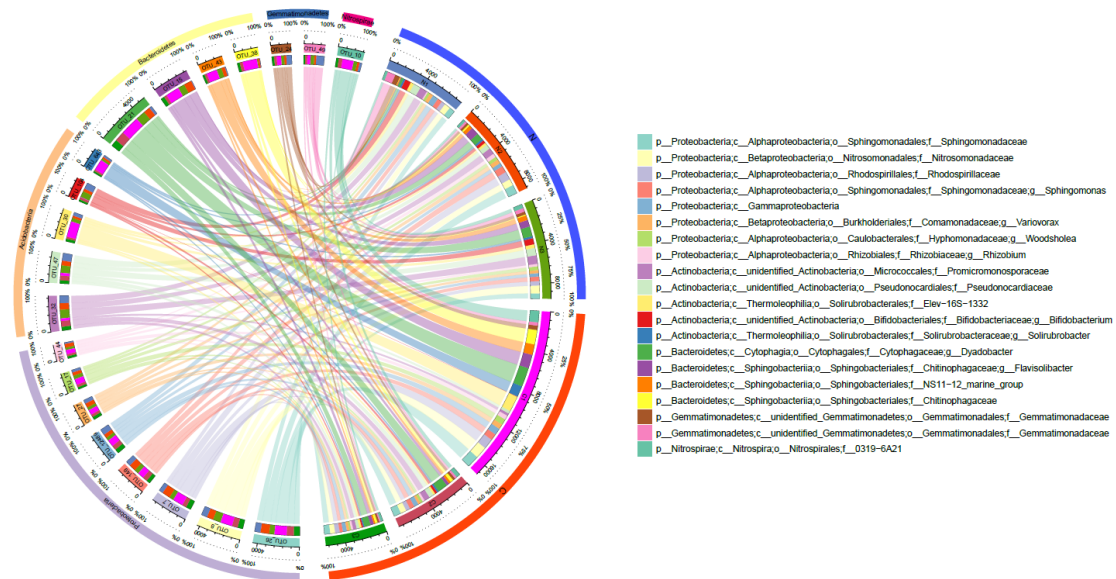


## R 作图-使用 circlize 绘制样本\_物种丰度关联弦状图

此处以样本-OTU 丰度的包含关系为例，与大家分享一例 circlize 绘制 circos 图的思路及 R 脚本。

作图示例文件、R 脚本（分为 UTF-8 以及 ANSI 两种编码格式，请自行选择）



该 circos 图展示了 20 种 OTU 在 6 个样本中的丰度关系。其中，20 种 OTU 可划分为 5 个门，6 个样本可划分为 2 个分组。

左侧为 circlize 做图结果，分为 5 圈。

第一圈，各 OTU 的门水平分类以及各样本的分组信息；

第二圈，OTU 相对丰度的百分比信息；

第三圈，OTU 及样本主区块，以不同颜色和标签区分，区块外周的刻度为 OTU 的绝对丰度信息；

第四圈，OTU 及样本副区块，与主区块（第三圈）对应，展示了各 OTU 在各样本中的丰度，以及各样本所含每种 OTU 的丰度信息；

第五圈，与 OTU 及样本副区块（第四圈）相对应，连线展示 OTU、样本关联信息。

右侧为图例（额外添加，非 circlize 作图所得，下文细说），展示了各 OTU（本示例中共计 20 种 OTU）的详细分类详情。

## 文件介绍

首先准备 3 个表格（以上传至百度盘中，内容均以 tab 分隔），分别为：

（1）物种分类信息文件（'taxonomy.txt'）。第一列为物种 ID（此处为 OTU，在 circos 图的中间区域展示），第二列为 OTU 的“门水平”分类信息（用于绘制 circos 外圈 OTU 分类），第三列为 OTU 分类详细信息（用于在图例中展示）。此外在后面的绘图过程中，会使用该文件确定 OTU 的排列顺序，所以请事先根据想要展示的顺序给该文件中的 OTU 进行排序。

（2）样本分组信息文件（group.txt）。第一列为样本 ID（此处为 N1、N2 等共计 6 个样本，在 circos 图的中间区域展示），第二列为样本分组信息（用于绘制 circos 外圈样本分组，此处共计 2 个分组 N、C）。此外在后面的绘图过程中，会使用该文件确定样本的排列顺序，所以请事先根据想要展示的顺序给该文件中的样本进行排序。

（3）物种丰度表格文件（otu\_table.txt）。每一列为样本，每一行为物种（此处为 OTU），交叉区为个 OTU 在各样本中的丰度（用于计算）。因 OTU 及样本的绘图顺序由前述 2 个文件确定，所以在该文件中无需纠结样本或 OTU 的排序。

OTU_ID	phylum	detail
OTU_26	Proteobacteria	p__Proteobacteria;c__Alphaproteobacteria;o__Sphingomonadales;f__Sphingomonadaceae
OTU_8	Proteobacteria	p__Proteobacteria;c__Betaproteobacteria;o__Nitrosomonadales;f__Nitrosomonadaceae
OTU_7	Proteobacteria	p__Proteobacteria;c__Alphaproteobacteria;o__Rhodospirillales;f__Rhodospirillaceae
OTU_149	Proteobacteria	p__Proteobacteria;c__Alphaproteobacteria;o__Sphingomonadales;f__Sphingomonadaceae;g__Sphingomonas
OTU_12489	Proteobacteria	p__Proteobacteria;c__Gammaproteobacteria
OTU_27	Proteobacteria	p__Proteobacteria;c__Betaproteobacteria;o__Burkholderiales;f__Comamonadaceae;g__Variovorax
OTU_17	Proteobacteria	p__Proteobacteria;c__Alphaproteobacteria;o__Caulobacteriales;f__Hyphomonadaceae;g__Woodsholea
OTU_44	Proteobacteria	p__Proteobacteria;c__Alphaproteobacteria;o__Rhizobiales;f__Rhizobiaceae;g__Rhizobium
OTU_32	Acidobacteria	p__Actinobacteria;c__unidentified_Actinobacteria;o__Micrococcales;f__Promicromonosporaceae
OTU_47	Acidobacteria	p__Actinobacteria;c__unidentified_Actinobacteria;o__Pseudonocardiales;f__Pseudonocardaceae
OTU_30	Acidobacteria	p__Actinobacteria;c__Thermoleophila;o__Solirubrobacterales;f__Elev-16S-1332
OTU_104	Acidobacteria	p__Actinobacteria;c__unidentified_Actinobacteria;o__Bifidobacteriales;f__Bifidobacteriaceae;g__Bifidobacterium
OTU_643	Acidobacteria	p__Actinobacteria;c__Thermoleophila;o__Solirubrobacterales;f__Solirubrobacteraceae;g__Solirubrobacter
OTU_21	Bacteroidetes	p__Bacteroidetes;c__Cytophagia;o__Cytophagales;f__Cytophagaceae;g__Dyadobacter
OTU_15	Bacteroidetes	p__Bacteroidetes;c__Sphingobacteriia;o__Sphingobacteriales;f__Chitinophagaceae;g__Flavisolibacter
OTU_43	Bacteroidetes	p__Bacteroidetes;c__Sphingobacteriia;o__Sphingobacteriales;f__NS11-12_marine_group
OTU_38	Bacteroidetes	p__Bacteroidetes;c__Sphingobacteriia;o__Sphingobacteriales;f__Chitinophagaceae
OTU_24	Gemmatimonadetes	p__Gemmatimonadetes;c__unidentified_Gemmatimonadetes;o__Gemmatimonadales;f__Gemmatimonadaceae
OTU_49	Gemmatimonadetes	p__Gemmatimonadetes;c__unidentified_Gemmatimonadetes;o__Gemmatimonadales;f__Gemmatimonadaceae
OTU_10	Nitrospirae	p__Nitrospirae;c__Nitrospira;o__Nitrospirales;f__O319-6A21

