Gregory Pitta

Mestrando pelo programa de pós graduação em Ecologia do Instituto de Biociências - USP.

Trabalha com estoques de biomassa vegetal arbórea nos remanescentes de Mata Atlântica. Com o objetivo de estimar quanto existia de biomassa, o quanto foi perdido pela ação humana e o quanto carbono pode ser sequestrado. Podendo, assim, identificar os hotspots de maior biomassa potencial e atual da Mata Atlântica, além de definir projeções para potenciais sequestros de carbono por esse bioma nas próximas décadas.

Inserido no projeto TreeCo, transita entre o LabTrop, o LET e o LEPaC

Página de Exercícios

Exercícios BIE5782-2017

Proposta de Trabalho Final

Proposta A

A função irá corrigir estimativas de biomassa arbórea baseadas em diferentes modelos alométricos.

Modelos alométricos costumam descrever a biomassa de um local:

```
  * ln(AGB) = 5.1218 + ln(q) + 2.4700 * ln(DBH) (Chave et al. 2005)
  * ln(AGB) = 5.5255 + ln(q) + 2.4600 * ln(DBH) (Komiyama et al. 2005)
  * ln(AGB) = 6.9538 + 0.8640 * ln(DBH) + 0.6350 * ln(Height) -1.3700 *
ln(q) (Ray et.al 2011)
  * AGB= 0.0559 *(q*(DBH^2)*Height) (Chave et al. 2014)
  * AGB= 0.0509*DBH^2*H*q  (Sato, 2015)
  * AGB= q* exp ( −1.499 + 2.148 ln (DBH) + 0.207 (ln (DBH))^2 − 0.0281 (ln
(DBH))^3) (Sato, 2015)
```

A função então realiza samples no dataframe com dados referentes as variáveis listadas nos modelos, por exemplo:

```
                        Latin Height   DBH        gx        gy
10597      Amaioua intermedia    5.0  21.0 133.90555 221.24582
184            Qualea cordata   10.0  39.0  25.29976  35.50473
568    Machaerium acutifolium    7.0  35.0  20.50771 111.72880
6559        Xylopia aromatica    7.0  21.0  83.82822 191.01873
```

```
70311         Eugenia sulcata   11.0  35.8 310.70000 221.80000
68735 Calophyllum brasiliense   16.5 189.3 291.70000  63.20000
33930         Ocotea indecora   13.0  69.5 180.43179  24.19011
45916       Meliosma sellowii   10.5  74.0 143.00059  76.79423
5983           Qualea cordata    6.5  40.0  99.59631  96.36049
12356  Copaifera langsdorffii    9.5  25.0 141.59383 236.46030
61112      Amaioua intermedia    3.0  32.3 148.40000  38.10000
14408           Pera glabrata   10.0  40.0 173.20565 315.94335
60971         Ormosia arborea   12.0  55.0 136.20000 301.70000
38390      Croton floribundus    7.0  20.0 286.00000  65.00000
50826          Rudgea recurva    6.5  23.0 254.00125 260.99448
```

E utilizando cada uma das equações fornecidas calcula o valor predito de biomassa nessa parcela amostrada aleatoriamente do banco de dados.

Em seguida após as 1000(?) interações de sample e estimativas de biomassa é produzido um lm() da relação entre os biomassa por parcela do modelo que o usuário escolheu e cada um dos demais modelos separadamente.

A função tem como entrada:

1. Uma lista equações para biomassa
2. Qual equação será tomada como padrão de referência, apenas o numero da linha
3. Um data-frame contendo os dados sobre os quais a equações

A função retorna:

Uma tabela contendo o intercepto e inclinação dos modelos estimadores de biomassa em relação ao escolhido pelo usuário:

Exemplo:

```
        Modelo               Fator de correção
        ------------------------------------------------------------
(1) (Chave,2014)x(Sato, 2015)    intercepto1     inclinação1
(2) (Chave,2014)x(Komiyama, 2005)    intercepto2    inclinação2
(3) (Chave,2014)x(Ray, 2011)    intercepto3    inclinação3
```

Comentários Rena

Oi Greg, Deu pra entender a ideia agora e acho que foi um ótimo exercício mesmo. Eu apostaria na proposta A, mas tenho algumas sugestões (veja se fazem sentido, pois não trabalho com modelos alométricos).

