

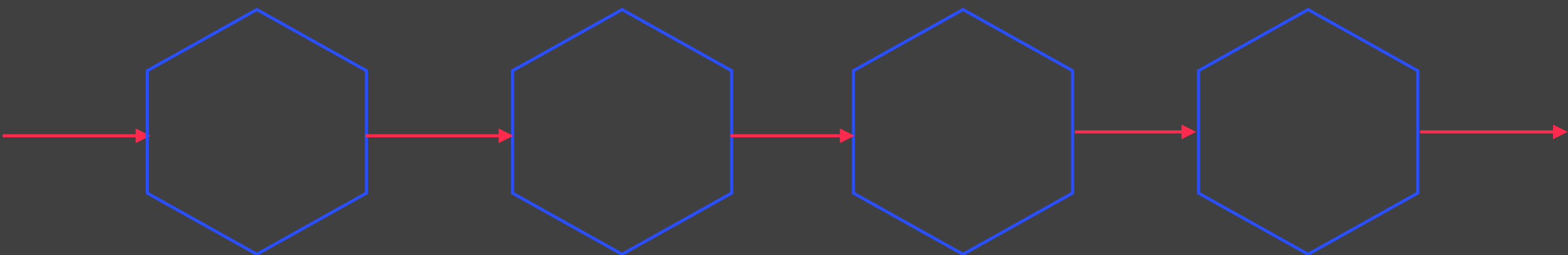
# R编程结构



- 矩阵运算: `t()`; `det()`; `array()`; `crossprod()`; `tcrossprod()`; `diag()`; `solve()`; `eigen()`;
- 缺失值: `NA`; `is.na()`; `na.rm = TRUE`; `na.omit()`;
- 类型函数: `is.numeric()`; `is.integer()`; `is.logical()`; `is.character()`; `as.xxxx()`
- 字符处理: `nchar()`; `substr()`; `strsplit()`; `toupper()`; `tolower()`; `paste()`;
- 日期和时间: `Sys.Date()`; `date()`; `difftime()`; `format()`; `as.Date()`; `%d`, `%a`, `%A`, `%m`, `%b`, `%B`, `%y`, `%Y`;
- 统计函数: `mean()`; `median()`; `sd()`; `var()`; `max()`; `min()`; `range()`; `sum()`; `quantile()`; `diff()`; `scale()`;
- 数据集合并: `rbind()`; `cbind()`;
- 其余: `apply()`;

