



易汉博 | 领先的大数据与健康解决方案
Leading solutions for big data and health

R, Cytoscape, AI 生信作图分析培训

易生信培训团队

联系我们: train@ehbio.com

2018-05-09

Contents

1 考题	9
1.1 理论题 (50 分)	9
1.2 代码题 (70 分)	10
1.3 错误识别题 (选答)	12
2 R 基础	13
2.1 R 安装	13
2.2 Rstudio 基础	14
2.2.1 Rstudio 版本	14
2.2.2 Rstudio 安装	15
2.2.3 Rstudio 使用	16
2.3 R 基本语法	17
2.3.1 获取帮助文档，查看命令或函数的使用方法、事例或适用范围	17
2.3.2 R 中的变量及其初始化	17
2.3.3 变量类型和转换	20
2.3.4 R 中矩阵运算	21
2.3.5 R 中矩阵筛选合并	28
2.3.6 str 的应用	30
2.3.7 R 的包管理	32

2.4 CheetSheets	33
2.5 参考	33
3 R plots	34
3.1 qplot 绘制图形 (王绪宁)	34
3.2 热图绘制	47
3.2.1 生成测试数据	47
3.2.2 转换数据格式	50
3.2.3 分解绘图	51
3.2.4 图形存储	60
3.3 热图美化	61
3.3.1 对数转换	62
3.3.2 Z-score 转换	64
3.3.3 抹去异常值	66
3.3.4 非线性颜色	68
3.3.5 调整行或列的顺序	71
3.4 热图绘制 - pheatmap	72
3.5 箱线图	77
3.5.1 一步步解析箱线图绘制	78
3.5.2 绘制单个基因 (A) 的箱线图	85
3.5.3 长矩阵绘制箱线图	87

3.6 线图	90
3.6.1 单线图	90
3.6.2 多线图	99
3.6.3 横轴文本线图	101
3.7 散点图	103
3.7.1 横纵轴都为数字的散点图解析	103
3.7.2 横纵轴都为字符串的散点图展示	110
3.8 功能富集泡泡图	113
3.8.1 单样品分开绘制	115
3.8.2 多样品合并绘制	118
3.9 韦恩图	121
3.9.1 韦恩图三个圈	121
3.9.2 韦恩图五个圈	122
3.9.3 UpSetView 展示	125
3.10 柱状图绘制	127
3.10.1 常规矩阵柱状图绘制	127
3.10.2 长矩阵分面绘制	136
3.11 图形支持中文字体	143
3.11.1 修改图形的字体	143
3.11.2 ggplot2 支持中文字体输出 PDF	144

3.11.3 系统可用字体	144
3.11.4 合并字体支持中英文	144
3.11.5 一个示例	145
3.12 PCA	146
3.12.1 主成分分析简介	147
3.12.2 主成分分析的意义	147
3.12.3 示例展示原始变量对样品的分类	148
3.12.4 PCA 的实现原理	162
3.12.5 简单的 PCA 实现	167
3.12.6 PCA 结果解释	174
3.12.7 PCA 应用于测试数据	175
3.12.8 PCA 注意事项	180
3.12.9 参考资料	180
3.13 表达聚类分析	181
3.14 生存分析	181
3.14.1 R 做生存分析	183
3.15 一步作图的优势	191
3.16 不改脚本的热图绘制	193
3.16.1 箱线图 - 一步绘制	198
3.16.2 线图 - 一步绘制	204

3.16.3 一网打进散点图绘制	216
3.17 参考资料	218
4 交互式图	219
4.1 交互式网络图	219
4.2 交互式桑基图	221
5 在线绘图	227
5.1 功能和数据概览	227
5.1.1 图形支持	227
5.1.2 整体界面	228
5.1.3 数据格式	229
5.2 图形展示和绘图展示	230
5.2.1 线图	230
5.2.2 富集分析泡泡图	232
5.2.3 热图	233
5.2.4 箱线图	234
5.2.5 散点图	235
5.2.6 柱状图	236
5.2.7 火山图	237
5.2.8 维恩图	237

5.2.9 UpsetView	238
5.2.10 共表达密度图	238
5.2.11 直方图	239
5.2.12 PCA	239
5.2.13 曼哈顿图	240
5.2.14 PCoA	240
5.2.15 CPcoA	240
5.2.16 桑基图	241
5.2.17 R 绘图文章	241
6 网络图	243
6.0.1 基本操作	243
6.0.2 miRNA-mRNA 调控网络	243
6.0.3 不同的布局的调试和修改	243
6.1 参考	249
7 图形排版	251
7.0.1 矢量图和标量图	251
7.0.2 矢量图的制作	251
7.0.3 矢量图编辑工具	252
7.0.4 作图基本原则	252

7.0.5 作图中的要点注意	253
7.0.6 Adobe Illustrator 中的基本概念和操作描述	254
7.0.7 视频教程	254
7.0.8 动图教程	255
7.1 参考	257
8 参考	258
8.1 程序学习心得	258
8.2 NGS 分析工具评估	258
8.3 文献精读	258
8.4 Linux 学习	258
8.5 R 统计和作图	259
8.6 NGS 基础	259
8.7 癌症数据库	260
8.8 Python 学习	260
8.9 NGS 软件	260
8.10 Cytoscape 网络图	260
8.11 分子对接	260
8.12 生信宝典之傻瓜式	261
8.13 生信人写程序	261
8.14 小技巧系列	261

8.15 招聘	261
9 公司简介	263

1 考题

1.1 理论题 (50 分)

无特殊标注的题为 1 分，其它为标注的分数。选答题特殊标注，若无标注，则必答。必答题不答着，不合格。

1. 简述 R,Rstudio 的区别?
2. R 中如何查看函数的帮助信息?
3. R 中变量名字命名有什么需要遵守的规则?
4. R 中怎么判断一个变量的类型是数值型?
5. scale 函数对数据做了什么? (2 分)
6. R 中如何把矩阵 (matrix) 转换为数据框 (data frame)?
7. 取出矩阵中所有值都为 0 的行? (2 分)
8. 写程序从矩阵和数据框中提取特定列?
9. 简述 reshape2 中定义的 wide format 和 long format 矩阵的区别? ggplot2 绘图接受的是那种格式? (3 分)
10. ggplot2 绘制热图使用的是什么函数? (2 分)
11. ggplot2 绘图时 fill 和 color 的区别是? (2 分)
12. ggplot2 绘图时如何调整横轴标签的旋转角度?
13. 因矩阵中数值相差很大导致绘制出的热图颜色区分不明显时有哪几种解决方式? (3 分)
14. ggplot2 绘制热图时如何自定义行或列的顺序? (3 分)
15. 箱线图怎么理解? (2 分)
16. ggplot2 绘图时如何调整 legend 的位置?
17. 火山图横轴和纵轴分别表示什么?
18. PCA 图怎么理解? (2 分)
19. 生存分析图怎么理解? (2 分)
20. R 中安装包的方式有几种? (2 分)
21. Cytoscape 输入数据最少需要几列? (2 分)
22. Cytoscape 中有哪几种常用的布局算法? (3 分)
23. Cytoscape 怎么给边增加箭头? (2 分)
24. Cytoscape 中怎么插入柱状图? (2 分)
25. 矢量图和标量图有什么区别? (2 分)

26. 什么是 Adobe Illustrator (AI) 的剪切蒙版？
27. AI 中选择工具和直接选择工具什么区别？
28. AI 中设置一组文字之间等距对齐使用哪个按钮？
29. AI 中怎么给一个组分赋予另一组分的属性，比如有 1 条线是 2 pt 红色，如何快速的让其它线跟其属性一致？
30. 请说出一起学习的 3 位小伙伴的名字。

1.2 代码题 (70 分)

无特殊标注的题为 3 分，其它为标注的分数。选答题特殊标注，若无标注，则必答。必答题不答着，不合格。

1. R 中写代码获得如下向量

```
"Gene_1" "Gene_2" "Gene_3" "Gene_4" "Gene_5" "Gene_6" "Gene_7" "Gene_8"
```

2. R 中运行 4+1:3 和 (4+1):3 得到的结果分别是什么？为什么？
3. 在 R 中运行以下代码得到的数据框的列名字是什么？

```
text="ID;2 cell;4 cell;8 cell;embryo
Pou5f1_1;2;3;4;5
Nanog_1;2;3.2;4.3;5
c-Myc_2;2;3;4;5
Tet1_3;2;3;4;5"

read.table(text=text,sep=";", header=T, row.names=1)
```

4. 题 (3) 中的矩阵转换为如下格式需要怎么操作？(5 分)

	embryo	variable	value
1	5	X2.cell	2.0
2	5	X2.cell	2.0
3	5	X2.cell	2.0
4	5	X2.cell	2.0
5	5	X4.cell	3.0
6	5	X4.cell	3.2
7	5	X4.cell	3.0
8	5	X4.cell	3.0
9	5	X8.cell	4.0

CONTENTS

```
10      5  X8.cell    4.3
11      5  X8.cell    4.0
12      5  X8.cell    4.0
```

5. R 中写程序把题 (3) 中的矩阵按照每行数字变化幅度由大到小排序。
6. 写代码利用题 (3) 的数据绘制热图。
7. 写代码利用题 (3) 的数据绘制线图，展示基因在不同样品表达变化趋势。
8. 写代码利用如下数据绘制 3 个集合的韦恩图。

```
list1 = sample(letters, 20)
list2 = sample(letters, 18)
list3 = sample(letters, 25)
```

9. 写代码绘制富集分析泡泡图 (最主要的是排序, 先按 ontology 排序, 再按 log_odds_ratio, 再按-log10(qvalue))。(5 分)

Ontology	Term	Count	log_odds_ratio	qvalue
biological_process	one-carbon metabolic process	34	1.012	0.001
biological_process	single-organism process	5781	0.070	8.140
biological_process	single-organism cellular process	4988	0.060	0.002
biological_process	cell communication	2169	0.100	0.007
biological_process	signal transduction	1955	0.107	0.006
biological_process	signaling	2100	0.102	0.006
biological_process	single organism signaling	2100	0.102	0.006
biological_process	response to stimulus	3251	0.074	0.012
molecular_function	protein binding	3299	0.101	3.321
cellular_component	cytoplasm	4711	0.065	0.001
cellular_component	Golgi apparatus	611	0.171	0.057
cellular_component	endomembrane system	1521	0.146	0.000
cellular_component	cytoplasmic part	3314	0.067	0.037
cellular_component	cell periphery	2059	0.086	0.065

10. 写代码用题 (3) 的矩阵绘制堆积柱状图，展示样品中不同基因表达的相对高低。(5 分)

11. 如下数据绘制火山图，横轴为 logFoldChange，纵轴为-log10(qvalue)。(5 分)

```
log2fc <- rnorm(5000, mean=0, sd=10)
qvalue <- 10^(-1 * abs(log2fc))
data <- data.frame(log2fc=log2fc, qvalue=qvalue)
```

12. 描述 Cytoscape 网络数据和节点数据输入格式。

CONTENTS

13. Cytoscape 中如何设置节点形状？
14. Cytoscape 中如何将节点数值性数据映射到节点的颜色？
15. Cytoscape 中如何设置边的线形？
16. 描述 Cytoscape 绘制 miRNA-gene 调控网络过程？
17. 描述 Cytoscape 使用 Bingo 进行功能富集分析的步骤？
18. 描述 Cytoscape 使用 KeggParser 如何导入一个通路，并映射表达量？(5 分)
19. 如何使用最新的 Cytoscape 直接获得关注的基因的调控网络？
20. 选择 2 种上面绘制的图利用 Adobe Illustrator 编辑。

1.3 错误识别题 (选答)

请辨识以下错误可能的原因是：

1. `read.table: Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, : No 11 elements in the first line.`
2. `read.table: duplicate 'row.name' are not allowed`

2 R 基础

R 语言是比较常用的统计分析和绘图语言，拥有强大的统计库、绘图库和生信分析的 Bioconductor 库，是学习生物信息分析的必备语言之一。

2.1 R 安装

Linux 下安装

如果使用的是新版的操作系统。直接可以用 `sudo apt-get install r-base` 或者 `yum install r-base` 来安装。

若系统版本老，或没有根用户权限，则需要下载编译源码安装，最新地址为<https://cran.r-project.org/src/base/R-latest.tar.gz>。

具体编译方式为 (Linux 下软件安装见 <http://blog.genesino.com/2016/06/bash1>):

```
# configure 是收集系统信息，生成Makefile 的过程
# --enable-R-shlib 需要设置，使得其他程序包括Rstudio 可以使用R 的动态库
# --prefix 指定软件安装目录，需使用绝对路径
./configure --prefix=/home/ehbio/R/3.4.0 --enable-R-shlib

# 也可以使用这个命令，共享系统的blas 库，提高运输速度
# ./configure --prefix=/home/ehbio/R/3.4.0 --enable-R-shlib --with-blas --with-lapack

# make 是编译的过程
make

# 安装到指定目录的过程
make install
```

安装完成之后，在 Linux 终端输入 `R` 即可启动交互式运行界面，`ctrl+d` 退出 `R` 运行界面。若提示找不到命令，需要判断有没有加入进环境变量。

Windows 下安装

下载 <https://cran.r-project.org/bin/windows/> 双击就可以了。

两者都需要考虑环境变量，若有问题，见文后的参考。

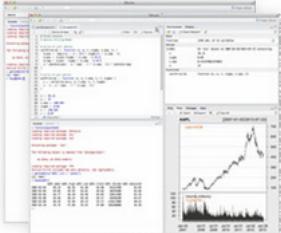
2.2 Rstudio 基础

Rstudio 是编辑、运行 R 语言的最为理想的工具之一，支持纯 R 脚本、Rmarkdown (脚本文档混排)、Bookdown (脚本文档混排成书)、Shiny (交互式网络应用) 等。

2.2.1 Rstudio 版本

Rsdutio 分为桌面版和服务器版，桌面版可以在单机使用，服务器版可以从浏览器访问供多人使用。

服务器版安装好之后，访问地址为 (8787 为默认端口号)，用户名和密码为 Linux 用户的用户名和密码。



Choose Your Version of RStudio

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace. [Learn More about RStudio features.](#)

RStudio Desktop Open Source License	RStudio Desktop Commercial License	RStudio Server Open Source License	RStudio Server Pro Commercial License	RStudio Server Pro + RStudio Connect Commercial License
FREE	\$995 per year	FREE	\$9,995 per year	\$29,995 per year
Integrated Tools for R	●	●	●	●
Priority Support	●		●	●
Access via Web Browser		●	●	●
Enterprise Security			●	●
Project Sharing			●	●
Manage Multiple R Sessions & Versions			●	●
Admin			●	●

2.2.2 Rstudio 安装

Linux 下安装服务器版

安装参考 <https://www.rstudio.com/products/rstudio/download-server/>

```
wget https://download2.rstudio.org/rstudio-server-rhel-1.0.136-x86_64.rpm
sudo yum install --nogpgcheck rstudio-server-rhel-1.0.136-x86_64.rpm
```

安装完之后的检测、启动和配置

```
sudo rstudio-server verify-installation #查看是否安装正确
sudo rstudio-server start ## 启动
sudo rstudio-server status ## 查看状态
sudo rstudio-server stop ## 停止
ifconfig | grep 'inet addr' ## 查看服务端ip地址
sudo rstudio-server start ## 修改配置文件后重启
sudo rstudio-server active-sessions ## 列出活跃的sessions:
sudo rstudio-server suspend-session <pid> ## 暂停session
sudo rstudio-server suspend-all ##暂停所有session
```

- Rstudio 日志目录，方便查看错误信息：/var/log/rstudio-server/
- 配置文件：
- /etc/rstudio/rserver.conf

```
www-port=8787 (default)
www-address=0.0.0.0 (default)
rsession-ld-library-path=/opt/local/lib:/opt/local/someapp/lib
rsession-which-r=/usr/local/bin/R
```

- /etc/rstudio/rsession.conf

- Timeout

```
[user]
session-timeout-minutes=30
[@powerusers]
session-timeout-minutes=0
```

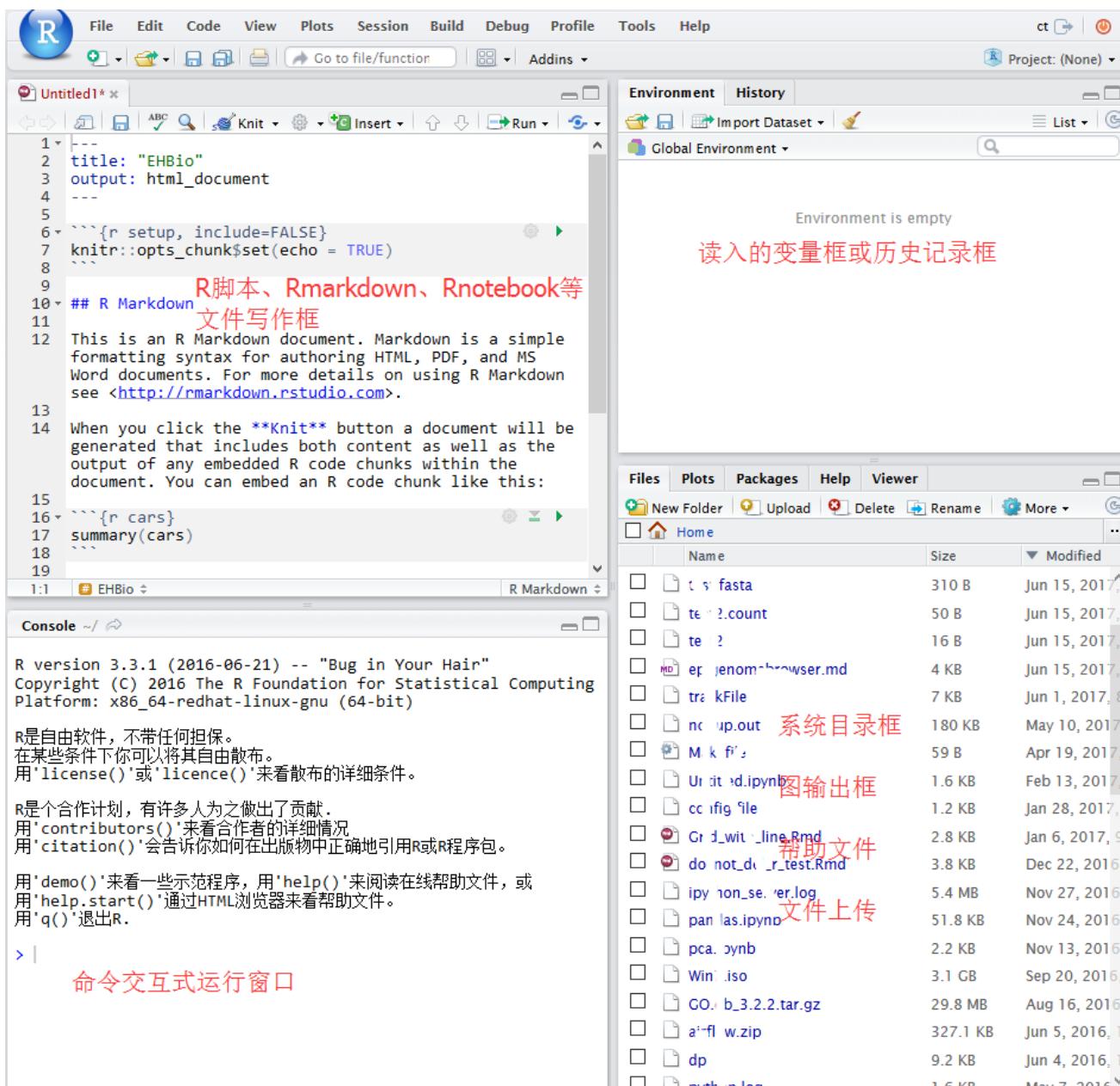
Windows 下安装桌面版

下载之后 (<https://www.rstudio.com/products/rstudio/download2/>) 双击安装，需要使用管理员权限，其它无需要注意的。

2.2.3 Rstudio 使用

Windows 下桌面版直接双击打开即可使用，Linux 服务器版访问地址为，用户名和密码为 Linux 用户的用户名和密码。

2.2.3.1 Rstudio 界面



2.2.3.2 Rstudio 中新建或打开文件 如果是桌面版，直接就可以访问“我的电脑”去打开之前写过的脚本。如果是服务器版，可直接访问服务器上写过的脚本。Rstudio 右下 1/4 部分可以切换目录，点击 more，设置工作目录。可以上传本地的脚本到对应目录打开。

2.3 R 基本语法

2.3.1 获取帮助文档，查看命令或函数的使用方法、事例或适用范围

```
>>> ?command
>>> ??command # 深度搜索或模糊搜索此命令

>>> example(command) # 得到命令的例子

>>> example(plot)
```

2.3.2 R 中的变量及其初始化

```
# 数字变量
a <- 10
a
```

```
## [1] 10
```

```
# 字符串变量
a <- "abc"
a
```

```
## [1] "abc"
```

```
# 逻辑变量
a <- TRUE
a
```

```
## [1] TRUE
```

CONTENTS

```
b <- T
b
```

```
## [1] TRUE
```

```
d <- FALSE
d
```

```
## [1] FALSE
```

```
# 向量
a <- vector(mode="logical", length=5)
a
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```
a <- c(1,2,3,4)
# 判断一个变量是不是 vector
is.vector(a)
```

```
## [1] TRUE
```

```
a <- list(element1=c(1,2,3,4), element2=1:5)
a
```

```
## $element1
## [1] 1 2 3 4
##
## $element2
## [1] 1 2 3 4 5
```

```
# 矩阵
a <- matrix(1:20, nrow=5, ncol=4, byrow=T)
a
```

```
##      [,1] [,2] [,3] [,4]
## [1,]     1     2     3     4
## [2,]     5     6     7     8
## [3,]     9    10    11    12
## [4,]    13    14    15    16
## [5,]    17    18    19    20
```

CONTENTS

```
is.matrix(a)
```

```
## [1] TRUE
```

```
dim(a) # 查看或设置数组的维度向量
```

```
## [1] 5 4
```

```
# 错误的用法
```

```
# dim(a) <- c(4, 4)
```

```
# 正确的用法
```

```
a <- 1:20
```

```
dim(a) <- c(5, 4) # 转换向量为矩阵
```

```
a
```

```
##      [,1] [,2] [,3] [,4]
## [1,]     1    6   11   16
## [2,]     2    7   12   17
## [3,]     3    8   13   18
## [4,]     4    9   14   19
## [5,]     5   10   15   20
```

```
print(paste("矩阵 a 的行数", nrow(a)))
```

```
## [1] "矩阵a的行数 5"
```

```
print(paste("矩阵 a 的列数", ncol(a)))
```

```
## [1] "矩阵a的列数 4"
```

```
# 查看或设置行列名
```

```
rownames(a)
```

```
## NULL
```

```
rownames(a) <- c('a', 'b', 'c', 'd', 'e')
```

```
a
```

CONTENTS

```
## [,1] [,2] [,3] [,4]
## a     1     6    11   16
## b     2     7    12   17
## c     3     8    13   18
## d     4     9    14   19
## e     5    10    15   20
```

R 中获取一系列的字母

```
letters[1:4]
```

```
## [1] "a" "b" "c" "d"
```

```
colnames(a) <- letters[1:4]
```

```
a
```

```
##   a   b   c   d
## a 1  6 11 16
## b 2  7 12 17
## c 3  8 13 18
## d 4  9 14 19
## e 5 10 15 20
```

2.3.3 变量类型和转换

不同的变量类型有不同的操作方式，`is` 系列和 `as` 系列函数用来判断变量的属性和转换变量的属性

```
is.character(a)
```

```
## [1] FALSE
```

```
is.numeric(a)
```

```
## [1] TRUE
```

```
is.matrix(a)
```

```
## [1] TRUE
```

```
is.data.frame(a)
```

```
## [1] FALSE
```

```
is.data.frame(as.data.frame(a))
```

```
## [1] TRUE
```

2.3.4 R 中矩阵运算

```
# 获得随机的正态分布数据
# random generation for the normal distribution with mean equal to 'mean'
# and standard deviation equal to 'sd'.
rnorm(10, mean = 0, sd = 1) # 正态分布的随机数
```

```
## [1] -1.7623698 0.1703914 -1.0272638 -1.2723234 -0.9341441 2.2230742
## [7] -0.3978386 1.6669337 -1.0560411 -2.8457878
```

```
# 获得随机的均匀分布的数据
# random generation for the uniform distribution with mean equal to 'mean'
# and standard deviation equal to 'sd'.
runif(10, min = 0, max = 1) # 平均分布的随机数
```

```
## [1] 0.1204988 0.2521150 0.9239175 0.7009465 0.8067070 0.4582831 0.9140327
## [8] 0.4635908 0.6317504 0.3384500
```

```
rep(1,5) # 把 1 重复 5 次
```

```
## [1] 1 1 1 1 1
```

```
scale(1:5) # 标准化数据
```

```
## [,1]
## [1,] -1.2649111
## [2,] -0.6324555
## [3,] 0.0000000
```

CONTENTS

```
## [4,] 0.6324555
## [5,] 1.2649111
## attr(),"scaled:center")
## [1] 3
## attr(),"scaled:scale")
## [1] 1.581139
```

`scale` 默认操作对等于

```
# 标准化数据
a <- 1:5
(a - mean(a)) / sd(a)
```

```
## [1] -1.2649111 -0.6324555 0.0000000 0.6324555 1.2649111

a <- c(rnorm(5), rnorm(5,1), runif(5), runif(5,-1,1), 1:5, rep(0,5), c(2,10,11,13,4),
       scale(1:5)[1:5])
a
```

```
## [1] 0.571515883 -0.786827497 -0.069386297 -0.366760489 -2.336466642
## [6] -0.256031715 0.976938913 2.239830880 1.307826807 1.689656291
## [11] 0.005828207 0.908370379 0.220653976 0.415340775 0.237234787
## [16] -0.996525097 0.179710705 -0.499995762 -0.680745059 -0.838659695
## [21] 1.0000000000 2.0000000000 3.0000000000 4.0000000000 5.0000000000
## [26] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [31] 2.0000000000 10.0000000000 11.0000000000 13.0000000000 4.0000000000
## [36] -1.264911064 -0.632455532 0.000000000 0.632455532 1.264911064
```

```
# ncol=5 5 列
# byrow=T: 先填充行
a <- matrix(a, ncol=5, byrow=T)
a
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.571515883 -0.7868275 -0.0693863 -0.3667605 -2.3364666
## [2,] -0.256031715 0.9769389 2.2398309 1.3078268 1.6896563
## [3,] 0.005828207 0.9083704 0.2206540 0.4153408 0.2372348
## [4,] -0.996525097 0.1797107 -0.4999958 -0.6807451 -0.8386597
## [5,] 1.0000000000 2.0000000 3.0000000 4.0000000 5.0000000
## [6,] 0.0000000000 0.0000000 0.0000000 0.0000000 0.0000000
## [7,] 2.0000000000 10.0000000 11.0000000 13.0000000 4.0000000
## [8,] -1.264911064 -0.6324555 0.0000000 0.6324555 1.2649111
```

```
# 按行加和
rowSums(a)
```

```
## [1] -2.987925 5.958221 1.787428 -2.836215 15.000000 0.000000 40.000000
## [8] 0.000000
```

```
# 注意检查括号的配对
#a <- a[rowSums(abs(a))!=0,]
# 错误：意外的'J' in "a <- a[rowSums(abs(a))!=0,]"
```

```
# 去除全部为 0 的行
a <- a[rowSums(abs(a))!=0,]
```

```
# 另外一种方式去除全部为 0 的行
#a[rowSums(a==0) < ncol(a),]
```

```
a
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  0.571515883 -0.7868275 -0.0693863 -0.3667605 -2.3364666
## [2,] -0.256031715  0.9769389  2.2398309  1.3078268  1.6896563
## [3,]  0.005828207  0.9083704  0.2206540  0.4153408  0.2372348
## [4,] -0.996525097  0.1797107 -0.4999958 -0.6807451 -0.8386597
## [5,]  1.000000000  2.0000000  3.0000000  4.0000000  5.0000000
## [6,]  2.000000000 10.0000000 11.0000000 13.0000000 4.0000000
## [7,] -1.264911064 -0.6324555  0.0000000  0.6324555  1.2649111
```

矩阵运算，R 默认针对整个数据进行常见运算

```
# 所有值都乘以 2
a * 2
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  1.14303177 -1.5736550 -0.1387726 -0.7335210 -4.6729333
## [2,] -0.51206343  1.9538778  4.4796618  2.6156536  3.3793126
## [3,]  0.01165641  1.8167408  0.4413080  0.8306815  0.4744696
## [4,] -1.99305019  0.3594214 -0.9999915 -1.3614901 -1.6773194
## [5,]  2.000000000 4.0000000  6.0000000  8.0000000 10.0000000
## [6,]  4.000000000 20.0000000 22.0000000 26.0000000 8.0000000
## [7,] -2.52982213 -1.2649111  0.0000000  1.2649111  2.5298221
```

所有值取绝对值，再取对数 (取对数前一般加一个数避免对 0 或负值取对数)

```
log2(abs(a) + 1)
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.652156854 0.8374004 0.0967831 0.4507604 1.7383211
## [2,] 0.328872893 0.9832683 1.6959185 1.2065350 1.4274218
## [3,] 0.008383917 0.9323412 0.2876543 0.5011495 0.3071193
## [4,] 0.997491207 0.2384331 0.5849584 0.7491009 0.8786545
## [5,] 1.000000000 1.5849625 2.0000000 2.3219281 2.5849625
## [6,] 1.584962501 3.4594316 3.5849625 3.8073549 2.3219281
## [7,] 1.179454401 0.7070437 0.0000000 0.7070437 1.1794544
```

取出最大值、最小值、行数、列数

```
max(a)
```

```
## [1] 13
```

```
min(a)
```

```
## [1] -2.336467
```

```
nrow(a)
```

```
## [1] 7
```

```
ncol(a)
```

```
## [1] 5
```

增加一列或一行

#*cbind: column bind*

```
cbind(a, 1:7)
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]  [,6]
## [1,] 0.571515883 -0.7868275 -0.0693863 -0.3667605 -2.3364666 1
## [2,] -0.256031715  0.9769389  2.2398309  1.3078268  1.6896563 2
## [3,]  0.005828207  0.9083704  0.2206540  0.4153408  0.2372348 3
## [4,] -0.996525097  0.1797107 -0.4999958 -0.6807451 -0.8386597 4
## [5,]  1.000000000  2.0000000  3.0000000  4.0000000  5.0000000 5
## [6,]  2.000000000 10.0000000 11.0000000 13.0000000 4.0000000 6
## [7,] -1.264911064 -0.6324555  0.0000000  0.6324555  1.2649111 7
```

```
# rbind: row bind
rbind(a, 1:5)
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  0.571515883 -0.7868275 -0.0693863 -0.3667605 -2.3364666
## [2,] -0.256031715  0.9769389  2.2398309  1.3078268  1.6896563
## [3,]  0.005828207  0.9083704  0.2206540  0.4153408  0.2372348
## [4,] -0.996525097  0.1797107 -0.4999958 -0.6807451 -0.8386597
## [5,]  1.000000000  2.0000000  3.0000000  4.0000000  5.0000000
## [6,]  2.000000000 10.0000000 11.0000000 13.0000000 4.0000000
## [7,] -1.264911064 -0.6324555  0.0000000  0.6324555  1.2649111
## [8,]  1.000000000  2.0000000  3.0000000  4.0000000  5.0000000
```

计算每一行的 var (方差)
apply 表示对数据 (第一个参数) 的每一行 (第二个参数赋值为 1) 或每一列 (2) 操作
最后返回一个列表

```
apply(a, 1, var)
```

```
## [1] 1.1896482 0.8755184 0.1159168 0.2083237 2.5000000 22.5000000
## [7] 1.0000000
```

计算每一行的 mad (中值绝对偏差)
一般认为比方差的鲁棒性更强，更少受异常值的影响，更能反映数据间的差异)
函数中的第二个参数 1 表示按行操作

```
apply(a, 1, mad)
```

```
## [1] 0.6227913 0.5661004 0.2640599 0.2679789 1.4826000 4.4478000 0.9376786
```

计算每一列的平均值

```
apply(a, 2, mean)
```

```
## [1] 0.1514109 1.8065339 2.2701575 2.6154454 1.2880965
```

取出中值绝对偏差大于 0.5 的行

```
b = a[apply(a, 1, mad) > 0.5, ]  
b
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  0.5715159 -0.7868275 -0.0693863 -0.3667605 -2.336467
## [2,] -0.2560317  0.9769389  2.2398309  1.3078268  1.689656
```

CONTENTS

```
## [3,] 1.0000000 2.0000000 3.0000000 4.0000000 5.0000000
## [4,] 2.0000000 10.0000000 11.0000000 13.0000000 4.0000000
## [5,] -1.2649111 -0.6324555 0.0000000 0.6324555 1.264911
```

```
# 输出 1 4 2 3 5
# 解释：原列表第一个元素显示在第一位
#       原列表第四个元素显示在第二位
#       原列表第二个元素显示在第三位
order(c(1,3,4,2,5))
```

```
## [1] 1 4 2 3 5
```

```
# 产生新的顺序
#
order(apply(b,1,mad), decreasing=T)
```

```
## [1] 4 3 5 1 2
```

```
# 矩阵按照 mad 的大小降序排列
c = b[order(apply(b,1,mad), decreasing=T), ]
c
```

```
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.0000000 10.0000000 11.0000000 13.0000000 4.0000000
## [2,] 1.0000000 2.0000000 3.0000000 4.0000000 5.0000000
## [3,] -1.2649111 -0.6324555 0.0000000 0.6324555 1.264911
## [4,] 0.5715159 -0.7868275 -0.0693863 -0.3667605 -2.336467
## [5,] -0.2560317 0.9769389 2.2398309 1.3078268 1.689656
```

```
rownames(c) <- paste('Gene', letters[1:nrow(c)], sep="_")
colnames(c) <- toupper(letters[1:ncol(c)])
c
```

	A	B	C	D	E
## Gene_a	2.0000000	10.0000000	11.0000000	13.0000000	4.0000000
## Gene_b	1.0000000	2.0000000	3.0000000	4.0000000	5.0000000
## Gene_c	-1.2649111	-0.6324555	0.0000000	0.6324555	1.264911
## Gene_d	0.5715159	-0.7868275	-0.0693863	-0.3667605	-2.336467
## Gene_e	-0.2560317	0.9769389	2.2398309	1.3078268	1.689656

矩阵转置

```
expr = t(c)
expr
```

```
##   Gene_a Gene_b      Gene_c      Gene_d      Gene_e
## A     2    1 -1.2649111  0.5715159 -0.2560317
## B    10    2 -0.6324555 -0.7868275  0.9769389
## C    11    3  0.0000000 -0.0693863  2.2398309
## D    13    4  0.6324555 -0.3667605  1.3078268
## E     4    5  1.2649111 -2.3364666  1.6896563
```

矩阵值的替换

```
expr2 = expr
expr2[expr2<0] = 0
expr2
```

```
##   Gene_a Gene_b      Gene_c      Gene_d      Gene_e
## A     2    1  0.0000000  0.5715159  0.0000000
## B    10    2  0.0000000  0.0000000  0.9769389
## C    11    3  0.0000000  0.0000000  2.2398309
## D    13    4  0.6324555  0.0000000  1.3078268
## E     4    5  1.2649111  0.0000000  1.6896563
```

矩阵中只针对某一列替换

```
# expr2 是个矩阵不是数据框，不能使用列名字索引
# expr2[expr2$Gene_b<1, "Gene_b"] <- 1
```

str 是一个最为常用、好用的查看变量信息的工具，尤其是对特别复杂的变量，

可以看清其层级结构，便于提取数据

```
str(expr2)
```

```
## num [1:5, 1:5] 2 10 11 13 4 1 2 3 4 5 ...
## - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:5] "A" "B" "C" "D" ...
##   ..$ : chr [1:5] "Gene_a" "Gene_b" "Gene_c" "Gene_d" ...
```

转换为数据框，再进行相应的操作

```
expr2 <- as.data.frame(expr2)
str(expr2)
```

```
## 'data.frame': 5 obs. of 5 variables:
```

CONTENTS

```
## $ Gene_a: num 2 10 11 13 4
## $ Gene_b: num 1 2 3 4 5
## $ Gene_c: num 0 0 0 0.632 1.265
## $ Gene_d: num 0.572 0 0 0 0
## $ Gene_e: num 0 0.977 2.24 1.308 1.69
```

```
expr2[expr2$Gene_b<1, "Gene_b"] <- 1
expr2
```

```
##   Gene_a Gene_b     Gene_c     Gene_d     Gene_e
## A      2       1 0.0000000 0.5715159 0.0000000
## B     10       2 0.0000000 0.0000000 0.9769389
## C     11       3 0.0000000 0.0000000 2.2398309
## D     13       4 0.6324555 0.0000000 1.3078268
## E      4       5 1.2649111 0.0000000 1.6896563
```

2.3.5 R 中矩阵筛选合并

```
# 读入样品信息
sampleInfo = "Samp;Group;Genotype
A;Control;WT
B;Control;WT
D;Treatment;Mutant
C;Treatment;Mutant
E;Treatment;WT
F;Treatment;WT"

phenoData = read.table(text=sampleInfo, sep=";", header=T, row.names=1, quote="")
phenoData
```

```
##           Group Genotype
## A     Control      WT
## B     Control      WT
## D Treatment    Mutant
## C Treatment    Mutant
## E Treatment      WT
## F Treatment      WT
```

```
# 把样品信息按照基因表达矩阵中的样品信息排序，并只保留有基因表达信息的样品
# '%in%' is a more intuitive interface as a binary operator, which
#     returns a logical vector indicating if there is a match or not for
```

```
# its left operand.

phenoData = phenoData[rownames(phenoData) %in% rownames(expr),]
```

```
##      Group Genotype
## A    Control     WT
## B    Control     WT
## D Treatment   Mutant
## C Treatment   Mutant
## E Treatment     WT
```

```
# 合并矩阵
# by=0 表示按照行的名字排序
# by=columnname 表示按照共有的某一列合并
# 合并后多出了新的一列 Row.names
merge_data = merge(expr, phenoData, by=0, all.x=T)
merge_data
```

```
##   Row.names Gene_a Gene_b     Gene_c     Gene_d     Gene_e     Group
## 1          A     2     1 -1.2649111  0.5715159 -0.2560317 Control
## 2          B    10     2 -0.6324555 -0.7868275  0.9769389 Control
## 3          C    11     3  0.0000000 -0.0693863  2.2398309 Treatment
## 4          D    13     4  0.6324555 -0.3667605  1.3078268 Treatment
## 5          E     4     5  1.2649111 -2.3364666  1.6896563 Treatment

##   Genotype
## 1     WT
## 2     WT
## 3  Mutant
## 4  Mutant
## 5     WT
```

```
rownames(merge_data) <- merge_data$Row.names
merge_data
```

```
##   Row.names Gene_a Gene_b     Gene_c     Gene_d     Gene_e     Group
## A          A     2     1 -1.2649111  0.5715159 -0.2560317 Control
## B          B    10     2 -0.6324555 -0.7868275  0.9769389 Control
## C          C    11     3  0.0000000 -0.0693863  2.2398309 Treatment
## D          D    13     4  0.6324555 -0.3667605  1.3078268 Treatment
## E          E     4     5  1.2649111 -2.3364666  1.6896563 Treatment

##   Genotype
## A     WT
```

```
## B      WT
## C    Mutant
## D    Mutant
## E      WT
```

去除一列 ; -1 表示去除第一列

```
merge_data = merge_data[,-1]
merge_data
```

	Gene_a	Gene_b	Gene_c	Gene_d	Gene_e	Group	Genotype
## A	2	1	-1.2649111	0.5715159	-0.2560317	Control	WT
## B	10	2	-0.6324555	-0.7868275	0.9769389	Control	WT
## C	11	3	0.0000000	-0.0693863	2.2398309	Treatment	Mutant
## D	13	4	0.6324555	-0.3667605	1.3078268	Treatment	Mutant
## E	4	5	1.2649111	-2.3364666	1.6896563	Treatment	WT

提取出所有的数值列

```
merge_data[sapply(merge_data, is.numeric)]
```

	Gene_a	Gene_b	Gene_c	Gene_d	Gene_e
## A	2	1	-1.2649111	0.5715159	-0.2560317
## B	10	2	-0.6324555	-0.7868275	0.9769389
## C	11	3	0.0000000	-0.0693863	2.2398309
## D	13	4	0.6324555	-0.3667605	1.3078268
## E	4	5	1.2649111	-2.3364666	1.6896563

2.3.6 str 的应用

`str`: Compactly display the internal structure of an R object, a diagnostic function and an alternative to ‘`summary` (and to some extent, ‘`dput`’). Ideally, only one line for each ‘basic’ structure is displayed. It is especially well suited to compactly display the (abbreviated) contents of (possibly nested) lists. The idea is to give reasonable output for *any* R object. It calls ‘`args`’ for (non-primitive) function objects.

`str` 用来告诉结果的构成方式，对于不少 Bioconductor 的包，或者复杂的 R 函数的输出，都是一堆列表的嵌套，`str(complex_result)` 会输出每个列表的名字，方便提取对应的信息。

`str` 的一个应用例子

```
str(list(a = "A", L = as.list(1:100)), list.len = 9)
```

```
## List of 2
## $ a: chr "A"
```

```
## $ L:List of 100
## ..$ : int 1
## ..$ : int 2
## ..$ : int 3
## ..$ : int 4
## ..$ : int 5
## ..$ : int 6
## ..$ : int 7
## ..$ : int 8
## ..$ : int 9
## .. [list output truncated]
```

利用 `str` 查看 `pca` 的结果，具体的 PCA 应用查看 <http://mp.weixin.qq.com/s/sREIBMkyR9rGa4TQp9KjNQ>。

```
pca_result <- prcomp(expr)
```

```
pca_result
```

```
## Standard deviations (1, ..., p=5):
## [1] 4.799777e+00 2.119814e+00 5.265774e-01 5.063709e-01 5.680469e-17
##
## Rotation (n x k) = (5 x 5):
##          PC1        PC2        PC3        PC4        PC5
## Gene_a  0.98679759 -0.1185723  0.08127263 -0.07460493  0.000000e+00
## Gene_b  0.09365872  0.7063884 -0.36055227 -0.27664287  5.345225e-01
## Gene_c  0.05923498  0.4467592 -0.22803328 -0.17496431 -8.451543e-01
## Gene_d  0.01529138 -0.4638203 -0.88544023 -0.02515012  1.186551e-15
## Gene_e  0.11711516  0.2687620 -0.16550907  0.94162508 -1.332268e-15
```

```
str(pca_result)
```

```
## List of 5
## $ sdev      : num [1:5] 4.80 2.12 5.27e-01 5.06e-01 5.68e-17
## $ rotation: num [1:5, 1:5] 0.9868 0.0937 0.0592 0.0153 0.1171 ...
##   .. attr(*, "dimnames")=List of 2
##     ..$ : chr [1:5] "Gene_a" "Gene_b" "Gene_c" "Gene_d" ...
##     ..$ : chr [1:5] "PC1" "PC2" "PC3" "PC4" ...
## $ center    : Named num [1:5] 8 3 0 -0.598 1.192
##   .. attr(*, "names")= chr [1:5] "Gene_a" "Gene_b" "Gene_c" "Gene_d" ...
## $ scale     : logi FALSE
## $ x         : num [1:5, 1:5] -6.33 1.81 3.09 5.08 -3.65 ...
##   .. attr(*, "dimnames")=List of 2
##     ..$ : chr [1:5] "A" "B" "C" "D" ...
##     ..$ : chr [1:5] "PC1" "PC2" "PC3" "PC4" ...
##   - attr(*, "class")= chr "prcomp"
```

CONTENTS

```
# 取出每个主成分解释的差异
pca_result$sdev
```

```
## [1] 4.799777e+00 2.119814e+00 5.265774e-01 5.063709e-01 5.680469e-17
```

2.3.7 R 的包管理

什么时候需要安装包

```
library('unExistedPackage')
Error in library("unExistedPackage") :
不存在叫'unExistedPackage'这个名字的程辑包
```

如何安装 R 包

```
install.packages("package_name")
# 指定安装来源
install.packages("package_name", repo="http://cran.us.r-project.org")

# 安装Bioconductor的包
source('https://bioconductor.org/biocLite.R')
biocLite('BiocInstaller')
biocLite(c("RUVSeq", "pcaMethods"))

# 安装Github的R包
install.packages("devtools")
devtools::install_github("JustinaZ/pcaReduce")

# 手动安装，首先下载包的源文件（压缩版就可），然后在终端运行下面的命令。
ct@ehbio:~$ R CMD INSTALL package.tar.gz

# 移除包
remove.packages("package_name")

# 查看所有安装的包
library()

# 查看特定安装包的版本
installed.packages() [c("ggplot2"), c("Package", "Version")]

#   Package Version
```

CONTENTS

```
# "DESeq2" "1.14.1"

# 查看默认安装包的位置
.libPaths()

# 查看已加载的包
.packages()

# 调用安装的包
library(package_name)
```

自动安装包

```
usePackage <- function(p) {
  if (!is.element(p, installed.packages() [, 1])) {
    install.packages(p, dep = TRUE)
  }
  require(p, character.only = TRUE)
}
```

2.4 CheetSheets

<https://www.rstudio.com/resources/cheatsheets/>

2.5 参考

- 生信宝典 Linux 系列
- 生信宝典 R 系列

3 R plots

3.1 qplot 绘制图形 (王绪宁)

ggplot2 中两大精髓的函数分别为 `qplot` (快速作图 quick plot)，类似于 R 中基本的 `plot` 和 `ggplot` (更符合 ggplot2 图层式绘图理念，一层层添加修改)。

测试数据是 ggplot2 包中自带的 `diamond` 数据，每一行为一种钻石，每一列为钻石不同的属性，如 `carat` (克拉), `cut` (切工), `color` (色泽), `clarity` (透明度) 等。

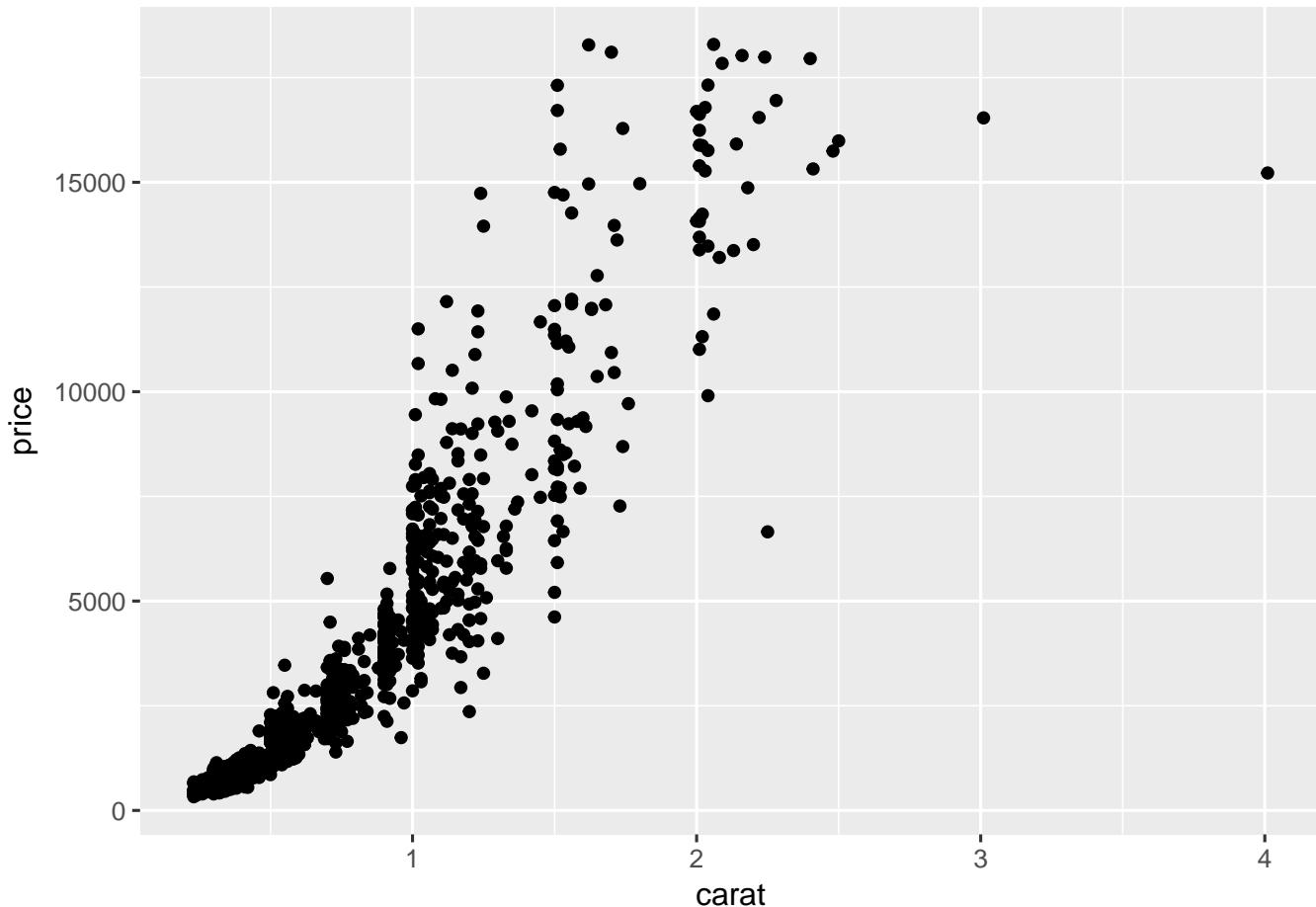
```
library(ggplot2)
set.seed(23)
dat <- diamonds[sample(nrow(diamonds), 1000), ]
head(dat)

## # A tibble: 6 x 10
##   carat cut      color clarity depth table price     x     y     z
##   <dbl> <ord>    <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.340 Ideal     D     VS2     61.0   58.    753   4.49   4.53   2.75
## 2 0.340 Ideal     G     VS2     61.0   57.    596   4.48   4.50   2.74
## 3 1.01   Ideal    F     VS2     61.6   56.    7229  6.48   6.45   3.98
## 4 0.320 Premium  E     VVS1    61.8   59.    1020  4.39   4.35   2.70
## 5 0.510 Ideal     D     SI1     61.7   55.    1569  5.11   5.16   3.17
## 6 1.22   Ideal    G     VVS1    61.1   56.    10888 6.91   6.94   4.23
```

数据读进来后，怎么绘制呢？

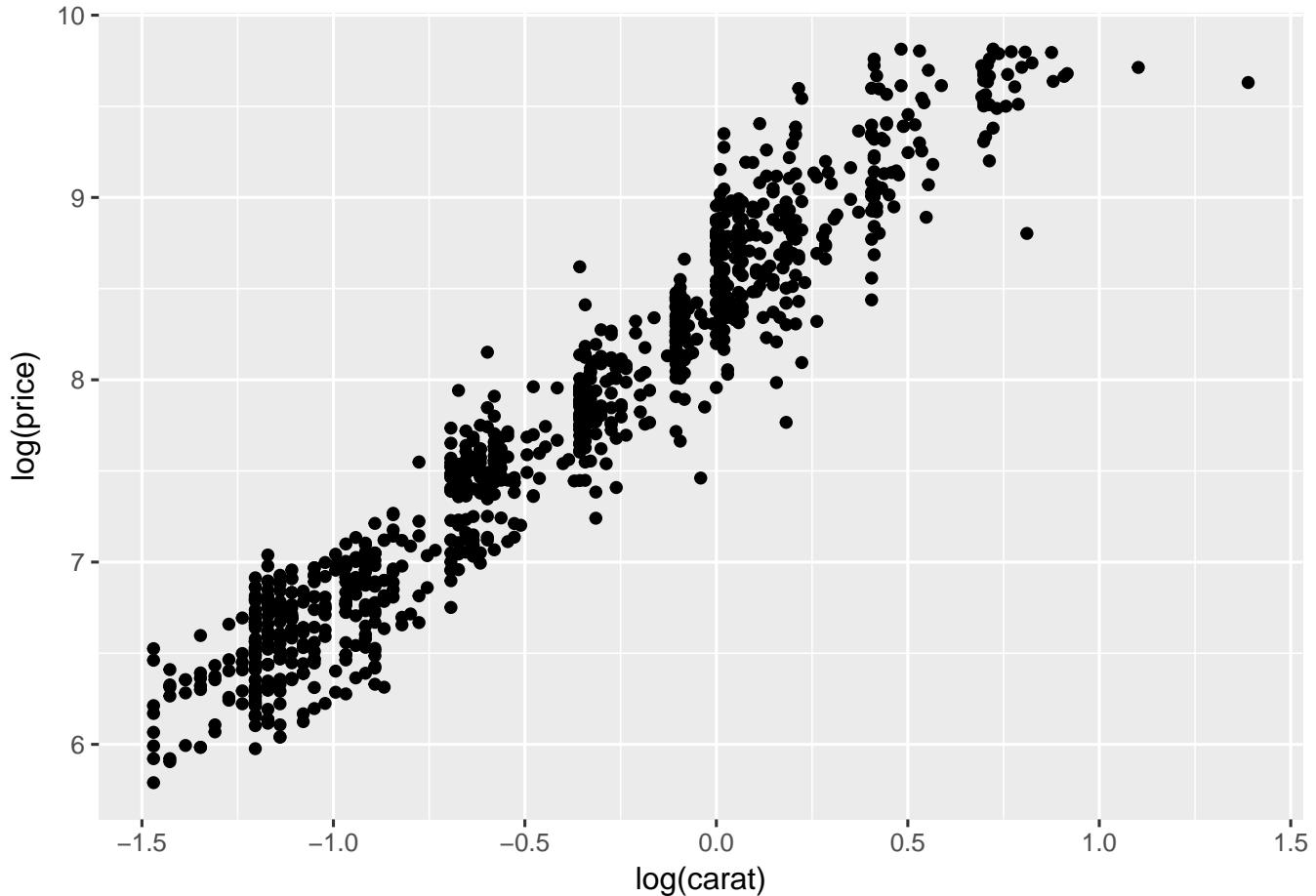
1. 绘制散点图，横轴是克拉数，纵轴是价格 (正相关)

```
qplot(carat, price, data=dat)
```



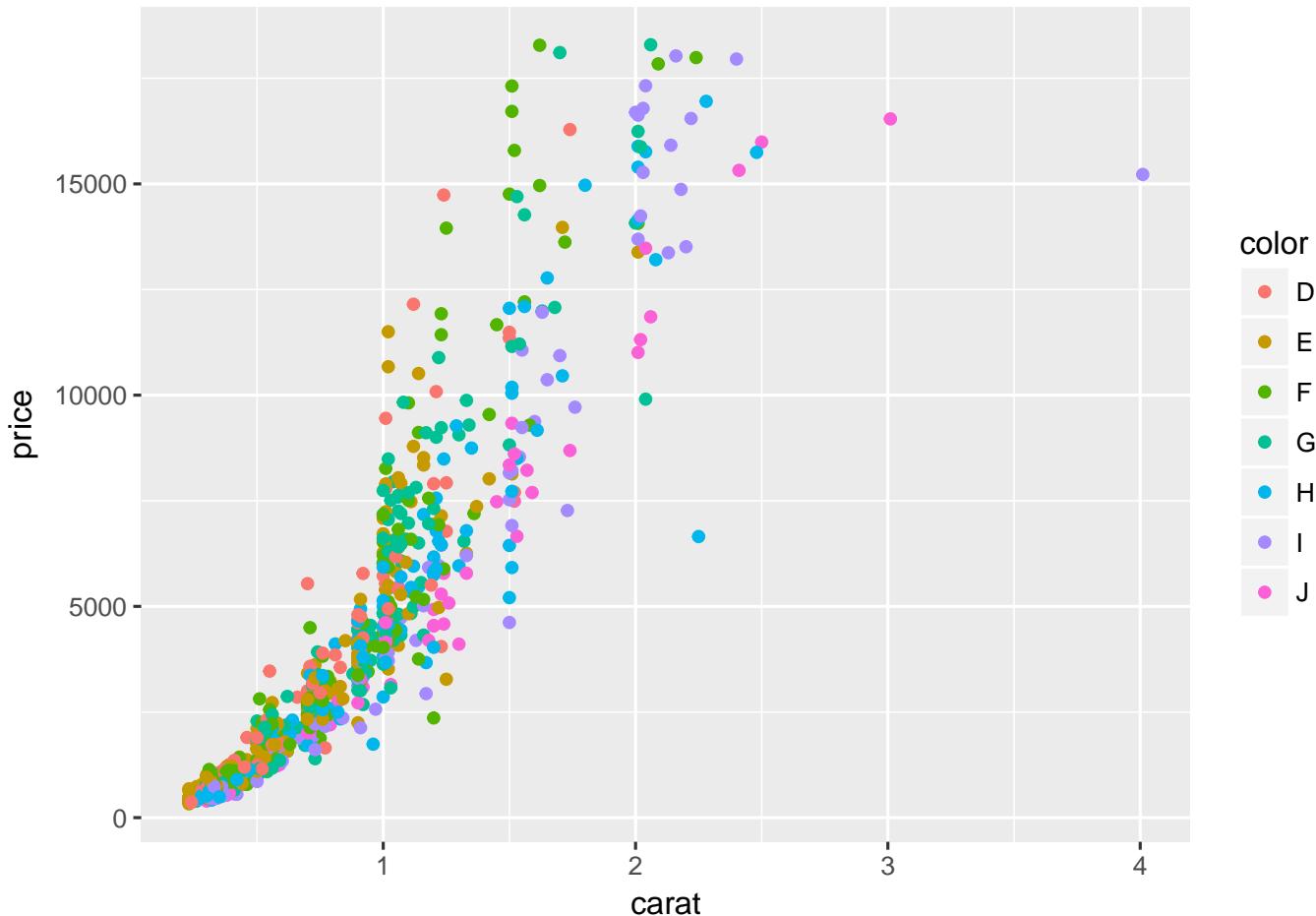
绘制散点图，对 x,y 值取 log

```
qplot(log(carat), log(price), data=dat)
```

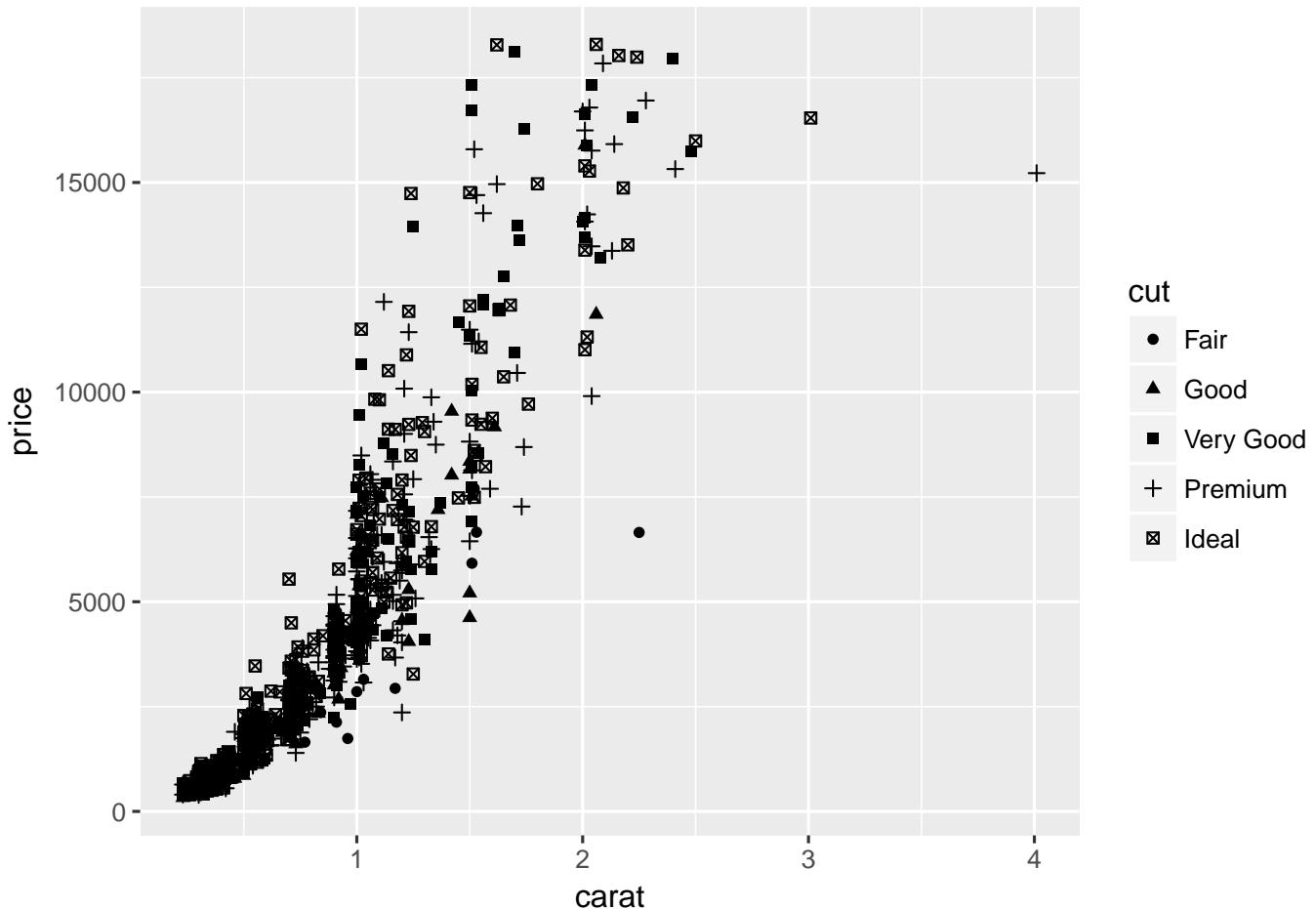


颜色、大小、性状和其他属性的设置

```
qplot(carat,price,data=dat,colour=color)
```



```
qplot(carat, price, data=dat, shape=cut) # 以 cut 为分类依据设置不同的形状
```



几何对象

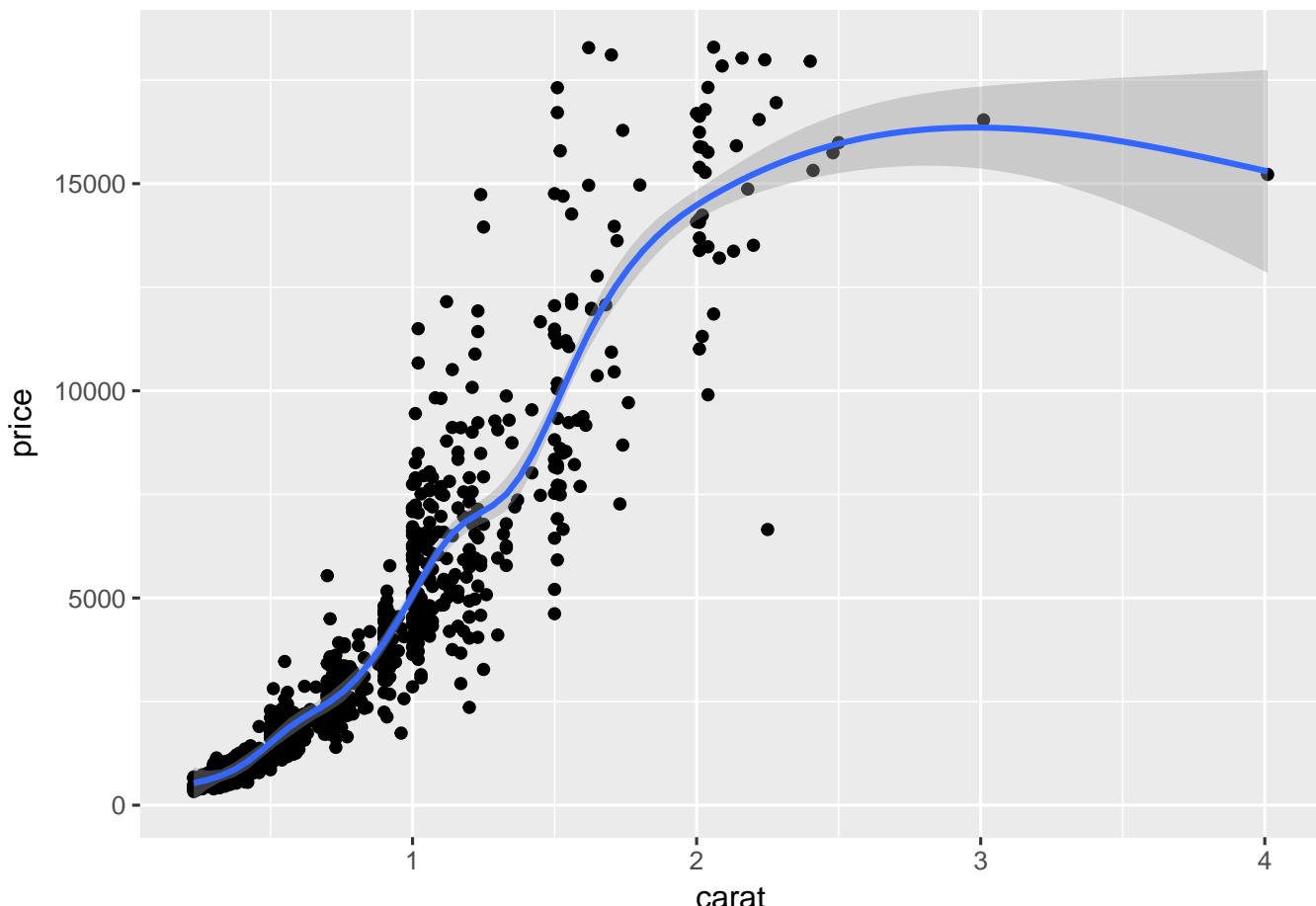
`qplot()` 函数配合不同的几何对象便可绘制出不同的图形:

- 点图 `geom="point"`
- 平滑曲线 `geom="smooth"`
- 箱线图 `geom="boxplot"`
- 任意方向的路径性 `geom="path"`
- 线条图 `geom="line"` (从左到右连接)
- 对于连续变量，直方图 `geom="histogram"`
- 频率多边图 `geom="freqpoly"`
- 绘制密度曲线 `geom="density"`
- 如果只有 `x` 参数传递给 `qplot()`, 那么默认是直方图
- 对于离散变量，`geom="bar"` 绘制条形图

```
# 向图形中加入平滑曲线 (# 从本张图片可以逐渐体会 ggplot 绘图的强大 ,  
# 后期应用 ggplot() 函数后 , 可以更加自由的绘制各种组合图形)  
qplot(carat, price, data=dat, geom=c("point", "smooth")) # 添加了一条拟合曲线
```

```
## `geom_smooth()` using method = 'gam'
```

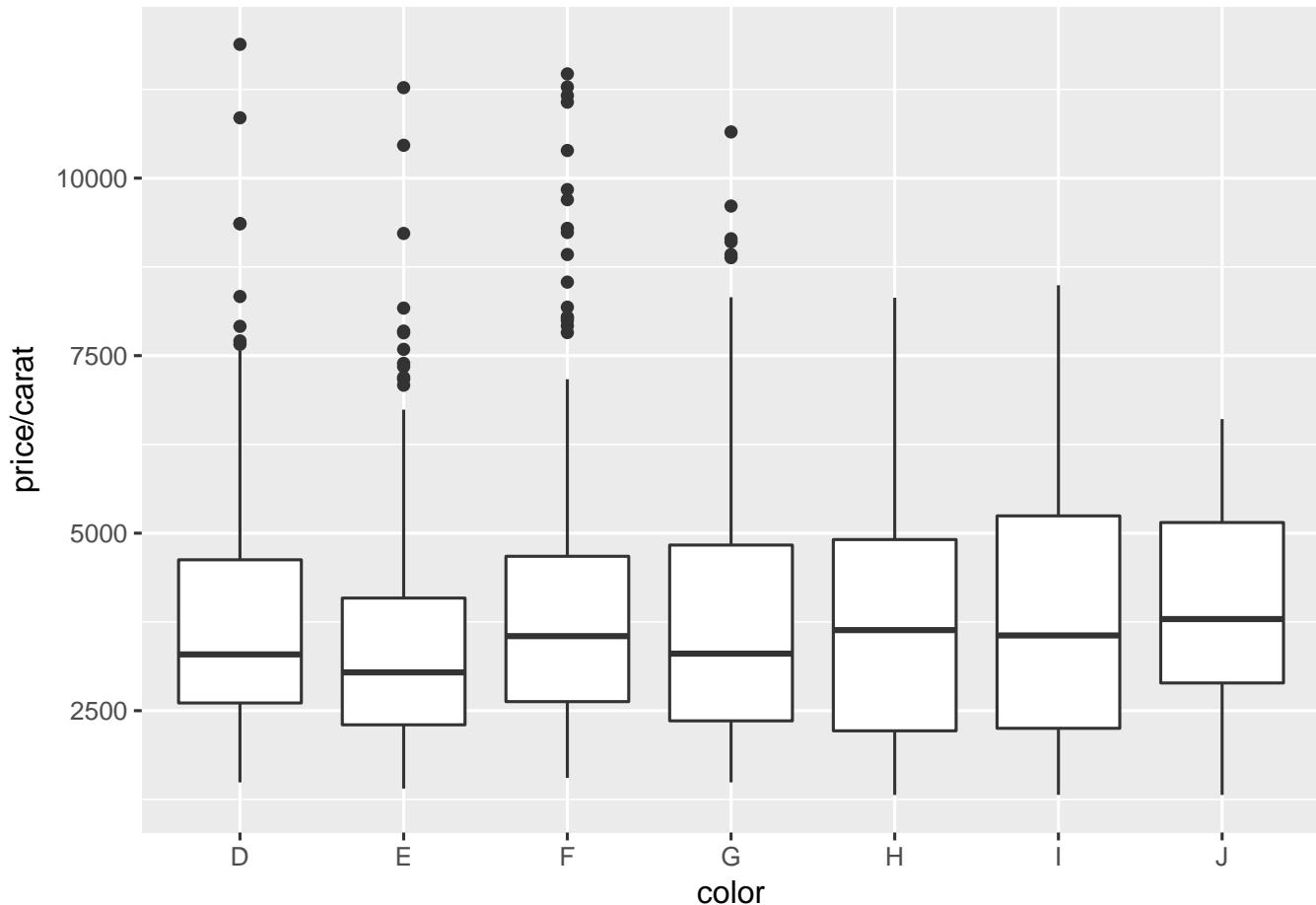
```
# 拟合曲线默认的方法为 method="loess" , 程序会根据数据点的多少自动选取 ,  
# 曲线周围的灰色部分为标准误 , 可以用 se=FALSE 曲线
```



绘制其他常见图形

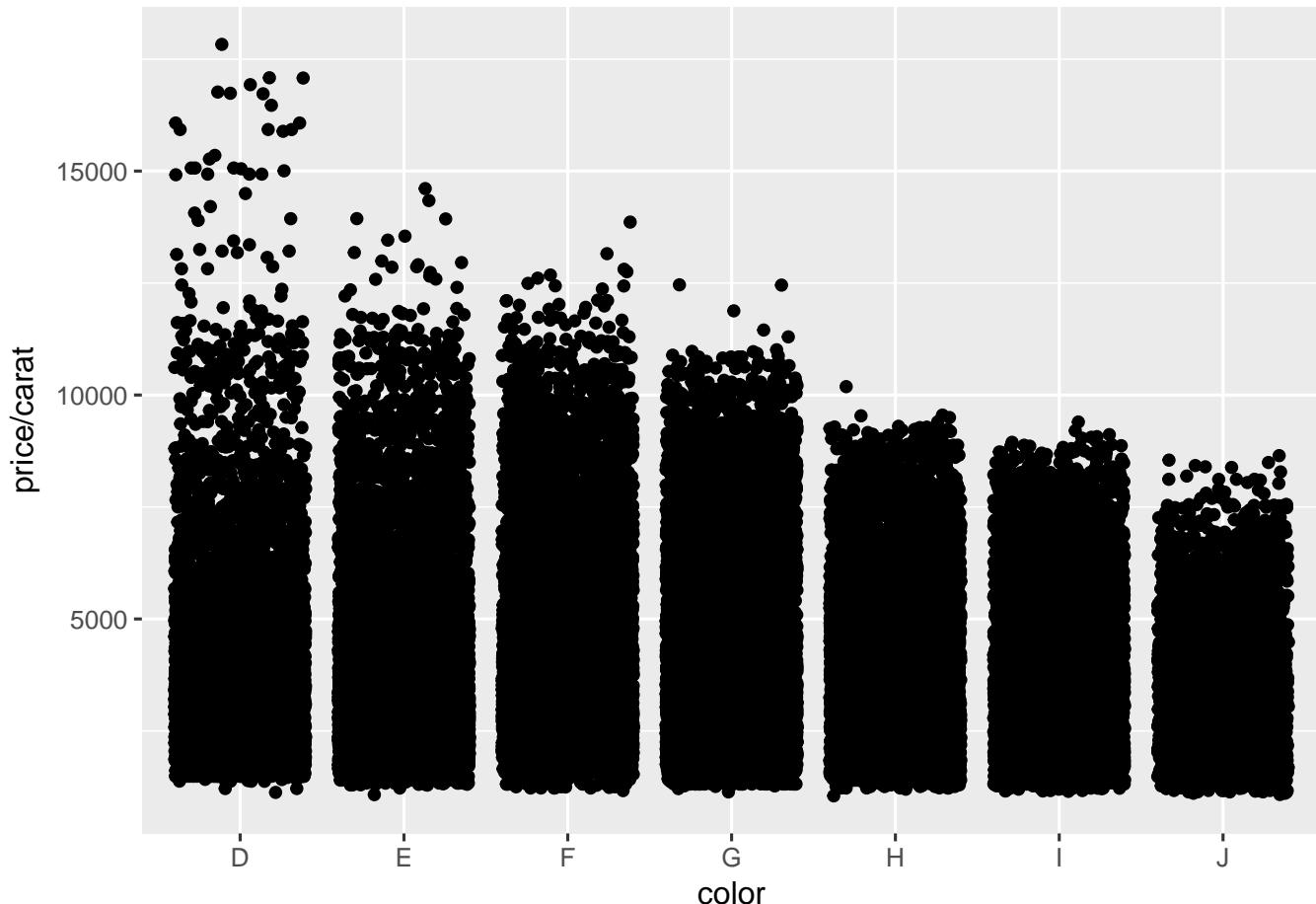
1. 箱线图

```
qplot(color, price/carat, data=dat, geom="boxplot")
```



2. 绕动图(抖动图)

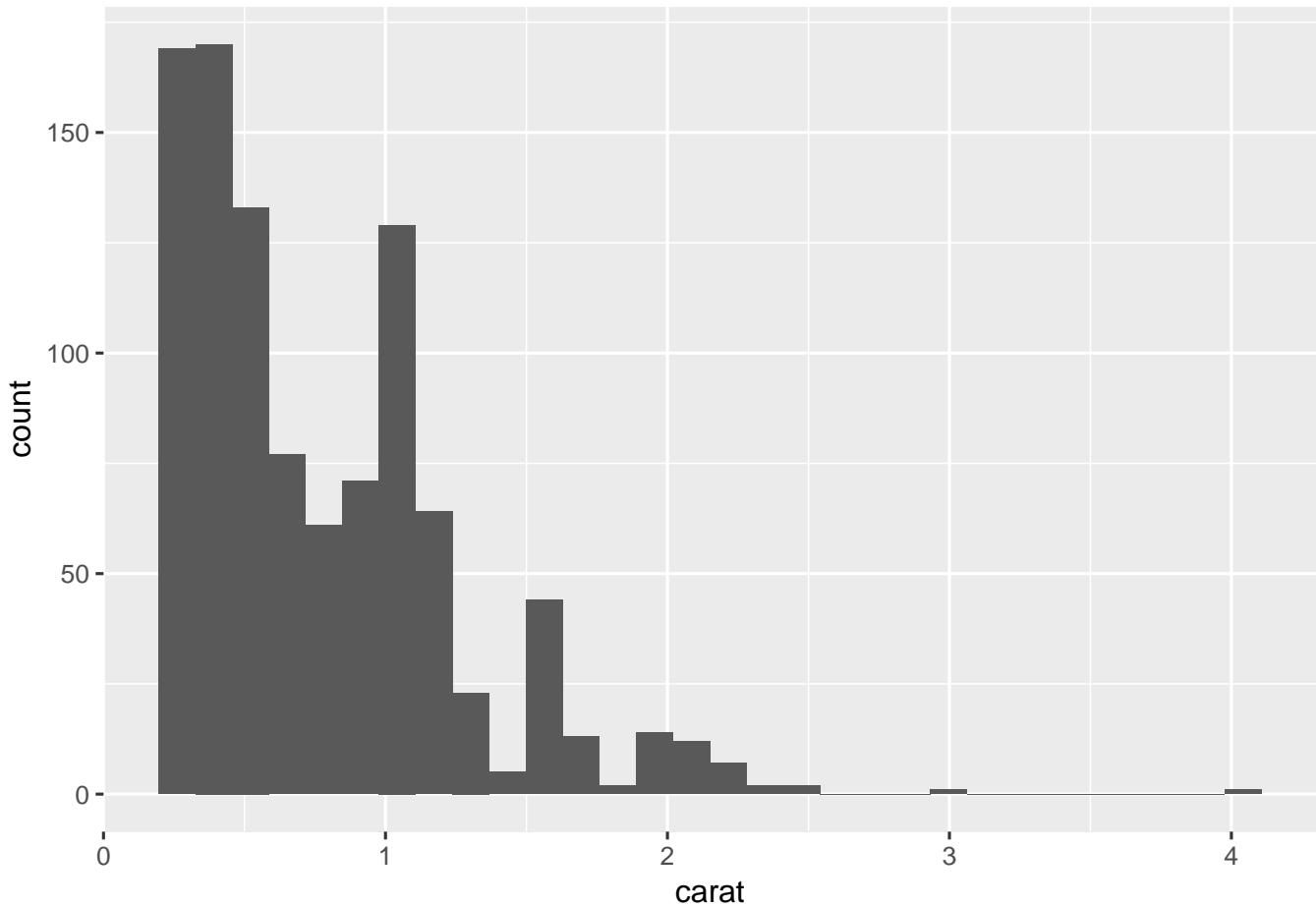
```
qplot(color,price/carat,data=diamonds,geom="jitter")
```



3. 直方图

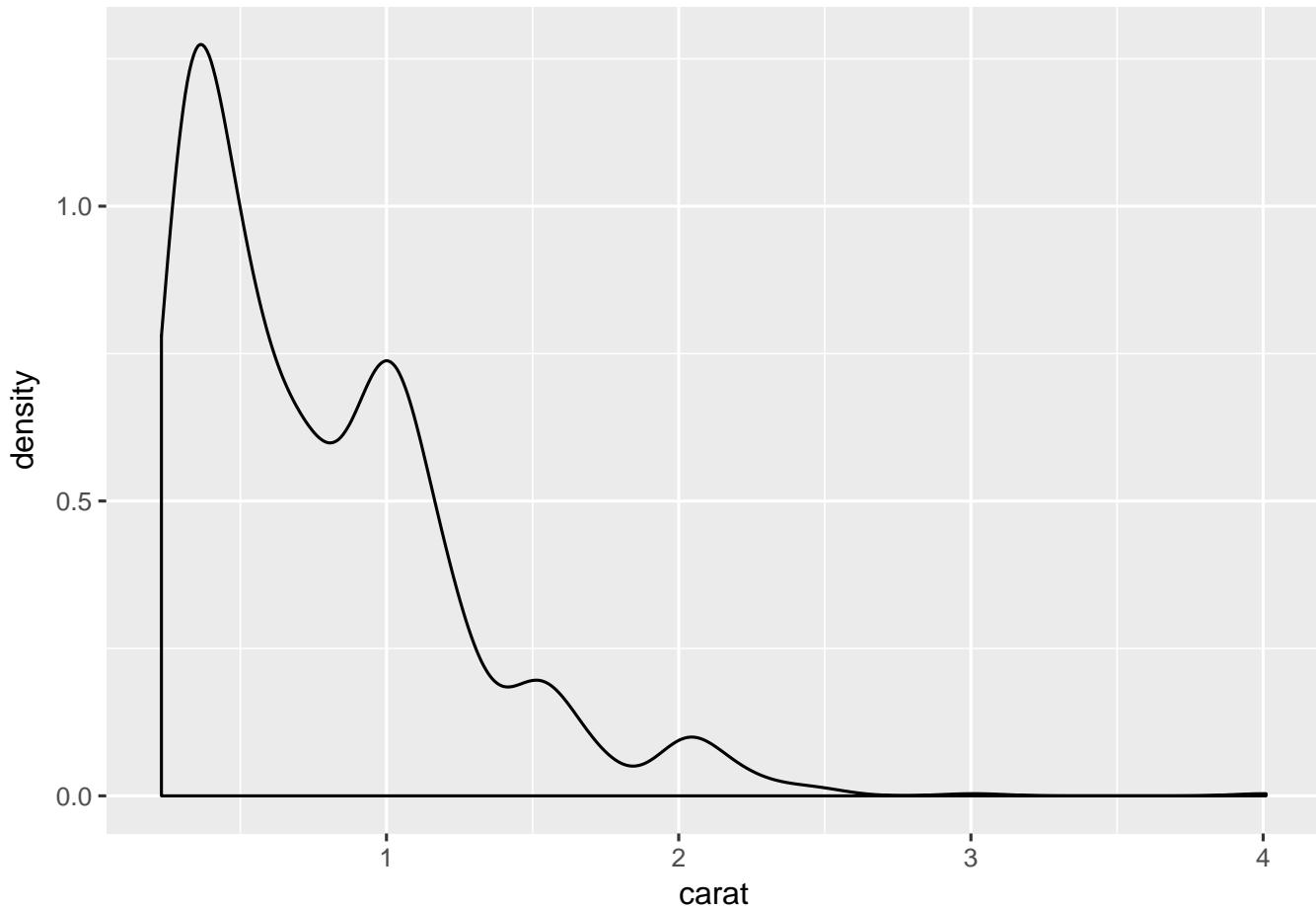
```
qplot(carat, data=dat, geom="histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



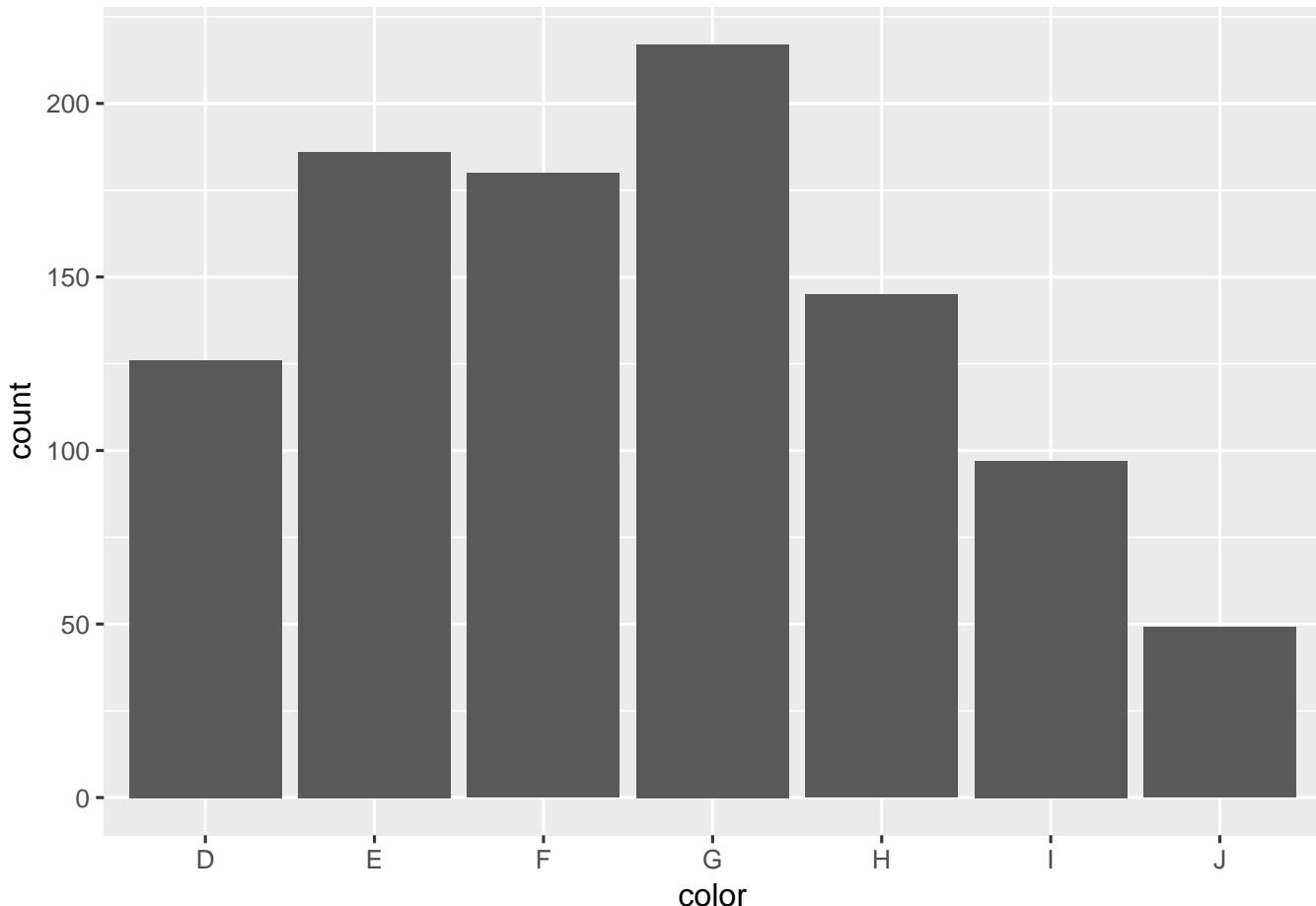
4. 密度曲线图

```
qplot(carat,data=dat,geom="density")
```



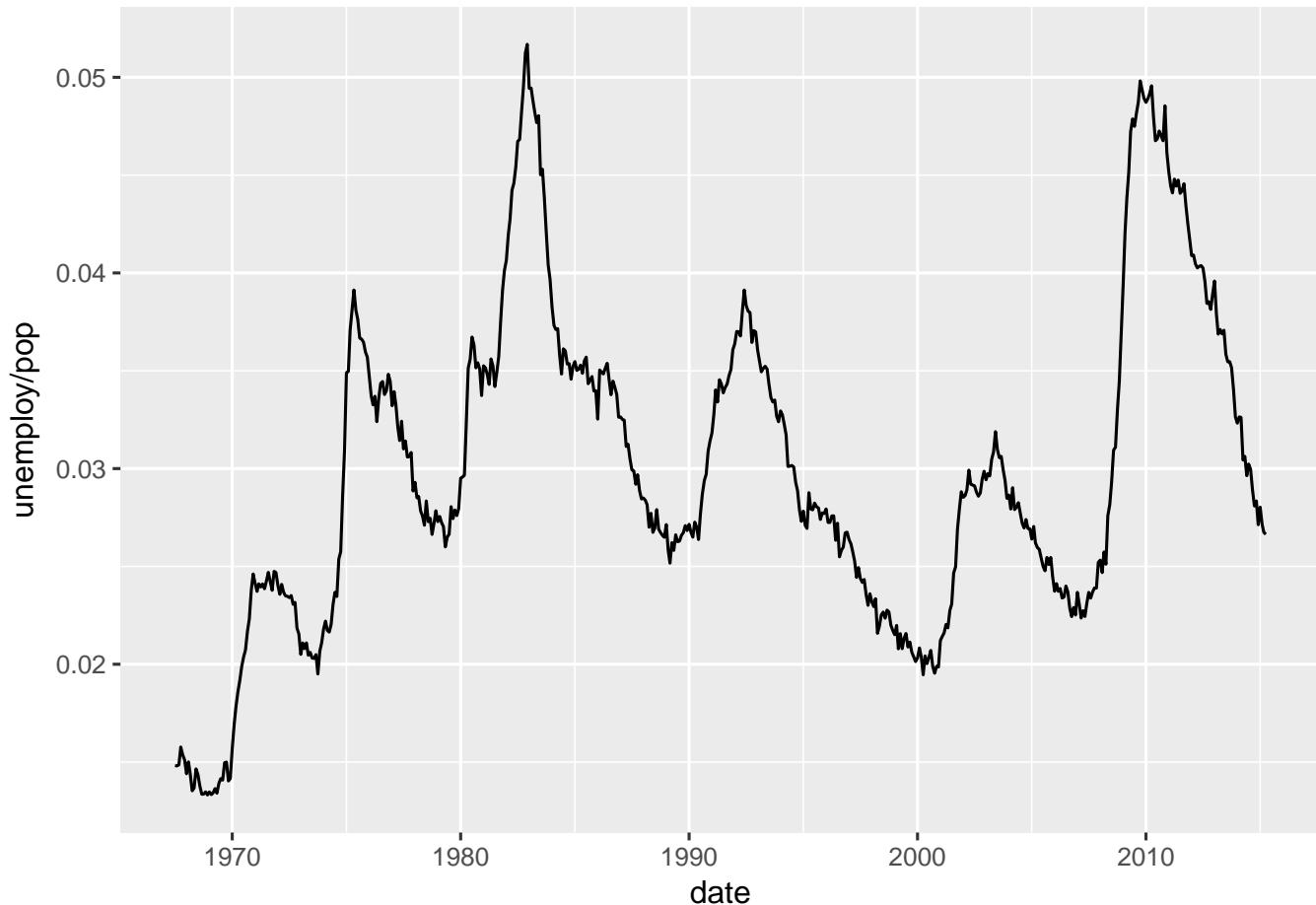
5. 条形图

```
qplot(color, data=dat, geom="bar")
```



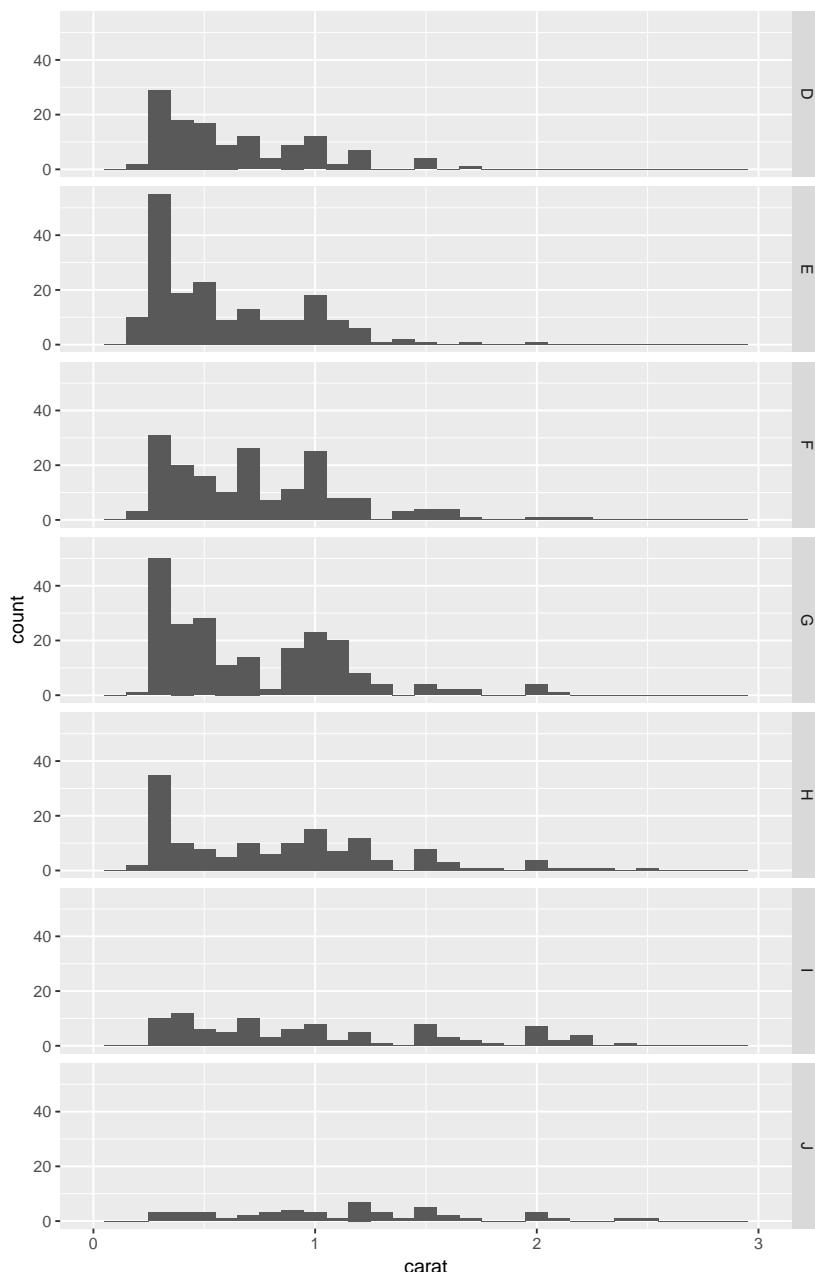
6. 时间序列线条图, 采用另一个数据集 economics

```
qplot(date,unemploy/pop,data=economics,geom="line")
```



7. 分面绘图 facet= 分类变量

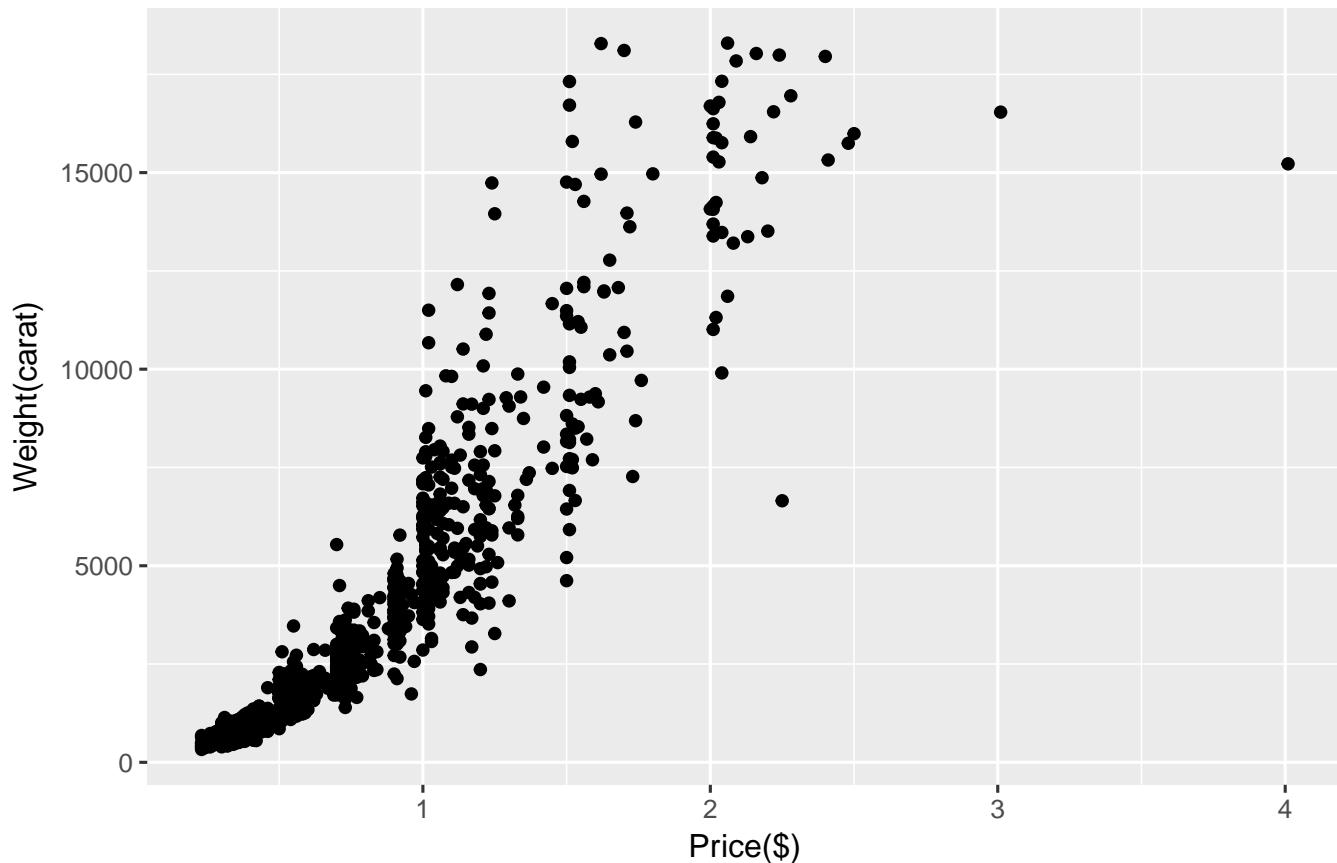
```
qplot(carat,data=dat,facets=color ~ .,geom="histogram",binwidth=0.1,xlim=c(0,3))
```



以上为 ggplot2 包中常见图形的快速绘制 (quickplot) 即 qplot 函数的应用。qplot 函数还有很多其他的参数, 对 xlim, ylim 设置 x,y 轴的区间例如 xlim=c(0,20); 对轴取 log 值, log="x", 对 x 轴取对数, log="xy" 表示对 x 轴和 y 轴取对数; main : 图形的主题 main="qplot title"; #xlab,ylab: 设置轴标签文字

