最基本的数据类型包括数字和字符串，数字包括整数、小数等，字符串是任何字符的组合，由单引号或双引号包括。它们是其它数据类型的基本组成部分，比如逻辑符号本质就是数字。科学计数法：1e-3 1e2，1L表示1这个整数

2 ^ 6 阶乘

5 %% 2 取余

数学函数log, log2, log10 exp(n)-e的n次方 sin, cos, tan sqrt

|或 &且 ≠0的数都是T

赋值：x<- 1 x = 1 1 -> x assign( "x", 1)

ls()显示当前环境下所有变量 rm( x )删除一个变量 rm(list=ls())删除当前环境下所有变量

特殊值：NULL：值未定义的表达式和函数；Inf：无限值（数字），分为+inf和-inf；NA：不可得到的值，不是有限值；NaN：不是数字（不存在）,属于NA的一种

c(-1, 0, 1) / 0 -Inf NaN Inf

判定特别值：is.finite()是不是有限值 is.infinite() is.na() is.nan() is.null()

判断数据类型：is.numeric() is.list()等，isTRUE函数只在参数是一个逻辑运算值true时才返回true，若传入的不是一个逻辑运算值，而是一个数字/字符等就都返回false

(x)中括号作用：如果不加括号，就不会输出x的值。括号将(x)整体作为一个表达式，这个表达式的值等于x的值，因此这一行实际上就是一个值，执行时会直接输出这个值

vector类型转换优先级：逻辑类型->数字类型 逻辑类型->字符串 数字类型->字符串

矩阵内的数值必须是同一种类型，输入混合有多种基本数据类型时，矩阵会按上面提到的规则进行强制转换

ab <- c(a,b)合并ab数组 length()-长度 rev()-将数组倒序

矩阵matrix( c(20, 30.1, 2, 45.8, 23, 14), nrow = 2, byrow = T, dimnames = list( c("row\_A", "row\_B"), c("A", "B", "C") ) ) 矩阵的指定长度：nrow行 × ncol列。矩阵长度较小时，输入数据会被截短；而矩阵长度较大时，输入数据则会被重复使用，byrow-是否按行将数据转化为matrix，dimnames指定矩阵的行名和列名

vector和矩阵都是数组。vector是一维数组，矩阵是二维数组。还可以有更多维的数组，高维数组与vector和矩阵一样，只能包含一种基本数据类型。高维数组可以由函数 array() 定array(data=LETTERS[1:16],dim=c(2,4,2),dimnames=list(c("A","B"), c("one","two","three","four"), c("一", "二") ))

m[1:2, ] 按行数取行 把1:2放逗号后面就是取列，结果都是vector

m[c( "row\_B", "row\_A" ), ] 按行名取行

m[1:2, 2:3] 取其中一部分

m[ 1:2 , ] <- matrix( 1:6, nrow = 2 ) 替换

a[is.na(a)] <- 0 将a中所有的NA替换为0

常用基础函数：

which()-遍历传入的vector，当vector中某元素满足判断条件时，which函数返回其索引

判断数据类型：str（）、class（）、typeof（）、mode();其中class、typeof显示整体数据类型，str显示变量内部每个元素类。mode函数只查看数据的大类，class函数查看数据的类，typeof函数则更细化，查看数据的细类

抽样：sample(x,size,replace = FALSE) x:数据size:抽样数replace:F，即抽样数size不能大于x长度；rnorm(n, mean = 0, sd = 1) n为产生随机值个数，mean 是平均数（用mu表示），sd 是标准差（用sigma表示）

排序：sort()-输出排序后新序列 order()-输出新序列的每个元素在原序列中的位置

数据计算：range()-输出最大值和最小值 summary()-min,max,median,mean,1、3等分点 t()取二维矩阵的转置

改变数据类型：names( ab) <- as.character( ab) names把某个参数/行/列的名称改变，as.character将数字对象转为字符串类型

seq(0,10,along.with=c(1,10,2)), along.with=Id就是说seq生成数据序列的长度与Id序列长度相同，输出1 5 10

unique()-返回一个没有任何重复元素或行的向量、数据框或数组

subset(a,logical expression)根据逻辑表达式筛选a中的数据，返回令表达式为T的a中数据

vectorisation：R最重要的一个概念，核心在于数据自动循环使用

ab[3:6] <- c("2","3") 遍历赋值

c(1,2)\*c(2,3)就是c(1\*2,2\*3)

x<- c(4,6,5,7,10,9,4,15) ;x < c( 8, 4 )- x中的元素依次与8 4 8 4 8 4 8 4进行比较

data.frame与tibble: 二维表格，由不同列组成；每列是一个vector，不同列的数据类型可以不同，但一列只包括一种数据类型（int, num, chr …）各列的长度相同

data.frame( data = sample( 1:100, 10 ), group = sample( LETTERS[1:3], 10, replace = TRUE), data2 = 0.1 )创建3列，分别为data、group、data2

tribble(

~x, ~y, ~z, #~表示列名

"a", 2, 3.6,

"b", 1, 8.5)

tb<- tibble( x = character(), y = integer(), z = double() )规定这一列的类型

tb <- add\_row( .before=2)增加行, 在第二行之前插入 add\_column增加列

相互转换：as\_tibble()和as.data.frame

区别：

（1）tibble可以按顺序计算列值（将前一列的值用于创建下一列的值）tibble(x = 1:5, y = x ^ 2)，而data.frame不行

（2）tibble取一行/一列时(t[1,])返回的仍是一个tibble，而data.frame取一列(df[1,])时返回的是一个vector

（3）tibble可以进行可控的数据类型转换（使用[[]]或$时返回一个vector）

（4）tibble在根据列名取某列时不会部分匹配，更准确（data.frame(abc = 1)$ab不会报错，而tibble会报错）

（5）构造自动循环补全：tibble-仅限于长度为1或等长,而data.frame则为整除（data.frame(a = 1:6, b = LETTERS[1:2])可以），而tibble不行，只能tibble(a = 1, b = 1:3)

