hierarchical Clustering.

Cluster Identification

The results that you get without automatic analysis of variants can provide some interaction. For example, the function `vcIdentifySampleClusters` enables you to select clusters by clicking on the dendrogram and selecting the region of your choice. The data for each selected cluster is returned as a sample list with the cluster IDs.

```r
selected_sample_list_auto_analysis <- vcIdentifySampleClusters()
```

For example:

![Identify Sample Clusters](image)

To accept your selected regions of interest, click OK. To return to the previous screen, click Cancel.

You can then save the selected data (samples in this case) as a tab-delimited *.txt file:

```r
saveData(selected_sample_list_auto_analysis)
```
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

For example:

```
selected_sample_list_auto_analysis.txt - Notepad
```

<table>
<thead>
<tr>
<th>SampleID</th>
<th>GroupID</th>
<th>SampleAnnotation</th>
<th>GroupID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT1-SC_1_S15</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S26</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S38</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S20</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S19</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S17</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S02</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
<tr>
<td>CT2-SC_1_S34</td>
<td>SC_1</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2-SC_1_S40</td>
<td>SC_1</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2-SC_1_S35</td>
<td>SC_1</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT2-SC_1_S19</td>
<td>SC_1</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>CT1-SC_1_S24</td>
<td>SC_1</td>
<td>CT1</td>
<td></td>
</tr>
</tbody>
</table>

Summary Output Files

Summary Output files for samples and sample groups are tab-delimited text that you can open in Excel. For more information on sample annotation and the calculation of metrics, see Data Evaluation.

For Individual Samples: In the Sample Summary Output file, samples are displayed in the order that they were entered into the text file. For example:

```
<table>
<thead>
<tr>
<th>Sample</th>
<th>Control</th>
<th>Calls</th>
<th>In</th>
<th>AA</th>
<th>AB</th>
<th>AB-AB</th>
<th>AA-AB</th>
<th>AB-AB</th>
<th>BB-AB</th>
<th>BB-AB</th>
<th>ADO (%)</th>
<th>LOO (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>34</td>
<td>283</td>
<td>25</td>
<td>0</td>
<td>172</td>
<td>90</td>
</tr>
<tr>
<td>CT1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>25</td>
<td>0</td>
<td>129</td>
<td>77</td>
</tr>
</tbody>
</table>
```

For Samples in Groups: In the Sample Group Summary Output file, samples are displayed in groups in the order that they were entered into the text file. For example:
Saving the VC Object as an *.fso Output File

The output file from the vcAutoAnalysis function is saved in the user-defined directory and is named vc (auto_analysis).fso by default. This tab-delimited file that stores the VC object dataset that contains the data frames described in Appendix D: Contents of the VC Object.

Advanced Functions for Targeted Resequencing or DNA Seq Analysis

The SINGuLAR Analysis Toolset includes a variety of advanced functions for Targeted Resequencing or DNA Seq Analysis of data files. This section provides you with some examples. To view the complete list of functions for Variant Analysis, see Appendix B: The List of Functions for Variant Analysis.

Reading Experimental Data

When analysis is initiated with auto analysis, an *.fso variant object file is created and saved in the specific output directory. This file can be called for further analysis.

To open a VC object file with a *.fso filename extension

If you have not performed the vcAutoAnalysis function in your current session but you have an *.fso file that was created and saved previously, enter the following function to read an existing *.fso expression object:

\[ \text{vc <- readVCObject()} \]

All selected files must be in the same folder. To select more than one file, press the Ctrl key while clicking the files.

This function reads single or multiple VCF files and returns a newly created VC object. For a detailed description of the VC object, see Appendix D: Contents of the VC Object. The VC object is not saved automatically. To save it, see Saving Data.

To open a VCF file with a *.vcf filename extension

\[ \text{vc <- readVCF()} \]

Annotating Samples

The release of the SINGuLAR Analysis Toolset includes the addition of a GroupID for gene list files. If sample and variant group information was not provided during your use of the function vcAutoAnalysis (or vcAutoAnalysis was not performed), you can update the sample and variants from a file to include group information. For more information, see Prepare the Input Files.
To annotate Sample groups from a file

vc <- vcUpdateSampleListFromFile (vc)

You can also update variant group information from a file like the one above. For more information, see this function in Appendix B: The List of Functions for Variant Analysis.

Annotating Variants

There are several ways to update variant group annotations directly from the variant annotation list in the VC object. For more information, see the genetic variant annotation and effect prediction toolbox SnpEff at http://snpeff.sourceforge.net/index.html. Variant annotations are parsed from the INFO field in the VCF file. For more information on this field, see Get the Sample and Variant Group Annotation Files and:

http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41

Note: These fields are specific to VCF files from snpEFF. There are three options for updating group information: CHROM, GENE, and VARIANT_TYPE. These options are columns that can be seen in the "variant_annot_list" data frame as part of the VC object.

To annotate variants by CHROM

This updates the chromosome number as the Group information.

vc <- vcUpdateVariantListFromVariantAnnotation(vc, by_annotation = "CHROM")

To annotate variants by GENE SYMBOL

This function allows you to update the list of variants by annotations that are their gene ID values (such as the gene EGFR):

vc <- vcUpdateVariantListFromVariantAnnotation(vc, by_annotation = "GENE_SYMBOL")

To annotate variants by VARIANT_TYPE

This will give three types: INSERTION, DELETION, and SNP

vc <- vcUpdateVariantListFromVariantAnnotation(vc, by_annotation = "VARIANT_TYPE")

To view the annotation

vc$variant_list
Here, you enter the object (such as vc) and the frame of the object (such as variant_list), separated by the $ sign. The output is the contents of the file displayed in the R console. For example:

<table>
<thead>
<tr>
<th>VARIANT_ID</th>
<th>GroupID</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2072454</td>
<td>SNP</td>
</tr>
<tr>
<td>7:55219065_T_G</td>
<td>SNP</td>
</tr>
<tr>
<td>rs4947986</td>
<td>SNP</td>
</tr>
<tr>
<td>7:55223449_A_T</td>
<td>SNP</td>
</tr>
<tr>
<td>7:55223452_T_C</td>
<td>SNP</td>
</tr>
<tr>
<td>7:55224348_A_AG</td>
<td>INSERTION</td>
</tr>
<tr>
<td>7:55229224_T_TC</td>
<td>INSERTION</td>
</tr>
</tbody>
</table>

**Setting Custom Colors and Symbols**

SINGuLAR Analysis Toolset assigns a default color and symbol to each sample and variant group for visualization of samples and variants in each variant analysis (VC) object.

To customize the colors and symbols for sample groups of a given VC object

```r
vc <- vcSetSampleGroupColorAndSymbols (vc)
```

This will display a dialog that allows you to click both the symbol and color for each sample group.

When you are done, click OK. This will also allow you to reset colors to the default to select again by clicking Reset.

Subsequent plots will include this sample group information as well as the colors selected automatically.

**Saving Data**

You can save any fluidigmSC data object (hc, vc .*.fso object), sample list, and variant list to a file for future use. For an explanation on how to create sample and variant lists, see Step 3 in Run the Automatic Analysis.
To write the VC dataset out as VCF file
writeVCF(vcf)

To save the VC object to files for future use
saveData (vc)

To save a variant or sample list to a *.txt file
saveData (example_variant_list)

To save a plot as an image file
Choose File > Save as, and then click the type of file you want. Your choices are: Metafile, Postscript, PDF, Png, Bmp, TIFF, and Jpeg.

HC Analysis on Filtered Variant Calls
HC analysis on the VC object clusters samples and variants using the modified Hamming distance method. It returns the HC object with a variant list (variant_list). For more information, see Hierarchical Clustering.

To conduct HC analysis on a VC object and omit specific data
hc <- vcHC(vc, missing_data_as_AA= FALSE)

With the default setting of missing_data_as_AA set as FALSE, new GT data will be generated, based the combination of GT and SCQC. For each genotype of sample, its value should be one of the five genotypes:
- 0 (missing_data)
- 1(AA)
- 2(AB)
- 3(BB)
- 4(NO-Call)

If missing_data_as_AA is set as TRUE, missing_data will be treated as 1 (AA) and GT data will be one of the four genotypes:
- 1(AA)
- 2(AB)
- 3(BB)
- 4(NO-Call)

With this revised GT data, the call distance is calculated by the Hamming distance method:
- Call Distance = 1 if two calls are different
- Call Distance = 0 if two calls are the same

**To change plot colors and omit specific data**

Use this function to conduct HC analysis on a VC object while customizing the color scheme of plot vcHC:

```r
hc <- vcHC(vc, color_scheme="black_blue_green_red_grey", missing_data_as_AA=FALSE)
```

**To display an existing HC analysis**

```r
vcDisplayHC(hc)
```

**Identifying sample and variant clusters**

The functions `vcIdentifySampleClusters` and `vcIdentifyVariantClusters` enable you to select clusters by clicking on the dendrogram. Selected clusters are returned as a sample or variant list along with the cluster group information.

**To identify sample clusters of interest**

Use the following function after you perform HC analysis:

```r
select_sample_clusters <- vcIdentifySampleClusters(hc)
```

**To identify variant clusters of interest**

```r
select_variant_clusters <- vcIdentifyVariantClusters(hc)
```

In the graphical user interface, use your cursor to select clusters. Your selections are within the red boundaries. To deselect a cluster, click the red line. For example:
To update the variant list and create a new VC object with a subset of variants

```r
new_vcf <- vcUpdateVariantListFromList(vcf, select_variant_clusters)
```

The new VC object can now be used for further analysis, such as for HC.

### Conducting a Fisher’s Exact Test for Case/Control Association Analysis

To determine whether two sample groups are associated for filtered variants, perform a **Fisher’s Exact test** for each variant in each pair of sample groups and return the results for each variant with the p-value for each pairwise sample group.

The data can be saved as an updated variant list with only the significant variables and also as an interactive Manhattan plot.

**IMPORTANT!** This function requires that samples be annotated with group information. Two sample groups must be used for comparison. For more information, see [Annotate Samples](#) and [Annotate Variants](#).

To perform the Fisher’s Exact test on VC data and create a list containing the calculation

```r
ft <- vcPerformFisherTest(vc)
```

To get the significant variants in the Fisher Exact test from the previous calculation

The default p value threshold is 0.05.

```r
fisher_variant_list <- vcGetSignificantVariants(ft, pvalue_threshold = 0.05)
```
To get the significant variants and display a Manhattan plot

```r
fisher_variant_list <- vcDisplayFisherTestResults(ft, "sample_group1", "sample_group2", pvalue_threshold = 0.05)
```

You must replace "sample_group1" and "sample group2" with the names of your samples groups. Default p value is 0.05. For example:

```r
fisher_variant_list <- vcDisplayFisherTestResults(ft, "CT1", "CT2", pvalue_threshold = 0.05)
```

For example:

To select variants of interest

In the Manhattan plot, you can:

- **Circle.** Draw a circle by connecting dots on the plot. Once a circle is formed, the plot displays each sample ID inside the circle.
- **Point.** Click a sample to display its ID.
- **Clear.** Returns the plot to its original state.
- **Save.** Save selected samples to a file.
- **Done.** Click this command to indicate that you are done locating samples of interest.
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

Updating Variant Quality, Processing a Variant List, and Updating the VC Object

Fluidigm uses predefined metrics (DP, GQ, AF, and minimum sample number for a variant call, VSN) and their thresholds to evaluate variant quality. With the \texttt{vcAutoAnalysis} function, you can use the graphical user interface (GUI) to define and view these variant quality metrics as histograms with plotted thresholds and determine their effects on hierarchical clustering.

The resulting output is a variant list that you can update iteratively and at any point, either during automatic analysis or later, using the functions described below. For more information on these metrics, see \texttt{Data Evaluation}.

One of the advantages of using the functions described below without the GUI for automatic analysis is the ability to change the default value of the DPGQAF\_operator argument from "AND" to "OR":

\begin{itemize}
  \item \textbf{AND}. High quality variant calls need to pass all thresholds.
  \item \textbf{OR}. High quality variant calls need to pass at least one of the thresholds.
\end{itemize}

This data is stored in the SCQC data frame of the VC object: 1 = pass, 0 = no-call, and -1 = missing data.

To update variant quality metrics for all samples

The defaults are shown here, which you can change. This function will update the SCQC numbers for each sample. However, it does not update the sample list to remove those samples that are now identified as low quality. The next function explains how to perform that step.

\begin{verbatim}
vc <- vcUpdateVariantQuality(vc, DP_cutoff=20, GQ_cutoff=80, AF_cutoff=0.1, Sample_Num_cutoff=2, DPGQAF_operator = "AND")
\end{verbatim}

To update the VC object to remove low-quality variants from variant list

You can check the variant quality of variant calls of samples, remove any variants from variant list if variants have low-quality cross all samples, and then return the updated variant object. This object will use the variant quality table for updating the variant list. The quality metrics can be changed using the previous function.

\begin{verbatim}
vc <- vcUpdateVariantListByVariantQuality(vc)
\end{verbatim}

To get a list of the high-quality variants

This function will return an \texttt{ok\_variant\_list} that can be saved according to the \texttt{Saving Data} section.

\begin{verbatim}
ok_variant_list <- vcGetFilteredVariantListByVariantQuality(vc)
\end{verbatim}
To use this list to update variant list by creating a new VC object without changing the original

new_vc<-vcUpdateVariantListFromList(vc, ok_variant_list)

You can now perform new HC analysis to VC and compare to original vcHC on hc

new_hc<-vcHC(new_vc)

For example:

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
</table>

Getting Variant Calls with Genotyping Information from a Sample

Use the GUI with the function vcAutoAnalysis to display predefined variant-quality metrics (DP, GQ, AF, and minimum sample number for a variant call, VSN) with user-defined thresholds. For more information, see Data Evaluation.

