# 1. Flowchart

#a、b、c三个数按照从大到小的顺序排列，利用判断语句来进行判断排列

Print\_values <- function(a,b,c) {

if(a>b)

if(b>c) Result<-c(a,b,c)

else {

if(a>c) Result<-c(a,c,b)

else Result<-c(c,a,b)

}

else{

if(b>c)

if(a>c) Result<-c(b,a,c)

else Result<-c(b,c,a)

else Result<-c(c,b,a)

}

print(Result)

}

Print\_values(10,8,22)

Print\_values(1,82,2)

Print\_values(10,3,9)

Print\_values(10,31,9)

Print\_values(10,31,91)

Print\_values(10,31,9)

> Print\_values(10,8,22)

[1] 22 10 8

> Print\_values(1,82,2)

[1] 82 2 1

> Print\_values(10,3,9)

[1] 10 9 3

> Print\_values(10,31,9)

[1] 31 10 9

> Print\_values(10,31,91)

[1] 91 31 10

> Print\_values(10,31,9)

[1] 31 10 9

**2. Matrix multiplication**

2.1 建立两个矩阵

M1 <- matrix(sample(0:50,50),nrow = 10,ncol = 5,byrow = TRUE)

M2 <- matrix(sample(0:50,50),nrow = 5,ncol = 10,byrow = TRUE)

2.2 矩阵的乘法

#function函数

Matrix\_multip<-function(Mat1,Mat2)

#根据运算法则，提取第一个矩阵的行列数和第二个矩阵的列数，且运算得到的新矩阵，行数同第一个矩阵的函数，列数同第二个矩阵的列数

Mat1\_rows <- nrow(Mat1)

Mat1\_cols <- ncol(Mat1)

Mat2\_cols <- ncol(Mat2)

Mat\_out = matrix(nrow = Mat1\_rows, ncol = Mat2\_cols)

#做运算，得到新矩阵的每一项的每一个值

for(i in 1:Mat1\_rows) {

for(j in 1:Mat2\_cols){

Mat\_value <- 0

for(k in 1:Mat1\_cols){

val\_row <- Mat1[i,k]

val\_col <- Mat2[k,j]

Mat\_value <- Mat\_value + val\_row\*val\_col

Mat\_out[i,j] <- Mat\_value

}

}

}

print(Mat\_out)

}

#对2.1中的M1和M2矩阵进行计算

Matrix\_multip(M1,M2)

#对运算结果进行验证

print(M1%\*%M2)

#结果

print(M1)

[,1] [,2] [,3] [,4] [,5]

[1,] 35 40 49 27 2

[2,] 12 37 36 30 13

[3,] 0 50 4 21 39

[4,] 41 45 32 10 7

[5,] 46 1 15 14 43

[6,] 24 22 3 5 47

[7,] 42 48 28 17 29

[8,] 33 23 11 6 19

[9,] 38 44 18 25 20

[10,] 31 16 9 26 34

> print(M2)

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 31 10 2 13 11 7 47 30 29 34

[2,] 48 21 9 22 19 3 25 24 37 33

[3,] 12 36 42 17 0 50 41 32 39 27

[4,] 38 23 14 16 28 43 44 49 1 26

[5,] 18 8 5 6 46 45 15 4 40 20

> Matrix\_multip(M1,M2)

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 4655 3591 2876 2612 1993 4066 5872 4909 4513 4575

[2,] 3954 2987 2354 2140 2273 3870 4480 3922 3671 3641

[3,] 3948 1989 1107 1738 3332 3008 2923 2513 3587 3084

[4,] 4321 2793 2006 2269 1908 2767 4909 3852 4392 4143

[5,] 2960 1687 1142 1357 2895 3612 4063 2742 3690 3226

[6,] 2872 1301 677 1209 2984 2714 2726 1777 3512 2693

[7,] 5110 3059 2075 2524 3184 3874 5505 4257 5263 4790

[8,] 2829 1499 914 1332 1842 1963 3126 2264 3003 2714

[9,] 4816 2687 1678 2288 2874 3273 5024 4077 4257 4280

[10,] 3437 1840 1118 1528 2937 3363 3880 3012 3228 3181

> print(M1%\*%M2)

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 4655 3591 2876 2612 1993 4066 5872 4909 4513 4575

