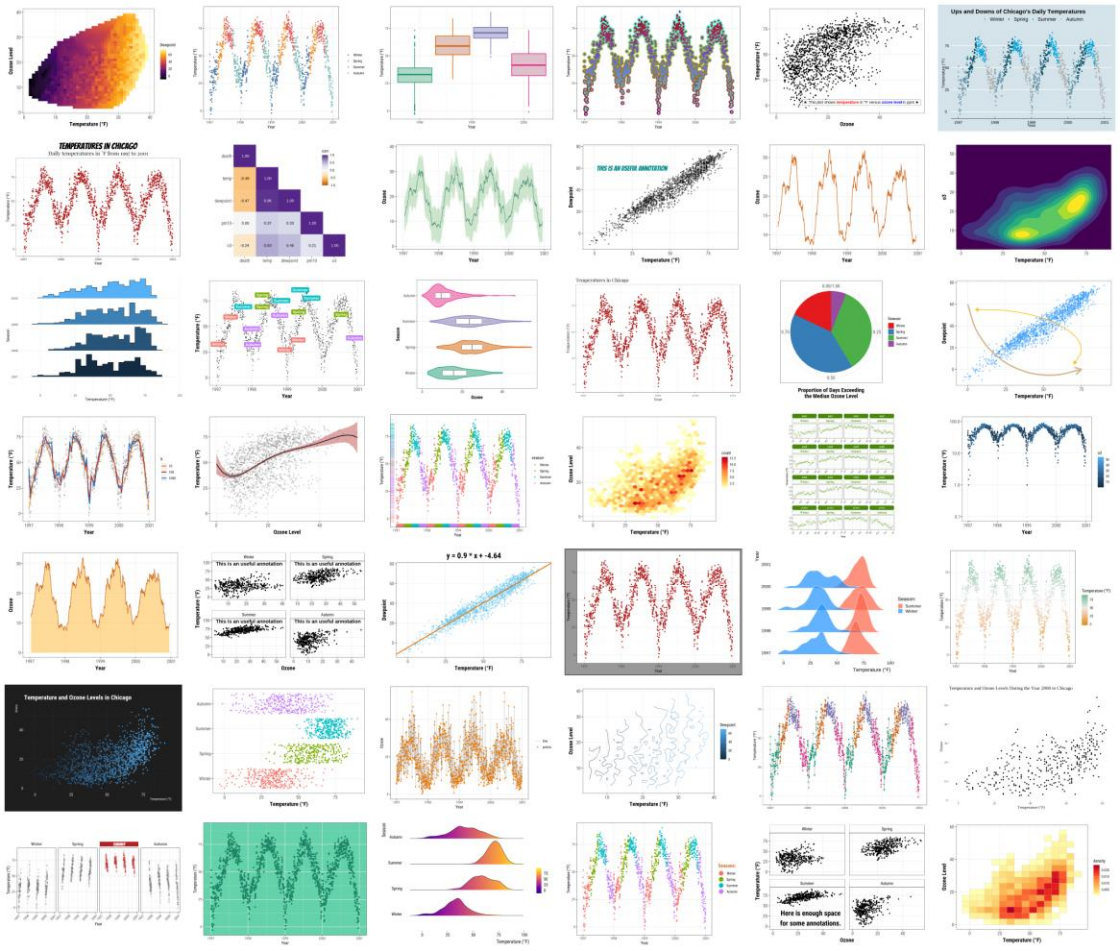
Ggplot

Material desenvolvido por Dr. Clarice Braúna Mendes

03-05-2024

O **ggplot** é possivelmente o pacote mais conhecido do ambiente R: ele possui uma sintaxe própria, é elegante e versátil (quase um poeta). Mas não se engane, assim como as *pipelines* do Tidyverse, ele pode causar estranhamento à primeira vista. Inclusive, ele também foi criado e desenvolvido pelo Hadley Wickham, cientista de dados do RStudio e criador do Tidyverse. De qualquer forma, o desafio vale a pena: ele possui diversos parâmetros, cores e possibilidades de visualização, tornando-o uma ferramenta poderosa, senão praticamente indispensável para todos aqueles que queiram se aprofundar no R.



O significado do nome do **ggplot** deriva do inglês *grammar of graphics*, ou "gramática dos gráficos". Essa gramática, sintaxe ou lógica de programação funciona por meio da *adição* de componentes aos seus *plots* iniciais, e iremos detalhá-la para que você possa compreendê-la e utilizá-la no seu dia a dia na pesquisa! Além disso,

iremos ver como que o **ggplot** se relaciona com as outras funcionalidades presentes no Tidyverse, como aquelas criadas para a limpeza e transformação de *dataframes* (e.g. pacotes **dplyr**, **tibble** entre outros).

Carregando pacotes

O ggplot é um dos pacotes do Tidyverse, por essa razão aplicamos o **library(tidyverse)**. Também iremos carregar o pacote **readxl** para ler os arquivos em Excel do nosso interesse.

```
library(readxl)
library(tidyverse)
```

Importando os dados

Vamos trabalhar novamente com os dados que utilizamos em nossa aula de Tidyverse. Além disso, iremos incluir a tabela que criamos ao final da aula passada, que relacionava o número de Áreas Protegidas e o uso de solo em cada Área de Planejamento da cidade do Rio de Janeiro.

```
#dados do uso do solo da cidade do Rio de Janeiro
uso_solo_rio<-read_excel("Classes de Uso do solo e Cobertura Vegetal - 
RJ.xlsx",sheet = "Dados")

#dados das áreas protegidas da cidade do Rio de Janeiro
aps_rio<-read_excel("Areas_Protegidas_Rio.xlsx")

#dados relacionais das áreas protegidas e do uso de solo para cada área 
de planejamento do Rio de Janeiro
dados_relacionais_uso_solo_aps_rio<-read_excel("Relação Nº Áreas 
Protegidas X Uso do solo - RJ.xlsx")
```

