Camila Bassi Fernandes da Silva

Sou bióloga, formada pelo Instituto de Biociências da USP e, atualmente, sou mestranda em Genética no IB, aluna da Profa. Dra. Maria Rita Passos Bueno do grupo de pesquisa de Fissura labiopalatina. Junto com o mestrado, exerço a profissão de Perita Criminal no Instituto de Criminalística de São Paulo, onde trabalho de acordo com uma escala de plantão, o que possibilita a conciliação das duas atividades.

I. Meus exercícios

Tabela com acesso aos meus exercícios resolvidos: Meus exercícios

II. Trabalho final

a) Proposta 1: Montagem de escala de plantões

Contextualização

Quando se trabalha seguindo uma escala de plantão, em que não há um horário de trabalho fixo, podendo variar entre os dias da semana e períodos de turnos, é necessária a montagem de uma escala, que modifica mensalmente de acordo com alguns fatores como: número de dias por mês e distribuição igualitária de turnos (diurno e noturno). Além disso, uma escala de plantões deve respeitar um descanso mínimo de 36 horas de um plantão ao outro. Tais fatores são mais dificultados ainda, quando esses funcionários também se revezam para trabalhar em duas unidades distintas e não devem realizar menos que 12 plantões ao mês, podendo ocorrer dupla de funcionários em alguns turnos (normalmente noturno).

Diante de tais dificuldades, que são enfrentadas mensalmente, a criação de uma função pouparia esse trabalho que, atualmente, é feito manualmente. Dessa forma, a proposta é desenvolver uma função que gere uma escala de funcionários com descanso mínimo de 36 horas, distribuição igualitária de turnos (diurno e noturno) e de unidades (unidades 1 e 2), e inclusão de duplas em alguns noturnos, variando de acordo com o número de dias no mês.

Planejamento da função

Entrada: escala (`ferias`, `ano`, `mes`, `dir`, `arquivo`)

- `ferias` = indicar quem sairá de férias ou licença prêmio (de 1 a 11)
- `ano` = ano desejado para montar a escala (2018 a 2020)
- `mes` = mês desejado para montar a escala (de 1 a 12)
- `dir` = diretório em que será salvo o arquivo
- `arquivo` = nome do arquivo em pdf (default=escala.pdf)

Verificando os parâmetros:

- `ferias` é um número inteiro de 1 a 11? Se não, escreve: “ferias precisa ser um numero inteiro entre 1 e 11.”
- `ano` é igual a 2018, 2019 ou 2020? Se não, escreve: “ano precisa ser igual a 2018, 2019 ou 2020.”
- `mes` é um número inteiro de 1 a 12? Se não, escreve: “mes precisa ser um numero inteiro entre 1 e 12.”
- `arquivo` contém caracter .pdf? Se não, escreve: “arquivo precisa ser extensão .pdf”

Pseudo-código:

1. Criação de vetor contendo números de 1 a 11 representando os funcionários, em que:
    1. Uma pessoa será removida mensalmente, caso esteja de férias ou de licença prêmio;
    2. Apenas uma pessoa poderá sair de férias para cada mês;
    3. Todo mês deve ter uma pessoa de férias;
    4. O funcionário que estará de férias será indicado pelo usuário no argumento `ferias`.
2. Criação de data frame contendo dia, mês, ano e dias da semana de 01/05/2018 a 30/04/2020.
3. Escolha de ano e mês no data frame para montagem de escala feita pelo usuário nos argumentos `ano` e `mes`, respectivamente.
4. Criação de uma matriz com 5 colunas representando *unidade 1 dia*, *unidade 2 dia*, *unidade 1 noite.2*, *unidade 2 noite* e *unidade 1 noite*.
5. Número de linhas da matriz definido pelo ano e mês gerados no data frame, após a escolha do usuário. O número de linhas representará o número de dias no mês.
6. A coluna *unidade 1 noite.2* poderá aparecer ou não na matriz de acordo com o número de dias no mês/número de linhas da matriz, com as seguintes condições:
7. A coluna permanecerá se o número de dias do data frame escolhido multiplicado pelo número de turnos (2 turnos) e pelo número de unidades (2 unidades) for diferente do número de funcionários ativos (10 funcionários ativos) multiplicado pelo número mínimo de plantões por mês (12 plantões por mês), ou seja, (4*número de dias ≠ 120);
8. Caso seja igual, a coluna será removida.
9. Uma vez que a coluna permaneça, serão preenchidas apenas as linhas 8, 9, 14, 15, 20, 21, 26 e 27 desta coluna, representando as formações de duplas.
10. Após os passos acima, a estrutura da matriz estará pronta para iniciar a montagem.
11. A montagem da matriz ocorrerá da seguinte forma:
    1. Preencher a matriz com o vetor de funcionarios ativos com as seguintes condições:
        1. Se forem 28 dias (ou linhas): seguir a sequência normalmente;
        2. Se forem 29 dias: iniciar a sequência, não preencher os dias 8, 9, 14 e 15 da coluna *unidade 1 noite.2* e seguir a sequência;
        3. Se forem 30 dias: seguir a sequência normalmente;
        4. Se forem 31 dias: iniciar a sequência, não preencher os dias 8 e 9 da coluna *unidade 1 noite.2* e seguir a sequência.
    2. Após o preenchimento, inverter a ordem das colunas para *unidade 1 dia*, *unidade 1 noite*, *unidade 1 noite.2* (se estiver presente), *unidade 2 dia* e *unidade 2 noite*.
    3. Posteriormente, inverter as colunas *unidade 1 dia+unidade 1 noite* com *unidade 2 dia +unidade 2 noite* apenas nas seguintes linhas:
        1. 6 a 10 para todos, exceto para 28 dias, que será de 6 a 9;
        2. 16 a 20 para todos;
        3. 26 a último dia do mês para todos, exceto para 29 dias, que será de 28 e 29.
12. Unir a matriz gerada com as informações de dia do mês e dia da semana do data frame.

