Tutorial_Escala

ACMartensen

8/16/2020

Contents

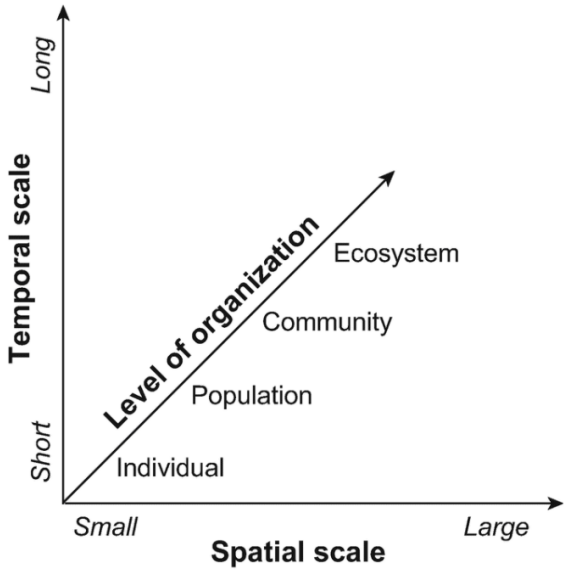

Fig. 2.1
The problem of scale and levels of organization, viewed through a modified space-time diagram.

Figure 1: Escalas espacial e temporal e níveis de organização, Fonte: Fletcher & Fortin (2019)

Apresentação

Esse exercício elabora em cima do que vimos na aula teórica sobre escala espacial. Como vimos na aula teórica, todos os fenômenos ecológicos tem uma dimensão temporal e espacial (Fig. 1 e 2), e o conceito de escala tenta capturar essas dimensões e mais importante, fazer sentido delas (Fletcher & Fortin 2019).

É muito importante ficar claro para você o que é escala (e o que não é!) e qual a importância desse conceito na elaboração do desenho experimental de um estudo, na coleta de dados, nas análises e na tomada de decisão. Esse tutorial te ajudará nisso!

Nesse tutorial iremos explorar aspectos de *resolução* e *extensão*, de forma que você possa ganhar maior familiaridade com esses conceitos, e usaremos um exercício baseado no Capítulo 2 do livro **Fletcher R, M-J Fortin. 2019. Spatial Ecology and Conservation Modeling: Applications with R. Springer.**. Ao longo do tutorial você precisará responder algumas perguntas em um formulário disponível no Google Classroom.

O Código

Como você já viu nos tutoriais do **R**, é em geral necessário baixar alguns pacotes para que possamos fazer as nossas análises. O pacote que iremos baixar é o *raster*. Lembre-se que todas as vezes que precisarmos baixar algum pacote você deverá usar o código abaixo.

```
install.packages("raster")
```

Quando você precisar baixar mais de um pacote, você pode usar o *c()* de concatenar, conforme o código abaixo:

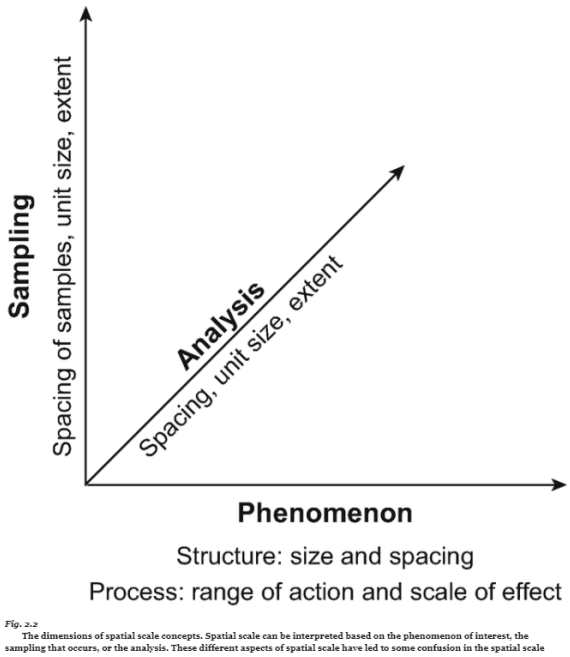

Fig. 2.2
The dimensions of spatial scale concepts. Spatial scale can be interpreted based on the phenomenon of interest, the sampling that occurs, or the analysis. These different aspects of spatial scale have led to some confusion in the spatial scale

Figure 2: As dimensões da escala espacial, Fonte: Fletcher & Fortin (2019)

```
install.packages(c("raster", "rgeos"))
```

Dessa maneira você baixa os dois (ou quantos quiser) pacotes numa linha de código única!