To retrieve variant list with all calls from a sample or sample group for a previously defined VC object

This function returns a variant_list that may be saved.

variant_list <-vcGetVariantListWithCallFromSample(vc, "mysample")

To return a variant list with variant calls only from a sample

If variant_only is set as TRUE, this function returns variant calls, but not the locations where the sample specified is a non-variant when compared to the common call.
variant_list_variant_call <- vcGetVariantListWithCallFromSample(vc, "sampleID", variant_only=TRUE)

To return a variant list with the most common genotype called above a user defined concordance threshold for all samples in the defined group

This function is useful when you are interested in variant calls with high concordance across the defined sample group. You can define your own concordance threshold, which will include all samples at or above that confidence threshold. The default is 0.51. When `variant_only` is set to `FALSE`, this will include both non-variant and variant calls when compared to common calls, by default.

sample_group_variant_list_all_call <- vcGetCommonVariantListWithCallFromSampleGroup(vc, "sample_group", concordance_threshold = 0.51, variant_only = FALSE)

Evaluating Variant Call Performance by Comparing Variant Calls of Samples with Common calls

Evaluation of variant call performance compares the calls of the samples in the VC object to give the common call list and returns a tab-delimited data table. You can compare a single sample:common call or with groups:common call comparisons. The common call group was created in the previous section using the function `vcGetCommonVariantListWithCallFromSampleGroup`.

With the `vcAutoAnalysis` function, you can use the graphical user interface (GUI) to define the common call sample source and to define and view the quality metrics.

The calculated variant call performance metrics are saved as the Sample Summary output file or the sample Group Summary Output file. For more information about these tables, see Examine the Results.

You can compare a single sample:common call or with groups:common call comparisons. The common call source was defined when creating the VC object or during AutoAnalysis.

IMPORTANT! The input table for this function was made in the previous section from the `vcGetCommonVariantListWithCallFromSampleGroup` function for sample groups. The quality metrics used for this section were defined using the functions `vcUpdateVariantQuality` and `vcUpdateVariantListByVariantQuality`.

The function `vcEvalSampleGroupVariants` calls the common variant table created by the `vcGetCommonVariantListWithCallFromSampleGroup` function for sample groups. The quality metrics from Get Variant Calls with Genotyping Information from a Sample.

`group_eval_table <- vcEvalSampleGroupVariants(vc, sample_group_variant_list_all_call)`

For variant call definitions, see Data Evaluation. These possible combinations of genotyping outcomes will be used to calculate the evaluation metrics ADO, LDO, GC, NRS, and NRD. For more
information on how these metrics are calculated, see Data Evaluation. The table format is described in Examine the Results.

**To compare variant calls of individual samples with common calls**

The function `vcEvalSampleVariants` calls the common variant table created by the `vcGetVariantListWithCallFromSampleGroup` from Get Variant Calls with Genotyping Information from a Sample.

```r
variant_list_all_call <- vcGetVariantListWithCallFromSample(vc, "sampleID", variant_only=FALSE)
eval_table <- vcEvalSampleVariants(vc, variant_list_all_call)
```

This can be saved as txt file see the Saving Data section.

**To compare variant calls of sample groups with common calls**

The function `vcEvalSampleGroupVariants` calls the common variant table created by the `vcGetCommonVariantListWithCallFromSampleGroup` from Get Variant Calls with Genotyping Information from a Sample.

```r
sample_group_variant_list_all_call <- vcGetVariantListWithCallFromSampleGroup(vc, "sampleID", variant_only=FALSE)
eval_table <- vcEvalSampleVariants(vc, sample_group_variant_list_all_call)
```

This can be saved as txt file see the Saving Data section.

---

**Advanced Functions for WES or Whole Genome DNA Seq Analysis**

This section provides you with some examples of advanced functions for WES or Whole Genome DNA Seq Analysis of data files.

**Note:** WES scale analysis can take up to several minutes to complete, and WGS scale analysis can take several hours. To minimize analysis time, be certain that your computer is not additionally running other software applications in parallel.

For information on the following topics, see "Advanced Functions for Targeted Resequencing or DNA Seq Analysis" in this document:

- Reading experimental data
- Annotating samples
- Annotating variants
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

- Setting custom colors and symbols (for sample and variant groups)
- Saving data
- HC analysis on filtered variant calls
- Identifying sample and variant clusters
- Conducting a Fisher's Exact Test (Case vs. Control) association analysis
- Updating Variant Quality, Processing a Variant List, and Updating the VC Object
- Getting Variant Calls with Genotyping Information from a Sample
- Evaluating Variant Call Performance by Comparing Variant Calls of Samples with Common calls

There is one function listed below for mutation analysis. To view the complete list of functions for variant analysis, see Appendix B: The List of Functions for Variant and Mutation Analysis.

Identifying Significant Variants and Mutations at Whole Exome Scale

The function \texttt{vcIdentifySigVariants} displays a Graphical User Interface (GUI) that allows you to detect significant variants or mutations from raw variant data with a set of filters, such as variant quality, variant event in number of single cells, and/or by case and control mutation analysis. It then performs hierarchical clustering analysis to group the heterogeneous samples, returns a summary report, and saves the VC object file for any downstream analysis.

1. Launch R.
2. \textbf{Do this only one time per session}: Enter the function \texttt{library(fluidigmSC)} to load the package.
3. Enter the function \texttt{vcIdentifySigVariants()} to display the first of three dialogs.
4. Select one of the following study types:

   - \texttt{Mutation Analysis (Case/Control)}
   - \texttt{Single-Cell Heterogeneity Analysis}

5. Import a VCF file into the Variant File (VCF) box:

   \begin{center}
   \textbf{Variant File (VCF):} \hspace{1cm} \textbf{Browse...}
   \end{center}

6. (Optional) Click the type of target to import, and then import one or more target files into the Targets box:

   \begin{center}
   \textbf{Targets (optional):} \hspace{1cm} \textbf{Browse...}
   \end{center}

   You can restrict the import of this data to target regions by providing targets of interest. Targets can be either (a) chromosome coordinates, such as 7:123-345;chr7:123-345 or chr7; or (b) gene symbol from a gene list, such as TP53. The targets can be a multiple of (a) and (b) or the combination of (a) and (b). The delimiter is ";".

7. In the Sample Annotation section, select the appropriate sample annotation method or import a Sample List file (with a *.txt filename extension) that contains pre-defined sample group annotation (the default):
File formats are located in Prepare the Input Files, and information about naming samples and sample groups are in Rules and Guidelines for Naming Genes and Samples.

8. In the Sample Group Categories section, click Define, enter your choice of single-cell groups in the dialog that appears, and then click OK.

If your study type is Mutation Analysis (Case/Control), select single-cell case and control groups from a set of dropdown lists, and if you have the bulk equivalents, you can optionally include them:

- **Single-cell control group (e.g. normal):** Choose one
- **Single-cell case group (e.g. tumor):** Choose one
- **Bulk cell control group (optional):** Choose one
- **Bulk cell case group (optional):** Choose one

If your study type is Single-Cell Heterogeneity Analysis, select single and bulk cells from a list. You can use the Shift and Control keys to select multiple values.

In each case, you see the specified groups in the lower region of the dialog.

9. Click Next.

10. In the dialog that appears, you can use the default filters and their values or change them in any combination, where applicable. If a variant call of sample fails in all of thresholds, it is assigned as "No-CALL", with a flag value as "0" in the SCQC data frame of VC object. The filters are as follows:

    **Filter Type: General Variant Quality**

    - **Read Depth (DP)** (read depth at this position for this sample: 5 - 1000)
    - **Genotype Quality (GQ)** (conditional genotype quality at this position for this sample; GQ 20 means 1/100 chance of being false positive: 10-100)

    **Filter Type: Single-Cell Related Variant Quality - for Mutation Analysis (Case/Control)**
Chapter 4: The SINGuLAR Analysis Toolset for Variant & Mutation Analysis

- **Variant Non-Reference Allele Frequency (AF)**  
  [AF is calculated by $AD2 / (AD1 + AD2)$ at this position for this single cell: 0.05 - 1]

- **Variant Found in Single Cells (VSN)**  
  [VSN cutoff is dynamically calculated based on a binominal test with given single-cell number and false discover rate (FDR) of 4E-6 and whole-exome target regions of ~38M bases: 2-99]

  For a VSN of 2 (the default), the expected probability of having a false-positive variant call at any target position is $1.004e-11$. If you change the VSN value, you also need to click **Recalculate** to recalculate this expected probability value.

  The false discovery rate (FDR) of single cell genotyping is determined based on high confidence homozygous sites in bulk genomic DNA, and then applying the FDR to a binomial test (cumulative distribution function) to determine the probability of observing a variant in a given number of cells amongst the total number of cells tested.

  **Note:** In the Fluidigm C1™ System, we observed that the FDR in WGA of a single-cell is 4E-6, and the most False variant calls were only found in one cell.

Filter Type: Single-Cell Related Variant Quality - for **Single-Cell Heterogeneity Analysis**

- **Variant Non-Reference Allele Frequency (AF)**  
  [AF is calculated by $AD2 / (AD1 + AD2)$ at this position for this single cell: 0.05 - 1]

- **Variant Found in Single Cells (VSN)**  
  [VSN cutoff is dynamically calculated based on a binominal test with given single-cell number and false discover rate (FDR) of 4E-6: 2-50]

  For a VSN of 2 (the default), the expected probability of having a false-positive variant call at any target position is $1.96e-08$. If you change the VSN value, you also need to click **Recalculate** to recalculate this expected probability value.

  The false discovery rate (FDR) of single cell genotyping is determined based on high confidence homozygous sites in bulk genomic DNA, and then applying the FDR to a binomial test (cumulative distribution function) to determine the probability of observing a variant in a given number of cells amongst the total number of cells tested.

Filter Type: Fisher's Exact Test (Case vs. Control) - for **Mutation Analysis (Case/Control)**

- **Pvalue cutoff**
  - **tails**  
    - **less**  
    - Your choices are "two.sided," "greater," and "less" (the default).

Filter Type: Additional Mutation Filters - for **Mutation Analysis (Case/Control)**

- **Not In Bulk-Cell Control Samples**
• □ In Bulk-Cell Control Samples
• □ FILTER flag as "PASS"
• □ No Indel
• □ Missense if Having Variant Annotation
• □ Non-dbSNP

Filter Type: Additional Mutation Filters - for **Single-Cell Heterogeneity Analysis**
• FILTER flag as "PASS"
• No Indel
• Missense if Having Variant Annotation
• Non-dbSNP

11. Click **Browse** at the right of the **Output Folder**, navigate to and select the name of a folder that will hold your variant object data. This data file will contain a *.*fso filename extension.

12. Click **Analysis**. As the analysis progresses, you see the progress listed in the R console.

**Examining the Results**

After you click **Analysis** for the function `vcIdentifySigVariants()`, a report is generated in the R window that might take a few minutes for all the data to be displayed. (To shorten the data-generation time, apply more filters to the `vcIdentifySigVariants()` function.)
Example: Report for Mutation Analysis

Total samples in VCF data 103
Total samples in provided sample list 103
Total variants in VCF data 129666
Total dbSNP variants in VCF data 34128
Total non-dbSNP variants in VCF data 95538
Multiple-allele variants removed = 681
Variants not passing VSN filter of 3 removed = 83191
Variants not passing Fisher's Exact Test with p-value cutoff of 0.05 removed = 43889
Final selected mutations = 1905

Saving analysis results to output file C:/MyFiles/Output/vcf.fso ...
Displaying analysis results ...

To locate the samples of interest in the Heatmap plot, please run:
vcIdentifySampleClusters()

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total samples in VCF data</td>
<td>103</td>
</tr>
<tr>
<td>Total samples in provided sample list</td>
<td>103</td>
</tr>
<tr>
<td>Total variants in VCF data</td>
<td>129666</td>
</tr>
<tr>
<td>Total dbSNP variants in VCF data</td>
<td>34128</td>
</tr>
<tr>
<td>Total non-dbSNP variants in VCF data</td>
<td>95538</td>
</tr>
<tr>
<td>Variants not passing VSN filter of 3 removed</td>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>Variants not passing Fisher's Exact Test with p-value cutoff of 0.05 removed</td>
<td>43889</td>
</tr>
<tr>
<td>Final selected mutations</td>
<td>1905</td>
</tr>
</tbody>
</table>

When the Mutation Analysis report is finished, you will see:

- A single heatmap for both sample and variant clusters (as shown in the graphic) if there are less than 2500 variants selected.
- Two heatmaps if samples are annotated: (1) one with both sample and variant clusters and (2) another with only variant clusters and samples ordered by the sample groups.
Example: Heatmap for Mutation Analysis

![Heatmap of Variants](image)

Example: Report for Single-Cell Heterogeneity

<table>
<thead>
<tr>
<th>Analysis Summary:</th>
<th>Category</th>
<th>Value</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 DP cutoff</td>
<td>10</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>2 GQ cutoff</td>
<td>10</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>3 Variant Allele Frequency cutoff (AF)</td>
<td>0.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4 Variant found in single cells cutoff (VSN)</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5 Variants with FILTER flag as PASS</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6 No Indel</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7 Missense SNPs</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8 Non-dbSNP</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9 Single cell groups tumor-SC; normal-SC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10 Bulk cell groups normal-gDNA; tumor-gDNA</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11 Total samples in VCF data</td>
<td>103</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>12 Total samples in provided sample list</td>
<td>103</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>13 Total variants in VCF data</td>
<td>129666</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>14 dbSNP variants</td>
<td>34128</td>
<td>26.32%</td>
<td></td>
</tr>
<tr>
<td>15 Non-dbSNP variants</td>
<td>95538</td>
<td>73.68%</td>
<td></td>
</tr>
<tr>
<td>16 Multiple-allele variants removed</td>
<td>681</td>
<td>0.53%</td>
<td></td>
</tr>
<tr>
<td>17 InDel variants removed</td>
<td>4421</td>
<td>3.41%</td>
<td></td>
</tr>
<tr>
<td>18 Variants with FILTER as not &quot;PASS&quot; removed</td>
<td>788856</td>
<td>60.81%</td>
<td></td>
</tr>
<tr>
<td>19 Non-Missense variants removed</td>
<td>22313</td>
<td>17.21%</td>
<td></td>
</tr>
<tr>
<td>20 dbSNP variants removed</td>
<td>11661</td>
<td>8.99%</td>
<td></td>
</tr>
<tr>
<td>21 Variants not passing VSN filter removed</td>
<td>11085</td>
<td>8.55%</td>
<td></td>
</tr>
<tr>
<td>22 Final selected variants</td>
<td>649</td>
<td>0.5%</td>
<td></td>
</tr>
</tbody>
</table>
Saving the VC Object as an *.fso Output File

The output file from the `vcIdentifySigVariants` function is saved in the user-defined directory and is named `vcf.fso` by default. This tab-delimited file that stores the VC object dataset that contains the data frames described in Appendix D: Contents of the VC Object.
Appendix A: The List of Functions for Gene Expression

This Appendix contains all the functions for gene expression used in the SINGuLAR Analysis Toolset.