- 输入输出
- 流程控制
- 循环控制
- 自写函数

输入输出



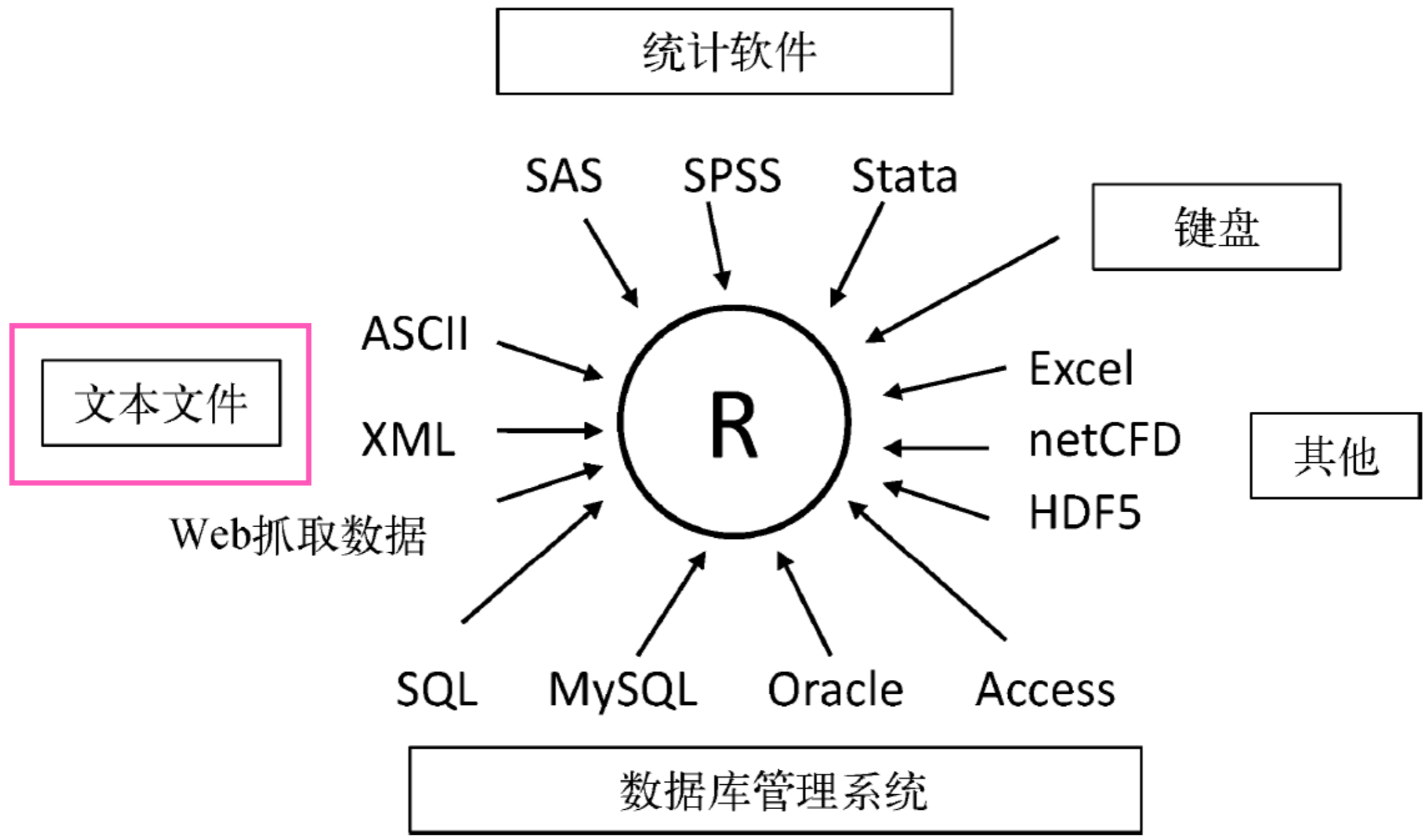



图2-2 可供R导入的数据源

表2-2 函数read.table()的选项

| 选 项              | 描 述  |
|------------------|--|
| header           | 一个表示文件是否在第一行包含了变量名的逻辑型变量   |
| sep              | 分开数据值的分隔符。默认是 <code>sep=""</code> ，这表示了一个或多个空格、制表符、换行或回车。使用 <code>sep=","</code> 来读取用逗号来分隔行内数据的文件，使用 <code>sep="\t"</code> 来读取使用制表符来分割行内数据的文件  |
| row.names        | 一个用于指定一个或多个行标记符的可选参数   |
| col.names        | 如果数据文件的第一行不包括变量名 ( <code>header=FALSE</code> )，你可以用 <code>col.names</code> 去指定一个包含变量名的字符向量。如果 <code>header=FALSE</code> 以及 <code>col.names</code> 选项被省略了，变量会被分别命名为 <code>v1</code> 、 <code>v2</code> ，以此类推   |
| na.strings       | 可选的用于表示缺失值的字符向量。比如说， <code>na.strings=c("-9", "?")</code> 把 <code>-9</code> 和 <code>?</code> 值在读取数据的时候转换成 <code>NA</code>  |
| colClasses       | 可选的分配到每一列的类向量。比如说， <code>colClasses=c("numeric", "numeric", "character", "NULL", "numeric")</code> 把前两列读取为数值型变量，把第三列读取为字符型向量，跳过第四列，把第五列读取为数值型向量。如果数据有多余五列， <code>colClasses</code> 的值会被循环。当你在读取大型文本文件的时候，加上 <code>colClasses</code> 选项可以可观地提升处理的速度 |
| quote            | 用于对有特殊字符的字符串划定界限的自负载。默认值是双引号 ( <code>"</code> ) 或单引号 ( <code>'</code> )  |
| skip             | 读取数据前跳过的行的数目。这个选项在跳过头注释的时候比较有用   |
| stringsAsFactors | 一个逻辑变量，标记处字符向量是否需要转化成因子。默认值是 <code>TRUE</code> ，除非它被 <code>colClasses</code> 所覆盖。当你在处理大型文本文件的时候，设置成 <code>stringsAsFactors=FALSE</code> 可以提升处理速度   |
| text             | 一个指定文字进行处理的字符串。如果 <code>text</code> 被设置了， <code>file</code> 应该被留空。2.3.1 节给出了一个例子   |


```
StudentID,First,Last,Math,Science,Social Studies
011,Bob,Smith,90,80,67
012,Jane,Weary,75,,80
010,Dan,"Thornton, III",65,75,70
040,Mary,"O'Leary",90,95,92
```



```
> grades <- read.table("studentgrades.csv", header=TRUE,
+                       row.names="StudentID", sep=",")
```


```
> grades # print data frame
```

