WOTD

51CTO

全球软件开发技术峰会

World Of Tech 2017

2017年12月1日-2日 • 深圳中洲万豪酒店

DEVELOPMENT

面向数据的思维模式和R语言编程

张丹
《R的极客理想》作者

前言

作为数据分析师，每天都有大量的数据需要处理，我们会根据业务的要求做各种复杂的报表。为了计算一个业务指标，你的SQL怎么写都不会少于10行时，这个时候**R语言绝对是不二的选择了**。

用R语言可以高效地、优雅地解决数据分析中的问题，让R来帮你打开面向数据的思维模式。

目录

面向数据的思维模式

数据很重要，要想把业务做好，需要有更高质量和更好来源的数据。

工程师，5年工作经验，熟练掌握一种语言。
他的能力：

- 理解需求
- 快速写代码完成开发任务
- 没有bug
- 性能很好

面向数据的思维模式

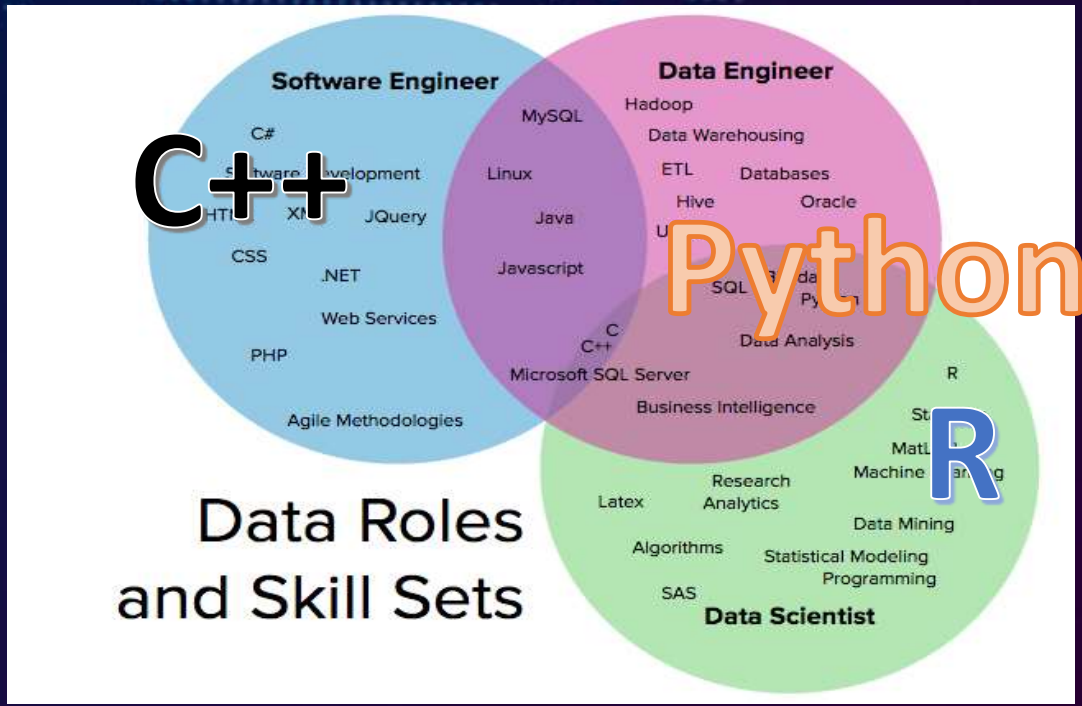
目标：不是如何做出来，而是如何让这个事情变现。那么要怎么去写代码？这就是一种思维模式的变化。