Installing

<table>
<thead>
<tr>
<th>Function Description</th>
<th>Code/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Fluidigm Single Cell Package Library for commands for the current session.</td>
<td>library(fluidigmSC)</td>
</tr>
<tr>
<td>The first method to be called after installing the fluidigmSC package. This function must only be run once.</td>
<td>firstRun() Downloads the following required packages: lattice, tcltk2, SDMTools, and rgl.</td>
</tr>
<tr>
<td>Set your working directory.</td>
<td>setwd(&quot;pathway of your working directory&quot;)</td>
</tr>
<tr>
<td></td>
<td>Example: setwd(&quot;C:/folder &quot;)</td>
</tr>
</tbody>
</table>

Getting R Help for the fluidigmSC Library

| Get the complete list of fluidigmSC functions, and then click any link to display the corresponding help. This will engage the default Internet program to display an interactive Help page. After you are done viewing the Help, click the R console to continue. | ?? fluidigmSC |
|                                                                                                                                   | Then click fluidigmSC::fluidigmSC_<version number>. |
|                                                                                                                                   | To see the complete list of R Help for fluidigmSC, click [Package fluidigmSC version <version number> Index] at the bottom of the page fluidigmSC_<version number>-package. |
| Get help on an individual function in the R application or in the fluidigmSC library. Do not include parenthesis or dependencies. R is case-sensitive. | ? functionName |
| List two subgroups for available functions. This will then force you to choose a                                              | scFunctions()                                      |
| For Gene Expression Analysis functions, type                                                                                     |                                                      |
### Identifying Outliers and Auto-Analyzing Data

Identify outliers in expression data and confirm and modify outlier candidates using a graphical user interface (GUI).

The EXP object is saved with outliers removed from further analysis. The default name of the EXP object is `exp(identify_outliers).fso`.

**identifyOutliers()**

Opens the **FluidigmSC Outlier Identification** dialog where you enter the files and instructions for automatic outlier identification and then run it.

You can enter:
- Expression file to be analyzed.
- LoD of your choice.
- Sample annotations (categorizing the samples into specific sample groups).

This then creates an interactive display. To identify outliers, click them. To save the file, click **Save**.

Auto-analyze expression data for a set of differentially expressed genes using a graphical user interface (GUI).

A PCA Score plot, Hierarchical Clustering (HC) heatmap, and a Violin plot by genes are displayed.

**autoAnalysis()**

Opens the **FluidigmSC Analysis** dialog where you enter the files and instructions for automatic analysis and then run it.

You can enter:
If sample annotation is provided, the Violin plot is ranked by the order of ANOVA p-values. Otherwise, it is ranked by PCA gene scores.

Also returns the text file **selected_gene_list** *(auto_analysis).txt* that contains genes ranked by ANOVA p-value between sample groups.

### Reading and Saving Expression Data

<table>
<thead>
<tr>
<th>Initialize the EXP object from one or more Ct expression data files that were generated by the BioMark™ system.</th>
<th>exp &lt;- readCtExp (exp_file=TRUE, lod=24)</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp_file: The filename of the data to be opened. The default is set to TRUE, which displays a file dialog that allows you to select one or more expression files.</td>
<td>exp_file: The filename of the data to be opened. The default is set to TRUE, which displays a file dialog that allows you to select one or more expression files.</td>
</tr>
<tr>
<td>lod: The defined detection limit for Ct expression data; and any expression data of genes or samples are assigned the LoD if their values are greater than the LoD value specified. The default LoD is set to a Ct of 24.</td>
<td>lod: The defined detection limit for Ct expression data; and any expression data of genes or samples are assigned the LoD if their values are greater than the LoD value specified. The default LoD is set to a Ct of 24.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initialize the EXP object from one or more linear expression data files.</th>
<th>exp &lt;- readLinearExp (exp_file=TRUE, lod=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp_file: The filename of the data to be opened. The default is set to TRUE, which displays a file dialog that allows you to select one or more expression files.</td>
<td>exp_file: The filename of the data to be opened. The default is set to TRUE, which displays a file dialog that allows you to select one or more expression files.</td>
</tr>
<tr>
<td>lod: The defined detection limit for mRNAseq expression data; and any expression data of genes or samples are assigned the LoD if their values are</td>
<td>lod: The defined detection limit for mRNAseq expression data; and any expression data of genes or samples are assigned the LoD if their values are</td>
</tr>
</tbody>
</table>
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Function Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>greater than the LoD value specified. The default linear value of lod is 1.</td>
<td>readData(file=TRUE)</td>
</tr>
<tr>
<td>Read any fluidigmSC data object (obj, such as EXP, HC, ANOVA, and PCA) file, a gene list file, and a sample list file for downstream analysis.</td>
<td>readExpObject(exp_file=TRUE)</td>
</tr>
<tr>
<td>Load one or more EXP object data files and create a merged EXP object.</td>
<td>saveData(data, file=TRUE)</td>
</tr>
<tr>
<td>Save any fluidigmSC data object (obj, such as EXP, HC, ANOVA, and PCA), a gene list, and a sample list to a file for future use.</td>
<td>data: The name of the file to be saved Can be a fluidigmSC object, a selected sample list, or a gene list. file:. The default is set to TRUE, which displays a file dialog that allows you to select one or more expression files.</td>
</tr>
</tbody>
</table>

### Managing Expression Data

#### Assign a Group ID

<table>
<thead>
<tr>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp &lt;- updateSampleListFromName (exp,sep=&quot; &quot;, prefix=TRUE)</td>
</tr>
<tr>
<td>data: The name of the file to be saved Can be a fluidigmSC object, a selected sample list, or a gene list. file:. The default is set to TRUE, which displays a file dialog that allows you to specify a file name and location to safe the defined data.</td>
</tr>
<tr>
<td>exp: A given EXP object.</td>
</tr>
<tr>
<td>sep: The character that the function will use to separate the sample group ID from the sample name. May also be &quot;-&quot;</td>
</tr>
</tbody>
</table>

---

SINGuLAR Analysis Toolset User Guide
### Managing Expression Data

<table>
<thead>
<tr>
<th>Update the EXP object</th>
<th>exp &lt;- updateSampleListFromFile (exp, sample_list_file=TRUE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the sample list of an EXP object using a sample list file as input and return the updated EXP object.</td>
<td>exp: The object to update. sample_list_file: The default is set to select a file through a file dialog. This argument can be set to the file name with file path if the file name is known. A sample list file contains two columns with headers SampleID and GroupID and has sample and group information in rows below.</td>
</tr>
<tr>
<td>Update the sample list of an EXP object using a sample list as input, and return the updated EXP object.</td>
<td>exp &lt;- updateSampleListFromList(exp, &quot;sample_list&quot;)</td>
</tr>
<tr>
<td>Update the gene list of an EXP object using a gene list as input, and return the updated EXP object.</td>
<td>exp &lt;- updateGeneListFromList (exp, &quot;gene_list&quot;)</td>
</tr>
<tr>
<td>Update an expression object with a new gene list from a gene list file.</td>
<td>exp &lt;- updateGeneListFromFile (exp, gene_list_file=TRUE)</td>
</tr>
</tbody>
</table>

- **prefix**: The default indicates that the group ID is a prefix of the sample name. If this argument is set to FALSE, the group ID will be assumed to be the suffix.
Appendix A: The List of Functions for Gene Expression

group, such as in HC and PCA plots.
can also be set to the filename with file path if it is known.
A gene list file contains two columns with headers GroupID and GroupID and has sample
and group information in rows below.

<table>
<thead>
<tr>
<th>Remove or Retain Samples and Sample Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove a sample from the expression's sample list and return the EXP object with the updated sample list.</td>
</tr>
<tr>
<td>- exp: A given Exp object.</td>
</tr>
<tr>
<td>- &quot;sample_name&quot;: The name of a sample to be removed from the list.</td>
</tr>
<tr>
<td>Remove a group of samples from the expression's sample list and return the updated EXP object with a modified sample list.</td>
</tr>
<tr>
<td>- exp: A given Exp object</td>
</tr>
<tr>
<td>- &quot;sample_group&quot;: The name of the sample group to be removed from the list.</td>
</tr>
<tr>
<td>Retain only the samples of a given group from the expression's sample list and return the updated EXP object with a modified sample list.</td>
</tr>
<tr>
<td>- exp: The object to update.</td>
</tr>
<tr>
<td>- &quot;sample_group&quot;: The name of the sample group to be retained in the sample list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remove or Retain Genes and Gene Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given an EXP object defined by the first argument (gene_name), remove a gene from the EXP gene list and return the EXP object with the updated gene list.</td>
</tr>
<tr>
<td>- exp: A given EXP object.</td>
</tr>
<tr>
<td>- &quot;gene_name&quot;: The name of a gene to be removed from the list.</td>
</tr>
<tr>
<td>Given an EXP object from linear expression data and a linear expression detection threshold, remove all genes in the expression</td>
</tr>
</tbody>
</table>
### Managing Expression Data

- **Given an EXP object from Ct expression data and a Ct detection threshold, remove all genes in the expression data that are greater than the Ct threshold.**

  ```r
  exp <- removeGeneByCtExp (exp, ct_threshold)
  ```

  - exp: A given EXP Object.
  - ct_threshold: The threshold used to compare and remove genes. Can be set to equal any number, such as ct_threshold=26.

- **Remove a group of genes from the expression's gene list and return the updated EXP object with a modified gene list.**

  ```r
  exp <- removeGeneGroup (exp, "gene_group")
  ```

  - exp: A given EXP Object.
  - "gene_group": The name of the gene group to be removed from the list.

- **Retain only the genes of a give group from the expression's gene list and return the updated EXP object with a modified gene list.**

  ```r
  exp <- retainGeneGroup (exp, "gene_group")
  ```

  - exp: A given EXP Object.
  - "gene_group": The name of the gene group to be retained in the list.

### Change Group Colors and Symbols

- **If your EXP object has more than one sample group, the SINGuLAR Analysis Toolset assigns a default color and symbol scheme to each sample group. You can customize the color and symbol schemes for each sample group, and the customized color and symbol will be used in the PCA Score plot and HC heatmap.**

  ```r
  exp<- setSampleGroupColorAndSymbols(exp)
  ```

  - exp: A given EXP Object.

  An interactive window appears that allows user to click on colors and symbols for each sample group. To save your choices, click **OK**.

  To change the settings back to the default values, click **Reset**. You can then select
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>These colors and symbols for all plots in the working session. This updated exp can be saved.</td>
<td>exp &lt;- setGeneGroupColorAndSymbols(exp) exp: A given EXP object with gene group annotation.</td>
</tr>
<tr>
<td>To return to the previously assigned colors, click Cancel.</td>
<td>exp &lt;- clearGeneGroupColorAndSymbols(exp) exp: A given EXP object with gene group annotation.</td>
</tr>
<tr>
<td>To return to the R console, close the window.</td>
<td></td>
</tr>
<tr>
<td>If you customize the color and symbol schemes for the sample groups, you can restore the default color and symbol settings for sample groups.</td>
<td>exp &lt;- setGeneGroupColorAndSymbols(exp) exp: A given EXP object with sample group annotation.</td>
</tr>
<tr>
<td>Returns an updated exp that will now use these colors and symbols for all plots in the working session. This updated exp can be saved.</td>
<td></td>
</tr>
<tr>
<td>If your EXP object has more than one gene group, the SINGuLAR Analysis Toolset assigns a default color and symbol scheme to each gene group. You can customize the color and symbol schemes for each gene group, and the customized color and symbol will be used in the PCA Loading plot and HC heatmap.</td>
<td>exp &lt;- clearGeneGroupColorAndSymbols(exp) exp: A given EXP object with sample group annotation.</td>
</tr>
<tr>
<td>Returns an updated exp that will now use these colors and symbols for all plots in the working session. This updated exp can be saved.</td>
<td></td>
</tr>
<tr>
<td>If you customize the color and symbol schemes for the gene groups, you can restore the default color and symbol settings for gene groups.</td>
<td>exp &lt;- clearGeneGroupColorAndSymbols(exp) exp: A given EXP object with gene group annotation.</td>
</tr>
<tr>
<td>Returns an updated exp that will now use these colors and symbols for all plots in the working session. This updated exp can be saved.</td>
<td></td>
</tr>
<tr>
<td><strong>Reset Sample and Gene Lists</strong></td>
<td></td>
</tr>
<tr>
<td>Replace an EXP object’s current list of genes with its original list of genes and return updated EXP.</td>
<td>exp &lt;- resetGeneList (exp) exp: A given EXP object.</td>
</tr>
</tbody>
</table>
## Managing Expression Data

Replace an EXP object’s current list of samples with its original list of samples, and return updated EXP.

```r
exp <- resetSampleList(exp)
```

**exp**: A given EXP object.

### Manage Outliers

Add a particular outlier back for analysis, and return an updated EXP.

```r
exp <- restoreOutlierFromName (exp, "sample_name")
```

- **exp**: The EXP object to be restored.
- **sample_name**: Name of the outlier sample to be restored.

Add all outliers defined by a sample list back for analysis, and return updated EXP.

```r
exp <- restoreOutlierFromList (exp, "sample_list")
```

- **exp**: The EXP object to be restored.
- **sample_list**: The list of outliers to be restored.

Restore all outliers back into the working sample list for downstream analysis, and return updated EXP.

```r
exp <- restoreAllOutlier (exp)
```

**exp**: The EXP object to be restored.

Move a sample identified by the name to the outlier list for a given EXP object, and return the updated EXP. The sample is removed from downstream analysis.

```r
exp <- addOutlierFromName (exp, "sample_name")
```

- **exp**: A given EXP object.
- **"sample_name"**: Name of the outlier sample to be added.

