- **Rclass_1**
  - 简单操作
    - +-*/
    - ^, sqrt(), exp()
    - 其他
  - 向量基础
    - 定义
    - 连结
    - 快捷生成
    - 序列
    - 重复
    - 排序
    - 长度
  - 向量操作
    - 索引
    - 运算
      - 赋值
      - 减法
      - 乘法
    - 其他运算
  - 矩阵
    - 定义矩阵
    - 合并行和列
    - 一些函数
    - 索引
    - diag：对角元<=>对角阵
    - 赋值
    - 矩阵运算
      - 转置
      - 乘法
      - 求逆
  - 多维数组
    - 定义
    - 索引
- **Rclass_2**
  - 逻辑数据、运算和索引
    - 布尔型
      - 与数值的关系
    - 关系运算符
    - any和all
    - 逻辑运算符
    - 布尔索引
  - 字符
    - 基础
    - 连接
      - 连接输出
      - 连接
    - 转义字符
    - 子串
    - 匹配
  - 因子
    - 基础
    - 有序因子
    - 合并
    - 切割（离散化）
  - 列表和数据框
    - 列表
    - 数据框
  - 特殊值
    - Inf
    - NaN(Not a Number)
    - NA(Not Available)
    - NULL: define an empty entity
  - 类型
    - 属性
    - 类型

- is判断函数
- as转换函数

Rclass_3

- 基础绘图
  - plot
    - type
    - main, xlab, ylab
    - col
    - pch
    - cex
    - lty
    - lwd
    - xlim，ylim
    - sub
    - 数字序号展示
  - 其他函数
    - abline
    - segment
    - points
    - lines：绘制折线
    - arrows
    - text
    - legend
  - plot进阶
  - par：修改参数
    - 拷贝当前参数
    - 修改参数
    - 再改回原来的参数
    - 画子图
  - RColorBrewer
    - 取色函数
    - 展示手段
- ggplot2——qplot
  - qplot
  - geom类
    - aes
    - geom_point
    - geom_line
    - geom_hline
    - geom_segment
    - labs
    - geom_smooth
    - geom_histogram
    - geom_boxplot
    - geom_rug
    - geom_density
    - geom_bar
- ggplot2——ggplot
  - ggplot
  - 用fill等颜色或形状特征来分类
  - 分面
    - facet_grid
    - facet_wrap
  - 用theme函数来自定义主题
  - 保存图片

RClass_4

- 条件语句
  - if语句
  - if-else语句
  - ifelse逐元素判断
  - switch
- 循环语句
  - for循环
  - while循环
  - break，next
  - repeat
- apply族
  - apply
  - tapply

Rclass_1

「 简单操作」

+-*/

^, sqrt(), exp()

其他

- `log`
  - **x**
  - **base**: 默认为自然对数的底数
- 赋值：用 `=` 或者 `<-`
- `ls()` ：展示当前工作空间中包含的内容，即目前定义的变量

「 向量基础」

定义

`c(ele1,el2,...)` : 元素可以是不同类型的

连结

`c(vec1,vec2,...)` ： 直接把待连结的向量作为新向量的元素即可实现连结

快捷生成

- `3:27`：生成从3到27的向量，包含头和尾，步长绝对值默认为1.
- `5.3:-45.5`：生成从5.3到-47+1.5的向量，步长绝对值为1

序列

- `seq()`
  - **from**
  - **to**
  - **by**：指定步长
  - **length.out**：输出个数。与by参数择其一即可

重复

- `rep()`
  - **x**
  - **times**：x整体重复的次数
  - **each**：x中每个元素重复的次数
- 例子：`rep(x = c(1,2,3), times = 3, each = 2)`
  - 返回：1,1,2,2,3,3,1,1,2,2,3,3,1,1,2,2,3,3,

排序

- `sort()`
  - **x**
  - **decreasing**：bool，是否降序

长度

`length(x = ...)`

「向量操作」

索引

- `myvec[myint]`：下标从1开始，而不是0
- `myvec[length(x = myvec)]`：索引倒数第一个元素
- `myvec[myint1:myint2]`：始终包含首尾
- `myvec[-myint]`：删除第 **myint** 个元素后的myvec
- `myvec[myvec2]`：索引出 **myvec2** 对应的元素
- `myvec[myvec2]`：删去 **myvec2** 对应的元素