常用函数：

head(n=),tail(n=)显示前/后n行数据

nrow()行数 ncol()列数 dim()维度 rownames()取出所有行名

str()表格结构：每列都是什么数据类型、有多少数据、数据值

rbind()：根据行进行合并 cbind():增加列 如果一个df的行数少于另一处，就循环调用行数不足的vector

bind\_rows()：合并tibble merge():默认按照相同的列名合并

tibble::enframe: 将vector或list转换为一列或两列的tibble。如enframe(c(1,2,3))，结果是两列的tibble，列名分别为name和value，列值都是(1,2,3)

res$Total=rowSums(res[,col\_list])求指定列的和，同理有rowMeans()

attach(iris)：之后可以直接用sepal直接代指iris$sepal detach(iris)取消该操作，with(iris, head(Sepal.Length)也可

within(airquality,{rm(Day)})：删除airquality中Day列

knitr::kable()：以更清晰的形式展示tibble

readr包：read\_csv()逗号分隔符文件

read\_tsv(): tab separated files用tab分隔的文件

read\_delim(): general delimited files

read\_fwf(): fixed width files

read\_table(): tabular files where columns are separated by white-space.读取txt文件

read\_log(): web log files

使用：

read\_csv("data/talk03/iris.csv", col\_types = cols(Sepal.Length = col\_double(),Species = col\_character()))col\_character()指定读取的列类型, col\_names = FALSE不读取文件中的列名, col\_names = col\_name指定列名,n\_max = 7是从开始读取往后7行, quote = FALSE使输出的文本string两边没有引号, skip跳过前n行读取9

write\_tsv(iris, "iris.tsv", quote\_escape = "none")将iris写入"iris.tsv"中，同理有write\_csv、write\_delim、write\_excel\_csv、write\_file、write\_lines、write\_rds、write\_tsv

系统自带读取函数read.table、read.csv: header = TRUE显示列名, skip跳过前n行读取，quote消除string数据前后的字符, na.strings = NA让空缺数据显示为NA,col.names = col\_name指定列名,nrows = 7是从开始读取往后7行,sep=”;”让读取时按分号捕获数据

R session工作空间：save.image(file = "xx.RData")保存 load(file = "xx.RData")载入 save(city, country, file="1.RData")保存指定变量

向量：as.factor()，ordered = T参数时factors排序。Factors会限制输入数据的选择范围：新加入的值必须是levels里已存在的值，levels不可更改，若更改需调用函数levels(x) <- c(levels(x), "widowed”), levels的顺序决定了排序的顺序。用as.factor创建factor时，得到的levels按字母表排列；但是，用levels(y)方式指定levels时，则按照指定的顺序，sort函数也不会改变已指定的levels顺序

reorder(a,b,func)就是根据b对a:b这个键值对进行排序，排序方法为func（可为median等）

forcars包：

fct\_c：合并两个factor

fct\_inorder：按照第一次出现的次序排列

fct\_infreq：按照出现的频率排列（从大到小）

fct\_inseq：按照数字大小排列（从小到大）

fct\_count( gender );#统计gender里的levels数量

fct\_collapse(gender,Male=c("m", "male "))把gender中c("m", "male ")这些levels都改成Male，或使用fct\_relabel

magrittr包提供管道操作，导入tidyverse包时会自动导入该包，%>%在使用时放在两个操作之间，前一个操作产生的数据将作为后面操作的输入数据，此外还有%$%等价于attach、%<>%代表双向数据传递等其他形式的pipe、%T>%返回上游的值，所有函数都支持pipe。通常需要用 . 指代传递来的数据，并以参数的形式赋予下游函数。

res <- matrix(ncol = 2) %T>% plot() 此时res是matrix

mtcars %$% cor.test( cyl, mpg ) 等效于with( mtcars, cor.test( cyl, mpg ) )

#a %in% b 判断a是否在b中，若在就返回true

dplyr包：

select() 选择列，根据列名规则，select(CHR = `Chromosome/scaffold name`)

filter() 按规则过滤行，filter( `Transcript type` %in% c( "protein\_coding", "miRNA", "lincRNA" ) )

mutate() 增加新列，从其它列计算而得（不改变行数）

summarise() 将多个值转换为单个值（通过mean, median, sd等操作），生成新列（总行数减少，通常与 group\_by配合使用 ），summarise( count = n\_distinct( GENE\_ID ))

arrange() 对行进行排序，arrange(-grade) 按成绩由高到低排序

group\_by() 分组，group\_by( CHR, TYPE )或group\_by( CHR )

gather( course, grade, -Name, -Occupation, na.rm = T ) 除了name和occupation列，都删去，转成course和grade列，其中第一个参数即course做键（接收原列的列名），第二个参数即grade做值（接收原列的列值）

mutate() 创建新列，mutate(bmi= mass/((height/100)^2))

