Chapter 17

Become Competent in Generating RNA-Seq Heat Maps in One Day for Novices Without Prior R Experience

Kejin Hu

Abstract

Heat map visualization of RNA-seq data is a commonplace task. However, most laboratories rely on bioinformaticians who are not always available. Biological scientists are afraid to prepare heat maps independently because R is a programming platform. Here, using RNA-seq data for 16 differentially expressed genes in *WNT* pathway between embryonic stem cells and fibroblasts, I share a tutorial for novices without any prior R experience to master the skills, in one day, required for preparation of heat maps using the *pheatmap* package. Procedures described include installation of R, RStudio, and the *pheatmap* package, as well as hands-on practices for some basic R commands, conversion of RNA-seq data frame to a numeric matrix suitable for generation of heat maps, and defining arguments for the *pheatmap* function to make a desired heat map. More than 20 template scripts are provided to generate heat maps and to control the dimensions and appearances of the heat maps.

Key words Heat maps, RNA-seq, RStudio, pheatmap, Tutorial, WNT pathways, Human embryonic stem cells

1 Introduction

RNA sequencing (RNA-seq) has become a commonplace technology for the biological and biomedical research laboratories [1–4]. RNA-seq generates genome wide transcription data. It is challenging to present the differential expressions for such a large set of genes. One popular method for visualization of RNA-seq data is heat map presentation using a color gradient. However, preparation of high-quality RNA-seq heat maps usually requires a programming platform. The widely used programming platform for generation of heat maps of RNA-seq data is R. Most biological and biomedical scientists are afraid to try the programming platforms for generation of RNA-seq heat maps, and rely on

Electronic Supplementary Material: The online version of this chapter (https://doi.org/10.1007/978-1-0716-1084-8_17) contains supplementary material, which is available to authorized users.

bioinformaticians who may not understand the biological problems well, and are not always conveniently available.

Here, I share a laboratory protocol designed for biological scientists without any prior experience in R to quickly become proficient in preparation of heat maps using the R platform. In a single day, audiences will be able to prepare professional heat maps to visualize RNA-seq data. There are several available packages for preparing heat maps [5–8]. This protocol uses the pretty heat map package *pheatmap* [6]. In the morning, the audiences will be guided to download and install R and RStudio onto their computers, and then practice some essential R commands, functions, and scripts prior to generating heat maps. In the afternoon, audiences will generate heat maps using a demo RNA-seq data frame. More than 20 template scripts along with heat map figures with different features are provided for novice users even without prior experience of R to generate heat maps and to manipulate the shape and appearance of the heat maps. This tutorial aims to turn a difficult and lengthy learning process into an easy, short, and enjoying experience. The scripts provided in the tutorial also serve as a handy reference for the more experienced users. It is a great refresher for the infrequent users of R and the heat map package.

2 Materials and Equipment

2.1 Computers (PC Windows or Mac OS X) and Softwares

The RStudio 1.2 requires a 64-bit PC operating system and works exclusively with the 64-bit version of R. This protocol was written based on the latest version of RStudio 1.2.1335 associated with the R 3.6.1 on platform of Microsoft Windows 10 Enterprise, and runs well on Windows 7 (64-bit), as well as with the older version RStudio 1.1.463 associated with the R3.3.3 on iMac OS X 10.9.5. The R and the *pheatmap* package were downloaded from The Comprehensive R Archive Network (CRAN) repository. A free version of RStudio was downloaded from www.rstudio.com.

2.2 The DESeq2 Read Count Data Frame

The demo data are based on real RNA-seq experiments in the author's laboratory, and some have been published [4]. The RNA-seq experiments sequenced four human fibroblast samples, and three human embryonic stem cell (ESC) samples. The format of the table is similar to what a bioinformatician generates using DESeq2 and passes to his/her biological scientists. For the convenience of presentation, the table includes only 16 genes in the WNT signaling pathways so that the heat maps in the tutorial appear larger, clear, and legible. Ten of the genes have at least two times higher expression in ESCs, while the remaining six have at least two times higher expression in human fibroblasts. All of the 16 genes have q values less than 0.01. The normalized read counts cover a large range from 1.7 to 7,659, and the log2 fold changes range from −6.9 to 11.6. Many genes are expressed only in one cell type. A gene is empirically deemed not expressed in a cell when the average normalized read counts are less than 50 [2, 4].

The original table is available as supplementary material. It is also available from the author upon request. If it is difficult to download the original RNA-seq data RNAseqHeatmap_wnt.csv, you can make a short table in Excel using the data in Table 1 and save the data in csv format.

After becoming proficient by practicing this protocol, users can generate heat maps with their own data with similar RNA-seq data frame in the format of *.csv*. One should convert his/her RNA-seq read count table in the Excel file to a comma separated values (csv) format. This can be done easily in Excel using the "Save As" function.

2.3 Download R and RStudio

RStudio is a user-friendly version of R. We will use RStudio to generate heat maps because it is more convenient to use. RStudio requires R to function. Therefore, we have to download R first. All commands and functions are the same in R and RStudio and you can practice in either one or in both after you will download the softwares onto your computer. The processes for downloading R and R Studio are provided in the following sections.

2.4 Downloading R

The steps may vary depending on the types of your computer systems, PC Windows, or Mac OS X. You may have to ask the IT personnel to help if unexpected problems occur.

1. Go to www.r-project.org/.
2. Click "CRAN mirror" under "Getting Started" subtitle.
3. Find the nearest mirror to you. In my case, it is National Institute for Computational Sciences, Oak Ridge, TN. Click on the link: https://cran.nics.utk.edu/cran/.
4. In the next page, click Download R for Windows.
5. In the R for Windows page, click "Install R for the First Time".
6. Click on "Download 3.6.1 for Windows". This is the latest version of R on the time of this protocol written, released on 2019-07-05. The version may be newer at the time of your downloading.
7. You will see a dialog box. Click Save.
8. In the dialog box, you will see a message of "R-3.6.1-win.exe" finished downloading. Click Open Folder.
9. In a new window, find the file of R-3.6.1-win and double click it.
10. You will see a new Windows dialog box asking "Do you want to allow this app to make changes to your device?". Click Yes.

Table 1
Normalized read counts of RNA-seq data

external_gene.x	Fibroblast_1	Fibroblast_2	Fibroblast_3	Fibroblast_4	ESC1	ESC2	ESC3
MYCN	5.021377	1.703858	0	0	6104.348	3031.534	4661.45
PCDH1	5.503161	8.065378	13.17402	26.95382	6143.231	3574.371	7659.542
CTNNA2	2.020892	8.358199	5.957563	0	456.1059	410.3579	170.5276
PCDHA11	18.78246	0	11.41216	16.11147	362.9158	184.3101	717.943
PCDHB16	30.09837	17.23551	35.65102	60.80093	339.4942	401.5489	429.4128
AXIN2	95.74123	122.3081	103.293	240.7407	1662.648	469.3609	1558.177
FZD9	18.21673	32.58625	12.29648	29.80337	118.6903	179.4875	49.63852
FRZB	241.2931	155.9666	61.00167	155.9059	1122.096	289.2833	190.3868
ADSS	2152.163	2494.122	2069.988	1857.153	4914.726	4859.624	4350.743
BMPR1A	1771.996	1030.246	2136.186	2000.28	3813.509	3350.53	3679.227
PPARD	2968.091	3330.526	3352.589	3989.729	929.0336	936.9425	922.8081
PCDHGA10	1104.69	645.9197	855.0272	1412.63	103.5171	384.8895	126.412
KREMEN1	1589.04	2297.112	584.7121	1178.002	176.29	211.9809	117.5155
PCDHGC5	123.0361	83.06663	180.6374	172.8532	6.188486	13.55698	17.84386
CDH13	3797.71	4029.024	1761.97	2025.046	53.61891	79.34874	40.25633
WNT5B	5736.458	5881.559	4439.834	4165.487	39.47173	46.66308	37.72429

11. In Select Setup Language window, select English, and Click OK.
12. Click Next in the Information window.
13. Click Next to install the R in the folder, C:\Program Files\R\R-3.6.1.
14. Select the component you want to install in the Select Components window. Deselect 32-bit Files and keep the others because we will use RStudio to generate heat map and RStudio 1.2 requires R and works exclusively with the 64-bit version of R. You can install all components though.
15. In the Startup Options window, click Next using the defaults option.
16. Click Next with Start Menu folder as R.
17. In the Select Additional Tasks window, check Create a desktop shortcut, and click Next.
18. Click Finish in the next window.
19. Now, you will see an icon for R on your computer desktop.
20. Double click the R icon to open it and you can start to practice as instructed below.

2.5 Downloading RStudio

RStudio requires R to function. You have to download and install R before you can work on RStuidio. However, with R installed on the same computer you do not need to open R to use RStudio.

1. Go to www.rstudio.com.
2. Click the "Download RStudio" button.
3. Choose the free download version and Click DOWNLOAD.
4. In Installers, click "RStudio 1.2.1335—Windows 7+(64-bit)", or the latest version at the time of your downloading.
5. Click Save in the pop-out dialog window.
6. Click "Open folder" in the dialog window.
7. In the new window find out the downloaded file, RStudio—1.2.1335, double click this file name.
8. Click Yes in the new dialog window to install.
9. Click Next in the RStudio Setup window.
10. Click Next in Choose Install Location window.
11. Click Install in the Choose Start Menu Folder Windows.
12. Click Finish in the final window to complete the installation.
13. Now, open RStudio and practice the R exercises provided below.

2.6 Know R by Hands-on Practice Before Generating Your Own Heat Maps

You may be able to make heat maps following instructions in the next subheading, i.e., Sect. 2.7 without practicing R commands provided in this section. However, it is a good idea to familiarize yourself with some basic commands, R operators, concepts, and functions before you generate heat maps if you do not have any prior experience with R. Readers can gain more R experience by practicing my recent R tutorial [10].

R is a programming platform and uses commands. When you open R, there is a prompt sign > in the console window. This is where you type your commands to conduct data calculation and manipulation. We will briefly practice some basic R commands and functions so that we can better understand the scripts used in generating RNA-seq heat maps. All executing R commands and scripts will be indicated by blue text. Notes are given after a pound sign # following the corresponding R commands or scripts.

I encourage the audiences to read this tutorial in front of your computer and do what the tutorial instructs you to. Please note that you have to hit the Enter key to execute an R function or script, which will not be indicated in the following text. For clarity, the output of your executions will not be included in the protocol and you will see those in your R pane. R uses a lot of functions to execute various tasks. An R function is a special R object, which consists of a set of codes to execute a specific task/calculation. Let us start with the *date()* function on the first day of your R and *pheatmap* experience. Type *date()* after the R prompt sign >, which will be omitted for all the commands and scripts in this tutorial in order to save space, to find out today's date and the current time.

```
date()
```

An R function includes its arguments in parentheses. Parentheses are needed even though no arguments are included (or default arguments). Now, type:

```
date    # When parentheses of a function is missing only the contents of the
        function are printed.
```

Type *version* to see which version of R you are using:

```
version    # the version function is an exception that it does not need parentheses.
```

To see what is the working directory the R is using, type:

```
getwd()
```

One very useful command is the help function: *help()*. R has an inbuilt help facility we can use to find information and usage of an R function. For example, if you want to know the function of *date* you type the following after the prompt sign >:

```
help(date)
```

A new window will pop out. You will see detailed information about the function in question. The help page will describe the function in several sections which may include Description, Usage, Arguments, Details, Reference, See Also, Examples, Value, and Note. In RStudio (see below), this information will appear in the Files/Plots/Packages/Help/Viewer window. You can use the *help ()* function for the commands used in this tutorial. For some R commands, you may have to enclose the command with quotation marks in the *help()* function:

```
help("+")	# An R help document for arithmetic operators will appear in the
		Help window. It returns a warning if + is not enclosed with
		quotation marks. Other arithmetic operators, the extract
		operator $, the assignment operators (=, <-, or ->), and the colon
		sequence operator: (see blow) also require quotation marks in the
		help() function.
```