运算

- 当运算符左右两边向量的长度不相等，则先循环**(rep)**至相等，再进行运算
- 例子：c(1,2,3,4)+c(1,2)=c(1,2,3,4)+c(1,2,1,2)=c(2,4,4,6)，**循环**
- 例子：c(1,2,3,4,5,6,7)+c(1,2,3)=c(1,2,3,4,5,6,7)+c(1,2,3,1,2,3,1)=...，**循环+截断**

赋值

索引出来后都可以**直接改值**

减法

乘法

其他运算

- `sum(myvec)`
- `prod(myvec)`：连乘

「矩阵」

定义矩阵

- `matrix`：
  - **data**: 常是一个向量
  - **nrow**：行数
  - **ncol**：列数
  - **byrow**：是否按列生成，默认T

合并行和列

- `rbind`：将传入的矩阵按行堆叠
- `cbind`：将传入的矩阵按列堆叠

一些函数

- `dim(mymat)`：返回一个向量，表维数
- `nrow(mymat)`：一个数
- `ncol(mymat)`
- `dimnames(mymat)`：行列的名字，默认是NULL，可用在定义时作为参数传入

索引

- `mymat[myint1,myint2]`
- `mymat[,myint2]`
- `mymat[myint1,]`
- `mymat[myvec1,myvec2]`
- 还有对应的添负号的情况，一边添负号的情况等等，只要知道负号表示删去就迎刃而解

diag：对角元<=>对角阵

赋值

- 广播机制同理，默认按列赋值
- 例子：B是3*3阵
  - ```
    B[c(1,3),2:1] <- c(65,-65,88,-88)

    88  65  -7
    1   2   900
    -88 -65 7
    ```
  - 按列赋值，但是索引的方式很灵活，导致赋值的方式也很灵活

矩阵运算

转置

`t(mymat)`

乘法

`mymat1 %*% mymat2`

求逆

- `solve(mymat)` = $(mymat)^{-1}$
- `solve(A,B)` = $A^{-1}B$

「多维数组」

定义

- `array()`
  - **data**：一个向量
  - **dim**：指定维数，也是一个向量
    - 前两个数是每个小矩阵的维数，后面的数决定小矩阵的排布和个数

索引

同矩阵，只是维数多一点罢了

Rclass_2

「逻辑数据、运算和索引」

布尔型

- `T`
- `F`
- 这两个可以参与向量和矩阵的组成
    ```

与数值的关系

- 认为T为数1，F为数0
- 认为非零数都是T，只有0是F

关系运算符

- ==, !=, >, <, >=, <=
- 当两边都是向量时，则为逐元素比较，返回一个向量。矩阵同理。
- 两边长度不一样时，则先循环，再比较

any和all

- `any(myvec)`：有一个T则返回T
- `all(myvec)`：全为T才返回T

逻辑运算符

- 逐元素比较的：
  - &
  - |
- 仅支持比较单个元素的：
  - &&
  - ||
- 例子：
  - c(T,T,F) & c(F,T,T)，返回c(F,T,F)
  - c(T,T,F) && c(F,T,T)，则仅比较第一对元素，返回F
- 运算顺序：`! > && > ||`

布尔索引

- `myvec[boolvec]`：取出T对应位置的元素
  - 当boolvec的长度小于myvec的时，仍然是先循环再索引
- 索引出来支持直接改值
- `which(x = boolvec)`：返回T所在的位置下标，即一个数值向量
  - 当boolvec是关于矩阵的关系运算语句时：
  - 先将矩阵按列展开成向量，即A->c(A[,1], A[,2], ...)
  - 再进行关系运算。
  - **arr.ind**: bool，是否按照原来矩阵的形状传入which里面，并按照原来矩阵的形状进行返回。