```
qplot(carat,price,data=dat,  
      xlab="Price ($)",ylab="Weight (carat)",  
      main="Price-Weight relationship")
```

Price–Weight relationship



3.2 热图绘制

热图是做分析时常用的展示方式，简单、直观、清晰。可以用来显示基因在不同样品中表达的高低、表观修饰水平的高低、样品之间的相关性等。任何一个数值矩阵都可以通过合适的方式用热图展示。

本篇使用 R 的 `ggplot2` 包实现从原始数据读入到热图输出的过程，并在教程结束后提供一份封装好的命令行绘图工具，只需要提供矩阵，即可一键绘图。

上一篇讲述了 Rstudio 的使用作为 R 写作和编译环境的入门，后面的命令都可以拷贝到 Rstudio 中运行，或写成一个 R 脚本，使用 `Rscript heatmap.r` 运行。我们还提供了 Bash 的封装，在不修改 R 脚本的情况下，改变参数绘制出不同的图形。

3.2.1 生成测试数据

绘图首先需要数据。通过生成几组向量，转换为矩阵，得到想要的数据。

```
data <- c(1:6, 6:1, 6:1, 1:6, (6:1)/10, (1:6)/10, (1:6)/10, (6:1)/10, 1:6, 6:1, 6:1, 1:6,
       6:1, 1:6, 1:6, 6:1)
data

## [1] 1.0 2.0 3.0 4.0 5.0 6.0 6.0 5.0 4.0 3.0 2.0 1.0 6.0 5.0 4.0 3.0 2.0
## [18] 1.0 1.0 2.0 3.0 4.0 5.0 6.0 0.6 0.5 0.4 0.3 0.2 0.1 0.1 0.2 0.3 0.4
## [35] 0.5 0.6 0.1 0.2 0.3 0.4 0.5 0.6 0.6 0.5 0.4 0.3 0.2 0.1 1.0 2.0 3.0
## [52] 4.0 5.0 6.0 6.0 5.0 4.0 3.0 2.0 1.0 6.0 5.0 4.0 3.0 2.0 1.0 1.0 2.0
## [69] 3.0 4.0 5.0 6.0 6.0 5.0 4.0 3.0 2.0 1.0 1.0 2.0 3.0 4.0 5.0 6.0 1.0
## [86] 2.0 3.0 4.0 5.0 6.0 5.0 4.0 3.0 2.0 1.0
```

注意：运算符的优先级。

`1:3+4`

```
## [1] 5 6 7
```

`(1:3)+4`

```
## [1] 5 6 7
```

`1:(3+4)`

```
## [1] 1 2 3 4 5 6 7
```

Vector 转为矩阵 (matrix)，再转为数据框 (data.frame)。

```
# ncol: 指定列数
# byrow: 先按行填充数据
# ?matrix 可查看函数的使用方法
# as.data.frame 的 as 系列是转换用的
data <- as.data.frame(matrix(data, ncol=12, byrow=T))
data
```

```
##      V1   V2   V3   V4   V5   V6   V7   V8   V9   V10  V11  V12
## 1 1.0 2.0 3.0 4.0 5.0 6.0 6.0 5.0 4.0 3.0 2.0 1.0
## 2 6.0 5.0 4.0 3.0 2.0 1.0 1.0 2.0 3.0 4.0 5.0 6.0
## 3 0.6 0.5 0.4 0.3 0.2 0.1 0.1 0.2 0.3 0.4 0.5 0.6
## 4 0.1 0.2 0.3 0.4 0.5 0.6 0.6 0.5 0.4 0.3 0.2 0.1
```

```

## 5 1.0 2.0 3.0 4.0 5.0 6.0 6.0 5.0 4.0 3.0 2.0 1.0
## 6 6.0 5.0 4.0 3.0 2.0 1.0 1.0 2.0 3.0 4.0 5.0 6.0
## 7 6.0 5.0 4.0 3.0 2.0 1.0 1.0 2.0 3.0 4.0 5.0 6.0
## 8 1.0 2.0 3.0 4.0 5.0 6.0 6.0 5.0 4.0 3.0 2.0 1.0

# 增加列的名字
colnames(data) <- c("Zygote", "2_cell", "4_cell", "8_cell", "Morula", "ICM", "ESC",
                      "4 week PGC", "7 week PGC", "10 week PGC", "17 week PGC", "Oocyte")

# 增加行的名字
# 注意 paste 和 paste0 的使用
rownames(data) <- paste("Gene", 1:8, sep="_")

# 只显示前 6 行和前 4 列
head(data) [,1:4]

```

```

##          Zygote 2_cell 4_cell 8_cell
## Gene_1     1.0    2.0    3.0    4.0
## Gene_2     6.0    5.0    4.0    3.0
## Gene_3     0.6    0.5    0.4    0.3
## Gene_4     0.1    0.2    0.3    0.4
## Gene_5     1.0    2.0    3.0    4.0
## Gene_6     6.0    5.0    4.0    3.0

```

虽然方法比较繁琐，但一个数值矩阵已经获得了。

还有另外 2 种获取数值矩阵的方式。

- 读入字符串

```

# 使用字符串的好处是不需要额外提供文件
# 简单测试时可使用，写起来不繁琐，又方便重复
# 尤其适用于在线提问时作为测试案例
txt <- "ID;Zygote;2_cell;4_cell;8_cell
+ Gene_1;1;2;3;4
+ Gene_2;6;5;4;5
+ Gene_3;0.6;0.5;0.4;0.4"

# 习惯设置 quote 为空，避免部分基因名字或注释中存在引号，导致读入文件错误。
# 具体错误可查看 http://blog.genesino.com/collections/R\_tips/ 中的记录
data2 <- read.table(text=txt, sep=";", header=T, row.names=1, quote="")
head(data2)

```

```
##           Zygote X2_cell X4_cell X8_cell
## + Gene_1      1.0     2.0     3.0     4.0
## + Gene_2      6.0     5.0     4.0     5.0
## + Gene_3      0.6     0.5     0.4     0.4
```

可以看到列名字中以数字开头的列都加了 **X**。一般要尽量避免行或列名字以数字开头，会给后续分析带来匹配问题；另外名字中出现的非字母、数字、下划线、点的字符都会被转为点，也需要注意，尽量只用字母、下划线和数字。

```
# 读入时，增加一个参数 `check.names=F` 也可以解决问题。
# 这次数字前没有再加 X 了
data2 <- read.table(text=txt, sep=";", header=T, row.names=1, quote="", check.names = F)
head(data2)
```

```
##           Zygote 2_cell 4_cell 8_cell
## + Gene_1      1.0     2.0     3.0     4.0
## + Gene_2      6.0     5.0     4.0     5.0
## + Gene_3      0.6     0.5     0.4     0.4
```

- 读入文件

与上一步类似，只是把 `txt` 代表的文字存到文件中，再利用文件名读取，不再赘述。

```
#data2 <- read.table("filename", sep=";", header=T, row.names=1, quote="")
```

3.2.2 转换数据格式

数据读入后，还需要一步格式转换。在使用 `ggplot2` 作图时，有一种长表格模式是最常用的，尤其是数据不规则时，更应该使用（这点，我们在讲解箱线图时再说）。

`melt`：把正常矩阵转换为长表格模式的函数。工作原理是把全部的非 `id` 列的数值列转为 1 列（列名默认为 `value`）；所有字符列转为一列，列名默认为 `variable`。

```
# 如果包没有安装，运行下面一句，安装包
#install.packages(c("reshape2", "ggplot2"))

library(reshape2)
library(ggplot2)

# 转换前，先增加一列 ID 列，保存行名字
data$ID <- rownames(data)
```

```
# id.vars 列用于指定哪些列为 id 列；这些列不会被 merge，会保留为完整一列。
```

```
data_m <- melt(data, id.vars=c("ID"))
head(data_m)
```

```
##           ID variable value
## 1 Gene_1    Zygote   1.0
## 2 Gene_2    Zygote   6.0
## 3 Gene_3    Zygote   0.6
## 4 Gene_4    Zygote   0.1
## 5 Gene_5    Zygote   1.0
## 6 Gene_6    Zygote   6.0
```

3.2.3 分解绘图

数据转换后就可以画图了，分解命令如下：

```
# data_m: 是前面费了九牛二虎之力得到的数据表
# aes: aesthetic 的缩写，一般指定整体的 X 轴、Y 轴、颜色、形状、大小等。
#       在最开始读入数据时，一般只指定 x 和 y，其它后续指定
p <- ggplot(data_m, aes(x=variable, y=ID))

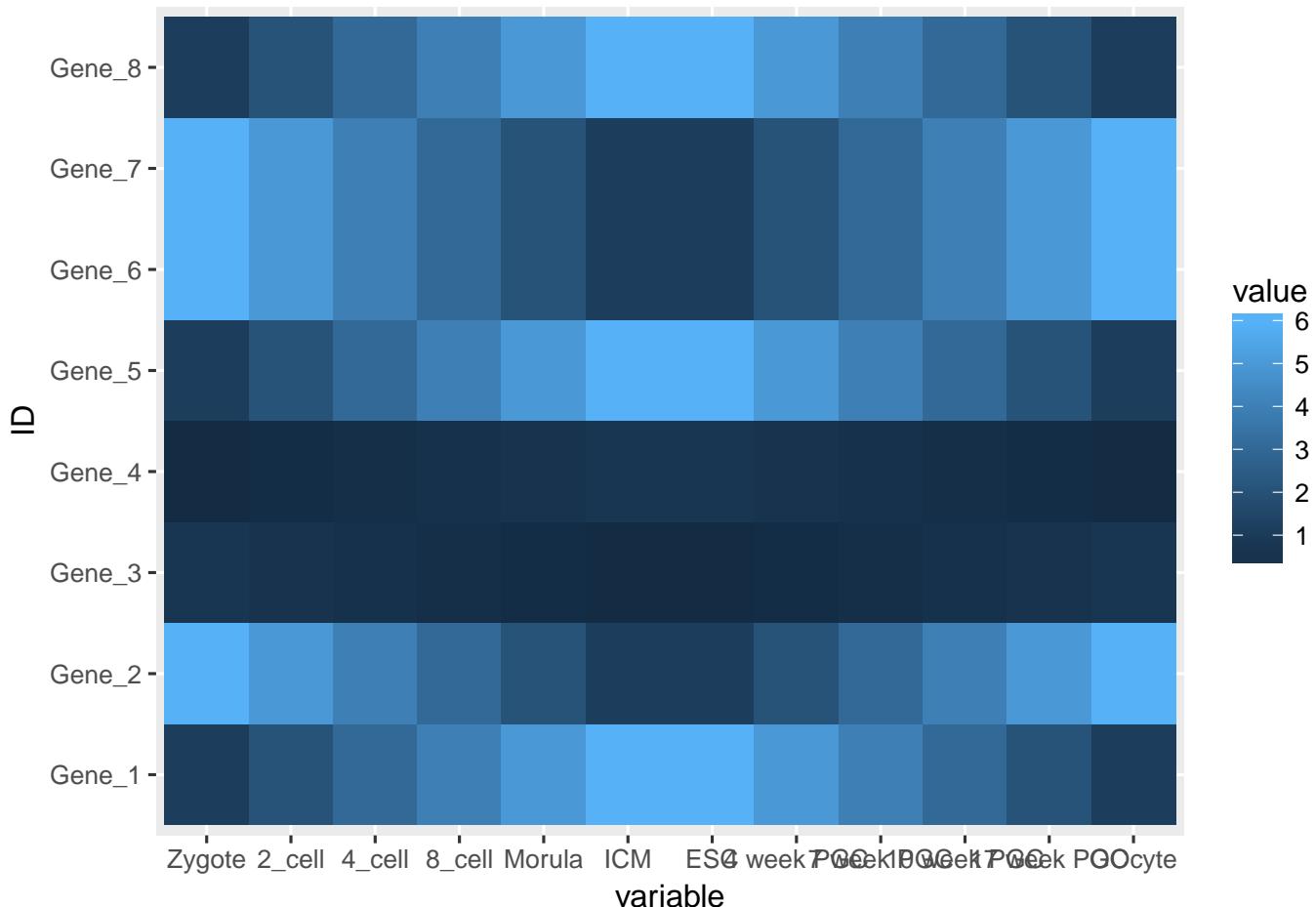
# 热图就是一堆方块根据其值赋予不同的颜色，所以这里使用 fill=value，用数值做填充色。
p <- p + geom_tile(aes(fill=value))

# ggplot2 为图层绘制，一层层添加，存储在 p 中，在输出 p 的内容时才会出图。
p

## 如果你没有使用 Rstudio 或其它 R 图形版工具，而是在远程登录的服务器上运行的交互式 R，
## 需要输入下面的语句，获得输出图形（图形存储于 R 的工作目录下的 Rplots.pdf 文件中）。

## 如何指定输出，后面会讲到。
#dev.off()
```

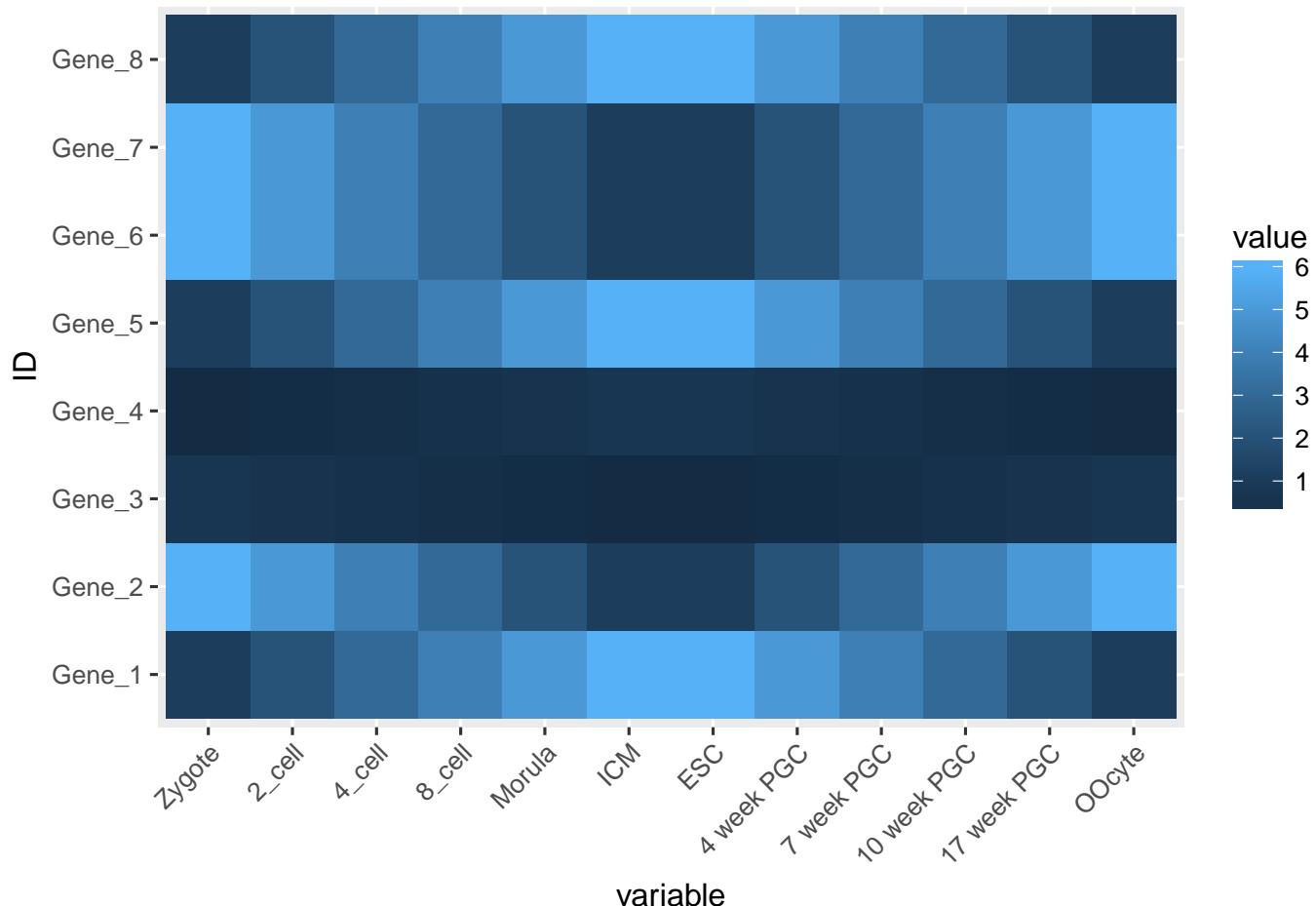
CONTENTS



热图出来了，但有点不对劲，横轴重叠一起了。一个办法是调整图像的宽度，另一个是旋转横轴标记。

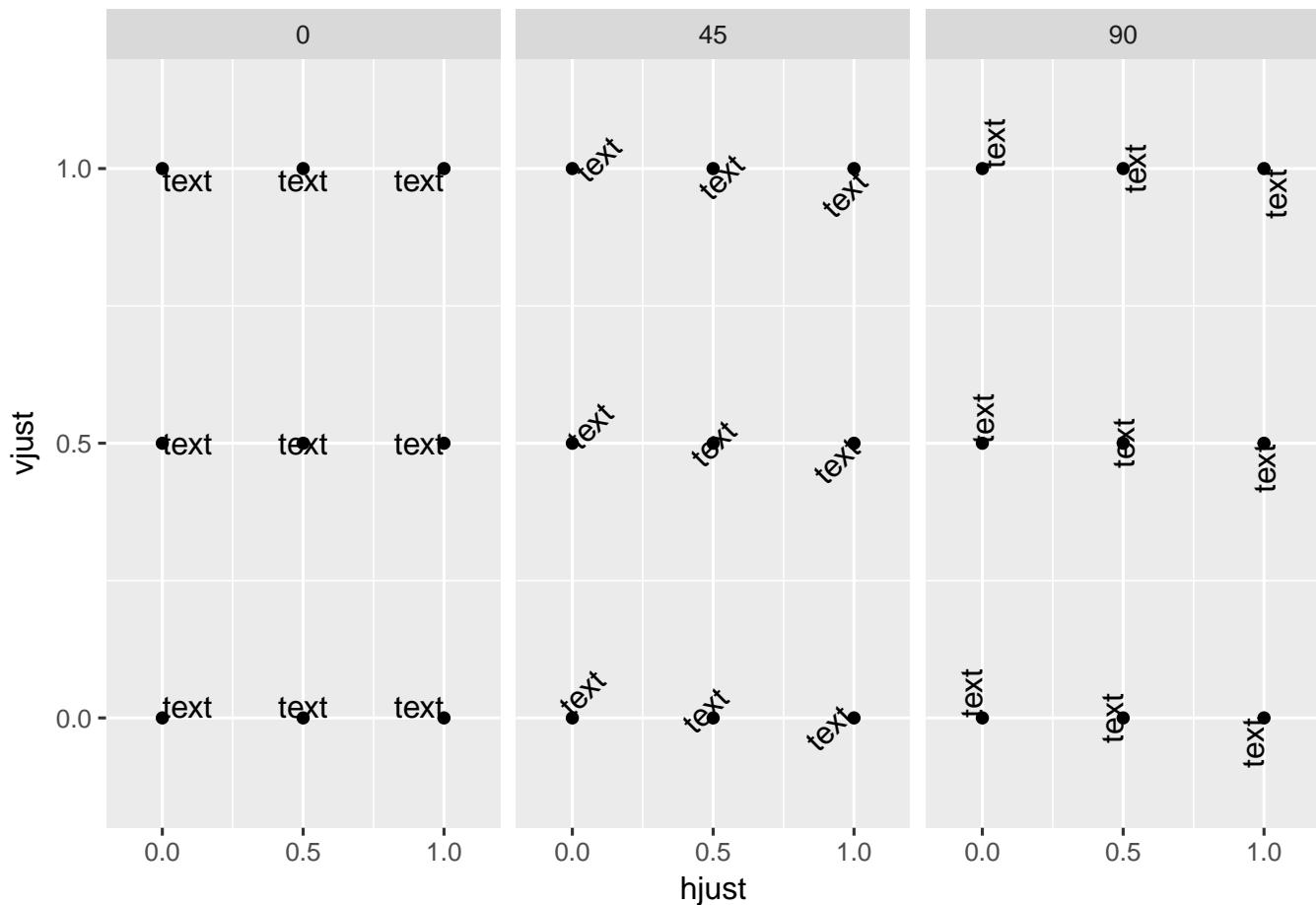
```
# theme: 是处理图美观的一个函数，可以调整横纵轴 label 的选择、图例的位置等。
# 这里选择 x 轴标签 45 度。
# hjust 和 vjust 调整标签的相对位置，
# 具体见下图。
# 简单说，hjust 是水平的对齐方式，0 为左，1 为右，0.5 居中，0-1 之间可以取任意值。
# vjust 是垂直对齐方式，0 底对齐，1 为顶对齐，0.5 居中，0-1 之间可以取任意值。

p <- p + theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1))
p
```



```
#knitr:::include_graphics("images/hjust_vjust.png")
td <- expand.grid(
  hjust=c(0, 0.5, 1),
  vjust=c(0, 0.5, 1),
  angle=c(0, 45, 90),
  text="text"
)

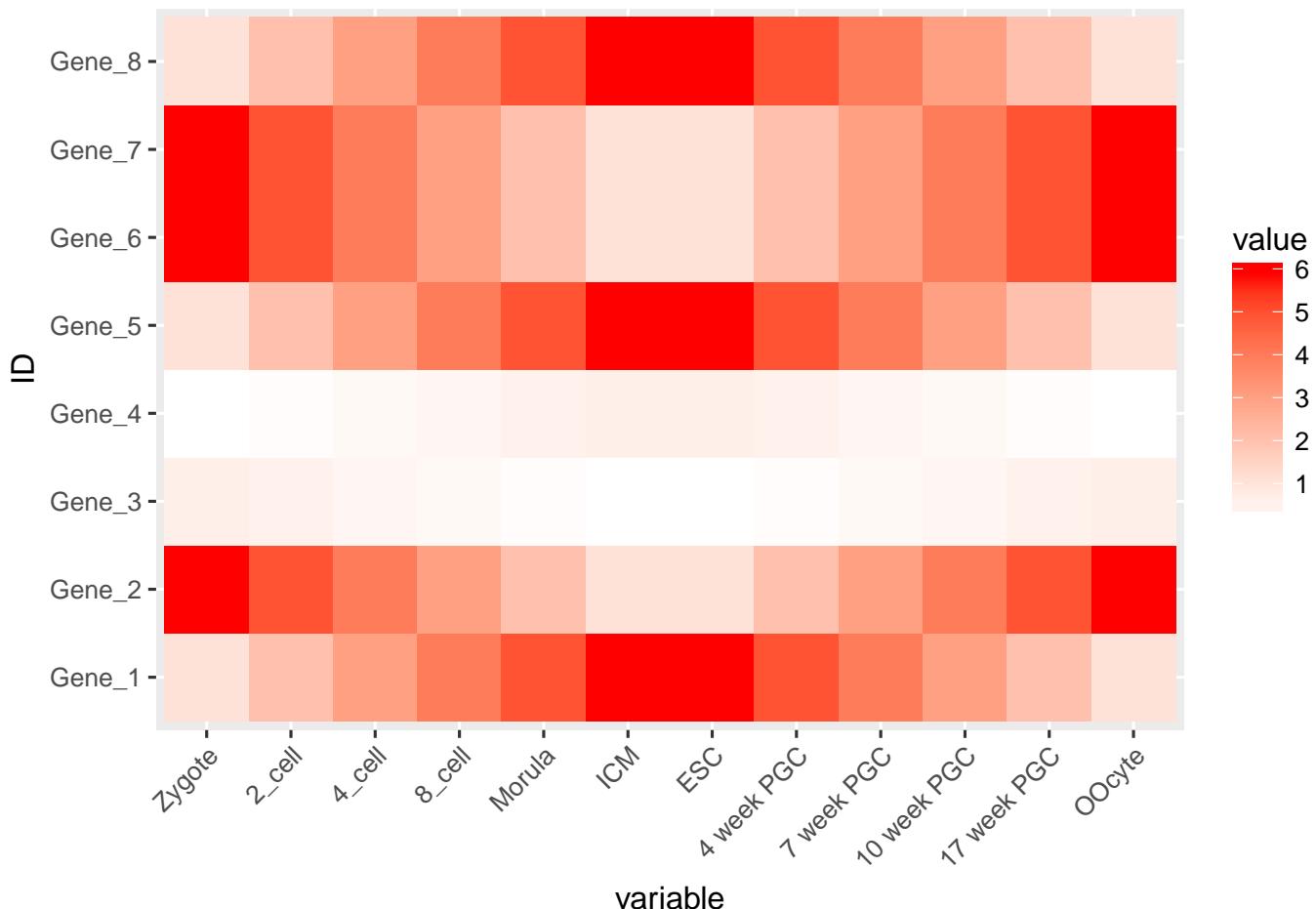
ggplot(td, aes(x=hjust, y=vjust)) +
  geom_point() +
  geom_text(aes(label=text, angle=angle, hjust=hjust, vjust=vjust)) +
  facet_grid(~angle) +
  scale_x_continuous(breaks=c(0, 0.5, 1), expand=c(0, 0.2)) +
  scale_y_continuous(breaks=c(0, 0.5, 1), expand=c(0, 0.2))
```



<http://stackoverflow.com/questions/7263849/what-do-hjust-and-vjust-do-when-making-a-plot-using-ggplot>

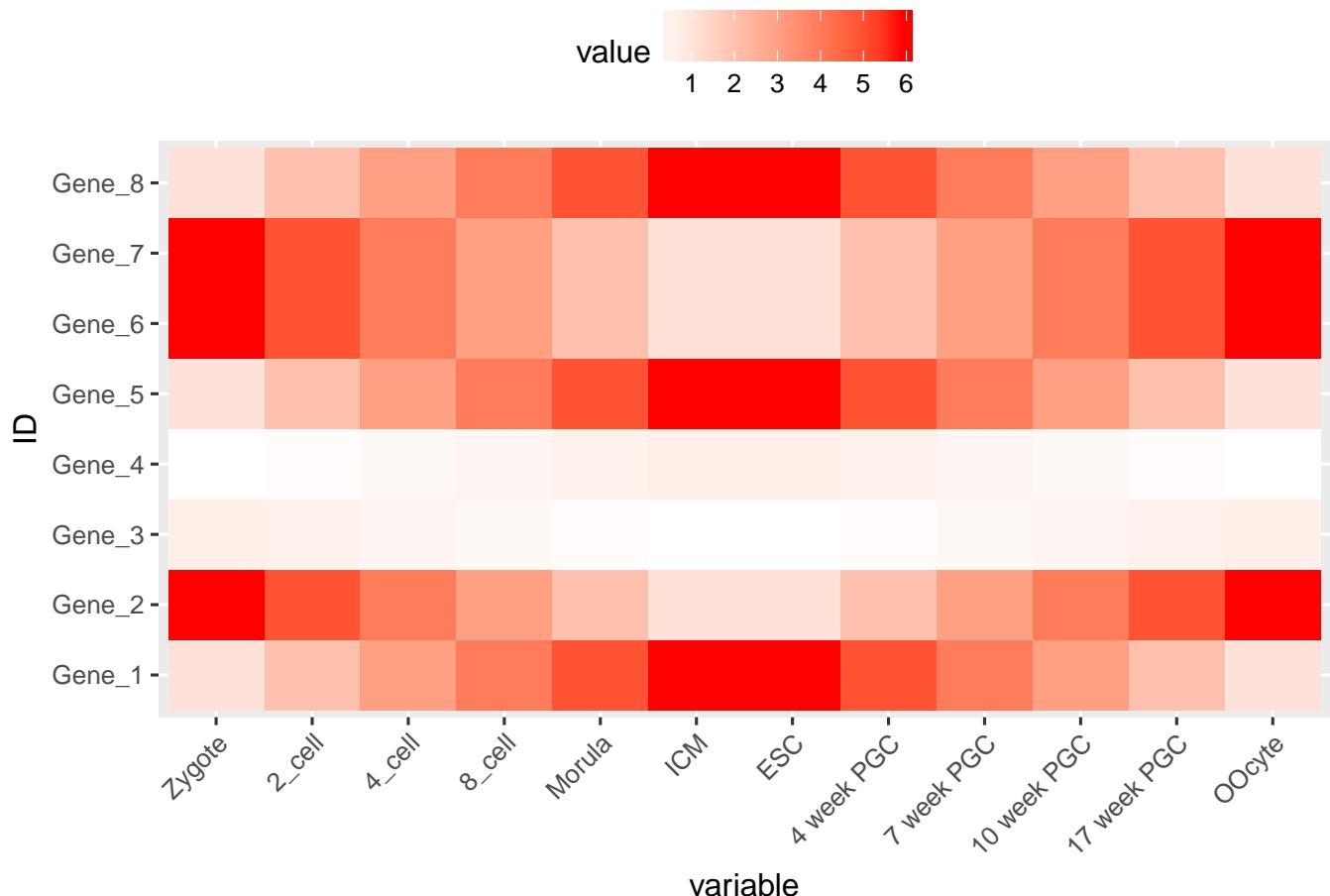
设置想要的颜色。

```
# 连续的数字，指定最小数值代表的颜色和最大数值赋予的颜色
# 注意 fill 和 color 的区别，fill 是填充，color 只针对边缘
p <- p + scale_fill_gradient(low = "white", high = "red")
p
```



调整 legend 的位置, legend.position, 可以接受的值有 top, bottom, left, right, 和一个坐标 c(0.05, 0.8) (左上角, 坐标是相对于图的左下角 (即原点) 计算的)

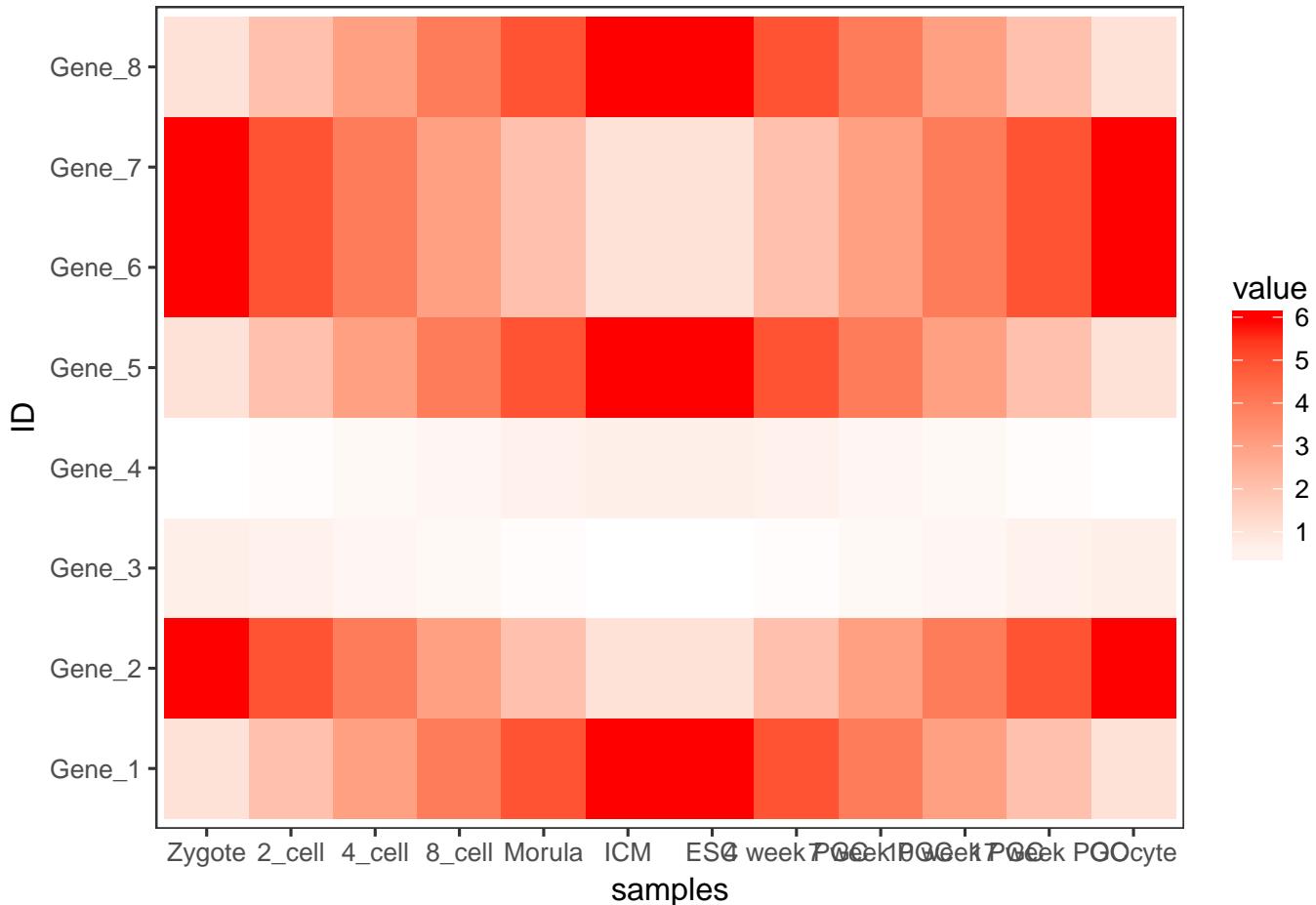
```
p <- p + theme(legend.position="top")
p
```



调整背景和背景格线以及 X 轴、Y 轴的标题。

```
p <- p + xlab("samples") + theme_bw() + theme(panel.grid.major = element_blank()) +
  theme(legend.key=element_blank())
p
```

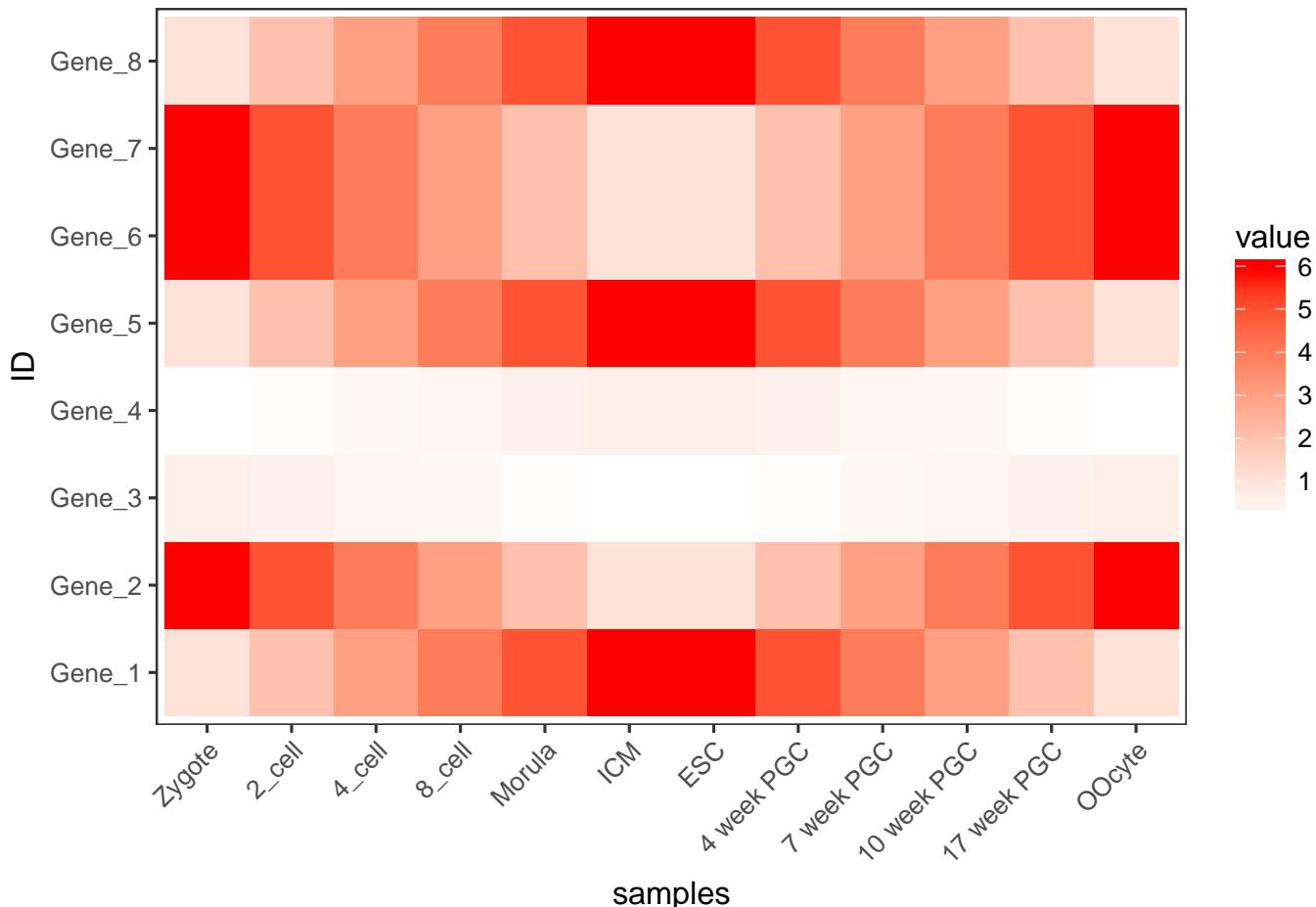
CONTENTS



为了使横轴旋转 45 度，需要把这句话 `theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1))` 放在 `theme_bw()` 的后面。

```
p <- p + theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1))
p
```

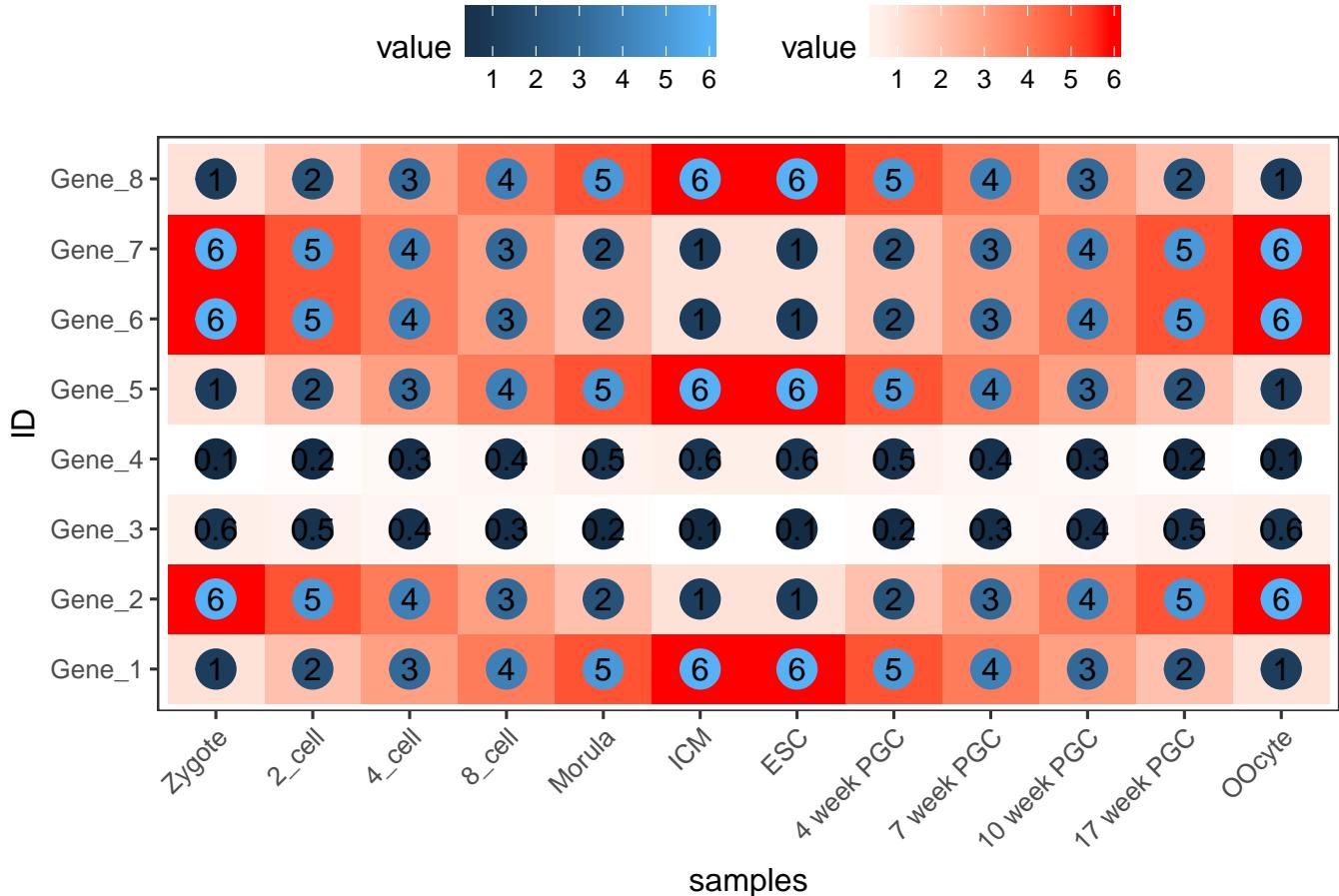
CONTENTS



合并以上命令，就得到了下面这个看似复杂的绘图命令。

```
p <- ggplot(data_m, aes(x=variable, y=ID)) + xlab("samples") + theme_bw() +
  theme(panel.grid.major = element_blank()) + theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1)) +
  theme(legend.position="top") + geom_tile(aes(fill=value)) +
  scale_fill_gradient(low = "white", high = "red") +
  geom_point(aes(color=value), size=6) +
  geom_text(aes(label=value))
```

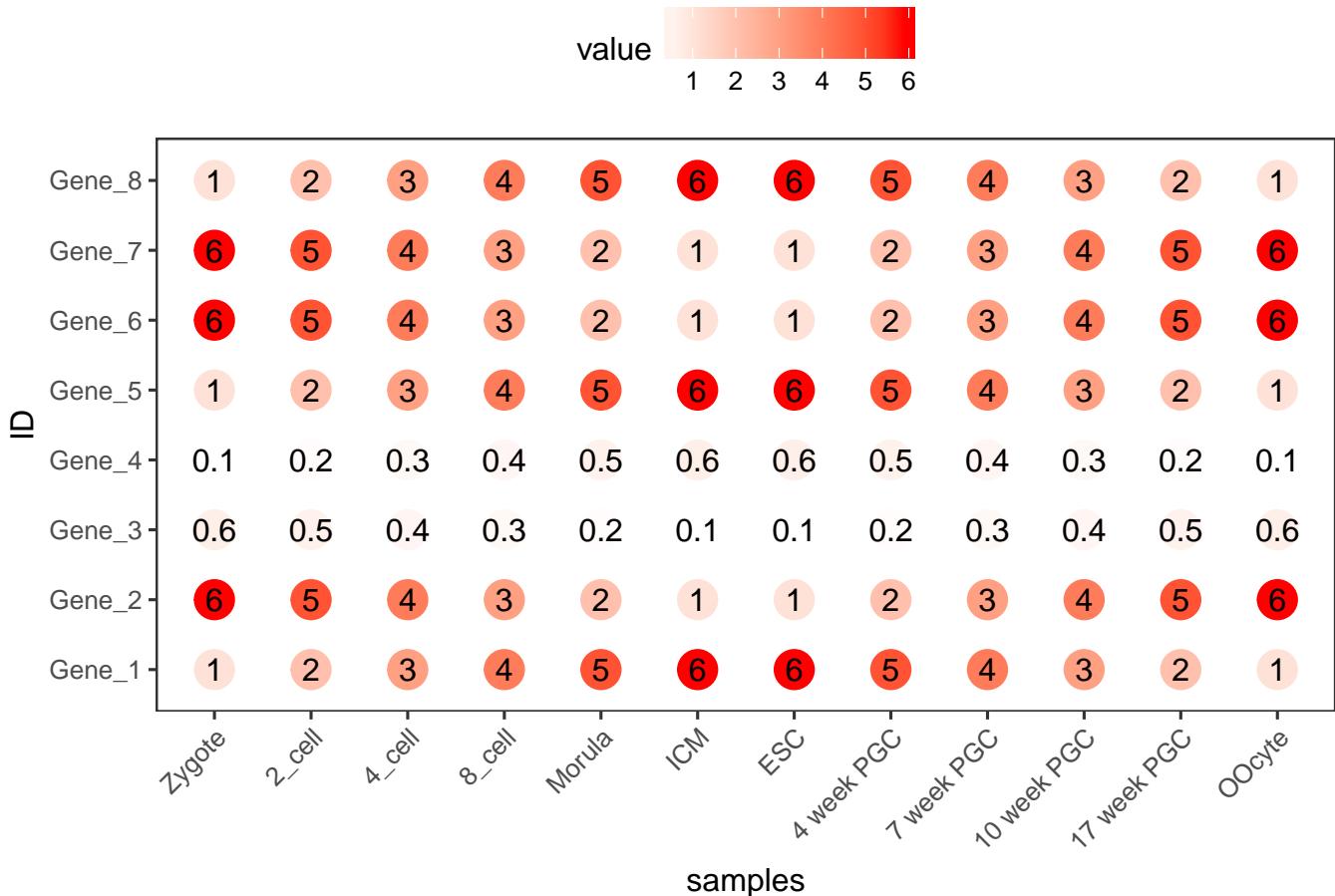
p



也可以只用 Point

```
p <- ggplot(data_m, aes(x=variable, y=ID)) + xlab("samples") + theme_bw() +
  theme(panel.grid.major = element_blank()) + theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1)) +
  theme(legend.position="top") +
  geom_point(aes(color=value), size=6) +
  scale_color_gradient(low = "white", high = "red") +
  geom_text(aes(label=value))
```

p



3.2.4 图形存储

图形出来了，就得考虑存储了，一般输出为 PDF 格式，方便后期的修改。

```
# 可以跟输出文件不同的后缀，以获得不同的输出格式
# colormode 支持 srgb (屏幕) 和 cmyk (打印，部分杂志需要，看上去有点褪色的感觉) 格式
ggsave(p, filename="heatmap.pdf", width=10,
       height=15, units=c("cm"), colormodel="srgb")
```

点击下载：[pdf](#)

至此，完成了简单的 heatmap 的绘图。但实际绘制时，经常会碰到由于数值变化很大，导致颜色过于集中，使得图的可读性下降很多。因此需要对数据进行一些处理，具体的下次再说。

3.3 热图美化

上面的测试数据，数值的分布比较均一，相差不是太大，但是 Gene_4 和 Gene_5 由于整体的值低于其它的基因，从颜色上看，不仔细看，看不出差别。

```

data <- c(rnorm(5,mean=5), rnorm(5,mean=20), rnorm(5, mean=100), c(600,700,800,900,10000))
data <- matrix(data, ncol=5, byrow=T)
data <- as.data.frame(data)
rownames(data) <- letters[1:4]
colnames(data) <- paste("Grp", 1:5, sep="_")
data

##          Grp_1      Grp_2      Grp_3      Grp_4      Grp_5
## a    6.087539   5.245061   7.075796   4.820246   4.374214
## b   18.391807  18.963660  20.210934  19.701757  21.620189
## c  100.005960  99.495593  99.766925  99.985531  100.411950
## d 600.000000 700.000000 800.000000 900.000000 10000.000000

data$ID <- rownames(data)
data

##          Grp_1      Grp_2      Grp_3      Grp_4      Grp_5 ID
## a    6.087539   5.245061   7.075796   4.820246   4.374214 a
## b   18.391807  18.963660  20.210934  19.701757  21.620189 b
## c  100.005960  99.495593  99.766925  99.985531  100.411950 c
## d 600.000000 700.000000 800.000000 900.000000 10000.000000 d

data_m <- melt(data, id.vars=c("ID"))
head(data_m)

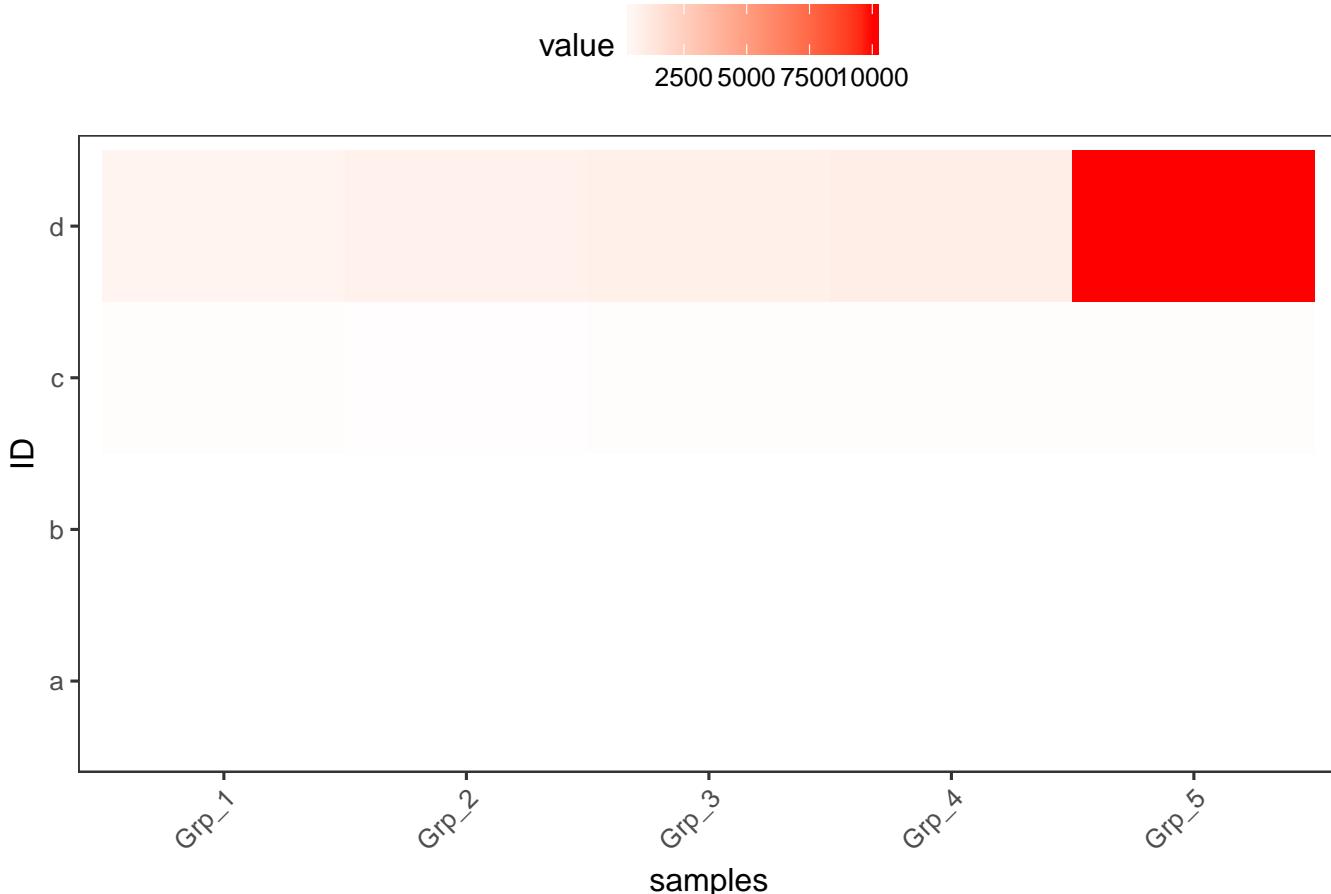
##   ID variable     value
## 1 a     Grp_1 6.087539
## 2 b     Grp_1 18.391807
## 3 c     Grp_1 100.005960
## 4 d     Grp_1 600.000000
## 5 a     Grp_2 5.245061
## 6 b     Grp_2 18.963660

p <- ggplot(data_m, aes(x=variable,y=ID)) + xlab("samples") + theme_bw() +
  theme(panel.grid.major = element_blank()) + theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1)) +
  theme(legend.position="top") + geom_tile(aes(fill=value)) +
  scale_fill_gradient(low = "white", high = "red")

```

CONTENTS

```
p
#dev.off()
```



图中只有右上角可以看到红色，其他地方就没了颜色的差异。这通常不是我们想要的。为了更好的可视化效果，需要对数据做些预处理，主要有 对数转换，z-score 转换，抹去异常值，非线性颜色等方式。

3.3.1 对数转换

假设下面的数据是基因表达数据，4 个基因 (a, b, c, d) 和 5 个样品 (Grp_1, Grp_2, Grp_3, Grp_4)，矩阵中的值代表基因表达 FPKM 值。

```
data <- c(rnorm(5,mean=5), rnorm(5,mean=20), rnorm(5, mean=100), c(600,700,800,900,10000))
data <- matrix(data, ncol=5, byrow=T)
data <- as.data.frame(data)
rownames(data) <- letters[1:4]
colnames(data) <- paste("Grp", 1:5, sep="_")
data
```

```

##          Grp_1      Grp_2      Grp_3      Grp_4      Grp_5
## a  4.125538  4.087054  3.52572  5.749561  2.420971
## b 21.296682 19.922518 19.41533 20.510619 20.319078
## c 100.188919 99.774166 99.88726 98.414636 100.442783
## d 600.000000 700.000000 800.000000 900.000000 10000.000000

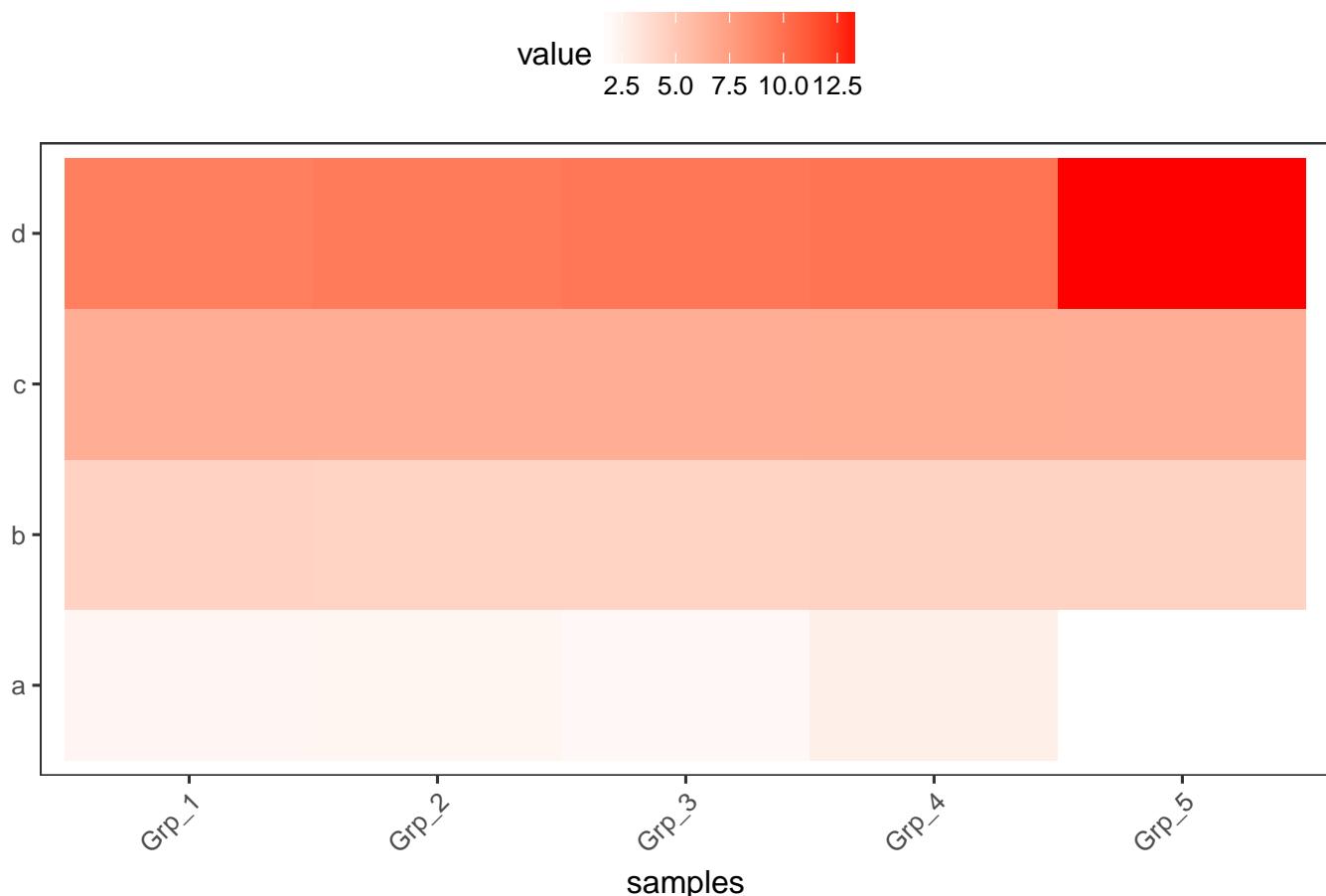
data_log <- log2(data+1)
data_log

##          Grp_1      Grp_2      Grp_3      Grp_4      Grp_5
## a 2.357704 2.346830 2.178147 2.754794 1.774406
## b 4.478757 4.386985 4.351581 4.426977 4.414073
## c 6.660907 6.654982 6.656600 6.635386 6.664522
## d 9.231221 9.453271 9.645658 9.815383 13.287857

data_log$ID = rownames(data_log)
data_log_m = melt(data_log, id.vars=c("ID"))

p <- ggplot(data_log_m, aes(x=variable,y=ID)) + xlab("samples") + ylab(NULL) +
  theme_bw() + theme(panel.grid.major = element_blank()) +
  theme(legend.key=element_blank()) + theme(legend.position="top") +
  theme(axis.text.x=element_text(angle=45,hjust=1,vjust=1)) +
  geom_tile(aes(fill=value)) + scale_fill_gradient(low = "white", high = "red")
p
#ggsave(p, filename="heatmap_log.pdf", width=8, height=12, units=c("cm"), colormodel="srgb")

```



对数转换后的数据，看起来就清晰的多了。而且对数转换后，数据还保留着之前的变化趋势，不只是基因在不同样品之间的表达可比 (同一行的不同列)，不同基因在同一样品的值也可比 (同一列的不同行) (不同基因之间比较表达值存在理论上的问题，即便是按照长度标准化之后的 FPKM 也不代表基因之间是完全可比的)。

3.3.2 Z-score 转换

Z-score 又称为标准分数，是一组数中的每个数减去这一组数的平均值再除以这一组数的标准差，代表的是原始分数距离原始平均值的距离，以标准差为单位。可以对不同分布的各原始分数进行比较，用来反映数据的相对变化趋势，而非绝对变化量。

```

data_ori <- "Grp_1;Grp_2;Grp_3;Grp_4;Grp_5
a;6.6;20.9;100.1;600.0;5.2
b;20.8;99.8;700.0;3.7;19.2
c;100.0;800.0;6.2;21.4;98.6
d;900;3.3;20.3;101.1;10000"

data <- read.table(text=data_ori, header=T, row.names=1, sep=";", quote="")
# 去掉方差为 0 的行，也就是值全都一致的行

```

CONTENTS

```

data <- data[apply(data, 1, var) != 0,]

data

##   Grp_1  Grp_2  Grp_3  Grp_4  Grp_5
## a    6.6   20.9 100.1  600.0     5.2
## b   20.8   99.8 700.0    3.7   19.2
## c 100.0   800.0    6.2   21.4   98.6
## d 900.0    3.3   20.3 101.1 10000.0

# 标准化数据，并转换为 data.frame
data_scale <- as.data.frame(t(apply(data, 1, scale)))

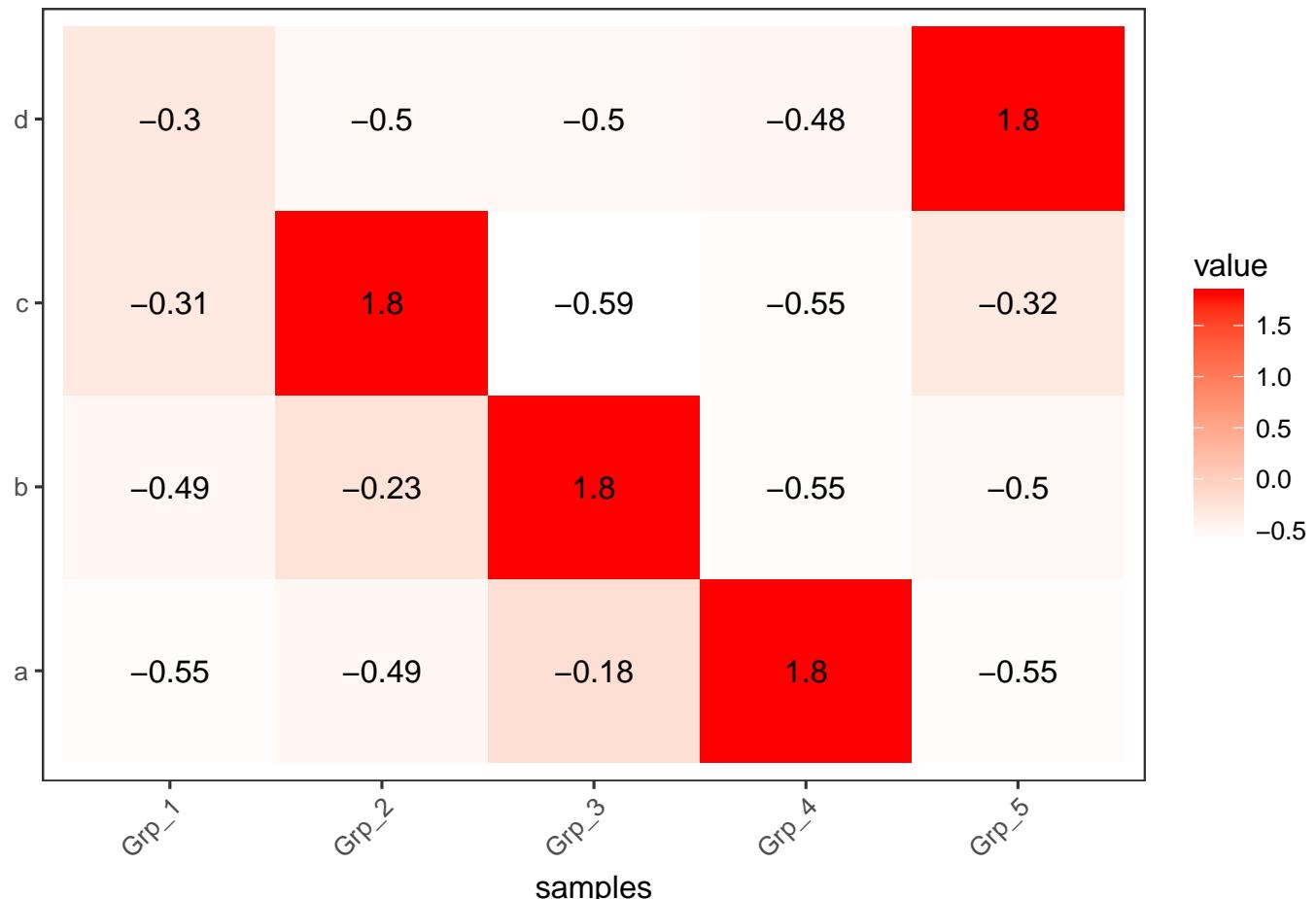
# 重命名列
colnames(data_scale) <- colnames(data)
data_scale

##           Grp_1       Grp_2       Grp_3       Grp_4       Grp_5
## a -0.5456953 -0.4899405 -0.1811446  1.7679341 -0.5511538
## b -0.4940465 -0.2301542  1.7747592 -0.5511674 -0.4993911
## c -0.3139042  1.7740182 -0.5936858 -0.5483481 -0.3180801
## d -0.2983707 -0.5033986 -0.4995116 -0.4810369  1.7823177

data_scale$ID = rownames(data_scale)
data_scale_m = melt(data_scale, id.vars=c("ID"))
data_scale_m$value <- as.numeric(prettyNum(data_scale_m$value, digits=2))
p <- ggplot(data_scale_m, aes(x=variable, y=ID)) + xlab("samples") + ylab(NULL) +
  theme_bw() + theme(panel.grid.major = element_blank()) +
  theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45, hjust=1, vjust=1)) +
  geom_tile(aes(fill=value)) + scale_fill_gradient(low = "white", high = "red") +
  geom_text(aes(label=value))

#ggsave(p, filename="heatmap_scale.pdf", width=8, height=12, units=c("cm"),
#       colormodel="srgb")

```



`z-score` 转换后，颜色分布也相对均一了，每个基因在不同样品之间的表达的高低一目了然。但是不同基因之间就完全不可比了。

3.3.3 抹去异常值

粗暴一点，假设检测饱和度为 100，大于 100 的值都视为 100 对待。

```
data_ori <- "Grp_1;Grp_2;Grp_3;Grp_4;Grp_5
a;6.6;20.9;100.1;600.0;5.2
b;20.8;99.8;700.0;3.7;19.2
c;100.0;800.0;6.2;21.4;98.6
d;900;3.3;20.3;101.1;10000"

data <- read.table(text=data_ori, header=T, row.names=1, sep=";", quote="")

data[data>100] <- 100
data
```

CONTENTS

```

##   Grp_1 Grp_2 Grp_3 Grp_4 Grp_5
## a    6.6  20.9 100.0 100.0   5.2
## b   20.8  99.8 100.0   3.7 19.2
## c 100.0 100.0   6.2  21.4  98.6
## d 100.0    3.3  20.3 100.0 100.0

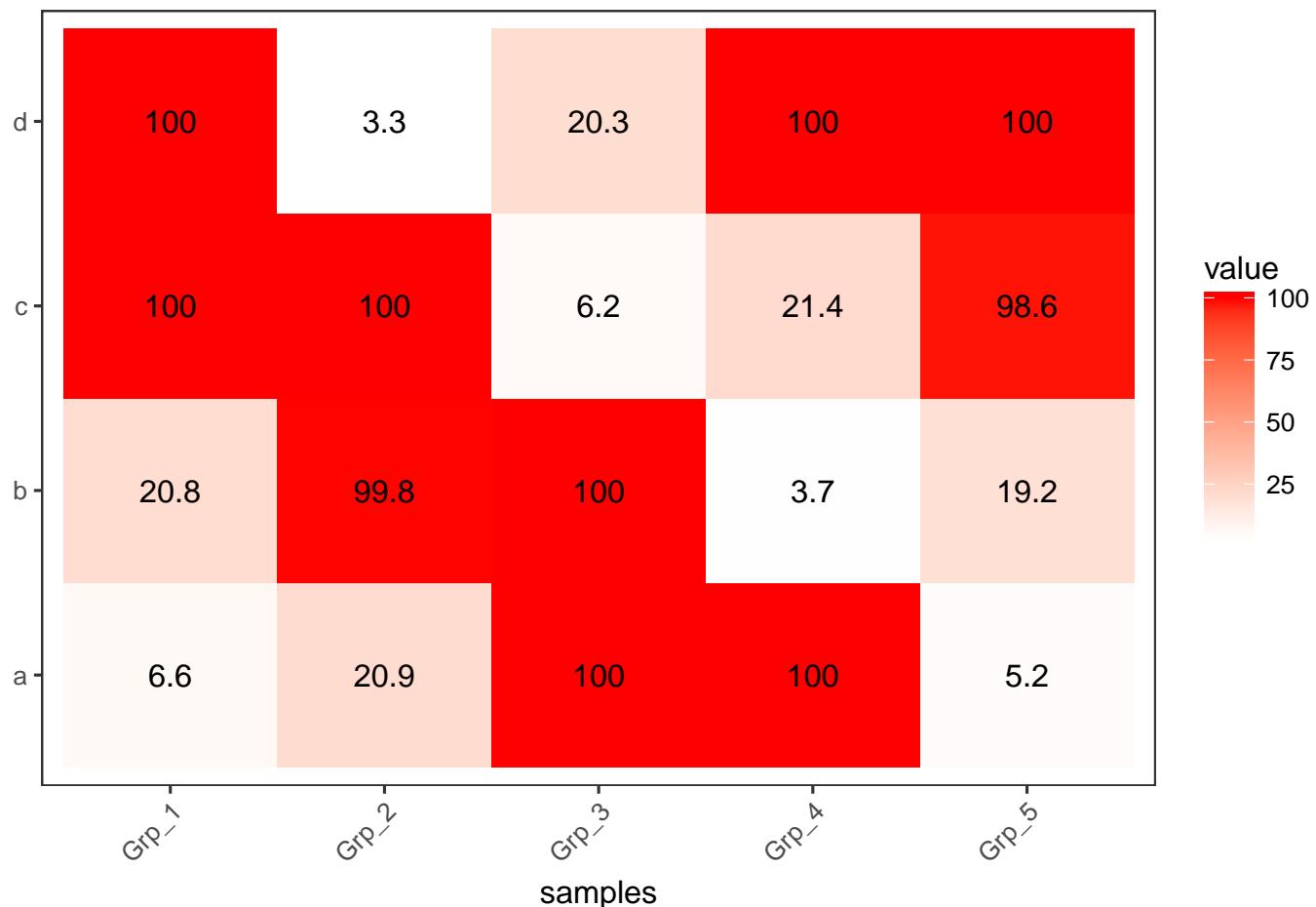
data$ID = rownames(data)
data_m = melt(data, id.vars=c("ID"))

p <- ggplot(data_m, aes(x=variable,y=ID)) + xlab("samples") + ylab(NULL) + theme_bw() +
  theme(panel.grid.major = element_blank()) + theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1)) +
  geom_tile(aes(fill=value)) + scale_fill_gradient(low = "white", high = "red") +
  geom_text(aes(label=value))



#ggsave(p, filename="heatmap_nooutlier.pdf, width=8, height=12, units=c("cm"),
#      colormodel="srgb")


```



虽然损失了一部分信息，但整体模式还是出来了。但是在选择异常值标准时需要根据实际确认。

3.3.4 非线性颜色

正常来讲，颜色的赋予在最小值到最大值之间是均匀分布的。如果最小值到最大值之间用 100 个颜色区分，则其中每一个 bin，不论其大小、有没有值都会赋予一个颜色。非线性颜色则是对数据比较小但密集的地方赋予更多颜色，数据大但分布散的地方赋予更少颜色，这样既能加大区分度，又最小的影响原始数值。通常可以根据数据模式，手动设置颜色区间。为了方便自动化处理，也可选择用四分位数的方式设置颜色区间。

```
data_ori <- "Grp_1;Grp_2;Grp_3;Grp_4;Grp_5
a;6.6;20.9;100.1;600.0;5.2
b;20.8;99.8;700.0;3.7;19.2
c;100.0;800.0;6.2;21.4;98.6
d;900;3.3;20.3;101.1;10000"

data <- read.table(text=data_ori, header=T, row.names=1, sep=";", quote="")

data
```

```
##   Grp_1  Grp_2  Grp_3  Grp_4    Grp_5
## a    6.6   20.9  100.1  600.0      5.2
## b   20.8   99.8  700.0    3.7    19.2
## c  100.0   800.0    6.2   21.4    98.6
## d  900.0    3.3   20.3  101.1  10000.0
```

获取数据的最大、最小、第一四分位数、中位数、第三四分位数

```
data$ID = rownames(data)
data_m = melt(data, id.vars=c("ID"))
summary_v <- summary(data_m$value)
summary_v
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##      3.30    16.05   60.00  681.36  225.82 10000.00
```

在最小值和第一四分位数之间划出 6 个区间，第一四分位数和中位数之间划出 6 个区间，中位数和第三四分位数之间划出 5 个区间，最后的数划出 5 个区间

```
break_v <- unique(c(seq(summary_v[1]*0.95,summary_v[2],length=6),
seq(summary_v[2],summary_v[3],length=6),seq(summary_v[3],summary_v[5],length=5),
seq(summary_v[5],summary_v[6]*1.05,length=5)))
break_v
```

```
## [1] 3.1350 5.7180 8.3010 10.8840 13.4670 16.0500
```

CONTENTS

```
## [7] 24.8400 33.6300 42.4200 51.2100 60.0000 101.4562
## [13] 142.9125 184.3687 225.8250 2794.3687 5362.9125 7931.4562
## [19] 10500.0000
```

按照设定的区间分割数据，原始数据替换为了其所在的区间的数值

```
data_m$value <- cut(data_m$value, breaks=break_v, labels=break_v[2:length(break_v)])
break_v=unique(data_m$value)

data_m
```

```
##      ID variable     value
## 1     a   Grp_1    8.301
## 2     b   Grp_1    24.84
## 3     c   Grp_1 101.45625
## 4     d   Grp_1 2794.36875
## 5     a   Grp_2    24.84
## 6     b   Grp_2 101.45625
## 7     c   Grp_2 2794.36875
## 8     d   Grp_2    5.718
## 9     a   Grp_3 101.45625
## 10    b   Grp_3 2794.36875
## 11    c   Grp_3    8.301
## 12    d   Grp_3    24.84
## 13    a   Grp_4 2794.36875
## 14    b   Grp_4    5.718
## 15    c   Grp_4    24.84
## 16    d   Grp_4 101.45625
## 17    a   Grp_5    5.718
## 18    b   Grp_5    24.84
## 19    c   Grp_5 101.45625
## 20    d   Grp_5 10500
```

虽然看上去还是数值，但已经不是数字类型了，而是不同的因子了，这样就可以对不同的因子赋予不同的颜色了

```
is.numeric(data_m$value)
```

```
## [1] FALSE
```

```
is.factor(data_m$value)
```

```
## [1] TRUE
```

```
break_v
```

```
## [1] 8.301      24.84      101.45625   2794.36875  5.718      10500
## 18 Levels: 5.718 8.301 10.884 13.467 16.05 24.84 33.63 42.42 51.21 ... 10500
```

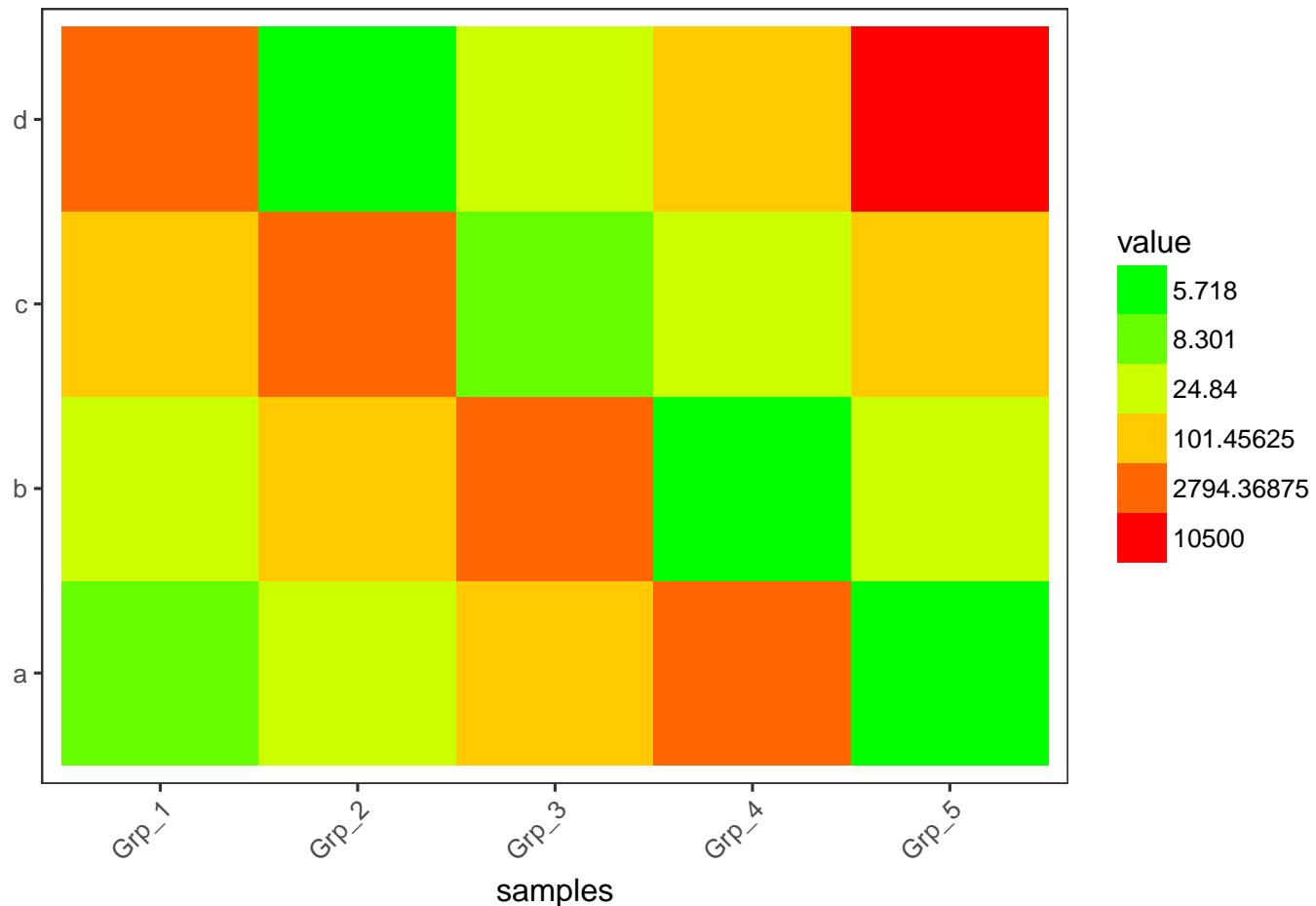
产生对应数目的颜色

```
gradientC=c('green','yellow','red')
col <- colorRampPalette(gradientC)(length(break_v))
col
```

```
## [1] "#00FF00" "#66FF00" "#CCFF00" "#FFCB00" "#FF6500" "#FF0000"
```

```
p <- ggplot(data_m, aes(x=variable,y=ID)) + xlab("samples") + ylab(NULL) + theme_bw() +
  theme(panel.grid.major = element_blank()) + theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45,hjust=1, vjust=1)) + geom_tile(aes(fill=value))

# 与上面不同的地方，使用的是 scale_fill_manual 逐个赋值
p <- p + scale_fill_manual(values=col)
p
#ggsave(p, filename="heatmap_nonlinear.pdf", width=8, height=12, units=c("cm"), colormodel="srgb")
```



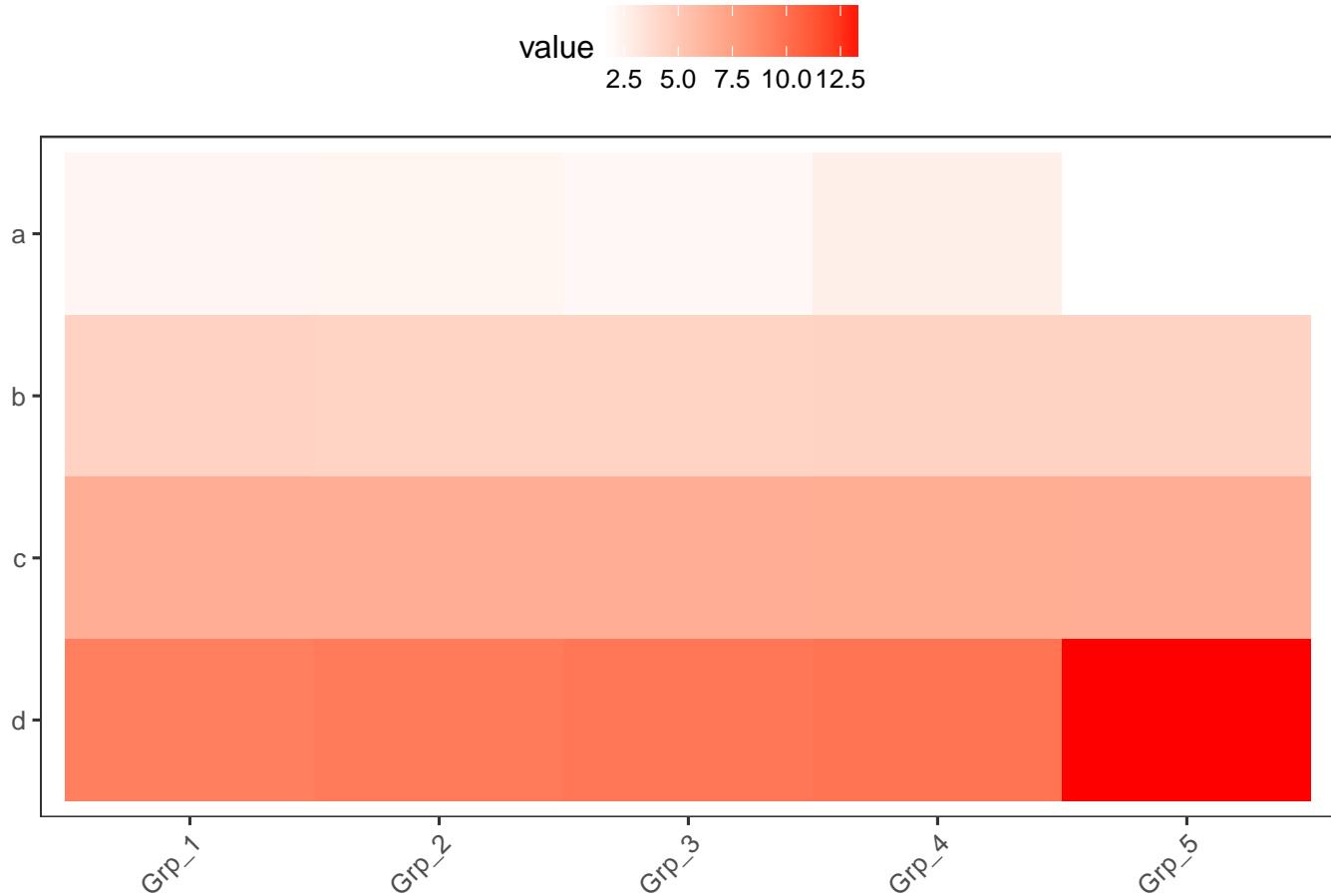
3.3.5 调整行或列的顺序

如果想保持图中每一行的顺序与输入的数据框一致，需要设置因子的水平。这也是 `ggplot2` 中调整图例或横纵轴字符顺序的常用方式。

```

data_rowname <- rownames(data)
data_rowname <- as.vector(rownames(data))
data_rownames <- rev(data_rowname)
data_log_m$ID <- factor(data_log_m$ID, levels=data_rownames, ordered=T)
p <- ggplot(data_log_m, aes(x=variable, y=ID)) + xlab(NULL) + ylab(NULL) + theme_bw() +
  theme(panel.grid.major = element_blank()) + theme(legend.key=element_blank()) +
  theme(axis.text.x=element_text(angle=45, hjust=1, vjust=1)) +
  theme(legend.position="top") + geom_tile(aes(fill=value)) +
  scale_fill_gradient(low = "white", high = "red")
p
#ggsave(p, filename="heatmap_log.pdf", width=8, height=12, units=c("cm"), colormodel="srgb")

```



基于 ggplot2 的 heatmap 绘制到现在就差不多了，但总是这么画下去也会觉得有点累，有没有办法更简化呢？

3.4 热图绘制 - pheatmap

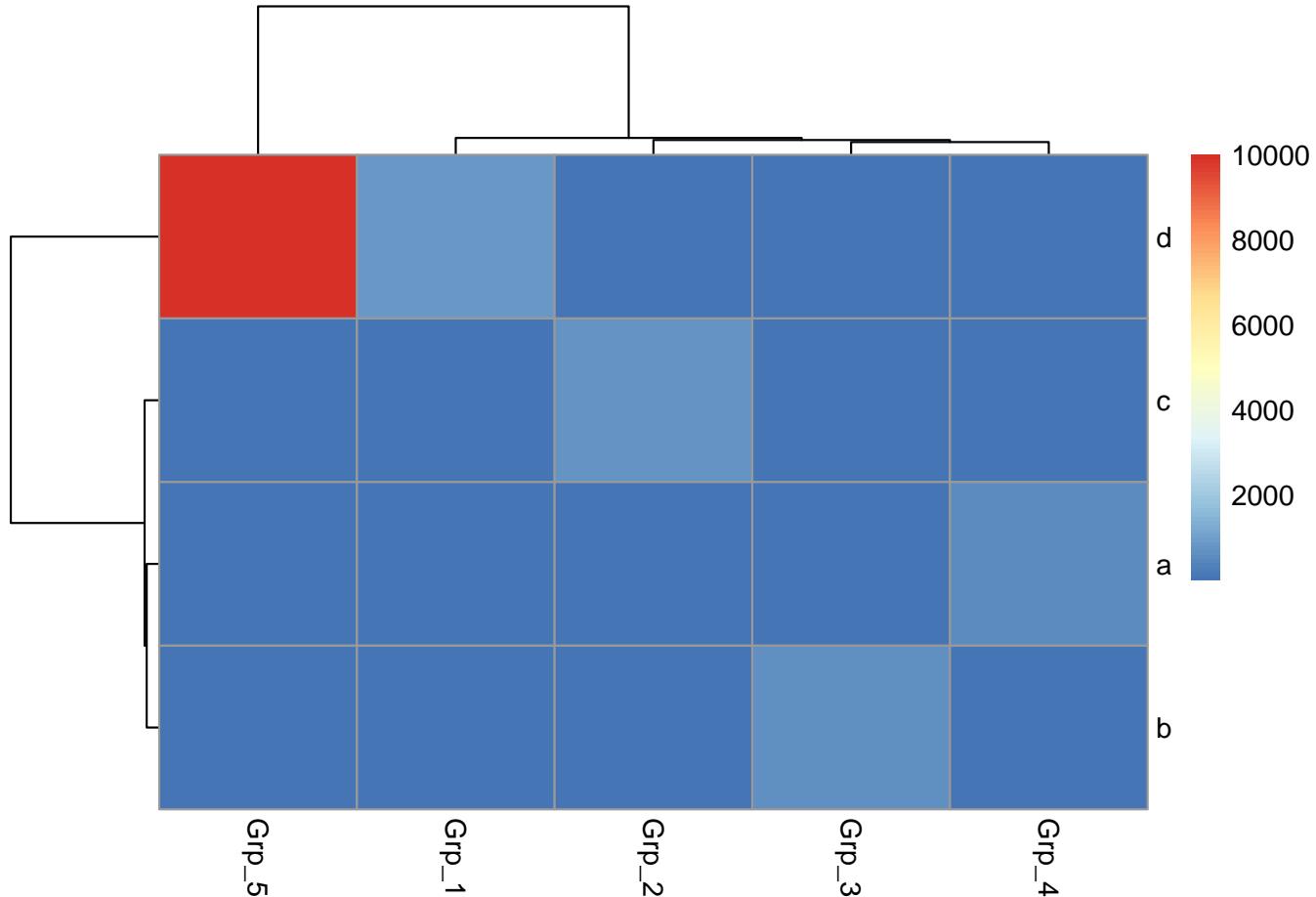
绘制热图除了使用 ggplot2，还可以有其它的包或函数，比如 `pheatmap::pheatmap` (`pheatmap` 包中的 `pheatmap` 函数)、`gplots::heatmap.2` 等。

相比于 ggplot2 作 heatmap, pheatmap 会更为简单一些，一个函数设置不同的参数，可以完成行列聚类、行列注释、Z-score 计算、颜色自定义等。那我们来看看效果怎样。

```
data_ori <- "Grp_1;Grp_2;Grp_3;Grp_4;Grp_5
a;6.6;20.9;100.1;600.0;5.2
b;20.8;99.8;700.0;3.7;19.2
c;100.0;800.0;6.2;21.4;98.6
d;900;3.3;20.3;101.1;10000"

data <- read.table(text=data_ori, header=T, row.names=1, sep=";", quote="")
```

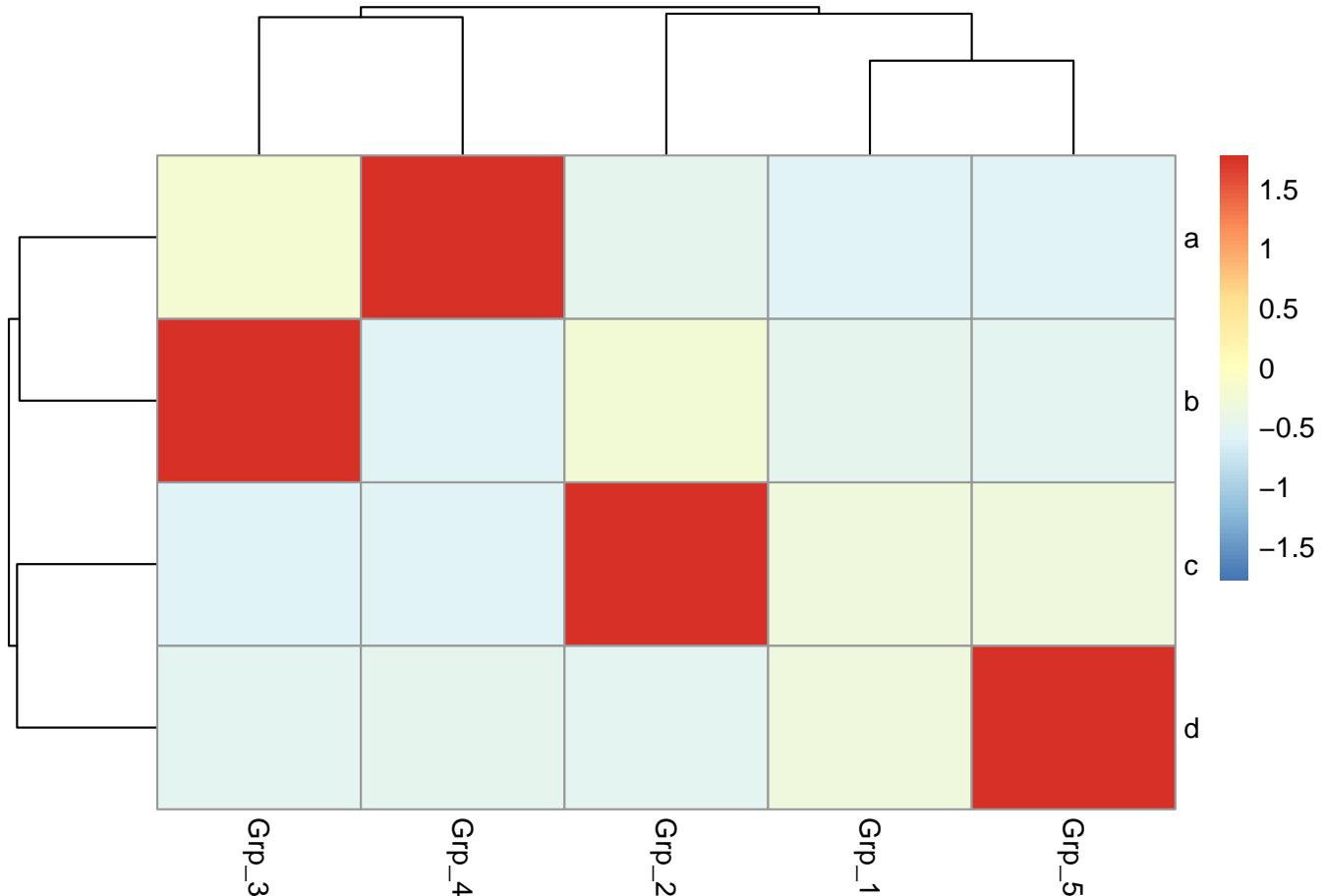
```
#pheatmap::pheatmap(data, filename="pheatmap_1.pdf")
pheatmap::pheatmap(data)
```



虽然有点丑，但一步就出来了。

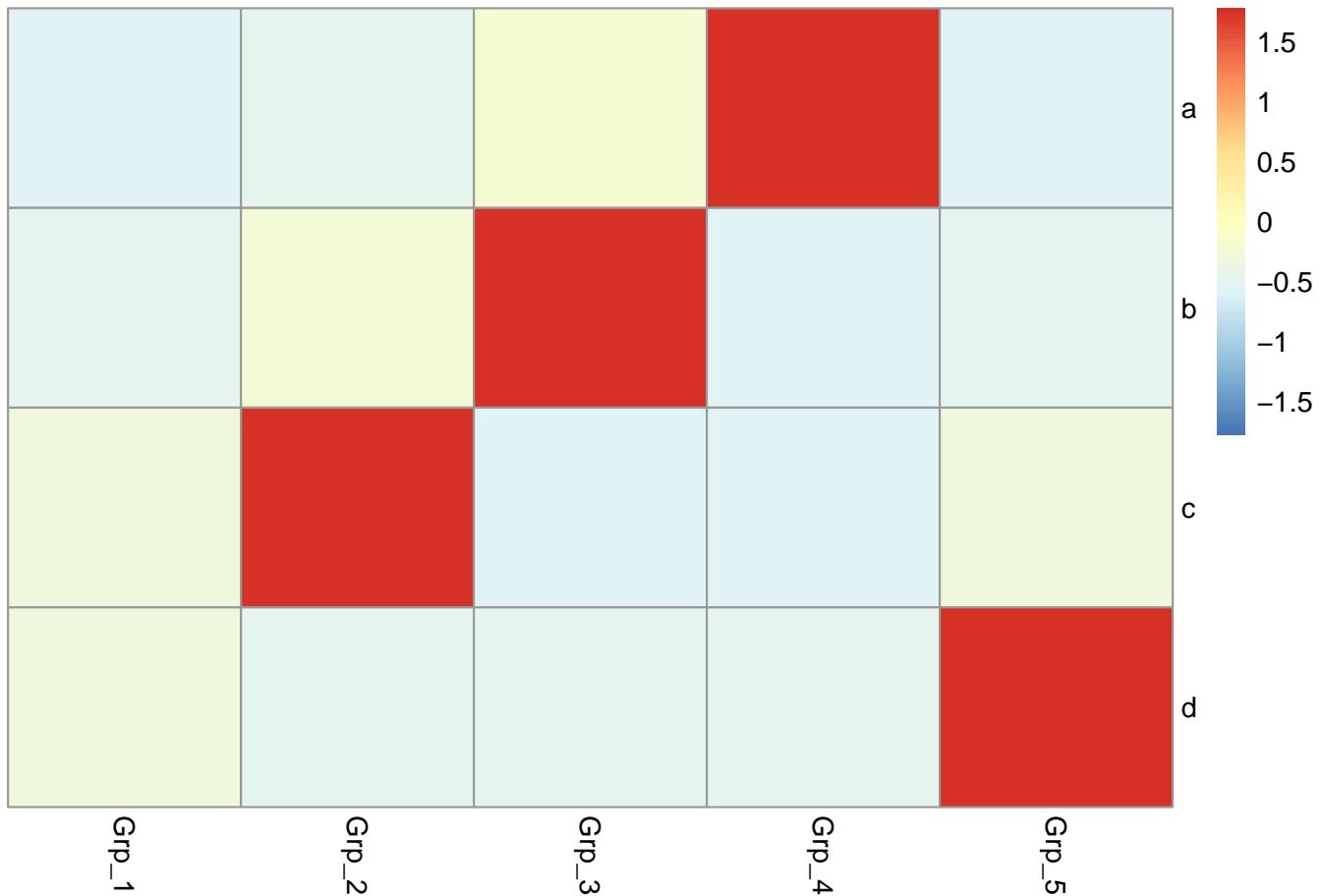
在 heatmap 美化篇提到的数据前期处理方式，都可以用于 pheatmap 的画图。此外 z-score 计算在 pheatmap 中只要一个参数就可以实现。