R can execute mathematic calculations using the intuitive commends: +, -, *, /, and ^ for raising to a power (arithmetic operators); log, log10, log2, and exp (functions for logarithms and exponentials); sin, cos, and tan (trigonometric functions); sqrt, abs (miscellaneous mathematical functions), and others. Try the following calculations:

```
2 + 4
9 - 2
2*9
9/3
3^2
log2(2)
log2(0.5)
```

The results of the above operations (calculations) are printed out immediately on the screen of R console. Alternatively, you can store each result in an object. Try the following:

```
x = 2 + 4
y = 9 - 2
z = 2*9
h = 9/3
i = 3^2
g = log2(2)
f = log2(0.5)
```

The results of the above calculations are not printed out on the screen, but stored in the active memory as an R object with a given name for each calculation. Each of the above object names becomes an R command now. You can type any of the object names to find out their contents. This is the simplest command in R.

```
x        # Typing an object name will return its contents.
y
z
h
i
g
f        # Typing an object name will return its contents.
```

The above commands of object names are an implicit use of the function *print()*. Try:

```
print(x)     # You will see the same result as you type x.
print(y)     # You will see the same result as you type y.
```

An R object is temporarily stored in the active memory, and its content can be replaced by a new assignment. This operation affects the contents of the objects in the active memory, but not any object stored in your hard drive. Try:

```
x              # Its current content, 6, is printed.
x = 4          # x is assigned a new value, 4.
x              # The new content of x object, 4, is printed.
x = 2*5 - 7    # x is assigned a new value again.
x              # The current value of x, 3, is printed out.
```

Recall the previously used R commands using the up and down arrow keys on the keyboard. Please try to press the up arrow 7 times, and then the down arrow key to see what happens. It is useful because you may want to re-execute or modify the calculations you previously entered.

```
x = 2 + 3*4
```

Press the up arrow key and you will see "x = 2 + 3*4" appears on your screen. Now you can edit the existing one to get a different one, for example, change x = 2 + 3*4 to x = 2 + 3/3 by deleting *4 and then adding /3. Then, hit the Enter key to see what will happen.

Instead of =, R commonly uses <- as an assignment operator. Try these:

```
x <- 10*2       # This is equivalent to x = 10*2.
x
y <- log2(2)    # This is equivalent to y = log2(2).
y
```

R object names are case sensitive. Uppercase and lowercase of the same letter are different objects. Try:

```
A <- 4 + 6
a <- 4 + 2
A
a
```

Generate a numeric vector using the combine function *c()* and the colon sequence operator :. Type:

```
c(1:10)
```

An object can be a vector. You can generate the same vector above and store it in a vector object:

```
v <- c(1:10)     # This generates a numeric vector containing 1, 2, 3, 4, 5, 6, 7,
                   8, 9, 10, and store it in v.
v                # print the vector of v on your screen.
```

We can log2-transform every element of the entire numeric vector.

```
log2(v)    # You will see the new vector containing the log2 values of the
             elements of the vector v.
```

We can repeat an element in a vector using the *rep()* function.

```
rep(2, 5)       # You will see that 2 is repeated by 5 times.
```

The above vector is numeric. We can generate a character vector using the combine function *c*.

```
c("female", "male")     # Character elements should be enclosed in quotation marks.
```

The above character vector can be stored in a vector object:

```
v1 <- c("female", "male")
v1
```

Elements of a character vector can be repeated as well using the *rep()* function.

```
v2 <- rep("male", 5)
```

We can use multiple functions to generate a vector, e.g., *c()* and *rep()* function.

```
v3 <- c(rep("female", 5), rep( "male", 5))
v3
```

The major function of R is data analysis and manipulation. In R, there are many built-in data sets. You can find out what built-in data sets are included in the current version of R using the *data* function:

```
data()
```

An R-data set window will pop out containing a list of inbuilt data sets in R. This list of data sets will appear in the Source window if you practice in RStudio (see below).

You can print any available data set in R console by typing the data set name after the prompt sign. For example,

```
mtcars        # This matrix-name command is equivalent to the print(mtcars) command.
```

Use *summary()* and other functions to check the details of a data set.

summary(mtcars)	# The generic summary function returns a summary of the data set.
head(mtcars)	# The *head()* function returns the first several rows of the table so that a small table can give you an idea about the table. The numbers of rows can be defined. Try:
head(mtcars, n = 5)	# Print the first 5 rows of the data frame.
head(mtcars, n = 3)	# Print the first 3 rows of the data frame.
min(mtcars)	# The minimum function *min()* returns the minimum value in the data frame. This works only for numeric matrix or vector.
max(mtcars)	# The maximum function *max()* returns the maximum value in the data frame. This works only for numeric matrix or vector.
ncol(mtcars)	# The numbers-of-columns function *ncol()* returns the number of columns in the data frame studied.
nrow(mtcars)	# The numbers-of-rows function *nrow()* returns the number of rows in the data frame studied.
dim(mtcars)	# The *dim()*function returns the dimension of the data frame, i.e. number of rows and columns.

We have seen numeric objects and vector objects above. An R object can be a matrix, a data frame, or a function. The mtcars is an inbuilt matrix object in R. You can take rows, or columns from a table/matrix using the index function [ ], and assign the new matrix a different matrix object name.

mtcars1 <- mtcars[, 1:4]	# This script takes the first four columns of the mtcars matix and assigns the new matrix to a new matrix object name mtcars1.
mtcars1	# It prints the new matrix mtcars1 on screen.
mtcars2 <- mtcars[1:5 ,]	# This script takes rows 1 to 5 of mtcars and assigns the new data set a matrix object name of mtcars2.
mtcars2	# It prints out the matrix of mtcars2 on screen.
mtcars3 <- log2(mtcars)	# This script transforms the entire mtcars data set using log2, and assigns the log2-transformed data a new object name mtcars3. The data set to be transformed by log2 should be numeric.
mtcars3	# This command is equivalent to print(mtcars3).

We can print specific column of the matrix using the extract operator $.

```
mtcars$mpg      # This script prints the mpg column of mtcars on screen. In
                Rstudio, when you type mtcars$, all column names will pop out
                for you to select.
```

We may want to sort a specific column. To sort, we use a combination of the *order* function, the index operator [ ], and the extract operator $:

```
mtcars4 <- mtcars[order(mtcars$mpg) ,]
```

To examine the sorted matrix, print the new matrix:

```
mtcars4
```

The above script sorts the column of *mpg* from low to high. If you want to sort from high to low, the argument "decreasing = TRUE" should be included.

```
mtcars5 <- mtcars[order(mtcars$mpg, decreasing = TRUE) ,]
```

To check the sorted results, type:

```
mtcars5
```

Give new names to row or columns using the *rownames()*, *paste ()*, and *sep* functions, and the colon sequence operator ":":

```
rownames(mtcars) = paste("cars", 1:32, sep = "")

mtcars          # It returns the table on your screen, and you will see that the row
                names have been changed  to a new set of names ranging from
                cars1 to cars32.
```

You can remove the objects from the active memory using the function *rm()*.

```
rm(mtcars)

mtcars          # You will see a table, but the row names become the original
                names because this is the inbuilt mtcars table, and you just
                removed the mtcars table you have assigned new row names.

mtcars1         # You will see the table of mtcars1 though because it is a
                different object although it is derived from mtcars.

rm(mtcars1)

mtcars1         # You will now see an error message: object 'mtcars1' not
                found. This is because mtcars1 was just removed from the
                active memory using the rm()function and mtcars1 is not
                an inbuilt data set.
```

To quit R, type:

```
q("yes")        # R will   ask if you want to "save workspace image". Click No.
                You can also quit your R session by clicking the "Quit session"
                in the pulldown menu under the "File" menu. A "Quit R
                Session" window pops out. Click "Save" or "Don't save",
                and R is quitted.
```

2.7 Generate Heat Maps in RStudio

The *pheatmap* package is an external one, which is not a part of the standard R installation, and it has to be installed before we can use it to generate heat maps. We use the *install.packages()* function to install it. Type:

```
install.packages("pheatmap")       # After installation, please check the libraries
                                     window to make sure pheatmap is in the list.
```

The *pheatmap* package is widely used to prepare heat maps for RNA-seq data. *pheatmap* is simple, yet it has a lot of arguments, which allow users to control the graphical parameters. Readers are encouraged to have a copy of the package documentation [6] in their hands when practicing this section. The document gives description about each argument of the *pheatmap ()* function.

To find out which working directory RStudio is currently directed to, type:

```
getwd()
```

You may see a return like this: *[1] "C:/Users/yourname/ Documents"*.

Now, direct the working directory to your heat map folder:

```
setwd("C:/Users/yourname/my1stpHeatmap")
```

You have to make the above folder on your computer beforehand. Please do not forget the "C:/". If this is missing, you will have a message of "*Error in setwd("Users/yourname/my1stpHeatmap")*". For folder name, it is not case sensitive. You can direct R to the same working directory by specifying setwd("c:/users/yourname/my1stpheatmap"). Please note the use of the lowercase letters of "c", "u", and 'h" here. These make no difference because they are just parts of folder names.

To confirm the change of working directory, type

```
getwd()     # You will see, "C:/Users/yourname/my1stpHeatmap" on the screen.
```

Now, download the RNA-seq read-count file RNAseqHeatmap_wnt.csv from the MiMB website into your folder, my1stpHeatmap. This table is a data frame in R terminology and contains both numeric and character elements. The hyperlink (https://doi.org/10.1007/978-1-0716-1084-8_17) for downloading the demo RNA-seq data table: The demo table is also available from the author upon request. If it is difficult to download the original RNA-seq data RNAseqHeatmap_wnt.csv, you can make a short table in Excel using the data in Table 1, and save the data in csv format. In this case, you can skip the steps below, i.e., steps for extracting the desired columns, or dropping the unwanted columns.

To read your RNA-seq read-count file from the computer folder into R, use the *read.csv()* function:

```
read.csv("RNAseqHeatmap_wnt.csv")
```

The above function reads the file and print the table in RStudio. To save the table as a data frame in R (i.e., the computer active memory), we assign it a data frame object name, my1stHeatmap.

```
my1stHeatmap  <- read.csv("RNAseqHeatmap_wnt.csv")
```

You will see your data frame name in the Environment window. Click on the data frame name my1stHeatmap on the Environment window, and your data will appear in the Source window. Please note that the extension of .csv is required in the file name RNA-seqHeatmap_wnt.csv for *read.csv* to function. If it is missing, you will see an error-warning message after running the *read.csv* function. When you will use your own RNA-seq data to generate heat maps, it is important to save your .xlsx file as a *csv* file.

From the source window, you see the rows of the uploaded table use numbers as row names. We will make the gene names as row names so that the row names of your heat maps will be the informative gene symbols. To set the gene symbols as their corresponding row names, we use the *rownames()* function and the extract operator $:

```
rownames(my1stHeatmap) <- my1stHeatmap$external_gene.x
```

Now, you see that the row names have been replaced with the corresponding gene names but the column of gene names, i.e., "external_gene.x", still remain in the table. Please note that the new name of the data frame *my1stHeatmap* includes intentionally an uppercase H to demonstrate that R object name is case sensitive. If you use my1stheatmap here in which a lower case h is used, you will find an error warning, "*Error in rownames(my1stheatmap) <- my1stheatmap$external_gene.x: object 'my1stheatmap' not found*". Please note that your column header for the gene identities may be different from the one used in this template script. If this is the case, please use your column header for the gene symbol column after the extract operator $ rather than external_gene.x used here.

The table (data frame in R terminology) contains many unwanted columns including the megadata. The following script will make a new matrix by taking the columns needed only for generation of heat maps, using the index operator [ ].

```
my1stHeatmap1 <- my1stHeatmap[, 10:16]    # This script takes columns 10 to
                                          16 from the data frame
                                          my1stHeatmap to make a matrix
                                          table of 7 columns. The comma
                                          here indicates that all rows will
                                          be retained, and the colon sequence
                                          operator means from column 10 to
                                          16 will be extracted.
```

Alternatively, to make the working table for generation of heat maps, we can drop the unwanted columns from the original table using the following script. This script uses the combine function *c()*, the index operator [ ], and the colon sequence operator :.

```
my1stHeatmap1 <- my1stHeatmap[,-c(1:9, 17, 18)]   # This script drops columns
                                                    1 to 9, as well as columns
                                                    17 and 18, leaving columns
                                                    10 to 16 to make a new
                                                    table of my1stHeatmap1.
```

If you have difficulty in downloading the original RNA-Seq table RNAseqHeatmap_wnt.csv, you can make a csv table by yourself using data in Table 1 in this chapter. Using this short table, you can skip the steps above for extracting columns 10 to 16 because the short table contains the required columns only.