[2,] 3954 2987 2354 2140 2273 3870 4480 3922 3671 3641

[3,] 3948 1989 1107 1738 3332 3008 2923 2513 3587 3084

[4,] 4321 2793 2006 2269 1908 2767 4909 3852 4392 4143

[5,] 2960 1687 1142 1357 2895 3612 4063 2742 3690 3226

[6,] 2872 1301 677 1209 2984 2714 2726 1777 3512 2693

[7,] 5110 3059 2075 2524 3184 3874 5505 4257 5263 4790

[8,] 2829 1499 914 1332 1842 1963 3126 2264 3003 2714

[9,] 4816 2687 1678 2288 2874 3273 5024 4077 4257 4280

[10,] 3437 1840 1118 1528 2937 3363 3880 3012 3228 3181

# 3. Pascal triangle

#将杨辉三角数据看成是一个n\*n的矩阵，初始值为0，将每一行的第一个数据都是1，因此从每一行的第二个数据开始循环计算，最后只输出n行的数据

Pascal\_triangle <- function(n){

Pas\_result <- matrix(0,nrow = n,ncol = n)

Pas\_result[,1] <- 1

for (i in 2:n){

for(j in 2:i){

Pas\_result[i,j] <- Pas\_result[i-1,j-1] + Pas\_result[i-1,j]

}

}

print(Pas\_result[n,])

}

#输出第100和200行的数据

Pascal\_triangle(100)

[1] 1.000000e+00 9.900000e+01 4.851000e+03 1.568490e+05 3.764376e+06

[6] 7.152314e+07 1.120529e+09 1.488703e+10 1.712009e+11 1.731031e+12

[11] 1.557928e+13 1.260505e+14 9.243705e+14 6.186172e+15 3.800077e+16

[16] 2.153377e+17 1.130523e+18 5.519612e+18 2.514490e+19 1.071967e+20

[21] 4.287867e+20 1.613055e+21 5.719012e+21 1.914626e+22 6.062982e+22

[26] 1.818895e+23 5.176854e+23 1.399668e+24 3.599146e+24 8.811702e+24

[31] 2.056064e+25 4.576400e+25 9.724850e+25 1.974439e+26 3.832735e+26

[36] 7.117936e+26 1.265411e+27 2.154619e+27 3.515430e+27 5.498494e+27

[41] 8.247740e+27 1.186870e+28 1.639011e+28 2.172642e+28 2.765181e+28

[46] 3.379666e+28 3.967434e+28 4.473915e+28 4.846741e+28 5.044567e+28

[51] 5.044567e+28 4.846741e+28 4.473915e+28 3.967434e+28 3.379666e+28

[56] 2.765181e+28 2.172642e+28 1.639011e+28 1.186870e+28 8.247740e+27

[61] 5.498494e+27 3.515430e+27 2.154619e+27 1.265411e+27 7.117936e+26

[66] 3.832735e+26 1.974439e+26 9.724850e+25 4.576400e+25 2.056064e+25

[71] 8.811702e+24 3.599146e+24 1.399668e+24 5.176854e+23 1.818895e+23

[76] 6.062982e+22 1.914626e+22 5.719012e+21 1.613055e+21 4.287867e+20

[81] 1.071967e+20 2.514490e+19 5.519612e+18 1.130523e+18 2.153377e+17

[86] 3.800077e+16 6.186172e+15 9.243705e+14 1.260505e+14 1.557928e+13

[91] 1.731031e+12 1.712009e+11 1.488703e+10 1.120529e+09 7.152314e+07

[96] 3.764376e+06 1.568490e+05 4.851000e+03 9.900000e+01 1.000000e+00

> Pascal\_triangle(200)