Vamos aplicar a função **glimpse**, que você aprendeu na aula passada, para analisar os dados das nossas tabelas, inclusive a que você criou. Observe que as Áreas de Planejamento são lidas como "double", ou seja, como números, em todos os *dataframes*. Entretanto, sabemos que estas Áreas na verdade são siglas, e que devem ser melhor compreendidas como "character".

```
glimpse(uso_solo_rio)
glimpse(aps_rio)
glimpse(dados_relacionais_uso_solo_aps_rio)

#transformando área de planejamento do uso de solo em "character"
uso_solo_rio$`Área de Planejamento (AP)`<-as.character(uso_solo_rio$`Área 
de Planejamento (AP)`)

#transformando área de planejamento das áreas de proteção em "character"
aps_rio$area_plane<-as.character(aps_rio$area_plane)

#transformando área de planejamento dos dados relacionais em "character"
```

```
dados_relacionais_uso_solo_aps_rio$`Área de Planejamento (AP)`<-
as.character(dados_relacionais_uso_solo_aps_rio$`Área de Planejamento
(AP)`)
```

Destrinchando a gramática de gráficos

O **ggplot** funciona por meio da *adição* de dados e parâmetros de interesse:

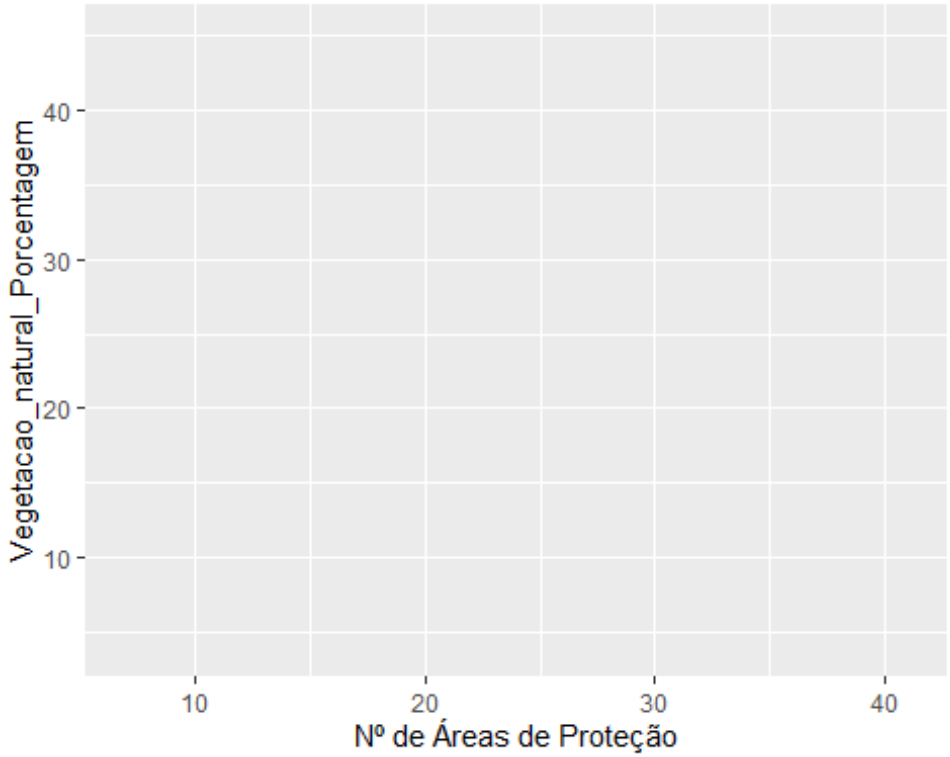
```
  ggplot()
```


Ao executar a função acima, perceba que apenas um *plot* vazio foi criado. Isso ocorreu porque você não adicionou nenhuma informação a ele! Vamos fazer isso no próximo passo, com os dados relacionais das Áreas Protegidas e do uso de solo para cada Área de Planejamento do Rio de Janeiro.

Inserindo o *dataframe* e suas relações representadas pelos eixos do gráfico

Suponha que você tenha interesse em saber se a porcentagem de Vegetação Natural (variável resposta = eixo y) aumenta conforme um maior número de Áreas Protegidas (variável preditora = eixo x), independentemente da Área de Planejamento analisada. Abaixo, é possível ver que, primeiramente, *adicionamos* o *dataframe* que queremos analisar, e logo em seguida, dizemos ao **ggplot** quais são nossas variáveis de interesse e como elas se relacionam.

```
  ggplot(dados_relacionais_uso_solo_aps_rio,aes(x = `Nº de Áreas de
Proteção`, y = `Vegetacao_natural_Porcentagem`))
```