物种分类信息文件

OTU_ID	N1	N2	N3	C1	C2	C3
OTU_21	235	671	989	1904	1003	637
OTU_26	572	872	519	1206	707	569
OTU_8	584	651	779	1183	578	482
OTU_32	741	642	860	259	572	350
OTU_47	735	469	734	276	468	283
OTU_7	407	808	440	1034	424	452
OTU_149	473	672	448	649	468	324
OTU_15	48	778	670	1362	280	332
OTU_12489	581	371	333	673	461	398
OTU_30	335	237	358	1567	197	226
OTU_10	223	227	374	1114	368	214
OTU_43	75	353	491	1052	234	169
OTU_104	615	294	612	30	235	146
OTU_27	353	427	308	504	256	269
OTU_24	517	223	302	438	249	195
OTU_49	783	204	369	320	205	122
OTU_17	243	329	369	586	255	178
OTU_38	36	159	89	1431	293	221
OTU_643	184	186	206	1091	174	128
OTU_44	123	375	242	422	209	193

sample_ID	group_ID
N1	N
N2	N
N3	N
C1	C
C2	C
C3	C

样本分组信息文件

物种丰度表格文件

## 读取文件内容及预处理

首先加载 `circlize` 包，用于绘制 `circos` 图。

然后在开始输出 3 个文件的名称，并事先根据 OTU 的种类数、样本数等预先定义颜色（讲真，样本或 OTU 一多，颜色就特别难以区分了）。

```
library(circlize)
library(reshape2) #在某步排列表格用

otu_table_file <- 'otu_table.txt'
group_file <- 'group.txt'
taxonomy_file <- 'taxonomy.txt'
color_otu <- c('#8DD3C7', '#FFFFB3', '#BEBADA', '#FB8072', '#80B1D3', '#FDB462',
               '#B3DE69', '#FCCDE5', '#BC80BD', '#CCEBC5', '#FFED6F', '#E41A1C', '#377EB8',
               '#4DAF4A', '#984EA3', '#FF7F00', '#FFFF33', '#A65628', '#F781BF', '#66C2A5')
color_sample <- c('#6181BD', '#F34800', '#64A10E', '#FF00FF', '#c7475b', '#049a0b')
color_phylum <- c('#BEAED4', '#FDC086', '#FFFF99', '#386CB0', '#F0027F')
color_group <- c('#4253ff', '#ff4308')
```

读取 3 个文件，并根据物种分类信息文件（`taxonomy.txt`）以及样本分组信息文件（`group.txt`）中的 OTU、样本等顺序，对物种丰度表格文件（`otu_table.txt`）中的 OTU、样本进行排序（最终得到数据框 `otu_table`）。

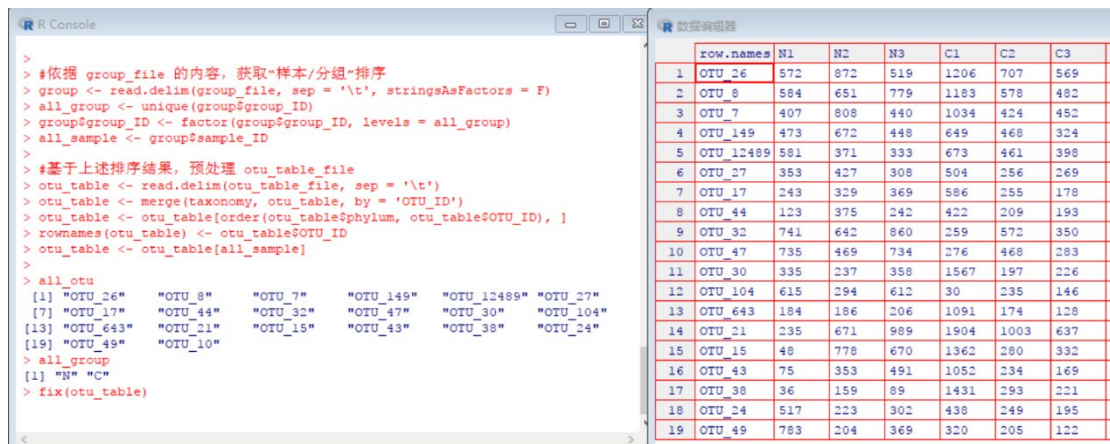
同时，预读取所有的 OTU 名称（向量 `all_otu`）、微生物门名称（向量 `tax_phylum`）、样本名称（向量 `all_sample`）以及样本分组名称（向量 `group`）。

```
#依据 taxonomy_file 的内容，获取“OTU/分类”排序
taxonomy <- read.delim(taxonomy_file, sep = '\t', stringsAsFactors = F)
tax_phylum <- unique(taxonomy$phylum)
taxonomy$phylum <- factor(taxonomy$phylum, levels = tax_phylum)
all_otu <- taxonomy$OTU_ID
taxonomy$OTU_ID <- factor(taxonomy$OTU_ID, levels = all_otu)

#依据 group_file 的内容，获取“样本/分组”排序
group <- read.delim(group_file, sep = '\t', stringsAsFactors = F)
all_group <- unique(group$group_ID)
group$group_ID <- factor(group$group_ID, levels = all_group)
all_sample <- group$sample_ID

#基于上述排序结果，预处理 otu_table_file
otu_table <- read.delim(otu_table_file, sep = '\t')
otu_table <- merge(taxonomy, otu_table, by = 'OTU_ID')
otu_table <- otu_table[order(otu_table$phylum, otu_table$OTU_ID), ]
rownames(otu_table) <- otu_table$OTU_ID
otu_table <- otu_table[all_sample]
```

