FLUIDIGM®

PN 100-5066 Rev 09

Singular Analysis Toolset

USER GUIDE

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

Information in this publication is subject to change without notice. It is Fluidigm policy to improve products as new techniques and components become available. Therefore, Fluidigm reserves the right to change specifications at any time. Every effort has been made to avoid errors in the text, diagrams, illustrations, figures, and screen captures. However, Fluidigm assumes no responsibility for any errors or omissions. In no event shall Fluidigm be liable for any damages in connection with or arising from the use of this publication.

Patent and Limited License Information
Fluidigm products are covered by issued and pending patents in the United States and other countries. Patent and limited license information is available at fluidigm.com/legalnotices.

Limited Use License to Perform Preamplification with Fluidigm IFCs
A license to use Thermo Fisher Scientific's patented preamplification method workflows involving a Fluidigm integrated fluidic circuit (IFC) can be obtained (i) with purchase of a Fluidigm IFC from Fluidigm Corporation or (ii) by a separate license from Thermo Fisher Scientific. For licensing information, contact outlicensing@lifetech.com or Out Licensing, Thermo Fisher Scientific, 5791 Van Allen Way, Carlsbad, CA 92008.

Limited Digital PCR License
A license to use Thermo Fisher Scientific's patented digital PCR method in all fields other than in the Sequencing Field, the Mass Spectrometry Field, and the Prenatal Field in workflows involving a Fluidigm IFC can be obtained (i) with purchase of a Fluidigm IFC from Fluidigm Corporation or (ii) by a separate license from Thermo Fisher Scientific. For licensing information, contact outlicensing@lifetech.com or Out Licensing, Thermo Fisher Scientific, 5791 Van Allen Way, Carlsbad, CA 92008.

Trademarks
Fluidigm, the Fluidigm logo, Biomark, C1, D3, Delta Gene, Dynamic Array, Juno, and Singular are trademarks or registered trademarks of Fluidigm Corporation in the United States and/or other countries. All other trademarks are the sole property of their respective owners.

For EU's WEEE directive information, go to fluidigm.com/compliance.

For technical support visit fluidigm.com/support

North America +1 650 266 6100 | Toll-free (US/CAN): 866 358 4354 | techsupport@fluidigm.com

Latin America +1 650 266 6100 | techsupportlatam@fluidigm.com

Europe/Middle East/Africa/Russia +33 1 60 92 42 40 | eu.support@fluidigm.com

Japan +81 3 3662 2150 | techsupportjapan@fluidigm.com

China (excluding Hong Kong) +86 21 3255 8368 | techsupportchina@fluidigm.com

All other Asian countries/India/Australia +1 650 266 6100 | techsupportasia@fluidigm.com

Contents

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 sequencing 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, tSNE, 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 HD 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 sequencing 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.

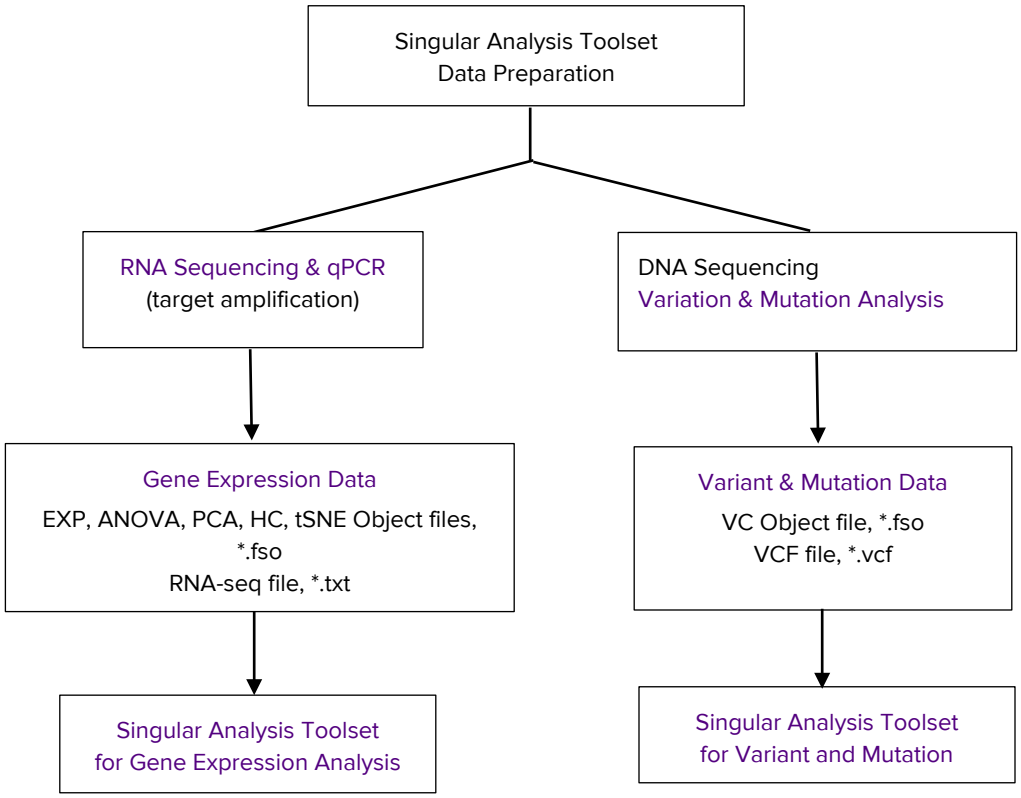
In single-cell variant analysis, besides using general QC metrics (such as DP, GQ, and so forth) to evaluate the quality of each variant call, researchers can use “variant event occurring in multiple numbers of single cells” to significantly eliminate false variant calls caused by random errors from 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.

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

- Biomark™ HD system
- Biomark™ HD Data Collection Software
- C1™ system
- C1™ IFC & Consumables
- D3™ Assay Design
- Delta Gene™ Assays
- Juno™ system
- Real-Time PCR Analysis Software
- 48.48, 96.96, and 192.24 Dynamic Array™ IFCs for Gene Expression

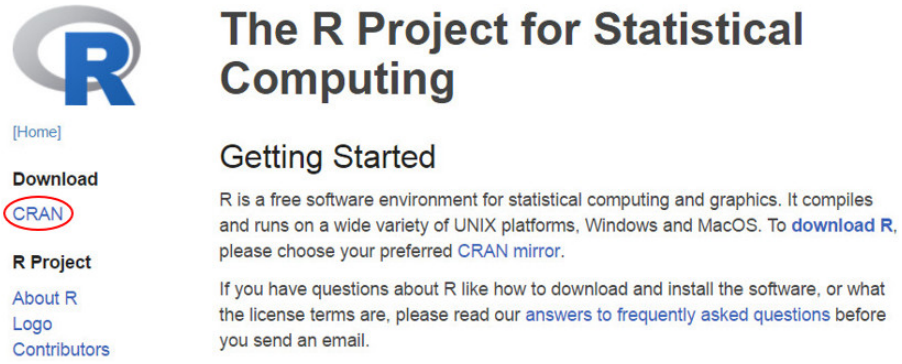
Chapter 2: Installing R and the Singular Analysis Toolset

Install R 3.3.3 for Windows

1 Ensure that you are installing the software on a computer running on the Windows operating system (XP or higher).

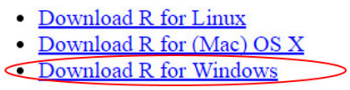
2 Navigate to the R-homepage at www.r-project.org.

3 Click the CRAN link on the left:



4 On the CRAN Mirrors page, scroll down to the closest CRAN mirror (replicate server) to your region, select it, and then click **OK**.

5 On the Comprehensive R Archive Network page, click the R download link for Windows:



6 On the R for Windows page, click **install R for the first time**.

7 On the R-3.4.0 (or later version) for Windows (32/64 bit) page, click **Previous releases** in the Other Builds section.

8 On the Previous Releases of R for Windows page, click **R 3.3.3** (March, 2017).

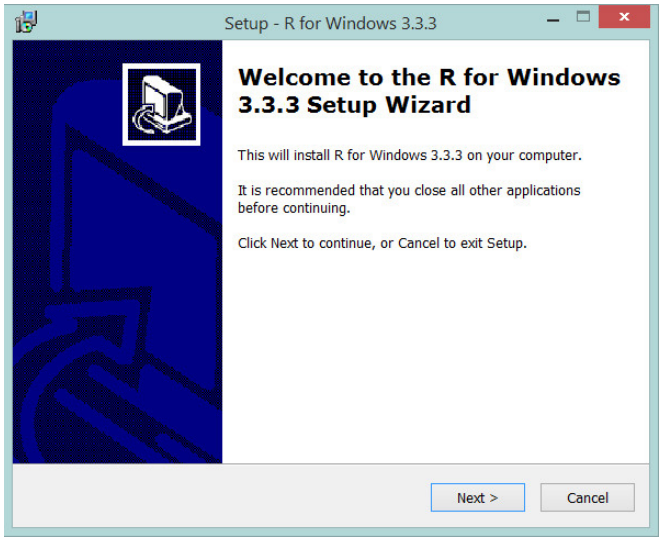
9 On the R-3.3.3 for Windows (32/64 bit) page, click **Download R 3.3.3 for Windows**.

10 Click the R-3.3.3 downloaded executable that you see in the bottom left of your screen.

11 When you are prompted to allow the software application to make changes on your computer, click **Yes**.

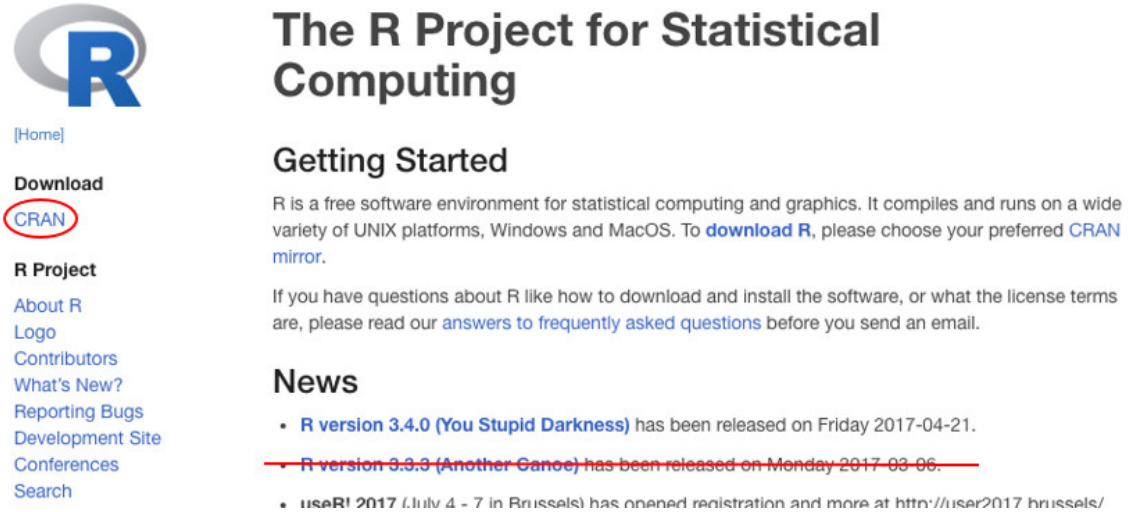
12 In the Select Setup Language dialog, select the language of your choice and click **OK**.

13 Follow the setup instructions in the installation wizard (Install the base subdirectories only.):



Install R 3.3.3 for Mac

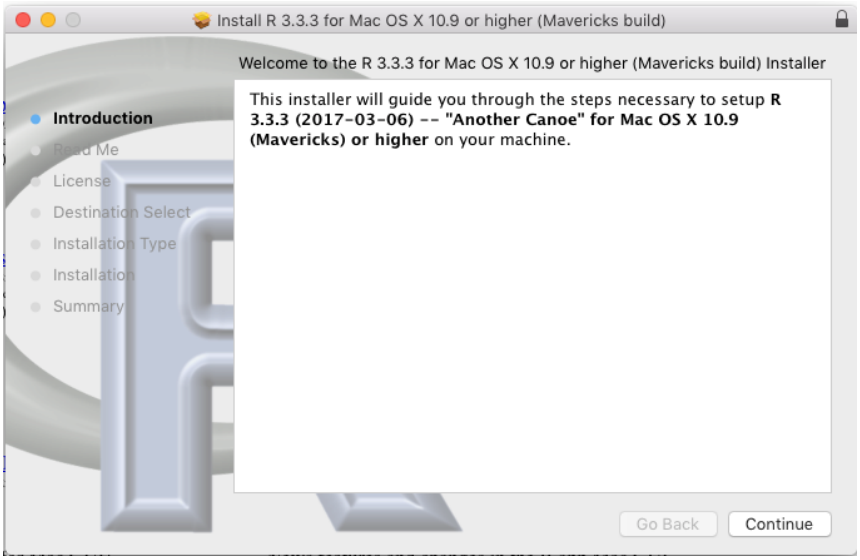
1. Ensure that you are installing the software on a computer running the Fluidigm-required Mac operating system OS X 11 (Sierra) to run R 3.3.3 on the Mac.
2. Navigate to the R-homepage at www.r-project.org.
3. Click the **CRAN** link on the left (but not the link for R version 3.3.3, if present):



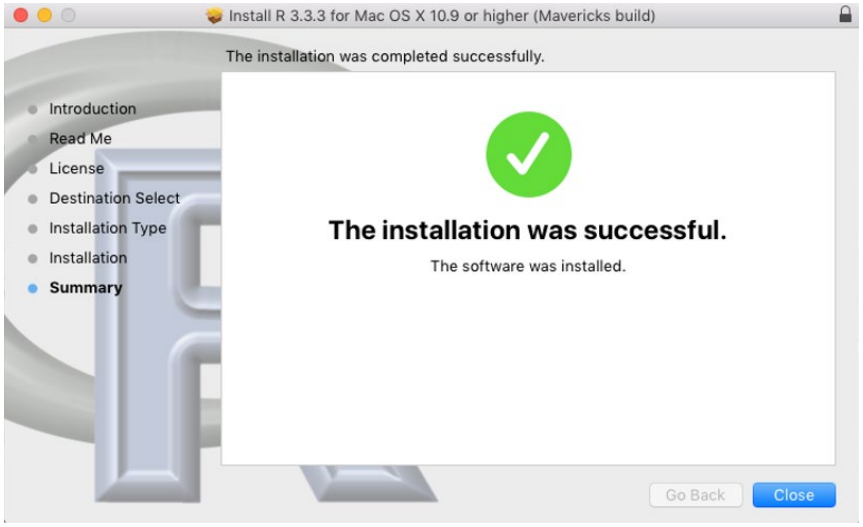
4. On the CRAN Mirrors page, scroll down to the closest CRAN mirror (replicate server) to your region, select it, and then click **OK**.
5. On the Comprehensive R Archive Network page, click the R download link for the Mac:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

6. On the R for Mac OS X page, click **R-3.3.3.pkg**, which contains binaries for a base distribution and packages to run on Mac OS X (release 10.9 and higher). This package also runs the Fluidigm-required Mac operating system OS X 11 (Sierra).
7. Click the R-3.3.3 downloaded package that you see in the bottom left of your screen.
8. In the **Install R 3.3.3 for Mac OS X 10.9 or highter (Mavericks build)** installer wizard, follow the instructions to set up R 3.3.3 (2017-03-06), beginning with the Introduction:



When you are finished with all the steps in the wizard, you see a Summary:



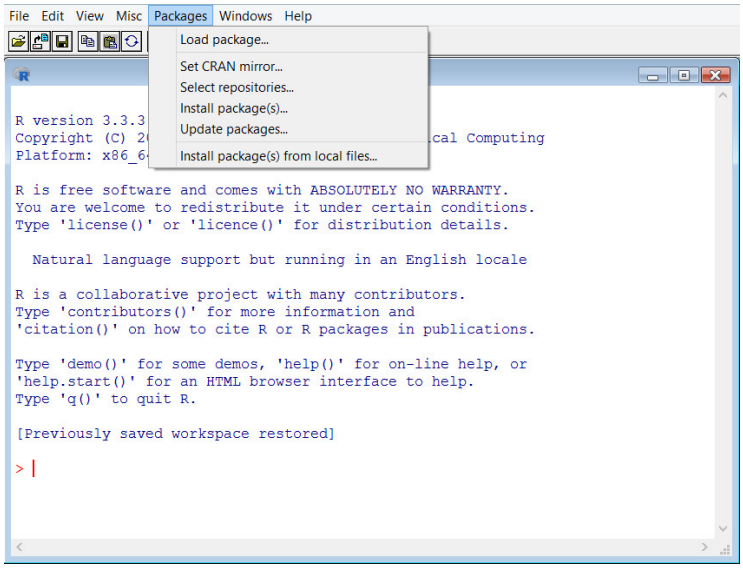
When you click **Close**, you are prompted to move the installer wizard to the Trash (**Move to Trash**) or keep it in its current location (**Keep**).

Install the Singular Analysis Toolset on Windows

1 Download Singular Analysis Toolset Software v3.6 from https://www.fluidigm.com/software.

2 Launch **R**. The R graphical user interface (GUI) displays.

3 On the Windows computer, click **Packages** > **Install package(s)** from local zip file:



4. In the R console, navigate to the **fluidigmSC_v 3.6.[#].zip** (for example, **fluidigmSC_3.6.2.zip**), and click **Open**.

5. At the **R** command prompt, type in **library(fluidigmSC)** and press **Enter**.

6. 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.
- **tcltk2**. The Tool Command Language (tcl) scripting language that includes the Tool Kit (tk) library of GUI elements, such as listboxes, scrollbars, and sliders.
- **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).

- **R.utils**. A set of utility functions whose application programming interfaces (API) methods and classes are used internally by other packages on CRAN. (Other APIs might require updates to R.utlils.) See Bengtsson, H. The R.oo package - Object-Oriented Programming with References Using Standard R Code, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), ISSN 1609-395X, Hornik, K.; Leisch, F. & Zeileis, A. (ed.), 2003.
- **tsne**. A function interface that provides a way to specify the t-distributed stochastic neighbor embedding (t-SNE) dimensionality reduction on R matrices or "dist" objects.

7. 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 Berkeley, CA, choose the **USA (CA 1)** mirror. This ensures that you receive R updates and can access online help.
   - 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.

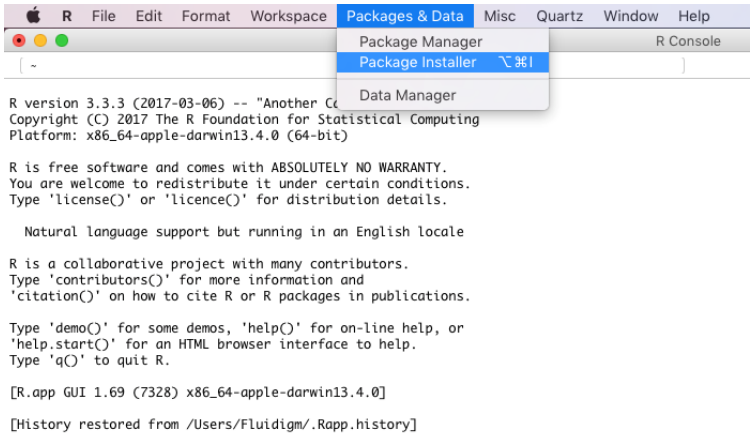
   The R GUI will display a series of messages. Proceed to set the directory for the Singular Analysis Toolset for data analysis.

Install the Singular Analysis Toolset on the Mac

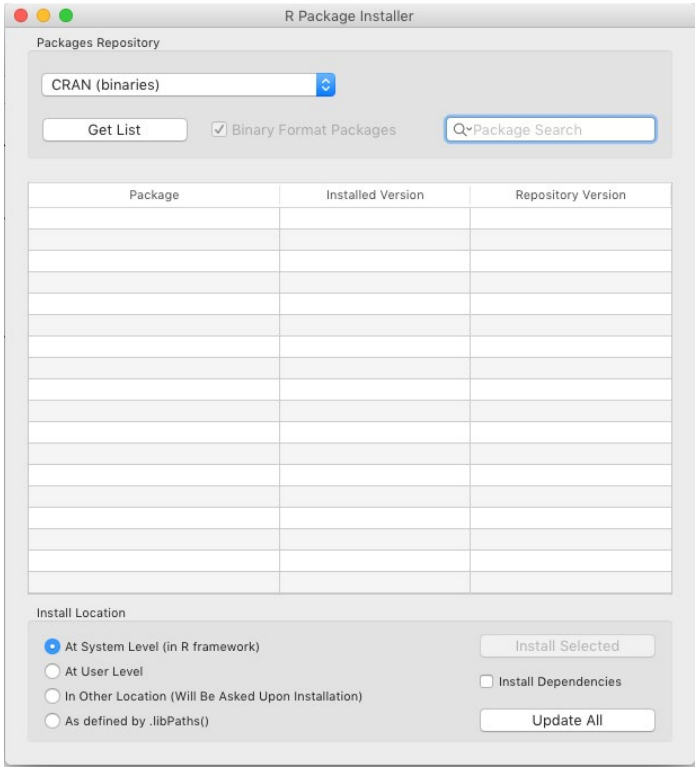
1 Download Singular Analysis Toolset Software v3.6 from https://www.fluidigm.com/software.

2 Launch **R**. The R graphical user interface (GUI) displays.

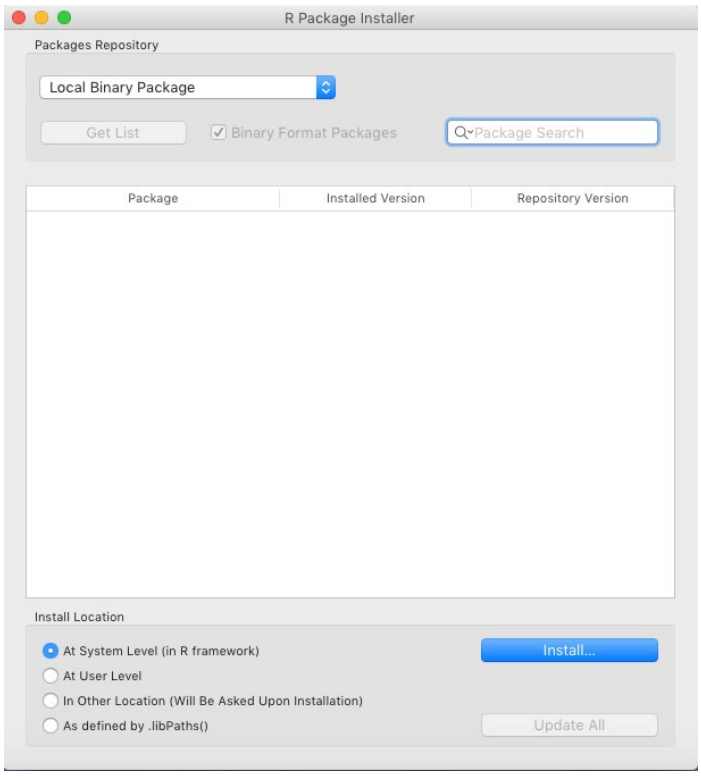
3 On the Mac computer, click **Packages & Data** > **Package Installer**:



In the R Console, you see the R Package Installer window:



4 In the Packages Repository section of the R Package Installer, change the default CRAN (binaries) to **Local Binary Package** and double-click it, which changes the inactive Install Selected button to an active **Install** button:



5 Click **Install**.

6 Navigate to the downloaded **fluidigmSC.3.6.[#].zip** (for example, **fluidigmSC_3.6.2.zip**), and click **Open**.

7 Close the R Package Installer window (even though the Install button is still in the active mode). In the R command line of the R Console, you see the message:

```
Package 'fluidigmSC' successfully unpacked and MD5 sums checked
```

8. At the **R** command prompt, type in **library(fluidigmSC)** and press **Enter**.

9. 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.
- **tcltk2**. The Tool Command Language (tcl) scripting language that includes the Tool Kit (tk) library of GUI elements, such as listboxes, scrollbars, and sliders.
- **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).

- **R.utils**. A set of utility functions whose application programming interfaces (API) methods and classes are used internally by other packages on CRAN. (Other APIs might require updates to R.utlils.) See Bengtsson, H. The R.oo package - Object-Oriented Programming with References Using Standard R Code, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), ISSN 1609-395X, Hornik, K.; Leisch, F. & Zeileis, A. (ed.), 2003.
- **tsne**. A function interface that provides a way to specify the t-distributed stochastic neighbor embedding (t-SNE) dimensionality reduction on R matrices or "dist" objects.

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

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 for the purpose of analysis. For example, a set of co-regulated genes.

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 set of variants for the purpose of analysis. For example, 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 avoid confusion, it is best to keep the character cases consistent.
- To improve the display of sample names in analysis plots, use short names.

Restrictions

Samples and gene names have non-alphanumeric character restrictions:

"Allowed"	Examples of "Not Allowed"
_ (underscore)	, (comma)
- (hyphen)	; (semi-colon)
< > (inequality signs)	/ (forward slash)
( ) (parentheses)	* (asterisk)
[ ] (brackets)	^ (caret)
& (ampersand)	+ (plus sign)

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

https://www.r-project.org

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.

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:

```
object <- function (arguments(s))
```

Thus:

```
object <- file/data calculation to use that object
```

Example 1: Creating an Object from a File

```
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

```
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 displayed 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

```
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:

`print(`*enter your function here*`)`

For example, to display the R code used for calculating and displaying an HC plot, enter:

```
print(HC)
```

Alternatively, you can type the following:

```
HC
```

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

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

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

```
> names(exp)
 [1] "obj_type"        "data_type"       "lod"             "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 object

```
object$dataframe
```

For example: `exp$gene_list`

Get R Help

Get a Specific Help Function

To get specific help for a specific function, type **?functionName** (for example, **?autoAnalysis**).

Get the Complete Set of Fluidigm Help Functions

To get the set of Fluidigm help functions, type **?fluidigmSC** at the R command line in the R console. Alternatively:

- On a Windows computer: type **??fluidigmSC** at the R command line. Your internet browser appears. At the upper left of your internet browser, click the link **fluidigmSC::fluidigmSC**. At the bottom right, click the link **Index**.
- On a Mac computer, select **Packages & Data > Package Manager**, and then select **Fluidigm Singular Analysis**.

Get Only the Expression or Variant Functions

To get the set of expression function, type **scExpFunctions()**. To get the set of variant functions, type **scVarFunctions()**. In each case, type in the number that corresponds to the category for the set of functions that you want to display. You see all the functions and arguments for that category.

Get the Complete Set of Available Variables for the Session

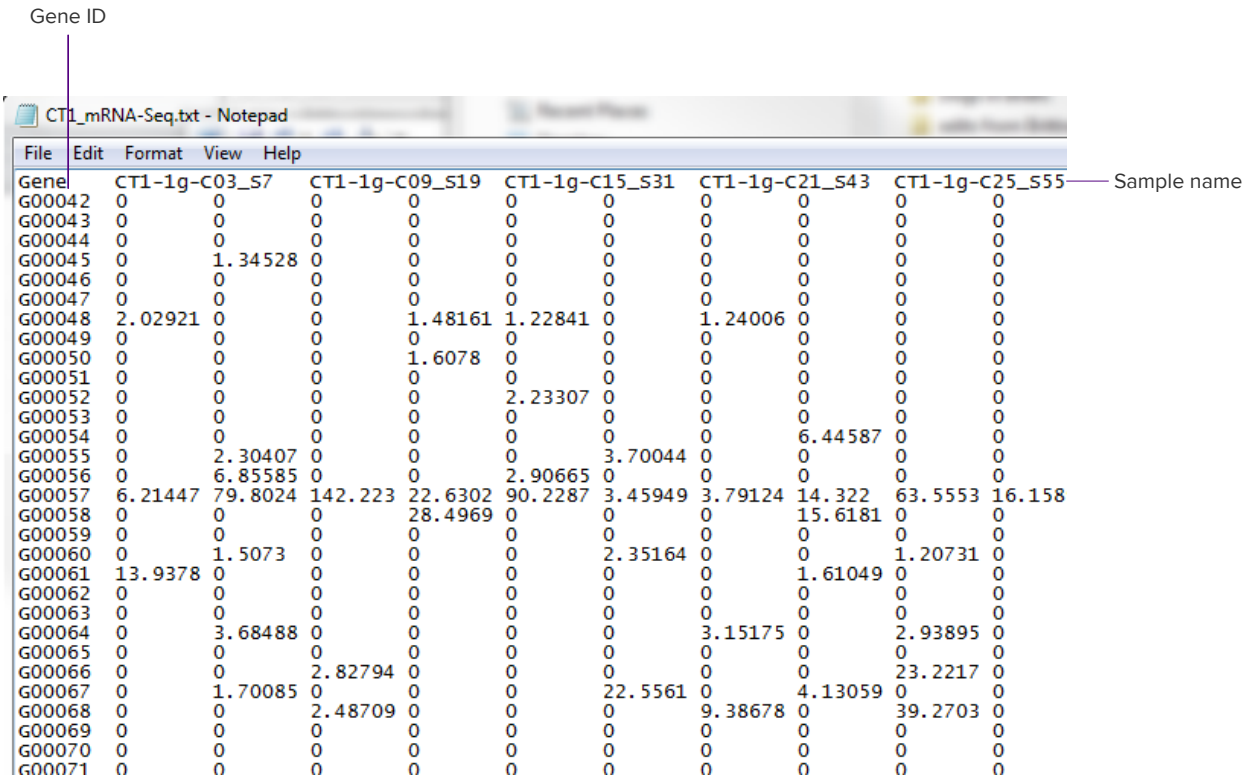
To get the available set of variables that can be used for the current session (such as anova, hc, pca, and tsne), type **scVariables()**.

