Juliana Borsoi

Aluna de doutorado do Departamento de Genética e Biologia Evolutiva do IB-USP, sob orientação da profa Dra Lygia da Veiga Pereira.

O objetivo do meu projeto é investigar os mecanismos moleculares dos fenótipos cardiovasculares da Síndrome de Marfan, uma doença monogênica causada por mutações na proteína de matriz extracelular fibrilina-1. Também tenho interese em estudar o papel das fibrilinas durante o desenvolvimento do sistema cardiovascular humano, analisando a dinâmica de expressão de seus respectivos genes.

Meus Exercícios

Link para os meus exercícios resolvidos: Meus Exercícios

Trabalho Final

1. Plano A - Ferramenta para ajuste da gravidade original (OG) e cálculo do índice de amargor (IBU) na produção caseira de cerveja

1.1. Contextualização

A produção de cerveja normalmente envolve cinco etapas básicas: a brassagem, a fervura, a fermentação, a maturação e a carbonatação/refermentação. Durante a etapa de brassagem, os grãos de malte moído são cozidos em água a uma temperatura específica (normalmente entre 60-70 °C) que propicie a atuação ótima de enzimas que degradam o amido presente nos grãos, convertendo-os em açúcares menores. O extrato obtido ao final desta etapa é chamado de *mosto* e contém os açúcares a serem utilizados pela levedura na produção de álcoois. A etapa de fervura consiste no cozimento do mosto a 100 °C. Durante esta fase, ocorre sua esterilização e concentração, além da caramelização dos açúcares e da evaporação de substâncias indesejáveis. Um outro evento muito importante que também ocorre normalmente durante a fervura é a lupulagem, que consiste na

extração dos sabores e aromas da chamada “alma da cerveja”, o lúpulo, que fornece o amargor característico da bebida devido à isomerização de seus alfa-ácidos em altas temperaturas. Ao final deste processo, o mosto, agora mais complexo, é resfriado e sofre a inoculação de leveduras que serão responsáveis pela etapa seguinte, a fermentação, em que os açúcares são convertidos em álcoois (principalmente o etanol) e CO2.

A concentração de açúcares presente no mosto após a fervura vai determinar o grau alcoólico do produto final e também o chamado “corpo” da cerveja. A densidade do mosto pode e deve ser medida durante várias das etapas de produção, sendo as mais importantes após a fervura, e após a fermentação. Mais comumente entre os cervejeiros caseiros, mede-se a densidade do mosto em relação à densidade da água, chamada de gravidade específica (SG) ou densidade específica. Este valor, sem unidade, é obtido utilizando-se instrumentos de medição como o densímetro. O valor de densidade específica obtido após a fervura é chamado gravidade original (OG) e o obtido após a fermentação é chamado gravidade final (FG), e variam dependendo do tipo de cerveja. A relação entre a OG e a FG é utilizada para o cálculo do “álcool por volume”, ou ABV, que é a concentração alcoólica final da cerveja, segundo a fórmula:

$$\%ABV = 131{,}25 * (OG - FG)$$

Desta forma, percebe-se a importância de se atingir o valor da OG desejável para o tipo de cerveja produzido, visto que dele dependem as características finais da bebida. Durante a produção, pode-se corrigir a gravidade específica aumentando-se o tempo de fervura, de forma a tornar o mosto mais denso (quando a OG está baixa) ou adicionando-se mais água após a fervura, diluindo-se o mosto (quando a OG está alta).

1.2. Descrição

A função irá calcular o volume de água (em litros) ou o tempo adicional de fervura necessário para se corrigir a gravidade específica (SG) obtida, de forma a se alcançar a gravidade original (OG) desejada. De modo opcional, a função também calculará o índice de amargor (IBU) da bebida após a fervura, utilizando informações a respeito do(s) lúpulo(s) utilizado(s) fornecidas pelo usuário.