Há uma quantidade finita de modelos? Fiquei pensando que sua função pode ter um ou dois modelos básicos embutidos além de aceitar os modelos que o usuário propõe, ou seja, sua função, por padrão, calcula a biomassa de acordo com Sato (2015) e Chaves et al. (2014). Além disso, vc permite que o usuário indique equações

por conta própria.

As medidas indicadas no exemplo são todas de campo, ou alguma delas é resultado de cálculos? Vc pode dar a opção pro usuário de calcular os valores antes de ajustar o modelo, com um argumentinho que a partir da altura, eu calculo o qq coisa outra q eu preciso pra usar no modelo (de novo, comentário de quem não é da área).

Você vai dar a opção pro usuário de quantas interações de sample ele quer que a função faça? Bom trabalho e qq coisa, to aqui :)

Beijo!

Proposta B

Conversor de coordenadas universal UTM decimal ou graus, retorna o município estado e país da coordenada.

A função tem como entrada:

1. Coordenada x
2. Coordenada y
3. zona (opcional, só para UTM)
4. local para acessar o GADM political map

A função retorna:

A função detecta o tipo de coordenada se UTM decimal ou graus, retorna uma tabela com as outras formas e o país estado e município

Detalhes metodológicos:

A função precisa ser capaz de ler e plotar as coordenadas num shapefile. Shp para isso irá requerer que o usuário tenha em sua máquina o mapa, além dos pacotes necessários para realizar tal tarefa.

Comentários Rena

Beleza,

Entendi, mas não tenho certeza se essa proposta é um desafio real pra você. Você consegue explicar melhor o trabalho que dá para encontrar o local a partir da coordenada, ou ainda pra casar a coordenada com o mapa? To precisando de um incentivo pra gostar dessa proposta...

Proposta C

Função que gera apresentações genéricas a partir de artigos científicos. Extrai o titulo, autor,

imagens e primeiras frases de cada parágrafo organizando por sessão.

A função tem como entrada:

1. caminho para o PDF do artigo

A função retorna:

Uma apresentção em html

Detalhes metodológicos:

A função precisa ser capaz de ler pdfs e extrair e reconhecer campos/ imagens

---

Comentários Rena

Beleza,

Achei essa proposta bem divertida, apesar de não achar q vai ser realmente útil pra vc...rs Acho que tem potencial se você adicionar possibilidades de edição de fonte, cores e/ou tamanho das imagens. Essa te empolga mais que a primeira?

Página de Ajuda

```
func.final               package:unknown               R Documentation

Calcula os desvios de um indice gerado entre métodos de calculo
sobre uma base de dados comum

Description:

     A função tem como objetivo auxiliar a corrigir estimativas de biomassa
     arbórea baseadas em diferentes modelos alométricos existentens na
literatura
     porém pode ser usado para qualquer indice que dependa de diversas
observações
     e cuja a literatura forneça diferentes metodos para tal


Usage:

     func.final(dados,path.formulas, f.interesse=1,n.sim=100, n.sample=10)

Arguments:

 dados                       um daraframe com as observaçoes, cada vetor
```

```
                            precisa ser uma observação de uma variável
                            utilizada pelas equações

 path.formulas,             arquivo com as formulas que serão comparadas
                            precisa ser diretamanete executável no R
                            gerando um resultado que depende das observaçoes
                            existentes no banco de dados

 f.interesse=1,             contra qual equação de interesse as demais serão
                            testadas, por padrão a primeira é a mais
                            importante
 n.sim=100,                 numero de simulações. quantas vezes a função
amostra
                            o banco de dados para gerar as comparações

 n.sample=10                nome infeliz, tamanho da amostra que é retirada
do
                            banco de dados.



Value:

     A função retorna uma janela com o plot dos resultados estimados por
cada equação
     simplesmente para ver se o lm fornece um ajuste aos dados que permite o
uso
     da tabela de co


Warning:

     A função gera variaveis com os nomes das colunas do data frame de dados
para que essas
     variaveis sejam utilizadas nas equações. data frames com nomes de
variaveis que sobre
     ponham nomes usados dentro da funçao vão interferir na execução. Existe
um testes
     embutido na função, mas esse tipo de codigo é propenso a erros
inesperados.
     a função não está otimizada n.sim ou n.sample muito grandes vão tornar
a execução
     muito lenta, o mesmo se n.sample for "n" indicando para ouso do banco
de dados
     completo. isso certamente causará lentidao.

Note:

     ~~further notes~~

     ~Make other sections like Warning with \section{Warning }{....} ~
```

```
Author(s):

     Greg Pitta
     Contact: gregory.pitta at usp dot br

References:

     Chave, J., Réjou-Méchain, M., Búrquez, A., Chidumayo, E., Colgan, M.
S., Delitti,
     W. B., ... & Henry, M. (2014). Improved allometric models to estimate
the
     aboveground biomass of tropical trees. Global change biology, 20(10),
3177-3190.
     GlobAllomeTree http://www.globallometree.org


Examples:

     func.final(data.tree, eq.sample)
     func.final(data.tree, eq.sample, 7, 120, 25)
     func.final(data.tree, eq.sample, 1, 50, n)
```