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.

Prepare mRNA Seq Data

Singular accepts a tab-delimited matrix file for mRNA Seq analysis. The first row contains the sample names and the first column contains the gene annotation (Gene ID, Isoform ID, and so forth). The expression values of an mRNA Seq analysis can be either TPM (Transcripts Per Million), FKPM (Fragments per Kilobase of transcript Per Million mapped reads) or RPM (Reads Per Million). TPM is recommended for single-cell analysis using C1 mRNA Seq IFC, and RPM is recommended for single-cell analysis using C1 mRNA Seq HT IFC. The mRNA Seq expression file format is illustrated as follows:



Gene	CT1-1g-C03_S7	CT1-1g-C09_S19	CT1-1g-C15_S31	CT1-1g-C21_S43	CT1-1g-C25_S55
G00042	0	0	0	0	0	0	0	0	0	0
G00043	0	0	0	0	0	0	0	0	0	0
G00044	0	0	0	0	0	0	0	0	0	0
G00045	0	1.34528	0	0	0	0	0	0	0	0
G00046	0	0	0	0	0	0	0	0	0	0
G00047	0	0	0	0	0	0	0	0	0	0
G00048	2.02921	0	0	1.48161	1.22841	0	1.24006	0	0	0
G00049	0	0	0	0	0	0	0	0	0	0
G00050	0	0	0	1.6078	0	0	0	0	0	0
G00051	0	0	0	0	0	0	0	0	0	0
G00052	0	0	0	0	2.23307	0	0	0	0	0
G00053	0	0	0	0	0	0	0	0	0	0
G00054	0	0	0	0	0	0	0	6.44587	0	0
G00055	0	2.30407	0	0	0	3.70044	0	0	0	0
G00056	0	6.85585	0	0	2.90665	0	0	0	0	0
G00057	6.21447	79.8024	142.223	22.6302	90.2287	3.45949	3.79124	14.322	63.5553	16.158
G00058	0	0	0	28.4969	0	0	0	15.6181	0	0
G00059	0	0	0	0	0	0	0	0	0	0
G00060	0	1.5073	0	0	0	2.35164	0	0	1.20731	0
G00061	13.9378	0	0	0	0	0	0	1.61049	0	0
G00062	0	0	0	0	0	0	0	0	0	0
G00063	0	0	0	0	0	0	0	0	0	0
G00064	0	3.68488	0	0	0	0	3.15175	0	2.93895	0
G00065	0	0	0	0	0	0	0	0	0	0
G00066	0	0	2.82794	0	0	0	0	0	23.2217	0
G00067	0	1.70085	0	0	0	22.5561	0	4.13059	0	0
G00068	0	0	2.48709	0	0	0	9.38678	0	39.2703	0
G00069	0	0	0	0	0	0	0	0	0	0
G00070	0	0	0	0	0	0	0	0	0	0
G00071	0	0	0	0	0	0	0	0	0	0

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 Delta Gene 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 data:

	A	B	C	D	E	F	G	H	I	J	K
1	Chip Run I	c:\BioMar	1.36E+09	96.96 (136	GE 96x96 F	ROX	EvaGreen	1/9/2013 14:51	0:33:31	BIOMARKHD081	
2	Applicatic	4.0.1									
3	Applicatic	20130423									
4	Export Typ	Table Results									
5	Quality Th	0.65									
6	Baseline C	Linear (Derivative)									
7	Ct Thresho	Auto (Global)									
8											
9											
10	Experimer	Experimer	Experimer	Experimer	Experimer	Experimer	EvaGreen	EvaGreen	EvaGreen	EvaGreen	EvaGre
11	Chamber	Sample	Sample	Sample	EvaGreen	EvaGreen	Ct	Ct	Ct	Ct	Tm
12	ID	Name	Type	rConc	Name	Type	Value	Quality	Call	Threshold	In Rang
13	S96-A01	CT1_94-1	Unknown	1	G01	Test	8.268477	1	Pass	0.018799	84.063
14	S96-A02	CT1_94-1	Unknown	1	G02	Test	999	0	Fail	0.018799	9
15	S96-A03	CT1_94-1	Unknown	1	G03	Test	999	0	Fail	0.018799	9
16	S96-A04	CT1_94-1	Unknown	1	G04	Test	19.36748	1	Pass	0.018799	88.840
17	S96-A05	CT1_94-1	Unknown	1	G05	Test	999	0	Fail	0.018799	9
18	S96-A06	CT1_94-1	Unknown	1	G06	Test	12.31763	1	Pass	0.018799	81.647
19	S96-A07	CT1_94-1	Unknown	1	G07	Test	18.78365	1	Pass	0.018799	86.268
20	S96-A08	CT1_94-1	Unknown	1	G08	Test	999	0	Fail	0.018799	9

Some Guidelines for Data Preparation

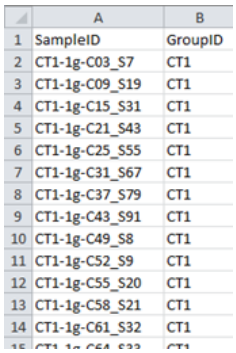
The Singular Analysis Toolset does not normalize data, since there is no definitive way to analyze burst-like expression for single cells. Normalization of data will be a future consideration.

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:



	A	B
1	SampleID	GroupID
2	CT1-1g-C03_S7	CT1
3	CT1-1g-C09_S19	CT1
4	CT1-1g-C15_S31	CT1
5	CT1-1g-C21_S43	CT1
6	CT1-1g-C25_S55	CT1
7	CT1-1g-C31_S67	CT1
8	CT1-1g-C37_S79	CT1
9	CT1-1g-C43_S91	CT1
10	CT1-1g-C49_S8	CT1
11	CT1-1g-C52_S9	CT1
12	CT1-1g-C55_S20	CT1
13	CT1-1g-C58_S21	CT1
14	CT1-1g-C61_S32	CT1

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."

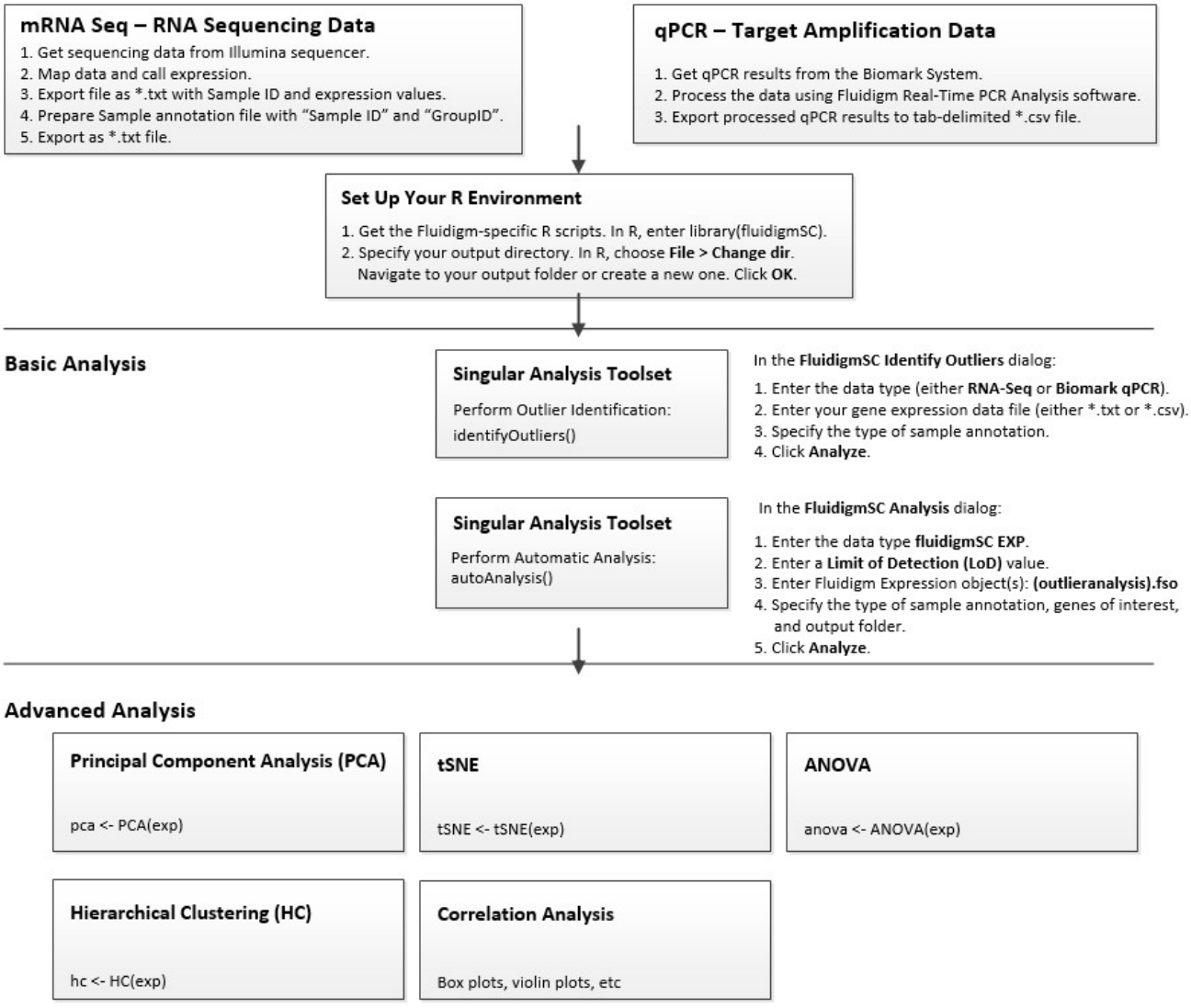
IMPORTANT When you open a tab-delimited file in Excel, Excel uses the "General" format for each cell by default. This is a problem for some gene names, which are converted to a Date. Thus, it is important to assign the Gene ID column as "Text" in the import wizard.

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.

Workflow for Gene Expression with the Singular Analysis Toolset



Log_2EX and Limit of Detection (LoD)

The Singular Analysis Toolset performs all statistical analyses based on the expression values in the Log_2 domain. It supports expression values (TPM, FPKM, or RPM in the linear domain) in mRNA sequencing experiments and C_t values in qPCR experiments.

Log_2EX and LoD in mRNA Seq Experiments

The input of expression levels in RNA Seq experiments are in the linear domain. Since the primary interest in expression analysis is the fold change, these values are converted to Log_2 values (Log_2Ex). Because small numbers (less than 1) can become very large negative values in the Log_2 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 Log_2 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 Log_2Ex by the following formula:

$Log_2Ex = Log_2(exp)$ if $\mathbf{exp} \geq LoD$ or

$Log_2Ex = Log_2(LoD)$ if $\mathbf{exp} < LoD$

Log_2EX 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 Log_2Ex 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 Log_2 by defining the term Log_2Ex:

$Log_2Ex = LoD\ C_t - C_t\ [Gene]$

If the value is negative, `Log2Ex = 0`

Log_2Ex 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_2Ex}, 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_2Ex), 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 by using the C_t value representing either a single copy of a transcript or the last cycle of PCR. For Biomark™ and Biomark HD™ systems, the recommended LoD default value is 24, as suggested by Livak et al. 2013. Log_2Ex values are as follows:

- If $Ct \leq LoD$, then $Log_2Ex = LoD - Ct$.
- If $Ct > LoD$, then $Log_2Ex = 0$.

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 Ct 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 Ct 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.

Missing Data Conversion. As most of the analysis functions in Singular require valid numerical values for all the data points, missing data will be converted to Log_2(LoD) for mRNA Seq results and 0 for qPCR results.

Example 1: Data for an LoD of 24 for qPCR

CT	PASS/FAIL	Converts to (Ct))	Log2Ex
20	Pass	20	4
22	Fail	-1	0
26	PASS	LoD	0
999	Fail	LoD	0

Example 2: Data for mRNA Seq data LoD =1

Expression	Converts to (Linear)	Log2Ex
10	10	3.321928
0.05	LoD	0
BLANK	-1	0
0	LoD	0

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.

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

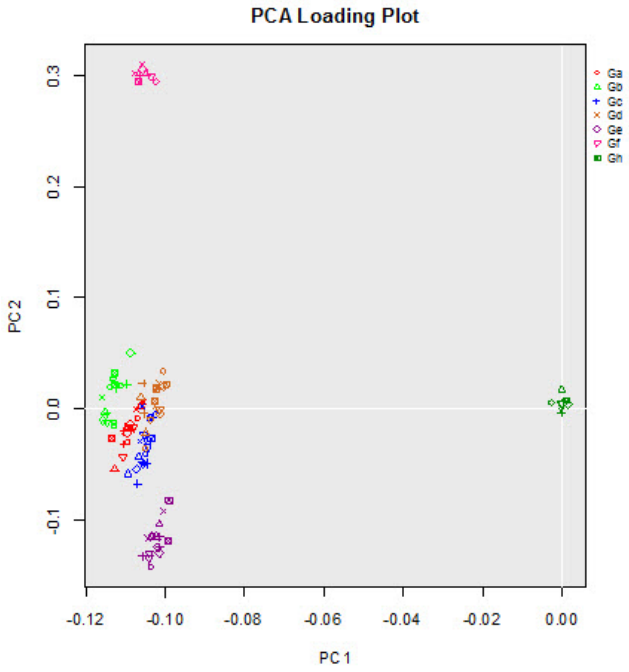
data set. It further permits you to define the number of components, or axes, used to analyze the data set.

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.

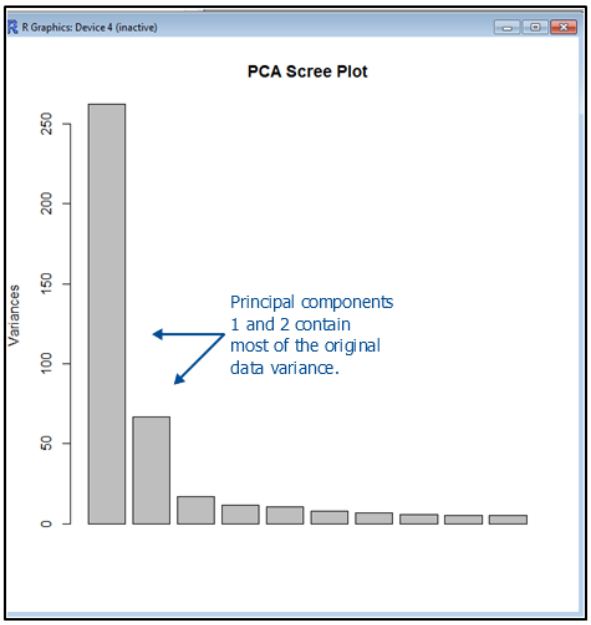
  You can also generate a PCA Loading plot at this step.

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



- **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:

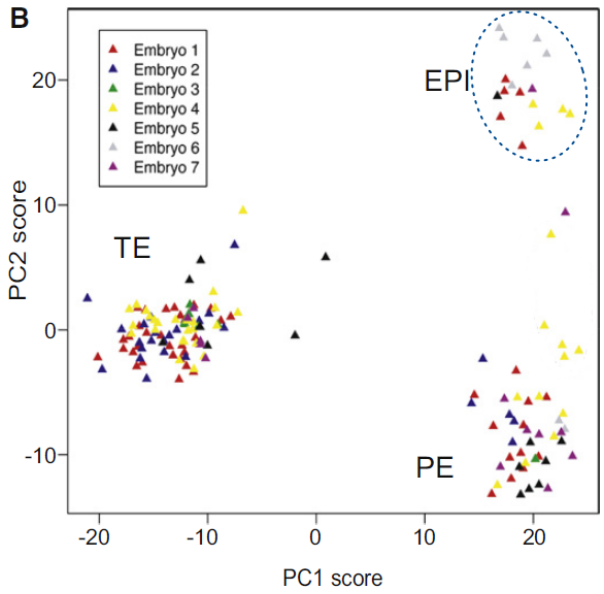


A scatter plot of PC scores can also provide interesting insights into data, as described in the case study here:

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. (This image was reprinted with permission from Developmental Cell.)



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.

t-Distributed Stochastic Neighbor Embedding (tSNE)

t-distributed stochastic neighbor embedding (tSNE) is a non-linear, dimensionality-reduction technique that takes a set of data points in a high-dimensional (high-D) space and finds faithful representation of those points in a lower-dimensional (low-D) space. The low-D space is typically a two-dimensional scatter plot with discernable clusters at several scales. In such a scatter plot, tSNE can capture much of the local structure of the high-D data while also revealing its global structure. tSNE is a discovery tool. Typically, it requires multiple tries to find your answer. The Singular Analysis Toolset provides three parameters to help you to tune your tSNE plot:

Initial Dimension. This parameter determines whether you want to perform an initial dimension reduction by PCA or not. By default (-1), there is no dimension reduction before performing the tSNE analysis. If the data has not been processed with PCA and/or ANOVA, you might want to set the initial dimension. The typical values for the initial dimension range from 100 to 400.

Iterations. A tSNE plot is made with a default number of 2000 iterations. Increase this number if the clusters are not converged at the end or reduce this number to accelerate the analysis if the clusters are converged quickly. In general, iterate until your outcome reaches a stable configuration. Singular reports the residual error every 50 iterations.

Perplexity. This parameter allows you to balance the local and global aspects of your data by setting the number of effective nearest neighbors. With a lower perplexity value, local variations dominate. In general, a larger (denser) dataset requires a larger perplexity. If the clusters are not well separated at the end of the analysis, try increasing and decreasing the perplexity value. If your analysis produces all equidistant points, try increasing the perplexity value. Typical values for the perplexity range between 5 and 50. (The default (-1) sets the value by dividing the number of data points by 10.)

TIPS:

(1) tSNE does not produce the same plot outcome upon successive runs with the same data and parameters. Thus, compare their residual error and select the solution with the lowest error.

(2) Outliers might significantly impact the tSNE analysis. If you want to study the structures of a set of cells with subtle differences in gene expression, perform the tSNE analysis only on the cells of interest.

Clustering. Cluster a tSNE result with k-mean clustering methods using an optional graphical user interface (GUI) for changing number of clusters:

K + 1. Increase the number of clusters by 1.

K – 1. Decrease the number of clusters by 1.

Retry. Recluster the data using different initial cluster centers.

Save. Save the selected cluster results (sample_list: sample id + cluster id).

Done. End your clustering session when you are finished clustering.

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

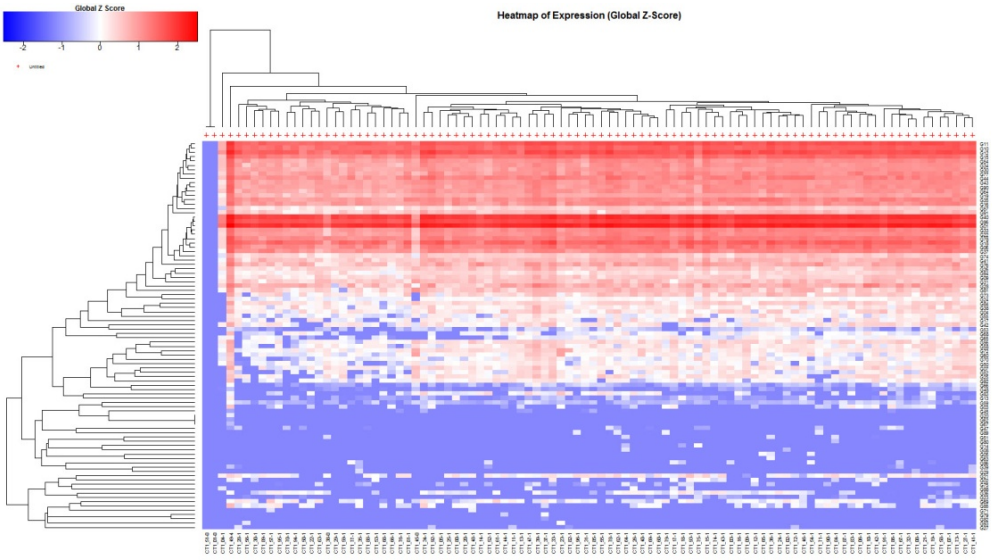
visual representations of genes where there are large, statistically significant changes between two sample groups. These plots have a Log2 fold change for all genes plotted on the X-axis and the negative Log10 p-value plotted on the Y-axis. The data points for the highly significant genes (low p-values 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.

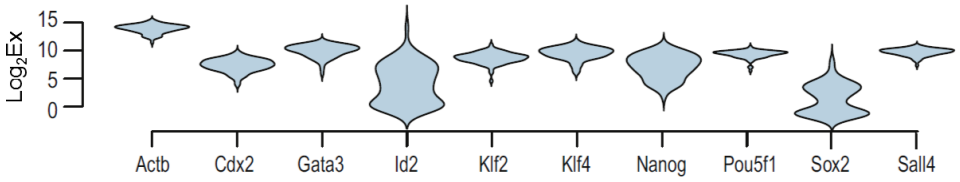
Visualization of Expression Profiles

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.
- **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.

A Study: Insights into Visualization with Violin Plots

The following figure shows a Violin plot from Guo et al. (2010) that compares 10 genes in 75 single cells derived from 16-cell stage mouse embryos. (This image was reprinted with permission from Developmental Cell.)



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 Log_2Ex axis) than *Cdx2*. This is because each gene has a characteristic transcriptional burst size, frequency, and decay rate.

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.

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 50th percentile expression value for the set of data.

The **Outlier Threshold** is defined as the expression value that is the 15th 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:

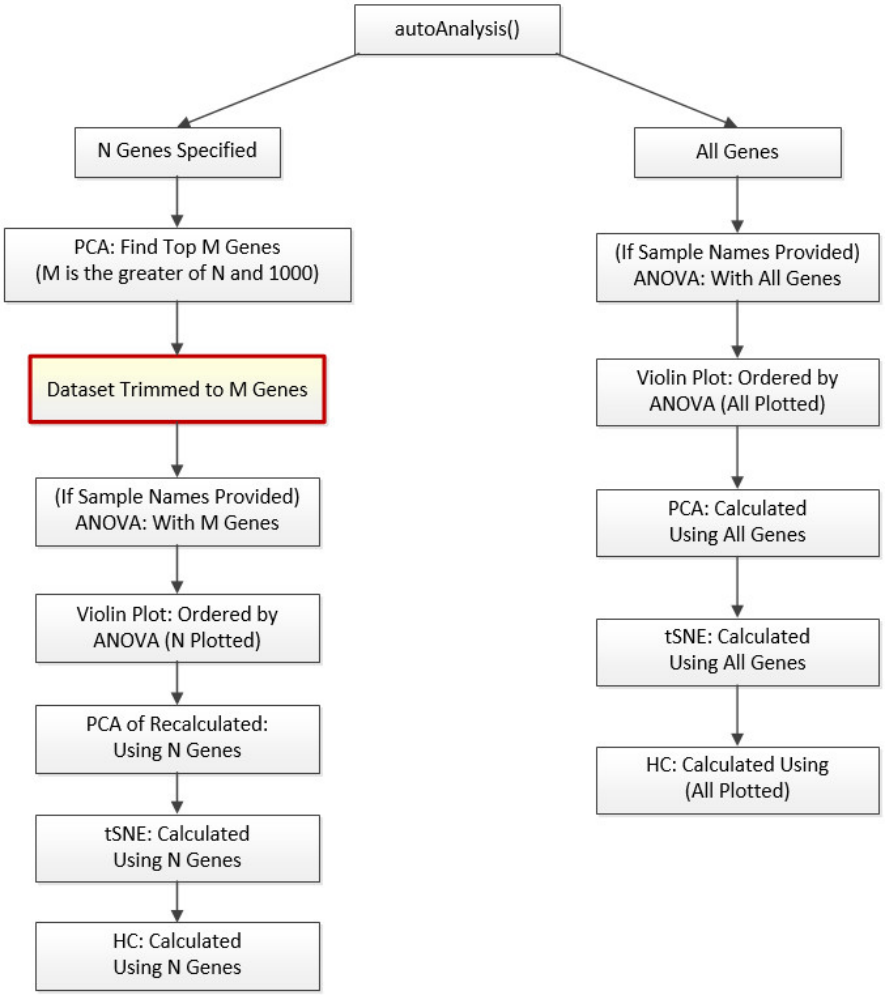
- Runs PCA and displays a 2D PCA Score plot, ANOVA (if sample groups are provided), a tSNE plot, Hierarchical Clustering and with a heatmap, and generates Violin plots for the top-ranked genes. Results are expressed as Log_2Ex values and exported to *.fso files and saved in the working directory set for the session. In this file, the Log_2Ex 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.
- 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 and ranked by ANOVA scores (when sample groups are provided). Lastly, a Hierarchical Clustering (HC) heatmap is displayed with the number of genes selected. If more than 1000 genes are specified, a warning message will be displayed.
- Saves (1) the gene lists, (2) the EXP, PCA, HC, and tSNE objects (in the output folder with the .fso filename extension) as the global variables fldm_exp, fldm_pca, fldm_hc, and fldm_tsne for further analysis, (2) the PCA Gene Ranking Score, and the (4) ANOVA p-value (if sample group information is provided).

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. Your choices of data types are:

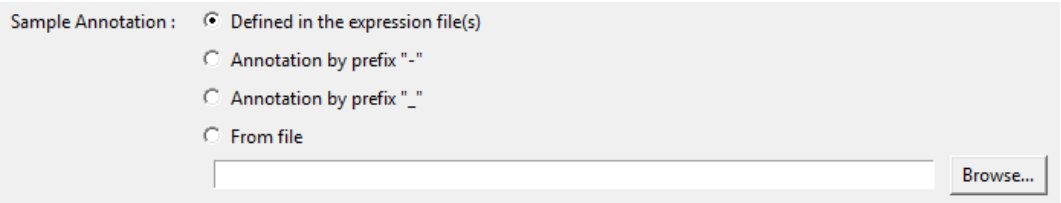
- **EXP**. Upload an existing *.fso file (created after performing the outlier identification procedure. For more information, see Performing Outlier Identification. The LoD in the case will have been defined during the identifyOutliers() function and does not need to be defined again here.
- **RNA-Seq**. (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.

  For example:

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

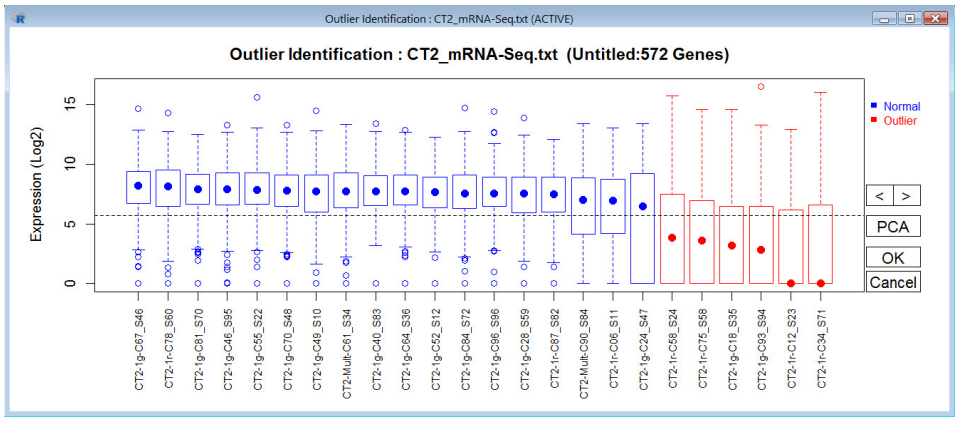
In the Sample Annotation section, select the way that you want sample group information to be annotated. You can add to the sample list with sample group annotation by sample prefix (**Annotation by prefix "-"** or **Annotation by prefix "_"**). The group information must be BEFORE the sample name, and 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.