行操作：

distinct()：删除重复行

sample\_frac(a,0.5)：随机选出50%的行

sample\_n(a,10)：随机选出10行

slice(a,1:5)：切片，选1-5行

top\_n(a,n)：选择并排序前n个条目

tidyr包：对应系统函数reshape2（重塑）/reshape（聚合）

宽数据（列多）--优点：自然，易理解；缺点：不易处理，稀疏时问题较大

pivot\_longer() 代替 gather，pivot\_longer( - name, names\_to = "course", values\_to = "grade"，values\_drop\_na = T)第一个参数要改变的列（- name表示除了name的剩下列），第二个参数是把原列名变成一个新列的列名，第三个参数是原列对应的值变成的新列的列名，values\_drop\_na可消除值为NA的行；或pivot\_longer(!c( name, class ),..)，!加上一个vector，元素是想保留的列，!相当于前面的-name中减号的作用

pivot\_wider() 代替 spread，pivot\_wider( names\_from = "course", values\_from = "grade" )列名来自哪列，列的值来自哪列

complete( name, course )：用于把隐式的缺失值转换成正常显示的缺失值。把grades按name和course分组，因为有3种name和5种course，分组后应有15行，而原数据只有9行；所以按此分组，原数据存在隐式的缺失值，complete可以将隐式缺失显示出来，使结果有15行，多出来的6行的grade列用NA代替

PPI network：使用igraph包，除此之外还有ggnet、interactive networkD3 R package、plotly.js、D3等

toppart <- ppi %>%

filter( gene1 == "KIF23" ) %>% arrange( desc( score ) ) %>%

slice( 1:10 )#取前10行（与KIF23最有关的10个基因），要不数据太多

genes <- unique(c("KIF23", toppart$gene2)) #得到"KIF23"与gene2的并集genes

netdata <- ppi %>% filter( gene1 %in% genes & gene2 %in% genes )#挑选出gene1和gene2列都在genes里的行（11个基因的相互联系）.因为是网络图，不能只是KIF23与其它基因的关系，还有其它基因之间的联系，所以要再在ppi总集里取10个基因

netdata.nr <- netdata %>% mutate( group =

if\_else( gene1 > gene2, #相当于if(gene1 > gene2) {str\_c( gene1, gene2, sep = "-" )} else {str\_c( gene2, gene1, sep = "-" ) )}

str\_c( gene1, gene2, sep = "-" ),str\_c( gene2, gene1, sep = "-" ) ) ) %>%

group\_by(group)%>%slice(1) #按group排完序之后，每组有2个元素，slice（1）只取第一个元素

netnet.nr <- graph\_from\_data\_frame( netdata.nr, directed = FALSE ); plot(netnet.nr)

计算每个基因的连接数： netdata %>% group\_by( gene1 ) %>% summarise( degree = n() ) #group\_by( gene1 )后每个基因都是一组

宏基因组：对直接从环境样本中回收的遗传物质的研究

排完序后将后面的丰度累加在一起，汇总为"Others"分类：使用tidytidbits:: lump\_rows( scientific\_name, relative\_abundance, n = 10, other\_level = "Others" ) ，(relative\_abundance, n = 10)保留前10行的相对丰度，并把剩下列的相对丰度加和，(scientific\_name,other\_level = "Others")把加和结果的scientific\_name列命名为Others

把 Others 和 Unknown 放在最后：fct\_relevel( fct\_reorder( abu.dat$scientific\_name, abu.dat$relative\_abundance, .desc = F), "Unknown", "Others" )，fct\_reorde的结果中虽然有"Unknown"和"Others"，但fct\_relevel会优先读取"Unknown", "Others"这两个参数放到scientific\_name列的最前面并锁定位置，之后再把reorder的结果排列

数据的整合：

使用 pivot\_wider：先合并，再pivot\_wider用bind\_rows合并两个tibble时，列名需要一致

expr %>% mutate( group = "rkpm" ) %>% #expr数据的group都为rkpm

select( gene, group, value=rkpm ) %>%

group\_by( gene ) %>% slice( 1 );#如果一个基因出现多次，取第一个为准

bind\_rows( meth2, expr2 )

也可使用left\_join(x,y):返回x中的所有行，以及x和y中的所有列。x中没有匹配项的行在新列中将具有NA值。如果x和y之间有多个匹配项，则返回所有匹配项的组合。

meth %>% pivot\_wider( names\_from = "site", values\_from = "methylation\_score" )