Alternatively, you can delete the unwanted columns in Excel before you transfer the table from Excel to R.

Click on the new data set name my1stHeatmap1 in the Environment window and display the new data in the Source window. Please note that the new table includes only 7 columns containing the normalized read counts for each repeats of the two cell types. Also, we keep the original table by assigning a new name for the new data set. The comma "," inside the index operator [,] indicates that all rows will be retained in the new table.

Load *pheatmap* library. This can be done simply by clicking the box next to the *pheatmap* package in the Packages window, or by executing the *library()* function.

```
library(pheatmap)     # You will see, after running this function, that the box
                        next to the package name pheatmap become checked.
```

Generating the first heat map using the default arguments of the *pheatmap()* function.

```
pheatmap(my1stHeatmap1)
```

You will see your very first heat map in the Plots window (Fig. 1). However, the differentiation power is very low. The colors have to cover read counts from 0 to 7,659. For many genes, the differences between the embryonic stem cells (ESCs) and fibroblasts are not displayed well, for example, *FZD9*, *PCDHGC5*, *PCDHA11*, and *CTNN2*. But, we know there are at least two fold of differences for all the 16 genes between the two types of cells.

To find out the range of the read counts in the entire table, we use the *summary()* function.

```
summary(my1stHeatmap1)
```

You will see the data profile for each column on the Console window after running the above function. They are from 0 to 7,659 for the entire data set.

To solve this problem, we can transform the data using the log2 function. Make sure you assign a new name for your log2-transformed data set.

```
heatmap_log2 <- log2(my1stHeatmap1)
```

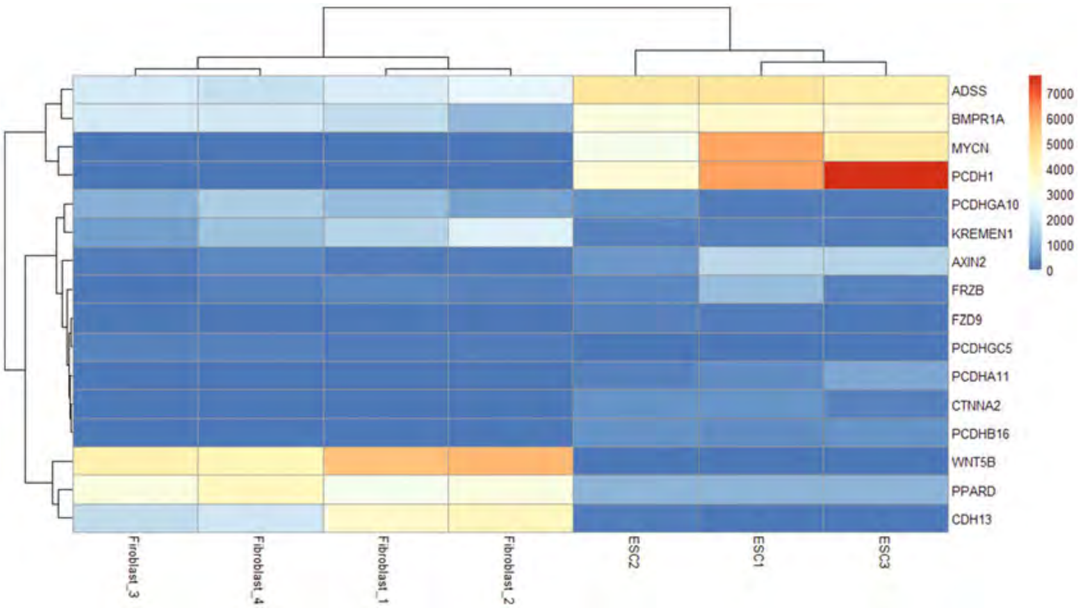


Fig. 1 A heat map prepared using the *pheatmap()* default arguments from the original normalized read counts without transformation and scaling

Run *pheatmap* again for the log2-transformed data.

```
pheatmap(heatmap_log2)
```

You will find an error message: "*Error in hclust(d, method = method): NA/NaN/Inf in foreign function call (arg 10)*". This is because log2 transformation turns the zeros to "-Inf". This can be seen using the *summary()* function.

```
summary(heatmap_log2)
```

To find the lowest log2 read count, we can sort each column using the *order()* function in combination with the index operator [ ], and the extract operator $.

```
heatmap_log2[order(heatmap_log2$Fibroblast_1) ,]
heatmap_log2[order(heatmap_log2$Fibroblast_2) ,]
heatmap_log2[order(heatmap_log2$Fibroblast_3) ,]
heatmap_log2[order(heatmap_log2$Fibroblast_4) ,]
```

Based on those sorting (Scripts for sorting for the three ESC columns are not given above), we found that the smallest transformed read count is 0.7688. Therefore, for practical purpose we replace each "-Inf" with a numeric element "-1" using index operator [ ] and the relational operator ==.

```
heatmap_log2[heatmap_log2 == -Inf] <- -1
```

Examine the table by scrolling around the table and using the *summary()* function.

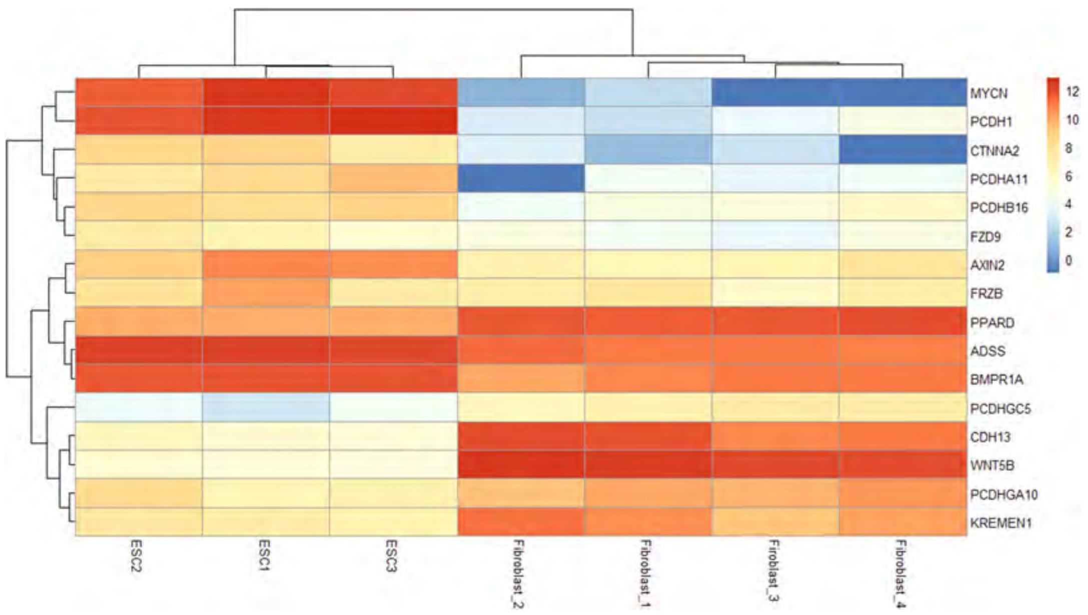


Fig. 2 A heat map prepared with the log2-transformed normalized read counts

Run the *pheatmap()* function again for the log2-transformed-data.

```
pheatmap(heatmap_log2)
```

Now, you see a heat map with improved resolution using the same default color scheme (Fig. 2). For examples, the genes *FZD9*, *PCDHGC5*, *PCDHA11*, and *CTNN2* are now resolved well between fibroblasts and ESCs. The scale of the color legend is now ranging from 0 to 12. But, some genes are still not well resolved between these two types of cells, for example, *ADSS* and *BMPR1A*.

To further improve resolution, we can use the *scale* argument of the *pheatmap* function as follows:

```
pheatmap(heatmap_log2, scale = “row” )
```

Now, as you can see that every gene is resolved well between the two types of cells (Fig. 3). Please note that we use a comma to separate multiple arguments in an R function.

To save the heat map image on computer, you can just simply use the Export menu in the Plots window by clicking the pull-down menu and selecting “Save as Image”. A new dialog window will pop out. To finish the saving process, you can do these steps: Select the format of image file (png, jpeg, tiff, bmp, svg, or eps), give a file name, and change the size of image if necessary, then click the “Save” button. You can save your heat maps within any folder by choosing folder from the Directory button.

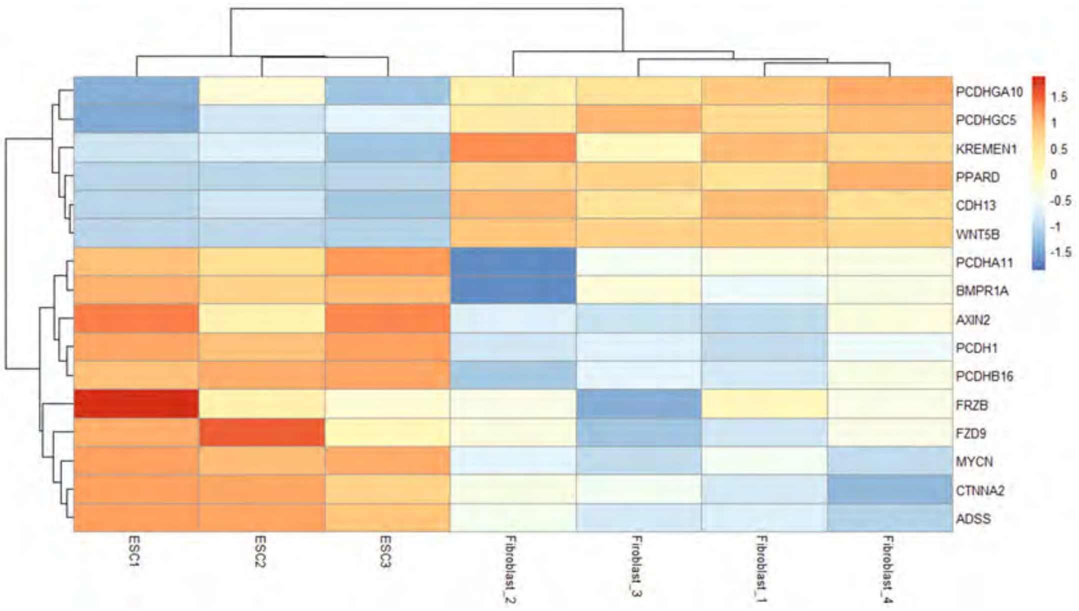


Fig. 3 A heat map generated with log2-transformed read counts and scaling of rows

Another way to save the heat map image is to include the *filename* argument:

```
pheatmap(heatmap_log2, scale = “row”, filename = “myRNAseqheatmap.pdf”  )
```

This will save your heat map image in the current folder of the working directory. The format of the image is defined by the extension of a file name, and it can be png, pdf, tiff, bmp, or jpeg formats.

Or, you can save the heat map by using an R graphics devices *jpeg()*, *bmp()*, *png()*, or *tiff()* at three steps:

```
jpeg(file = “myjpegheatmap.jpeg”)           # Open an R graphics device.
pheatmap(heatmap_log2, scale = “row”)       # Create a heat map using the
                                            pheatmap function.
dev.off()                                   # Close the graphics device and return
                                            graphics device to the computer screen,
                                            i.e., the Plots window. This is
                                            important. Without running the
                                            dev.off() function, your image may not
                                            be saved correctly.
```

The heat map is now saved in the working directory. Go to the working directory to find out your newly generated heat map file. Other graphics devices work the same way as *jpeg()*.

After you have used a graphics device to save your image on computer, you may not see your image to show up in the Plots window when you run *pheatmap()* again for a new heat map. If this happens, you can solve this problem by calling *dev.off()* one more time. You will see "null device", which indicates all other devices are closed and the Plots pane becomes the active graphics device.