数据分析师，5年工作经验，熟练掌握一种语言。

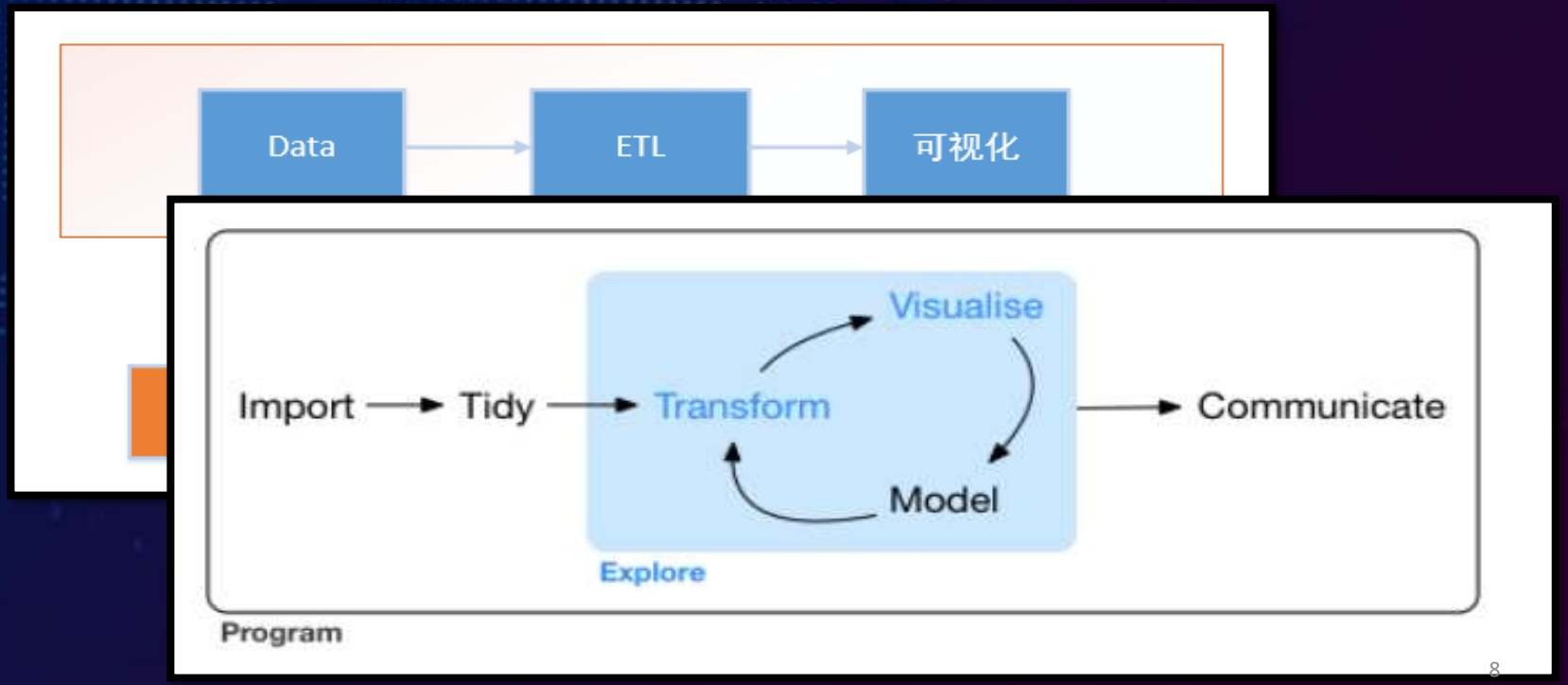
他的能力：

- 提出需求
- 发现数据的规律
- 找到业务价值点
- 变成数据产品

数据与工程


Software Engineer
C#
JQuery
CSS
.NET
Web Services
PHP
Agile Methodologies
Data Engineer
MySQL
Hadoop
Data Warehousing
Linux
ETL
Databases
Hive
Oracle
Java
Javascript
SQL
C
C++
Microsoft SQL Server
Business Intelligence
R
Research
Analytics
Latex
Data Mining
Algorithms
Statistical Modeling
Programming
SAS
Data Scientist
C++
Python
R
Data Roles
and Skill Sets