可查看读取及赋值结果。



The screenshot shows the R Console on the left and the Data Editor on the right. The R Console contains the following code and output:

```
>
> #依据 group_file 的内容, 获取“样本/分组”排序
> group <- read.delim(group_file, sep = '\t', stringsAsFactors = F)
> all_group <- unique(group$group_ID)
> group$group_ID <- factor(group$group_ID, levels = all_group)
> all_sample <- group$sample_ID
>
> #基于上述排序结果, 预处理 otu_table file
> otu_table <- read.delim(otu_table_file, sep = '\t')
> otu_table <- merge(taxonomy, otu_table, by = 'OTU_ID')
> otu_table <- otu_table[order(otu_table$phylum, otu_table$OTU_ID), ]
> rownames(otu_table) <- otu_table$OTU_ID
> otu_table <- otu_table[all_sample, ]
>
> all_otu
[1] "OTU_26" "OTU_8" "OTU_7" "OTU_149" "OTU_12489" "OTU_27"
[7] "OTU_17" "OTU_44" "OTU_32" "OTU_47" "OTU_30" "OTU_104"
[13] "OTU_643" "OTU_21" "OTU_15" "OTU_43" "OTU_38" "OTU_24"
[19] "OTU_49" "OTU_10"
> all_group
[1] "N" "C"
> fix(otu_table)
```

The Data Editor on the right displays the first 19 rows of the OTU table, with columns: row.names, N1, N2, N3, C1, C2, C3.

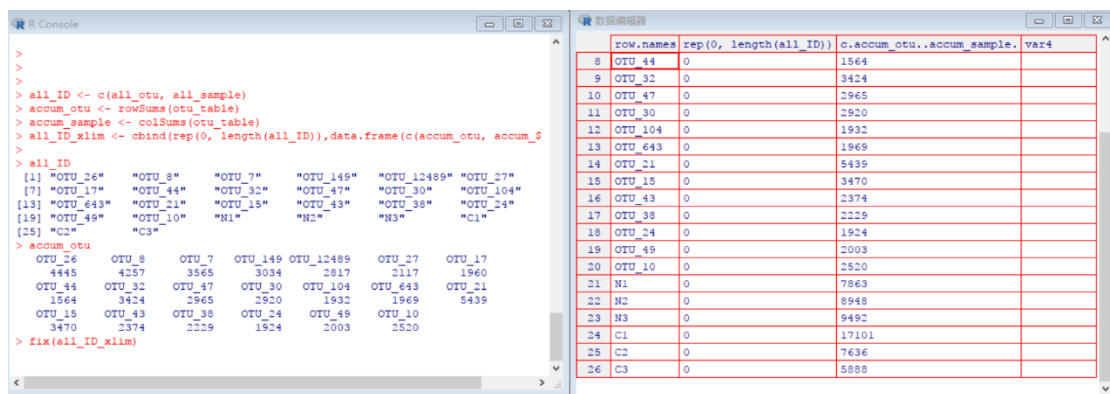
row.names	N1	N2	N3	C1	C2	C3
1 OTU_26	572	872	519	1206	707	569
2 OTU_8	584	651	779	1183	578	482
3 OTU_7	407	808	440	1034	424	452
4 OTU_149	473	672	448	649	468	324
5 OTU_12489	581	371	333	673	461	398
6 OTU_27	353	427	308	504	256	269
7 OTU_17	243	329	369	586	255	178
8 OTU_44	123	375	242	422	209	193
9 OTU_32	741	642	860	259	572	350
10 OTU_47	735	469	734	276	468	283
11 OTU_30	335	237	358	1567	197	226
12 OTU_104	615	294	612	30	235	146
13 OTU_643	184	186	206	1091	174	128
14 OTU_21	235	671	989	1904	1003	637
15 OTU_15	48	778	670	1362	280	332
16 OTU_43	75	353	491	1052	234	169
17 OTU_38	36	159	89	1431	293	221
18 OTU_24	517	223	302	438	249	195
19 OTU_49	783	204	369	320	205	122

变量读取完毕, 根据读取的内容进行统计计算, 得到绘图数据。

首先计算 circlize 外圈属性数据, 即每种 OTU 的总丰度, 以及各样本中所含 OTU 的总丰度。说白了就是给 otu\_table 的行和列求个总和, 如下。

```
#circlize 外圈属性数据
all_ID <- c(all_otu, all_sample)
accum_otu <- rowSums(otu_table)
accum_sample <- colSums(otu_table)
all_ID_xlim <- cbind(rep(0, length(all_ID)), data.frame(c(accum_otu, accum_sample)))
```

查看得到的数据框 all\_ID\_xlim, 如下, 很好理解不多说。



The screenshot shows the R Console on the left and the Data Editor on the right. The R Console contains the following code and output:

```
>
>
> all_ID <- c(all_otu, all_sample)
> accum_otu <- rowSums(otu_table)
> accum_sample <- colSums(otu_table)
> all_ID_xlim <- cbind(rep(0, length(all_ID)), data.frame(c(accum_otu, accum_sample)))
>
> all_ID
[1] "OTU_26" "OTU_8" "OTU_7" "OTU_149" "OTU_12489" "OTU_27"
[7] "OTU_17" "OTU_44" "OTU_32" "OTU_47" "OTU_30" "OTU_104"
[13] "OTU_643" "OTU_21" "OTU_15" "OTU_43" "OTU_38" "OTU_24"
[19] "OTU_49" "OTU_10" "N1" "N2" "N3" "C1"
[25] "C2" "C3"
> accum_otu
OTU_26 OTU_8 OTU_7 OTU_149 OTU_12489 OTU_27 OTU_17
4445 4257 3565 3034 2817 2117 1960
OTU_44 OTU_32 OTU_47 OTU_30 OTU_104 OTU_643 OTU_21
1564 3424 2965 2920 1932 1969 5439
OTU_15 OTU_43 OTU_38 OTU_24 OTU_49 OTU_10
3470 2374 2229 1924 2003 2520
> fix(all_ID_xlim)
```