Based on the defined sample list of outliers, move outlier samples from the sample list to the outlier list for the given EXP, and return the updated EXP object.

```r
exp <- addOutlierFromList (exp, "sample_list")
```

- **exp**: A given EXP object.
- **"sample_list"**: The list of outliers to be added.
## Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Function Description</th>
<th>R Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open a list of samples defined in a file, move them from the sample list to the outlier list for the given EXP, and return the updated EXP object.</strong></td>
<td><code>exp &lt;- addOutlierFromFile (exp)</code></td>
</tr>
<tr>
<td></td>
<td><code>exp: A given EXP object.</code></td>
</tr>
<tr>
<td><strong>Update the LoD</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Return a new EXP object after updating the given EXP object’s limit of detection (LoD) and recalculating the Log2Ex expression data.</strong></td>
<td><code>exp &lt;- updateLod(exp, new_lod)</code></td>
</tr>
<tr>
<td>For mRNA Seq data, the LoD is defined in the linear domain. For qPCR data, the LoD is defined in the Ct domain.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Merge Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Merge two expression objects that share either the same sample list or the same gene list (but not both).</strong></td>
<td><code>exp &lt;- mergeExpData(exp1, exp2, overlap_flag = -1)</code></td>
</tr>
<tr>
<td>Returns a single expression object containing expanded list of either samples or genes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Merge two gene lists and return a merged gene list.</strong></td>
<td><code>merged_gene_list &lt;- mergeGeneList (gene_list1, gene_list2, method=&quot;union&quot;)</code></td>
</tr>
<tr>
<td>You can replace gene_list1 and gene_list2 with sample list names separated by a comma.</td>
<td></td>
</tr>
</tbody>
</table>
that can be merged as "union", "intersect", or "subtract".

For the "union" method, the merged sample list includes all genes from both source lists.

For the "intersect" method, the merged sample list includes only genes that are common between both source lists.

For the "subtract" method, the merged sample list includes only genes that are in list1 but not in list2.

### Displaying Expression Data

Given an EXP object defined by the first argument, prints a summary of that expression object containing the number of original samples, number of original genes, the number of working genes, number of working sample groups, number of working samples in each group, and the number of outliers.

<table>
<thead>
<tr>
<th>Merge two sample lists and return a merged sample list.</th>
<th>merged_sample_list &lt;- mergeSampleList(sample_list1, sample_list2, method=&quot;intersect&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can replace gene_list1 and gene_list2 with gene list names separated by a comma that can be merged as &quot;union&quot;, &quot;intersect&quot;, or &quot;subtract&quot;.</td>
<td>For the &quot;union&quot; method, the merged gene list includes all genes from both source lists.</td>
</tr>
<tr>
<td>For the &quot;intersect&quot; method, the merged gene list includes only genes that are common between both source lists.</td>
<td>For the &quot;subtract&quot; method, the merged gene list includes only genes that are in list1 but not in list2.</td>
</tr>
</tbody>
</table>

\[
\text{summaryExp(exp)}
\]

\[
\text{summary_table <- summaryExp(exp)}
\]

\[
\text{exp: The object to display.}
\]
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Displays the information or when set to create a list, this can be saved as text delimited file.</th>
<th>boxPlotBySamples (exp, by_sample=TRUE, sample_group=&quot;all&quot;, max_sample_num=100, &quot;title&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given an EXP object and the defined a set of genes, create a Box plot of gene expression for individual samples or individual sample groups. Sample names are along the x axis, colored by default or customized group color. Log2Ex values are on the y axis.</td>
<td><em>by_sample</em>: The default (TRUE) displays gene profiles for each sample. FALSE displays the gene profile for the sample group.</td>
</tr>
<tr>
<td><em>sample_group</em>: The default is set to display all sample groups. When set to the name of a sample group, it displays the gene profile for that particular sample group.</td>
<td></td>
</tr>
<tr>
<td><em>max_sample_number</em>: The maximum number of samples allowed to print. When the maximum is reached, you are prompted to either (a) stop the creation process to cancel the display or (b) continue. To continue graphing until all samples are plotted, type in a number and press Enter.</td>
<td></td>
</tr>
<tr>
<td>&quot;title&quot;: Title of the plot. If missing, the function will provide a default &quot;Boxplot of Gene Expression by Samples (Genes = N)&quot; where N is the number of genes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Given an EXP object and a defined a set of genes, create Violin plots of gene expression for individual samples or individual sample groups. Sample names are along the x axis, colored by default or customized group color. Log2Ex values are on the y axis.</th>
<th>violinPlotBySamples (exp, by_sample=TRUE, sample_group=&quot;all&quot;, max_sample_num=100, &quot;title&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>by_sample</em>: The default (TRUE) displays gene profiles for each sample. FALSE displays the gene profile for the sample group.</td>
<td></td>
</tr>
<tr>
<td><em>sample_group</em>: The default is set to display all sample groups. When set to the name of a sample group, it displays the gene profile for that particular</td>
<td></td>
</tr>
</tbody>
</table>
### Displaying Expression Data

**max_sample_number**: The maximum number of samples allowed to print. When the maximum is reached, you are prompted to either (a) stop the creation process to cancel the display or (b) continue. To continue graphing until all samples are plotted, type in a number and press Enter.

**"title"**: Title of the plot. If missing, the function will provide a default "Violin Plot of Gene Expression by Samples (Genes = N)" where N is the number of genes.

<table>
<thead>
<tr>
<th>Given an EXP object, displays histograms of the average gene expression values for each sample group, bucketed in bins of 2 of Log2Ex values. Also displays a scatter plot of Log2Ex values between samples groups, where the axis can be traced out to find the group on that axis, and you can see the Correlation Coefficient between any two sample groups at the opposing intersection of group names.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairwiseScatterPlotBetweenSampleGroups(exp, &quot;title&quot;)</td>
</tr>
<tr>
<td>exp: A given EXP object.</td>
</tr>
<tr>
<td>&quot;title&quot;: Title of the plot. If missing, the function will provide a default &quot;Pairwise Comparison of Sample Groups&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Given an EXP object and a defined gene limit of detection (LoD), calculate how many genes in each sample will be detected (expression value above threshold) or will drop out (expression value below threshold). For mRNA Seq data, the LoD is defined in the linear domain. For qPCR data, the LoD is defined in the Ct domain. This command will display a graph with number of detected genes on the y axis and sample names on the x axis, and a bar graph representing gene numbers for each sample.</th>
</tr>
</thead>
<tbody>
<tr>
<td>analyzeGeneDetection(exp, threshold=1, dropout_flag=FALSE, display=TRUE, &quot;title&quot;)</td>
</tr>
<tr>
<td>exp: The object containing the detected genes.</td>
</tr>
<tr>
<td>threshold: The defined threshold for gene detection. This represents a linear value for mRNAseq data and a Ct value for qPCR data.</td>
</tr>
<tr>
<td>dropout_flag: The default is set to FALSE to calculate detection. When set to TRUE will calculate the gene drop_out by sample</td>
</tr>
<tr>
<td>display: The logical value to determine the result to be displayed. The default</td>
</tr>
</tbody>
</table>
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boxPlotByGenes (exp, gene_list_file=FALSE, gene_list=FALSE, num_gene_per_plot=100, max_gene_num=200, &quot;title&quot;)</td>
<td>Given an EXP object and a set of genes, create Box plots to display gene expression profiles across samples for each gene in a gene list. The gene list can be in the EXP object (by default), from a file, or from another source. Log2Ex values are on the y axis and gene names are along the x axis, which each sample group having an individual plot for each gene. Sample groups are colored by group color set by default or chosen.</td>
</tr>
</tbody>
</table>

- **exp**: The object containing the gene list to be displayed (by default).
- **gene_list_file**: The default is set to FALSE. To use a gene list from a file, set this argument to TRUE. A GUI window appears, allowing the user to choose the gene list from a file.
- **gene_list**: The default is set to FALSE. To use a different gene list, set this argument to the gene list.
- **num_gene_per_plot**: Number of genes per plot. If there are more genes than the maximum allowed in a plot, another plot will be created to display the excess. The default is 100 genes.
- **max_gene_num**: The maximum number of genes allowed to print. When the maximum is reached, the user is prompted to either 1) stop the creation process or 2) continue. The user will need to type a number and press enter. If stop is chosen, the display is cancelled. If continue is chosen, the function continues graphing until all samples are plotted.
- **"title"**: Title of the plot. If missing, the function will provide a default "Boxplot"
Given an EXP object and a set of genes, create Violin plots to display the gene expression profile (across samples) for each gene in a gene list.

The gene list can be the gene list in the EXP object, a gene list from a file, or a gene list.

Log2Ex values are on the y axis and gene names are along the x axis, which each sample group having an individual plot for each gene. Sample groups are colored by group color set by default or chosen.

### Summary of PCA Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform PCA on a given EXP object, calculate the PCA gene score and displays results.</td>
<td>pca &lt;- PCA (exp, rank_component_num=3, display_plots = TRUE)</td>
<td>exp: The EXP object.</td>
</tr>
</tbody>
</table>
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PCA Score Plot</strong>: Each point is a sample. PC1 scores on x and PC2 scores on y.</td>
<td><strong>rank_component_num</strong>: The number of principal components used to calculate the PCA scores of genes. (The default value is 3.)</td>
</tr>
<tr>
<td><strong>PCA Loading Plot</strong>: Each point is a gene. PC1 scores on x and PC2 scores on y.</td>
<td><strong>display_plots</strong>: When set to TRUE this will display all plots. This is the default. If the display_plot is set to FALSE, only the new PCA analysis object will be returned.</td>
</tr>
</tbody>
</table>

Apply an existing PCA model to a expression data of new samples. A PCA model must have been calculated on an EXP object using command:

```r
pca <- PCA(exp)
```

Then a second or new exp can be called up using:

```r
exp <- readExpObject()
```

The second expression object should share the same set of genes as those in the expression object used to calculate the PCA model.

Returns a new PCA object containing the PCA calculations for the second exp object as well as three 2-dimensional plots:

- **PCA Scree**: Variance on y axis and first ten PC scores on x axis. Visually identify how many PC scores contain most of the variance.
- **PCA Score Plot**: Each point is a sample. PC1 scores on x and PC2 scores on y.
- **PCA Loading Plot**: Each point is a gene. PC1 scores on x and PC2 scores on y.

```r
pca <- applyPCA (exp, pca, display_plot=TRUE, include_outliers=FALSE)
```

- **exp**: The second exp object to which the PCA model will be applied.
- **pca**: The pca model calculated from the original exp object.
- **display_plot**: When set to TRUE this will display all plots. This is the default. If the display_plot is set to FALSE, only the new PCA analysis object will be returned.
- **include_outliers**: If TRUE, outliers are added to the PCA. If FALSE (default), outliers from the second exp are not added to the PCA.

Display a PCA Score plot with an optional graphical user interface for locating samples of interest.

The plot is interactive:

- **Circle**: Draw a circle by connecting dots on the plot. Once a circle is formed, the plot

```r
sample_list <- displayPCAScore (pca= fldm_pca, x_axis = 1, y_axis = 2, locate=TRUE, "title")
```

- **pca=fldm_pca**: The default is to call on the autoAnalysis generated pca object.
This can be set to equal any pca object.

- **x_axis**: The default is PC score one. This is the PC score used for this axis.
- **y_axis**: The default is PC score 2. This is the PC score used for this axis.
- **z_axis**: The default is PC score three. This is the PC score used for this axis.
- **locate**: The default is TRUE and will display an interactive plot that allows you to click anywhere on the 3D plot and

```r
sample_list <- display3DPCAScore (pca, x_axis = 1, y_axis = 2, z_axis = 3, locate = TRUE)
```

- **pca**: The default is call on the autoAnalysis generated PCA object. This can be set to equal any pca object.
- **x_axis**: The default is PC score one. This is the PC score used for this axis.
- **y_axis**: The default is PC score two. This is the PC score used for this axis.
- **z_axis**: The default is PC score three. This is the PC score used for this axis.
- **locate**: The default is TRUE and will display an interactive plot that allows you to click anywhere on the 3D plot and

```r
sample_list <- display3DPCAScore (pca, x_axis = 1, y_axis = 2, z_axis = 3, locate = TRUE)
```
**Done.** Indicates that you are done locating samples of interest.

Returns a sample list of selected samples. This can be created as an object list and saved or displayed in the R command line, if not included.

You can also save an image copy it to the clipboard, or print it:

To save the image, choose **File > save as**, and then click any of the following: Metafile, Postscript, PDF, Png, Bmp, TIFF, or Jpeg. If Jpeg, additionally click the quality (50%, 75%, or 100%).

To copy to clipboard, choose **File > Copy to the clipboard**, and then click either as a Bitmap or as a Metafile.

To print, choose **File > Print.**

| Displays a PCA Loading plot with an optional graphical user interface for locating genes of interest | gene_list <- displayPCALoading(pca=fldm_pca, x_axis = 1, y_axis = 2, locate=TRUE, "title") |
| Circle. Draw a circle by connecting dots on the plot. Once a circle is formed, the plot displays each gene ID inside the circle. |  pca=fldm_pca: The default is to call on the autoAnalysis generated pca object. this can be set to equal any pca object. |
| Point. Click a gene to display its ID. |  x_axis: The default is PC score one. This is the PC score used for this axis. |
| Clear. Returns the plot to its original state. |  y_axis: default is PC score 2. This is the PC score used for this axis. |
| Save. Save selected genes to a file. |  locate: The default is TRUE and will display an interactive plot that allows user selection of individual samples on the plot. |
| Done. Indicates that you are done locating genes of interest. |  "title": Title of the plot. If missing, the function will provide a default "PCA Loading Plot". |

Returns a sample list of selected samples. This can be created as an object list and saved or displayed in the R command line, if not included.

You can also save an image copy it to the clipboard, or print it:

drag the cursor to turn the plot. You can also select samples on the plot and save the plot.
To save the image, choose **File > save as**, and then click any of the following: Metafile, Postscript, PDF, Png, Bmp, TIFF, or Jpeg. If Jpeg, additionally click the quality (50%, 75%, or 100%).

To copy to clipboard, choose **File > Copy to the clipboard**, and then click either as a Bitmap or as a Metafile.

To print, choose **File > Print**.