BI & 数据科学


Data
ETL
可视化
Visualise
Import
Tidy
Transform
Communicate
Model
Explore
Program

如何突围

跨学科的结合，而不是沉浸在一个单独的学科领域里！单一领域已经有很多大神，你不可能颠覆他们的位置。

但一旦你带着原有的东西进入到另一个学科，或者多结合几个学科，就没有那么多大神了，而且现在社会就需要这种人才。

R语言的价值

R语言的价值，主要体现在数据分析的效率上！

用R语言，验证想法的可行性，再用其他语言来改写。

R语言擅长的事有很多，包括做数学计算、数据处理、统计分析、可视化等，但它并不擅长做Web开发的工程项目。

目录

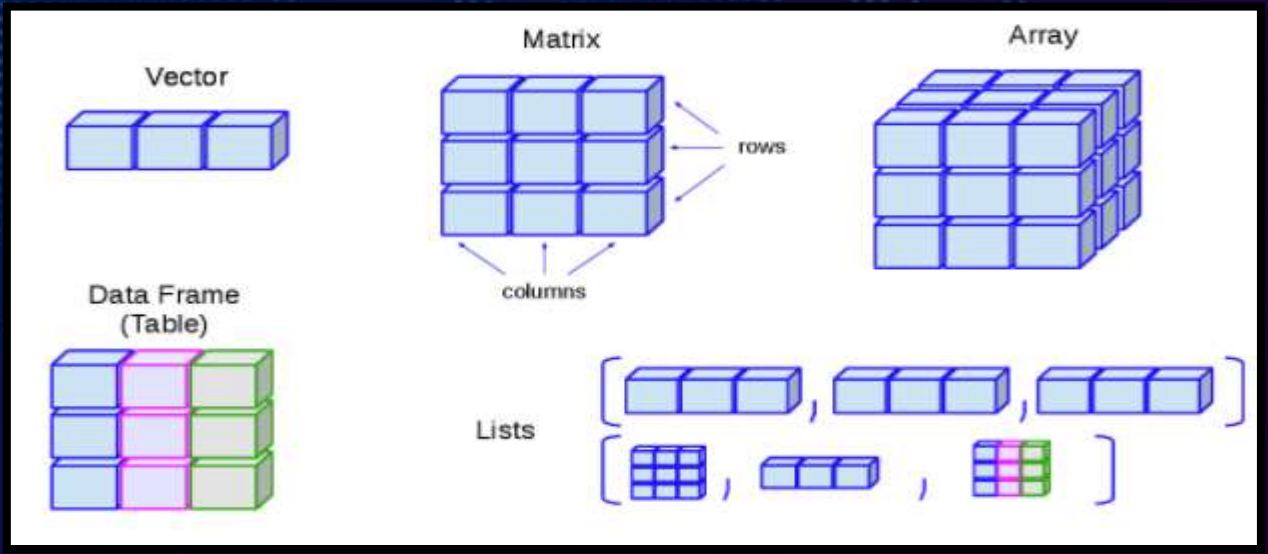
用R语言进行数据处理

合并、分组、排序、筛选、转置、差分、填充、移动、清洗、回归、分布检验、高数计算 等等。

R语言数据类型

- 向量 Vector : c()
- 矩阵 Matrix: matrix()
- 数据框 DataFrame: data.frame()
- 时间序列 XTS: xts()

R语言数据类型



初始化数据集

```
> x<-1:20;x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
> m<-matrix(1:40,ncol=5);m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    9   17   25   33
[2,]    2   10   18   26   34
[3,]    3   11   19   27   35
[4,]    4   12   20   28   36
[5,]    5   13   21   29   37
[6,]    6   14   22   30   38
[7,]    7   15   23   31   39
[8,]    8   16   24   32   40
```

```
> library(xts)
> xts(1:10,order.by=as.Date('2017-01-01')+1:10)
           [,1]
2017-01-02    1
2017-01-03    2
2017-01-04    3
2017-01-05    4
2017-01-06    5
2017-01-07    6
2017-01-08    7
2017-01-09    8
2017-01-10    9
2017-01-11   10
```

```
> df<-data.frame(a=1:5,b=c('A','A','B','B','A'),c=rnorm(5));df
  a b          c
1 1 A  1.1519118
2 2 A  0.9921604
3 3 B -0.4295131
4 4 B  1.2383041
5 5 A -0.2793463
```

概况(Summary)

```
# 查看矩阵数据集的概况
> m<-matrix(1:40,ncol=5)
> summary(m)
```

V1	V2	V3	V4	V5
Min. :1.00	Min. : 9.00	Min. :17.00	Min. :25.00	Min. :33.00
1st Qu.:2.75	1st Qu.:10.75	1st Qu.:18.75	1st Qu.:26.75	1st Qu.:34.75
Median :4.50	Median :12.50	Median :20.50	Median :28.50	Median :36.50
Mean :4.50	Mean :12.50	Mean :20.50	Mean :28.50	Mean :36.50
3rd Qu.:6.25	3rd Qu.:14.25	3rd Qu.:22.25		
Max. :8.00	Max. :16.00	Max. :24.00		

```
# 查看数据框数据集的概况信息
> df<-data.frame(a=1:5,b=c('A','A','B','B','A'),c=rnorm(5))
> summary(df)
```

a	b	c
Min. :1	A:3	Min. :-1.5638
1st Qu.:2	B:2	1st Qu.:-1.0656
Median :3		Median :-0.2273
Mean :3		Mean :-0.1736
3rd Qu.:4		3rd Qu.: 0.8320
Max. :5		Max. : 1.1565