1.3. Planejamento da função

Entrada: brewtool (SG, OG, `t.mosto`, `t.cal`, `vol`, `tempo.f`, IBU, `input`)

- SG = gravidade específica pós-fervura (OG) obtida pelo usuário por medida com densímetro (classe: numeric, com 3 casas decimais, 1.020 < SG < 1.120)
- OG = gravidade original desejada (classe: numeric, com 3 casas decimais, 1.020 < OG < 1.120)
- `t.mosto` = temperatura do mosto, em graus Celsius, no momento da medida de SG (classe: numeric, 0 < `t.mosto` < 85)
- `t.cal` = temperatura de calibração do densímetro utilizado, em graus Celsius (classe = numeric, `t.cal` > 0, default = 20)
- `vol` = vetor contendo os volumes do mosto antes e após a fervura, em litros (classe: numeric, `vol[i]` > 0)
- `tempo.f` = tempo de fervura, em minutos (classe: numeric, `tempo.f` > 0, default = 60)
- IBU = opção para o cálculo do índice de amargor (IBU) (classe: logic, default = FALSE)
- `input` = tabela de dados contendo as informações - peso (em gramas), tempo (em minutos

pós início da fervura) e % de alfa-ácidos - do(s) lúpulo(s) utilizado(s), necessária apenas se IBU = TRUE (classe = data frame)

Verificação dos parâmetros:

- SG maior que 1.020 e menor que 1.120? Se não, stop("SG fora do intervalo permitido, 1.020 < SG < 1.120")
- OG maior que 1.020 e menor que 1.120? Se não, stop("OG fora do intervalo permitido, 1.020 < OG < 1.120")
- SG é um número com 3 casas decimais? Se não, warning("SG deve ser um número com 3 casas decimais. O valor digitado foi ajustado para 3 casas.")
- OG um número com 3 casas decimais? Se não, warning("OG deve ser um número com 3 casas decimais. O valor digitado foi ajustado para 3 casas.")
- `t.mosto` é um número inteiro entre 0 e 85? Se não, stop("t.mosto deve ser um número inteiro entre 0 e 85")
- `t.cal` é um número inteiro maior que 0? Se não, stop("t.cal deve ser um número inteiro > 0")
- `vol[i]` é um número maior que 0? Se não, stop("vol deve conter apenas números > 0")
- `vol[1]` menor que `vol[2]`? Se sim, warning("Volume pré-fervura menor que o volume pós-fervura. A taxa de evaporação não poderá ser calculada. Verifique a ordem dos elementos do vetor (vol = c(pré, pós)).")
- `tempo.f` é um número inteiro maior que 0? Se não, stop("tempo.f deve ser um número inteiro > 0")
- Se IBU = TRUE, `input` é um data frame contendo colunas de mesmo tamanho nomeadas "peso", "tempo", e "aa"? Se não, stop(paste("As informações de", input, "não estão corretas. Verifique a nomenclatura e o tamanho das colunas".))

Pseudo-código:

1. Modificar SG formatando seu valor para 3 casas decimais
2. Modificar OG formatando seu valor para 3 casas decimais
3. Criar o objeto `SG.cal` corrigindo a SG pela temperatura de calibração (`t.cal`)
4. Criar um objeto `SG.dig` com os 2 últimos dígitos de `SG.cal`
5. Criar um objeto `OG.dig` com os 2 últimos dígitos de OG
6. Criar um objeto `SG.pre` com o cálculo do valor de SG pré-fervura
7. Criar o objeto `dif` subtraindo SG de OG –> `dif` = OG - `SG.cal`
8. Teste condicional para `dif`
    1. Se (if) `dif` = 0
        1. Criar o objeto `print.og` = print(paste("Sua OG", (`SG.cal`), "já é igual à OG desejada, nada precisa ser feito. Prossiga com a receita.", \n))
        2. Se (if) IBU = TRUE
            1. Criar o data frame `a.acidos`, contendo as informações de utilização dos alfa-ácidos para cada valor de SG pré-fervura de acordo com o tempo de fervura;
            2. Criar o objeto `IBU.tot` = rep(NA, nrow(`input`))
            3. Criar um ciclo for, com contador i = 1:length(`IBU.tot`) para calcular o IBU de cada lúpulo
                1. Criar o objeto U, contendo a % de utilização de alfa-ácidos para `SG.pre` correspondente ao `tempo.f` - `input$tempo[i]`, de acordo com o data frame `a.acidos`
                2. Calcular o valor de IBU para o lúpulo [i] utilizando as informações de `a.acidos` e armazenar seu valor em `IBU.tot[i]`
            4. Criar o objeto `print.ibu` = print(paste("Seu IBU é igual a", sum(`IBU.tot`,), "."))