<table>
<thead>
<tr>
<th>From a given PCA object, return a defined number of top-ranked PCA genes.</th>
<th>pca_gene_list &lt;- getTopPCAGenes (pca, top_gene_num=100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>￭ pca : Previously created pca object containing gene list.</td>
<td>￭ pca : Previously created pca object containing gene list.</td>
</tr>
<tr>
<td>￭ top_gene_num: The default is the top 100 genes, sorted from high to low gene values calculated from PC scores 1,2,&amp;3, however this can be set to equal any desired number.</td>
<td>￭ top_gene_num: The default is the top 100 genes, sorted from high to low gene values calculated from PC scores 1,2,&amp;3, however this can be set to equal any desired number.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Given a PCA object, display a graph with the PCA score of genes on the y-axis and accumulated gene numbers on the x-axis, sorted by scores (by default from large to small) calculated from PC scores 1,2,&amp;3.</th>
<th>displayPCAGeneScores (pca, top_gene_num = -1,&quot;title&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>￭ pca: Previously created pca object containing gene list.</td>
<td>￭ pca: Previously created pca object containing gene list.</td>
</tr>
<tr>
<td>￭ top_gene_num: The maximum number of ranked genes to be plotted. If the value is negative (such as the default value of -1), all genes are plotted. This can be changed to equal any number and will display no more than chosen number of genes.</td>
<td>￭ top_gene_num: The maximum number of ranked genes to be plotted. If the value is negative (such as the default value of -1), all genes are plotted. This can be changed to equal any number and will display no more than chosen number of genes.</td>
</tr>
<tr>
<td>￭ &quot;title&quot;: Title of the plot. If missing, the function will provide a default &quot;PCA Gene Score Plot&quot;.</td>
<td>￭ &quot;title&quot;: Title of the plot. If missing, the function will provide a default &quot;PCA Gene Score Plot&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display the Scree plot for the PCA object saved from the autoAnalysis. You can also display the Scree plot with an explicitly-</th>
<th>displayPCAScree (pca=fldm_pca, &quot;title&quot;)</th>
</tr>
</thead>
</table>
### Appendix A: The List of Functions for Gene Expression

| defined PCA object. | - pca=fldm_pca: The PCA analysis result by `autoAnalysis()` can be changed to any pca object. |
| - “title”: Title of the plot. If missing, the function will provide a default "PCA Scree Plot". |

#### Summary of ANOVA Functions

An ANOVA object, containing the result of ANOVA analysis. The ANOVA object is a list of the following data frames:

1. **obj_type**: type of object, "ANOVA"
2. **p_values**: the first column holds gene names, and the second column holds p-values for all groups, followed by p-values for each pairwise groups. The remaining columns hold average expression values for each sample group.
3. **sample_list**: the sample names and group IDs used for this analysis.
4. **gene_list**: the gene names and group IDs used for this analysis.

The function also displays the summary plot for number of significantly expressed genes in pairwise sample groups.

| Display a graph with p-values on the y-axis and the number of genes with p-values less than or equal to that number (cumulative) on the x-axis—and sorted according to p-values (small to large). |
| displayANOVAPValues (anova, top_gene_num=-1, pvalue_threshold=1, sample_group1=FALSE, sample_group2=FALSE, "title") |
| - anova: Previously generated ANOVA object from which p-values will be displayed. |
| - top_gene_num: The maximum number of ranked genes to be plotted. If the value is negative, all genes are plotted. The default is set to -1. |
Displaying Expression Data

- **pvalue_threshold**: The threshold to measure statistical significance. The threshold must be between 0 and 1. The default is set to 1.
- **sample_group1, sample_group2**: The sorting instructions for genes:
  
  To Sort genes by overall p-value of ANOVA, set these value to **FALSE**.
  
  To Sort genes by p-value of T Test in two sample groups, set these values equal to the group names of interest in quotes:
  
  ```r
  sample_group1 = "group_name1",
  sample_group2 = "group_name2"
  ```
- **"title"**: Title of the ANOVA graph. If missing, the function will provide a default title.

Get the top-ranked ANOVA genes (sorted by ANOVA p-values) from an ANOVA object for all sample groups or for a pair of sample groups.

When two sample groups are to be compared, `sample_group1` and `sample_group2` must be provided. If only one of the two is provided, an error will result.

If `pvalue_threshold` and `top_gene_num` are specified, only those genes with p-values equal to or less than the threshold are displayed. The number of genes returned will not exceed `top_gene_num`.

```r
anova_gene_list <- getTopANOVAGenes
(anova, top_gene_num=100,
pvalue_threshold=0.05,
sample_group1=FALSE,
sample_group2=FALSE)
```

- **anova**: Previously generated ANOVA object from which p-values will be displayed.
- **top_gene_num**: Number of ranked genes to be returned. The default is set to 100.
- **pvalue_threshold**: The desired threshold to define statistical significance. The threshold must be between 0 and 1. The default is set to 0.05.
- **sample_group1, sample_group2**: The sorting instructions for genes:
  
  To Sort genes by overall p-value of ANOVA, set these value to **FALSE**.
  
  To Sort genes by p-value of T Test in two sample groups, set these values equal to
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Display the number of significantly expressed genes (less than the given <code>pvalue_threshold</code>) for each pair of sample groups.</th>
<th><code>pairwiseANOVASummary</code> (<code>anova</code>, <code>pvalue_threshold</code>=0.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <code>anova</code>: Previously generated ANOVA object from which p-values will be displayed.</td>
<td></td>
</tr>
<tr>
<td>- <code>pvalue_threshold</code>: The standard level of significance to be used. By default this is set to 0.05.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Find the genes with mean expression ratios between two given sample groups that are equal to or greater than a given fold-change threshold and the difference is statistically significant.</th>
<th><code>foldChangeAnalysis</code> (<code>anova</code>, <code>sample_group1</code>, <code>sample_group2</code>, <code>foldchange_threshold</code> = 2, <code>pvalue_threshold</code> = 0.05, <code>display_plot</code> = TRUE, <code>locate</code> = FALSE, &quot;title&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <code>anova</code>: Previously generated ANOVA object from which p-values will be displayed.</td>
<td></td>
</tr>
<tr>
<td>- <code>sample_group1</code>, <code>sample_group2</code>: The sorting instructions for genes:</td>
<td></td>
</tr>
<tr>
<td>To Sort genes by overall p-value of ANOVA, set these value to <code>FALSE</code>.</td>
<td></td>
</tr>
<tr>
<td>To Sort genes by p-value of T Test in two sample groups, set these values equal to the group names of interest in quotes:</td>
<td></td>
</tr>
<tr>
<td><code>sample_group1</code> = &quot;group_name1&quot;, <code>sample_group2</code> = &quot;group_name2&quot;</td>
<td></td>
</tr>
<tr>
<td>- <code>foldchange_threshold</code>: Only those genes whose mean expression values are equal to or greater than this fold change threshold will be returned. The default is a 2-fold change.</td>
<td></td>
</tr>
<tr>
<td>- <code>pvalue_threshold</code>: Only those genes with p-values equal or less than the threshold</td>
<td></td>
</tr>
</tbody>
</table>

```sample_group1 = "group_name1", sample_group2 = "group_name2"```
### Displaying Expression Data

- Displaying Expression Data are displayed. The default is 0.05, so all genes will be displayed up to but not greater than the top_gene_number.
- display_plot: When set to TRUE this will display a Volcano plot. If FALSE, no plot will be displayed.
- locate: The default is set to FALSE. When set to TRUE, user will be able to interactively select interested genes in the plot.
- "title": Title of the plot.

### Summary of HC Functions

Given an EXP object, conduct unsupervised hierarchical clustering (HC) analysis for expression data.

```r
hc <- HC (exp,
    dendrogram="both",
    color_scheme="blue_white_red",
    display="global_z_score",
    display_sample_names=TRUE,
    display_gene_names=TRUE,
    heatmap_display=TRUE)
```

- **exp**: The EXP object.
- **dendrogram**: Clustering method. The default to set to cluster both sample and gene. Options are: "both", "sample", "gene", and "none".
- **color_scheme**: The predefined color scheme in HC. The default is "blue_white_red". If a different color scheme is preferred, set this argument to "" and the function helps you create a desired color scheme.
- **display**: The method for the heatmap display. Options are: "global_z_score", "gene_z_score", and "expression".
- **display_sample_names**: The default (TRUE) is set to display sample names. If FALSE, sample names will not be
### Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Display Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display_gene_names</td>
<td>The default (TRUE) is set to display gene names in the HC plot. If FALSE, gene names will not be displayed.</td>
</tr>
<tr>
<td>heatmap_display</td>
<td>The default (TRUE) is set to display the heatmap. If FALSE, the heatmap will not be displayed.</td>
</tr>
</tbody>
</table>

Given HC and EXP objects, append the expression data from the EXP object to the existing HC object and return an updated HC object.

```r
hc <- applyHC(hc, exp, 
  cluster = TRUE, 
  color_scheme = "blue_white_red", 
  display = "global_z_score", 
  display_sample_names = TRUE, 
  display_gene_names = TRUE)
```

**hc**: The hierarchical clustering object.

**exp**: The expression object.

**cluster**: The default is set to cluster the additional expression data. When set to FALSE, no clustering will be performed and the data will be added according to the order defined in the EXP.

**color_scheme**: The predefined color scheme in HC. The default is "blue_white_red". If a different color scheme is preferred, set this argument to "" and the function helps you create a desired color scheme.

**display**: The method for the heatmap display. Can be "global_z_score", "gene_z_score", or "expression".

**display_sample_names**: The default is set to display sample names in the HC plot. When set to FALSE, sample names would not be displayed.

**display_gene_names**: The default is set to display gene names in the HC plot. When set to FALSE, gene names would not be displayed.
| Display the results of HC analysis. | displayHC (hc,  
color_scheme="blue_white_red",  
display="global_z_score",  
display_sample_names=TRUE,  
display_gene_names=TRUE,  
sample_name_trimed_size=15,  
gene_name_trimed_size=10, "title") |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ hc: The object containing the hierarchical clustering information.</td>
<td></td>
</tr>
<tr>
<td>▪ color_scheme: The predefined color scheme in HC. The default is &quot;blue_white_red&quot;. If a different color scheme is preferred, set this argument to &quot;&quot; and the function helps you create the desired color scheme.</td>
<td></td>
</tr>
<tr>
<td>▪ display: The method for the heatmap display. Options are: &quot;global_z_score&quot;, &quot;gene_z_score&quot;, and &quot;expression&quot;.</td>
<td></td>
</tr>
<tr>
<td>▪ display_sample_names: The default is set to display sample names in the HC plot. When set to FALSE, sample names will not be displayed.</td>
<td></td>
</tr>
<tr>
<td>▪ display_gene_names: The default is set to display gene names in the HC plot. When set to FALSE, gene names will not be displayed.</td>
<td></td>
</tr>
<tr>
<td>▪ sample_name_trimed_size: The default is set to the maximum length of characters for displaying any sample name in the HC plot. The default is set as 15 characters.</td>
<td></td>
</tr>
<tr>
<td>▪ gene_name_trimed_size: The default is set to the maximum length of characters for displaying any gene name in the HC plot. The default is set as 10 characters.</td>
<td></td>
</tr>
<tr>
<td>▪ &quot;title&quot;: Title of the Heatmap of Expression display, where the term &quot;display&quot; replaced with the type selected. If missing, the function will provide a</td>
<td></td>
</tr>
<tr>
<td>Function Description</td>
<td>Code Example</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Export the heatmap of a given hierarchical clustering analysis to a tab-delimited text file with the values selected in the display. These values are in the same location as the values in the heatmap. | `exportHCHeatmap(hc, display="global_z_score")`  
  - `hc`: The HC object.  
  - `display`: The method for the heatmap display. Options are: "global_z_score", "gene_z_score", and "expression". |
| Gets gene clusters by threshold. Since the SINGuLAR™ system clusters genes by the Pearson Correlation method, this function enables you to identify a group of co-expressed genes by the correlation coefficient threshold. | `hc_gene_list <- getGeneClusterByThreshold(hc, threshold=TRUE, min_gene_num=3, include_subclusters=FALSE)`  
  - `hc`: The HC object.  
  - `threshold`: The threshold can be either a logical variable or a number. If the logical variable is set to `TRUE` (default), the function displays the gene cluster dendrogram to let users select the gene correlation threshold from the plot.  
  - `min_gene_num`: The minimum number of genes required for a gene cluster. The default is 3.  
  - `include_subclusters`: A logical variable to determine whether to include selected gene clusters, including any of their sub-clusters that meet two selection criteria described in the description section. When set to `FALSE`, only the top cluster will be listed. If `TRUE`, this will also display all subclusters. |
| Interactively identify gene clusters using the gene cluster dendrogram and return the identified cluster(s) as a gene with cluster ID. | `select_gene_clusters <- identifyGeneClusters(hc)`  
  - `hc`: The HC object. |
| Get sample clusters by a threshold. The threshold is the minimum fold change | `hc_sample_list <- getSampleClusterByThreshold(hc, threshold=TRUE, min_gene_num=3, include_subclusters=FALSE)`  
  - `hc`: The HC object.  
  - `threshold`: The threshold can be either a logical variable or a number. If the logical variable is set to `TRUE` (default), the function displays the gene cluster dendrogram to let users select the gene correlation threshold from the plot.  
  - `min_gene_num`: The minimum number of genes required for a gene cluster. The default is 3.  
  - `include_subclusters`: A logical variable to determine whether to include selected gene clusters, including any of their sub-clusters that meet two selection criteria described in the description section. When set to `FALSE`, only the top cluster will be listed. If `TRUE`, this will also display all subclusters. |
### Displaying Expression Data

between any two samples in the cluster.

<table>
<thead>
<tr>
<th>threshold=TRUE, min_sample_num=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>h: The HC object.</td>
</tr>
<tr>
<td>threshold: The threshold can be either a logical variable or a number. If the logical variable is set to TRUE (default), the function displays the sample cluster dendrogram to let users select the sample correlation threshold from a plot. You can also set the threshold to an actual value instead of to TRUE or FALSE.</td>
</tr>
<tr>
<td>min_sample_num: The minimum number of samples required for a sample cluster.</td>
</tr>
</tbody>
</table>

Interactively identify sample clusters using the sample cluster dendrogram and return the identified cluster(s) as a sample list with cluster ID.

<table>
<thead>
<tr>
<th>hc_sample_list &lt;- identifySampleClusters (hc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc: The HC object.</td>
</tr>
</tbody>
</table>

Does not allow nested clusters. If a selected cluster is inside another cluster, the function automatically removes the previous outer selection. If a selected cluster is outside another cluster, the function automatically removes the previous inner selection.

#### Summary of Correlation Functions

Given a CORR object and a target gene with a defined Correlation Coefficient threshold, this function displays the gene expression profile of co-expressed genes for the samples in selected sample groups.

<table>
<thead>
<tr>
<th>displayCorrGenes(corr, query_gene_name = &quot;&quot;, corr_threshold = 0.5, corr_pattern = &quot;positive&quot;, &quot;title&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>corr: A given CORR object.</td>
</tr>
<tr>
<td>query_gene_name: A target gene name in used in gene correlation analysis, if query_gene_name is missing, it will retrieve the first gene in the table of corr_gene_result in CORR object as query_gene_name.</td>
</tr>
<tr>
<td>corr_threshold: The Correlation</td>
</tr>
</tbody>
</table>
Appendix A: The List of Functions for Gene Expression

<table>
<thead>
<tr>
<th>Coefficient threshold, range is 0 - 1. The default is set as 0.5.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- corr_pattern: A flag to indicate genes are co-expressed or anti co-expressed with target genes. The default is set as &quot;positive&quot;, options have &quot;negative&quot;, &quot;both&quot;.</td>
</tr>
<tr>
<td>- &quot;title&quot;: Title of the plot. If missing, the function will provide a default.</td>
</tr>
</tbody>
</table>

Given an Exp object and a gene list file, this function helps you to find other genes that are co-expressed with these target genes having correlation coefficient values greater than the defined threshold.