累计(Cumulate)

```
# 向量x
> x<-1:10;x
 [1]  1  2  3  4  5  6  7  8  9 10

# 累计求和
> cum_sum<-cumsum(x)

# 累计求积
> cum_prod<-cumprod(x)

# 拼接成data.frame
> data.frame(x, cum_sum, cum_prod)
   x cum_sum cum_prod
1  1       1        1
2  2       3        2
3  3       6        6
4  4      10       24
5  5      15      120
6  6      21      720
7  7      28     5040
8  8      36    40320
9  9      45   362880
10 10     55  3628800
```

差分(Difference)

```
> x<-1:10;x
 [1]  1  2  3  4  5  6  7  8  9 10

# 计算1阶差分
> diff(x)
[1] 1 1 1 1 1 1 1 1 1

# 计算2阶差分
> diff(x,2)
[1] 2 2 2 2 2 2 2 2

# 计算2阶差分，迭代2次
> diff(x,2,2)
[1] 0 0 0 0 0 0
```

```
# 对向量2次累积求和
> x <- cumsum(cumsum(1:10));x
 [1]   1   4  10  20  35  56  84 120 165 220

# 计算2阶差分
> diff(x, lag = 2)
[1]  9 16 25 36 49 64 81 100

# 计算1阶差分，迭代2次
> diff(x, differences = 2)
[1] 3 4 5 6 7 8 9 10

# 同上
> diff(diff(x))
[1] 3 4 5 6 7 8 9 10
```

分组(Group by)

```
# 创建数据框
> df<-data.frame(a=1:5,b=c('A','A','B','B','A'),c=rnorm(5));df
  a b           c
1 1 A  1.28505418
2 2 A -0.04687263
3 3 B  0.25383533
4 4 B  0.70145787
5 5 A -0.11470372

# 执行分组操作
> aggregate(. ~ b, data = df, mean)
  b        a         c
1 A 2.666667 0.3744926
2 B 3.500000 0.4776466
```

```
# 加载plyr库
> library(plyr)

# 执行分组操作
> ddply(df,.(b),summarise,
+       sum_a=sum(a),
+       mean_c=mean(c))
  b sum_a       mean_c
1 A     8 -0.05514761
2 B     7  0.82301276
```

分裂(Split)

```
> split(1:10, 1:2)
$`1`
[1] 1 3 5 7 9

$`2`
[1]  2  4  6  8 10
```

```
# 生成因子规则
> n <- 3; size <- 5
> fat <- factor(round(n * runif(n * size))); fat
 [1] 2 3 2 1 1 0 0 2 0 1 2 3 1 1 1
Levels: 0 1 2 3

# 生成数据向量
> x <- rnorm(n * size); x
 [1]  0.68973936  0.02800216 -0.74327321  0.18879230 -1.80495863  1.46555486  0.15325334
[10] -0.70994643  0.61072635 -0.93409763 -1.25363340  0.29144624 -0.44329187

# 对向量以因子的规则进行拆分
> split(x, fat)
$`0`
[1] 1.4655549 0.1532533 0.4755095

$`1`
[1]  0.1887923 -1.8049586 -0.7099464 -1.2536334  0.2914462 -0.4432919

$`2`
[1]  0.6897394 -0.7432732  2.1726117  0.6107264

$`3`
[1]  0.02800216 -0.93409763
```