3. Retornar `print.og` e `print.ibu`

2. Se (else if) `dif` < 0
    1. Criar o objeto `v.final` com o cálculo do volume final de mosto necessário para que `SG.cal` se iguale a `OG` –> `v.final` = (`SG.dig`*`vol[2]`/`OG.dig`)
    2. Criar o objeto `v.adic` com o cálculo do volume a ser adicionado para o ajuste
    3. Criar o objeto `print.og` = print(paste("Sua OG é igual a", `SG.cal`, ". Para ajustá-la para", `OG`, "adicione", `v.adic`, "litros de água esterilizada.", \n))
    4. Se (if) IBU = TRUE
        1. Criar o data frame `a.acidos`, contendo as informações de utilização dos alfa-ácidos para cada valor de SG pré-fervura de acordo com o tempo de fervura;
        2. Criar o objeto `IBU.pre` = rep(NA, nrow(`input`))
        3. Criar o objeto `IBU.pos` = rep(NA, nrow(`input`))
        4. Criar um ciclo for, com contador i = 1:length(`IBU.pre`) para calcular o IBU de cada lúpulo
            1. Calcular o valor de IBU para o lúpulo [i] utilizando as informações de `a.acidos` e usando o volume atual (`vol[2]`). Armazenar seu valor em `IBU.pre[i]`
            2. Calcular o valor de IBU para o lúpulo [i] utilizando as informações de `a.acidos` e usando o novo volume (`v.final`). Armazenar seu valor em `IBU.pos[i]`
        5. Criar o objeto `print.ibu` = print(paste("Seu IBU atual é igual a", sum(`IBU.pre`,), "e seu IBU após o ajuste da OG será igual a", sum(`IBU.pos`), "."))
    5. Retornar `print.og` e `print.ibu`
3. Se (else if) `dif` > 0
    1. Se (if) `vol[1]` ⇐ `vol[2]`
        1. Criar o objeto `print.og` = print(paste("Sua OG é igual a", `SG.cal`, ". Como não foi possível calcular a taxa de evaporação, não será possível ajustá-la para", `OG`, ".", \n))
    2. Se `vol[1]` > `vol[2]` (else)
        1. Criar o objeto `evap` com o cálculo da taxa de evaporação de líquido por minuto de fervura –> `evap` = `abs`(diff(`vol`))/`tempo.f`
        2. Criar o objeto `v.final` com o volume final em que `SG.cal` é igual a `OG`
        3. Criar o objeto `v.evap` com o cálculo do volume necessário de evaporação –> `v.evap` = `vol[2]` - `v.final`
        4. Criar o objeto `tempo` com o cálculo do tempo necessário para evaporação de `v.evap` –> `tempo` = `v.evap`/`evap`
        5. Criar o objeto `print.og` = print(paste("Sua OG é igual a", `SG.cal`, ". Para ajustá-la para", `OG`, "ferva o mosto por mais", `tempo`, "minutos."))
    3. Se (if) IBU = TRUE
        1. Criar o data frame `a.acidos`, contendo as informações de utilização dos alfa-ácidos para cada valor de SG pré-fervura de acordo com o tempo de fervura;
        2. Criar o objeto `IBU.pre` = rep(NA, nrow(`input`))
        3. Criar o objeto `IBU.pos` = rep(NA, nrow(`input`))
        4. Criar um ciclo for, com contador i = 1:length(`IBU.pre`) para calcular o IBU de cada lúpulo
            1. Criar o objeto U, contendo a % de utilização de alfa-ácidos para `SG.pre` correspondente ao `tempo.f` - `input$tempo[i]`, de acordo com o data frame `a.acidos`
            2. Calcular o valor de IBU para o lúpulo [i] utilizando as informações de

`a.acidos` e usando o volume atual (`vol[2]`). Armazenar seu valor em `IBU.pre[i]`

3. Se (if) `v.final` = TRUE
    1. Criar o objeto U, contendo a % de utilização de alfa-ácidos para `SG.pre` correspondente ao `tempo.f - input$tempo[i] + tempo`, de acordo com o data frame `a.acidos`
    2. Calcular o valor de IBU para o lúpulo [i] utilizando as informações de `a.acidos` e usando o novo volume (`v.final`). Armazenar seu valor em `IBU.pos[i]`
    3. Criar o objeto `print.ibu.pos` = print(paste("Seu IBU após o ajuste da OG será igual a", sum(`IBU.pos`), "."")
4. Retornar `print.og`, `print.ibu` e `print.ibu.pos`