|    | First | Last          | Math | Science | Social.Studies |
|----|-------|---------------|------|---------|----------------|
| 11 | Bob   | Smith         | 90   | 80      | 67             |
| 12 | Jane  | Weary         | 75   | NA      | 80             |
| 10 | Dan   | Thornton, III | 65   | 75      | 70             |
| 40 | Mary  | O'Leary       | 90   | 95      | 92             |



```
> str(grades) # view data frame structure
```

```
'data.frame':  4 obs. of  5 variables:
 $ First      : Factor w/ 4 levels "Bob","Dan","Jane",...: 1 3 2 4
 $ Last       : Factor w/ 4 levels "O'Leary","Smith",...: 2 4 3 1
 $ Math       : int  90 75 65 90
 $ Science    : int  80 NA 75 95
 $ Social.Studies: int  67 80 70 92
```





```
> grades <- read.table("studentgrades.csv", header=TRUE,  
+                       row.names="StudentID", sep=";",  
+                       colClasses=c("character", "character", "character",  
+                                     "numeric", "numeric", "numeric"))
```

```
> grades # print data frame
```

|     | First | Last          | Math | Science | Social.Studies |
|-----|-------|---------------|------|---------|----------------|
| 011 | Bob   | Smith         | 90   | 80      | 67             |
| 012 | Jane  | Weary         | 75   | NA      | 80             |
| 010 | Dan   | Thornton, III | 65   | 75      | 70             |
| 040 | Mary  | O'Leary       | 90   | 95      | 92             |

```
> str(grades) # view data frame structure
```

```
'data.frame':  4 obs. of  5 variables:  
 $ First      : chr  "Bob" "Jane" "Dan" "Mary"  
 $ Last       : chr  "Smith" "Weary" "Thornton, III" "O'Leary"  
 $ Math       : num  90 75 65 90  
 $ Science    : num  80 NA 75 95  
 $ Social.Studies: num  67 80 70 92
```





```
read.table(file,  
           header=FALSE,  
           sep="||",  
           row.names="||")
```

---

```
write.table(file,  
            append=FALSE,  
            sep="||",  
            row.names=TRUE,  
            col.names=TRUE)
```