The Data Editor on the right displays the first 26 rows of the all\_ID\_xlim data frame, with columns: row.names, rep(0, length(all\_ID)), c.accum\_otu..accum\_sample., var4.

row.names	rep(0, length(all_ID))	c.accum_otu..accum_sample.	var4
8 OTU_44	0	1564	
9 OTU_32	0	3424	
10 OTU_47	0	2965	
11 OTU_30	0	2920	
12 OTU_104	0	1932	
13 OTU_643	0	1969	
14 OTU_21	0	5439	
15 OTU_15	0	3470	
16 OTU_43	0	2374	
17 OTU_38	0	2229	
18 OTU_24	0	1924	
19 OTU_49	0	2003	
20 OTU_10	0	2520	
21 N1	0	7863	
22 N2	0	8948	
23 N3	0	9492	
24 C1	0	17101	
25 C2	0	7636	
26 C3	0	5888	

然后计算 circlize 内圈连线数据, 即得到每种 OTU 在各样本中的丰度, 以及各样本所含每种 OTU 的丰度。呃呃, 有点绕口, 而且二者也是一个意思, 看下面就知道了。

```
#circlize 内圈连线数据
otu_table$otu_ID <- all_otu
plot_data <- melt(otu_table, id = 'otu_ID') #此处使用 reshape2 包的 melt()命令排列数据
colnames(plot_data)[2] <- 'sample_ID'
plot_data$otu_ID <- factor(plot_data$otu_ID, levels = all_otu)
plot_data$sample_ID <- factor(plot_data$sample_ID, levels = all_sample)
plot_data <- plot_data[order(plot_data$otu_ID, plot_data$sample_ID), ]
plot_data <- plot_data[c(2, 1, 3, 3)]
```

查看得到的数据框 `plot_data`，如下。

新数据框 `plot_data` 中，记录了各 OTU 和各样本的丰度对应关系。`sample_ID`，各样本；`otu_ID`，各 OTU；第 3 列和第 4 列内容完全一致，均代表了每种 OTU 在各样本中的丰度。至于为什么为 2 列，是因为作图需要，自己运行一下即可理解了。

The screenshot shows two windows from an RStudio interface:

- R Console (Left):** Displays the following commands and their output:
 

```
> 
> 
> 
> 
> 
> 
> 
> otu_table$otu_ID <- all_otu
> plot_data <- melt(otu_table, id = 'otu_ID') #此处使用reshape2包的melt()命令
> colnames(plot_data)[2] <- 'sample_ID'
> plot_data$otu_ID <- factor(plot_data$otu_ID, levels = all_otu)
> plot_data$sample_ID <- factor(plot_data$sample_ID, levels = all_sample)
> plot_data <- plot_data[order(plot_data$otu_ID, plot_data$sample_ID), ]
> plot_data <- plot_data[c(2, 1, 3, 3)]
> 
> fix(plot_data)
```
- Data Editor (Right):** Shows a table titled "数据编辑器" (Data Editor) containing the transformed data. The columns are row.names, sample\_ID, otu\_ID, value, and value.1. The rows represent individual samples with their corresponding OTU IDs and values.
 

	row.names	sample_ID	otu_ID	value	value.1
1	1	N1	OTU_26	572	572
2	21	N2	OTU_26	872	872
3	41	N3	OTU_26	519	519
4	61	C1	OTU_26	1206	1206
5	81	C2	OTU_26	707	707
6	101	C3	OTU_26	569	569
7	2	N1	OTU_8	584	584
8	22	N2	OTU_8	651	651
9	42	N3	OTU_8	779	779
10	62	C1	OTU_8	1183	1183
11	82	C2	OTU_8	578	578
12	102	C3	OTU_8	482	482
13	3	N1	OTU_7	407	407
14	23	N2	OTU_7	808	808
15	43	N3	OTU_7	440	440
16	63	C1	OTU_7	1034	1034
17	83	C2	OTU_7	424	424
18	103	C3	OTU_7	452	452
19	4	N1	OTU_149	473	473

顺便给各 OTU 及样本定义颜色。

```
#颜色设置
names(color_otu) <- all_otu
names(color_sample) <- all_sample
```

这样就把预先设定的颜色与各 OTU 及样本对应起来了。

```
> names(color_otu) <- all_otu
> names(color_sample) <- all_sample
> color_sample
      N1      N2      N3      C1      C2      C3
"#6181BD" "#F34800" "#64A10E" "#FF00FF" "#c7475b" "#049a0b"
> color_otu
      OTU_26      OTU_8      OTU_7      OTU_149      OTU_12489      OTU_27      OTU_17
"#8DD3C7" "#FFFFB3" "#BEBADA" "#FB8072" "#80B1D3" "#FDB462" "#B3DE69"
      OTU_44      OTU_32      OTU_47      OTU_30      OTU_104      OTU_643      OTU_21
"#FCCDE5" "#BC80BD" "#CCEBC5" "#FFED6F" "#E41A1C" "#377EB8" "#4DAF4A"
      OTU_15      OTU_43      OTU_38      OTU_24      OTU_49      OTU_10
"#984EA3" "#FF7F00" "#FFFF33" "#A65628" "#F781BF" "#66C2A5"
```



