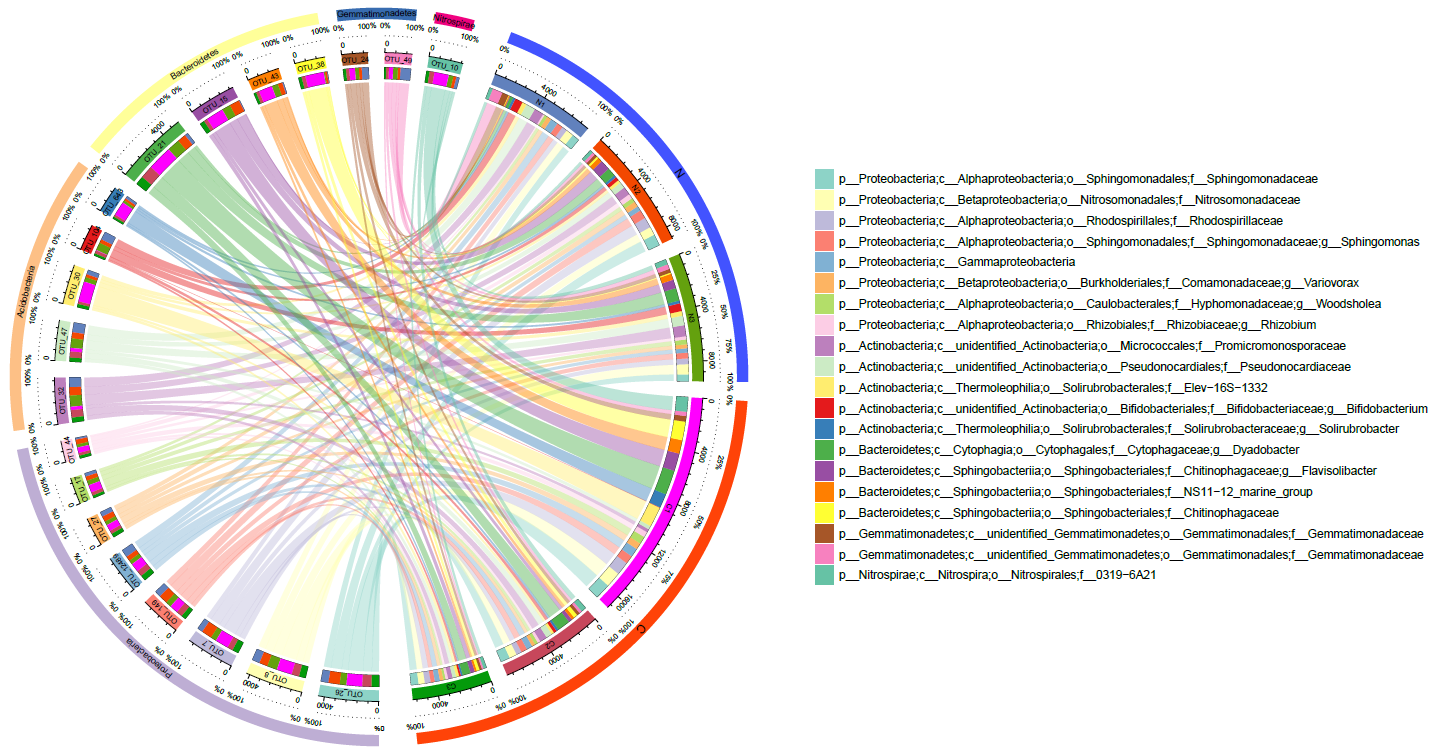
R作图-使用circlize绘制样本\_物种丰度关联弦状图

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）等，已上传至百度盘，无提取码（若失效请私信我）

<https://pan.baidu.com/s/10BCdWESl6Xt7DliUyT6drw>



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：

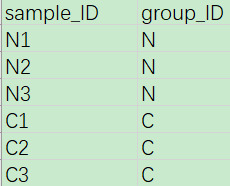
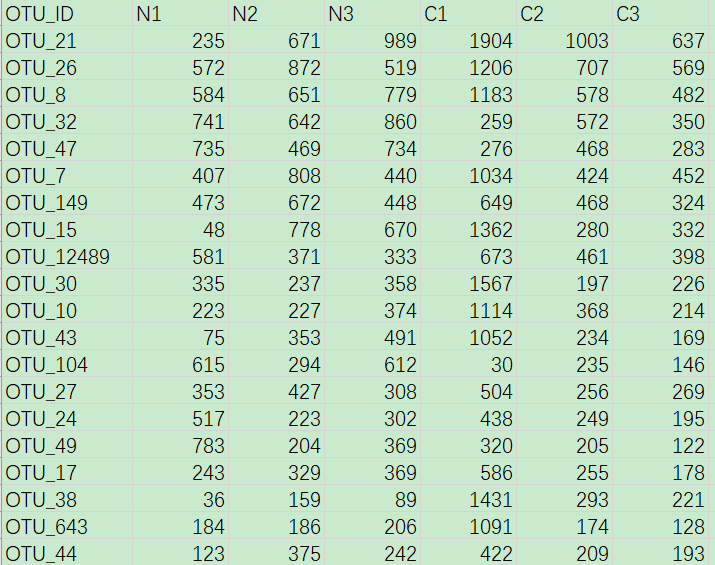
（1）物种分类信息文件（'taxonomy.txt）。