13. Salvar o arquivo no formato tabela em pdf, contendo as colunas com unidades/turnos e as linhas com os dias e os dias da semana com a distribuição final dos funcionários.

Saída:

- Arquivo no formato pdf que conterá a escala criada em que:
  - Se for de 28 dias: 12 plantões com a distribuição de no mínimo 5 e no máximo 7 plantões para cada turno/unidade;
  - Se for de 29 dias: 12 plantões com a distribuição de no mínimo 5 e no máximo 7 plantões para cada turno/unidade;
  - Se for de 30 dias: 12 plantões com a distribuição de 6 plantões para cada turno/unidade;
  - Se for de 31 dias: 13 plantões com a distribuição de no mínimo 5 e no máximo 8 plantões para cada turno/unidade.

Comentários gerais: Sua proposta está clara e bem organizada. Ela se propõe a resolver um problema relevante e acho que ela poderia ser bem útil. Seu algoritmo já está bem desenvolvido, então você pode começar a pensar em como tornar a função mais geral. Em outras palavras: ela está específica para um determinado contexto e seria mais interessante dar mais liberdade para o usuário especificar as características do seu local de trabalho.

Para pensar:

- Por que a função só considera 11 funcionários? Seria bacana torná-la mais geral para que ela possa ser empregada em outros contextos. Como você faria para que a função considere um número variável de funcionários?
- Seguindo o mesmo questionamento do último ponto levantado, a premissa de que “todo mês deve ter uma pessoa de férias” é específica de um contexto. Por que não torná-la mais geral também?
- A função se limita ao período de 01/05/2018 a 30/04/2020. Vamos tentar romper essa premissa também? Eu pesquisei um pouco e acho que, apesar de não ser simples, é possível criar o dataframe para qualquer mês e ano. Além disso, do jeito que está colocado, toda vez que a função for rodada, será necessário criar primeiro o calendário completo para depois extrair o mês de interesse. Acho que isso pode ser otimizado também.
- Bônus (só para pensar mesmo): como você faria para a sua função levar em consideração as restrições de horário de cada funcionário?

Sugestões:

- Ao invés de usar números para representar os funcionários, você poderia ter um argumento na sua função com um arquivo que tenha uma lista dos nomes.
- Você colocou o argumento “arquivo” para o usuário especificar o nome que ele deseja usar. Caso não seja especificado, o nome padrão é “escala.pdf”. Eu imagino que você tenha feito isso para o usuário especificar o mês e o ano para o qual a escala está sendo montada. Se for isso mesmo, existe como salvar o arquivo com um nome que inclui uma parte padrão (“escala_”) e outra variável (“10.2018”). Dessa forma, você evitaria que o usuário insira o nome manualmente e garantiria que os arquivos gerados pela função tenham sempre o mesmo padrão de nome.
- Os critérios para distribuição dos funcionários nos plantões são novamente específicos (e.g.: número mínimo de plantões por mês e unidades). Todos esses números poderiam entrar como argumentos da função para torná-la mais abrangente.

- Por que você optou pelo arquivo no formato .pdf? Não é tão simples criar um dataframe com esse formato no R. Existem outros mais simples e talvez até mais úteis por permitirem que o usuário faça modificações a posteriori.

b) Proposta 2: Jogo Sudoku

Contextualização

Sudoku é um jogo de raciocínio em que o objetivo é preencher os números de 1 a 9 em cada espaço vazio de uma matriz 9×9, de forma que o número não se repita na mesma linha, coluna ou blocos de números (3×3). Assim como qualquer outro jogo, o Sudoku também apresenta diferentes níveis de dificuldade, que será determinado pelo número de espaços vazios. Quanto maior a dificuldade, mais espaços vazios deverão ser preenchidos.

Muitas pessoas tem o hábito de comprar revistas com esses jogos em bancas de jornal constantemente. Esses gastos poderiam ser economizados se fosse possível gerar esses jogos e imprimir do seu próprio computador. Dessa forma, a proposta é criar uma função que irá gerar diferentes sudokus de níveis variados (fácil, médio ou difícil) para as pessoas imprimirem, sem precisar sair de casa.

