Aula 3 - Estruturas de dados no R

Eduardo Koerich Nery

Nesta aula iremos explorar as estruturas de dados na linguagem R. Se na língua portuguesa nós unimos frases para formar diferentes tipos de texto, na linguagem R nós podemos organizar os vetores em diferentes estruturas de dados: **matriz, arranjo, fator, tabela de dados e lista**. Essas estruturas diferem quanto ao número de dimensões e tipos de dados armazenados, sendo portanto mais úteis para diferentes tipos de tarefas.

Matriz

Uma matriz é *um conjunto de vetores numéricos de mesmo tamanho*. As matrizes são estruturas bidimensionais, ou seja, possuem linhas e colunas. Uma vez que as matrizes possuem *apenas dados numéricos*, elas são passíveis das operações aritméticas. As operações com as matrizes vão seguir as regras da álgebra matricial. Uma matriz pode resultar da (a) união de vetores com mesmo tamanho ou da (b) divisão de um único vetor.

a) Para construir uma matriz a partir da união de vetores, nós podemos usar as funções `rbind()` ou `cbind()`, "colar linhas" e "colar colunas", respectivamente. Por exemplo, dada as alturas, massas e idades dos membros da minha família, eu posso organizar esses vetores como linhas ou como colunas de uma matriz. Perceba que o nome dos vetores são preservados na matriz, identificando as linhas ou as colunas que os vetores formaram.

```
alturas <- c(1.84,1.74,1.69)
massas <- c(72,95,67)
idades <- c(29,38,59)

rbind(alturas,massas,idades)

cbind(alturas,massas,idades)
```

b) Para construir uma matriz a partir da divisão de um vetor, nós podemos usar a função `matrix()`, especificando o número de linhas e de colunas desejadas e como serão organizados os dados. Por exemplo, digamos que eu tenha os dados de altura, massa e idade de um grupo de pessoas em um único vetor `pessoas`. Eu posso dividir esse vetor em uma matriz de 3 colunas e 3 linhas de duas formas: organizando por linhas `byrow = TRUE` ou organizando por colunas `byrow = FALSE`.

```
pessoas <- c(1.84,1.74,1.69,72,95,67,29,38,59)

m.pessoas <- matrix(pessoas, nrow = 3, ncol = 3, byrow = TRUE)

m.pessoas
```

Em alguns casos, podemos encontrar uma estrutura de dados com símbolos numéricos, mas que não é interpretada como matriz. Quando isso ocorre, podemos transformar essa estrutura em uma matriz com a função `as.matrix()`. Para verificar se um objeto é uma matriz ou não, podemos usar a função `is.matrix()`.

Assim como nos vetores, cada dado de uma matriz está indexado a uma posição numérica. Essa posição é especificada pelo número da linha e pelo número da coluna ocupada pelo dado. Para acessar uma posição dentro de uma matriz, nós usamos os colchetes `[]` ao lado da matriz, e especificamos dentro dos colchetes o número da linha e da coluna de interesse. Por exemplo, dada a matriz `m.pessoas`, eu posso acessar a segunda linha na terceira coluna com o comando `m.pessoas[2,3]`; acessar apenas a segunda linha `m.pessoas[2,]`; ou apenas a terceira coluna e `m.pessoas[,3]`.

Para verificar o tamanho de uma matriz, podemos usar a função `dim()`. Isto é importante pois algumas operações precisam que as matrizes tenham dimensões iguais (ou que a transposição de uma matriz tenha as mesma dimensões de uma outra matriz). Também , como veremos a seguir, matrizes com mesmo tamanho podem ser unidas em uma estrutura maior de dados, os arranjos.

Arranjo

Um arranjo é *um conjunto de matrizes de mesmo tamanho.* Aqui vamos explorar apenas os arranjos tridimensionais, ou seja, os que possuem linhas, colunas e camadas. Numa comparação simples, podemos imaginar um arranjo tridimensional como uma pilha de folhas, onde cada folha é uma matriz. Para construir um arranjo, nós podemos usar a função `array()`, especificando o número de linhas, colunas e camadas.

Por exemplo, para guardar os dados de idade e massa das pessoas em dois anos diferentes, eu criei duas matrizes com três linhas e duas colunas, `ano.1` e `ano.2`. Posteriormente eu organizei essas matrizes em um arranjo de duas camadas `dois.anos`. Perceba que no argumento `dim` eu especifiquei as três dimensões do arranjo `c(3,2,2)`, ou seja, 3 linhas, 2 colunas e 2 camadas. As dimensões sempre são especificadas nesta ordem!

```
ano.1 <- matrix(c(29,38,59,72,95,67),  nrow = 3, ncol = 2, byrow = FALSE)

ano.2 <- matrix(c(19,28,49,67,77,60), nrow = 3, ncol = 2, byrow = FALSE)

arr.pessoas <- array(c(ano.1,ano.2), dim=c(3,2,2))

arr.pessoas
```

Cada dado de um arranjo está indexado a uma posição. No arranjo tridimensional, essa posição é especificada pelo número da linha, da coluna e da camada ocupada pelo dado. Para acessar uma posição dentro de um arranjo, nós usamos os colchetes `[]` ao lado do arranjo, e especificamos dentro dos colchetes o número da linha, da coluna e da camada. Por exemplo, uma vez feito o arranjo sobre os dados das pessoas em dois diferentes anos, eu posso acessar apenas a segunda camada `arr.pessoas[,,2]`; apenas a primeira coluna da segunda camada `arr.pessoas[,1,2]`; ou então apenas a terceira linha, na primeira coluna, na segunda camada `arr.pessoas[3,1,2]`.