[1] 1.000000e+00 1.990000e+02 1.970100e+04 1.293699e+06 6.339125e+07

[6] 2.472259e+09 7.993637e+10 2.203960e+12 5.289504e+13 1.122550e+15

[11] 2.132845e+16 3.664616e+17 5.741232e+18 8.258541e+19 1.097206e+21

[16] 1.353221e+22 1.556204e+23 1.675208e+24 1.693821e+25 1.613588e+26

[21] 1.452229e+27 1.237852e+28 1.001535e+29 7.707466e+29 5.652141e+30

[26] 3.956499e+31 2.647811e+32 1.696560e+33 1.042173e+34 6.145226e+34

[31] 3.482294e+35 1.898412e+36 9.966664e+36 5.043736e+37 2.462530e+38

[36] 1.160907e+39 5.288576e+39 2.329832e+40 9.932442e+40 4.100316e+41

[41] 1.640126e+42 6.360490e+42 2.392756e+43 8.736341e+43 3.097430e+44

[46] 1.066893e+45 3.571771e+45 1.162725e+46 3.681964e+46 1.134646e+47

[51] 3.403938e+47 9.944838e+47 2.830454e+48 7.850504e+48 2.122544e+49

[56] 5.595797e+49 1.438919e+50 3.609920e+50 8.838081e+50 2.112151e+51

[61] 4.928353e+51 1.123018e+52 2.499621e+52 5.435684e+52 1.155083e+53

[66] 2.399018e+53 4.870734e+53 9.668771e+53 1.876879e+54 3.563350e+54

[71] 6.617650e+54 1.202362e+55 2.137532e+55 3.718720e+55 6.331875e+55

[76] 1.055312e+56 1.721826e+56 2.750449e+56 4.301984e+56 6.589115e+56

[81] 9.883672e+56 1.452046e+57 2.089529e+57 2.945481e+57 4.067569e+57

[86] 5.503181e+57 7.294914e+57 9.475004e+57 1.205910e+58 1.504000e+58

[91] 1.838222e+58 2.201826e+58 2.584752e+58 2.973855e+58 3.353496e+58

[96] 3.706495e+58 4.015370e+58 4.263743e+58 4.437774e+58 4.527426e+58

[101] 4.527426e+58 4.437774e+58 4.263743e+58 4.015370e+58 3.706495e+58

[106] 3.353496e+58 2.973855e+58 2.584752e+58 2.201826e+58 1.838222e+58

[111] 1.504000e+58 1.205910e+58 9.475004e+57 7.294914e+57 5.503181e+57

[116] 4.067569e+57 2.945481e+57 2.089529e+57 1.452046e+57 9.883672e+56

[121] 6.589115e+56 4.301984e+56 2.750449e+56 1.721826e+56 1.055312e+56

[126] 6.331875e+55 3.718720e+55 2.137532e+55 1.202362e+55 6.617650e+54

[131] 3.563350e+54 1.876879e+54 9.668771e+53 4.870734e+53 2.399018e+53

[136] 1.155083e+53 5.435684e+52 2.499621e+52 1.123018e+52 4.928353e+51

[141] 2.112151e+51 8.838081e+50 3.609920e+50 1.438919e+50 5.595797e+49

[146] 2.122544e+49 7.850504e+48 2.830454e+48 9.944838e+47 3.403938e+47

[151] 1.134646e+47 3.681964e+46 1.162725e+46 3.571771e+45 1.066893e+45

[156] 3.097430e+44 8.736341e+43 2.392756e+43 6.360490e+42 1.640126e+42

[161] 4.100316e+41 9.932442e+40 2.329832e+40 5.288576e+39 1.160907e+39

[166] 2.462530e+38 5.043736e+37 9.966664e+36 1.898412e+36 3.482294e+35

[171] 6.145226e+34 1.042173e+34 1.696560e+33 2.647811e+32 3.956499e+31

[176] 5.652141e+30 7.707466e+29 1.001535e+29 1.237852e+28 1.452229e+27

[181] 1.613588e+26 1.693821e+25 1.675208e+24 1.556204e+23 1.353221e+22

[186] 1.097206e+21 8.258541e+19 5.741232e+18 3.664616e+17 2.132845e+16

[191] 1.122550e+15 5.289504e+13 2.203960e+12 7.993637e+10 2.472259e+09

[196] 6.339125e+07 1.293699e+06 1.970100e+04 1.990000e+02 1.000000e+00

# 4. Add or double

#将任意数分为奇数、偶数来确定最小的move

#Create a random value between 1 and 100.

random\_val <- sample(1:100,1)

#Build the function

Least\_moves <- function(num){

if (num <= 3) move\_res <- num-1

if (num > 3){

if (num%%2!=0) move\_res <- 1+ Least\_moves(num-1)

else

move\_res <- 1 + min(c(Least\_moves(num-1), Least\_moves(round(num/2))))

}

return(move\_res)

}

#输入2和5来进行检查

Least\_moves(2)

Least\_moves(5)

> Least\_moves(2)

[1] 1

> Least\_moves(5)