排序(Order)

```
# 生成一个乱序的向量
> x<-sample(1:10);x
 [1]  6  2  5  1  9 10  8  3  7  4

# 对向量排序
> x[order(x)]
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> df<-data.frame(a=1:5,b=c('A','A','B','B','A'),c=rnorm(5));df
  a b          c
1 1 A  1.1780870
2 2 A -1.5235668
3 3 B  0.5939462
4 4 B  0.3329504
5 5 A  1.0630998

# 自定义排序函数
> order_df<-function(df,col,decreasing=FALSE){
+     df[order(df[,c(col)],decreasing=decreasing),]
+ }

# 以c列倒序排序
> order_df(df,'c',decreasing=TRUE)
  a b          c
1 1 A  1.1780870
5 5 A  1.0630998
3 3 B  0.5939462
4 4 B  0.3329504
2 2 A -1.5235668
```

转置(Transpose)

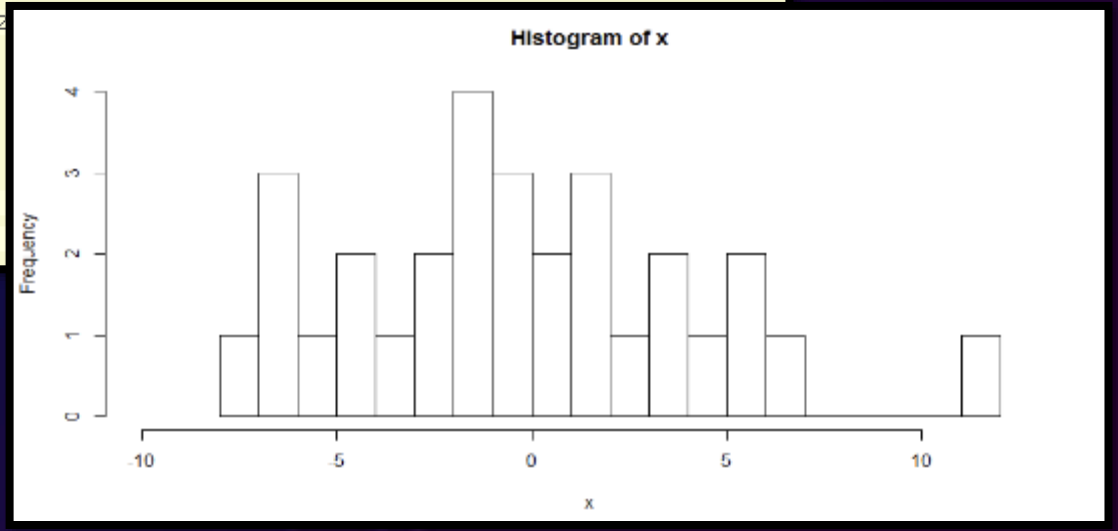
```
# 创建一个3行5列的矩阵
> m<-matrix(1:15,ncol=5);m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15

# 转置后，变成5行3列的矩阵
> t(m)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15
```

填充(Fill)

```
# 生成数据框
> df<-data.frame(a=c(1,NA,NA,2,NA),
+       b=c('B','A','B','B',NA),
+       c=c(rnorm(2),NA,NA,NA));df
   a    b          c
1  1    B  0.2670988
2 NA    A -0.5425200
3 NA    B         NA
4  2    B         NA
5 NA <NA>         NA

# 把数据框a列的NA，用9进行填充
> na.fill(df$a,9)
[1] 1 9 9 2 9

# 把数据框中的NA，用1进行填充
> na.fill(df,1)
     a      b      c
[1,] " 1"   "B"    " 0.2670988"
[2,] "TRUE" "A"    "-0.5425200"
[3,] "TRUE" "B"    "TRUE"
[4,] " 2"   "B"    "TRUE"
[5,] "TRUE" "TRUE" "TRUE"
```

计数(Table)

```
# 生成30个随机数的向量
> set.seed(0)
> x<-round(rnorm(30)*5);x
 [1]  6 -2  7  6  2 -8 -5 -1  0 12

# 统计每个值出现的次数
> table(x)
x
-8 -6 -5 -4 -3 -2 -1  0  1  2  3
 1  3  1  2  1  2  4  3  2  3  1
```