第一列为物种ID（此处为OTU，在circos图的中间区域展示），第二列为OTU的“门水平”分类信息（用于绘制circos外圈OTU分类），第三列为OTU分类详细信息（用于在图例中展示）。此外在后面的绘图过程中，会使用该文件确定OTU的排列顺序，所以请事先根据想要展示的顺序给该文件中的OTU进行排序。

（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。



物种分类信息文件

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。

然后在开始出输入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')

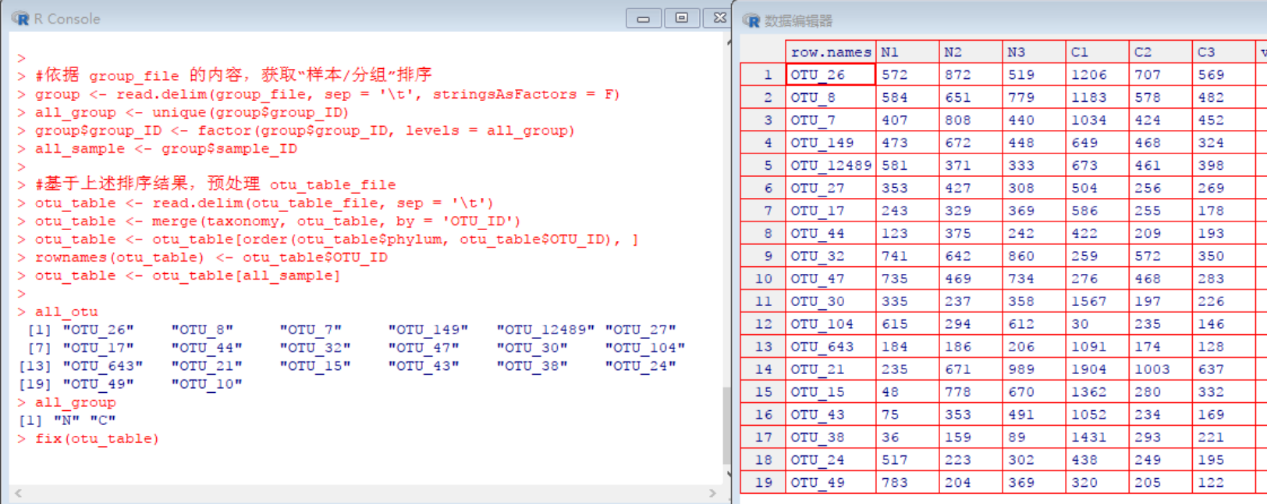
otu\_table <- merge(taxonomy, otu\_table, by = 'OTU\_ID')

