João Callefe

Mestrando em Ciência Animal no programa de Epidemiologia Experimental Aplicada às Zoonoses (FMVZ-USP).

O título da minha dissertação de mestrado é “Sistemas de vigilância em Saúde Animal: proposta para brucelose e tuberculose bovinas”, orientado pelo Prof. Dr. José Soares Ferreira Neto.

Meus exercícios

Exercícios

Proposta de trabalho final

Proposta A

Contextualização

Muitos bancos e planilhas de dados gerados a partir da coleta de informações de interesse podem apresentar problemas de completude. A completude consiste na proporção entre de dados coletados e não coletados, mas que tinham intenção de coleta. A falta de preenchimento, assim como erros ou variações na grafia dificultam o aproveitamento dos dados e podem tornar a análise inviável.

Na medicina veterinária, isso é bastante comum devido aos diferentes modos de se referir a uma espécie animal no momento em que esse dado é coletado. Por exemplo, ao referir-se a um bovino, o operador pode escrever e/ou abreviar de diferentes maneiras: bovino, bovis, boi, vaca, gado, bov, bovi, etc. Isso pode dificultar e atrasar a análise dos dados, pois o responsável pela análise precisará corrigir cada observação, e isso se torna bastante inapropriado quando o número de registros é alto.

Para resolver este problema, uma função que ajude a filtrar, corrigir e dimensionar os erros de preenchimento pode der bastante útil. A função (complete.vet) permite que o usuário tenha uma visão geral do banco de dados, sobre a quantidade de campos não preenchidos e corrija o campo de espécie em diferentes grafias dentre as especies de animais domésticos utilizados na produção.

Planejamento da função

Entrada: `complete.vet` (`file`, `colm`, `specie`, `fill`, `completeness`)

`file`: class `data.frame` no formato .csv

`colm`:vetor de classe `character` com a espécie do animal que foi registrada.

`specie`: determinar qual espécie devem ser identificadas no vetor `colm`.

`fill`: definir qual nome padrão deve ser utilizado para preencher o vetor `colm`.

`completness`: vetor lógico = TRUE calcula a proporção de NAs no banco se houver upload de `file`.

Verificando parâmetros

`file` deve ser no fomato .csv, se não, escrever: "`file` deve ser em formato .csv"

`colm` deve ser vetor da classe `character`.Se não, escrever: "`colm` deve ser da classe `character`.

Pseudo-código

Criar objeto `file` para carregar arquivos

Criar objeto `colm` para determinar qual coluna do `data.frame` ou vetor deve ser utilizado para identificar os padrões de escrita

Criar objeto `specie` que contenha diveros níveis com as possíveis grafias que se refiram aos animais domésticos: bovino, suino, ave, equino, caprino, ovino, cão, gato.

Filtrar todas as possíveis grafias

Criar objeto `fill` atribuido pelo usuário para substituir as grafias por um padrão determinado

Criar objeto `completness` com um vetor lógico = TRUE quando houver interesse em determinar a proporção de incompletude do banco. Soma os NAs e divide pelo número de campos disponíveis para o preenchimento.

Saída

Data frame ou vetor corrigido com o nome padrão para espécie

Proporção de completude

Proposta B

Contextualização

Doenças infecciosas de importância para saúde pública e animal são contempladas, geralmente, com sistemas de vigilância epidemiológica que visam estratégias de prevenção, rápida detecção e combate a focos e disseminação. Para isso, além de componentes de vigilância passiva, como as notificações mandatórias respaldadas por legislação específica (dependendo do agravo em questão), são necessárias medidas ativas, ou seja, busca de potenciais casos de doença numa população suscetível.

Deste modo, componentes ativos de busca de doenças podem ser realizados a partir da coleta de espécimes biológicos para diagóstico, uma vez que o processo de investigação seja desencadeado.

Para tal, o processo de amostragem é essencial, dado que a coleta censitária seria extremanete onerosa e infactível.

Esse processo de amostragem é geralmente realizado em dois níveis: agragado (representando fazendas, abatedouros, granjas, por exemplo) ou individual (animais ou carcaças).

A ideia desta função (smart.sample) é definir o número necessário de fazendas e/ou de animais que devem ser amostrados. Além disso, quando necessario, o usuario poderá de imediato saber quais animais devem ser coletados para avaliar a prevalência aparente de determinada região ou fazenda.

