Análise exploratória de dados: carregamento de dados, sua reresentação e exploração inicial

Representação dos dados, sua exploração; implementação em R

Motivação

- Evolução tecnológica: capacidade de criar grandes quantidades de dados– era do Big data
- Cálculo de 1.7 Mbytes de dados criados no mundo cada segundo; mais de 40 zetabytes (10^{21}) criados até 2020
- Cada vez maior dificuldades dos peritus humanos em cada área de processor a analisar dados em bruto
- Num mundo de competitividade crescente, há grandes vantagens em ser capaz de **extrair conhecimento a partir dos dados** em larga-escala
- As áreas biomédicas são as que geram mais dados entre todas as áreas científicas
- A Biologia é cada vez mais uma ciência “data-driven”

Definição de dados

- Valores de **variáveis (atributos, campos)** **quantitativas** ou **qualitativas** que caraterizam um conjunto de **entidades (objetos, exemplos)**
- Conjunto de **entidades**: denominado **população**; conjunto de itens em que estamos interessados
- **Variáveis**: medida ou característica das entidades
- **Qualitativas** (ou discretas, nominais): medem características com conjunto (finito) de valores (e.g. sexo, tipo de tratamento); podem ser chamadas de **ordinais** – se valores tiverem uma ordem natural; caso particular - **binárias**
- **Quantitativas** (ou contínuas): medem características com valores numéricos num dado intervalo (e.g. altura, pressão sanguínea)

Dados em bruto vs processados

Dados em **bruto**:

- normalmente designam dados originais, tal como são recolhidos
- são difíceis de usar de forma direta na análise de dados
- necessitam de **preparação de dados** (normalmente feito apenas uma vez) para transformar os dados de forma a serem usados na análise
- Preparação dos dados pode incluir junção de dados, seleção de dados, limpeza dos dados, normalização e outras transformações
- Passos do processamento devem ser guardados
- Exemplos: dados em bruto de DNA microarrays ou dados de GC-MS ou RMN

Dados **processados**:

- dados “prontos” para análise
- tipicamente numa forma standard (matricial)

Dados processados: características desejáveis

Nomes das variáveis fáceis de usar e informativas

Dados não devem ter valores errados

Desejavelmente, dados não devem ter valores omissos (embora tal seja em muitos casos impossível)

Valores das variáveis são consistentes e todas as variáveis transformadas necessárias foram adicionadas

Estrutura dos dados

Os dados processados são tipicamente apresentados numa forma **matricial**:

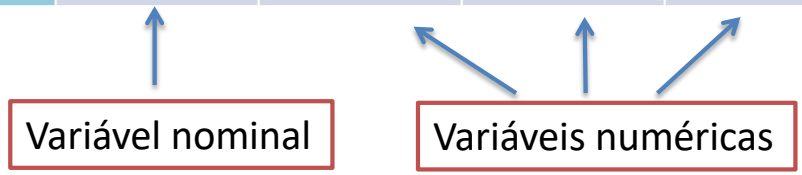
- Cada **variável** apresenta-se numa **coluna**; também designados por campos ou atributos; o nome da coluna é o identificador da variável
- Cada **entidade** (ou observação) está representada numa **linha;** também designados por exemplos (em alguns casos cada entidade tem um identificador que dá nome à linha)
- Cada **matriz** ou **tabela** representa dados sobre um tipo de observação (também designado por **conjunto de dados** ou *dataset*)

Estrutura dos dados

Variáveis, campos, atributos

Entidades, objetos, itens, exemplos

	Sexo	Tensão Max	Tensão Min	Pulsações
id1	M	14.4	10.1	67
id2	F	12.1	7.3	84
id3	F	11.5	7.0	76
id4	M	15.1	9.9	112
id5	M	13.6	8.9	68
id6	F	12.9	8.0	76
…	…	…	…	…



Representação dos dados em R

Estruturas de dados centrais na representação de dados em R são os **data frames** e as **matrizes**, ambos com uma estrutura matricial capaz de representar dados processados

Diferença principal: nas matrizes todos os dados têm que ser do mesmo tipo, pelo que apenas são adequadas para dados numéricos

Os **data frames** permitem representar variáveis contínuas (numéricas) e variáveis discretas

- Um data frame é uma ***list*** em R, em que cada um dos campos corresponde a uma variável (coluna)
- Cada campo é um vetor de valores numéricos (variável contínua) ou um fator (variável discreta)
- Todos os campos têm o mesmo comprimento – número de entidades (linhas)
- Cada linhas e colunas podem ter um nome (string)