```
pheatmap::pheatmap(data, scale="row")
```



有时可能不需要行或列的聚类，原始展示就可以了。

```
pheatmap::pheatmap(data, scale="row", cluster_rows=FALSE, cluster_cols=FALSE)
```



给矩阵 (data) 中行和列不同的分组注释。假如有两个文件，第一个文件为行注释，其第一列与矩阵中的第一列内容相同(顺序没有关系)，其它列为第一列的不同的标记，如下面示例中 (假设行为基因，列为样品) 的 2,3 列对应基因的不同类型 (TF or enzyme) 和不同分组。第二个文件为列注释，其第一列与矩阵中第一行内容相同，其它列则为样品的注释。

```
row_anno = data.frame(type=c("TF", "Enzyme", "Enzyme", "TF"),
                       class=c("clu1", "clu1", "clu2", "clu2"), row.names=rownames(data))
row_anno
```

```
##      type class
## a      TF   clu1
## b Enzyme  clu1
## c Enzyme  clu2
## d      TF   clu2
```

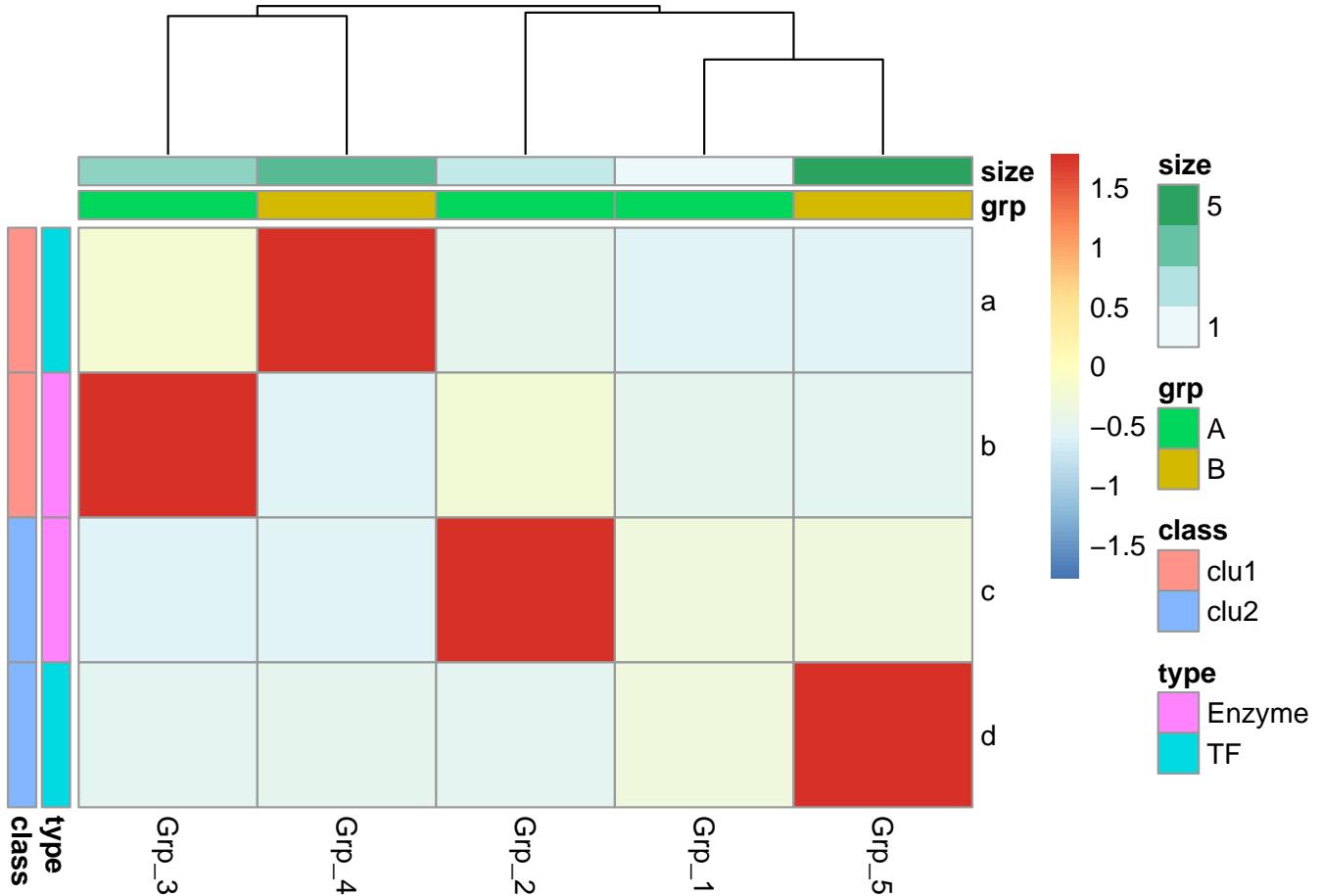
```
col_anno = data.frame(grp=c("A", "A", "A", "B", "B"), size=1:5, row.names=colnames(data))
col_anno
```

```
##      grp size
## Grp_1  A    1
```

CONTENTS

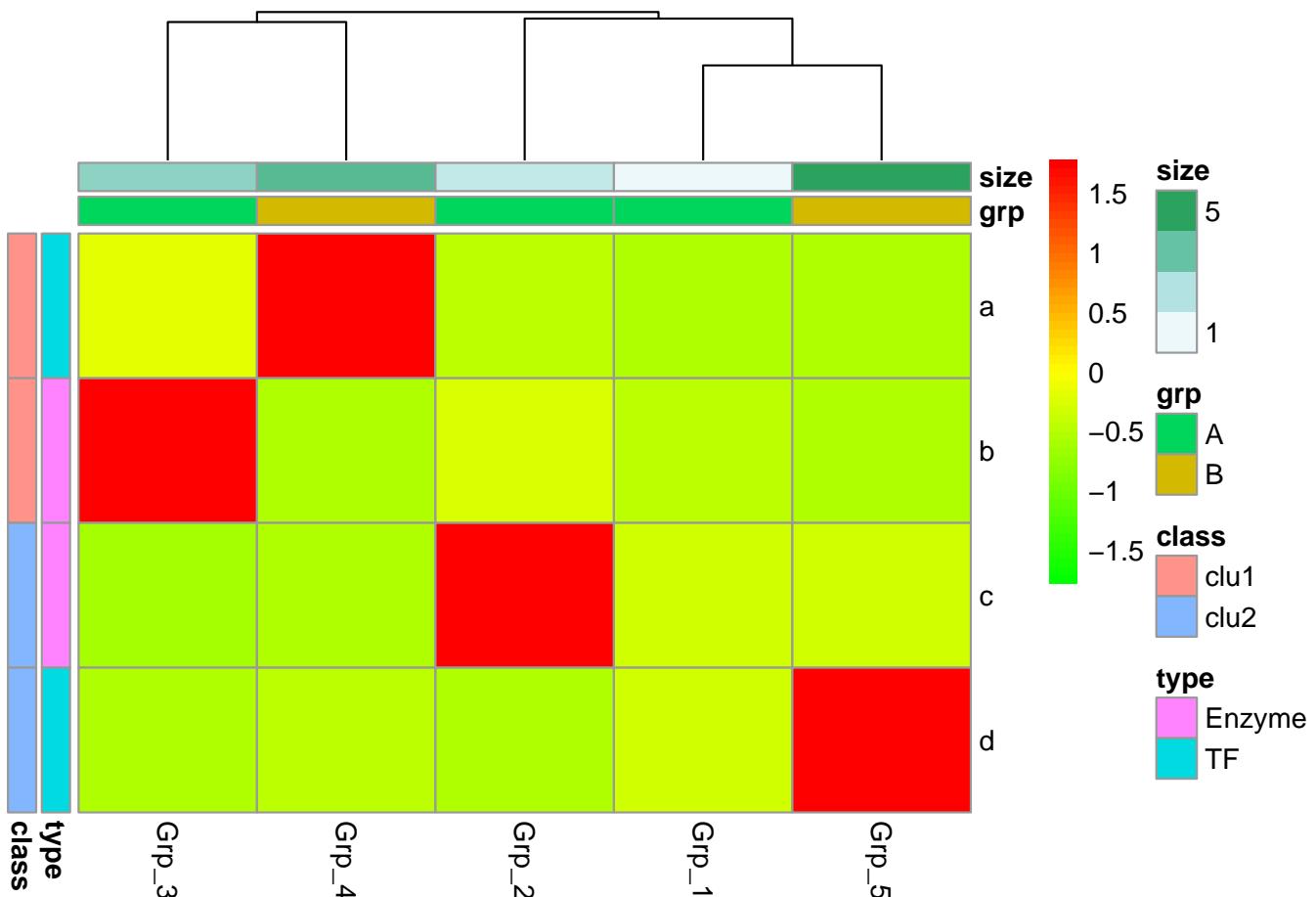
```
## Grp_2    A    2
## Grp_3    A    3
## Grp_4    B    4
## Grp_5    B    5
```

```
pheatmap::pheatmap(data, scale="row",
cluster_rows=FALSE, annotation_col=col_anno, annotation_row=row_anno)
```



自定义下颜色吧。

```
# <bias> values larger than 1 will give more color for high end.
# Values between 0-1 will give more color for low end.
pheatmap::pheatmap(data, scale="row", cluster_rows=FALSE,
annotation_col=col_anno, annotation_row=row_anno,
color=colorRampPalette(c('green','yellow','red')), bias=1)(50))
```



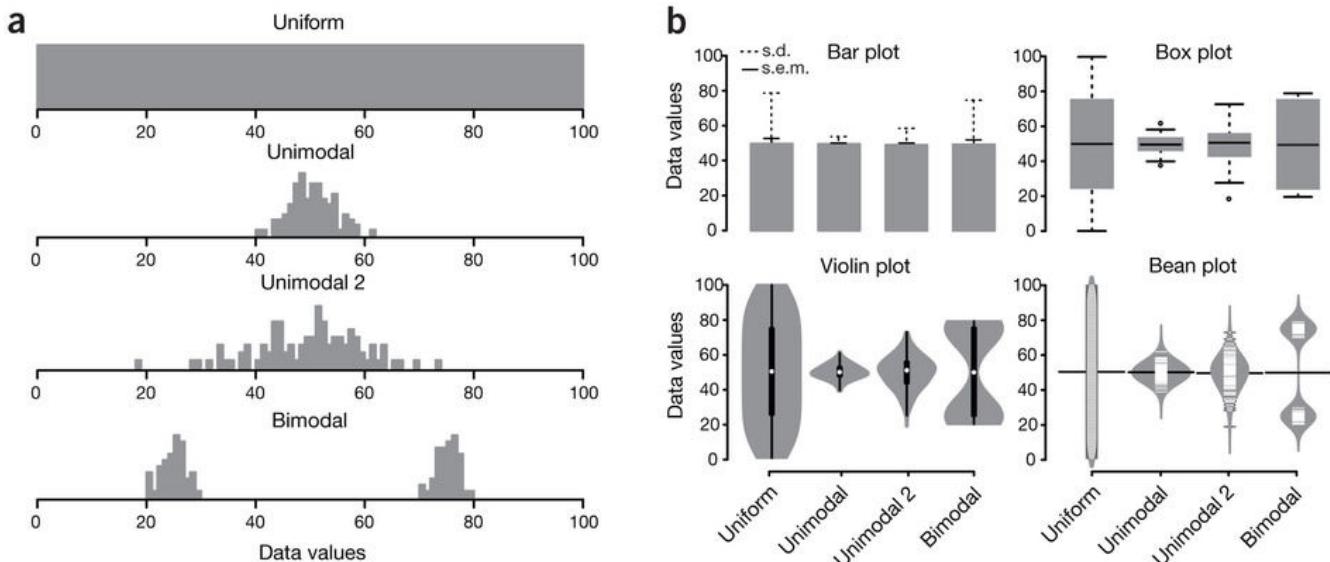
`heatmap.2` 的使用在上一期转录组分析绘制相关性热图时有提到，这次就不介绍了，跟 `pheatmap` 有些类似，而且也有不少教程。

3.5 箱线图

箱线图是能同时反映数据统计量和整体分布，又很漂亮的展示图。在 2014 年的 Nature Method 上有 2 篇 Correspondence 论述了使用箱线图的好处和一个在线绘制箱线图的工具。就这样都可以发两篇 Nature method，没天理，但也说明了箱线图的重要意义。

下面这张图展示了 Bar plot、Box plot、Violin plot 和 Bean plot 对数据分布的反应。从 Bar plot 上只能看到数据标准差或标准误不同；Box plot 可以看到数据分布的集中性不同；Violin plot 和 Bean plot 展示的是数据真正的分布，尤其是对 Biomodal 数据的展示。

Boxplot 从下到上展示的是最小值，第一四分位数（箱子的下边线）、中位数（箱子中间的线）、第三四分位数（箱子上边线）、最大值，具体解读参见 http://mp.weixin.qq.com/s/t3UTI_qAli0cy1g6ZmHtwg。



(a) Hypothetical sample data sets of 100 data points each that are uniform, unimodal with one of two different variances or bimodal. Simple bar plot representations and statistical parameters may obscure such different data distributions. (b) Comparison of data visualization methods. Bar plots typically represent only the mean and s.d. or s.e.m. Box plots visualize the five-number summary of a data set (minimum, lower quartile, median, upper quartile and maximum). Violin and bean plots represent the actual distribution of the individual data sets.

- Nature Method 文章 <http://www.nature.com/nmeth/journal/v11/n2/full/nmeth.2811.html>

3.5.1 一步步解析箱线图绘制

假设你有这么一个基因表达矩阵，第一列为基因名字，第一行为样品名字，想绘制样品中基因表达的整体分布。

```
profile="Name;2cell_1;2cell_2;2cell_3;4cell_1;4cell_2;4cell_3;zygote_1;zygote_2;zygote_3
A;4;6;7;3.2;5.2;5.6;2;4;3
B;6;8;9;5.2;7.2;7.6;4;6;5
C;8;10;11;7.2;9.2;9.6;6;8;7
D;10;12;13;9.2;11.2;11.6;8;10;9
E;12;14;15;11.2;13.2;13.6;10;12;11
F;14;16;17;13.2;15.2;15.6;12;14;13
G;15;17;18;14.2;16.2;16.6;13;15;14
H;16;18;19;15.2;17.2;17.6;14;16;15
I;17;19;20;16.2;18.2;18.6;15;17;16
J;18;20;21;17.2;19.2;19.6;16;18;17
L;19;21;22;18.2;20.2;20.6;17;19;18
M;20;22;23;19.2;21.2;21.6;18;20;19
N;21;23;24;20.2;22.2;22.6;19;21;20
O;22;24;25;21.2;23.2;23.6;20;22;21"
```

读入数据并转换为 ggplot2 需要的长数据表格式，好好体会下这个格式，虽然多占用了不少空间，但是确实很方便。

CONTENTS

```

profile_text <- read.table(text=profile, header=T, row.names=1, quote="", sep="; ", check.names=F
# 在 melt 时保留位置信息
# melt 格式是 ggplot2 画图最喜欢的格式
#
library(ggplot2)
library(reshape2)
data_m <- melt(profile_text)

## No id variables; using all as measure variables

head(data_m)

##     variable value
## 1 2cell_1      4
## 2 2cell_1      6
## 3 2cell_1      8
## 4 2cell_1     10
## 5 2cell_1     12
## 6 2cell_1     14

summary(data_m)

##       variable      value
## 2cell_1:14   Min.    : 2.00
## 2cell_2:14   1st Qu.:10.25
## 2cell_3:14   Median  :16.00
## 4cell_1:14   Mean    :14.87
## 4cell_2:14   3rd Qu.:19.20
## 4cell_3:14   Max.    :25.00
## (Other):42

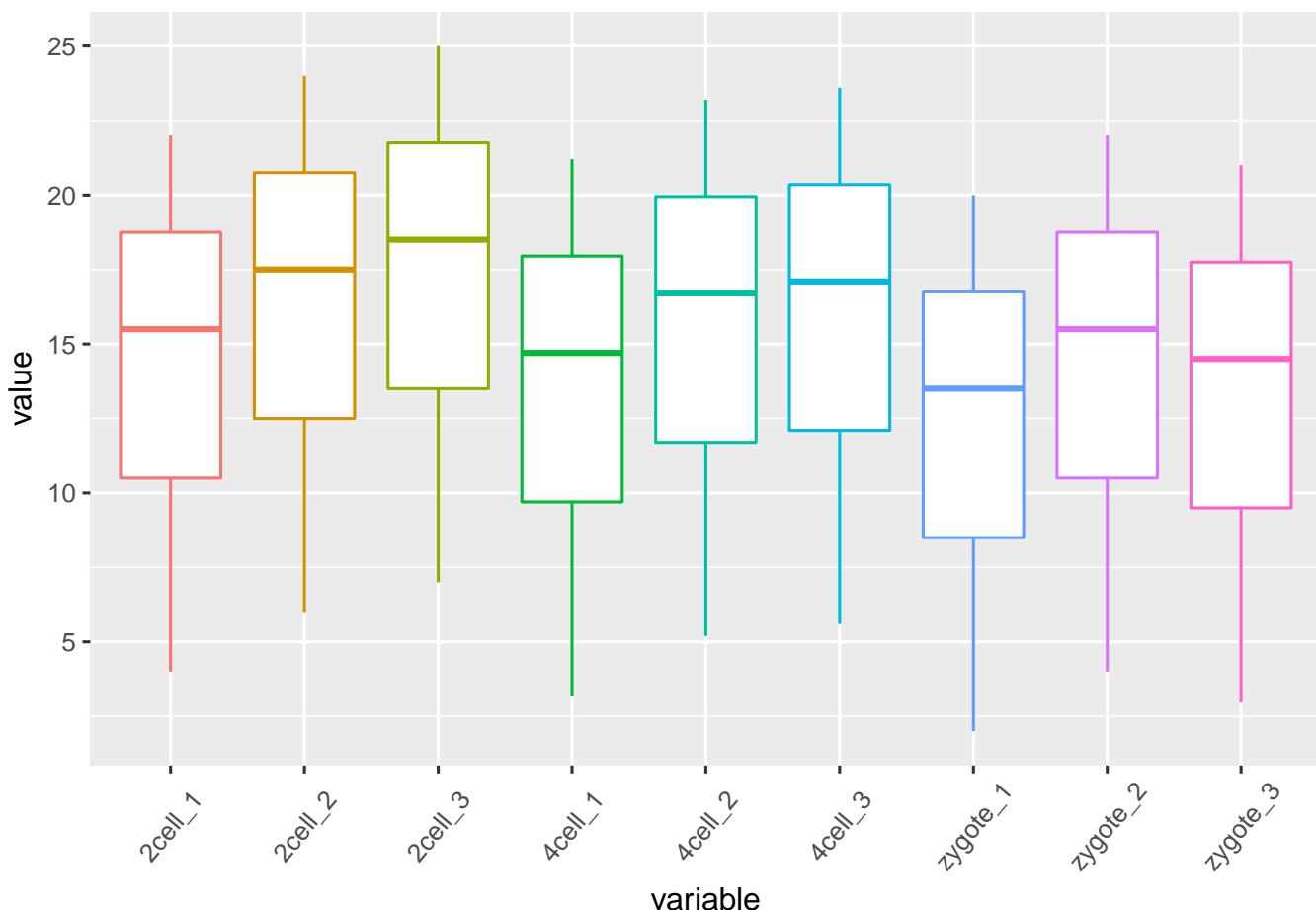
```

variable 和 value 为矩阵 melt 后的两列的名字，内部变量，可以通过?melt 查看如何修改。variable 代表了点线的属性，value 代表对应的值。像往常一样，就可以直接画图了。

```

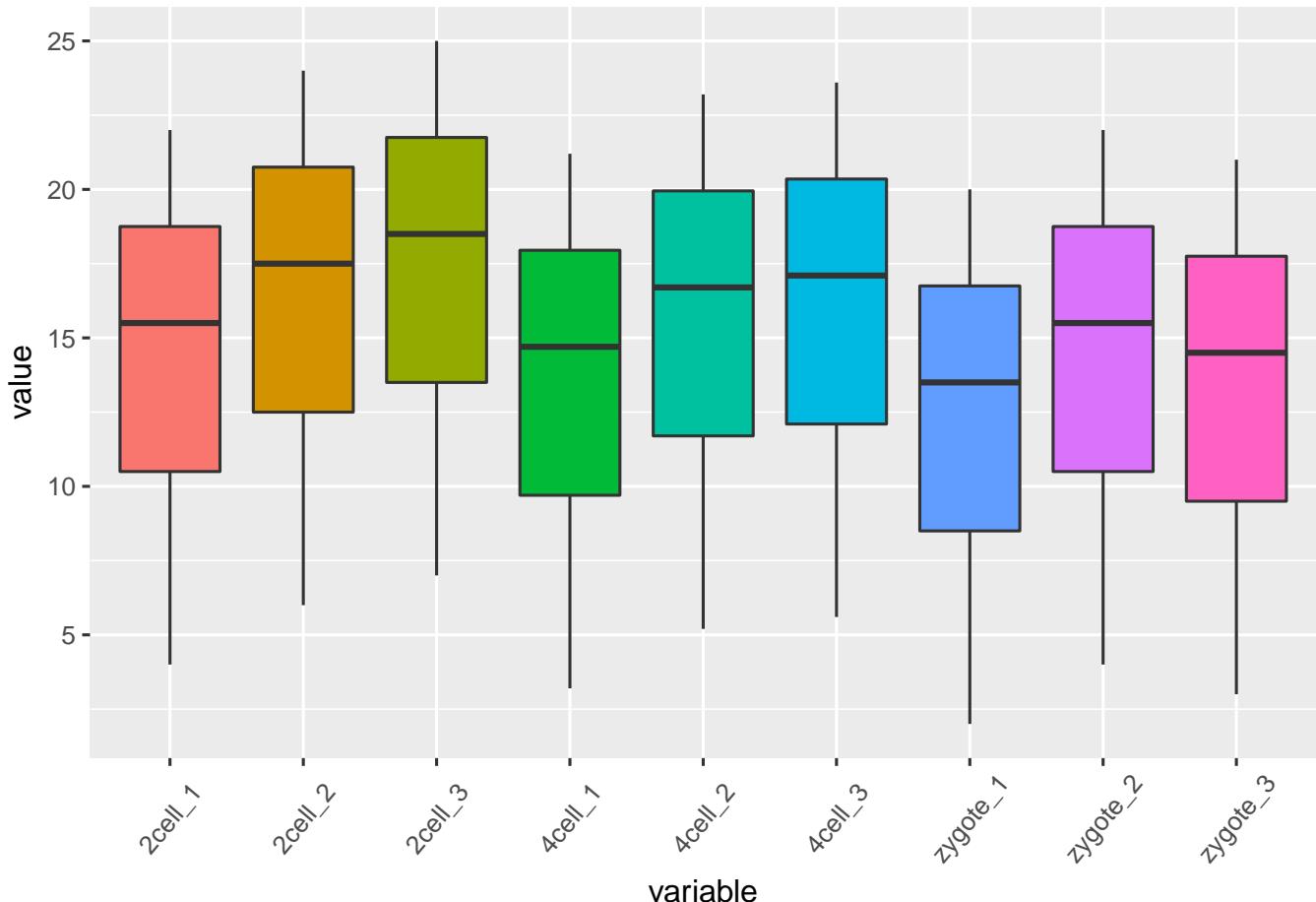
p <- ggplot(data_m, aes(x=variable, y=value, color=variable)) +
geom_boxplot() +
theme(axis.text.x=element_text(angle=50, hjust=0.5, vjust=0.5)) +
theme(legend.position="none")
p
# 图会存储在当前目录的 Rplots.pdf 文件中，如果用 Rstudio，可以不运行 dev.off()
# dev.off()

```



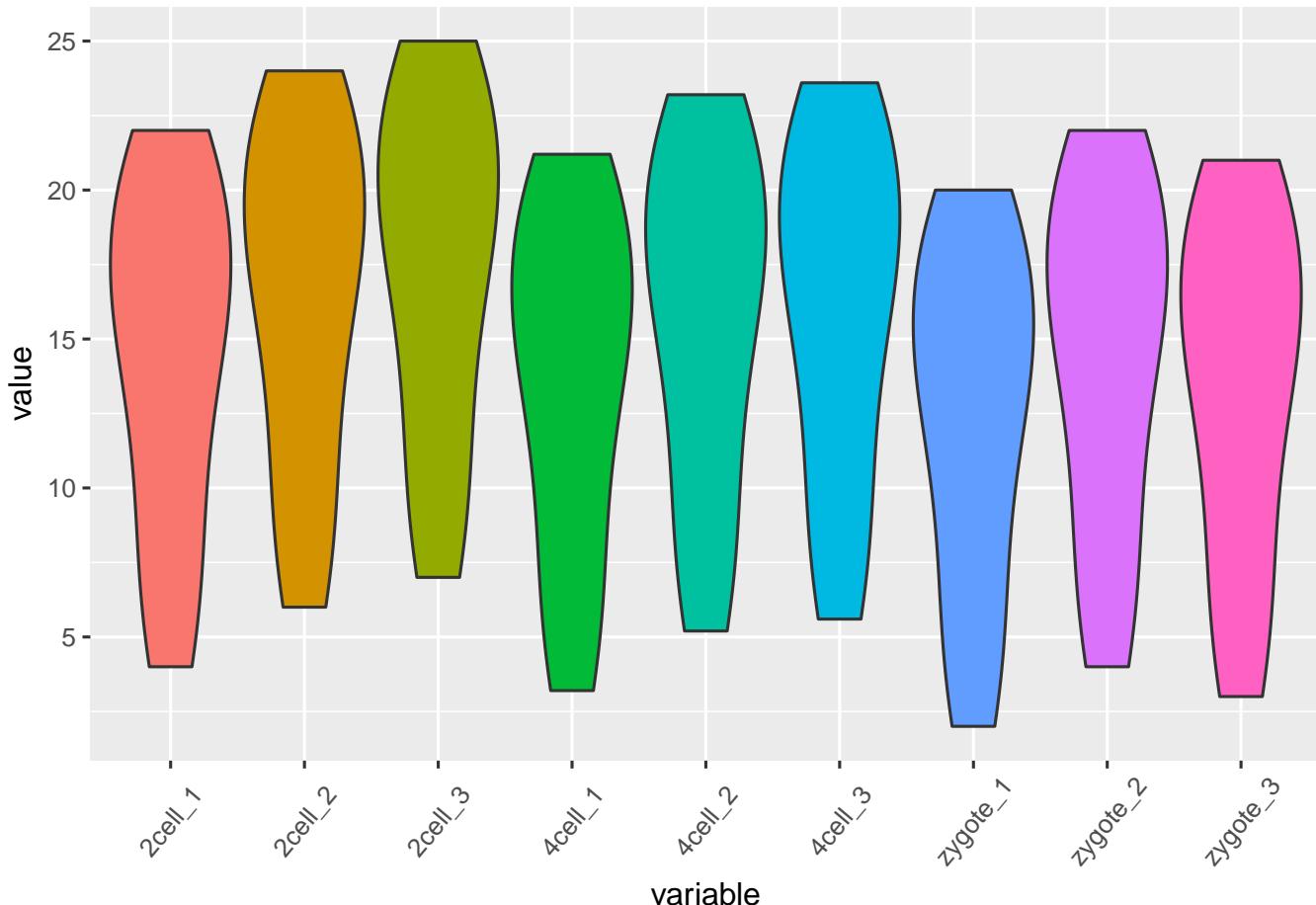
箱线图出来了，看上去还可以，再加点色彩 (fill)。

```
# variable 和 value 为矩阵 melt 后的两列的名字，内部变量， variable 代表了点线的属性， value 代表对应的值。
p <- ggplot(data_m, aes(x=variable, y=value)) +
  geom_boxplot(aes(fill=factor(variable))) +
  theme(axis.text.x=element_text(angle=50,hjust=0.5, vjust=0.5)) +
  theme(legend.position="none")
p
# 图会存储在当前目录的 Rplots.pdf 文件中，如果用 Rstudio，可以不运行 dev.off()
#dev.off()
```

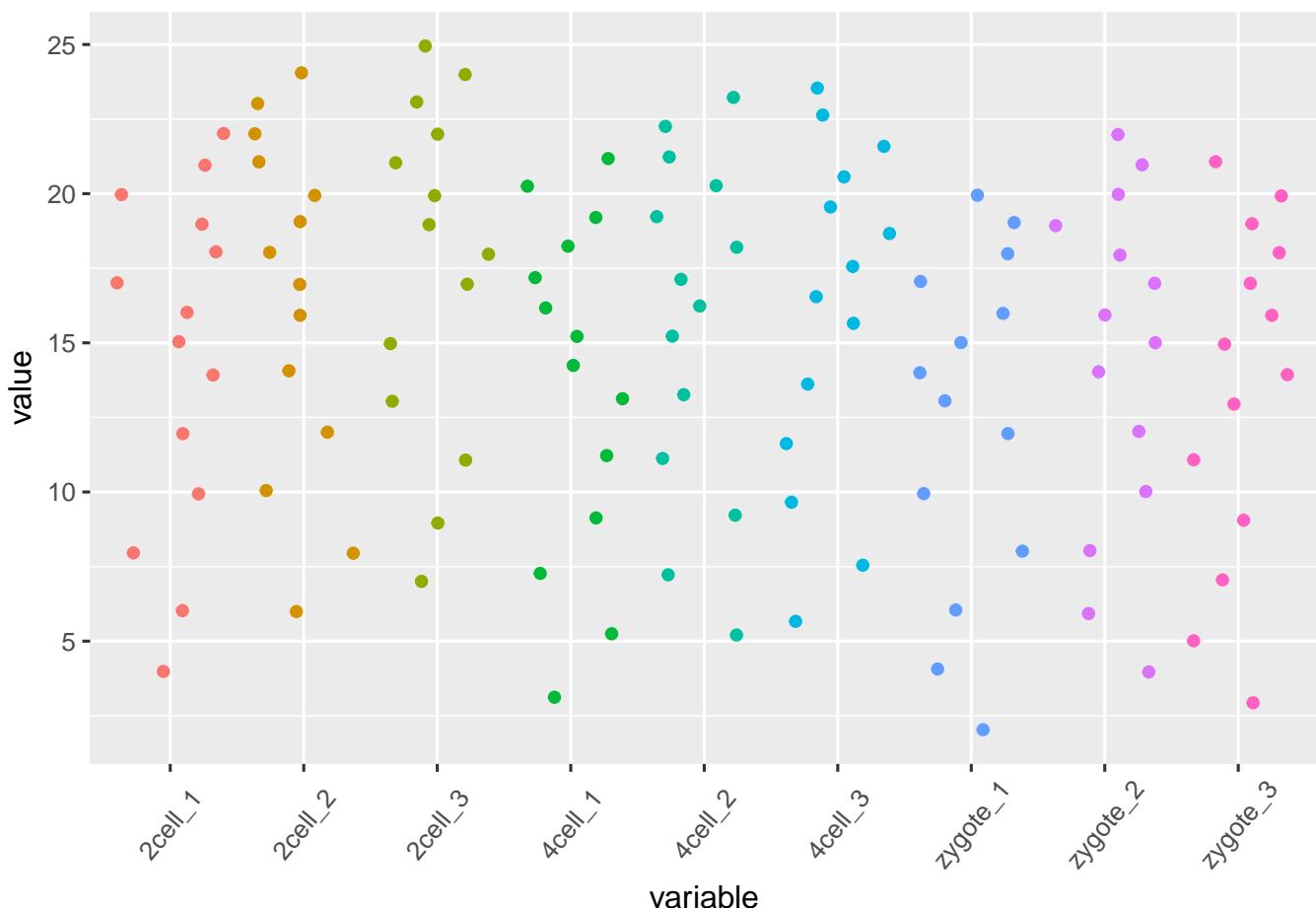


再看看 violin plot

```
# variable 和 value 为矩阵 melt 后的两列的名字，内部变量，variable 代表了点线的属性，value 代表对应的值。
p <- ggplot(data_m, aes(x=variable, y=value)) +
  geom_violin(aes(fill=factor(variable))) +
  theme(axis.text.x=element_text(angle=50,hjust=0.5, vjust=0.5)) +
  theme(legend.position="none")
p
# 图会存储在当前目录的 Rplots.pdf 文件中，如果用 Rstudio，可以不运行 dev.off()
#dev.off()
```



```
# variable 和 value 为矩阵 melt 后的两列的名字，内部变量，variable 代表了点线的属性，value 代表对应的值。
p <- ggplot(data_m, aes(x=variable, y=value)) +
  geom_jitter(aes(color=factor(variable))) +
  theme(axis.text.x=element_text(angle=50,hjust=0.5, vjust=0.5)) +
  theme(legend.position="none")
p
# 图会存储在当前目录的 Rplots.pdf 文件中，如果用 Rstudio，可以不运行 dev.off()
#dev.off()
```



还有 Jitter plot (这里使用的是 ggbeeswarm 包)

```

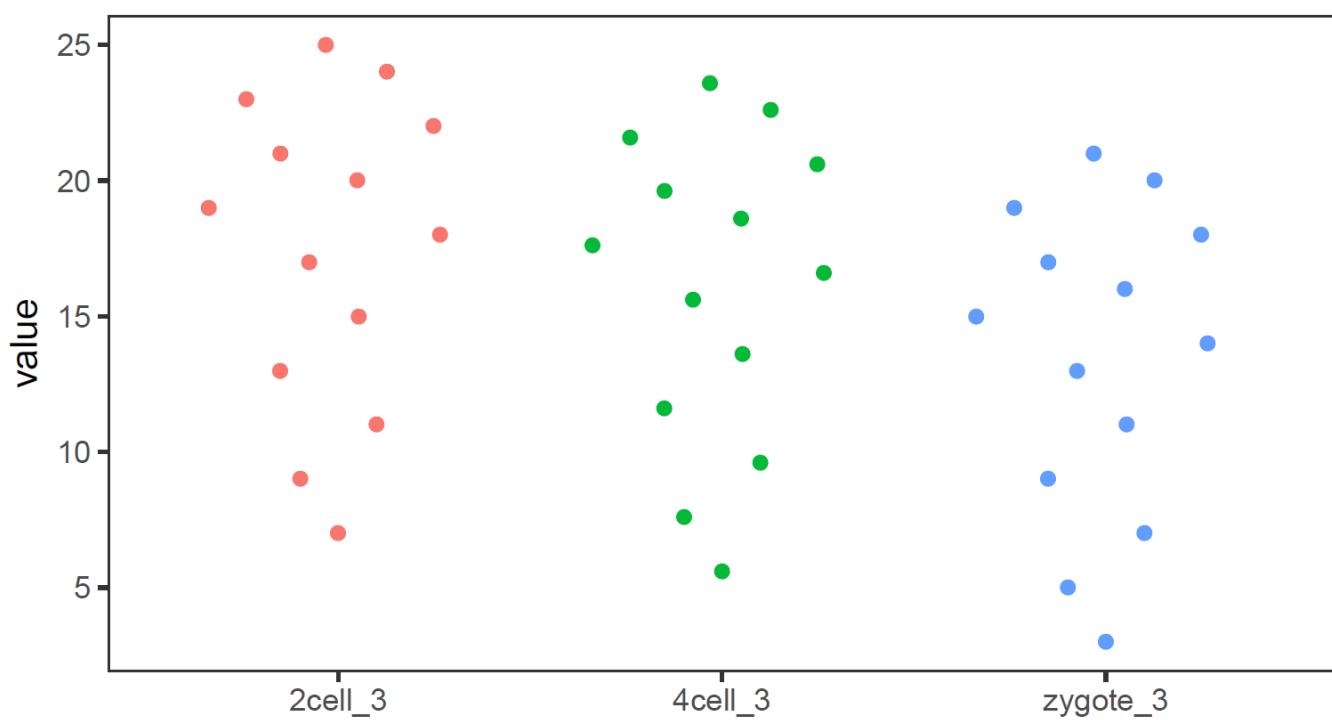
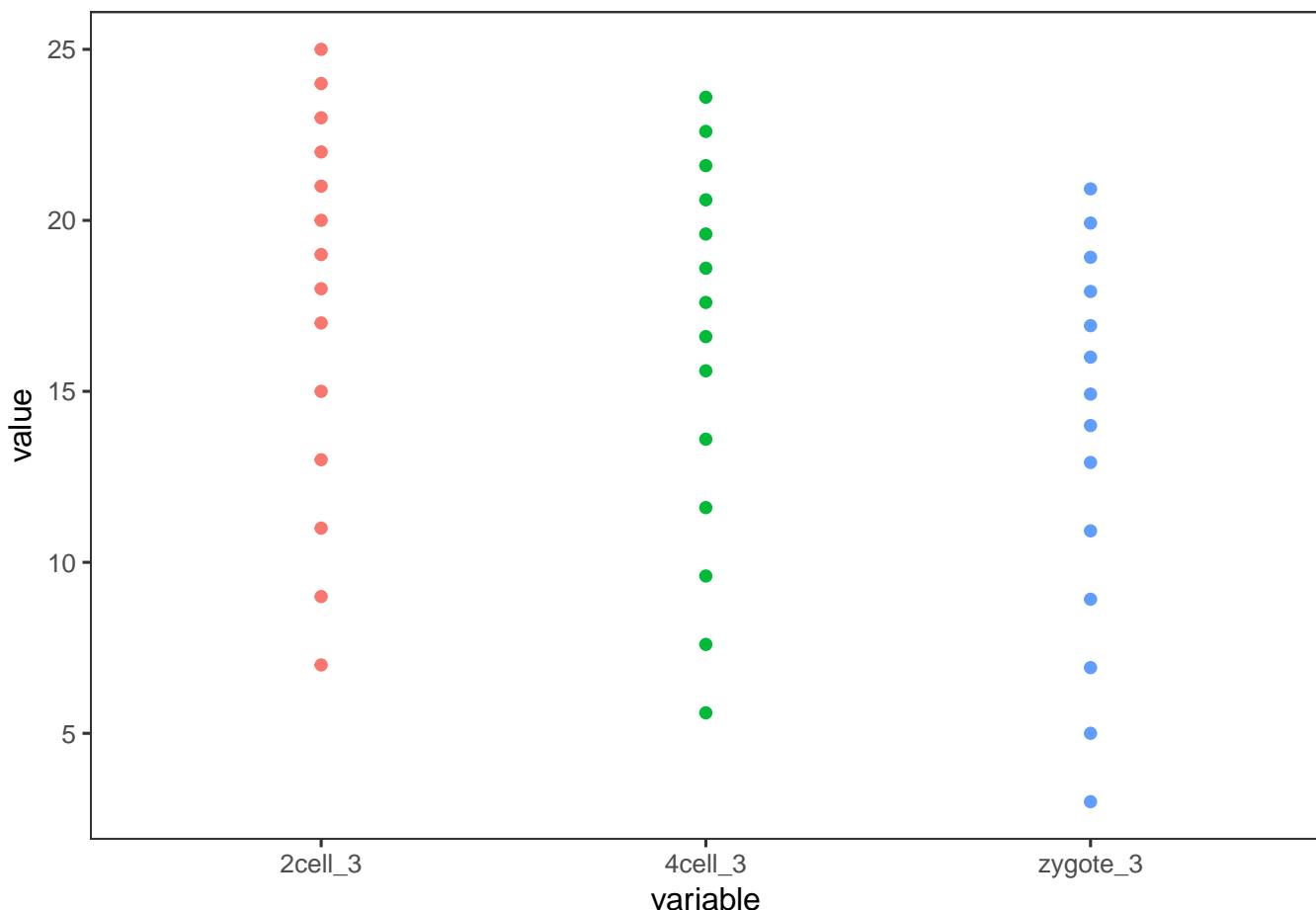
library(ggbeeswarm)
# 为了更好的效果，只保留其中一个样品的数据
# grep1 类似于 Linux 的 grep 命令，获取特定模式的字符串

data_m2 <- data_m[grep1("_3", data_m$variable),]

# variable 和 value 为矩阵 melt 后的两列的名字，内部变量，
# variable 代表了点线的属性，value 代表对应的值。
p <- ggplot(data_m2, aes(x=variable, y=value), color=variable) +
  geom_quasirandom(aes(colour=factor(variable)), groupOnX=FALSE) +
  theme_bw() + theme(panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(), legend.key=element_blank()) +
  theme(legend.position="none")
p
#ggsave(p, filename="jitterplot.pdf", width=14, height=8, units=c("cm"))

```

CONTENTS



也可以用 `geom_jitter(aes(colour=factor(variable)))` 替代 `geom_quasirandom(aes(colour=factor(variable)))`

绘制抖动图，但个人认为 `geom_quasirandom` 给出的结果更有特色。

3.5.2 绘制单个基因 (A) 的箱线图

为了更好的展示效果，下面的矩阵增加了样品数量和样品的分组信息。

```
#profile="Name;2cell_1;2cell_2;2cell_3;2cell_4;2cell_5;2cell_6;4cell_1;4cell_2;4cell_3;\  
#4cell_4;4cell_5;4cell_6;zygote_1;zygote_2;zygote_3;zygote_4;zygote_5;zygote_6  
#A;4;6;7;5;8;6;3.2;5.2;5.6;3.6;7.6;4.8;2;4;3;2;4;2.5  
#B;6;8;9;7;10;8;5.2;7.2;7.6;5.6;9.6;6.8;4;6;5;4;6;4.5"  
  
profile_text <- read.table("data/boxplot_singleGene.data", header=T, row.names=1, quote="",  
sep="\t", check.names=F)  
  
data_m = data.frame(t(profile_text['A',]))  
data_m$sample = rownames(data_m)  
# 只挑选显示部分  
# grep1 前面已经讲过用于匹配  
data_m[grep1('_[123]', data_m$sample),]
```

```
##          A   sample  
## 2cell_1 4.0 2cell_1  
## 2cell_2 6.0 2cell_2  
## 2cell_3 7.0 2cell_3  
## 4cell_1 3.2 4cell_1  
## 4cell_2 5.2 4cell_2  
## 4cell_3 5.6 4cell_3  
## zygote_1 2.0 zygote_1  
## zygote_2 4.0 zygote_2  
## zygote_3 3.0 zygote_3
```

获得样品分组信息 (这个例子比较特殊，样品的分组信息就是样品名字下划线前面的部分)

```
# 可以利用 strsplit 分割，取出其前面的字符串  
# R 中复杂的输出结果多数以列表的形式体现，在之前的矩阵操作教程中  
# 提到过用 str 函数来查看复杂结果的结构，并从中获取信息  
group = unlist(lapply(strsplit(data_m$sample, "_"), function(x) x[1]))  
data_m$group = group  
data_m[grep1('_[123]', data_m$sample),]
```

```
##          A   sample   group  
## 2cell_1 4.0 2cell_1 2cell
```

CONTENTS

```
## 2cell_2 6.0 2cell_2 2cell
## 2cell_3 7.0 2cell_3 2cell
## 4cell_1 3.2 4cell_1 4cell
## 4cell_2 5.2 4cell_2 4cell
## 4cell_3 5.6 4cell_3 4cell
## zygote_1 2.0 zygote_1 zygote
## zygote_2 4.0 zygote_2 zygote
## zygote_3 3.0 zygote_3 zygote
```

如果没有这个规律，也可以提到类似于下面的文件，指定样品所属的组的信息。

```
sampleGroup_text="Sample;Group
zygote_1;zygote
zygote_2;zygote
zygote_3;zygote
zygote_4;zygote
zygote_5;zygote
zygote_6;zygote
2cell_1;2cell
2cell_2;2cell
2cell_3;2cell
2cell_4;2cell
2cell_5;2cell
2cell_6;2cell
4cell_1;4cell
4cell_2;4cell
4cell_3;4cell
4cell_4;4cell
4cell_5;4cell
4cell_6;4cell"

#sampleGroup = read.table(text=sampleGroup_text,sep="\t",header=1,check.names=F,row.names=1)

#data_m <- merge(data_m, sampleGroup, by="row.names")

# 会获得相同的结果，脚本注释掉了以免重复执行引起问题。
```

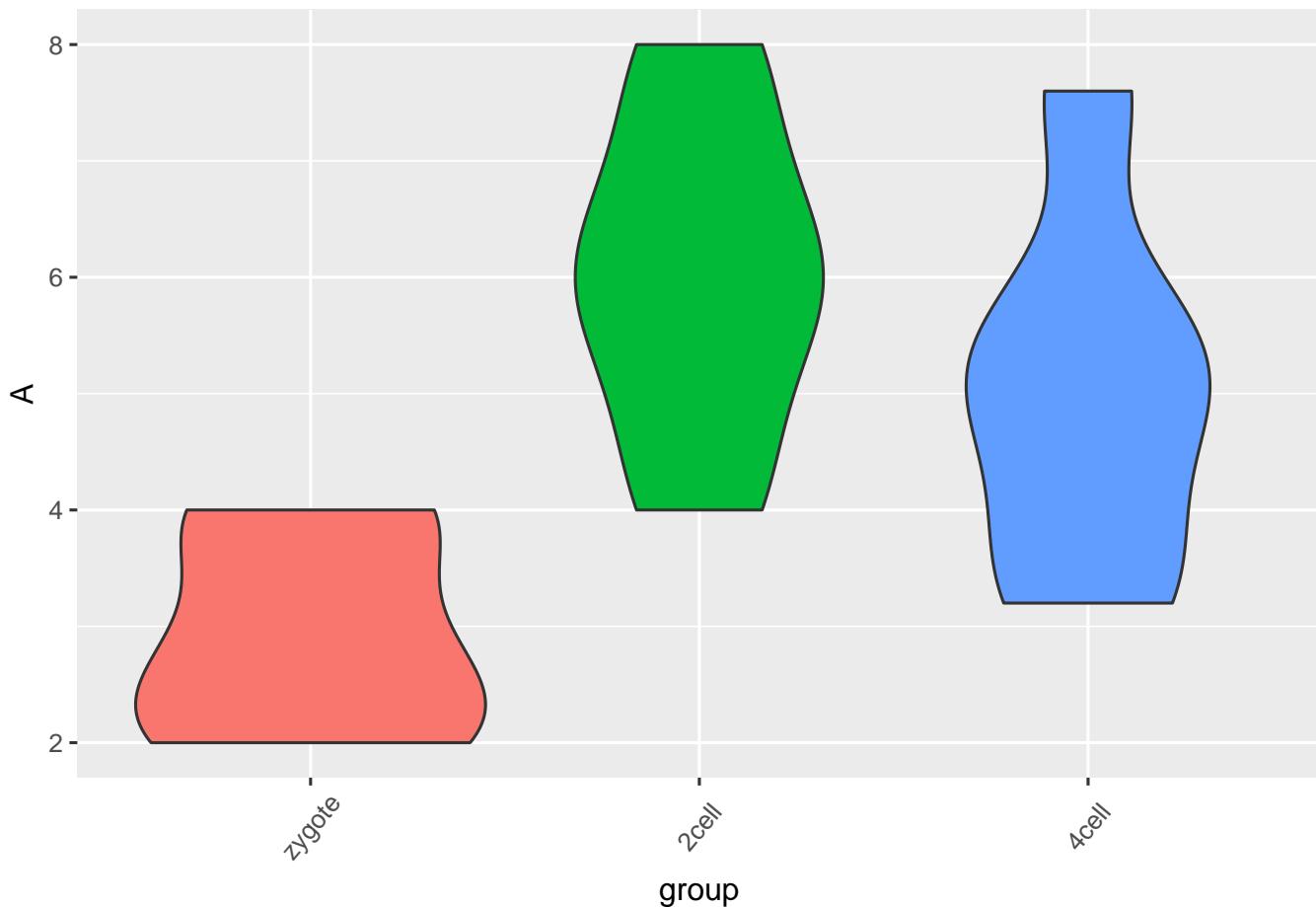
矩阵准备好了，开始画图了（小提琴图做例子，其它类似）

```
# 调整下样品出现的顺序
data_m$group <- factor(data_m$group, levels=c("zygote","2cell","4cell"))
# group 和 A 为矩阵中两列的名字，group 代表了值的属性，A 代表基因 A 对应的表达值。
# 注意看修改了的地方
p <- ggplot(data_m, aes(x=group, y=A)) +
  geom_violin(aes(fill=factor(group))) +
```

```

theme(axis.text.x=element_text(angle=50,hjust=0.5, vjust=0.5)) +
theme(legend.position="none")
p
# 图会存储在当前目录的 Rplots.pdf 文件中，如果用 Rstudio，可以不运行 dev.off()
#dev.off()

```



3.5.3 长矩阵绘制箱线图

常规矩阵绘制箱线图要求必须是个方正的矩阵输入，而有时想比较的几个组里面检测的值数目不同。比如有三个组，GrpA 组检测了 6 个病人，GrpB 组检测了 10 个病人，GrpC 组是 12 个正常人的检测数据。这时就很难形成一个行为检测值，列为样品的矩阵，就需要长表格模式解决这一问题。

```

long_table <- "Grp;Value
GrpA;10
GrpA;11
GrpA;11
GrpA;11
GrpA;12
GrpA;11

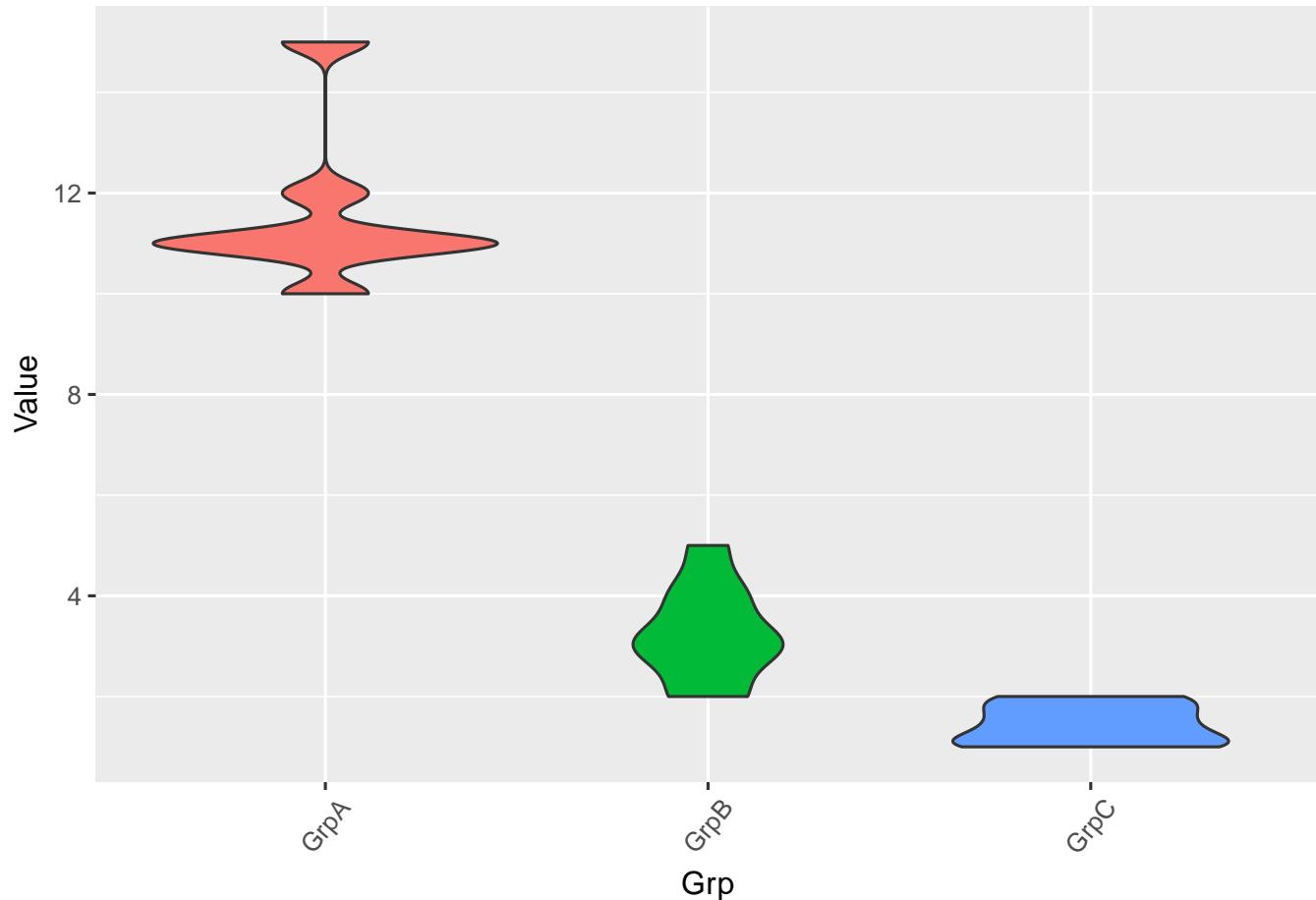
```

CONTENTS

```
GrpA;15
GrpB;5
GrpB;4
GrpB;3
GrpB;2
GrpB;4
GrpB;3
GrpB;2
GrpB;3
GrpB;3.1
GrpC;2
GrpC;1
GrpC;1
GrpC;1.1
GrpC;1.5
GrpC;1.1
GrpC;1.5
GrpC;1.8
GrpC;2"
```

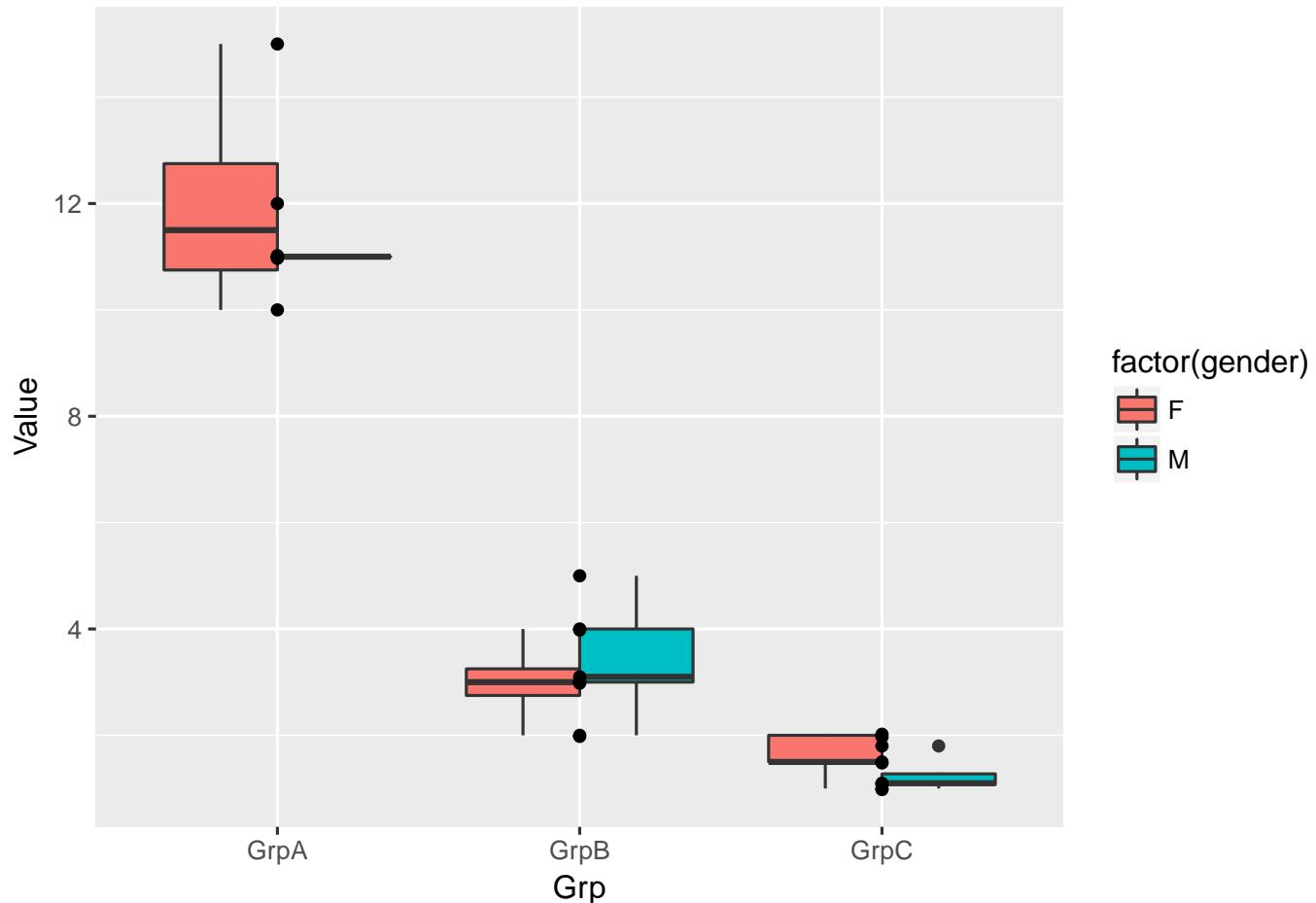
```
long_table <- read.table(text=long_table, sep=";", header=T, check.names=F)

p <- ggplot(long_table, aes(x=Grp, y=Value)) +
geom_violin(aes(fill=factor(Grp))) +
theme(axis.text.x=element_text(angle=50, hjust=0.5, vjust=0.5)) +
theme(legend.position="none")
p
```



```
a = c(rep(c('F', "M"), 12), 'F')
long_table$gender <- a

p <- ggplot(long_table, aes(x=Grp, y=Value)) +
  geom_boxplot(aes(fill=factor(gender))) +
  geom_quasirandom(groupOnX=FALSE)
p
```



长表格形式自身就是常规矩阵 melt 后的格式，这种用来绘制箱线图就很简单了，就不举例子了。

3.6 线图

线图是反映趋势变化的一种方式，其输入数据一般也是一个矩阵。

3.6.1 单线图

假设有这么一个矩阵，第一列为转录起始位点及其上下游 5 kb 的区域，第二列为 H3K27ac 修饰在这些区域的丰度，想绘制一张线图展示。

```
profile="Pos;H3K27ac
-5000;8.7
-4000;8.4
-3000;8.3
-2000;7.2
```

CONTENTS

```
-1000;3.6
0;3.6
1000;7.1
2000;8.2
3000;8.4
4000;8.5
5000;8.5"
```

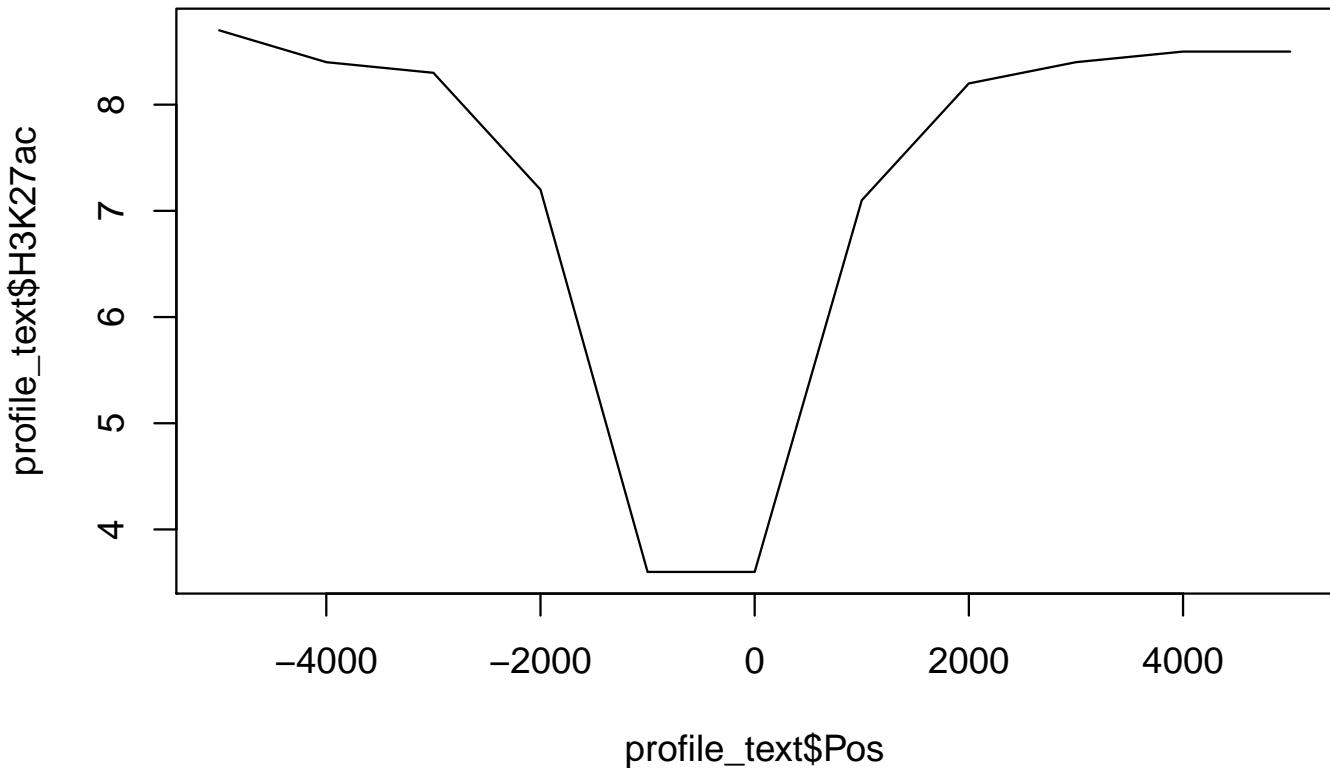
读入数据

```
profile_text <- read.table(text=profile, header=T, row.names=NULL, quote="", sep=";")  
profile_text
```

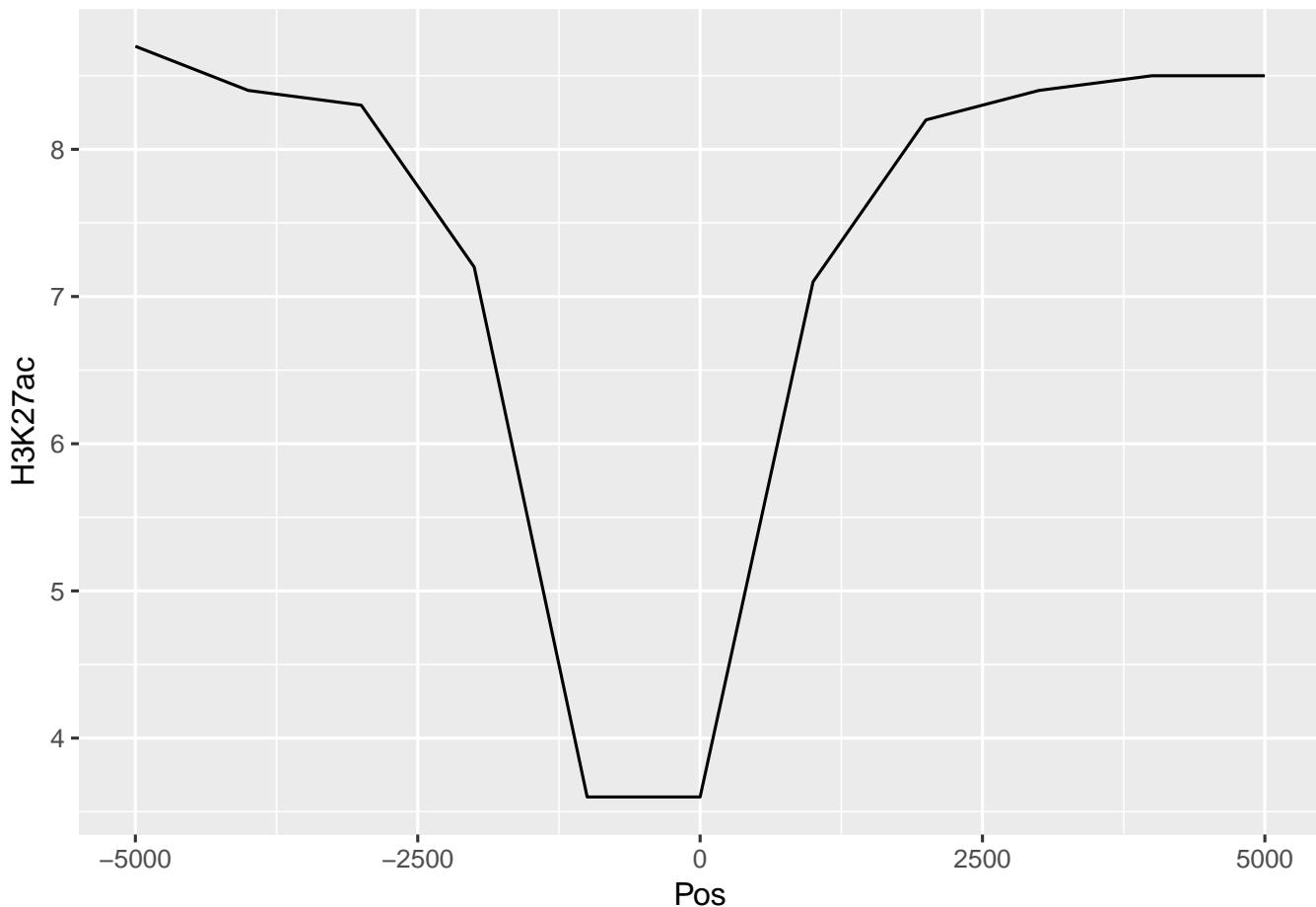
```
##          Pos H3K27ac
## 1      -5000     8.7
## 2      -4000     8.4
## 3      -3000     8.3
## 4      -2000     7.2
## 5      -1000     3.6
## 6         0     3.6
## 7      1000     7.1
## 8      2000     8.2
## 9      3000     8.4
## 10     4000     8.5
## 11     5000     8.5
```

```
plot(profile_text$Pos, profile_text$H3K27ac, type = "l")
```

CONTENTS



```
ggplot(profile_text, aes(x=Pos, y=H3K27ac)) + geom_line()
```



```
profile_text <- read.table(text=profile, header=T, row.names=1, quote="", sep=";")  
profile_text
```

```
##      H3K27ac  
## -5000     8.7  
## -4000     8.4  
## -3000     8.3  
## -2000     7.2  
## -1000     3.6  
##  0         3.6  
##  1000     7.1  
##  2000     8.2  
##  3000     8.4  
##  4000     8.5  
##  5000     8.5
```

```
# 在 melt 时保留位置信息  
# melt 格式是 ggplot2 画图最喜欢的格式  
# 好好体会下这个格式，虽然多占用了不少空间，但是确实很方便  
  
# 这里可以用 `xvariable`，也可以是其它字符串，但需要保证后面与这里的一致
```

```
# 因为这一列是要在 x 轴显示，所以起名为 `xvariable`。
profile_text$xvariable = rownames(profile_text)
#library(ggplot2)
#library(reshape2)
data_m <- melt(profile_text, id.vars=c("xvariable"))
data_m
```

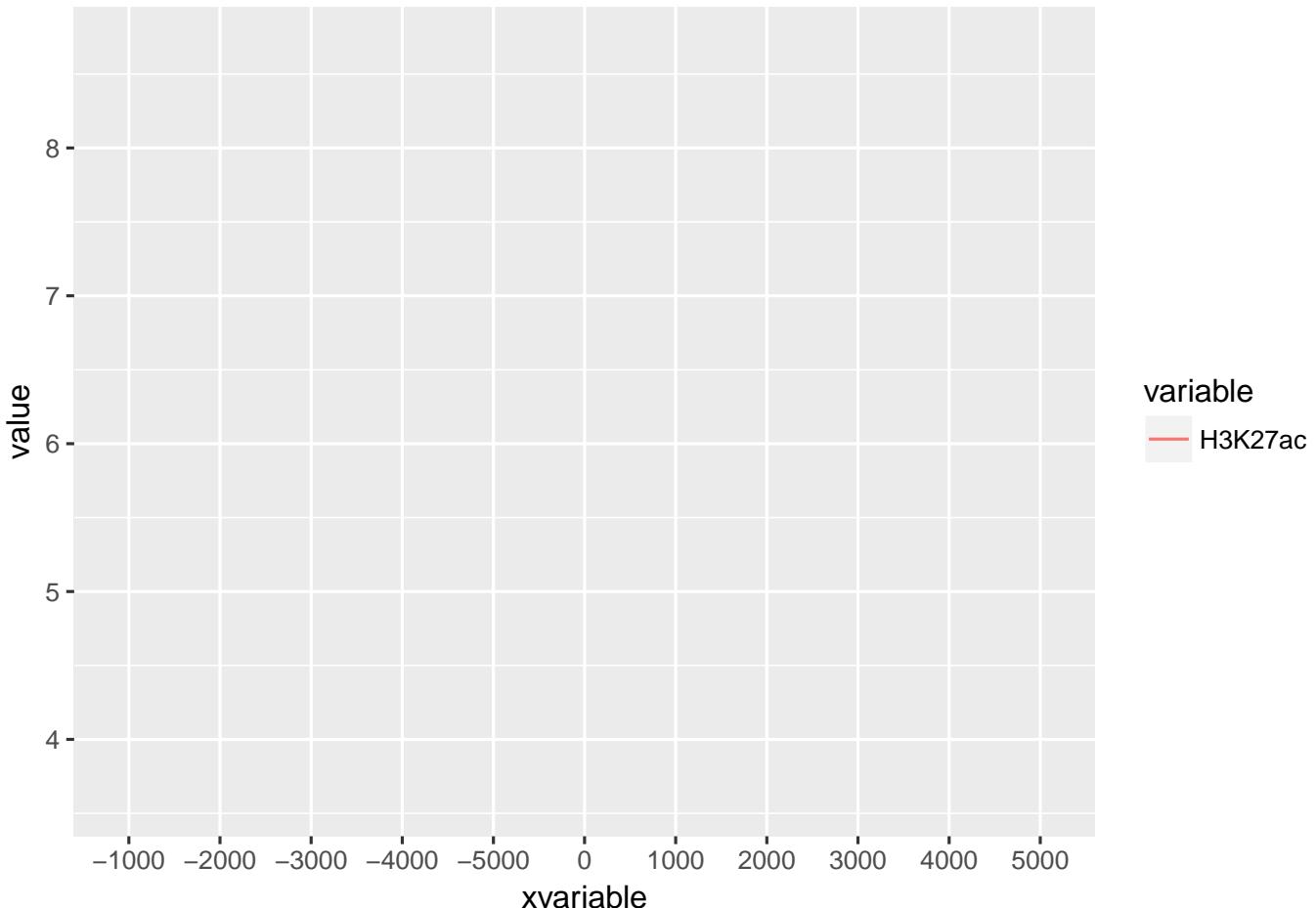
```
##      xvariable variable value
## 1      -5000  H3K27ac  8.7
## 2      -4000  H3K27ac  8.4
## 3      -3000  H3K27ac  8.3
## 4      -2000  H3K27ac  7.2
## 5      -1000  H3K27ac  3.6
## 6          0  H3K27ac  3.6
## 7      1000  H3K27ac  7.1
## 8      2000  H3K27ac  8.2
## 9      3000  H3K27ac  8.4
## 10     4000  H3K27ac  8.5
## 11     5000  H3K27ac  8.5
```

然后开始画图，与上面画 heatmap 一样。

```
# variable 和 value 为矩阵 melt 后的两列的名字，内部变量，variable 代表了点线的属性，value 代表对应的值。
p <- ggplot(data_m, aes(x=xvariable, y=value, color=variable)) + geom_line()
p
```

```
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
```

```
# 图会存储在当前目录的 Rplots.pdf 文件中，如果用 Rstudio，可以不运行 dev.off()
# dev.off()
```



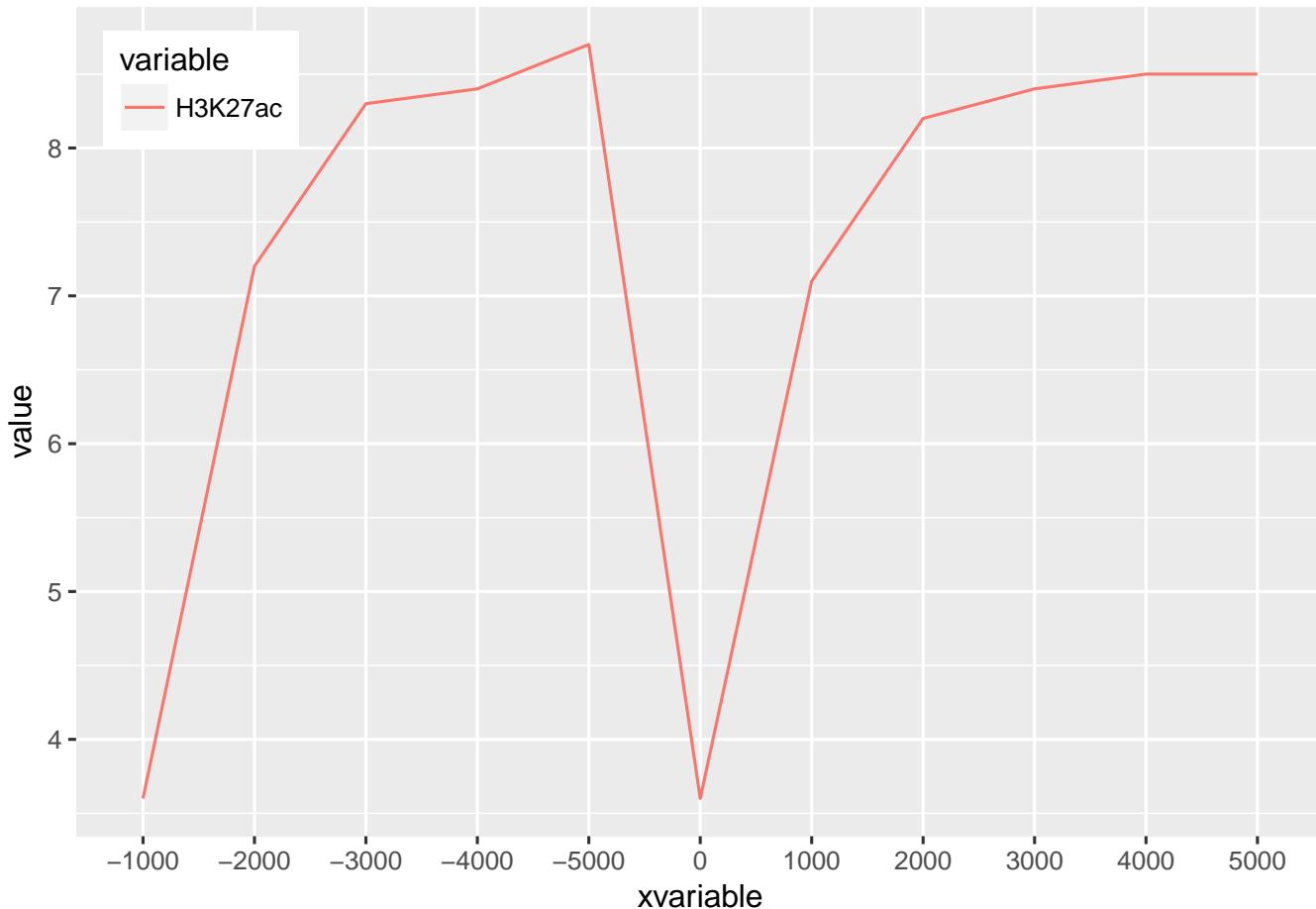
满心期待一个倒钟形曲线，结果，什么也没有。

仔细看，出来一段提示

```
geom_path: Each group consists of only one observation.
Do you need to adjust the group aesthetic?
```

原来默认 ggplot2 把每个点都视作了一个分组，什么都没画出来。而 `data_m` 中的数据都来源于一个分组 `H3K27ac`，分组的名字为 `variable`，修改下脚本，看看效果。

```
p <- ggplot(data_m, aes(x=xvariable, y=value, color=variable, group=variable)) +
  geom_line() + theme(legend.position=c(0.1, 0.9))
p
# dev.off()
```



图出来了，一条线，看一眼没问题；再仔细看，不对了，怎么还不是倒钟形，原来横坐标错位了。

检查下数据格式

```
summary(data_m)
```

```
##   xvariable      variable     value
##  Length:11    H3K27ac:11   Min.   :3.600
##  Class :character          1st Qu.:7.150
##  Mode   :character          Median :8.300
##                               Mean   :7.318
##                               3rd Qu.:8.450
##                               Max.   :8.700
```

问题来了，`xvariable` 虽然看上去数字，但存储的实际是字符串（因为是作为行名字读取的），需要转换为数字。

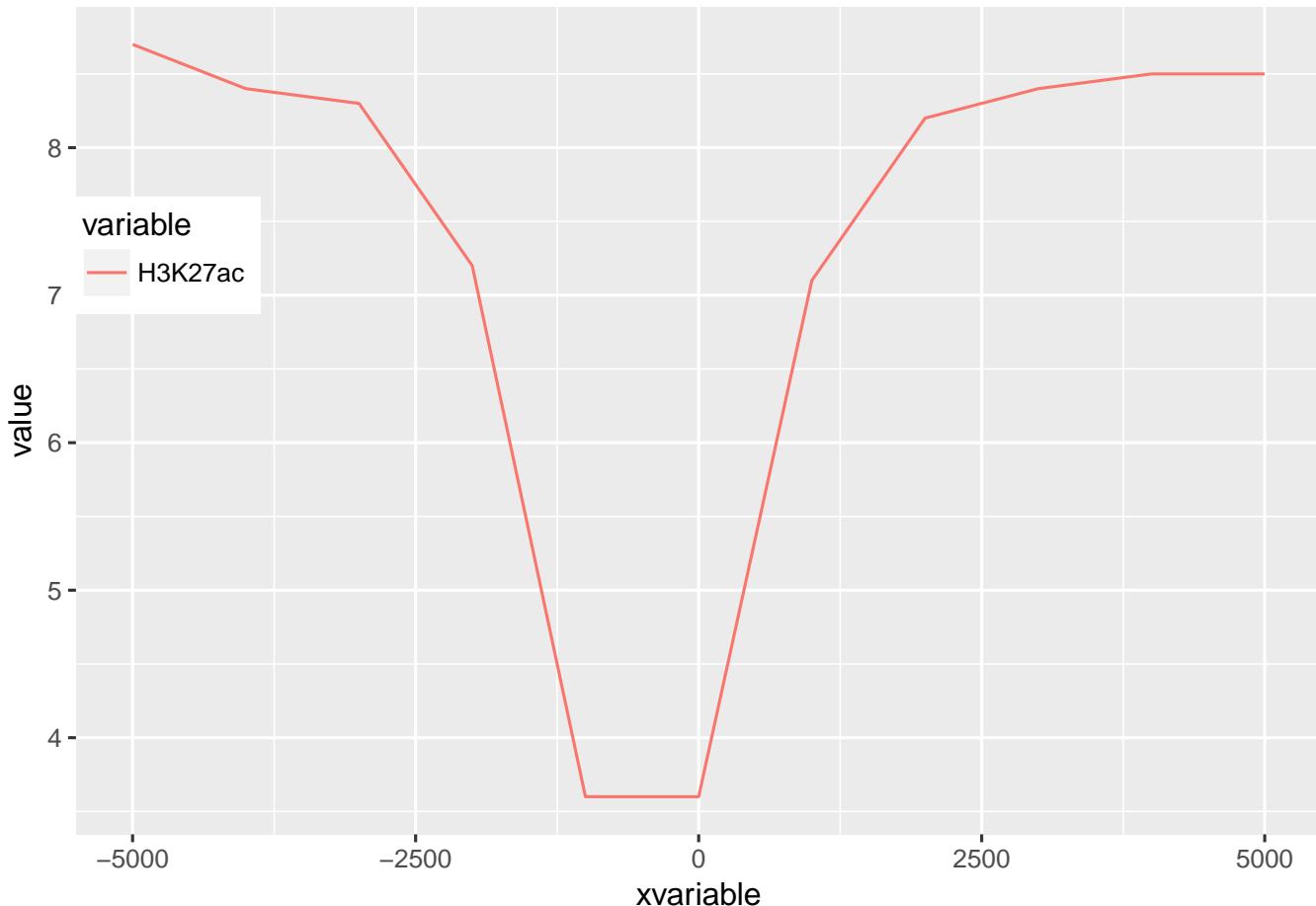
```
data_m$xvariable <- as.numeric(data_m$xvariable)
```

```
# 再检验下
summary(data_m)
```

```
##   xvariable      variable     value
## Min. :-5000  H3K27ac:11 Min. :3.600
## 1st Qu.:-2500                   1st Qu.:7.150
## Median :    0                   Median :8.300
## Mean   :    0                   Mean   :7.318
## 3rd Qu.: 2500                  3rd Qu.:8.450
## Max.  : 5000                  Max.  :8.700
```

好了，继续画图。

```
# 注意断行时，加号在行尾，不能放在行首
p <- ggplot(data_m, aes(x=xvariable, y=value,color=variable,group=variable)) +
  geom_line() + theme(legend.position=c(0.07,0.7))
p
#dev.off()
```



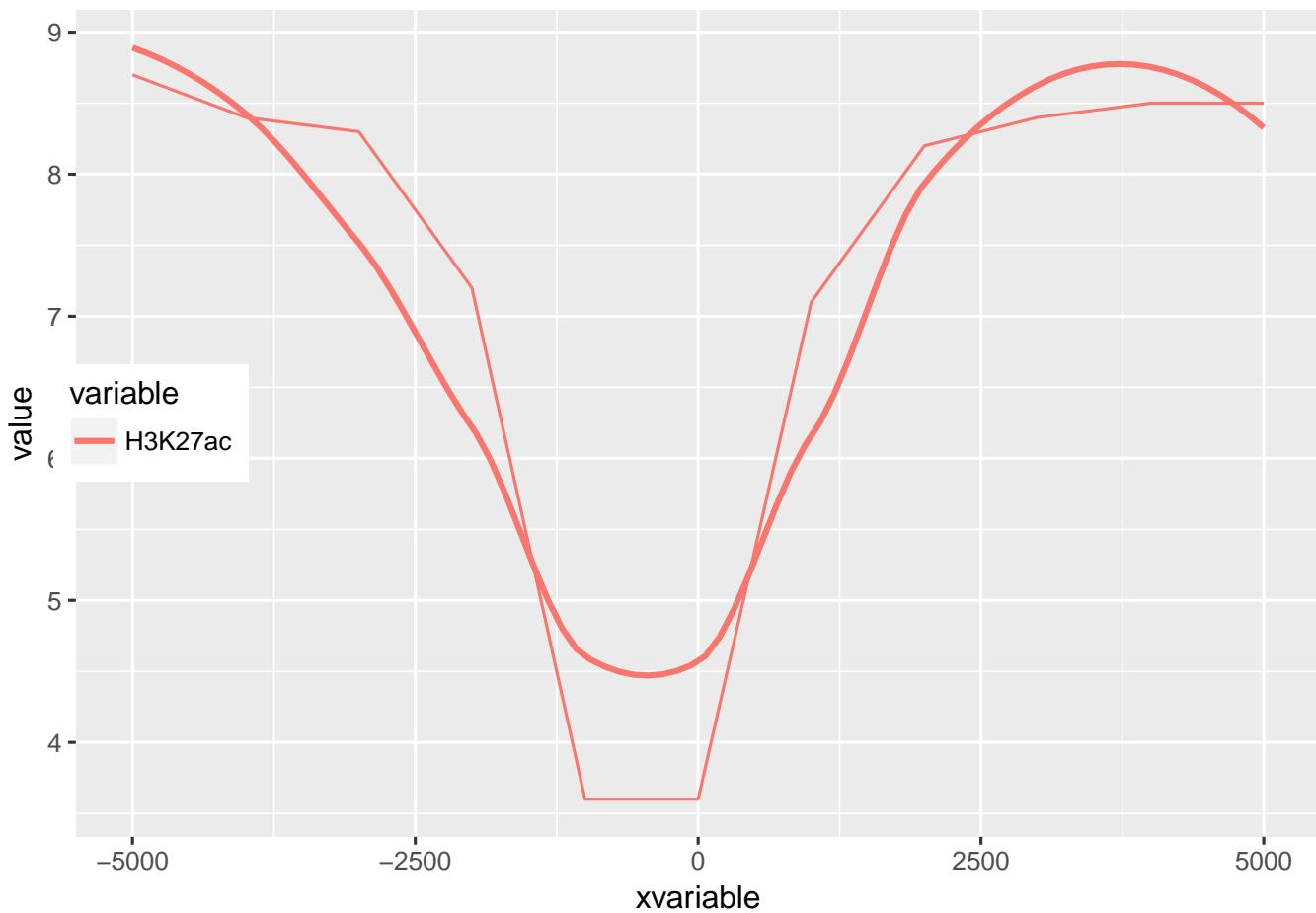
图终于出来了，调了下 legend 的位置，看上去有点意思了。

有点难看，如果平滑下，会不会好一些，`stat_smooth` 可以对绘制的线进行局部拟合。在不影响变化趋势的情况下，可以使用（但慎用）。

```
p <- ggplot(data_m, aes(x=xvariable, y=value, color=variable, group=variable)) +
  geom_line() + stat_smooth(method="auto", se=FALSE) +
  theme(legend.position=c(0.06, 0.5))
p
```

```
## `geom_smooth()` using method = 'loess'
```

```
#dev.off()
```



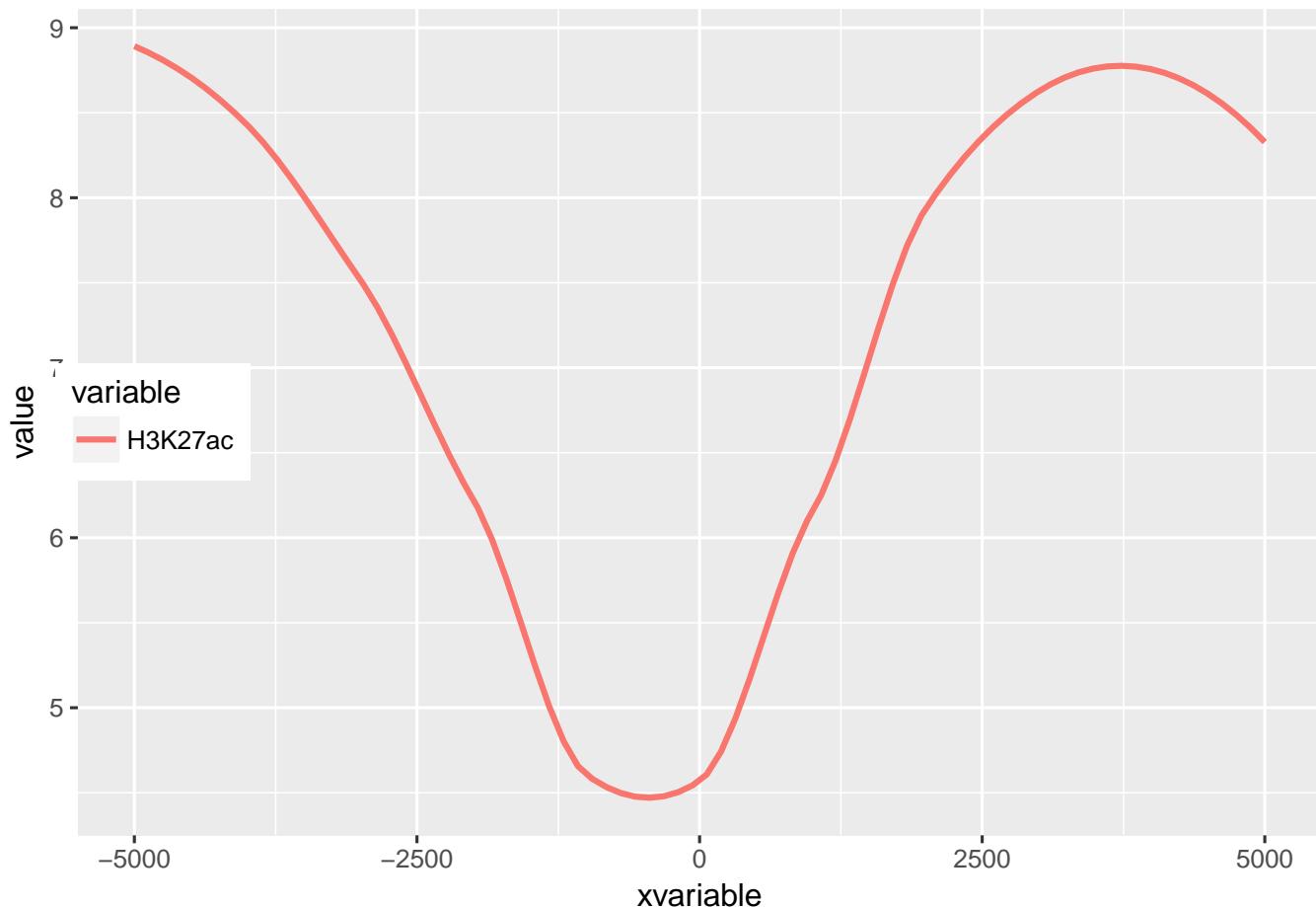
从图中看，趋势还是一致的，线条更优美了。另外一个方式是增加区间的数量，线也会好些，而且更真实。

`stat_smooth` 和 `geom_line` 各绘制了一条线，只保留一条就好。

```
p <- ggplot(data_m, aes(x=xvariable, y=value,color=variable,group=variable)) +
  stat_smooth(method="auto", se=FALSE) + theme(legend.position=c(0.06,0.5))
p
```

```
## `geom_smooth()` using method = 'loess'
```

```
#dev.off()
```



好了，终于完成了单条线图的绘制。

3.6.2 多线图

那么再来一个多线图的例子吧，只要给之前的数据矩阵多加几列就好了。

```
profile = "Pos;h3k27ac;ctcf;enhancer;h3k4me3;polII
-5000;8.7;10.7;11.7;10;8.3"
```

CONTENTS

```

-4000;8.4;10.8;11.8;9.8;7.8
-3000;8.3;10.5;12.2;9.4;7
-2000;7.2;10.9;12.7;8.4;4.8
-1000;3.6;8.5;12.8;4.8;1.3
0;3.6;8.5;13.4;5.2;1.5
1000;7.1;10.9;12.4;8.1;4.9
2000;8.2;10.7;12.4;9.5;7.7
3000;8.4;10.4;12;9.8;7.9
4000;8.5;10.6;11.7;9.7;8.2
5000;8.5;10.6;11.7;10;8.2"

profile_text <- read.table(text=profile, header=T, row.names=1, quote="", sep="; ")

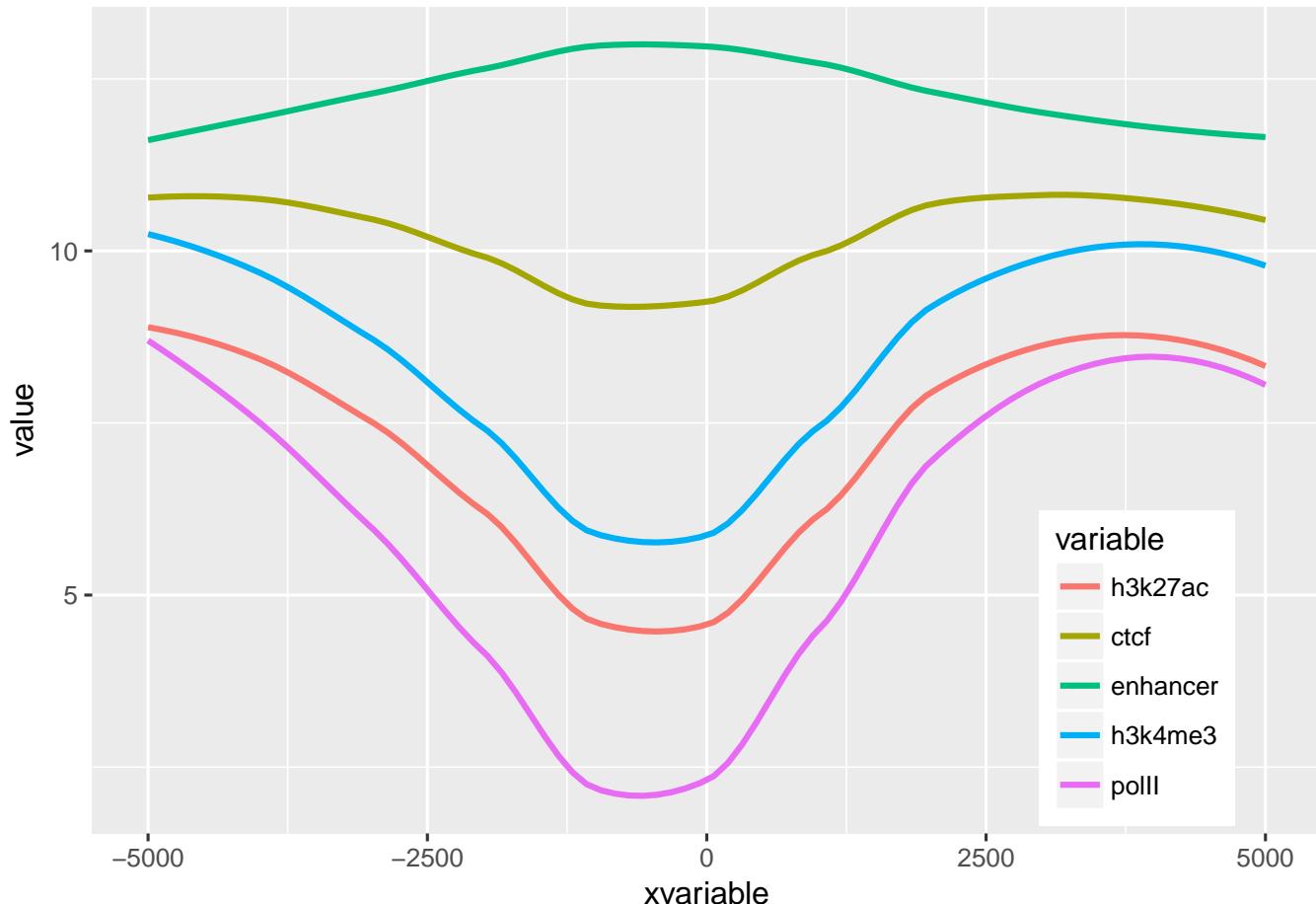
profile_text$xvariable = rownames(profile_text)
data_m <- melt(profile_text, id.vars=c("xvariable"))
data_m$xvariable <- as.numeric(data_m$xvariable)