2.8 Change Dimensions and Appearance of Heat Maps

The heat map looks great now. For the purpose of presentation or publication, we may need to edit the heat maps to show or not to show the associated information in a desired way. We achieve this by defining additional arguments. In R, the arguments in a specific function can be in any order. In the following scripts, the arguments in discussion will be the last argument(s) in the *pheatmap()* function and is highlighted in bold face.

The shapes of dendrograms are affected by the methods of clustering and methods for calculating clustering distance. The default method for clustering in *pheatmap()* is complete linkage with the value of "*complete*". Other available methods for clustering with *pheatmap()* include "*ward.D*", "*ward.D2*", "*single*", "*complete*", "*average*", "*mcquitty*", "*median*", and "*centroid*". Methods for clustering are based on the calculation of distance. The default distance measure in *pheatmap()* is "euclidean". The *pheatmap()* also supports other distance measure methods for calculating dissimilarity/distance including "correlation", "*maximum*", "*manhattan*", "*canberra*", "*binary*", or "*minkowski*". The distance measure "*correlation*" is widely used in gene expression clustering considering that it focuses on expression behavior (upregulation or downregulation) rather than the levels of expression (Fig. 4). The default Euclidean method tends to cluster the genes with high values together and the genes with low values in separate groups.

```
pheatmap(heatmap_log2, scale = "row", clustering_method = "mcquitty",
clustering_distance_rows = "correlation")   # This script gives a heat map with
                                               different shape of dendrogram and
                                               different arrangement of cells
                                               (Fig. 4).
```

We can show the expression levels for each gene in number using the logical argument of *display_numbers* (Fig. 5).

```
pheatmap(heatmap_log2, scale = "row", display_numbers = TRUE)
```

The size of the heat map may not be optimal. We can include the *cellwidth* argument to adjust the width of the cell to obtain a desired size of the heat map (Fig. 6):

```
pheatmap(heatmap_log2, scale = "row", display_numbers = TRUE, cellwidth = 22 )
```

Now, the cell width is just enough to accommodate the numbers inside, but the cell height may not be optimal. We use *cellheight* argument to make them just fit (Fig. 7).

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 22, display_numbers =
TRUE, cellheight = 20)
```

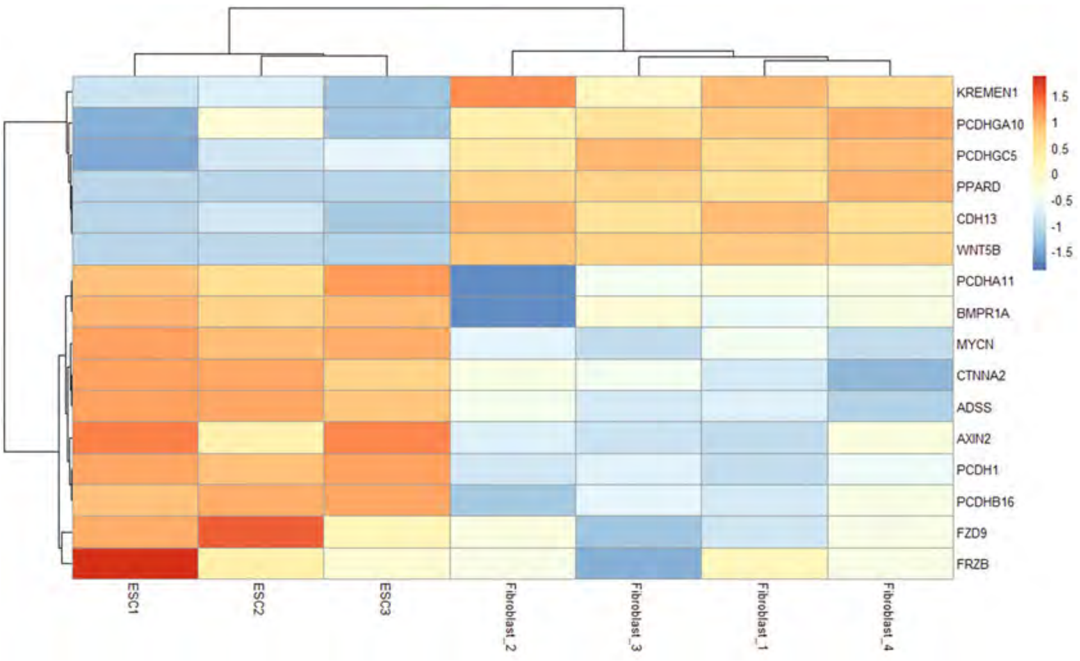
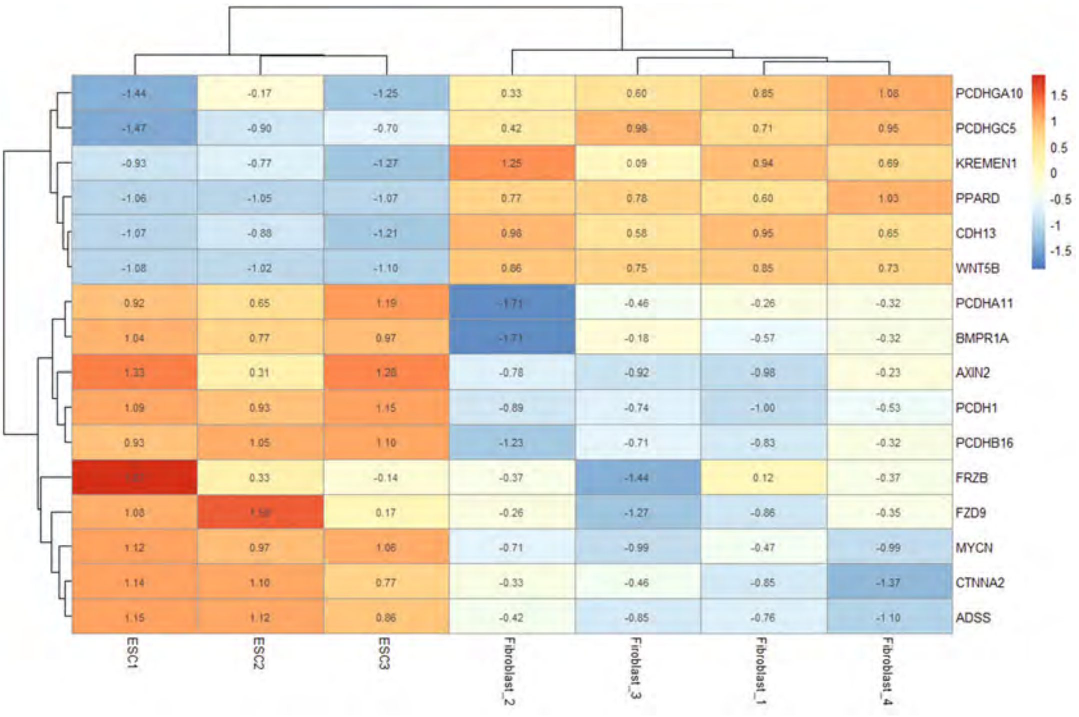


Fig. 4 A heat map with a different clustering method ("*mcquitty*") and distance measure ("*correlation*") from the default setting ("*complete*" and "*Euclidean*", respectively)



	ESC1	ESC2	ESC3	Fibroblast_2	Firoblast_3	Fibroblast_1	Fibroblast_4
PCDHGA10	-1.44	-0.17	-1.25	0.33	0.60	0.85	1.08
PCDHGC5	-1.47	-0.90	-0.70	0.42	0.98	0.71	0.95
KREMEN1	-0.93	-0.77	-1.27	1.25	0.09	0.94	0.69
PPARD	-1.06	-1.05	-1.07	0.77	0.78	0.60	1.03
CDH13	-1.07	-0.88	-1.21	0.98	0.58	0.95	0.65
WNT5B	-1.08	-1.02	-1.10	0.86	0.75	0.85	0.73
PCDHA11	0.92	0.65	1.19	-1.71	-0.46	-0.26	-0.32
BMPR1A	1.04	0.77	0.97	-1.71	-0.18	-0.57	-0.32
AXIN2	1.33	0.31	1.28	-0.78	-0.92	-0.98	-0.23
PCDH1	1.09	0.93	1.15	-0.89	-0.74	-1.00	-0.53
PCDHB16	0.93	1.05	1.10	-1.23	-0.71	-0.83	-0.32
FRZB	[illegible]	0.33	-0.14	-0.37	-1.44	0.12	-0.37
FZD9	1.08	[illegible]	0.17	-0.26	-1.27	-0.86	-0.35
MYCN	1.12	0.97	1.06	-0.71	-0.99	-0.47	-0.99
CTNNA2	1.14	1.10	0.77	-0.33	-0.46	-0.85	-1.37
ADSS	1.15	1.12	0.86	-0.42	-0.85	-0.76	-1.10

Fig. 5 The same heat map as in Fig. 4, but with the expression levels shown in each cell

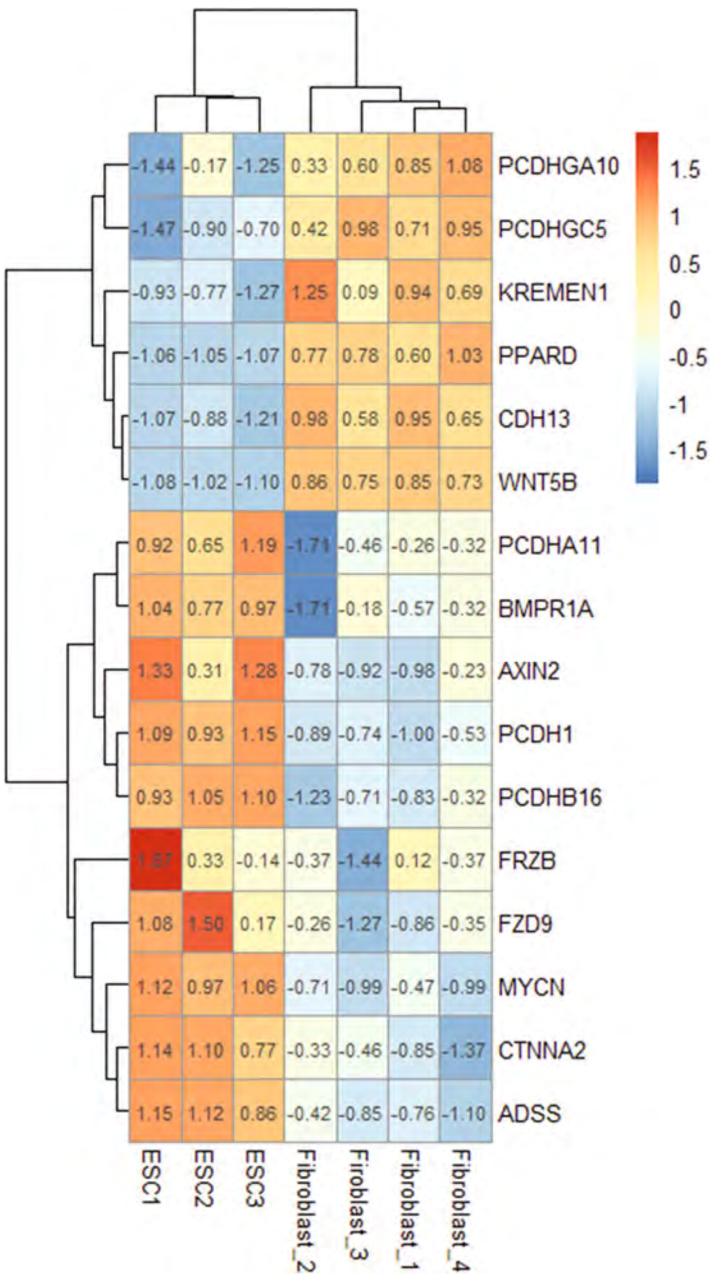


Fig. 6 The same heat map as in Fig. 5 but with the cell width modified

We can separate the heat map into two panels based on clusters representing the two types of cells using the argument of *cutree_cols* (Fig. 8):

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 22, cellheight = 20 , display_numbers = TRUE,
cutree_cols = 2)
```