left\_join( meth3, expr2, by = "gene" ) #按gene列进行合并，结果是增加列

list相关内容：

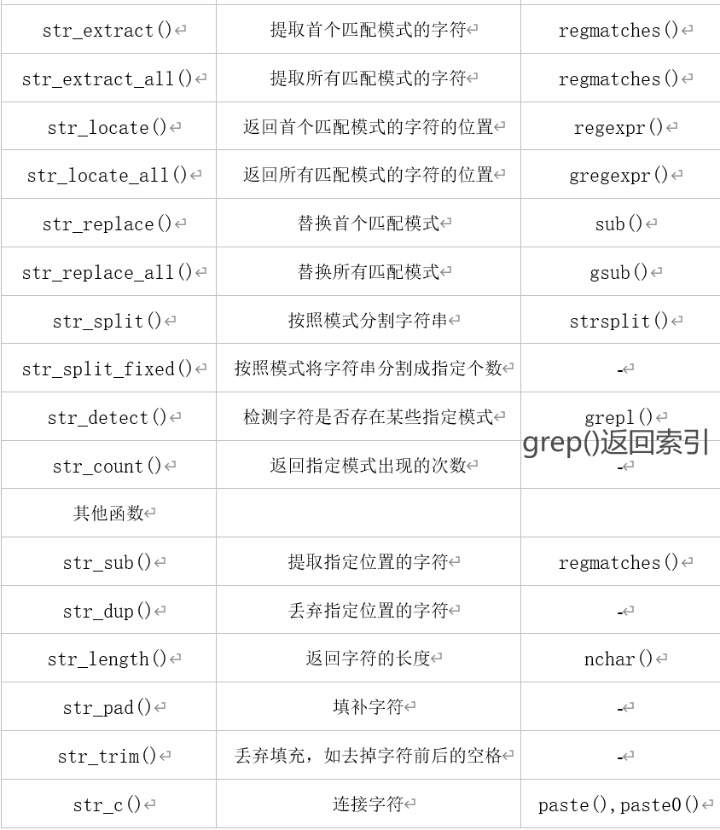
list[[1]]是取该list的第一个元素，可以是vector，若是[1]会返回一个list

data中films列的元素是一个list，该list里面的元素是一个chr型的vector，该列在rmd中输出的是<chr [5]>的形式，[5]表示该chr型vector中有5个元素。在rstudio中此句输出结果与上面tibble的相同，但生成的PDF中输出是里面包含的电影名称，因此data.frame不能实现

stringr和stringi

字符串中要想打单引号，需在前面加上\，即\'输出为‘；字符串中若换行，输出时会用\n替代换行，若使用writeLines()函数，就把字符串在换行处切开为多个字串进行输出（会换行）

"\""输出"\""，'\''输出"'"，"\\"输出"\\"，若用writeLines()函数，会输出转义后形式，且两端无引号



str\_subset 则是返回包含子字符串的字符串

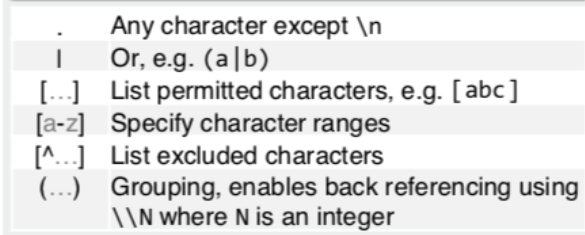
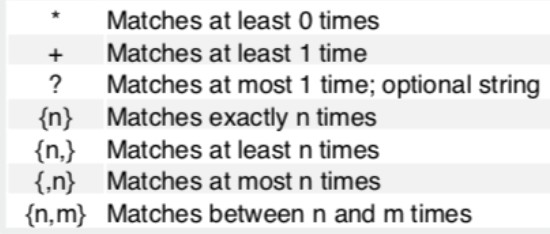
str\_extract 会返回匹配的字符串

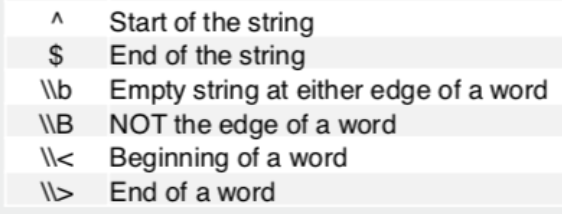
str\_match 抽取匹配的字串，第一列是整体匹配结果，后面是每一组(正则)匹配结果。

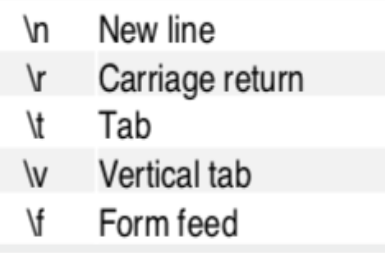
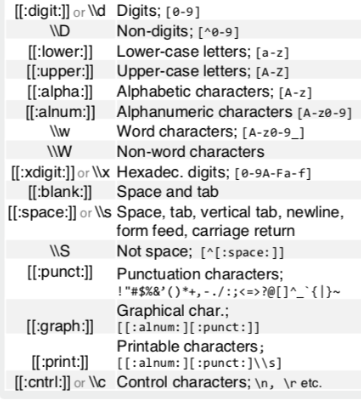
字符串比较：系统-直接"A" > "abc" pracma-strcmp()和strcmpi()，i是忽略大小写进行比较

toupper()小写变大写 tolower()大写变小写（系统自带）

倒序输出：stringi-stri\_reverse字符规则







stringr的其它函数

fixed(): match exact bytes

coll(): match human letters

boundary(): match boundaries

str\_to\_title()将首字母大写

统计元音个数：str\_count(x, "[aeiou]")

反向互补序列：seq %>% str\_replace\_all("A","t") %>% str\_replace\_all("T","a") %>%

str\_replace\_all("C","g") %>% str\_replace\_all("G","c") %>% toupper() %>% stri\_reverse()

填充字符串：str\_pad(处理类型,width,side=c("left","right","both"),pad="")，处理类型:字符串/向量 width:字符串填充后的宽度 side:从哪边开始填充，默认从左边 pad:用什么字符串填充

提取字符串中的大于等于2次的重复字段：str\_extract(x,"(\\w{2}){2,}")

有理数识别（123、-1.5等）： "^([\\-|\\+)?\\d+(\\.\\d+)?$](file:///\\-|\\+)%3f\\d+(\\.\\d+)%3f$)"，