# 这里 group=variable, 而不是 group=1 (如果上面你用的是 1 的话)
# variable 和 value 为矩阵 melt 后的两列的名字, 内部变量,
# variable 代表了点线的属性, value 代表对应的值。
p <- ggplot(data_m, aes(x=xvariable, y=value,color=variable,group=variable)) +
    stat_smooth(method="auto", se=FALSE) + theme(legend.position=c(0.85,0.2))
p

## `geom_smooth()` using method = 'loess'

#dev.off()

```



3.6.3 横轴文本线图

如果横轴是文本，又该怎么调整顺序呢？还记得之前热图旁的行或列的顺序调整吗？重新设置变量的 `factor` 水平就可以控制其顺序。

```
profile = "Pos;h3k27ac;ctcf;enhancer;h3k4me3;polII
-5000;8.7;10.7;11.7;10;8.3
-4000;8.4;10.8;11.8;9.8;7.8
-3000;8.3;10.5;12.2;9.4;7
-2000;7.2;10.9;12.7;8.4;4.8
-1000;3.6;8.5;12.8;4.8;1.3
0;3.6;8.5;13.4;5.2;1.5
1000;7.1;10.9;12.4;8.1;4.9
2000;8.2;10.7;12.4;9.5;7.7
3000;8.4;10.4;12;9.8;7.9
4000;8.5;10.6;11.7;9.7;8.2
5000;8.5;10.6;11.7;10;8.2"

profile_text <- read.table(text=profile, header=T, row.names=1, quote="", sep=";")
```

```
profile_text_rownames <- row.names(profile_text)

profile_text$xvariable = rownames(profile_text)
data_m <- melt(profile_text, id.vars=c("xvariable"))
data_m$xvariable <- factor(data_m$xvariable, levels=profile_text_rownames, ordered=T)

# geom_line 设置线的粗细和透明度
p1 <- ggplot(data_m, aes(x=xvariable, y=value, color=variable, group=variable)) +
  geom_line(size=1, alpha=0.9) + theme(legend.position=c(0.8,0.25)) +
  theme(axis.text.x=element_text(angle=45, hjust=1, vjust=1))

# stat_smooth
p2 <- ggplot(data_m, aes(x=xvariable, y=value, color=variable, group=variable)) +
  stat_smooth(method="auto", se=FALSE) + theme(legend.position=c(0.8,0.25)) +
  theme(axis.text.x=element_text(angle=45, hjust=1, vjust=1))

library("cowplot")

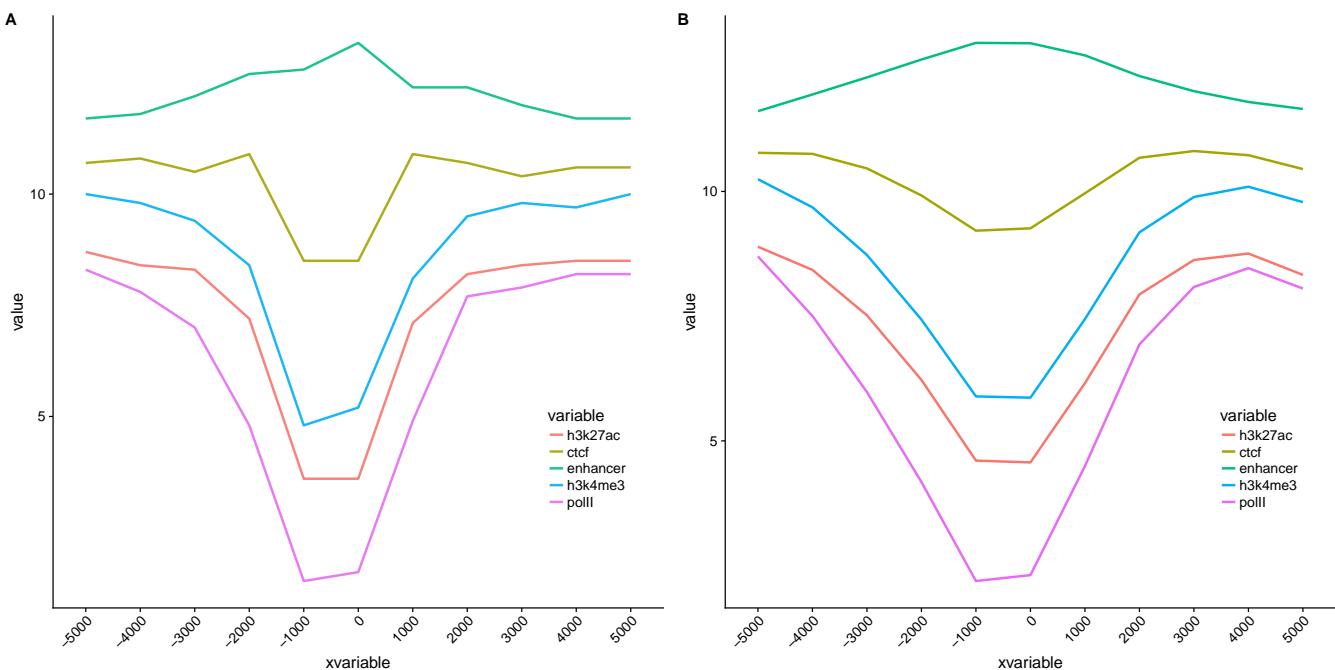
## 
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggplot2':
## 
##     ggsave

plot_grid(p1,p2, labels=c("A","B"), ncol=2, nrow=1)

## `geom_smooth()` using method = 'loess'

#dev.off()
```



比较下位置信息做为数字(前面的线图)和位置信息横轴的差别。当为数值时，ggplot2会选择合适的几个刻度做标记，当为文本时，会全部标记。另外文本横轴，smooth效果不明显。

3.7 散点图

散点图在生物信息分析中是应用比较广的一个图，常见的差异基因火山图、功能富集分析泡泡图、相关性分析散点图、抖动图、PCA 样品分类图等。凡是想展示分布状态的都可以用散点图。

3.7.1 横纵轴都为数字的散点图解析

绘制散点图的输入一般都是规规矩矩的矩阵，可以让不同的列分别代表 X 轴、Y 轴、点的大小、颜色、形状、名称等。

3.7.1.1 输入数据格式 (使用火山图的输入数据为例) 火山图用于展示基因表达差异的分布，横轴为 Log2 Fold Change，越偏离中心差异倍数越大；纵轴为 $(-1) * \text{Log10 P_adjust}$ ，值越大差异越显著。一般横轴越偏离中心的点其纵轴值也会比较大，因此呈现火山喷发的形状。

火山图需要的数据格式如下

- id: 不是必须的，但一般的软件输出结果中都会包含，表示基因名字。
- log2FoldChange: 差异倍数的对数，一般的差异分析输出结果中也会给出对数处理的值，因此程序没有提供这一步的计算操作。

CONTENTS

- **padj:** 多重假设检验矫正过的差异显著性 P 值；一般的差异分析输出结果为原始值，程序提供一个参数对其求取负对数。
- **significant:** 可选列，标记哪些基因是上调、下调、无差异；若无此列或未在参数中指定此列，默认程序会根据 padj 列和 log2FoldChange 列根据给定的阈值自动计算差异基因，并作出不同颜色的标记。
- **label:** 可选列，一般用于在图中标记出感兴趣的基因的名字。非-行的字符串都会标记在图上。

```
volcano = "id;log2FoldChange;padj;significant;label
E00007;4.28238;0;EH BIO_UP;A
E00008;-1.1036;0.476466843393901;Unchanged;-
E00009;-0.274368;1;Unchanged;-
E00010;4.62347;7.37606076333335e-103;EH BIO_UP;-
E00012;0.973987;0.482982440163204;Unchanged;-
E00017;-1.30205;0.000555693857439792;Baodian_UP;B
E00024;0.617636;2.78047837287061e-13;Unchanged;-
E00033;1.48669;2.56000581595275e-60;EH BIO_UP;-
E00034;-0.783716;0.00341521725291801;Unchanged;-
E00036;2.01592;6.03136656016401e-06;EH BIO_UP;C
E00040;-1.89657;4.73663890849056e-21;Baodian_UP;-
E00041;-0.268168;0.563429434558031;Unchanged;-
E00042;0.0861048;0.367700939634328;Unchanged;-
E00043;-1.19328;1.42673872027352e-153;Baodian_UP;-
E00044;-0.887981;2.43067804654905e-26;Unchanged;-
E00047;-0.610941;5.51696648645932e-57;Unchanged;-
"
```

数据的读取之前的 R 语言统计和绘图系列都已解释过，不再赘述

文末也有链接可直达之前的文章，新学者建议从头开始

```
#volcanoData <- read.table(text=volcano, sep=";", header=T, quote="", check.names=F)
volcanoData <- read.table("data/volcano.txt", sep="\t", header=T,
                           row.names=NULL, quote="", check.names=F)
```

head(volcanoData)

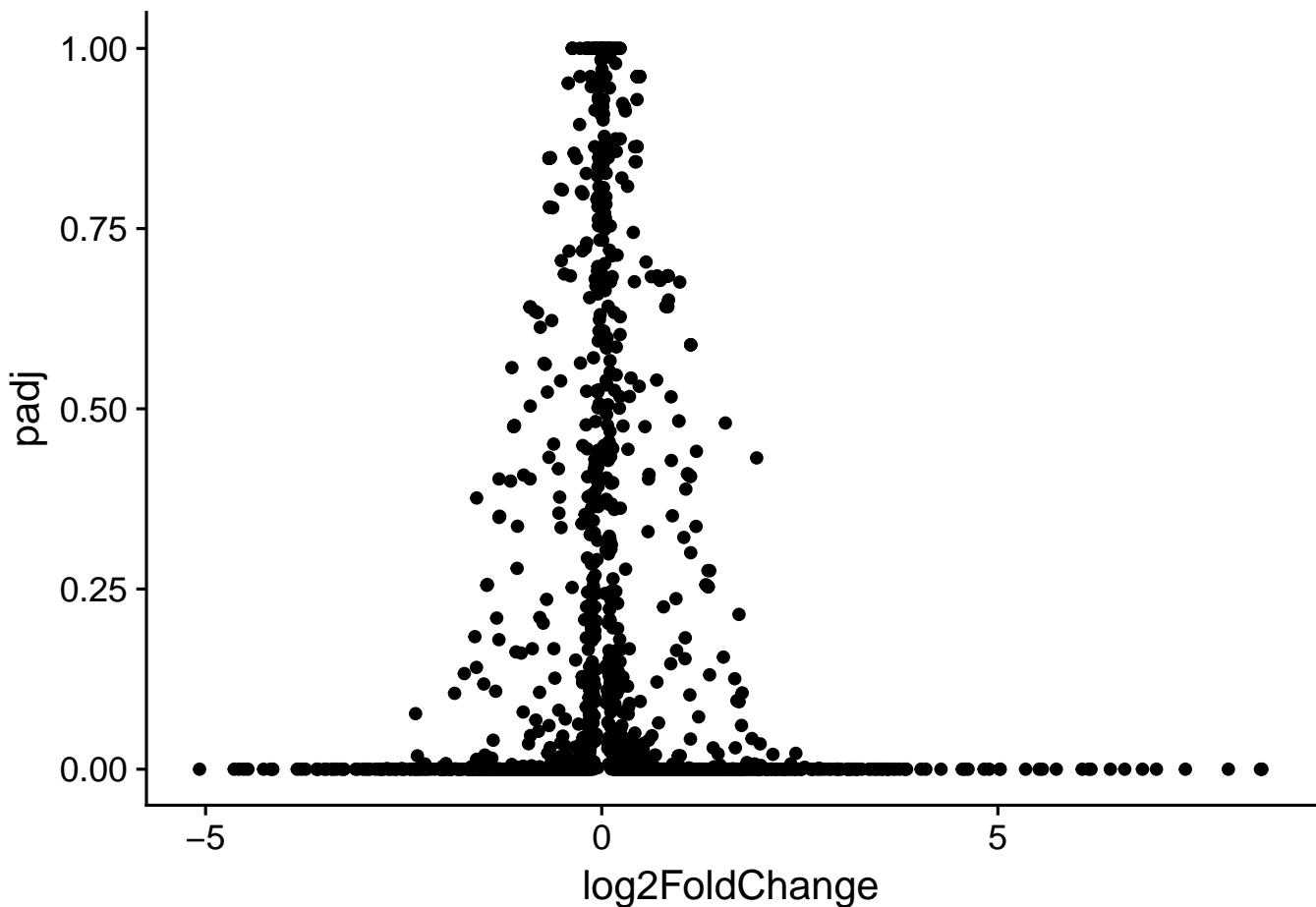
```
##      id log2FoldChange          padj significant label
## 1 E00007     4.282380 0.000000e+00    EH BIO_UP     A
## 2 E00008    -1.103600 4.764668e-01    Unchanged     -
## 3 E00009    -0.274368 1.000000e+00    Unchanged     -
## 4 E00010     4.623470 7.376061e-103   EH BIO_UP     -
## 5 E00012     0.973987 4.829824e-01    Unchanged     -
## 6 E00017    -1.302050 5.556939e-04  Baodian_UP     B
```

dim(volcanoData)

[1] 1985 5

绘制散点图，只需要指定 X 轴和 Y 轴，再加上 geom_point 即可。

```
#library(ggplot2)
p <- ggplot(volcanoData, aes(x=log2FoldChange, y=padj))
p <- p + geom_point()
# 前面是给 p 不断添加图层的过程
# 单输入一个 p 是真正作图
# 前面有人说，上面都输完了，怎么没出图
# 就因为差了一个 p
p
```



说好的火山图的例子，但怎么也看不出喷发的态势。

对数据坐下预处理，差异大的基因 `padj` 小，先对其求取负对数，所谓负负得正，差异大的基因就会处于图的上方了。

```
# 从示例数据中看到，最小的 padj 值为 0，求取负对数为正无穷。
# 实际上 padj 值小到一个点对我们来讲就是个数
# 所以可以给所有小于 1e-6 的 padj 都让其等于 1e-6，再小也没意义
#
volcanoData[volcanoData$padj<1e-6, "padj"] <- 1e-6
volcanoData$padj <- (-1)* log10(volcanoData$padj)
```

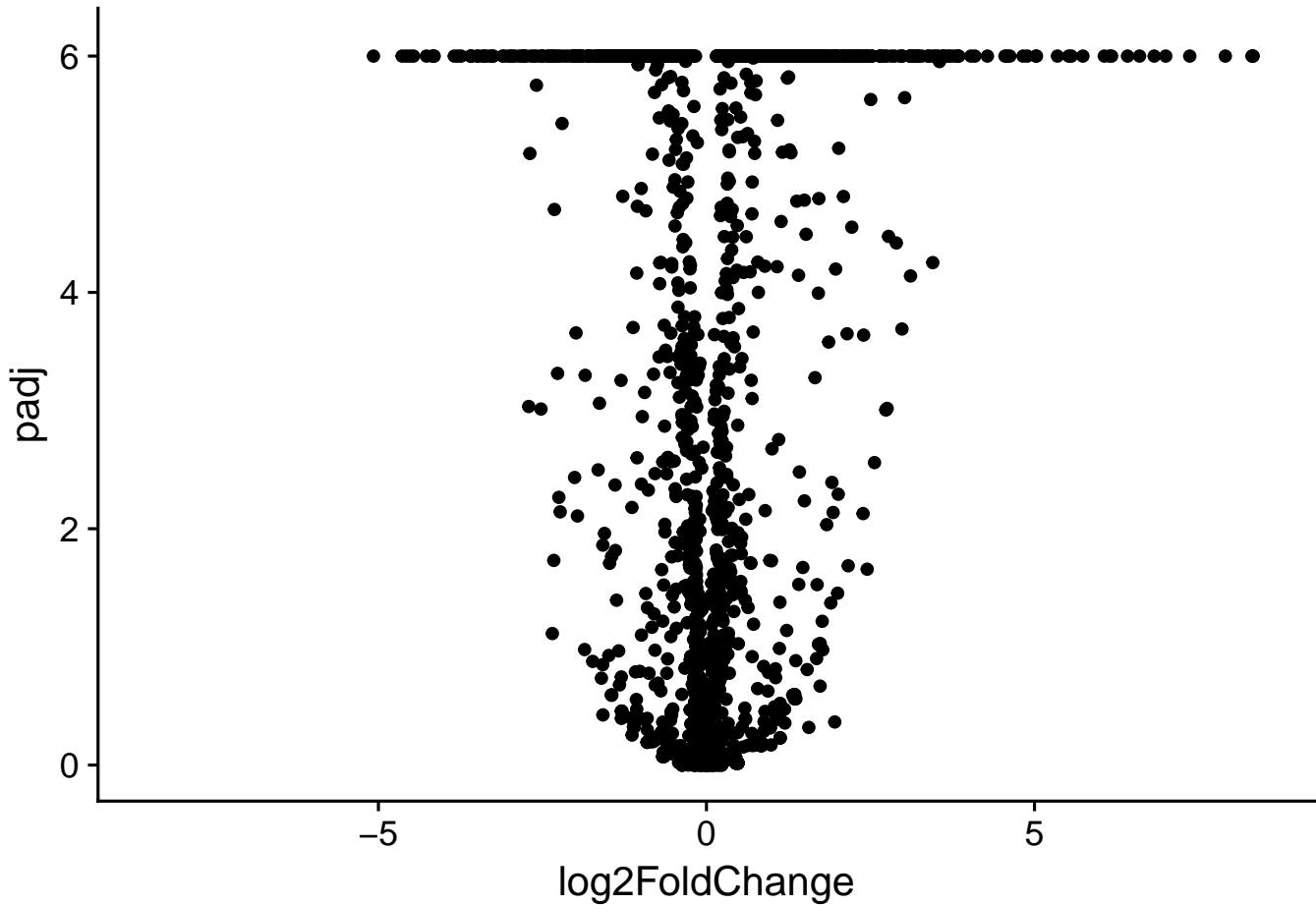
CONTENTS

```
summary(volcanoData)
```

```
##      id      log2FoldChange      padj      significant
## E00007 : 1   Min. :-5.07612   Min. :0.0000 Baodian_UP: 201
## E00008 : 1   1st Qu.:-0.52292  1st Qu.:0.9619 EHBIo_UP : 270
## E00009 : 1   Median : 0.03515  Median :5.4514 Unchanged :1514
## E00010 : 1   Mean   : 0.10928  Mean   :3.7426
## E00012 : 1   3rd Qu.: 0.58246  3rd Qu.:6.0000
## E00017 : 1   Max.   : 8.33085  Max.   :6.0000
## (Other):1979
## label
## -:1981
## A: 1
## B: 1
## C: 1
## D: 1
##
##
```

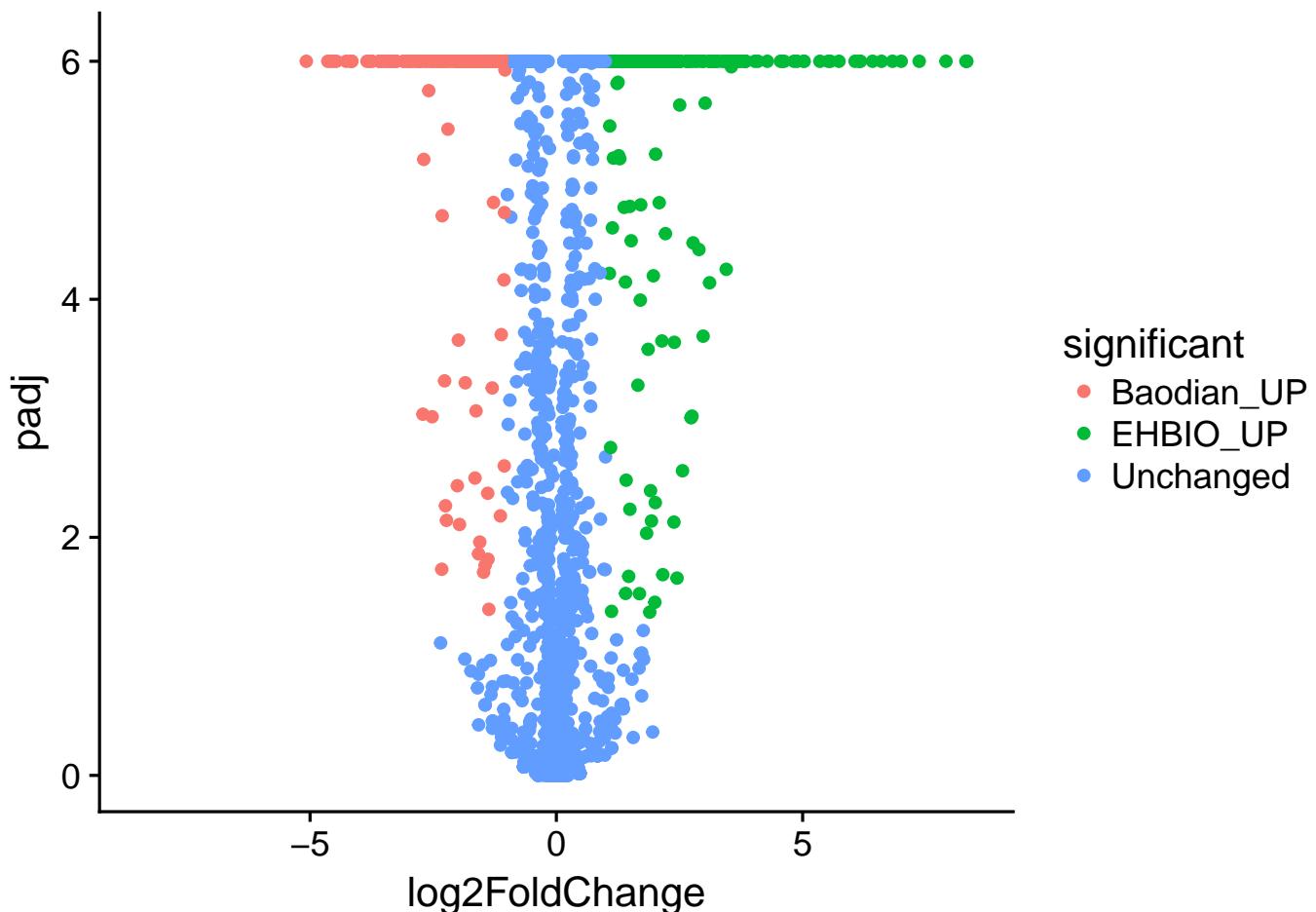
数据中基因的上调倍数远高于下调倍数，使得出来的图是偏的，这次画图时调整下 X 轴的区间使图对称。

```
log2fc_max_abs = max(abs(volcanoData$log2FoldChange)) + 0.1
padj_max = max(volcanoData$padj) + 0.1
p <- ggplot(volcanoData, aes(x=log2FoldChange, y=padj)) +
  geom_point() + xlim(-1*log2fc_max_abs, log2fc_max_abs) + ylim(0, padj_max)
p
```

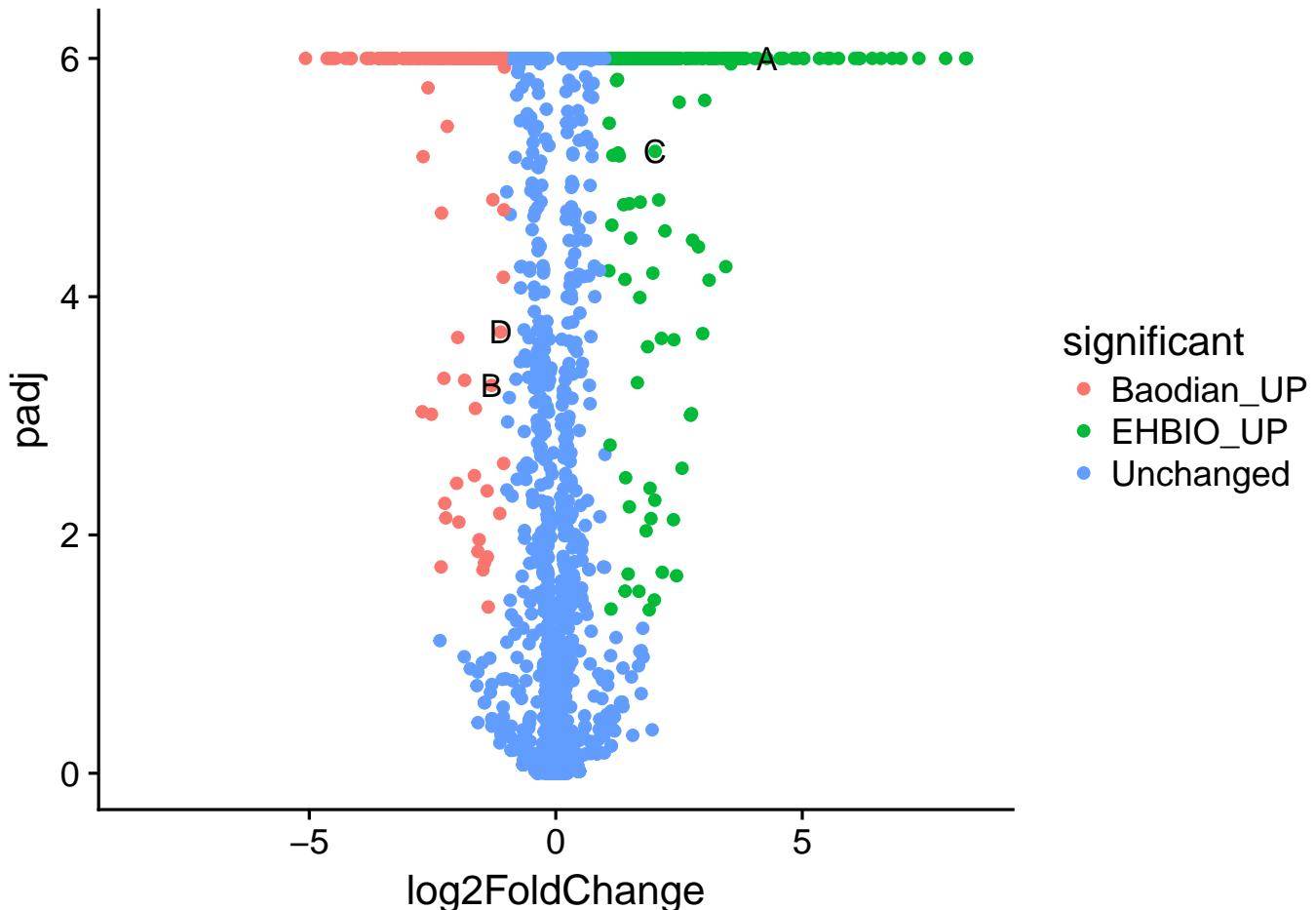


有点意思了，数据太少不明显，下一步加上颜色看看。

```
p <- ggplot(volcanoData, aes(x=log2FoldChange, y=padj)) +  
  geom_point(aes(color=significant)) +  
  xlim(-1*log2fc_max_abs, log2fc_max_abs) + ylim(0, padj_max)  
p
```



```
volcanoData[volcanoData$label=="-", "label"] = NA
p <- ggplot(volcanoData, aes(x=log2FoldChange, y=padj)) +
  geom_point(aes(color=significant)) + xlim(-1*log2fc_max_abs, log2fc_max_abs) +
  ylim(0,padj_max) +
  geom_text(aes(label=label))
p
```



利用现有的数据，基本上就是这个样子了。虽然还不太像，原理都已经都点到了。

```

volcanoData$sig <- ifelse(
  volcanoData$padj>1.30103,
  ifelse(
    volcanoData$log2FoldChange>=1,
    "UP",
    ifelse(
      volcanoData$log2FoldChange<=-1,
      "DW",
      "NoDiff")
  ),
  "NoDiff")

volcanoData$sig <- factor(volcanoData$sig, levels=c("UP", "DW", "NoDiff"))
summary(volcanoData)

```

	id	log2FoldChange	padj	significant
##	E00007 :	1	Min. : -5.07612	Min. : 0.0000 Baodian_UP: 201
##	E00008 :	1	1st Qu.: -0.52292	1st Qu.: 0.9619 EHBIO_UP : 270
##	E00009 :	1	Median : 0.03515	Median : 5.4514 Unchanged : 1514

CONTENTS

```
## E00010 : 1 Mean : 0.10928 Mean : 3.7426
## E00012 : 1 3rd Qu.: 0.58246 3rd Qu.: 6.0000
## E00017 : 1 Max. : 8.33085 Max. : 6.0000
## (Other):1979
##   label          sig
## - : 0    UP : 271
## A : 1    DW : 201
## B : 1    NoDiff:1513
## C : 1
## D : 1
## NA's:1981
##
```

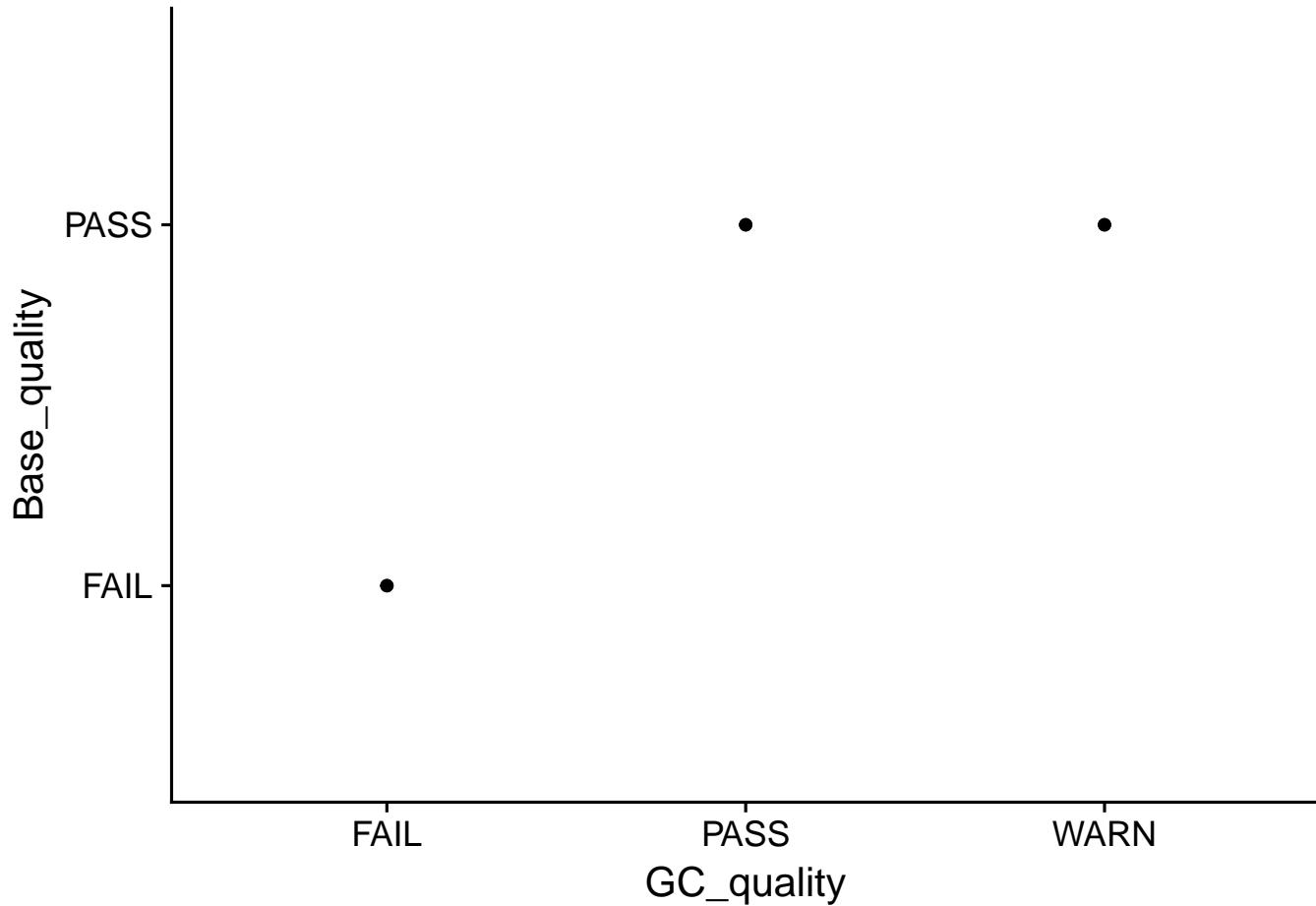
3.7.2 横纵轴都为字符串的散点图展示

3.7.2.1 输入数据格式如下 这个数据是 FASTQC 结果总结中的直观的查看所有样品测序碱基质量和 GC 含量的散点图的示例数据。

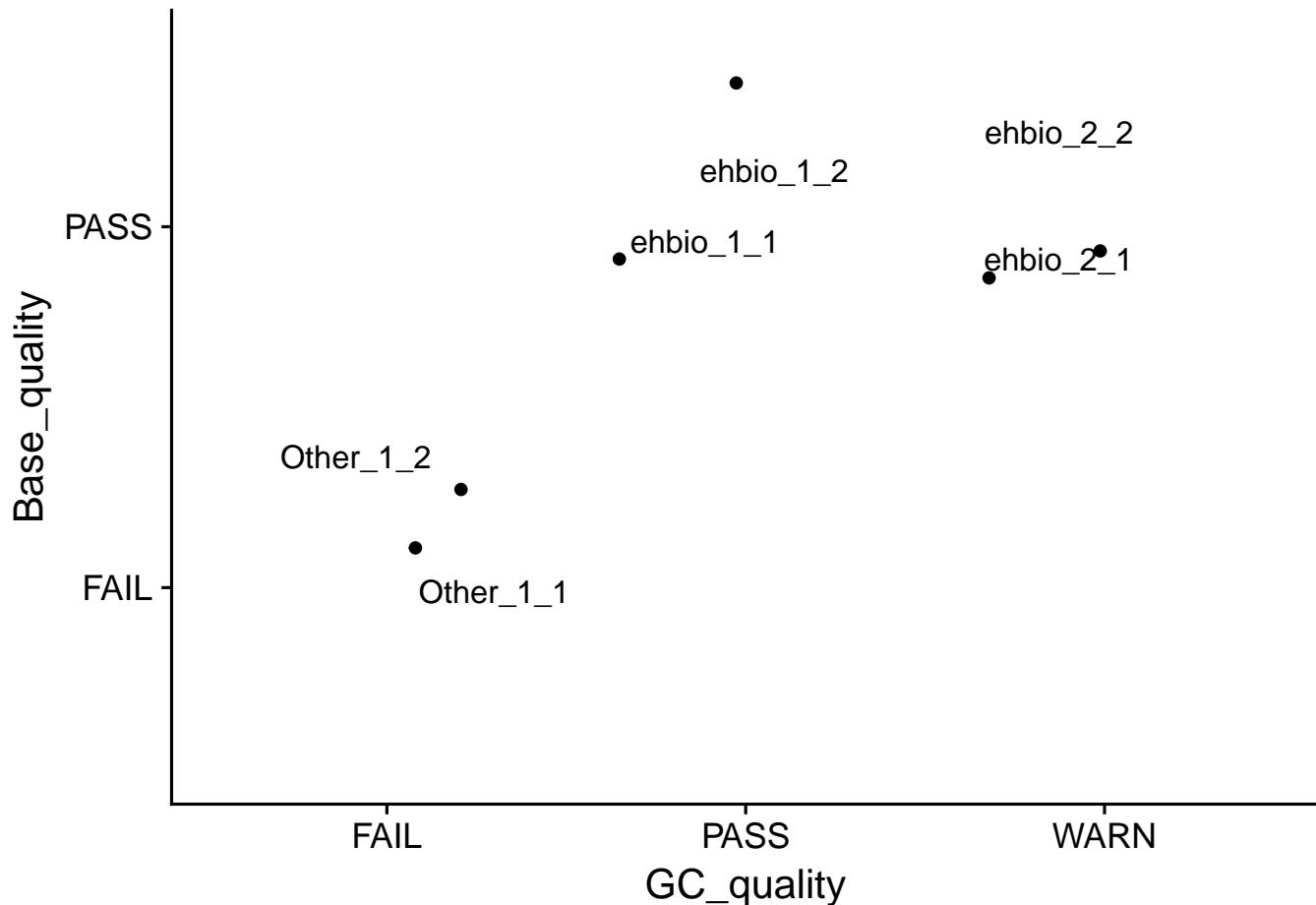
```
fastqc<-"ID;GC_quality;Base_quality
ehbio_1_1;PASS;PASS
ehbio_1_2;PASS;PASS
ehbio_2_1;WARN;PASS
ehbio_2_2;WARN;PASS
Other_1_1;FAIL;FAIL
Other_1_2;FAIL;FAIL"

fastqc_data <- read.table(text=fastqc, sep=";", header=T)
# 就不查看了

p <- ggplot(fastqc_data, aes(x=GC_quality, y=Base_quality)) + geom_point()
p
```

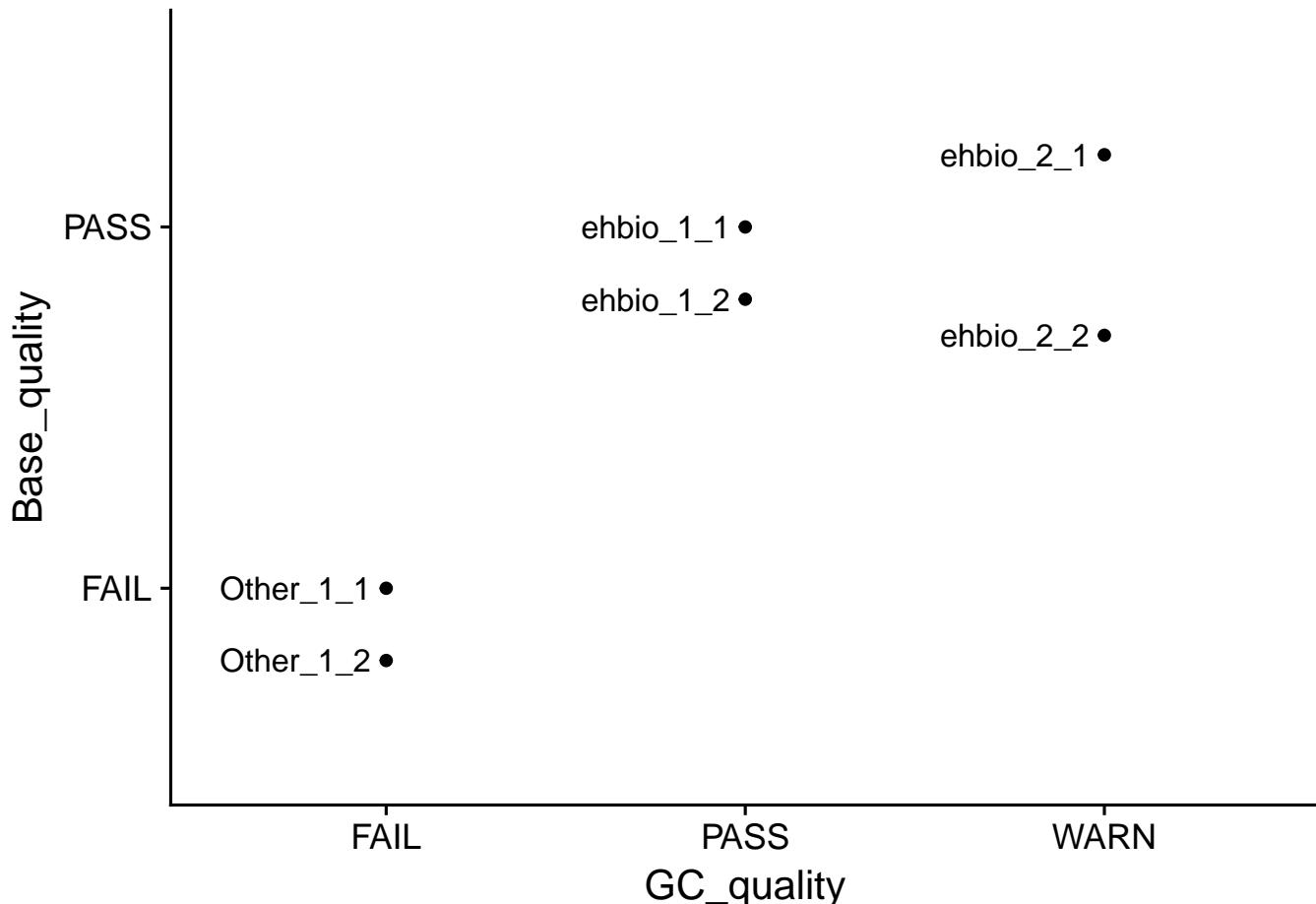


```
p <- ggplot(fastqc_data, aes(x=GC_quality, y=Base_quality)) +  
  geom_jitter() + geom_text(aes(label=ID), position="jitter")  
p
```



六个点少了只剩下了 3 个，重叠在一起了，而且也不知道哪个点代表什么样品。这时需要把点抖动下，用到一个包 ggbeeswarm，抖动图的神器。

```
library(ggbeeswarm)
p <- ggplot(fastqc_data, aes(x=GC_quality, y=Base_quality)) +
  geom_quasirandom(groupOnX=FALSE)
# 使用 geom_text 增加点的标记
# label 表示标记哪一列的数值
# position_quasirandom 获取点偏移后的位置
# xjust 调整对齐方式; hjust 是水平的对齐方式，0 为左，1 为右，0.5 居中，0-1 之间可以取任意值。
# vjust 是垂直对齐方式，0 底对齐，1 为顶对齐，0.5 居中，0-1 之间可以取任意值。
# check_overlap 检查名字在图上是否重叠，若有重叠，只显示一个
p <- p + geom_text(aes(label=ID), position=position_quasirandom(groupOnX=FALSE),
                    hjust=1.1, check_overlap=T)
p
```



3.8 功能富集泡泡图

功能富集分析用来展示某一组基因 (一般是单个样品上调或下调的基因) 倾向参与哪些功能调控通路，对从整体理解变化了的基因的功能和潜在的调控意义具有指导作用，也是文章发表中一个有意义的美图。通常会用柱状图、泡泡图和热图进行展示。热图的画法之前已经介绍过，这次介绍下富集分析泡泡图，其展示的信息是最为全面的，也是比较抓人眼球的。

假设有一个富集分析结果矩阵 (文件名为 GOenrichement.xls) 存储了 EHBIOS 样品和 Baodian 样品中各自上调的基因富集的通路。

<http://omicslab.genetics.ac.cn/GOEAST>

- Description 为 GO 通路的描述，也可以是 KEGG 通路。
- GeneRatio 为对应通路差异基因占总差异基因的比例，本列可以用分数或小数表示，都可以处理。
- qvalue 表示对应通路富集的显著性程度，可以是 log 处理过的，也可以是原始的。
- Count 为对应通路差异基因数目。
- Type 这个矩阵合并了 EHBIOS 样品和 Baodian 样品中各自上调的基因富集的通路，用 Type 列做区分。如果只有一个样品可不要。

CONTENTS

Description	GeneRatio	qvalue	Count	Type
ERBB signaling pathway	7/320	0.001836081	7	EHBIO_up
regulation of ERBB signaling pathway	5/320	0.003886659	5	EHBIO_up
negative regulation of cell cycle G1/S phase transition	4/320	0.016153254	4	EHBIO_up
Wnt signaling pathway	13/320	0.01680096	13	EHBIO_up
cell-cell signaling by wnt	13/320	0.0171473	13	EHBIO_up
negative regulation of cell cycle process	8/320	0.019453085	8	EHBIO_up
extrinsic apoptotic signaling pathway	9/320	0.024164034	9	EHBIO_up
positive regulation of extrinsic apoptotic signaling pathway	4/320	0.025708228	4	EHBIO_up
cell cycle G1/S phase transition	7/320	0.035797856	7	EHBIO_up
negative regulation of apoptotic signaling pathway	8/320	0.038684745	8	EHBIO_up
regulation of Notch signaling pathway	4/320	0.041592045	4	EHBIO_up
regulation of cell cycle G1/S phase transition	5/320	0.047407619	5	EHBIO_up
negative regulation of BMP signaling pathway	3/320	0.049460847	3	EHBIO_up
regulation of ERK1 and ERK2 cascade	14/342	0.000629602	14	Baodian_up
positive regulation of cell adhesion	17/342	0.000827275	17	Baodian_up
ERK1 and ERK2 cascade	14/342	0.001086508	14	Baodian_up
regulation of cell growth	17/342	0.002228511	17	Baodian_up
positive regulation of cytoskeleton organization	10/342	0.004406867	10	Baodian_up
regulation of cell-cell adhesion	15/342	0.005075219	15	Baodian_up
regulation of cytoskeleton organization	15/342	0.019685646	15	Baodian_up
negative regulation of Notch signaling pathway	3/342	0.020578211	3	Baodian_up
neuron apoptotic process	10/342	0.040284925	10	Baodian_up

```
enrichment = read.table("data/GOenrichement.xls", header=T, row.names=NULL,
  sep="\t", quote="")
head(enrichment)
```

```
##                                     Description GeneRatio
## 1                      ERBB signaling pathway    7/320
## 2      regulation of ERBB signaling pathway    5/320
## 3 negative regulation of cell cycle G1/S phase transition    4/320
## 4                      Wnt signaling pathway   13/320
## 5      cell-cell signaling by wnt       13/320
## 6 negative regulation of cell cycle process     8/320
##   qvalue Count Type
## 1 0.001836081    7 EHBIO_up
## 2 0.003886659    5 EHBIO_up
## 3 0.016153254    4 EHBIO_up
## 4 0.016800960   13 EHBIO_up
## 5 0.017147300   13 EHBIO_up
## 6 0.019453085    8 EHBIO_up
```

3.8.1 单样品分开绘制

示例矩阵中包含两个样品上调基因的富集通路，现在先取出一个样品绘制。

```
enrichment_sxbd = droplevels(enrichment[enrichment$type=="Baodian_up", ])
```

构造一个函数，转换分数为小数。

```
library(plyr)
library(stringr)
library(ggplot2)
library(grid)

mixedToFloat <- function(x){
  x <- sapply(x, as.character)
  is.integer <- grep("^-?\\d+$", x)
  is.fraction <- grep("^-?\\d+\\.\\d+$", x)
  is.float <- grep("^-?\\d+\\.\\d+\\.\\d+$", x)
  is.mixed <- grep("^-?\\d+ \\d+\\.\\d+\\.\\d+$", x)
  stopifnot(all(is.integer | is.fraction | is.float | is.mixed))

  numbers <- strsplit(x, "[ /]")

  ifelse(is.integer, as.numeric(sapply(numbers, `[, 1])),
  ifelse(is.float, as.numeric(sapply(numbers, `[, 1])),
  ifelse(is.fraction, as.numeric(sapply(numbers, `[, 1)) /
    as.numeric(sapply(numbers, `[, 2)),
    as.numeric(sapply(numbers, `[, 1)) +
    as.numeric(sapply(numbers, `[, 2)) /
    as.numeric(sapply(numbers, `[, 3))))))

}
```

```
mixedToFloat(c('1 1/2', '2 3/4', '2/3', '11 1/4', '1'))
```

```
## [1] 1.5000000 2.7500000 0.6666667 11.2500000 1.0000000
```

转换数据列为小数或整数

```
enrichment_sxbd$GeneRatio = mixedToFloat(enrichment_sxbd$GeneRatio)
enrichment_sxbd$Count = mixedToFloat(enrichment_sxbd$Count)
```

qvalue 转换负对数，并作为新的一列

```
# qvalue 转换
log_name = paste0("negLog10_", "qvalue")
col_name_enrichment_sxbd <- colnames(enrichment_sxbd)
col_name_enrichment_sxbd <- c(col_name_enrichment_sxbd, log_name)
enrichment_sxbd$log_name <- log10(enrichment_sxbd$qvalue) * (-1)
colnames(enrichment_sxbd) <- col_name_enrichment_sxbd
```

Term 排序

1. 获取 Term 出现的次数

```
# 计算每个 Term 出现的顺序，用于排序，出现次数最多的排在前面
enrichment_sxbd_freq <- as.data.frame(table(enrichment_sxbd$Description))
colnames(enrichment_sxbd_freq) <- c("Description", "IDctct")
head(enrichment_sxbd_freq)
```

```
##                                     Description IDctct
## 1           ERK1 and ERK2 cascade      1
## 2 negative regulation of Notch signaling pathway 1
## 3          neuron apoptotic process   1
## 4          positive regulation of cell adhesion 1
## 5 positive regulation of cytoskeleton organization 1
## 6          regulation of cell growth     1
```

根据出现次数、GeneRatio、-log10(qvalue) 排序

```
enrichment_sxbd2 <- merge(enrichment_sxbd, enrichment_sxbd_freq, by="Description")

# 首先根据出现次数排序、然后根据 GeneRatio、然后根据-log10(qvalue)
enrichment_sxbd3 <- enrichment_sxbd2[order(enrichment_sxbd2$IDctct,
  enrichment_sxbd2$GeneRatio, enrichment_sxbd2$negLog10_qvalue), ]

term_order <- unique(enrichment_sxbd3$Description)

# 设置排序顺序
enrichment_sxbd$Description <- factor(enrichment_sxbd$Description,
  levels=term_order, ordered=T)
```

```
color_v <- c("green", "red")

# 指定 x,y
p <- ggplot(enrichment_sxbd, aes(x=GeneRatio, y=Description)) +
```

```


 labs(x="GeneRatio", y="GO description") + labs(title="")

p <- p + geom_point(aes(size=Count, color=negLog10_qvalue)) +
  scale_colour_gradient(low=color_v[1], high=color_v[2], name="negLog10_qvalue")

# Term 单行长度不超过 60 字符
p <- p + scale_y_discrete(labels=function(x) str_wrap(x, width=60))

p <- p + theme_bw() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())

top='top'
bottom='bottom'
left='left'
right='right'
none='none'
legend_pos_par <- right

uwid = 0
vhig = 12

# 自动估算图形长宽
if (uwid == 0 || vhig == 0) {
  x_len = length(unique(enrichment_sxbd$Description))
  if(x_len<10) {
    vhig = 10
  } else if(x_len<20) {
    vhig = 10 + (x_len-10)/3
  } else if(x_len<100) {
    vhig = 13 + (x_len-20)/5
  } else {
    vhig = 40
  }
  uwid = vhig
  if(legend_pos_par %in% c("left", "right")){
    uwid = 1.5 * uwid
  }
}

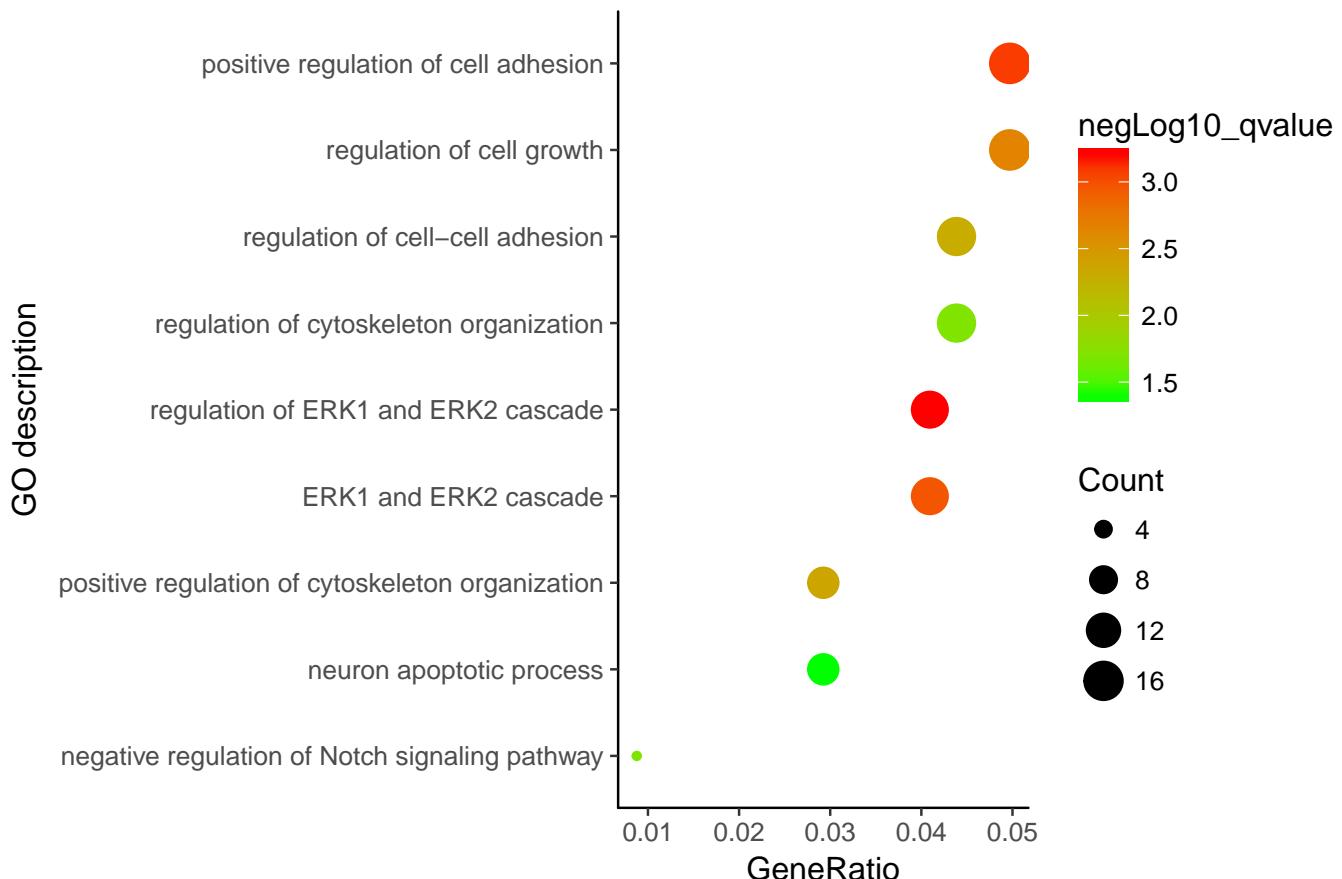
p <- p + theme(legend.position=legend_pos_par)

p <- p + theme( panel.grid = element_blank(), panel.border=element_blank(),
  legend.background = element_blank(),
  axis.line.x=element_line(size=0.4, colour="black", linetype='solid'),
  axis.line.y=element_line(size=0.4, colour="black", linetype='solid'),


```

```
axis.ticks = element_line(size=0.4)
)
```

```
#ggsave(p, filename="GOenrichement.ehbio.xls.scatterplot.dv.pdf", dpi=300, width=uwid,
#height=vhig, units=c("cm"))
p
```



3.8.2 多样品合并绘制

```
#enrichment$type <- factor(enrichment$type, levels=sample_ho, ordered=T)

# First order by Term, then order by Sample
enrichment <- enrichment[order(enrichment$Description, enrichment$type), ]

enrichment$GeneRatio = mixedToFloat(enrichment$GeneRatio)
enrichment$Count = mixedToFloat(enrichment$Count)
```

```

log_name = paste0("negLog10_", "qvalue")
col_name_enrichment <- colnames(enrichment)
col_name_enrichment <- c(col_name_enrichment, log_name)
enrichment$log_name <- log10(enrichment$qvalue) * (-1)
colnames(enrichment) <- col_name_enrichment

# Get the count of each unique Term
enrichment_freq <- as.data.frame(table(enrichment$Description))
colnames(enrichment_freq) <- c("Description", "IDctct")
enrichment2 <- merge(enrichment, enrichment_freq, by="Description")

# 增加一列，样品信息用于排序
enrichment_samp <- ddply(enrichment2, "Description", summarize,
    sam_ct_ct_ct=paste(Type, collapse="_"))

enrichment2 <- merge(enrichment2, enrichment_samp, by="Description")

# 排序与上面相同，但增加了按样品组合排序
enrichment3 <- enrichment2[order(enrichment2$IDctct, enrichment2$sam_ct_ct_ct,
    enrichment2$Type, enrichment2$GeneRatio, enrichment2$negLog10_qvalue), ]
#print(enrichment3)

term_order <- unique(enrichment3$Description)

enrichment$Description <- factor(enrichment$Description, levels=term_order, ordered=T)

#print(enrichment)
rm(enrichment_freq, enrichment2, enrichment3)

color_v <- c("green", "red")

p <- ggplot(enrichment, aes(x=GeneRatio, y=Description)) +
    labs(x="GeneRatio", y="GO description") + labs(title="")

p <- p + geom_point(aes(size=Count, color=negLog10_qvalue, shape=Type)) +
    scale_colour_gradient(low=color_v[1], high=color_v[2], name="negLog10_qvalue")

p <- p + scale_y_discrete(labels=function(x) str_wrap(x, width=60))

p <- p + theme_bw() + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank())

p <- p + theme(axis.text.x=element_text(angle=45, hjust=0.5, vjust=1))

top='top'
bottom='bottom'
left='left'

```

```
right='right'
none='none'
legend_pos_par <- right

uwid = 0
vhig = 12

if (uwid == 0 || vhig == 0) {
  x_len = length(unique(enrichment$Description))
  if(x_len<10) {
    vhig = 10
  } else if(x_len<20) {
    vhig = 10 + (x_len-10)/3
  } else if(x_len<100) {
    vhig = 13 + (x_len-20)/5
  } else {
    vhig = 40
  }
  uwid = vhig
  if(legend_pos_par %in% c("left", "right")){
    uwid = 1.5 * uwid
  }
}

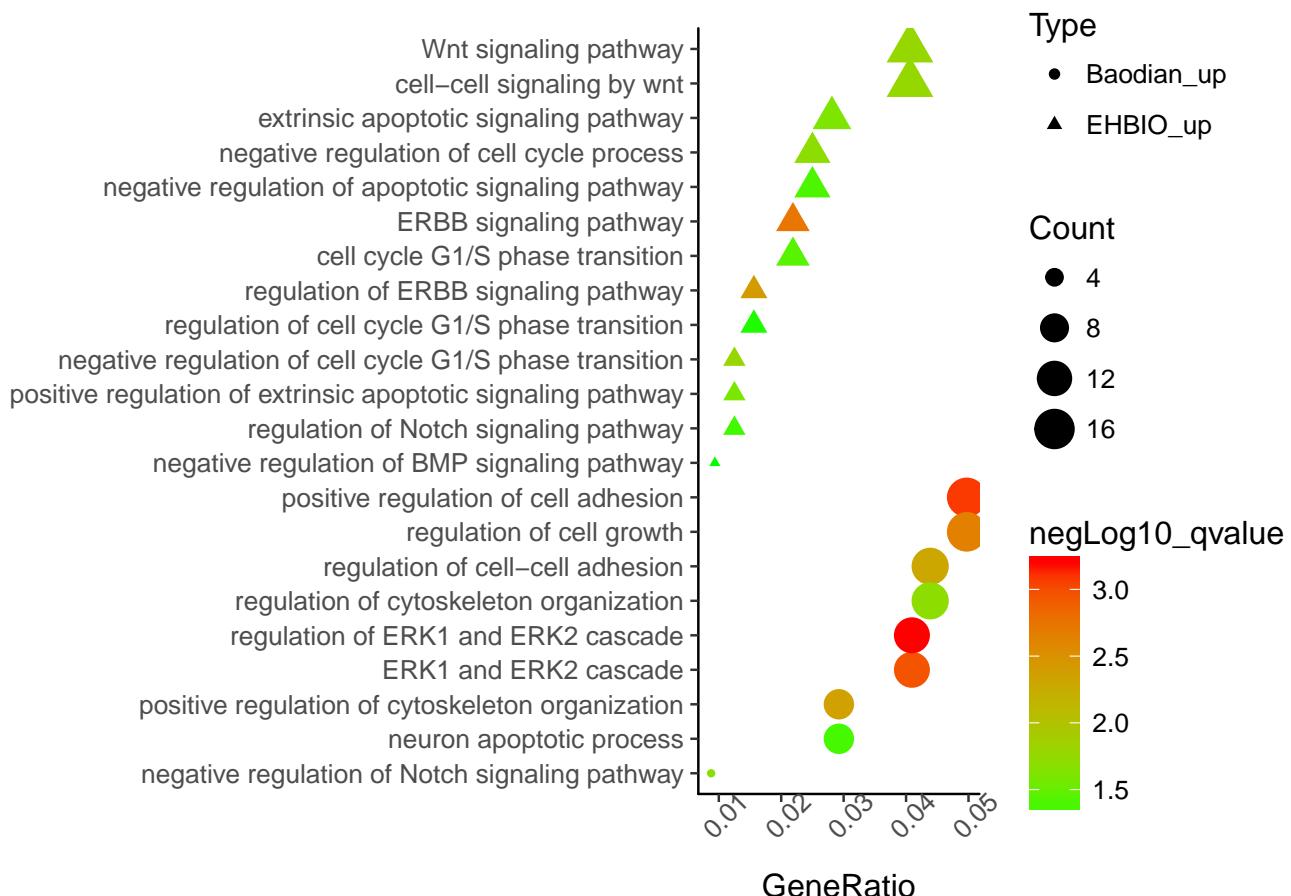
p <- p + theme(legend.position=legend_pos_par)

p <- p + theme( panel.grid = element_blank(),
  panel.border=element_blank(),
  legend.background = element_blank(),
  axis.line.x=element_line(size=0.4, colour="black", linetype='solid'),
  axis.line.y=element_line(size=0.4, colour="black", linetype='solid'),
  axis.ticks = element_line(size=0.4)
)

#ggsave(p, filename="GOenrichement.xls.scatterplot.dv.pdf", dpi=300, width=uwid,
#height=vhig, units=c("cm"))

p
```

GO description



通过这张图解释下，富集分析的结果怎么解读。富集分析实际是查找哪些通路里面包含的差异基因占总差异基因的比例显著高于通路中总基因占所有已经注释的基因的比例。这一显著性通常用多重假设检验矫正过的 pvalue(又称 qvalue, FDR 或 p.adjust) 来表示。在图中体现为点的颜色。从绿到红富集显著性逐渐增高。点的大小表示对应通路中包含的差异基因的数目。点的形状代表了不同类型的基因，如 EH BIO 中上调的基因和 Baodian 中上调的基因。横轴表示对应通路包含的差异基因占总的差异基因的比例，本图中最高不过 5%，这个值越大说明通路被影响的越多。

3.9 韦恩图

维恩图是用来反映不同集合之间的交集和并集情况的展示图。一般用于展示 2-5 个集合之间的交并关系。集合数目更多时，将会比较难分辨，更多集合的展示方式一般使用 upSetView。

较早的文章列举了多个在线工具 <http://mp.weixin.qq.com/s/zn654JqG9OeO71rJUTDr2Q>。

3.9.1 韦恩图三个圈

```
library(VennDiagram)
```

CONTENTS

```

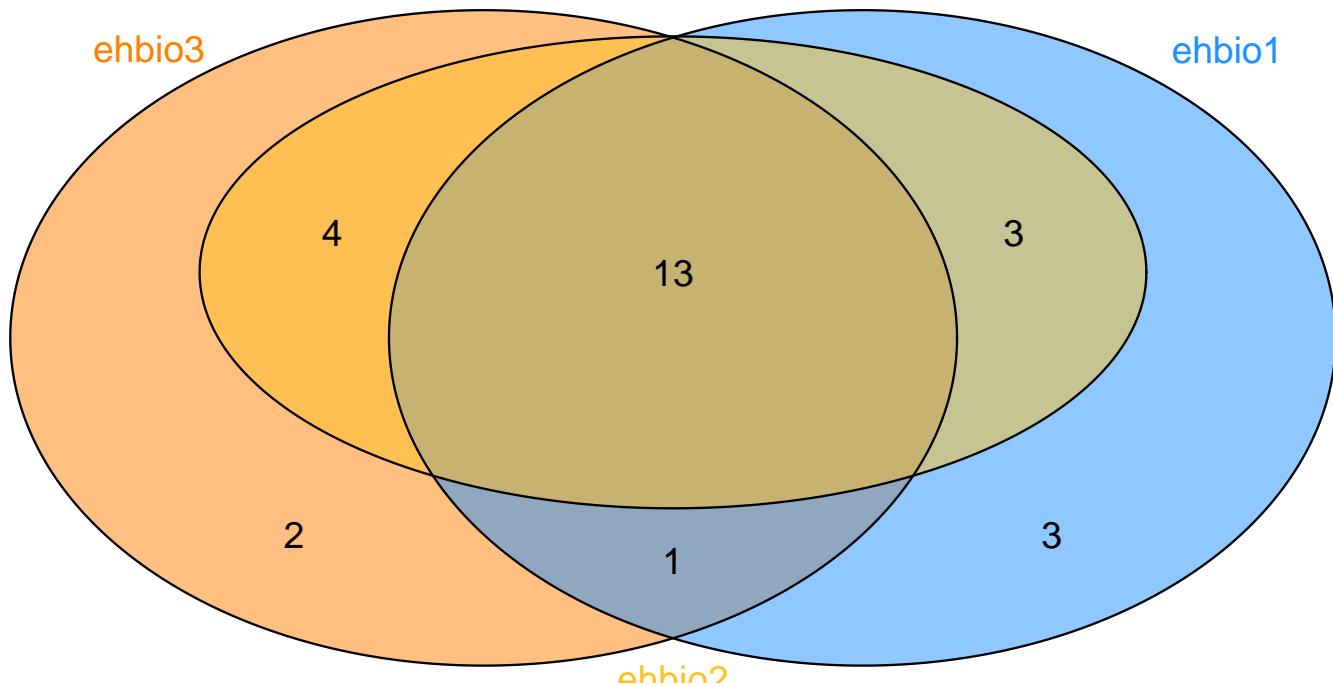
## Loading required package: futile.logger

list1 = sample(letters, 20)
list2 = sample(letters, 20)
list3 = sample(letters, 20)

color_v <- c("dodgerblue", "goldenrod1", "darkorange1")

label_size = 1
margin = 0.1
p <- venn.diagram(
  x = list(ehbio1=list1, ehbio2=list2, ehbio3=list3),
  filename = NULL, col = "black", lwd = 1,
  fill = color_v, alpha = 0.50, main="",
  label.col = c("black"), cex = 1, fontfamily = "Helvetica",
  cat.col = color_v, cat.cex = label_size,
  margin=margin,
  cat.fontfamily = "Helvetica"
)
grid.draw(p)

```



3.9.2 韦恩图五个圈

假设有这么一个矩阵，第一列为不同集合中的 ID，第二列为集合的名字，无标题行，存储为 `venn.txt`。

CONTENTS

```
a  ehbio1
b  ehbio1
c  ehbio1
d  ehbio1
e  ehbio1
f  ehbio1
g  ehbio1
h  ehbio2
i  ehbio2
j  ehbio2
k  ehbio2
e  ehbio2
f  ehbio2
g  ehbio2
a  ehbio3
b  ehbio3
h  ehbio3
j  ehbio3
i  ehbio3
f  ehbio3
g  ehbio3
a  ehbio4
b  ehbio4
h  ehbio4
d  ehbio5
e  ehbio5
y  ehbio5
x  ehbio5
```

```
library(VennDiagram)

#pdf(file="venn.txt.vennDiagram.pdf", onefile=FALSE, paper="special")

data <- read.table(file="data/venn.txt", sep="\t", quote="")

num <- 0

ehbio1 <- data[grep1("\\<ehbio1\\>",data[,2]),1]
num <- num + 1

ehbio2 <- data[grep1("\\<ehbio2\\>",data[,2]),1]
num <- num + 1

ehbio3 <- data[grep1("\\<ehbio3\\>",data[,2]),1]
num <- num + 1

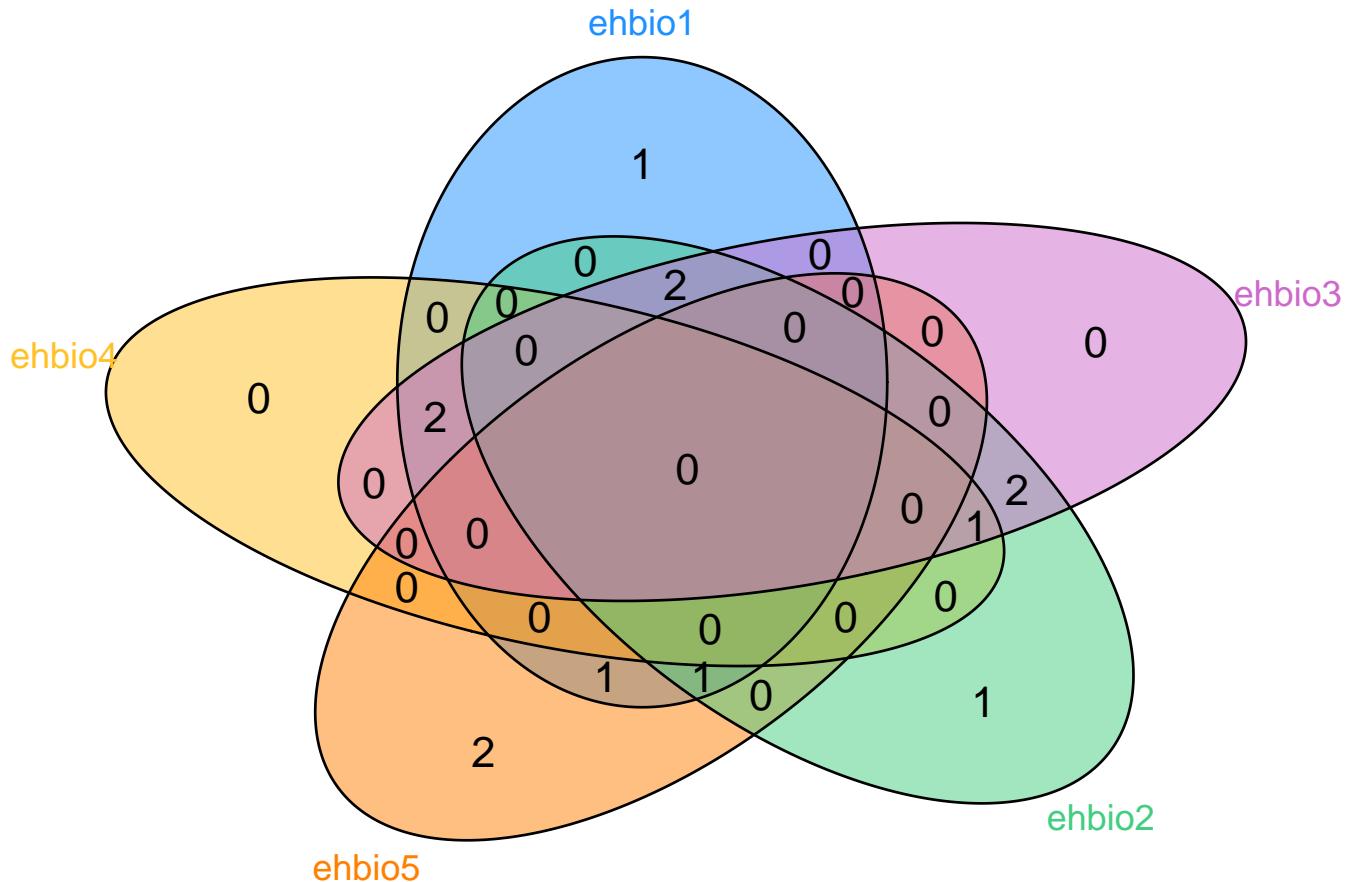
ehbio4 <- data[grep1("\\<ehbio4\\>",data[,2]),1]
```

```
num <- num + 1

ehbio5 <- data[grep1("\\\<ehbio5\\\>", data[,2]),1]
num <- num + 1

color_v <- c("dodgerblue", "goldenrod1", "darkorange1", "seagreen3", "orchid3") [1:num]
# label.col = c("orange", "white", "darkorchid4", "white", "white", "white", "white",
# "white", "darkblue", "white", "white", "white", "darkgreen", "white"),
# "white", "darkblue", "white", "white", "white", "darkgreen", "white"),

label_size = 0.8
margin = 0.3
p <- venn.diagram(
  x = list(ehbio1=ehbio1, ehbio4=ehbio4,
    ehbio5=ehbio5, ehbio2=ehbio2,
    ehbio3=ehbio3),
  filename = NULL, col = "black", lwd = 1,
  fill = color_v,
  alpha = 0.50, main="",
  label.col = c("black"),
  cex = 1, fontfamily = "Helvetica",
  cat.col = color_v, cat.cex = label_size,
  margin=margin,
  cat.fontfamily = "Helvetica"
)
grid.draw(p)
#dev.off()
```



3.9.3 UpSetView 展示

对于集合比较多的时候，包括上面提到的 5 个集合的交并集情况，如果只是为了展示个炫图，还可以，但如果想解释结果，就会比较头疼，难判断区域的归属。

因此对于这种多集合情况，推荐使用 UpSetView 展示，看效果如下。

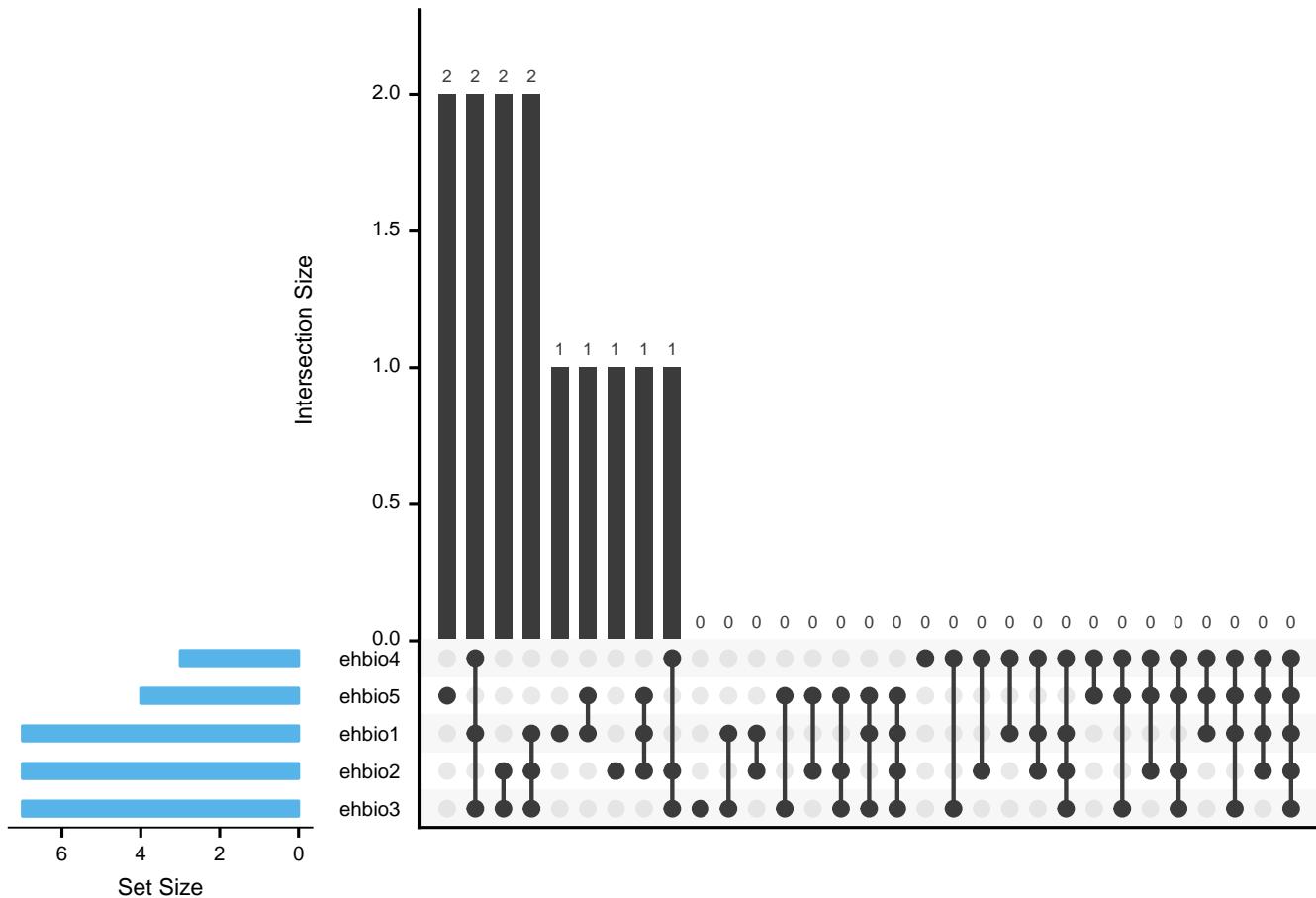
测试数据，存储为 `upsetview.txt`（第一行为集合的名，每个集合一列；每一行为一个 ID，如果对应 ID 出现在这个集合则标记 1，否则标记 0）：

pattern	ehbio1	ehbio2	ehbio3	ehbio4	ehbio5
a	1	0	1	1	0
b	1	0	1	1	0
c	1	0	0	0	0
d	1	0	0	0	1
e	1	1	0	0	1
f	1	1	1	0	0
g	1	1	1	0	0
h	0	1	1	1	0
i	0	1	1	0	0

```
j 0 1 1 0 0
k 0 1 0 0 0
x 0 0 0 0 1
y 0 0 0 0 1
```

```
library(UpSetR)
matrix = read.table("data/upsetview.txt", header=T, row.names=NULL, sep="\t")
nsets = dim(matrix)[2]-1

#pdf(file="upsetview.txt.upsetV.pdf", onefile=FALSE, paper="special", width=10,
#height=5, bg="white", pointsize=12)
upset(matrix, nsets=nsets, sets.bar.color = "#56B4E9", order.by = "freq",
      empty.intersections = "on")
#dev.off()
```



UpSetR: <http://www.caleydo.org/tools/upset/> 采用连线的方式展示不同的组合之间共有的和特有的项目，对于特别多的组合尤其适用。

单个点表示特有，连起来的点表示共有，相当于 venn 图中重叠的部分。

垂直的柱子代表的是 Venn 图中的数字，看连接的点判断归属。

水平的柱子代表对应样品中 Item 的总数。

3.10 柱状图绘制

柱状图也是较为常见的一种数据展示方式，可以展示基因的表达量，也可以展示 GO 富集分析结果，基因注释数据等。[39 个转录组分析工具，120 种组合评估 \(转录组分析工具哪家强-导读版\)](#)中提到了较多堆积柱状图的使用。下面就详细介绍下怎么绘制。

3.10.1 常规矩阵柱状图绘制

有如下 4 个基因在 5 组样品中的表达值

```
data_ori <- "Grp_1;Grp_2;Grp_3;Grp_4;Grp_5
a;2.6;2.9;2.1;2.0;2.2
b;20.8;9.8;7.0;3.7;19.2
c;10.0;11.0;9.2;12.4;9.6
d;9;3.3;10.3;11.1;10"

data <- read.table(text=data_ori, header=T, row.names=1, sep=";", quote="")
```

```
##   Grp_1  Grp_2  Grp_3  Grp_4  Grp_5
## a    2.6    2.9    2.1    2.0    2.2
## b   20.8    9.8    7.0    3.7   19.2
## c   10.0   11.0    9.2   12.4    9.6
## d     9.0    3.3   10.3   11.1   10.0
```

整理数据格式，保留基因名字信息

```
library(ggplot2)
library(reshape2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
```

CONTENTS

```

##      arrange, count, desc, failwith, id, mutate, rename, summarise,
##      summarize

## The following objects are masked from 'package:stats':
##      filter, lag

## The following objects are masked from 'package:base':
##      intersect, setdiff, setequal, union

data_rownames <- rownames(data)
data_colnames <- colnames(data)
data$gene <- data_rownames
data_m <- melt(data, id.vars=c("gene"))
data_m

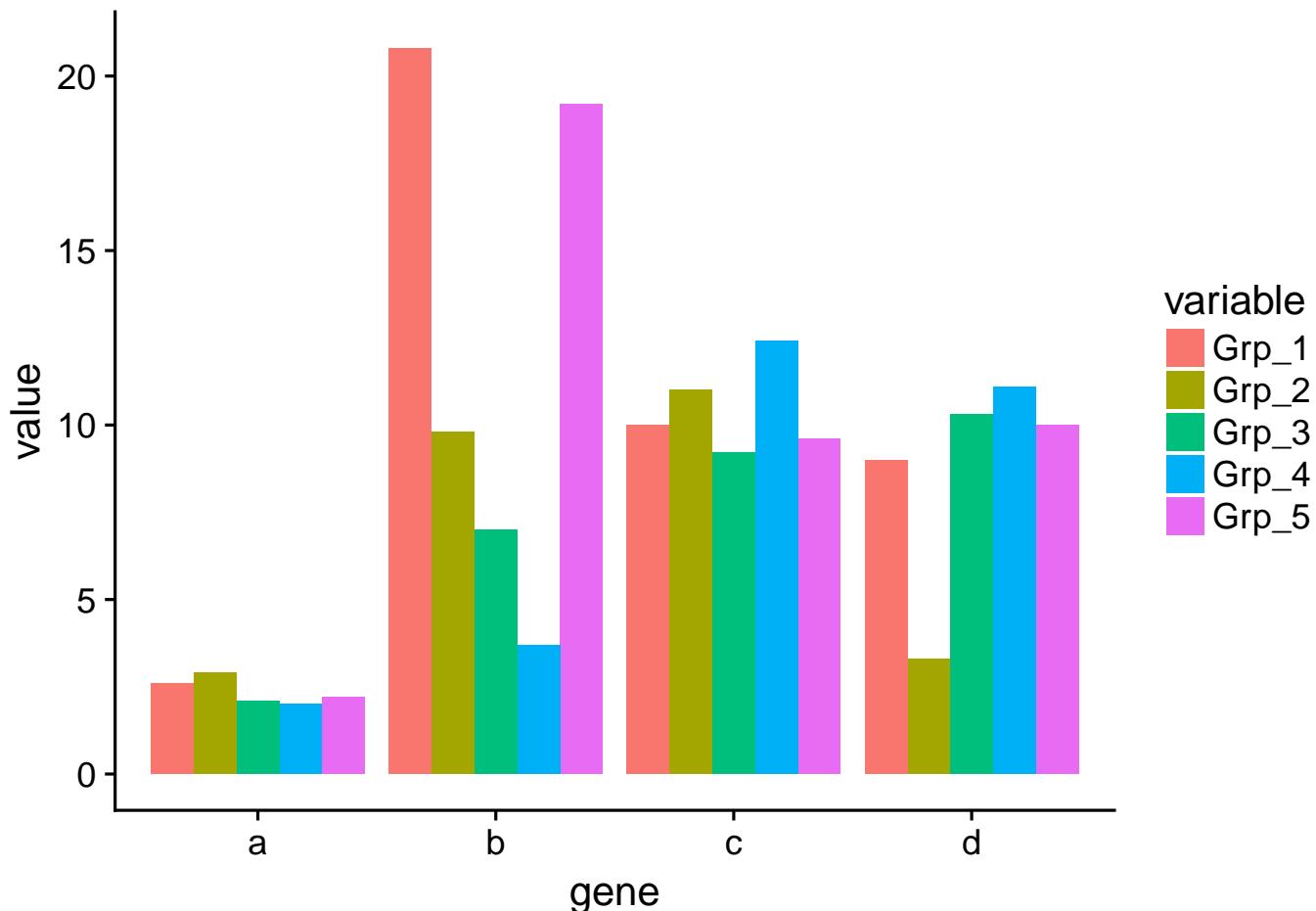
##      gene variable value
## 1     a    Grp_1   2.6
## 2     b    Grp_1  20.8
## 3     c    Grp_1  10.0
## 4     d    Grp_1   9.0
## 5     a    Grp_2   2.9
## 6     b    Grp_2   9.8
## 7     c    Grp_2  11.0
## 8     d    Grp_2   3.3
## 9     a    Grp_3   2.1
## 10    b    Grp_3   7.0
## 11    c    Grp_3   9.2
## 12    d    Grp_3  10.3
## 13    a    Grp_4   2.0
## 14    b    Grp_4   3.7
## 15    c    Grp_4  12.4
## 16    d    Grp_4  11.1
## 17    a    Grp_5   2.2
## 18    b    Grp_5  19.2
## 19    c    Grp_5   9.6
## 20    d    Grp_5  10.0

```

首先看下每个基因在不同组的表达情况

```
# 给定数据，和 x 轴、y 轴所在列名字
# 直接使用 geom_bar 就可以绘制柱状图
# position: dodge: 柱子并排放置
p <- ggplot(data_m, aes(x=gene, y=value))
p + geom_bar(stat="identity", position="dodge", aes(fill=variable))

# 如果没有图形界面，运行下面的语句把图存在工作目录下的 Rplots.pdf 文件中
#dev.off()
```



柱子有点多，也可以利用 $\text{mean} \pm \text{SD}$ 的形式展现

首先计算平均值和标准差，使用 group_by 按 gene 分组，对每组做 summarize

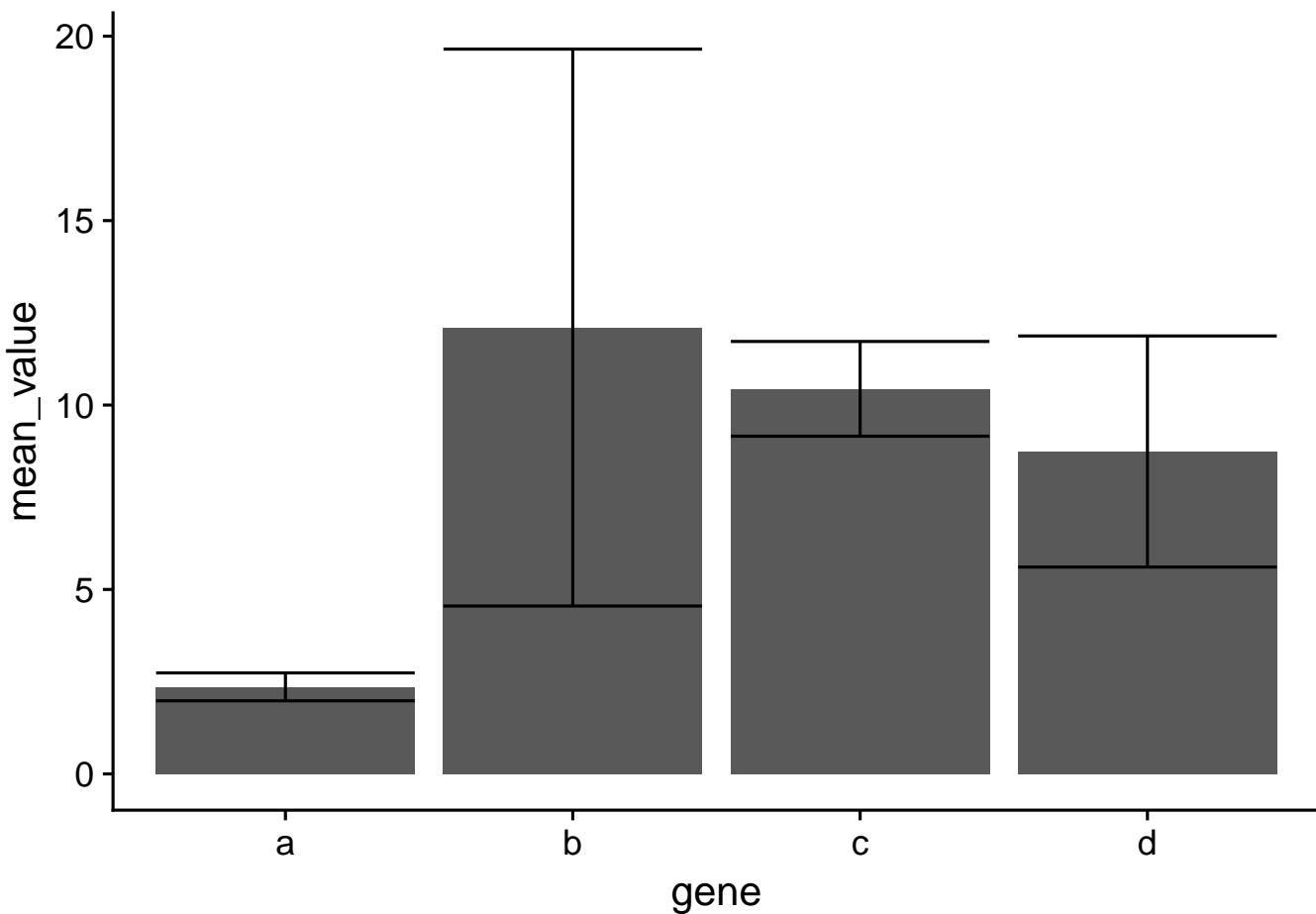
```
# 获取平均值和标准差
data_m_sd_mean <- data_m %>% group_by(gene) %>%
  dplyr::summarise(sd=sd(value), mean_value=mean(value))
data_m_sd_mean <- as.data.frame(data_m_sd_mean)
data_m_sd_mean
```

CONTENTS

```
##   gene      sd mean_value
## 1     a 0.3781534    2.36
## 2     b 7.5491721   12.10
## 3     c 1.2837445   10.44
## 4     d 3.1325708    8.74
```

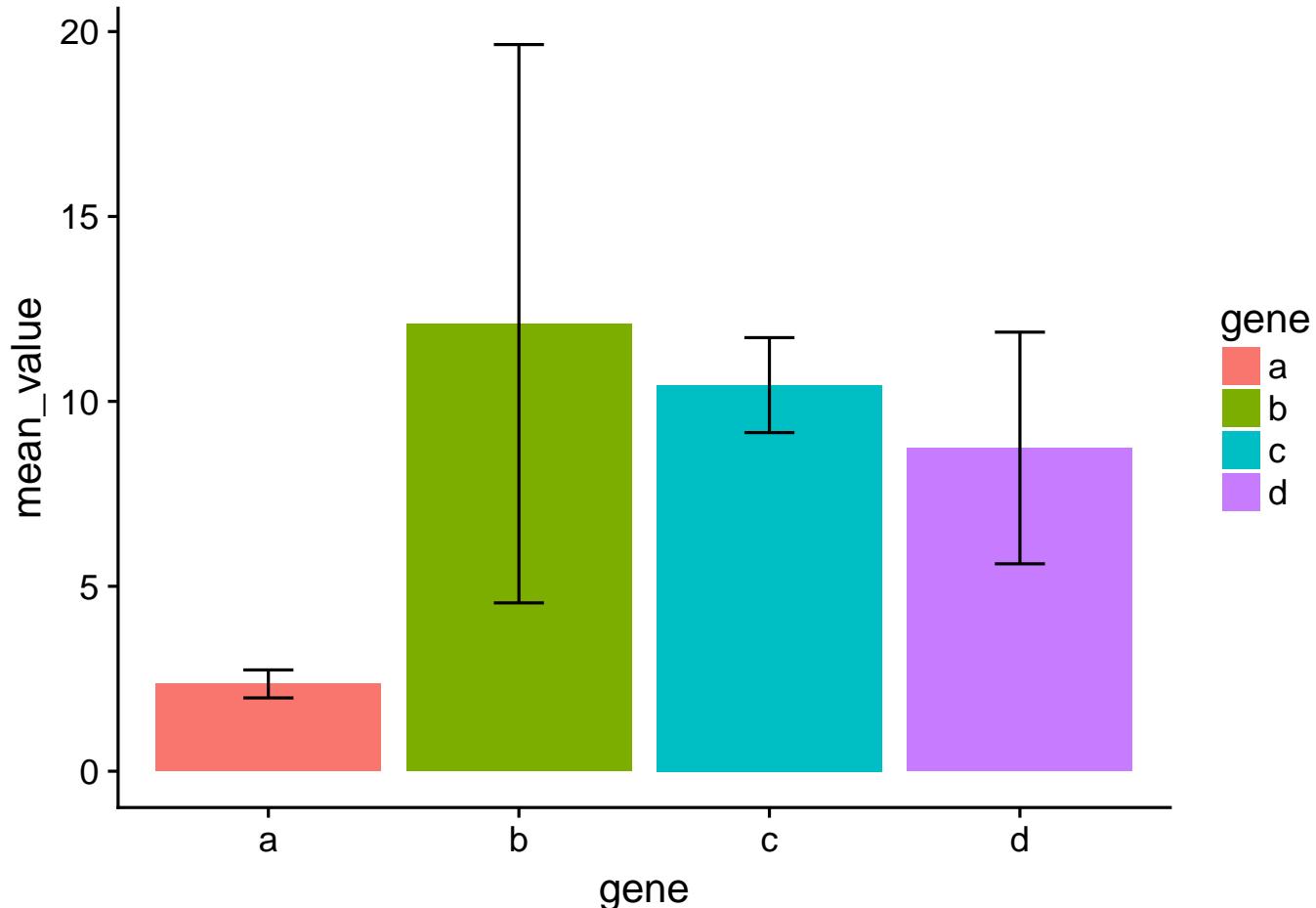
使用 `geom_errorbar` 添加误差线

```
p <- ggplot(data_m_sd_mean, aes(x=gene, y=mean_value)) +
  geom_bar(stat="identity") +
  geom_errorbar(aes(ymin=mean_value-sd, ymax=mean_value+sd))
p
```



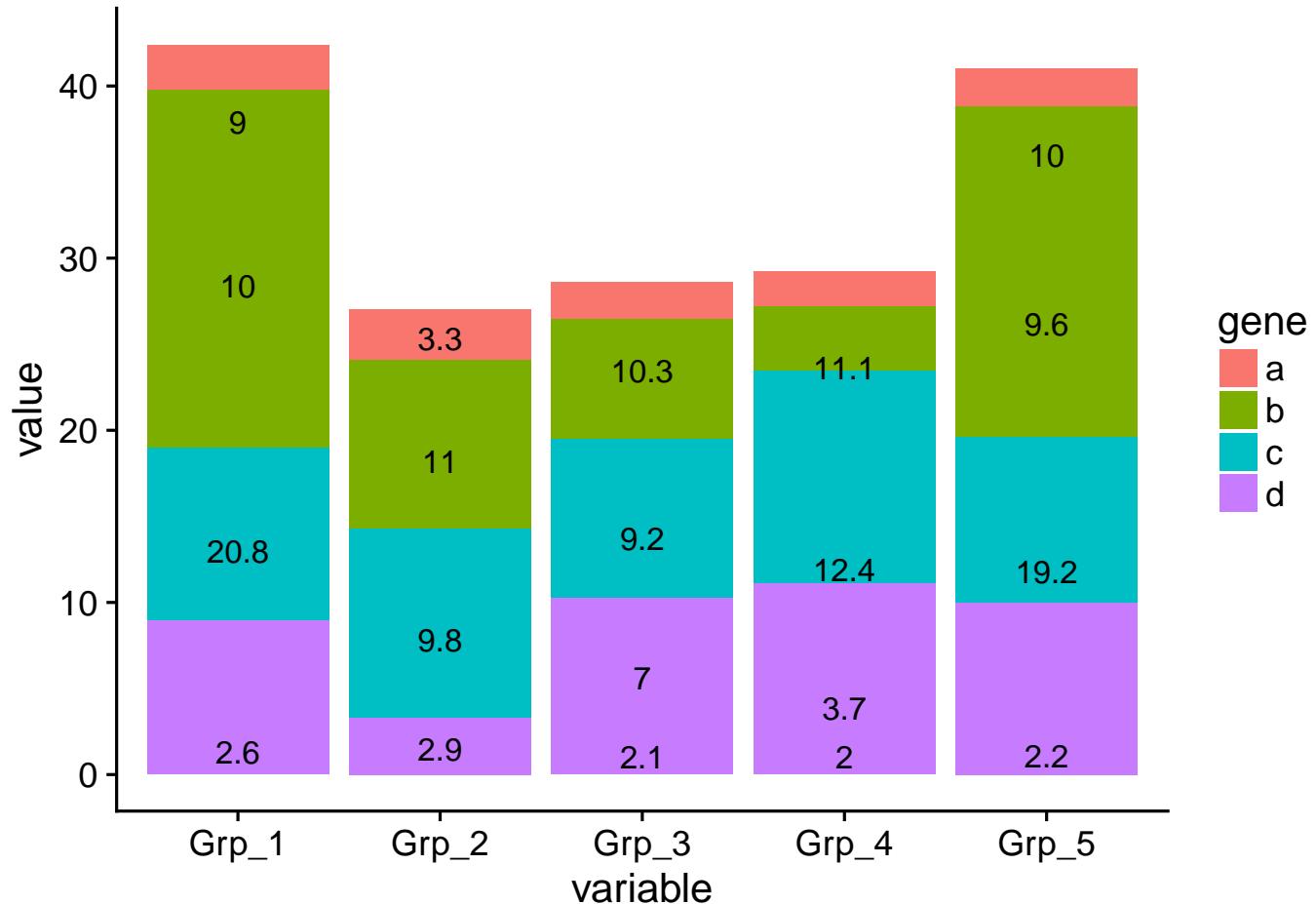
设置误差线的宽度和位置