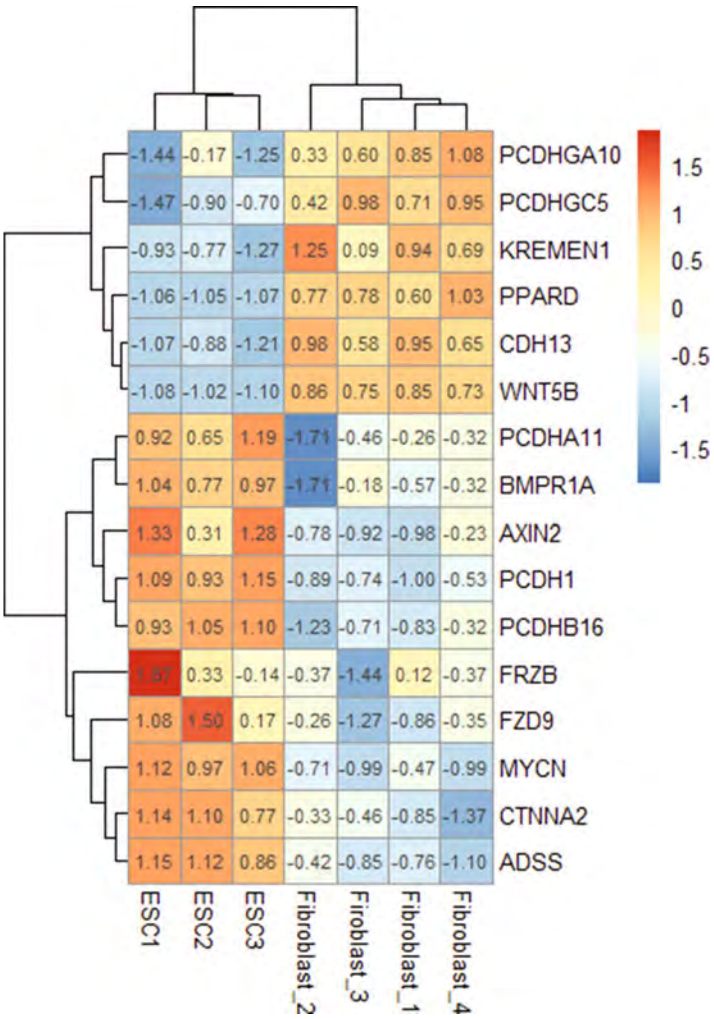


Fig. 7 The same heat map as in Fig. 6 except for additional modification in cell heights

We can include a heat map title using the argument of *main* (Fig. 9):

```
pheatmap(heatmap_log2, scale = “row”, cellwidth = 22, cellheight =20 , display_numbers = TRUE,
cutree_cols = 2, main = “Differential expression of some genes in WNT pathway between human
fibroblasts and ESCs”)
```

The heat map title is too wide compared with that of the heat map image. We can split it into two or three lines using the *paste* function along with the new line operator \n (Fig. 10):

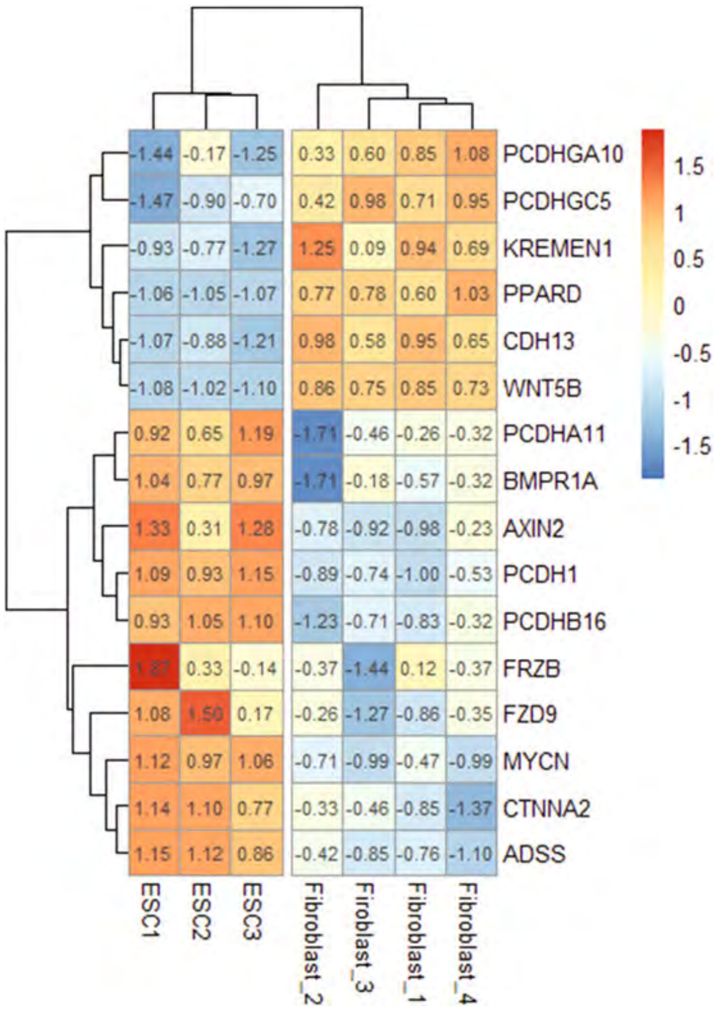
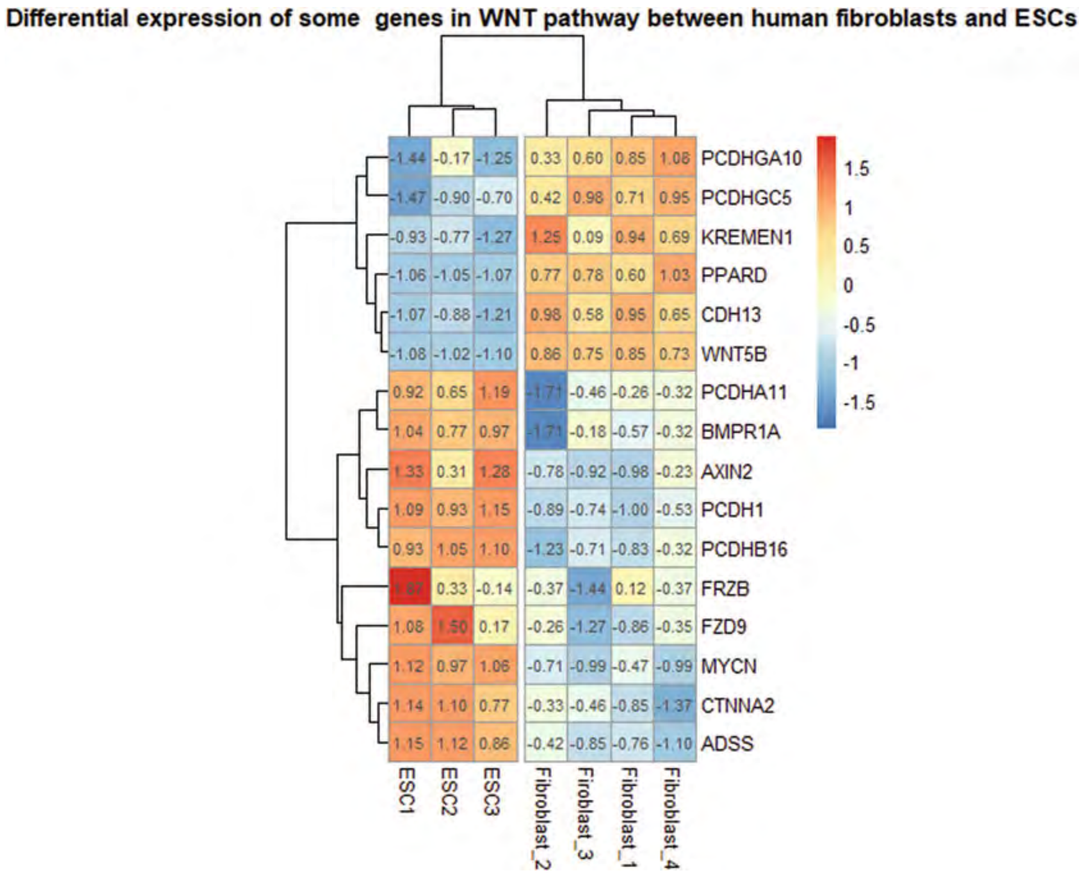


Fig. 8 The same heat map as in Fig. 7 except for a gap introduced to separate panels for the two cell types

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 22, cellheight =20 , display_numbers = TRUE,
cutree_cols = 2, main = paste("Differential expression of", "\nsome genes in WNT pathway",
"\nbetween human fibroblasts and ESCs"))
```

We can change the font sizes for numbers and text in the heat map, using the arguments of *fontsize*, *fontsize_row*, *fontsize_col*, or *fontsize_number* for all, row names, column names, or numbers in the cells, respectively (Fig. 11):



	ESC1	ESC2	ESC3	Fibroblast_2	Firoblast_3	Fibroblast_1	Fibroblast_4
PCDHGA10	-1.44	-0.17	-1.25	0.33	0.60	0.85	1.08
PCDHGC5	-1.47	-0.90	-0.70	0.42	0.98	0.71	0.95
KREMEN1	-0.93	-0.77	-1.27	1.25	0.09	0.94	0.69
PPARD	-1.06	-1.05	-1.07	0.77	0.78	0.60	1.03
CDH13	-1.07	-0.88	-1.21	0.98	0.58	0.95	0.65
WNT5B	-1.08	-1.02	-1.10	0.86	0.75	0.85	0.73
PCDHA11	0.92	0.65	1.19	-1.71	-0.46	-0.26	-0.32
BMPR1A	1.04	0.77	0.97	-1.71	-0.18	-0.57	-0.32
AXIN2	1.33	0.31	1.28	-0.78	-0.92	-0.98	-0.23
PCDH1	1.09	0.93	1.15	-0.89	-0.74	-1.00	-0.53
PCDHB16	0.93	1.05	1.10	-1.23	-0.71	-0.83	-0.32
FRZB	1.87	0.33	-0.14	-0.37	-1.44	0.12	-0.37
FZD9	1.08	1.50	0.17	-0.26	-1.27	-0.86	-0.35
MYCN	1.12	0.97	1.06	-0.71	-0.99	-0.47	-0.99
CTNNA2	1.14	1.10	0.77	-0.33	-0.46	-0.85	-1.37
ADSS	1.15	1.12	0.86	-0.42	-0.85	-0.76	-1.10

Fig. 9 The heat map in Fig. 8 except that a title is included

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 22, cellheight =20 , display_numbers = TRUE,
cutree_cols = 2, main = paste("Differential expression of", "\nsome genes in WNT pathway",
"\nbetween human fibroblasts and ESCs"), fontsize = 14)
```

The default font size is 10. We found that the new font size for the numbers is too large for the cell size. We can change either the size of cell or the font size of numbers to make them matched. We redefine the cellwidth = 28 (*see* Fig. 12).