[1] 3

# 5. Dynamic programming

1-9, 9个数字两两之间一共有8个空位置，可以任意放入+、-或者合并的符号，对数据进行处理和运算，例如

1+2合并3+4合并5-6+7+8+9 表示1+23+45-6+7+8+9

8个空位置（依次表示为a, b, c, d, e, f, g, h），3种符号，共有3^8个排列方式，每一种排列方式对应一个结果，因此建立一个矩阵，3^8行，9列，如下，每一行都记录了一种计算排列方式和一个计算结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | 计算结果 |
| · |  |  |  |  |  |  |  |  |
| · |  |  |  |  |  |  |  |  |
| · |  |  |  |  |  |  |  |  |

在计算时，考虑8个位置符号的可能性，对每个位置可能的符号进行排列，对应于相应的计算。建立一个位置符号的指针，记为symbol（1, a, b, c, d, e, f, g, h, 4, 4, 4, 4, 4, 4, 4, 4）。位置数记为index，index从1-8进行循环。location=index+1。将+记为1，减号记为2，合并记为3，symbol的每一个值记为isyb。计算时，+表示为（-1）^（3-isyb），减号表示为（-1）^（3-isyb）来进行运算。设一个ival的初始值为1，之后每一次对每一个位置进行判断的值，都在ival的基础上进行计算。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| num |  | 1 |  | 2 |  | 3 |  | 4 |  | 5 |  | 6 |  | 7 |  | 8 |  | 9 |  |  |
| Symbol | 1 |  | a |  | b |  | c |  | d |  | e |  | f |  | g |  | h |  | 4 | 4 |
| index | 1 |  | 2 |  | 3 |  | 4 |  | 5 |  | 6 |  | 7 |  | 8 |  |  |  |  |  |
| location |  |  | 1 |  | 2 |  | 3 |  | 4 |  | 5 |  | 6 |  | 7 |  | 8 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

当index=1时，对location=1，2的位置进行判断，如果index+1不等于3，且index+2的位置也不等于3，那么，ival=ival（初始值为1）+（-1）^（3-isyb）（index+1），

# 6. Visibility in Shenzhen during the past 10 years

#读取数据，对VIS数据进行整理和计算，并绘制图

Airport\_Data <- read.csv(file = "2281305.csv", header = T)

names(Airport\_Data)

BaoAn\_V<-Airport\_Data$VIS

Obs\_Time<-Airport\_Data$DATE

BaoAn\_V\_value <- substr(BaoAn\_V,1,6)

BaoAn\_V\_flag <- substr(BaoAn\_V,12,12)

BaoAn\_V\_value2 <- as.numeric(BaoAn\_V\_value)

BaoAn\_V\_flag2 <- as.numeric(BaoAn\_V\_flag)

BaoAn\_V\_value2[which(BaoAn\_V\_value2 == 999999)] <- NA

BaoAn\_V\_value2[!which(BaoAn\_V\_flag2 == 1)] <- NA

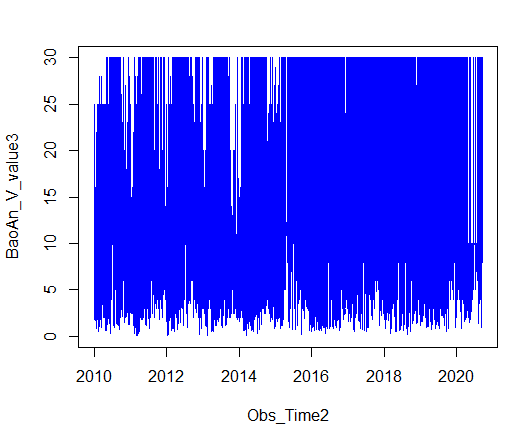
BaoAn\_V\_value3 <- BaoAn\_V\_value2 \* 0.001

Obs\_Time\_Date <- substr(Obs\_Time,1,10)

Obs\_Time\_Hour <- substr(Obs\_Time,12,13)

Obs\_Time2 <- as.Date(Obs\_Time)

plot(Obs\_Time2,BaoAn\_V\_value3, lwd=0.5,type="l",col="blue")



#将数据按照[0,5km),[5km,10km),[10km,15km),[15km,20km),[20km,25km),[25km,30km), and >=30km进行分类计数

BaoAn\_V\_flag3 <- BaoAn\_V\_flag2[which(BaoAn\_V\_flag2 == 1)]

result\_5 <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 < 5)])

result\_10 <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 < 10)]) - result\_5

result\_15 <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 < 15)]) - result\_10

result\_20 <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 < 20)]) - result\_15