```
p <- ggplot(data_m_sd_mean, aes(x=gene, y=mean_value)) +
  geom_bar(stat="identity", aes(fill=gene)) +
  geom_errorbar(aes(ymin=mean_value-sd, ymax=mean_value+sd), width=0.2,
                position=position_dodge(width=0.75))
p
```



每个基因的原始表达值堆积柱状图 (只需要修改 position=stack)

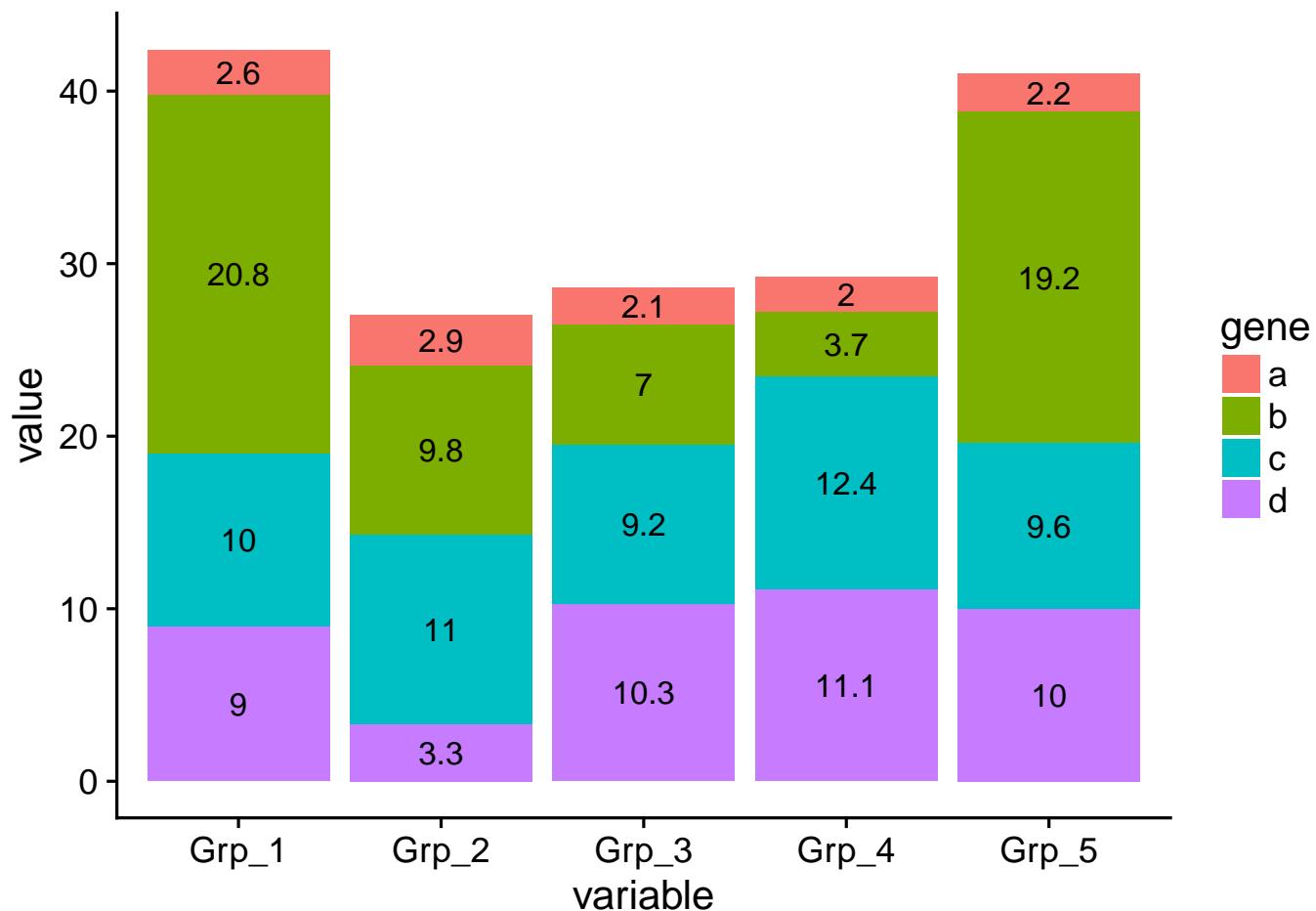
```
# position="fill" 展示的是堆积柱状图各部分的相对比例
# position="stack" 展示的是堆积柱状图的原始值
p <- ggplot(data_m, aes(x=variable, y=value)) +
  geom_bar(stat="identity", position="stack", aes(fill=gene)) +
  geom_text(aes(label=value), position=position_stack(vjust=0.5))
p
```



堆积柱状图显示没问题，但文本标记错位了

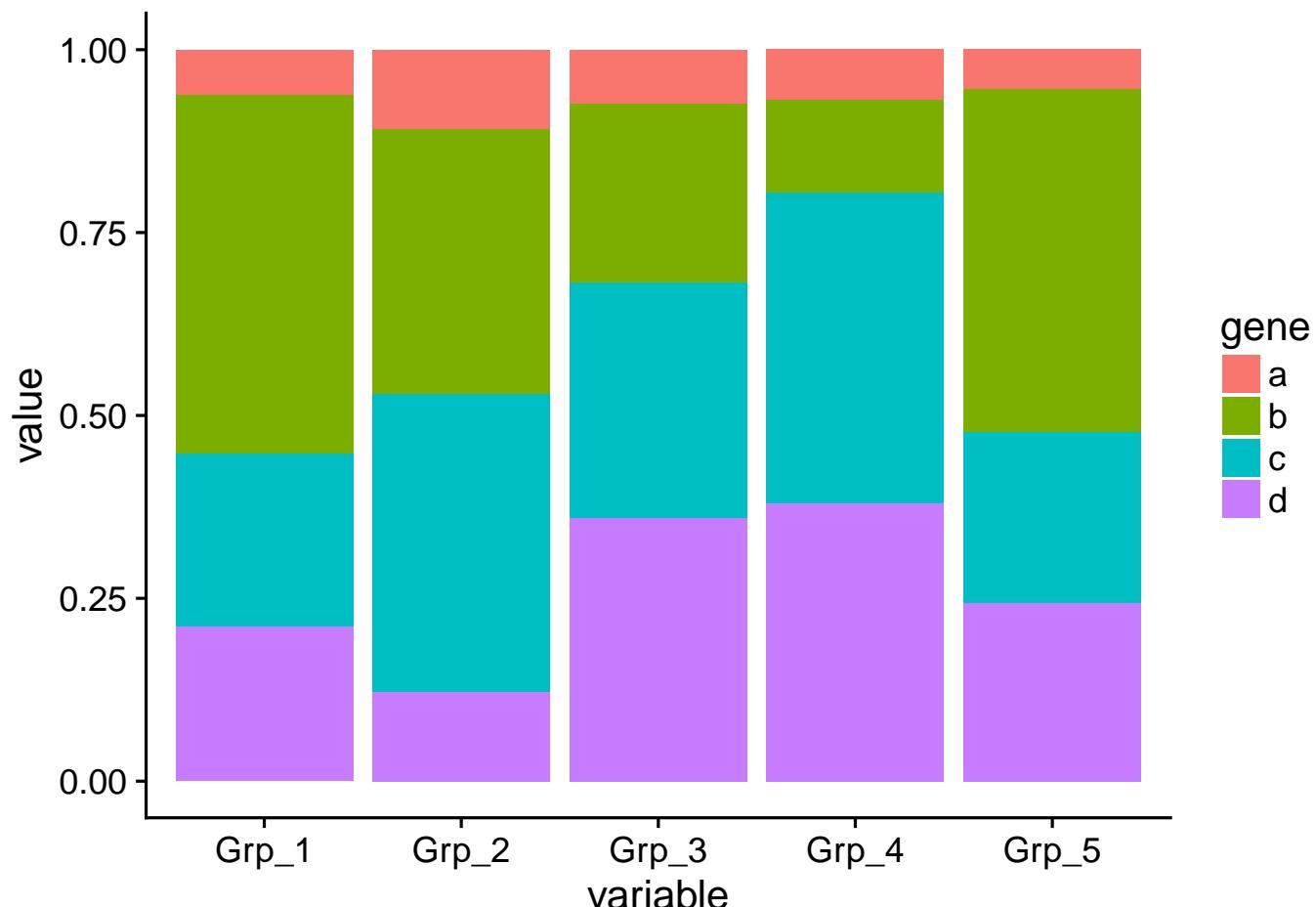
指定下分组信息，位置计算就正确了

```
# position="fill" 展示的是堆积柱状图各部分的相对比例
# position="stack" 展示的是堆积柱状图的原始值
p <- ggplot(data_m, aes(x=variable, y=value, group=gene)) +
  geom_bar(stat="identity", position="stack", aes(fill=gene)) +
  geom_text(aes(label=value), position=position_stack(vjust=0.5))
p
```



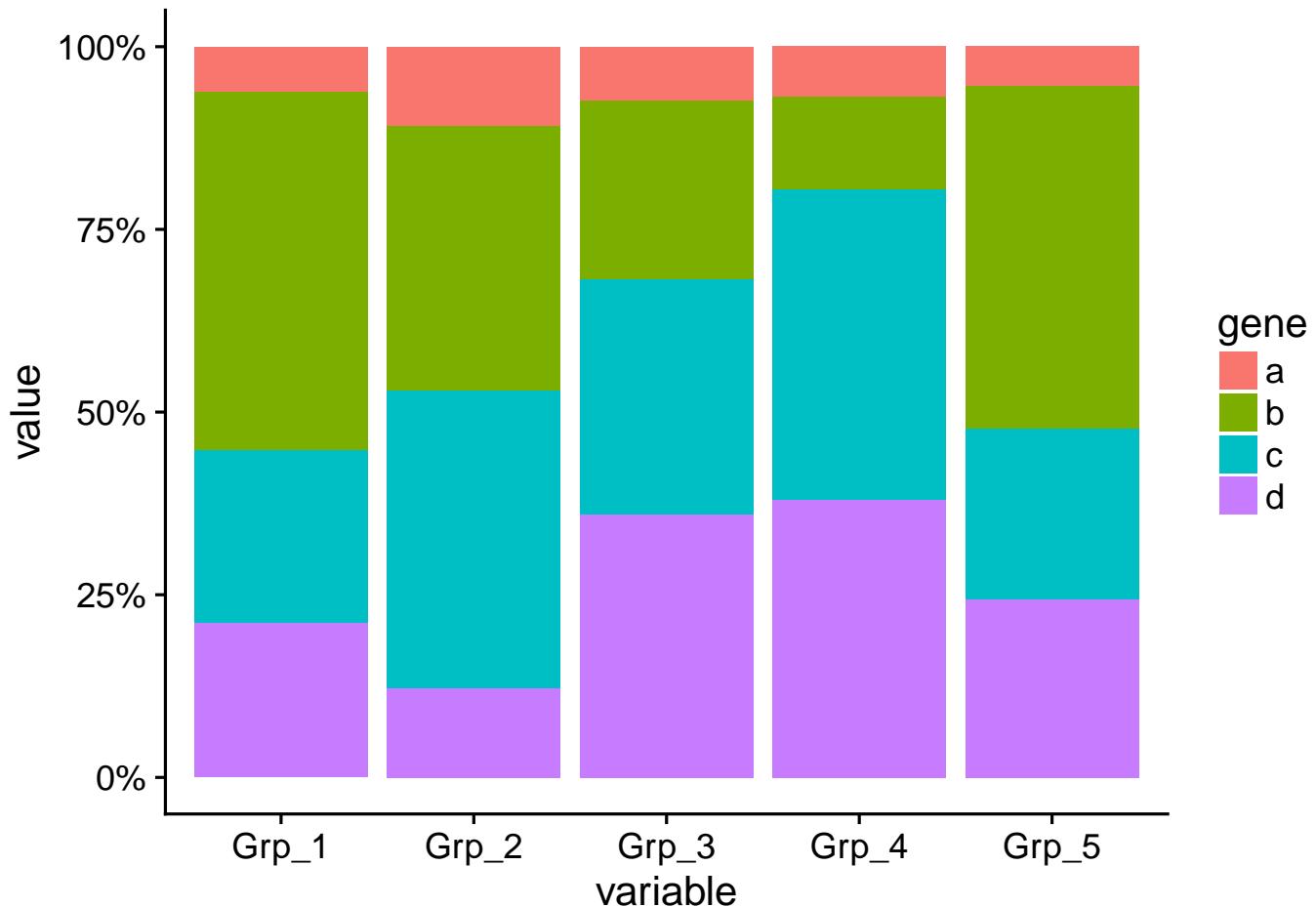
比较每组各个基因的相对表达 (position=fill)

```
# position="fill" 展示的是堆积柱状图各部分的相对比例
# position="stack" 展示的是堆积柱状图的原始值，可以自己体现下看卡差别
p <- ggplot(data_m, aes(x=variable, y=value)) +
  geom_bar(stat="identity", position="fill", aes(fill=gene))
p
```



纵轴的显示改为百分比

```
p <- ggplot(data_m, aes(x=variable, y=value)) +  
  geom_bar(stat="identity", position="fill", aes(fill=gene)) +  
  scale_y_continuous(labels = scales::percent)  
p
```



在柱子中标记百分比值

首先计算百分比，同样是 `group_by` (按照给定的变量分组，然后按组操作) 和 `mutate` 两个函数 (在当前数据表增加新变量)

```
# group_by: 按照给定的变量分组，然后按组操作
# mutate: 在当前数据表增加新变量
# 第一步增加每个组的加和，第二步计算比例
data_m <- data_m %>% group_by(variable) %>% mutate(count=sum(value)) %>%
  mutate(freq=round(100*value/count,2))
head(data_m)
```

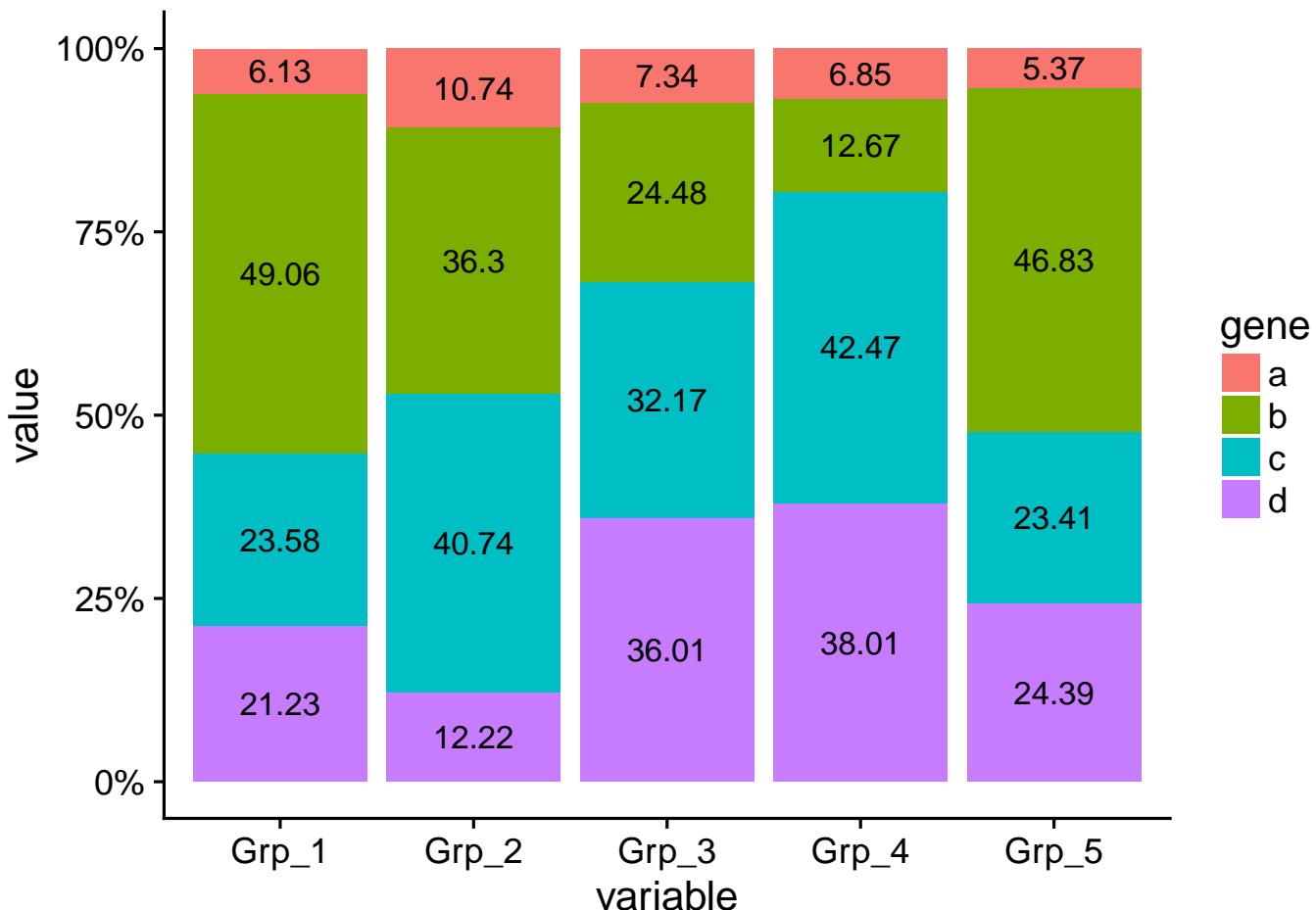
```
## # A tibble: 6 x 5
## # Groups:   variable [2]
##   gene  variable value count freq
##   <chr> <fct>    <dbl> <dbl> <dbl>
## 1 a     Grp_1     2.60  42.4  6.13
## 2 b     Grp_1     20.8   42.4  49.1
## 3 c     Grp_1     10.0   42.4  23.6
```

CONTENTS

```
#> 4 d      Grp_1      9.00  42.4 21.2
#> 5 a      Grp_2      2.90  27.0 10.7
#> 6 b      Grp_2      9.80  27.0 36.3
```

再标记相对比例信息

```
p <- ggplot(data_m, aes(x=variable, y=value, group=gene)) +
  geom_bar(stat="identity", position="fill", aes(fill= gene)) +
  scale_y_continuous(labels = scales::percent) +
  geom_text(aes(label=freq), position=position_fill(vjust=0.5))
p
```



3.10.2 长矩阵分面绘制

再复杂一些的矩阵 (除了有不同时间点的信息，再增加对照和处理的信息)

```
library(ggplot2)
library(reshape2)
library(dplyr)

data_ori <- "Gene;Group;Expr;Condition
a;T1;2.6;Control
b;T1;20.8;Control
c;T1;10;Control
d;T1;9;Control
a;T2;2.9;Control
b;T2;9.8;Control
c;T2;11;Control
d;T2;3.3;Control
a;T3;2.1;Control
b;T3;7;Control
c;T3;9.2;Control
d;T3;10.3;Control
a;T4;2;Control
b;T4;3.7;Control
c;T4;12.4;Control
d;T4;11.1;Control
a;T5;2.2;Control
b;T5;19.2;Control
c;T5;9.6;Control
d;T5;10;Control
d;T1;2.6;Treatment
b;T1;20.8;Treatment
c;T1;10;Treatment
a;T1;9;Treatment
d;T2;2.9;Treatment
b;T2;9.8;Treatment
c;T2;11;Treatment
a;T2;3.3;Treatment
a;T3;2.1;Treatment
c;T3;7;Treatment
b;T3;9.2;Treatment
d;T3;10.3;Treatment
a;T4;2;Treatment
c;T4;3.7;Treatment
b;T4;12.4;Treatment
d;T4;11.1;Treatment
a;T5;2.2;Treatment
d;T5;19.2;Treatment
c;T5;9.6;Treatment
b;T5;10;Treatment"

data_m <- read.table(text=data_ori, header=T, sep=";", quote="")
```

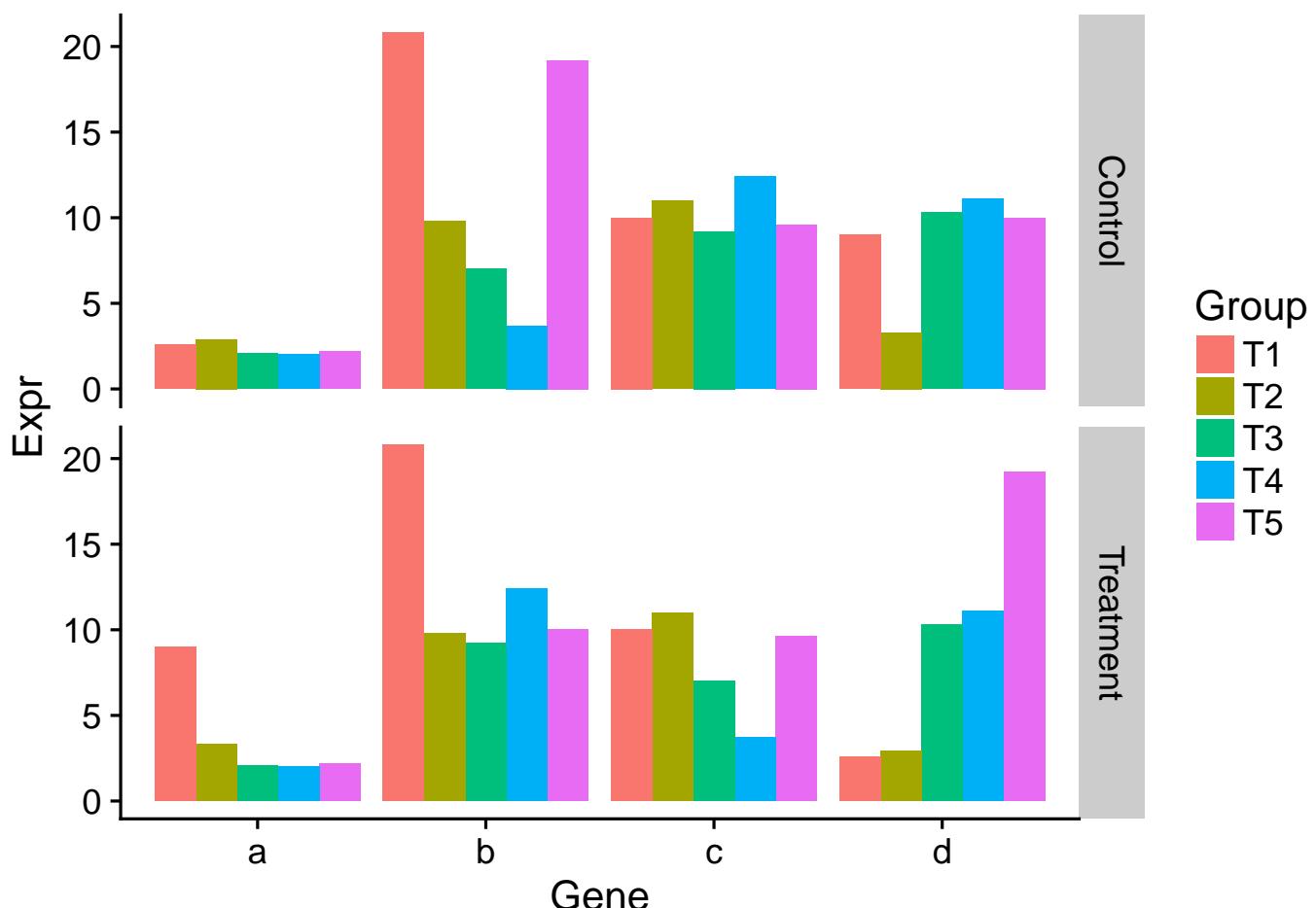
CONTENTS

```
head(data_m)
```

```
##   Gene Group Expr Condition
## 1     a    T1  2.6  Control
## 2     b    T1 20.8  Control
## 3     c    T1 10.0  Control
## 4     d    T1  9.0  Control
## 5     a    T2  2.9  Control
## 6     b    T2  9.8  Control
```

首先看下每个基因在不同组的表达情况, `facet_grid` 和 `facet_wrap` 可以对图形分面显示。

```
# scales: free_y 表示不同子图之间使用独立的 Y 轴信息
#         但 x 轴使用同样的信息。
#         其它可选参数有 free_x, free, fixed
p <- ggplot(data_m, aes(x=Gene, y=Expr)) +
  geom_bar(stat="identity", position="dodge", aes(fill=Group)) +
  facet_grid(Condition~., scales="free_y")
p
# 如果没有图形界面，运行下面的语句把图存在工作目录下的 Rplots.pdf 文件中
#dev.off()
```



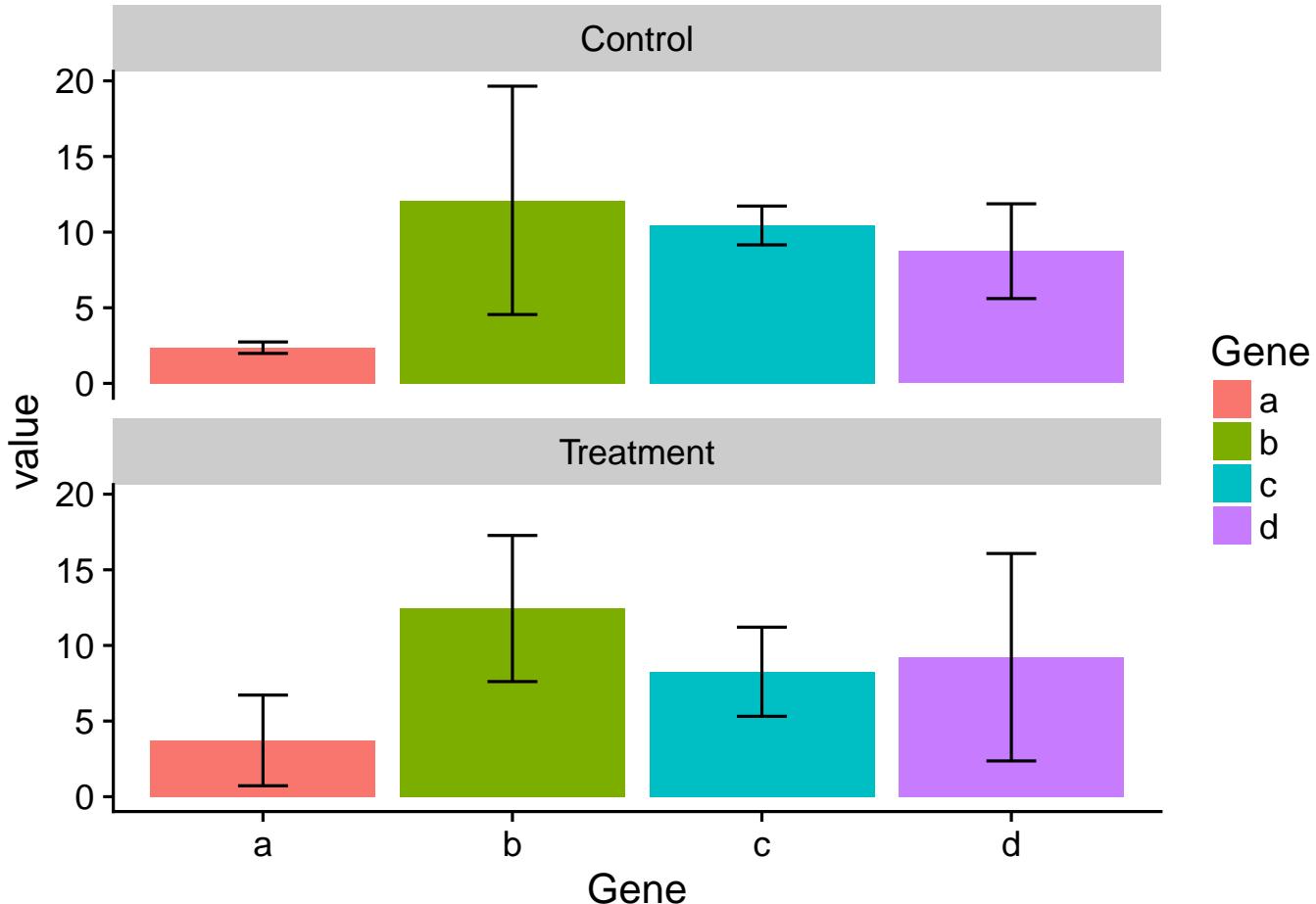
柱子有点多，也可以利用 $\text{mean} \pm \text{SD}$ 的形式展现

```
# 获取平均值和标准差
# 分组时不只 Gene 一个变量了，还需要考虑 Condition
data_m_sd_mean <- data_m %>% group_by(Gene, Condition) %>%
  dplyr::summarise(sd=sd(Expr), value=mean(Expr))
data_m_sd_mean <- as.data.frame(data_m_sd_mean)
data_m_sd_mean
```

```
##   Gene Condition      sd value
## 1   a   Control 0.3781534  2.36
## 2   a Treatment 2.9978326  3.72
## 3   b   Control 7.5491721 12.10
## 4   b Treatment 4.8299068 12.44
## 5   c   Control 1.2837445 10.44
## 6   c Treatment 2.9458445  8.26
## 7   d   Control 3.1325708  8.74
## 8   d Treatment 6.8568943  9.22
```

```
p <- ggplot(data_m_sd_mean, aes(x=Gene, y=value)) +
  geom_bar(stat="identity", aes(fill=Gene)) +
  geom_errorbar(aes(ymin=value-sd, ymax=value+sd), width=0.2,
    position=position_dodge(width=0.75)) +
  facet_wrap(~Condition, ncol=1)

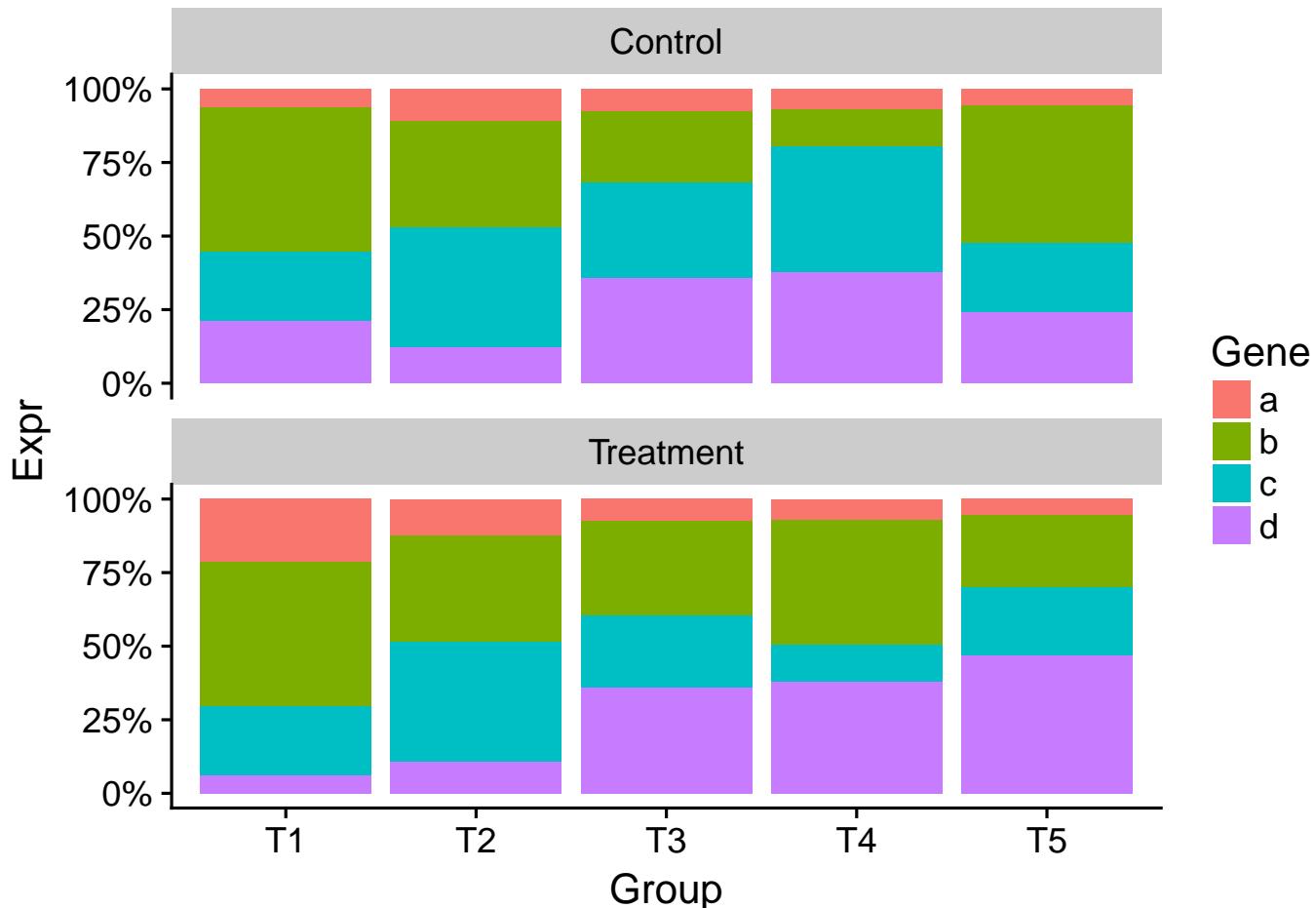
p
```



每组里面各个基因的相对表达, 纵轴的显示改为百分比

```
# position="fill" 展示的是堆积柱状图各部分的相对比例
# position="stack" 展示的是堆积柱状图的原始值, 可以自己体现下看卡差别
p <- ggplot(data_m, aes(x=Group, y=Expr)) +
  geom_bar(stat="identity", position="fill", aes(fill=Gene)) +
  scale_y_continuous(labels = scales::percent) +
  facet_wrap(~Condition, ncol=1)

p
```



facet 后，显示正常，不需要做特别的修改

在柱子中标记百分比值 (计算百分比值需要注意了，文本显示位置还是跟之前一致)

```
# group_by: 按照给定的变量分组，然后按组操作
# mutate: 在当前数据表增加新变量
# 第一步增加每个组 (Group 和 Condition 共同定义分组) 的加和，第二步计算比例
data_m <- data_m %>% group_by(Group, Condition) %>% mutate(count=sum(Expr)) %>%
  mutate(freq=round(100*Expr/count,2))
head(data_m)
```

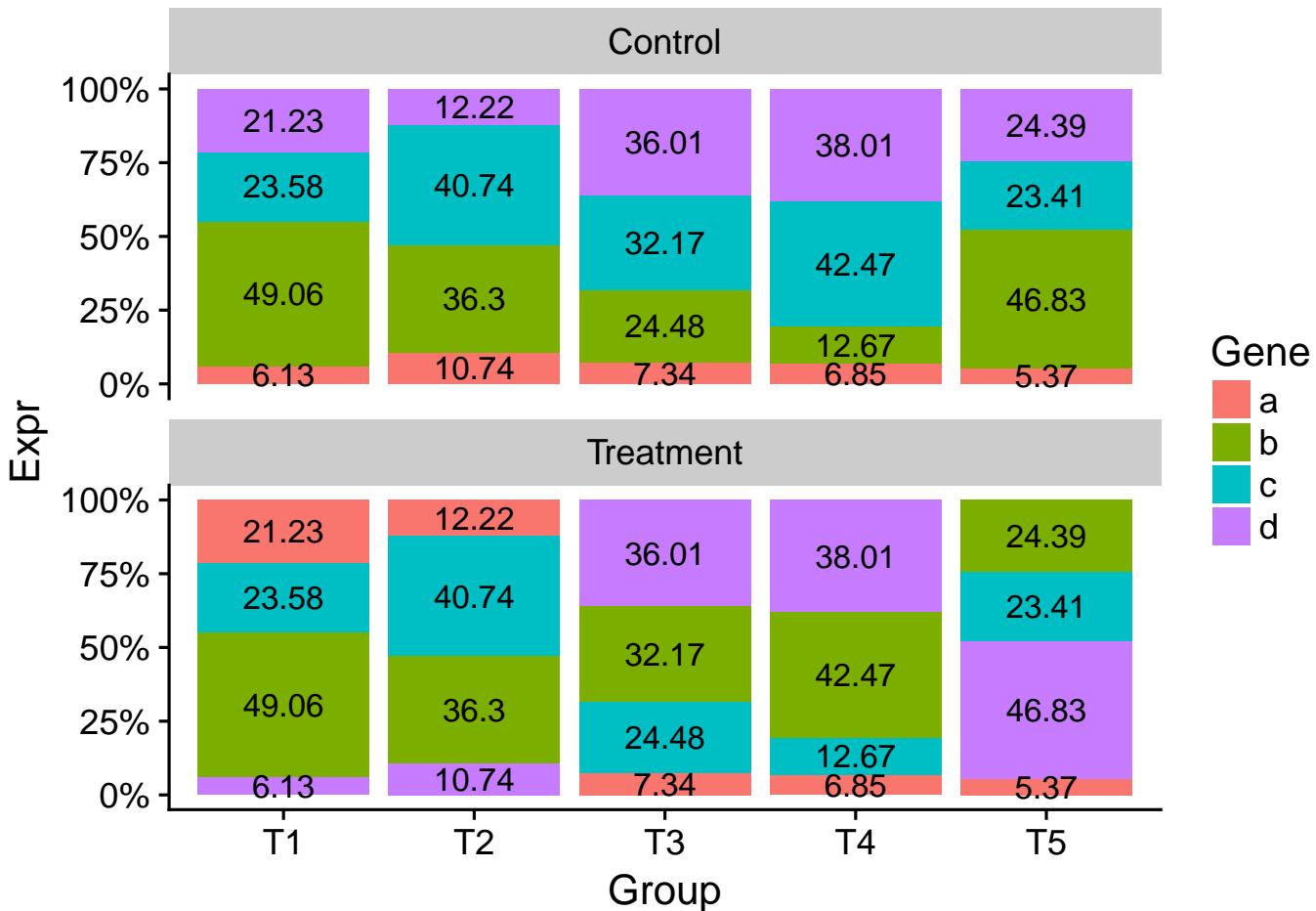
```
## # A tibble: 6 x 6
## # Groups:   Group, Condition [2]
##   Gene  Group  Expr Condition count freq
##   <fct> <fct> <dbl> <fct>     <dbl> <dbl>
## 1 a      T1     2.60 Control    42.4  6.13
## 2 b      T1     20.8  Control    42.4  49.1
## 3 c      T1     10.0  Control    42.4  23.6
## 4 d      T1     9.00 Control    42.4  21.2
```

CONTENTS

```
#> 5 a      T2      2.90 Control    27.0 10.7
#> 6 b      T2      9.80 Control    27.0 36.3
```

```
p <- ggplot(data_m, aes(x=Group, y=Expr, group=Group)) +
  geom_bar(stat="identity", position="fill", aes(fill=Gene)) +
  scale_y_continuous(labels = scales::percent) +
  geom_text(aes(label=freq), position=position_fill(vjust=0.5)) +
  facet_wrap(~Condition, ncol=1)

p
```



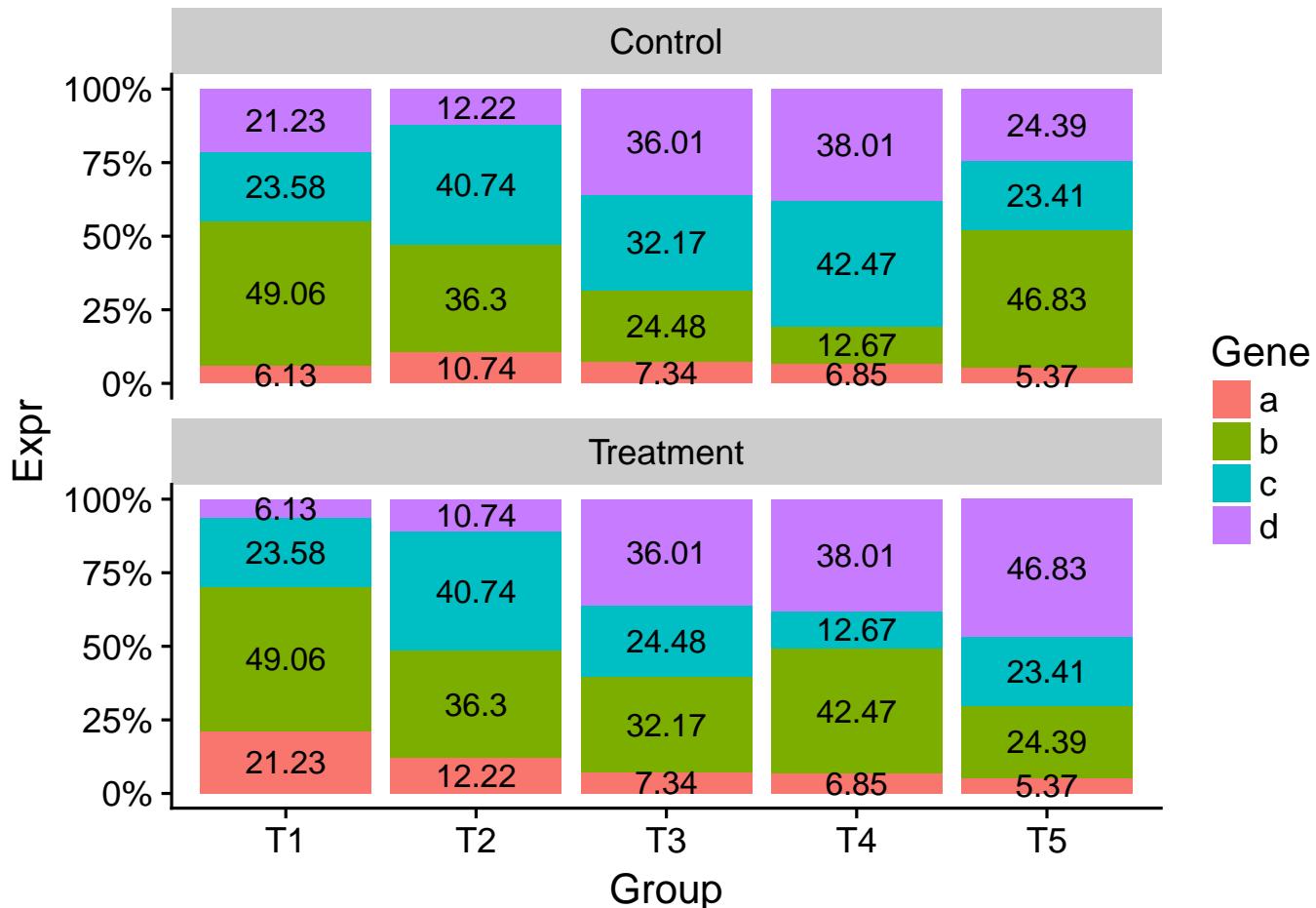
文本显示位置没有问题，但柱子的位置有些奇怪，使得两组之间不可比。

先对数据做下排序，然后再标记文本 (这部分可以看下视频，或者自己输出下数据看看为什么要排序。底层原因可能是这种新的数据结构的问题。)

```
# with: 产生一个由 data_m 组成的局部环境，再这个环境里，列名字可以直接使用
data_m <- data_m[with(data_m, order(Condition, Group, Gene)),]
p <- ggplot(data_m, aes(x=Group, y=Expr, group=Group)) +
  geom_bar(stat="identity", position="fill", aes(fill=Gene)) +
```

```
scale_y_continuous(labels = scales::percent) +
geom_text(aes(label=freq), position=position_fill(vjust=0.5)) +
facet_wrap(~Condition, ncol=1)
```

p



这样两种条件下的比较更容易了。

3.11 图形支持中文字体

3.11.1 修改图形的字体

ggplot2 中修改图形字体。

```
# 修改坐标轴和legend、标题的字体
theme(text=element_text(family="Arial"))
```

```
# 或者
theme_bw(base_family="Arial")

# 修改geom_text的字体
geom_text(family="Arial")
```

3.11.2 ggplot2 支持中文字体输出 PDF

`showtext` 包可给定字体文件，加载到 R 环境中，生成新的字体家族名字，后期调用这个名字设定字体，并且支持中文写入 pdf 不乱码

```
library(showtext)
showtext.auto(enable=TRUE)

font_path = "FZSTK.TTF"
font_name = tools::file_path_sans_ext(basename(font_path))
font.add(font_name, font_path)

# 修改坐标轴和legend、标题的字体
theme(text=element_text(family=font_name))

# 修改geom_text的字体
geom_text(family=font_name)
```

3.11.3 系统可用字体

- Linux 字体一般在 `/usr/share/fonts` 下，也可以使用 `fc-list` 列出所以加载的字体。
- Windows 字体在 `C:\Windows\Fonts\` 下，直接可以看到，也可以拷贝到 Linux 下使用。

3.11.4 合并字体支持中英文

通常情况下，作图的字体都是英文，`ggplot2` 默认的或按需求加载一种字体就可以了。但如果中英文混合出现时，单个字体只能支持一种文字，最好的方式是合并两种字体，类似于 Word 中设置中英文分别使用不同的字体。

软件[FontForge](#)可以方便的合并中英文字体，其安装也比较简单，直接 `yum install fontforge.x86_64`。

假如需要合并 `FZSTK.TTF` (windows 下获取) 和 `Schoolbell-Regular.ttf` (谷歌下载)，这两个都是手写字体。按如下，把字体文件和程序脚本 `mergefont.pe` 放在同一目录下，运行 `fontforge -script mergefont.pe` 即可获得合并后的字体 `FZ_School.ttf`。

CONTENTS

```

ct@ehbio $ ls
FZSTK.TTF mergefont.pe Schoolbell-Regular.ttf
ct@ehbio $ cat mergefont.pe
Open("FZSTK.TTF")
SelectAll()
ScaleToEm(1024)
Generate("temp.ttf", "", 0x14)
Close()

# Open English font and merge to the Chinese font
Open("Schoolbell-Regular.ttf")
SelectAll()
ScaleToEm(1024)

MergeFonts("temp.ttf")
SetFontNames("FZ_School", "FZST", "Schoolbel", "Regular", "")
Generate("FZ_School.ttf", "", 0x14)
Close()

ct@ehbio $ fontforge -script mergefont.pe
ct@ehbio $ ls
FZ_School.ttf FZSTK.TTF mergefont.pe Schoolbell-Regular.ttf

```

然后安装前面的介绍使用 `showtext` 导入即可使用。

3.11.5 一个示例

字体文件自己从 Windows 获取，School bell 从 Google fonts 获取。

```

library(showtext)
## Add fonts that are available on current path

# 方正字体+schoole bell (中英混合)
font.add("FZ_School", "font/FZ_School.ttf")
# 黑体
font.add("simhei", "font/simhei.ttf")
font.add("Arial", "font/arial.ttf")

# 黑体和Arial的合体
font.add("HeiArial", "font/HeiArial.ttf")
showtext.auto() ## automatically use showtext for new devices

library(ggplot2)

```

```

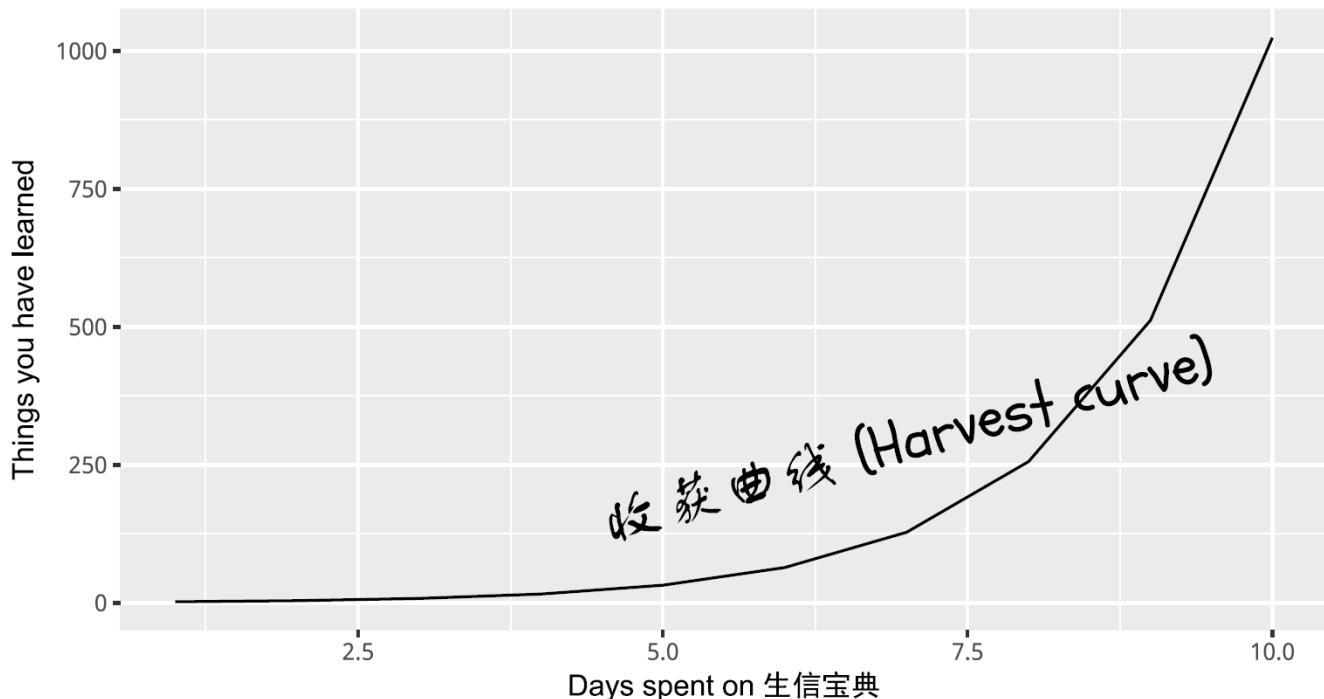
p = ggplot(NULL, aes(x = 1:10, y = 2^(1:10), group=1)) + geom_line() +
  theme(axis.title.y=element_text(family="Arial"),
        axis.title.x=element_text(family="HeiArial"),
        plot.title=element_text(family="simhei")) +
  xlab("Days spent on 生信宝典") +
  ylab("Things you have learned") +
  ggtitle("生信宝典，换个角度学生信") +
  annotate("text", 7, 300, family = "FZ_School", size = 8,
           label = "收获曲线 (Harvest curve)", angle=15)

# annotate指定的是文字的中间部分的位置

#ggsave(p, filename="example-SXBD.pdf", width = 7, height = 4) ## PDF device
p

```

生信宝典，换个角度学生信



3.12 PCA

```

library(knitr)
library(psych)

## 

```

```

## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
## 
##     %+%, alpha

library(reshape2)
library(ggplot2)
library(ggbeeswarm)
library(scatterplot3d)
library(useful)
library(ggfortify)

## Loading required package: methods

```

3.12.1 主成分分析简介

主成分分析 (PCA, principal component analysis) 是一种数学降维方法, 利用正交变换 (orthogonal transformation) 把一系列可能线性相关的变量转换为一组线性不相关的新变量, 也称为主成分, 从而利用新变量在更小的维度下展示数据的特征。

主成分是原有变量的线性组合, 其数目不多于原始变量。组合之后, 相当于我们获得了一批新的观测数据, 这些数据的含义不同于原有数据, 但包含了之前数据的大部分特征, 并且有着较低的维度, 便于进一步的分析。

在空间上, PCA 可以理解为把原始数据投射到一个新的坐标系统, 第一主成分为第一坐标轴, 它的含义代表了原始数据中多个变量经过某种变换得到的新变量的变化区间; 第二成分为第二坐标轴, 代表了原始数据中多个变量经过某种变换得到的第二个新变量的变化区间。这样我们把利用原始数据解释样品的差异转变为利用新变量解释样品的差异。

这种投射方式会有很多, 为了最大限度保留对原始数据的解释, 一般会用最大方差理论或最小损失理论, 使得第一主成分有着最大的方差或变异数 (就是说其能尽量多的解释原始数据的差异); 随后的每一个主成分都与前面的主成分正交, 且有着仅次于前一主成分的最大方差 (正交简单的理解就是两个主成分空间夹角为 90° , 两者之间无线性关联, 从而完成去冗余操作)。

3.12.2 主成分分析的意义

1. 简化运算。

在问题研究中, 为了全面系统地分析问题, 我们通常会收集众多的影响因素也就是众多的变量。这样会使得研究更丰富, 通常也会带来较多的冗余数据和复杂的计算量。

比如我们测序了 100 种样品的基因表达谱借以通过分子表达水平的差异对这 100 种样品进行分类。在这个问题中，研究的变量就是不同的基因。每个基因的表达都可以在一定程度上反应样品之间的差异，但某些基因之间却有着调控、协同或拮抗的关系，表现为它们的表达值存在一些相关性，这就造成了统计数据所反映的信息存在一定程度的冗余。另外假如某些基因如持家基因在所有样本中表达都一样，它们对于解释样本的差异也没有意义。这么多的变量在后续统计分析中会增大运算量和计算复杂度，应用 PCA 就可以在尽量多的保持变量所包含的信息又能维持尽量少的变量数目，帮助简化运算和结果解释。

2. 去除数据噪音。

比如说我们在样品的制备过程中，由于不完全一致的操作，导致样品的状态有细微的改变，从而造成一些持家基因也发生了相应的变化，但变化幅度远小于核心基因（一般认为噪音的方差小于信息的方差）。而 PCA 在降维的过程中滤去了这些变化幅度较小的噪音变化，增大了数据的信噪比。

3. 利用散点图实现多维数据可视化。

在上面的表达谱分析中，假如我们有 1 个基因，可以在线性层面对样本进行分类；如果我们有 2 个基因，可以在一个平面对样本进行分类；如果我们有 3 个基因，可以在一个立体空间对样本进行分类；如果有更多的基因，比如说 n 个，那么每个样品就是 n 维空间的一个点，则很难在图形上展示样品的分类关系。利用 PCA 分析，我们可以选取贡献最大的 2 个或 3 个主成分作为数据代表用以可视化。这比直接选取三个表达变化最大的基因更能反映样品之间的差异。（利用 Pearson 相关系数对样品进行聚类在样品数目比较少时是一个解决办法）

4. 发现隐性相关变量。

我们在合并冗余原始变量得到主成分过程中，会发现某些原始变量对同一主成分有着相似的贡献，也就是说这些变量之间存在着某种相关性，为相关变量。同时也可获得这些变量对主成分的贡献程度。对基因表达数据可以理解为发现了存在协同或拮抗关系的基因。

3.12.3 示例展示原始变量对样品的分类

假设有一套数据集，包含 100 个样品中某一基因的表达量。如下所示，每一行为一个样品，每一列为基因的表达值。这也是做 PCA 分析的基本数据组织方式，每一行代表一个样品，每一列代表一组观察数据即一个变量。

```
count <- 50
Gene1_a <- rnorm(count, 5, 0.5)
Gene1_b <- rnorm(count, 20, 0.5)
grp_a <- rep('a', count)
grp_b <- rep('b', count)
cy_data <- data.frame(Gene1 = c(Gene1_a, Gene1_b), Group=c(grp_a, grp_b))
cy_data <- as.data.frame(cy_data)
label <- c(paste0(grp_a, 1:count), paste0(grp_b, 1:count))
row.names(cy_data) <- label
library(knitr)
library(psych)
# Add additional column to data only for plotting
cy_data$Y <- rep(0, count*2)
kable(headTail(cy_data), booktabs=TRUE, caption="Expression profile for Gene1 in 100 samples")
```

从下图可以看出，100 个样品根据 Gene1 表达量的不同在横轴上被分为了 2 类，可以看做是在线性水平的分类。

Table 3.1: Expression profile for Gene1 in 100 samples

	Gene1	Group	Y
a1	4.71	a	0
a2	4.66	a	0
a3	4.74	a	0
a4	4.66	a	0
...	...	NA	...
b47	20.11	b	0
b48	19.44	b	0
b49	19.39	b	0
b50	20.08	b	0

```

library("ggplot2")
library("ggbeeswarm")

# geom_quasirandom: 用于画 Jitter Plot
# theme(axis.*.y): 去除 Y 轴
# xlim, ylim 设定坐标轴的区间
ggplot(cy_data,aes(Gene1, Y))+geom_quasirandom(aes(color=factor(Group)))+
  theme(legend.position=c(0.5,0.7)) + theme(legend.title=element_blank()) +
  scale_fill_discrete(name="Group") + theme(axis.line.y=element_blank(),
  axis.text.y=element_blank(), axis.ticks.y=element_blank(),
  axis.title.y=element_blank()) + ylim(-0.5,5) + xlim(0,25)
  
```

- a
- b

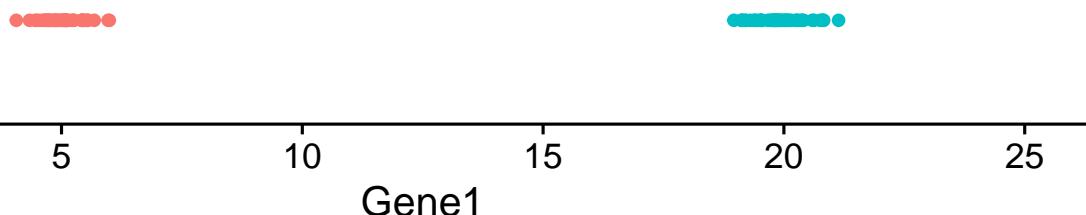


Table 3.2: Expression profile for Gene1 and Gene2 in 100 samples

	Gene1	Gene2	Group
a1	4.71	5.01	a
a2	4.66	4.82	a
a3	4.74	4.84	a
a4	4.66	5.02	a
...	NA
b47	20.11	4.84	b
b48	19.44	5.07	b
b49	19.39	5.3	b
b50	20.08	4.95	b

那么如果有 2 个基因呢？

```
count <- 50
Gene1_a <- rnorm(count, 5, 0.2)
Gene1_b <- rnorm(count, 5, 0.2)

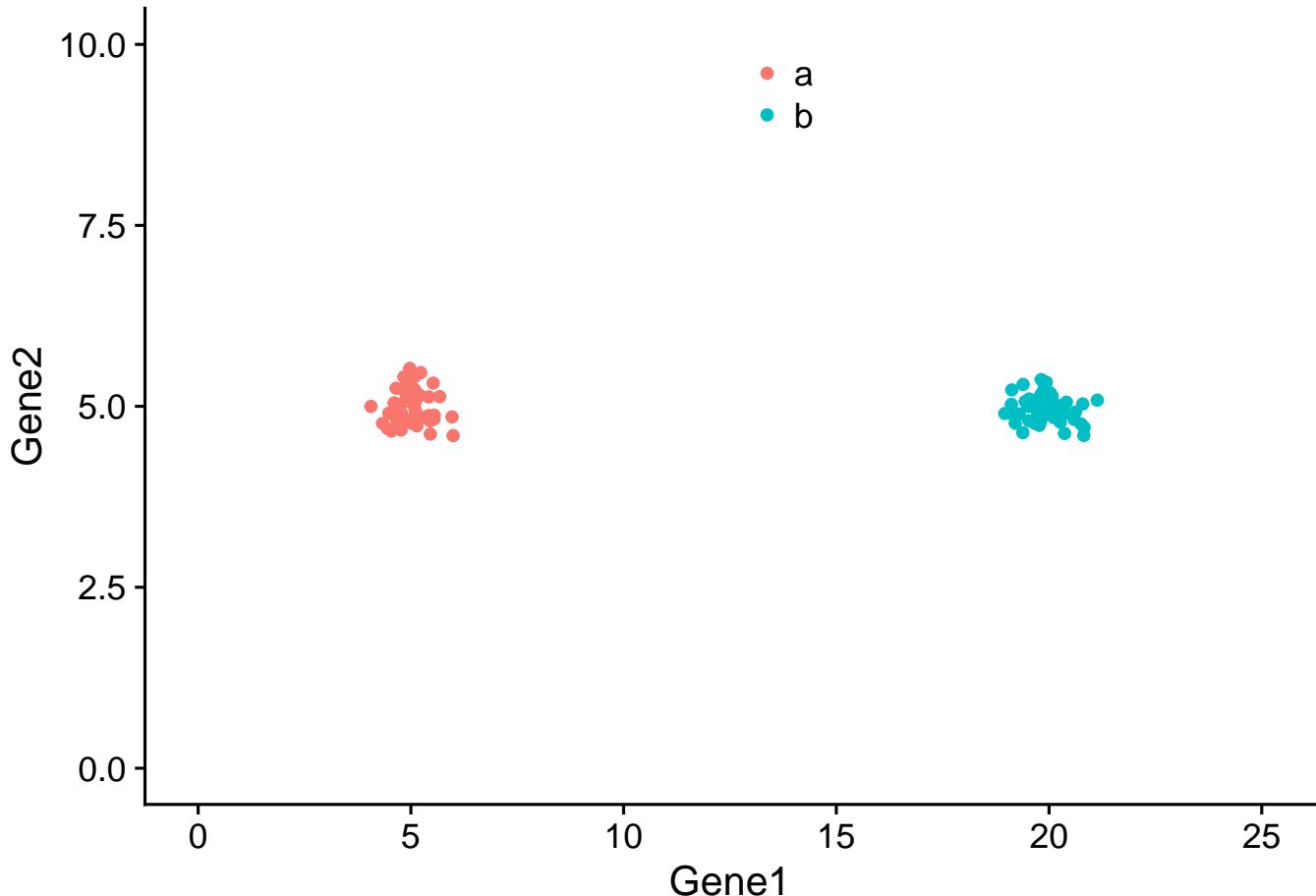
cy_data2 <- data.frame(Gene1 = c(Gene1_a, Gene1_b), Gene2 = c(Gene2_a, Gene2_b),
                        Group=c(grp_a, grp_b))
cy_data2 <- as.data.frame(cy_data2)

row.names(cy_data2) <- label

kable(headTail(cy_data2), booktabs=T,
      caption="Expression profile for Gene1 and Gene2 in 100 samples")
```

从下图可以看出，100 个样品根据 Gene1 和 Gene2 的表达量的不同在坐标轴上被分为了 2 类，可以看做是在平面水平的分类。而且在这个例子中，我们可以很容易的看出 Gene1 对样品分类的贡献要比 Gene2 大，因为 Gene1 在样品间的表达差异大。

```
ggplot(cy_data2,aes(Gene1, Gene2))+geom_point(aes(color=factor(Group)))+
  theme(legend.position=c(0.5,0.9)) + theme(legend.title=element_blank()) +
  ylim(0,10) + xlim(0,25)
```



如果有 3 个基因呢？

```
count <- 50
Gene3_a <- c(rnorm(count/2, 5, 0.2), rnorm(count/2, 15, 0.2))
Gene3_b <- c(rnorm(count/2, 15, 0.2), rnorm(count/2, 5, 0.2))

data3 <- data.frame(Gene1 = c(Gene1_a, Gene1_b), Gene2 = c(Gene2_a, Gene2_b),
                     Gene3 = c(Gene3_a, Gene3_b), Group=c(grp_a, grp_b))
data3 <- as.data.frame(data3)

row.names(data3) <- label

kable(headTail(data3), booktabs=T, caption="Expression profile for 3 genes in 100 samples")
```

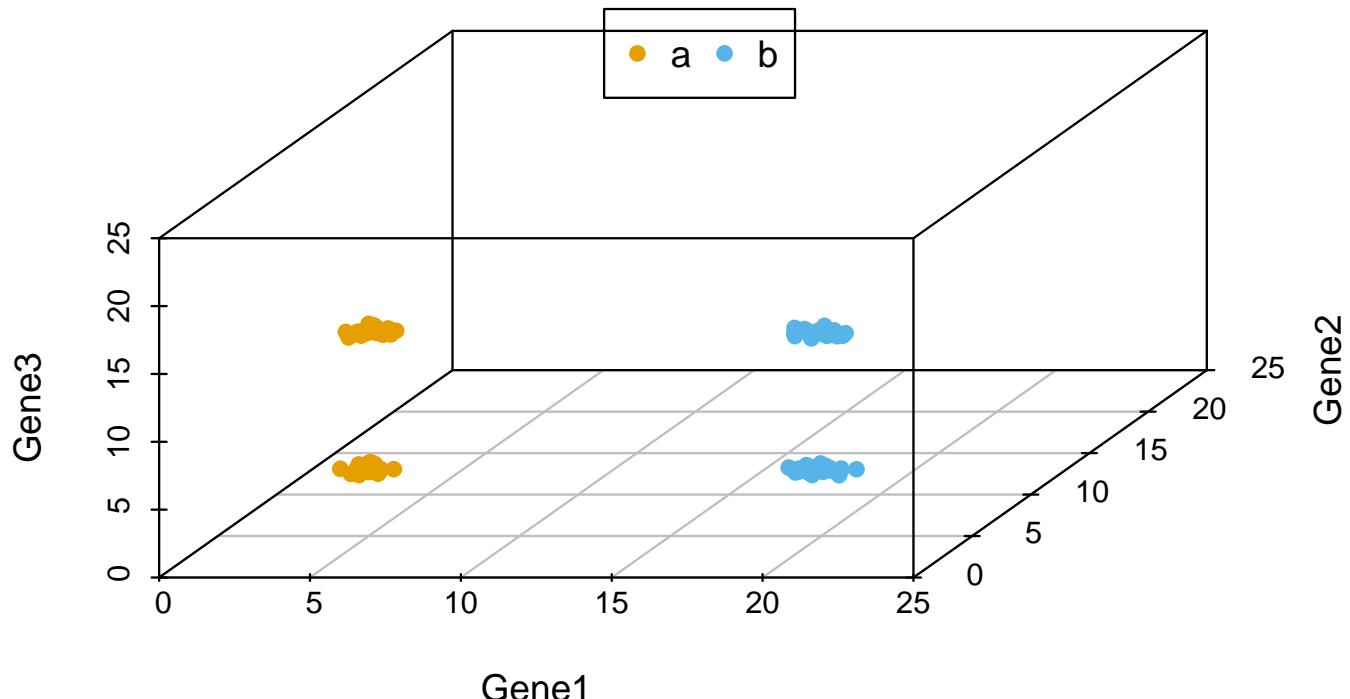
从下图可以看出，100 个样品根据 Gene1、Gene2 和 Gene3 的表达量的不同在坐标轴上被分为了 4 类，可以看做是立体空间的分类。而且在这个例子中，我们可以很容易的看出 Gene1 和 Gene3 对样品分类的贡献要比 Gene2 大。

```
library(scatterplot3d)
color1 <- c("#E69F00", "#56B4E9")
# Extract same number of colors as the Group and same Group would have same color.
```

Table 3.3: Expression profile for 3 genes in 100 samples

	Gene1	Gene2	Gene3	Group
a1	4.71	5.01	4.64	a
a2	4.66	4.82	5.03	a
a3	4.74	4.84	4.93	a
a4	4.66	5.02	5.26	a
...	NA
b47	20.11	4.84	4.83	b
b48	19.44	5.07	4.99	b
b49	19.39	5.3	5.04	b
b50	20.08	4.95	5.23	b

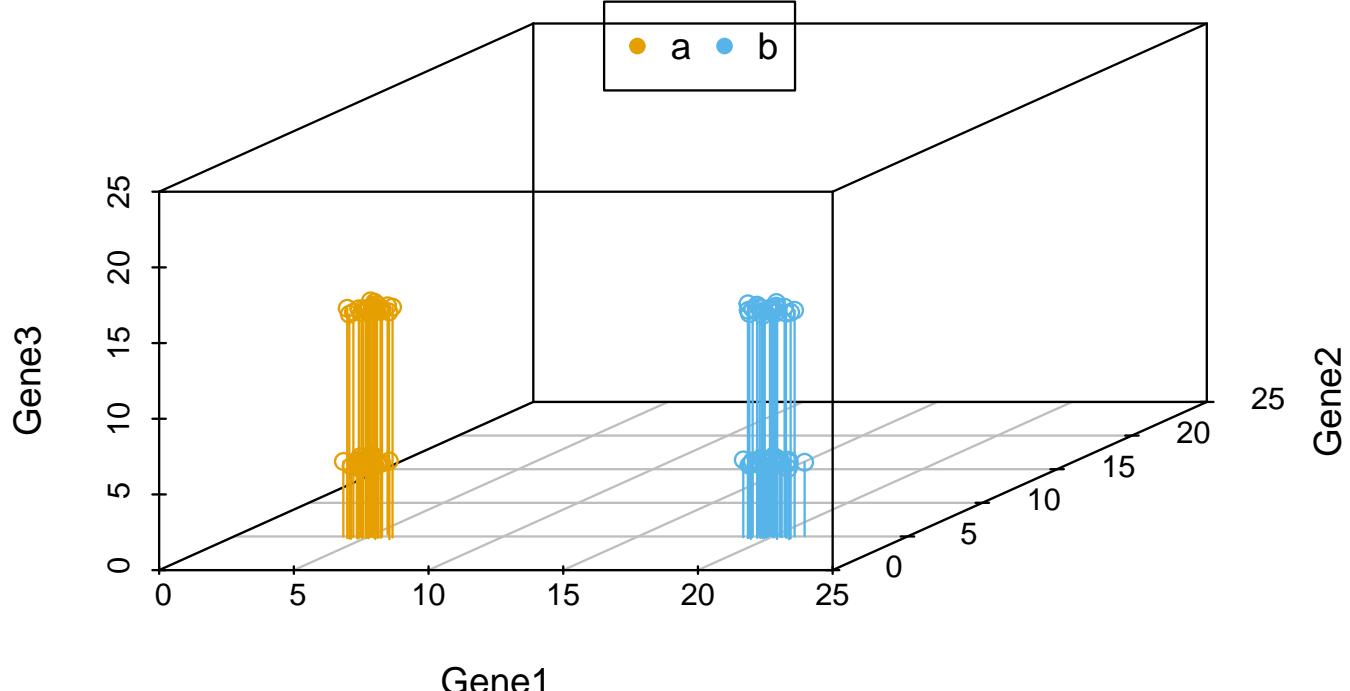
```
colors <- color1[as.numeric(data3$Group)]
scatterplot3d(data3[,1:3], color=colors, xlim=c(0,25), ylim=c(0,25), zlim=c(0,25),
angle=55, pch=16)
legend("top", legend=levels(data3$Group), col=color1, pch=16, xpd=T, horiz=T)
```



当我们向由 Gene1 和 Gene2 构成的 X-Y 平面做垂线时，可以很明显的看出，Gene2 所在的轴对样品的分类没有贡献。因为投射到 X-Y 屏幕上的点在 Y 轴几乎处于同一位置。

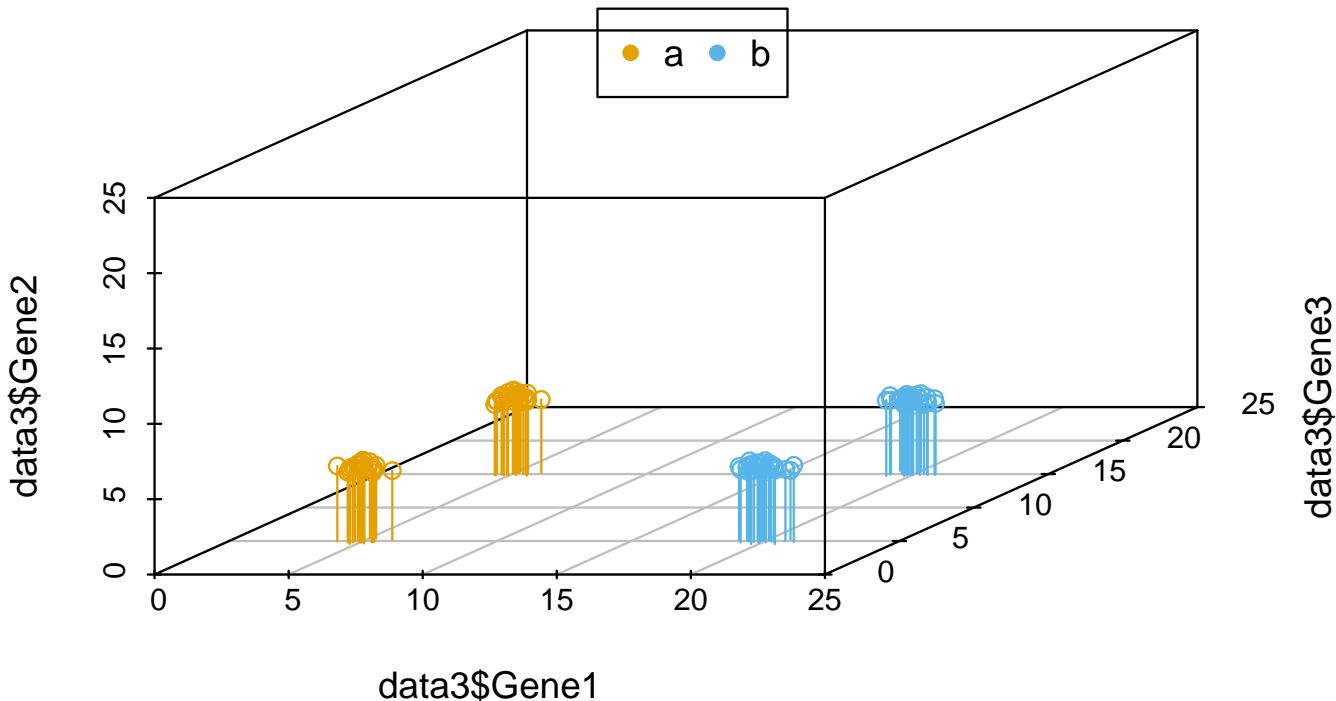
```
library(scatterplot3d)
color1 <- c("#E69F00", "#56B4E9")
colors <- color1[as.numeric(data3$Group)]
```

```
scatterplot3d(data3[,1:3], color=colors, xlim=c(0,25), ylim=c(0,25), zlim=c(0,25), type='h')
legend("top", legend=levels(data3$Group), col=colorl, pch=16, xpd=T, horiz=T)
```



我们把坐标轴做一个转换，可以看到在由 Gene1 和 Gene3 构成的 X-Y 平面上，样品被分为了 4 类。Gene2 对样品的分类几乎没有贡献，因为几乎所有样品在 Gene2 维度上的值都一样。

```
library(scatterplot3d)
colorl <- c("#E69F00", "#56B4E9")
colors <- colorl[as.numeric(data3$Group)]
scatterplot3d(x=data3$Gene1, y= data3$Gene3, z= data3$Gene2, color=colors,
               xlim=c(0,25), ylim=c(0,25), zlim=c(0,25), type='h')
legend("top", legend=levels(data3$Group), col=colorl, pch=16, xpd=T, horiz=T)
```



在上述例子中，我们可以很容易的区分出 Gene1 和 Gene3 可以作为分类的主成分，而 Gene2 则对分类没有帮助，可以在计算中去除。

但是如果我们测序了几万个基因的表达时，就很难通过肉眼去看，或者作出一个图供我们筛选哪些基因对样本分类贡献大。这时我们应该怎么做呢？

其中有一个方法是，在这个基因表达矩阵中选出 3 个变化最大的基因做为 3 个主成分对样品进行分类。我们试验下效果怎么样。

```
# 数据集来源于 http://satijalab.org/seurat/old-get-started/
# 原始下载链接 http://www.broadinstitute.org/~rahuls/seurat/seurat\_files\_nbt.zip
# 为了保证文章的使用，文末附有数据的新下载链接，以防原链接失效
data4 <- read.table("data/HiSeq301-RSEM-linear_values.txt", header=T, row.names=1, sep="\t")
dim(data4)
```

```
## [1] 23730 301
```

```
library(useful)
kable(corner(data4, r=15, c=8), booktabs=T, caption="Gene expression matrix")
```

我们筛选变异系数最大的 3 个基因。在这之前我们先剔除在少于 5 个样品中表达的基因和少于 1000 个表达的基因样品（这里我们把表达值不小于 1 的基因视为表达的基因），并把所有基因根据其在不同样品中表达值的变异系数排序。

Table 3.4: Gene expression matrix

	Hi_2338_1	Hi_2338_2	Hi_2338_3	Hi_2338_4	Hi_2338_5	Hi_2338_6	Hi_2338_7	Hi_2338_8
A1BG	9.08	0.00	0.00	1.75	0.00	0.40	0.00	0.78
A1BG-AS1	0.00	0.00	3.47	0.36	0.00	0.00	0.00	0.00
A1CF	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00
A2LD1	0.00	0.00	0.00	0.29	0.00	9.19	0.00	0.00
A2M	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A2M-AS1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A2ML1	0.10	0.00	0.14	0.00	0.00	0.00	0.00	0.00
A2MP1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A4GALT	0.57	0.00	0.00	0.00	0.00	0.00	0.35	0.00
A4GNT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AA06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AAAS	38.95	0.00	0.00	4.44	0.00	32.90	0.00	5.58
AACS	0.12	0.00	0.00	0.00	0.58	1.03	2.16	65.74
AACSP1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AADAC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

```
# 去除表达值全为 0 的行
#data4_nonzero <- data4[rowSums(data4) != 0,]

# 筛选符合要求的表达的行和列
#data4_use <- data4[apply(data4,1,function(row) sum(row>=1)>=5),]
#data4_use <- data4[,apply(data4,2,function(col) sum(col>=1)>=1000),]
data4_use <- data4[rowSums(data4>=1)>5,colSums(data4>=1)>1000]
```

```
# 对于表达谱数据，因为涉及到 PCR 的指数扩增，一般会取 log 处理
# 其它数据 log 处理会降低数据之间的差异，不一定适用
data4_use_log2 <- log2(data4_use+1)

dim(data4_use_log2)
```

```
## [1] 16482 301
```

```
# 计算变异系数 (标准差除以平均值) 度量基因表达变化幅度
#cv <- apply(data4_use_log2,1,sd)/rowMeans(data4_use_log2)
# 根据变异系数排序
#data4_use_log2 <- data4_use_log2[order(cv,decreasing = T),]

# 计算中值绝对偏差 (MAD, median absolute deviation) 度量基因表达变化幅度
# 在基因表达中，尽管某些基因很小的变化会导致重要的生物学意义，
# 但是很小的观察值会引入很大的背景噪音，因此也意义不大。
mads <- apply(data4_use_log2, 1, mad)
```

Table 3.5: A table of the 3 most variable genes

	MT2A	ANXA1	ARHGDI
Hi_2338_1	12.211493	11.837198	9.543283
Hi_2338_2	11.306216	8.098769	8.071623
Hi_2338_3	11.926226	10.688626	9.720535
Hi_2338_4	10.974207	9.386574	9.883376
Hi_2338_5	14.603994	10.375072	9.970379
Hi_2338_6	6.904604	11.155349	10.093510
Hi_2338_7	12.436719	10.852249	7.742882
Hi_2338_8	9.798375	9.783227	6.270716
Hi_2338_9	11.743673	9.626476	9.250251
Hi_2338_10	11.240016	11.303056	0.000000

```
data4_use_log2 <- data4_use_log2[rev(order(mads)), ]
```

筛选前 3 行

```
data_var3 <- data4_use_log2[1:3, ]
```

转置矩阵使得每一行为一个样品，每一列为一组变量

```
data_var3_forPCA <- t(data_var3)
```

```
dim(data_var3_forPCA)
```

```
## [1] 301 3
```

```
kable(corner(data_var3_forPCA, r=10, c=5), booktabs=TRUE,
      caption="A table of the 3 most variable genes")
```

获得样品分组信息

```
sample <- rownames(data_var3_forPCA)
```

把样品名字按 <_> 分割，取出其第二部分作为样品的组名

lapply(X, FUC) 对列表或向量中每个元素执行 FUC 操作，FUNC 为自定义或 R 自带的函数

One better way to generate group

```
group <- unlist(lapply(strsplit(sample, "_"), function(x) x[2]))
```

##One way to generate group

```
#sample_split <- strsplit(sample, "_")
```

```
#group <- matrix(unlist(sample_split), ncol=3, byrow=T)[,2]
```

```
print(sample[1:4])
```

```
## [1] "Hi_2338_1" "Hi_2338_2" "Hi_2338_3" "Hi_2338_4"
```

Table 3.6: A table of the 3 most variable genes

	MT2A	ANXA1	ARHGDI	group
Hi_2338_1	12.211493	11.837198	9.543283	2338
Hi_2338_2	11.306216	8.098769	8.071623	2338
Hi_2338_3	11.926226	10.688626	9.720535	2338
Hi_2338_4	10.974207	9.386574	9.883376	2338
Hi_2338_5	14.603994	10.375072	9.970379	2338
Hi_2338_6	6.904604	11.155349	10.093510	2338
Hi_2338_7	12.436719	10.852249	7.742882	2338
Hi_2338_8	9.798375	9.783227	6.270716	2338
Hi_2338_9	11.743673	9.626476	9.250251	2338
Hi_2338_10	11.240016	11.303056	0.000000	2338

Table 3.7: A table of the 3 most variable genes in melted format

group	variable	value
2338	MT2A	12.211493
2338	MT2A	11.306216
2338	MT2A	11.926226
2338	MT2A	10.974207
2338	MT2A	14.603994
2338	MT2A	6.904604
2338	MT2A	12.436719
2338	MT2A	9.798375
2338	MT2A	11.743673
2338	MT2A	11.240016

```

print(group[1:4])

## [1] "2338" "2338" "2338" "2338"

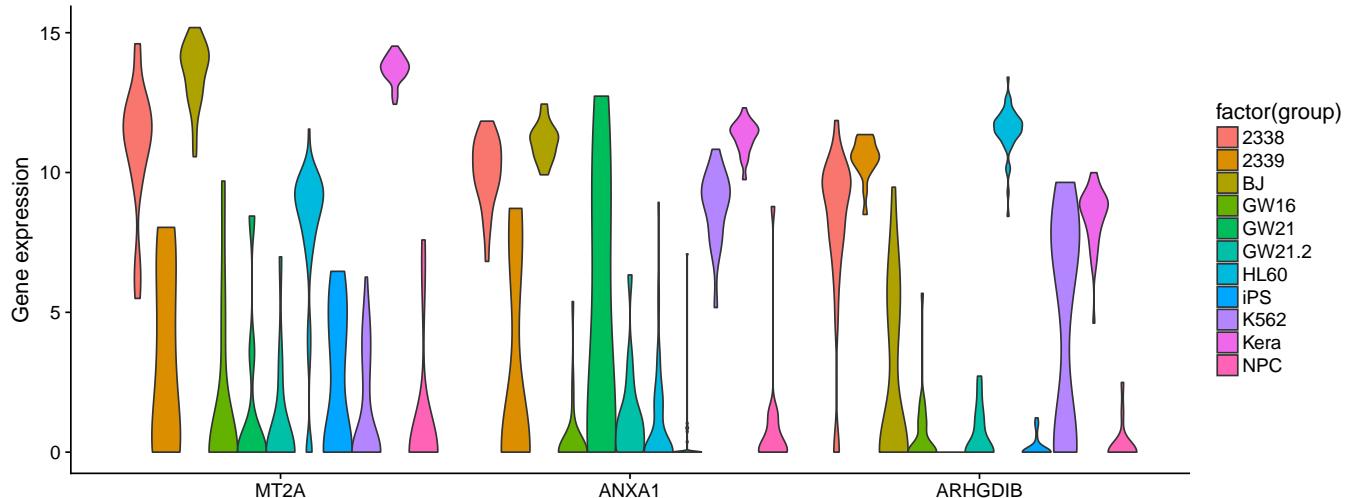
data_var3_scatter <- as.data.frame(data_var3_forPCA)
data_var3_scatter$group <- group
kable(corner(data_var3_scatter, r=10,c=5), booktabs=TRUE,
      caption="A table of the 3 most variable genes")

library(reshape2)
library(ggplot2)
data_var3_melt <- melt(data_var3_scatter, id.vars=c("group"))
kable(corner(data_var3_melt, r=10,c=5), booktabs=TRUE,
      caption="A table of the 3 most variable genes in melted format")

```

```
ggplot(data_var3_melt, aes(factor(variable), value)) + ylab("Gene expression") +
  geom_violin(aes(fill=factor(group)), stat="ydensity", position="dodge",
              scale="width", trim=TRUE) + xlab(NULL)

#ggplot(data_var3_melt, aes(factor(variable), value)) + ylab("Gene expression") +
#  geom_quasirandom(aes(color=factor(group))) + xlab(NULL)
```



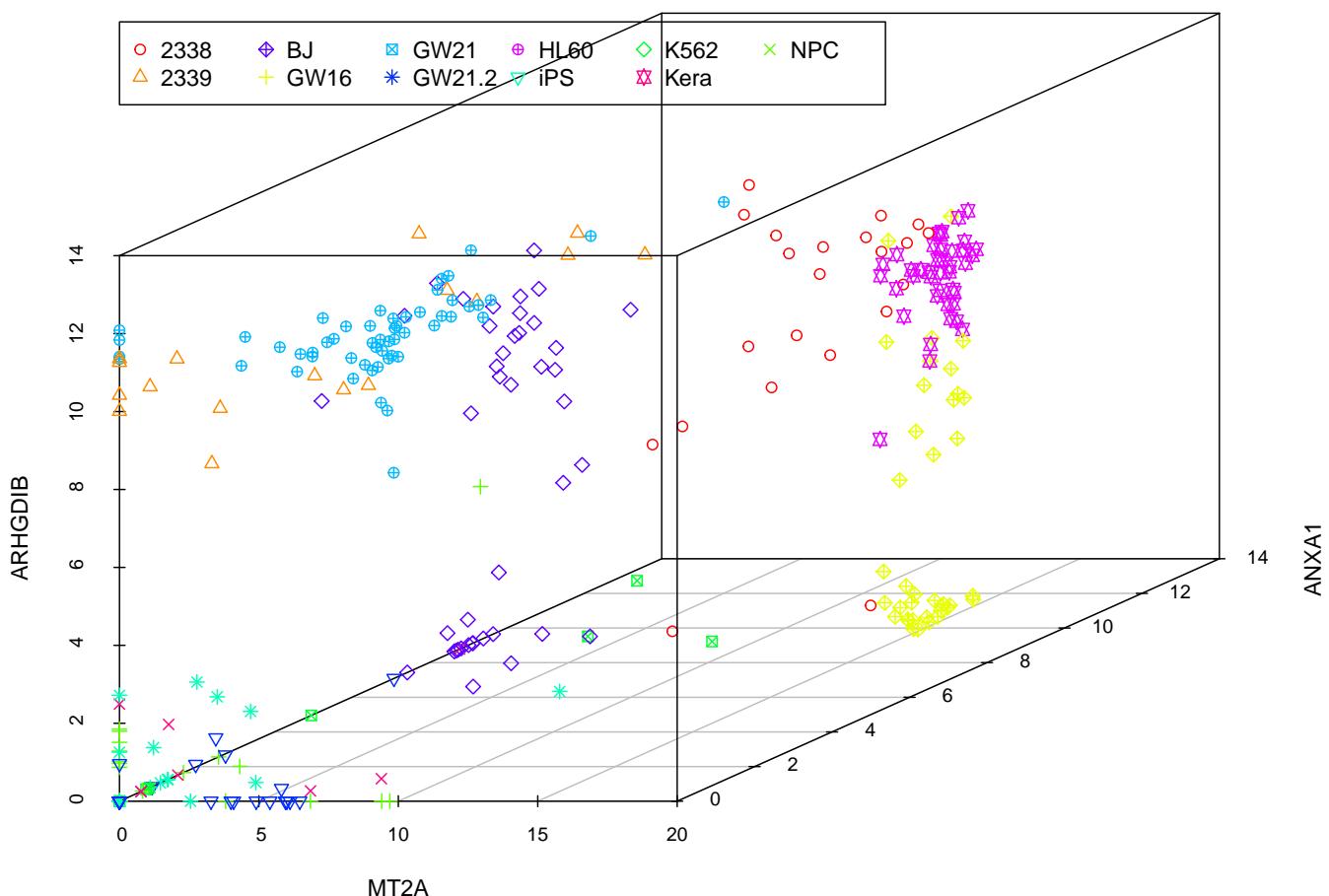
```
# 根据分组数目确定颜色变量
colorA <- rainbow(length(unique(group)))

# 根据每个样品的分组信息获取对应的颜色变量
colors <- colorA[as.factor(group)]

# 根据样品分组信息获得 legend 的颜色
colorl <- colorA[as.factor(unique(group))]

# 获得 PCH symbol 列表
pch_l <- as.numeric(as.factor(unique(group)))
# 产生每个样品的 pch symbol
pch <- pch_l[as.factor(group)]

scatterplot3d(data_var3_forPCA[,1:3], color=colors, pch=pch)
legend(0,10, legend=levels(as.factor(group)), col=colorl, pch=pch_l, xpd=T, horiz=F, ncol=6)
```



我们看到图中的样品并没有按照预先设定的标签完全分开。当然我们也可以通过其他方法筛选变异最大的三个基因，最终的分类效果不会相差很大。因为不管怎么筛选，我们都只用到了 3 个基因的表达量。

假如我们把这个数据用 PCA 来分类，结果是怎样的呢？

```
# Pay attention to the format of PCA input
# Rows are samples and columns are variables
data4_use_log2_t <- t(data4_use_log2)

# Add group column for plotting
data4_use_log2_label <- as.data.frame(data4_use_log2_t)
data4_use_log2_label$group <- group

# By default, prcomp will centralized the data using mean.
# Normalize data for PCA by dividing each data by column standard deviation.
# Often, we would normalize data.
# Only when we care about the real number changes other than the trends,
# `scale` can be set to TRUE.
# We will show the differences of scaling and un-scaling effects.
pca <- prcomp(data4_use_log2_t, scale=T)
```

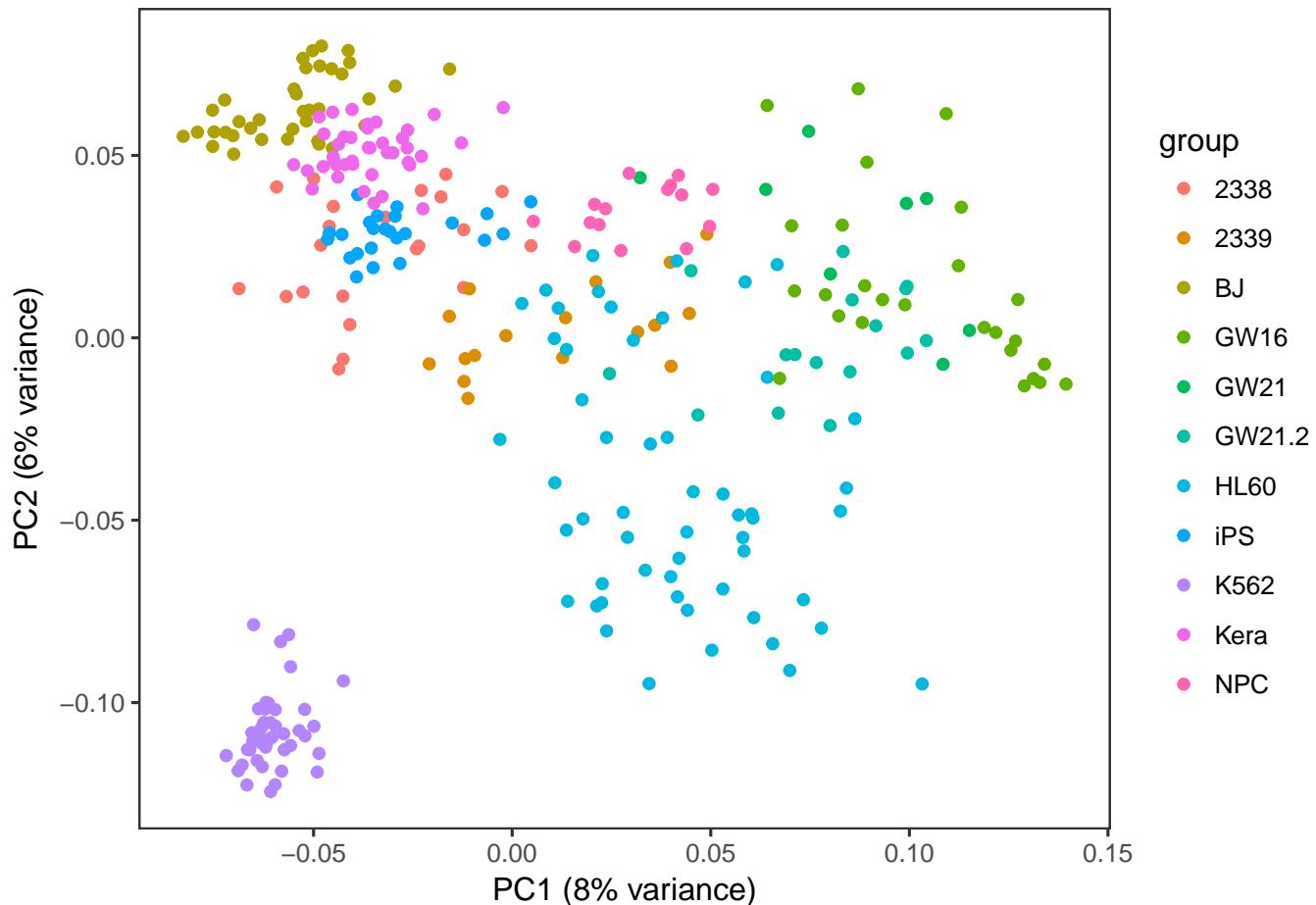
```
# sdev: standard deviation of the principle components.
# Square to get variance
percentVar <- pca$sdev^2 / sum(pca$sdev^2)

# To check what's in pca
print(str(pca))

## List of 5
## $ sdev      : num [1:301] 36.6 30.4 23.3 21.6 19.9 ...
## $ rotation: num [1:16482, 1:301] -0.01133 -0.01955 -0.00199 -0.00569 -0.0204 ...
## ... attr(*, "dimnames")=List of 2
##   ..$ : chr [1:16482] "MT2A" "ANXA1" "ARHGDIB" "RND3" ...
##   ..$ : chr [1:301] "PC1" "PC2" "PC3" "PC4" ...
## $ center    : Named num [1:16482] 6.5 5.47 5.43 4.23 4.1 ...
##   .. attr(*, "names")= chr [1:16482] "MT2A" "ANXA1" "ARHGDIB" "RND3" ...
## $ scale     : Named num [1:16482] 5.62 4.96 4.78 4.13 3.98 ...
##   .. attr(*, "names")= chr [1:16482] "MT2A" "ANXA1" "ARHGDIB" "RND3" ...
## $ x         : num [1:301, 1:301] -37.6 -28.6 -30.6 -43.7 -11.4 ...
##   .. attr(*, "dimnames")=List of 2
##     ..$ : chr [1:301] "Hi_2338_1" "Hi_2338_2" "Hi_2338_3" "Hi_2338_4" ...
##     ..$ : chr [1:301] "PC1" "PC2" "PC3" "PC4" ...
## - attr(*, "class")= chr "prcomp"
## NULL
```

从图中可以看到，数据呈现了一定的分类模式(当然这个分类结果也不理想，我们随后再进一步优化)。