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。



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

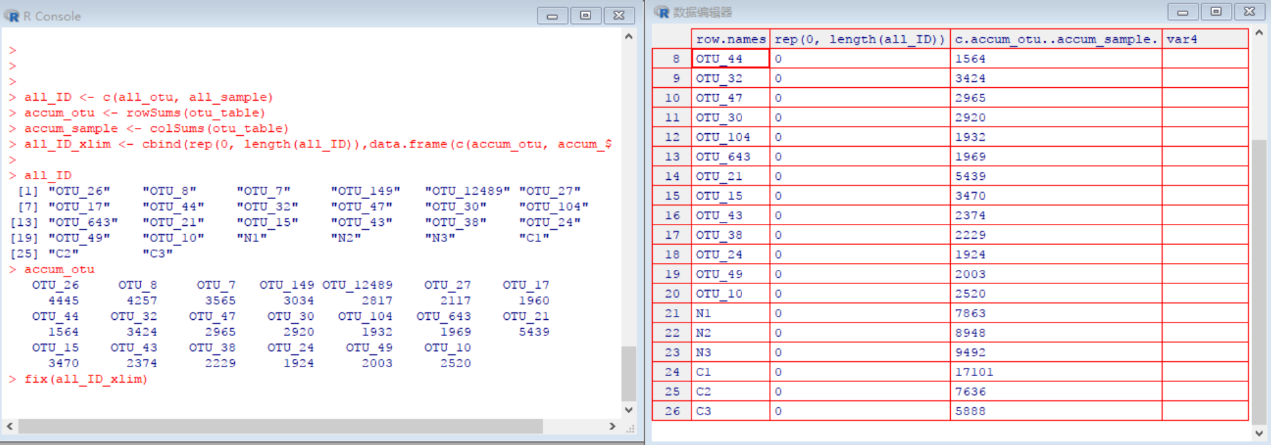
all\_ID <- c(all\_otu, all\_sample)

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。



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)

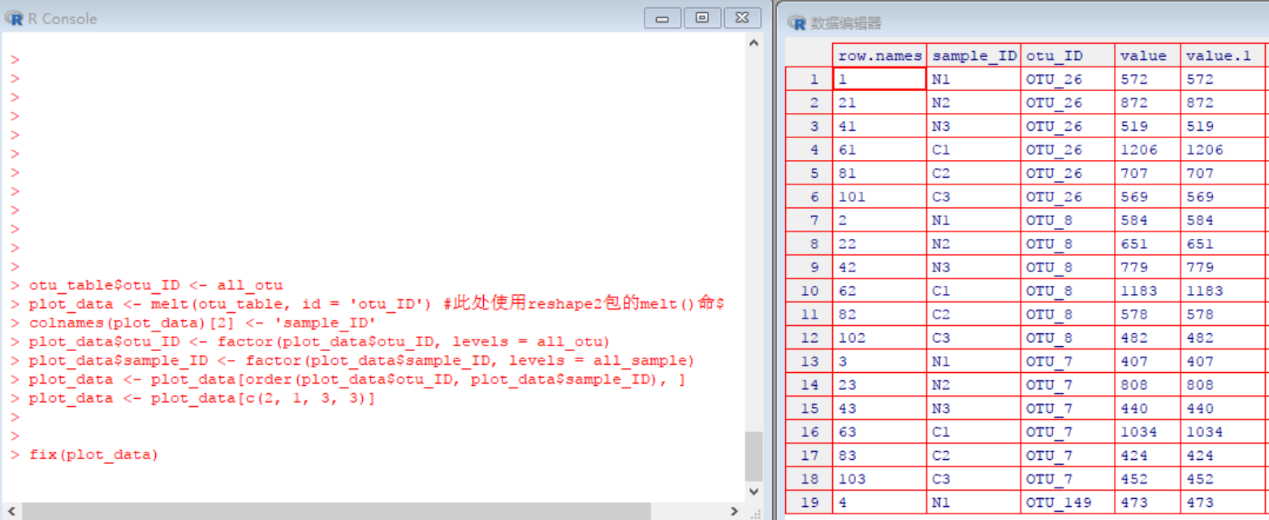
plot\_data$sample\_ID <- factor(plot\_data$sample\_ID, levels = all\_sample)