```
print(which(x = A > 25))
[1] 3 4 6 9
#print(which(x=c(A[,1],A[,2],A[,3])>25))与上相同
print(which(x = A > 25, arr.ind = T))
     row col
[1,]   3   1
[2,]   1   2
[3,]   3   2
[4,]   3   3
```

- `myvec[-which(x = boolvec)]`：删去T对应位置的元素
- 其他操作与矩阵、向量索引同理

「字符」

基础

- 将字符串看成一个整体，而不是看成一个向量
- `nchar(x = mystr)`：返回字符串的长度
- 可以直接换值
- 比大小：从前往后挨个字母比，直到比出来大小关系为止
  - 一般按照字母表顺序
  - 同一字母，大写字母大于小写字母

连接

连接输出

`cat(...)`：输出连接之后的结果，不自动在末尾加回车，无返回值（或返回NULL）。

连接

- `paste(...)`：返回连接之后的向量
  - **sep**：str，分隔符，默认为空格
- `paste0(...)`：同上，只是默认分隔符变成了空，即无分隔符

转义字符

Escape sequence	Result
\n	Starts a newline
\t	Horizontal tab
\b	Invokes a backspace
\\	Used as a single backslash
\"	Includes a double quote

子串

- `substr`：查看字符串的子串
  - **x**
  - **start**：Int，是子串的起始索引
  - **stop**：Int，是子串的结束索引

匹配

- `sub`：字符串替换，只替换第一个出现的字符串
  - **pattern**：要替换的子串，在原字符串中查找
  - **replacement**：用来替换的字符串
  - **x**：目标字符串
- `gsub`：替换全部符合要求的字符串，参数同上

「因子」

基础

- `factor()`：定义因子
  - **x**：一个向量，里面的元素可以是数字，也可以是字符串等
  - **levels**：指定水平（可缺省）
  - **labels**：给各水平取的别名（可缺省）
- `levels(myfac)`：返回一个字符向量，即因子的水平
  - 可以用于改因子的水平的名字，则因子的内容也会对应地改变
- 布尔索引：`myfac==mystr`：返回一个逻辑向量，T指示对应的元素是**mystr**对应的水平。

有序因子

- `factor()`
  - **ordered**：bool，是否生成有序因子
    - 如果T，则因子水平按照传入**levels**参数的向量给出的顺序从小到大定序。

合并

- `c(myfac1, myfac2, ...)`：合并，合并后水平和值都取并集

切割（离散化）

- `cut()`：化成区间型数据
  - **x**
  - **breaks**：数值向量，其中的元素代表分割区间的端点
  - **right**：bool，T代表左开右闭，F代表左闭右开
  - **include.lowest**：bool，T代表最大与最小的区间端点均取闭，F代表所有区间端点都按照 **right** 参数给定的规则来取