## circlize 绘图

然后接下来就是使用 `circlize` 包绘制 `circos` 图了。

注：绘图细节部分的调整很多也很繁琐，此处默认使用的各细节参数设置（如字体大小等）均根据示例文件而来。若是大家后续更换为自己的数据时，还需多加调试参数了。

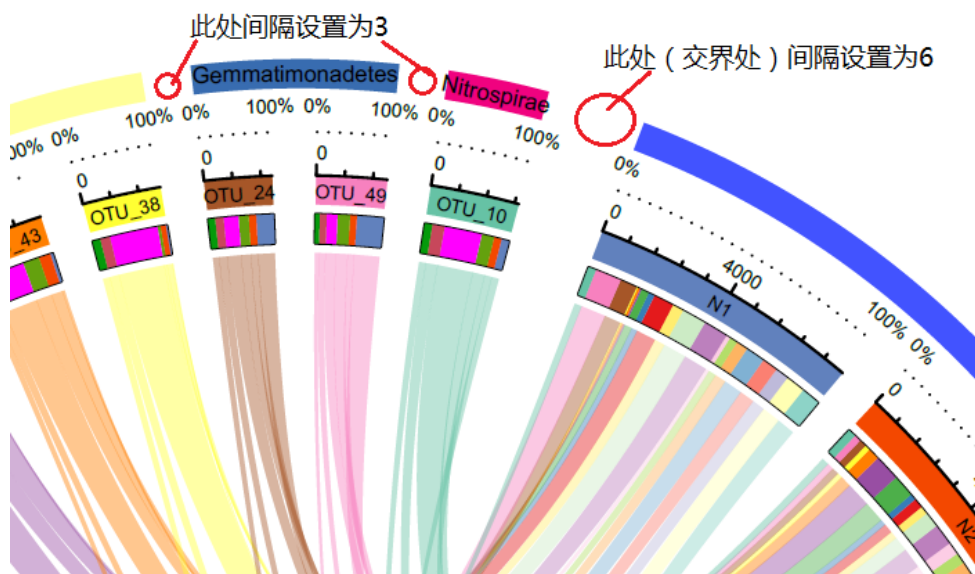
首先定义整体的布局。

```
pdf('circlize_plot.pdf', width = 8, height = 8)

##整体布局
gap_size <- c(rep(3, length(all_otu) - 1), 6, rep(3, length(all_sample) - 1), 6)
circos.par(cell.padding = c(0, 0, 0, 0), start.degree = 270, gap.degree = gap_size)
circos.initialize(factors = factor(all_ID, levels = all_ID), xlim = all_ID_xlim)
```

预先生成 pdf 文件，将图片绘制到 pdf 中。我这里设置了一个 8×8 大小的 pdf。

`gap_size` 为设定的 `circos` 图中，各 OTU 或样本区块之间的间距（如下图）；`circos.par` 就是画板设置了；`circos.initialize` 定义绘图因子，此处使用了前述得到的 `circlize` 外圈属性数据框 `all_ID_xlim`。



然后开始绘图。

与 `perl` 的 `circos` 图的绘图顺序一致，R 的 `circlize` 也是由外圈往内圈逐个添加。在下文中，由外圈到内圈简称为第一圈、第二圈……

在本次绘图的示例图中，可以看到处在最外圈的部分为 OTU 的门分类信息以及样本的分组信息，因此我们首先将 OTU 的门分类信息以及样本的分组信息添加在最外圈（第一圈）。

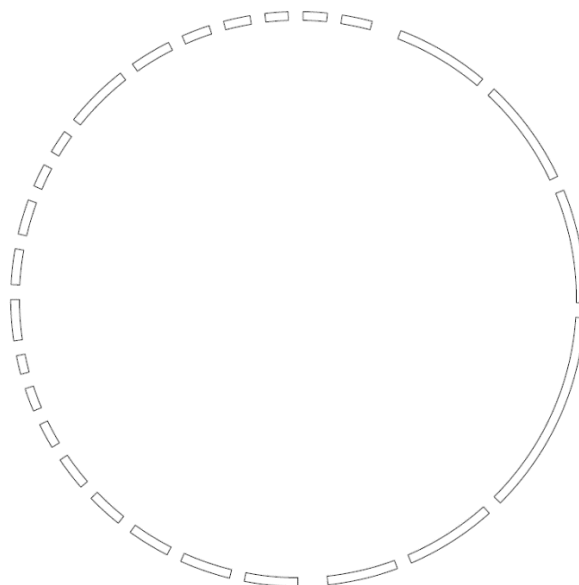
```
##绘制 OTU 分类、样本分组区块（第一圈）
circos.trackPlotRegion(
  ylim = c(0, 1), track.height = 0.03, bg.border = NA,
  panel.fun = function(x, y) {
    sector.index = get.cell.meta.data('sector.index')
    xlim = get.cell.meta.data('xlim')
    ylim = get.cell.meta.data('ylim')
  })

for (i in 1:length(tax_phylum)) {
  tax_OTU <- {subset(taxonomy, phylum == tax_phylum[i])}$OTU_ID
  highlight.sector(tax_OTU, track.index = 1, col = color_phylum[i], text = tax_phylum[i],
    cex = 0.5, text.col = 'black', niceFacing = FALSE)
}

for (i in 1:length(all_group)) {
  group_sample <- {subset(group, group_ID == all_group[i])}$sample_ID
  highlight.sector(group_sample, track.index = 1, col = color_group[i], text = all_group[i],
    cex = 0.7, text.col = 'black', niceFacing = FALSE)
}
```

第一条命令用于绘制第一圈。ylim 控制 y 轴范围，track.height 控制高度，bg.border 控制边框颜色；定义的函数 panel.fun() 以循环的方式，读取 all\_ID\_xlim 中的内容，逐一绘制外圈区块。

运行完上述的第一条命令之后，发现绘图区域啥也没显示。但事实上，并非没有绘制任何元素，只是将绘制元素的填充颜色及边框颜色设置为 NA 了，因为此处不需要任何颜色作为添加。大家可将 bg.border=NA 更改为 bg.border='black'，即可明白（更改后出图如下）。更多参数信息可使用 help(circos.trackPlotRegion) 查看。



然后接下来使用两个循环，将预先设定的 OTU 门分类颜色以及样本分组颜色添加在刚才未定义颜色的第一圈。`highlight.sector()`实现这个功能，并可以将处于同一分组的样本合并在一起。`track.index = 1` 指定了将颜色添加在 `circlize` 图的第一圈；`col` 和 `text` 参数分别指定颜色和标签内容；`cex` 指定标签字体大小，`text.col` 指定标签字体颜色，`niceFacing` 用于展示标签字体的方向，默认为 `TRUE` 但个人推荐使用 `FALSE`。更多参数信息可使用 `help(highlight.sector)`查看。