Data frame: exemplo dataset interno iris

```
> data(iris)
> dim(iris)
[1] 150   5
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6  5.4         3.9          1.7         0.4  setosa

> unlist(lapply(iris,class))
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
   "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

Iris

Data frame contendo 4 variáveis numéricas e uma variável discreta. Os vetores numéricos representam comprimento e largura de pétalas e sépalas de 150 flores de 3 espécies: *setosa*, *virginica* and *versicolor* (dado no último campo, um vetor de fatores com 3 níveis possíveis)

Fontes de dados

Os dados para análise estão disponíveis em diversas fontes e em múltiplos formatos e estruturas

Alguns exemplos de formatos típicos:

- Ficheiros de texto tipo CSV ou TSV
- Ficheiros de folhas de cálculo (e.g. Excel)
- Bases de dados (e.g. MySQL)
- Ficheiros em formato XML ou noutros formatos como JSON

Possíveis fontes

- WWW / internet
- Recolhida por aplicações específicas
- Cedidas por colegas ou outros investigadores
- Provenientes de trabalhos de investigação próprios
- ...

Leitura de dados em R

Ficheiros em formato CSV / TSV

- Função **read.table**: função base de leitura de ficheiros texto em formato tabular; retorna um data frame
- Argumentos:
  - Nome do ficheiro (assumido na pasta de trabalho) ou caminho completo
  - Separador (*sep*) – por omissão considerado espaço ou tab
  - *header* – define se a primeira linha tem headers de colunas ou não (TRUE ou FALSE); por omissão F
  - *dec* – carácter usado como ponto decimal
  - *row.names* e *col.names* – vetores com nomes de linhas / colunas
  - *na.strings* – string usada para definir valores omissos (NA)
  - *as.is, stringAsFactors* – controlam que campos são transformados em fatores
  - Outros – ver *help(read.table)*
- Variante **read.csv** – assume *sep = ","* e *header = T*

Ficheiros em formato Excel: função **read.xlsx** (package xlsx)

Função para carregar ficheiro: **download.file**

Leitura de dados em R: exemplo

```
> fileUrl = "http://archive.ics.uci.edu/ml/machine-learning-
databases/ecoli/ecoli.data"
> download.file(fileUrl, destfile="ecoli.csv")

> ecoli = read.table("ecoli.csv")
> dim(ecoli)
[1] 336   9
> head(ecoli)
          V1   V2   V3   V4  V5   V6   V7   V8 V9
1  AAT_ECOLI 0.49 0.29 0.48 0.5 0.56 0.24 0.35 cp
2 ACEA_ECOLI 0.07 0.40 0.48 0.5 0.54 0.35 0.44 cp
3 ACEK_ECOLI 0.56 0.40 0.48 0.5 0.49 0.37 0.46 cp
4 ACKA_ECOLI 0.59 0.49 0.48 0.5 0.52 0.45 0.36 cp
5  ADI_ECOLI 0.23 0.32 0.48 0.5 0.55 0.25 0.35 cp
6 ALKH_ECOLI 0.67 0.39 0.48 0.5 0.36 0.38 0.46 cp
```

Note que o ficheiro não tem headers para as colunas

Leitura de dados em R: exemplo *earthquakes*

```
> ficheiro = "http://darwin.di.uminho.pt/cursoAnaliseDados/earthquakeData.csv"
> download.file(ficheiro, destfile = "earthquakes.csv")

> eData = read.csv("earthquakes.csv")
> dim(eData)
[1] 360  10
> names(eData)
 [1] "Src"       "Eqid"      "Version"   "Datetime"  "Lat"       "Lon"
 [7] "Magnitude" "Depth"     "NST"       "Region"
> head(eData)
  Src     Eqid Version                                Datetime     Lat
1  nc 72289771       1 Monday, September  1, 2014 17:09:40 UTC 37.4703
2  (…)
```

Ficheiro com headers; read.csv assume headers por omissão

Escrita / exportação de dados

A tarefa mais comum é escrever uma matriz ou data frame para ficheiro num formato tabular (e.g. CSV/ TSV)

```
> write.table(mydata, "c:/mydata.txt", sep="\t")
```

```
> write.csv(MyData, file = "MyData.csv”,row.names=FALSE, na="")
```

É possível importar / exportar dados para inúmeras plataformas: SPSS, SAS, Stata (**foreign**). Para Excel, pode usar-se o package **xlsx**.