  - **labels**：给原来的水平取别名

「列表和数据框」

列表

- `list(...)`
  - 里面可以放矩阵、向量、字符串、数值等等数据类型
- 索引为：**mylist[[myint]]**，最大是 **mylist[[length(x = mylist)]]**
  - **mylist[myint]**也可起到索引的效果，但是它返回的还是一个list

    - **mylist[[c(myint1, myint2)]]** = **mylist[[myint1]] [myint2]**
    - **mylist[c(myint1, myint2)]**返回列表的第myint1和第myint2个元素
- `names(mylist)`
    - 对它赋值，可以给mylist的各元素取名字
    - 取名字之后，可以用 **mylist$mystr**的形式进行索引，注意**mystr**不需要加双引号，直接写字符串的内容即可
- 嵌套：如将 **mylist1** 作为 **mylist2** 的一个元素
    - 索引时用两次**[[]]**或两次**$**即可

数据框

- `data.frame(mykey1 = myvalue1, ...)`
    - **myvalue**的长度必须都相等，可以是向量或因子
- 索引
    - 可以如矩阵一般索引
    - 也可以用 **[[]]** 和 **$** 如列表一般索引
- 维数
    - `dim()`：返回一个数值向量
    - `nrow()`
    - `ncol()`
- 加数据
    - `rbind`：对齐特征加样本
    - `cbind`：可以直接加列，所加的列可以是向量或因子
    - 用 **$** 也可以齐到加列的目的，直接取名赋值就好
- 布尔索引，同矩阵索引的方法

「特殊值」

Inf

- 表示超出了R语言能表达的最大值，与数学中严格的无穷大定义稍有不同
    - 如90000^100会直接返回Inf，故log(90000^100)还是会返回Inf（log(Inf)=Inf）
    - -Inf
    - 运算等数学性质与数学中的无穷大基本相同
- 判断：
    - `is.infinite(x = ...)`
    - `is.finite(x = ...)`
    - 直接用 **==** 来判断
    - 由数学易知，Inf和Inf之间不能比大小

NaN(Not a Number)

- 所有数学上的不定式都返回NaN
    - **-Inf+Inf**
    - **Inf/Inf**
    - **0/0**
- 判断：`is.nan()`

- 仅限在数值运算之中使用

NA(Not Available)

- 可在任何类型中使用，NaN是数值运算中的NA
- 判断：`is.na()`
- 清洗：`na.omit(object = ...)`
  - 移除NA和NaN

NULL: define an empty entity

- 就是“空”，前面提到的特殊值都不是“空”，都是占据空间和长度的
- NULL是没有长度的，c(2,4,NULL,8) #length = 3
- 判断：`is.null(x = ...)`，是不是空集?
- NULL与数值做运算返回 **numeric(0)**，长度也是0