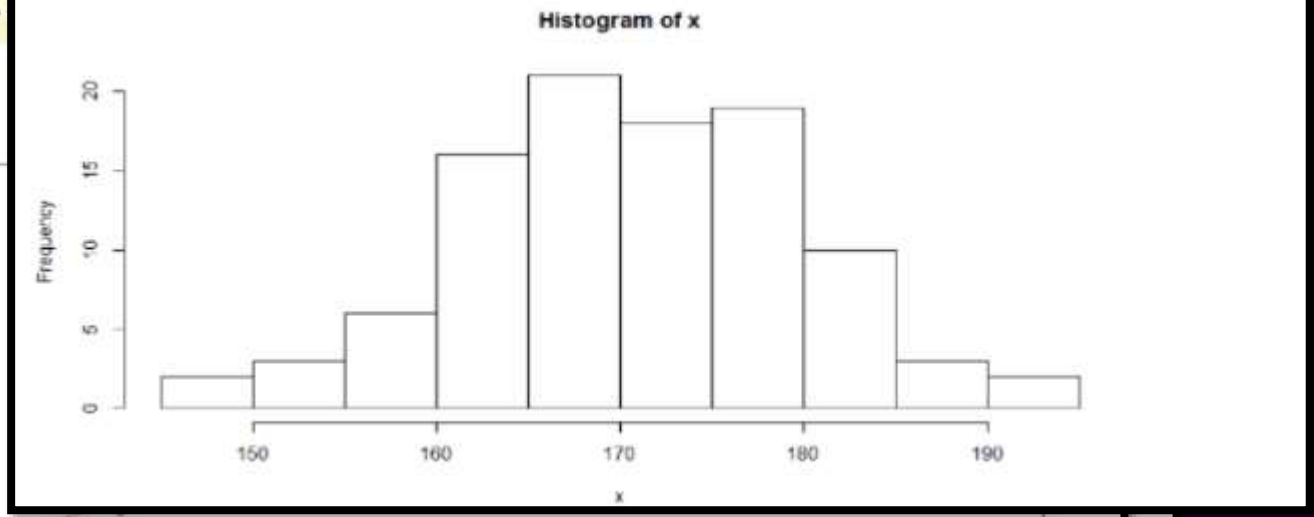


正态

```
# 生成身高数据
> set.seed(1)
> x<-round(rnor
> head(x,20)
 [1] 164 172 16

# 画出散点图
> plot(x)
```

```
> shapiro.test(x)
```

```
> qqnorm(x)
> qqline(x,
```



数值分段(Cut)

```
> x<-1:10;x
 [1]  1  2  3  4  5  6  7  8  9 10

# 把向量转换为3段因子，分别列出每个值对应因子
> cut(x, 3)
 [1] (0.991,4] (0.991,4] (0.991,4] (0.991,4] (4,7]     (4,7]     (4,7]     (7,10]    (7,10]    (7,10]
Levels: (0.991,4] (4,7] (7,10]

# 对因子保留2位精度，并支持排序
> cut(x, 3, dig.lab = 2, ordered = TRUE)
 [1] (0.99,4] (0.99,4] (0.99,4] (0.99,4] (4,7]    (4,7]    (4,7]    (7,10]   (7,10]   (7,10]
Levels: (0.99,4] < (4,7] < (7,10]
```