We can change the color of the cell border using the *border_color* argument (Fig. 13):

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 28, cellheight = 20 , display_numbers = TRUE,
cutree_cols = 2, fontsize = 14, border_color = "magenta")
```

You can try other colors by replacing magenta in the above script with white, black, grey, pink, yellow, cyan, green, and others.

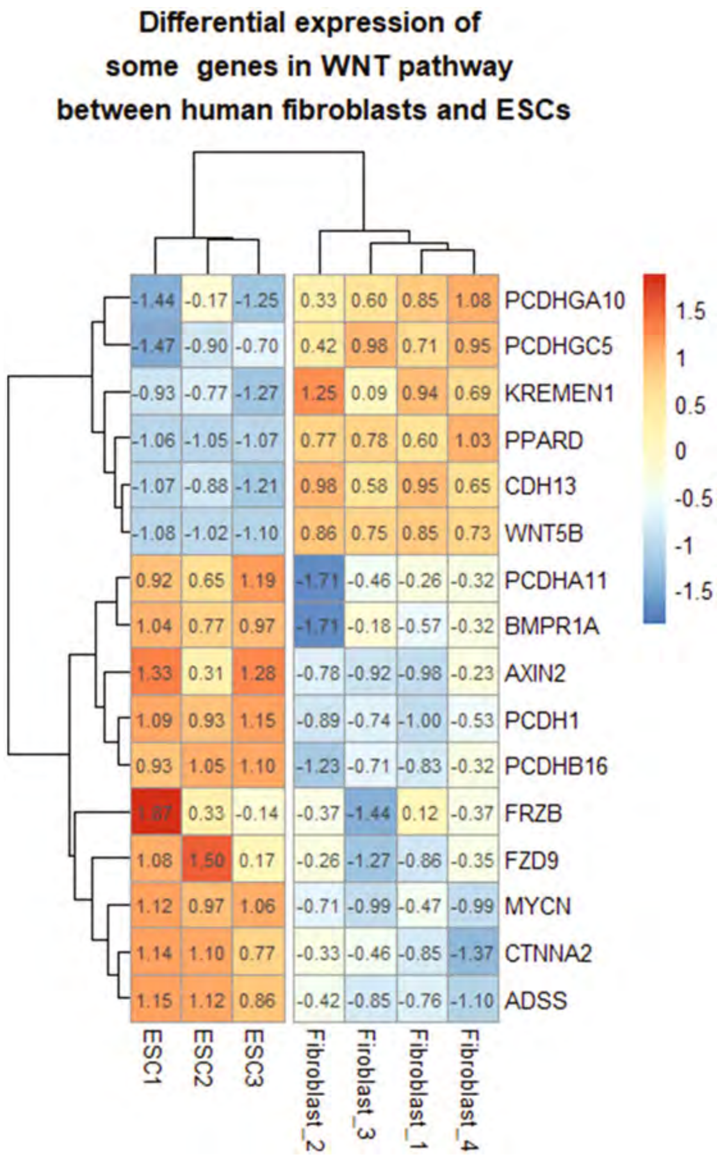


Fig. 10 The heat map in Fig. 9 but with its title split in three lines

We can change the angle of the text for column labels using argument of *angle_col* (Fig. 14) (Please note that only five predefined angles are available, 0, 45, 90, 270, and 315):

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 28, cellheight = 20 , display_numbers = TRUE,
cutree_cols = 2, fontsize = 14, border_color = "magenta", main = paste("Differential expression of",
"\nsome genes in WNT pathway",  "\nbetween human fibroblasts and ESCs"),  angle_col = "45")
```

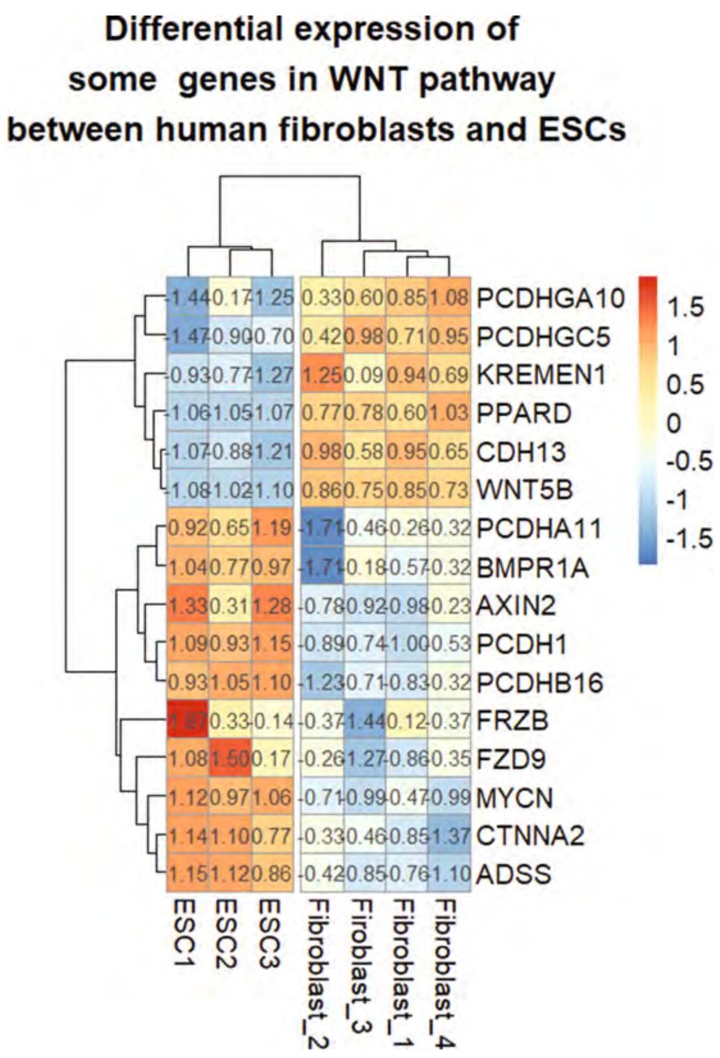


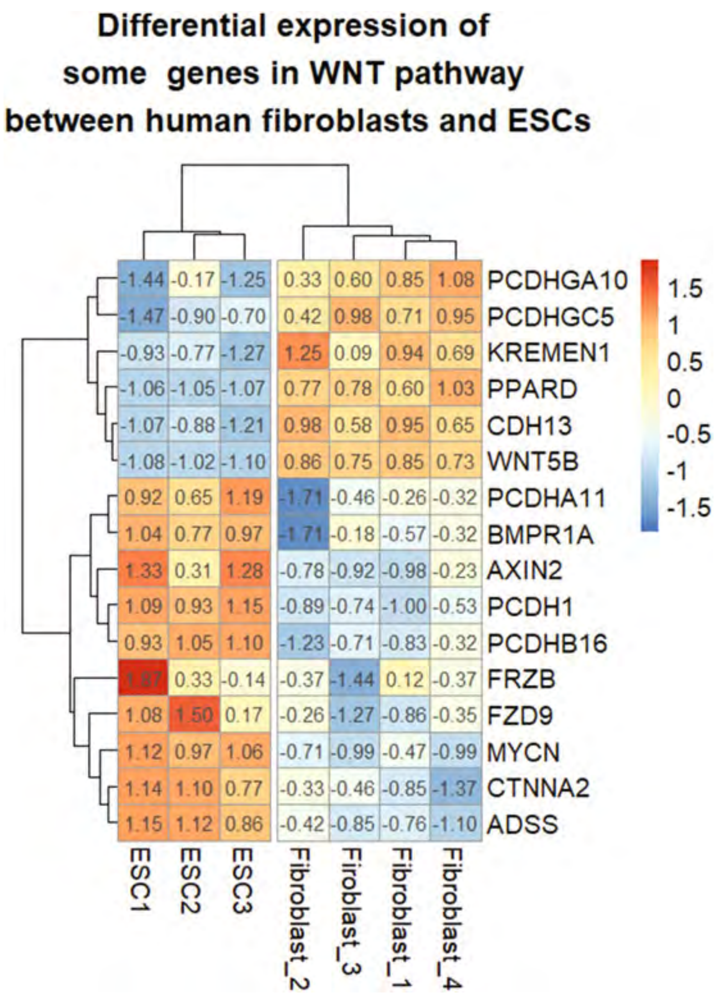
Fig. 11 The same heat map as in Fig. 10 but with a larger font size

We can change the height of the dendrogram using arguments of *treeheight_row*, and/or *treeheight_col* (Fig. 15):

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 28, cellheight = 20 , display_numbers = TRUE,
cutree_cols = 2, fontsize = 14, border_color = "magenta", main = paste("Differential expression of",
"\nsome genes in WNT pathway",  "\nbetween human fibroblasts and ESCs"),  treeheight_row = 200,
treeheight_col = 100)
```

If you do not want the tree to show up in the image, the *treeheight* value(s) can be set to 0 (Fig. 16).

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 28, cellheight = 20 , display_numbers = TRUE,
cutree_cols = 2, fontsize = 14, border_color = "magenta", main = paste("Differential expression of",
"\nsome genes in WNT pathway",  "\nbetween human fibroblasts and ESCs"),  treeheight_row = 0,
treeheight_col = 0)
```



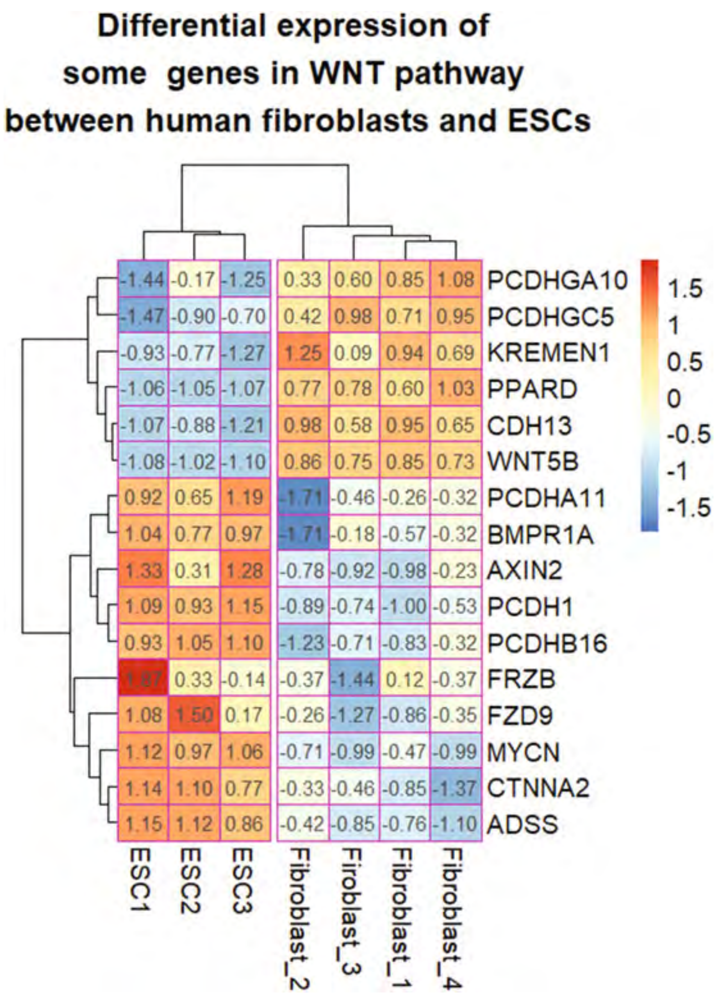
	ESC1	ESC2	ESC3	Fibroblast_2	Firoblast_3	Fibroblast_1	Fibroblast_4
PCDHGA10	-1.44	-0.17	-1.25	0.33	0.60	0.85	1.08
PCDHGC5	-1.47	-0.90	-0.70	0.42	0.98	0.71	0.95
KREMEN1	-0.93	-0.77	-1.27	1.25	0.09	0.94	0.69
PPARD	-1.06	-1.05	-1.07	0.77	0.78	0.60	1.03
CDH13	-1.07	-0.88	-1.21	0.98	0.58	0.95	0.65
WNT5B	-1.08	-1.02	-1.10	0.86	0.75	0.85	0.73
PCDHA11	0.92	0.65	1.19	-1.71	-0.46	-0.26	-0.32
BMPR1A	1.04	0.77	0.97	-1.71	-0.18	-0.57	-0.32
AXIN2	1.33	0.31	1.28	-0.78	-0.92	-0.98	-0.23
PCDH1	1.09	0.93	1.15	-0.89	-0.74	-1.00	-0.53
PCDHB16	0.93	1.05	1.10	-1.23	-0.71	-0.83	-0.32
FRZB	1.87	0.33	-0.14	-0.37	-1.44	0.12	-0.37
FZD9	1.08	1.50	0.17	-0.26	-1.27	-0.86	-0.35
MYCN	1.12	0.97	1.06	-0.71	-0.99	-0.47	-0.99
CTNNA2	1.14	1.10	0.77	-0.33	-0.46	-0.85	-1.37
ADSS	1.15	1.12	0.86	-0.42	-0.85	-0.76	-1.10

Fig. 12 The same heat map as in Fig. 11 but with the cell width increased to accommodate the larger font size

You may just need heat map and will add title, gene names, sample names later in Illustrator to produce a high-quality presentation. To this end, you can remove the *main* argument, and set *show_rownames*, *show_colnames*, and *display_numbers* to F or FALSE (Fig. 17).

```
pheatmap(heatmap_log2, scale = “row”, cellwidth = 28, cellheight = 20 , cutree_cols = 2, fontsize = 14,
border_color = “magenta”, treeheight_row = 0, treeheight_col = 0, display_numbers = FALSE,
show_rownames = F, show_colnames = F)
```