Código da Função

```
##############################################################################
###
#Author: Gregory Pitta
#Institution: Universidade de São Paulo
#Contact: gregory.pitta at usp dot br
#This work is licensed under the Creative Commons Attribution 4.0
International
#License. To view a copy of this license, visit
#http://creativecommons.org/licenses/by/4.0/ or send a letter to Creative
#Commons, PO Box 1866, Mountain View, CA 94042, USA.

##############################################################################
###
#### Função Final
##############################################################################
###
func.final<-function(dados,path.formulas, f.interesse=1,n.sim=100,
n.sample=10)
{

##############################################################################
###
#### Função que transforma  o P em Signif. codes
##############################################################################
```

```
###
    gen.sig.code<- function (modelobject) # definindao a função e sues
parâmetros
    { #apenas uma série de ifs para substituir
        result=""                  # valores P por códigos de significancia
        if (modelobject<=0.001){
            result="***"
        }else if(modelobject<=0.01){
            result="**"
        }else if(modelobject<=0.05){
            result="*"
        }else if(modelobject<=0.1){
            result="."
        }
        return(result)
    }
################################################################################
###
###TESTES DE VERIFICAÇÃO
################################################################################
###
#verificando se o banco de dados não contém variáveis que serão usadas na
func
    v.nomes.var.user<-names(dados) # criamos vetor com nomes dos vetores de
dados
    # e variáveis
    v.nomes.in.funct<-c("count1", "count2", "count3","count4","count5",
                        "v.nomes.var.user",
                        "v.nomes.in.funct","dados","path.formulas",
"f.interesse",
                        "n.sim", "n.sample", "mat.calc","mat.calc.samp"
    )# vetor com os nomes de todas as variaves usados na função
    if (length(na.omit(match(v.nomes.var.user,v.nomes.in.funct)))>=1){
paste(na.omit(v.nomes.in.funct[match(v.nomes.var.user,v.nomes.in.funct)]),
              "is a var name that halts this funcion, please chage it")
        stop(func.final)
    }
################################################################################
###
#verificando se o usuário não quer amostrar seus dados, opção "n"
################################################################################
###
    if (n.sample=="n"){
        n.sample=nrow(dados)
    }
################################################################################
###
###CORPO DA FUNÇãO
################################################################################
###
    # corpo do loop, deve preencher o valor das formulas para todas as
```

```
repetições
    # Primeiramentecriamaos a matriz de valores calculados com a dimensão do
numero
    #de repetições e a quantidade de formulas dadas pelo usuário

        # esse loop estabelece o número de as repetições
        #que o usuário pede
        #criamos uma matriz que guardará os valors caluculados sob cada
equação
        mat.calc<- matrix(NA, n.sim, nrow(path.formulas))
        #esse loop estabelece o tamanho da amostra tomada em cada repetição,
        #essas são as linhas do dataframe amostrado
        for (count1 in 1:n.sim){ # esse loop estabelece o número de as
repetições
            #que o usuário pede
            #criaos uma matriz que guardará os valors caluculados sob cada
equação
            mat.calc.samp<-matrix(NA,n.sample,nrow(path.formulas))
            #geramos uma amostra nova do banco de dados (ou o banco de dados
todo)
            dados.samp<-dados[sample(nrow(dados),n.sample),]
            #esse loop estabelece o tamanho da amostra tomada em cada
repetição,
            #essas são as linhas do dataframe amostrado
            for (count2 in 1:n.sample){
                #esse loop estabelece para qual formula estamos realizando
as
                #operações (arquivo de formulas)