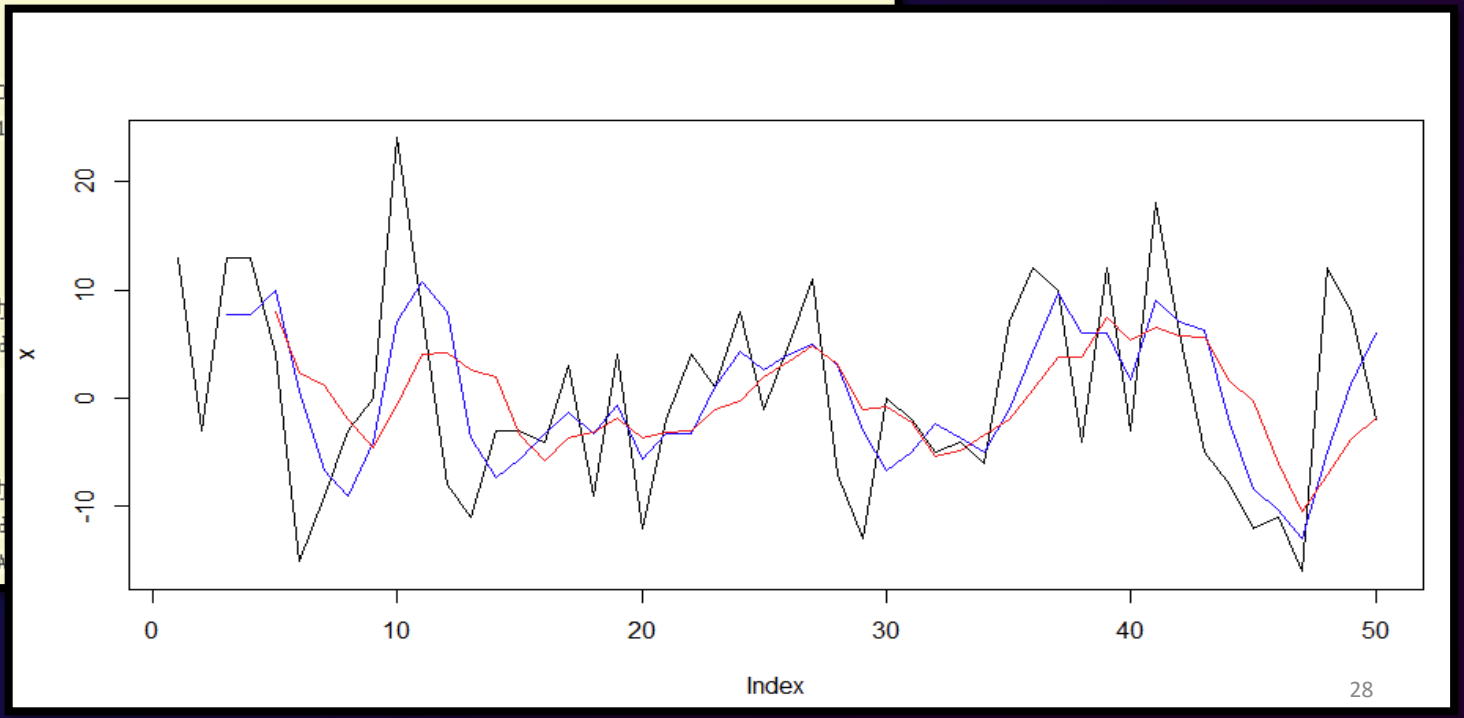
集合(Set)

```
# 定义2个向量x,y
> x<-c(3:8,NA);x
[1]  3  4  5  6  7  8 NA
> y<-c(NA,6:10,NA);y
[1] NA  6  7  8  9 10 NA

# 判断x与y重复的元素的位置
> is.element(x, y)
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE

# 判断y与x重复的元素的位置
> is.element(y, x)
[1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
```

```
# 取并集
> union(x, y)
[1]  3  4  5  6  7  8 NA  9 10

# 取交集
> intersect(x, y)
[1]  6  7  8 NA

# 取x有，y没有元素
> setdiff(x, y)
[1] 3 4 5

# 取y有，x没有元素
> setdiff(y, x)
[1]  9 10

# 判断2个向量是否相等
> setequal(x, y)
[1] FALSE
```

移动平均(MA)

```
# 生成50个随机数
> set.seed(0)
> x<-round(rnorm(50
 [1]  13  -3  13  1

# 加载TTR包
> library(TTR)

# 计算周期为3的移动
> m3<-SMA(x,3);head
 [1]        NA
[10]  7.0000000

# 计算周期为5的移动
> m5<-SMA(x,5);head
 [1]  NA  NA  NA
```