^(\-|\+)?中\-和\+分别代表-和+，|是“或”，?表示-和+可有可无，整体表示以1个-或+或什么也不加开头（\\为转义符号，实际上是\），\d+表示至少要有1个数字，(\.\d+)?中\.表示小数点，\d+表示小数点后至少要有1个数字，?表示小数点和小数点后数字这部分可有可无

系统自带循环函数：apply，对应dplyr包；tapply—dplyr::filter sapply/lapply—dplyr::select

df %>% apply( ., 1, median )取行的 median，df %>% apply( ., 2, median )取列的 median

df %>% apply( ., c(1,2), median )取both的 median：计算每个元素的中值，结果的n行m列的元素就对应原数据集中的n行m列的元素的中值

系统自带的dplyr行操作：tapply

mtcars %$% tapply(mpg,cyl, mean)按cyl分组，相同cyl的为一组，计算每组的mpg的平均值，等效于mtcars %>% group\_by( cyl ) %>% summarise( mean = mean( mpg ) )

系统自带列操作：lapply和sapply--对vector每次取一个element，对data.frame, tibble,matrix 每次取一列，对list每次取一个成员。若输入是tibble, matrix, data.frame时，功能与apply( x, 2, FUN)类似，如lapply( mean )，两者区别是展示数据的方式不同（list和表格形式）

purrr包

对应lapply：map( FUN )遍历每列(tibble)或每个元素(list)，运行FUN函数，将计算结果返回至list，如df %>% map( summary ) 对每一列都进行summary操作

对应sapply：map\_lgl() makes a logical vector；map\_int() makes an integer vector；map\_dbl() makes a double vector；map\_chr() makes a character vector.FUN只能返回单个double/character/…值

mtcars %>% split( .$cyl ) %>%#将mtcars按cyl列分为三个tibble，返回值为包含它们的list，而group\_by只是隐式分组，仍返回原数据

map( ~ cor.test( .$wt, .$mpg ) ) %>% #.在pipe中代表从上游传递而来的数据在某些函数中比如cor.test()，必须指定输入数据可以用.代替

map\_dbl( ~.$estimate );#map( function(df) { cor.test( df$wt, df$mpg ) } )等价于map( ~ cor.test( .$wt, .$mpg ) )，map\_dbl等函数同理 **（~符号作用）**

更多函数：

map2(mu,sigma,rnorm,n = 5)依次将mu的第i个元素和sigma的第i个元素传入rnorm中，rnorm函数运算结果即为输出list的第i个元素

pmap(.l,.f,...)其中.l为数据框，.f为要应用的多元函数。.f是几元函数，对应数据框.l有几列，.f 将依次在数据框.l的每一行上进行迭代，即.f接收.l第i行的所有数据作为自己的参数，.f的输出作为结果的第i个元素

reduce(list(c1,c2,c3),function(x,y){intersect(x,y)})：取交集，将c1和c2传入function(x,y)中，并将函数返回的结果和c3继续传入，直到没元素可传就输出结果

accumulate(c(1,2,3,4),`+`)：累加x中的元素，返回c(1,1+2,1+2+3,1+2+3+4)

keep/discard：接收一个判断条件的函数，当即传入函数中的列满足条件时，保留/舍弃该列

并行计算：需要三个步骤--分解并发放任务、分别计算、回收结果并保存

parallel包：显示CPU core数量，将全部或部分分配给任务，包括parLapply、parSapply、parRapply、parCapply等。foreach包：提供%do%和%dopar%操作符，以提交任务，进行顺序或并行计算，.packages=c()可将需要的包传递给任务。如果每个任务需要提前装入某些包，可以此方法。嵌套循环：foreach(..) %:% {foreach(..) %dopar% {}}，外层的循环部分用%:%操作符

辅助包--iterators 包：将 data.frame, tibble, matrix 分割为行/列 用于提交并行任务。

注意：任务完成后，要回收分配的 CPU core。

cpus <- parallel::detectCores() 检测有多少个 CPU

cl <- makeCluster( cpus - 1 ) make a cluster

res <- foreach( row = iter( d, by = "row" ) ) %dopar% {return ( row$x \* row$y );} 分配任务

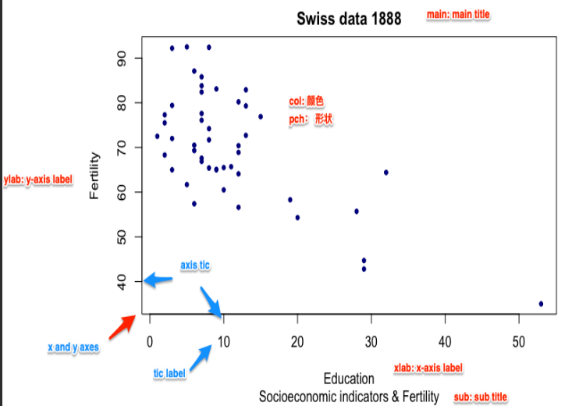
#row = iter( d, by = "row" ):将输入数据d (data.frame)按行/列遍历，每次取出一行或列，赋予row这个变量（可随意取名）；foreach将数据row分发给cl（这里没有体现出来），进行计算 row$x \* row$y ，并返回结果；.combine = 'c'参数规定将返回结果合并为vector，’cbind' : 将返回值按列合并、'rbind' : 将返回值按行合并、默认情况下返回 list

stopCluster( cl ) 注意在最后关闭创建的 cluster

ggplot2：代替基础作图函数plot散点图、abline画线、legend图例、boxplot箱图

优点：1. 将常见的统计变换融入到了绘图中，可以直接输入原始数据就可以绘制统计结果图，较为方便。2.绘图时对数据有整体认识，不会出现plot函数绘制一条曲线确定横纵轴范围后，再添加曲线时超出范围的值无法绘制的情况。3.具有更多的内置函数，可以绘制更复杂的图，绘制的图也更加漂亮，功能更强大。4.图层的概念使其非常符合画图的常识，逻辑很清晰，易于看懂每一步在干什么。