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

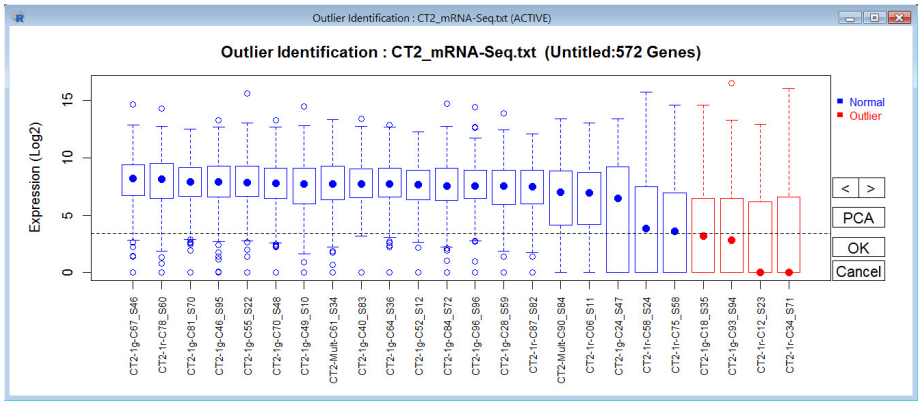
For example, you see:



7. To change Normal (, blue) data point values to Outliers (, red) or vice versa, do either of the following:

   - Click a single data point (suh as , blue). The color for the single data point is changed (such as , red). (To change it back, click the single data point again.)
   - Use the left (reduce) and right (increase) interface icons < > to move Expression threshold on the Y axis:

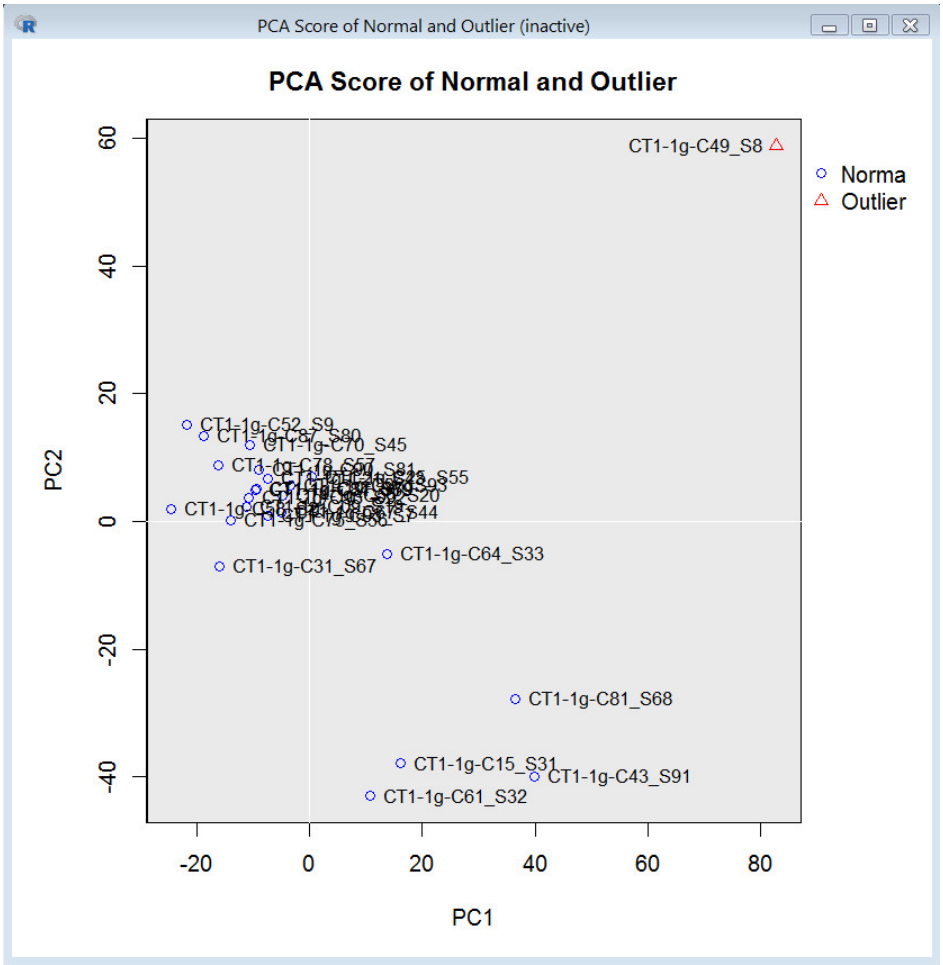
     For example, the left interface icon was clicked 9 times to lower the threshold, which placed 2 of the outliers in the normal range:



     In addition, the command line in the R console displays the Y-axis content in text format:

```
Reduce threshold from 5.689 to 5.439.
Reduce threshold from 5.439 to 5.189.
Reduce threshold from 5.189 to 4.939.
Reduce threshold from 4.939 to 4.689.
Reduce threshold from 4.689 to 4.439.
Reduce threshold from 4.439 to 4.189.
Reduce threshold from 4.189 to 3.939.
Reduce threshold from 3.939 to 3.689.
Reduce threshold from 3.689 to 3.439.
```

8. 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:



9. 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.

10. 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, see any of the following:

- Appendix A: The List of Functions for Gene Expression.

- The section Getting R Help.
- The section **Without Automatic Analysis** in the section Principal Component Analysis (PCA).

Run the Automatic Analysis

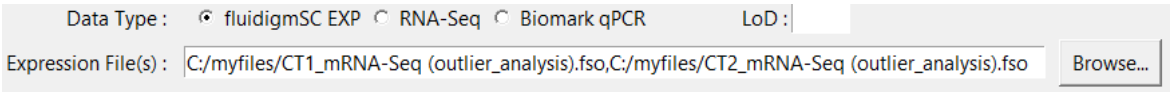
Automated analysis in the Singular Analysis Toolset is a PCA-driven method.

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.)

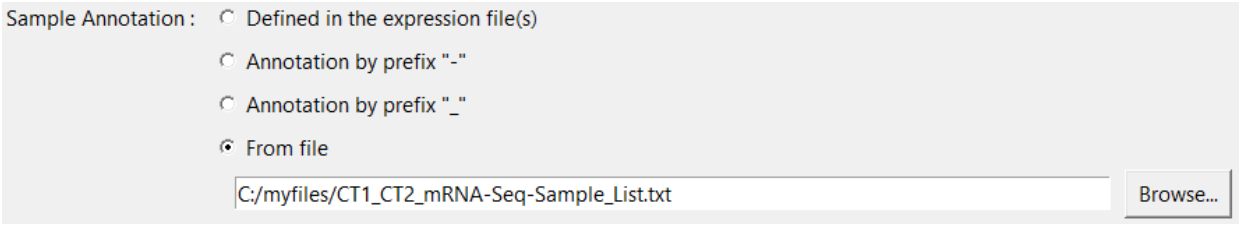


Your choices of data types are:

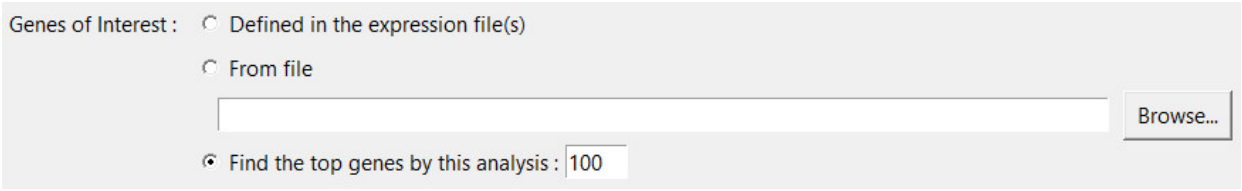
- **fluidigmSC EXP**. Upload an existing *.fso file (created after performing the outlier identification procedure. For more information, see Performing Outlier Identification.) The LoD in the case will have been defined during the identifyOutliers() function and does not need to be defined again here.
- **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, find the genes of interest by one of these ways:



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:

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 the top 1000 genes ranked by PCA Analysis ...
Finding the top 100 differentially expressed genes by ANOVA analysis ...
Performing PCA analysis with the selected genes ...
Performing tSNE analysis with the selected genes ...
  tSNE Parameters:  initial_dims=100  perplexity=5  iteration=2000
```

In addition, the content of the tSNE results is also displayed. For example:

```
sigma summary: Min. : 0.1744 |1st Qu. : 0.2275 |Med
Epoch: Iteration #50 error is: 17.2006261763202
Epoch: Iteration #100 error is: 19.3880224846483
Epoch: Iteration #150 error is: 2.56399320681204
Epoch: Iteration #200 error is: 2.10088834063369
Epoch: Iteration #250 error is: 1.58335903878667
Epoch: Iteration #300 error is: 1.17227316590276
Epoch: Iteration #350 error is: 1.08275295149067
Epoch: Iteration #400 error is: 1.03140632811684
Epoch: Iteration #450 error is: 0.981526128937179
Epoch: Iteration #500 error is: 0.958196630771283
Epoch: Iteration #550 error is: 0.92723010664169
Epoch: Iteration #600 error is: 0.870474447798298
Epoch: Iteration #650 error is: 0.736393813706924
Epoch: Iteration #700 error is: 0.719788919773153
Epoch: Iteration #750 error is: 0.715015143335327
Epoch: Iteration #800 error is: 0.712079315131528
Epoch: Iteration #850 error is:
```

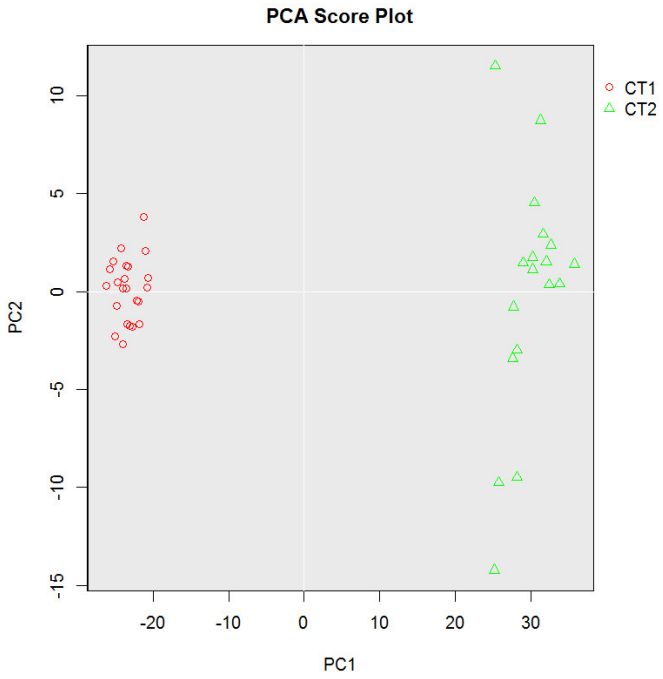
NOTE When multiple expression files are selected, the Singular Analysis Toolset merges them into a single expression object.

Examine the Results

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

PCA Plots

The autoAnalysis function outputs a two-dimensional **PCA Score** plot by default. Each point represents a single cell:



NOTE Outlier identification and elimination before running PCA is highly recommended.

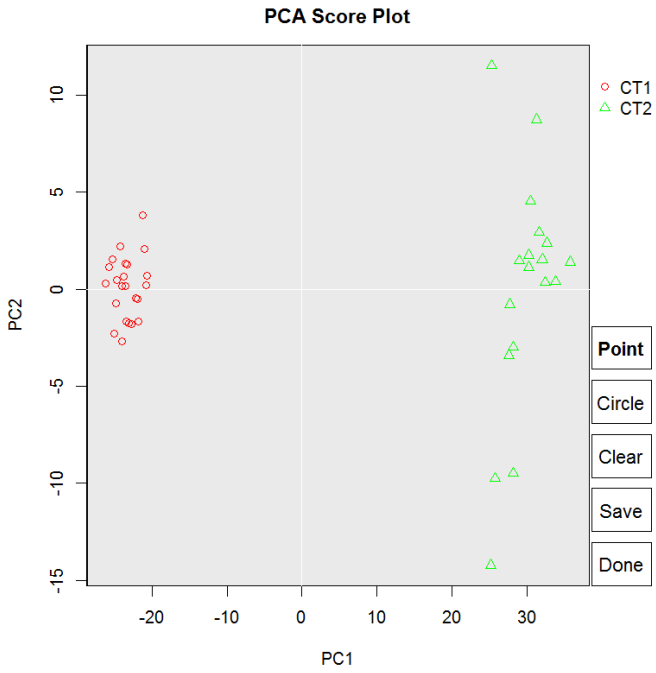
To assist with analysis of clusters, the Singular Analysis Toolset provides you with the ability to identify each point in a PCA plot.

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(),

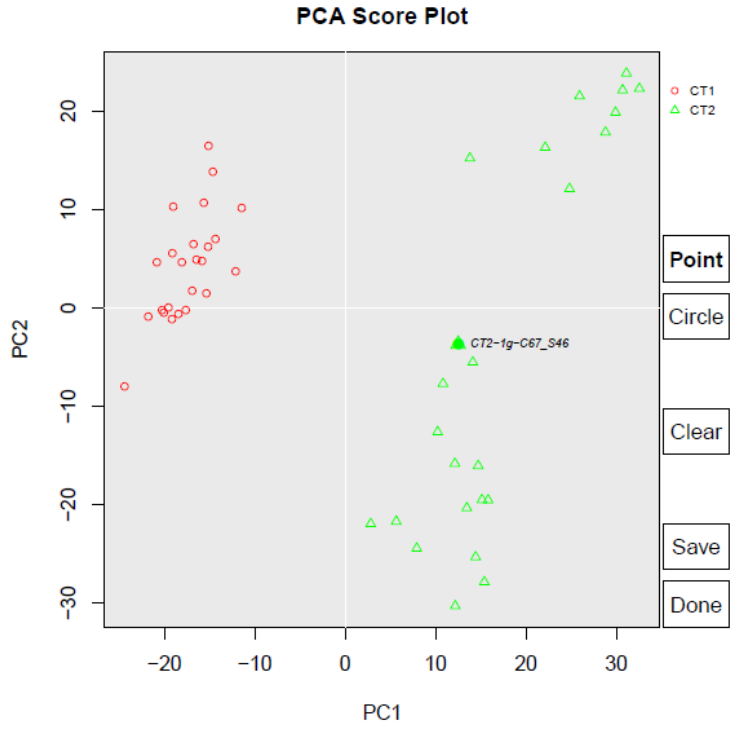
The PCA plot is displayed plot is displayed with several buttons along the right margin. For example:



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. For example:



- 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.

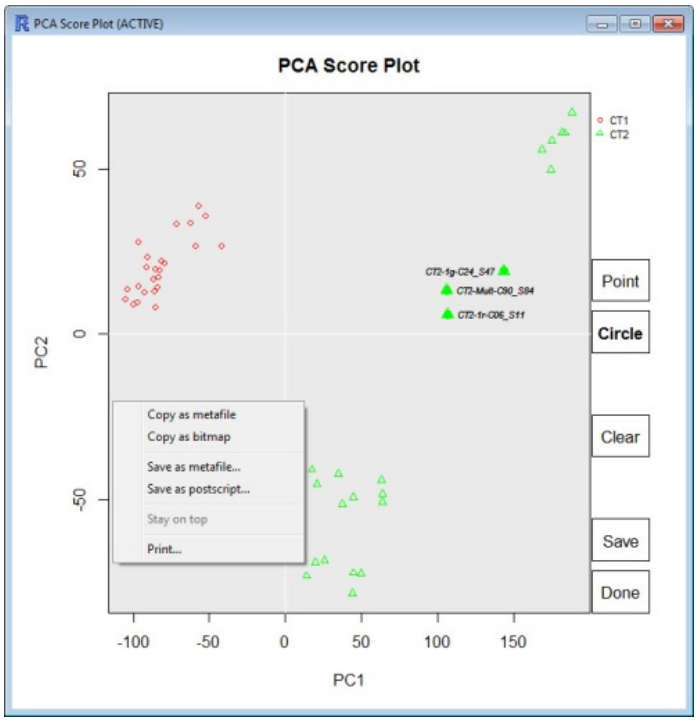
3 You can save the plot, copy it to the clipboard, or print it. Do any of the following:

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.

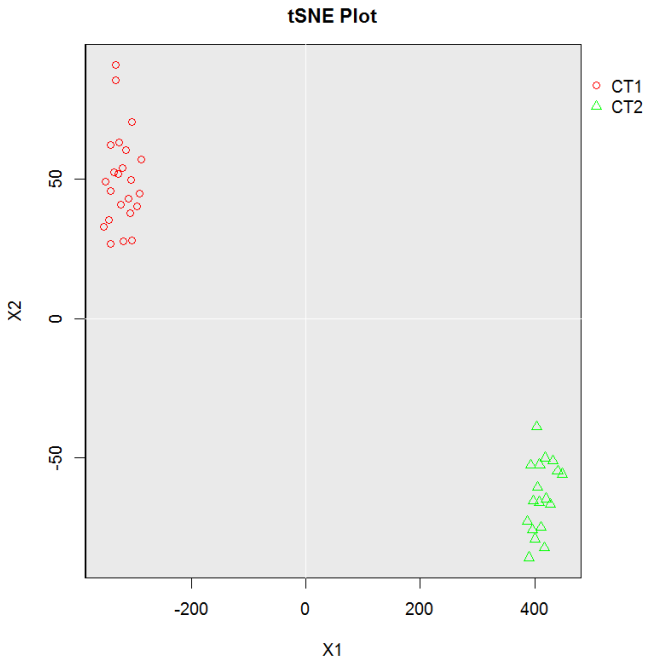
To copy the PCA Score plot to the clipboard, do either of the following:

- 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:



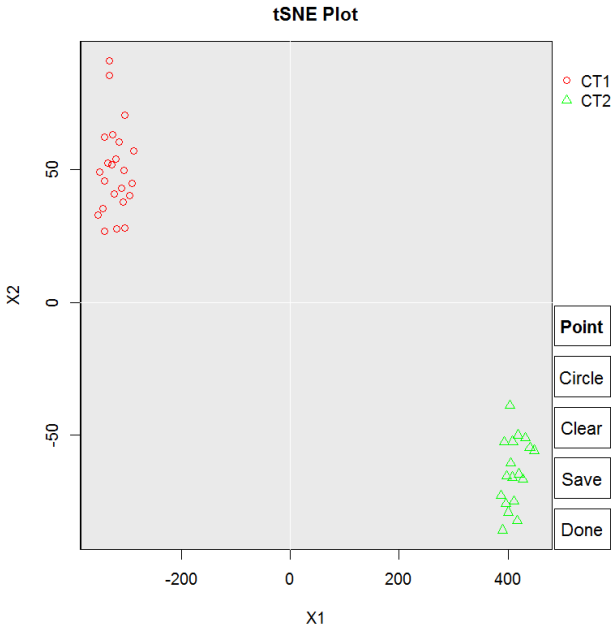
tSNE Plot

The autoAnalysis function outputs a two-dimensional **tSNE** plot by default. The points in the scatter plot are generated algorhthmically upon reduction of a set of data points in a high-dimensional (high D) space and are meant to show discernable sample clusters at several scales:



To assist with analysis of clusters, the Singular Analysis Toolset provides you with the ability to identify individual samples in a tSNE plot.

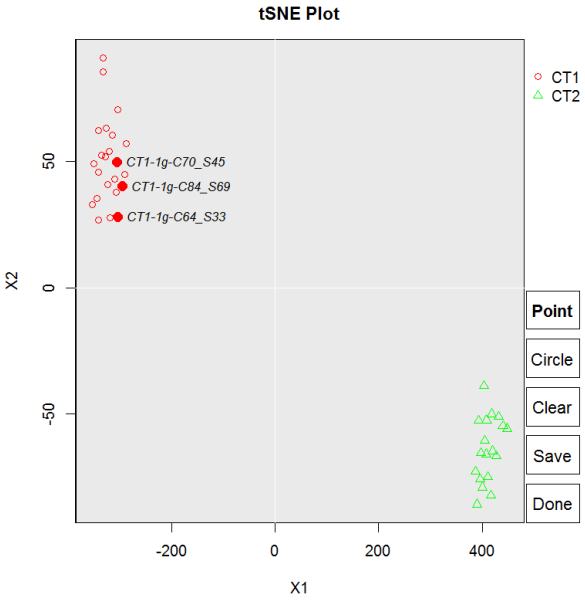
1 At the command line, enter **displaytSNE()**. The tSNE plot displayed with several buttons along the right margin. For example:



2 You can identify samples of interest in a tSNE plot:

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. For example:



- 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, and the values are ranked by score. For example:

```
> displaytSNE()
          SampleID GroupID
19 CT1-1g-C84_S69     CT1
13 CT1-1g-C64_S33     CT1
15 CT1-1g-C70_S45     CT1
```

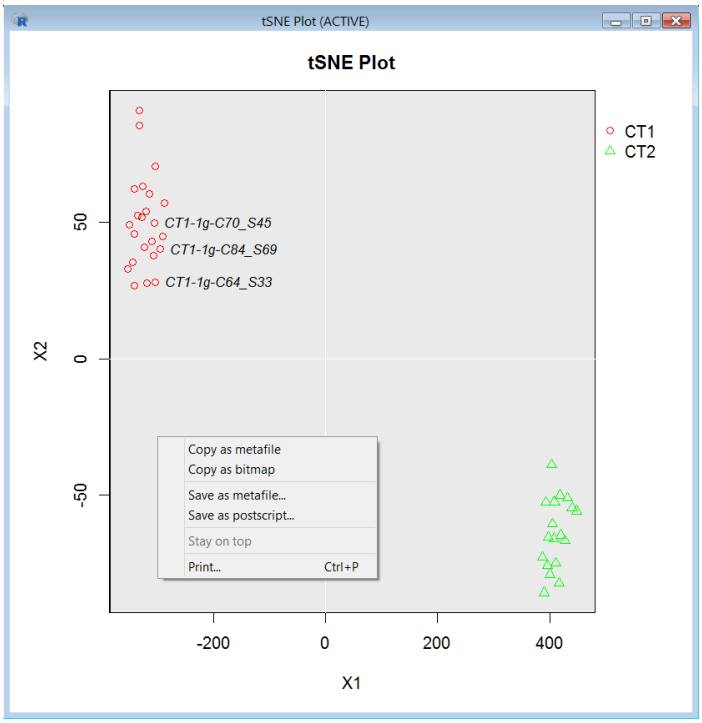
3. You can save the plot, copy it to the clipboard, or print it. Do any of the following:

To save the tSNE Score plot as an image file, choose **File > Save as**, and then click any of the following options:

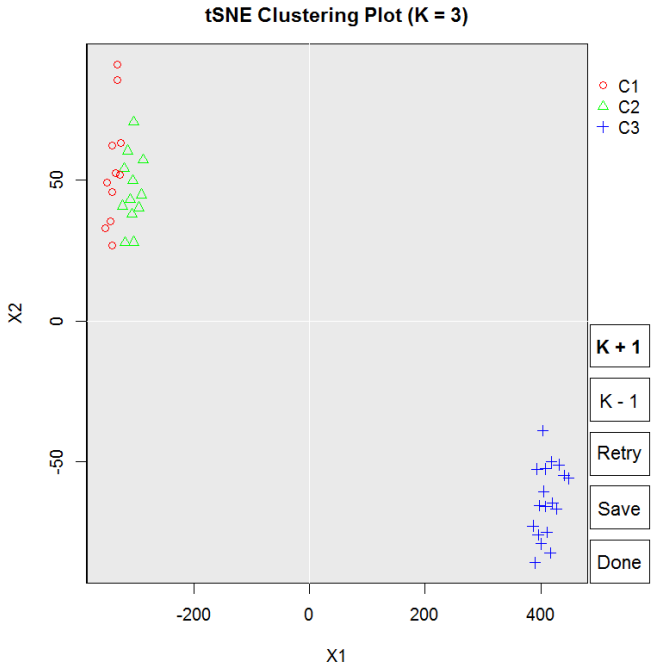
- **Metafile**. Opens a dialog that allows you to save the plot as an enhanced metafile with an *.emf filename extension.
- **Postscript**. Opens a dialog that allows you to save the plot as an encapsulated postscript file with an *.eps filename extension.
- **PDF**. Opens a dialog that allows you to 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 tSNE 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:

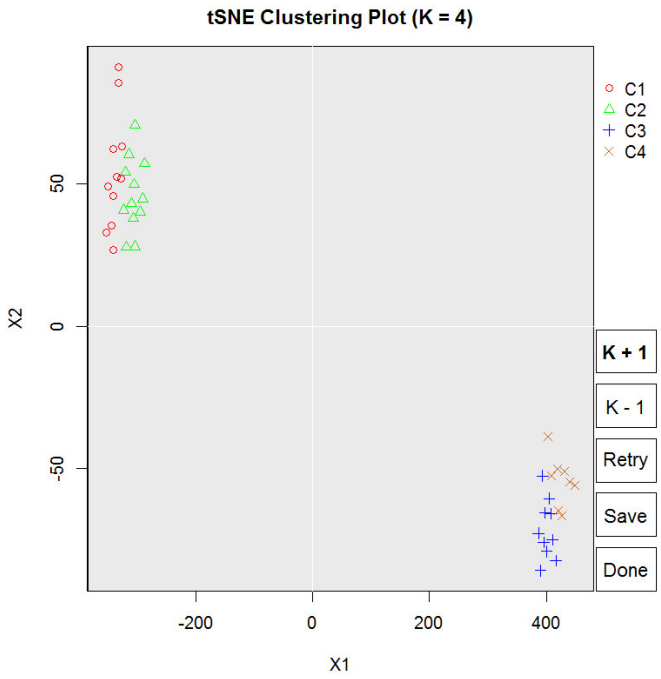


4. To change the number of clusters, enter **clustertSNE()** at the R command line. The tSNE clustering plot is displayed with several buttons along the right margin. For example:



a. Do any of the following:

- To increase the number of clusters by 1, click **K+1**. Repeat as needed. For example:



- To decrease the number of clusters by 1, click **K-1**. Repeat as needed.
- To recluster the data using different initial cluster centers, click **Retry**.
- To save the selected cluster results, click **Save**.

b. When you care done changing the number of clusters, click **Done** to return to the main **R** window.

The **R** window displays the contents of your sample list, and the values are ranked by score. For example:

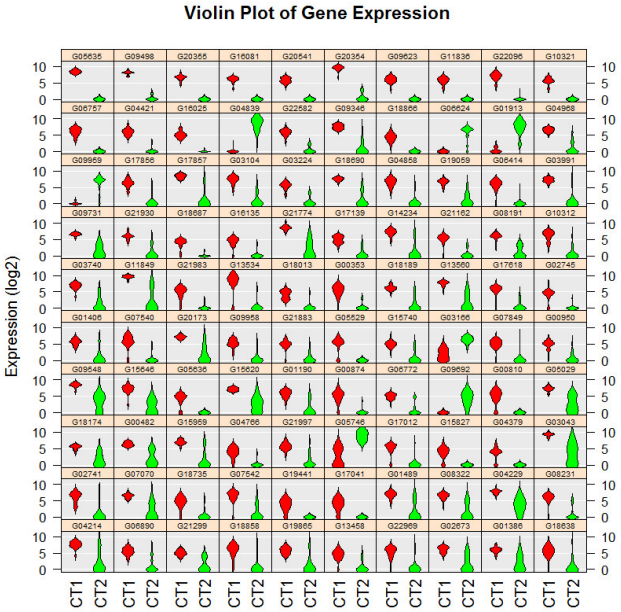
```
  Perform k-mean clustering with 3 clusters ...  Done
  Perform k-mean clustering with 4 clusters ...  Done
          SampleID GroupID OriginalGroupID
1     CT1-1g-C03_S7      C1              CT1
2    CT1-1g-C09_S19      C2              CT1
3    CT1-1g-C15_S31      C2              CT1
4    CT1-1g-C21_S43      C1              CT1
5    CT1-1g-C25_S55      C2              CT1
6    CT1-1g-C31_S67      C1              CT1
7    CT1-1g-C37_S79      C3              CT1
8    CT1-1g-C43_S91      C3              CT1
9     CT1-1g-C52_S9      C1              CT1
10   CT1-1g-C55_S20      C1              CT1
11   CT1-1g-C58_S21      C2              CT1
12   CT1-1g-C61_S32      C2              CT1
13   CT1-1g-C64_S33      C2              CT1
14   CT1-1g-C67_S44      C3              CT1
```

5. You can save the plot, copy it to the clipboard, or print it, as described in step 3.

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

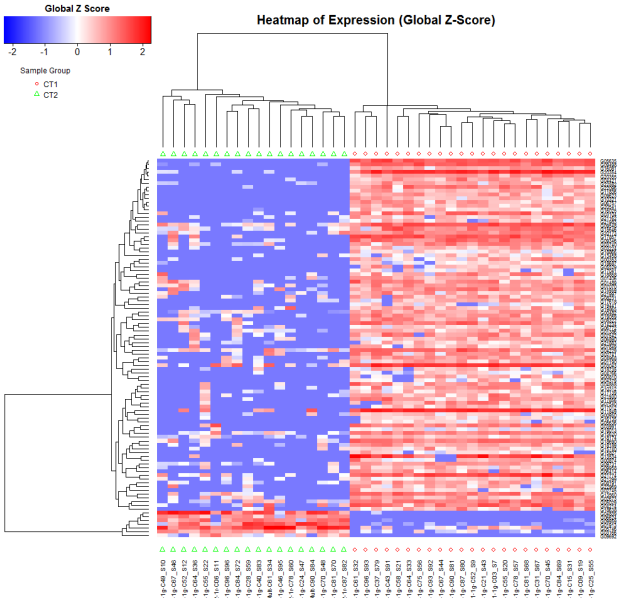
existed, a small p-value (typically <0.05) indicates the possibility of a biological process of interest happening randomly. For example:



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:



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.

GeneID	GroupID	GroupID.x	Anova.pValue	CT2_vs_CT1.pValue	CT1.Average	CT2.Average	GroupID.y	Score(PC1-3)	Rank
G05635	Untitled	Untitled	3.61E-50	0	8.474375529	0	Untitled	0.04807372	12
G09498	Untitled	Untitled	4.20E-41	0	8.090431975	0.251348963	Untitled	0.042251668	31
G20355	Untitled	Untitled	5.23E-38	0	6.575623727	0	Untitled	0.036304633	78
G16081	Untitled	Untitled	4.39E-35	0	6.038817499	0	Untitled	0.034213418	100
G22096	Untitled	Untitled	4.99E-30	0	6.92021624	0.099159883	Untitled	0.038123055	53
G20354	Untitled	Untitled	1.71E-27	0	9.491456385	0.860213614	Untitled	0.052536391	4
G04839	Untitled	Untitled	2.39E-24	0	0.107889335	8.294444679	Untitled	0.046517589	16
G17857	Untitled	Untitled	2.57E-20	0	8.350182257	0.811629315	Untitled	0.037718274	58
G03991	Untitled	Untitled	2.96E-19	0	7.251667136	0.597781138	Untitled	0.037032594	67
G21774	Untitled	Untitled	5.70E-18	0	8.574837858	1.550850434	Untitled	0.036358588	76
G01913	Untitled	Untitled	1.15E-17	0	0.457902512	6.884561125	Untitled	0.047113591	14
G11849	Untitled	Untitled	3.12E-17	0	9.606908888	1.485228787	Untitled	0.04066949	39
G06624	Untitled	Untitled	1.03E-16	0	0	5.937647583	Untitled	0.035406852	87
G13534	Untitled	Untitled	2.60E-16	0	7.936397739	0.242622464	Untitled	0.0418288	32
G08361	Untitled	Untitled	2.82E-11	2.75E-11	10.37351501	3.699265897	Untitled	0.037288579	64
G09959	Untitled	Untitled	6.06E-10	6.06E-10	0.137613626	5.205934446	Untitled	0.066764288	2
G05746	Untitled	Untitled	7.92E-10	7.91E-10	2.527979453	8.215650456	Untitled	0.051892911	6

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

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

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

```
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

Like sample annotation, genes can also be grouped with GroupID..