Após baixar os pacotes você precisa "ligá-los", e você pode fazer isso através da linha de código abaixo. Você também pode usar a função *require(raster)* que vai gerar o mesmo resultado do que a *library(raster)* apresentada abaixo.

```
library(raster)
```

Uma paisagem bem simples

Inicialmente iremos gerar uma paisagem bem simples, de 6 por 6 pixels. Você já deve saber que pixel eh a unidade básica de uma imagem (lembra da camera do seu celular, 10Mb ou algo assim?!). Vocês devem ter visto sobre pixels e resolução no meso de geoprocessamento. E na aula teórica vimos que podemos tratar o pixel aqui como a resolução. Vamos dizer que temos um pixel de 10m (res=10 no bloco de código), ou seja, uma quadrado de 10 por 10m, sendo essa, a menor unidade mapeável (tem relação com escala cartográfica, lembra da aula teórica?!).

```
pai<-raster(ncol=6, nrow=6, xmn=1, xmx=60, ymn=1, ymx=60, res=10) #Essa linha de comando gera a paisage
values(pai)<-1:ncell(pai) #E essa atribui valores para os pixels criados acima
plot(pai) #essa plota
text(pai, digits=2) #Essa coloca os valores dos pixels
```

Mas nessa simulação a gente só colocou os números dos pixels, e como cada um tem o seu número, cada um ganhou uma cor diferente. Mas e se a gente quiser gerar um "mapa mesmo", com classes diferentes?

Usaremos aqui a função **rpois** "irmã" da rnorm que usamos no tutorial do **R**, lembra dela? Dá uma olhada com **?rpois** e veja o que ela faz.