We can give the rows or columns specific names rather than row names and column names in the original matrix using the arguments of *label_col* and/or *label_row*. In this case, we need to set *show_rownames* or *show_colnames* to T or TRUE, or just remove these two arguments in order to display the new names (Fig. 18):



	ESC1	ESC2	ESC3	Fibroblast_2	Firoblast_3	Fibroblast_1	Fibroblast_4
PCDHGA10	-1.44	-0.17	-1.25	0.33	0.60	0.85	1.08
PCDHGC5	-1.47	-0.90	-0.70	0.42	0.98	0.71	0.95
KREMEN1	-0.93	-0.77	-1.27	1.25	0.09	0.94	0.69
PPARD	-1.06	-1.05	-1.07	0.77	0.78	0.60	1.03
CDH13	-1.07	-0.88	-1.21	0.98	0.58	0.95	0.65
WNT5B	-1.08	-1.02	-1.10	0.86	0.75	0.85	0.73
PCDHA11	0.92	0.65	1.19	-1.71	-0.46	-0.26	-0.32
BMPR1A	1.04	0.77	0.97	-1.71	-0.18	-0.57	-0.32
AXIN2	1.33	0.31	1.28	-0.78	-0.92	-0.98	-0.23
PCDH1	1.09	0.93	1.15	-0.89	-0.74	-1.00	-0.53
PCDHB16	0.93	1.05	1.10	-1.23	-0.71	-0.83	-0.32
FRZB	1.87	0.33	-0.14	-0.37	-1.44	0.12	-0.37
FZD9	1.08	1.50	0.17	-0.26	-1.27	-0.86	-0.35
MYCN	1.12	0.97	1.06	-0.71	-0.99	-0.47	-0.99
CTNNA2	1.14	1.10	0.77	-0.33	-0.46	-0.85	-1.37
ADSS	1.15	1.12	0.86	-0.42	-0.85	-0.76	-1.10

Fig. 13 The same heat map as in Fig. 12 but with magenta cell borders

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 28, cellheight = 20 , display_numbers = FALSE,
cutree_cols = 2, fontsize = 14, border_color = "magenta", treeheight_row = 0, treeheight_col = 0,
labels_col = c("F1", "F2", "F3", "F4", "PSC1", "PSC2", "PSC3"))
```

We can use a color palette rather than the default one. For this purpose, we will load the color library RColorBrewer [9] using the *library()* function, or by checking the box next to RColorBrewer in the Packages window.

```
library(RcolorBrewer)
```

To know what color palettes RColorBrewer includes, use the following R command:

```
display.brewer.all()      # The RcolorBrewer palettes appear in Plots window.
```

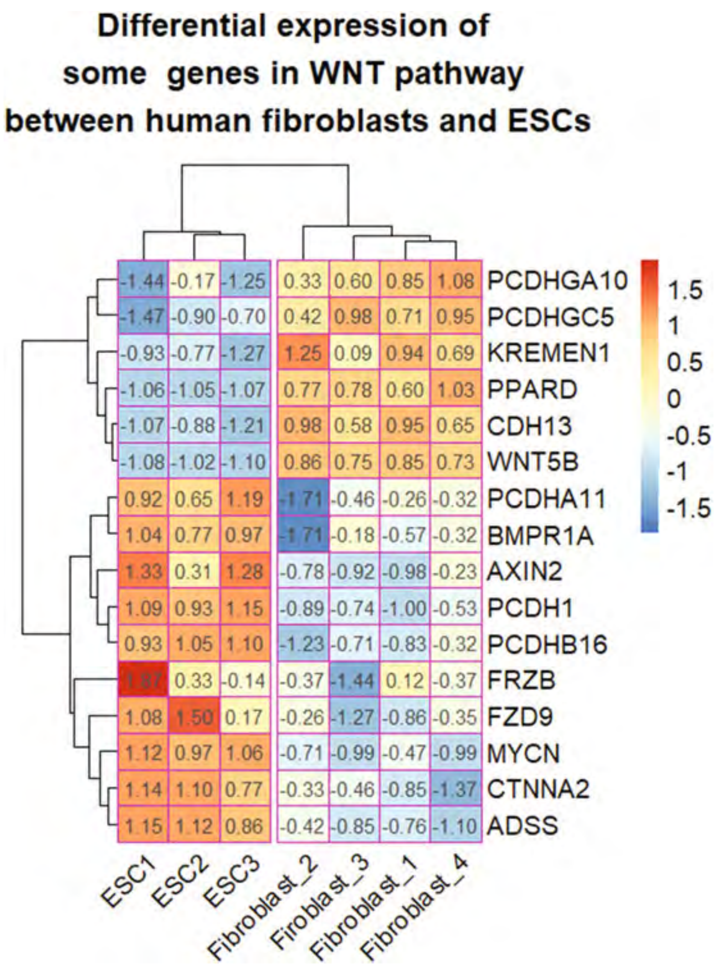


Fig. 14 The same heat map as in Fig. 13 but with its column names tilted

For scientific publications, more and more journals encourage the authors to use color-blind friendly palettes. To find out which color palettes are color-blind friendly, use the following script:

```
display.brewer.all(colorblindFriendly = T)        # All colorblind friendly palettes
                                                  appear in the Plots window.
```

Now, you can test another color-blind diverging palettes PuOr using the following R script (Fig. 19):

```
pheatmap(heatmap_log2, scale = "row", cellwidth = 28, cellheight = 20 ,
fontsize = 14, border_color = "magenta", treeheight_row = 0,
treeheight_col = 0, angle_col = "45", color = brewer.pal(11, "PuOr"))
```

The resolution looks good as well with the new color palette (Fig. 19).

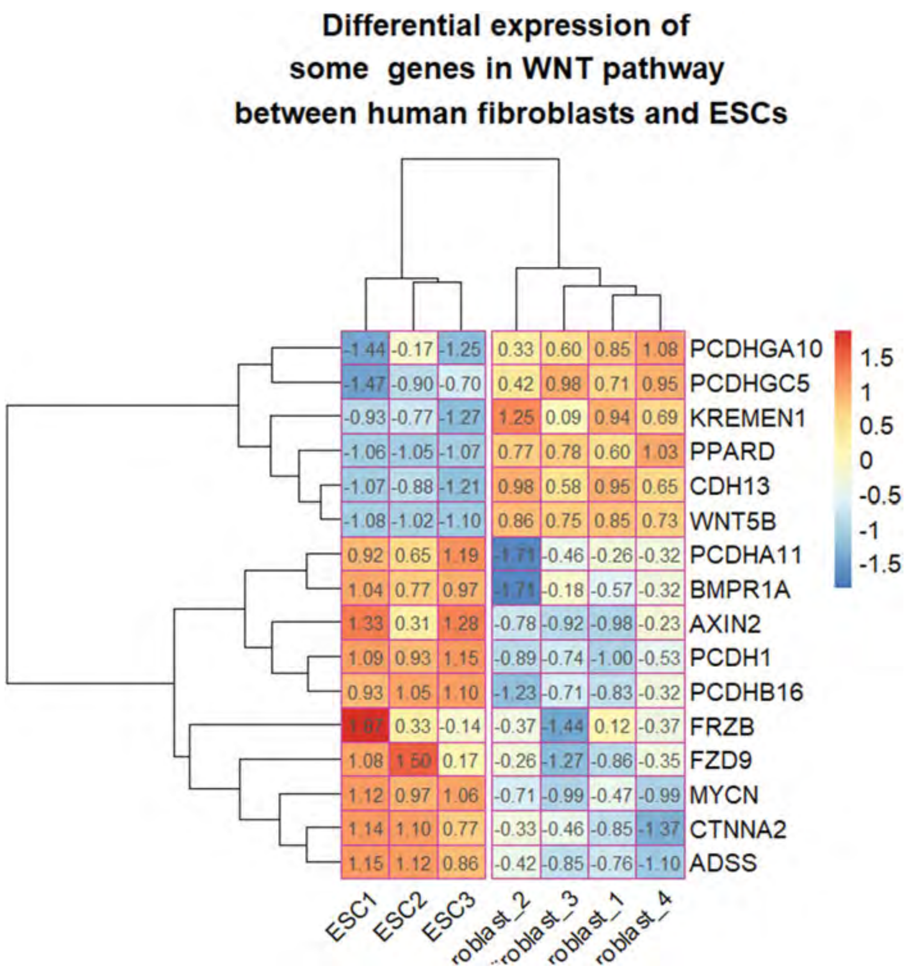


Fig. 15 The same heat map as in Fig. 14 but with increased heights of trees

Try not to scale the rows with the color palette *PuOr* (Fig. 20):

```
pheatmap(heatmap_log2, cellwidth = 28, cellheight = 20 , fontsize = 14, border_color = “magenta”,
treeheight_row = 0, treeheight_col = 0, angle_col = “45”, color = brewer.pal(11, “PuOr”), scale =
“none”)
```

The differentiation power has decreased without scaling (Fig. 20).

Try to use the original normalized read counts for a heat map with the new colors without log2 transformation (Fig. 21).