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。



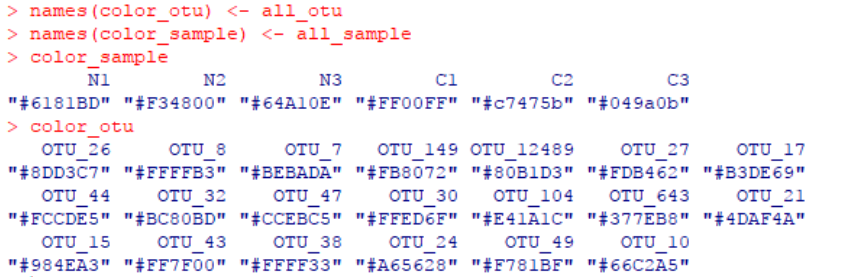
顺便给各OTU及样本定义颜色。

#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。



## 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

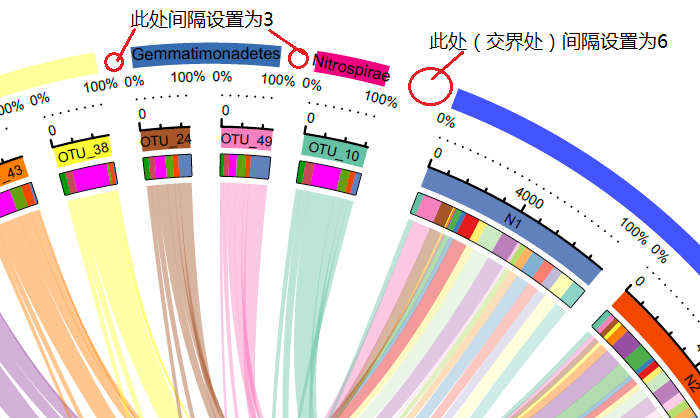
gap\_size <- c(rep(3, length(all\_otu) - 1), 6, rep(3, length(all\_sample) - 1), 6)

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)) {

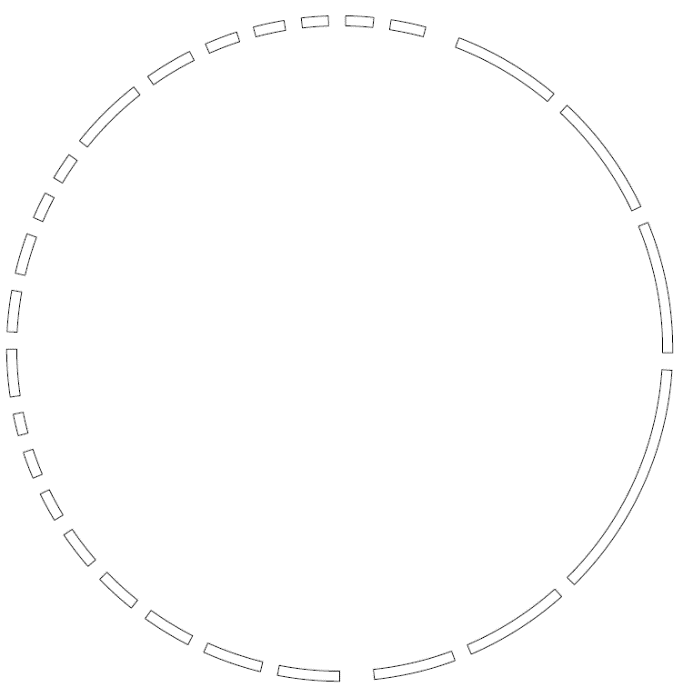
group\_sample <- {subset(group, group\_ID == all\_group[i])}$sample\_ID

highlight.sector(group\_sample, track.index = 1, col = color\_group[i], text = all\_group[i], cex = 0.7, text.col = 'black', niceFacing = FALSE)

