---

title: "getting and cleaning data"

author: "Dingchong"

output: html\_document

---

# WEEK 1 概述和读取数据

##1.概述

数据获取和清洗，这个过程所费的时间精力可能会占到整个项目的80%，有了干净的数据才有接下去才的数据分析Raw data -> Processing script -> tidy data

这中间有4样东西：

raw data 原始数据，未经加工的，从各处拿过来的数据

tidy data set 整理好的数据集，变量名应该是人类可以读懂的，比如AgeAtDiagnosis而不是AgeDx。。。

A code book describing each variable and its values in the tidy data set.

An explicit and exact recipe you used to go from 1 -> 2, 3.（The instruction list）

The code book应包含的要件如下

1.tidy data 里面没有包含的变量信息

2.数据汇总的说明

3.采用的实验设计说明

4.应该是word/txt文档

5.应该说明你获取原始数据的情况

6.描述每一个变量

The instruction list应该是一个可执行的代码脚本（R，或者其他语言），

输入是raw data，输出是tidy data，可以重复执行。

总结一下这个部分，为了做到科学严谨（在商业应用中一样有重要性）：

从raw data到tidy data的处理过程一定要记录下来，可重复执行和检查；

处理的方式和流程要有文档，越清楚越方便后面检查和优化，至于形式，自己斟酌就好，不必严格按照课程的来。

##2.获取数据——internet

download.file()

可重复执行，可下载txt,csv等格式，关键参数包括url,destfile,method。

例子：

```{r}

setwd("e:/")

fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"

download.file( fileUrl, destfile = "./cameras.csv" ) # mac下加这个 , method ="curl" ，未证实

list.files("./") #下好了

```

注意事项

1.http开头的，默认下载就好；https开头的在MAC下需要加method ="curl"

##3.获取本地数据——read.table()

read.table()

最常用的函数，但是数据量受内存限制；

重要参数包括

  file,文件名；

  header,是否包含表头；

  sep,间隔符号；

  row.names,可以定义行名，默认或者NULL则是序号；

  nrows，读入行数，数据量大是分批读入用；

  quote，指定字符引用的符号`'" 之类的；

  na.string，代表NA的字符，如果raw data是SQL出来的，这里可以指定NULL；

  skip，从第几行开始读；

相关read.csv(), read.csv2(), read.delim()...

##4.读excel表格

read.xlsx() { xlsx package}

略

##5.读取XML

library(XML)

fileUrl <- "http://www.w3schools.com/xml/simple.xml"

doc <- xmlTreeParse( fileUrl, useInternal=T)

rootNode <- xmlRoot(doc)

xmlName(rootNode)

xmlValue(rootNode)

rootNode[[1]]

rootNode[[1]][[1]]

xmlSApply( rootNode, xmlValue)

xpathSApply( rootNode, "//name", xmlValue)

xpathSApply( rootNode, "//price", xmlValue)

```

##6.read JSON data

```{r}

library(jsonlite)

# read

jsonData <- fromJSON( "https://api.github.com/users/jtleek/repos" )

names(jsonData)

jsonData$owner$login

# write

myjson <- toJSON(iris, pretty = T )

myjson

```

### WEEK2

##1.read from APIs

```{r}

# twitter

```

## quiz 1

```{r}

#1.2

fileUrl <- "https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06hid.csv"

download.file( fileUrl, destfile = "./cameras.csv" )

cameras <- read.csv("E:/Program Files/RStudio/cameras.csv")

table(cameras$VAL)

table(cameras$FES)

#3

library(xlsx)

dat <- read.xlsx("E:/getdata-data-DATA.gov\_NGAP.xlsx", sheetIndex=1,

                  startRow= 18, endRow= 23, colIndex= 7:15)

sum(dat$Zip\*dat$Ext,na.rm=T)

#4

library(XML)

fileUrl <- "http://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml"

doc <- xmlTreeParse( fileUrl, useInternal=T )

Node <- xmlRoot(doc)

zipc <- xpathSApply( Node[[1]], "//zipcode", xmlValue )

length( zipc[ zipc=="21231"] )

#5

library(data.table)

DT <- fread( "e:/getdata-data-ss06pid.csv")

rowMeans(DT)[DT$SEX==1]; rowMeans(DT)[DT$SEX==2] # ==

system.time( {

    mean(DT[DT$SEX==1,]$pwgtp15); mean(DT[DT$SEX==2,]$pwgtp15)

} )

system.time( {

  sapply(split(DT$pwgtp15,DT$SEX),mean)

})

system.time( {

  mean(DT$pwgtp15,by=DT$SEX)

})

tapply(DT$pwgtp15,DT$SEX,mean)

DT[,mean(pwgtp15),by=SEX]

```

Getting and Cleaning data

# week 2

## Reading from MySQL

hg19 <- dbConnect(MySQL(), user="genome", db="hg19",

host="genome-mysql.cse.ucsc.edu")

allTables <- dbListTables(hg19)

dbListFields(hg19, "affyU133Plus2")

dbGetQuery(hg19, "select count(\*) from affyU133Plus2")

affyData <- dbReadTable(hg19, "affyU133Plus2")

Quiz Q1