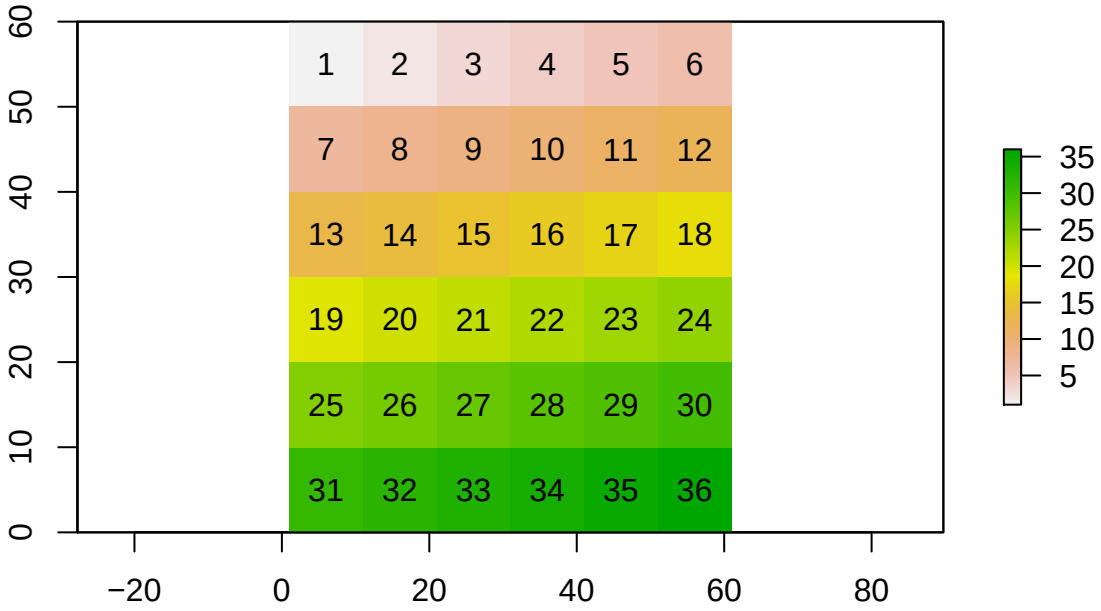

Figure 3: Paisagem simples de 36 pixels numerados

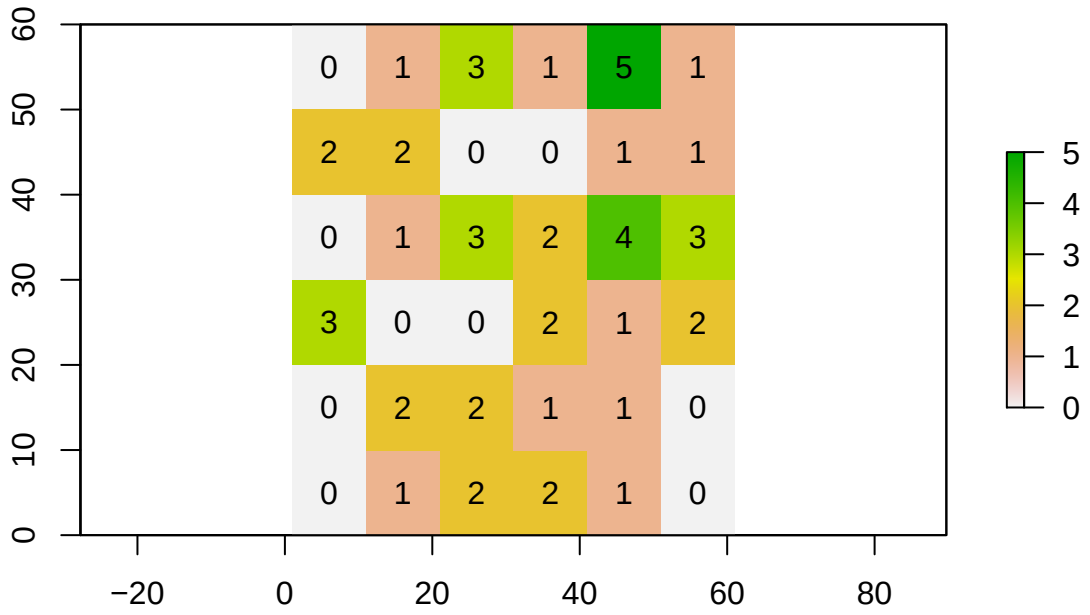

Figure 4: Agora sorteamos classes para os pixels e eles não estão mais apenas numerados, mas apresentam classes determinadas de uso e cobertura do solo, como por ex: cana, pastagem, floresta...

```
set.seed(1229) #Esse valor está aqui só para garantir que todas as simulações terão o mesmo resultado.
values(pai) <-rpois(ncell(pai), lambda=1) #Aqui simulamos valores (não a ordem dos pixels como acima)
plot(pai)
text(pai, digits=2)
```

P.01: Explique o que é a função rpois e o que ela fez nessa linha de código.

Perceba que no mapa acima temos 6 classes (0, 1, 2, 3, 4 e 5) e elas tem a seguinte frequência de ocorrência:

```
table(values(pai))
```

P.02: Inclua esses valores no formulário de respostas.

Alterando a resolução

Agora iremos degradar a resolução desses dados, ou seja, iremos alterar o tamanho dos pixels. Por exemplo, iremos juntar 4 pixels em um único pixel. Como você acha que podemos fazer isso? Quais valores esse pixel que vai substituir os 4 originais deve ter?

Existem diversas maneiras de se fazer isso, uma das formas é através da média.

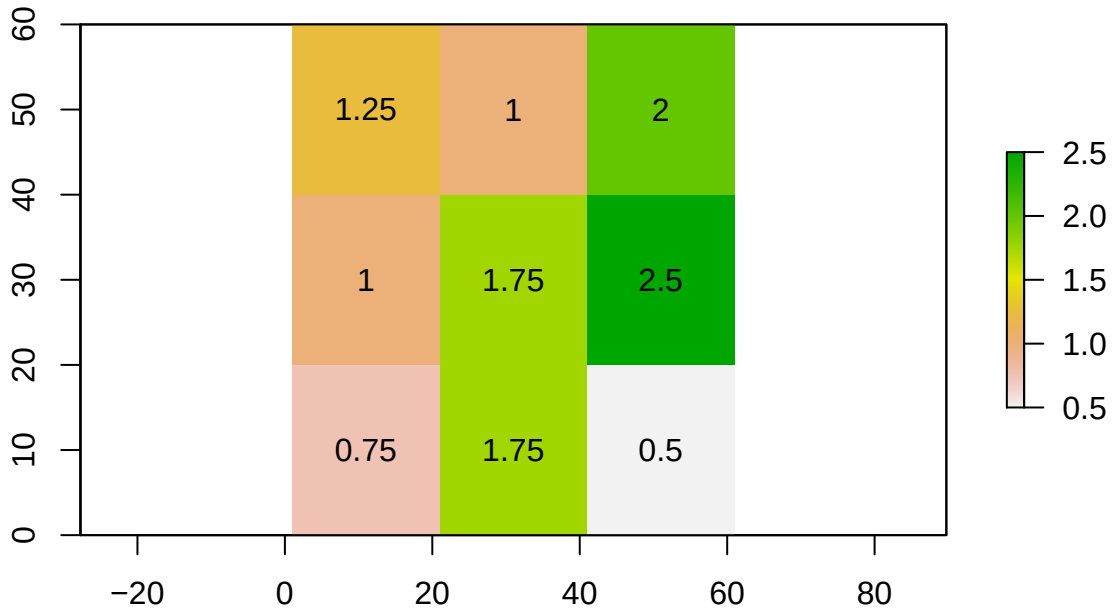

Figure 5: Reamostragem pela média

```
pai_media<-aggregate(pai, fact=2, fun=mean)
```

A média é uma forma particularmente adequada para se executar com variáveis contínuas, como por exemplo, cobertura de dossel, número de flores, etc.

```
plot(pai_media)
text(pai_media, digits=2)
```

Confira os resultados com o mapa original.

P.03: Agora mude o *digits* para *digits=3* e explique o que ocorreu.

Outra opção é utilizar o valor mais comum da área, o que é particularmente adequado quando temos um mapa categórico, como por exemplo mata/não-mata, ou soja, floresta e pasto.

```
pai_dom<-aggregate(pai, fact=2, fun=modal)
```

```
plot(pai_dom)
text(pai_dom, digits=2)
```