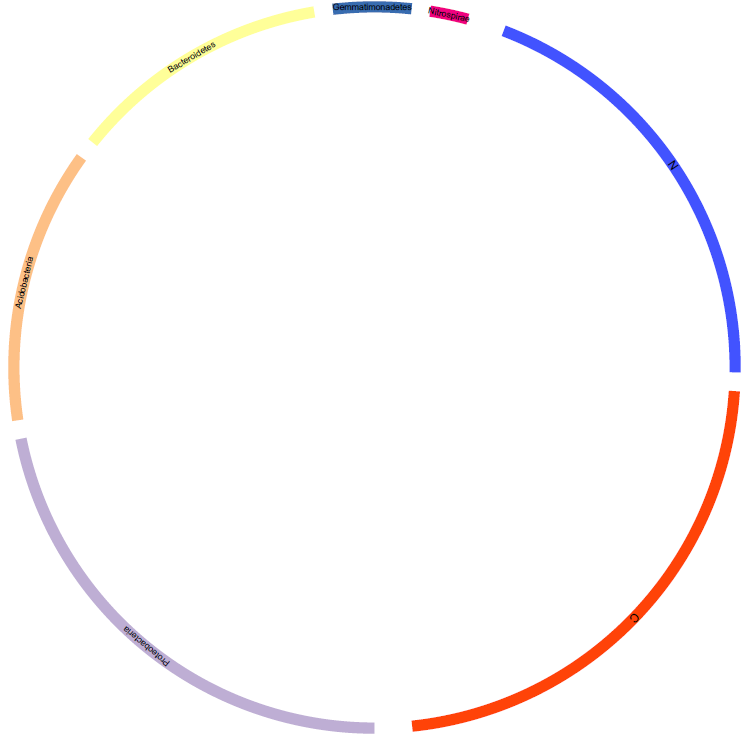
}

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)

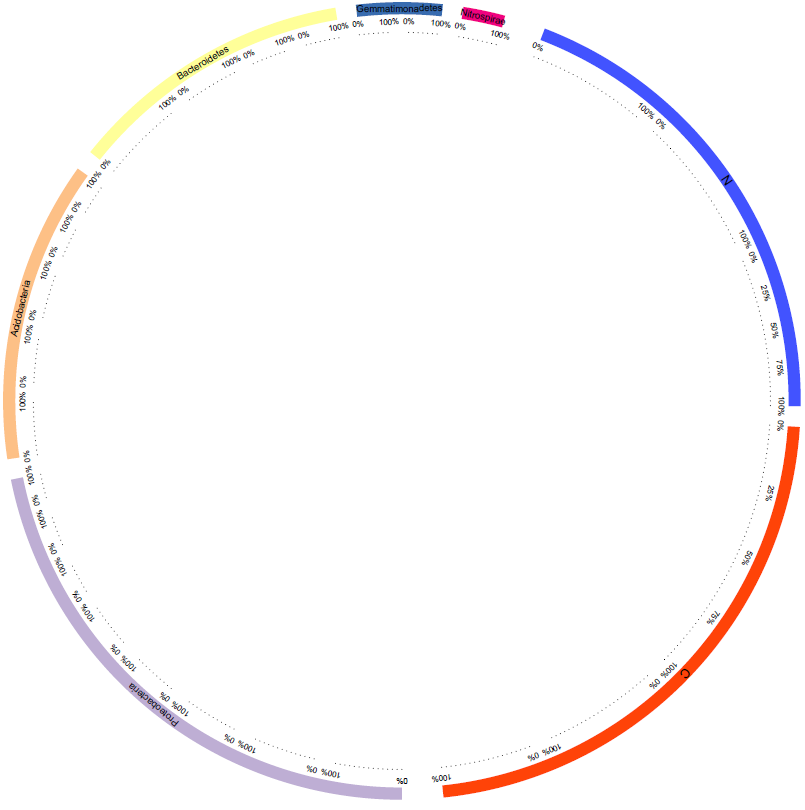
for (p in c(0, seq(by, 1, by = by))) circos.text(p\*(xlim[2] - xlim[1]) + xlim[1], mean(ylim) + 0.4, paste0(p\*100, '%'), cex = 0.4, adj = c(0.5, 0), niceFacing = FALSE)