To update genes with group information

The gene 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, PCA, and tSNE), 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

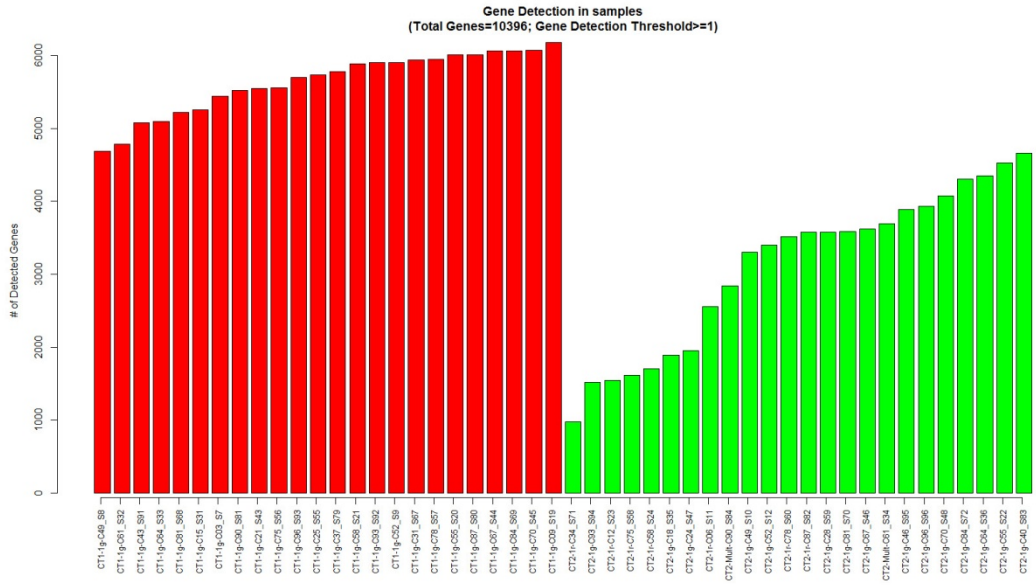
The Excel macro **fluidigmSCObjectToExcel** (100-6988) with an *.xlsm filename extension is a utility to open an *.fso object and convert each data member or frame into an individual worksheet. Each worksheet is a data frame. For more information on the contents of each data frame, see Appendix C: Contents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

```
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):

```
analyzeGeneDetection(exp, threshold)
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:

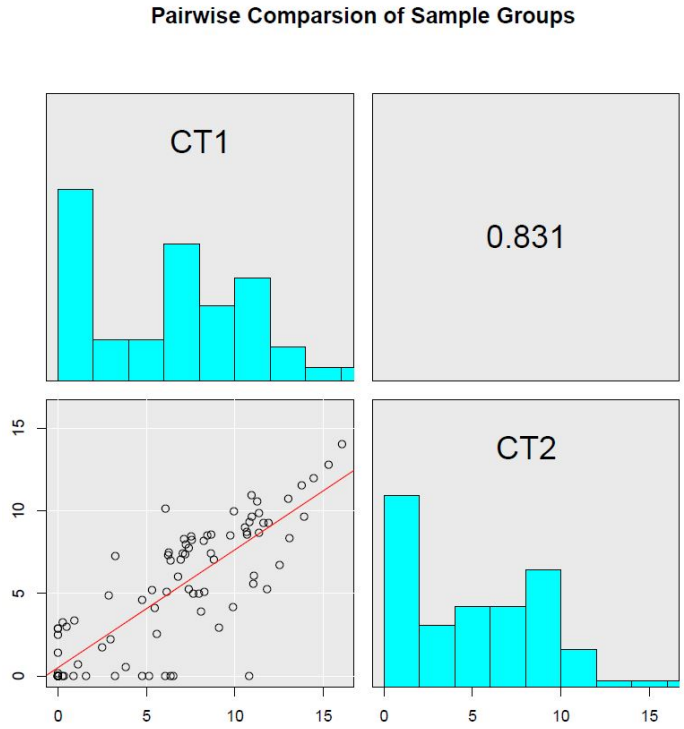
```
violinPlotByGenes (exp, gene_list=HC_gene_list)
```

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

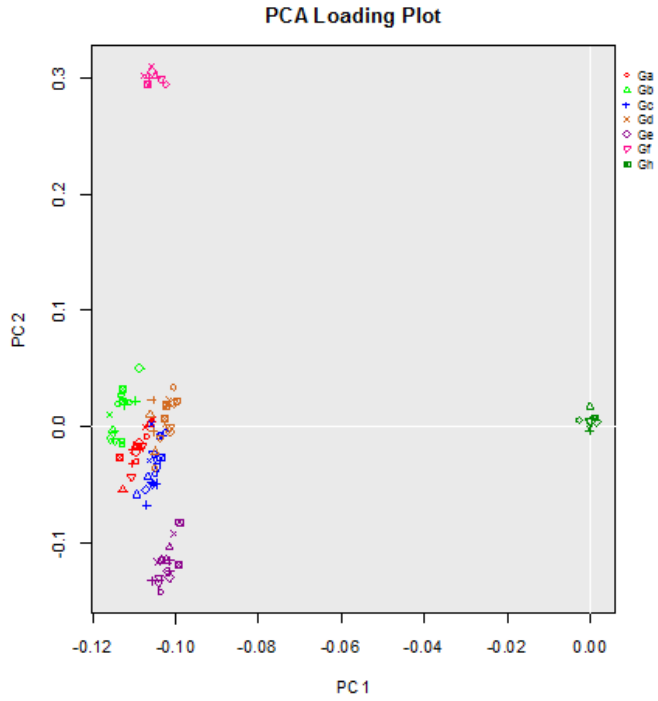
The Singular Analysis Toolset performs a variety of PCA functions. For more information, see Appendix A: The List of Functions for Gene Expression.

To analyze with PCA using a previously generated expression object

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

This function displays three PCA plots automatically:

- **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:



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):

```
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

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

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

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

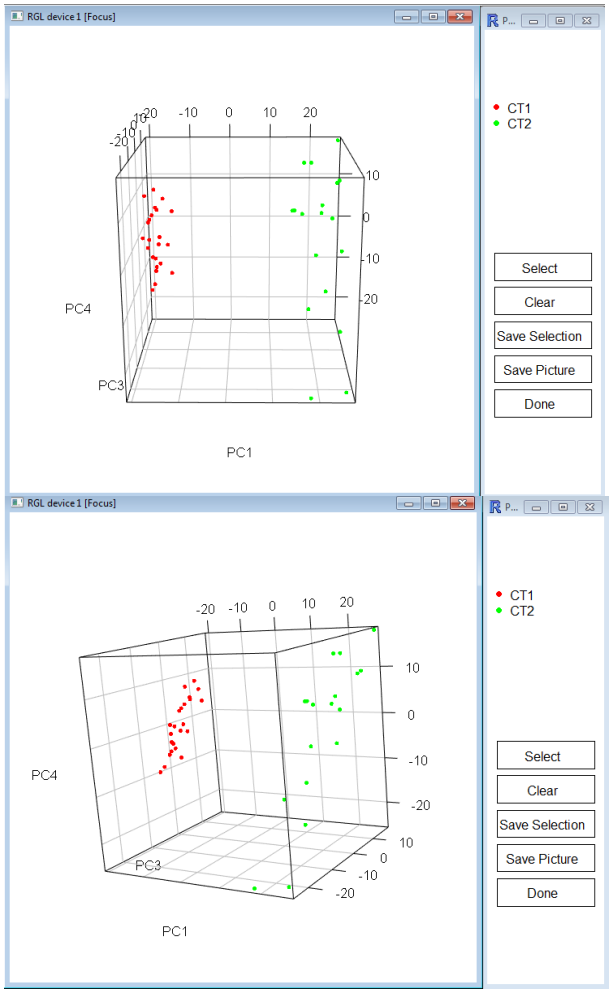
To display a 3D PCA Score Plot

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:

```
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.
- **Save Picture**. Save the current 3D plot to a *.png file.
- **Done**. Click this command to indicate that you are done locating samples of interest.

Examples of Advanced ANOVA Functions

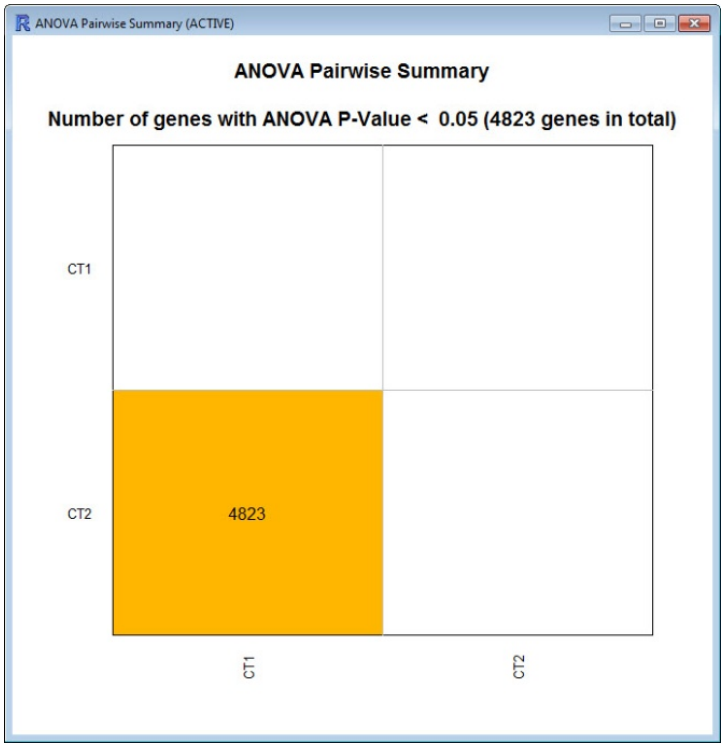
The Singular Analysis Toolset performs ANOVA based on either sample group annotation from user-defined annotation or on identified sample groups. Genes can be obtained according to P-value for all groups or any two groups (if you have more than two sample groups). For more information, see Appendix A: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

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

For example:



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:

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 = "SC_1", sample_group2 = "SC_2")
```

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

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

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

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

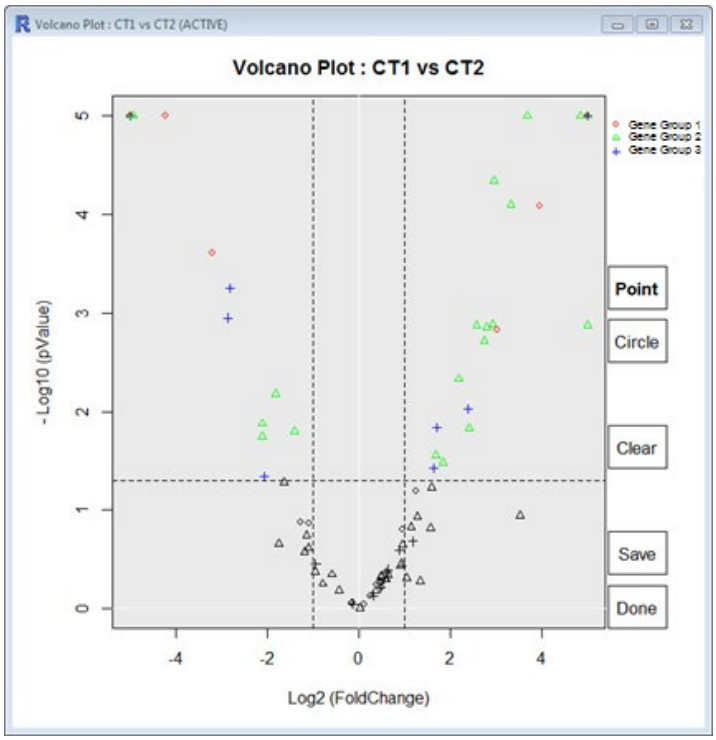
To identify differentially expressed genes via a threshold value in a Volcano plot

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:

```
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 a 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)
```

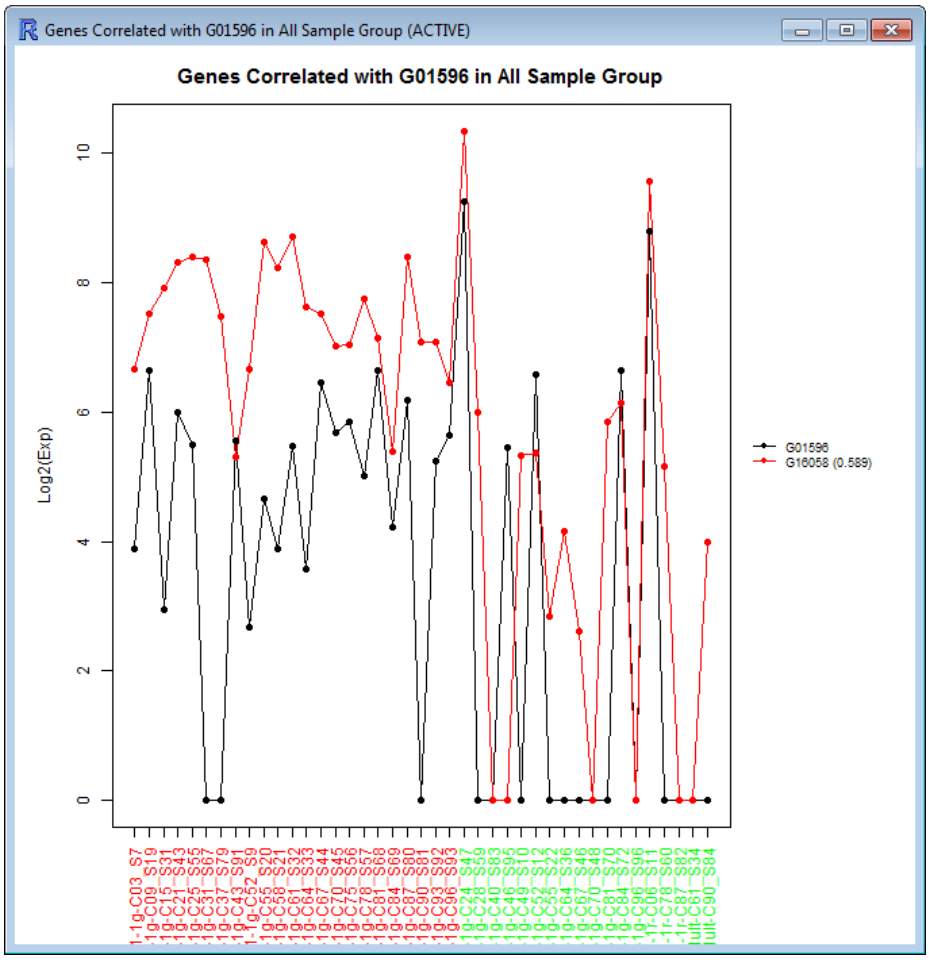
This corr object can be saved and opened according to the Saving Data section.

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="G01", corr_threshold = 0.5, corr_pattern =
"positive", "title")
```

In this example, there is one gene correlated with Gene G01596 in the two sample groups:



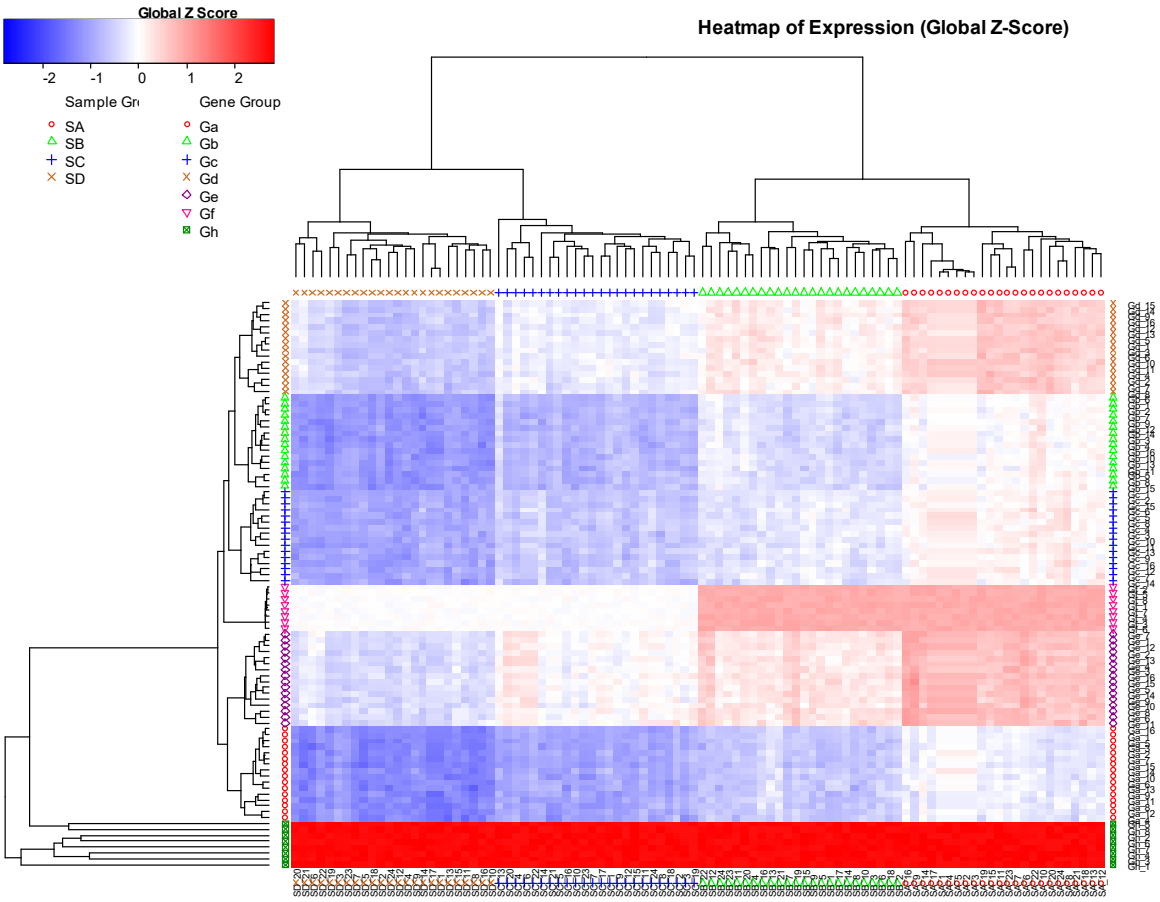
Examples of Advanced HC Functions

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:

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



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:

```
hc <- HC (exp, color_scheme= "green_gold_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, and 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

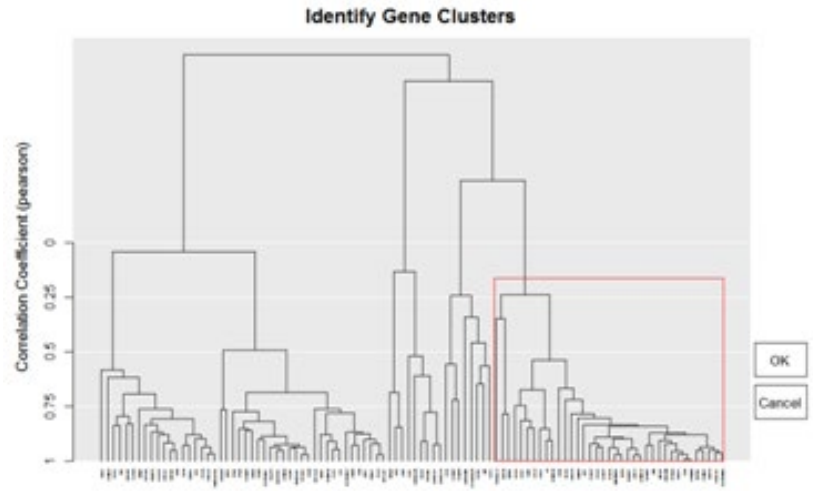
The functions **identifySampleClusters** and **identifyGeneClusters** enable you to select clusters by clicking on the dendrogram and selecting the region of your choice. The data for each selected cluster is returned as either a sample or gene list with the cluster IDs.

For a gene list:

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

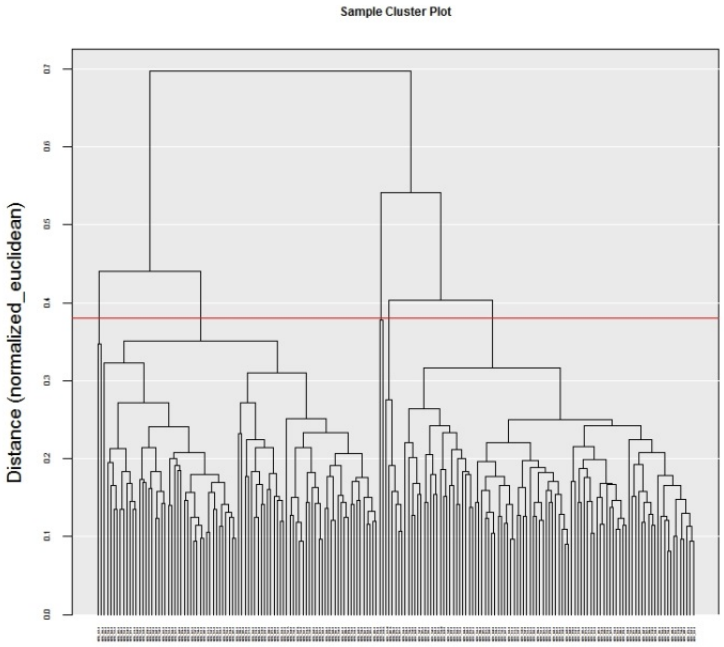
For a sample list:

```
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
    SampleID GroupID   ClusterDistance SampleAnnotationGroupID
1   CT1_04-1  SC_3_2 0.351357930523433                     CT1
2   CT2_32-1  SC_3_2 0.351357930523433                     CT2
3   CT2_84-1  SC_3_2 0.351357930523433                     CT2
4   CT2_55-1  SC_3_2 0.351357930523433                     CT2
5   CT2_05-1  SC_3_2 0.351357930523433                     CT2
6   CT2_02-1  SC_3_2 0.351357930523433                     CT2
7   CT2_31-1  SC_3_2 0.351357930523433                     CT2
8   CT2_10-1  SC_3_2 0.351357930523433                     CT2
9   CT2_71-1  SC_3_2 0.351357930523433                     CT2
10  CT2_14-1  SC_3_2 0.351357930523433                     CT2
11  CT2_20-1  SC_3_2 0.351357930523433                     CT2
```

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

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

To remove the outliers from the experiment

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

This newly updated expression object will no longer contain the outliers that have been identified.

Chapter 4: The Singular Analysis Toolset for Variant & Mutation Analysis

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.

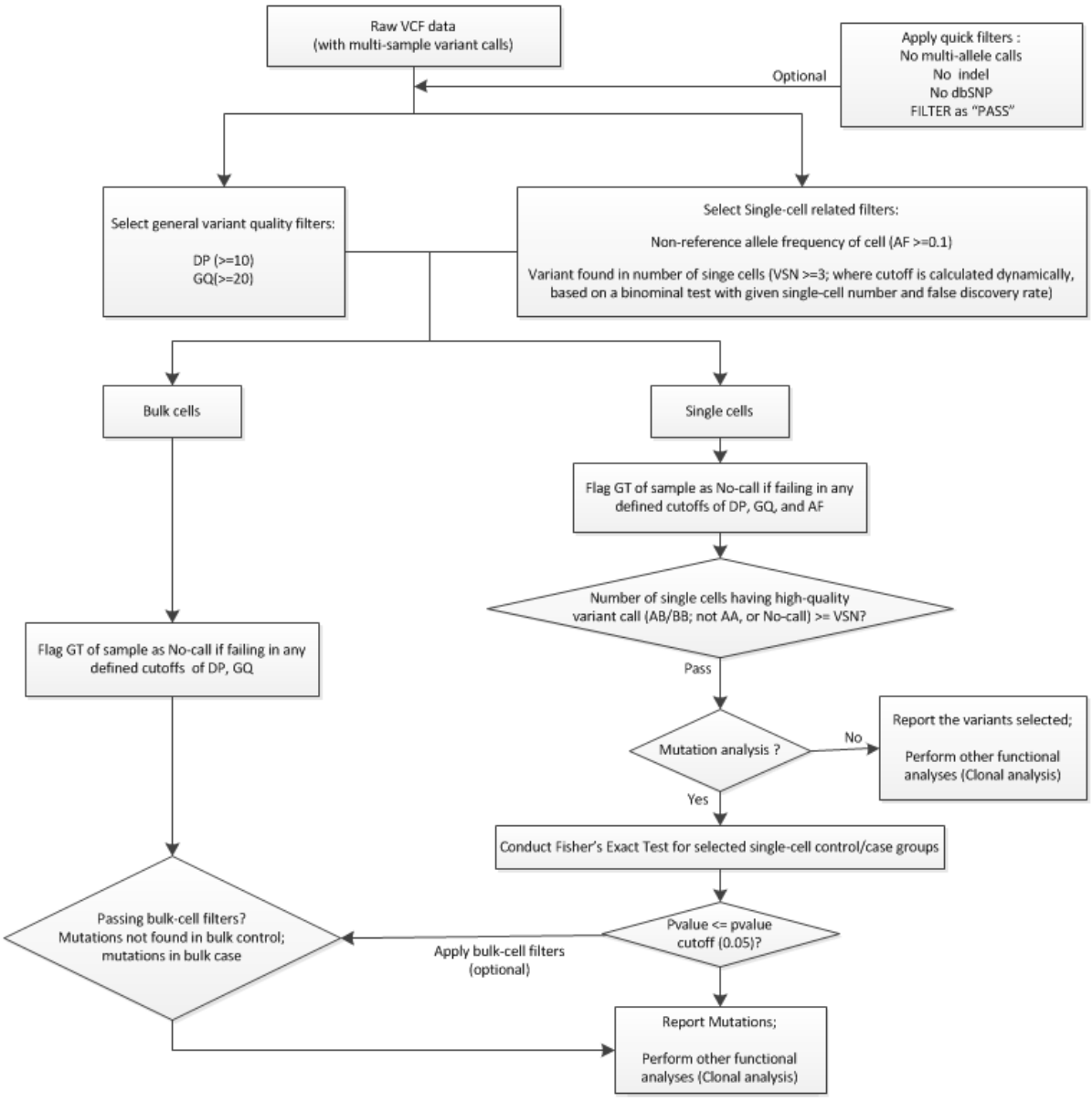
The types of data that can be analyzed for variants in the Singular Analysis Toolset are singleton SNPs, insertions, and deletions. If a SNP contains more than one alternate allele or variant identified in the VC object, then these data points will not be analyzed in this current version of the Singular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

The Singular Analysis Toolset accepts VCF files as input for single cell variant and mutation analysis. The overall mutation analysis workflow is depicted below.

Mutation Analysis Strategy



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:

	A	B	C
1	**SampleID**	**GroupID**	
2	CT1-gDNA_S01	CT1-gDNA	
3	CT1-gDNA_S02	CT1-gDNA	
4	CT1-gDNA_S03	CT1-gDNA	
5	CT1-SC_1_S01	CT1-SC	
6	CT1-SC_1_S02	CT1-SC	
7	CT1-SC_1_S03	CT1-SC	
8	CT1-SC_1_S04	CT1-SC	
9	CT1-SC_1_S05	CT1-SC	
10	CT1-SC_1_S06	CT1-SC	
11	CT1-SC_1_S07	CT1-SC	
12	CT1-SC_1_S08	CT1-SC	
13	CT1-SC_1_S09	CT1-SC	
14	CT1-SC_1_S10	CT1-SC	
15	CT1-SC_1_S11	CT1-SC	

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:

	A	B	C
1	**VariantID**	**GroupID**	
2	rs2072454	VC_1	
3	rs4947986	VC_1	
4	7:55240708_T_TG	VC_1	
5	7:55241604_T_TC	VC_1	
6	rs2293347	VC_1	

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.

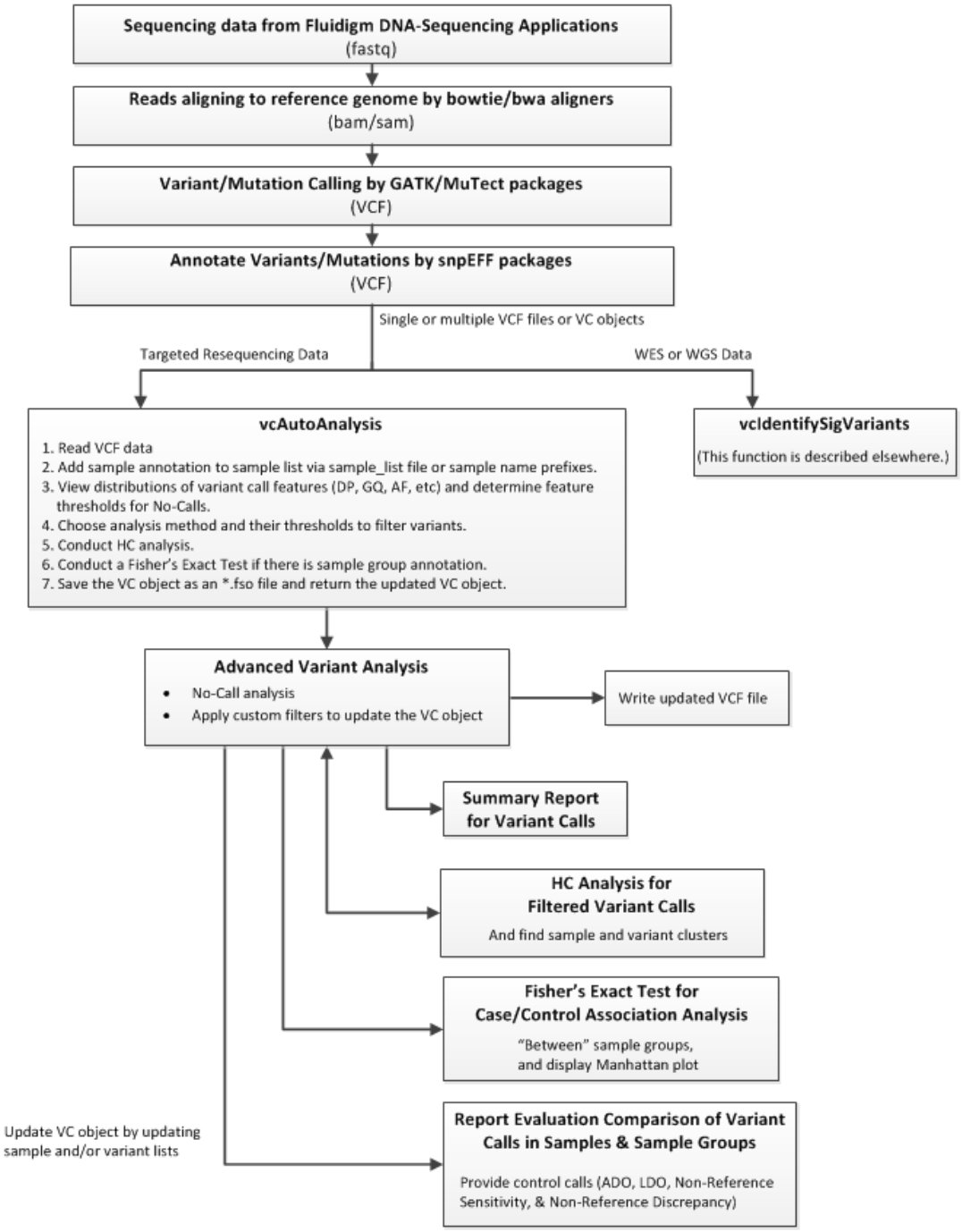
Preparing the Input Files

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



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

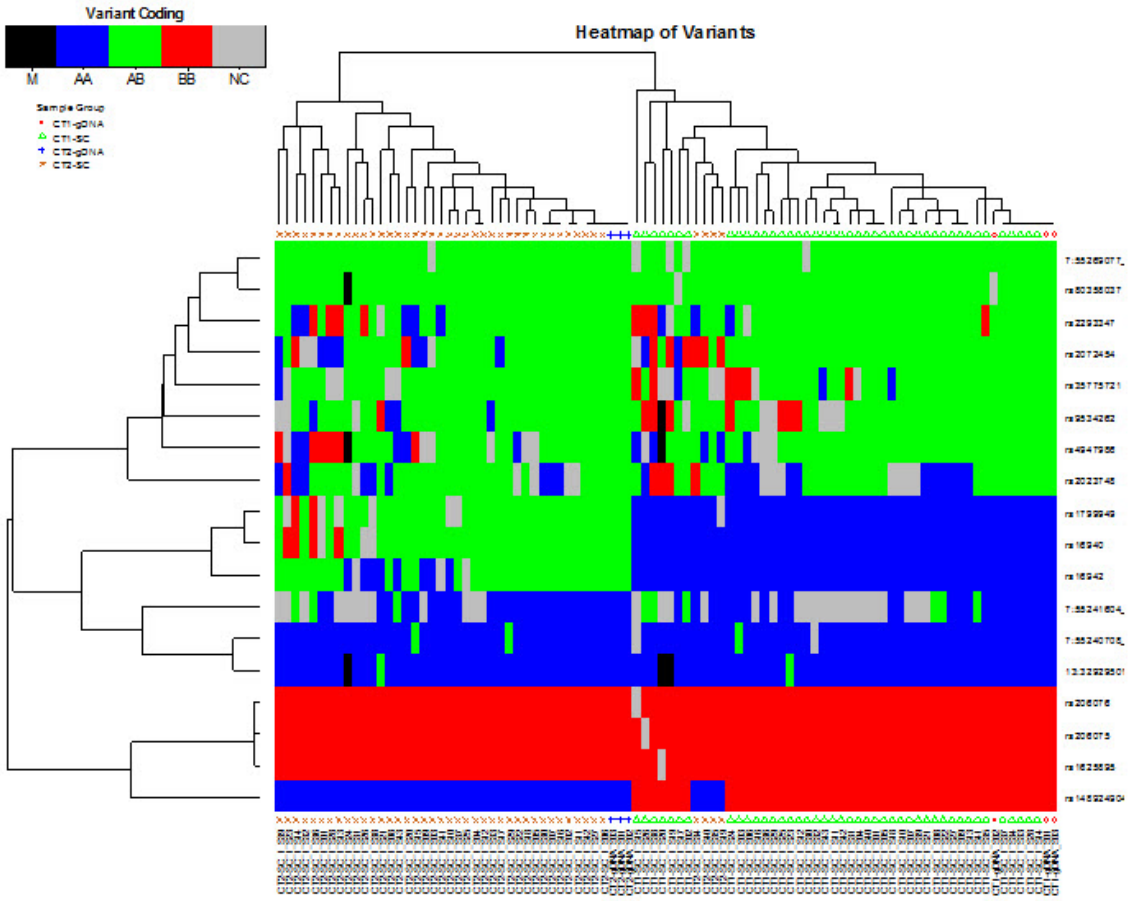
to build a binary tree or dendrogram of data that successively merges similar groups of data points.

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:

	Variants: A/B, B/B	**Non-Variants: A/A**	**Row Total**
Sample Group 1	a	b	a + b
Sample Group 2	c	d	c + d
Column Total	a + c	b + d	a + b + c + d (=n)

Table 1. 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)

$$p = \frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)!\ (c+d)!\ (a+c)!\ (b+d)!}{a!\ \ b!\ \ c!\ \ d!\ \ n!}$$

Modified from: http://en.wikipedia.org/wiki/Fisher's_exact_test

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.

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

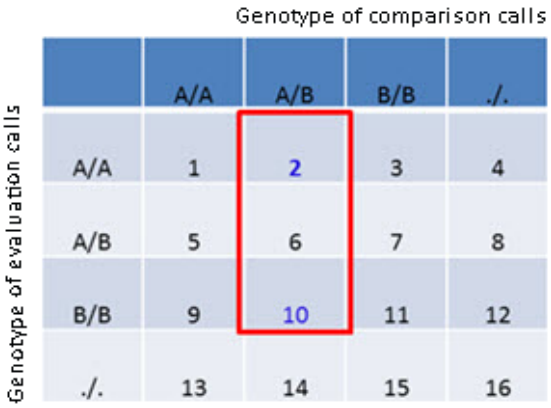
1. **aa_aa**. The number of samples of a variant with GT call as AA; and its GT in common call list is AA.
2. **aa_ab**. The number of samples of a variant with GT call as AB; and its GT in common call list is AA.
3. **aa_bb**. The number of samples of a variant with GT call as BB; and its GT in common 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)

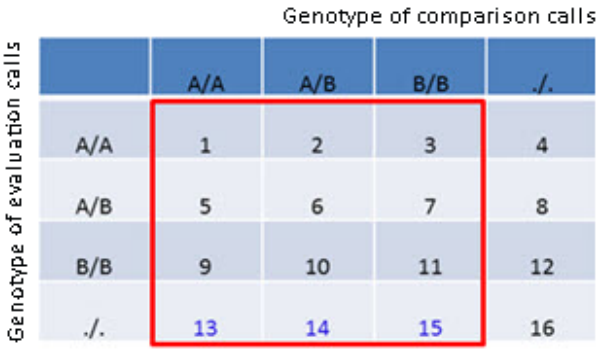
This table shows the loss of one allele from a heterozygous site during amplification.



Calculation = (2+10) / (2+6+10) = 12/18

Locus Dropout (LDO)

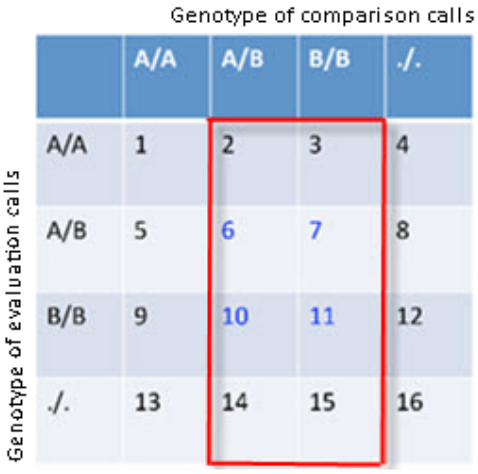
This table shows the loss of both alleles during amplification.



Calculation = (13+14+15) / (1+2+3+5+6+7+9+10+11+13+14+15) = 42/96

Non-Reference Sensitivity

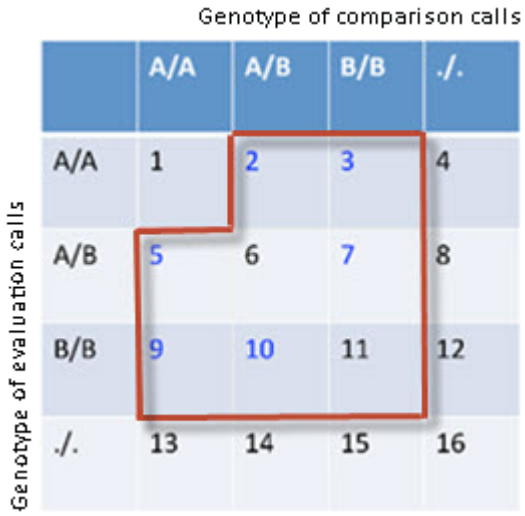
This table shows the fraction of sites called variant that are also variant in the evaluation set.



Calculation = (6+7+10+11) / (2+3+6+7+10+11+14+15) = 34/68

Non-Reference Discrepancy Rate

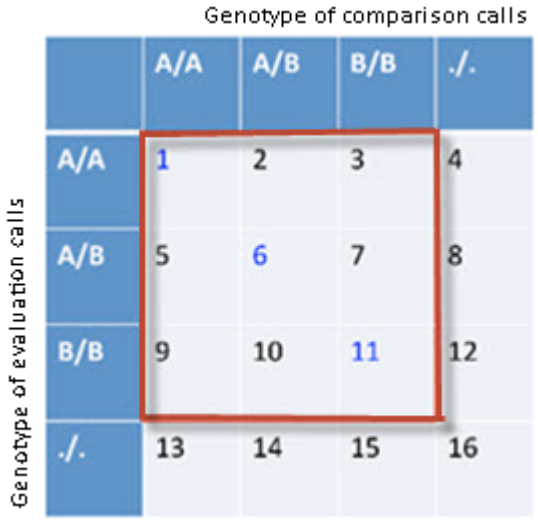
This table shows the accuracy of genotype calls at sites called in both the comparison and evaluation set, excluding concordant homozygous reference calls.



Calculation = (2+3+5+7+9+10) / (2+3+5+6+7+9+10+11) = 36/53

Overall Genotype Concordance

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



Calculation = (1+6+11) / (1+2+3+5+6+7+9+10+11) = 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.

  NOTE The samples filtered by this analysis will be in the Outlier list.

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

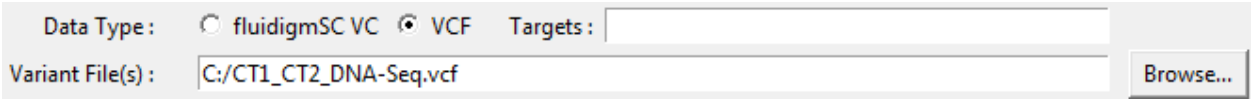
Type in **?fluidigmSC** at the R command line in the R console. At the bottom right, click the link **Index**.

For more information, see Get R Help.

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.)

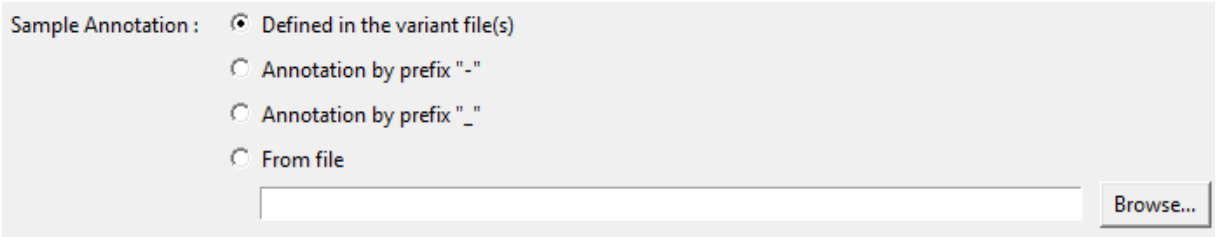


Your choices of data types are:

- **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 7:123-345;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 ";".

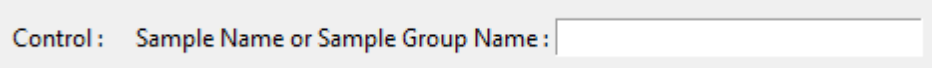
5. In the **Sample Annotation** section, select the appropriate sample annotation method.

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 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:

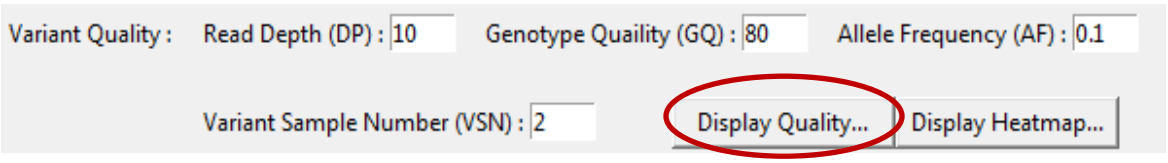


The name you enter will be used in variant filter analysis, which will only include calls that contain variant calls in the defined control.

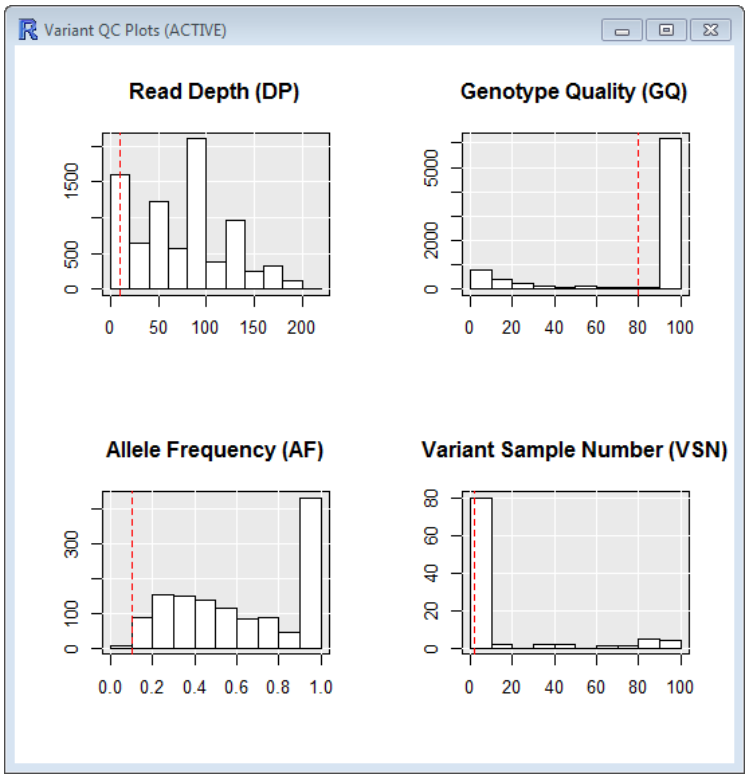
7 In the **Variant Quality** section:

- 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.

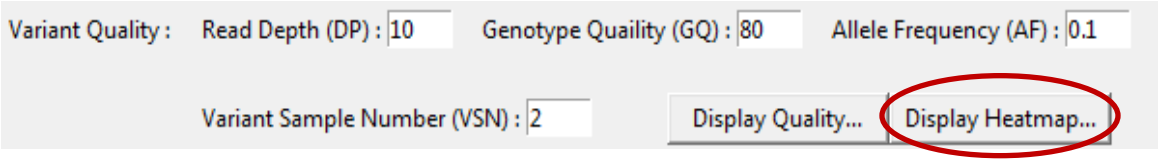
  When you click **Display Quality** in the **Variant Quality** section, you see histogram plots that contain a red line to highlight the defined threshold.



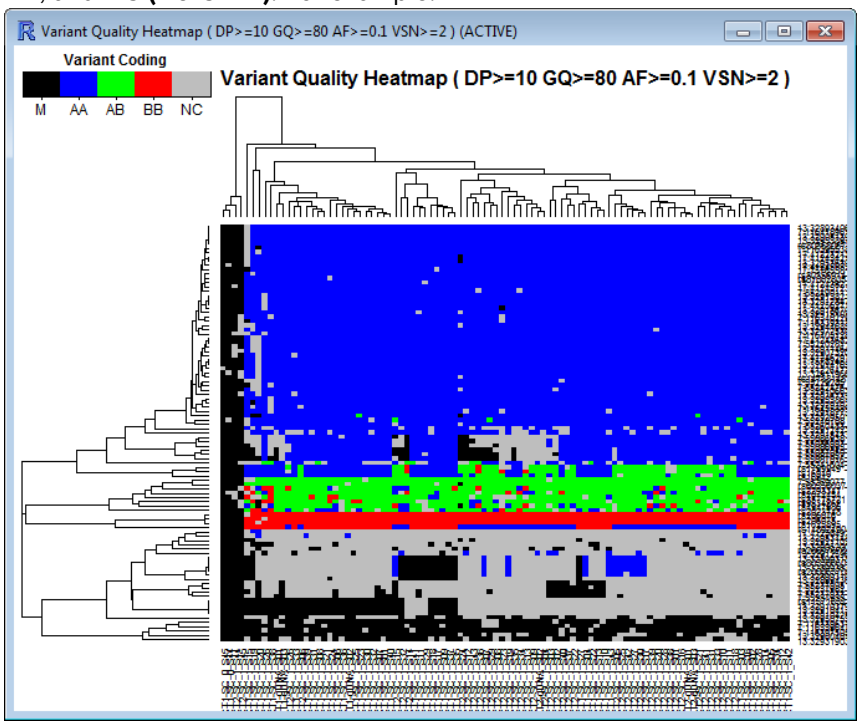
For example:



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

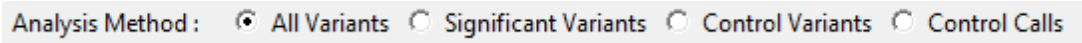


Each genotyping call of sample is assigned as one of four categories: **M**, **AA**, **AB**, **BB**, and **NC (No-CALL)**. For example:

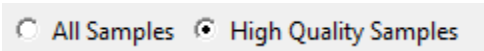


8 In the **Analysis Method** section:

a Select the method to filter variant calls:



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 ...
Dislaying 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

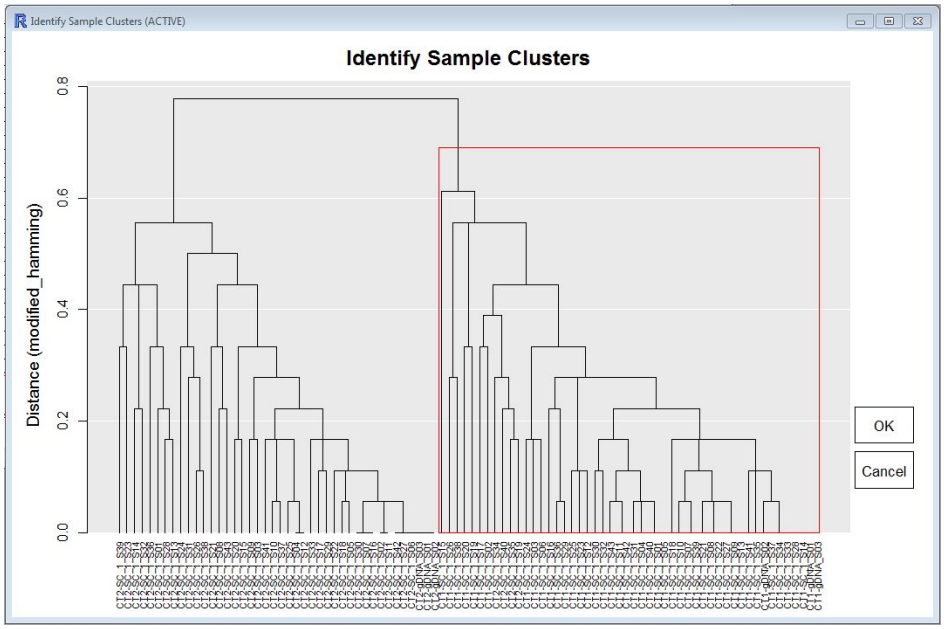
The **Heatmap of Variants** plot is displayed automatically upon running the function `vcAutoAnalysis`. For more information, see Hierarchical Clustering.

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.

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

For example:

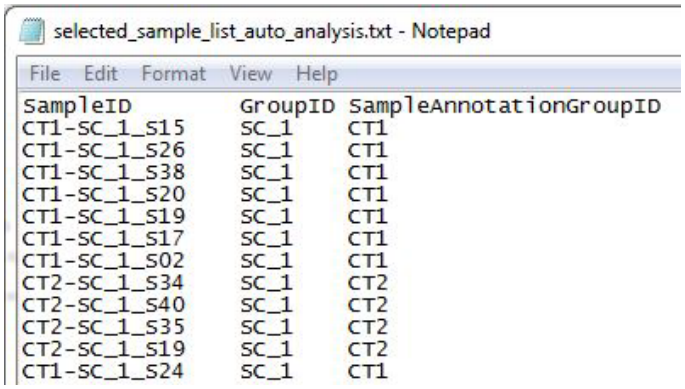


To accept your selected regions of interest, click **OK**. To return to the previous screen, click **Cancel**.

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

```
saveData(selected_sample_list_auto_analysis)
```

For example:



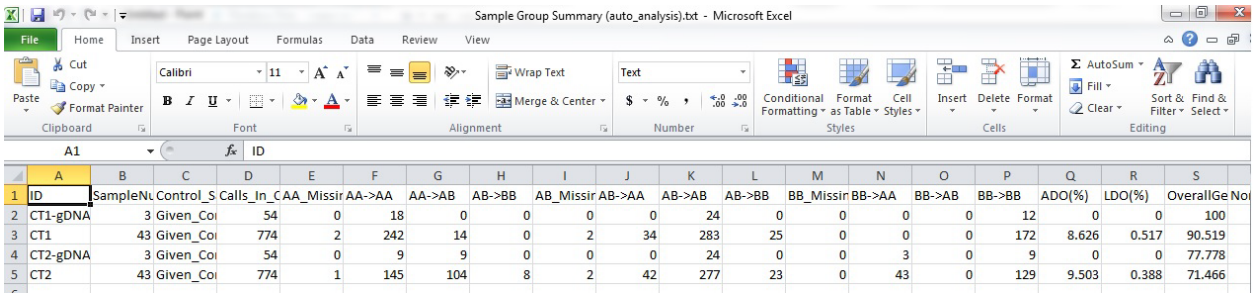
Summary Output Files

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:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	ID	SampleGr	Control_S	Calls_In_C	AA_Missir	AA->AA	AA->AB	AA->BB	AB_Missir	AB->AA	AB->AB	AB->BB	BB_Missir	BB->AA	BB->AB	BB->BB	ADO(%)	LDO(%)	OverallGe
2	CT1-gDNA	CT1-gDNA	Given_Co	18	0	6	0	0	0	0	8	0	0	0	0	4	0	0	100
3	CT1-gDNA	CT1-gDNA	Given_Co	18	0	6	0	0	0	0	8	0	0	0	0	4	0	0	100
4	CT1-gDNA	CT1-gDNA	Given_Co	18	0	6	0	0	0	0	8	0	0	0	0	4	0	0	100
5	CT1-SC_1_	CT1	Given_Co	18	0	5	1	0	0	0	8	0	0	0	0	4	0	0	94.444
6	CT1-SC_1_	CT1	Given_Co	18	0	5	1	0	0	1	6	1	0	0	0	4	12.5	0	83.333
7	CT1-SC_1_	CT1	Given_Co	18	0	5	1	0	0	1	6	1	0	0	0	4	12.5	0	83.333
8	CT1-SC_1_	CT1	Given_Co	18	0	5	1	0	0	0	8	0	0	0	0	4	0	0	94.444
9	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	0	8	0	0	0	0	4	0	0	100
10	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	2	5	1	0	0	0	4	18.75	0	83.333
11	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	1	7	0	0	0	0	4	6.25	0	94.444
12	CT1-SC_1_	CT1	Given_Co	18	0	5	1	0	0	1	7	0	0	0	0	4	6.25	0	88.889
13	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	1	7	0	0	0	0	4	6.25	0	94.444
14	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	1	7	0	0	0	0	4	6.25	0	94.444
15	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	0	8	0	0	0	0	4	0	0	100
16	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	1	6	1	0	0	0	4	12.5	0	88.889
17	CT1-SC_1_	CT1	Given_Co	18	0	6	0	0	0	1	7	0	0	0	0	4	6.25	0	94.444

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:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	ID	SampleNu	Control_S	Calls_In_C	AA_Missir	AA->AA	AA->AB	AB->BB	AB_Missir	AB->AA	AB->AB	AB->BB	BB_Missir	BB->AA	BB->AB	BB->BB	ADO(%)	LDO(%)	OverallGe
2	CT1-gDNA	3	Given_Co	54	0	18	0	0	0	0	24	0	0	0	0	12	0	0	100
3	CT1	43	Given_Co	774	2	242	14	0	2	34	283	25	0	0	0	172	8.626	0.517	90.519
4	CT2-gDNA	3	Given_Co	54	0	9	9	0	0	0	24	0	0	3	0	9	0	0	77.778
5	CT2	43	Given_Co	774	1	145	104	8	2	42	277	23	0	43	0	129	9.503	0.388	71.466

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 Sequencing or DNA Seq Analysis

The Singular Analysis Toolset includes a variety of advanced functions for Targeted Sequencing 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:

```
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

```
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:

	VARIANT_ID	GroupID
1	rs2072454	SNP
2	7:55219065_T_G	SNP
3	rs4947986	SNP
4	7:55223449_A_T	SNP
5	7:55223452_T_C	SNP
6	7:55224348_A_AG	INSERTION
7	7:55229224_T_TC	INSERTION

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

```
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:

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

To display an existing HC analysis

```
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

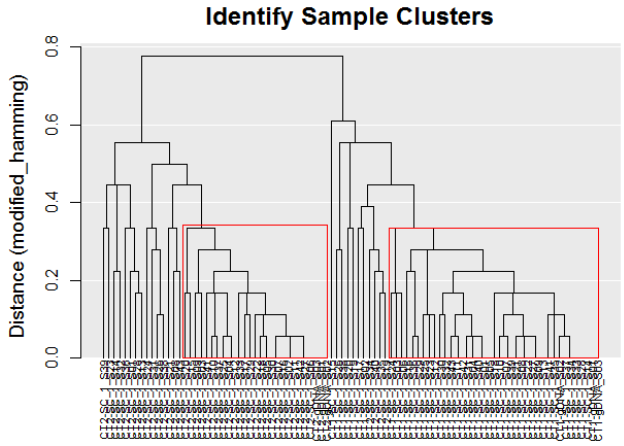
Use the following function after you perform HC analysis:

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

To identify variant clusters of interest

```
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

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

The new VC object can now be used for futher 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

```
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.