```r
corr <- findCorrGenesFromFile(exp, gene_list_file = TRUE, sample_group = "all", method = "pearson", corr_threshold = 0.5)
```

- `exp`: A given EXP object.
- `gene_list_file`: A gene list file. If missing, it would pop-out a GUI to let user select a gene list file.
- `sample_group`: Given one or more sample groups (delimited by ","), The default is set as "all", meaning all samples in sample list.
- `method`: The method used to calculate gene correlation coefficient. The current option is "pearson".
- `corr_threshold`: Defined sample group(s), multiple samples are delimited by ",". The default is set as "all", meaning all samples in sample list.

Given an Exp object and a gene list file, this function helps you to find other genes that are co-expressed with these target genes having correlation coefficient values greater than the defined threshold.

```r
corr <- findCorrGenesFromList(exp, gene_list = TRUE, sample_group = "all", method = "pearson", corr_threshold = 0.5)
```

- `exp`: A given EXP object.
- `gene_list`: A gene list as target genes for correlation analysis.
- `sample_group`: Given one or more sample groups (delimited by ","), The default is set as "all", meaning all samples in
sample list.

- method: The method used to calculate gene correlation coefficient. The current option is "pearson".
- corr_threshold: Correlation coefficient threshold used to select co-expressed genes with target gene. The default is set as 0.5.

Given an Exp object and a target gene name, this function helps you to find other genes that are co-expressed with these target genes having correlation coefficient values greater than the defined threshold. (The default is 0.5.)

```r
findCorrGenesFromName(exp, gene_name, sample_group = "all", method = "pearson", corr_threshold = 0.5)
```

- exp: A given EXP object.
- gene_name: A gene list as target genes for correlation analysis.
- sample_group: Defined sample group(s), multiple samples are delimited by ",". The default is set as "all", meaning all samples in sample list.
- method: The method used to calculate gene correlation coefficient. The default is "pearson".
- corr_threshold: Correlation coefficient threshold used to select co-expressed genes with target gene. The default is set as 0.5.
Appendix B: The List of Functions for Variant and Mutation Analysis

This Appendix contains all the functions for variant analysis used in the SINGuLAR Analysis Toolset, as well as the function `vcIdentifySigVariants` for mutation analysis.

Installing

<table>
<thead>
<tr>
<th>Task</th>
<th>Code/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Fluidigm Single Cell Package Library for commands for the current session.</td>
<td><code>library(fluidigmSC)</code></td>
</tr>
<tr>
<td>The first method to be called after installing the <code>fluidigmSC</code> package. This function must only be run once.</td>
<td><code>firstRun()</code>&lt;br&gt;Downloads the following required packages: lattice, tcltk2, SDMTools, and rgl.</td>
</tr>
<tr>
<td>Set your working directory.</td>
<td><code>setwd(&quot;pathway of your working directory&quot;)</code>&lt;br&gt;Example: <code>setwd(&quot;C:/folder &quot;)</code></td>
</tr>
</tbody>
</table>

Getting R Help for the fluidigmSC Library

<table>
<thead>
<tr>
<th>Task</th>
<th>Code/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the complete list of fluidigmSC functions, and then click any link to display the corresponding help. This will engage the default Internet program to display an interactive Help page. After you are done viewing the Help, click the R console to continue.</td>
<td><code>?? fluidigmSC</code>&lt;br&gt;Then click <code>fluidigmSC::fluidigmSC_&lt;version number&gt;</code>.&lt;br&gt;To see the complete list of R Help for fluidigmSC, click [Package fluidigmSC version &lt;version number&gt; Index] at the bottom of the page fluidigmSC_&lt;version number&gt;-package.</td>
</tr>
<tr>
<td>Get help on an individual function in the R application or in the fluidigmSC library. Do not include parenthesis or dependencies. R is case-sensitive.</td>
<td><code>? functionName</code></td>
</tr>
<tr>
<td>List two subgroups for available functions. This will then force you to choose a</td>
<td><code>scFunctions()</code>&lt;br&gt;For Variant Analysis functions, type</td>
</tr>
</tbody>
</table>
### Auto-Analyzing Variant Data

Auto-analyzes raw variant data, annotate samples, and assign a QC flag to each variant of sample, based on user defined threshold of variant metrics (DP, GQ, AF, VSN).

Returns a list of filtered samples (the outlier_list), an HC heatmap, and the VC object. This also automatically saves 2 text files containing variant call performance by group and by sample respectively: Sample Group Summary (auto_analysis).txt, and Sample Summary (auto_analysis).txt.

```r
vcAutoAnalysis()
```

Opens the FluidigmSC Analysis dialog where you enter the files and instructions for automatic analysis and then run it.

You can enter:
- One or more VCF files to be analyzed.
- Sample and sample group annotations.
- A control sample or a control sample group to include only those variant calls that are in the defined control.
- A filtering method to filter out low quality samples.
- Location to save the VC object file.

### Reading and Writing Variant Data

Initialize the VC object from one or more VCF data files. This is the initialization function

```r
vc <- readVCF(vcf_file = TRUE, targets = "")
```
for reading variant data and creating a VC object for any downstream analysis.

Targets are the regions that you define to read only partial VCF data of interest instead of entire VCF data.

This returns a VC object.

- **vc_file**: The filename of the data to be opened. The default is set TRUE, which displays a file dialog that allows you to select one or more VCF files.
- **targets**: If left blank, this will read the entire VCF data file. Each target can be either (a) chromosome coordinates, such as 7:123-345;chr7:123-345 or chr7; or (b) gene symbol such as TP53. Targets can have multiple (a) and (b) or the combination of (a) and (b). The delimiter is ";".

Load one or more VC object data files and create a merged VC object.

vc <- readVCObject(vc_file = TRUE)

vc_file: The filename of the data to be opened. The default TRUE is set to display a file dialog box to select VC object file(s). This argument can be set to the file name with a file path.

After completing variant analysis, you can write out the filtered variant data as a VCF file for any downstream analysis.

writeVCF(vc, vcf_file = TRUE)

- **vc**: A given VC object.
- **vcf_file**: The default TRUE is set to enter a file through a file dialog. This argument can be set to the file name with a file path.

Save VC object to files for future use in R or to convert to data frames. Default saves as *.fso, which can be extracted using the Excel macro **FluidigmSCObjectToExcel.xlsm**.

saveData (vc)

vc: A given VC object.

Save a list of data. These lists can contain a variety of data, including samples, genes, group information and metadata. Returns a tab delimited text file (*.txt filename extension) that can be opened in Excel.

saveData (example_gene_list)

example_gene_list: An example of a user-defined name, which also calls a dialog to save with a *.txt filename extension.
### Save the automatically generated outlier_list from vcAutoAnalysis.

The format of this command is `saveData(object$frame)`

```r
saveData(vc $outlier_list)
```

### Managing Variant Data

#### Assign a Group ID

Extracts the sample group from the sample name with a separator from the VC object. The sample group ID must be either a prefix or suffix in the sample name.

```r
cv <- vcUpdateSampleListFromName(vc, sep = "_", prefix = TRUE)
```

- `vc`: The active VC object to update.
- `sep`: The character the function will use to separate the sample group ID from the sample name. The default is set equal to "_".
- `prefix`: Any other permitted symbol.

#### Update the VC object

Update the sample list of a VC object using a sample list file as input and return the updated VC object.

```r
cv <- vcUpdateSampleListFromFile(vc, sample_list_file = TRUE)
```

- `vc`: The VC object to update.
- `sample_list_file`: A tab-delimited text file (with a *.txt filename extension) with the first two columns containing sample names and group names with headers as "SampleID" and "GroupID", respectively. The default (TRUE) will open a GUI to select the sample. This can be set to equal the file and path of the file.

Update the sample list of a VC object using a sample list as input and return the updated VC object.

```r
cv <- vcUpdateSampleListFromList(vc, sample_list)
```

- `vc`: The VC object to update.
### Managing Variant Data

- **sample_list**: A data frame with the first two columns containing sample names and group names with headers as "SampleID" and "GroupID", respectively. This should be set to the `sample_list`.

According to variant quality, remove low-quality variants from the working variant list and return the updated VC object with updated variant list.

This function should be used in conjunction with `vcUpdateVariantQuality`, and it will return a VC object with variants that have at least one sample with an AB or BB calls.

```r
vc <- vcUpdateVariantListByVariantQuality(vc)
```

- `vc`: A given VC object with variant quality table.

Update the variant list of a VC object using a variant list file as input, and return the updated VC object.

```r
vc <- vcUpdateVariantListFromFile(vc, variant_list_file = TRUE)
```

- `vc`: The VC object to update.
- `variant_list_file`: A tab-delimited text file (with a *.txt filename extension) with the first two columns containing variant names and group names with headers as "VariantID" and "GroupID", respectively. The default (TRUE) will open a GUI to select the variant file. This can be set to equal the file and path of the file.

Update the variant list of a VC object using a variant list as input, and return the updated VC object.

```r
vc <- vcUpdateSampleListFromList(vc, variant_list)
```

- `vc`: The VC object to update.
- `variant_list`: A data frame with the first two columns containing sample names and group names with headers as "VariantID" and "GroupID", respectively. This should be set to the `variant_list`.

Update variant group annotations directly from variant_annotat_list in VC in which the variant annotations are parsed from INFO

```r
vc <- vcUpdateVariantListFromVariantAnnotation(vc, byAnnotation = "CHROM")
```
Appendix B: The List of Functions for Variant and Mutation Analysis

during VCF data reading.

Currently the supported variant annotation package is snpEFF.

### Determine quality of each variant call in individual samples based on predefined variant metrics (minimum DP, GQ, AF and sample number for each variant call, VSN) and the given thresholds.

Returns an updated vc with the calculated quality of variants kept in SCQC, a data frame in VC object.

<table>
<thead>
<tr>
<th>vc: The VC object to update.</th>
</tr>
</thead>
<tbody>
<tr>
<td>by_annotation: Variant annotations include: CHROM (as default), GENE_SYMBOL, and VARIANT_TYPE.</td>
</tr>
</tbody>
</table>

```r
vc <- vcUpdateVariantQuality(vc, DP_cutoff=20, GQ_cutoff=80, AF_cutoff=0.1, Sample_Num_cutoff=2, DPGQAF_operator = "AND")
```

| vc: A given VC object with variant quality table. |
| DP_cutoff: Threshold for minimum DP, where DP is read depth at this position for this sample. The default is 20. |
| GQ_cutoff: Threshold for minimum GQ, where GQ is conditional genotype quality, encoded as a phred quality \(-10\log_{10}p\) (genotype call is wrong, conditioned on the site's being variant). The default is 80. |
| AF_cutoff: Threshold for minimum AF, where AF is allele frequency for the ALT allele in this sample. It can be from VCF data directly or be calculated from AD with calculation formulation as \(AF=AD2/(AD1 + AD2)\). The default is 0.10. |
| Sample_Num_cutoff: Threshold for minimum sample number having such variant call (AB or BB) among samples. The default is 2. |
| DPGQAF_operator: Logical operator with default as "AND", option = "OR". It is used to determine how to combine the above metrics to determine final quality flag of variants of samples. |

If DPGQAF_operator is set as:

"AND" (default), the quality of a variant of sample is calculated as 1 if variant metrics pass all metric thresholds. Otherwise the
### Remove or Retain Sample and Variant Groups

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Example Code</th>
</tr>
</thead>
</table>
| Remove low-quality samples from sample list of the VC object. The low-quality samples are determined in SCQC table of the VC object. | `vc <- vcRemoveLowQualitySamples(vc, nocall_missing_threshold=0.5)` | - `vc`: A given VC object.  
- `nocall_missing_threshold`: Threshold of percent of samples having NO-CALL (low-quality) for a variant. The default is 0.5. |
| Remove a group of samples from the VC sample list, and return the updated VC object with a modified sample list. | `vc <- vcRemoveSampleGroup(vc, sample_group)` | - `vc`: A given VC object.  
- `sample_group`: The group name of the group of samples to be removed from the list.  
If there is only one sample group in a VC object, it cannot be removed since a VC object must contain at least one sample group. |
| Remove a sample from the VC sample list, and return the VC object with the updated sample list. | `vc <- vcRemoveSamples(vc, sample_name)` | - `vc`: A given VC object.  
- `sample_name`: The name of a sample to be removed from the list. |
| Remove a variant from the VC variant list, and return the VC object with the updated variant list. | `vc <- vcRemoveVariant(vc, variant_name)` | - `vc`: A given VC object.  
- `variant_name`: The name of a variant to be removed from the list. |
| Remove a variant group from the VC variant | `vc <- vcRemoveVariantGroup(vc, variant_group)` | - `vc`: A given VC object.  
- `variant_group`: The group name of the group of variants to be removed from the list. |
list, and return the VC object with the updated variant list.

- vc: A given VC object.
- variant_group_name: The group name of a group of variants to be removed from the list.

Retain only the samples of a given group from the VC sample list, and return the updated VC object with the modified sample list.

- vc <- vcRetainSampleGroup(vc, sample_group)
- vc: A given VC object.
- sample_group_name: The name of a sample group to be retained in the list.

Retain only the variants of a given group from the VC variant list and return the updated VC object with the modified variant list.

- vc <- vcRetainVariantGroup(vc, variant_group)
- vc: A given VC object.
- variant_group_name: The name of the variant group to be retained in the variant list.

**Get Variants**

Given a VC, a sample group, and a defined minimum GT concordance threshold in this sample group, return a variant list with common GT.

This is useful when you are interested in variant calls with high concordance across control samples. You can define your own concordance threshold. The default is 0.51.

- ref_variant_list_wcall <- vcGetCommonVariantListWithCallFromSampleGroup(vc, sample_group, concordance_threshold = 0.51, variant_only = FALSE)
- vc: A given VC object.
- sample_group: A given sample group. Replace with the group name in quotes.
- concordance_threshold: A GT concordance threshold within samples of this sample group. The default is 0.51.
- variant_only: A logical to determine whether to return all common calls or variant calls with hetero AB or homovariant BB only.

Given a VC object, check working variants in

- failed_filter_variants <-
### Managing Variant Data

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vcGetFailVariantsFromVariantCall(vc)</code></td>
<td>Given a VC object, return variants that failed in filter in the original variant calls.</td>
</tr>
<tr>
<td><code>vcGetFailVariantsFromVariantCall(vc)</code></td>
<td>Given a VC object with quality table of SCQC, return high-quality variants. This function should be used in conjunction with <code>vcUpdateVariantQuality</code>, and it will return all the variants that have at least one sample with an AB or BB call.</td>
</tr>
<tr>
<td><code>vcGetPassVariantsFromVariantCall(vc)</code></td>
<td>Given a VC object, check working variants in variant list, and return variants passing filter in original VCF data.</td>
</tr>
<tr>
<td><code>vcGetSignificantVariants(ft, pvalue_threshold = 0.05)</code></td>
<td>Given Fisher's Exact Test result, return variants with significant changes in any pairwise sample groups.</td>
</tr>
<tr>
<td><code>vcGetVariantListBySampleGroupAlleleFrequency(vc, sample_group, allele_freq_threshold = 0.1)</code></td>
<td>Given a VC and sample group and defined minimum allele frequency in sample group, returns variant list with common GT in which their allele frequency are greater than the threshold.</td>
</tr>
</tbody>
</table>

Get variant list with genotyping information from a given sample.
Appendix B: The List of Functions for Variant and Mutation Analysis