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circlize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circlize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

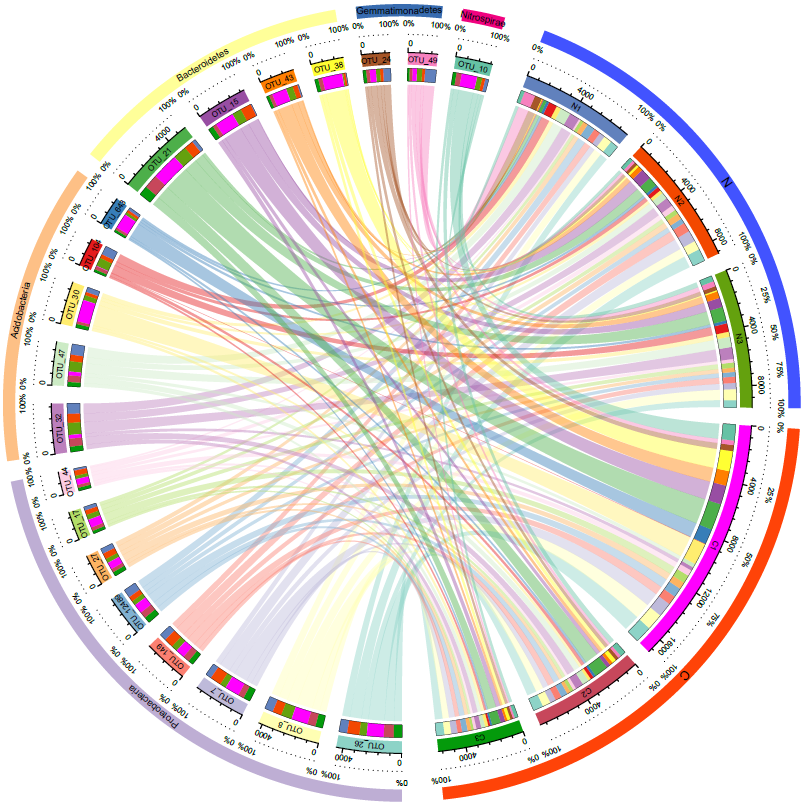
circos.clear()