Saída:

- Valor da SG ajustada pela temperatura (OG do usuário –> `SG.cal`)
- Volume de água a ser adicionado para se corrigir `SG.cal` para `OG` (`v.adic`), se `SG.cal` > `OG`
- Tempo adicional de fervura (`tempo`), se `SG.cal` < `OG`
- Valores de IBU antes e após a correção de `SG.cal` para a `OG`.

Julia Barreto

Oi, Juliana, tudo bem? Achei o tema da proposta muito massa mas essa ideia é muito simples já que apenas executa uma equação. Vamos tentar trabalhar possibilidades que complementem a ideia de forma que fique com uma cara de função. Procura reacessar as atividades da última aula, a função deve ser recorrente, útil, inédita e não trivial.

Entendo pouco da produção de cerveja mas vou tentar sugerir possibilidades. Uma seria a gente pensar numa demanda maior, tipo uma certa cervejaria interessada em produzir diferentes tipos de cerveja para um festival. Dessa forma, poderia entrar com um data-frame que contivesse diferentes valores de SG e OG para mais de uma cerveja produzida simultaneamente. Outra ideia seria também pensar na produção de diferentes tipos de cerveja amargor diferente. Tentando incluir mais variáveis que podem ser calculadas juntas ou uma em conseguencia da outra.

Me escreve se quiser conversar sobre essas ou outras ideias que venha a ter. Estou à disposição Julia Barreto

Oi Juliana,

Concordo com a Júlia. A ideéia é muito legal e vc. fez um bom trabalho na descrição e no pseudo-código. Já há algum controle de fluxo previsto com `if`, mas ficaria muito legal se incluisse alguma complexidade que necessitasse de um `for`. Uma possibilidade e ir pelo caminho sugerido pela Julia, criando a possibilidade do usuário poder definir mais de uma meta de OG desejada. Dessa forma, o usuário poderia tomar a decisão de qual caminho tomar (quanto de agua ou tempo de fervura) para diferentes metas de OG. Colocando isso em um loop, o desafio de programação ficaria mais completo. Avalie se faz sentido no contexto da sua ideia...

Juliana

Olá! Muito obrigada pelas sugestões! Eu incluí o cálculo do amargor como sugerido pela Julia e acho que a função ficou bem mais interessante e completa. Espero que esteja ok. :)

Referências

Blog Homini Lúpulo: https://www.hominilupulo.com.br/

2. Plano B - Ferramenta para cálculo de número de células e ajuste de concentração

2.1. Contextualização

Em pesquisas em que a cultura de células é necessária, a contagem destas é crucial para a equiparação dos parâmetros iniciais do experimento, de forma a se poder comparar diferentes condições e diferentes eventos experimentais. Existem diferentes equipamentos, de diversas marcas, que efetuam a contagem automática de células, porém, o instrumento ainda mais utilizado em grande parte dos laboratórios é o hemocitômetro, muito conhecido como câmara de Neubauer. Esta câmara contém uma malha de leitura, que consiste em um desenho de linhas igualmente espaçadas, organizadas em quadrantes e com dimensões conhecidas, como ilustrado abaixo:

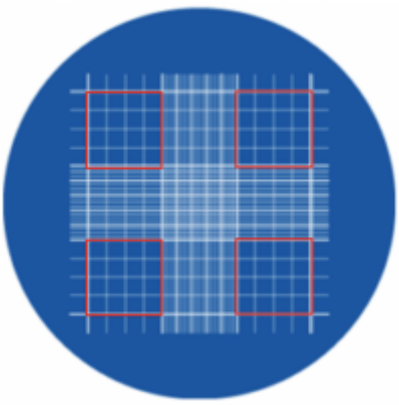

Figura 1. Esquema demonstrando a malha de leitura da câmara de Neubauer. Os quadrantes laterais, marcados em vermelho, são os mais comumente utilizados para contagem de células grandes. (Modificado de https://kasvi.com.br/como-e-realizada-contagem-de-celulas/)