```
fisher_variant_list <-vcGetSignificantVariants(ft, pvalue_threshold = 0.05)
```

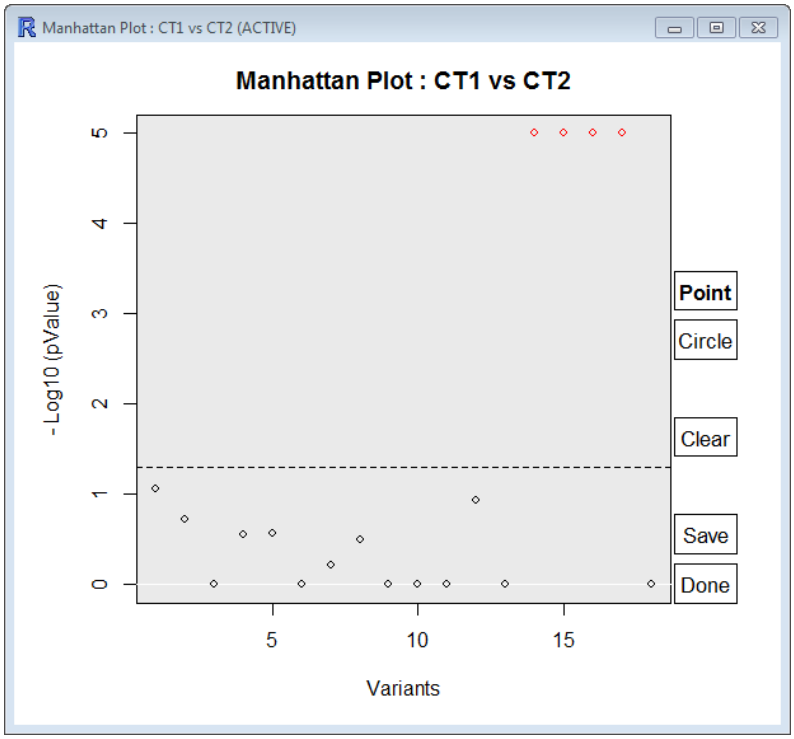
To get the significant variants and display a Manhattan plot

```
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:

```
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.

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 **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 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":

- **AND**. High quality variant calls need to pass all thresholds.
- **OR**. High quality variant calls need to pass at least one of the thresholds.

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.

```
vc <-vcUpdateVariantQuality(vc, DP_cutoff=20, GQ_cutoff=80, AF_cutoff=0.1,
Sample_Num_cutoff=2, DPGQAF_operator = "AND")
```

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.

```
vc <- vcUpdateVariantListByVariantQuality(vc)
```

To get a list of the high-quality variants

This function will return an ok_variant_list that can be saved according to the Saving Data section.

```
ok_variant_list <-vcGetFilterVariantListByVariantQuality(vc)
```

To use this list to update variant list by creating a new VC object without changing the original

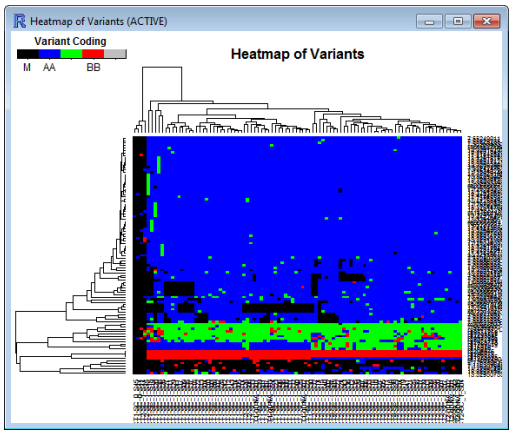
```
new_vc<-vcUpdateVariantListFromList(vc, ok_variant_list)
```

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

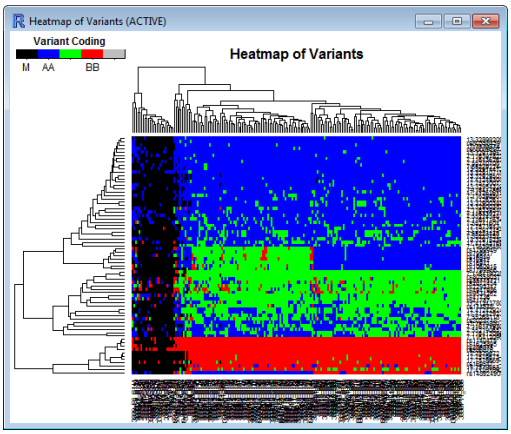
```
new_hc<-vcHC(new_vc)
```

For example:

Before



After



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** 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 **vcGetCommonVariantListWithCallFromSampleGroup** from Get Variant Calls with Genotyping Information from a Sample.

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 sectio sample_list <-vcIdentifySampleClusters(), 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 Sequencing or DNA Seq Analysis" in this document:

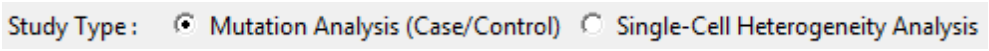
- Reading experimental data
- Annotating samples
- Annotating variants
- Setting custom colors and symbols (for sample and variant groups)
- Saving data
- HC analysis on filtered variant calls
- Identifying sample and variant clusters
- Conducting a Fisher's Exact Test (Case vs. Control) association 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

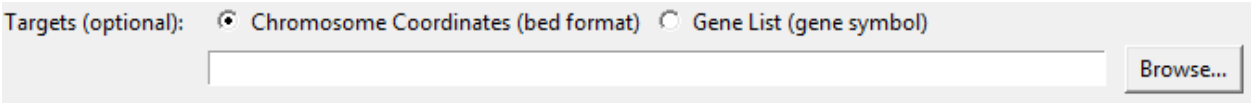
The function **vcIdentifySigVariants** displays a Graphical User Interface (GUI) that allows you to detect significant variants or mutations from raw variant data with a set of filters, such as variant quality, variant event in number of single cells, and/or by case and control mutation analysis. It 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. Do this only one time per session: Enter the function **library(fluidigmSC)** to load the package.
3. Enter the function **vcIdentifySigVariants()** to display the first of three dialogs.
4. Select one of the following study types:



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

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



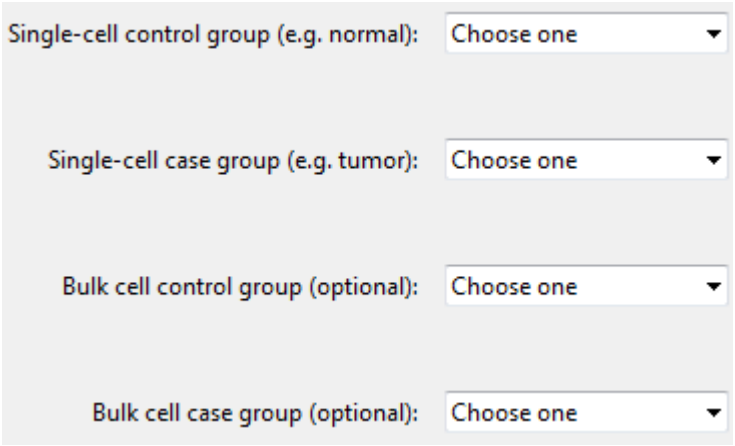
You can restrict the import of this data to target regions by providing targets of interest. Targets can be either (a) chromosome coordinates, such as 7:123-345;chr7:123-345 or chr7 ; or (b) gene symbol from a gene list, such as TP53. The targets can be a multiple of (a) and (b) or the combination of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



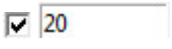
If your study type is **Single-Cell Heterogeneity Analysis**, select single and bulk cells from a list. You can use the **Shift** and **Control** keys to select multiple values.

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

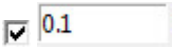
9. Click **Next**.

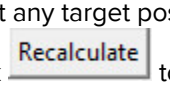
10. In the dialog that appears, you can use the default filters and their values or change them in any combination, where applicable. If a variant call of sample fails in all of thresholds, it is assigned as "No-CALL", with a flag value as "0" in the SCQC data frame of VC object. The filters are as follows:

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)**

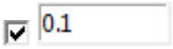
- **Variant Non-Reference Allele Frequency (AF)**  [AF is calculated by AD2 / (AD1 + AD2) at this position for this single cell: 0.05 – 1]
- **Variant Found in Single Cells (VSN)** [VSN cutoff is dynamically calculated based on a binominal test with given single-cell number and false discover rate (FDR) 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  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.

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)** ☑ 2 [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 0.05
- 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.)

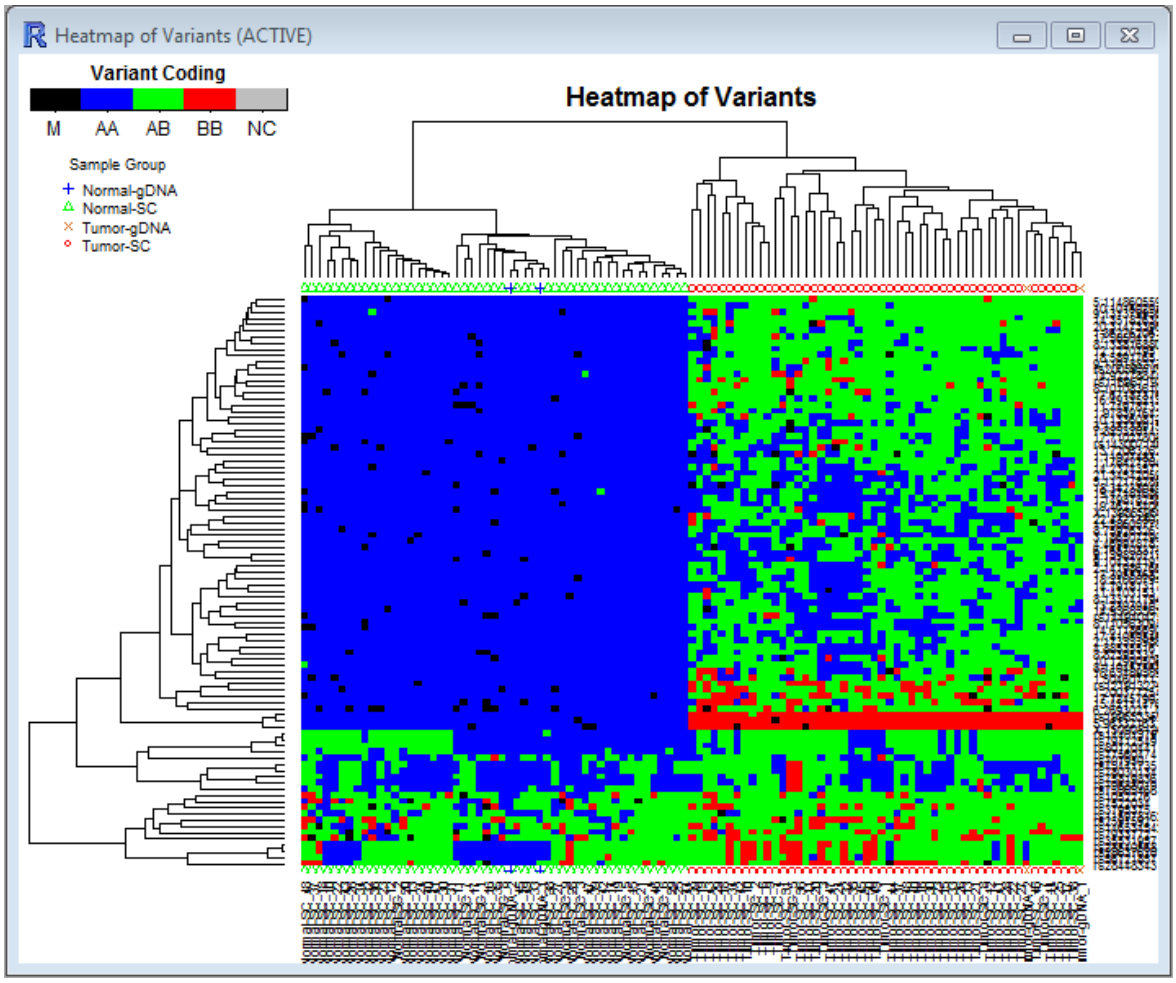
Example: Report for Mutation Analysis

```
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 pvalue cutoff of 0.05 removed = 43889
Final selected mutations = 1905

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


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

                                                    Category
1                                      Total samples in VCF data
2                                Total samples in provided sample list
3                                      Total variants in VCF data
4                                   Total dbSNP variants in VCF data
5                                Total non-dbSNP variants in VCF data
6                                                           <NA>
7                          Variants not passing VSN filter of 3 removed
8 Variants not passing Fisher's Exact Test with pvalue cutoff of 0.05 removed
9                                      Final selected mutations
   Value
1    103
2    103
3 129666
4  34128
5  95538
6   <NA>
7  83191
8  43889
9   1905
```

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



Example: Report for Single-Cell Heterogeneity

Analysis Summary:

	Category	Value	Percentage
1	DP cutoff	10	-
2	GQ cutoff	10	-
3	Variant Allele Frequency cutoff(AF)	0.1	-
4	Variant found in single cells cutoff(VSN)	3	-
5	Variants with FILTER flag as PASS	Yes	-
6	No Indel	Yes	-
7	Missense SNPs	Yes	-
8	non dbSNP	Yes	-
9	Single cell groups	tumor-SC; normal-SC	-
10	Bulk cell groups	normal-gDNA; tumor-gDNA	-
11	Total samples in VCF data	103	100%
12	Total samples in provided sample list	103	100%
13	Total variants in VCF data	129666	100%
14	dbSNP variants	34128	26.32%
15	Non-dbSNP variants	95538	73.68%
16	Multiple-allele variants removed	681	0.53%
17	Indel variants removed	4421	3.41%
18	Variants with FILTER as not "PASS" removed	78856	60.81%
19	Non-Missense variants removed	22313	17.21%
20	dbSNP variants removed	11661	8.99%
21	Variants not passing VSN filter removed	11085	8.55%
22	Final selected variants	649	0.5%

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

Launches the Fluidigm Single Cell Package Library for commands for the current session.	library(fluidigmSC)
This is the first method to be called after installing the **fluidigmSC** package. Only run this function one time per session. It downloads the required packages lattice, tcltk2, rgl, R.utils, and tsne.	firstRun()
Set your working directory.	setwd("pathway of your working directory") Example: setwd("C:/folder ")

Getting R Help for the fluidigmSC Library

Provides the complete list of fluidigmSC functions. Click any link to display the 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 For example: ? applyPCA

Lists the available types of gene expression analysis functions.	scExpFunctions() Now enter one of the 9 numbers to select the gene expression analysis function of your choice: 1. Get-Started Functions 2. Read and Save Functions 3. Expression Data Management Functions 4. Sample and Gene Display Functions 5. PCA Analysis Functions 6. Hierarchical Clustering Analysis Functions 7. ANOVA Analysis Functions 8. Co-Profiling Analysis Functions 9. tSNE Analysis Functions
Lists the available types of variant analysis functions.	scVarFunctions() Now enter one of the 7 numbers to select the variant analysis function of your choice: 1. Get-Started Functions 2. Read and Save Functions 3. Variant Data Management Functions 4. Variant Quality Functions 5. Hierarchical Clustering Analysis Functions 6. Fisher Test Functions 7. Variant Report Functions
Lists all the user-defined variables in the current session.	scVariables()

Identifying Outliers and Auto-Analyzing Data

Opens the interactive **FluidigmSC Outlier Identification** dialog where you enter the files and instructions for automatic outlier identification. By default (when an EXP object is not provided), this function will display a dialog box to select expression data type and an expression file. You can enter: • Expression file to be analyzed. • LoD of your choice. • Sample annotations (categorizing the samples into specific sample groups).	identifyOutliers(exp=NULL) exp: The EXP object. The default is set to NULL and opens the graphical user interface (GUI) for selecting expression file. When an EXP object is provided, the analysis will bypass the file selection and directly go to the outlier analysis. The function automatically identifies outliers and returns a Box plot of expression for each sample with outlier candidates labeled, where you can manually select and deselect outlier samples:

Outlier analysis is based on the notion that the same type of samples (for example, cells) have a set of commonly-expressed genes. Analysis for outliers is performed iteratively by trimming the low-expressing genes until at least 95 percent of remaining genes have their expression values above the limit of detection in at least half the samples (assuming the outlier is less than 50 percent of samples). These are the control samples that are used to identify outlier samples. If there are multiple sample groups in the expression object, the outlier analysis procedure is performed on each sample group.

When you click **Analyze**, the interactive **Outlier Identification** dialog opens that allows you to confirm and modify outlier candidates in your expression data.

The EXP object is saved as **exp(identify_outliers).fso** with outliers removed from further analysis.

- Adjust the outlier threshold using the "<" and ">" buttons or by clicking left to the y axis.
- Click a sample labeled as an outlier will restore the oultier to a normal sample.
- Click a normal sample will label it as an outlier.
- Click the PCA button will display the PCA score plot of all samples with outliers labeled. The sample labels will be displayed next to the data points in the PCA score plot when the total number of samples is less than 100.
- Click the Done button will remove identified outliers from the EXP object and allow you to save the EXP object.
- If there are more than one sample groups, the list of sample groups will be displayed on the top-right. You can click on each sample group to review and modify the outlier candidate.

NOTE The "Outlier Identification" dialog box is not linked to the R application, and it is important not to click anywhere outside this dialog box before completing the file selection. If you click anywhere outside this dialog box, it will be hidden behind the R application window. To find it, minimize the R application window or press Alt+Esc to toggle through the open windows.

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

On completion of autoAnalysis, a PCA Score plot, a Hierarchical Clustering (HC) heatmap, a Violin plot by genes, and a tSNE plot are displayed. 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.

In the output folder, the autoAnalysis function additionally returns

- A gene list [as the text file **selected_gene_list (auto_analysis).txt**]. The first column is the GeneID. The remaining columns are ANOVA p-values for all groups, pairwise p-value and average expression for each sample group, and PCA gene scores. ANOVA information is available only if sample annotation is provided.
- The objects EXP, PCA, HC, and tSNE objects the global variables fldm_exp, fldm_pca, fldm_hc, and fldm_tsne for further analysis.

  Each object file that is returned [for example **exp (auto_analysis).fso**] is merged with the genes of

autoAnalysis()

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 expression data files to be analyzed.
- Sample annotations (categorizing the samples into specific sample groups).
- A gene list with selected genes of interest, including gene group annotation or the number of significant genes to be found by the analysis.

NOTE The "FluidigmSC Analysis" dialog box is not linked to the R application, and it is important not to click anywhere outside this dialog box before completing the file selection. If you click anywhere outside this dialog box, it will be hidden behind the R application window. To find it, minimize the R application window or press Alt+Esc to toggle through the open windows.

interest and updated sample annotations, if provided. **IMPORTANT** If more than 1000 genes are specified, a warning message will be displayed.	
Save the automatically generated outlier list from autoAnalysis.	saveData(exp $outlier_list)

Reading and Saving Expression Data

Initialize the EXP object from one or more Ct expression data files that were generated by the Biomark HD system by Fluidigm. Ct expression data should be input as table or heatmap .csv (comma-delimited) files from the Fluidigm Real-Time PCR Analysis software. Sample names must be unique. Multiple Ct expression data files are merged into a single EXP object if all data share an identical gene or sample set, but not both.	exp <- readCtExp (exp_file=TRUE, lod=24) • exp_file: The filename of the data to be opened. The default (TRUE) displays a file dialog that allows you to select one or more expression files. • lod: The defined limit of detection (LoD) for Ct expression data. 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.
Initialize the EXP object from one or more linear expression data files. This is the initialization function for reading linear expression data and creating an EXP object for any downstream analysis.	exp <- readLinearExp (exp_file=TRUE, lod=1) • exp_file: The filename of the data to be opened. The default (TRUE) displays a file dialog that allows you to select one or more expression files. • lod: The defined detection limit for mRNA-seq 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 linear value of lod is 1.
Remove rows (genes) and renormalizes the expression data (for example, TPM, RPKM or FPKM) when reading the tab delimited data files.	excludeExpDataRowByName(excluded_rownames, renormalization=TRUE) • excluded_rownames: A set of names to be excluded in the form of c("Name1", "Name2",...). • renormalization: The default (TRUE) performs renormalization after removing the given set of rows to be excluded. If FALSE, renormalization will not be performed. For example: excludeExpDataRowByName (c("Undetermined")) or excludeExpDataRowByName (c("Spike1", "Spike2"))

Clear the excluded row names set by the excludeExpDataRowByName function.	clearExcludedExpDataRowNames ()
Remove columns (samples) when reading the tab delimited data files.	excludeExpDataColumnByName(excluded_colnames) excluded_colnames: A set of names to be excluded in the form of c("Name1", "Name2",...). For example: excludeExpDataColumnByName (c("Undetermined")) or excludeExpDataColumnByName (c("Spike1", "Spike2"))
Clear the excluded column names set by the excludeExpDataColumnByName function.	clearExcludedExpDataColumnNames ()
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.	readData(file=TRUE) file: The name of the file to be opened. The default (TRUE) displays a file dialog that allows you to select one or more expression files.
Load one or more EXP object data files and create a merged EXP object.	readExpObject(exp_file=TRUE) exp_file: The filename of the data to be opened. The default (TRUE) displays a file dialog that allows you to select one or more expression files.
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.	saveData(data, file=TRUE) • 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 (TRUE) displays a file dialog that allows you to specify a file name and location to safe the defined data.

Managing Expression Data

Assign a Group ID

Extract the sample group from the sample name with a separator in the Exp object.	exp <- updateSampleListFromName (exp, sep="_", prefix=TRUE) • exp: A given EXP object. • sep: The character that the function will use to separate the sample group ID from the sample name. May also be "-"

	• prefix: The default (TRUE) indicates that the group ID is a prefix of the sample name. If FALSE, the group ID will be assumed to be the suffix.
Update the EXP Object	
Update the sample list of an EXP object using a sample list file as input and return the updated EXP object. Since this function provides a SampleID column in your sample list, you can then manipulate samples by sample group and display them by group, such as in HC and PCA plots.	exp <- updateSampleListFromFile (exp, sample_list_file=TRUE) • exp: The object to update. • sample_list_file: The default (TRUE) allows you 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 and in quotes. The sample list file is a tab-delimited text file with the first two columns containing sample names and group names with headers as "SampleID" and "GroupID", respectively.
Update the sample list of an EXP object using a sample list as input and return the updated EXP object.	exp <- updateSampleListFromList(exp, "sample_list") • exp: The object to update. • "sample_list": new sample list to replace the sample list in the EXP object.
Update the gene list of an EXP object using a gene list as input, and return the updated EXP object.	exp <- updateGeneListFromList (exp, "gene_list") • exp: The object to update. • "gene_list": A new gene list to replace the gene list in the given EXP object.
Update an expression object with a new gene list from a gene list file. This function provides a GroupID column in your gene list. You can then manipulate genes by gene group and display them by group, such as in HC and PCA plots.	exp <- updateGeneListFromFile (exp, gene_list_file=TRUE) • exp: The object to update. • gene_list_file: The default (TRUE) allows you to select a file through a dialog. This argument 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.
Remove or Retain Samples and Sample Groups	
Remove a sample from the expression's sample list and return the EXP object with the updated sample list.	exp <- removeSample(exp, "sample_name") • exp: A given Exp object.

	• "sample_name": The name of a sample to be removed from the list.
Remove a group of samples from the expression's sample list and return the updated EXP object with a modified sample list.	exp <- removeSampleGroup (exp, "sample_group") • exp: A given Exp object • "sample_group": The name of the sample group to be removed from the list.
Retain only the samples of a give group from the expression's sample list and return the updated EXP object with a modified sample list.	exp <- retainSampleGroup (exp, "sample_group") • exp: The object to update. • "sample_group": The name of the sample group to be retained in the sample list.
Remove or Retain Genes and Gene Groups	
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.	exp <- removeGene (exp, "gene_name") • exp: A given EXP object. • "gene_name": The name of a gene to be removed from the list.
Given an EXP object from linear expression data and a linear expression detection threshold, remove all genes in the expression data that are lower than the linear threshold. Returns the updated exp with updated gene list.	exp <- removeGenesByLinearExp (exp, linear_threshold) • exp: A given EXP object. • linear_threshold: The threshold used to compare and remove genes. Can be set to equal any number, such as linear_threshold=2.
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. Returns an updated exp with updated gene list.	exp <- removeGenesByCtExp (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.	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.	exp <- retainGeneGroup (exp, "gene_group") • exp: A given EXP Object. • "gene_group": The name of the gene group to be retained in the list.
Create Gene Lists	
Create a gene list from a set of gene names. Returns gene names that you can create after reading a previously saved EXP object. You can then display plots with gene names.	createGeneListFromGeneNames(gene_names = c()) The inner parentheses contain a set of gene names in quotes. Example: createGeneListFromGeneNames(gene_names = c("Gene1", "Gene2",...))
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. 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.	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 different colors. To return to the previously assigned colors, click **Cancel**. To return to the R console, close the window.
If you customize the color and symbol schemes for the sample groups, you can restore the default color and symbol settings for sample groups. 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.	exp<-clearSampleGroupColorAndSymbols(exp) exp: A given EXP object with sample group annotation.
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. 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.	exp<- setGeneGroupColorAndSymbols(exp) exp: A given EXP object with sample group annotation.

If you customize the color and symbol schemes for the gene groups, you can restore the default color and symbol settings for gene groups. 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.	exp<-clearGeneGroupColorAndSymbols(exp) exp: A given EXP object with gene group annotation.
Reset Sample and Gene Lists	
Replace an EXP object's current list of genes with its original list of genes and return updated EXP.	exp <- resetGeneList (exp) exp: A given EXP object.
Replace an EXP object's current list of samples with its original list of samples. and return updated EXP.	exp <- resetSampleList(exp) exp: A given EXP object.
Manage Outliers	
Add a particular outlier back for analysis, and return an updated EXP.	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 definied by a sample list back for analysis. and return updated EXP.	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.	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.	exp <- addOutlierFromName (exp, "sample_name") • exp: A given EXP object. • "sample_name": Name of the outlier sample to be added. Replace with the name of the sample set in quotes.
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.	exp <- addOutlierFromList (exp, "sample_list") • exp: A given EXP object.

	• "sample_list": The list of outliers to be added. Replace with the list name in quotes.
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. The default (TRUE) for the outlier_list_file dispays a file dialog box to select a sample list file. Alternatively, this can be set to the file path in quotes. This dialog box is not linked to the R application, and it is important not to click anywhere outside this dialog box before completing your file selection. If you click outside this dialog box, it will be hidden behind the R application window. To find it, minimize the R application window or press Alt+Esc to toggle through the open windows.	exp <- addOutlierFromFile (exp, outlier_list_file=TRUE) exp: A given EXP object. Outlier_list_file: An outlier list file
Update the LoD	
Return a new EXP object after updating the given EXP object's limit of detection (LoD) and recalculating the Log2Ex expression data. For mRNA seq data, the LoD is defined in the linear domain. For qPCR data, the LoD is defined in the Ct domain.	exp <- updateLod(exp, lod) • exp: The object to update. • lod: The new LoD value to be associated with this EXP object. This value becomes the desired LoD number.
Merge Data	
Merge two expression objects that share either the same sample list or the same gene list (but not both). Returns a single expression object containing expanded list of either samples or genes.	exp <- mergeExpData(exp1, exp2, overlap_flag = -1) • exp1: Must be an existing exp object within the R session. • exp2: Must be a second existing exp object within the R session. • overlap_flag: When set to -1 (default), first determines whether the two EXP objects share a common set of genes or a common set of samples, and then merges them accordingly. **overlap_flag = "gene_overlap"** merges the two EXP objects with a common set of genes. **overlap_flag = "sample_overlap"** merges the two EXP objects with a common set of samples.
Merge two gene lists and return a merged gene list.	merged_gene_list <- mergeGeneList (gene_list1, gene_list2, method="union") You can replace gene_list1 and gene_list2 with sample list names separated by a comma 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.
Merge two sample lists and return a merged sample list .	merged_sample_list <- mergeSampleList (sample_list1, sample_list2, method="intersect") You can replace gene_list1 and gene_list2 with gene list names separated by a comma that can be merged as "union", "intersect", or "subtract". For the "union" method, the merged gene list includes all genes from both source lists. For the "intersect" method, the merged gene list includes only genes that are common between both source lists. For the "subtract" method, the merged gene list includes only genes that are in the 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. Displays the information or when set to create a list, this can be saved as text delimited file.	summaryExp(exp) summary_table <-summaryExp(exp) exp: The object to display.
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. The Box plot will order samples by sample groups. Sample groups are in alphabetical order and samples within groups are in their original order. Sample names are along the x axis, colored by default or customized group color. Log2Ex values are on the y axis.	boxPlotBySamples (exp, by_sample=TRUE, sample_group="all", max_sample_num=1000, "title") • by_sample: The default (TRUE) displays gene profiles for each sample. FALSE displays the gene profile for the sample group. • sample_group: 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. • 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": The title of the plot in quotes. If missing, the function will provide a default "Boxplot of Gene Expression by Samples (Genes = N)" where N is the number of genes.
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.	violinPlotBySamples (exp, by_sample=TRUE, sample_group="all", max_sample_num=1000, "title") • by_sample: The default (TRUE) displays gene profiles for each sample. FALSE displays the gene profile for the sample group. • sample_group: 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. • 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.
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 Log_2Ex 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.	pairwiseScatterPlotBetweenSampleGroups (exp, "title") exp: A given EXP object. "title": Title of the plot. If missing, the function will provide a default "Pairwise Comparison of Sample Groups".
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.	analyzeGeneDetection(exp, threshold=1, dropout_flag=FALSE, display=TRUE, "title") exp: The object containing the detected genes. • threshold: The defined threshold for gene detection. This represents a linear value for mRNAseq data and a Ct value for qPCR data. • dropout_flag: The default (FALSE) calculates gene detection. When set to TRUE, it calculates gene dropout by sample. • display: The logical value to determine the result to be displayed. The default (TRUE) is set to display the result of gene detection or dropout. When changed to FALSE, no display is created.

	• "title": The title of the plot in quotes. If the quotes are missing, the function will provide a default of Gene Detection in Sample (Total Genes=N; Gene Detection Threshold>=#), where N is the number of genes in the exp input and # is the defined threshold.
Given an EXP object and a set of genes, creates 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. Log_2Ex 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. The box plots will remain in the same order as genes in the gene list used.	boxPlotByGenes (exp, gene_list_file=FALSE, gene_list=FALSE, num_gene_per_plot=100, max_gene_num=1000, "title") • exp: The EXP object containing the gene list to be displayed (by default). • gene_list_file: The default (FALSE) does not give you access to select a gene list from a file. When set to TRUE, you see a window that allows you to choose a gene list from a file. • gene_list: The default (FALSE) does not allow you to select a different gene list. When set to TRUE, you can select a different 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 to be plotted. The default is 1000 genes. If the maximum is reached, you are prompted to type in a value of 1 to stop the plot-creation process where it is interrupted or a value of 2 to continue graphing until all samples are plotted. • "title": The title of the plot in quotes. If missing, the function will provide a default: "Boxplot of Gene Expression "
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. Log_2Ex 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.	violinPlotByGenes (exp, gene_list_file = FALSE, gene_list = FALSE, num_gene_per_plot = 100, max_gene_num = 1000, "title") exp: The object containing the gene list to be displayed. 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 "Violin Plot of Gene Expression".

Summary of PCA Functions

Perform PCA on a given EXP object, calculate the PCA gene score and displays results. Displays three 2-dimensional plots: PCA Scree: Variance on y axis and first ten pc scores on x axis. visually identify how man 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 a gene. PC1 scores on x and PC2 scores on y.	pca <- PCA (exp, rank_component_num=3, display_plots = TRUE) • exp: The EXP object. • rank_component_num: The number of principal components used to calculate the PCA scores of genes. (The default value is 3.) • display_plots: 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.
Applies an existing PCA model for an EXP object to the expression data of new samples in a new EXP object and 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 man 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. The PCA model must have been calculated on the existing EXP object using: pca <-PCA(exp) Then a second or new exp can be called up using: exp <- readExpObject() To apply a model PCA to a set of new samples, the new data set must contain all genes in the PCA model. When	pca <- applyPCA (exp, pca, display_plots=TRUE, include_outliers=FALSE) • exp: The new EXP object to which the PCA model will the applied. • pca: The pca model calculated from the original EXP object. • display_plots: The default (TRUE) displays all PCA plots. If FALSE, only the new PCA object will be returned. • include_outliers: The logical value to display the results with outliers from the second EXP object. The default (FALSE) excludes outliers from the second EXP object.

the model is applied to the new expression object, only genes present in the PCA model will be applied.	
Cluster a PCA Score with k-mean clustering methods using an optional graphical user interface (GUI) for changing number of clusters: **K + 1**. Increase the number of clusters by 1. **K – 1**. Decrease the number of clusters by 1. **Retry**. Recluster the data using different initial cluster centers. **Save**. Save the selected cluster results (sample_list: sample id + cluster id). **Done**. End your clustering session when you are finished clustering.	clusterPCAScore (pca=fldm_pca, x_axis=1, y_axis=2, num_clusters=3, num_iterations=20, change_num=TRUE, title="") • pca: The PCA object to be clustered. The default is set to fldm_pca, which is the output from the autoAnalysis() function. • x_axis: The default is set to PCA component 1. • y_axis: The default is set to PCA component 2. • num_clusters: Number of clusters, and the default is set to 3. • num_iterations: Number of iterations to be performed, and the default is set to 20. • change_num: The default (TRUE) is set to interactively change the number of clustering according to the data and to save cluster results. When set to FALSE, it only displays the cluster plot and return the cluster results. • title: Title of the plot in quotes. If missing, the function will provide a default.
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 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**. 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**.	sample_list <- displayPCAScore (pca=fldm_pca, x_axis = 1, y_axis = 2, sample_list=NULL, 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; on the x axis. • y_axis: The default is PC score 2; on the y axis. • sample_list: Displays the PCA score according to the new sample list. The displayed PCA score will contain the common samples from both the PCA and the provided sample list. • locate: The default (TRUE) allows you to select and save individual samples of interest on the plot. If FALSE, it only displays the Score plot. • "title": The title of the plot. If missing, the function will provide a default "PCA Score Plot”.

Display a 3D PCA Score plot with an optional graphical user interface for locating samples of interest. The plot is interactive. To move the plot to different angles, press and hold the mouse in the plot as you move your cursor. You can also use any of the following commands: **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. **Save Picture**. Save the current 3D plot to a *.png file. **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**.	sample_list <- display3DPCAScore (pca, x_axis = 1, y_axis = 2, z_axis=3, locate=TRUE) • 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; on the x axis. • y_axis: The default is PC score two; on the y axis. • z_axis: The default is PC score three; on the z axis. • sample_list: Displays the PCA score according to the new sample list. The displayed PCA score will contain the common samples from both the PCA and the provided sample list. • locate: The default is TRUE and will display an interactive plot that allows you to click anywhere on the 3D plot and drag the cursor to turn the plot. You can also select and save individual samples of interest on the plot. If FALSE, it only displays the Score plot.
Displays a PCA Loading plot with an optional graphical user interface for locating genes of interest The plot is interactive: **Circle**. Draw a circle by connecting dots on the plot. Once a circle is formed, the plot displays each gene ID inside the circle. **Point**. Click a gene to display its ID. **Clear**. Returns the plot to its original state. **Save**. Save selected genes to a file. **Done**. Indicates that you are done locating genes 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%).	gene_list <- displayPCALoading (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: default is PC score 2. This is the PC score used for this axis. • locate: The default is TRUE and will display an interactive plot that allows user selection of individual samples on the plot. • "title": Title of the plot. If missing, the function will provide a default "PCA Loading Plot".

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**.	
From a given PCA object, return a defined number of top-ranked PCA genes.	pca_gene_list <- getTopPCAGenes (pca, top_gene_num=100) • pca : Previously created pca object containing gene list. • top_gene_num: The default is the top 100 genes, sorted from high to low gene values calculated from PC scores 1,2,&3, however this can be set to equal any desired number.
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,&3.	displayPCAGeneScores (pca, top_gene_num = -1,"title") • pca: Previously created pca object containing gene list. • 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. • "title": Title of the plot. If missing, the function will provide a default "PCA Gene Score Plot".
Display the Scree plot for the PCA object saved from the autoAnalysis. You can also display the Scree plot with an explicitly-defined PCA object.	displayPCAScree (pca=fldm_pca, "title") • pca=fldm_pca: The PCA analysis result by autoAnalysis(). This 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 tSNE Functions

Performs a tSNE (T-distributed stochastic neighbor embedding) technique on a given EXP object. This technique uses an unsupervised nonlinear dimensionality reduction algorithm to visualize high-dimensional gene expression data sets in a dimension-reduced data space, which can then be viewed in a scatter plot. Unlike PCA analysis, each tSNE run is an explorative analysis method that results in a different plot, even if the parameter setting is the same. To unveil the internal data structure, multiple tSNE analysis is required with or without changing the analysis parameters.	tSNE(exp, initial_dims = -1, perplexity = -1, iteration = 2000, display_plots = TRUE) • exp: The EXP object. • initial_dims: The initial dimensions (Default: -1). If -1, tSNE will use the number of genes defined the EXP object for the analysis. If initial_dims is greater than 0, the gene dimension will be reduced by PCA and then the perform the

A tSNE object is a list of the following data frames:

- obj_type: The “tSNE” object
- analysis_parameter: The initial_dims, perplexity, and iteration used for this analysis
- tsne_result: The location (X1, X2) of each sample
- sample_list: The sample names and group IDs used for this analysis in two columns
- gene_list: The gene names and group IDs used for this analysis in two columns
- sample_group_color_symbols: The color and symbol for each sample group inherited from the EXP object

analysis the the reduced dimension equal to initial_dims.

- perplexity: The optimal number of neighbors (Default: -1). Typical values should range from 5 to 50, where the default is to let the function set this value automatically according to the number of samples. In general, if the clusters are not well separated at the end of the analysis, increase this parameter if clusters should contain large number of data points or reduce it otherwise.
- iteration: The number of iterations (Default: 2000). Increase this number if the clusters are not converged at the end or reduce this number to accelerate the analysis if the clusters are converged quickly.
- display_plots: The default (TRUE) will display the new tSNE plot after analysis. If FALSE, no tSNE plot will be displayed.

Display a tSNE plot with an optional GUI for locating samples of interest.

displaytSNE(tsne, sample_list=NULL, locate = TRUE, title = "")

The plot is interactive:

Circle. Draw a circle by connecting dots on the plot. Once a circle is formed, the plot displays each gene ID inside the circle.

Point. Click a gene to display its ID.

Clear. Returns the plot to its original state.

Save. Save selected genes to a file.

Done. Indicates that you are done locating genes 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**.

- tsne: The tSNE object to be plotted. The default is set to fldm_tsne, which is the output from the autoAnalysis() function.
- sample_list: Displays the tSNE result according to the new sample list. The displayed tSNE plot will contain the common samples from both the tSNE and the provided sample list.
- locate: The default (TRUE) allows you to select and save individual samples of interest on the plot. If FALSE, it only displays the tSNE plot.
- title: The title of the plot, in quotes. If missing, the function will provide a default.

Cluster a tSNE result with k-mean clustering methods using an optional graphical user interface (GUI) for changing number of clusters: **K + 1**. Increase the number of clusters by 1. **K – 1**. Decrease the number of clusters by 1. **Retry**. Recluster the data using different initial cluster centers. **Save**. Save the selected cluster results (sample_list: sample id + cluster id). **Done**. End your clustering session when you are finished clustering.	clustertSNE (tsne=fldm_tsne, num_clusters=3, num_iterations=20, change_num=TRUE, title="") • tsne: The tSNE object to be clustered. The default is set to fldm_tsne, which is the output from the autoAnalysis() function. • num_clusters: Number of clusters, and the default is set to 3. • num_iterations: Number of iterations to be performed, and the default is set to 20. • change_num: The default (TRUE) is set to interactively change the number of clustering according to the data and to save cluster results. When set to FALSE, it only displays the cluster plot and return the cluster results. • title: Title of the plot in quotes. If missing, the function will provide a default.

Summary of ANOVA Functions

Given an EXP object with two or more sample groups, performs analysis of variance for all the sample groups and pairwise analysis for any two groups and eturns an ANOVA object with 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 pairwised 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.	anova <- ANOVA(exp, sample_list=FALSE, sample_list_file=FALSE, display_summary=TRUE) • exp: The exp object containing the sample list and group information to be used for calculations. • sample_list_file: The default is FALSE. To use a sample list from a file, set this argument to TRUE. A GUI window appears, allowing the user to choose the sample list from a file. • sample_list: The default is FALSE. To use a different sample list, set this argument to the sample list. • display_summary: The default is TRUE to display a plot with ANOVA summary information.
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.

	• 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: • 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.	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 the group names of interest in quotes: • sample_group1 = "group_name1", sample_group2 = "group_name2"
Display the number of significantly expressed genes (less than the given pvalue_threshold) for each pair of sample groups.	pairwiseANOVASummary (anova, pvalue_threshold=0.05, title) • anova: Previously generated ANOVA object from which p-values will be displayed. • pvalue_threshold: The standard level of significance to be used. By default this is set to 0.05. • title: The title of the plot in quotes.

Find statistically significant genes with mean expression ratios between two given sample groups that are equal to or greater than a given threshold.

Genes are considered 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.

The function returns the gene list and displays a Volcano plot.

foldChangeAnalysis(anova, sample_group1, sample_group2, foldchange_threshold = 1, pvalue_threshold = 0.05, xlim=c(-5,5), ylim=c(0,5), display_plot = TRUE, locate = FALSE, title)

- anova: Previously generated ANOVA object from which p-values will be displayed.
- sample_group1, sample_group2: Sorting instructions, as follows:

  To Sort genes by overall p-value of ANOVA, set these values 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:

  sample_group1 = "group_name1", sample_group2 = "group_name2"
- foldchange_threshold: 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.
- pvalue_threshold: Only those genes with p-values equal or less than the threshold are displayed. The default is 0.05, so all genes will be displayed up to but not greater than the top_gene_number.
- xlim: The fold change display range, default is -5 to 5 (log2 space).
- ylim: The p-value display range, default is 0 to 5 (log10 space).
- display_plot: The default (TRUE) displays a Volcano plot. The returned gene list will contain all the selected genes. If FALSE, no plot will be displayed. The returned gene list contains all the genes whose mean expression ratios between two sample groups (1) exceed the foldchange_threshold and p_value and are (2) equal to or less than the pvalue_threshold.
- locate: The default (FALSE) returns a gene list that contains all the genes whose mean expression ratios between two sample groups (1) exceed the foldchange_threshold and p_value and are (2) equal to or less than the pvalue_threshold. If TRUE, the returned gene list contains all the selected genes and you will be able to interactively select interested genes in the plot.
- "title": Title of the plot, in quotes. If missing, the function will provide a default.

Summary of HC Functions

Given an EXP object, conduct unsupervised hierarchical clustering (HC) analysis by the Pearson Correlation method. Co-profiled genes are clustered together and samples are clustered by normalized Euclidian distance (distance/number of genes), representing the average fold change. HC analysis uses the “complete linkage” method (a bottom-up method) to find similar clusters.

This function returns the HC object, which lists the following data frames:

- obj_type, the type of object: “HC”
- sample_cluster_method: The distance method used for sample clustering.
- gene_cluster_method: The distance method used for gene clustering.
- sample_cluster: The cluster structure and cluster distance for each sample. Column 1 contains sample names. The remaining columns are cluster structures (from first level to the last level) and cluster distances (from first level to the last level).
- gene_cluster: The cluster structure and cluster distance for each gene. Column 1 contains gene IDs. The remaining columns are cluster structures (from first level to the last level) and cluster distances (from first level to the last level). Cluster distances are converted to correlation coefficients when genes are clustered by the Pearson method.
- sample_list: Sample list of EXP object
- gene_list: Gene list of EXP object
- log2ex_data: Gene expression in log2 domain of EXP object

When samples and/or genes are annotated with group information, the HC plot will display the group symbols along the dendrogram and the names.

If the figure margins for the HC plot are incorrect, see setHCMarginFactor and resetHCMarginFactor.