Fator

Um fator é *um vetor para classificar os dados*, ou seja, uma categorização dos dados. Um fator sempre deriva de um vetor que pode ser numérico ou de caracteres. Para criar um fator, nós podemos usar a função `factor()`. Por exemplo, para classificar os membros da minha família segundo o sexo biológico, eu criei um vetor de caracteres `sexo` e posteriormente transformei o vetor em um fator `f.sexo`.

```
sexo <- c("M","M","F")

f.sexo <- factor(sexo)

f.sexo
```

Dentro de um fator, cada valor (categoria) é chamado de **nível**. Para acessar os níveis que existem dentro de um fator, nós podemos usar a função `levels()`. Essa mesma função permite que os níveis do fator sejam substituídos pela atribuição de um novo vetor. Por exemplo, eu posso acessar os níveis do dentro do vetor sexo `levels(f.sexo)`, e em seguida eu posso substituir o nome dos níveis por um código numérico `levels(f.sexo)<-c("0","1")`. Perceba que para substituir os níveis de um fator, nós precisamos atribuir um vetor de caracteres.

Tabela de dados

Uma tabela de dados é *um conjunto de vetores de mesmo tamanho, mas com diferentes classes de dados*. Tabelas de dados são estruturas bidimensionais, ou seja, possuem linhas e colunas. As linhas da tabela são chamadas de *observações*, enquanto que as colunas são chamadas de *variáveis*. Diferentemente das estruturas vistas até agora, os dados não estão indexados apenas a uma posição, mas também a um *nome*. Para construir uma tabela de dados, nós podemos usar a função `data.frame()` sobre um conjunto de vetores.

Por exemplo, com os dados sobre as pessoas organizados nos vetores `alturas`, `massas` e `sexo`, eu construi uma tabela de dados `t.pessoas`.

```
alturas <- c(1.84,1.74,1.69)

massas <- c(72,95,67)

sexo <- c("M","M","F")

t.pessoas <- data.frame(alturas,massas,sexo)

t.pessoas
```

Dentro da tabela de dados, cada variável possui um nome que nós podemos acessar com a função `colnames()`. De modo similar, cada observação possui um nome que nós podemos acessar com a função `rownames()`. Essas funções também permitem substituir os nomes das colunas e das observações pela atribuição de um vetor de caracteres. Por exemplo, para acessar os nomes das variáveis da tabela de dados, eu executei `colnames(t.pessoas)`. Posteriormente, para atribuir novos nomes às variáveis, eu executei `colnames(t.pessoas) <- c("metros","kg","sexo")`. Perceba que para substituir os nomes eu tive que atribuir um vetor com caracteres. Os nomes das observações e das variáveis devem ser únicos, ou seja, sem repetição de nomes.

Visto que cada variável possui um nome único dentro da tabela de dados, nós podemos acessar as variávies pelos seus nomes. Para acessar uma variável pelo nome, nós adicionamos o operador sifrão `$` ao lado da tabela de dados e especificamos o nome da variável desejada. Por exemplo, para acessar apenas a variável `kg` na tabela de dados das pessoas, eu executei `t.pessoas$kg`. Uma vez acessada a variável, nós podemos manipular a mesma como qualquer vetor.

Lista

Uma lista é *um conjunto de qualquer tipo de estrutura de dados, independente do tamanho*. As listas são unidimensionais, logo as posições dentro das listas estão indexadas a um único número. De maneira similar a tabela de dados, as unidades de uma lista podem receber nomes, e posteriormente essas unidades podem ser acessadas pelos seus nomes. Para criar uma lista, nós podemos usar a função `list()` e adicionar dentro dos parênteses as diferentes estruturas de dados, separando as mesmas por vírgula `,`.

Por exemplo, para compilar diferentes estururas de dados sobre as pessoas, eu criei uma lista `l.pessoas`. A lista possui a matriz `m.pessoas`, o arranjo `arr.pessoas` e a tabela `t.pessoas`, cada um indexado a uma posição. A posição dentro da lista é indicada dentro dos colchetes duplos `[[]]`.

```
l.pessoas <- list(m.pessoas, arr.pessoas, t.pessoas)

l.pessoas
```

Uma vez criada a lista, nós podemos acessar as unidades da lista pelo número (posição) em que elas estão indexados. Para acessar um componente de uma lista, nós usamos o duplo colchete `[[]]` ao lado da lista e especificamos a posição de interesse. Especificado o componente da lista, nós ainda podemos acessar os dados com os colchetes simples `[]`. Por exemplo, eu posso acessar a matriz armazenada na primeira posição da lista e adicionalmente acessar a terceira coluna dessa matriz `l.familia[[1]][,3]`.