然后我们继续往内圈绘制。

第二圈为 OTU 丰度的百分比刻度信息。



```

##添加百分比注释（第二圈）
circos.trackPlotRegion(
  ylim = c(0, 1), track.height = 0.05, bg.border = NA,
  panel.fun = function(x, y) {
    sector.index = get.cell.meta.data('sector.index')
    xlim = get.cell.meta.data('xlim')
    ylim = get.cell.meta.data('ylim')
  })

circos.track(
  track.index = 2, bg.border = NA,
  panel.fun = function(x, y) {
    xlim = get.cell.meta.data('xlim')
    ylim = get.cell.meta.data('ylim')
    sector.name = get.cell.meta.data('sector.index')
    xplot = get.cell.meta.data('xplot')

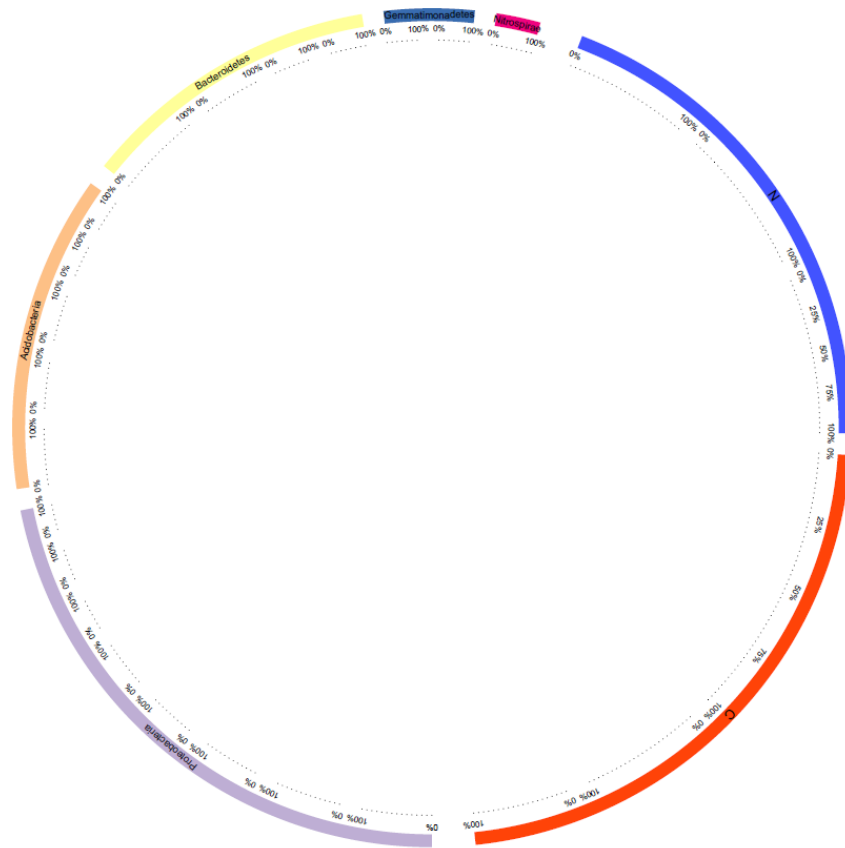
    by = ifelse(abs(xplot[2] - xplot[1]) > 30, 0.25, 1)
    for (p in c(0, seq(by, 1, by = by))) circos.text(p*(xlim[2] - xlim[1]) + xlim[1],
mean(ylim) + 0.4, paste0(p*100, '%'), cex = 0.4, adj = c(0.5, 0), niceFacing = FALSE)

    circos.lines(xlim, c(mean(ylim), mean(ylim)), lty = 3)
  })

```

首先我们使用 `circos.trackPlotRegion()` 绘制一圈无边框、无颜色的空白区域，目的为添加的刻度标签等内容留出展示的空间。

然后使用 `circos.track()` 添加百分比刻度标签。其中，`track.index = 2` 意为将刻度标签展示在 `circos` 图的第二圈；定义的函数 `panel.fun()` 使用循环的方式，读取 `all_ID_xlim` 中的内容（丰度信息）并在读取后进行计算得到百分比数值，同时额外使用 `if` 判断 OTU 丰度的绝对数值，若数值足够大则考虑将展示更多的刻度（以 25% 为一刻度），若数值过小则仅展示 0 和 100% 这两个刻度；`circos.lines()` 添加了外周的点状虚线。