- 数据框调用（$）不存在的元素名，返回NULL

「类型」

属性

- `attr()`
  - **x**
  - **which**：str，说明要返回x的哪个属性的值
  - 可用 `attributes(x)` 来查看实例对象x都具有哪些属性
    - x一般具有在定义时赋予它的那些属性
  - 可以直接改值

类型

- `class()`
  - 传入向量时，返回向量的元素类型。字符串和数值混杂时，算字符型向量。
    - 特别地，传入有序因子时，返回"ordered" "factor"
  - 传入矩阵时，一律返回'matrix'·'array'

is判断函数

`is.integer, is.numeric, is.matrix, is.data.frame, is.vector, is.logical`

as转换函数

- 隐式转换在布尔型-与数值的关系中提到过。T->1, F->0
- `as.numeric, as.character, as.logical`
  - 不能跨两个类型，如不能：`as.logical(c("1","0","1","0","0"))`
- `as.vector, as.data.frame, as.matrix`

Rclass_3

「基础绘图」

plot

type

- 缺省为散点图
- **l**：直线
- **b**：点线图
- **n**：空白图

main, xlab, ylab

- 标题和轴名

col

- 颜色，可用色彩插件来选择
- 或用数字序号

pch

- 点的形状，用数字序号选择

cex

- 点的大小，是一个浮点数

lty

- 线的形状，用数字序号选择

lwd

- 线的宽度，是一个浮点数

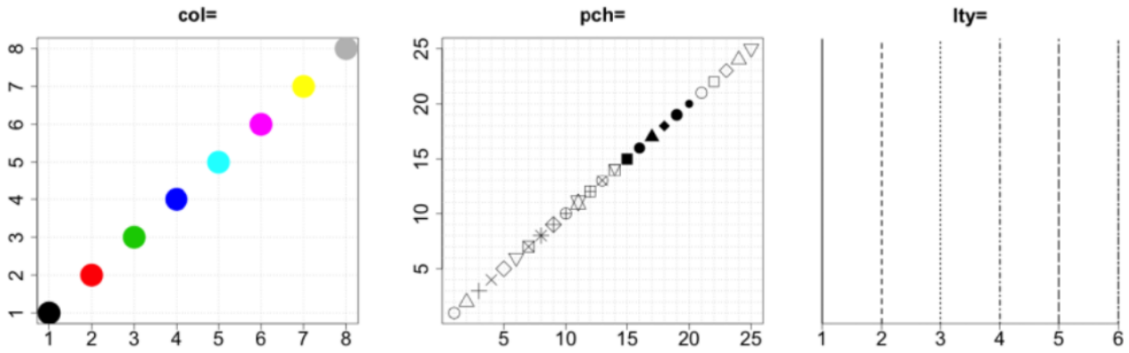
xlim，ylim

- 坐标范围，输入一个向量

sub

小标题

数字序号展示



其他函数

abline

- **a, b**：截距和斜率
- **h**： the y-value(s) for horizontal line(s).
- **v**：the x-value(s) for vertical line(s).
- 其他参数同plot
- 传入回归对象lm时，可画出回归直线

segment

- **x0, y0**：线段的起始点坐标
- **x1,y1**：线段的终点坐标
- 其他参数同plot

points

- 第一个参数x：可为一个向量，代表点的横坐标
- 第一个参数y：可为一个向量，代表点的纵坐标

lines：绘制折线

- 第一个参数x：可为一个向量，代表折点的横坐标
- 第一个参数y：可为一个向量，代表折点的纵坐标

arrows

- **x0, y0**：箭头的起始点坐标
- **x1,y1**：箭头的终点坐标

text

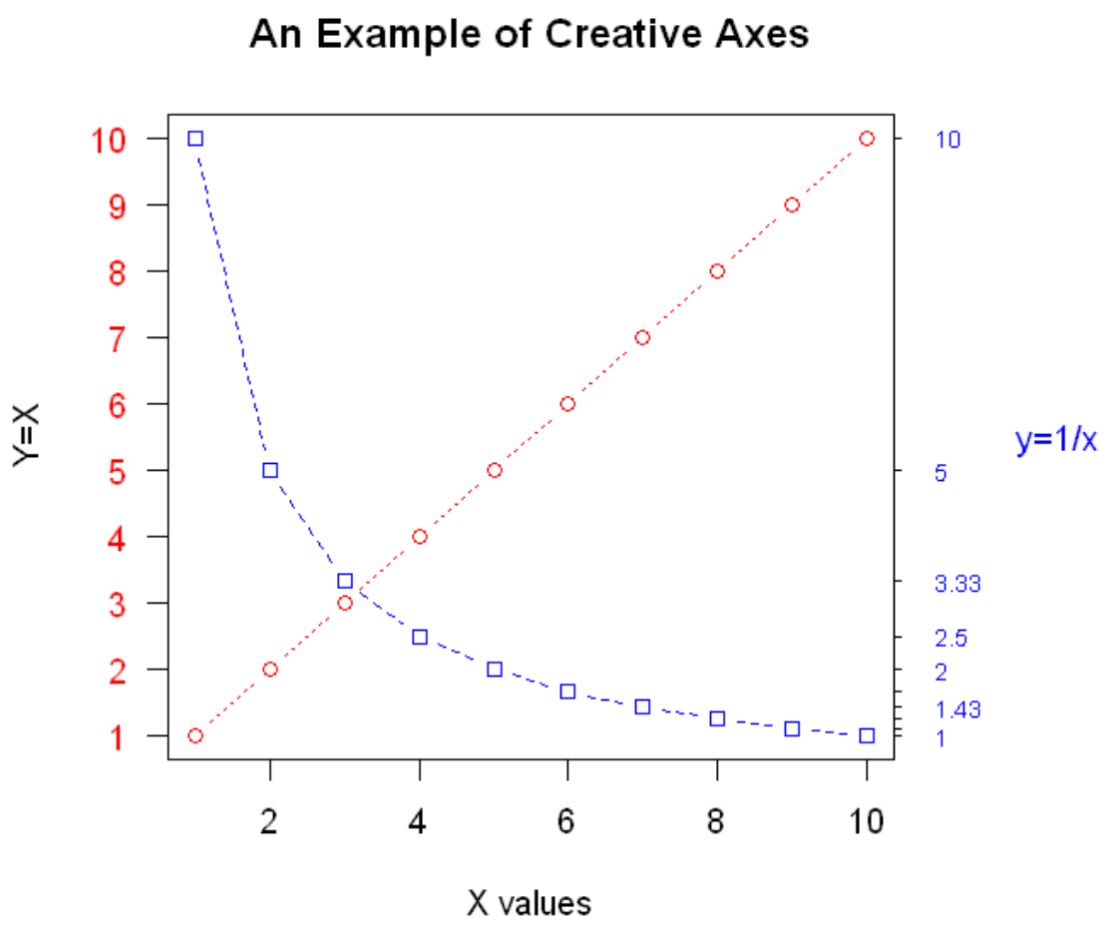
- **x, y**：定位出文本的中点
- **labels**：文本内容

legend

- **legend**：图例中的文本
- **pch, lty, col, lwd, pt.cex**：用来描述点或线。输入一个向量，向量与legend等长
  - 如某位置想要描述一个点，则 **lty=NA**
  - **pt.cex**：点的大小

plot进阶