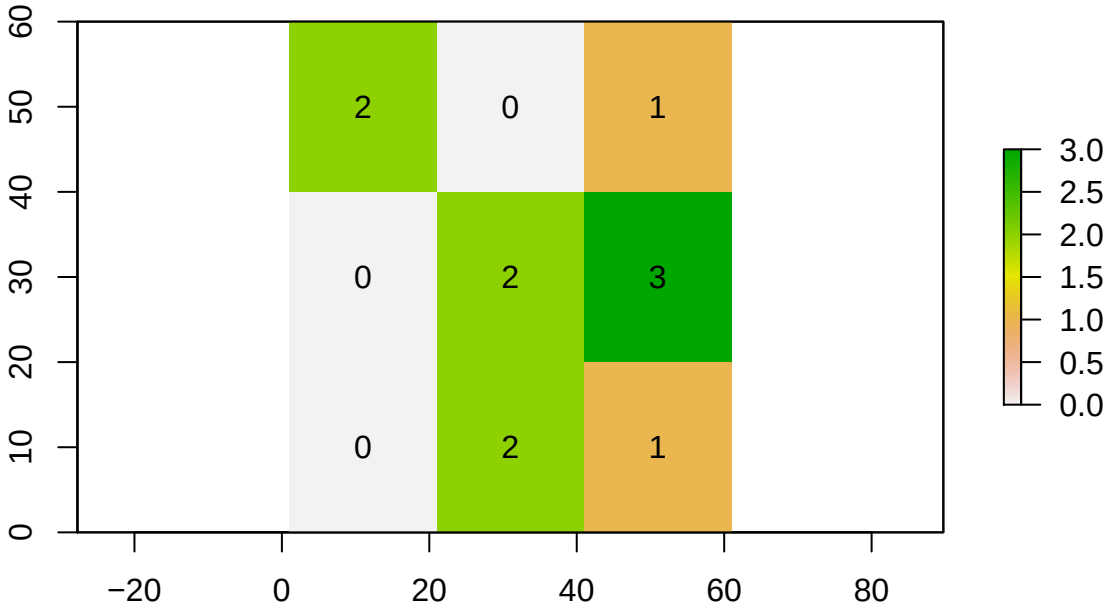

Figure 6: Reamostragem pela dominante

P.04: Confira os resultados e explique o que aconteceu.

P.05: Após esse comando, qual o tamanho do pixel resultante?

P.06: Compare os resultados obtidos pelos dois métodos numa mesma resolução e discuta.

Abaixo, você pode observar a média e a variância para o mapa **pai** original.

```
cellStats(pai, mean)
```

```
## [1] 1.388889
```

```
cellStats(pai, var)
```

```
## [1] 1.55873
```

P.07: Execute os mesmos comandos para os mapas pai_media e pai_dom e discuta os resultados comparando-os com os do mapa original.

Podemos também diminuir o tamanho dos pixels, através de reamostragem. Novamente existem diferentes técnicas para se executar essa tarefa. Uma das formas é simplesmente dividir o pixel original.

```
pai_dis<-disaggregate(pai, fact=2)
```

```
par(mfrow=c(1,2)) #Este comando indica que o R deve plotar duas figuras, no caso em 1 linha e duas colu
plot(pai)
text(pai, digits=2)
plot(pai_dis)
text(pai_dis, digits=2)
```

Outra opção é reamostrar utilizando o método de interpolação bilinear, que utiliza a média ponderada pela distância dos pixels do entorno, tanto no eixo x, quanto no eixo y.

```
pai_bil<-disaggregate(pai, fact=2, method="bilinear")
```

```
par(mfrow=c(1,2))
plot(pai)
text(pai, digits=2)
plot(pai_bil)
text(pai_bil, digits=2)
```

O método de reamostragem por interpolação bilinear é mais indicado para dados contínuos e em geral, com bordas não claramente delimitadas. Para dados categóricos ele não é muito recomendado.

Alterando a extensão

Vamos agora alterar a extensão, e já utilizaremos essa mudança para avaliar melhor os resultados da interpolação bilinear.

Inicialmente iremos criar uma nova área menor, ou seja, iremos explorar uma extensão menor.