---

```
read.csv()  
write.csv()
```

2.5 已知有 5 名学生的数据，如表 2.3 所示。用数据框的形式读入数据。

表 2.3: 学生数据

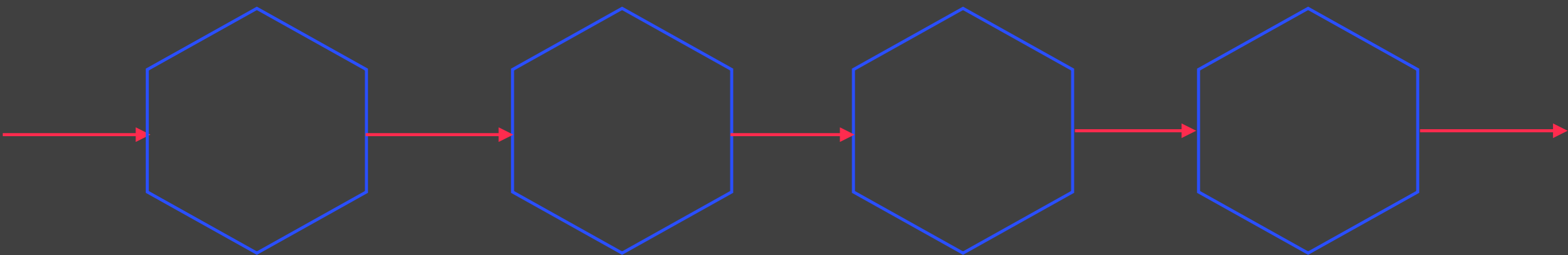
| 序号 | 姓名 | 性别 | 年龄 | 身高 (cm) | 体重 (kg) |
|----|----|----|----|---------|---------|
| 1  | 张三 | 女  | 14 | 156     | 42.0    |
| 2  | 李四 | 男  | 15 | 165     | 49.0    |
| 3  | 王五 | 女  | 16 | 157     | 41.5    |
| 4  | 赵六 | 男  | 14 | 162     | 52.0    |
| 5  | 丁一 | 女  | 15 | 159     | 45.5    |

2.6 将例 2.5 中的数据表 2.3 的数据写成一个纯文本文件，用函数 `read.table()` 读该文件，然后再用函数 `write.csv()` 写成一个能用 *Excel* 表能打开的文件，并用 *Excel* 表打开。

10分钟完成

随机找学生讲解

# 流程控制



|         |   |
|---------|---|
| if-else | <i>if(cond) statement<br/>if(cond) statement1 else statement2</i> |
| ifelse  | <i>if(cond, statement1, statement2)</i>                           |

```
> if(FALSE)
+ {
+   message("This won't execute...")
+ }else
+ {
+   message("and you'll get an error before you reach this.")
+ }
and you'll get an error before you reach this.
```

一个表达式，不能是NA

```
> ifelse(rbinom(10, 1, 0.5), "Head", "Tail")
[1] "Tail" "Head" "Head" "Head" "Tail" "Head" "Head" "Tail"
[9] "Tail" "Tail"
```

switch

*switch(expr, ...)*

```
> feelings <- c("sad", "afraid")
> for (i in feelings)
+   print(
+     switch(i,
+       happy = "I am glad you are happy",
+       afraid = "There is nothing to fear",
+       sad = "Cheer up",
+       angry = "Calm down now"
+     )
+   )
[1] "Cheer up"
[1] "There is nothing to fear"
```

|        |                                  |
|--------|----------------------------------|
| repeat | <i>repeat(statement)</i>         |
| for    | <i>for(var in seq) statement</i> |
| while  | <i>while(cond) statement</i>     |

```

> repeat
+ {
+   message("Happy Groundhog Day!")
+   action <- sample(
+     c(
+       "Learn French",
+       "Make an ice statue",
+       "Rob a bank",
+       "Win heart of Andie McDowell"
+     ),
+     1
+   )
+   message("action = ", action)
+   if(action == "Win heart of Andie McDowell") break
+ }

```

```

-
Happy Groundhog Day!
action = Make an ice statue
Happy Groundhog Day!
action = Rob a bank
Happy Groundhog Day!
action = Win heart of Andie McDowell