```
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", representing low, medium, and high gene expression. 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 displayed.
- display_gene_names: The default (TRUE) is set to display gene names in the HC plot. If FALSE, gene names will not be displayed.
- heatmap_display: The default (TRUE) is set to display the heatmap. If FALSE, the heatmap will not be displayed.

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.

The display option changes the display only, not the clustering. 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 resulting HC can be displayed again with different options with displayHC function.
The HC heatmap display with multiple sample or gene groups can be problematic for low computer display resolution (e.g., 1024x768) or the text size in the Windows OS is set more than 100 percent. The displayHC function will automatically adjust the symbol display margin to avoid the "figure margins too large" error, and the new margin factor will be reported. However, the symbols could be displayed improperly. Use this function to increase the margin factor if the spacing between the symbols and the names is too large or to reduce the margin factor if the spacing is too small (the symbols are not completely visible). The set margin factor that you used will be persisted for all the subsequent displayHC functions.	setHCMarginFactor(mf) mf: The margin factor between 0.5 to 1. The typical value for a system display size of 125% is 0.8. If the figure margins are still too large, try lowering the mf.
Reset the HC display margin factor to the default value of 1. This is 100% of the system display size. The set margin factor that you used will be persisted for all the subsequent displayHC functions. To reset this setting to the default of 1, enter either of the following functions: resetHCMarginFactor or setHCMarginFactor(1)	resetHCMarginFactor ()
Given the hierarchical clustering (HC) and expression (EXP) objects, appends the expression data from the EXP object to the existing HC object and returns an updated HC object.	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 (HC) • exp: The expression object • cluster: The default (TRUE) clusters 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", representing low, medium, and high gene expression. 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, which can be "global_z_score", "gene_z_score", or "expression" • display_sample_names: The default (TRUE) displays sample names in the HC plot. When set to FALSE, sample names are not displayed. • display_gene_names: The default is set to display gene names in the HC plot. When set to FALSE, gene names are not displayed.
Display the results of hierarchical clustering (HC) analysis. When samples and/or genes are annotated with group information, the HC plot will display the group symbols along the dendrogram and the names. When the computer display resolution is low (e.g., 1024x768) or the text size in the Windows OS is set more than 100 percent, this display function will automatically adjust the symbol display margin to avoid the plot error, and the new margin factor will be reported. However, the symbols could be displayed improperly. If the spacing between the symbols and the names is too large, the margin factor should be increased by using setHCMarginFactor. If the spacing is too small (the symbols are not completely visible), the margin factor should be reduced. In the example below, the computer display resolution is only 1024x768 and the spacing between the sample group symbols and sample names are too large (where the reported margin factor from the previous HC is 0.6). Thus: setHCMarginFactor (0.7) displayHC (exp, display = "gene_z_score")	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") • hc: The object containing the hierarchical clustering information. • color_scheme: The predefined color scheme in HC. The default is "blue_white_red", representing low, medium, and high gene expression. 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 in the HC plot. When set to FALSE, sample names will not be displayed. • display_gene_names: The default (TRUE) is set to display gene names in the HC plot. When set to FALSE, gene names will not be displayed. • sample_name_trimed_size: The default (15) is set to the maximum length of characters for displaying any sample name in the HC plot. • gene_name_trimed_size: The default (10) is set to the maximum length of characters for displaying any gene name in the HC plot. • "title": Title of the plot in quotes. If missing, the function will provide a default. 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 any normalization. The heatmap of "global_z_score" and "expression" best represents similarity between samples while the heatmap of "gene_z_score" best represents the similarity between genes.
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 identity 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 subsclusters.
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 between any two samples in the cluster.	hc_sample_list <- getSampleClusterByThreshold (hc, threshold=TRUE, min_sample_num=3) • 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 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. • min_sample_num: The minimum number of samples required for a sample cluster.
Interactively identify sample clusters using the sample cluster dendrogram and return the identified cluster(s) as a sample list with cluster ID. 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.	hc_sample_list <- identifySampleClusters (hc) hc: The HC object.

Summary of Correlation Functions

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. The CORR object is a list of the following data frames: • obj_type: The “CORR” object • corr_method : The correlation method used for calculating gene correlation coefficient between two genes cross the selected samples • corr_gene_result: The result of gene correlation analysis, containing the following columns: TargetGeneID, QueryGeneID , Coefficient, GroupID • log2ex_data: The gene expression data (log2) of selected samples in sample groups sample_list: The sample list having the selected sample groups	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. The CORR object is a list of the following data frames: • obj_type: The “CORR” object • corr_method : The correlation method used for calculating gene correlation coefficient between two genes cross the selected samples • corr_gene_result: The result of gene correlation analysis, containing the following columns: TargetGeneID, QueryGeneID , Coefficient, GroupID • log2ex_data: The gene expression data (log2) of selected samples in sample groups sample_list: The sample list having the selected sample groups	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.) The CORR object is a list of the following data frames: • obj_type: The “CORR” object • corr_method : The correlation method used for calculating gene correlation coefficient between two genes cross the selected samples • corr_gene_result: The result of gene correlation analysis, containing the following columns: TargetGeneID, QueryGeneID , Coefficient, GroupID • log2ex_data: The gene expression data (log2) of selected samples in sample groups • sample_list: The sample list having the selected sample groups	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.

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.	displayCorrGenes(corr, query_gene_name = "", corr_threshold = 0.5, corr_pattern = "positive", "title") • corr: A given CORR object. • 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. • corr_threshold: The Correlation Coefficient threshold, range is 0 - 1. The default is set as 0.5. • corr_pattern: A flag to indicate genes are co-expressed or anti co-expressed with target genes. The default is set as "positive", options have "negative", "both". • "title": Title of the plot. If missing, the function will provide a default.

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 for mutation analysis.

Installing

Launch the Fluidigm Single Cell Package Library for commands for the current session.	library(fluidigmSC)
The first method to be called after installing the **fluidigmSC** package. This function must only be run once.	firstRun() Downloads the following required packages: lattice, tcltk2, rgl, plyr, IRanges, and DNAcopy.
Set your working directory.	setwd("pathway of your working directory") Example: setwd("C:/folder ")

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 all user-defined variables in the current session.	scVariables()

Categorize the R help functions for variant analysis into seven categories: 1. Get Started Functions 2. Read and Save Functions 3. Variant Data Management Functions 4. Variant Quality Functions 5. Hierarchical Clustering Analysis Functions 6. Fisher's Exact Test Functions 7. Variant Report Functions	scVarFunctions() When you are prompted, type a number between **1** and **7**, and then press **Enter**.

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.	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 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 <- readVCF(vcf_file = TRUE, targets = "") • vcf_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)	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.	vc <- 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 “_” options: 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.	vc <- 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.	vc <- vcUpdateSampleListFromList(vc, sample_list) • vc: The VC object to update. • 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.	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.	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.	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.
Updat variant group annotations directly from variant_annotat_list in VC in which the variant annotations are parsed from INFO during VCF data reading. Currently the supported variant annotation package is snpEFF.	vc <- vcUpdateVariantListFromVariantAnnotation(vc, by_annotation = "CHROM") • vc: The VC object to update. • by_annotation: Variant annotations include: CHROM (as default), GENE_SYMBOL, and VARIANT_TYPE.
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.	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 -10log_10p (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 quality is 0. • "OR", the quality of a variant of sample is calculated as 1 if variant metrics pass any one of metric thresholds. Otherwise the quality is 0.

Remove or Retain Sample and Variant Groups

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 0.50.
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 list, and return the VC object with the updated variant list.	vc <- vcRemoveVariantGroup(vc, variant_group_name) • 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 the variant list to determine whether they pass filter in VCF, and return the variants that failed in filter in the original variant calls.	failed_filter_variants <- vcGetFailVariantsFromVariantCall(vc) vc: A given VC object.
Given a VC object with quality table of SCQC, return high-quality variants. This function should be used in conjunction with vcUpdateVariantQuality, and it will return all the variants that have at least one sample with an AB or BB call.	variant_list <- vcGetFilteredVariantListByVariantQuality(vc) vc: A given VC object.
Given a VC object, check working variants in variant list, and return variants passing filter in original VCF data.	pass_filter_variants <- vcGetPassVariantsFromVariantCall(vc) vc: A given VC object.
Given Fisher's Exact Test result, return variants with significant changes in any pairwise sample groups.	selected_variant_list <- vcGetSignificantVariants(ft, pvalue_threshold = 0.05) • ft: A given Fisher Exact Test result. • pvalue_threshold: Threshold of pvalue in Fisher Exact test. The default is 0.50.

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 is greater than the threshold.	ref_variant_list_wcall <- vcGetVariantListBySampleGroupAlleleFrequency(vc, sample_group, allele_freq_threshold = 0.1) • vc: A given VC object. • sample_group: A given sample group. Replace with sample group name in quotes. • allele_freq_threshold: A threshold for minimum allele frequency of each variant in the given group. The default is 0.1.
Get variant list with genotyping information from a given sample.	ref_variant_list_wcall <- vcGetVariantListWithCallFromSample(vc, 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 toTRUE, only returns variant calls with GT as either hetero (AB) or homo variant (BB).
Change Group Colors and Symbols	
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.	vc <- vcSetSampleGroupColorAndSymbols (vc) vc: A given VC object.
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.	vc <- vcSetVariantGroupColorAndSymbols (vc) vc: A given VC object.
Restore the current color and symbol settings of sample groups back to their default settings.	vc <- vcClearSampleGroupColorAndSymbols (vc) vc: A given VC object with sample group annotation.
Restore the current color and symbol settings of the sample groups back to their default settings.	vc <- vcClearVariantGroupColorAndSymbols (vc) vc: A given VC object with variant group annotation.

Reset Sample and Variant Lists	
Replace a VC object's current list of samples with its original list of samples.	vc <- vcResetSampleList(vc) vc: A given VC object.
Replace a VC object's current list of variant with its original list of variants.	vc <- vcResetVariantList(vc) vc: A given VC object.
Merge Data	
Merge two VC objects into a single VC object.	vc <- vcMergeVCObj(vc1, vc2) • vc1: The first VC object. • vc2: The second VC object.
Evaluate Variant-Calling Performance	
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.	result <-vcEvalSampleVariants(vc, ref_variant_list_wcall) • vc: A given VC object. • ref_variant_list_wcall: A variant list with common GT. This should be replaced with the list.
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.	result <-vcEvalSampleGroupVariants(vc, ref_variant_list_wcall) • vc: A given VC object. • ref_variant_list_wcall: A variant list with common GT. This should be replaced with the list.

Managing Outliers

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. The default (TRUE) for the outlier_list_file dispays a file dialog box to select a sample list file. Alternatively, this can be set to the file path in quotes. This dialog box is not linked to the R application, and it is important not to click anywhere outside this	vc <- vcAddOutlierFromFile(vc, outlier_list_file=TRUE) • vc: A given VC object. • outlier_list_file: An outlier list file.

dialog box before completing your file selection. If you click outside this dialog box, it will be hidden behind the R application window. To find it, minimize the R application window or press Alt+Esc to toggle through the open windows.	
Based on the defined sample list of outliers, move outlier samples 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 in quotes.
Move an outlier sample based on given outlier sample name to the outlier list, and return the updated VC object.	vc <- vcAddOutlierFromName (vc, “sample_name”) • vc: A given VC object. • “sample_name”: The outlier to be moved to the outlier list. Replace with the name of the sample set 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 for a given VC object.	• vc <- vcRestoreOutlierFromName(vc, sample_name) • • vc: A given VC object. • sample_name: Name of the outlier sample to be restored. Replace with the sample name in quotes.

Displaying Variant Data

Summary of Fisher's Exact Test Functions

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. Returns an ft object that contains test result.	ft <- vcPerformFisherTest(vc) vc: A given VC object.

Display the results of the Fisher's Exact test. Interactively locate and save variants of interest and display a Manhattan plot. You can select significant variants in the test either by variants having significant p-value (e.g. pvalue<=0.05) or by manually selecting variants (where the locate function if locate is set as TRUE). Returns selected_variant_list.	selected_variant_list <- vcDisplayFisherTestResults(ft, sample_group1, sample_group2, pvalue_threshold = 0.05, locate = FALSE, "title") • ft: Fisher test result from the function of vcPerformFisherTest. • sample_group1: The first given sample group name. Replace with the name in quotes. • sample_group2: The second given sample group name. Replace with the name in quotes. • pvalue_threshold: The significant pvalue threshold set for the Fisher EXACT Test, where the default is 0.05. • locate: The default (TRUE) is set to interactively locate and to save variants of interest. When set to FALSE, it only displays the plot. • "title": Title of the plot. If missing, the function will provide a default.

Summary of HC Functions

Given a VC object, conduct unsupervised hierarchical clustering (HC) analysis for variant data.	vcHC(vc, dendrogram = "both", color_scheme="black_blue_green_red_grey", missing_data_as_AA = FALSE, display_sample_names = TRUE, display_variant_names = TRUE, heatmap_display = TRUE) • 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.
Displays the results of HC analysis of variant data.	vcDisplayHC(hc, color_scheme = "black_red_green_blue_grey", display_sample_names = TRUE, display_variant_names = TRUE, "title") • 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.
Exports the heatmap of a given hierarchical clustering analysis to a tab-delimited text file. Values will remain in their respective locations.	vcExportHCHeatmap(hc) hc: The HC object for variant data.
Interactively identify sample clusters using the Sample Cluster dendrogram, and return the identified cluster(s) as a sample list containing SampleID and a ClusterID.	select_sample_clusters <- vcIdentifySampleClusters (hc) hc: The HC object.
Interactively identify variant clusters using the Variant Cluster dendrogram. Returns a list.	select_variant_clusters <- vcIdentifyVariantClusters (hc) hc: The HC object.