```
library(ggfortify)
autoplot(pca, data=data4_use_log2_label, colour="group") +
  xlab(paste0("PC1 (", round(percentVar[1]*100), "% variance)")) +
  ylab(paste0("PC2 (", round(percentVar[2]*100), "% variance)")) + theme_bw() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
  theme(legend.position="right")
```



采用 3 个主成分获得的分类效果优于 2 个主成分，因为这样保留的原始信息更多。

```
# 根据分组数目确定颜色变量
colorA <- rainbow(length(unique(group)) )

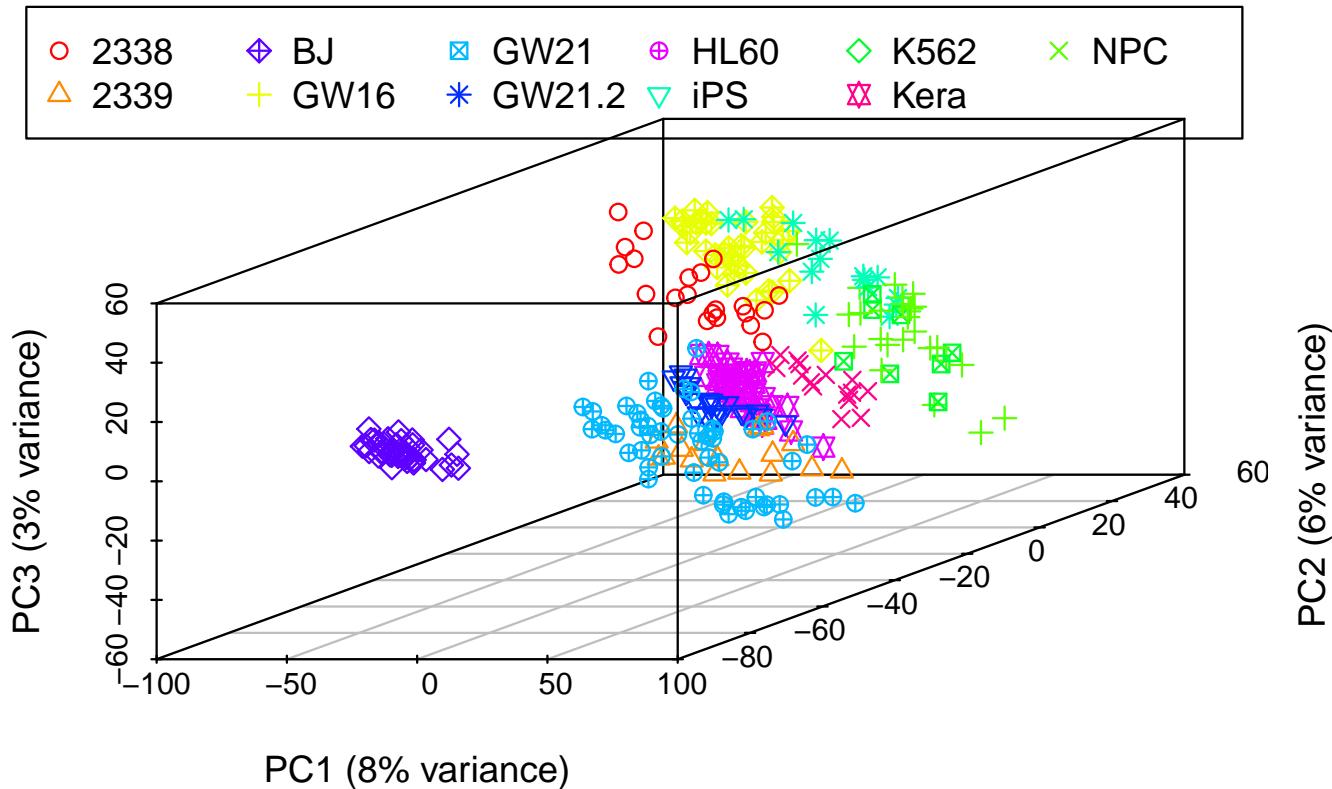
# 根据每个样品的分组信息获取对应的颜色变量
colors <- colorA[as.factor(group) ]

# 根据样品分组信息获得 legend 的颜色
colorl <- colorA[as.factor(unique(group))]

# 获得 PCH symbol 列表
pch_l <- as.numeric(as.factor(unique(group)))
# 产生每个样品的 pch symbol
pch <- pch_l[as.factor(group) ]

pc <- as.data.frame(pca$x)
scatterplot3d(x=pc$PC1, y=pc$PC2, z=pc$PC3, pch=pch, color=colors,
  xlab=paste0("PC1 (", round(percentVar[1]*100), "% variance)"),
  ylab=paste0("PC2 (", round(percentVar[2]*100), "% variance)"),
  zlab=paste0("PC3 (", round(percentVar[3]*100), "% variance)"))
```

```
legend(-3, 8, legend=levels(as.factor(group)), col=colorrl, pch=pch_l, xpd=T, horiz=F, ncol=6)
```



3.12.4 PCA 的实现原理

在上面的例子中，PCA 分析不是简单地选取 2 个或 3 个变化最大的基因，而是先把原始的变量做一个评估，计算各个变量各自的变异度（方差）和两两变量的相关度（协方差），得到一个协方差矩阵。在这个协方差矩阵中，对角线的值为每一个变量的方差，其它值为每两个变量的协方差。随后对原变量的协方差矩阵对角化处理，即求解其特征值和特征向量。原变量与特征向量的乘积（对原始变量的线性组合）即为新变量（回顾下线性代数中的矩阵乘法）；新变量的协方差矩阵为对角协方差矩阵且对角线上的方差由大到小排列；然后从新变量中选择信息最丰富也就是方差最大的前 2 个或前 3 个新变量也就是主成分用以可视化。下面我们一步步阐释这是怎么做的。

我们先回忆两个数学概念，方差和协方差。方差用来表示一组一维数据的离散程度。协方差表示 2 组一维数据的相关性。当协方差为 0 时，表示两组数据完全独立。当协方差为正时，表示一组数据增加时另外一组也会增加；当协方差为负时表示一组数据增加时另外一组数据会降低（与相关系数类似）。如果我们有很多组一维数据，比如很多基因的表达数据，就会得到很多协方差，这就构成了协方差矩阵。

方差和协方差的计算公式如下：

$$Var(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n - 1}$$

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{n - 1}$$

如果数据的均值为 0，这个公式可以进一步简化。简化后的公式把计算协方差转变为了矩阵乘法运算。这也是为什么 PCA 需要中心化数据。

$$\text{Var}(X) = \frac{\sum_{i=1}^n X_i^2}{n - 1}$$

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^n X_i Y_i}{n - 1}$$

$$\text{cov}(X, Y) = \frac{X_{n,m}^T Y_{n,m}}{n - 1}$$

假如我们有一个矩阵如下，

```
mat <- as.data.frame(matrix(rnorm(20, 0, 1), nrow=4))
colnames(mat) <- paste0("Gene_", letters[1:5])
rownames(mat) <- paste0("Samp_", 1:4)
mat

##           Gene_a     Gene_b     Gene_c     Gene_d     Gene_e
## Samp_1 -0.95854605 -1.4641695  1.4118119 -1.87124035  2.26763879
## Samp_2 -1.07372321  0.2325534  0.3048656 -1.22271308 -0.02195949
## Samp_3  0.27186238 -0.4106860  0.9816736  0.09329151 -0.17824977
## Samp_4   0.05494982  0.1682696 -1.6447469 -0.73149523  0.62384560
```

平均值中心化 (mean centering)：中心化数据使其平均值为 0

```
# mean-centering data for columns
# Get mean-value matrix first
mat_mean_norm <- mat - rep(colMeans(mat), rep.int(nrow(mat), ncol(mat)))
mat_mean_norm
```

```
##           Gene_a     Gene_b     Gene_c     Gene_d     Gene_e
## Samp_1 -0.5321818 -1.09566138  1.14841086 -0.9382011  1.59482000
## Samp_2 -0.6473589  0.60106152  0.04146455 -0.2896738 -0.69477827
## Samp_3  0.6982266 -0.04217784  0.71827253  1.0263308 -0.85106855
## Samp_4   0.4813141  0.53677770 -1.90814794  0.2015441 -0.04897318
```

CONTENTS

```
# mean-centering using scale for columns
scale(mat, center=T, scale=F)

##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## Samp_1 -0.5321818 -1.09566138  1.14841086 -0.9382011  1.59482000
## Samp_2 -0.6473589  0.60106152  0.04146455 -0.2896738 -0.69477827
## Samp_3  0.6982266 -0.04217784  0.71827253  1.0263308 -0.85106855
## Samp_4  0.4813141  0.53677770 -1.90814794  0.2015441 -0.04897318
## attr(,"scaled:center")
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## -0.4263643 -0.3685081  0.2634011 -0.9330393  0.6728188
```

中位数中心化 (median centering) : 如果数据变换范围很大或有异常值，中位数标准化效果会更好。

```
# median-centering data for columns
mat_median_norm <- mat - rep(apply(mat, 2, median), rep.int(nrow(mat), ncol(mat)))
mat_mean_norm
```

```
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## Samp_1 -0.5321818 -1.09566138  1.14841086 -0.9382011  1.59482000
## Samp_2 -0.6473589  0.60106152  0.04146455 -0.2896738 -0.69477827
## Samp_3  0.6982266 -0.04217784  0.71827253  1.0263308 -0.85106855
## Samp_4  0.4813141  0.53677770 -1.90814794  0.2015441 -0.04897318
```

我们可以计算 Gene_a 的方差为 0.4738249 (`var(mat$Gene_a)`) ; Gene_a 和 Gene_b 的协方差为 0.1409658。

mat 中 5 组基因的表达值的方差计算如下：

```
apply(mat, 2, var)
```

```
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## 0.4738249 0.6172194 1.8258369 0.6860357 1.2509613
```

mat 中 5 组基因表达值的协方差计算如下：

```
cov(mat)
```

```
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## Gene_a  0.4738249  0.1409658 -0.3516357  0.5001446 -0.3389245
```

CONTENTS

```
## Gene_b 0.1409658 0.6172194 -0.7626310 0.3062449 -0.7184596
## Gene_c -0.3516357 -0.7626310 1.8258369 -0.2456140 0.4282830
## Gene_d 0.5001446 0.3062449 -0.2456140 0.6860357 -0.7261170
## Gene_e -0.3389245 -0.7184596 0.4282830 -0.7261170 1.2509613
```

如果均值为 0，数值矩阵的协方差矩阵为矩阵的乘积（实际上是矩阵的转置与其本身的乘积除以变量的维数减 1）。

```
# Covariance matrix for Mean normalized matrix
cov(mat_mean_norm)
```

```
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## Gene_a 0.4738249 0.1409658 -0.3516357 0.5001446 -0.3389245
## Gene_b 0.1409658 0.6172194 -0.7626310 0.3062449 -0.7184596
## Gene_c -0.3516357 -0.7626310 1.8258369 -0.2456140 0.4282830
## Gene_d 0.5001446 0.3062449 -0.2456140 0.6860357 -0.7261170
## Gene_e -0.3389245 -0.7184596 0.4282830 -0.7261170 1.2509613
```

```
# Covariance matrix for Mean normalized matrix
# crossprod: matrix multiplication
crossprod(as.matrix(mat_mean_norm)) / (nrow(mat_mean_norm)-1)
```

```
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## Gene_a 0.4738249 0.1409658 -0.3516357 0.5001446 -0.3389245
## Gene_b 0.1409658 0.6172194 -0.7626310 0.3062449 -0.7184596
## Gene_c -0.3516357 -0.7626310 1.8258369 -0.2456140 0.4282830
## Gene_d 0.5001446 0.3062449 -0.2456140 0.6860357 -0.7261170
## Gene_e -0.3389245 -0.7184596 0.4282830 -0.7261170 1.2509613
```

```
# Use %*% for matrix multiplication (slower)
t(as.matrix(mat_mean_norm)) %*% as.matrix(mat_mean_norm) / (nrow(mat_mean_norm)-1)
```

```
##           Gene_a      Gene_b      Gene_c      Gene_d      Gene_e
## Gene_a 0.4738249 0.1409658 -0.3516357 0.5001446 -0.3389245
## Gene_b 0.1409658 0.6172194 -0.7626310 0.3062449 -0.7184596
## Gene_c -0.3516357 -0.7626310 1.8258369 -0.2456140 0.4282830
## Gene_d 0.5001446 0.3062449 -0.2456140 0.6860357 -0.7261170
## Gene_e -0.3389245 -0.7184596 0.4282830 -0.7261170 1.2509613
```

用矩阵形式书写如下，便于理解

$$\text{cov}(\text{mat}) = \frac{1}{3} \begin{bmatrix} -0.53 & -0.65 & 0.7 & 0.48 \\ -1.1 & 0.6 & -0.04 & 0.54 \\ 1.15 & 0.04 & 0.72 & -1.91 \\ -0.94 & -0.29 & 1.03 & 0.2 \\ 1.59 & -0.69 & -0.85 & -0.05 \end{bmatrix} \begin{bmatrix} -0.53 & -1.1 & 1.15 & -0.94 & 1.59 \\ -0.65 & 0.6 & 0.04 & -0.29 & -0.69 \\ 0.7 & -0.04 & 0.72 & 1.03 & -0.85 \\ 0.48 & 0.54 & -1.91 & 0.2 & -0.05 \end{bmatrix} = \begin{bmatrix} 0.47 & 0.14 & -0.35 & 0.5 \\ 0.14 & 0.62 & -0.76 & 0.31 \\ -0.35 & -0.76 & 1.83 & -0.25 \\ 0.5 & 0.31 & -0.25 & 0.69 \\ -0.34 & -0.72 & 0.43 & -0.73 \end{bmatrix}$$

根据前面的描述，原始变量的协方差矩阵表示原始变量自身的方差（协方差矩阵的主对角线位置）和原始变量之间的相关程度（非主对角线位置）。如果从这些数据中筛选主成分，则要选择方差大（主对角线值大），且与其它已选变量之间相关性最小的变量（非主对角线值很小）。如果这些原始变量之间毫不相关，则它们的协方差矩阵在除主对角线外其它地方的值都为 0，这种矩阵成为对角矩阵。

而做 PCA 分析就是产生一组新的变量，使得新变量的协方差矩阵为对角阵，满足上面的要求。从而达到去冗余的目的。然后再选取方差大的变量，实现降维和去噪。

如果正向推导，这种组合可能会有很多种，一一计算会比较麻烦。那反过来看呢？我们不去寻找这种组合，而是计算如何使原变量的协方差矩阵变为对角阵。

数学推导中谨记的两个概念：

1. 假设：把未求解到的变量假设出来，用符号代替；这样有助于思考和演算
2. 逆向：如果正向推导求不出，不妨倒着来；尽量多的利用已有信息

前面提到，新变量 $(Y_{m,k})$ 是原始变量 $(X_{m,n})$ （原始变量的协方差矩阵为 $(C_{n,n})$ ）的线性组合，那么假设我们找到了这么一个线性组合（命名为特征矩阵 $(P_{n,k})$ ），得到一组新变量 $Y_{m,k} = X_{m,n}P_{n,k}$ ，并且新变量的协方差矩阵 $(D_{k,k})$ 为对角阵。那么这个特征矩阵 $(P_{n,k})$ 需要符合什么条件呢？

从矩阵运算可以看出，最终的特征矩阵 $(P_{n,k})$ 需要把原变量协方差矩阵 $(C_{n,n})$ 转换为对角阵（因为新变量的协方差矩阵 $(D_{k,k})$ 为对角阵），并且对角元素从大到小排列（保证每个主成分的贡献度依次降低）。

现在就把求解新变量的任务转变为了求解原变量协方差矩阵的对角化问题了。在线性代数中，矩阵对角化的问题就是求解矩阵的特征值和特征向量的问题。

我们举一个例子讲述怎么求解特征值和特征向量。

假设 $A_{n,n}$ 为 n 阶对称阵，如存在 λ 和非零向量 x ，使得 $Ax = \lambda x$ ，则称 λ 为矩阵 $A_{n,n}$ 的特征值，非零向量 x 为矩阵 $A_{n,n}$ 对应于特征值 λ 的特征向量。

根据这个定义可以得出 $(A_{n,n} - \lambda E)x = 0$ ，由于 x 为非零向量，所以行列式 $|A - \lambda E| = 0$ 。

由此求解出 n 个根 $\lambda_1, \lambda_2, \dots, \lambda_n$ 就是矩阵 A 的特征值。

回顾下行列式的计算：

- 行列式的值为行列式第一列的每一个数乘以它的余子式（余子式是行列式中除去当前元素所在行和列之后剩下的行列式）。
- 当行列式中存在线性相关的行或列或者有一行或一列元素全为 0 时，行列式的值为 0。
- 上三角形行列式的值为其主对角线上元素的乘积。
- 互换行列式的两行或两列，行列式变号。
- 行列式的某一列（行）乘以同意书加到另一列（列）对应元素上去，行列式不变。

假如我们有一个矩阵 $A = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$ ，如何计算它的特征值和特征向量呢？

则 λ 的值为 2 或 4。

对 $\lambda_1 = 2$ 时，求解 $(A - 2E)x = \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} x = 0$ ，得 $x = k \begin{vmatrix} 1 \\ 1 \end{vmatrix}$ ，则对于 $\lambda_1 = 2$ 时的特征向量 $p_1 = \begin{vmatrix} 1 \\ 1 \end{vmatrix}$

对 $\lambda_2 = 4$ 时，求解 $(A - 2E)x = \begin{vmatrix} -1 & -1 \\ -1 & -1 \end{vmatrix} x = 0$ ，得 $x = k \begin{vmatrix} 1 \\ -1 \end{vmatrix}$ ，则对于 $\lambda_2 = 4$ 时的特征向量 $p_2 = \begin{vmatrix} 1 \\ -1 \end{vmatrix}$

以上就完成了 PCA 的数学推导。

3.12.5 简单的 PCA 实现

我们使用前面用到的数据 `data3` 来演示下如何用 R 函数实现 PCA 的计算，并与 R 中自带的 `prcomp` 做个比较。

```
library(knitr)
kable(headTail(data3), booktabs=T, caption="Expression profile for 3 genes in 100 samples")
```

标准化数据

```
data3_center_scale <- scale(data3[, 1:3], center=T, scale=T)
kable(headTail(data3_center_scale), booktabs=T,
      caption="Normalized expression for 3 genes in 100 samples")
```

计算协方差矩阵

```
data3_center_scale_cov <- cov(data3_center_scale)
kable(data3_center_scale_cov, booktabs=T,
      caption="Covariance matrix for 3 genes in 100 samples")
```

Table 3.8: Expression profile for 3 genes in 100 samples

	Gene1	Gene2	Gene3	Group
a1	4.71	5.01	4.64	a
a2	4.66	4.82	5.03	a
a3	4.74	4.84	4.93	a
a4	4.66	5.02	5.26	a
...	NA
b47	20.11	4.84	4.83	b
b48	19.44	5.07	4.99	b
b49	19.39	5.3	5.04	b
b50	20.08	4.95	5.23	b

Table 3.9: Normalized expression for 3 genes in 100 samples

	Gene1	Gene2	Gene3
a1	-1.04	0.14	-1.06
a2	-1.04	-0.8	-0.98
a3	-1.03	-0.73	-1
a4	-1.04	0.19	-0.93
...
b47	1.02	-0.68	-1.02
b48	0.93	0.4	-0.99
b49	0.92	1.53	-0.98
b50	1.02	-0.2	-0.94

Table 3.10: Covariance matrix for 3 genes in 100 samples

	Gene1	Gene2	Gene3
Gene1	1.0000000	-0.0640112	-0.0004707
Gene2	-0.0640112	1.0000000	0.1878111
Gene3	-0.0004707	0.1878111	1.0000000

Table 3.11: PCA generated matrix for the expression of 3 genes in 100 samples

	PC1	PC2	PC3
a1	0.37	-1.32	-0.57
a2	0.98	-1.3	0.15
a3	0.95	-1.3	0.08
a4	0.25	-1.29	-0.52
...
b47	1.4	0.64	-0.43
b48	0.59	0.56	-1.15
b49	-0.22	0.56	-1.94
b50	1	0.66	-0.72

求解特征值和特征向量

```
data3_center_scale_cov_eigen <- eigen(data3_center_scale_cov)

# 特征值，从大到小排序
data3_center_scale_cov_eigen$values
```

```
## [1] 1.1985640 0.9997125 0.8017235
```

```
# 特征向量，每一列为对应特征值的特征向量
data3_center_scale_cov_eigen$vectors
```

```
## [,1]      [,2]      [,3]
## [1,]  0.2294536 0.946533046 -0.2267736
## [2,] -0.7068494 0.001878625 -0.7073616
## [3,] -0.6691151 0.322601400  0.6694873
```

产生新的矩阵

```
pc_select = 3
label = paste0("PC", 1:pc_select)
data3_new <- data3_center_scale %*% data3_center_scale_cov_eigen$vectors[, 1:pc_select]
colnames(data3_new) <- label
kable(headTail(data3_new), booktabs=T,
       caption="PCA generated matrix for the expression of 3 genes in 100 samples")
```

比较原始数据和新产生的主成分对样品的聚类

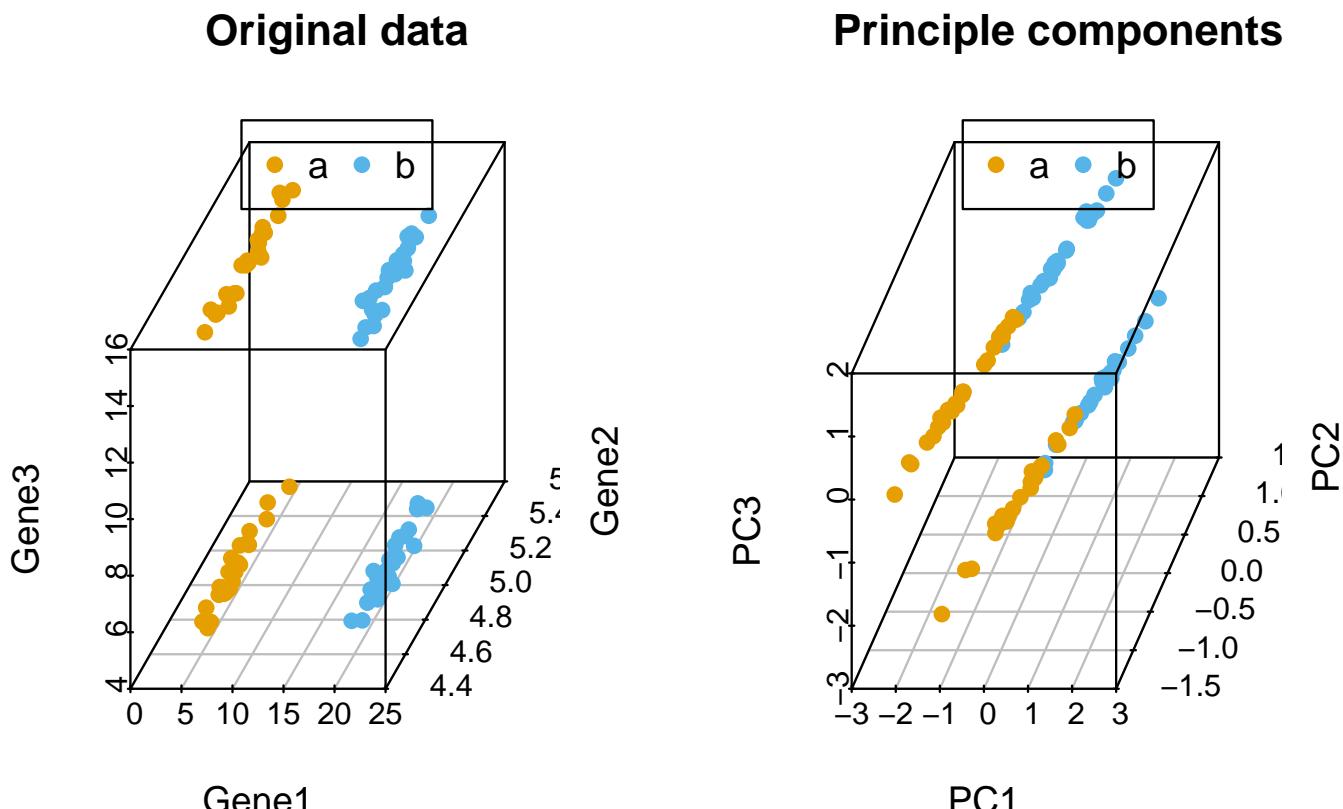
```
#library(scatterplot3d)
color1 <- c("#E69F00", "#56B4E9")
# Extract same number of colors as the Group and same Group would have same color.
colors <- color1[as.numeric(data3$Group)]

# 1 row 2 columns
par(mfrow=c(1, 2))

scatterplot3d(data3[,1:3], color=colors, angle=55, pch=16, main="Original data")
legend("top", legend=levels(data3$Group), col=color1, pch=16, xpd=T, horiz=T)

scatterplot3d(data3_new, color=colors, angle=55, pch=16, main="Principle components")
legend("top", legend=levels(data3$Group), col=color1, pch=16, xpd=T, horiz=T)

#par(mfrow=c(1,1))
```



利用 prcomp 进行主成分分析

```
pca_data3 <- prcomp(data3[,1:3], center=TRUE, scale=TRUE)

#Show what's in the result returned by prcomp
str(pca_data3)
```

Table 3.12: PCA generated matrix usig princomp for the expression of 3 genes in 100 samples

	PC1	PC2	PC3
a1	0.37	1.32	-0.57
a2	0.98	1.3	0.15
a3	0.95	1.3	0.08
a4	0.25	1.29	-0.52
...
b47	1.4	-0.64	-0.43
b48	0.59	-0.56	-1.15
b49	-0.22	-0.56	-1.94
b50	1	-0.66	-0.72

```

## List of 5
## $ sdev     : num [1:3] 1.095 1 0.895
## $ rotation: num [1:3, 1:3] 0.22945 -0.70685 -0.66912 -0.94653 -0.00188 ...
## ... attr(*, "dimnames")=List of 2
##   ..$ : chr [1:3] "Gene1" "Gene2" "Gene3"
##   ..$ : chr [1:3] "PC1" "PC2" "PC3"
## $ center   : Named num [1:3] 12.47 4.99 9.98
##   .. attr(*, "names")= chr [1:3] "Gene1" "Gene2" "Gene3"
## $ scale    : Named num [1:3] 7.495 0.207 5.063
##   .. attr(*, "names")= chr [1:3] "Gene1" "Gene2" "Gene3"
## $ x        : num [1:100, 1:3] 0.373 0.98 0.946 0.255 1.109 ...
##   .. attr(*, "dimnames")=List of 2
##     ..$ : chr [1:100] "a1" "a2" "a3" "a4" ...
##     ..$ : chr [1:3] "PC1" "PC2" "PC3"
##   - attr(*, "class")= chr "prcomp"

# 新的数据，与前面计算的抑制
data3_pca_new <- pca_data3$x
kable(headTail(data3_pca_new), booktabs=T,
      caption="PCA generated matrix usig princomp for the expression of 3 genes in 100 samples")

# 特征向量，与我们前面计算的一致 (特征向量的符号是任意的)
pca_data3$rotation

##          PC1         PC2         PC3
## Gene1  0.2294536 -0.946533046 -0.2267736
## Gene2 -0.7068494 -0.001878625 -0.7073616
## Gene3 -0.6691151 -0.322601400  0.6694873

```

比较手动实现的 PCA 与 princomp 实现的 PCA 的聚类结果

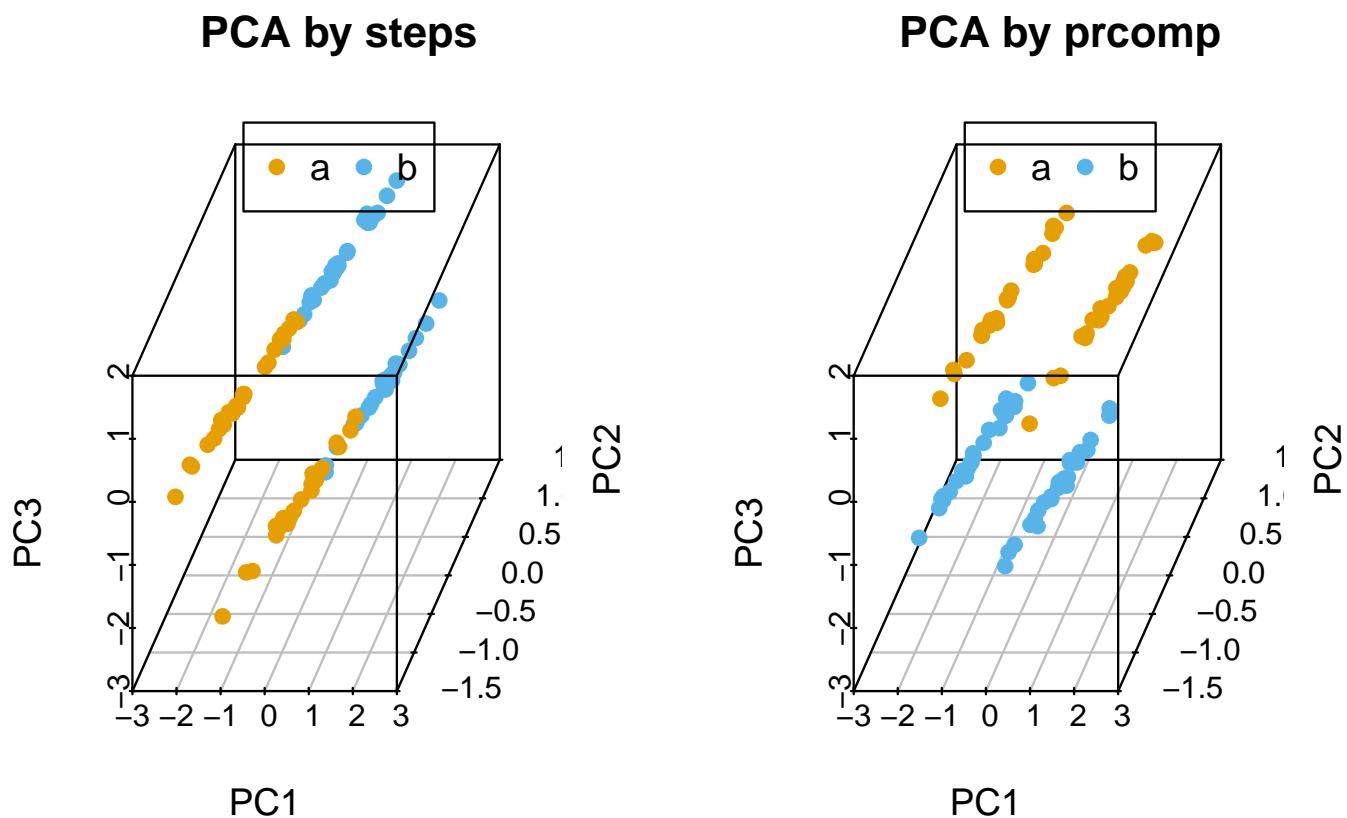
```
#library(scatterplot3d)
colorl <- c("#E69F00", "#56B4E9")
# Extract same number of colors as the Group and same Group would have same color.
colors <- colorl[as.numeric(data3$Group)]

# 1 row 2 columns
par(mfrow=c(1, 2))

scatterplot3d(data3_new, color=colors, angle=55, pch=16, main="PCA by steps")
legend("top", legend=levels(data3$Group), col=colorl, pch=16, xpd=T, horiz=T)

scatterplot3d(data3_pca_new, color=colors, angle=55, pch=16, main="PCA by prcomp")
legend("top", legend=levels(data3$Group), col=colorl, pch=16, xpd=T, horiz=T)

#par(mfrow=c(1,1))
```



自定义 PCA 计算函数

```
ct_PCA <- function(data, center=TRUE, scale=TRUE) {
  data_norm <- scale(data, center=center, scale=scale)
  data_norm_cov <- crossprod(as.matrix(data_norm)) / (nrow(data_norm)-1)
  data_eigen <- eigen(data_norm_cov)
```

```

rotation <- data_eigen$vectors
label <- paste0('PC', c(1:ncol(rotation)))
colnames(rotation) <- label
sdev <- sqrt(data_eigen$values)
data_new <- data_norm %*% rotation
colnames(data_new) <- label
ct_pca <- list('rotation'=rotation, 'x'=data_new, 'sdev'=sdev)
return(ct_pca)
}

```

比较有无 scale 对聚类的影响，从图中可以看到，如果不对数据进行 scale 处理，样品的聚类结果更像原始数据，本身数值大的基因对主成分的贡献会大。如果关注的是每个变量自身的实际方差对样品分类的贡献，则不应该 SCALE；如果关注的是变量的相对大小对样品分类的贡献，则应该 SCALE，以防数值高的变量导入的大方差引入的偏见。

```
data3_pca_noscale_step = ct_PCA(data3[,1:3], center=TRUE, scale=FALSE)
```

特征向量

```
data3_pca_noscale_step$rotation
```

```

##          PC1        PC2        PC3
## [1,] 0.9999982499 0.000609949 -0.001768688
## [2,] -0.0017733305 0.007697251 -0.999968803
## [3,] -0.0005963159 0.999970190  0.007698319

```

新变量

```
data3_pca_noscale_pc <- data3_pca_noscale_step$x
```

```

#library(scatterplot3d)
color1 <- c("#E69F00", "#56B4E9")
# Extract same number of colors as the Group and same Group would have same color.
colors <- color1[as.numeric(data3$Group)]

# 1 row 2 columns
par(mfrow=c(2, 2))

scatterplot3d(data3[,c(1,3,2)], color=colors, angle=55, pch=16, main="Original data")
legend("top", legend=levels(data3$Group), col=color1, pch=16, xpd=T, horiz=T)

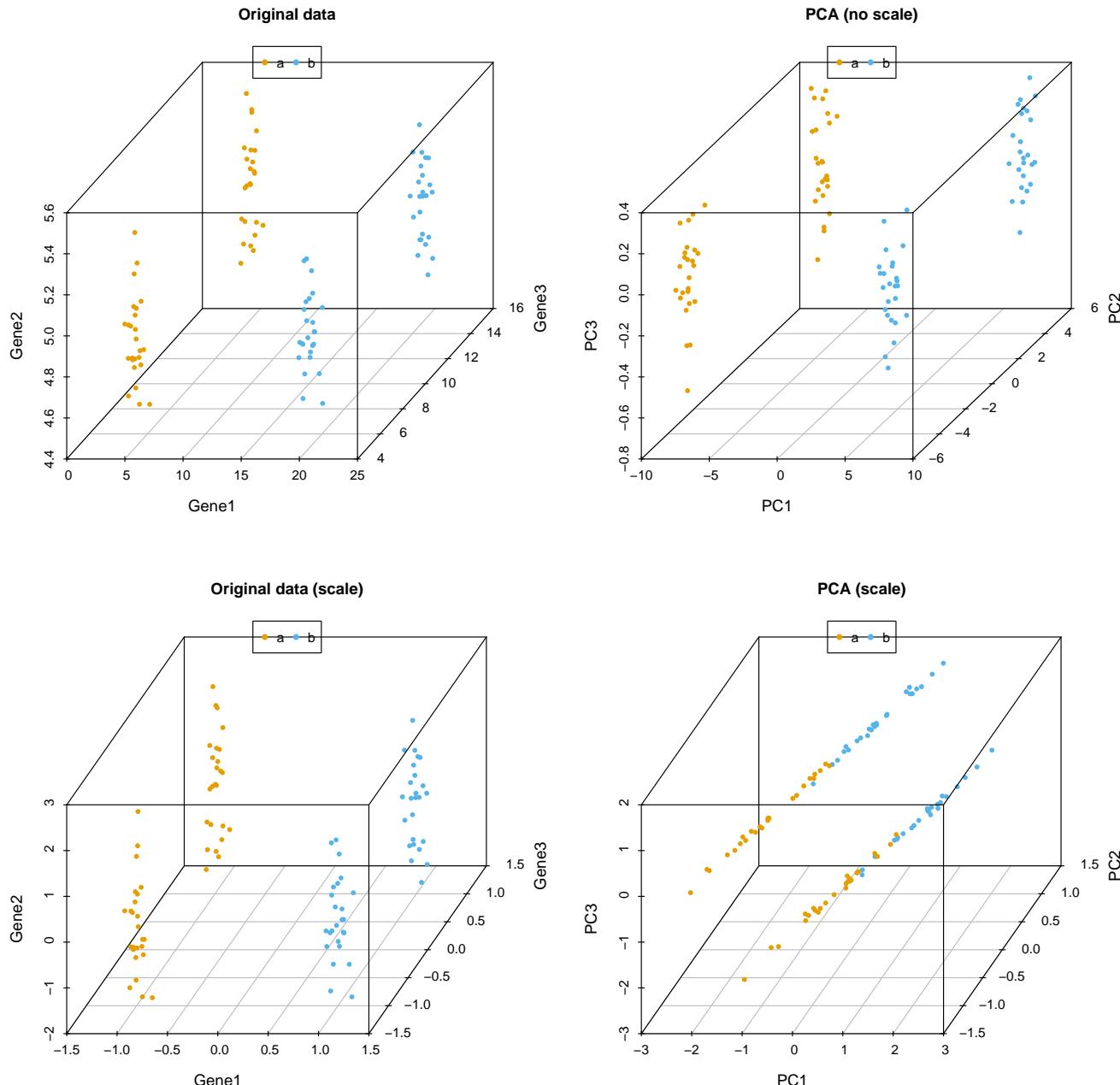
scatterplot3d(data3_pca_noscale_pc, color=colors, angle=55, pch=16, main="PCA (no scale)")
legend("top", legend=levels(data3$Group), col=color1, pch=16, xpd=T, horiz=T)

scatterplot3d(data3_center_scale[,c(1,3,2)], color=colors, angle=55, pch=16,
               main="Original data (scale)")
legend("top", legend=levels(data3$Group), col=color1, pch=16, xpd=T, horiz=T)

```

```
scatterplot3d(data3_new, color=colors, angle=55, pch=16, main="PCA (scale)")
legend("top", legend=levels(data3$Group), col=colorl, pch=16, xpd=T, horiz=T)

#par(mfrow=c(1,1))
```



3.12.6 PCA 结果解释

`prcomp` 函数会返回主成分的标准差、特征向量和主成分构成的新矩阵。接下来，探索下不同主成分对数据差异的贡献和主成分与原始变量的关系。

CONTENTS

- 主成分的平方为特征值，其含义为每个主成分可以解释的数据差异，计算方式为 `eigenvalues = (pca$sdev)^2`
- 每个主成分可以解释的数据差异的比例为 `percent_var = eigenvalues*100/sum(eigenvalues)`
- 可以使用 `summary(pca)` 获取以上两条信息。

这两个信息可以判断主成分分析的质量：

- 成功的降维需要保证在前几个为数不多的主成分对数据差异的解释可以达到 80-90%。

指导选择主成分的数目：

- 选择的主成分足以解释的总方差大于 80% (方差比例碎石图)
- 从前面的协方差矩阵可以看到，自动定标 (scale) 的变量的方差为 1 (协方差矩阵对角线的值)。待选择的主成分应该是那些方差大于 1 的主成分，即其解释的方差大于原始变量 (特征值碎石图，方差大于 1，特征值也会大于 1，反之亦然)。

鉴定核心变量和变量间的隐性关系：

- 原始变量与主成分的相关性 `Variable correlation with PCs (var.cor) = loadings * sdev`
- 原始数据对主成分的贡献度 `var.cor^2 / (total var.cor^2)`

在测试数据中，`scale` 后，三个主成分对数据差异的贡献度大都在 30% 左右，而未 `scale` 的数据，三个主成分对数据差异的贡献度相差很大。这是因为三个基因由于自身表达量级所引起的方差的差异导致它们各自对数据的权重差异，从而使主成分偏向于数值大的变量。

3.12.7 PCA 应用于测试数据

前面用到一组比较大的测试数据集，并做了 PCA 分析，现在测试不同的处理对结果的影响。

首先回顾下我们用到的数据。

```
library("gridExtra")
## 
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
```

Table 3.13: Single cell gene expression data

	MT2A	ANXA1	ARHGDI	RND3	BLVRB
Hi_2338_1	12.211	11.837	9.543	9.762	9.162
Hi_2338_2	11.306	8.099	8.072	10.107	9.423
Hi_2338_3	11.926	10.689	9.721	9.388	9.694
Hi_2338_4	10.974	9.387	9.883	8.792	9.767
Hi_2338_5	14.604	10.375	9.970	8.815	9.350

```
##  
##      combine  
  
# Pay attention to the format of PCA input  
# Rows are samples and columns are variables  
# data4_use_log2_t <- t(data4_use_log2)  
  
# Add group column for plotting  
# data4_use_log2_label <- as.data.frame(data4_use_log2_t)  
# data4_use_log2_label$group <- group  
  
kable(corner(data4_use_log2_label), digits=3, caption="Single cell gene expression data")
```

比较对数运算和 scale 对样品分类的影响。

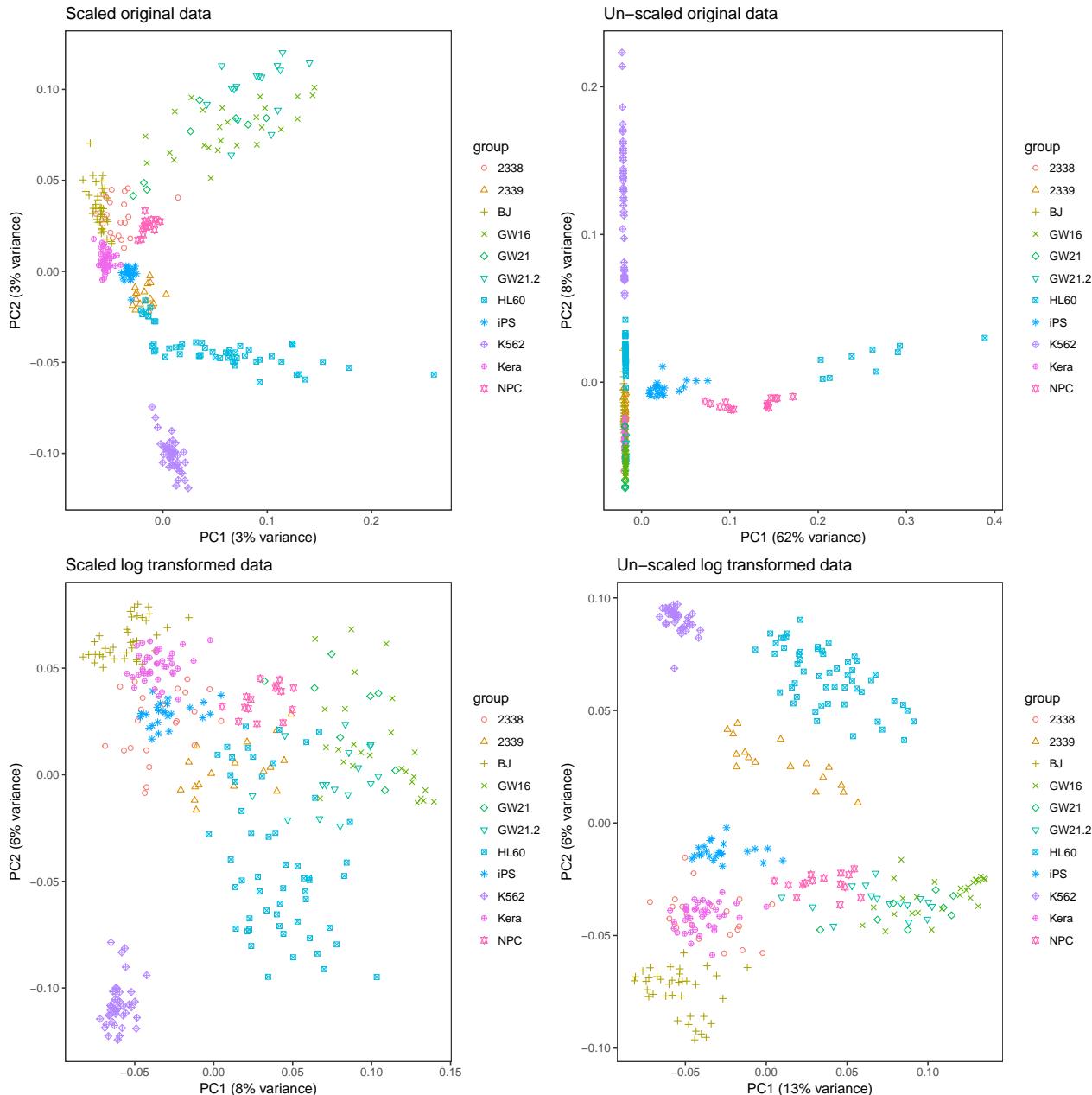
```
ct_pca_2d_plot <- function(pca, data_with_label, labelName='group', title='PCA') {  
  # sdev: standard deviation of the principle components.  
  # Square to get variance  
  percentVar <- pca$sdev^2 / sum( pca$sdev^2 )  
  #data <- data_with_label  
  #data[labelName] <- factor(unlist(data[labelName]))  
  level <- length(unique(unlist(data_with_label[labelName])))  
  shapes = (1:level)%%30  # maximum allow 30 types of symbols  
  p = autoplot(pca, data=data_with_label, colour=labelName, shape=labelName) +  
    scale_shape_manual(values=shapes) +  
    xlab(paste0("PC1 (", round(percentVar[1]*100), "% variance)")) +  
    ylab(paste0("PC2 (", round(percentVar[2]*100), "% variance)")) +  
    theme_bw() + theme(legend.position="right") + labs(title=title) +  
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())  
  return(p)  
}
```

```
# By default, prcomp will centralized the data using mean.  
# Normalize data for PCA by dividing each data by column standard deviation.  
# Often, we would normalize data.  
# Only when we care about the real number changes other than the trends,  
# `scale` can be set to TRUE.
```

```
# We will show the differences of scaling and un-scaling effects.
data4_use_t <- t(data4_use)
ori_scale_pca_test <- prcomp(data4_use_t, scale=T)
ori_no_scale_pca_test <- prcomp(data4_use_t, scale=F)
log_scale_pca_test <- prcomp(data4_use_log2_t, scale=T)
log_no_scale_pca_test <- prcomp(data4_use_log2_t, scale=F)

ori_scale_pca_plot = ct_pca_2d_plot(ori_scale_pca_test, data4_use_log2_label,
                                    title="Scaled original data")
ori_no_scale_pca_plot = ct_pca_2d_plot(ori_no_scale_pca_test, data4_use_log2_label,
                                       title="Un-scaled original data")
log_scale_pca_plot = ct_pca_2d_plot(log_scale_pca_test, data4_use_log2_label,
                                      title="Scaled log transformed data")
log_no_scale_pca_plot = ct_pca_2d_plot(log_no_scale_pca_test, data4_use_log2_label,
                                         title="Un-scaled log transformed data")

a__ <- grid.arrange(ori_scale_pca_plot, ori_no_scale_pca_plot, log_scale_pca_plot,
                     log_no_scale_pca_plot, ncol=2)
```



如果首先提取 500 个变化最大的基因，再执行 PCA 分析会怎样呢？

```

data4_use_mad <- apply(data4_use, 1, mad)
data4_use_mad_top500 <- t(data4_use[rev(order(data4_use_mad))][1:500],)

data4_use_log2_mad <- apply(data4_use_log2, 1, mad)
data4_use_log2_mad_top500 <- t(data4_use_log2[rev(order(data4_use_log2_mad))][1:500],)

ori_scale_pca_top500 <- prcomp(data4_use_mad_top500, scale=T)
ori_no_scale_pca_top500 <- prcomp(data4_use_mad_top500, scale=F)
log_scale_pca_top500 <- prcomp(data4_use_log2_mad_top500, scale=T)
log_no_scale_pca_top500 <- prcomp(data4_use_log2_mad_top500, scale=F)

```

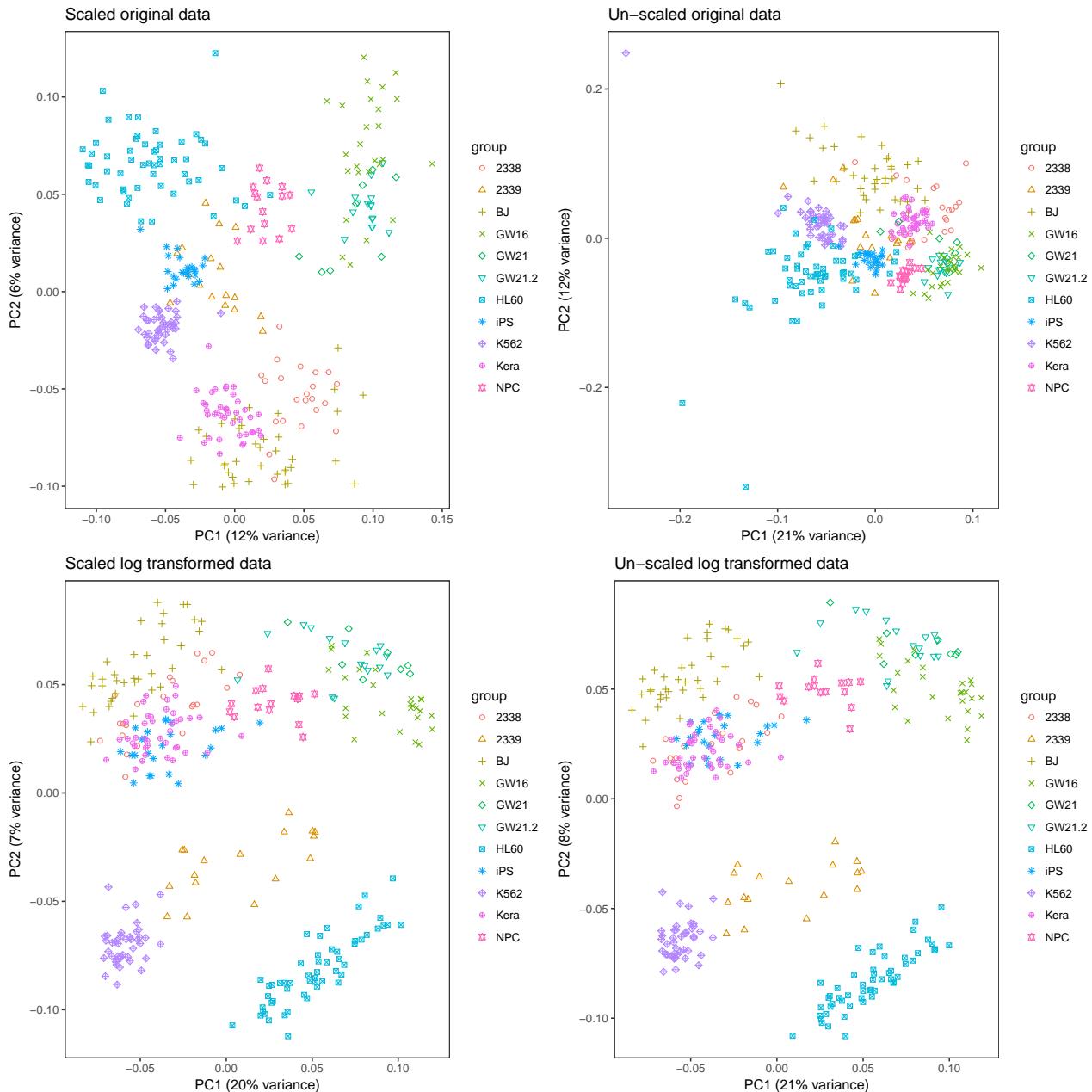
CONTENTS

```

ori_scale_pca_plot_t5 = ct_pca_2d_plot(ori_scale_pca_top500, data4_use_log2_label,
                                         title="Scaled original data")
ori_no_scale_pca_plot_t5 = ct_pca_2d_plot(ori_no_scale_pca_top500,
                                            data4_use_log2_label, title="Un-scaled original data")
log_scale_pca_plot_t5 = ct_pca_2d_plot(log_scale_pca_top500,
                                         data4_use_log2_label, title="Scaled log transformed data")
log_no_scale_pca_plot_t5 = ct_pca_2d_plot(log_no_scale_pca_top500,
                                            data4_use_log2_label, title="Un-scaled log transformed data")

a__ <- grid.arrange(ori_scale_pca_plot_t5, ori_no_scale_pca_plot_t5,
                     log_scale_pca_plot_t5, log_no_scale_pca_plot_t5, ncol=2)

```



3.12.8 PCA 注意事项

1. 一般说来，在PCA之前原始数据需要中心化（centering，数值减去平均值）。中心化的方法很多，除了平均值中心化（mean-centering）外，还包括其它更稳健的方法，比如中位数中心化等。
2. 除了中心化以外，定标（Scale，数值除以标准差）也是数据前处理中需要考虑的一点。如果数据没有定标，则原始数据中方差大的变量对主成分的贡献会很大。数据的方差与其量级成指数关系，比如一组数据 $(1, 2, 3, 4)$ 的方差是 1.67，而 $(10, 20, 30, 40)$ 的方差就是 167，数据变大 10 倍，方差放大了 100 倍。
3. 但是定标（scale）可能会有一些负面效果，因为定标后变量之间的权重就是变得相同。如果我们的变量中有噪音的话，我们就在无形中把噪音和信息的权重变得相同，但PCA本身无法区分信号和噪音。在这样的情形下，我们就不必做定标。
4. 一般而言，对于度量单位不同的指标或是取值范围彼此差异非常大的指标不直接由其协方差矩阵出发进行主成分分析，而应该考虑对数据的标准化。比如度量单位不同，有万人、万吨、万元、亿元，而数据间的差异性也非常大，小则几十大则几万，因此在用协方差矩阵求解主成分时存在协方差矩阵中数据的差异性很大。在后面提取主成分时发现，只提取了一个主成分，而此时并不能将所有的变量都解释到，这就没有真正起到降维的作用。此时就需要对数据进行定标（scale），这样提取的主成分可以覆盖更多的变量，这就实现主成分分析的最终目的。但是对原始数据进行标准化后更倾向于使得各个指标的作用在主成分分析构成中相等。对于数据取值范围不大或是度量单位相同的指标进行标准化处理后，其主成分分析的结果与仍由协方差矩阵出发求得的结果有较大区别。这是因为对数据标准化的过程实际上就是抹杀原有变量离散程度差异的过程，标准化后方差均为 1，而实际上方差是对数据信息的重要概括形式，也就是说，对原始数据进行标准化后抹杀了一部分重要信息，因此才使得标准化后各变量在主成分构成中的作用趋于相等。因此，对同度量或是取值范围在同量级的数据还是直接使用非定标数据求解主成分为宜。
5. 中心化和定标都会受数据中离群值（outliers）或者数据不均匀（比如数据被分为若干个小组）的影响，应该用更稳健的中心化和定标方法。
6. PCA 也存在一些限制，例如它可以很好的解除线性相关，但是对于高阶相关性就没有办法了，对于存在高阶相关性的数据，可以考虑 Kernel PCA，通过 Kernel 函数将非线性相关转为线性相关，关于这点就不展开讨论了。另外，PCA 假设数据各主特征是分布在正交方向上，如果在非正交方向上存在几个方差较大的方向，PCA 的效果就大打折扣了。

3.12.9 参考资料

- <https://www.zhihu.com/question/20998460>
- PCA 教程 1
- PCA 文字化描述
- pca1
- ggplot2 axis
- scatterplot3D
- 稳健 PCA
- http://www.nlPCA.org/pca_principal_component_analysis.html
- Data centering

- Sample R markdown
- 矩阵特征值，对称矩阵的对角化
- Detail usage and visualization of prcomp result
- ggplot2 side by side plot

3.13 表达聚类分析

```
#
```

3.14 生存分析

生存分析指根据试验或调查得到的数据对生物或人的生存时间进行分析和推断，研究生存时间和结局与众多影响因素间关系及其程度大小的方法，也称生存率分析或存活率分析。常用于肿瘤等疾病的标志物筛选、疗效及预后的考核。

简单地说，比较两组或多组人群随着时间的延续，存活个体的比例变化趋势。活着的个体越少的组危险性越大，对应的基因对疾病影响越大，对应的药物治疗效果越差。

生存分析适合于处理时间-事件数据，如下

ID	Status	Days.survival	surv
166	0	2	2+
244	0	2	2+
202	1	7	7
135	1	13	13
190	1	17	17
247	1	19	19
127	1	26	26
220	0	28	28+
230	0	28	28+
203	1	29	29
212	1	30	30
261	1	33	33
114	1	35	35
248	1	35	35
196	1	37	37
213	1	41	41
237	1	41	41
266	1	47	47
199	1	49	49
241	1	50	50

生存时间数据有两种类型：

- 完全数据 (complete data) 指被观测对象从观察起点到出现终点事件所经历的时间; 一般用状态值 1 或 TRUE 表示。
- 截尾数据 (censored data) 或删失数据, 指在出现终点事件前, 被观测对象的观测过程终止了。由于被观测对象所提供的信息是不完全的, 只知道他们的生存事件超过了截尾时间。截尾主要由于失访、退出和终止产生。一般用状态值 0 或 FALSE 表示。
- TCGA 中的临床数据标记也符合这个规律, 在下面软件运行时也可修改状态值的含义, 但一般遵循这个规律。

生存概率 (survival probability) 指某段时间开始时存活的个体至该时间结束时仍然存活的可能性大小。

生存概率 = 某人群活过某段时间例数 / 该人群同时间段期初观察例数。

生存率 (Survival rate) , 用 $S(t)$ 表示 , 指经历 t 个单位时间后仍存活的概率 , 若无删失数据 , 则为活过了 t 时刻仍然存活的例数 / 观察开始的总例数。如果有删失数据 , 分母则需要按时段进行校正。

生存分析一个常用的方法是寿命表法。

寿命表是描述一段时间内生存状况、终点事件和生存概率的表格 , 需计算累积生存概率即每一步生存概率的乘积 (也可能是原始生存概率) , 可完成对病例随访资料在任意指定时点的生存状况评价。

生存分析数据共计238人

寿命表

ID	Status	Days.survival	surv	time	n.risk	n.event	survival
166	0		2 2+	2	238	0	1.0000000
244	0		2 2+	7	236	1	0.9957627
202	1		7 7	13	235	1	0.9915254
135	1		13 13	17	234	1	0.9872881
190	1		17 17	19	233	1	0.9830508
247	1		19 19	26	232	1	0.9788136
127	1		26 26	28	231	0	0.9788136
220	0		28 28+	29	229	1	0.9745393
230	0		28 28+	30	228	1	0.9702650
203	1		29 29	30	227	1	0.9659907
212	1		30 30	33	226	2	0.9574421
261	1		33 33	35	224	1	0.9531678
114	1		35 35	41	223	2	0.9446192
248	1		35 35	41	221	1	0.9403449
196	1		37 37	47	220	1	0.9360706
213	1		41 41	49	219	1	0.9317963
237	1		41 41	49	218	0	0.9317963
266	1		47 47	50	216	1	0.9274824
199	1		49 49				
241	1		50 50				

3.14.1 R 做生存分析

R 中做生存分析需要用到包 `survival` 和 `survminer`。输入数据至少两列 , 存活时间和生存状态 , 也就是测试数据中的 `Days.survival` 和 `vital_status` 列。如果需要比较不同组之间的差异 , 也需要提供个体的分组信息 , 如测试数据中的 `PAM50` 列。对应 TCGA 的数据 , 一般根据某个基因的表达量或突变有无对个体进行分组。

读入数据

```
library(survival)
BRCA <- read.table('data/BRCA.tsv', sep="\t", header=T)
head(BRCA)
```

```
##          ID SampleType PAM50Call_RNAseq Days.survival
## 1 TCGA-E9-A2JT-01 Tumor_type        LumA            288
## 2 TCGA-BH-A0W4-01 Tumor_type        LumA            759
## 3 TCGA-BH-A0B5-01 Tumor_type        LumA           2136
## 4 TCGA-AC-A3TM-01 Tumor_type      Unknown           762
## 5 TCGA-E9-A5FL-01 Tumor_type      Unknown            24
## 6 TCGA-AC-A3TN-01 Tumor_type      Unknown           456
##   pathologic_stage vital_status
## 1         stage iia          0
## 2         stage iia          0
## 3         stage iiia         0
## 4         stage iiia         0
## 5         stage iib          0
## 6         stage iib          0
```

简单地看下每一列都有什么内容，方便对数据整体有个了解，比如有无特殊值。

```
summary(BRCA)
```

```
##          ID SampleType PAM50Call_RNAseq
## 1 TCGA-3C-AAAU-01: 1 Tumor_type:1090 Basal    :138
## 2 TCGA-3C-AALI-01: 1                      Her2     : 65
## 3 TCGA-3C-AALJ-01: 1                      LumA    :415
## 4 TCGA-3C-AALK-01: 1                      LumB    :194
## 5 TCGA-4H-AAAK-01: 1                      Normal   : 24
## 6 TCGA-5L-AAT0-01: 1                      Unknown:254
## (Other)       :1084
##   Days.survival   pathologic_stage vital_status
##   Min.    : 0.0   stage iia :359   Min.    :0.0000
##   1st Qu.:450.2   stage iib :259   1st Qu.:0.0000
##   Median :848.0   stage iiia:156   Median :0.0000
##   Mean   :1247.0   stage i    : 90   Mean   :0.1394
##   3rd Qu.:1682.8   stage ia   : 85   3rd Qu.:0.0000
##   Max.   :8605.0   stage iiic: 67   Max.   :1.0000
##                   (Other)    : 74
```

计算寿命表

```
# Days.survival: 跟踪到的存活时间
# vital_status: 跟踪到的存活状态
# ~1 表示不进行分组
fit <- survfit(Surv(Days.survival, vital_status)~1, data=BRCA)

# 获得的 survival 列就是生存率
summary(fit)
```

```
## Call: survfit(formula = Surv(Days.survival, vital_status) ~ 1, data = BRCA)
##
##   time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    116    1021      1 0.999 0.000979      0.997    1.000
##    158    1017      1 0.998 0.001386      0.995    1.000
##    160    1016      1 0.997 0.001697      0.994    1.000
##    172    1010      1 0.996 0.001962      0.992    1.000
##    174    1008      1 0.995 0.002195      0.991    0.999
##    197    1003      1 0.994 0.002406      0.989    0.999
##    224     993      1 0.993 0.002604      0.988    0.998
##    227     990      1 0.992 0.002788      0.987    0.998
##    239     987      1 0.991 0.002961      0.985    0.997
##    255     981      1 0.990 0.003125      0.984    0.996
##    266     978      1 0.989 0.003282      0.983    0.996
##    295     965      1 0.988 0.003435      0.981    0.995
##    302     962      1 0.987 0.003581      0.980    0.994
##    304     958      1 0.986 0.003723      0.979    0.993
##    320     948      1 0.985 0.003862      0.977    0.993
##    322     946      1 0.984 0.003995      0.976    0.992
##    336     938      1 0.983 0.004127      0.975    0.991
##    348     929      1 0.982 0.004256      0.973    0.990
##    362     921      1 0.981 0.004382      0.972    0.989
##    365     918      1 0.980 0.004506      0.971    0.989
##    377     896      1 0.979 0.004632      0.970    0.988
##    385     887      2 0.976 0.004877      0.967    0.986
##    426     845      1 0.975 0.005006      0.965    0.985
##    446     826      1 0.974 0.005137      0.964    0.984
##    524     765      1 0.973 0.005286      0.962    0.983
##    538     751      1 0.971 0.005435      0.961    0.982
##    548     743      1 0.970 0.005583      0.959    0.981
##    558     732      1 0.969 0.005731      0.958    0.980
##    563     729      1 0.967 0.005875      0.956    0.979
##    571     723      1 0.966 0.006017      0.954    0.978
##    573     721      1 0.965 0.006156      0.953    0.977
##    584     708      1 0.963 0.006297      0.951    0.976
##    612     684      1 0.962 0.006443      0.949    0.975
##    614     680      1 0.961 0.006587      0.948    0.974
##    616     678      1 0.959 0.006728      0.946    0.972
```

CONTENTS

## 639	660	1	0.958	0.006873	0.944	0.971
## 678	632	1	0.956	0.007027	0.943	0.970
## 723	608	1	0.955	0.007189	0.941	0.969
## 749	592	1	0.953	0.007356	0.939	0.968
## 754	589	1	0.951	0.007519	0.937	0.966
## 763	577	1	0.950	0.007685	0.935	0.965
## 785	570	1	0.948	0.007850	0.933	0.964
## 786	568	1	0.946	0.008012	0.931	0.962
## 792	564	2	0.943	0.008327	0.927	0.960
## 811	557	1	0.941	0.008483	0.925	0.958
## 821	552	1	0.940	0.008637	0.923	0.957
## 825	550	1	0.938	0.008789	0.921	0.955
## 860	541	1	0.936	0.008942	0.919	0.954
## 879	533	1	0.934	0.009096	0.917	0.952
## 883	531	1	0.933	0.009248	0.915	0.951
## 904	526	1	0.931	0.009399	0.913	0.950
## 912	522	1	0.929	0.009548	0.911	0.948
## 921	517	1	0.927	0.009697	0.909	0.947
## 943	511	1	0.926	0.009847	0.906	0.945
## 959	505	1	0.924	0.009996	0.904	0.944
## 967	502	1	0.922	0.010144	0.902	0.942
## 976	494	1	0.920	0.010294	0.900	0.940
## 991	489	2	0.916	0.010590	0.896	0.937
## 1004	480	1	0.914	0.010739	0.894	0.936
## 1009	473	1	0.912	0.010889	0.891	0.934
## 1032	464	1	0.910	0.011042	0.889	0.932
## 1034	463	2	0.907	0.011339	0.885	0.929
## 1048	453	1	0.905	0.011489	0.882	0.927
## 1072	444	1	0.902	0.011642	0.880	0.926
## 1093	439	1	0.900	0.011796	0.878	0.924
## 1104	434	1	0.898	0.011950	0.875	0.922
## 1127	426	1	0.896	0.012106	0.873	0.920
## 1142	419	1	0.894	0.012265	0.870	0.918
## 1148	418	1	0.892	0.012421	0.868	0.917
## 1152	414	1	0.890	0.012576	0.866	0.915
## 1174	404	1	0.888	0.012736	0.863	0.913
## 1272	371	1	0.885	0.012925	0.860	0.911
## 1275	370	1	0.883	0.013109	0.858	0.909
## 1286	367	1	0.880	0.013293	0.855	0.907
## 1324	356	1	0.878	0.013483	0.852	0.905
## 1365	347	1	0.875	0.013680	0.849	0.903
## 1388	343	1	0.873	0.013876	0.846	0.900
## 1411	342	1	0.870	0.014068	0.843	0.898
## 1430	338	1	0.868	0.014260	0.840	0.896
## 1439	335	1	0.865	0.014451	0.837	0.894
## 1508	322	1	0.862	0.014654	0.834	0.892
## 1542	313	1	0.860	0.014864	0.831	0.889

CONTENTS

# #	1556	305	2	0.854	0.015291	0.825	0.885
# #	1563	302	1	0.851	0.015500	0.821	0.882
# #	1642	283	1	0.848	0.015735	0.818	0.880
# #	1649	279	1	0.845	0.015969	0.814	0.877
# #	1673	277	1	0.842	0.016200	0.811	0.874
# #	1688	271	1	0.839	0.016436	0.807	0.872
# #	1692	269	1	0.836	0.016668	0.804	0.869
# #	1694	267	1	0.833	0.016897	0.800	0.867
# #	1699	266	1	0.830	0.017121	0.797	0.864
# #	1759	260	1	0.826	0.017350	0.793	0.861
# #	1781	258	1	0.823	0.017576	0.790	0.858
# #	1793	256	1	0.820	0.017799	0.786	0.856
# #	1812	254	1	0.817	0.018020	0.782	0.853
# #	1884	241	1	0.813	0.018261	0.778	0.850
# #	1900	239	1	0.810	0.018499	0.775	0.847
# #	1920	236	1	0.807	0.018736	0.771	0.844
# #	1927	233	1	0.803	0.018973	0.767	0.841
# #	1993	225	1	0.800	0.019221	0.763	0.838
# #	2009	223	1	0.796	0.019467	0.759	0.835
# #	2097	212	1	0.792	0.019734	0.754	0.832
# #	2127	208	1	0.788	0.020003	0.750	0.829
# #	2192	195	1	0.784	0.020305	0.746	0.825
# #	2207	192	1	0.780	0.020606	0.741	0.822
# #	2273	181	2	0.772	0.021261	0.731	0.814
# #	2296	174	1	0.767	0.021596	0.726	0.811
# #	2348	168	1	0.763	0.021945	0.721	0.807
# #	2361	167	1	0.758	0.022284	0.716	0.803
# #	2373	161	1	0.753	0.022638	0.710	0.799
# #	2417	156	1	0.749	0.023002	0.705	0.795
# #	2469	152	1	0.744	0.023372	0.699	0.791
# #	2483	150	1	0.739	0.023736	0.694	0.787
# #	2520	144	1	0.734	0.024119	0.688	0.782
# #	2534	143	1	0.728	0.024490	0.682	0.778
# #	2551	140	1	0.723	0.024861	0.676	0.774
# #	2573	137	1	0.718	0.025234	0.670	0.769
# #	2636	127	1	0.712	0.025661	0.664	0.764
# #	2712	118	1	0.706	0.026144	0.657	0.759
# #	2763	115	1	0.700	0.026628	0.650	0.754
# #	2798	112	1	0.694	0.027114	0.643	0.749
# #	2854	109	1	0.687	0.027602	0.635	0.744
# #	2866	107	1	0.681	0.028082	0.628	0.738
# #	2911	105	1	0.675	0.028554	0.621	0.733
# #	2965	102	1	0.668	0.029030	0.613	0.727
# #	3063	86	1	0.660	0.029713	0.604	0.721
# #	3262	66	1	0.650	0.030901	0.592	0.714
# #	3409	54	1	0.638	0.032590	0.577	0.705
# #	3418	53	1	0.626	0.034127	0.563	0.697

# #	3461	50	1	0.614	0.035668	0.548	0.688
# #	3462	49	2	0.589	0.038357	0.518	0.669
# #	3472	47	1	0.576	0.039532	0.504	0.659
# #	3492	46	1	0.563	0.040608	0.489	0.649
# #	3669	40	1	0.549	0.041965	0.473	0.638
# #	3736	38	1	0.535	0.043280	0.457	0.627
# #	3873	36	1	0.520	0.044555	0.440	0.615
# #	3926	35	1	0.505	0.045693	0.423	0.603
# #	3941	34	1	0.490	0.046703	0.407	0.591
# #	3945	33	1	0.476	0.047593	0.391	0.579
# #	3959	31	1	0.460	0.048467	0.374	0.566
# #	4267	24	1	0.441	0.050097	0.353	0.551
# #	4456	19	1	0.418	0.052562	0.326	0.535
# #	6456	10	1	0.376	0.061715	0.273	0.519
# #	6593	9	1	0.334	0.067535	0.225	0.497
# #	7455	6	1	0.279	0.075850	0.163	0.475

绘制生存曲线，横轴表示生存时间，纵轴表示生存概率，为一条梯形下降的曲线。下降幅度越大，表示生存率越低或生存时间越短。

```
library(survminer)
```

```
## Loading required package: ggpubr
```

```
## Loading required package: magrittr
```

```
##
```

```
## Attaching package: 'ggpubr'
```

```
## The following object is masked from 'package:VennDiagram':
```

```
##
```

```
##      rotate
```

```
## The following object is masked from 'package:cowplot':
```

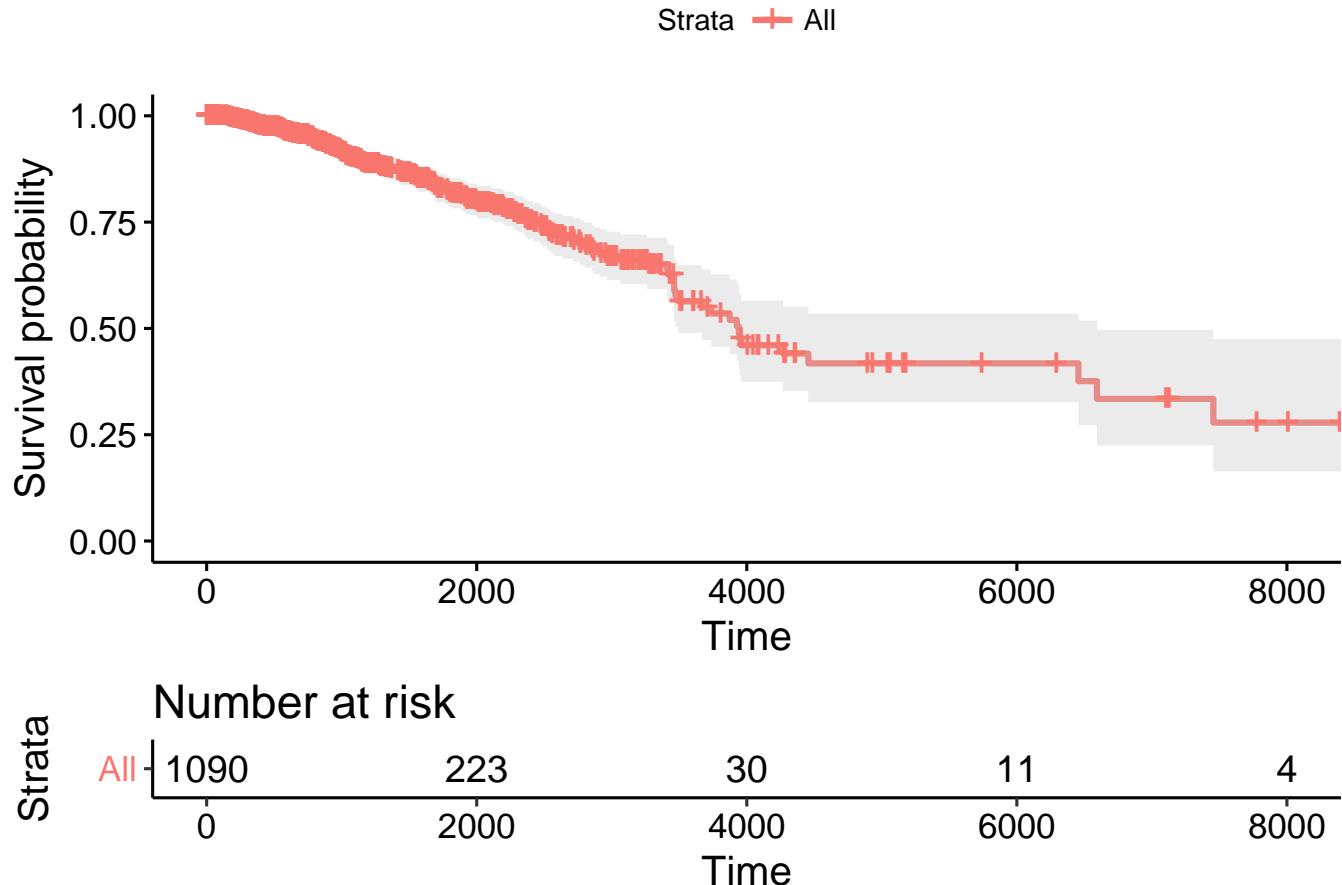
```
##
```

```
##      get_legend
```

```
# conf.int: 是否显示置信区间
```

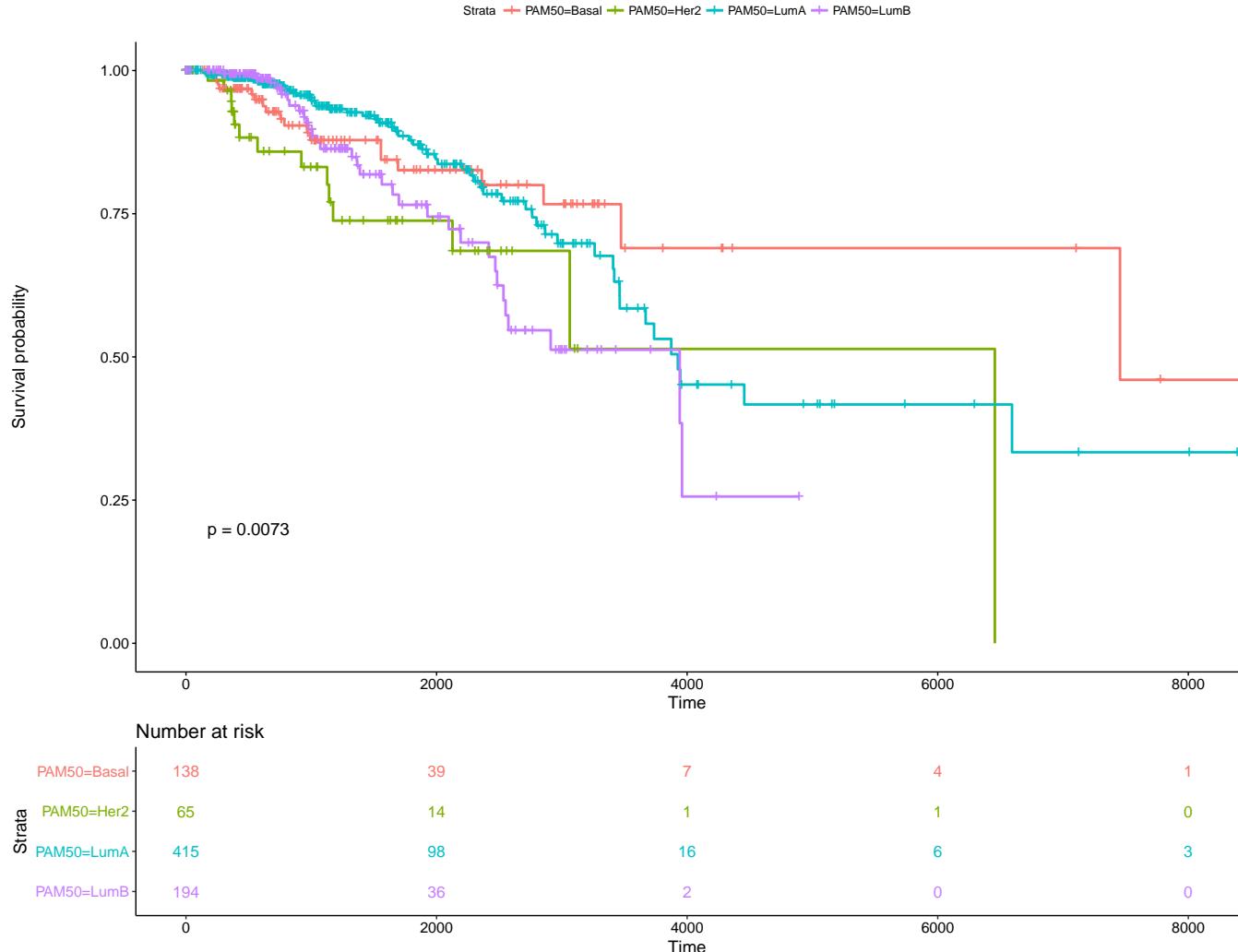
```
# risk.table: 对应时间存活个体总结表格
```

```
ggsurvplot(fit, conf.int=T,risk.table=T)
```



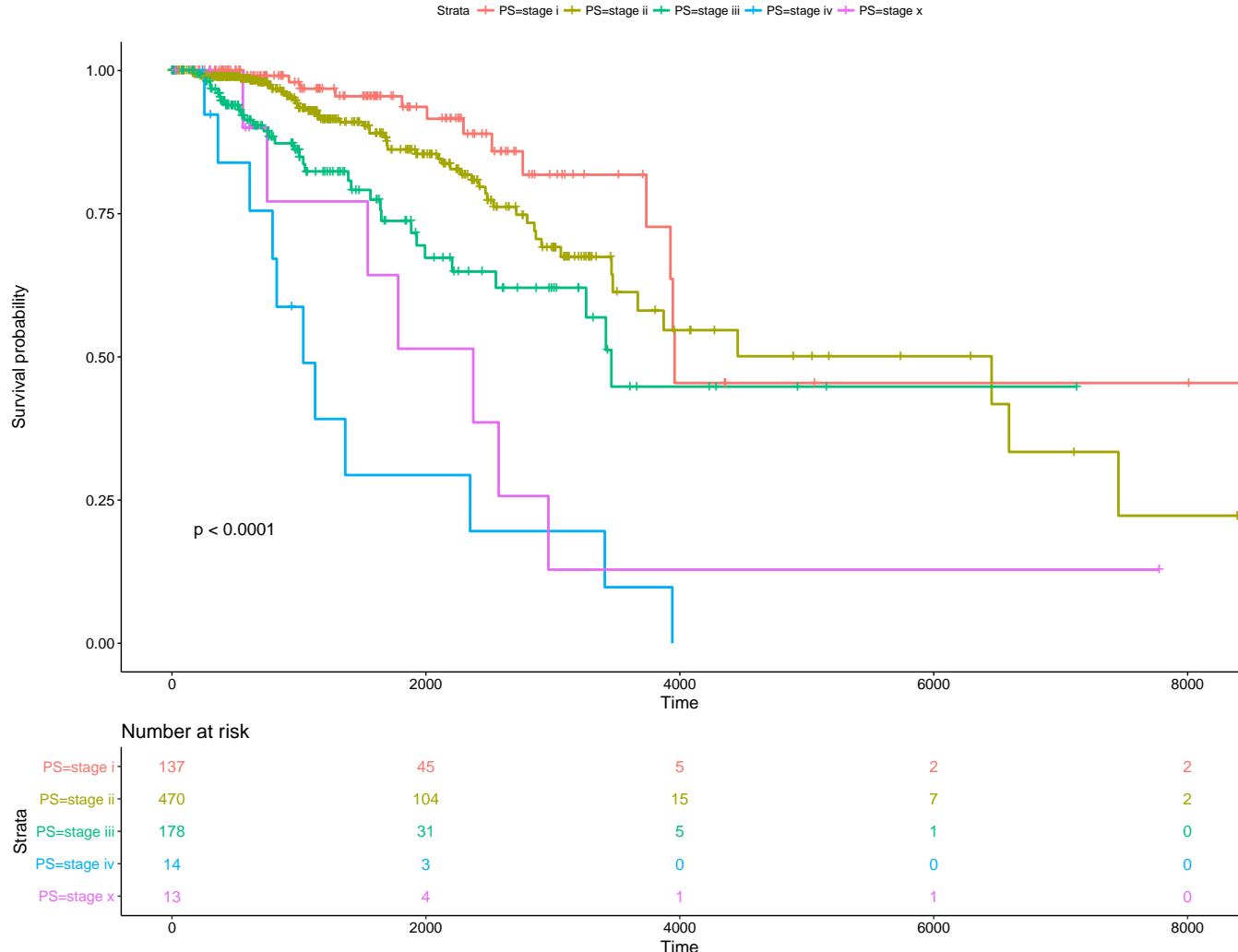
PAM50 是通过 50 个基因的表达量把乳腺癌分为四种类型 (Luminal A, Luminal B, HER2-enriched, and Basal-like) 作为预后的标志。根据 PAM50 属性对病人进行分组，评估比较两组之间生存率的差别。

```
# 这三步不是必须的，只是为了方便，选择其中的 4 个确定了的分组进行分析
# 同时为了简化图例，给列重命名一下，使得列名不那么长
BRCA_PAM50 <- BRCA[grep("Basal|Her2|LumA|LumB", BRCA$PAM50Call_RNAseq), ]
BRCA_PAM50 <- droplevels(BRCA_PAM50)
colnames(BRCA_PAM50)[colnames(BRCA_PAM50)=="PAM50Call_RNAseq"] <- 'PAM50'
# 按 PAM50 分组
fit <- survfit(Surv(Days.survival, vital_status)~PAM50, data=BRCA_PAM50)
# 绘制曲线
ggsurvplot(fit, conf.int=F, risk.table=T, risk.table.col="strata", pval=T)
```



简化 Stage 信息，先只查看大的阶段

```
BRCA_PAM50$pathologic_stage <- gsub('^(i+v*)\.*', "\\\1", BRCA_PAM50$pathologic_stage)
BRCA_PAM50$pathologic_stage <- as.factor(BRCA_PAM50$pathologic_stage)
colnames(BRCA_PAM50)[colnames(BRCA_PAM50)=="pathologic_stage"] <- 'PS'
fit <- survfit(Surv(Days.survival, vital_status)~PS, data=BRCA_PAM50)
# 绘制曲线
ggsurvplot(fit, conf.int=F, risk.table=T, risk.table.col="strata", pval=T)
```



3.15 一步作图的优势

一步作图相比于直接写 R 代码有什么好处呢？

1. 模块化好。数据处理和可视化分开；一步作图，只是作图，不做无关的处理，更随意。
2. 易用性强。敲代码，总不如给改数来得快。
3. 重用性强。假如我要做十几个分开的基因集合呢？一段段复制代码？改错了或忘记改某个地方了怎么办？
4. 快速出图。先快速出个原型，再接着调整。

获取地址 <https://github.com/Tong-Chen/s-plot>.

在线绘图 <http://www.ehbio.com/ImageGP/>

CONTENTS

s-plot 可以绘制的图的类型还有一些，列举如下；

Usage:

```
s-plot options
```

Function:

This software is designed to simply the process of plotting and help researchers focus more on data rather than technology.

Currently, the following types of plot are supported.

Bars

```
s-plot barPlot
```

Lines

```
s-plot lines
```

Dots

```
s-plot pca
s-plot scatterplot3d
s-plot scatterplot2
s-plot scatterplotColor
s-plot scatterplotContour
s-plot scatterplotLotsData
s-plot scatterplotMatrix
s-plot scatterplotDoubleVariable
s-plot contourPlot
s-plot density2d
```

Distribution

```
s-plot areaplot
s-plot boxplot
s-plot densityHistPlot
```

Cluster

```
s-plot hcluster_gg (latest)
```

Heatmap

```
s-plot heatmaps
s-plot heatmapM
s-plot heatmap.2
s-plot pheatmap
s-plot prettyHeatmap
```

Others

```
s-plot volcano
s-plot vennDiagram
s-plot upsetView
```

3.16 不改脚本的热图绘制

绘图时通常会碰到两个头疼的问题：1. 需要画很多的图，唯一的不同就是输出文件，其它都不需要修改。如果用 R 脚本，需要反复替换文件名，繁琐又容易出错。

- 每次绘图都需要不断的调整参数，时间久了不用，就忘记参数放哪了；或者调整次数过多，有了很多版本，最后不知道用哪个了。

为了简化绘图、维持脚本的一致，我用 bash 对 R 做了一个封装，然后就可以通过修改命令好参数绘制不同的图了。

先看一看怎么使用

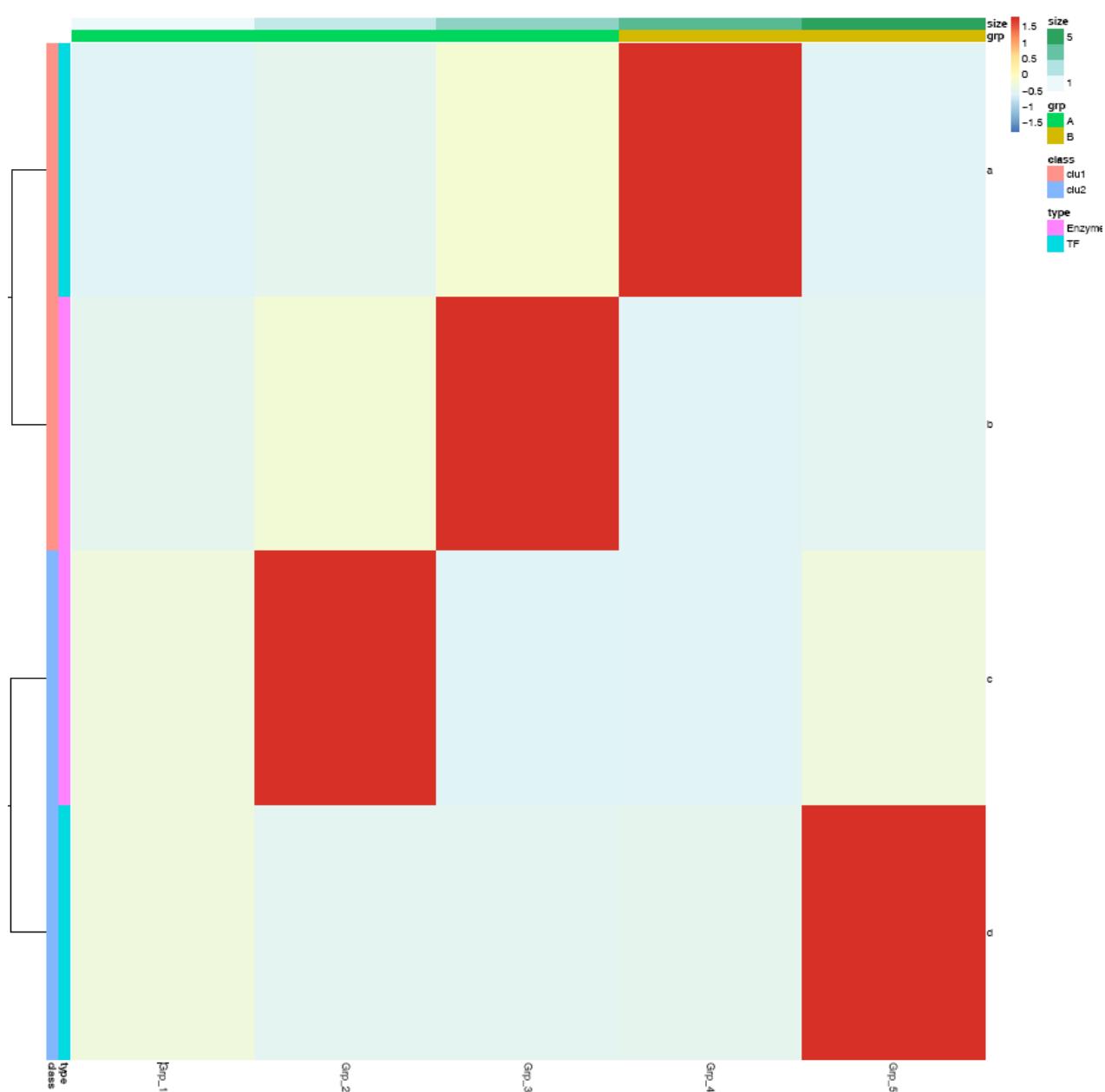
首先把测试数据存储到文件中方便调用。数据矩阵存储在 heatmap_data.xls 文件中；行注释存储在 heatmap_row_anno.xls 文件中；列注释存储在 heatmap_col_anno.xls 文件中。

```
# tab 键分割，每列不加引号
write.table(data, file="heatmap_data.xls", sep="\t", row.names=T, col.names=T, quote=F)
# 如果看着第一行少了 ID 列不爽，可以填补下
system("sed -i '1 s/^/ID\t/' heatmap_data.xls")

write.table(row_anno, file="heatmap_row_anno.xls", sep="\t", row.names=T,
            col.names=T, quote=F)
write.table(col_anno, file="heatmap_col_anno.xls", sep="\t", row.names=T,
            col.names=T, quote=F)
```

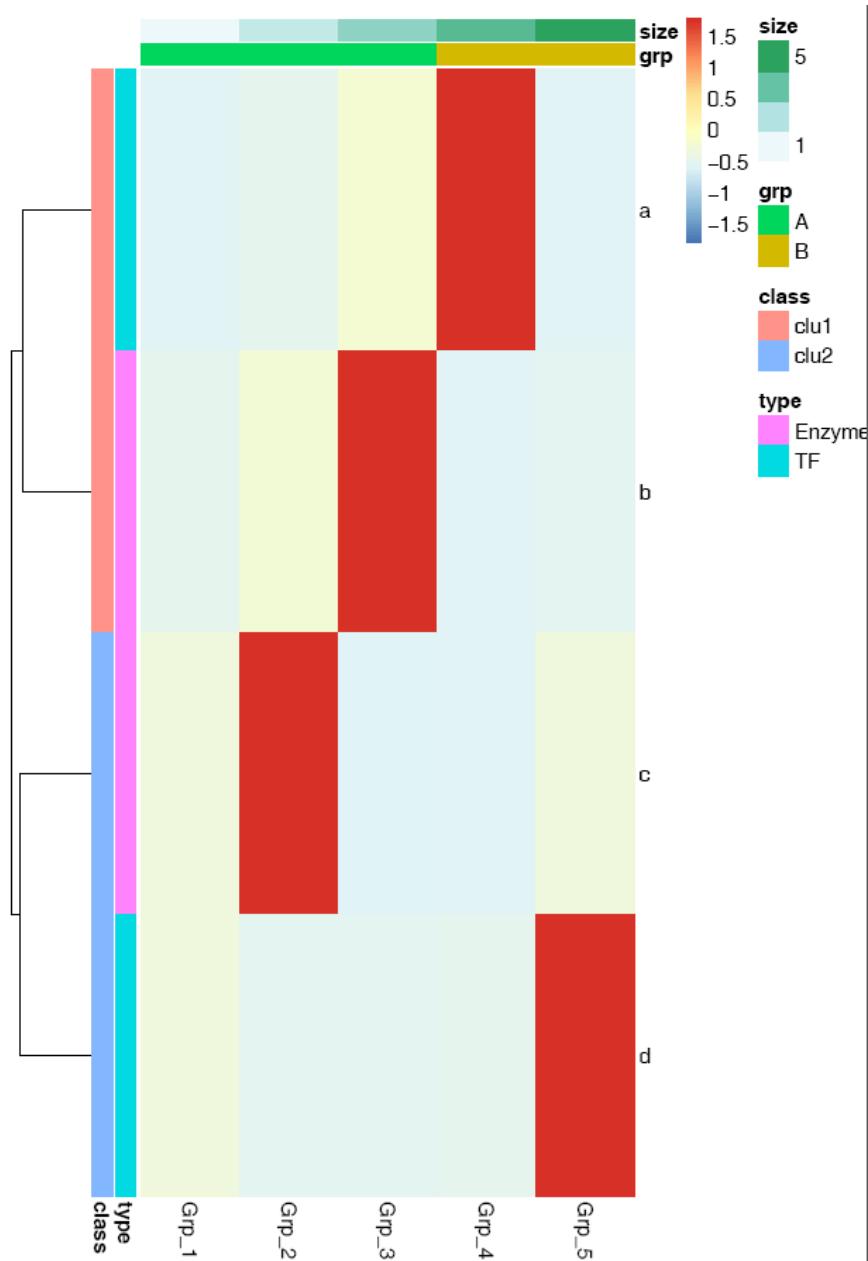
然后用程序 sp_pheatmap.sh 绘图。

```
# -f: 指定输入的矩阵文件
# -d: 指定是否计算 Z-score, <none> (否), <row> (按行算), <col> (按列算)
# -P: 行注释文件
# -Q: 列注释文件
ct@ehbio:~/ $ sp_pheatmap.sh -f heatmap_data.xls -d row -P heatmap_row_anno.xls \
-Q heatmap_col_anno.xls
```



字有点小，是因为图太大了，把图的宽和高缩小下试试。

```
# -f: 指定输入的矩阵文件
# -d: 指定是否计算 Z-score, <none> (否), <row> (按行算), <col> (按列算)
# -P: 行注释文件
# -Q: 列注释文件
# -u: 设置宽度, 单位是 inch
# -v: 设置高度, 单位是 inch
ct@ehbio:~/ $ sp_pheatmap.sh -f heatmap_data.xls -d row -P heatmap_row_anno.xls \
-Q heatmap_col_anno.xls -u 8 -v 12
```



横轴的标记水平放置

```
# -A: 0, X 轴标签选择 0 度
# -C: 自定义颜色，注意引号的使用，最外层引号与内层引号不同，引号之间无交叉
# -T: 指定给定的颜色的类型；如果给的是 vector (如下面的例子)，则-T 需要指定为 vector;
#       否则结果会很怪异，只有俩颜色。
# -t: 指定图形的题目，注意引号的使用；参数中包含空格或特殊字符等都要用引号包起来作为一个整体。
ct@ehbio:~/\$ sp_pheatmap.sh -f heatmap_data.xls -d row -P heatmap_row_anno.xls \
-Q heatmap_col_anno.xls -u 8 -v 12 -A 0 -C 'c("white", "blue")' -T vector \
-t "Heatmap of gene expression profile"
```

CONTENTS

sp_pheatmap.sh 的参数还有一些，可以完成前面讲述过的所有热图的绘制，具体如下：

```
***CREATED BY Chen Tong (chentong_biology@163.com) ***
```

----Matrix file-----

Name	T0_1	T0_2	T0_3	T4_1	T4_2
TR19267 c0_g1 CYP703A2	1.431	0.77	1.309	1.247	0.485
TR19612 c1_g3 CYP707A1	0.72	0.161	0.301	2.457	2.794
TR60337 c4_g9 CYP707A1	0.056	0.09	0.038	7.643	15.379
TR19612 c0_g1 CYP707A3	2.011	0.689	1.29	0	0
TR35761 c0_g1 CYP707A4	1.946	1.575	1.892	1.019	0.999
TR58054 c0_g2 CYP707A4	12.338	10.016	9.387	0.782	0.563
TR14082 c7_g4 CYP707A4	10.505	8.709	7.212	4.395	6.103
TR60509 c0_g1 CYP707A7	3.527	3.348	2.128	3.257	2.338
TR26914 c0_g1 CYP710A1	1.899	1.54	0.998	0.255	0.427

----Matrix file-----

----Row annorarion file -----

-----1. At least two columns-----

-----2. The first column should be the same as the first column in
matrix (order does not matter)-----

Name Clan Family

TR19267 c0_g1 CYP703A2	CYP71	CYP703
TR19612 c1_g3 CYP707A1	CYP85	CYP707
TR60337 c4_g9 CYP707A1	CYP85	CYP707
TR19612 c0_g1 CYP707A3	CYP85	CYP707
TR35761 c0_g1 CYP707A4	CYP85	CYP707
TR58054 c0_g2 CYP707A4	CYP85	CYP707
TR14082 c7_g4 CYP707A4	CYP85	CYP707
TR60509 c0_g1 CYP707A7	CYP85	CYP707
TR26914 c0_g1 CYP710A1	CYP710	CYP710

----Row annorarion file -----

----Column annorarion file -----

-----1. At least two columns-----

-----2. The first column should be the same as the first row in
matrix (order does not matter)-----

Name Sample

T0_1	T0
T0_2	T0
T0_3	T0
T4_1	T4
T4_2	T4

----Column annorarion file -----

Usage:

sp_pheatmap.sh options**Function:**

This script is used to do heatmap using package pheatmap.

The parameters for logical variable are either TRUE or FALSE.

OPTIONS:

- f Data file (with header line, the first column is the **rownames**, tab separated. Colnames must be unique unless you **know** what you are doing.) [**NECESSARY**]
- t Title of picture [Default empty title]
["Heatmap of gene expression profile"]
- a Display xtics. [Default TRUE]
- A Rotation angle for x-axis value (anti clockwise)
[**Default** 90]
- b Display ytics. [Default TRUE]
- H Hieratical cluster for columns.
Default FALSE, accept TRUE
- R Hieratical cluster for rows.
Default TRUE, accept FALSE
- c Clustering method, Default "complete".
Accept "ward.D", "ward.D2", "single", "average" (=UPGMA), "mcquitty" (=WPGMA), "median" (=WPGMC) or "centroid" (=UPGMC)
- C Color vector.
Default pheatmap_default.
Accept a vector containing multiple colors such as
<'c("white", "blue")'> will be transferred
to <colorRampPalette(c("white", "blue"), bias=1) (30)>
or an R function
<colorRampPalette(rev(brewer.pal(n=7, name="RdYlBu")))> (100)>
generating a list of colors.
- T Color type, a vector which will be transferred as described in <-C> [vector] or a raw vector [direct vector] or a function [function (default)].
- B A positive number. Default 1. Values larger than 1 will give more color for high end. Values between 0-1 will give more color for low end.
- D Clustering distance method for rows.
Default 'correlation', accept 'euclidean', "manhattan", "maximum", "canberra", "binary", "minkowski".
- I Clustering distance method for cols.
Default 'correlation', accept 'euclidean', "manhattan", "maximum", "canberra", "binary", "minkowski".
- L First get log-value, then do other analysis.
Accept an R function log2 or log10.