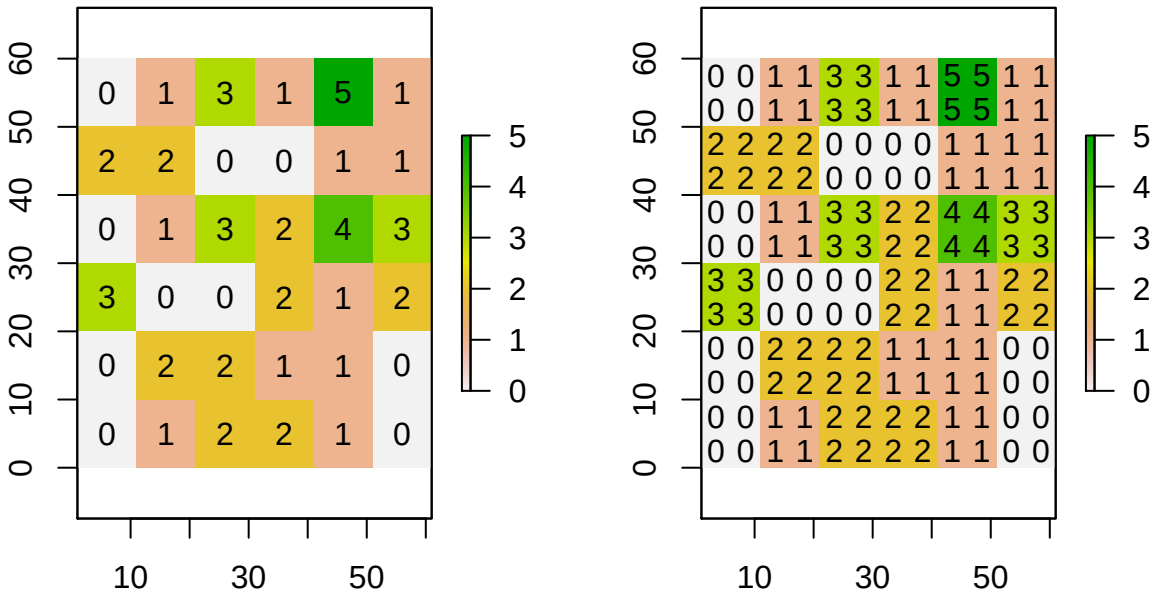

Figure 7: Paisagem original e paisagem reamostrada

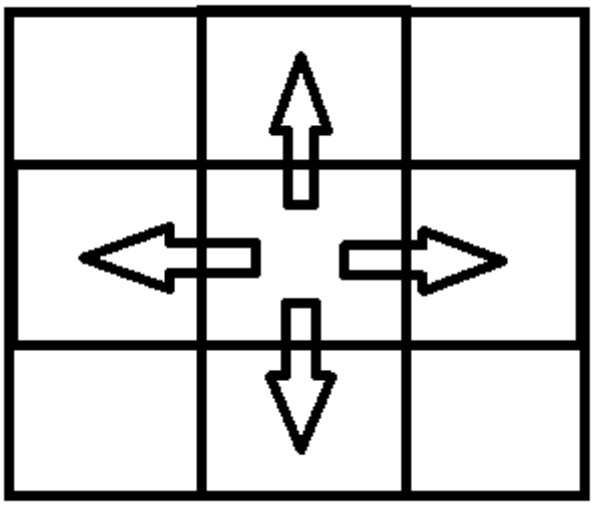

Figure 8: Fig. 8: Esquema do método de reamostragem bilinear

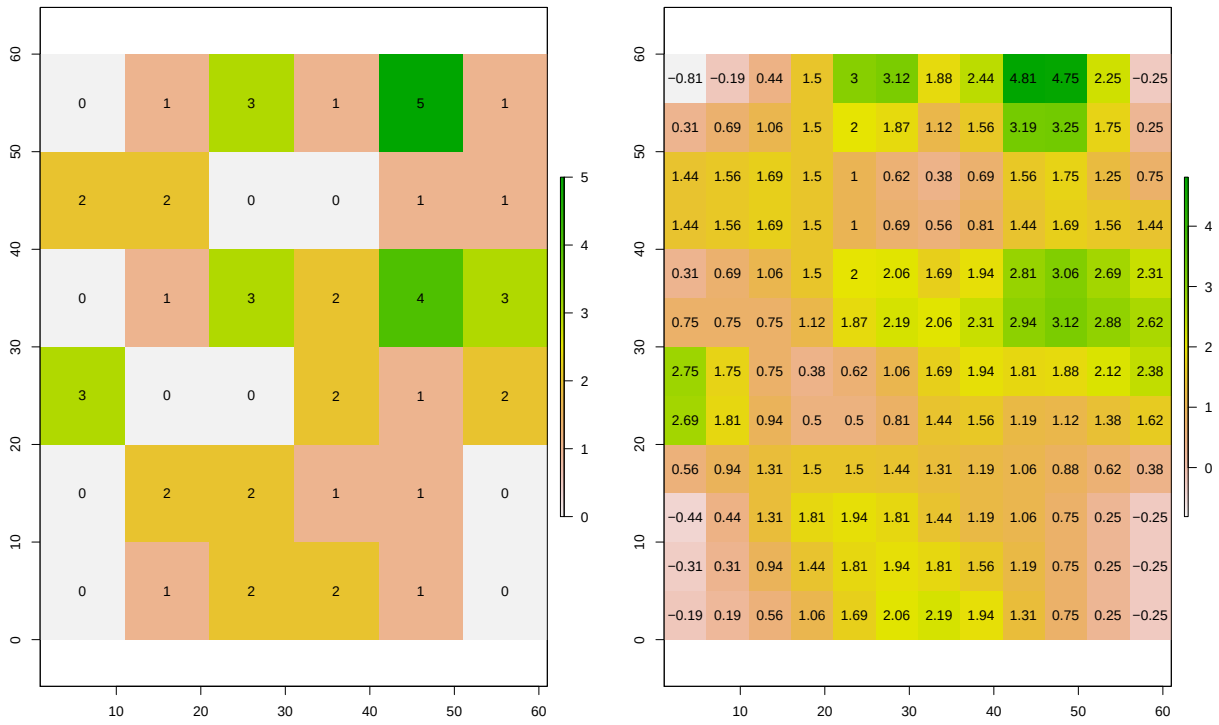

Figure 9: Paisagem original e reamostrada pelo método bilinear

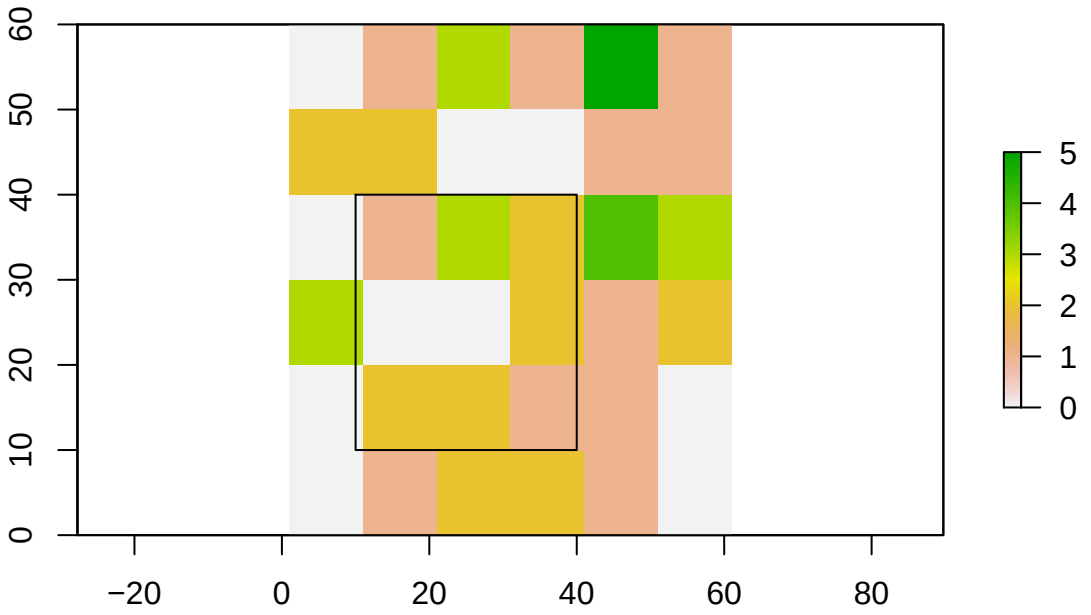

Figure 10: O mapa pai com a nova extensão menor em preto

```
e<-extent(10,40,10,40)
```

Veja a área que selecionamos

```
plot(pai)
plot(e, add=TRUE)
```

Agora iremos recortar essa área

```
pai_crop<-crop(pai, e)
```

```
plot(pai_crop)
```

```
par(mfrow=c(1,2))
plot(pai, col=c("white", "pink", "orange", "green", "blue", "red"))
text(pai, digits=2)
plot(e, add=TRUE)
plot(pai_crop, col=c("white", "pink", "orange", "green"))
text(pai_crop, digits=2)
```