```r
sample_name, variant_only = FALSE)
```

- `vc`: A given VC object.
- `sample_name`: A given sample. Replace with sample name in quotes.
- `variant_only`: A logical to determine to get all calls or just variant calls of a sample. The default is set as FALSE. When set to FALSE, returns all calls compared to common calls. When set to TRUE, only returns variant calls with GT as either hetero (AB) or homo variant (BB).

### Change Group Colors and Symbols

<table>
<thead>
<tr>
<th>Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>If VC has more than one sample group, the SINGuLAR™ system assigns the default color and symbol scheme to the sample groups. You can customize the color and symbol schemes for each sample group, and the customized color and symbol will be used in the HC heatmap.</td>
<td><code>vc &lt;- vcSetSampleGroupColorAndSymbols (vc)</code></td>
</tr>
<tr>
<td><code>vc</code>: A given VC object.</td>
<td></td>
</tr>
<tr>
<td>If VC has more than one variant group, the SINGuLAR™ system assigns the default color and symbol scheme to the variant groups. You can customize the color and symbol schemes for each variant group, and the customized color and symbol will be used in the HC heatmap.</td>
<td><code>vc &lt;- vcSetVariantGroupColorAndSymbols (vc)</code></td>
</tr>
<tr>
<td><code>vc</code>: A given VC object.</td>
<td></td>
</tr>
<tr>
<td>Restore the current color and symbol settings of sample groups back to their default settings.</td>
<td><code>vc &lt;- vcClearSampleGroupColorAndSymbols (vc)</code></td>
</tr>
<tr>
<td><code>vc</code>: A given VC object with sample group annotation.</td>
<td></td>
</tr>
<tr>
<td>Restore the current color and symbol settings of the sample groups back to their default settings.</td>
<td><code>vc &lt;- vcClearVariantGroupColorAndSymbols (vc)</code></td>
</tr>
<tr>
<td><code>vc</code>: A given VC object with variant group annotation.</td>
<td></td>
</tr>
</tbody>
</table>
## Managing Variant Data

### Reset Sample and Variant Lists

<table>
<thead>
<tr>
<th>Action</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace a VC object’s current list of samples with its original list of samples.</td>
<td><code>vc &lt;- vcResetSampleList(vc)</code></td>
<td>vc: A given VC object.</td>
</tr>
<tr>
<td>Replace a VC object’s current list of variant with its original list of variants.</td>
<td><code>vc &lt;- vcResetVariantList(vc)</code></td>
<td>vc: A given VC object.</td>
</tr>
</tbody>
</table>

### Merge Data

<table>
<thead>
<tr>
<th>Action</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge two VC objects into a single VC object.</td>
<td><code>vc &lt;- vcMergeVCObj(vc1, vc2)</code></td>
<td>vc1: The first VC object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vc2: The second VC object.</td>
</tr>
</tbody>
</table>

### Evaluate Variant-Calling Performance

<table>
<thead>
<tr>
<th>Action</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given a VC object and a variant list with common call, conduct an evaluation of variant-calling performance for each variant in individual samples by comparing the variant calls of samples with the given common call list.</td>
<td><code>result &lt;- vcEvalSampleVariants(vc, ref_variant_list_wcall)</code></td>
<td>vc: A given VC object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ref_variant_list_wcall: A variant list with common GT. This should be replaced with the list.</td>
</tr>
<tr>
<td>Given a VC object and a variant list with common call, conduct an evaluation of variant-calling performance for each variant in sample groups by comparing the variant calls in samples of each sample group with the given common call list.</td>
<td><code>result &lt;- vcEvalSampleGroupVariants(vc, ref_variant_list_wcall)</code></td>
<td>vc: A given VC object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ref_variant_list_wcall: A variant list with common GT. This should be replaced with the list.</td>
</tr>
</tbody>
</table>
### Managing Outliers

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Function Call</th>
<th>Parameters</th>
</tr>
</thead>
</table>
| Open a list of samples defined in a file, move them from the sample list to the outlier list for the given VC, and return the updated VC object. | `vc <- vcAddOutlierFromFile(vc, outlier_list_file=TRUE)`                      | - `vc`: A given VC object.  
  - `outlier_list_file`: An outlier list file. If TRUE, this function displays a file dialog to select a sample list file. |
| Provides a list of samples, moves them from the sample list to the outlier list for the given VC, and return the updated VC object. | `vc <- vcAddOutlierFromList(vc, sample_list)`                                | - `vc`: A given VC object.  
  - `sample_list`: A given outlier sample list. Replace with the list name. |
| Move an outlier sample based on given outlier sample name to the outlier list, and return the updated VC object. | `vc <- vcAddOutlierFromName(vc, outlier_sample)`                             | - `vc`: A given VC object.  
  - `outlier_sample`: The outlier to be moved to the outlier list. Replace with the sample name in quotes. |
| Restore all outliers back into the working sample list for downstream VC analysis. | `vc <- vcRestoreAllOutlier(vc)`                                               | - `vc`: A given VC object. |
| Given an outlier list, restore outliers from the outlier list back to the sample list of a given VC object. | `vc <- vcRestoreOutlierFromList(vc, sample_list)`                            | - `vc`: A given VC object.  
  - `sample_list`: List of sample outliers to be restored. Replace with the list name. |
| Given a sample name, restore that sample from the outlier list back to the sample list | `vc <- vcRestoreOutlierFromName(vc, sample_name)`                             | - `vc`: A given VC object.  
  - `sample_name`: Replace with the sample name in quotes. |
Displaying Variant Data

<table>
<thead>
<tr>
<th>Summary of Fisher's Exact Test Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Fisher's Exact test for VC data if sample group annotation is present and generate p-value to each variant in all pairwise sample groups.</td>
</tr>
<tr>
<td>vc: A given VC object.</td>
</tr>
<tr>
<td>Returns an ft object that contains test result.</td>
</tr>
<tr>
<td>Display the results of the Fisher's Exact test. Interactively locate and save variants of interest and display a Manhattan plot.</td>
</tr>
<tr>
<td>You can select significant variants in the test either by variants having significant p-value (e.g. pvalue&lt;=0.05) or by manually selecting variants (where the locate function if locate is set as TRUE).</td>
</tr>
<tr>
<td>Returns selected_variant_list.</td>
</tr>
<tr>
<td>locate: The default (TRUE) is set to interactively locate and to save variants of interest. When set to FALSE, it only displays the plot.</td>
</tr>
</tbody>
</table>
### Summary of HC Functions

<table>
<thead>
<tr>
<th>Function Description</th>
<th>Function Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given a VC object, conduct unsupervised hierarchical clustering (HC) analysis for</td>
<td>`vcHC(vc, dendrogram = &quot;both&quot;, color_scheme=&quot;black_blue_green_red_grey&quot;, missing_data_as_AA = FALSE,</td>
</tr>
<tr>
<td>variant data.</td>
<td>display_sample_names = TRUE, display_variant_names = TRUE, heatmap_display = TRUE)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Displays the results of HC analysis of variant</td>
<td><code>vcDisplayHC(hc, color_scheme =</code></td>
</tr>
</tbody>
</table>

- **vc**: A given VC object.
- **dendrogram**: Clustering method. The default to set to cluster both sample and variant. Options are: "both", "sample", "variant", and "none".
- **color_scheme**: The predefined color scheme in HC. The default is "black_blue_green_red_grey". If a different color scheme is preferred, set this argument to "" and the function helps you create a desired color scheme.
- **missing_data_as_AA**: A logical to determine whether convert missing data as AA call in HC analysis. The default is FALSE and will not convert missing data to AA.
- **display_sample_names**: The default is set to display sample names in the HC plot. When set to FALSE, sample names will not be displayed.
- **display_variant_names**: The default (TRUE) is set to display variant names in the HC plot. When set to FALSE, variant names will not be displayed.
- **heatmap_display**: The default (TRUE) is set to display clustering result. When set to FALSE, no heatmap will be displayed.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcExportHCHeatmap(hc)</td>
<td>Exports the heatmap of a given hierarchical clustering analysis to a tab-delimited text file. Values will remain in their respective locations.</td>
</tr>
<tr>
<td>select_sample_clusters &lt;- vcIdentifySampleClusters (hc)</td>
<td>Interactively identify sample clusters using the Sample Cluster dendrogram, and return the identified cluster(s) as a sample list containing SampleID and a ClusterID.</td>
</tr>
<tr>
<td>select_variant_clusters &lt;- vcIdentifyVariantClusters (hc)</td>
<td>Interactively identify variant clusters using the Variant Cluster dendrogram. Returns a list.</td>
</tr>
</tbody>
</table>

- **hc**: The HC object from vcHC analysis.
- **color_scheme**: The predefined color scheme in HC for genotyping. The default is "black_blue_green_red_grey", with black for missing data, blue for AA, green for AB, red for BB, and grey for NO-CALL. If a different color scheme is preferred, set this argument to "" and the function helps you create the desired color scheme.
- **display_sample_names**: The default (TRUE) is set to display sample names in the HC plot. When set to FALSE, sample names will not be displayed.
- **display_variant_names**: The default (TRUE) is set to display variant names in the HC plot. When set to FALSE, gene names will not be displayed.
- **"title"**: Title of the plot. If missing, the function will provide a default.
## Identifying Significant Variants

Opens a set of dialogs that allows you to identify significant variants by single-cell mutation analysis (case/control) or single-cell heterogeneity analysis. Methods to extract variants or mutations of interest include:

- General variant quality filters, such as DP and GQ of variant of sample
- Single-cell related variant quality filters, such as non-reference allele frequency in variant of single cells (AF) and variant found in number of single cells (VSN)
- Mutation detection based on Fisher’s Exact Test for Case vs. Control analysis
- Other filters as optional:
  1. Mutations not in bulk cell control samples
  2. Mutations in bulk cell case samples
  3. No indel
  4. FLITER flag as “PASS”
  5. MISSENSE
  6. Non dbSNP

Filters 1-2 are mutation filters; and 3, 4, 5 are general variant filters.

For all filters except the one for the Fisher's Exact Test in mutation analysis, you can decide your own combination of filters.

For all of the above filters, SINGuLAR provides suggested (default) cutoff values and accompanying descriptions and allows you to change values within reasonable ranges.

If there are less than 2500 variants to be selected, this function displays a single heatmap that clusters the samples and variants selected. If the samples are annotated, this function displays two heatmaps: (1) one with both sample and variant clusters and (2) one with only variant clusters. Samples are ordered by the sample groups.
Appendix C: Contents of the EXP Object

In the SINGuLAR Analysis Toolset, EXP data is read, parsed, and converted to a Fluidigm VC object, which is a list containing a series of data frames. You can then use that EXP object for further analysis.

Your training data was converted to the VC object (auto_analysis).fso through automatic analysis [using AutoAnalysis()] and then subjected to some of the advanced analyses illustrated in Chapter 3 on Gene Expression. The data frames that follow are illustrations of the contents of (auto_analysis).fso.

**Data Frame: obj_type**

This data frame contains only the name of the Fluidigm EXP object.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>obj_type</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>EXP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data Frame: data_type**

This data frame contains the type of org_data, which can be Ct or LinearExp.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>data_type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Ct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data Frame: lod**

This data frame contains the Limit of Detection (LoD) for the org_data.
Appendix C: Contents of the EXP Object

Data Frame: gene_lods

This data frame is unused.

Data Frame: org_data

This data contains the data from imported expression files. The first row holds the sample names and the first column holds the gene names (or assay names). Missing values are replaced with -1.
Data Frame: gene_list

This data contains the an array of gene names (or assay names) with the "Gene ID" as the first row. They are a subset of those in the org_data.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GeneID</td>
</tr>
<tr>
<td>2</td>
<td>G49</td>
</tr>
<tr>
<td>3</td>
<td>G46</td>
</tr>
<tr>
<td>4</td>
<td>G05</td>
</tr>
<tr>
<td>5</td>
<td>G86</td>
</tr>
<tr>
<td>6</td>
<td>G89</td>
</tr>
<tr>
<td>7</td>
<td>G03</td>
</tr>
</tbody>
</table>

Data Frame: sample_list

This data frame contains a data frame of sample names (first column) and types (second column). They are a subset of those in the org_data.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SampleID</td>
</tr>
<tr>
<td>2</td>
<td>CT1_94-1</td>
</tr>
<tr>
<td>3</td>
<td>CT1_95-1</td>
</tr>
<tr>
<td>4</td>
<td>CT1_50-1</td>
</tr>
<tr>
<td>5</td>
<td>CT1_96-1</td>
</tr>
<tr>
<td>6</td>
<td>CT1_49-1</td>
</tr>
<tr>
<td>7</td>
<td>CT1_10-1</td>
</tr>
</tbody>
</table>

Data Frame: outlier_list

This data frame contains a list of identified outliers: sample names with sample group ID.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SampleID</td>
</tr>
<tr>
<td>2</td>
<td>CT2_51-0</td>
</tr>
<tr>
<td>3</td>
<td>CT2_01-0</td>
</tr>
<tr>
<td>4</td>
<td>CT2_27-0</td>
</tr>
<tr>
<td>5</td>
<td>CT2_56-1</td>
</tr>
<tr>
<td>6</td>
<td>CT2_06-0</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C: Contents of the EXP Object

Data Frame: `log2ex_data`

This data frame contains the transformed data from `org_data` with limit of detection and trimmed with `gene_list` (if defined) and `sample_list` (if defined). The format is the same as that of the `org_data` data frame.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>CT1_94-1</td>