Quando uma solução de células em suspensão é aplicada à câmara, estas se espalham em camada única, distribuindo-se sobre as linhas. As células dispostas nos quadrantes de medida 4×4 (em vermelho na figura acima) são então contadas e a média, calculada. Este valor pode então ser multiplicado pelo fator 10.000, que foi obtido considerando-se a área do quadrante e sua profundidade na câmara, obtendo-se assim, a concentração de células por mililitro (ml). Caso a solução aplicada tenha sido diluída, o valor obtido deve ser multiplicado pela diluição. Desta forma, a fórmula para o cálculo pode ser resumida em:

$$\frac{nº\ de\ células}{nº\ de\ quadrantes\ contados} * 10^4 * diluição$$

Obtendo-se este valor, é possível então calcular o volume necessário para se diluir a solução à concentração de células desejada (final) para o experimento:

$$Conc.inicial * Vol.inicial = Conc.final * Vol.final$$

$$Vol.final - Vol.inicial = Vol.necessário$$

2.2. Descrição

A função irá calcular a concentração da solução de células a partir da contagem obtida pelo usuário na câmara de Neubauer e também o volume necessário a ser adicionado à solução para ajuste desta concentração a uma concentração específica, também fornecida pelo usuário.

2.3. Planejamento da função

Entrada: neubauer (q, `dil`, `vol`, `conc.f`)

- q = vetor de tamanho n contendo os valores obtidos de contagem para n quadrantes da câmara de Neubauer (classe = numeric, q = (c(q1 ... qn)), sendo cada elemento >= 0)
- `dil` = fator de diluição (classe = numeric, `dil` > 0, default = 1)
- `vol` = volume da solução de células, em ml (classe = numeric, `vol` > 0)
- `conc.f` = concentração final de células desejada por ml (classe = numeric, `conc.f` > 0)

Verificação dos parâmetros:

- q é um vetor contendo apenas números inteiros maiores ou iguais a 0? Se não, stop("Os valores de q devem ser números inteiros >= 0.")
- `dil` é um número inteiro maior que 0? Se não, stop("O fator de diluição dil deve ser um número inteiro > 0.")
- `vol` é um número maior que 0? Se não, stop("vol deve ser um número > 0.")
- `conc.f` é um número maior que 0? Se não, stop("conc.f deve ser um número > 0.")

Pseudo-código:

1. Criar o objeto `tam.q` contendo o tamanho do vetor q
2. Criar o objeto `soma.q` contendo a soma dos valores de q
3. Criar o objeto `conc.i` contendo a concentração inicial de células
4. Criar o objeto `total` contendo a quantidade total de células na solução
5. Criar o objeto `vol.adic` contendo o volume necessário a ser adicionado para o ajuste de `conc.i` a `conc.f`
6. Retornar os valores calculados para o usuário:
    1. `conc.i` –> print(paste("A concentração da sua solução é de", `conc.i`, "células/ml."))
    2. `total` –> print(paste("O total de células na sua solução é", `total`,"."))
    3. `vol.adic` –> print(paste("O volume a ser adicionado à solução para a concentração de", `conc.f`, "células/ml é de", `vol.adic`, "ml."))

Saída:

- Concentração de células na solução (`conc.i`)
- Total células na solução (`total`)
- Volume a ser adicionado para se atingir a concentração final (`vol.adic`)

> Julia Barreto
> Oi de novo! Acho que essa proposta também ficou muito simples, teria que ser melhor trabahada e complementada para que se justificasse uma função. Como a A está mais completa e me parece ter mais recursos para caprichar, vamos deixar essa aqui de lado por ora. Julia Barreto

---

Links para o Trabalho Final

Como sugerido, decidi prosseguir com o plano A, adicionando o cálculo do índice de amargor (IBU) à função `brewtool`. Como explicado anteriormente, a função tem como objetivo auxiliar produtores caseiros no preparo da cerveja artesanal, indicando o valor de gravidade original (OG) obtido e as ações a serem tomadas para sua correção, quando necessário. De modo opcional, a função também calcula o IBU da cerveja antes e depois do ajuste da OG.

- Link para a página da função: Brewtool.
- Link para a página de ajuda da função: Help.

Arquivos da função

brewtool

help

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

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2019:alunos:trabalho_final:juliana-borsoi:start

Last update: **2020/09/23 17:15**