                for (count3 in 1:nrow(path.formulas)){
                    # esse loop cria as variaveis e dá o valor para elas de
acordo
                    #com o valor da linha do contador
                    for (count4 in 1:(ncol(dados))){
                        assign(colnames(dados[count4]),
                               dados.samp[count2,colnames(dados[count4])])
                    }
                    #Aqui linha a linha puxamos objetos pela posição
indexada
                    texto = path.formulas[count3,1]
                    #o comando parse torna o texto do objeto com equações em
uma
                    #expressao real
                    expressao = parse(text=texto)
                    # fechamos ford count4, temos todas as variaveis com
                    #valores atribuidos
                    #salvamos na nossa tabela temporária os valroes
calculados
                    #por cada formula, para um individuo
                    mat.calc.samp[count2,count3]<-eval(expressao)
                } # fechamos for count3, temos uma linha da nossa tabela
```

```
                #temporária completa para todas as formulas
            }# fechamos for count2, temos a tabela completa, para todos os
            #individuos da amostra
            # Realizamos um apply para condensar (somando) a amostra em uma
linha da
            #tabela final,temos os valores totais somados para uma amostra,
calculados
            #por diferentes formulas
            mat.calc[count1,]<-apply(mat.calc.samp, 2, FUN="sum")
        }# fechamos o for count1 a tabela com todas as simulações está
completa.

#############################################################################
###
    ##Registrando os resultados
#############################################################################
###
    #temos que agora comparar com a função escolhida pelo usuário e gerar
    # o resultado final da função
    #criamos uma matriz para salvar os objetos do resultado
    result<-matrix(NA,nrow(path.formulas), 7)
    # e nomeamos os vetores
    colnames(result)<-c("eq","intercepto", "Pr(>|t|)","S.code",
                        "inclinação", "Pr(>|t|)","S.code")
    #abrimos um dispositivo gráfico
    graphics.off()
    x11()
    #e dividimos ele apropriadamente para o número de equações testadas
    par(mfrow=c(floor(sqrt(nrow(path.formulas))),
                ceiling(sqrt(nrow(path.formulas)))))
    # criamos um loop que testa a equação ecolhida contra cada uma das
demais
    for (count5 in 1:nrow(path.formulas)){
        #pulamos aqui o contador para não fazer lm de uma formula sobre ela
        #mesma
        if(count5==f.interesse){
            count5=count5+1
        }
        # lm é uma maneira simples de teststar os resultados da eq de
interesse
        # contra os calculos gerados pelas demais equações
        lm1<-lm(mat.calc[,f.interesse]~mat.calc[,count5])

        # um simples Plot com as legendas das funções, não temos os nomes
pois
        # ao ler um arquivo .R não temos as colunas nomeadas
        plot(mat.calc[,f.interesse]~mat.calc[,count5],
             ylab=path.formulas[f.interesse,2],
xlab=path.formulas[count5,2])
        #linha da regressão lm no plot
        abline(lm1, col="red")
```

```
        # criando a matriz de resultados, para cada linha:
        # salvamos o intercepto
        result[count5,1]<-
paste(eq.sample[f.interesse,2],"vs.",eq.sample[count5,2])
        result[count5,2]<-round(lm1$coefficients[1],2)
        #salvamos o p valor para o intercepto
        result[count5,3]<-round(summary(lm1)$coefficients[1,4],2)
        #geramos o codigo de significancia
        result[count5,4]<-gen.sig.code(summary(lm1)$coefficients[1,4])
        # salvamos a inclinação
        result[count5,5]<-round(lm1$coefficients[2],2)
        # salvamos o p valor para a inclinação
        result[count5,6]<-round(summary(lm1)$coefficients[2,4],2)
        #geramos o codigo de significancia
        result[count5,7]<-gen.sig.code(summary(lm1)$coefficients[2,4])
    }
    #imprimimos na tela os resultados
    return(result)
    # e a legenda dos codigos de significancia
    paste("Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1")
}
```