```
> library(xlsx)
> write.xlsx(mydata, "c:/mydata.xlsx")
```

Preparação dos dados

Para chegar à estrutura desejável dos dados, dada anteriormente, podem ser necessárias diversas etapas a partir dos dados reais, “em bruto”

Primeiro passo deve passar por **selecionar** os dados desejáveis e **carregá-los** para um formato de trabalho, podendo haver necessidade de selecionar dados, juntar dados de várias fontes e formatos, etc.

Esta fase não será coberta neste curso dada a sua heterogeneidade e dependência dos objetivos (cada caso é um caso)

Assumindo que após essa fase, temos os dados num formato matricial (tipo *data frame*), ainda haverão tipicamente várias tarefas a realizar para termos os dados prontos para a análise de dados, que veremos em seguida

Exploração inicial dos dados

A primeira etapa no tratamento preliminar dos dados será a sua **exploração**, com fim a diagnosticar possíveis problemas e desenhar soluções para resolvê-los

Existem várias formas de explorar os dados, com métricas diversas e com ferramentas gráficas; ambos os casos serão explorados

Este processo será iterativo, correndo-se ferramentas de exploração dos dados, fazendo a sua correção e tornando a correr para validar os métodos, até que não subsistam problemas

Soluções podem passar pela correção da estrutura dos dados, pela mudança de nomes de variáveis, pela transformação de variáveis, pelo tratamento de valores omissos, pela limpeza de dados erróneos ou inconsistentes.

Verificando estrutura dos dados

Objectivo: procurar problemas nos dados que requeiram ações de pré-processamento:

Tarefas

- Verificar **nomes dos campos**
- Verificar **dimensões** dos dados: nº de linhas e colunas
- Verificar **estrutura dos dados**: classes dos campos/ tipos de dados, valores de cada fator

Se detectados problemas devem corrigir-se

Em alguns casos, problemas advêm de leituras incorretas (e.g. parâmetros no **read.table**) ou de problemas nos próprios dados (e.g. ficheiros incorretos); casos comuns são separadores errados, mau uso da flag que define headers, etc.

Noutros casos, é necessário escrever scripts para corrigir problemas

Verificando estrutura dos dados: exemplo

```
> dim(eData)
[1] 360 10
> nrow(eData)
[1] 360
> ncol(eData)
[1] 10
> names(eData)
 [1] "Src"       "Eqid"      "Version"   "Datetime"  "Lat"       "Lon"       "Magnitude" "Depth"
"NST"       "Region"
> class(eData)
[1] "data.frame"
> unlist(lapply(eData, class))
Src      Eqid   Version  Datetime      Lat       Lon    Magnitude     Depth       NST    Region
"factor" "factor"  "factor"  "factor" "numeric" "numeric" "numeric" "numeric" "integer" "factor"
> unlist(lapply(eData, typeof))
Src        Eqid   Version     Datetime    Lat       Lon    Magnitude   Depth      NST    Region
"integer" "integer" "integer" "integer"  "double"  "double" "double" "double" "integer" "integer"
```

Corrigir estrutura dos dados: exemplos de mudança de nomes

```
> names(eData)
 [1] "Src"       "Eqid"      "Version"   "Datetime"  "Lat"       "Lon"       "Magnitude"
"Depth"     "NST"       "Region"
> names(eData)[5]="Latitude"
> names(eData)[6]="Longitude"
> names(eData)
 [1] "Src"       "Eqid"      "Version"   "Datetime"  "Latitude"  "Longitude" "Magnitude"
"Depth"     "NST"       "Region"
> row.names(eData)
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8" …
> row.names(eData) = paste("id-", row.names(eData), sep="")
> row.names(eData)
  [1] "id-1"   "id-2"   "id-3"   "id-4"   "id-5"   "id-6"   "id-7"   "id-8" …
```

Exemplos acima mostram como mudar nome de campos ou linhas

Para manipulações mais complexas pode haver necessidade de recorrer a

Funções sobre strings: ver exemplos – *sub*, *gsub*, *grep*, *paste*, *toupper*, *tolower*, *chartr*, etc.

Filtros

Filtros podem ser aplicados tendo em conta valores específicos de campos ou a sua combinação (e.g. usando operadores lógicos)

Permitem criar "**views**" parciais de data frames ou matrizes que podem ser úteis na sumarização dos dados

Podem ser criadas segmentações dos dados de acordo com o o valor de alguns campos

Podem ser usados para verificar se dadas combinações de valores ocorrem na medida do expectável

Filtros: exemplos

```
> eData$Latitude[1:10] > 40
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
> any(eData$Lat[1:10] > 40)
[1] TRUE
> all(eData$Lat[1:10] > 40)
[1] FALSE
> which(eData$Lat[1:10] > 37)
[1]  1  7 10
> sum(eData$Lat > 60)
[1] 3
> eData[eData$Latitude > 38 & eData$Longitude > -119, c("Latitude","Longitude")]
   Latitude Longitude
22  45.4572 -117.6165
43 38.0143 -118.6437
…
```

```
> subData = subset(eData, Longitude < -125 | Latitude > 40)
> dim(subData)
[1] 101  10
```