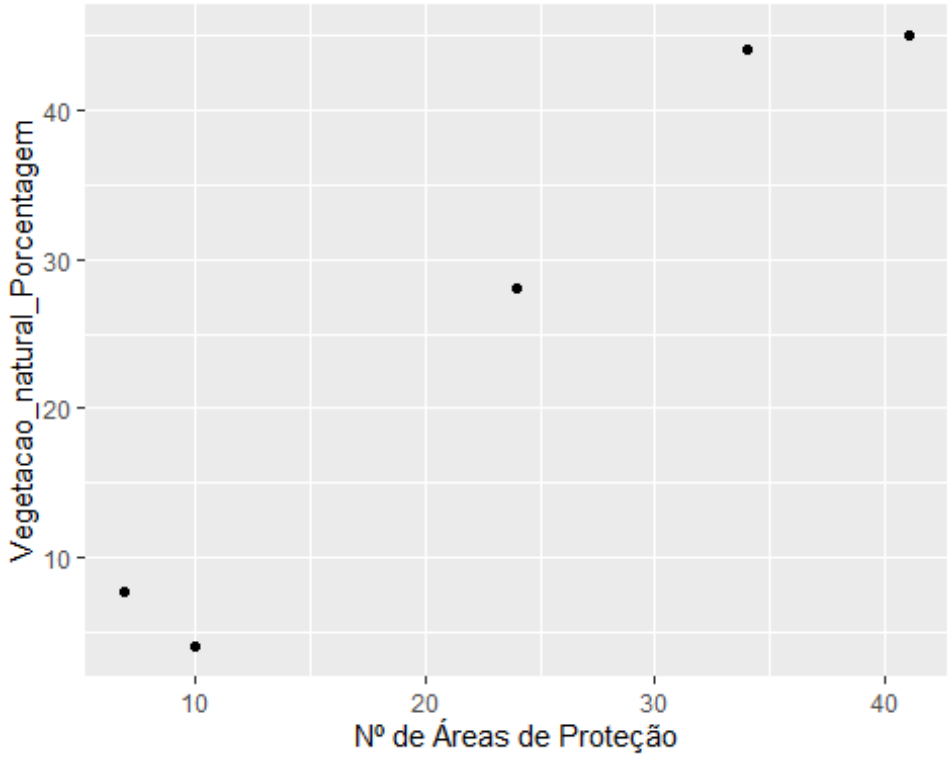


O termo **aes** é derivado do inglês *aestetics*, ou "estética" em inglês. Dessa forma, você acabou de definir os primeiros parâmetros que darão a "estética" inicial do seu gráfico.

Definindo os formatos geométricos para visualizar seus dados

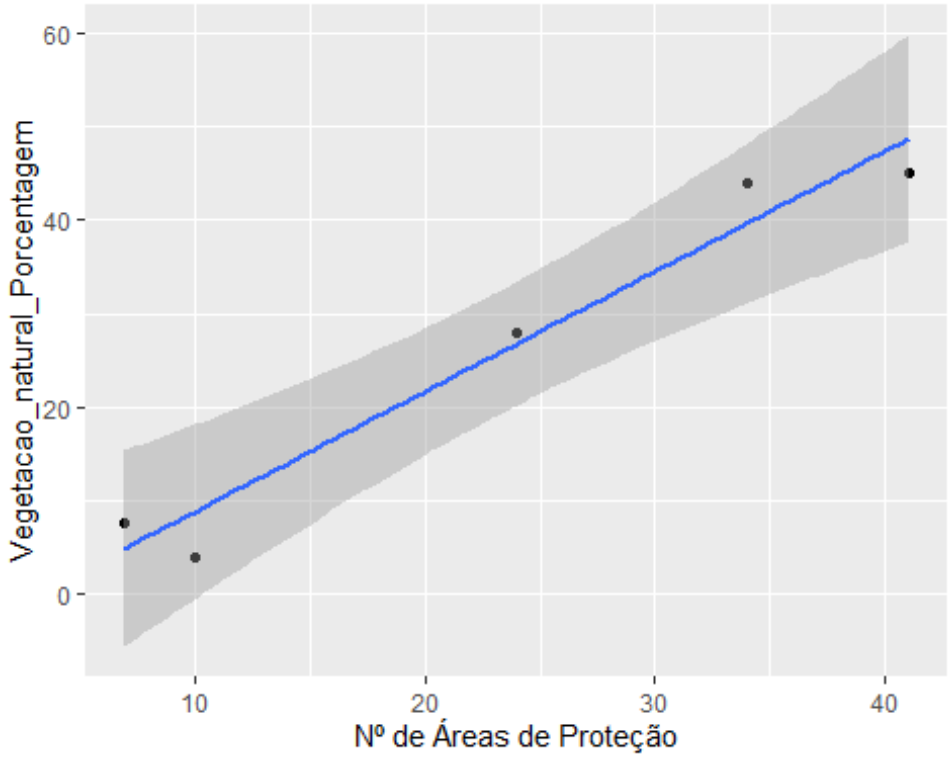
Na etapa anterior, foram estabelecidas as variáveis preditora e resposta, as quais estão de fato ilustradas nos eixos do *plot*. Porém, ainda não vimos os dados em si e a relação entre eles! Isso pode ser feito com diferentes formatos geométricos: pontos, linhas de tendência, barras, *boxplot*, entre outros! Tudo depende da natureza dos seus dados (se categóricos ou contínuos) e da sua criatividade! Como nossas variáveis são categóricas, iremos utilizar um *scatterplot*, ou "gráfico de dispersão", para visualizar nossos dados.

```
  ggplot(dados_relacionais_uso_solo_aps_rio,aes(x = `Nº de Áreas de
Proteção`, y = `Vegetacao_natural_Porcentagem`))+
  geom_point()
```



É possível constatar uma relação positiva entre nossas variáveis! Caso queira tornar essa relação ainda mais aparente, você pode criar uma linha de tendência, por meio da sua *adição* ao sistema do **ggplot**.