```
x <- c(1:10)
y <- x
z <- 10/x
opar <- par(no.readonly=TRUE)  #先保存
par(mar=c(5, 4, 4, 8) + 0.1)   #define the margin:bottom, left, top and right
plot(x, y, type="b",
     pch=21, col="red",
     yaxt="n", lty=3, ann=FALSE)   #ann indicating whether the default
annotation (title and x and y axis labels) should appear on the plot
# yaxt indicating whether y axis is plotted.
lines(x, z, type="b", pch=22, col="blue", lty=2)
axis(2, at=x, labels=x, col.axis="red", las=2)   #1=below, 2=left, 3=above and
4=right, las indicating labels are parallel (=0) or perpendicular(=2) to axis
axis(4, at=z, labels=round(z, digits=2),
     col.axis="blue", las=2, cex.axis=0.7, tck=-.01)   #tck for tick mark
length and direction
mtext("y=1/x", side=4, line=3, cex.lab=1, las=2, col="blue") #Text is written
in one of the four margins
title("An Example of Creative Axes",
      xlab="X values",
      ylab="Y=X")
par(opar)
```



par：修改参数

拷贝当前参数

```
opar <- par(no.readonly=T)
```

修改参数

```
par(...=...,...)
```

再进行绘图操作

再改回原来的参数

```
par(opar)
```

画子图

- 使用参数 **mfrow**，传入一个向量c(x,y)，表示新建x行y列的子图

RColorBrewer

取色函数

- brewer.pal
  - n: 取几种颜色
  - name（str）：取哪种套餐（取色板）里的颜色
- rainbow
  - n
- gray
  - level：a vector of desired gray levels between `0` and `1`

展示手段

- barplot
  - 取定横坐标，一般为：`rep(1,n)`
  - col=...
- pie
  - `rep(1,n)`
  - labels=...
  - col=...

「ggplot2——qplot」

qplot

- 参数和用法均与plot相似
- qplot会返回图片对象，而plot没有返回值。据此可以在已有对象上加东西。
- **type='n'**改为 **geom="blank"**
- 给参数 **color, fill, shape** 输入一个分类向量或因子后，会自动给出图例。可据此分类。
- **fill**：是填充区域的颜色，**color**是点、线和边界的颜色

geom类

- 里面可以传入一些参数
- 注意：+号不能位于行首

aes

- 使用aes函数向mapping中传递一些重要参数，每个函数不同，具体看help
- 缺省时一般会继承前面qplot中的数据
- 传入 **group=1** 是将所有的点都视为一个组的意思

geom_point

- **size, shape, color**
- aes: x, y
- position="jitter"：跟boxplot搭配时，把横坐标一样的点的横坐标错开画，方便观看

geom_line

- **color, linetype(lty)**
- aes: x, y

geom_hline

- aes: yintercept

geom_segment

- aes: x, y, xend, yend

labs

- title：标题
- x：横轴名
- y：纵轴名

geom_smooth

- method：传入一个字符串，如回归"lm"

geom_histogram

- **bins**：柱形数

geom_boxplot

- notch
- width

geom_rug

geom_density

- alpha

geom_bar

- position:
  - "stack"：堆叠式
  - "dodge"：平放式
  - "fill"：填满式
- 默认用来计数用的

「ggplot2——ggplot」

ggplot

- `data`：传入一个数据框
- `aes`：参数可有x、y或者其他后面各geom类函数都能用的aes参数

用fill等颜色或形状特征来分类

- 例子：`ggplot(data=Salaries, aes(x=salary, fill=rank))`
- `ggplot(Salaries, aes(x=yrs.since.phd, y=salary, color=rank,shape=sex)) + geom_point()`

分面

facet_grid

- 输入一个formula：rowvar~colvar
- 可以用.来说明不按此位置来分面
  - 例：`facet_grid(.~sex)`，只按列分面，按sex来分面，出来一个一行两列的图

facet_wrap

- 输入：`~var`
- 第二个参数决定排列方式
  - `ncol = n`：分面后排成n列
  - `nrow = n`：分面后排成n行

用theme函数来自定义主题

例子：