Por fim, as listas permitem que suas unidades tenham nomes. Para acessar os nomes das unidades de uma lista, nós usamos a função `names()` sobre a lista de interesse. Ainda, nós podemos renomear as unidades de uma lista ao atriburi um vetor de caracteres. Uma vez atribuídos os nomes, podemos acessar as undiades de uma lista pelos nomes, utilizando o operador sifrão `$`. Por exemplo, eu posso verificar os nomes das unidades da lista `names(l.pessoas)`; posteriormente eu posso renomear os componentes da lista `names(l.pessoas)<-c("matriz", "arranjo", "tabela")`; por último, eu posso usar os nomes atribuíudos para acessar apenas a tabela de dados dentro da lista `l.familia$tabela`.

Práticas

Francis Galton (1822-1911) foi um cientista britânico que se destacou positivamente pelas suas contribuições à estatística e genética. Mais precisamente, Galton formulou o conceito de correlação, e desenvolveu métodos de quantificar a herança de caracteríticas. Por outro lado, Galton foi o maior proponente da eugenia, uma ideia que trouxe grande atraso e dano para a humanidade.
Para o desenvolvimento do conceito de correlação, Galton utilizou um conjunto de dados sobre a altura de crianças e de seus respectivos parentais. A coluna `family` identifica a família por meio de um numeral; a coluna `father` traz as alturas dos pais das crianças, em polegadas; a coluna `mother` traz as alturas das mães das crianças, em polegadas; a coluna `gender` traz o sexo biológico das crianças; a coluna `height` traz a altura das crianças, em polegas; e por fim, a coluna `kids` traz o número de crianças na família.

Dada essas informações, faça as seguintes manipulações e avalições sobre esse conjunto de dados:

1) Primeiro, mude o diretório de trabalho para a pasta onde você armazenou o arquivo `galton_dataset.tab`. Agora carregue o conunto de dado com o comando `galton <- read.table(file="Galton_dataset.csv", header= T, sep=",")`.

2) Os dados coletados por Galton foram armazenados em um objeto chamado `galton` no nosso console do R. Acesse as primeiras linhas desse conjunto de dados com o comando `head(galton)`, e também podemos verificar a estrutura e tipos de dados do conjunto com o comando `str(galton)` (str é uma abreviação de *structure*, ou seja, estrutura).

3) O nome das colunas é pouco informativo e também não está na nossa língua nativa. Substitua o nome das colunas do conjunto de dados, de modo que: Family = familia; Father = alt_pai; Mother = alt_mae; Gender = sexo; Height = alt_crianca; Kids = n_crianca. [DICA: você vai precisar da função colnames()]

4) A escala de medida das alturas está em polegadas, o que é pouco intuítivo para nós que usamos o sistema métrico. Substitua as altura em polegadas por alturas em metros. [DICA: 1 polegada = 0,0254 metros]

5) A correlação ocorre quando duas variáveis numéricas contínuas variam juntas, sendo que 0 indica nenhuma correlação e 1 indica correlção total. Quando ambas variáveis variam na mesma direção, a correlação é positiva; quando uma variável aumenta e a outra diminui, a correlação é negativa. A correlação **não** indica causalidade entre variáveis, apenas uma associação entre as mesmas. A altura das crianças está correlacionada com as alturas dos pais e das mães? [DICA: a correlação pode ser calculada com a função `cor()`]

6) O *status* sócio-econômico de uma família pode influenciar o crescimento das crianças e consequentemente suas alturas. O número de filhos de uma família é um possível indicador desse *status* socio-econômico, onde famílias maiores possivelmente terão um status de maior vulnerabilidade social. A partir do número de crianças das famílias, crie uma nova coluna classificando o tamanho das famílias, onde: 3 ou menos crianças = pequena; de 4 a 6 crianças = media; 7 ou mais crianças = grande. Essa nova coluna deve ter o nome `tamanho_familia`.

7) Agora podemos avaliar os dados considerando o *status* sócio-econômico das famílias, representado no tamanho das famílias. Para isso, teremos que reorganizar os dados usando a classificação feita anteriormente. Separe as famílias por categoria de tamanho, atribuindo cada categoria a um novo objeto. Em seguida, armazene *apenas* os dados de altura de cada um desses objetos em matrizes, e armazene as matrizes em uma lista. Por fim, nomei as posições da lista de acordo com a categoria de tamanho.

8) Assumindo o tamanho das famílias como indicador do *status* sócio-econômico, podemos avaliar se o *status* sócio-econômico interfere com as alturas das crianças. Para cada categoria de tamanho de família, calcule a média e o desvio-padrão das altura das crianças. Existem diferenças entre famílias de categorias diferentes? [DICA: a média e o desvio-padrão podem ser calculados com as funções `mean()`e `sd()`, respectivamente.]