Arquivo da Função

Função Final

Exemplo 1

Dados de Biomassa

O site GlobAllomeTree, disponibiliza uma rica base de dados que inclui medidas alométricas para espécies arbóreas de todo o mundo, incluindo as densidades de madeira para algumas espécies. Mais importante de tudo, contém um banco de equações alométricas (mais de 13.000!) que pode ser utilizado livremente para testar a função.

Os arquivos da base de dados estão sob Copyright do GlobAllomeTree mas estão em parte aqui com fins educativos. versões parciais podem ser baixadas a seguir allometric equations wood densities raw tree data

Obs. esses arquivos são pesados e grandes!

Vamos agora como nos tutoriais tratar esses dados para que entrem bem na função! (h8-) muahaha)

```
equations<-read.csv("allometric_equations3.csv")
equations$Reference_short<-paste(equations$Reference_author,"
","(",equations$Reference_year,")",sep="")
```

```
rawdata<-read.csv("raw_data.csv")
densities<-read.csv("wood_densities.csv")

v<-c("DBH_cm","H_m","ABG_kg","Family","Genus","Species","Subspecies")
data.tree<-rawdata[,v]

s<-c("Density_g_cm3","Genus","Species","Subspecies")
data.dens<-densities[,s]


data1.eq<-matrix(NA,nrow(equations),2)

grep("log Biomass=",equations$Substitute_equation[i])

for (i in 1:nrow(equations)){
    if(length(grep("log Biomass=",equations$Substitute_equation[i]))>=1){
    data1.eq[i,1]<-sub("log Biomass=", "exp(",
equations$Substitute_equation[i])
    data1.eq[i,1]<-paste(data1.eq[i,1],")",sep="")
    data1.eq[i,2]<-equations$Reference_short[i]
    }else{
        data1.eq[i,1]<-sub("","",equations$Substitute_equation[i])
        data1.eq[i,2]<-equations$Reference_short[i]
    }
}


for (i in 1:nrow(equations)){
    if(length(grep("log10 Biomass=",data1.eq[i,1]))>=1){
        data1.eq[i,1]<-sub("log10 Biomass=", "10^(", data1.eq[i,1])
        data1.eq[i,1]<-paste(data1.eq[i,1],")",sep="")
    }else{
        data1.eq[i,1]<-sub("","",data1.eq[i,1])
    }
}


    for (i in 1:nrow(equations)){
    if(length(grep("Biomass=",data1.eq[i,1]))>=1){
        data1.eq[i,1]<-sub("Biomass=", "", data1.eq[i,1])
    }
    if(length(grep("biomass=",data1.eq[i,1]))>=1){
            data1.eq[i,1]<-sub("biomass=", "", data1.eq[i,1])
    }
    if(length(grep("Volume",data1.eq[i,1]))>=1){
        data1.eq[i,1]<-NA
    }
    if(length(grep("#NAME?",data1.eq[i,1]))>=1){
            data1.eq[i,1]<-NA
        }
    }
```

```
for (i in 1:nrow(equations)){
    data1.eq[i,1]<-sub("LOG", "log", data1.eq[i,1])
    data1.eq[i,1]<-sub("ln", "log", data1.eq[i,1])
    data1.eq[i,1]<-sub("Exp", "exp", data1.eq[i,1])
}


data.eq<-na.omit(data1.eq)
unique.eq<-unique(data.eq)
eq.sample<-unique.eq[sample(nrow(unique.eq), 10), ]


data.tree$WD<-NA
data.dens$latin<-paste(data.dens$Genus,data.dens$Species)
data.tree$latin<-paste(data.tree$Genus,data.tree$Species)

Genus_density<-tapply(data.dens$Density_g_cm3,list(data.dens$Genus),FUN
="mean")

Species_density<-tapply(data.dens$Density_g_cm3,list(data.dens$latin),FUN
="mean")

for (i in 1:(nrow(data.tree))){
    if(is.na(Species_density[as.character(data.tree$latin[i])])==TRUE) {
        data.tree$WD[i]<-Genus_density[as.character(data.tree$Genus[i])]
        }else
        {
        data.tree$WD[i]<-Species_density[as.character(data.tree$latin[i])]
        }
}

data.tree<-na.omit(data.tree)
names(data.tree)<-c("DBH","H","AGB","Family","Genus",
"Species","Subspecies", "WD", "latin")
```

Verifique o objeto eq.sample, embora tenhamos tratado a maioria das adversidades do banco de dados são mais de 8000 equações e houve uma série de erros na montagem delas...

caso as formulas pareçam ok tente rodar:

```
func.final(data.tree,eq.sample)
```

cada vez que você criar o objeto eq.sample um novo set de equações estará disponível para testar a função

Obs. essas equações em grande parte tem um contexto ou um grupo taxonômico ao qual se aplicam e o sample não considera isso ou a composição do banco de árvores. Estamos aqui apenas fazendo um exercício com a função