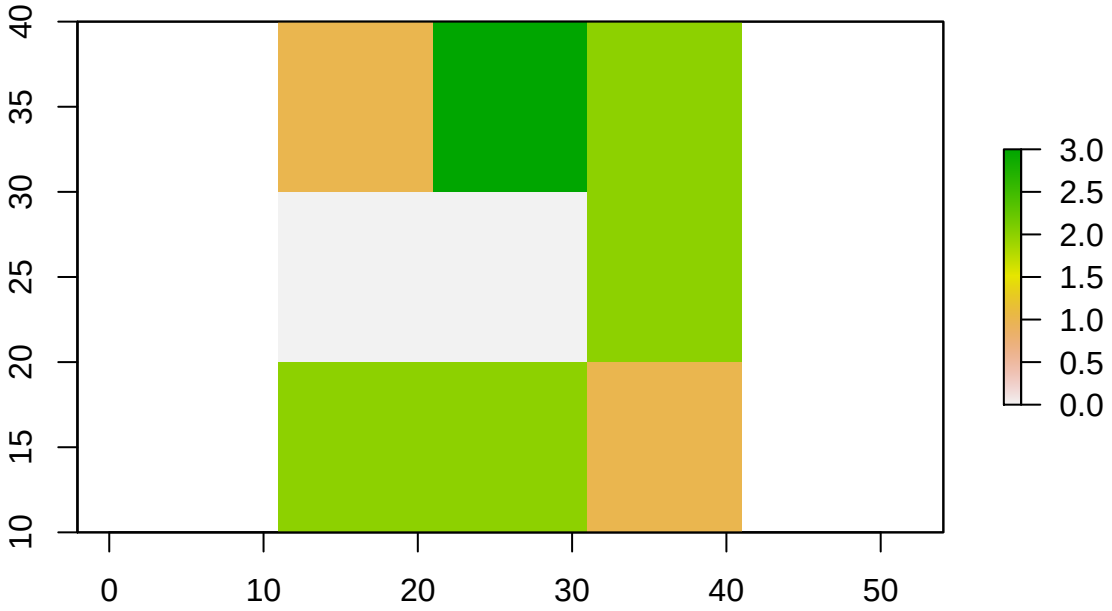

Figure 11: Plotando o recorte - mudando a extensão

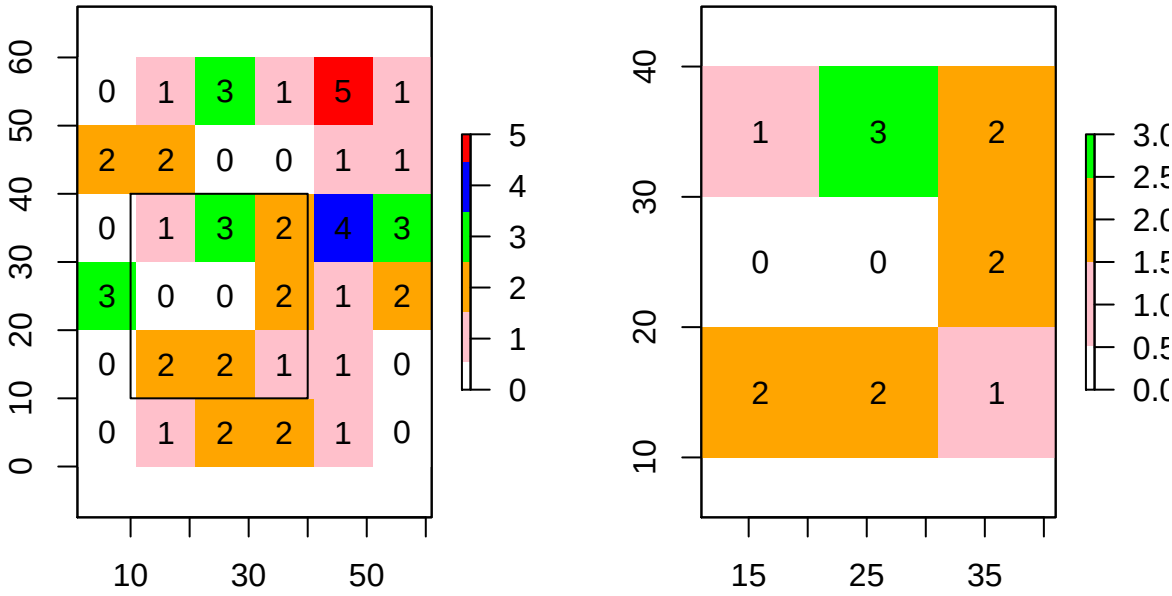

Figure 12: Plotando o original, a área de recorte e o recorte

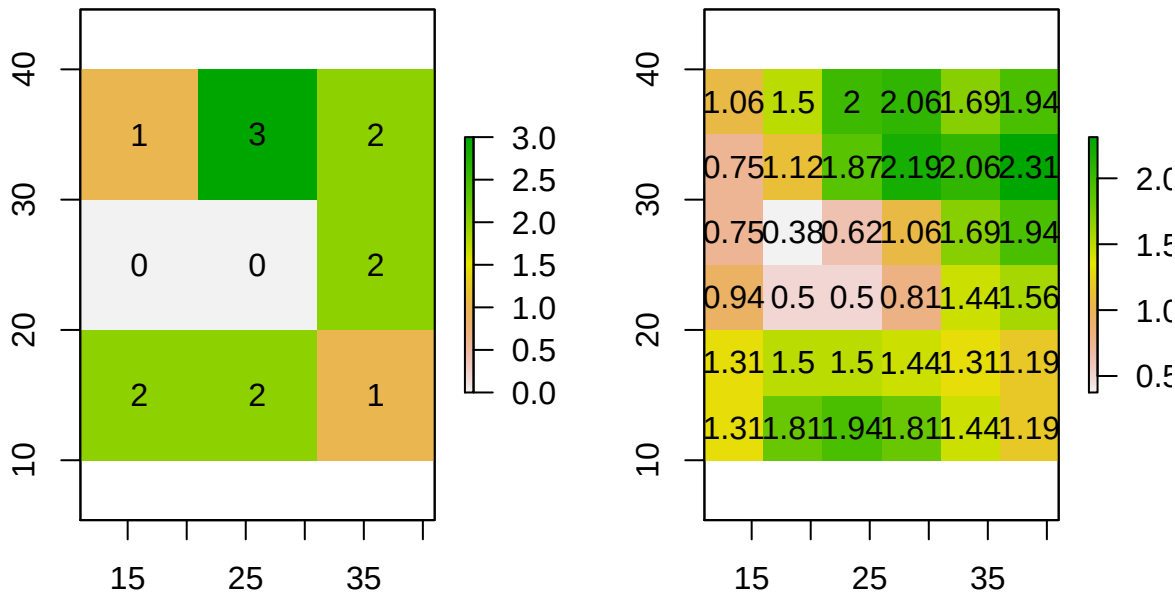

Figure 13: Plotando o recorte original e o recorte reamostrado pelo método bilinear

P.08: Qual é a extensão em número de pixels desse recorte?