Planejamento da função

Entrada: sudoku (nivel, resposta, dir, arquivo)

- `nivel` = nível de dificuldade (1 = Fácil, 2 = Médio, 3 = Difícil)
- `resposta` = Jogo acompanhado da resposta (sim) ou não (nao)
- `dir` = diretório em que será salvo o arquivo
- `arquivo` = nome do arquivo em jpeg (default=sudoku.jpeg)

Verificando os parâmetros:

- `nivel` é igual a 1, 2 ou 3? Se não, escreve: “nivel precisa ser igual a 1 (facil), 2 (medio) ou 3 (dificil).”
- `resposta` é um caracter sim ou nao? Se não, escreve: “resposta precisa ser igual a sim ou nao.”
- `arquivo` contém caracter .jpeg? Se não, escreve: “arquivo precisa ser extensão .jpeg.”

Pseudo-código:

1. Criação de matriz 9×9 com NA.
2. Para cada célula da matriz:
    1. Criação de vetor de números de 1 a 9;
    2. Substituir números deste vetor que já existam na mesma linha da célula por NA;
    3. Substituir números deste vetor que já existam na mesma coluna da célula por NA;
    4. Substituir números deste vetor que já existam no bloco 3×3 ao qual esta célula pertence por NA;

    5. Eliminar os NA's gerados no vetor de 1 a 9;
    6. Selecionar amostra aleatória dos números que restaram no vetor.
3. Substituir aleatoriamente números por espaços dependendo do nível de dificuldade selecionado pelo usuário no argumento `nivel`, em que:
    1. Nível 1 (Fácil) - Substituição de 36 a 42 números por espaços;
    2. Nível 2 (Médio) - Substituição de 43 a 49 números por espaços;
    3. Nível 3 (Difícil) - Substituição de 50 a 56 números por espaços.
4. A matriz criada será salva no formato de imagem jpeg acompanhada da resposta, caso o usuário desejar no argumento `resposta`.

Saída:

- Arquivo no formato jpeg que conterá o sudoku criado com ou sem a resposta

Comentários gerais: Eu gostei da ideia, mas acho que se for seguir adiante com esse plano, seria necessário pensar em outras ações para a função realizar. Seu principal desafio vai ser pensar no pseudo-código para gerar diferentes versões do jogo. Eu fiz uma pesquisa rápida e vi que é possível criar um jogo de sudoku em apenas algumas linhas de comando. Acho que você pode fazer essa função como um desafio pessoal, mas me parece que seria pouco para a proposta da disciplina.

Para pensar:

- Por que a função tem o argumento “arquivo”?
- A função só gera um tipo de jogo de sudoku. Como você faria para ela gerar matrizes de jogos diferentes cada vez que é rodada?
- Inicialmente, a matriz vai estar vazia. Sendo assim, não vai ter nenhum número na linha/coluna/bloco cujo correspondente no vetor seria substituído. O passo 2 do pseudo-código precisa ser repensado.

Sugestões:

- Que tal o usuário definir o tamanho do sudoku? Ele não precisar ser sempre 9×9.
- Além de gerar jogos, a função poderia também solucionar jogos. Para isso, você poderia ter um argumento do tipo caractere para o usuário especificar qual ação ele deseja realizar.
- Por que você optou pelo arquivo no formato .pdf? Não é tão simples criar um dataframe com esse formato no R.

Conclusão: Camila, eu gostei bastante das suas propostas. Apesar de ter me empolgado com o sudoku, acho que você deve investir no Plano A. No momento que você matar o algoritmo para gerar o jogo, a função vai ficar muito simples. Acho que vai ser um desafio interessante você se propor a tornar o Plano A mais generalizado. Se quiser conversar, pode me mandar um e-mail (carolina.mendonca@usp.br) e a gente combina um horário. Bom trabalho!

Carol Mendonça

III. Links para o trabalho final

Para o trabalho final, resolvi seguir com o plano A, seguindo as recomendações da Carol. A minha função se chama escala e diferentemente do que propus anteriormente, ela funciona para um número variável de funcionários, sendo mais geral, mas servindo para apenas uma unidade de trabalho, cumprindo os pré requisitos de descanso de no mínimo 36 horas e com distribuição do mesmo número de plantões para cada funcionário e aproximadamente igualitária de plantões diurnos e noturnos.

- Link para a página da minha função: Escala
- Link para a página de ajuda da minha função: Help

IV. Limitações da função

A função executa por meio de um sorteio de dias, podendo gerar escala que não respeita descanso ou então não executa na primeira tentativa. Dessa forma, caso isso aconteça, recomenda-se que a função seja executada novamente. A função também funciona para 6 e 8 funcionários, mas ao longo dos testes realizados, foram verificados com alta frequência diversos erros na distribuição de plantões e de descanso. Entretanto, não houve tempo hábil para corrigir tais limitações, uma vez que necessitei de um tempo para reestruturar a proposta inicial e tornar a função mais geral.

From:
http://ecor.ib.usp.br/ - **ecoR**

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2018:alunos:trabalho_final:camila.bassi.silva:start

Last update: **2020/08/12 06:04**