```
mytheme <- theme(
    plot.title = element_text(face = "bold.italic", size = "14", color =
"brown"),
    axis.title = element_text(face = "bold.italic", size = 10, color =
"brown"),
    axis.text = element_text(face = "bold", size = 9, color = "darkblue"),
    panel.background = element_rect(fill = "white", color = "darkblue"),
    panel.grid.major.y = element_line(color = "grey", linetype = 1),
    panel.grid.minor.y = element_line(color = "grey", linetype = 2),
    panel.grid.minor.x = element_blank(), legend.position = "top"
)
# 用element族的函数来自定义主题
ggplot(Salaries, aes(x = rank, y = salary, fill = sex)) +
    geom_boxplot() +
    labs(title = "Salary by Rank and Sex", x = "Rank", y = "Salary") +
    mytheme
```

保存图片

独立地写 `ggsave`，写file=文件名就行

RClass_4

「条件语句」

if语句

`if(<loopcond>){...}`

if-else语句

`if(<loopcond>){...}else{...}`

ifelse逐元素判断

- `test`：输入一个向量
- `yes`：若为T应该输出的结果
- `no`：若为F应该输出的结果
- 可以嵌套，如：`ifelse(x<5,ifelse(x<3, 'A','B'), 'C')`
- 反例：`if(c(F,T,F))`，只能使用一个元素，故只使用第一个元素

switch

- `EXPR`：输入的内容，可以是一个变量
- 后面的都是键值对，键匹配EXPR的内容，值是该键触发时switch语句的返回值
- 最后一个键值对一般为：`NA`
- 后面也可以不是键值对，直接按位置放值，则EXPR处应该输入一个自然数，switch语句的输出值则为EXPR值对应的位置处的值。

「循环语句」

for循环

- `for(loopind in loopvec){...}`

while循环

- `while(loopcond){...}`

break，next

- break：跳出本层循环
- next：直接跳转到本层循环的入口处，相当于continue

repeat

条件永真的循环，相当于while(T){}

「apply族」

apply

对矩阵或数组等逐行或逐列使用某函数，返回一个向量

- `X`：输入的数据
- `MARGIN`：运算方向，输入整数1或2.
  - 1表示以1轴（横轴）方向为运算方向，2同理
  - MARGIN顾名思义，就是运算后这个维度没了
- `FUN`：输入函数名
- `...`：可把FUN的参数放在后面

例子：

```
bar <- array(1:18,dim=c(3,3,2))
print(apply(bar,3,FUN=diag))

     [,1] [,2]
[1,]    1   10
[2,]    5   14
[3,]    9   18
```

- 第一列是第一个矩阵的对角元，第二列是第二个矩阵的对角元

tapply

依据传入的分组变量对数据进行分组，再对各组使用函数

- `X`：输入的数据
- `INDEX`：分组变量，向量或者因子均可
- `FUN`：函数名

lapply

对列表或向量的每个元素使用函数，返回一个等长的列表或向量

- `X`：输入的数据
- `FUN`：函数名

sapply

简化版本的lapply，统一返回一个向量了，若输入的是列表则返回一个带名字的向量

「函数」

- 格式：

```
functionname <- function(arg1,arg2,arg3,...){
do any code in here when called
return(returnobject)
}
```

- 可用键值对的方式传参，也可直接按位置传参
- 可以有return，也可以没有

lazy evaluation

当函数声明中没有给默认值的参数未传入实参，若这个参数恰好没在函数体中用到，则不会报错。（逻辑：不调用则不初始化，不初始化就不会发现没传参）

设置默认值

直接在定义里面要加默认值的arg后面写 **= 初始值**

检查缺失

用 `missing(argname)`，返回T则argname没传参，F则传了参。

传入任意数量的参数

声明中，放在最后的 `...` 表示传入任意多个参数。函数体中就用 `...` 来代表和调用那任意多个参数。