```
pheatmap(cellwidth = 28, cellheight = 20 , fontsize = 14, border_color = “magenta”, treeheight_row = 0,
treeheight_col = 0, angle_col = “45”, color = brewer.pal(11, “PuOr”), my1stHeatmap1)
```

The differentiation power becomes even worse without log2 transformation and scaling (Fig. 21).

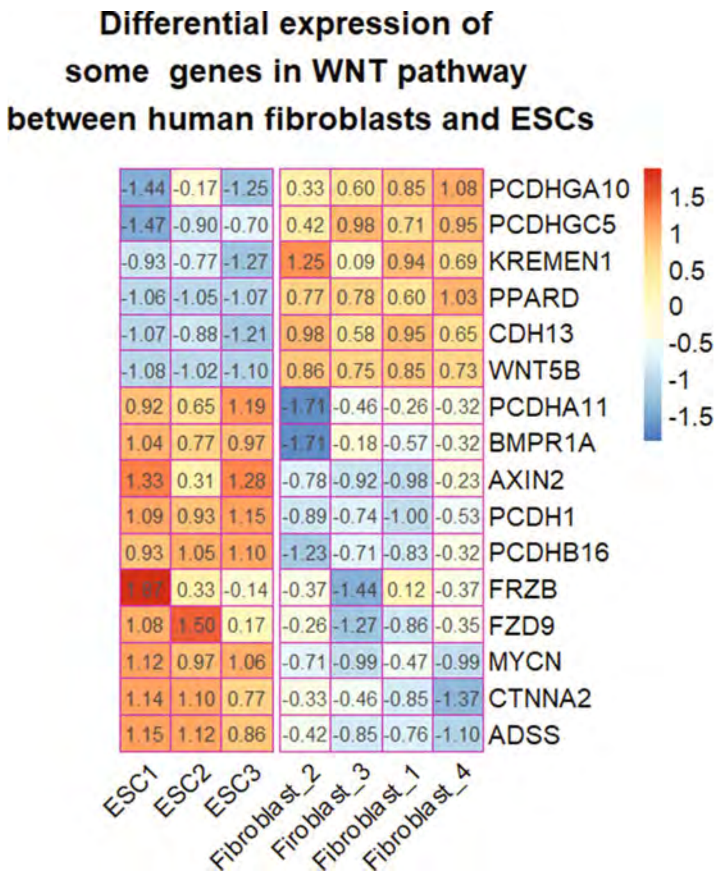


Fig. 16 The same heat map as in Fig. 15 but with its dendrograms removed

2.9 Timing

For a novice without any prior R experience, it takes one day to become competent in generating heat maps using the procedure provided in this protocol. After being proficient, one can generate a heat map from a read count table in less than 30 min.

2.10 Trouble-shooting

Troubleshooting advices can be found in Table 2. For additional R experience and skills, audiences can use my recent tutorial about generating boxplots and violin plots using R [10].

2.11 Anticipated Results

At the end of the morning practice, audiences will download and install R, as well as RStudio on to his/her own computers. The audiences will then become familiar, by hands-on experience, with some basic R concepts, commands, functions, and some scripts including some basic manipulation of tables, which will help the audiences to use the demo table for preparation of RNA-seq data to generate their own heat maps.

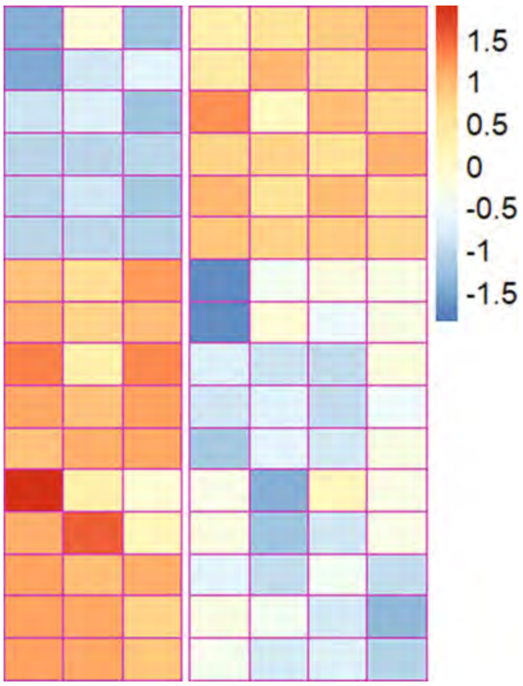


Fig. 17 The same heat map as in Fig. 16 but with all of its labels removed

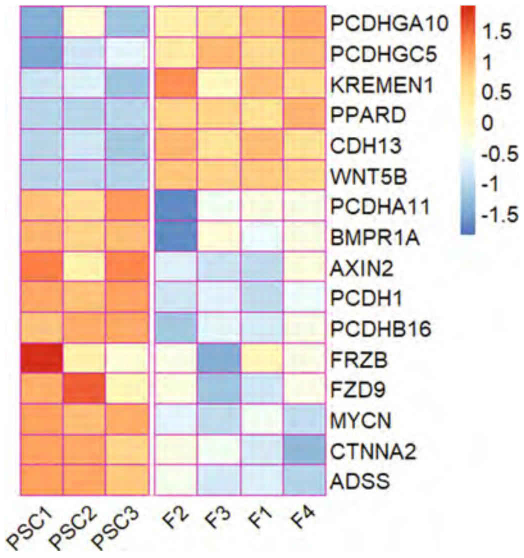


Fig. 18 Heat map with customized column labels

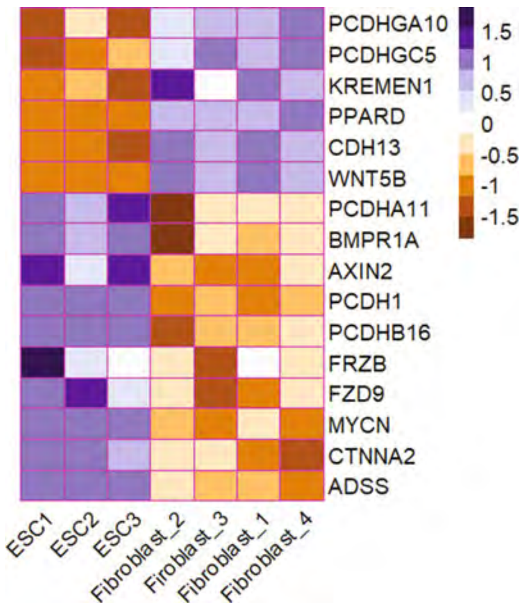


Fig. 19 Heat map for log2-transformed read counts and row scaling, with a color palette (PuOr) different from the default

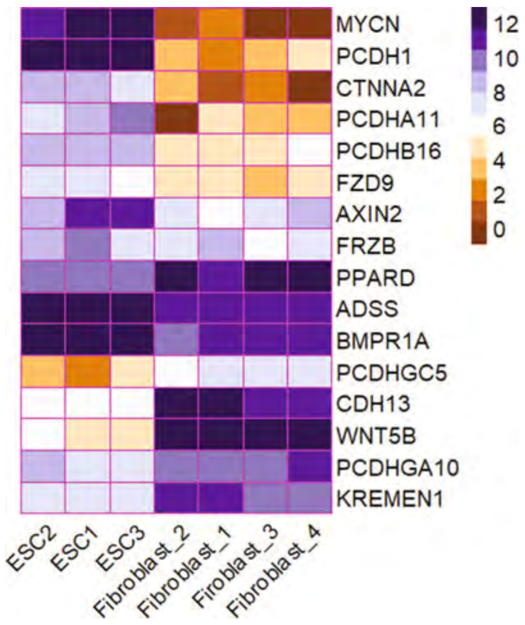


Fig. 20 A low-resolution heat map without scaling. The same heat map as in Fig. 19, but without row scaling. Similar to Fig. 2, but with different colors

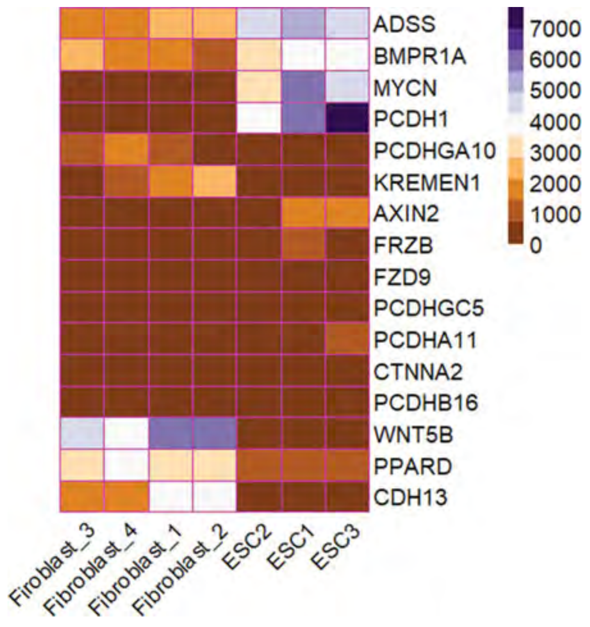


Fig. 21 A heat map with very low differentiation power. The same heat map as in Fig. 19, but with the original read counts and without scaling. Similar to Fig. 1, but with different colors

At the end of the day, audiences will be able to generate heat maps using data frame of RNA-seq read counts. The audiences will also become proficient in manipulating the resulting heat maps for a desired size and appearance.

Acknowledgments

The author's work is supported by NIH (1R01GM127411-01A1) and the American Heart Association (17GRNT33670780). The author appreciates the testing of this tutorial by Drs. Ricardo Cevallos and Jobaida Akther, both of whom were at the level of fresh postdoctoral scientists in biology and medical sciences without any prior experience in R and preparation of heat maps at the time of testing. This tutorial was also tested by the group members of Dr. Ren-he Xu at Macau University including Roma Borkar, Dejin Zheng, Ye Yi, Siyi Fu, Borong Huang, Sen Ye, Guihui Qin, Haibo Peng, Changkun Yang, Daryl Yichi Zhang, and Philip Naderev Lagniton.

Table 2
Troubleshooting table

Problems	Possible causes	Potential solutions
Cannot download R or Rstudio	Different computer systems and browsers used.	Downloading and installation is generally easy. Just follow instruction on your computer rather than steps here. If problems persist, ask your IT personnel for help.
RStudio not open after installation	Computer system is too old.	Download and install earlier version of RStudio that can run on your computer.
R does not read my read count table	The format may be wrong. The file extension is missing.	Save the table as .csv format. Check if the file extension .csv is missing.
Could not find the function *pheatmap*	*pheatmap* package is not installed; *pheatmap* is not loaded.	Install *pheatmap*; load *pheatmap*
Heat map fails with warning: NA/NaN/Inf in function call	The read count table contain characters: NA (not available, NaN (not a number), or Inf.	Replace NA, NaN, or Inf with appropriate numeric numbers.
pheatmap does not work with a warning: object “yourobjectname” not found.	Object names in R are case sensitive, and your input may use the wrong cases.	Check if the object name (matrix name) to be heat mapped is correctly spelled. Pay attention to capital letters.
Heat map does not save with a warning: object “yourfilename” not found.	File name is not enclosed in quotation marks.	Enclose your heat map file name with quotation marks.
Heat map does not save with a warning: improper filename	The extension of the file name is missing.	Add file format extension such as . pdf, .png, or .tiff
Heat map does not show in plots Window after saving previous heat maps	R is using other graphics device.	Run *dev.off()* one more time. It will work when you see “null device” after running *dev.off()*.
Heat map fails with a warning: object “true” not found	The logical objects TRUE or FALSE should be capitalized.	Change “true” to “TRUE” or “T”. This is true for “false”.
Heat map fails with a warning: unexpected “=”	When use two-part object such as *display_numbers*, a hyphen is used instead of an underscore.	Change the hyphen sign to an underscore sign.
Heat map fails with a warning: could not find function “brewer.pal”	The RColorBrewer is not loaded.	Load the color package RColorBrewer.
RColorBrewer will not load	Package name is not spelled correctly.	Use capital R and B in RColorBrewer, or just click the box next to RcolorBrewer in the Packages window.

(continued)

Table 2
(continued)

Problems	Possible causes	Potential solutions
Heat map fails when use new color with a warning: not a valid palette name for brewer.pal	The color object is not spelled correctly.	Check the color object name and make sure appropriate uppercase letters are used.
Cannot sort using the *order()* function	The comma is missing.	Add a comma "," within the index operator [ ].
Customized labels for columns are with the wrong samples	The labels in the character vector for the argument *labels_col* is in the wrong order.	Order the labels in the same order as they appear in the matrix columns.

References

1. Shao Z, Yao C, Khodadadi-Jamayran A, Xu W, Townes TM, Crowley MR, Hu K (2016) Reprogramming by de-bookmarking the somatic transcriptional program through targeting of BET bromodomains. Cell Rep 16 (12):3138–3145. https://doi.org/10.1016/j.celrep.2016.08.060
2. Shao Z, Zhang R, Khodadadi-Jamayran A, Chen B, Crowley MR, Festok MA, Crossman DK, Townes TM, Hu K (2016) The acetyllysine reader BRD3R promotes human nuclear reprogramming and regulates mitosis. Nat Commun 7:10869. https://doi.org/10.1038/ncomms10869
3. Kang L, Yao C, Khodadadi-Jamayran A, Xu W, Zhang R, Banerjee NS, Chang CW, Chow LT, Townes T, Hu K (2016) The universal 3D3 antibody of human PODXL is pluripotent cytotoxic, and identifies a residual population after extended differentiation of pluripotent stem cells. Stem Cells Dev 25(7):556–568. https://doi.org/10.1089/scd.2015.0321
4. Hu K, Ianov L, Crossman D (2020) Profiling and quantification of pluripotency reprogramming reveal that WNT pathways and cell morphology have to be reprogramed extensively. Heliyon 6(5):e04035. https://doi.org/10.1016/j.heliyon.2020.e04035
5. Gu Z (2019) Complexheatmap complete. https://jokergoo.github.io/ComplexHeatmap-reference/book/
6. Kolde R (2019) pheatmap: pretty heatmaps https://cran.r-project.org/web/packages/pheatmap/pheatmap.pdf
7. Khomtchouk BB, Van Booven DJ, Wahlestedt C (2014) HeatmapGenerator: high performance RNAseq and microarray visualization software suite to examine differential gene expression levels using an R and C++ hybrid computational pipeline. Source Code Biol Med 9(1):30. https://doi.org/10.1186/s13029-014-0030-2
8. Liaw A, Genteman R, Maechler M, Huber W, Warnes G. heatmap.3: enhanced heatmap based on gplots::heatmap.2 https://cran.r-project.org/web/packages/heatmap3/vignettes/vignette.pdf
9. Neuwirth E (2015) ColorBrewer Pallettes http://ugrad.stat.ubc.ca/R/library/RColorBrewer/html/ColorBrewer.html
10. Hu K (2020) Become competent within one day in generating boxplots and violin plots for a novice without prior R experience. Methods and Protocols 3(4):64. https://doi.org/10.3390/mps3040064