<td>CT1_95-1</td>
<td>CT1_50-1</td>
<td>CT1_49-1</td>
<td>CT1_48-1</td>
<td>CT1_47-1</td>
<td>CT1_02-1</td>
<td>CT1_46-1</td>
<td>CT1_03-1</td>
<td>CT1_01-1</td>
</tr>
<tr>
<td>2</td>
<td>G49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>G46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>G05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.876596</td>
</tr>
<tr>
<td>5</td>
<td>G86</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>G89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000001</td>
</tr>
<tr>
<td>7</td>
<td>G93</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>G82</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>G79</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>G08</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.51087</td>
</tr>
<tr>
<td>11</td>
<td>G14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>G80</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>G51</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>G71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.11113</td>
</tr>
<tr>
<td>15</td>
<td>G27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Data Frame: `log2ex_avg_data`

This data frame contains the same format as the `org_data`, except the first row is sample type and expression value is the ensemble averaged log2ex from `org_data` if the sample type is provided in the `sample_list`.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>CT1</td>
<td>CT2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>G49</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>G46</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>G05</td>
<td>0.241061</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>G86</td>
<td>1.125515</td>
<td>0.696523</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>G89</td>
<td>0.334612</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>G93</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>G82</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>G79</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>G08</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>G14</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>G80</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>G51</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>G71</td>
<td>0</td>
<td>2.893981</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>G27</td>
<td>0</td>
<td>2.870704</td>
<td></td>
</tr>
</tbody>
</table>
Data Frame: summary

This data frame contains a summary of the experiment data.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Category</td>
<td>Value</td>
</tr>
<tr>
<td>2</td>
<td>Number of original samples</td>
<td>192</td>
</tr>
<tr>
<td>3</td>
<td>Number of original genes</td>
<td>96</td>
</tr>
<tr>
<td>4</td>
<td>Number of working genes</td>
<td>94</td>
</tr>
<tr>
<td>5</td>
<td>Number of working sample groups</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Number of working samples in CT1</td>
<td>94</td>
</tr>
<tr>
<td>7</td>
<td>Number of working samples in CT2</td>
<td>93</td>
</tr>
<tr>
<td>8</td>
<td>Number of outliers</td>
<td>5</td>
</tr>
</tbody>
</table>
Appendix D: Contents of the VC Object

In the SINGuLAR Analysis Toolset, VCF data is read, parsed, and converted to a Fluidigm VC object, which is a list containing a series of data frames. You can then use that VC object for further analysis.

Your training data, CT1_CT2_DNA-Seq.vcf, was converted to the VC object vc (auto_analysis).fso through automatic analysis [using vcAutoAnalysis()] and then subjected to some of the advanced analyses illustrated in Chapter 4 on Variant Analysis. The data frames that follow are illustrations of the contents of vc (auto_analysis).fso.

**Data Frame: obj_type**

This data frame contains only the name of the Fluidigm VC object.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>obj_type</td>
<td>VC</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data Frame: data_format**

This data frame contains the version of the VCF file.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>data_format</td>
<td>VCFv4.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data Frame: genome_build**

This data frame uses either the CRCR37 or hg19 reference genome.
Data Frame: sample_list

This data frame contains sample names (Column A) and types (Column B). They are a subset of those in the org_data data frame.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT1-gDNA_S01</td>
<td>CT1-gDNA</td>
</tr>
<tr>
<td>CT1-gDNA_S02</td>
<td>CT1-gDNA</td>
</tr>
<tr>
<td>CT1-gDNA_S03</td>
<td>CT1-gDNA</td>
</tr>
<tr>
<td>CT1-SC_1_S01</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S02</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S03</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S04</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S05</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S06</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S07</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S08</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S09</td>
<td>CT1-SC</td>
</tr>
<tr>
<td>CT1-SC_1_S10</td>
<td>CT1-SC</td>
</tr>
</tbody>
</table>

Data Frame: variant_list

This data frame contains an array of variant names (or assay names) with the column names of "VariantID" and "GroupID" as the first row. They are a subset of those in the org_data data frame.

A variant name is either rs ID or UID with the combination of chrome, pos, and ref, and alt.
**Data Frame: mult_allele_data**

This data frame contains a list of variants having multiple alleles in variant call.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VariantID</td>
<td>GroupID</td>
</tr>
<tr>
<td>2</td>
<td>rs2072454</td>
<td>VC_1</td>
</tr>
<tr>
<td>3</td>
<td>rs4947986</td>
<td>VC_1</td>
</tr>
<tr>
<td>4</td>
<td>7:55240708_T TG</td>
<td>VC_1</td>
</tr>
<tr>
<td>5</td>
<td>7:55241604_T TC</td>
<td>VC_1</td>
</tr>
<tr>
<td>6</td>
<td>rs2293347</td>
<td>VC_1</td>
</tr>
<tr>
<td>7</td>
<td>7:55269077_T A</td>
<td>VC_1</td>
</tr>
<tr>
<td>8</td>
<td>rs35775721</td>
<td>VC_1</td>
</tr>
<tr>
<td>9</td>
<td>rs2023748</td>
<td>VC_1</td>
</tr>
<tr>
<td>10</td>
<td>rs206075</td>
<td>VC_2</td>
</tr>
<tr>
<td>11</td>
<td>rs206076</td>
<td>VC_2</td>
</tr>
<tr>
<td>12</td>
<td>13:32925501_G GT</td>
<td>VC_2</td>
</tr>
<tr>
<td>13</td>
<td>rs9534262</td>
<td>VC_2</td>
</tr>
</tbody>
</table>

**Data Frame: org_data**

This data frame contains the data in the original VCF.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CHROM</td>
<td>POS</td>
<td>ID</td>
<td>REF</td>
<td>ALT</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>55238261</td>
<td>.</td>
<td>TC</td>
<td>TCC,T</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SINGuLAR Analysis Toolset User Guide**
Appendix D: Contents of the VC Object

Data Frame: org_data_header

This data frame contains the header of the original VCF.

Data Frame: variant_annot_list

This data frame contains an array of variant names (or assay names) with the column names of "VariantID" and the parsed annotation from the VCF. They are a subset of those in the org_data data frame.

Data Frame: GT

This data frame contains the encoded genotyping data. They are a subset of those in the org_data data frame.
Data Frame: AF

This data frame contains the parsed allele frequencies, based on AD in variant calls. They are a subset of those in the org_data data frame.

<table>
<thead>
<tr>
<th>ID</th>
<th>CT1_gDNA_S01</th>
<th>CT1_gDNA_S02</th>
<th>CT1_SC_1_S01</th>
<th>CT1_SC_1_S02</th>
<th>CT1_SC_1_S03</th>
<th>CT1_SC_1_S04</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2072454</td>
<td>0.437956204</td>
<td>0.529850746</td>
<td>0.35</td>
<td>0.0864131817</td>
<td>0.992753623</td>
<td>0.781021898</td>
</tr>
<tr>
<td>rs4947986</td>
<td>0.35</td>
<td>0.558139535</td>
<td>0.372091023</td>
<td>0.571428571</td>
<td>0.181818182</td>
<td>0.463414534</td>
</tr>
<tr>
<td>rs20724708_T_TG</td>
<td>0.0001363636</td>
<td>0.0013513614</td>
<td>0.0012121212</td>
<td>0.0012121212</td>
<td>0.0012121212</td>
<td>0.0012121212</td>
</tr>
<tr>
<td>rs207241604_T_TIC</td>
<td>0.0001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
</tr>
<tr>
<td>rs2056757</td>
<td>0.562043796</td>
<td>0.4609375</td>
<td>0.457746479</td>
<td>0.335862821</td>
<td>0.335862821</td>
<td>0.335862821</td>
</tr>
<tr>
<td>rs20567721</td>
<td>0.197183099</td>
<td>0.001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
<td>0.001150476</td>
</tr>
<tr>
<td>rs2056787</td>
<td>0.281666667</td>
<td>0.266666667</td>
<td>0.2361111111</td>
<td>0.197183099</td>
<td>0.001150476</td>
<td>0.001150476</td>
</tr>
<tr>
<td>rs2056795</td>
<td>0.192307692</td>
<td>0.992307692</td>
<td>0.992307692</td>
<td>0.992307692</td>
<td>0.992307692</td>
<td>0.992307692</td>
</tr>
<tr>
<td>rs2056805</td>
<td>0.758975611</td>
<td>0.6375</td>
<td>0.80952381</td>
<td>0.821428571</td>
<td>0.97438972</td>
<td>0.46835443</td>
</tr>
</tbody>
</table>

Data Frame: DP

This data frame contains the coverage depth of variant calls. They are a subset of those in the org_data data frame.

<table>
<thead>
<tr>
<th>ID</th>
<th>CT1_gDNA_S01</th>
<th>CT1_gDNA_S02</th>
<th>CT1_gDNA_S03</th>
<th>CT1_SC_1_S01</th>
<th>CT1_SC_1_S02</th>
<th>CT1_SC_1_S03</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2072454</td>
<td>131</td>
<td>129</td>
<td>134</td>
<td>131</td>
<td>137</td>
<td>129</td>
</tr>
<tr>
<td>rs4947986</td>
<td>38</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td>42</td>
<td>39</td>
</tr>
<tr>
<td>rs20724708_T_TG</td>
<td>95</td>
<td>86</td>
<td>94</td>
<td>75</td>
<td>88</td>
<td>104</td>
</tr>
<tr>
<td>rs207241604_T_TIC</td>
<td>44</td>
<td>46</td>
<td>40</td>
<td>44</td>
<td>43</td>
<td>42</td>
</tr>
<tr>
<td>rs209325901_G_GT</td>
<td>47</td>
<td>46</td>
<td>53</td>
<td>55</td>
<td>52</td>
<td>56</td>
</tr>
<tr>
<td>rs2056757</td>
<td>88</td>
<td>88</td>
<td>94</td>
<td>93</td>
<td>99</td>
<td>84</td>
</tr>
<tr>
<td>rs2060756</td>
<td>122</td>
<td>125</td>
<td>125</td>
<td>121</td>
<td>127</td>
<td>129</td>
</tr>
<tr>
<td>rs2056805</td>
<td>174</td>
<td>176</td>
<td>176</td>
<td>174</td>
<td>188</td>
<td>199</td>
</tr>
<tr>
<td>rs2056795</td>
<td>78</td>
<td>77</td>
<td>80</td>
<td>80</td>
<td>77</td>
<td>76</td>
</tr>
</tbody>
</table>

SINGuLAR Analysis Toolset User Guide 167
Appendix D: Contents of the VC Object

**Data Frame: GQ**

This data frame contains conditional genotype quality, encoded as a phred quality $-10\log_{10} p$, and are a subset of those in the org_data data frame.

<table>
<thead>
<tr>
<th>ID</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2072454</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs4047986</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>7:55240708_T_TG</td>
<td>99</td>
<td>99</td>
<td>87</td>
<td>63</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs2293347</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>7:55260077_T_A</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>6</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs35775721</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs2023748</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs206075</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs206076</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs9534262</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>rs1625895</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
</tbody>
</table>

**Data Frame: gt_group_data**

This data frame contains a **Summary of Genotype** of each variant call in their respective sample groups. They are a subset of those in the org_data data frame.

<table>
<thead>
<tr>
<th>ID</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2072454</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs4047986</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:55240708_T_TG</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:55241604_T_TC</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs2293347</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:55260077_T_A</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs35775721</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs2023748</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs206075</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs206076</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:32029501_G_GT</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs9534262</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs1625895</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data Frame: SCQC**

This data frame contains a QC flag to represent passing QC filters, which are currently based on DP, GQ and AF and variant call frequency of sample group (also referred to as VSN). They are a subset of those in the org_data data frame. The values are defined as follows:

- 1=passing
- 0=NoCall
Data Frame: outlier_list

This data frame contains a list of identified outliers, which are sample names (Column A) and their sample types (Column B).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>CT1-gDNA_S01</td>
<td>CT1-gDNA_S02</td>
<td>CT1-gDNA_S03</td>
<td>CT1-SC_1_S01</td>
<td>CT1-SC_1_S02</td>
<td>CT1-SC_1_S03</td>
</tr>
<tr>
<td>2</td>
<td>rs2072454</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>rs4947986</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>7:55240708_T_TG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>7:55241064_T_TC</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>rs2293347</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7:55269077_I_A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>rs35775721</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>rs2023748</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>rs206075</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>rs206076</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>13:32929501_G_GT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>rs9534262</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>rs1625895</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>rs1625895</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix D: Contents of the VC Object
Appendix E: References for Gene Expression


Appendix E: References for Gene Expression


For Statistical Methodologies


Appendix F: References for Variant and Mutation Analysis


GATK Software Package:

http://www.broadinstitute.org/gatk/about/citing-gatk

Matric Definition for Evaluation of Genotyping Performances:

http://gatkforums.broadinstitute.org/discussion/48/using-varianteval