```

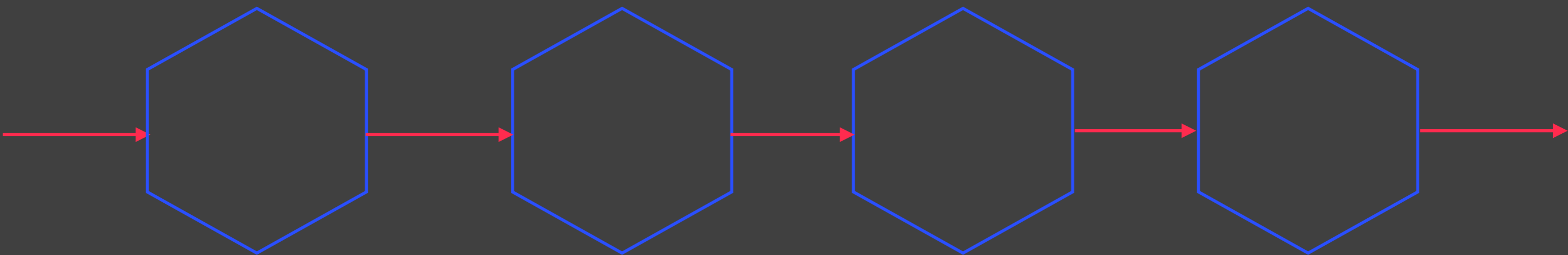
break

next





# 自写函数



```
function(arg1,arg2,...) {  
  statements  
  return (object)  
}
```

```
myfun <- function() {  
  print("hello world")  
  return ()  
}
```

```
> f <- function(x,y) x + y  
> f  
function(x,y) x + y  
> f(1,2)  
[1] 3
```

**代码清单5-8** `mystats()`: 一个由用户编写的描述性统计量计算函数

```
mystats <- function(x, parametric=TRUE, print=FALSE) {  
  if (parametric) {  
    center <- mean(x); spread <- sd(x)  
  } else {  
    center <- median(x); spread <- mad(x)  
  }  
  if (print & parametric) {  
    cat("Mean=", center, "\n", "SD=", spread, "\n")  
  } else if (print & !parametric) {  
    cat("Median=", center, "\n", "MAD=", spread, "\n")  
  }  
  result <- list(center=center, spread=spread)  
  return(result)  
}
```

---

```
set.seed(1234)  
x <- rnorm(500)
```

---

```
y <- mystats(x)
```

---

```
y <- mystats(x, parametric=FALSE, print=TRUE)
```

```
Median= -0.0207  
MAD= 1
```

```
mydate <- function(type="long") {  
  switch(type,  
    long = format(Sys.time(), "%A %B %d %Y"),  
    short = format(Sys.time(), "%m-%d-%y"),  
    cat(type, "is not a recognized type\n")  
  )  
}
```

---

```
> mydate("long")  
[1] "Monday July 14 2014"  
> mydate("short")
```

```
[1] "07-14-14"  
> mydate()  
[1] "Monday July 14 2014"  
> mydate("medium")  
medium is not a recognized type
```

**2.7** 编写一个 R 程序 (函数). 输入一个整数  $n$ , 如果  $n \leq 0$ , 则中止运算, 并输出一句话: “要求输入一个正整数”; 否则, 如果  $n$  是偶数, 则将  $n$  除 2, 并赋给  $n$ ; 否则, 将  $3n + 1$  赋给  $n$ . 不断循环, 只到  $n = 1$ , 才停止计算, 并输出一句话: “运算成功”. 这个例子是为了检验数论中的一个简单的定理.

● 0011-1

**例 2.4** 编写一个用二分法求非线性方程根的函数, 并求方程

● 0011-2

$$x^3 - x - 1 = 0$$

在区间  $[1, 2]$  内的根, 精度要求  $\varepsilon = 10^{-6}$ .

二分法计算过程如下: 取中点  $x = \frac{a+b}{2}$ , 若  $f(a)$  与  $f(x)$  异号, 则置  $b = x$ ; 否则  $a = x$ . 当区间长度小于指定要求时, 停止计算.

