05

Aumentando limite e ordenando a exibição

Transcrição

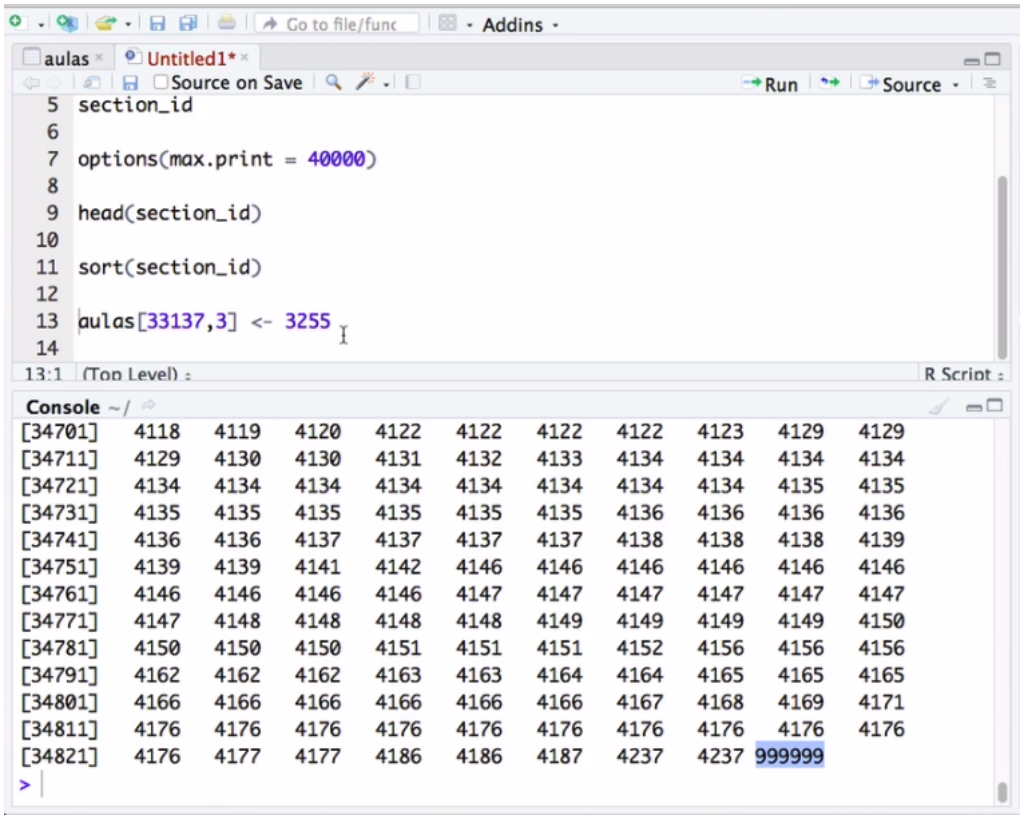
A empresa que nos contratou confirmou que `999999` foi inserido erroneamente no banco de dados. O desenvolvedor responsável nos passou o valor correto, que é `3255`. Vamos corrigir o banco de dados antes de continuarmos a análise. Precisamos especificar em qual linha e coluna o valor deve ser alterado e, para isso, manipularemos a planilha, ou **matriz**, conceito que utilizaremos por se tratar de números em linhas e colunas.

No Script, digitaremos o banco de dados (`aulas`) e, entre colchetes (`[]`), indicaremos a linha e a coluna que sofrerão alterações, nessa ordem, e separadas por vírgula (`,`). Ao informar o valor correto, o desenvolvedor responsável nos passou que o dado foi inserido na linha `33137` e coluna `3`.

O número da coluna tem a planilha do arquivo `aulas.xlsl` como referência, na qual encontramos a variável que estamos analisando (`section_id`) na terceira coluna. A substituição do valor correto será feita por meio do comando de atribuição, com o atalho "*Option* + -" (MAC) ou digitando o sinal de menor, seguido de sinal de menos, sem espaço entre eles (`<-`). Esse comando forma o desenho de uma seta. O valor à direita será atribuído à posição indicada à esquerda dela.

```
aulas[33137, 3] <- 3255
```

Ao clicarmos em "Run", o comando será executado sem problemas. O valor foi inserido no banco de dados. Em seguida, posicionaremos o cursor em `sort(section_id)`, no Script, para executá-lo e verificar se o valor foi inserido corretamente. No retorno, o valor fora do padrão, `999999`, permanece.