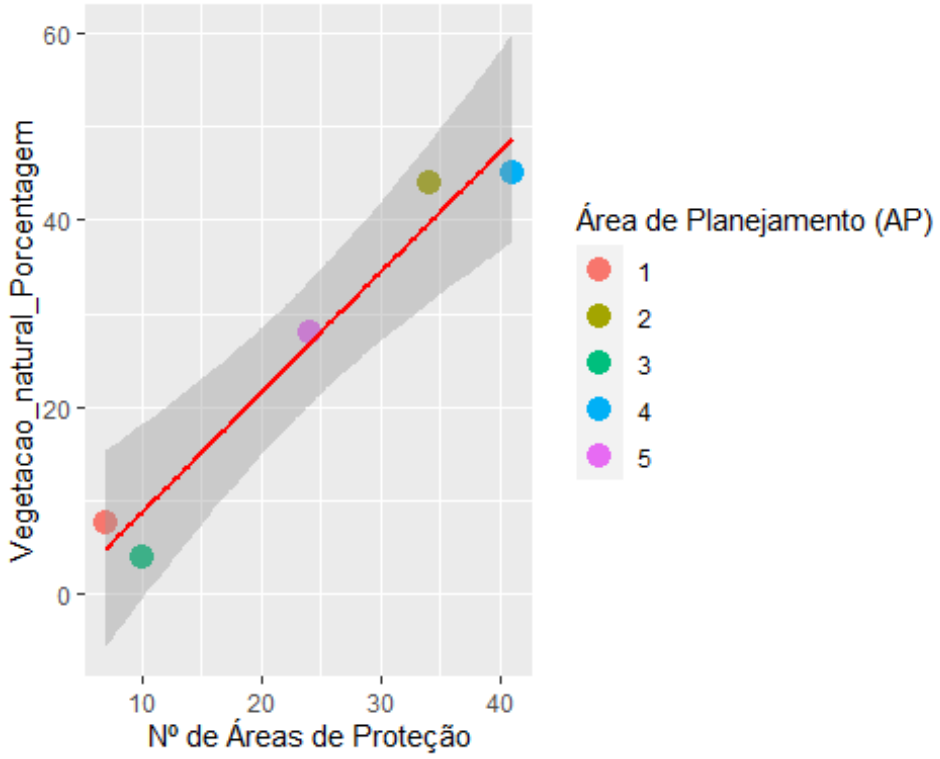
```
ggplot(dados_relacionais_uso_solo_aps_rio,aes(x = `Nº de Áreas de
Proteção`, y = `Vegetacao_natural_Porcentagem`))+
  geom_point()+
  geom_smooth(method = lm)
```



Legendas, destaques e anotações que ajudem você a comunicar seus dados

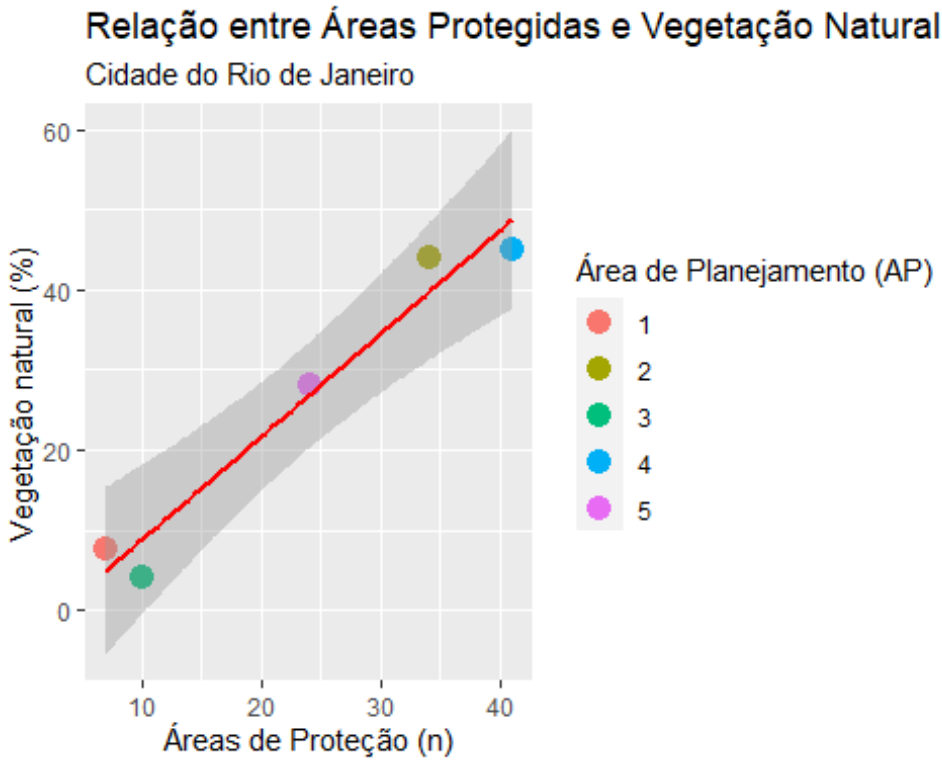
Os elementos que criamos até então para nossos gráficos são os chamados elementos obrigatórios: sem eles, não há construção de sentido dentro da sintaxe do **ggplot**. Todos os elementos adicionados a partir de então são opcionais e podem variar de acordo com o que você deseja comunicar com suas análises ou seus dados. Seria interesse, por exemplo, que fôssemos capazes de ilustrar a qual Área de Planejamento da cidade cada ponto do nosso *plot* pertence, além de dar mais destaque à nossa linha de tendência e aos nossos próprios pontos.

```
ggplot(dados_relacionais_uso_solo_aps_rio,aes(x = `Nº de Áreas de
Proteção`, y = `Vegetacao_natural_Porcentagem`))+
  geom_point(aes(color=`Área de Planejamento (AP)`),size=4)+
  geom_smooth(color="red",linewidth=1,method = lm)
```