第三圈和第四圈开始正式绘制 OTU 及样本区块。

其中第三圈为主区块，第四圈为副区块。

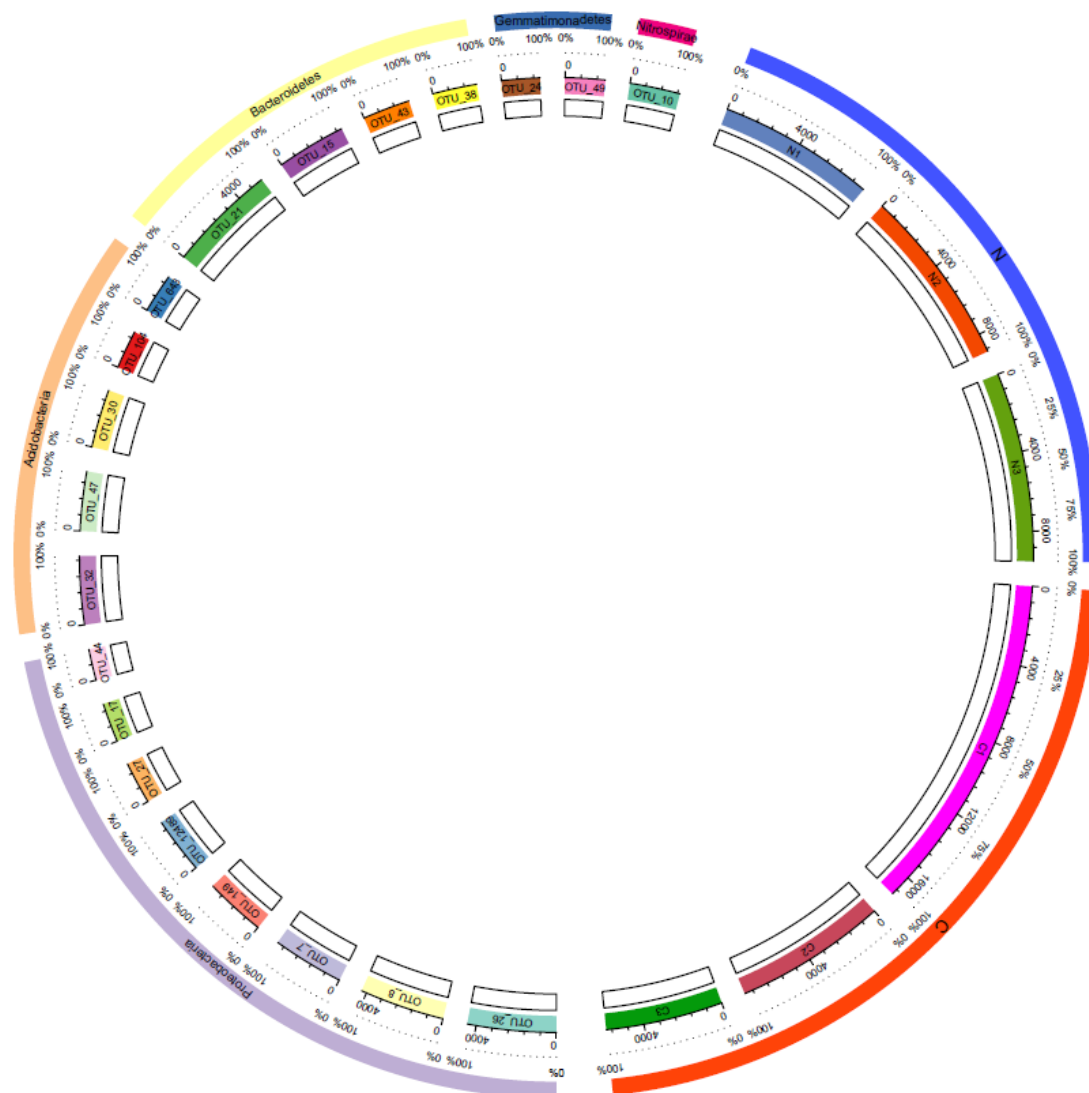
```
##绘制 OTU、样本主区块（第三圈）
circos.trackPlotRegion(
  ylim = c(0, 1), track.height = 0.03, bg.col = c(color_otu, color_sample), bg.border = NA,
  track.margin = c(0, 0.01),
  panel.fun = function(x, y) {
    xlim = get.cell.meta.data('xlim')
    sector.name = get.cell.meta.data('sector.index')
    circos.axis(h = 'top', labels.cex = 0.4, major.tick.percentage = 0.4, labels.niceFacing
= FALSE)
    circos.text(mean(xlim), 0.2, sector.name, cex = 0.4, niceFacing = FALSE, adj = c(0.5,
0))
  })

##绘制 OTU、样本副区块（第四圈）
circos.trackPlotRegion(ylim = c(0, 1), track.height = 0.03, track.margin = c(0, 0.01))
```

继续使用 `circos.trackPlotRegion()` 循环读取 `all_ID_xlim` 中的内容并绘制，此时可以将预先定义的 OTU 颜色及样本颜色添加上。同时在内部使用 `circos.axis()` 和 `circos.text()` 函数，在循环绘图的同时，将刻度线和标签添加上。此处，`sector.name` 为在循环过程中每次读取的 OTU 或样本名称；`circos.text()` 中的 `mean(xlim)` 和 `0.2` 分别意为将给定的标签文字添加在每个

区块的 X 轴中间位置以及 Y=0.2 的位置。其余参数项类似上述不再多说，更细致的参数可使用 `help()` 查看帮助。

第三圈主圈绘制完成，第四圈副圈暂未添加颜色（将在后续操作中添加）。



至此，外周区域绘制完成。

最内圈为样本- OTU 丰度关联信息，以连线的方式展示。此处需使用之前计算得到的 `circulize` 内圈连线数据框 `plot_data`。

```

##绘制 OTU-样本关联连线（最内圈）
for (i in seq_len(nrow(plot_data))) {
  circos.link(
    plot_data[i,2], c(accum_otu[plot_data[i,2]], accum_otu[plot_data[i,2]] -
plot_data[i,4]),
    plot_data[i,1], c(accum_sample[plot_data[i,1]], accum_sample[plot_data[i,1]] -
plot_data[i,3]),
    col = paste0(color_otu[plot_data[i,2]], '70'), border = NA )

  circos.rect(accum_otu[plot_data[i,2]], 0, accum_otu[plot_data[i,2]] - plot_data[i,4], 1,
sector.index = plot_data[i,2], col = color_sample[plot_data[i,1]], border = NA)
  circos.rect(accum_sample[plot_data[i,1]], 0, accum_sample[plot_data[i,1]] -
plot_data[i,3], 1, sector.index = plot_data[i,1], col = color_otu[plot_data[i,2]], border = NA)

  accum_otu[plot_data[i,2]] = accum_otu[plot_data[i,2]] - plot_data[i,4]
  accum_sample[plot_data[i,1]] = accum_sample[plot_data[i,1]] - plot_data[i,3]
}

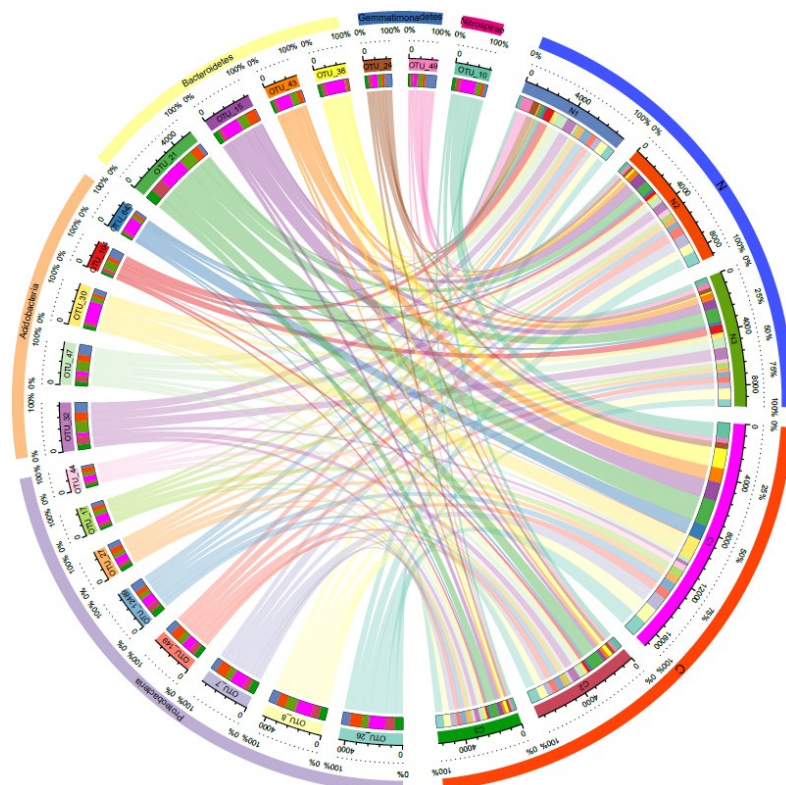
##清除 circlize 样式并关闭画板，保存 pdf
circos.clear()
dev.off()