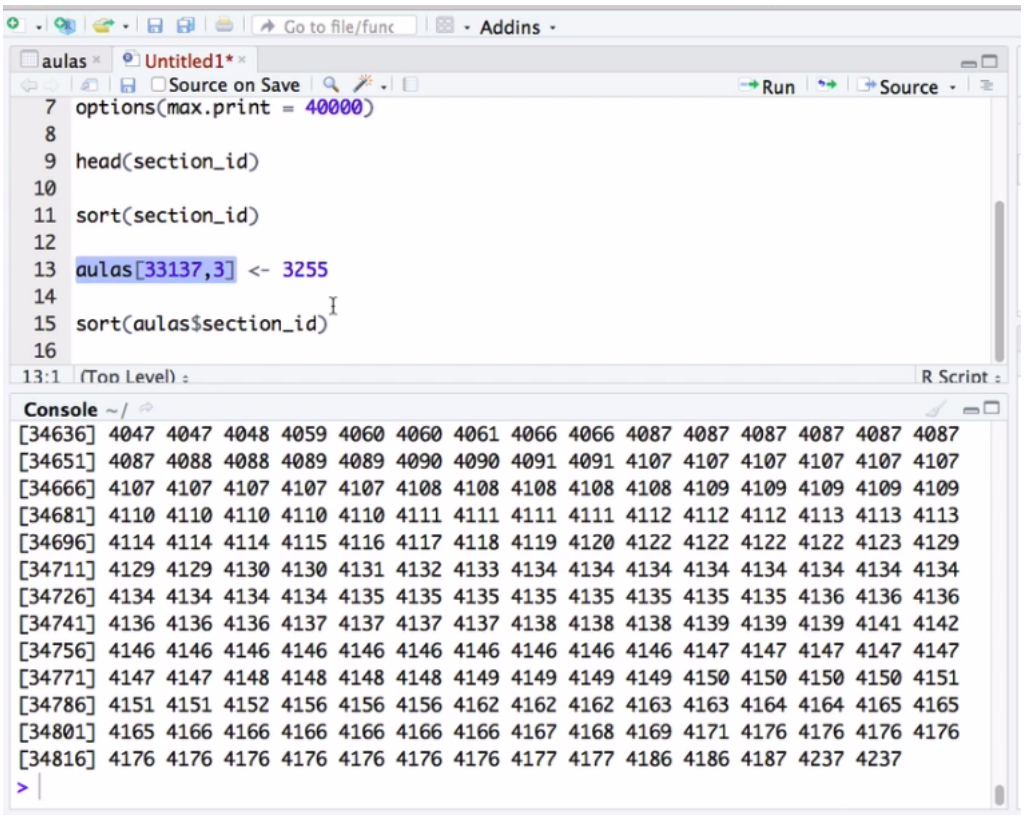
Isso acontece porque trabalhamos com a cópia da variável (`section_id`), que se encontra na memória do RStudio. Isto é, não trabalhamos diretamente com o banco de dados. Devemos prestar atenção ao especificarmos a variável, após o

uso de `attach()`. Essa função de anexação cria uma cópia da variável na memória do programa.

Quando trabalhamos com a cópia, não trabalhamos com o banco de dados, e vice-versa. Na análise, precisamos ficar atentos para manipular os objetos obtendo resultados nele, e não em outros. Nesse caso, para conseguirmos o resultado esperado, digitaremos o nome do banco de dados na função `sort`.

```
sort(aulas$section_id)
```

Funciona. No Console, o valor `999999` não aparece mais:

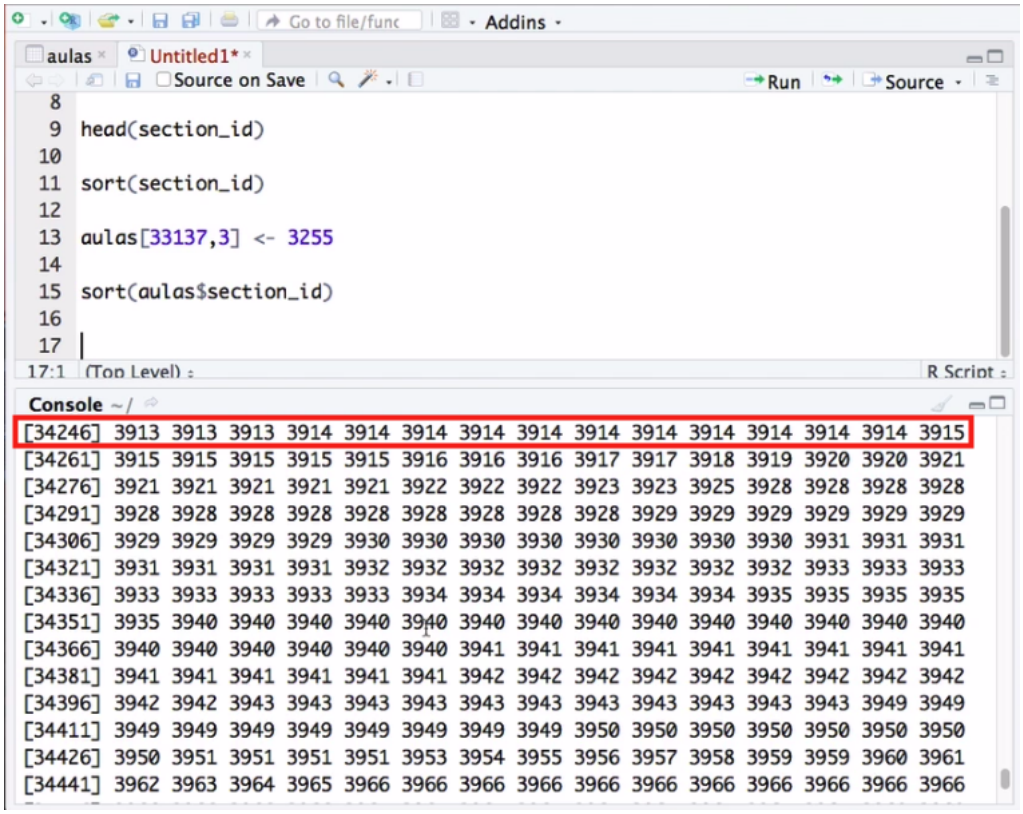


Agora, localizaremos o valor correto `3255` no banco de dados, selecionando e executando `aulas[33137, 3]`. Assim, estaremos solicitando a execução do trecho selecionado, que contém o nome do banco de dados, a linha e a coluna que estamos procurando. Ao executarmos, teremos como retorno no Console:

```
> aulas[33137, 3]
# A tibble: 1 x 1
    section_id
         <dbl>
1         3255
>
```

O valor foi inserido corretamente no banco de dados, e não em sua cópia. Podemos fazer a mesma análise com dados de cursos e alunos, mas a princípio estamos interessados nos vídeos. Trabalharemos com as outras variáveis mais adiante.

Precisaremos descobrir a quantidade de vídeos únicos, pois há uma porção de valores repetidos no Console, indicando que um mesmo vídeo foi assistido mais de uma vez.



Na primeira linha, destacada na imagem acima, por exemplo, `3914` é repetido diversas vezes. Esse comportamento é esperado, considerando que vários alunos fizeram a mesma aula ou um mesmo aluno visualizou o mesmo vídeo diversas vezes.

Se repararmos, veremos que o primeiro código de curso não segue linearmente até o último. Isso significa que na hora de codificar os cursos, a empresa pulou códigos propositalmente, ou que na amostra que estamos analisando a sequência não esteja linear, o que dificulta a análise.

Para descobrirmos a quantidade de cursos únicos, analisaremos somente uma exibição por curso. Para isso, utilizaremos a função `unique()`, que fornecerá uma exibição por curso.

```
unique(aulas$section_id)
```

No Console, após executarmos o comando, serão exibidas somente uma observação por curso. O RStudio fornece uma facilidade gráfica que nos permite saber quantos cursos existem no banco de dados. Na coluna à esquerda do Console, entre colchetes (`[ ]`), está o índice das observações exibidas. Nele, podemos ver a ordem do primeiro valor da linha.

Por exemplo, a observação `[1966]` é o vídeo `4135`, e a observação `[1981]` é o vídeo `3872`. Para sabermos quantos vídeos temos no total, contamos os códigos da linha. Por exemplo, na linha de `[1981]`, há 3 códigos: `3872`, `4237` e `4187`. Isso significa que nela, temos 3 vídeos: `[1981]`, `[1982]` e `[1983]`.

A contagem nessa linha é fácil, por conter poucos valores, mas caso isto não ocorra, fica difícil, e nos sujeitamos a erros na contagem. No RStudio, a função `length()` nos fornece diretamente o tamanho do vetor. Vamos especificá-la com `unique()`.

```
length(unique(aulas$section_id))
```

Executando a linha desse comando, teremos de retorno no Console:

```
> length(unique(aulas$section_id))
[1] 1983
```

Obtém-se o mesmo resultado da contagem anterior, de uma maneira mais prática. Essa função é útil para conseguirmos o tamanho do vetor. Notem que copiamos a função `unique()` e a inserimos dentro de outra, `length()`. Esse alinhamento, com inserção de uma função, que terá valor *input* em outra define o conceito de *nesting*, muito utilizado no RStudio.

Obtivemos a quantidade significativa de vídeos contidos na amostra, são `1983` vídeos em apenas uma delas. No banco de dados da empresa que estamos atendendo devem haver muito mais. Lembrem-se que o objetivo é analisar a popularidade dos cursos, precisamos entender o universo em que estamos trabalhando. Não é interessante para a empresa analisar todo o conteúdo do banco de dados de uma vez só, assim como analisar uma pequena parte dele.

Se esse fosse o caso, caberia a nós sugerir uma análise qualitativa para poucos vídeos. No entanto, verificamos que a quantidade é significativa, portanto, identificaremos quais são os mais e menos assistidos. Serão informações importantes, a partir das quais a empresa poderá economizar gastos, esforço e tempo de trabalho.