```

[Default FALSE]
-d Scale the data or not for clustering and visualization.
[Default 'none' means no scale, accept 'row', 'column' to
scale by row or column.]
-m The maximum value you want to keep, any number larger will
be taken as this given maximum value.
[Default Inf, Optional]
-s The smallest value you want to keep, any number smaller will
be taken as this given minimum value.
[Default -Inf, Optional]
-k Aggregate the rows using kmeans clustering.
This is advisable if number of rows is so big that R cannot
handle their hierarchical clustering anymore, roughly more than 1000.
Instead of showing all the rows separately one can cluster the
rows in advance and show only the cluster centers. The number
of clusters can be tuned here.
[Default 'NA' which means no
cluster, other positive integer is accepted for executing
kmeans cluster, also the parameter represents the number of
expected clusters.]
-P A file to specify row-annotation with format described above.
[Default NA]
-Q A file to specify col-annotation with format described above.
[Default NA]
-u The width of output picture.[Default 20]
-v The height of output picture.[Default 20]
-E The type of output figures.[Default pdf, accept
eps/ps, tex (pictex), png, jpeg, tiff, bmp, svg and wmf)]
-r The resolution of output picture.[Default 300 ppi]
-F Font size [Default 14]
-p Preprocess data matrix to avoid 'STDERR 0 in cor(t(mat))'.
Lowercase <p>.
[Default TRUE]
-e Execute script (Default) or just output the script.
[Default TRUE]
-i Install the required packages. Normally should be TRUE if this is
your first time run s-plot.[Default FALSE]

```

3.16.1 箱线图 - 一步绘制

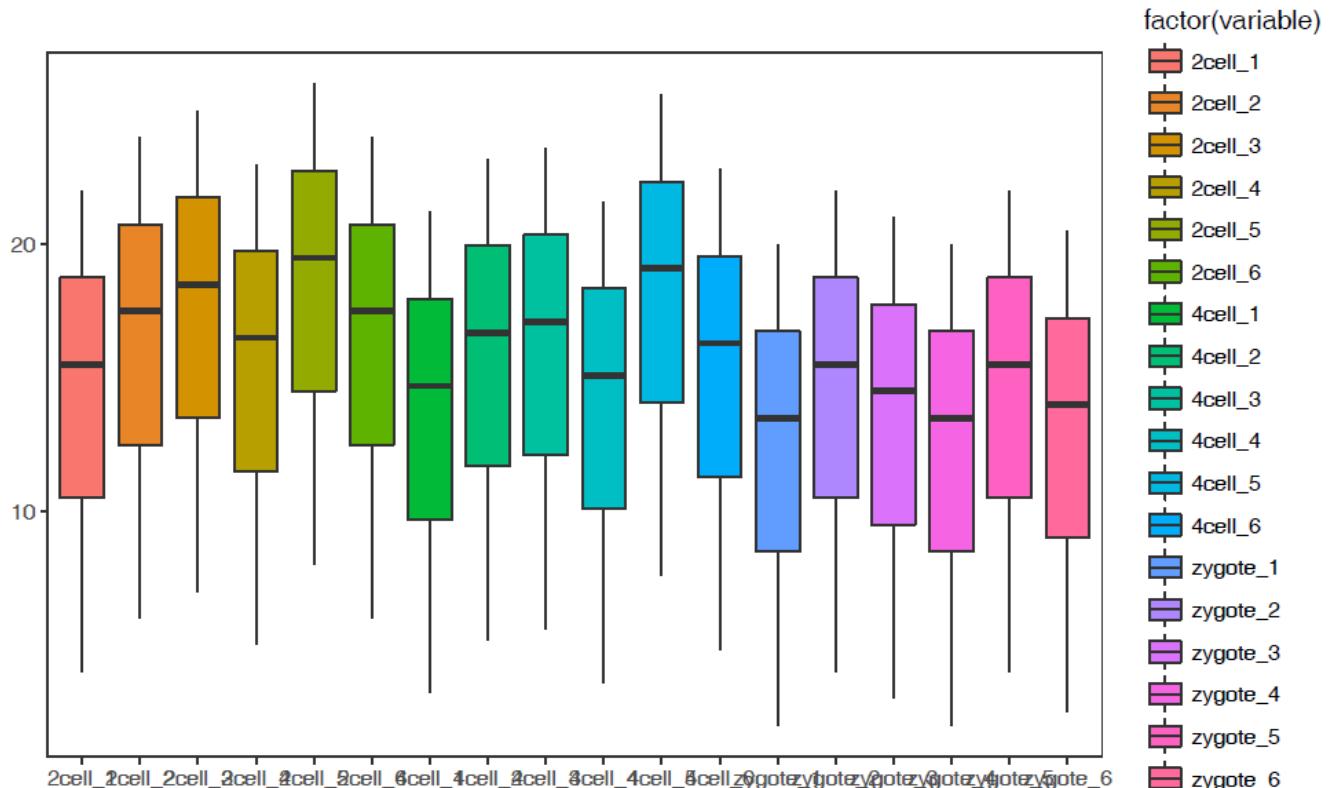
首先把测试数据存储到文件中方便调用。数据矩阵存储在 `boxplot.normal.data`、`sampleGroup` 和 `boxplot.melt.data` 文件中 (TAB 键分割，内容在文档最后。如果你手上有自己的数据，也可以拿来用)。

使用正常矩阵默认参数绘制箱线图

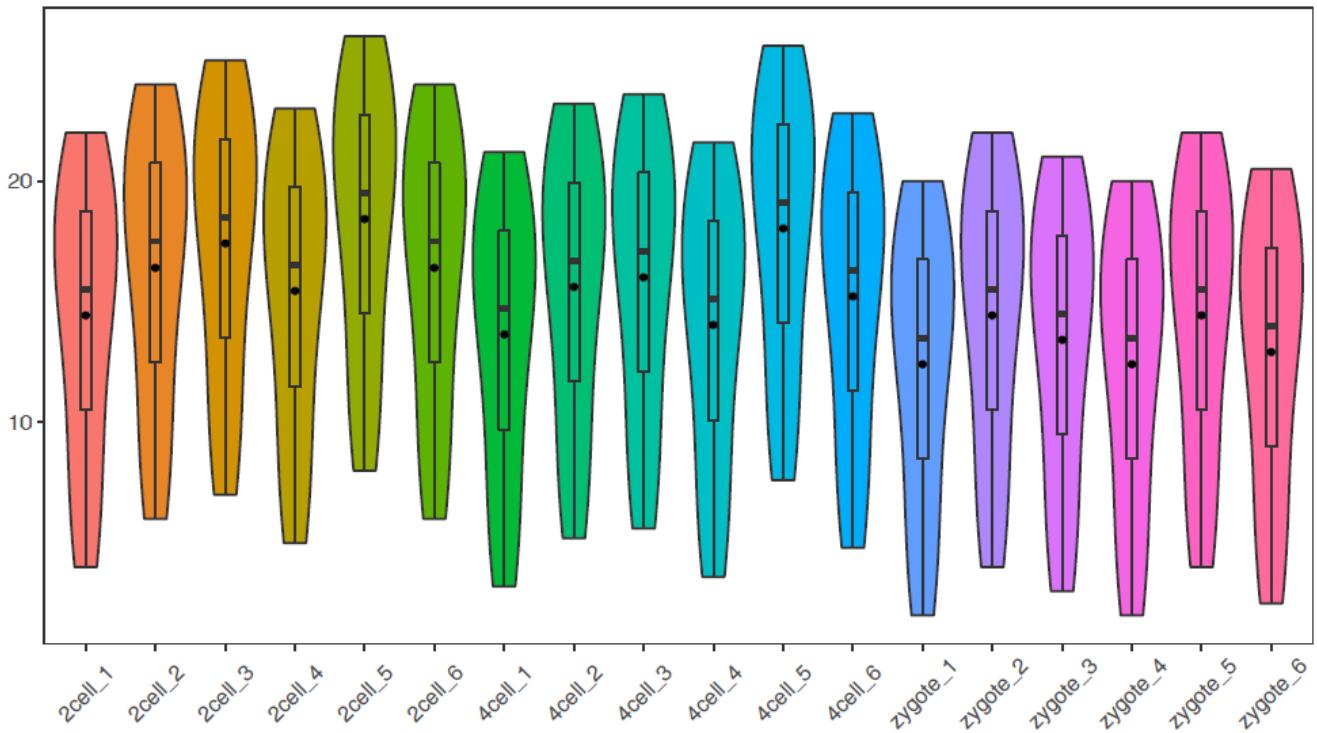
CONTENTS

```
# -f: 指定输入的矩阵文件，第一列为行名字，第一行为header
    列数不限，列名字不限；行数不限，行名字默认为文本
sp_boxplot.sh -f boxplot.normal.data
```

箱线图出来了，但有点小乱。



```
# -f: 指定输入的矩阵文件，第一列为行名字，第一行为header
    列数不限，列名字不限；行数不限，行名字默认为文本
# -P: none, 去掉legend (uppercase P)
# -b: X-axis旋转45度
# -V: TRUE 绘制小提琴图
sp_boxplot.sh -f boxplot.normal.data -P none -b 45 -V TRUE
```

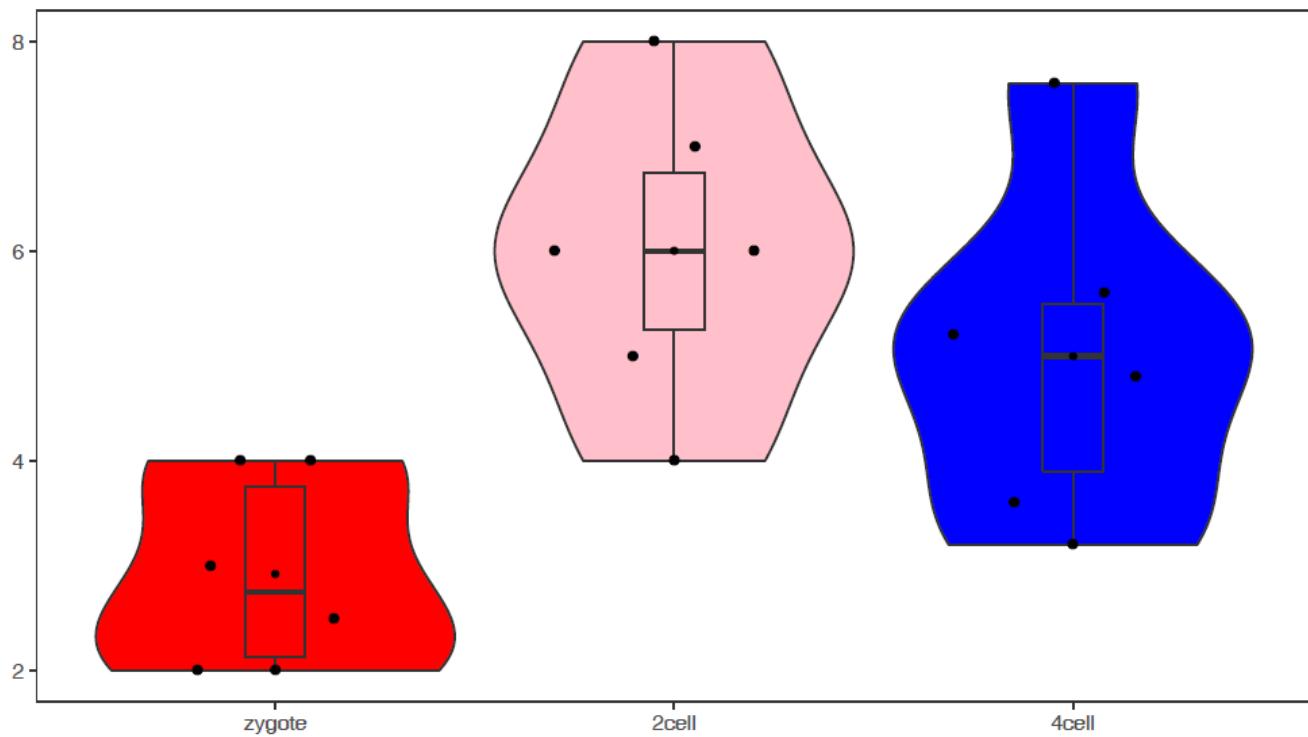


绘制单个基因的小提琴图加抖动图

```

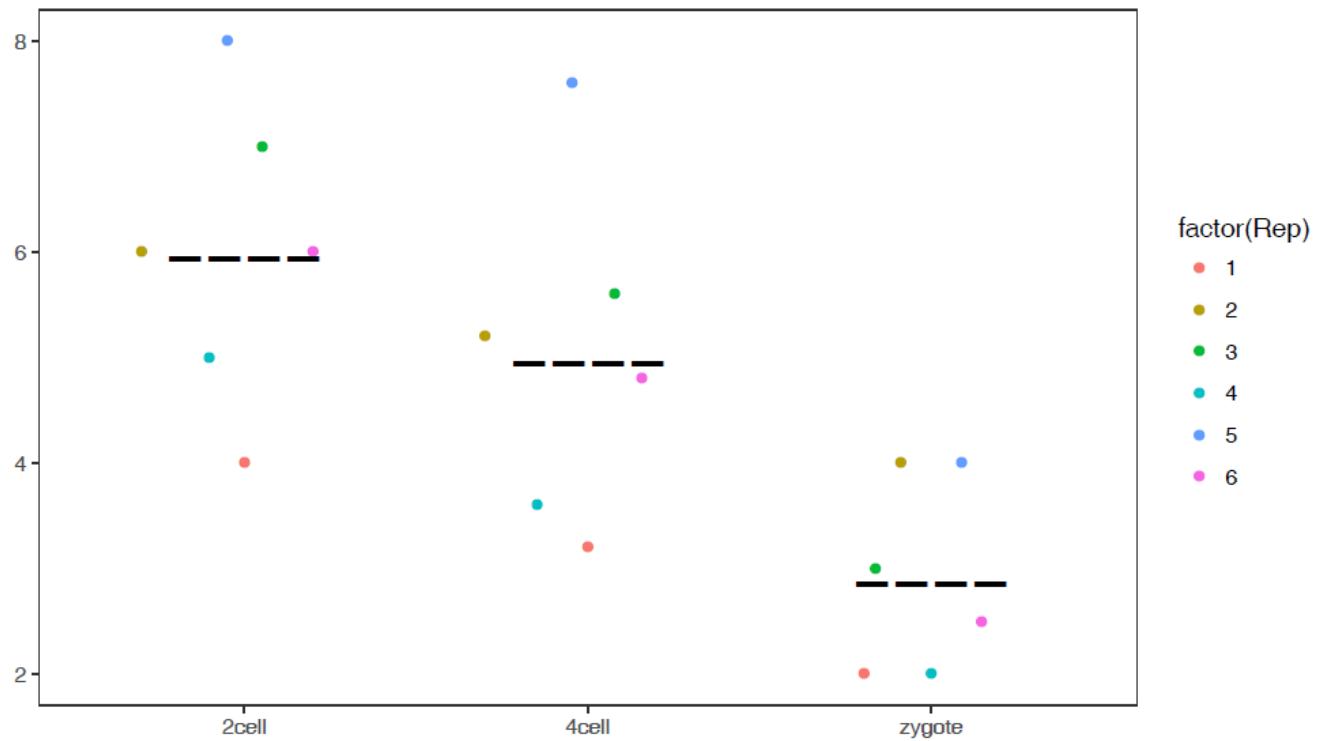
# -q: 指定某一行的名字，此处为基因名，绘制基因A的表达图谱
# -Q: 指定样本分组，绘制基因A在不同样品组的表达趋势
# -F Group: sampleGroup中第二列的名字，指代分组信息，根据需要修改
# -J TRUE: 绘制抖动图 jitter plot
# -L: 设置x轴样品组顺序
# -c TRUE -C "'red', 'pink', 'blue)": 指定每个箱线图的颜色
sp_boxplot.sh -f boxplot.normal.data -q A -Q sampleGroup -F Group -V TRUE -J TRUE \
-L "'zygote','2cell','4cell'" -c TRUE -C "'red', 'pink', 'blue'" -P none

```



使用 melted 矩阵默认参数绘箱线图

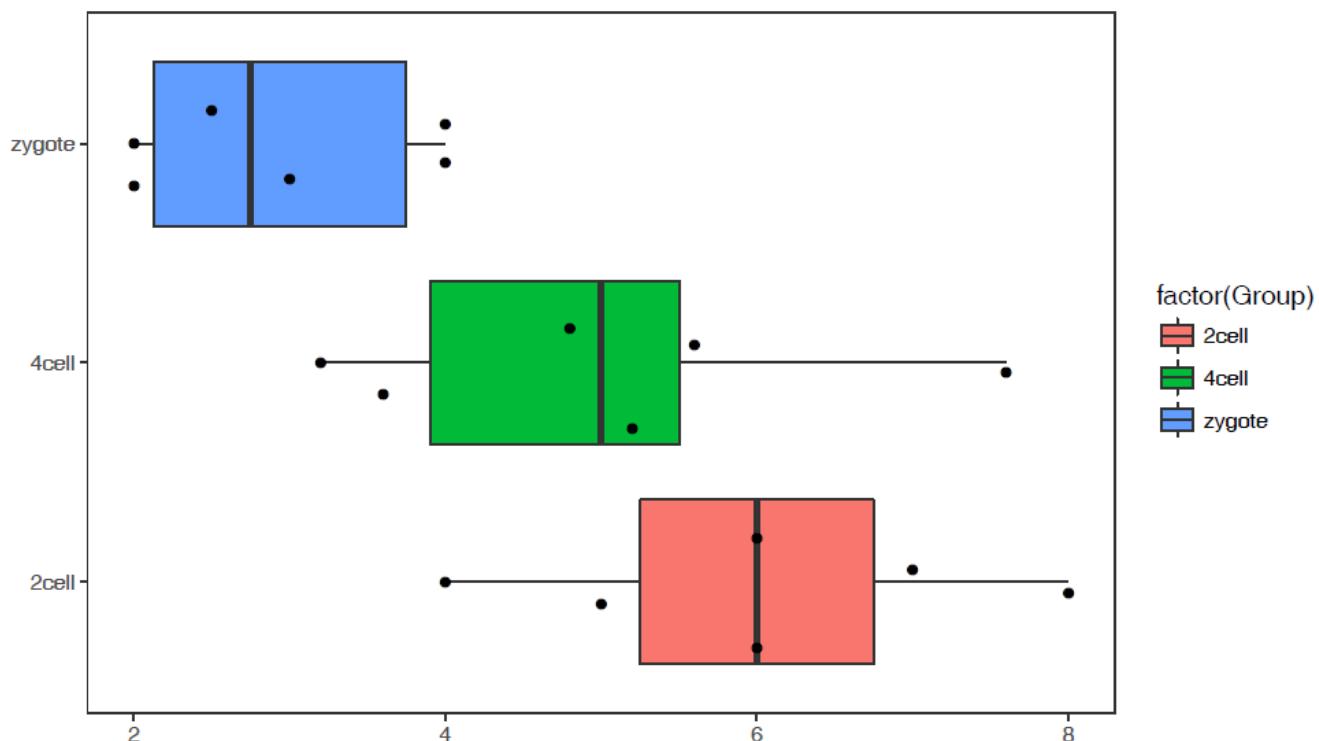
```
# -f: 指定输入文件
# -m TRUE: 指定输入的矩阵为melted format
# -d Expr : 指定表达值所在的列
# -F Rep: 指定子类所在列，也就是legend
# -a Group : 指定x轴分组信息
# -j TRUE: jitter plot
sp_boxplot.sh -f boxplot.melt.data -m TRUE -d Expr -F Rep -a Group -j TRUE
```



```
# 如果没有子类，则-a和-F指定为同一值
```

```
# -R TRUE: 旋转boxplot
```

```
sp_boxplot.sh -f boxplot.melt.data -m TRUE -d Expr -a Group -F Group -J TRUE -R TRUE
```



CONTENTS

参数中最需要注意的是引号的使用：

- 外层引号与内层引号不能相同
- 凡参数值中包括了空格，括号，逗号等都用引号括起来作为一个整体。

```
#boxplot.normal.data
Name      2cell_1 2cell_2 2cell_3 2cell_4 2cell_5 2cell_6 4cell_1 4cell_2 4cell_3 4cell_4 4cell_5
A     4       6    7    5    8    6    3.2   5.2   5.6   3.6   7.6   4.8   2     4     3     2     4     2.5
B     6       8    9    7   10    8    5.2   7.2   7.6   5.6   9.6   6.8   4     6     5     4     6     4.5
C     8      10   11   9   12   10    7.2   9.2   9.6   7.6  11.6   8.8   6     8     7     6     8     6.5
D    10      12   13   11   14   12    9.2  11.2    11.6    9.6  13.6  10.8   8     10    9     8     10    8.5
E    12      14   15   13   16   14   11.2   13.2   13.6   11.6  15.6  12.8   10    12    11    10    12
F    14      16   17   15   18   16   13.2   15.2   15.6   13.6  17.6  14.8   12    14    13    12    14
G    15      17   18   16   19   17   14.2   16.2   16.6   14.6  18.6  15.8   13    15    14    13    15
H    16      18   19   17   20   18   15.2   17.2   17.6   15.6  19.6  16.8   14    16    15    14    16
I    17      19   20   18   21   19   16.2   18.2   18.6   16.6  20.6  17.8   15    17    16    15    17
J    18      20   21   19   22   20   17.2   19.2   19.6   17.6  21.6  18.8   16    18    17    16    18
L    19      21   22   20   23   21   18.2   20.2   20.6   18.6  22.6  19.8   17    19    18    17    19
M    20      22   23   21   24   22   19.2   21.2   21.6   19.6  23.6  20.8   18    20    19    18    20
N    21      23   24   22   25   23   20.2   22.2   22.6   20.6  24.6  21.8   19    21    20    19    21
O    22      24   25   23   26   24   21.2   23.2   23.6   21.6  25.6  22.8   20    22    21    20    22
```

```
#boxplot.melt.data
```

Gene	Sample	Group	Expr	Rep
A	zygote_1	zygote	2	1
A	zygote_2	zygote	4	2
A	zygote_3	zygote	3	3
A	zygote_4	zygote	2	4
A	zygote_5	zygote	4	5
A	zygote_6	zygote	2.5	6
A	2cell_1	2cell	4	1
A	2cell_2	2cell	6	2
A	2cell_3	2cell	7	3
A	2cell_4	2cell	5	4
A	2cell_5	2cell	8	5
A	2cell_6	2cell	6	6
A	4cell_1	4cell	3.2	1
A	4cell_2	4cell	5.2	2
A	4cell_3	4cell	5.6	3
A	4cell_4	4cell	3.6	4
A	4cell_5	4cell	7.6	5
A	4cell_6	4cell	4.8	6

```
#sampleGroup
Sample Group
zygote_1      zygote
zygote_2      zygote
zygote_3      zygote
zygote_4      zygote
zygote_5      zygote
zygote_6      zygote
2cell_1       2cell
2cell_2       2cell
2cell_3       2cell
2cell_4       2cell
2cell_5       2cell
2cell_6       2cell
4cell_1       4cell
4cell_2       4cell
4cell_3       4cell
4cell_4       4cell
4cell_5       4cell
4cell_6       4cell
```

3.16.2 线图 - 一步绘制

首先把测试数据存储到文件中方便调用。数据矩阵存储在 `line_data.xls` 和 `line_data_melt.xls` 文件中 (直接拷贝到文件中也可以，这里这么操作只是为了随文章提供个测试文件，方便使用。如果你手上有自己的数据，也可以拿来用)。

```
profile = "Pos;H3K27ac;CTCF;Enhancer;H3K4me3;polII
-5000;8.7;10.7;11.7;10;8.3
-4000;8.4;10.8;11.8;9.8;7.8
-3000;8.3;10.5;12.2;9.4;7
-2000;7.2;10.9;12.7;8.4;4.8
-1000;3.6;8.5;12.8;4.8;1.3
0;3.6;8.5;13.4;5.2;1.5
1000;7.1;10.9;12.4;8.1;4.9
2000;8.2;10.7;12.4;9.5;7.7
3000;8.4;10.4;12;9.8;7.9
4000;8.5;10.6;11.7;9.7;8.2
5000;8.5;10.6;11.7;10;8.2"

profile_text <- read.table(text=profile, header=T, row.names=1, quote="", sep=";")
# tab 键分割，每列不加引号
write.table(profile_text, file="line_data.xls", sep="\t", row.names=T, col.names=T, quote=F)
# 如果看着第一行少了 ID 列不爽，可以填补下
#system("sed -i '1 s/^/ID\t/' line_data.xls")
```

CONTENTS

```

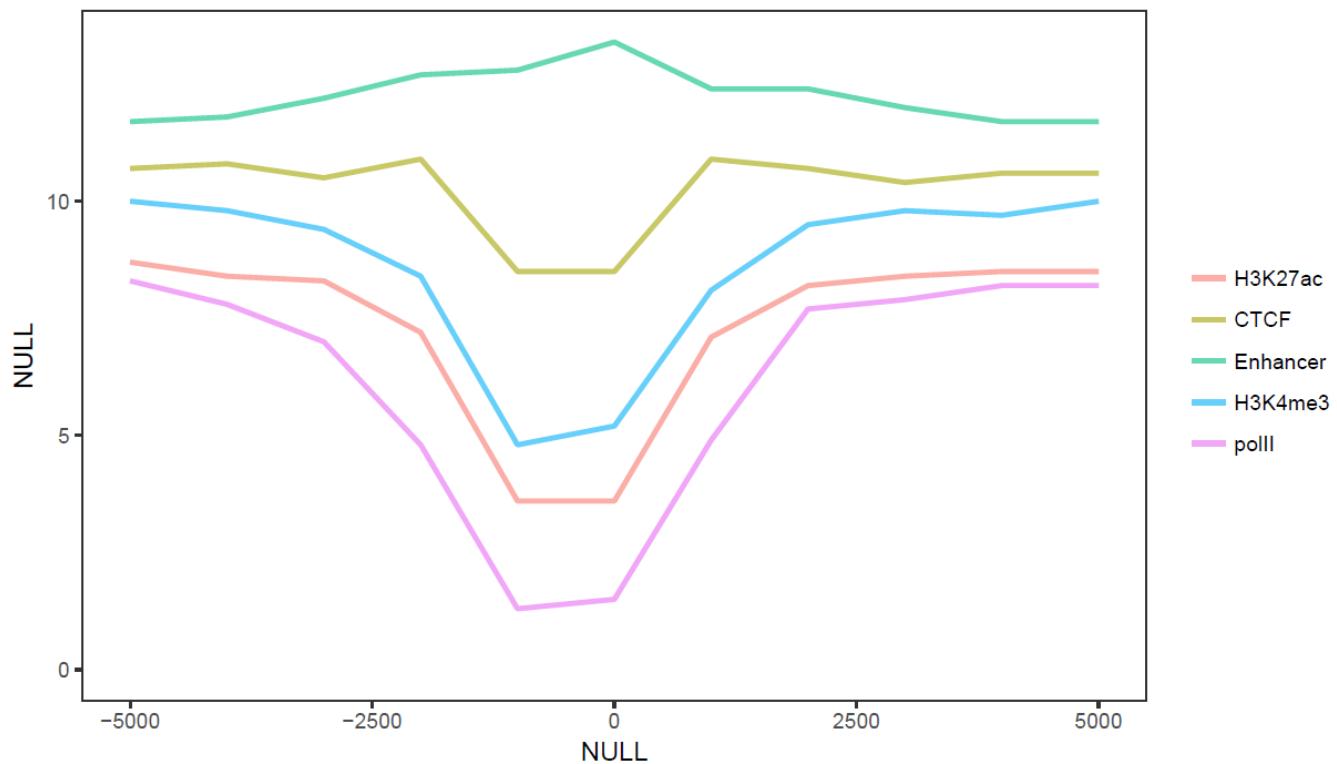
profile = "Pos;variable;value;set
-5000;H3K27ac;8.71298;A
-4000;H3K27ac;8.43246;A
-3000;H3K27ac;8.25497;A
-2000;H3K27ac;7.16265;A
-1000;H3K27ac;3.55341;A
0;H3K27ac;3.5503;A
1000;H3K27ac;7.07502;A
2000;H3K27ac;8.24328;A
3000;H3K27ac;8.43869;A
4000;H3K27ac;8.48877;A
-5000;CTCF;10.6913;A
-4000;CTCF;10.7668;A
-3000;CTCF;10.5441;A
-2000;CTCF;10.8635;A
-1000;CTCF;8.45751;A
0;CTCF;8.50316;A
1000;CTCF;10.9143;A
2000;CTCF;10.7022;A
3000;CTCF;10.4101;A
4000;CTCF;10.5757;A
-5000;H3K27ac;8.71298;B
-4000;H3K27ac;8.43246;B
-3000;H3K27ac;8.25497;B
-2000;H3K27ac;7.16265;B
-1000;H3K27ac;3.55341;B
0;H3K27ac;3.5503;B
1000;H3K27ac;7.07502;B
2000;H3K27ac;8.24328;B
3000;H3K27ac;8.43869;B
4000;H3K27ac;8.48877;B
-5000;CTCF;10.6913;B
-4000;CTCF;10.7668;B
-3000;CTCF;10.5441;B
-2000;CTCF;10.8635;B
-1000;CTCF;8.45751;B
0;CTCF;8.50316;B
1000;CTCF;10.9143;B
2000;CTCF;10.7022;B
3000;CTCF;10.4101;B
4000;CTCF;10.5757;B"

profile_text <- read.table(text=profile, header=T, quote="", sep="; ")
# tab 键分割，每列不加引号
write.table(profile_text, file="line_data_melt.xls", sep="\t", row.names=T, col.names=T, quote="")
# 如果看着第一行少了 ID 列不爽，可以填补下
# system("sed -i '1 s/^/ID\t/' line_data_melt.xls")

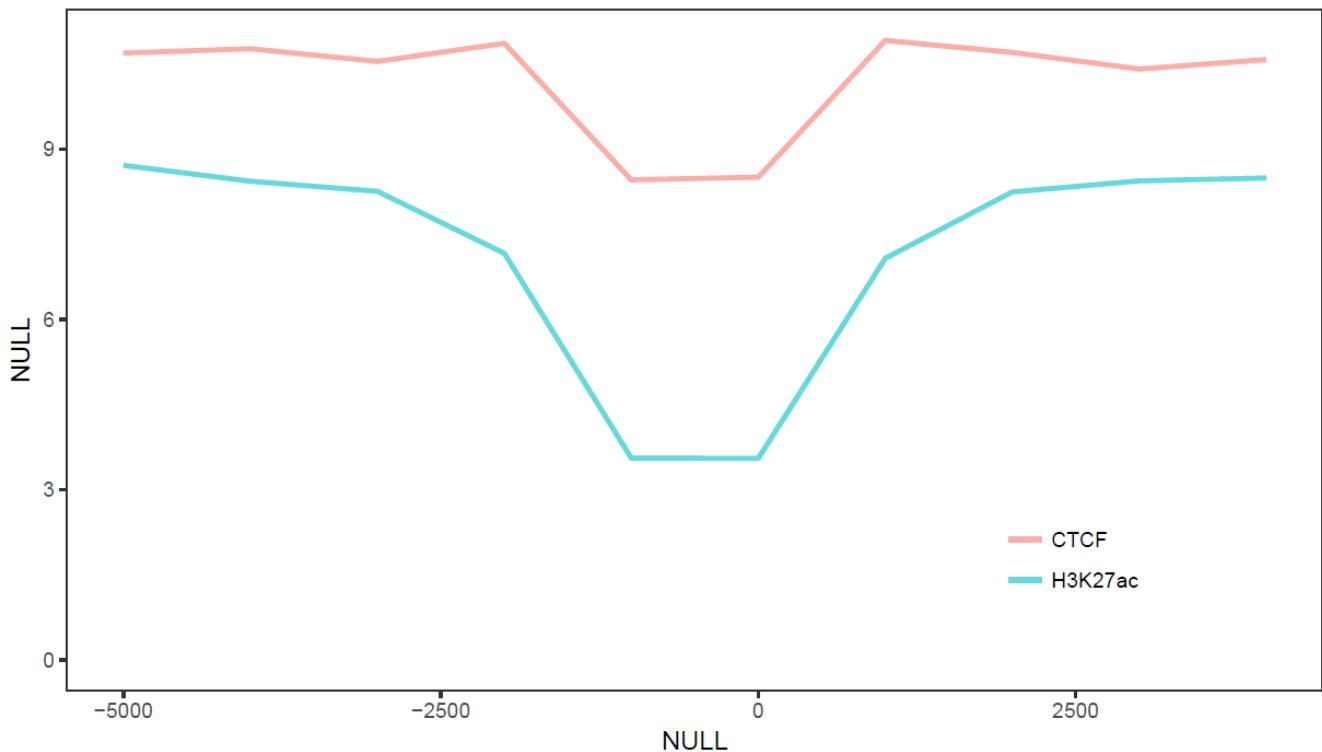
```

使用正常矩阵默认参数绘制个线图

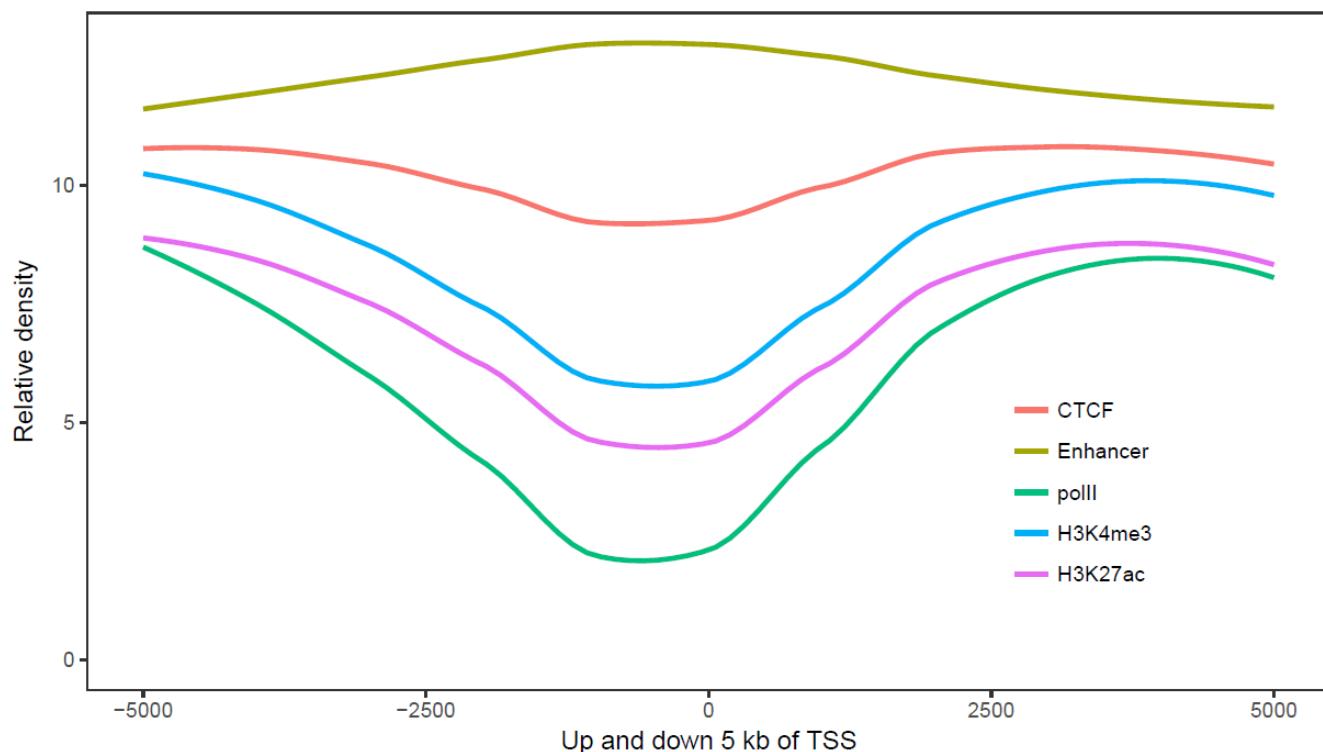
```
# -f: 指定输入的矩阵文件，第一列为行名字，第一行为header
    列数不限，列名字不限；行数不限，行名字默认为文本
# -A FALSE: 指定行名为数字
sp_lines.sh -f line_data.xls -A FALSE
```



```
# -l: 设定图例的顺序
# -o TRUE: 局部拟合获得平滑曲线
# -A FALSE: 指定行名为数字
# -P: 设置legend位置，相对于原点的坐标
# -x, -y指定横纵轴标记
sp_lines.sh -f line_data.xls -l "'CTCF','Enhancer','polII','H3K4me3','H3K27ac'" \
-P 'c(0.8,0.3)' -o TRUE -A FALSE -x 'Up and down 5 kb of TSS' -y 'Relative density'
```

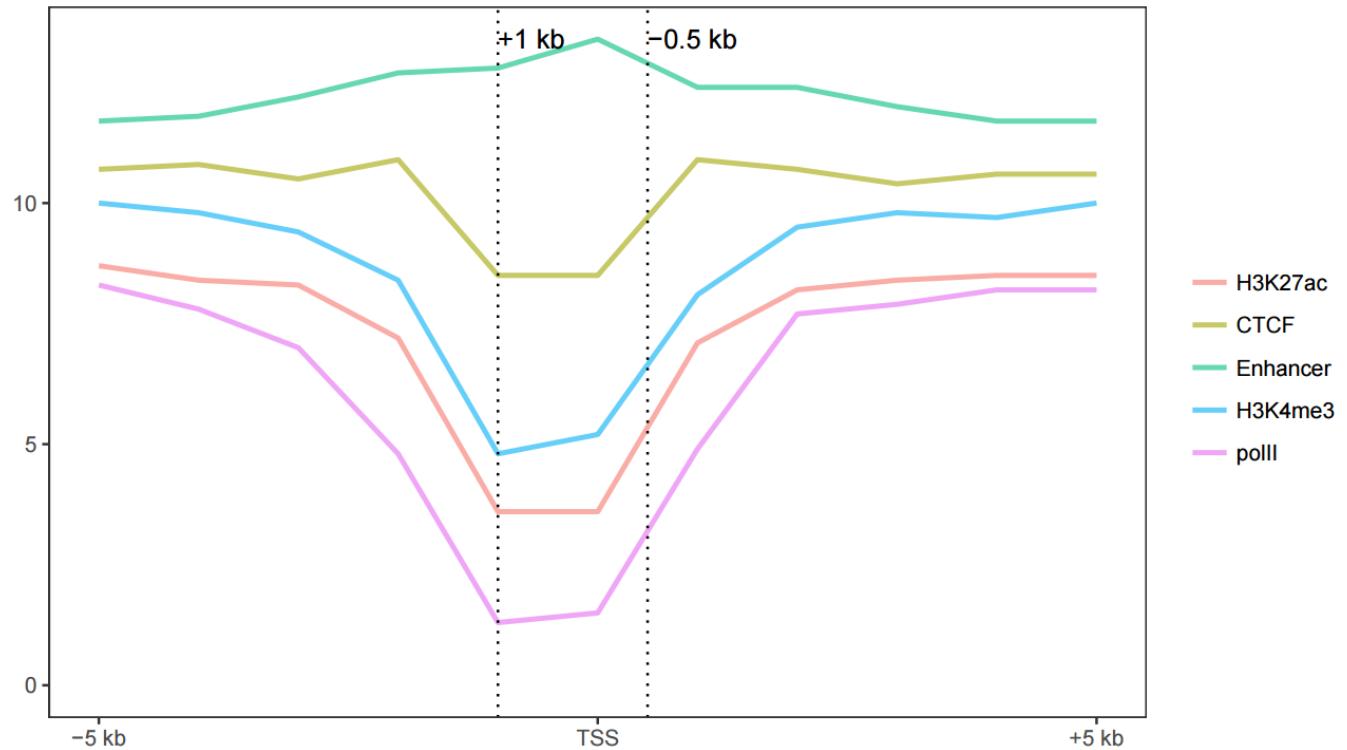


```
# -A FALSE: 指定行名为数字
# -V 'c(-1000, 500)': 设置垂线的位置
# -D: 设置垂线的文本标记，参数为引号引起来的vector，注意引号的嵌套
# -I: 设置横轴的标记的位置
# -b: 设置横轴标记的文字
sp_lines.sh -f line_data.xls -A FALSE -V 'c(-1000,500)' -D "c('+1 kb', '-0.5 kb')" \
-I "c(-5000,0,5000)" -b "c(' -5 kb', 'TSS', '+5 kb')"
```



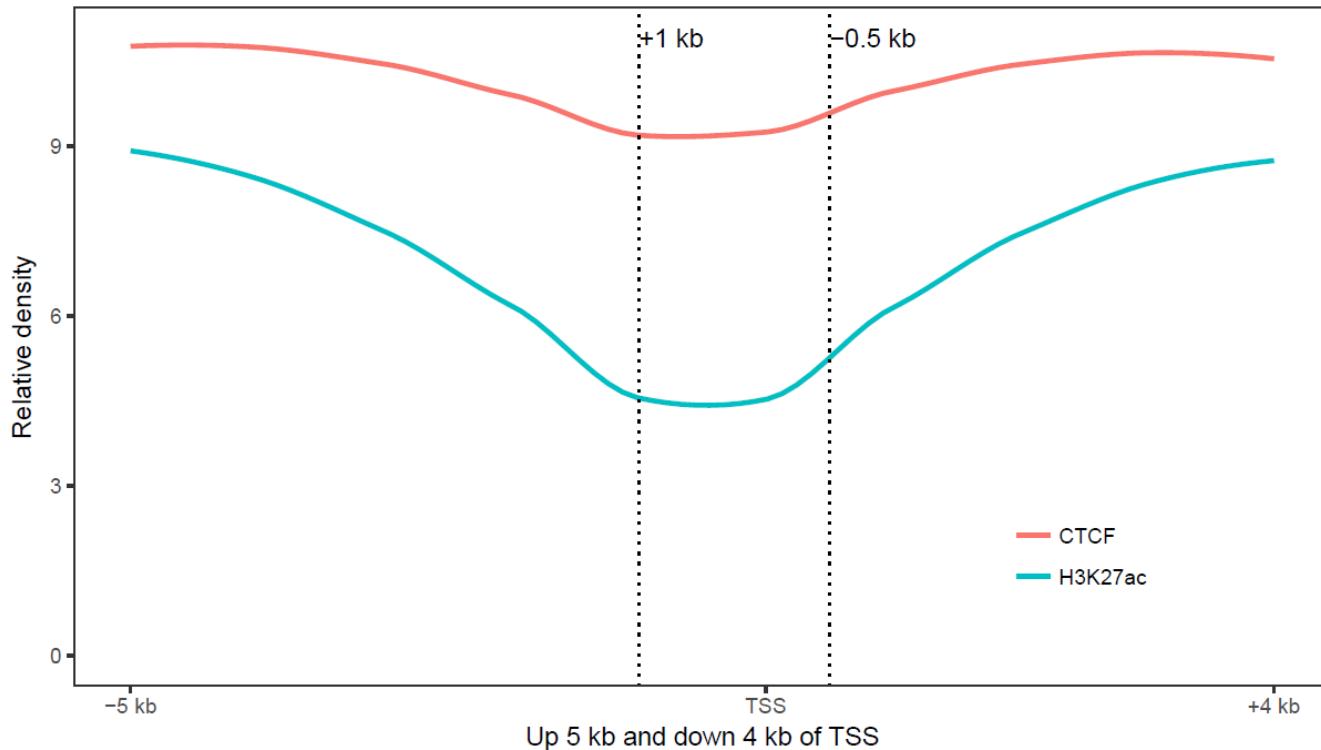
使用 melted 矩阵默认参数绘制个线图 (除需要改变文件格式，指定 -m TRUE, -a xvariable 外其它与正常矩阵一样)

```
# -f: 指定输入文件
# -m TRUE: 指定输入的矩阵为melted format, 三列, 第一列为Pos (给-a)
#           第二列为variable (给-H, -H默认即为variable)
#           第三列为value, 名字不可修改
# -A FALSE: 指定行名为数字
# -P 'c(0.8,0.2)': 设置legend位置, 相对于原点的坐标
sp_lines.sh -f line_data_melt.xls -a Pos -m TRUE -A FALSE -P 'c(0.8,0.2)'
```



完整的图

```
# -c: 自定义线的颜色
sp_lines.sh -f line_data_melt.xls -a Pos -m TRUE -A FALSE -P 'c(0.8,0.2)' -o TRUE \
-v 'c(-1000,500)' -D "c('+1 kb','-0.5 kb')" \
-I "c(-5000,0,4000)" -b "c('-5 kb', 'TSS', '+4 kb')" \
-x 'Up 5 kb and down 4 kb of TSS' -y 'Relative density' -C "'pink', 'blue'"
```



参数中最需要注意的是引号的使用：

- 外层引号与内层引号不能相同
- 凡参数值中包括了空格，括号，逗号等都用引号括起来作为一个整体。

完整参数列表如下：

```
ct@ehbio:~ $sp_lines.sh
***CREATED BY Chen Tong (chentong_biology@163.com) ***
Usage:
```

```
/MPATHB/self/s-plot/sp_lines.sh options
```

Function:

This script is used to draw a line or multiple lines using ggplot2.
You can specify whether or not smooth your line or lines.

Two types of input files are supported, normal matrix or melted matrix format. Column separator

Here is an example of normal matrix format. The first column will be treated as X-axis variable

****Set**** column is not needed. If given, <facet_plot> (multiple plots in one page) **could** be disp

Pos	H3K27ac	CTCF	Enhancer	H3K4me3	polII	
-5000	8.71298	10.69130		11.7359	10.02510	8.26866
-4000	8.43246	10.76680		11.8442	9.76927	7.78358
-3000	8.25497	10.54410		12.2470	9.40346	6.96859
-2000	7.16265	10.86350		12.6889	8.35070	4.84365
-1000	3.55341	8.45751	12.8372	4.84680	1.26110	
0	3.55030	8.50316	13.4152	5.17401	1.50022	
1000	7.07502	10.91430		12.3588	8.13909	4.88096
2000	8.24328	10.70220		12.3888	9.47255	7.67968
3000	8.43869	10.41010		11.9760	9.80665	7.94148
4000	8.48877	10.57570		11.6562	9.71986	8.17849

--With SET--

Pos	H3K27ac	CTCF	Enhancer	H3K4me3	polII	Set
-5000	8.71298	10.69130		11.7359	10.02510	8.26866 1
-4000	8.43246	10.76680		11.8442	9.76927	7.78358 1
-3000	8.25497	10.54410		12.2470	9.40346	6.96859 1
-2000	7.16265	10.86350		12.6889	8.35070	4.84365 1
-1000	3.55341	8.45751	12.8372	4.84680	1.26110	1
0	3.55030	8.50316	13.4152	5.17401	1.50022	1
1000	7.07502	10.91430		12.3588	8.13909	4.88096 1
2000	8.24328	10.70220		12.3888	9.47255	7.67968 1
3000	8.43869	10.41010		11.9760	9.80665	7.94148 1
4000	8.48877	10.57570		11.6562	9.71986	8.17849 1
-5000	8.71298	10.69130		11.7359	10.02510	8.26866 2
-4000	8.43246	10.76680		11.8442	9.76927	7.78358 2
-3000	8.25497	10.54410		12.2470	9.40346	6.96859 2
-2000	7.16265	10.86350		12.6889	8.35070	4.84365 2
-1000	3.55341	8.45751	12.8372	4.84680	1.26110	2
0	3.55030	8.50316	13.4152	5.17401	1.50022	2
1000	7.07502	10.91430		12.3588	8.13909	4.88096 2
2000	8.24328	10.70220		12.3888	9.47255	7.67968 2
3000	8.43869	10.41010		11.9760	9.80665	7.94148 2
4000	8.48877	10.57570		11.6562	9.71986	8.17849 2

For matrix format, example command lines include:

* Attribute of X-axis value (first column of matrix) **is <number>**

s-plot lines -f matrix.file -A FALSE

* Attribute of X-axis value (first column of matrix) **is <text>**

```
s-plot lines -f matrix.file

* Attribute of X-axis value (first column of matrix) is numbers, change legned order (default a

s-plot lines -f matrix.file -l "'polII', 'CTCF', 'Enhancer', 'H3K27ac', 'H3K4me3'"

* Attribute of X-axis value (first column of matrix) is numbers, change legned order (default a

s-plot lines -f matrix.file -l "'polII', 'CTCF', 'Enhancer', 'H3K27ac', 'H3K4me3'" -o TRUE

* Attribute of X-axis value (first column of matrix) is numbers, with <Set> (Set is column name)

s-plot lines -f matrix.file -F "+facet_grid(Set ~ ., scale='free_y')"
```

FILEFORMAT when -m is true

#The name "value" shoud **not** be altered.
#variable can be altered using -H
#Actually this format is the melted result of last format.

Pos	variable	value
-5000	H3K27ac	8.71298
-4000	H3K27ac	8.43246
-3000	H3K27ac	8.25497
-2000	H3K27ac	7.16265
-1000	H3K27ac	3.55341
0	H3K27ac	3.55030
1000	H3K27ac	7.07502
2000	H3K27ac	8.24328
3000	H3K27ac	8.43869
4000	H3K27ac	8.48877
-5000	CTCF	10.69130
-4000	CTCF	10.76680
-3000	CTCF	10.54410
-2000	CTCF	10.86350
-1000	CTCF	8.45751
0	CTCF	8.50316
1000	CTCF	10.91430
2000	CTCF	10.70220
3000	CTCF	10.41010
4000	CTCF	10.57570

* Attribute of X-axis value (melt format) **is** <number>

```
s-plot lines -f matrix.file -m TRUE -a Pos -A FALSE
```

- * Attribute of X-axis value (first column of matrix) **is** <text>

```
s-plot lines -f matrix.file -m TRUE -a Pos
```
- * If the name of the second column is <type> not <variable>, one should specify with <-H>.

```
s-plot lines -f matrix.file -A FALSE -m TRUE -a Pos -H type
```
- * Attribute of X-axis value (first column of matrix) **is** numbers, change legned order (default a

```
s-plot lines -f matrix.file -m TRUE -a Pos -l "'polII', 'CTCF', 'Enhancer', 'H3K27ac', 'H3K
```
- * Attribute of X-axis value (first column of matrix) **is** numbers, change legned order (default a

```
s-plot lines -f matrix.file -m TRUE -a Pos -l "'polII', 'CTCF', 'Enhancer', 'H3K27ac', 'H3K
```
- * Attribute of X-axis value (first column of matrix) **is** numbers, with <Set> (Set is column name

```
s-plot lines -f matrix.file -F "+facet_grid(Set ~ ., scale='free_y')"
```

OPTIONS:

- f** Data file (with header line, the first column would be treated as rownames for **normal** matrix. No rownames for melted format. Columns are tab seperated)
[NECESSARY]
- m** When true, it will skip melt preprocesses. But the format must be **the** same as listed before.
[Default FALSE, accept TRUE]
- a** Name for x-axis variable
[Only needed when <-m> is <TRUE>].
For the melted data, '**Pos**' should be given here.
For normal matrix, default the first column will be used,
program will assign an value '**xvariable**' to represent it.
`]]`
- A** Are x-axis variables numbers.
[Default <TRUE>, meaning X-axis label is <text>.
<FALSE> means X-axis label is <numerical>.)
- H** Name for legend variable.
[Default variable, this should only be set when -m is TRUE]
- J** Name for color variable.
[Default same as -H, this should only be set when -m is TRUE]
- l** Set orders of legend variable.
**[Default column order for normal matrix, accept a string like
`"'CTCF', 'H3K27ac', 'Enhancer'"` to set your own order.**
Pay attention to the usage of two types of quotes.
*****When** -m is TRUE, default order would be alphabet order.*****
`]`

-P Legend position[Default right. Accept
top, bottom, left, none, or 'c(0.08,0.8)'.]

-L Levels for x-axis variable, suitable when x-axis is not treated as numerical.
[Default] the order of first column for normal matrix.
Accept a string like "'g','a','j','x','s','c','o','u'" to set your own order.
This will only be considered when -A is TRUE.
*****When** -m is used, this default order would be alphabet order.*****
]

-o Smooth lines or not.
[Default] FALSE means no smooth. Accept TRUE to smooth lines.]

-O The smooth method you want to use.
[smoothing method (function) **to** use, eg. lm, glm, gam, loess, rlm.
For datasets with n < 1000 default is '**loess**'.
For datasets with 1000 or more observations defaults to '**gam**'.
]

-v Add vertical lines.[Default FALSE, accept a series of
numbers in following format "c(1,2,3,4,5)" or other
R code that can generate a vector.]

-D Add labels to vlines.
[Default] same as -V.
Accept a series of numbers in following format "c(1,2,3,4,5)" or other R code
that can generate a vector as labels.
Or one can give '1' to disallow labels]

-j Add horizontal lines.[Default FALSE, accept a series of
numbers in following format "c(1,2,3,4,5)" or other
R code that can generate a vector]

-d Add labels to hline.
[Default] same as -j
Accept a series of numbers in following format "c(1,2,3,4,5)" or other R code
that can generate a vector as labels.
Or one can give '1' to disallow labels]

-I Manually set the position of xtics.
[Default FALSE, accept a series of
numbers in following format "c(1,2,3,4,5)" or other R code
that can generate a vector to set the position of xtics]

-b Manually set the value of xtics when -I is specified.
[Default the content of -I when -I is specified,
accept a series of numbers in following format "c(1,2,3,4,5)" or other R code
that can generate a vector to set the position of xtics]

-x Display xtics. [Default TRUE]

-y Display ytics. [Default TRUE]

-R Rotation angle for x-axis labels (anti clockwise)
[Default 0]

-B line size. [Default 1. Accept a number.]

-t Title of picture[Default empty title]

-x xlab of picture[Default empty xlab]

-y ylab of picture[Default empty ylab]

-c Manually set colors for each line.[Default FALSE, meaning using ggplot2 default.]
-C Color for each line.

When -c is TRUE, one has two options:

1. Supplying a function to generate colors,

like "rainbow(11)" or "rainbow(11, alpha=0.6)",

rainbow is an R color pallettes,

11 is the number of colors you want to get,

0.6 is the alpha value.

The R pallettes include <heat.colors>, <terrain.colors>,

<topo.colors>, <cm.colors>.

2. Supplying a list of colors in given format,

the number of colors should be equal to the number of

bars like "'red','pink','blue','cyan','green','yellow'" or

"rgb(255/255,0/255,0/255),rgb(255/255,0/255,255/255),

rgb(0/255,0/255,255/255),rgb(0/255,255/255,255/255),

rgb(0/255,255/255,0/255),rgb(255/255,255/255,0/255)"

One can use R fucntion <colors()> **to** list all available colors.

-s Scale y axis

[**Default** null. Accept TRUE. This function is depleted.]

But if the supplied number after -S is not 0, this parameter will be set to TRUE]

-F The formula for facets.[Default no facets,

"+facet_grid(level ~ .)" **means** divide by levels of 'level' vertically.

"+facet_grid(. ~ level)" **means** divide by levels of 'level' horizontally.

"+facet_grid(lev1 ~ lev2)" **means** divide by lev1 vertically and lev2 horizontally.

"+facet_wrap(~level, ncol=2)" **means** wrap horizontally with 2 columns.

#Pay attention to the single quote for parameters in function for scale.

Example: "+facet_wrap(~Size,ncol=6,scale='free')"

Example: "+facet_grid(Size ~ .,scale='free_y')"

]

-G If facet is given, you may want to specifize the order of **variable** in your facet, default alphabetical order.

[**Accept** sth like (one level one sentence, separate by';')

'data\$size <- factor(data\$size, levels=c("11", "12",...,"110"), ordered=T)']

-v If scale is TRUE, give the following 'scale_y_log10()' [default], 'coord_trans(y="log10")'
or other legal command for ggplot2 or simply 'log2'.]

-S A number to add if scale is used.

[**Default** 0. If a non-zero number is given, -s would be set to TRUE.]

-p Other legal R codes for gggplot2 could be given here.

[**Begin** with '+']

-w The width of output picture (cm).[Default 20]

-u The height of output picture (cm).[Default 12]

-E The type of output figures.[Default pdf, accept

```

eps/ps, tex (pictex), png, jpeg, tiff, bmp, svg and wmf) ]
-r The resolution of output picture.[Default 300 ppi]
-z Is there a header. Must be TRUE. [Default TRUE]
-e Execute or not[Default TRUE]
-i Install depended packages[Default FALSE]

```

3.16.3 一网打进散点图绘制

假如有一个输入数据如下所示 (存储于文件 scatterplot.xls 中)

Samp		Gene1	Gene2	Color	Size	GC_quality	Base_quality
a	1	1	grp1	10	PASS	PASS	
b	2	2	grp1	10	PASS	PASS	
c	1	3	grp1	10	WARN	PASS	
d	3	1	grp2	15	WARN	WARN	
e	2	2	grp2	15	PASS	WARN	
f	3	3	grp3	5	PASS	PASS	
g	2	1	grp3	5	WARN	PASS	

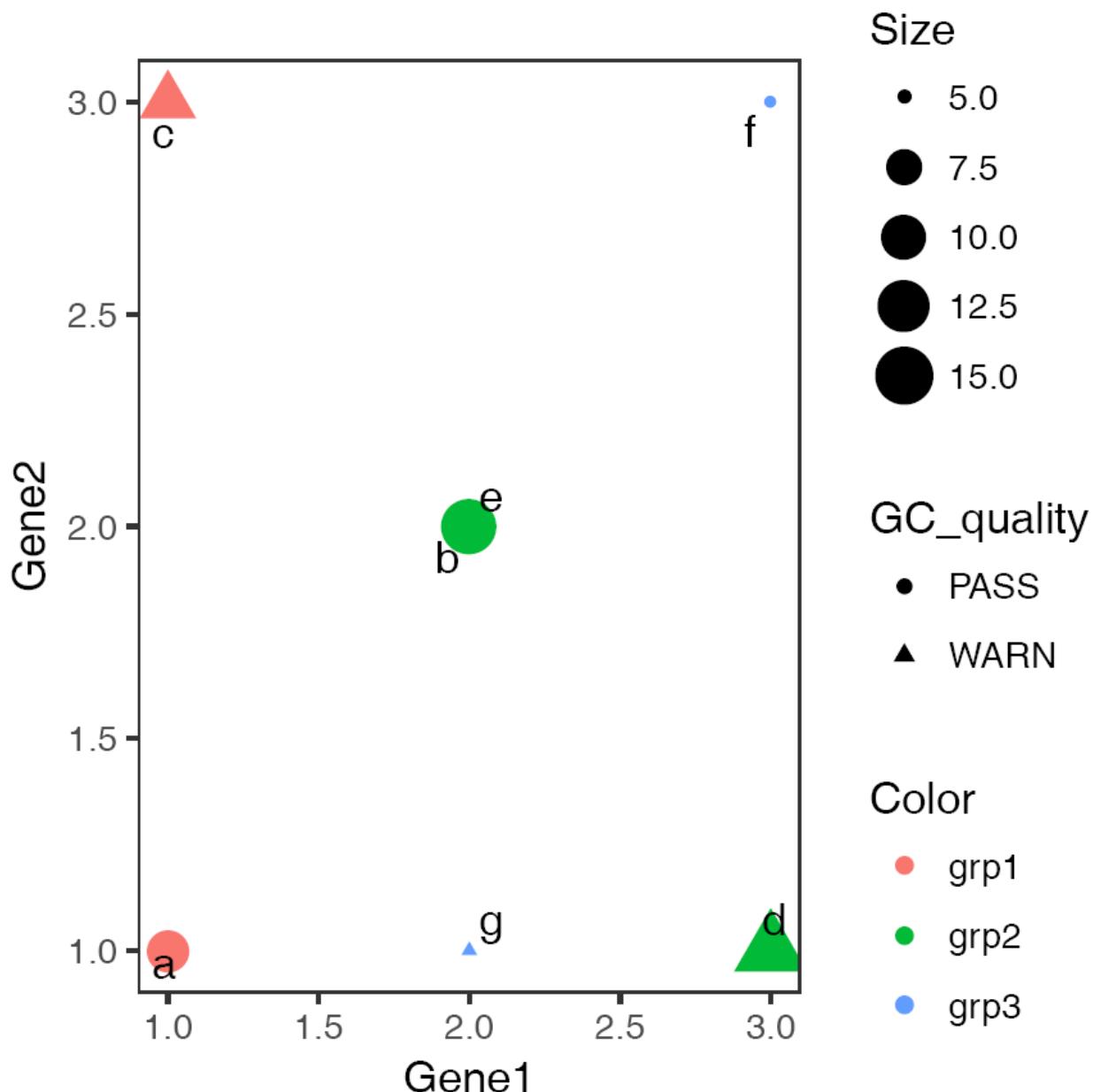
想绘制样品在这两个 Gene 为轴的空间的分布，并标记样品的属性，只需要运行如下命令

```

# -f: 指定输入文件，列数不限，顺序不限；第一行为列名字，第一列无特殊要求，必选
# -X: 指定哪一列为 X 轴信息，必选
# -Y: 指定哪一列为 Y 轴信息，必选
# -c: 指定用哪一列标记颜色，可选
# -s: 指定哪一列标记大小，一般为数字列，可选
# -S: 指定哪一列标记形状，可选
# -L: 指定哪一列用来作为文本标记
# -w, -u: 指定图的长宽

sp_scatterplot2.sh -f scatterplot.xls -X Gene1 -Y Gene2 -c Color -s Size \
-S GC_quality -L Samp -w 10 -u 10

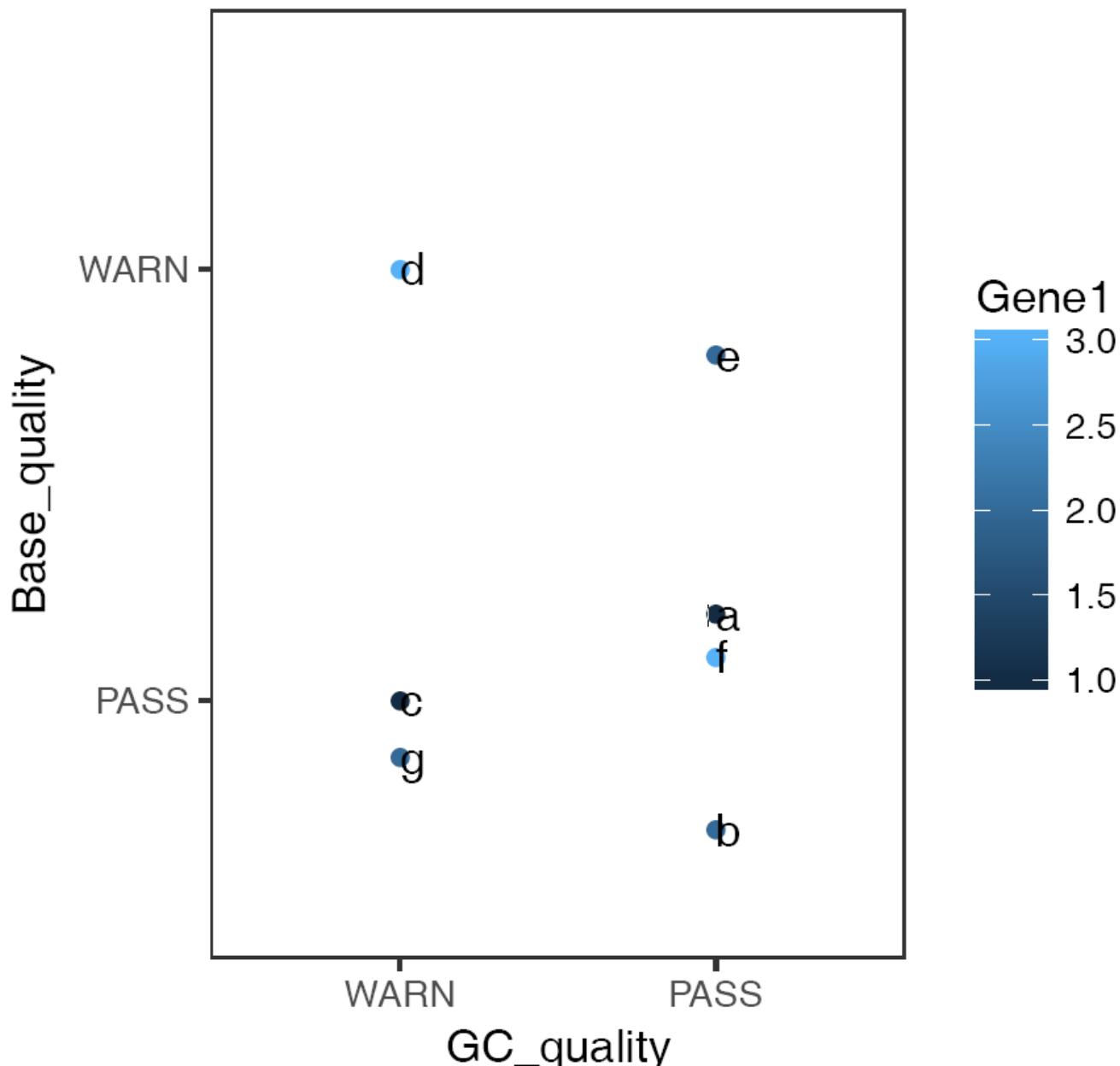
```



如果横纵轴为字符串，且有重复，则需指定参数-J TRUE 以错开重叠的点，具体如下

```
# -o: 指定 x 轴变量的顺序, 默认是字母顺序
# 其它列或其它属性的顺序也可以用相应的方式指示, 具体看程序的帮助提示
# -c Gene1: 用特定基因的表达对点着色, 单细胞分析图中常用
# -J TRUE: 见上
# -z FALSE: 默认使用 geom_text_repel 添加点的标记, 及其智能, 不会出现标签过多覆盖的情况
# 但对 jitterplot, 会有些冲突, 所以在 ` -J TRUE` 且出来的图中点的标签不符合预期时, 设定
# 此参数为 FALSE, 使用 geom_text 标记点。

sp_scatterplot2.sh -f scatterplot.xls -X GC_quality -Y Base_quality \
-o "'WARN', 'PASS'" -c Gene1 -w 10 -u 10 -J TRUE -L Samp -Z FALSE
```



只有想不到，没有做不到，`sp_scatterplot2.sh` 还可以完成更多你想做的散点图，而且只需调参数，无需改代码，简单可重用。

3.17 参考资料

- <http://rpubs.com/xuefliang/153247>
- <http://www.sthda.com/english/wiki/survminer-r-package-survival-data-analysis-and-visualization>

4 交互式图

networkD3 是基于 D3JS 的 R 包交互式绘图工具，用于转换 R 语言生成的图为交互式网页嵌套图。目前支持网络图，桑基图，树枝图等。

关于网络图的绘制，我们之前有 5 篇文章，可点击查看。

- [Cytoscape 教程 1](#)
- [Cytoscape 之操作界面介绍](#)
- [新出炉的 Cytoscape 视频教程](#)
- [Cytoscape: MCODE 增强包的网络模块化分析](#)
- [一文学会网络分析——Co-occurrence 网络图在 R 中的实现](#)

4.1 交互式网络图

也可以使用此文介绍的 network3D 绘制交互式网络图，输入数据与 Cytoscape 需要的数据格式一致。

运行下方脚本，可得到这个网络图。是关于我们培训现在开通报名的课程、开过的课程和即将要开的课程。

如果需要用自己的数据，也只需替换数据部分，其它部分都是写好的通用脚本。

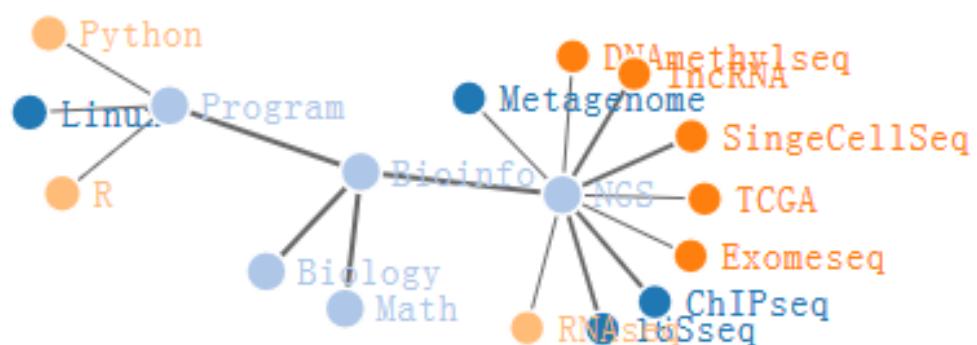
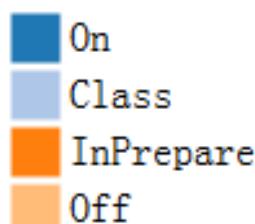


Figure 4.1: 交互式网络图

```
#install.packages("networkD3")
library("networkD3")

# 网络数据和节点属性数据以类似格式存入文本文件即可
# 网络文件有 3 列组成，第一列为
network <- "Src;Target;Value
Bioinfo;Biology;4
Bioinfo;Math;4
Bioinfo;Program;4
Bioinfo;NGS;4
Program;Linux;1
Program;Python;1
Program;R;1
NGS;RNASEQ;1
NGS;ChIPseq;3
NGS;16Sseq;3
NGS;Metagenome;1
NGS;SingleCellSeq;3
NGS;DNAmethylseq;1
NGS;lncRNA;3
NGS;Exomeseq;1
NGS;TCGA;1
"

attribute <- "name;group;size
Bioinfo;Class;4
Biology;Class;4
Math;Class;4
Program;Class;4
NGS;Class;4
Linux;On;2
Python;Off;2
R;Off;2
RNAseq;Off;1
ChIPseq;On;1
16Sseq;On;1
Metagenome;On;1
SingleCellSeq;InPrepare;1
DNAmethylseq;InPrepare;1
lncRNA;InPrepare;1
Exomeseq;InPrepare;1
TCGA;InPrepare;1"

network <- read.table(text=network, sep=";", header=T, row.names=NULL, quote="", comment="")

network <- network[,1:3]
colnames(network) <- c("Src", "Target", "Value")
```

```

nodes <- unique(c(network$Src, network$Target))
factor_list <- sort(unique(c(levels(network$Src), levels(network$Target))))
num_list <- 0:(length(factor_list)-1)
levels(network$Src) <- num_list[factor_list %in% levels(network$Src)]
levels(network$Target) <- num_list[factor_list %in% levels(network$Target)]

attribute <- read.table(text=attribute, sep=";", header=T, row.names=NULL, quote="", comment="")
attribute <- attribute[match(factor_list, attribute$name),]

forceNetwork(Links = network, Nodes = attribute,
            width = 600, height=400,
            Source = "Src", Target = "Target",
            Value = "Value", NodeID = "name",
            Group = "group", opacity = 1,
            legend = T, zoom = T, Nodesize = "size",
            bounded = T, opacityNoHover = 1, fontSize = 15)

```

4.2 交互式桑基图

桑基图（Sankey diagram），即桑基能量分流图，也叫桑基能量平衡图。它是一种特定类型的流程图，图中延伸的分支的宽度对应数据流量的大小，通常应用于能源、材料成分、金融等数据的可视化分析。

也可以视为一种层级网络图，比如展示[上一篇文章中的生物信息课程网络图](#)；也可以展示菌群随时间变化的趋势，如[3分和30分文章差距在哪里](#)文章所示哈扎人肠道菌群的季节变化规律。

下面将用 2 个例子，展示如何用常见网络图数据绘制桑基图。

最简单桑基图

第一列为上游，第二列为下游，第三列为联通值，值越大线越粗。如果您自己有数据，只需要替换输入部分，后面数据格式转换代码是通用的。

```

network <- "Src;Target;Value
Bioinfo;Biology;20
Bioinfo;Math;20
Bioinfo;Program;20
Bioinfo;NGS;20
Program;Linux;8
Program;Python;8
Program;R;6

```

```

NGS;RNAseq;1
NGS;ChIPseq;1
NGS;m16Sseq;1
NGS;Metagenome;1
NGS;SingleCellSeq;1
NGS;DNAmethylseq;1
NGS;lncRNA;1
NGS;Exomeseq;1
NGS;TCGA;1
"

network <- read.table(text=network, sep=";", header=T, row.names=NULL, quote="", comment="")

network <- network[,1:3]
colnames(network) <- c("Src", "Target", "Value")

# 转换原始数据点为0起始的一系列整数表示
factor_list <- sort(unique(c(levels(network$Src), levels(network$Target))))
num_list <- 0:(length(factor_list)-1)
levels(network$Src) <- num_list[factor_list %in% levels(network$Src)]
levels(network$Target) <- num_list[factor_list %in% levels(network$Target)]

network$Src <- as.numeric(as.character(network$Src))
network$Target <- as.numeric(as.character(network$Target))

attribute <- data.frame(name=c(factor_list))

```

network

```
Src Target Value 1 1 2 20 2 1 8 20 3 1 11 20 4 1 10 20 5 11 6 8 6 11 12 8
```

attribute

```
head(attribute[, 1]) [1] 16Sseq Bioinfo Biology ChIPseq DNAmethylseq [6] Exomeseq
```

```

sankeyNetwork(Links = network, Nodes = attribute,
  Source = "Src", Target = "Target",
  Value = "Value", NodeID = "name",
  font_size= 12, nodeWidth = 30)

```

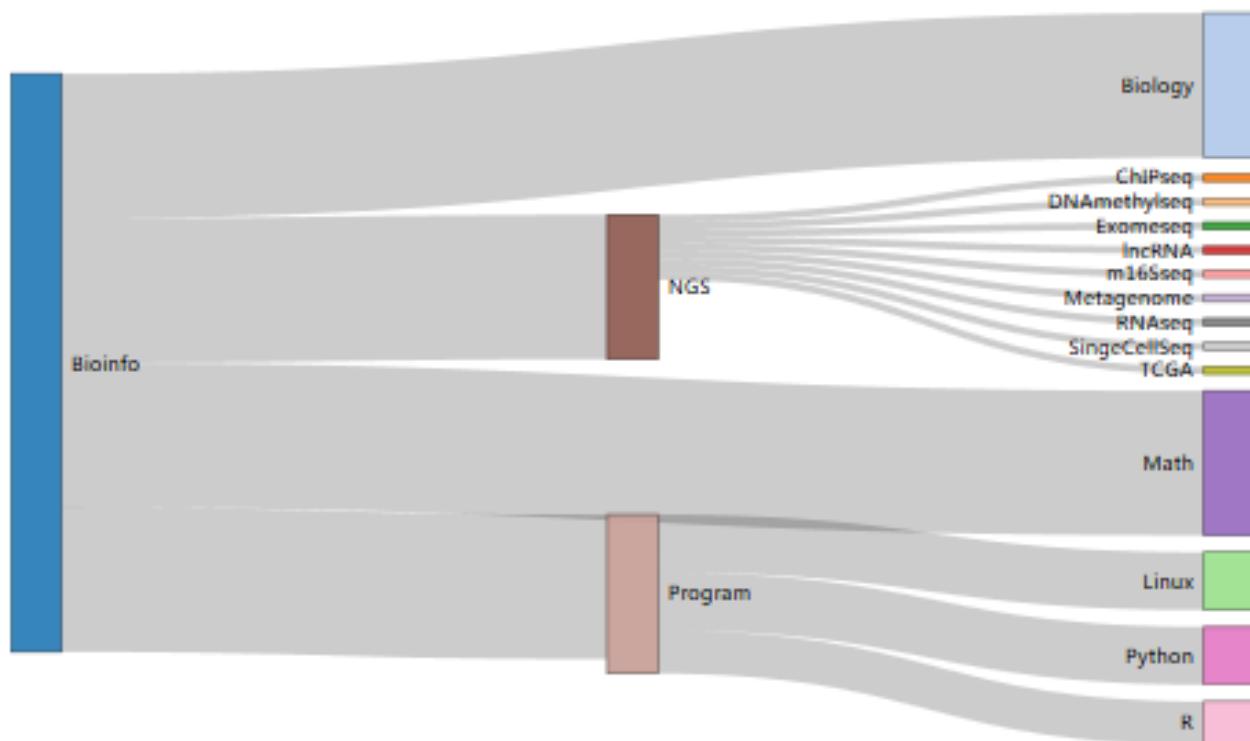


Figure 4.2: 交互式桑基图

点线分组桑基图

网络数据比上一步的桑基图多一列，指示线的属性；再提供一个节点分组信息文件，获得层次更鲜明的桑基图。

只需要修改对应的数据，后面格式转换的代码通用。

```
network <- "Src;Target;Value;Link_Grp
Bioinfo;Biology;20;Main
Bioinfo;Math;20;Main
Bioinfo;Program;20;Main
Bioinfo;NGS;20;Main
Program;Linux;8;Sub
Program;Python;8;Sub
Program;R;6;Sub
NGS;RNaseq;1;Sub
NGS;ChIPseq;1;Sub
NGS;16Sseq;1;Sub
NGS;Metagenome;1;Sub
NGS;SingleCellSeq;1;Sub
NGS;DNAmethylseq;1;Sub
NGS;lncRNA;1;Sub
```

CONTENTS

```

NGS;Exomeseq;1;Sub
NGS;TCGA;1;Sub
"

network <- read.table(text=network, sep=";", header=T, row.names=NULL, quote="", comment="")

network <- network[,1:4]
colnames(network) <- c("Src", "Target", "Value", "Link_Grp")

factor_list <- sort(unique(c(levels(network$Src), levels(network$Target) )))
num_list <- 0:(length(factor_list)-1)
levels(network$Src) <- num_list[factor_list %in% levels(network$Src) ]
levels(network$Target) <- num_list[factor_list %in% levels(network$Target) ]

network$Src <- as.numeric(as.character(network$Src))
network$Target <- as.numeric(as.character(network$Target))