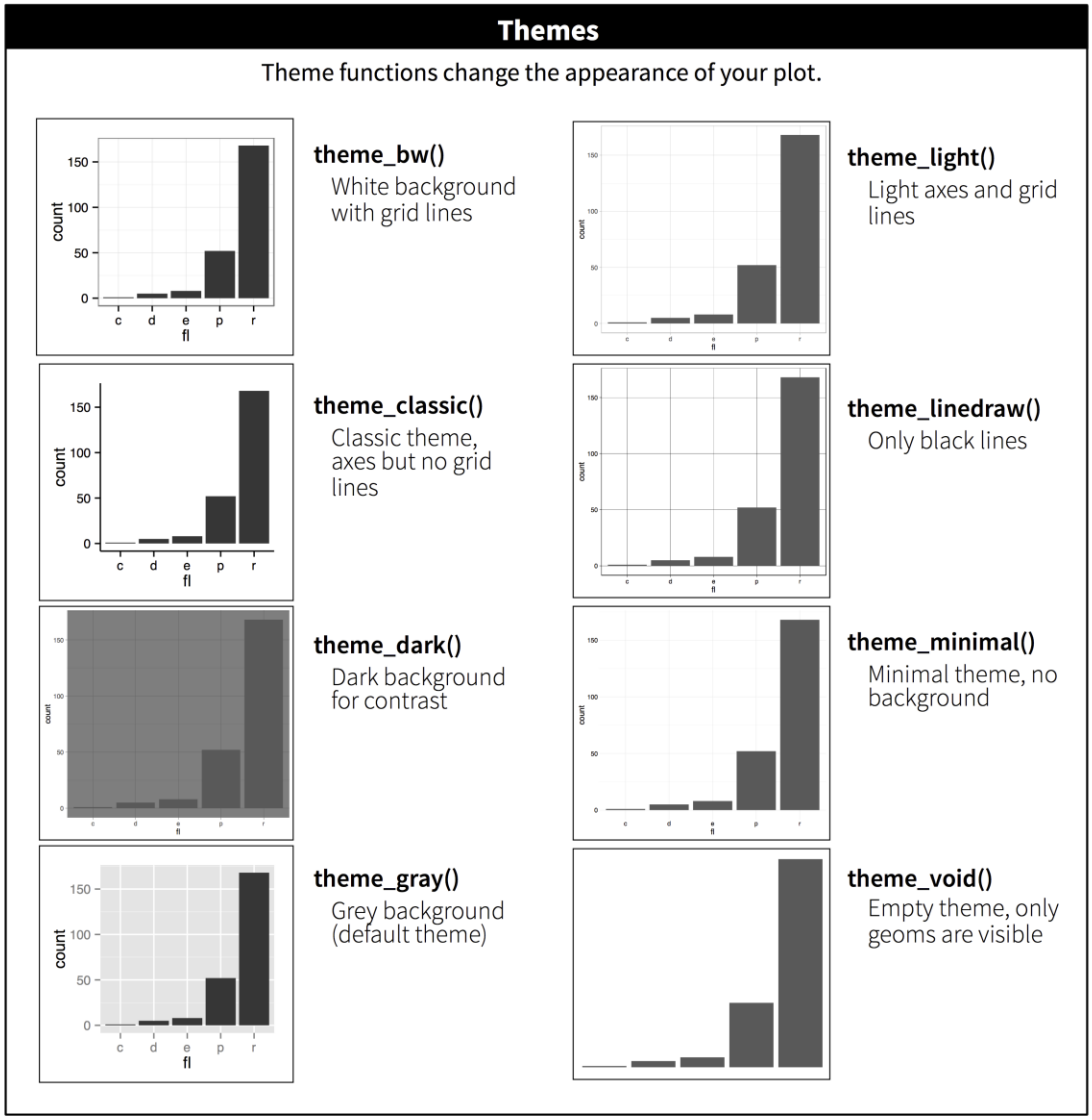


A aparência do nosso gráfico já está bem mais cativante! Mas nossas legedas ainda estão de acordo com os nomes do nosso *dataframe* original, além de ainda não termos um título informativo. Para isso, utilizamos a função **labs**, derivada do inglês *labels*, que significa "rótulos", ou "legendas".

```
ggplot(dados_relacionais_uso_solo_aps_rio,aes(x = `Nº de Áreas de
Proteção`, y = `Vegetacao_natural_Porcentagem`))+
  geom_point(aes(color=`Área de Planejamento (AP)`),size=4)+
  geom_smooth(color="red",linewidth=1,method = lm)+
  labs(title = "Relação entre Áreas Protegidas e Vegetação Natural",
       subtitle = "Cidade do Rio de Janeiro",
       x="Áreas de Proteção (n)",
       y="Vegetação natural (%)")
```



Mais melhoramentos!! Mas o fundo cinza e o *grid* branco em contraste podem incomodar alguns leitores. Para isso, existem diferentes formatos de tema/estilo do **ggplot**:

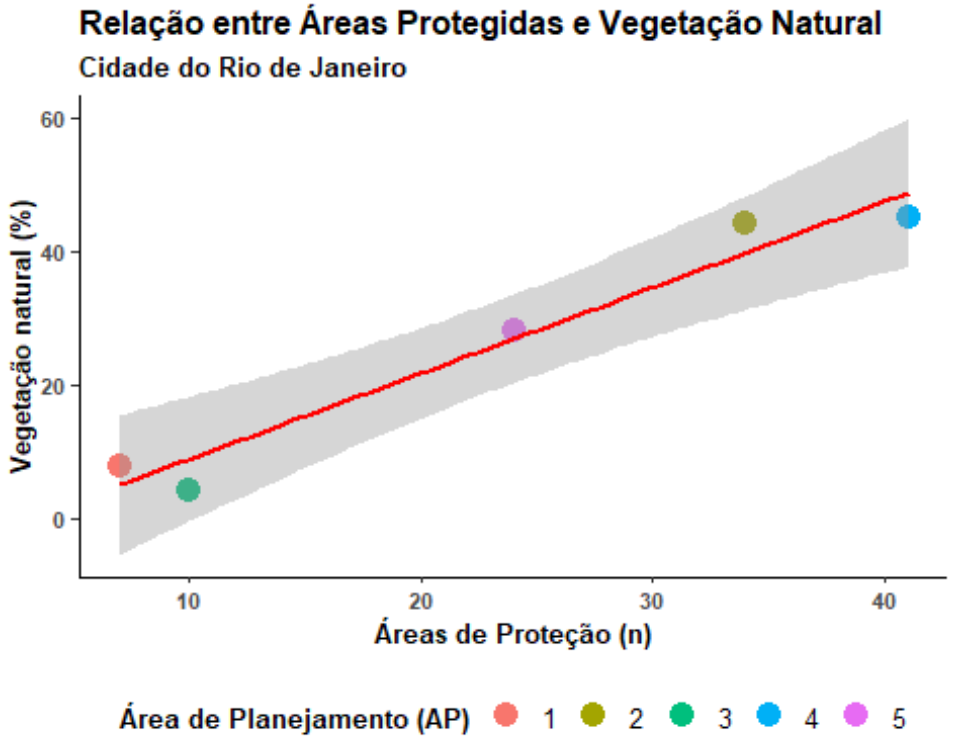


Esquema do livro "R for Data Science"

Vamos aplicar o tema clássico! Finalmente, também podemos modificar mais um pouco nosso gráfico, alterando os tamanhos das fontes das nossas legendas e rótulos dos eixos, e dos nossos títulos e subtítulos. Isso é feito através da função **theme**. Com essa função, também podemos alterar a posição da nossa legenda das Áreas de Planejamento, que está ocupando muito espaço na área do nosso *plot*. **IMPORTANTE**: caso haja modificação do tipo de tema/estilo do gráfico, ela deve ser feita *antes* da aplicação da função **theme**, como ilustrado abaixo.

```
ggplot(dados_relacionais_uso_solo_aps_rio,aes(x = `Nº de Áreas de
Proteção`, y = `Vegetacao_natural_Porcentagem`))+
  geom_point(aes(color=`Área de Planejamento (AP)`),size=4)+
  geom_smooth(color="red",linewidth=1,method = lm)+
  labs(title = "Relação entre Áreas Protegidas e Vegetação Natural",
```

```
      subtitle = "Cidade do Rio de Janeiro",
      x="Áreas de Proteção (n)",
      y="Vegetação natural (%)")+
theme_classic()+
theme(title = element_text(face = "bold",size = 10),
      axis.title.x = element_text(face="bold",size=10),
      axis.title.y = element_text(face="bold",size=10),
      axis.text.x = element_text(face="bold",size=8),
      axis.text.y = element_text(face="bold",size=8),
      legend.title = element_text(face="bold",size=10),
      legend.text = element_text(size=10),
      legend.position = "bottom")
```

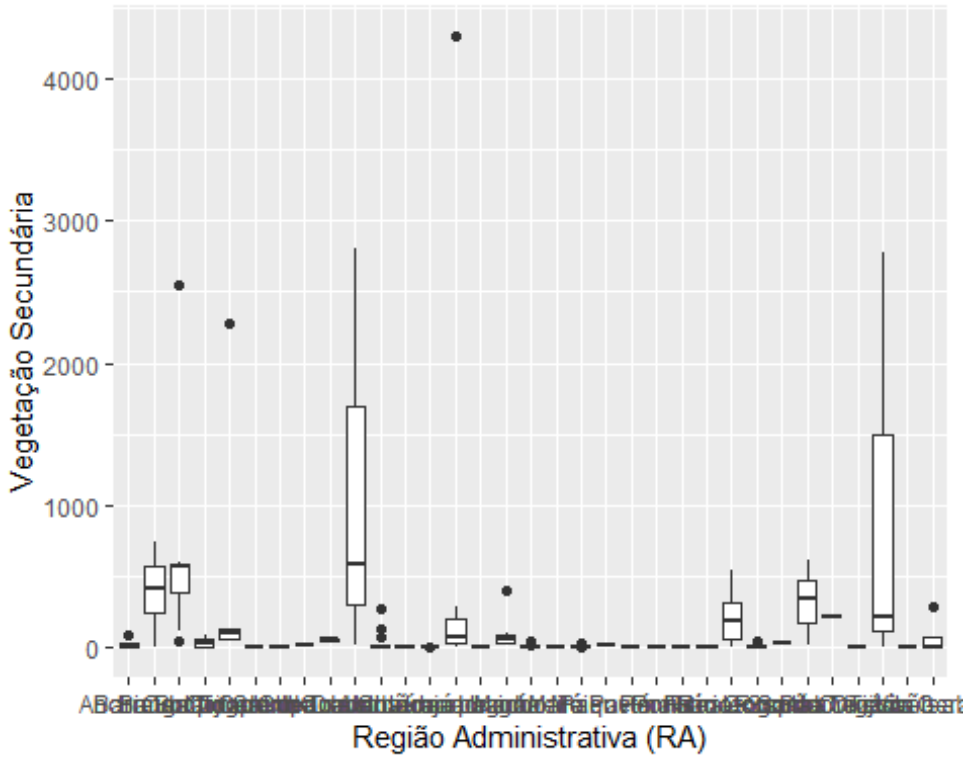


Pronto! Evoluímos bastante nosso gráfico desde o resultado inicial, não é? Agora, vamos explorar outros tipos de combinação de dados com as funcionalidades do **ggplot**.