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。

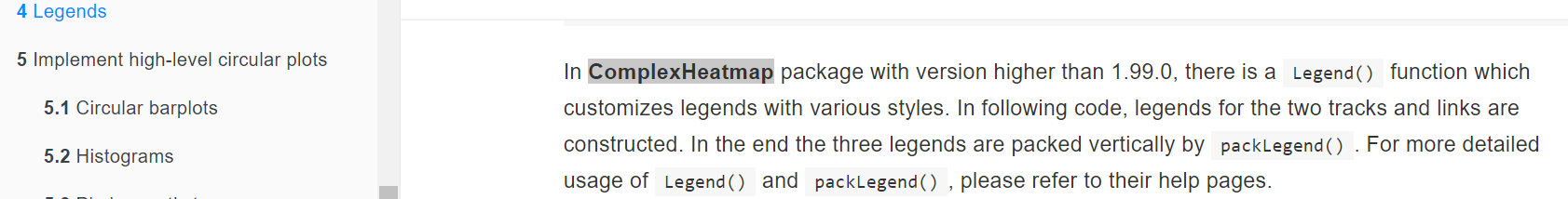


## 添加图例

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),

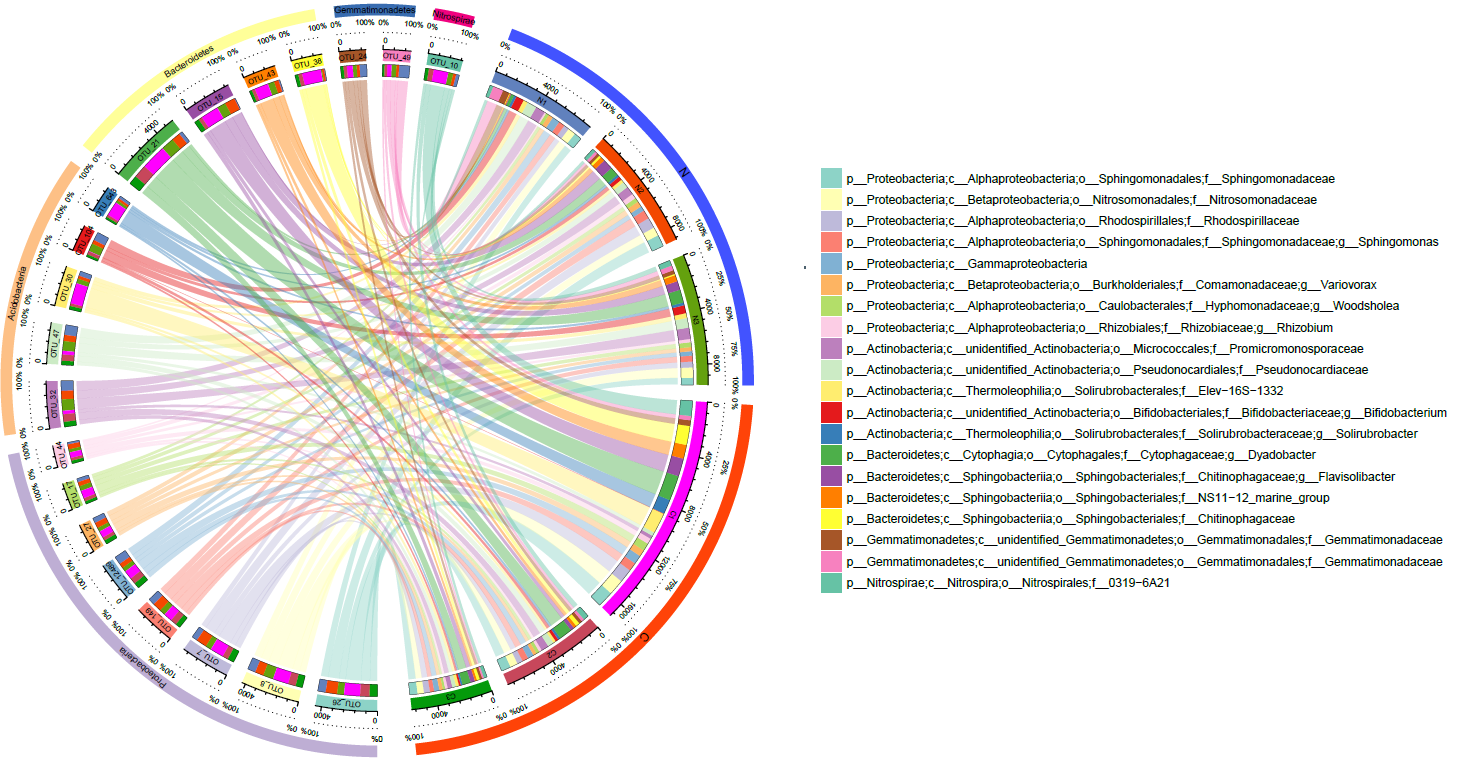
grid\_height = unit(0.5, 'cm'), grid\_width = unit(0.5, 'cm'), type = 'points', pch = NA, background = color\_otu)

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>

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>