缺点：1.函数太多且复杂，学起来很麻烦。2.在做参数修改的时候，基础函数画一种图，一个函数，所有参数都在这个函数里面，函数可以做的就是所以你可以调节的内容，而ggplot2作一个图需要调用很多函数来完成，修改的时候需要在各个函数中调整，比较难掌握。3.无法针对多个legends进行调整。4.效率不高，绘图速度较慢，表示二次开发的可能性不高。



high level: plotting functions create a new plot在图像区域，包括plot通用画图函数、pairs、coplot、qqnorm、hist、dotchart、image、contour等

low level: plotting functions add more information to an existing plot，包括points点图、lines线图、abline直线、polygon多边形、legend图例、title标题、axis轴 等

可以用 add = TRUE 参数（如果可用）将 high level 函数强制转换为 low level

图形相关参数（系统提供）--par() 函数： 显示或修改当前图形设备的参数。mar = c(1,2,3,4)设置图形边距、mfrow=c(2,3)画2行3列的panel

图形设备是指图形输出的设备，可以将图形设备理解为保存格式。默认设备是：X11() : \*nix、windows() : windows、quartz() : OS X图形显示在显示器上。常用其它设备有：pdf()、png()、jpeg()分别对应输出文件格式，默认文件名为 Rplots.pdf，必须用dev.off()关闭。关闭后，返回到最近使用的图形设备，height 和 width 参数的单位是 inch，如果运行多个 high level 作图命令，则会产生多页pdf。尽量使用 pdf 作为文件输出格式，生信图片大多是点线图，适合保存为矢量格式（如pdf, ps 等），矢量图可无限放大而不失真（变成像素），可由 Adobe Illustrator 等矢量图软件进行编辑

ggplot2参数：aes (aesthetics)美学：控制全局参数，包括：x,y轴使用的数据，颜色(colour, fill)，形状(shape)，大小(size)，分组(group)等等；图层：geom\_<layer\_name> 每张图可有多个图层（此处有两个）；图层可使用全局数据(df)和参数(aes)，也可以使用自己的aes和数据

图层可从ggplot继承数据和其它参数，包括aes，但也可使用自己的。xy -axes 会随数据自动调整，ggplot2作图结果可以保存在变量中，并可累加更多图层，图层使用自己的数据时，需要用data = 指定而全局数据则不用 ggplot ( data.frame( ... ) )

ggplot2作图的四个基本组成部分/四要素/ layered grammer图层语法成分：

1.图层geom\_<图层名>：geom\_point , geom\_line: 点线图，用于揭示两组数据间的关系；geom\_smooth：常与geom\_point联合使用，揭示数据走势；geom\_bar：bar图；geom\_boxplot : 箱线图，用于比较N组数据，揭示区别；geom\_path : 与geom\_line 相似，但也可以画其它复杂图形；geom\_histogram, geom\_density : 数据的分布，也可用于多组间的比较

2. scale 显示控制：scale\_<控制内容>\_<控制手段>，四种控制类型：（1）scale\_color\_...e.g. scale\_color\_manual()以手选方式控制颜色、scale\_color\_brewer( palette = "<palette name>" )色板、scale\_color\_gradient() 改变渐变颜色、scale\_color\_gradient2()使用3个渐变色、scale\_colour\_gradientn更多颜色、scale\_colour\_binned更多渐变色（2）scale\_fill\_...（3）scale\_shape\_...（4）scale\_size\_... aes(size = cyl) 用 cyl 列的值确定点大小

ggplot(colour = Species)：根据Species列决定颜色, 由默认色板决定(根据需要的颜色数量，取相应的行)

ggplot2中颜色：fill与colou区别：colour defines the colour with which a geom is outlined (the shape's "stroke")；fill defines the colour with which a geom is filled。Points generally only have a colour and no fill.However, point shapes 21–25 that include both a colour and a fill. colour 在 shape = 21时，为描边（stroke）色，可用 stroke 控制 线条粗细；fill是填充色。要做到两个匹配，即aes 参数 (如 fill =) 与 函数 (如 scale\_fill\_xxx) 匹配（colour 也是如此）、数据类型与色板类型匹配--数量配渐变，离散配离散；两者色板是通用的，函数也是，比如 scale\_fill\_xx 对应 scale\_colour\_xx；有些 shape / pch 只有 colour (stroke)，有些则两者都有。ggsci: 论文发表用的色板，包括scale\_color\_npg等

size, colour 等参数可在aes()内部，也可在外部，区别：在内部时，以指定列的值确定大小，或按 factor 的数量确定颜色、形状的数量；在外部时，则以指定值为准

3. 坐标系统--线性坐标系统coord\_cartesian(),coord\_flip(),coord\_fixed();非线性坐标系统coord\_trans()、coord\_polar()、coord\_map()。默认的坐标系统，可使用xlim, ylim等参数，实现缩放局部。线性系统：coord\_flip() exchange the x and y axes、coord\_fixed()改xy刻度对应图中的长度；非线性：coord\_trans()通过改刻度修改曲线、coord\_polar(), 柱图变饼图、coord\_map()画地图