Dados contínuos X categóricos

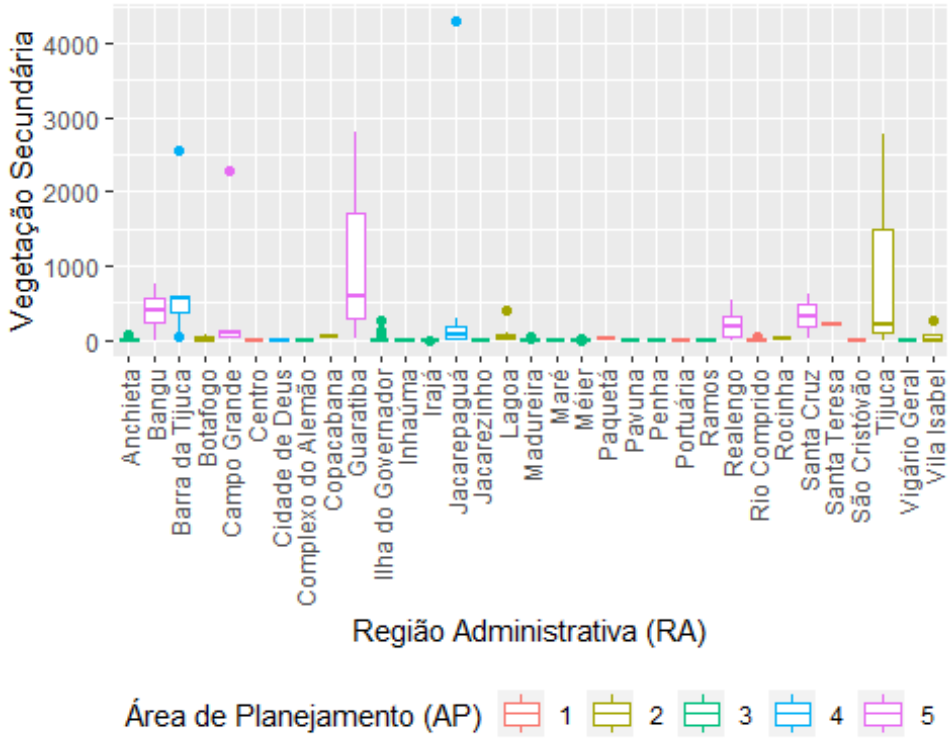
É muito comum utilizar gráficos com *boxplots* quando fazemos uma ANOVA, ou quando criamos um modelo linear generalizado (GLM) com variáveis preditoras categóricas. A função **geom_boxplot** permite que adicionemos esse tipo de formato geométrico para ilustrar nossos dados. Vamos aplica-lá para saber como a quantidade de Vegeteção Secundária (em hectares) está relacionada às Regiões Administrativas da cidade do Rio de Janeiro. Esses dados estão disponibilizados no *dataframe* referente ao uso de solo na cidade.

```
ggplot(uso_solo_rio,aes(x=`Região Administrativa (RA)`,y=`Vegetação
Secundária`))+
  geom_boxplot()
```



Os nomes das Regiões Administrativas estão sobrepostos, dificultando a sua visualização. Vamos mudar seu ângulo de orientação para 90 graus dentro da função **theme**. Além disso, podemos inserir uma noma camada de informação ao nosso *plot*: como cada Região Administrativa encontra-se dentro de uma Área de Planejamento, podemos assinalar cada *boxplot* de uma Região a uma cor distinta. Essas cores irão corresponder às Áreas de Planejamento às quais cada Região pertence, o que será feito novamente pelo parâmetro **aes** do **geom_boxplot**. Finalmente, iremos alocar a legenda novamente na parte de baixo do gráfico.

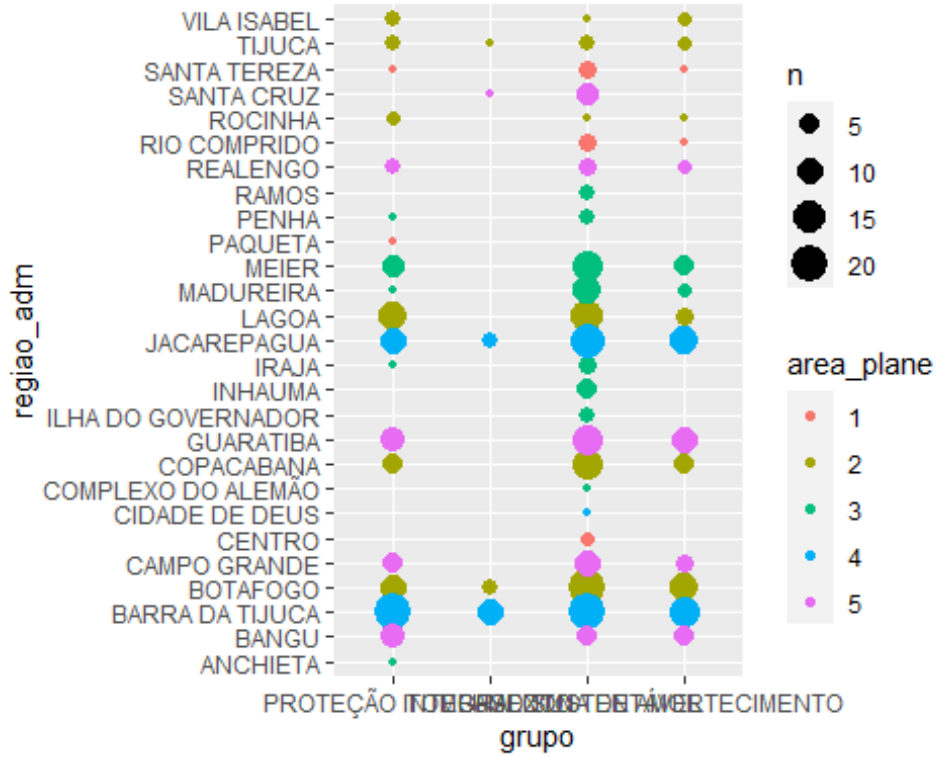
```
ggplot(uso_solo_rio,aes(x=`Região Administrativa (RA)`,y=`Vegetação
Secundária`))+
  geom_boxplot(aes(color=`Área de Planejamento (AP)`))+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        legend.position = "bottom")
```



Dados categóricos X categóricos

Alguns exemplos de análises envolvendo variáveis preditora e resposta categóricas são teste de qui-quadrdo e modelos lineares generalizados (GLM). Sua visualização pode ser feita por meio da função **geom_count**, que iremos aplicar a nossos dados de Áreas de Proteção no Rio de Janeiro. Nosso interesse é de compreender a quantidade de Áreas de Proteção dentro de cada Região Administrativa da cidade, de acordo com o seu tipo de uso (Uso Sustentável, Proteção Integral, etc.). Vamos também ilustrar como isso está distribuído dentro de cada Área de Planejamento do Rio de Janeiro. Uma maneira de integrar os pacotes de organização e transformação de dados do Tidyverse ao **ggplot** é por meio do operador de *pipelines*, como ilustrado abaixo.