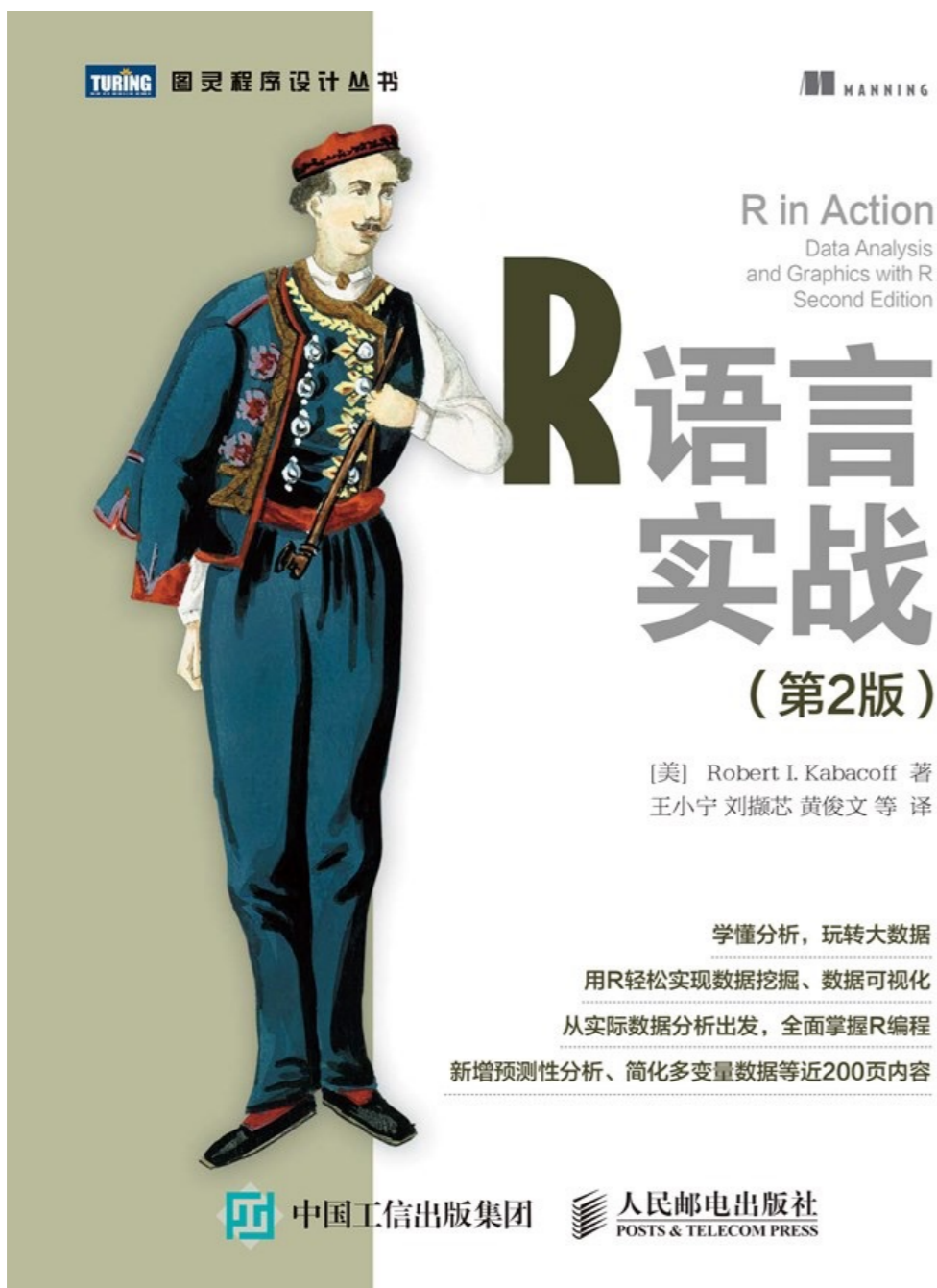
10分钟完成

随机找学生讲解

# 提问时间!

孙惠平

[sunhp@ss.pku.edu.cn](mailto:sunhp@ss.pku.edu.cn)



2.3、5.4、5.5  
例子5-6、5-8




第四章



INTERACTIVE COURSE

### Case Study: Exploring Baseball Pitching Data in R

Start Course For Free | Play Intro Video | Bookmark



4 hours | 14 Videos | 69 Exercises | 7,934 Participants | 5,750 XP


提交方式和上节课一样!

<https://www.datacamp.com/courses>

INTERACTIVE COURSE

### Intermediate R: Practice

Start Course For Free | Bookmark



4 hours | 0 Videos | 52 Exercises | 58,845 Participants | 4,800 XP

谢谢!

孙惠平

[sunhp@ss.pku.edu.cn](mailto:sunhp@ss.pku.edu.cn)