# 只需要前两列
attribute <- "name;group;size
Bioinfo;Class;4
Biology;Class;4
Math;Class;4
Program;Class;4
NGS;Class;4
Linux;On;2
Python;Off;2
R;Off;2
RNAseq;Off;1
ChIPseq;On;1
16Sseq;On;1
Metagenome;On;1
SingleCellSeq;InPrepare;1
DNAmethylseq;InPrepare;1
lncRNA;InPrepare;1
Exomeseq;InPrepare;1
TCGA;InPrepare;1"

attribute <- read.table(text=attribute, sep=";", header=T, row.names=NULL, quote="", comment="")
attribute <- attribute[,1:2]
colnames(attribute) <- c("name", "group")
attribute <- attribute[match(factor_list, attribute$name),]

sankeyNetwork(Links = network, Nodes = attribute,
  Source = "Src", Target = "Target",
  Value = "Value", NodeID = "name",
  NodeGroup = "group", LinkGroup = "Link_Grp",

```

CONTENTS

fontSize= 14, nodeWidth = 30)

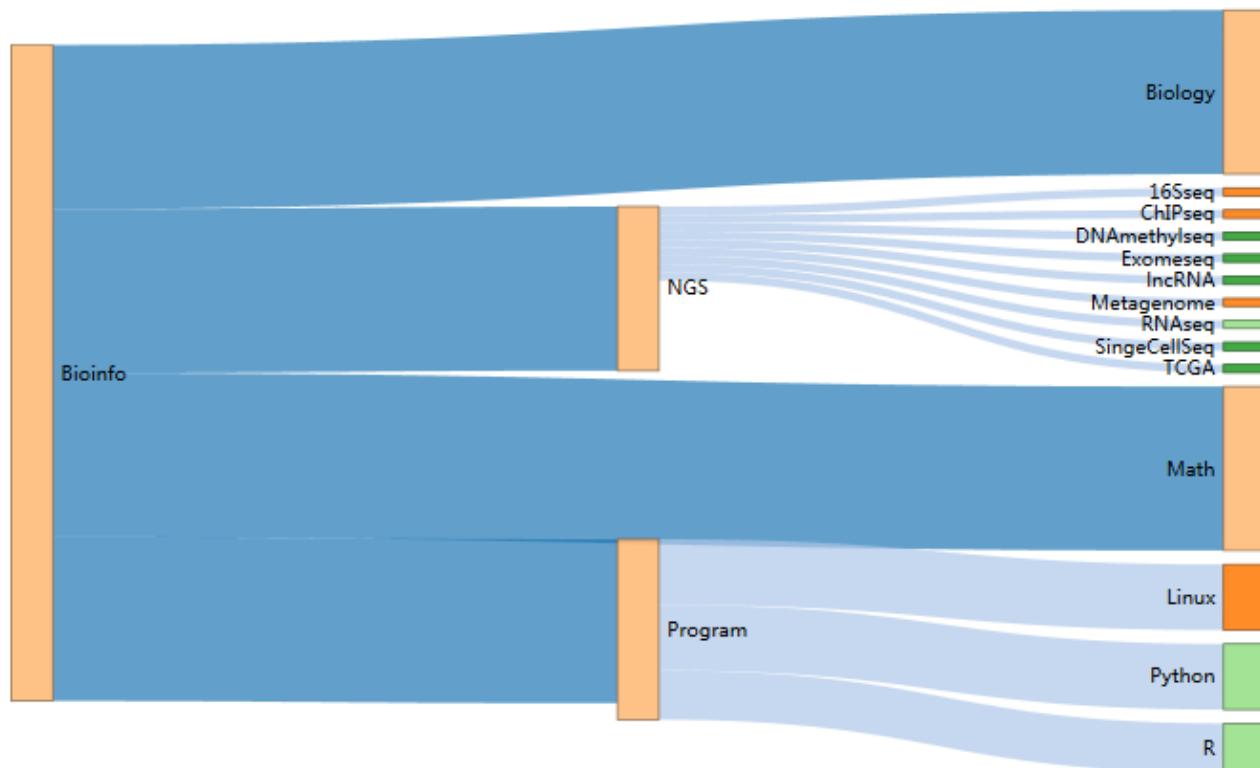


Figure 4.3: 交互式桑基图

桑基图还有类似的称为冲击图 (alluvial diagram) 的展示，具体可见[ggalluvial：冲击图展示组间变化、时间序列和复杂多属性 alluvial diagram](#)。

说到交互式可视化，还有之前推出的：

- R 语言交互式可视化包 CanvasXpress
- 视频教程：R 语言 recharts 包绘制交互式图形

关于 R 绘图, 更多文章如下：

- 在 R 中赞扬下努力工作的你，奖励一份 CheatSheet
- 别人的电子书，你的电子书，都在 bookdown
- R 语言 - 入门环境 Rstudio
- R 语言 - 热图绘制 (heatmap)
- R 语言 - 基础概念和矩阵操作
- R 语言 - 热图简化
- R 语言 - 热图美化

CONTENTS

- R 语言 - 线图绘制
- R 语言 - 线图一步法
- R 语言 - 箱线图 (小提琴图、抖动图、区域散点图)
- R 语言 - 箱线图一步法
- R 语言 - 火山图
- R 语言 - 富集分析泡泡图 (文末有彩蛋)
- R 语言 - 散点图绘制
- 一文看懂 PCA 主成分分析
- 富集分析 DotPlot，可以服
- R 语言 - 韦恩图
- R 语言 - 柱状图
- R 语言 - 图形设置中英字体
- R 语言 - 非参数法生存分析
- 基因共表达聚类分析和可视化
- R 中 1010 个热图绘制方法
- 还在用 PCA 降维？快学学大牛最爱的 t-SNE 算法吧, 附 Python/R 代码
- 一个函数抓取代谢组学权威数据库 HMDB 的所有表格数据
- 文章用图的修改和排版

[点击阅读原文](#) , 了解更多培训信息。

5 在线绘图

访问网站 <http://www.ehbio.com/ImageGP>。

5.1 功能和数据概览

5.1.1 图形支持

设计有生信分析常见的 14 种图，都已实现。

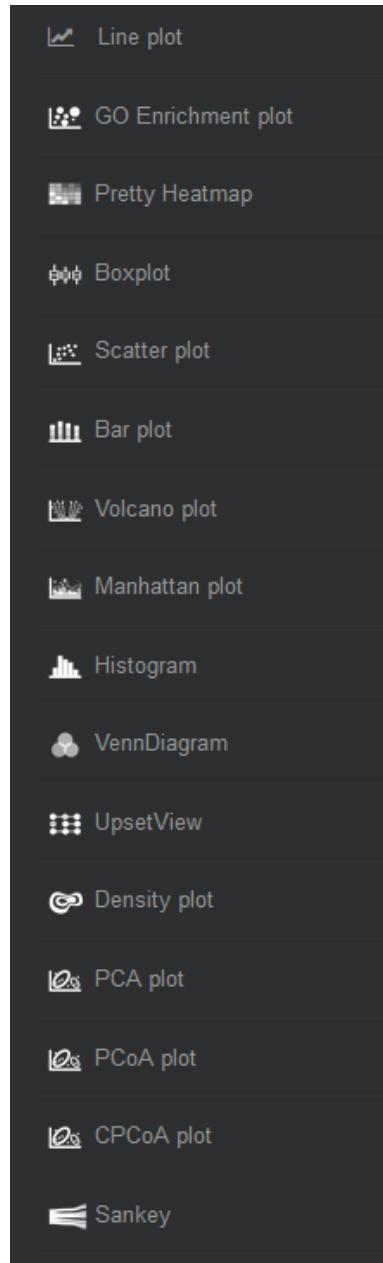


Figure 5.1: ImageGP 生信绘图

5.1.2 整体界面

初进入是网站展示和图形 Demo 界面。轮播图中贴心地提供了 R 中支持的颜色的名字和代码，方便绘图时查询使用，用户可点击右上角按钮全屏查看。下方每个 Demo 图点击都可进入相应绘图页面。

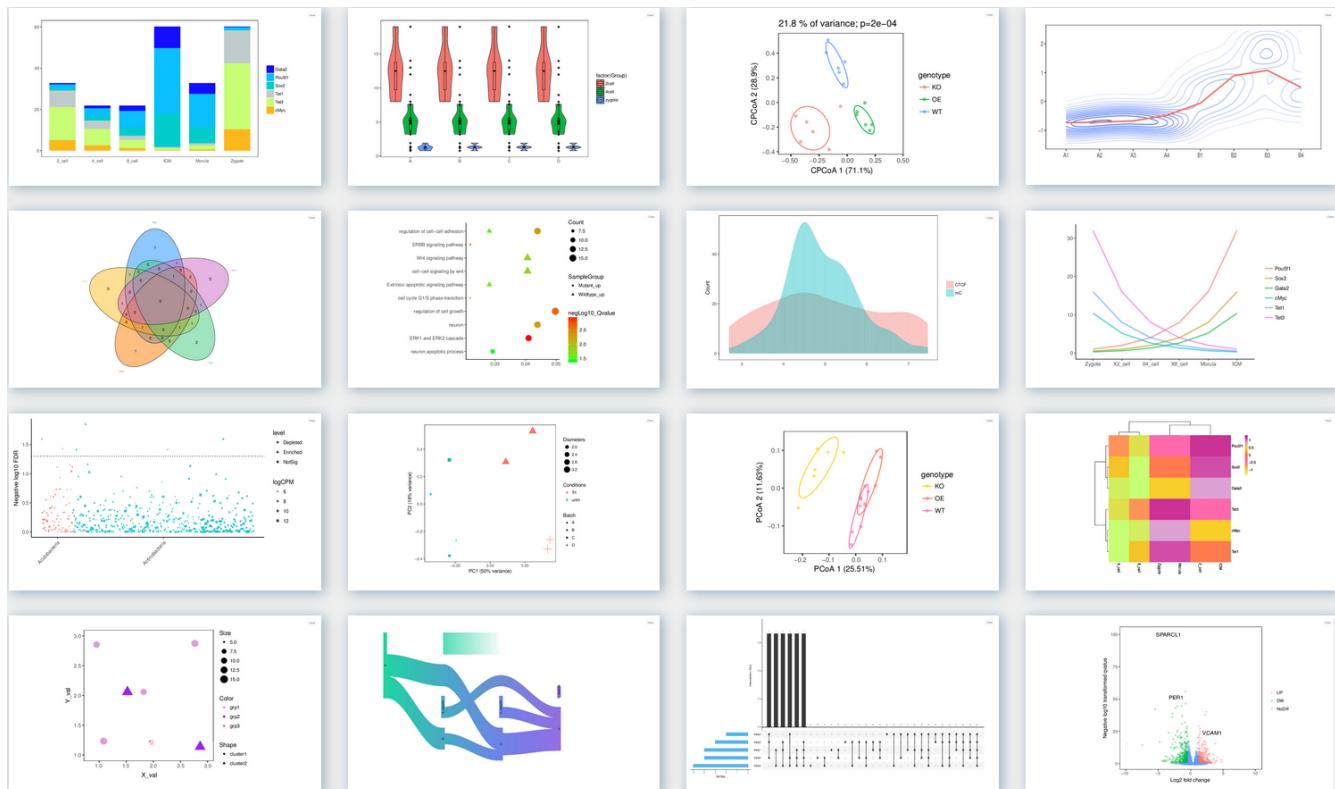


Figure 5.2: ImageGP preview (缩略图)

5.1.3 数据格式

输入数据格式根据绘图需要支持两种格式，Wide format 和 Long format。

Wide format 每一列为一个变量，更像常见的基因表达数据，比如每一行为一个基因，每一列为一个样品，适合于中规中矩的矩阵，可读性好。

Long format 最简单的格式是两列，一列包含所有变量类型，如前面提到的样品，一列包含所有值。常见的基因表达矩阵从 Wide format 转换为 Long format 后有三列，一列为 GeneID, 一列为样品名字，再一列(通常为 value)为基因表达值，表示对应基因在对应样品的表达量。这种格式方便程序处理，也适合不规则数据。

初使用此网站对输入数据结构不熟悉的用户，可以试用给出的默认输入数据，并仿照其格式编辑自己的数据。注意表格类输入数据要以 tab 分割。点击 Demo 按钮，查看对应的必填参数（会高亮显示）和填写方式。

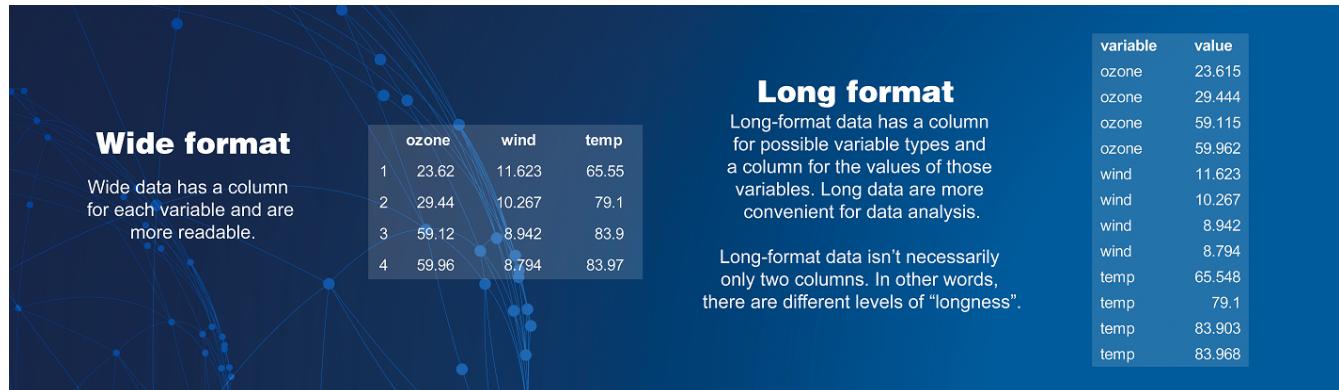


Figure 5.3: 数据格式

5.2 图形展示和绘图展示

5.2.1 线图

线图支持 2 种输入格式，基因表达趋势图使用常见的 `Wide format`，只是样品为行，基因为列，在 Excel 中可以很轻松转置；如果有需要，后期也会提供在线处理功能。

`Wide format` 没有必填的参数，输入数据后，直接点击 `Plot` 就可以出图了。

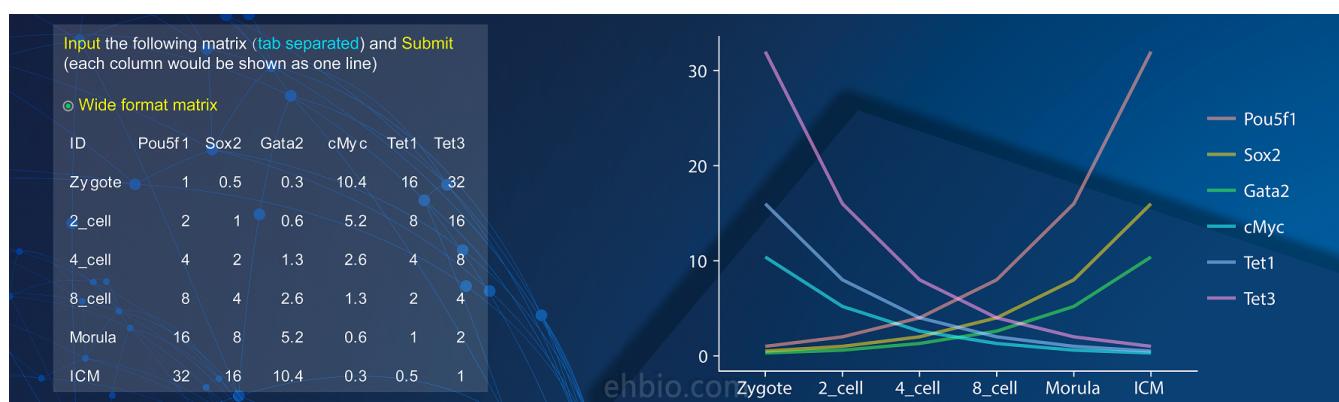


Figure 5.4: 线图

再复杂一点，增加标题、X-轴、Y-轴标题，调整图例位置和 X 轴旋转，就得到了右边的图。(黄色字体为对应参数，绿色文字为参数设置的值)

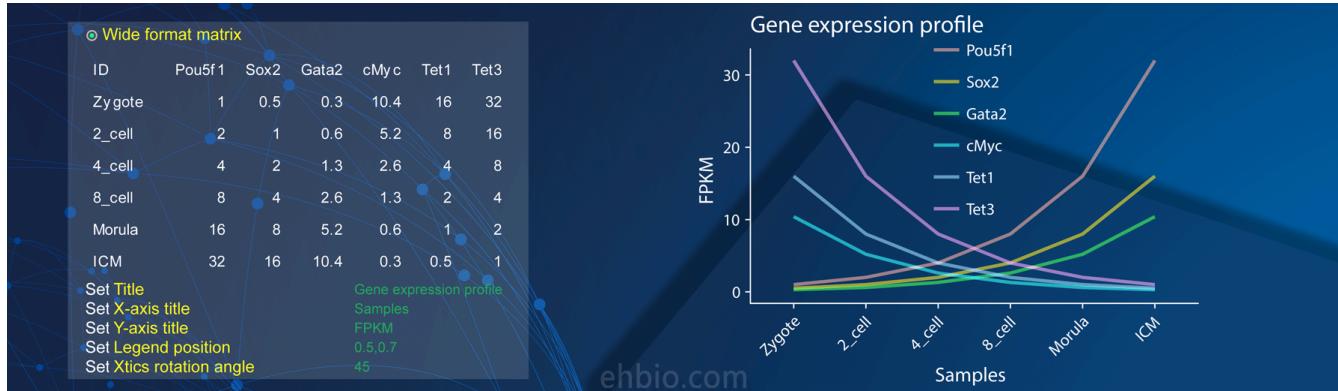


Figure 5.5: 线图

线图还支持长表格格式，需要特别注意的是 value 列是必须的，而且列的名字必须是 value。这种格式下有必填项需要指定哪一列作为 X 轴 X-axis variable 和哪一列做分组信息(图例信息)Legend variable，同时需要指定 X 轴是数值类型 Continuous 还是字符串类型 Discrete(数字可以做为数值处理，也可以作为字符串处理，根据图形意义的需要来选择)。然后就可以点击 Plot 出图了。

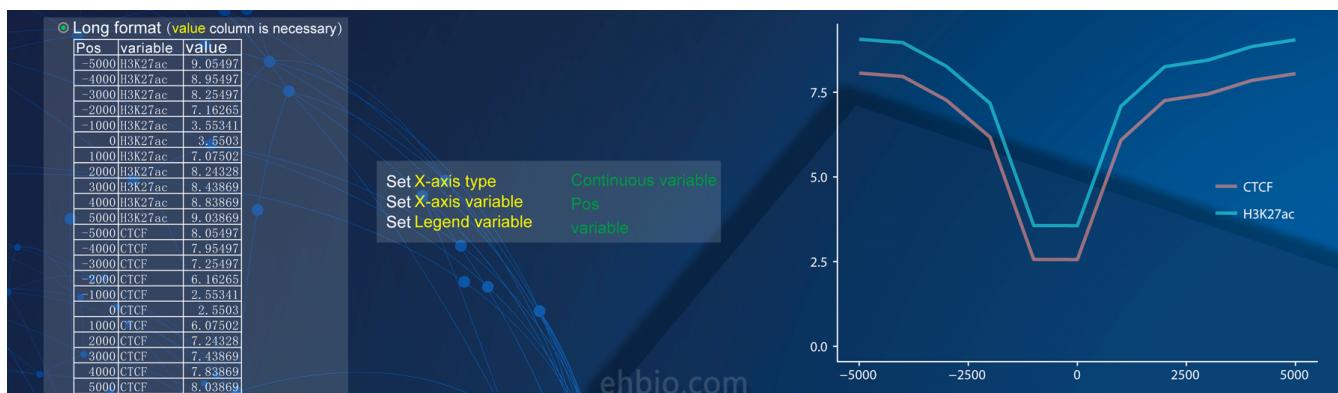


Figure 5.6: 线图

这个图还可以继续美化，比如增加垂线标记启动子区域，横轴改为生物含义的标示，线可在不违反趋势的条件下更圆润一些，更改线的颜色和排列顺序等。

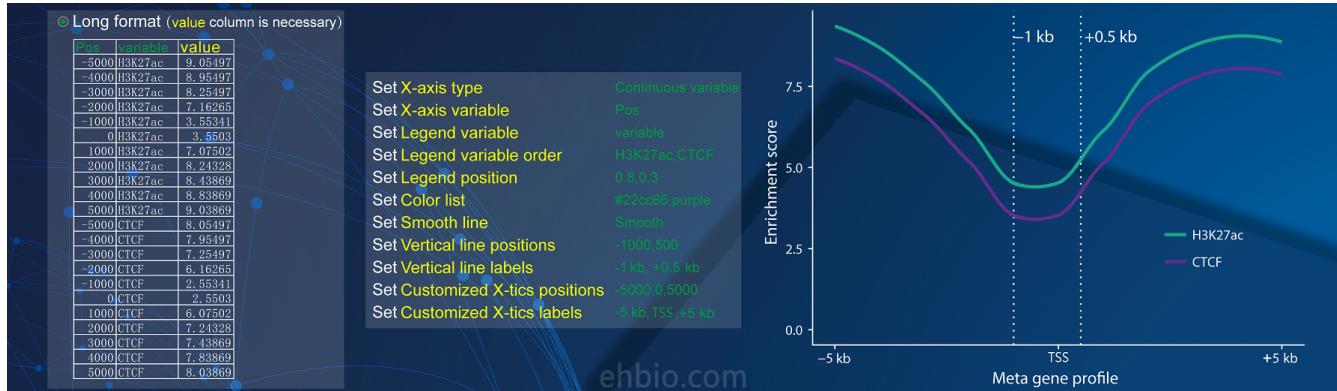


Figure 5.7: 线图

5.2.2 富集分析泡泡图

富集分析泡泡图只需要输入 Long format 的输入格式，也是之前推荐的富集分析工具[去东方](#)，最好用的在线 GO 富集分析工具和[GO、GSEA 富集分析一网打进](#)直接输出的格式。

输入数据后，必须填入的参数为 X-轴类型，X-轴信息所在列名字，Y-轴信息所在列名字，点的颜色变量和大小变量，具体见下图标识 (黄色字体为对应参数，绿色为参数设置的值) 或按网页提示填写即可，直接点击 Plot 就可以出图了。

若只有一组数据，X-axis variable 一般选择 GeneRatio (continuous variable)；若有多组数据，X-axis 一般选择 SampleGroup (用于标识分组的列)。

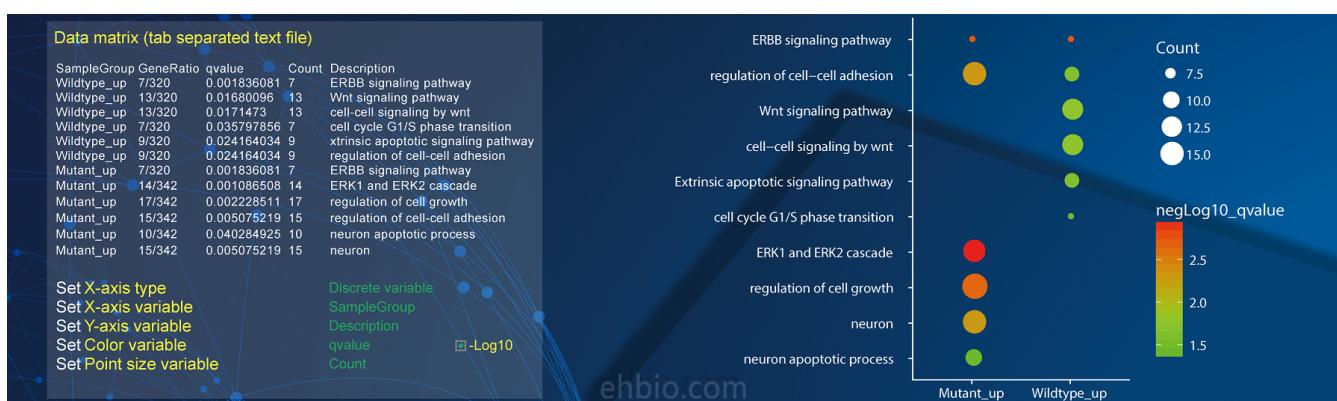


Figure 5.8: 富集分析泡泡图

这是另外一种情况，同时用上了 GeneRatio 作为 X-axis variable，SampleGroup 作为 Shape variable，并且改变了点的颜色等。

CONTENTS

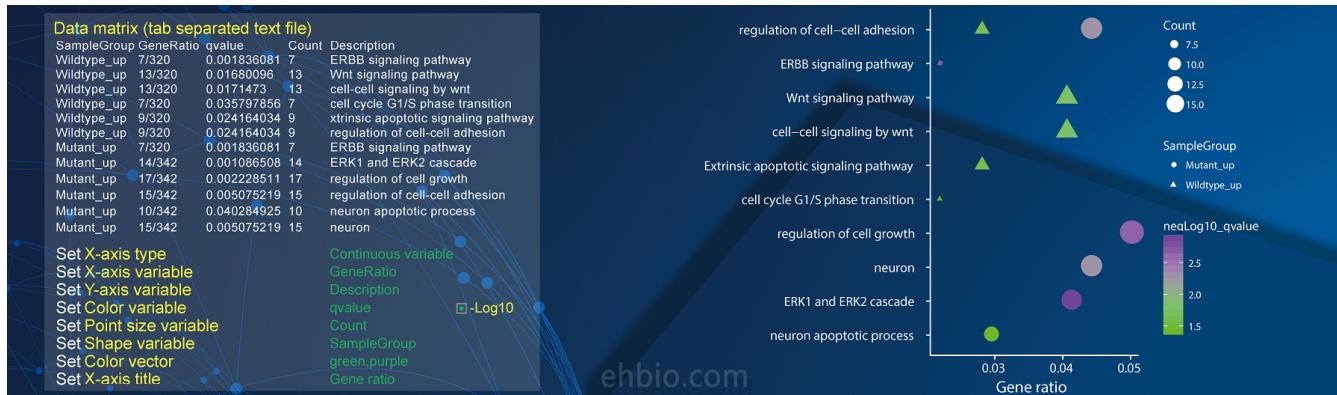


Figure 5.9: 富集分析泡泡图

5.2.3 热图

热图的输入数据主要包括 Data matrix (wide format) 和 Annotation matrix 两部分 (注释不是必须的)。

最简单情况下，只需要输入数据矩阵，点击 Plot 就可以出图了。

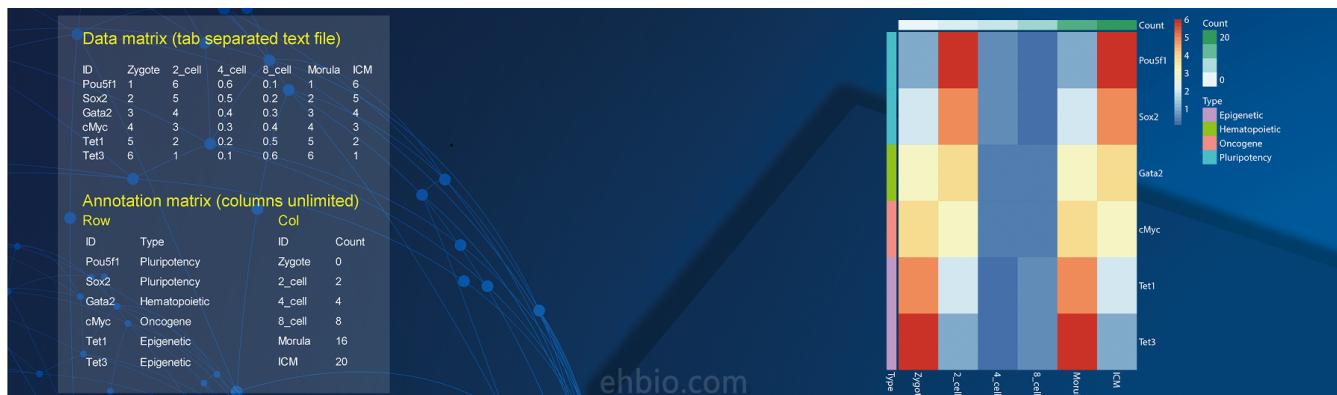


Figure 5.10: 热图

还可以选择是否需要按行或列聚类，X-轴标记倾斜角度，是否进行标准化和更改颜色。



Figure 5.11: 热图聚类

颜色设置有多种方式，可以提供两个边界颜色 (white, blue (colortype 设置为 continuous，表示最小值为白色，最大值为蓝色，生成一系列的色阶))；如果只有有限的数，想单独赋给颜色，可以选择 colorype 为 Discrete。

另外一个重要功能是添加注释和更改注释项的颜色，添加注释比较简单，如果是行注释，那么文件的第一列与 Data matrix 的第一列内容一致；如果是列注释，文件第一列与 Data matrix 的第一行内容一致，其它的列根据分组信息自由标记就好。系统会自动给注释赋值颜色，通常都是足够使用的。

如果想自定义颜色，就复杂一些了，按图中的示例，连续型变量 Count 需要赋值一个颜色向量，表示对 Count 列的值从小到大颜色赋值为 white 到 red。离散型变量 Type，则需要对每个值分别指定颜色。Type 和 Count 都是注释矩阵中的列名字，需要根据需要修改。如果是离散变量，每个值都需要设置对应颜色。

5.2.4 箱线图

只需要输入数据矩阵，指定分组变量 (Legend variable) 和数据列 (Y-axis variable) 就可以了。



Figure 5.12: 箱线图

美化一点，调整下图例位置，box 方向，X、Y、Title 等，Box 类型

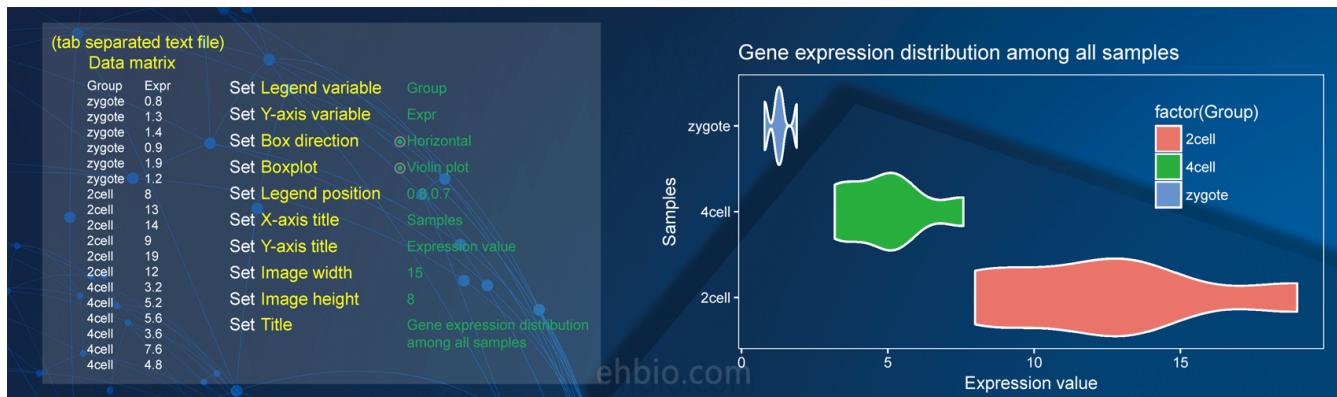


Figure 5.13: 小提琴图横版

如果数据有亚组，则需同时指定横轴变量 (X-axis variable)、亚组变量 (Legend variable)、纵轴变量 (Y-axis variable)。

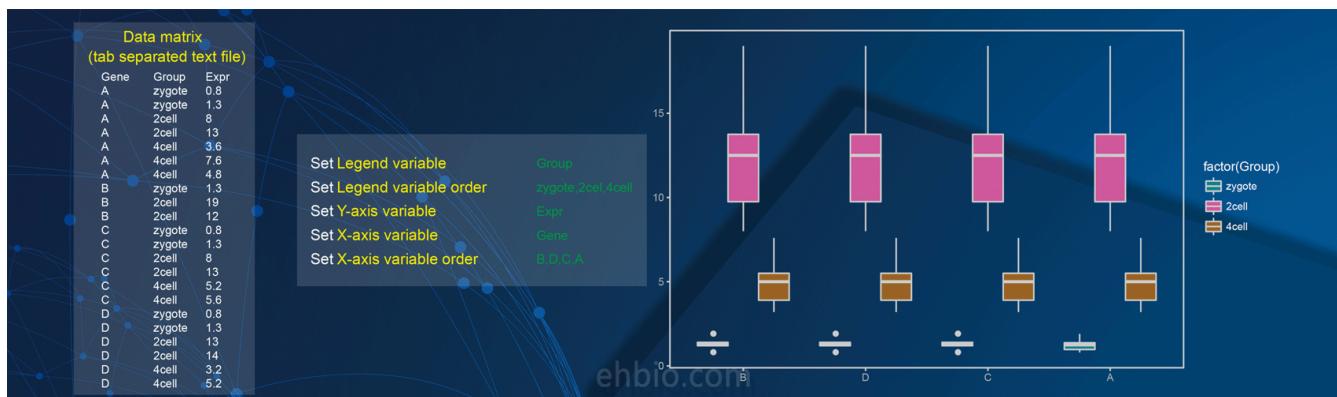


Figure 5.14: 多组箱线图

5.2.5 散点图

比较直观，指定哪一列做横轴、纵轴、用于上色、形状、大小 (大小除了可以是某一列，也可以是一个数字)



Figure 5.15: 散点图

5.2.6 柱状图

只需要输入符合格式的数据，就可以出图了



Figure 5.16: 柱状图



Figure 5.17: 柱状图

如果需要显示 Error-bar，则需要使用 Long format，并且指定哪一列数据是平均值，哪一列是标准差。

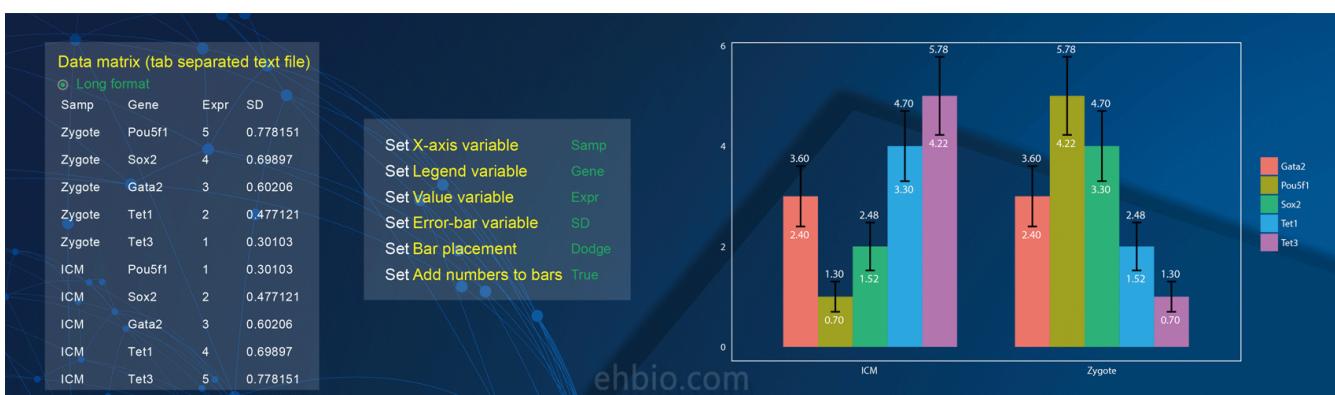


Figure 5.18: 柱状图误差线

5.2.7 火山图

最少需要 2 列数据，设置下阙值，就可以了。



Figure 5.19: 火山图

也可以提供计算好的差异基因，如果需要标记基因，则需提供一列(列内的非'-'都会被标记)



Figure 5.20: 火山图标记基因名字

5.2.8 维恩图

直接上图吧



Figure 5.21: 维恩图

5.2.9 UpsetView

更多集合的维恩图时，推荐使用

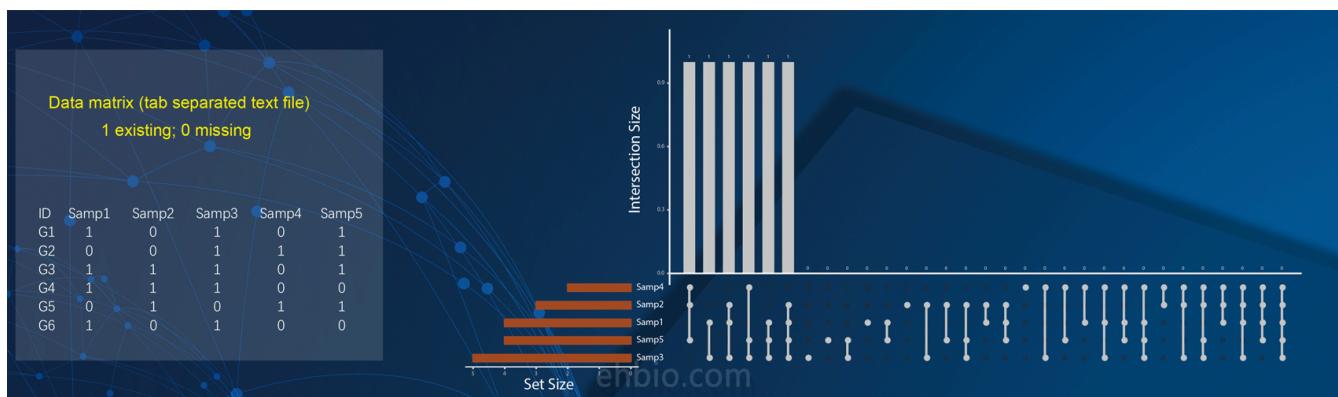


Figure 5.22: UpsetView

5.2.10 共表达密度图

输入常規矩阵就可以

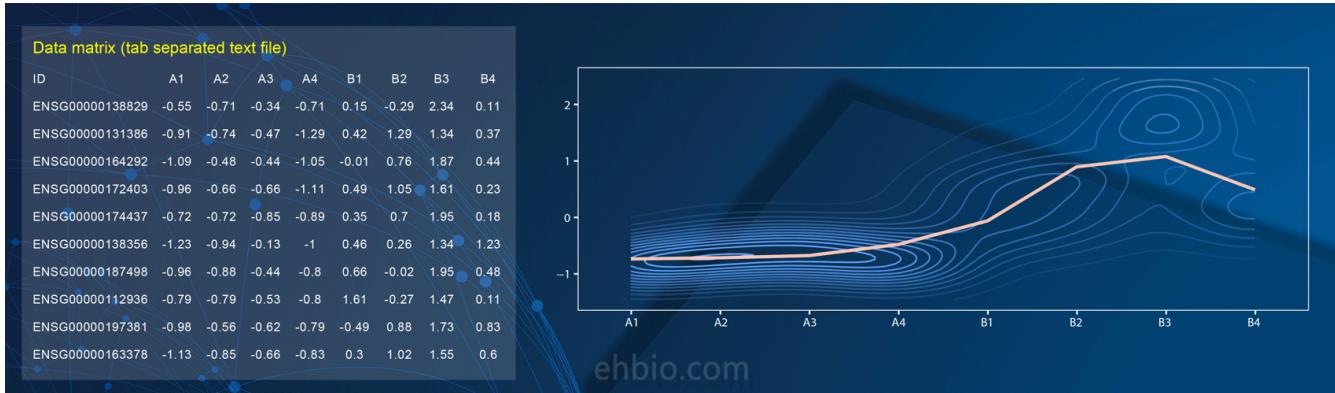


Figure 5.23: 共表达趋势图

5.2.11 直方图

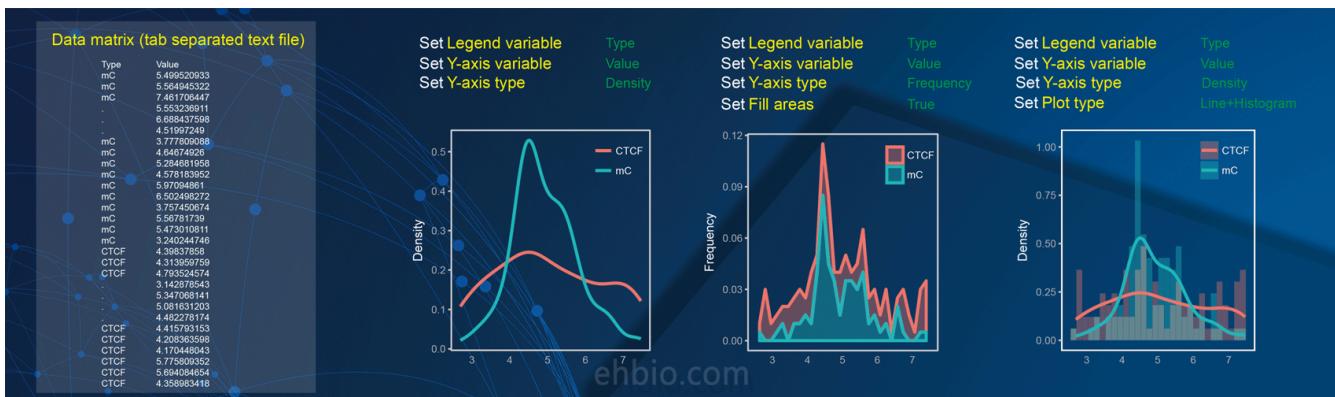


Figure 5.24: 数据分布直方图

5.2.12 PCA

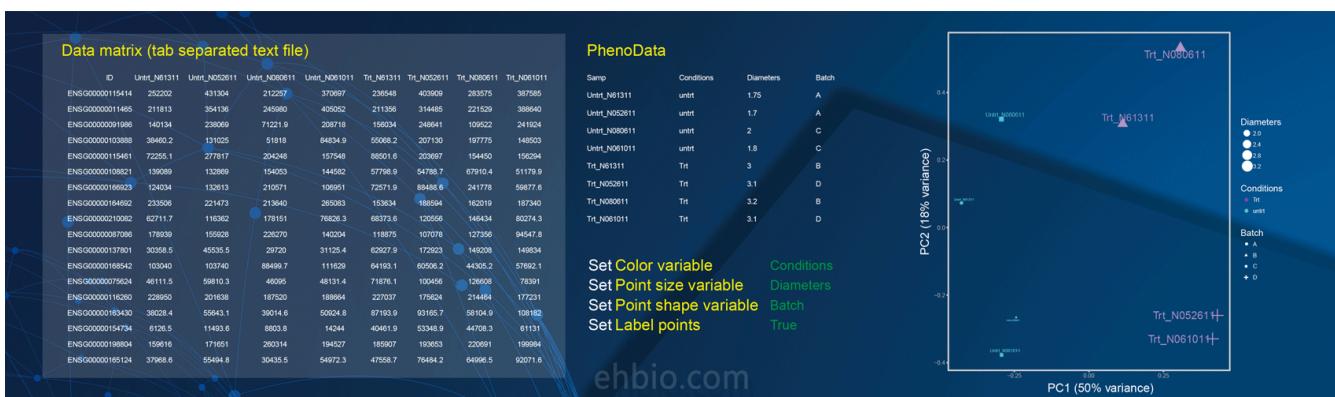


Figure 5.25: 主成分分析

5.2.13 曼哈顿图

与火山图类似，主要是点的属性调整。

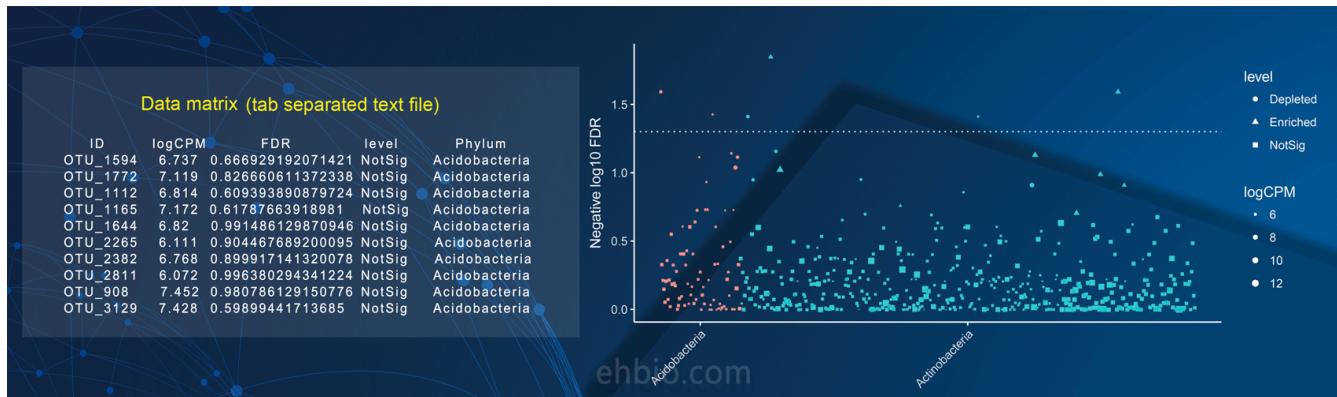


Figure 5.26: 曼哈顿图

5.2.14 PCoA

基于距离矩阵的样品聚类

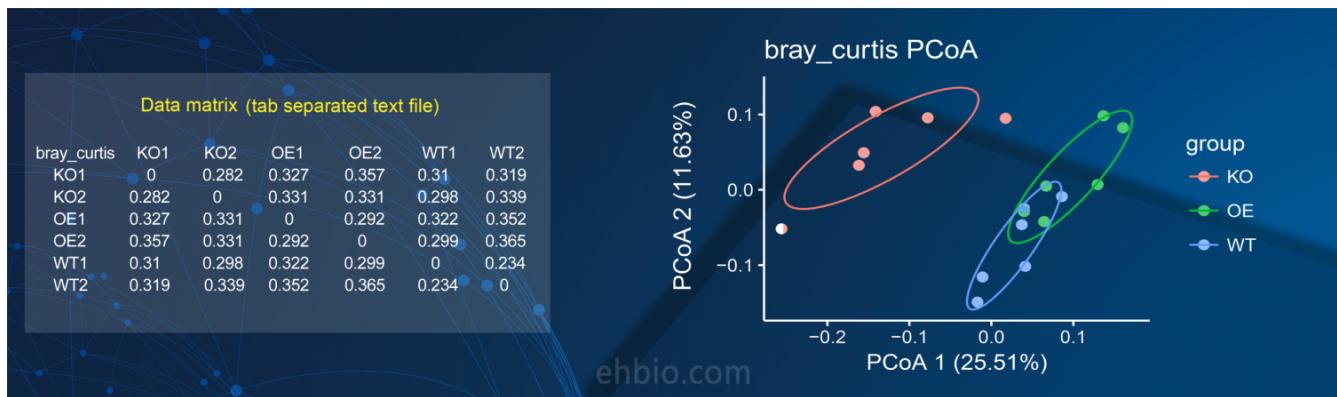


Figure 5.27: 主坐标轴分析

5.2.15 CPcOA

限制性主坐标轴分析，提取最大分类子平面进行展示

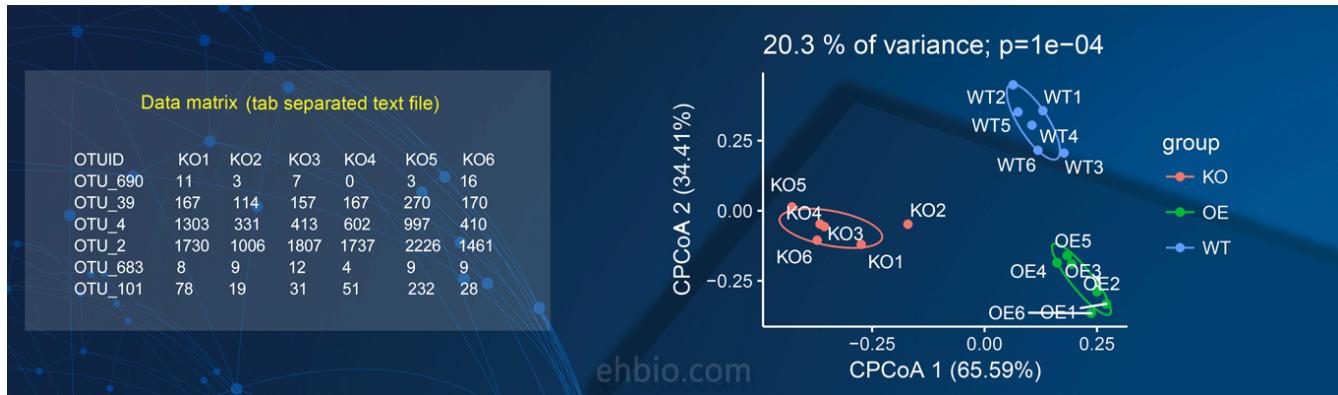


Figure 5.28: 限制性主坐标轴分析

5.2.16 桑基图

一种流图，某种程度上可以视为韦恩图的变种，只需要提供类似 Venn 图的数据就可以。

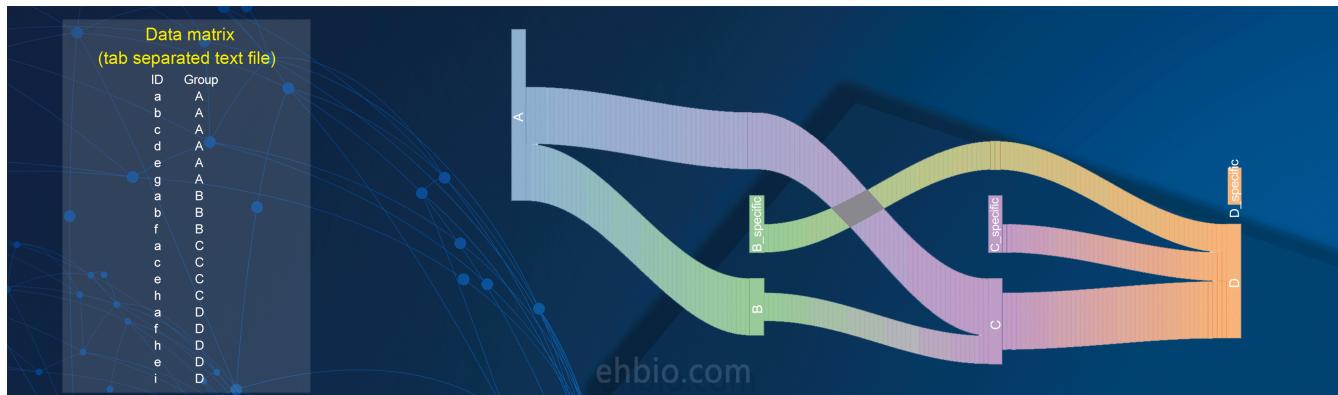


Figure 5.29: 桑基图

5.2.17 R 绘图文章

- 在 R 中赞扬下努力工作的你，奖励一份 CheatSheet
- R 语言学习 - 入门环境 Rstudio
- R 语言学习 - 热图绘制 (heatmap)
- R 语言学习 - 基础概念和矩阵操作
- R 语言学习 - 热图简化
- R 语言学习 - 热图美化
- R 语言学习 - 线图绘制
- R 语言学习 - 线图一步法
- R 语言学习 - 箱线图 (小提琴图、抖动图、区域散点图)
- R 语言学习 - 箱线图一步法
- R 语言学习 - 火山图

CONTENTS

- R 语言学习 - 富集分析泡泡图（文末有彩蛋）
- R 语言学习 - 散点图绘制
- 一文看懂 PCA 主成分分析
- 富集分析 DotPlot，可以服
- R 语言学习 - 韦恩图
- R 语言学习 - 柱状图
- R 语言学习 - 图形设置中英字体
- 教师节献礼 - 文章用图的修改和排版
- R 语言学习 - 非参数法生存分析
- 基因共表达聚类分析和可视化
- R 中 1010 个热图绘制方法
- network3D: 交互式桑基图
- network3D 交互式网络生成

6 网络图

Cytoscape 已成为网络图绘制的核心工具，基因表达调控网络、蛋白互作网络、miRNA-gene 调节关系、分析流程、组织架构等任何与网络、结构、层级有关系的事情都可以用 Cytoscape 来绘制。前期的教程中有[Cytoscape 的基本使用](#)，[早期录制的 Cytoscape 视频教程, 含安装和基本使用](#)，[更新录制的 Cytoscape 视频教程](#)等。

这次主要以实际操作的形式来讲述 Cytoscape 的使用，内容以 PPT 教案为主。

下面是补充内容。

本次三个视频，截图不同的角度。

6.0.1 基本操作

利用一个简单的组织构建图展示 Cytoscape 的基本使用，导入网络、导入节点属性，更改点的形状、颜色，在点上插入图片、绘制饼图、柱状图、加入颜色条纹，点的手动对齐、横向分布等。

网络文件和节点属性文件都是正常矩阵格式，若有中文需使用 UTF-8 编码；若非 EXCEL 文件，不能有 xls 后缀。

https://imgcache.qq.com/tencentvideo_v1/playerv3/TPout.swf?max_age=86400&v=20161117&vid=p0541x250n9&auto=0

6.0.2 miRNA-mRNA 调控网络

Cytoscape 可用于绘制基因共表达网络，这儿选取 miRNA-gene 调控网络（含 miRNA-gene 表达相关性数据）做为例子，涉及到根据表达变化倍数对每个点进行着色、根据表达相关性对线进行着色、miRNA 和靶基因采用不同的形状表示、微调获得合适的展示图形、结果导出 PDF 格式、导出图例等。

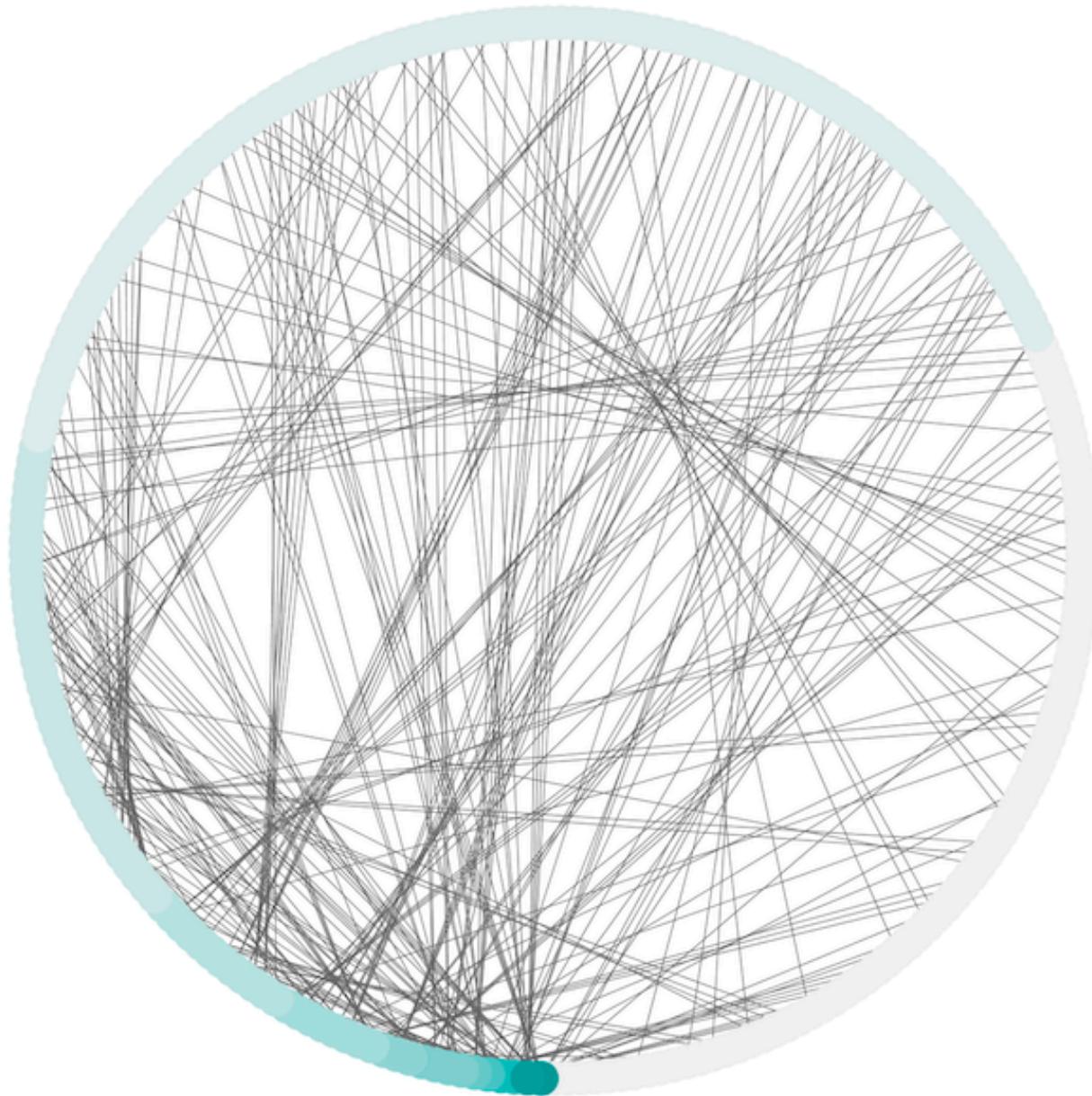
https://imgcache.qq.com/tencentvideo_v1/playerv3/TPout.swf?max_age=86400&v=20161117&vid=z0541oby69q&auto=0

6.0.3 不同的布局的调试和修改

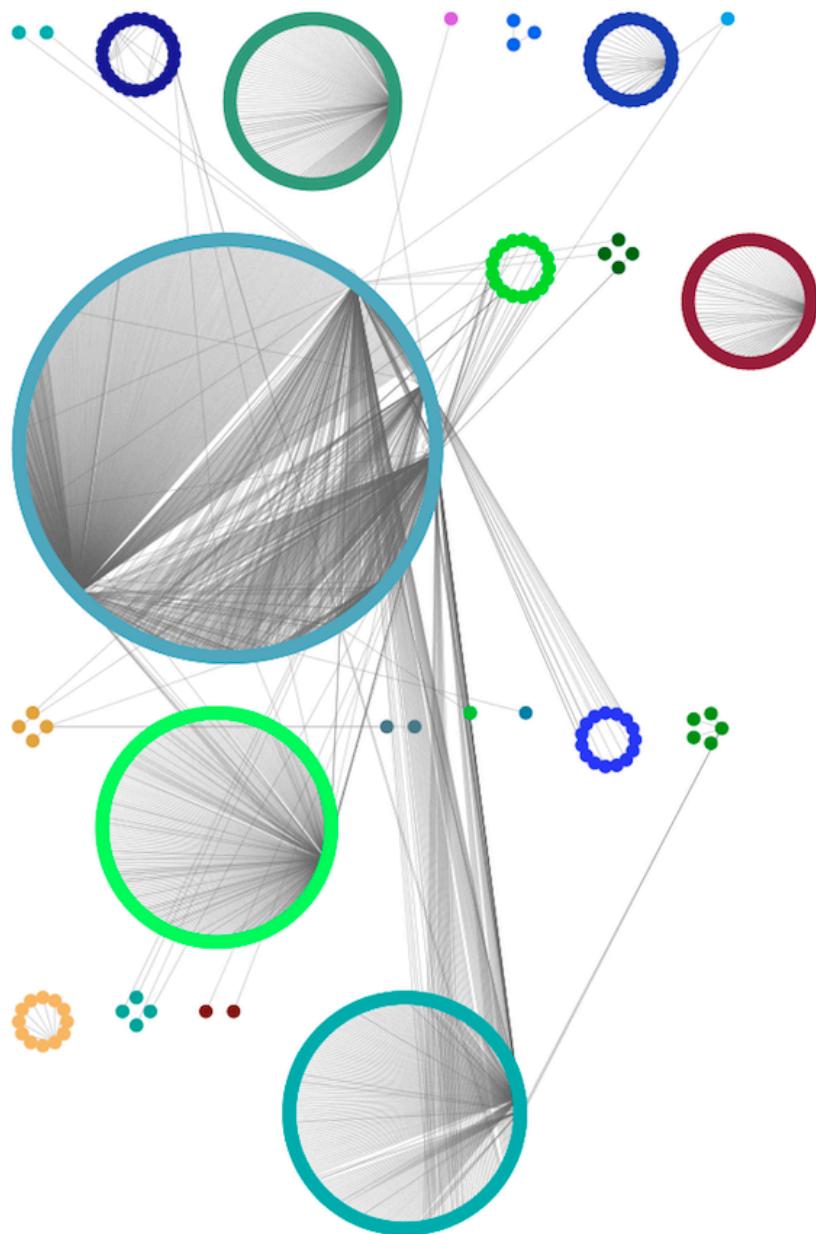
网络绘制根据网络图的大小、展示的用意可以选择合适的布局。Cytoscape 提供了多种布局算法，具体见http://manual.cytoscape.org/en/stable/Navigation_and_Layout.html。

下面列出几种示例

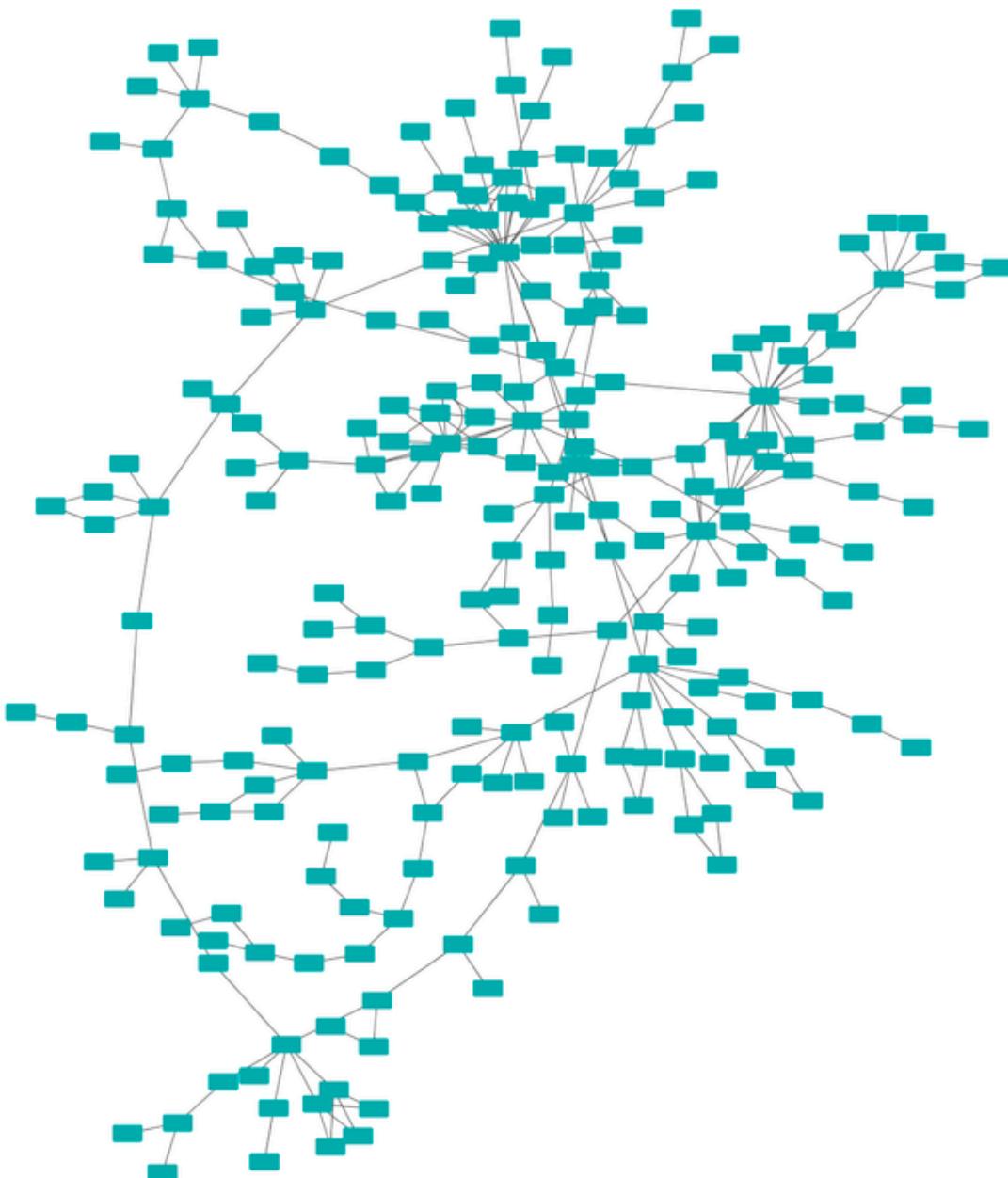
6.0.3.1 Attribute Circle Layout 属性环展示方式所有的节点都在环上，适用于点比较少的时候。环上点的顺序可以根据某一列来调整。



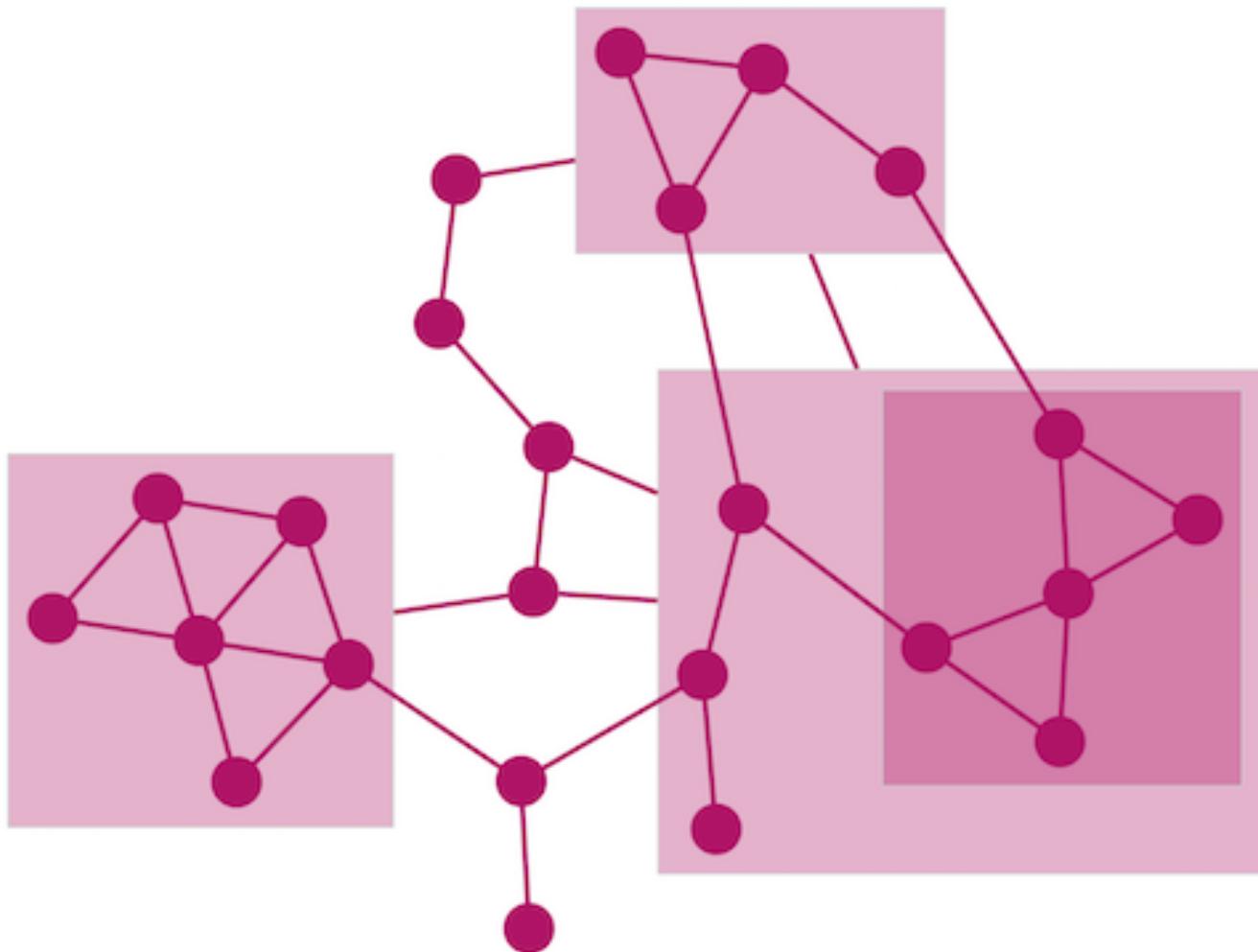
6.0.3.2 Group Attributes Layout 先按某一列的值对点进行分组，每个分组的点成为一个独立的环。在区分上下调基因时，可以按这个来分类，看不同类的基因的调控关系。



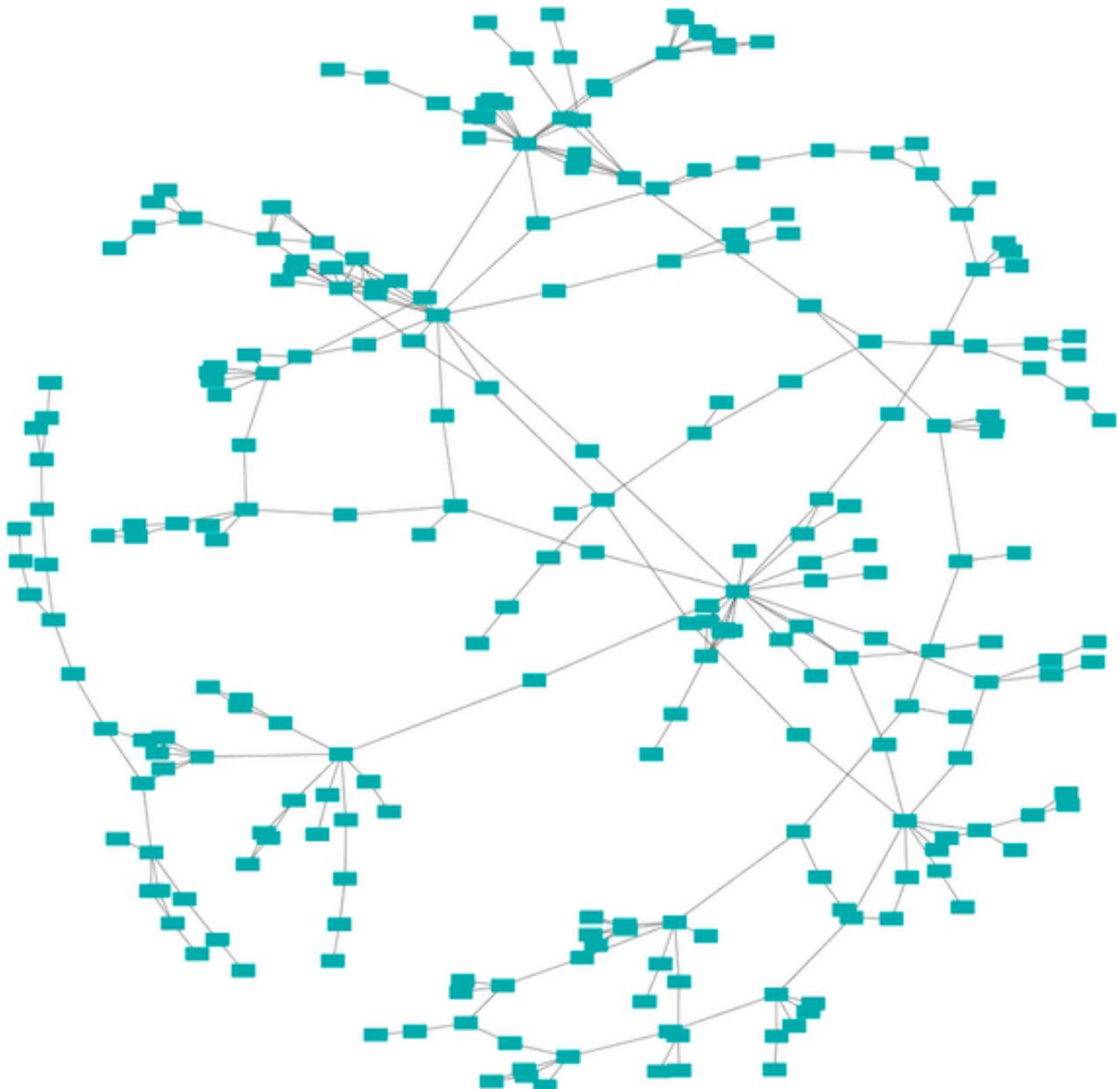
6.0.3.3 Prefuse Force Directed Layout 通常也可以获得比较好的结果，相连的点邻接，其它点较远。同时可以根据某一列设置边的强度作为连线的长度。



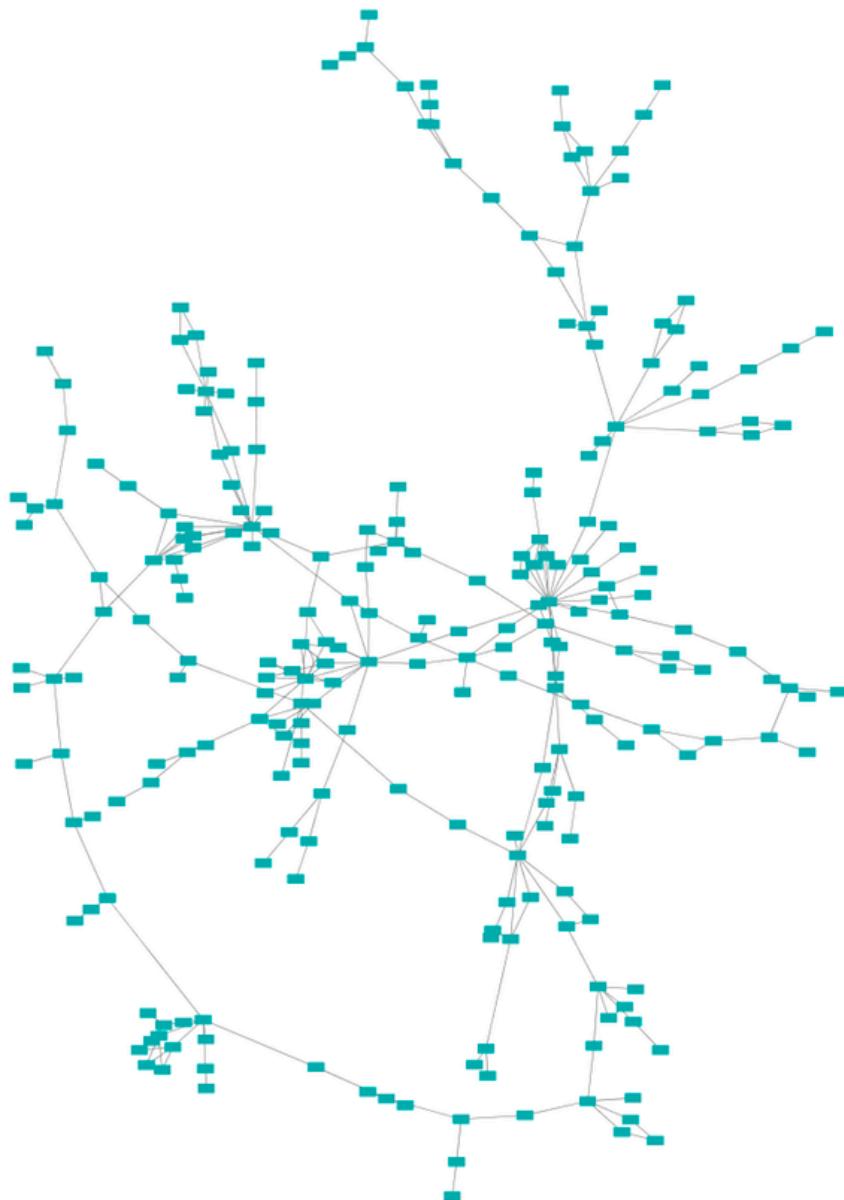
6.0.3.4 Compound Spring Embedder Layout 复杂网络适用这个，尤其是连线特别多时。



6.0.3.5 Edge-weighted Spring-Embedded Layout 所有的边视为连接 2 个节点的弹簧。每一个弹簧有一个松弛状态下的期望长度 (resting length) 和压缩强度。算法会迭代所有的弹簧，并进行仿真模拟获得每个弹簧的最佳长度。是比较常用的一种，其获得结果也是一个环，但对空间的利用比下面两种要好。



6.0.3.6 yFiles Organic Layout 也是一种 Spring-embeded 的算法，可以展示网络图的聚类结构。



https://imgcache.qq.com/tencentvideo_v1/playerv3/TPout.swf?max_age=86400&v=20161117&vid=w05410fbssys&auto=0

6.1 参考

1. [Cytoscape 的基本使用](#)
2. [早期录制的 Cytoscape 视频教程, 含安装和基本使用](#)
3. <http://blog.genesino.com/2012/04/cytoscape-basic-usage/>
4. http://manual.cytoscape.org/en/stable/Navigation_and_Layout.html
5. <https://www.slideshare.net/keiono/introduction-to-biological-network-analysis-and-visualization-with-cytoscape-part1>

6. 新出炉的 Cytoscape 视频教程

7 图形排版

成果发表是科研过程中不可缺的一部分，发表成果又少不了图形展示。文章图表排版是否整齐规范、协调一致、重点突出对一篇文章的发表也是有不少贡献的。此外做科研的人都爱看脸，文章中的图表是重要的颜面之一。生信宝典系列文章中，[R 作图](#)也是受到最多欢迎的一部分。

读文献时，看到文章中的图，就一直好奇是怎么拼出来的，尤其是怎么保证图形中字体的大小一致的。如果是统一用 R 画图，也许可以实现这一点，设置一样的字体、一样的长宽，这样只要图形缩放比例一致，字体理论上也一致。或者使用 cowplot, gridExtra, ggpublisher 等工具也可以组合多个图倒一起（像前面的示例）。但不同的图有不同的边缘设置，实际操作起来，却也不总是顺利。高手们，比如推出[生信宝典傻瓜系列](#)的海哥擅用 PPT 进行修改排版，我学了下，没学会。后来查阅资料，发现有这么一款工具很强大，Adobe Illustrator 简称 AI，翻译成汉语就是人工智能。

著名的 Adobe 公司出品，应该都不陌生。试用后（买不起，也只能用试用版；如果有钱，还是推荐购买正版，维护原作者的权益），发现果然很强大。

7.0.1 矢量图和标量图

矢量图是使用直线和曲线来描述图形，这些图形的元素是一些点、线、矩形、多边形、圆和弧线等等，它们都是通过数学公式计算获得的。矢量图形最大的优点是无论放大、缩小或旋转等不会失真；最大的缺点是难以表现色彩层次丰富的逼真图像效果。常见的矢量图有 PDF, SVG, EPS 等格式。如果图形中有文字，并且文字可以复制，则可初步判断为矢量图。

矢量图的任何一个地方都可以挑出来修改，某个边框不好看，删掉；线条的粗细不统一，设置成一样的；某条线的颜色想重点突出下，单独修改（这部分最好还是在画图时就修改好，会更协调，当然 AI 也没问题）；图中有了多余的元素，删掉；添加文字、设置成统一字体、统一大小更没问题；画个简单的模式图，没问题；不同的子图拼在一起，没问题；自此，再也不愁文章的拼图了。

与矢量图相对应的就是标量图了，常见的 png, jpg, gif 格式等，是由像素点构成，放大到一定程度会出现马赛克效果。图中的文字不可复制，元素不可拆分。

7.0.2 矢量图的制作

- 常规图：

- Excel，生成的图可以直接拷贝到 AI 里面修改
- R, Perl, Python 等程序语言输出 pdf, eps 格式的图（详见公众号中[R 作图系列](#)）

- 常用工具的出图

- 二代测序出图 UCSC – PDF; IGV – SVG; epigenomegateway - SVG; 在高通量数据可视化文章中也有介绍
- Motif WebLogo - eps
- 作图软件 Graphpad

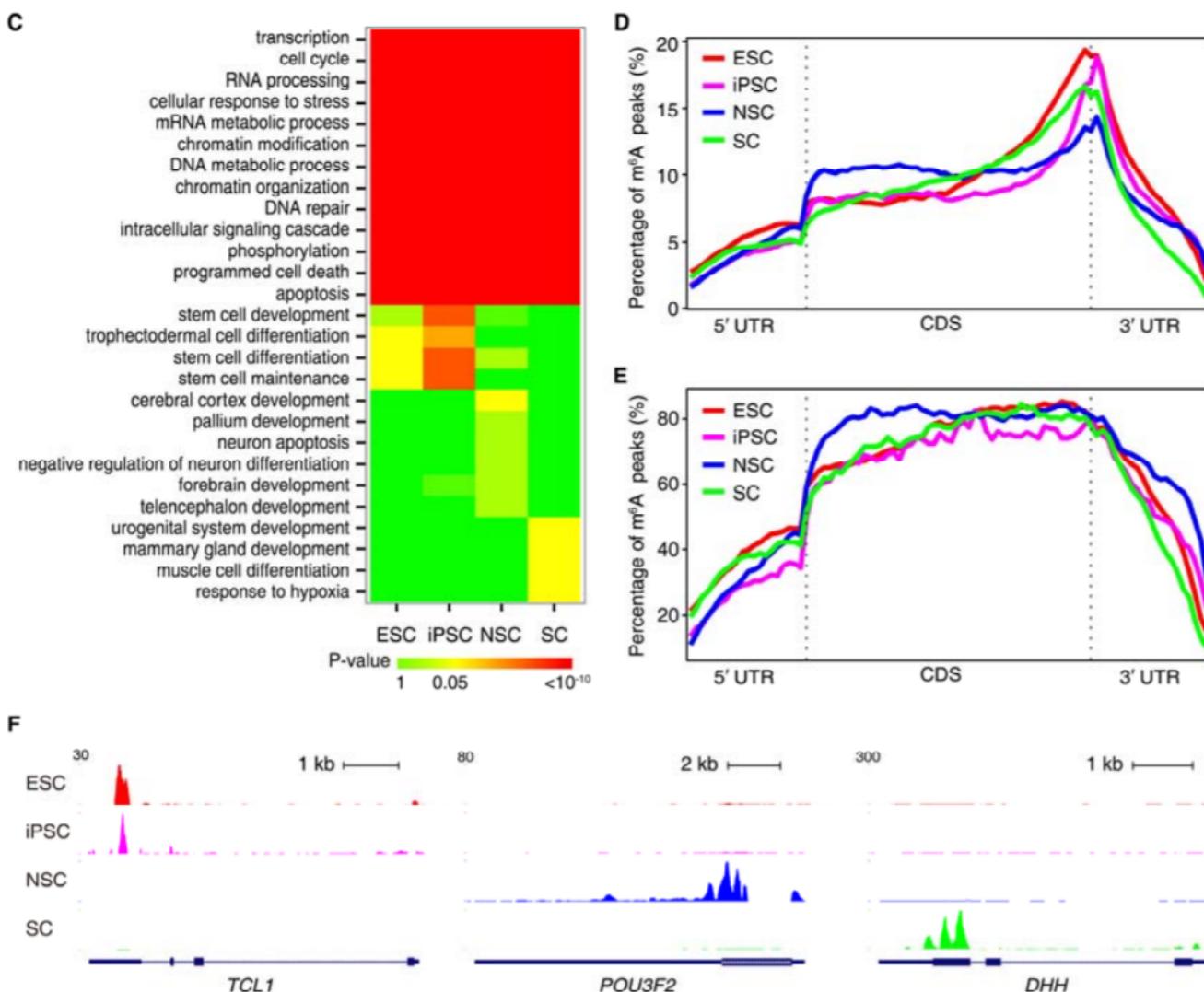
CONTENTS

7.0.3 矢量图编辑工具

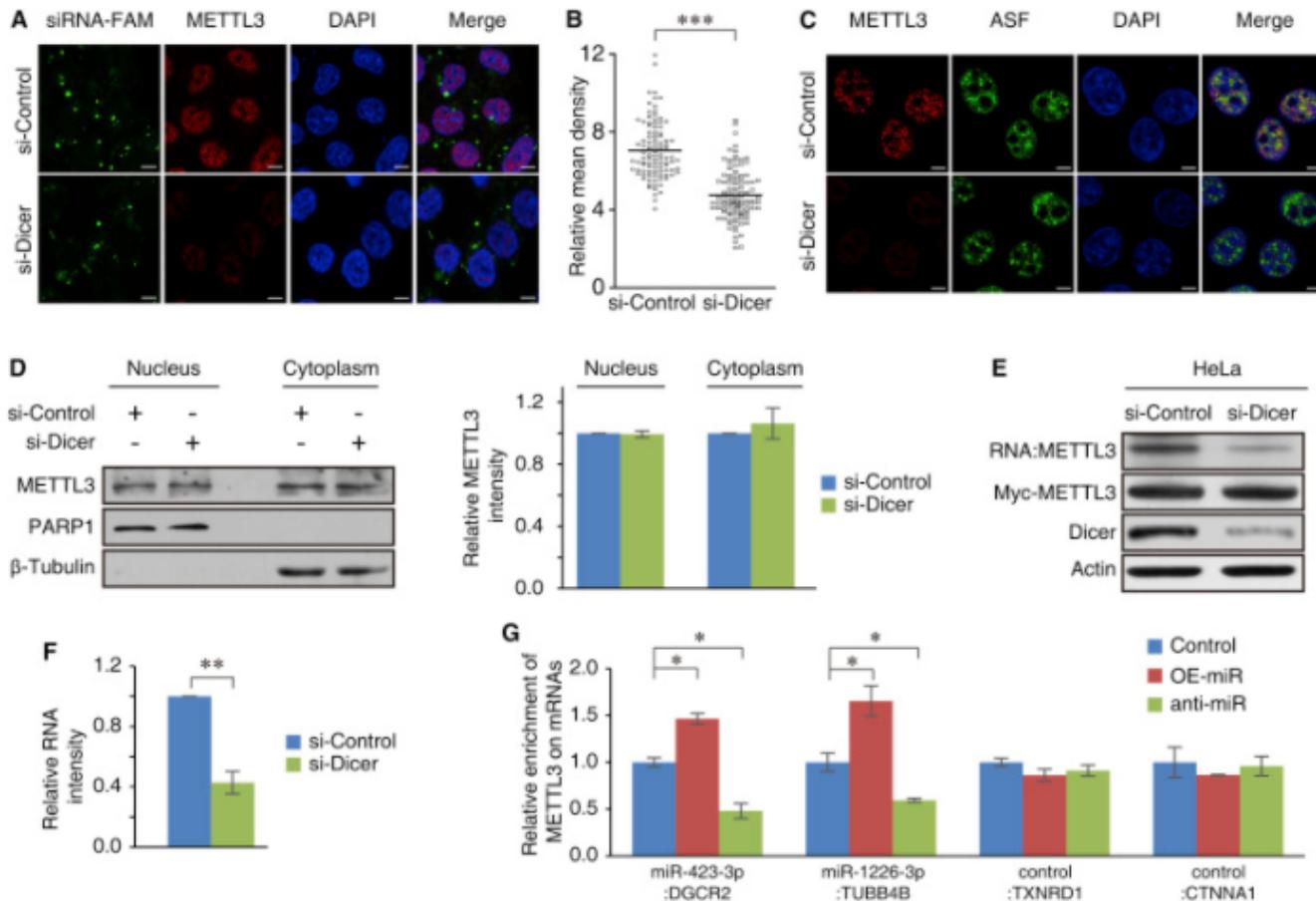
主要有 Gimp, Adobe illustrator, Inkscape, image magik, photoshop, latex。适用之后, 从稳定性还是易用性来讲, Adobe illustrator 是最好的一款。但是是收费软件, 在线会有一些试用版, 供测试时用。

7.0.4 作图基本原则

- 图形中文字的字体保持统一, 一般使用 Helvetica 或 Arial
- 符号一般使用 Symbol 字体, 常见符号有 ', β 等
- Panel 的字号 (A,B,C,D) 一般比其余的文字大一号, 上下左右对齐
- 文字特别密集的地方字体可适当缩小, 原则是看着协调
- 图和图之间的距离在空间允许的情况下尽可能的大



- 一篇文稿所有柱状图理论上柱子的宽度保持一致
- 柱状图的 Error bar 宽度一致
- 坐标轴上的刻度尺宽度, 长度一致
- 坐标轴的宽度、颜色一致
- 胶图的泳道对齐



7.0.5 作图中的要点注意

- 标准化, 便于位置调整
 - 从最开始作图, 到文章投稿、修改、定稿, 中间会不断调整, 子图会根据文章需要不断删减, 调整位置。因此标准化之后, 就可以很简单的互换位置就可以了
 - 每个子图的长宽尽量一致
 - 每个相似子图内部元素的特征一致, 比如柱子的宽度 (6 mm), 柱子之间的距离, 坐标轴的刻度的宽度 (0.7 mm), 误差线的宽度 (1 mm), P-value 连接线宽度 (6 mm), 胶图泳道的宽度等
- 合理利用对齐工具, 左右对齐、横向分布、纵向分布等, 即保证对齐效果, 又免去人为调整的繁琐
- 在选择单个元素时尽量使用直接选择工具
- 作图要做到自己满意, 自己对自己负责; 当你觉得一个地方不合适需要调整时, 一定要及时修改; 如果怕麻烦现在没调整, 过几天别人发现或自己觉得不舒服也还是会再调整的。所有要做好充足的准备和充足的工作

- 保留备份, 保留备份, 保留备份. 每次大的修改都要保留原始版本, 因为不知道明天是否还会改回来

7.0.6 Adobe Illustrator 中的基本概念和操作描述

1. 编组 : 性质相似或者需要同时修改的部分可以编为一组, 方便处理。双击一组内容, 就可以进入编组内部, 对编组的每个元素修改; 并且编组外的元素处于屏蔽状态, 操作起来不会受到干扰。
2. 剪切蒙版 : 如果想剪切掉图中的某一部分, 可以绘制一个矩形、圆形或任意不规则形状覆盖住需要保留的部分, 然后同时选中这两个元素(绘制的形状框在被剪切的图之上), 按右键, 选择剪切蒙版, 就可以完成剪切操作。而在修改图时, 也可以不断的释放剪切蒙版, 方便对不同图层的操作。
3. 直接选择工具 : 可以无视编组和剪切蒙版, 对选中并且只是选中的部分进行操作。这在删除多余的内容和边框时会经常用到。
4. 魔棒工具 : 选择类似属性的组分, 统一操作。
5. 吸管工具 : 给一个组分赋予另一个组分的属性。
6. 对齐工具 : 用于组分的对齐和分布, 在设置坐标轴的标记文字时很有用, 省去了一个个手动对齐的操作。只要对齐两端, 按一下按钮中间的内容就自动与刻度线对齐了。
7. 其它的就靠大家不断的尝试、体验、操作了。多选、多点、多查, 慢慢就都熟练了。

另外翻阅到之前准备文章期间做的图形排版教程。当时每天都在不断的调整图和排版, 心得体会比现在更多些, 也录制了视频, 但是无声版, 录制完之后为 EXE 格式。如果您已观看过[文章用图的修改和排版](#)视频教程, 也自己操作过, 有了一定基础, 还想进一步的了解, 可以看看这份无声版的记录是否能有些帮助。

7.0.7 视频教程

第一个视频以[绘制的线图](#)为例, 展示如何修改、调整矢量图的每个部分。ggplot2 出品的矢量图整体逻辑比较清晰, 一层层的叠加, 修改起来也比较方便, 没有太多难点; 关键是熟悉用到的按钮的使用方式和快捷键的操作。

https://imgcache.qq.com/tencentvideo_v1/playerv3/TPout.swf?max_age=86400&v=20161117&vid=m0548j1ufql&auto=0

第二个视频以[UCSC 绘制的测序峰图](#)为例, 展示如何对稍微复杂一些的图进行修改、删除多余内容。

https://imgcache.qq.com/tencentvideo_v1/playerv3/TPout.swf?max_age=86400&v=20161117&vid=z0548i168ie&auto=0

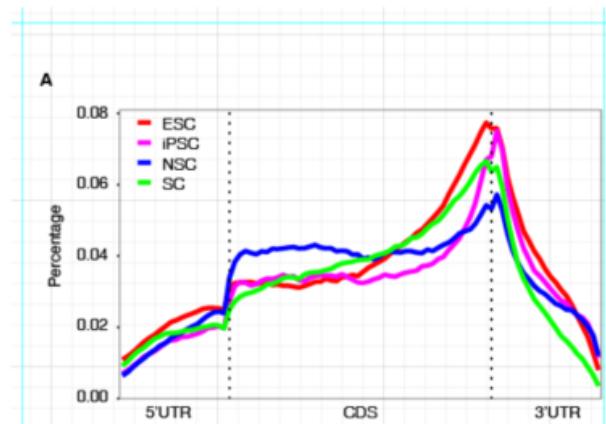
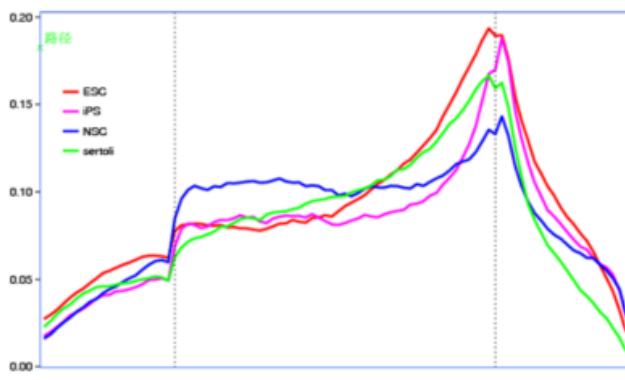
第三个视频是剪切蒙版的使用和针对其它不同类型图的特殊操作。

https://imgcache.qq.com/tencentvideo_v1/playerv3/TPout.swf?max_age=86400&v=20161117&vid=e0548h4prm2&auto=0

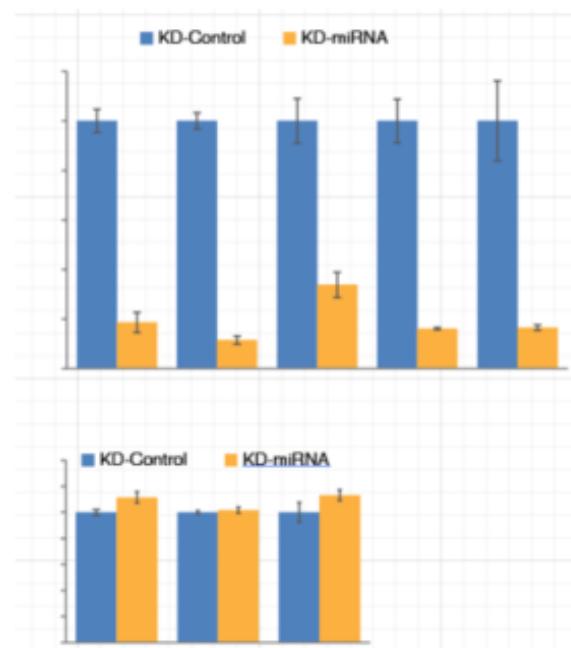
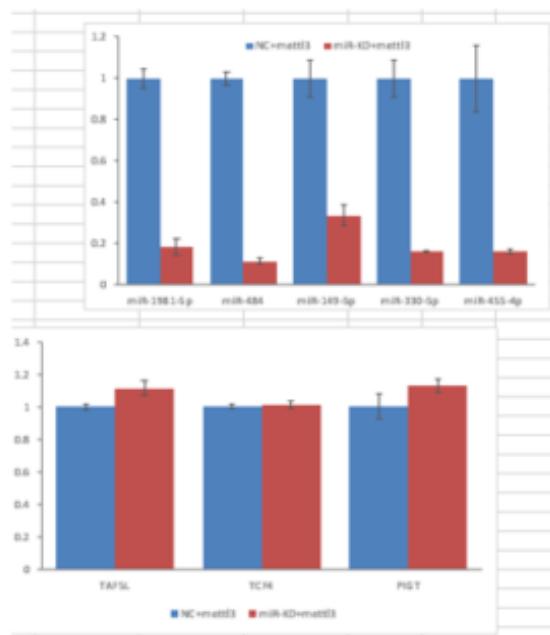
7.0.8 动图教程

左边是修改前的，右边是成品或近成品截图。

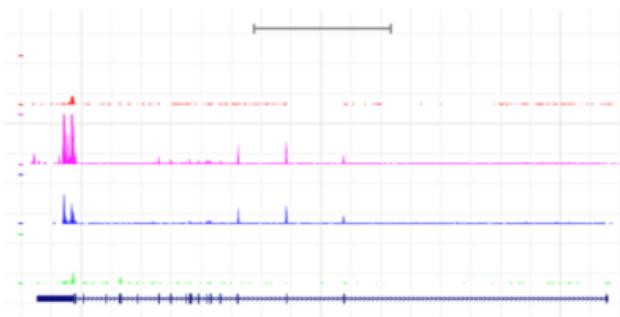
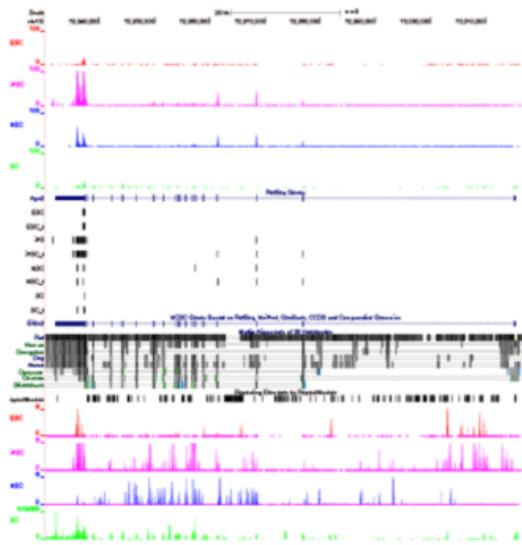
1. AI 编辑线图.exe



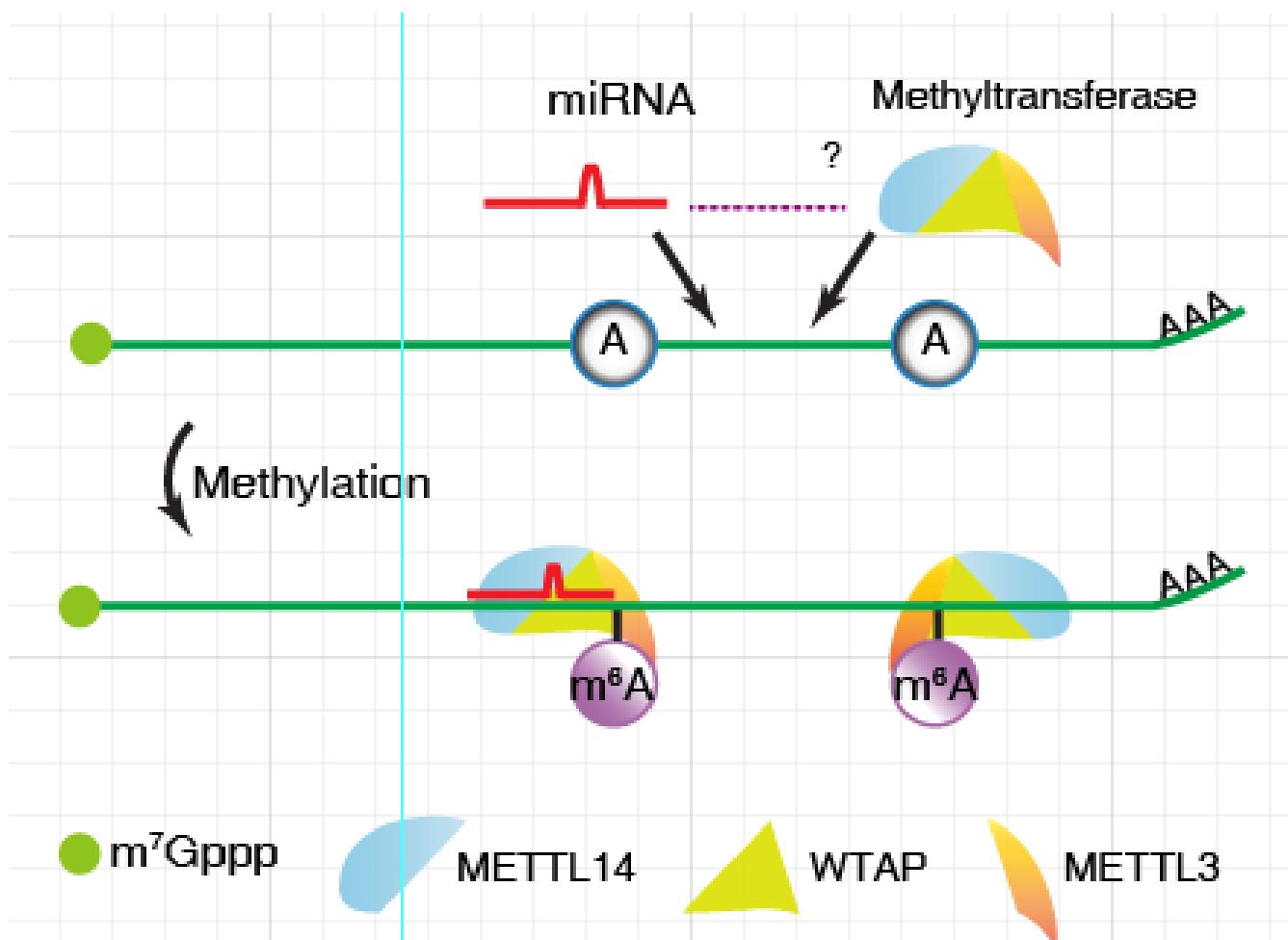
2. AI_柱状图.exe



3. AI_UCSC.exe



4. AI_设计.exe



5. 图形导出

可以导出为 PDF、TIFF 格式；最好用 **PhotoShop** 导出，分辨率高，文件小。

6. 视频文件统一下载地址：链接: <https://pan.baidu.com/s/1dFcWadV> 密码: (b4mw)

7.1 参考

1. 教师节献礼 - 文章用图的修改和排版
2. 简单强大的在线绘图
3. 文章用图的修改和排版 (二)
4. 论文图表基本规范

8 参考

8.1 程序学习心得

- 生物信息之程序学习
- 如何优雅的提问
- 生信宝典视频教程
- 好色之旅-画图三字经
- 转录组分析的正确姿势
- 学习生信的系列教程
- 学习生信的系列书籍
- 教师节献礼 - 文章用图的修改和排版
- 简单强大的在线绘图

8.2 NGS 分析工具评估

- 39 个转录组分析工具，120 种组合评估 (转录组分析工具哪家强-导读版)
- 39 个转录组分析工具，120 种组合评估 (转录组分析工具大比拼 (完整翻译版))
- 无参转录组分析工具评估和流程展示

8.3 文献精读

- 一场大病引起的诺贝尔 2017 年生理学奖角逐

8.4 Linux 学习

- Linux 学习-总目录
- Linux 学习-文件和目录
- Linux 学习-文件操作
- Linux 文件内容操作
- Linux 学习-环境变量和可执行属性
- Linux 学习 - 管道、标准输入输出
- Linux 学习 - 命令运行监测和软件安装
- Linux 学习-常见错误和快捷操作
- Linux 学习-文件列太多，很难识别想要的信息在哪列；别焦急，看这里。
- Linux 学习-文件排序和 FASTA 文件操作
- 用了 Docker，妈妈再也不担心我的软件安装了 - 基础篇

CONTENTS

- Linux 服务器数据定期同步和备份方式
- 不用 Linux 也可以的强大文本处理方法
- Linux 学习 - 又双叒叕一个软件安装方法
- 查看服务器配置信息
- Linux 学习 - SED 操作 , awk 的姊妹篇
- Linux 学习 - 常用和不太常用的实用 awk 命令

8.5 R 统计和作图

- R 语言学习 - 入门环境 Rstudio
- R 语言学习 - 热图绘制 (heatmap)
- R 语言学习 - 基础概念和矩阵操作
- R 语言学习 - 热图简化
- R 语言学习 - 热图美化
- R 语言学习 - 线图绘制
- R 语言学习 - 线图一步法
- R 语言学习 - 箱线图 (小提琴图、抖动图、区域散点图)
- R 语言学习 - 箱线图一步法
- R 语言学习 - 火山图
- R 语言学习 - 富集分析泡泡图 (文末有彩蛋)
- R 语言学习 - 散点图绘制
- 一文看懂 PCA 主成分分析
- 富集分析 DotPlot , 可以服
- R 语言学习 - 韦恩图
- R 语言学习 - 柱状图
- R 语言学习 - 图形设置中英字体
- 教师节献礼 - 文章用图的修改和排版
- R 语言学习 - 非参数法生存分析

8.6 NGS 基础

- NGS 基础 - FASTQ 格式解释和质量评估
- NGS 基础 - 高通量测序原理
- NGS 基础 - 参考基因组和基因注释文件
- NGS 基础 - GTF/GFF 文件格式解读和转换
- 本地安装 UCSC 基因组浏览器
- 测序数据可视化 (一)
- 测序文章数据上传找哪里
- GO、GSEA 富集分析一网打进
- GSEA 富集分析 - 界面操作

8.7 癌症数据库

- UCSC XENA - 集大成者 (TCGA, ICGC)
- ICGC 数据库使用
- TCGA 数据库在线使用

8.8 Python 学习

- Python 学习极简教程 (一)
- Python 学习教程 (二)
- Python 学习教程 (三)
- Python 学习教程 (四)
- Python 学习教程 (五)
- Python 学习教程 (六)
- Pandas , 让 Python 像 R 一样处理数据 , 但快
- Python 解析 psiBlast 输出的 JSON 文件结果
- 为啥我的 Python 这么慢 - 项查找 (二)
- 为啥我的 Python 这么慢 (一)

8.9 NGS 软件

- Rfam 12.0+ 本地使用 (最新版教程)
- 轻松绘制各种 Venn 图
- ETE 构建、绘制进化树
- psRobot : 植物小 RNA 分析系统
- 生信软件系列 - NCBI 使用
- 去东方 , 最好用的在线 GO 富集分析工具

8.10 Cytoscape 网络图

- Cytoscape 教程 1
- Cytoscape 之操作界面介绍
- 新出炉的 Cytoscape 视频教程

8.11 分子对接

- 来一场蛋白和小分子的风花雪月

CONTENTS

- 不是原配也可以-对接非原生配体
- 简单可视化-送你一双发现美的眼睛
- 你需要知道的那些前奏

8.12 生信宝典之傻瓜式

- 生信宝典之傻瓜式 (一) 如何提取指定位置的基因组序列
- 生信宝典之傻瓜式 (二) 如何快速查找指定基因的调控网络
- 生信宝典之傻瓜式 (三) 我的基因在哪里发光 - 如何查找基因在发表研究中的表达

8.13 生信人写程序

- 生信人写程序 1. Perl 语言模板及配置
- 生信人写程序 2. Editplus 添加 Perl, Shell, R, markdown 模板和语法高亮

8.14 小技巧系列

- 参考文献中杂志名字格式混乱问题一次解决

8.15 招聘

- 易汉博欢迎您加入

9 公司简介



易汉博基因科技(北京)有限公司

公司简介

易汉博基因科技（北京）有限公司由中国科学院及武汉大学等高校的优秀博士、硕士毕业生创建，致力于利用最前沿与最精确的生物信息分析技术，帮助解析基础研究和临床应用相关的高通量数据的专业数据服务公司。

易汉博技术团队在生物信息领域具有很高的造诣，团队成员以科学问题为导向，依托精湛的生物信息分析技术和丰富的实验生物学设计经验，在攻读学位或博士后进修期间获得了多个创新性研究成果，并发表于 *Cell*, *PNAS*, *JBC*, *NAR*, *New Phytologist*, *Bioinformatics* 等杂志，其中创始人陈博士发现并协调验证的 miRNA 调控 RNA 甲基化修饰 m⁶A 形成的全新作用机制的工作更以封面文章的形式发表于干细胞领域顶尖杂志 *Cell Stem Cell* 中。这一工作在解析 m⁶A 修饰形成的位点选择机制、拓展 miRNA 的新功能和发现新的细胞重编程调控因素方面均取得了开创性的成果。

易汉博为您提供最具性价比的标准生物信息服务和科研合作式的定制化生物信息分析和数据解读服务，帮助您充分地利用生物大数据来促进新的科学发现。

因为专注，所以专业。让易汉博同您一起，领略生命之美，护航健康生活。

业务范围

- 转录组数据分析
- lncRNA 测序分析
- smRNA 测序分析
- 单基因功能注释
- IncRNA 鉴定分析
- 基因组测序分析
- 生物网络分析
- 芯片数据分析
- ChIP-seq 数据分析
- 生物数据定制分析

服务特色



超值的标准分析

周期短
宽度广
价格低

生物信息专业博士
国家重大项目经验
国际一流科研视角



邮箱: vip@ehbo.com

电话: 010-51732515 15210822685

网址: www.ehbio.com

地址: 北京市西城区德胜门外大街 11 号 1 檐 220



深入的定制分析 精准的结果呈现

科研问题为导向
数据模式为依据
合作交流为模式

信得过的分析
看得懂的结果
用得上的图文

联系我们