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:
  - Mutations not in bulk cell control samples
  - Mutations in bulk cell case samples
  - No indel
  - FLITER flag as "PASS"
  - MISSENSE
  - 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.

vcIdentifySigVariants()

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.

	A	B
1	**obj_type**	
2	EXP	
3		
4		

Data Frame: data_type

This data frame contains the type of org_data, which can be Ct or LinearExp.

	A	B	C
1	**data_type**		
2	Ct		
3			
4			
5			
6			

Data Frame: lod

This data frame contains the Limit of Detection (LoD) for the org_data.

	A	B
1	lod	
2	24	
3		
4		
5		
6		

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 be sample names and the first column holds the gene names (or assay names). Missing values are replaced with
-1.

	A	B	C	D	E	F	G	H	I	J	K	L	
1	ID	CT1_94-1	CT1_51-0	CT1_95-1	CT1_50-1	CT1_96-1	CT1_49-4	CT1_48-1	CT1_01-0	CT1_47-1	CT1_02-1	CT1_46-1	CT
2	G01	8.268477	999	8.134847	8.462971	8.14192	6.086499	8.644814	16.02776	8.241851	7.411101	7.441316	9.
3	G02	999	999	999	999	999	18.85328	999	999	999	999	999	
4	G03	999	999	18.65291	999	999	15.63785	999	999	999	999	999	
5	G04	19.36748	999	17.08223	16.01653	16.47481	14.67247	16.25628	24.21015	15.80213	17.72738	16.91809	15
6	G05	999	999	999	999	999	999	999	999	999	999	999	
7	G06	12.31763	999	12.66304	12.90143	11.9511	10.70307	12.77088	999	12.25398	12.55043	11.68322	11
8	G07	18.78365	999	19.86548	19.8807	17.09944	14.37997	16.50371	999	16.4024	17.16569	17.11522	16
9	G08	999	999	999	999	999	999	999	999	999	21.38913	999	
10	G09	17.1008	999	16.73603	17.56185	15.11695	13.45578	14.94789	999	15.47933	15.97848	15.50741	15
11	G10	11.26945	999	10.91833	12.62333	10.91519	9.110173	10.66574	15.52096	10.39701	10.4834	10.28686	10
12	G11	11.44553	999	11.14735	11.03208	10.20762	8.466394	10.7116	28.60625	9.53697	10.06127	9.740999	1
13	G12	10.5622	999	10.32265	12.55868	10.20786	7.954943	9.990094	999	9.454955	9.506376	9.46297	10
14	G13	21.05811	999	25.55137	999	22.62463	17.22987	26.1052	999	21.88399	22.5879	19.52199	25
15	G14	999	999	999	999	999	999	999	999	999	999	999	

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.

	A	B	C
1	**GeneID**	**GroupID**	
2	G49	Ga	
3	G46	Ga	
4	G05	Ga	
5	G86	Ga	
6	G89	Ga	

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.

	A	B	C
1	**SampleID**	**GroupID**	
2	CT1_94-1	CT1	
3	CT1_95-1	CT1	
4	CT1_50-1	CT1	
5	CT1_96-1	CT1	
6	CT1_49-4	CT1	

Data Frame: outlier_list

This data frame contains a list of identified outliers: sample names with sample group ID.

	A	B	C
1	**SampleID**	**GroupID**	
2	CT1_51-0	CT1	
3	CT1_01-0	CT1	
4	CT2_27-0	CT2	
5	CT2_56-1	CT2	
6	CT2_06-0	CT2	
7			

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.

	A	B	C	D	E	F	G	H	I	J	K
1	ID	CT1_94-1	CT1_95-1	CT1_50-1	CT1_96-1	CT1_49-4	CT1_48-1	CT1_47-1	CT1_02-1	CT1_46-1	CT1_03-1
2	G49	0	0	0	0	0	0	0	0	0	0
3	G46	0	0	0	0	0	0	0	0	0	0
4	G05	0	0	0	0	0	0	0	0	0	0
5	G86	0	0	0	0	0	0	0	0	0	4.086101
6	G89	0	0	0	4.876596	0	0	0	0	0	0
7	G93	0	0	0	0	0	0	0	0	0	0
8	G82	0	0	0	0	0	0	0	0	0	0
9	G79	0	0	0	0	0	0	0	0	0	0
10	G08	0	0	0	0	0	0	0	2.61087	0	0
11	G14	0	0	0	0	0	0	0	0	0	0
12	G80	0	0	0	0	0	0	0	0	0	0
13	G51	0	0	0	0	0	0	0	0	0	0
14	G71	0	0	0	0	0	5.11123	0	0	0	0
15	G27	0	0	0	0	0	0	0	0	0	0

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.

	A	B	C	D
1	ID	CT1	CT2	
2	G49	0	0	
3	G46	0	0	
4	G05	0.241061	0	
5	G86	1.125515	0.696523	
6	G89	0.334612	0	
7	G93	0	0	
8	G82	0	0	
9	G79	0	0	
10	G08	0	0	
11	G14	0	0	
12	G80	0	0	
13	G51	0	0	
14	G71	0	2.893981	
15	G27	0	2.870704	

Data Frame: summary

This data frame contains a summary of the experiment data.

	A	B
1	**Category**	**Value**
2	Number of original samples	192
3	Number of original genes	96
4	Number of working genes	94
5	Number of working sample groups	2
6	Number of working samples in CT1	94
7	Number of working samples in CT2	93
8	Number of outliers	5
9		

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.

	A	B	C
1	**obj_type**		
2	VC		
3			
4			
5			
6			

Data Frame: data_format

This data frame contains the version of the VCF file.

	A	B	C
1	**data_format**		
2	VCFv4.1		
3			
4			
5			
6			

Data Frame: genome_build

This data frame uses either the CRCR37 or hg19 reference genome.

	A	B	C	D	E	F	G	H	I	J	K
1	**genome_build**										
2	##reference=file:///home/references/genomes/human/ucsc_hg19_all/Homo_sapiens_assembly19.fasta										
3											
4											
5											
6											

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.

	A	B	C
1	**SampleID**	**GroupID**	
2	CT1-gDNA_S01	CT1-gDNA	
3	CT1-gDNA_S02	CT1-gDNA	
4	CT1-gDNA_S03	CT1-gDNA	
5	CT1-SC_1_S01	CT1-SC	
6	CT1-SC_1_S02	CT1-SC	
7	CT1-SC_1_S03	CT1-SC	
8	CT1-SC_1_S04	CT1-SC	
9	CT1-SC_1_S05	CT1-SC	
10	CT1-SC_1_S06	CT1-SC	
11	CT1-SC_1_S07	CT1-SC	
12	CT1-SC_1_S08	CT1-SC	
13	CT1-SC_1_S09	CT1-SC	
14	CT1-SC_1_S10	CT1-SC	
15	CT1-SC_1_S11	CT1-SC	

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.

	A	B	C
1	**VariantID**	**GroupID**	
2	rs2072454	VC_1	
3	rs4947986	VC_1	
4	7:55240708_T_TG	VC_1	
5	7:55241604_T_TC	VC_1	
6	rs2293347	VC_1	
7	7:55269077_T_A	VC_1	
8	rs35775721	VC_1	
9	rs2023748	VC_1	
10	rs206075	VC_2	
11	rs206076	VC_2	
12	13:32929501_G_GT	VC_2	
13	rs9534262	VC_2	
14	rs1625895	VC_2	

Data Frame: mult_allele_data

This data frame contains a list of variants having multiple alleles in variant call.

	A	B	C	D	E	F
1	**CHROM**	**POS**	**ID**	**REF**	**ALT**	
2	7	55238261	.	TC	TCC,T	
3						
4						
5						
6						

Data Frame: org_data

This data frame contains the data in the original VCF.

B	C	D	E	F	G	H	I	J	K	L	M	N	
#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	CT1-gDNA_S01	CT1-gDNA_S02	CT1-gDNA_S03	CT1-SC_1_S01	C
7	55086938	.	G	A	311.54	PASS	AC=1;AF=5.6	GT:AD:DP	0/0:4,0:4:12:0,1	0/0:40,0:40:99:0	0/0:6,0:6:18:0,1	0/0:42,0:42:99:(	0/
7	55086993	.	G	A	882.63	PASS	AC=1;AF=5.6	GT:AD:DP	0/0:5,0:5:9:0,9,1	0/0:78,0:78:99:0	0/0:7,0:7:21:0,21	0/0:73,0:73:99:(	0/
7	55086995	.	G	A	902.67	PASS	AC=1;AF=5.7	GT:AD:DP	0/0:4,0:4:9:0,9,1	0/0:79,0:79:99:0	0/0:5,0:5:12:0,12	0/0:66,0:66:99:(	0/
7	55087016	.	G	A	583.16	PASS	AC=1;AF=6.4	GT:AD:DP	./.	0/0:40,0:40:99:0	0/0:2,0:2:6:0,6,8	0/0:42,0:42:99:(	0/
7	55087050	.	A	T	622.17	PASS	AC=1;AF=6.3	GT:AD:DP	./.	0/0:40,0:40:99:0	0/0:2,0:2:6:0,6,7	0/0:44,0:44:99:(	0/
7	55087057	.	A	G	112.22	PASS	AC=1;AF=6.4	GT:AD:DP	./.	0/0:38,2:38:99:0	0/0:2,0:2:3:0,3,4	0/0:43,0:43:99:(	0/
7	55214348	rs2072454	C	T	183560.1	PASS	AC=92;AF=0.4	GT:AD:DP	0/1:77,60:131:99	0/1:63,71:129:99	0/1:63,77:134:99	0/1:43,94:131:9	1/
7	55220281	.	G	T	1457.06	PASS	AC=1;AF=5.4	GT:AD:DP	0/0:94,0:94:99:0	0/0:89,0:89:99:0	0/0:87,1:87:99:0	0/0:84,0:84:99:(	0/
7	55221655	rs4947986	G	A	48951.58	PASS	AC=86;AF=0.4	GT:AD:DP	0/1:26,14:38:99:	0/1:19,24:41:99:	0/1:27,16:41:99:	0/1:18,24:41:99	0/
7	55229285	.	G	T	1462.12	PASS	AC=1;AF=5.4	GT:AD:DP	0/0:132,0:132:99	0/0:135,0:135:99	0/0:143,0:143:99	0/0:161,1:161:9	0/
7	55237933	.	T	C	326.14	PASS	AC=21;AF=0.1	GT:AD:DP	0/0:1,0:1:3:0,3,3	0/0:1,0:1:3:0,3,3	./.	./.	0/
7	55237935	.	T	C	38.12	PASS	AC=5;AF=0.0	GT:AD:DP	./.	0/0:1,0:1:3:0,3,3	./.	./.	0/
7	55237982	.	T	C	34.44	PASS	AC=2;AF=0.0	GT:AD:DP	0/0:3,0:3:6:0,6,8	0/0:2,0:2:3:0,3,4	./.	0/0:1,0:1:3:0,3,	0/

Data Frame: org_data_header

This data frame contains the header of the original VCF.

	A
1	as.vector(header)
2	##fileformat=VCFv4.1
3	##FILTER=<ID=LowQual,Description="Low quality">
4	##FILTER=<ID=WGApipe_variant_filter,Description="QD < 2.0 || FS > 60.0 || MQ < 40.0 || MappingQualityRankSum < -12.5 || ReadPosRankSum < -8.0 || AF < 0.1 || QUAL < 200 || DP <20 ">
5	##FORMAT=<ID=AD,Number=.,Type=Integer,Description="Allelic depths for the ref and alt alleles in the order listed">
6	##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth (reads with MQ=255 or with bad mates are filtered)">
7	##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
8	##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
9	##FORMAT=<ID=PL,Number=G,Type=Integer,Description="Normalized, Phred-scaled likelihoods for genotypes as defined in the VCF specification">
10	##GATKCommandLine=<ID=UnifiedGenotyper,Version=2.6-4-g3e5ff60,Date="Tue Jan 14 14:17:39 PST 2014",Epoch=1389737859241,CommandLineOptions="analysis_type=UnifiedGenotyper
11	##GATKCommandLine=<ID=VariantFiltration,Version=2.6-4-g3e5ff60,Date="Tue Jan 14 14:34:14 PST 2014",Epoch=1389738854610,CommandLineOptions="analysis_type=VariantFiltration inp
12	##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in genotypes, for each ALT allele, in the same order as listed">
13	##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT allele, in the same order as listed">
14	##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles in called genotypes">

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.

	A	B	C	D	E	F	G	H	I	J	K	L
1	ID	VARIANT_ID	#CHROM	POS	ID	REF	ALT	GENE_SYMBOL	VARIANT_TYPE	VARIANT_FUNCTION	POSITION_CATEGORY	AMINO_ACID_CHANGE
2	rs2072454	rs2072454	7	55214348	rs2072454	C	T	EGFR	SNP	SYNONYMOUS_CODING	CODING	Untitled
3	rs4947986	rs4947986	7	55221655	rs4947986	G	A	EGFR	SNP	SYNONYMOUS	Untitled	Untitled
4	7:55240708_T_TG	7:55240708_T_TG	7	55240708	.	T	TG	EGFR	INSERTION	FRAME_SHIFT	Untitled	Untitled
5	7:55241604_T_TC	7:55241604_T_TC	7	55241604	.	T	TC	EGFR	INSERTION	SYNONYMOUS	Untitled	Untitled
6	rs2293347	rs2293347	7	55268916	rs2293347	C	T	EGFR	SNP	SYNONYMOUS_CODING	CODING	Untitled
7	7:55269077_T_A	7:55269077_T_A	7	55269077	.	T	A	EGFR	SNP	SYNONYMOUS	Untitled	Untitled
8	rs35775721	rs35775721	7	1.16E+08	rs35775721	C	T	MET	SNP	SYNONYMOUS_CODING	CODING	Untitled
9	rs2023748	rs2023748	7	1.16E+08	rs2023748	G	A	MET	SNP	SYNONYMOUS_CODING	CODING	Untitled
10	rs206075	rs206075	13	32913055	rs206075	A	G	BRCA2	SNP	SYNONYMOUS_CODING	CODING	Untitled
11	rs206076	rs206076	13	32915005	rs206076	G	C	BRCA2	SNP	SYNONYMOUS_CODING	CODING	Untitled
12	13:32929501_G_GT	13:32929501_G_GT	13	32929501	.	G	GT	BRCA2	INSERTION	SYNONYMOUS	Untitled	Untitled
13	rs9534262	rs9534262	13	32936646	rs9534262	T	C	BRCA2	SNP	SYNONYMOUS	Untitled	Untitled
14	rs1625895	rs1625895	17	7578115	rs1625895	T	C	TP53	SNP	SYNONYMOUS	Untitled	Untitled

Data Frame: GT

This data frame contains the encoded genotyping data. They are a subset of those in the org_data data frame.

	A	B	C	D	E	F	G	H	I
1	ID	CT1-gDNA_S01	CT1-gDNA_S02	CT1-gDNA_S03	CT1-SC_1_S01	CT1-SC_1_S02	CT1-SC_1_S03	CT1-SC_1_S04	CT1-SC_1_S05
2	rs2072454	2	2	2	2	3	2	2	2
3	rs4947986	2	2	2	2	2	2	2	2
4	7:55240708_T_TG	1	1	1	1	1	2	1	1
5	7:55241604_T_TC	1	1	1	2	2	1	2	1
6	rs2293347	2	2	2	2	2	2	2	2
7	7:55269077_T_A	2	2	2	2	1	2	2	2
8	rs35775721	2	2	2	2	2	3	2	2
9	rs2023748	2	2	2	2	2	1	2	2
10	rs206075	3	3	3	3	3	3	3	3
11	rs206076	3	3	3	3	3	3	3	3
12	13:32929501_G_GT	1	1	1	1	1	1	1	1
13	rs9534262	2	2	2	2	2	2	2	2
14	rs1625895	3	3	3	3	3	3	3	3

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.

	A	B	C	D	E	F	G	H
1	ID	CT1-gDNA_S01	CT1-gDNA_S02	CT1-gDNA_S03	CT1-SC_1_S01	CT1-SC_1_S02	CT1-SC_1_S03	CT1-SC_1_S04
2	rs2072454	0.437956204	0.529850746	0.55	0.686131387	0.992753623	0.503875969	0.781021898
3	rs4947986	0.35	0.558139535	0.372093023	0.571428571	0.181818182	0.463414634	0.170731707
4	7:55240708_T_TG	0	0	0.011363636	0.013513514	0	0.121212121	0
5	7:55241604_T_TC	0	0	0	0.119047619	0.195121951	0	0.1
6	rs2293347	0.562043796	0.4609375	0.457746479	0.339869281	0.3	0.511278195	0.315384615
7	7:55269077_T_A	0.281690141	0.266666667	0.236111111	0.197183099	0	0.388235294	0.317073171
8	rs35775721	0.5	0.5	0.547619048	0.214285714	0.204545455	0.953488372	0.152173913
9	rs2023748	0.25	0.184782609	0.298969072	0.206185567	0.192307692	0.079545455	0.72826087
10	rs206075	1	1	1	1	1	0.992307692	1
11	rs206076	1	1	0.994350282	1	1	1	1
12	13:32929501_G_GT	0	0	0	0	0	0	0
13	rs9534262	0.756097561	0.6375	0.80952381	0.821428571	0.897435897	0.46835443	0.825
14	rs1625895	0.976744186	1	1	1	1	1	1

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.

	A	B	C	D	E	F	G	H	
1	ID	CT1-gDNA_S01	CT1-gDNA_S02	CT1-gDNA_S03	CT1-SC_1_S01	CT1-SC_1_S02	CT1-SC_1_S03	CT1-SC_1_S04	CT
2	rs2072454	131	129	134	131	137	129	131	
3	rs4947986	38	41	41	41	42	39	39	
4	7:55240708_T_TG	95	86	94	75	88	104	84	
5	7:55241604_T_TC	44	46	40	44	43	42	42	
6	rs2293347	131	124	141	147	10	130	125	
7	7:55269077_T_A	71	75	71	69	2	86	78	
8	rs35775721	42	46	42	42	42	41	44	
9	rs2023748	88	88	94	93	99	84	88	
10	rs206075	122	125	125	121	127	129	128	
11	rs206076	174	176	176	174	188	199	193	
12	13:32929501_G_GT	47	46	53	55	52	56	47	
13	rs9534262	78	77	80	80	77	76	76	
14	rs1625895	42	41	43	41	48	43	42	

Data Frame: GQ

This data frame contains conditional genotype quality, encoded as a phred quality - 10log_10p, They are a subset of those in the org_data data frame.

	A	B	C	D	E	F	G	H	
1	ID	CT1-gDNA_S01	CT1-gDNA_S02	CT1-gDNA_S03	CT1-SC_1_S01	CT1-SC_1_S02	CT1-SC_1_S03	CT1-SC_1_S04	CT
2	rs2072454	99	99	99	99	99	99	99	
3	rs4947986	99	99	99	99	99	99	99	
4	7:55240708_T_TG	99	99	99	99	99	99	99	
5	7:55241604_T_TC	99	99	87	63	99	96	36	
6	rs2293347	99	99	99	99	86	99	99	
7	7:55269077_T_A	99	99	99	99	6	99	99	
8	rs35775721	99	99	99	99	99	99	39	
9	rs2023748	99	99	99	99	99	99	99	
10	rs206075	99	99	99	99	99	99	99	
11	rs206076	99	99	99	99	99	99	99	
12	13:32929501_G_GT	99	99	99	99	99	99	99	
13	rs9534262	99	99	99	99	37	99	99	
14	rs1625895	99	99	99	99	99	99	99	

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.

	A	B	C	D	E	F	G	H	I	J	
1	ID	CT1-gDNA_N	CT1-gDNA_VC	CT1-gDNA_M	CT1-gDNA_AA	CT1-gDNA_AB	CT1-gDNA_BB	CT1-gDNA_NC	CT1-gDNA_AF	CT1-gDNA_VF	CT1
2	rs2072454	3	3	0	0	3	0	0	0.5	1	
3	rs4947986	3	3	0	0	3	0	0	0.5	1	
4	7:55240708_T_	3	0	0	3	0	0	0	0	0	
5	7:55241604_T_	3	0	0	3	0	0	0	0	0	
6	rs2293347	3	3	0	0	3	0	0	0.5	1	
7	7:55269077_T_	3	3	0	0	3	0	0	0.5	1	
8	rs35775721	3	3	0	0	3	0	0	0.5	1	
9	rs2023748	3	3	0	0	3	0	0	0.5	1	
10	rs206075	3	3	0	0	0	3	0	1	1	
11	rs206076	3	3	0	0	0	3	0	1	1	
12	13:32929501_G	3	0	0	3	0	0	0	0	0	
13	rs9534262	3	3	0	0	3	0	0	0.5	1	
14	rs1625895	3	3	0	0	0	3	0	1	1	
15	rs148924904	3	3	0	0	0	3	0	1	1	

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

	A	B	C	D	E	F	G	
1	ID	CT1-gDNA_S01	CT1-gDNA_S02	CT1-gDNA_S03	CT1-SC_1_S01	CT1-SC_1_S02	CT1-SC_1_S03	CT
2	rs2072454	1	1	1	1	1	1	
3	rs4947986	1	1	1	1	1	1	
4	7:55240708_T_TG	1	1	1	1	1	1	
5	7:55241604_T_TC	1	1	1	0	1	1	
6	rs2293347	1	1	1	1	1	1	
7	7:55269077_T_A	1	1	1	1	0	1	
8	rs35775721	1	1	1	1	1	1	
9	rs2023748	1	1	1	1	1	1	
10	rs206075	1	1	1	1	1	1	
11	rs206076	1	1	1	1	1	1	
12	13:32929501_G_GT	1	1	1	1	1	1	
13	rs9534262	1	1	1	1	0	1	
14	rs1625895	1	1	1	1	1	1	
15	rs148924904	1	1	1	1	1	1	

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).

Data Frame: outlier_list

	A	B	C
1	**SampleID**	**GroupID**	
2	CT1-SC_0_S45	CT1-SC	
3	CT1-SC_0_S44	CT1-SC	
4	CT2-SC_1_S44	CT2-SC	
5	CT2-SC_1_S45	CT2-SC	
6			

Appendix E: References for Gene Expression

Aguilo, F., S. Avagyan, A. Labar, A. Sevilla, D. F. Lee, P. Kumar, I. R. Lemischka, B. Y. Zhou, and H. W. Snoeck. "*Prdm16* is a physiologic regulator of hematopoietic stem cells." *Blood* 117 (2011): 5057-5066.

Bengtsson, M., A. Ståhlberg, P. Rorsman, and M. Kubista. "Gene expression profiling in single cells from the pancreatic islets of Langerhans reveals lognormal distribution of mRNA levels." *Genome Research* 15 (2005): 1388-1392.

Brennecke P., S. Anders, J.K. Kim, A.A. Kołodziejczyk, X. Zhang, V. Proserpio, B. Baying, V. Benes, S.A. Teichmann, J.C. Marioni, and M.G. Heisler. “Accounting for technical noise in single-cell RNA seq experiments.” *Nat Methods* (2013): Nov 10(11):1093-5. doi: 10.1038/nmeth.2645. Epub 2013 Sep 22.

Buettner, Florian, Victoria Moignard, Berthold Göttgens, and Fabian J. Theis. "Probabilistic PCA of censored data: accounting for uncertainties in the visualization of high-throughput single-cell qPCR data." *Bioinformatics.* published 31 March 2014, 10.1093/bioinformatics/btu134.

Buganim, J. et al. "Single-cell expression analyses during cellular reprogramming reveal an early stochastic and a late hierarchic phase." *Cell* 150 (2012) 6:1209-22.

Chubb, J. R., T. Trcek, S. M. Shenoy, and R. H. Singer. "Transcriptional pulsing of a developmental gene." *Current Biology* 16 (2006): 1018-1025.

Daehwan, Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L. Salzberg. "*TopHat2*: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions." *Genome Biology* 14 (2013): R36, doi:10.1186/gb-2013-14-4-r36.

Dalerba, P. et al. "Single-cell dissection of transcriptional heterogeneity in human colon tumors." *Nat. Biotechnol.* 29 (2011): 1120-1127.

Devonshire, A. S., R. Elaswarapu, and C. A. Foy. "Applicability of RNA standards for evaluating RT-qPCR assays and platforms." *BMC Genomics* 12 (2011)*:* 118-127.

Diehn, M. et al. "Association of reactive oxygen species levels and radioresistance in cancer stem cells." *Nature* 458 (2009): 780-783.

Flatz, L. et al. "Single-cell gene-expression profiling reveals qualitatively distinct CD8 T cells elicited by different gene-based vaccines." *Proc. Natl. Acad. Sci. USA* 108 (2011): 5724-5729.

Guo, G., M. Huss, G. Q. Tong, C. Wang, L. L. Sun, N. E. Clarke, and P. Robson. "Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst." *Developmental Cell* 18 (2010): 675-685.

Langmead B. and S. Salzberg. "Fast gapped-read alignment with Bowtie 2." *Nature Methods* 9 (2012): 357-359

Li, Bo and C. Dewey. "RSEM: accurate transcript quantification from RNA Seq data with or without a reference genome." *BMC Bioinformatics* 12 (2011): 323, doi:10.1186/1471-2105-12-323.

Livak, K. J. and T. D. Schmittgen. "Analysis of relative gene expression data using real-time quantitative PCR and the 2(-Delta Delta C(T)) method." *Methods* 25 (2001): 402-408.

Livak, K.J. et al. "Methods for qPCR gene expression profiling applied to 1440 lymphoblastoid single cells." *Methods* 59 (2013): 71-9.

Moignard, V. et al. "Characterization of transcriptional networks in blood stem and progenitor cells using high-throughput single-cell gene expression analysis." *Nat Cell Biol*. 15(2013) 4:363-72

Pang, Z. P. et al. "Induction of human neuronal cells by defined transcription factors." *Nature* 476 (2011): 220-223.

Raj, A., C. S. Peskin, et al. "Stochastic mRNA synthesis in mammalian cells." *PLoS Biol*. 4 (2006): e309.

Tanabe, K. et al. "Maturation, not initiation, is the major roadblock during reprogramming toward pluripotency from human fibroblasts." *PNAS* 110(2013) 30:12172-9

Trapnell C, B. A. Williams, G. Pertea, A. M. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. Wold, and L. Pachter. "Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation." *Nature Biotechnology*: doi:10.1038/nbt.1621.

Trapnell, Cole, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J. Lennon, Kenneth J. Livak, Tarjei S. Mikkelsen, and John L. Rinn. "The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells." *Nat Biotechnol.* (2014): doi:10.1038/nbt.2859.

Vandesompele, J., K. De Preter, F. Pattyn, B. Poppe, N. Van Roy, A. De Paepe, and F. Speleman. "Accurate normalization of real-time quantitative RT- PCR data by geometric averaging of multiple internal control genes." *Genome Biology* 3 (2002): research0034.1-research0034.11.

Vincent, J. J. et al. "Single cell analysis facilitates staging of Blimp1-dependent primordial germ cells derived from mouse embryonic stem cells." *PLoS ONE* 6 (2011): e28960.

Wills, Q.F. et al. “Single-cell gene expression analysis reveals genetic associations masked in whole-tissue experiments.” *Nat Biotechnol* 31 (2013): 748-52.

For Statistical Methodologies

Gordon, A.D. (1999), Classification. Second Edition. London: Chapman and Hall / CRC.

Freedman, David A., Robert Pisani, and Roger Purves. *Statistics*, 4th ed. New York: W. W. Norton & Company, 2007.

Appendix F: References for Variant and Mutation Analysis

DePristo, M., E. Banks, R. Poplin, K. Garimella, J. Maguire, C. Hartl, A. Philippakis, G. del Angel, M.A. Rivas, M. Hanna, A. McKenna, T. Fennell, A. Kernytsky, A. Sivachenko, K. Cibulskis, S. Gabriel, D. Altshuler, and M. Daly. "A framework for variation discovery and genotyping using next-generation DNA sequencing data." Nature Genetics.43 (2011):491-498.

Yong Hou, Luting Song, Ping Zhu, Bo Zhang, Ye Tao, Xun Xu, Fuqiang Li, Kui Wu, Jie Liang, Di Shao, Hanjie Wu, Xiaofei Ye, Chen Ye, Renhua Wu, Min Jian, Yan Chen, Wei Xie, Ruren Zhang, Lei Chen, Xin Liu, Xiaotian Yao, Hancheng Zheng, Chang Yu, Qibin Li, Zhuolin Gong, Mao Mao, Xu Yang, Lin Yang, Jingxiang Li, Wen Wang, Zuhong Lu, Ning Gu, Goodman Laurie, Lars Bolund, Karsten Kristiansen, Jian Wang, Huanming Yang, Yingrui Li, Xiuqing Zhang, Jun Wang. " Single-Cell Exome Sequencing and Monoclonal Evolution of a JAK2-Negative Myeloproliferative Neoplasm." Cell. 148 (5) (2012):873-85.

McKenna A, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, M. A. DePristo. "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data." Genome Res. 20 (2010):1297-303.

Van der Auwera, G.A., M. Carneiro, C. Hartl, R. Poplin, G. del Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K. Garimella, D. Altshuler, S. Gabriel, M. DePristo. "From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline." Current Protocols in Bioinformatics.43 (2013):11.10.1-11.10.33.

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