4. faceting同时作多个图，每个图显示不同的数据子集。facet\_grid( cyl ~ . );#按cyl分成三个图，scales = "free"根据每张图的数据范围确定x轴范围

如何在一张图中画多个panel：关键要素-- order / position、labeling、layout，使用包—gridExtra、cowplot、grid、lattice、patchwork，函数-- cowplot::plot\_grid、cowplot::draw\_plot调整相对大小、gridExtra::grid.arrange

写公式：将两个公式上下放置 atop ( <equation\_1> , <equation\_2> )；将公式中的某些值替换为数值substitute(<equation>,list())。eq<- substitute(atop(italic(r)^2~"="~r2\*",list(r2=...)); eq <- as.character(as.expression(eq)); geom\_text(aes(label= eq)) 其中引号两边必须有 \* 或 ~ 字符，~ 表示空格，\* 表示什么都没有，~~ 表示两个空格；geom\_text中hjust为横向调整，vjust纵向调整。也可使用paste—substitute组合

ggplot2核心在于先计算再做图. default stat behaviors (默认计算方法)：geom\_bar–count、geom\_boxplot-boxplot、geom\_count—sum、geom\_density—density、geom\_histogram—bin、geom\_quantile-quantile。显示数值：使用dplyr的cumsum()函数，计算位置=当前累加值-自身值/2，使数字显示在当前值的中间，累加前，要对数据按 factors 进行排序，通过arrange函数实现。geom\_text中position取值："dodge"、"stack"、position\_identity()--在指定位置，不改变、position\_jitter()--随机往别的地方移动使点不重叠，不同的图层有不同默认值

改变 theme()：theme调整包括theme()函数，用于调整各个elements和theme\_xxx()函数，直接使用已经定制好的内容。ggplot2中的主题：theme\_gray（系统默认主题）、theme\_bw、theme\_linedraw、theme\_light、theme\_dark、theme\_minimal、theme\_classic、theme\_void()

theme()函数用于细调，包括line, rect, text, title整体框架、axis.<compoment>调整坐标轴、legend.<parameter>调整图例、plot.<>控制title, subtitle等细节、panel.<...>调整facet情况下的panel、strip.<...>调整facet的标题细节

labs(): 修改axis, legend, and plot labels

正态分布normal distributions：可用mean和sd描述（它是对称的、平均值和中位数相同、大多数常见值接近平均值、不太常见的值离平均值远、标准差表示从平均值到拐点的距离），(mean+1\*sd)包含68%的数据、(mean+2\*sd)包含95%的数据(mean+3\*sd)包含99% 的数据

Uniform Distribution：均匀分布—runif()、Binomial Distribution：二项式分布—rbinom()、Poisson Distribution：泊松分布—rpois()、Exponential Distribution：指数分布—rexp()

Gamma Distribution：伽玛分布(γ分布)—rgamma()、正态分布—rnorm()

量化描述函数：

mean( norm$dat );#平均值 median( norm$dat );#中值 mode( norm$dat );#数据类型

sd(norm$dat);#标准差var(norm$dat);#测量一组数字的分布程度range(norm$dat);#minmax

quantile(probs = seq(0, 1, length = 11))分位值 summary()对数字：最大最小值、1 3分位值、平均值、中值；对字符：长度、类型；可应用于整个表格;，相当于对每列进行 summary

table函数：返回vector当中 unique 值和它们的出现次数

dplyr::count：统计一列中各元素数量

ggsignif::geom\_signif显著性检验，geom\_signif(comparisons = list(1:2, 2:3, 3:4, 4:5),#第1和2组数据比较、第2和3组数据比较...

test = wilcox.test,#检测方法

step\_increase = 0.1 );#让结果数值分开（下一组比上一组结果高0.1个单位）

geom\_smooth( method = "lm" ); #默认是lowess，lm是线性回归

当趋势不明显时，可以按另一组数据分组，两种分组（binning）方法：equal-distance(cut)--取最大最小值，分为多个区间，把数据值在相同区间的行分为1组、equal-size binning(ntile)：每组的数量相同。cut 函数：按指定的间隔 (breaks) 对数据进行分割，不仅可用于 equal distance，还可以用于任意间距

parametric tests参数检验：基于总体服从某种特定概率分布的假设的，对于不同的总体参数，有不同的参数检验，比如T检验，F检验，Z检验都是参数检验，检验均值，方差等等。包括t-test、analysis of variance、linear regression；用于数据有较明确的分布（例如正态分布），或假设数据有明确的分布；当假设不成立时，检测会无效；更灵敏（相比nonparametric test），p-value更低。适用于：数量化性状（身高、体重、产量、污染值）、整数值（成绩、年龄、每天步数），不适用于其它 count data或者discrete data、或者有太多趋向于min或max的值、百分比或比例。需要psych和rcompanion包。使用t.test() anova()等函数

parametric test 的要求：随机取样、值或 residuals 为正态分布（residules 是指观察值与预测值(mean)之差）、有相同的variance方差

异常值是远远超出其他观测值的极值。例如，在正态分布中，异常值可能是分布尾部的值。对于 normal distribution，通常 mean +- 2 or 3 \* sd；对于 non-parametric distribution是第1等分点-1.5\*IRQ和第3等分点+1.5\*IRQ之外的值

two samples t-test：比较 sd 和 mean，可应用于正态分布。t.test( Steps ~ Sex )或t.test( Steps[ Sex == "male" ], Steps[ Sex == "female" ])