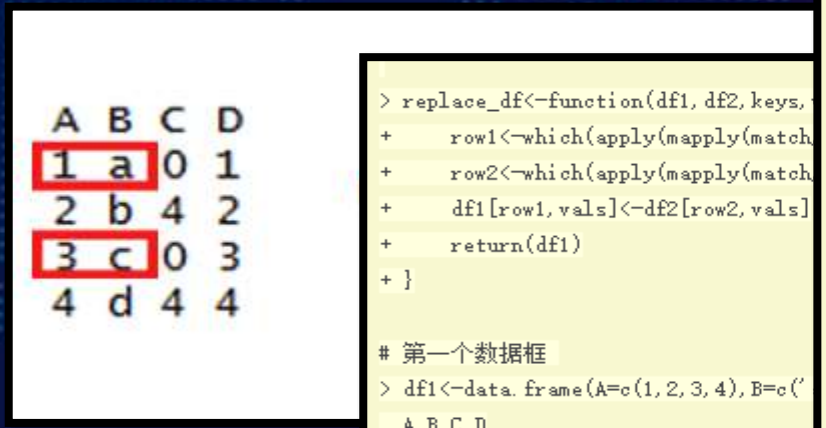
目录

过滤数据框中，列数据全部为空的列

a	b	c
1	NA	1
NA	NA	2
2	NA	3
4	NA	4

```
# 判断哪列的值都是NA
na_col_del_df<-function(df){
  df[,which(!apply(df,2,function(x) all(is.na(x))))]
}

# 生成一个数据集
> df<-data.frame(a=c(1,NA,2,4),b=rep(NA,4),c=1:4);df
   a  b c
1  1 NA 1
2 NA NA 2
3  2 NA 3
4  4 NA 4

# 保留非NA的列
> na_col_del_df(df)
   a c
1  1 1
2 NA 2
3  2 3
4  4 4
```

替换数据框中某个区域的数据



A	B	C	D
1	a	0	1
2	b	4	2
3	c	0	3
4	d	4	4

```
> replace_df<-function(df1,df2,keys,
+     row1<-which(apply(mapply(match
+     row2<-which(apply(mapply(match
+     df1[row1,vals]<-df2[row2,vals]
+     return(df1)
+ }

# 第一个数据框
> df1<-data.frame(A=c(1,2,3,4),B=c('
  A B C D
1 1 a 0 1
2 2 b 4 2
3 3 c 0 3
4 4 d 4 4
```

```
# 第二个数据框
> df2<-data.frame(A=c(1,3),B=c('a','c'),C=c(9,9),D=rep(8,2));df2
  A B C D
1 1 a 9 8
2 3 c 9 8

# 定义匹配条件列
> keys=c("A","B")

# 定义替换的列
> vals=c("C","D")

# 数据替换
> replace_df(df1,df2,keys,vals)
  A B C D
1 1 a 9 8
2 2 b 4 2
3 3 c 9 8
4 4 d 4 4
```

长表和宽

```
# 创建数据框
> df<-data.frame(
+     program=rep(c('R','Java','PHP','Python'),3),
+     fun=rep(c('fun1','fun2','fun3'),each = 4),
+     time=round(rnorm(12,10,3),2)
+ );df
   program  fun  time
1        R fun1 15.01
2     Java fun1  7.17
3      PHP fun1 10.84
4   Python fun1  8.96
5        R fun2 10.30
6     Java fun2  9.45
7      PHP fun2  8.87
8   Python fun2  8.18
9        R fun3  6.30
10    Java fun3  9.70
11     PHP fun3  8.89
12  Python fun3  5.19

# 加载reshape2库
> library(reshape2)

# 长表变宽表
> wide <- reshape(df,v.names="time",idvar="program",timevar="fun",direction = "wide");wide
  program time.fun1 time.fun2 time.fun3
1       R      8.31      8.72     10.10
2    Java      8.45      4.15     13.86
3     PHP     10.49     11.47      9.96
4  Python     10.45     13.25     14.64
```

```
   program  fun
1        R fun1
2     Java fun1
3      PHP fun1
4   Python fun1
5        R fun2
6     Java fun2
7      PHP fun2
8   Python fun2
9        R fun3
10    Java fun3
11     PHP fun3
12  Python fun3
```

```
> reshape(wide, direction = "long", varying =3:4)
        program time.fun1  time id
1.fun2        R      8.31  8.72  1
2.fun2     Java      8.45  4.15  2
3.fun2      PHP     10.49 11.47  3
4.fun2   Python     10.45 13.25  4
1.fun3        R      8.31 10.10  1
2.fun3     Java      8.45 13.86  2
3.fun3      PHP     10.49  9.96  3
4.fun3   Python     10.45 14.64  4
```

总结

从本文的介绍来看，要做好数据处理是相当不容易的。你要知道数据是什么样的，业务逻辑是什么，怎么写程序以及数据变形，最后怎么进行BI展示，表达出正确的分析维度。

试试R语言，忘掉程序员的思维，换成数据的思维，也许繁琐的数据处理工作会让你开心起来。

作者介绍

张丹，《R的极客理想》系列图书作者

Thank you!

用R语言把数据玩出花样

http://blog.fens.me/r-transform/