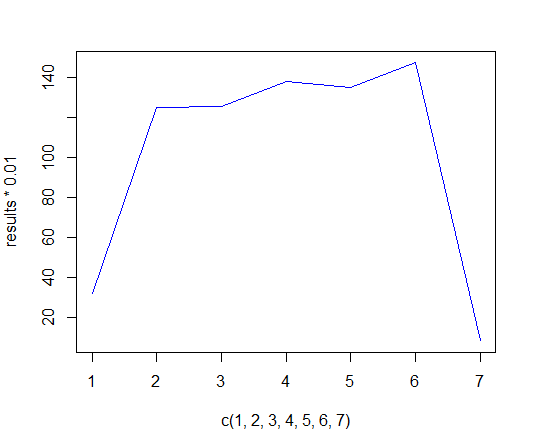
result\_25 <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 < 25)]) - result\_20

result\_30 <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 < 30)]) - result\_25

result\_30plus <- table(BaoAn\_V\_flag3[which(BaoAn\_V\_value3 >= 30)])

results <- c(result\_5,result\_10,result\_15,result\_20,result\_25,result\_30,result\_30plus)

plot(c(1,2,3,4,5,6,7),results\*0.01, lwd=0.5,type="l",col="blue")



# 7. Explore a data set

#下载降雨量数据——VQC00670480.csv，找到DlySum列，通过flag进行筛选，同时筛选出2016年以后的时间序列的数据并进行画图

Precipitation\_Data <- read.csv(file = "VQC00670480.csv", header = T)

names(Precipitation\_Data)

Pre\_V<-Precipitation\_Data$DlySum

head(Pre\_V)

tail(Pre\_V)

typeof(Pre\_V)

Obs\_Time<-Precipitation\_Data$DATE

head(Obs\_Time)

tail(Obs\_Time)

typeof(Obs\_Time)

#Clean the data, make the missing data NA; retain the data after 2016 to plot

Pre\_Vnum <- as.numeric(Pre\_V)

Pre\_V\_flag <- Precipitation\_Data$DlySumQF

Pre\_Vnum[which(Pre\_V\_flag == "M")] <- NA

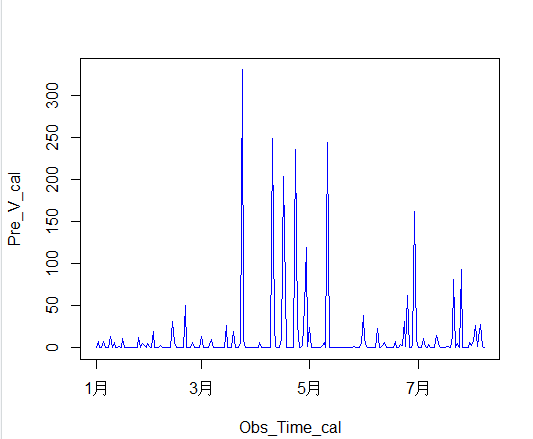
Pre\_V\_cal <- Pre\_Vnum[which(Obs\_Time > "2016-12-31")]

Obs\_Time2 <- Obs\_Time[which(Obs\_Time > "2016-12-31")]

Obs\_Time\_cal <- as.Date(Obs\_Time2)

#Plot the time series of the daily sum precipitation after 2016.

plot(Obs\_Time\_cal,Pre\_V\_cal, lwd=0.5,type="l",col="blue")



#做5个统计数据计算，求平均值，中位数，和，最小，最大值。

sum\_cal <- sum(Pre\_V\_cal,na.rm=T)

mean\_cal <- mean(Pre\_V\_cal,na.rm=T)

median\_cal <- median(Pre\_V\_cal,na.rm=T)

min\_cal <- min(Pre\_V\_cal,na.rm=T)

max\_cal <- max(Pre\_V\_cal,na.rm=T)

min\_date <- Obs\_Time\_cal[which.min(Pre\_V\_cal)]

max\_date <- Obs\_Time\_cal[which.max(Pre\_V\_cal)]

print(c(sum\_cal,mean\_cal,median\_cal,min\_cal,max\_cal))

print(c(min\_date,max\_date))

> print(c(sum\_cal,mean\_cal,median\_cal,min\_cal,max\_cal))

[1] 2557.00000 11.78341 0.00000 0.00000 331.00000

> print(c(min\_date,max\_date))

[1] "2017-01-01" "2017-03-24"