ANOVA: similar to independent t-test, but can be applied to multiple groups主要分析一个变量（多组）对另一变量的贡献度/解释度/影响度。比如：年级对学生的体重是否有影响

two way ANOVA：一个变量受另外两个因素影响model3<-lm( wt~class+age+class:age, data = wts2);anova( model3 ) 其中class和age称为main effects，class:age称为interaction effects

repeated measures ANOVA：同一变量、不同时间段的重复测量

相关性检验：cor(x,y)

获取p值：cor.test(a,b) $p.value

R方：with(data, cor.test(a,b)$estimate )^2或m=lm(y~x,data); summary(m)$r.squared

y~x的线性回归公式：y=a+bx a= coef(m)[1] b= coef(m)[2]

用模型进行预测：model2 = lm(formula = wt ~ class + age, data = wts2); wt.predicted <- predict( model2, newdata )

在一个linear model（线性模型）中， wt = intercept + a \* class + b \* age

linear regression线性回归：是 parametric test、假设变量之间独立（比如：年龄和班级之间没有关联）、homogeneity of variance方差齐性

non-parametric test非参数检验：不考虑总体分布是否已知，常常也不是针对总体参数，而是针对总体的某些一般性假设（如总体分布的位置是否相同，总体分布是否正态）进行检验。使用的函数：wilcox.test()、kruskal.test()等。适用于：非参数性质使它们适用于不符合参数分析假设的数据，并且适用于区间/比率或序数因变量。

wilcox.test和t.test区别：t.test是一种用于比较两组数值型数据之间平均值差异的方法，当两组数据来自正态分布的总体时，通常可以使用t.test，其中p.value衡量了两组数据的平均值之间是否存在显著差异，p.value越小代表差异越大；wilcox.test是一种非参数检验方法，用于比较两组数据的中位数差异，适用于数据不满足正态分布假设，或者更想关注数据的中位数差异时，p.value衡量了两组数据的中位数之间是否存在显著差异，p.value越小代表差异越大。如果数据满足t.test的条件，且平均值差异更重要，则t.test的p.value更显著；如果满足wilcox.test的条件，且中位数差异更重要，则wilcox.test的p.value更显著。

线性回归：是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，Y可以被一个变量X解释--一元线性回归，Y可以被X, Z等多个变量解释--multivariate linear regression。

统计函数：m为lm的线性模型

coefficients(m)--model coefficients、confint(m, level=0.95)-- CIs for model parameters、fitted(m)--predicted values、residuals(m)—residuals、anova(m)--anova table、vcov(m)--covariance matrix for model parameters、influence(m)--regression diagnostics

RMSE(predictions, 原数据)--均方根误差，越小模型拟合效果越好

R2(predictions, 原数据)--值越->1，模型拟合效果越好

multivariate linear modeling ：

m=lm( sales ~ youtube + facebook + newspaper, data = marketing )

coef(m)--取summary的estimate列，值越大影响越大

relaimpo:: calc.relimp-- relative importance analysis

interactions考虑因素之间的依赖关系或互作关系

lm(a~b+c+b:c)等价于lm(a~b\*c)

任何检验都有基本的假设，将检验应用于不符合假设的数据是统计学最大的滥用。线性：X和Y的平均值之间的关系是线性的；同调性：残差的方差对于X的任何值都是相同的；独立性：观察是相互独立的；正态性：对于X的任何固定值，Y都是正态分布的。

glm 还可用于其它类型数据的分析，如Logistic regression、泊松回归（一种特殊类型的回归，其中因变量由计数数据组成，观察是独立的，该模型的均值和方差相等）

Non-linear regression (nls)：使用由参数和一个或多个自变量组成的非线性函数来预测目标变量，使用同名函数nls()计算（系统自带）

non-linear least squares, using drc包

aomisc 包和 drc 包带了非常多数据生成的函数

non-linear functions：Polynomials--Linear equation、Quadratic polynomial；Concave/Convex curves (no inflection)—Exponential/Asymptotic/Negative exponential/Power curve/ Logarithmic equation、Rectangular hyperbola；Sygmoidal curves—Logistic/Gompertz/Log-logistic equation、Weibull-type 1/2；Curves with a maximum--Brain-Cousens equation

Exponential equation：(Exponential decay) y = a \* (exp(k \* X) \* k)

Power curve：a\* (X^(b-1) \*b)使用drm()，DRC.powerCurve作为建模的参数, 来自aomisc包

多元 non-linear regression—mars：Multivariate Adaptive Regression Splines (多元自适应回归样条)、machine learning

MARS提供了一种方便的方法来通过评估切点（结）来捕捉数据中的非线性关系。该过程将每个预测器的每个数据点评估为一个结，并创建具有候选特征的线性回归模型。

参数调整—采用交叉验证cross validation方式，做K fold, N times折叠，每个iteration为随机split并得到 K \* N 个模型。非常耗时，尽量使用分布式计算

variable importance plot 使用VIP包

MARS优点：能自然地处理混合类型的预测因子（定量和定性）；需要最小的feature engineering（例如，特征缩放），并执行自动特征选择；高度相关的预测因子不会像OLS模型那样阻碍预测准确性。缺点：通常训练速度较慢；此外，尽管相关的预测因子并不一定会阻碍模型的性能，但它们可能会使模型解释变得困难。