Agora iremos tentar entender um pouco melhor da reamostragem bilinear. Para isso, iremos usar o mesmo recorte que fizemos anteriormente *e* para cortar e comparar a paisagem original e a com resolução alterada através do método bilinear.

```
bil_crop<-crop(pai_bil, e)
```

E agora vamos plotar

```
par(mfrow=c(1,2))
plot(pai_crop)
text(pai_crop, digits=2)
plot(bil_crop)
text(bil_crop, digits=2)
```

Veja essa figura para lhe ajudar a pensar no método bilinear de reamostragem (Ver fig. 8).

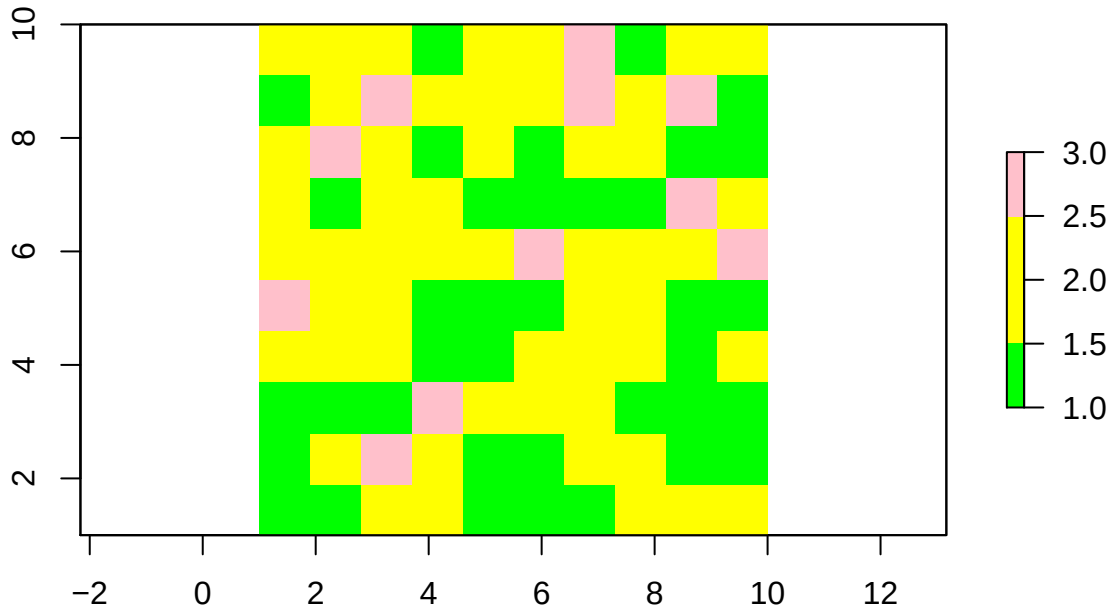

Figure 14: Plotando a paisagem mais complexa

P.09: Sabendo que o método bilinear considera os 4 pixels do entorno no cálculo (ver figura acima), explique quando você optaria por usar esse método em uma análise. Aborde aspectos como que tipo de espécie, de pergunta, etc.

Uma paisagem um pouco mais complexa

Para uma paisagem de 10x10 onde temos 3 classes (ou seja, uma mapa categórico, você lembra da primeira aula?!): ~30% de floresta (verde), ~60% de pastagens (amarelo) e ~10% de agricultura (rosa)

```
set.seed(1142)
pai.c<-raster(ncol=10, nrow=10, xmn=1, xmx=10, ymn=1, ymx=10)
d<-rmultinom(100, size = 1, prob = c(0.3, 0.6, 0.1))
j<-vector()
for (i in 1:length(which(d==1))){
  dd<-which(d==1)[i]-3*(i-1)
  j<-c(j, dd)
}
values(pai.c)<-j
```

```
plot(pai.c, col=c("green", "yellow", "pink"))
```

Agora você deve realizar as seguintes etapas e responder as perguntas abaixo.

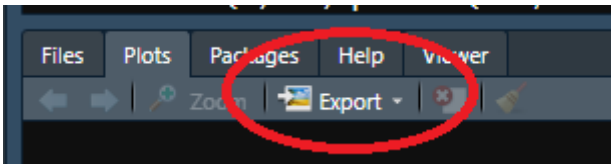

Figure 15: Método de salvar a figura para incluir no formulário

Você precisará salvar as figuras. Nesse momento (mais pra frente no curso eu irei explicar outros modos!) utilizaremos o botão Export > Save as Image do *painel Plots* no canto inferior direito do RStudio (caso você não tenha alterado a organização dos painéis).

P.10: Mude a resolução pela média e utilize fact=2. Posteriormente, mude a resolução pela dominante (moda). Gere uma figura com os dois mapas lado a lado utilizando *par(mfrow=c(1,2))*. Inclua a figura no formulário de respostas.

P.11: Crie uma área para recortar a paisagem de 10x10. Recorte a paisagem para uma extensão menor. Plote a paisagem de 10x10 e inclua a área a ser recortada em preto. Ao lado, plote apenas a áreas recortada. Inclua a figura no formulário de respostas.

P.12: Ao invés de plotar uma figura ao lado da outra, plote uma em cima da outra. Como você alteraria o comando para que o R execute dessa maneira. Inclua apenas a linha de código no formulário.