```
myfibplot <- function(thresh,plotit=TRUE,...){
略
plot(1:length(fibseq),fibseq,...)
略
}
```

RClass_5

「错误」

- 写warning用warning(mystr)，写报错用stop(mystr)
- attempt<-try(...,silent=...)，里面写试错语句，就算错了也不会终止程序，第二个参数若为T则还不会弹出报错信息。（warning还是会弹出）
  - 若错了，则class(attempt)=="try-error"，里面包含了报错信息

「基础统计」

- `mean(...)`：均值
- `median(...)`：中位数
- `table(mydf)`：计数，相当于 `value_counts()`
- `min(...)`
- `max(...)`
- 以上各函数均有na.rm参数，为T的时候移除缺失值，只处理数值
- `round(...,digits=...)`：保留几位小数
- `quantile()`：返回样本分位点
  - x
  - prob：输入一个向量，元素都是0~1之间的小数
- `summary()`：描述性统计
- `var()`
- `sd()`
- `IQR()`
- `sqrt()`

- `cov()`
- `cor()`
  - 可以选择method

「分布」

前缀

- `d`：density，密度函数，输入x，输出P(X=x)
- `p`：cdf，分布函数，输入q，输出P(X<=q)
- `q`：分位数，输入p，输出分布的p分位数
- `r`：随机变量，输入n，输出n个随机数

后缀

- `binom`：二项分布，自然参数为size, prob
- `pois`：泊松分布，自然参数为lambda
- `unif`：均匀分布，自然参数为min, max
- `norm`：正态分布，自然参数为mean, sd（标准差，即sigma）
- `t`：t分布，自然参数为df

验证正态性

用 `qqnorm(data)` 与 `qqline(data)`

RClass_6

「置信区间」

- 用公式手撸

「假设检验」

p值

- 检验统计量的值比目前给定值更加极端的概率称为该给定值下的p值
  - 更加极端是指更往备择假设的方向走
  - p值与置信系数比较，可得出目前给定值与拒绝域端点值的大小关系
- 手撸检验统计量的值并和临界值比较
- 注意p值的算法和备择假设有关：拒绝域方向与备择假设方向一致，拒绝域方向影响第一类错误概率的算法，第一类错误概率算法决定p值的算法。

用函数做

- `t.test`
  - x：数据
  - y：可能给出数据2，这时做两均值检验
  - mu：真值

- alternative：备择假设的方向
  - less, two.sided
- conf.level
- var.equal: `TRUE` then the pooled variance is used to estimate the variance, otherwise the variance is estimated separately for both groups andthe Welch (or Satterthwaite) approximation to the degrees of freedom is used
- paired: `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are silently removed (in pairs if `paired` is `TRUE`)

模拟

- 模拟拒真

```
typeI.tester <- function(mu0,sigma,n,alpha,ITERATIONS=10000){
  pvals <- rep(NA,ITERATIONS)
  for(i in 1:ITERATIONS){
    temporary.sample <- rnorm(n=n,mean=mu0,sd=sigma) #这里就是真，即目标总体的均值就是mu0
    temporary.mean <- mean(temporary.sample)
    temporary.sd <- sd(temporary.sample)
    pvals[i] <- 1-pt((temporary.mean-mu0)/(temporary.sd/sqrt(n)),df=n-1)
 #H1:mu>mu0
  }
  return(mean(pvals<alpha)) #用Iteration次试验的犯错频率来估计每次犯一类错误的概率
}  #第一类错误（上面是对逻辑向量取均值）
```

- 模拟纳伪

```
typeII.tester <- function(mu0,muA,sigma,n,alpha,ITERATIONS=10000){
  pvals <- rep(NA,ITERATIONS)
  for(i in 1:ITERATIONS){
    temporary.sample <- rnorm(n=n,mean=muA,sd=sigma)  #这里就是伪，即目标总体的均值不是mu0
    temporary.mean <- mean(temporary.sample)
    temporary.sd <- sd(temporary.sample)
    pvals[i] <- 1-pt((temporary.mean-mu0)/(temporary.sd/sqrt(n)),df=n-1)
  }
  return(mean(pvals>=alpha)) #落在了拒绝域外纳伪了
}
```

  - ```
    1-typeII.tester(mu0=0,muA=0,sigma=1.2,n=40,alpha=0.01) #typeI
    #mu0和muA相等时，这里就是Type1，因为mA就是真，生成样本也即是真样本
    ```

- 模拟功效：即1-type2，可直接用 `...` 传进上面定义的函数中，类似一个皮套

```
power.tester <- function(nvec,...){
  nlen <- length(nvec)
  result <- rep(NA,nlen)
  pbar <- txtProgressBar(min=0,max=nlen,style=3)
  for(i in 1:nlen){
    result[i] <- 1-typeII.tester(n=nvec[i],...) #1-第二类错误
    setTxtProgressBar(pbar,i)  #画一个进度条
  }
  close(pbar)
  return(result)   #返回一个vec
}
```

-
```

「回归」

线性回归

- `lm(formula,data)`
  - formula：因变量~自变量
- `predict(fitted,newdata,interval,level)`
  - 还能顺便获得置信区间

含分类变量的回归

与上面写法一样，不需要额外进行编码化

多元线性回归

多个自变量之间用+号连接，若还想要交互效应，则用*号连接