- Em nível agregado: utilização de amostragem aleatória simples entre as fazendas ou abatedoutros de uma região determinada
- Amostragem a nível individual:

Amostragem para proporção ou prevalência aparente (estudo transversal em população infinita ou finita), sem considerar a imperfeição do teste. Onde P é a proporção esperada:

Planejamento da função

Entrada: `smart.sample` (`dado`, `population`, `N`, `confidence`, `precision`, `position`, `P`)

`dado`:`data.frame` deve conter a identificação do agregado (agr) e/ou indivíduo (ind), caso `position = TRUE`. class `data.frame`.

`population`: magnitude da população. =TRUE considerado finita e utiliza o valor ajustado. Default (= FALSE).

N: tamanho de cada população (classe: integer, N > 0). `population` deve ser = TRUE.

`confidence`: confiança desejada (classe: numeric com 2 casas decimais 0 > `confidence` < 1). Default (= `0.95`)

`precision`:precisão desejada (classe: numeric com 2 casas decimais e > 0).

`position`: vetor lógico (=TRUE) determina que a função mostre quais animais ou fazendas devem ser amostrados. Default (= FALSE).

P: proporcao esperada para o agravo. (`numeric`)

Verificando parâmetros

`file`: deve conter a identificação do agregado e/ou indivíduo, caso `position = TRUE`. class (`data.frame`). Se não, escrever:“O arquivo deve ser em formato `.csv` separado por ”;” e conter a identificação de agregados e/ou indivíduos. `position` deve ser = TRUE.

N: tamanho de cada população (classe: integer, N > 0). `population` deve ser = TRUE. Se não, escrever: N deve ser maior que zero)aplicável se `population = TRUE` e `imperfection = TRUE`.

`confidence`:(classe: numeric com 2 casas decimais 0 > `confidence` < 1). Se não, escrever: class deve ser numeric e entre 0 e 1.

`precision`: classe: numeric com 2 casas decimais e > 0). Se não, escrever: class deve ser numeric e maior que 0.

Pseudo-código

Cria objeto `dado`

Cria objeto `population` determinando a partir de um vetor lógico se a população é finita (=FALSE) ou infinita (=TRUE). Se `population`=FALSEutiliza formula ajustada e necessita do valor de `N`, quando `imperfection = TRUE`.

Cria objeto `N` para identificar o número de individuos de uma população finita. Objeto `N` com NAs.

Cria objeto `confidence` que determina o valor de tabela Z para o valor de confiança atribuida.

Cria objeto `precision` para o valor designado pelo usuário.

Cria objeto `position` quando existe o upload de `file` e se `position = TRUE`. Determina quais individuos ou agregados foram amostrados.

Os parâmetros e vetores lógicos são criados a partir desta interação por meio dos controladores de fluxo `if`/`else`.

Preenche a fórmula de acordo com os parâmetros e as perguntas respondidas pelo usuário

Para casos onde `position = TRUE` faz um sorteio dentro do `data.frame` (aleatória simples) para determinar quais os individuos ou agregados devem ser amostrados, baseados no número determinado pela fórmula.

Saída:

Determina o número de individuos ou agregados que devem ser amostrados para determinar a prevalência de um agravo ou fenômeno determinado.

Determina quais individuos ou agregados devem ser amostrados frente ao valor estipulado em populacoes finitas.

> Oi João, achei que os dois planos são factíveis e trazem algo de interessante, mas gostei mais do plano B.
>
> Só tenho como recomendação que vc explique melhor o que está acontecendo na hora de escrever o help da função. Muitos dos termos que vc usou parecem ser jargões da área de epidemiologia/zoonoses que eu não entendi, mas acho que vc consegue explicar melhor.
>
> Uma dica sobre data.frames e arquivos. É melhor que sua função receba um data.frame, e não um nome de arquivo. Com isso, o usuário pode guardar os dados no formato que quiser, ler conforme esse formato, e depois passar pra sua

função só o data.frame. Facilita a vida de todo mundo!

Uma dica sobre usabilidade. NÃO É UMA BOA IDÉIA escrever funções interativas. O melhor é que a função tenha argumentos que representem as respostas a essas perguntas. Não tem problema se forem muitos argumentos. Funções interativas geram códigos de baixa reproducibilidade, o que tira uma das grandes vantagens de usar o R.

Minha recomendação, então, é seguir o plano B, mas sem interatividade e sem receber nomes de arquivos. A função receberia o data.frame no formato especificado (tem que ficar bem explicado no help!) e uma série de parâmetros descrevendo o que o usuário quer/espera de sua amostragem.

Danilo G Muniz

Trabalho final: Determinacao do numero amostral e identificacao de elementos sorteados

Código da funçao 'smart.sample'

Código Smart.sample

Help da funçao 'smart.sample'

Help Smart.sample