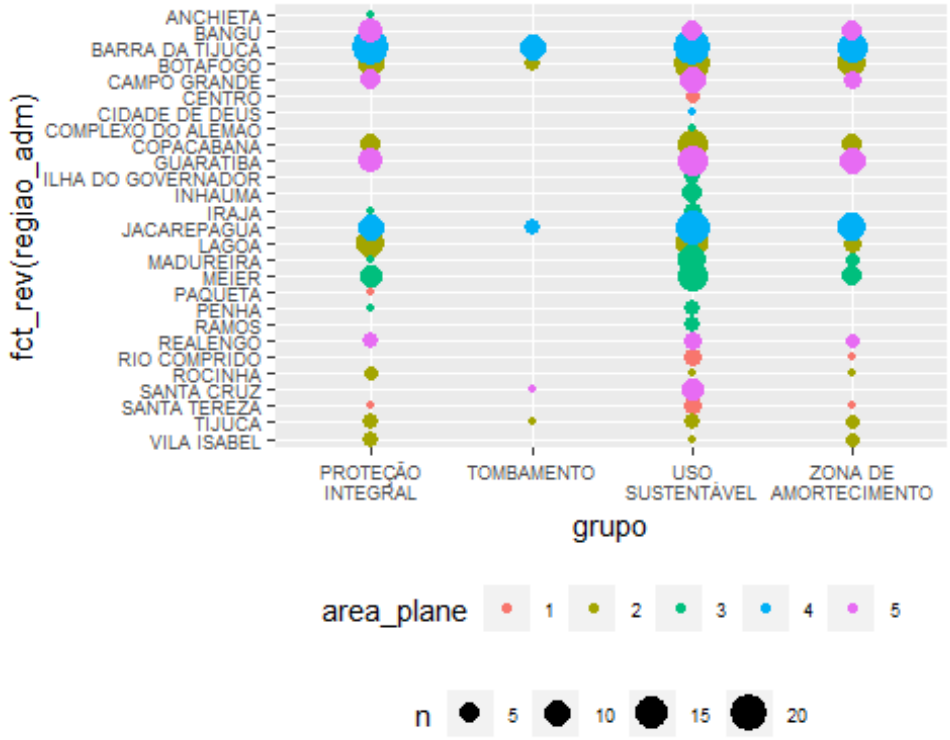
```
aps_rio%>%
  ggplot(aes(x=grupo,y=regiao_adm))+
  geom_count(aes(color=area_plane))
```



Perceba que as categorias de tipo de uso estão sobrepostas umas às outras, dificultando nossa visualização. Vamos "quebrar" esses nomes para caber em nosso gráfico de forma mais organizada através da função **scale_x_discrete** do **ggplot**. Dentro desta função, chamaremos a função **label_wrap**, do pacote **scales**. Você irá fazer isso por meio do operador (::), como você aprendeu na aula passada.

Repare, ainda, que os nomes das Regiões Administrativas estão em ordem alfabética invertida. Vamos corrigir esses problemas com a função **fct_rev**, derivada do inglês *factor reverse*, ou "reversão de fatores", dentro da estética do seu *plot*. Finalmente, iremos alocar as legendas para a parte de baixo do gráfico, empilhadas de forma vertical, através da modificação dos parâmetros da função **theme**.

```
aps_rio%>%
  ggplot(aes(x=grupo,y=fct_rev(regiao_adm)))+
  geom_count(aes(color=area_plane))+
  scale_x_discrete(labels = scales::label_wrap(10))+
  theme(axis.text.x = element_text(size=7),
        axis.text.y = element_text(size=7),
        legend.text = element_text(size=7),
        legend.position = "bottom",
        legend.box = "vertical")
```



Pronto! Criamos mais uma visualização que pode ajudar muito os tomadores de decisão na Prefeitura do Rio de Janeiro. Você pode salvar as suas figuras individualmente por meio da função **ggsave**. Ela é muito útil, pois permite que você possa salvar um mesmo gráfico em diferentes tipos de arquivo, com diferentes tipos de resolução e dimensão. O *default* da função é de sempre salvar o último *plot* criado na área de trabalho do R.

```
ggsave("Uso de Áreas de Proteção nas Regiões Administrativas do Rio de Janeiro.png",dpi = 600)
ggsave("Uso de Áreas de Proteção nas Regiões Administrativas do Rio de Janeiro.pdf")
```

Exercício - E agora?

As possibilidades de criação de visualização com o **ggplot** são imensas. Na internet, é possível acessar vários materiais gratuitos que lhe auxiliarão na descoberta do universo deste pacote. Um dos mais úteis é o *ggplot cheat sheet*, do inglês "tabela para trapacear o ggplot". Como exercício, sugerimos que você explore esses materiais e os dados desta aula para criar uma visualização completa, com título e legendas, que você considere como importante para o planejamento e gestão ambientais na cidade do Rio Janeiro. Você irá contar com a ajuda das professoras e monitoras para essa tarefa! Salve sua visualização com o auxílio do **ggsave**.