```

此处以循环的方式添加最内圈连线，连线颜色以 OTU 的预设颜色为准（绘制的同时将颜色的不透明度设置为 70%）。

先前绘制外圈第四圈的时候未添加颜色，也在此处将颜色添加上，即命令中使用的两个 `circos.rect()` 函数分别添加。

运行完毕后，我们的 `circlize` 绘图就完成了。最终结果如下图所示。

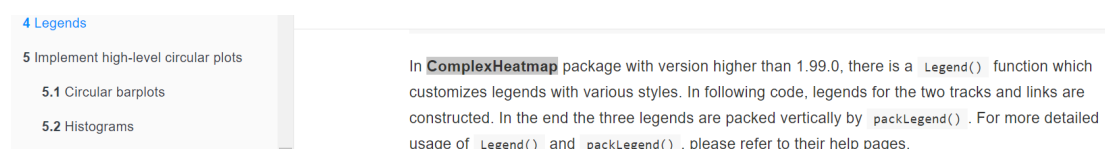


## 添加图例

其实到了这一步就已经完成了，不过示例图中还包含了图例信息。算是额外的一个补充内容吧，教给大家一个可用于绘制图例的包（虽然绘制图例的包有很多）。

首先需要声明一点，`circlize` 包中是没有绘制图例的命令的，这点与 `perl` 的 `circos` 一样，均无法直接绘制图例。若是添加图例的话，还需额外调用其它的包绘制图例。

在 `circlize` 的官方说明文档中，提及了一个叫 `ComplexHeatmap` 的包，可用于为 `circlize` 图添加图例。



然后参照示例文档，我们可以使用 `Legend()` 命令添加图例。

注：个人感觉 `ComplexHeatmap` 包的坑很多.....

除了加载 `ComplexHeatmap` 包外，我们再加载 `grid` 包，用于调节画板，便于存放图例。

```
library(ComplexHeatmap) #可用此包添加图例
library(grid) #可用此包调整画板

##读取数据、预处理操作等，见前述，这里不再展示
##.....
##直接展示 circlize 绘图

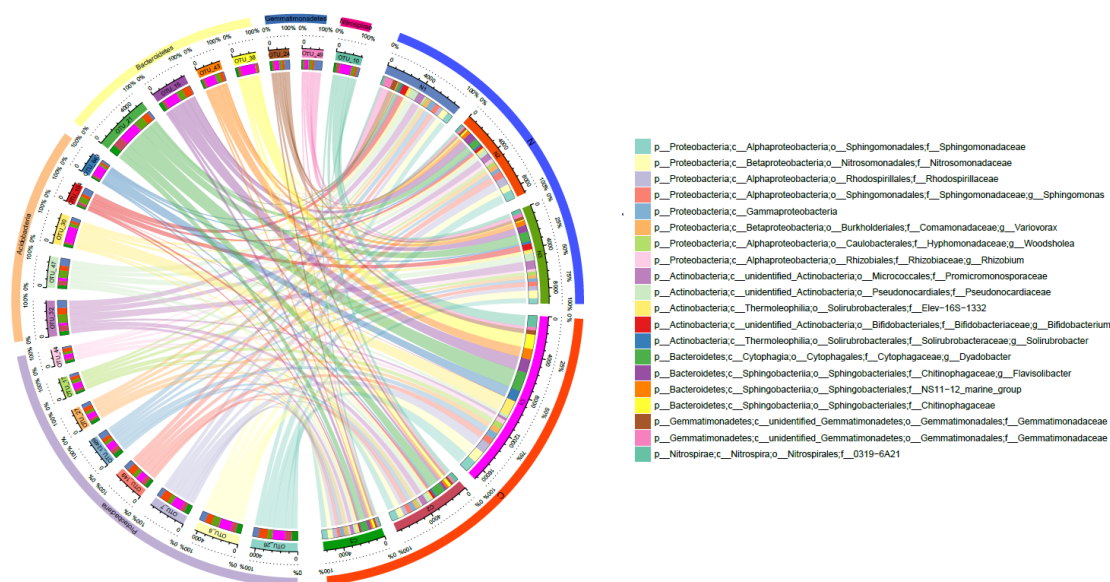
##首先将 pdf 的宽设置的大一些
pdf('circlize_plot.pdf', width = 20, height = 8)
circle_size = unit(1, 'snpc')

##然后中间为 circlize 绘图，见前述，此处不展示
##.....

##最后添加图例，Legend()绘制图例，配合 grid 包调节画板
otu_legend <- Legend(
  at = all_otu, labels = taxonomy$detail, labels_gp = gpar(fontsize = 8),
  grid_height = unit(0.5, 'cm'), grid_width = unit(0.5, 'cm'), type = 'points', pch = NA,
  background = color_otu)

pushViewport(viewport(x = 0.85, y = 0.5))
grid.draw(otu_legend)
upViewport()
```

图例添加完毕后，最终结果如下所示。



## 补充说明

最后再补充一点内容。

就是使用过 `grid` 包设置过画板，组合过多个图片的同学们会很清楚，`grid` 包在多图排列组合方面很好用。很多作图包（如 `ggplot2`）的作图结果均可用 `grid` 包进行调整。

但是 `circlize` 包的结果似乎无法使用 `grid` 包来调节（我尝试过 N 次了……），无论怎样设置画板，`circlize` 的结果总位于画板的正中心。所以最后添加图例的时候，也是很无奈将画板的宽度设置的很大，以便将图例展示在图的右侧（此时可以看到作图结果的左侧存在很大的空白区域，此处不再展示了，大家运行一下即可知道）。对于左侧空白处，可以在后期使用 AI、PS 等工具截掉。

若是大家有针对 `circlize` 添加图例的更好方法，还请大家私信我，万分感谢。

## 参考文档

`circlize` 包的官方说明文档，链接

[http://jokergoo.github.io/circlize\\_book/book](http://jokergoo.github.io/circlize_book/book)

主体部分借鉴了此大神的思路，表示感谢

<http://jokergoo.github.io/circlize/example/otu.html>