Programação em R

Julho, 2016

Base R: desenhando seus gráficos

Dados

Vamos resgatar nossa base de dados de imóveis e passar alguns filtros.

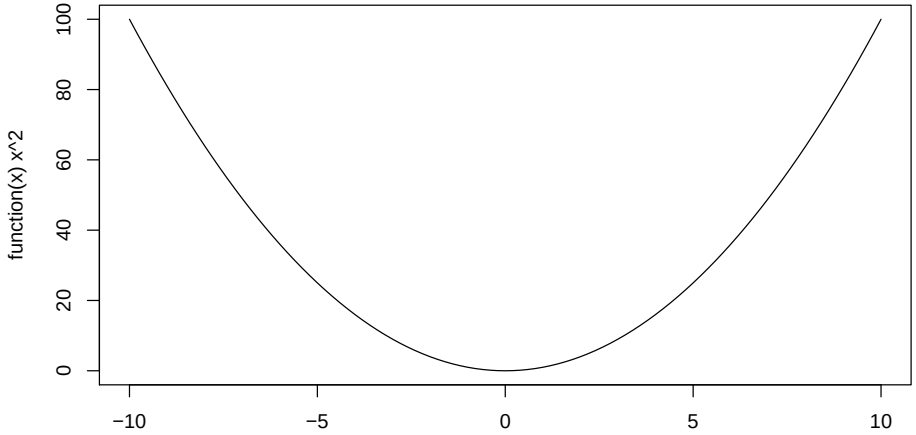
```
library(ggplot2)
library(dplyr)
limpos <- readRDS("Dados/limpos.rds")
venda <- limpos %>%
         filter(tipo == "venda",
                quartos < 10,
                imovel == "apartamento",
                bairro %in% c("Asa Sul","Asa Norte",
                              "Noroeste","Sudoeste"))
```

Função plot

A função plot é uma função genérica, e possui vários métodos diferentes dependendo do tipo de objeto.

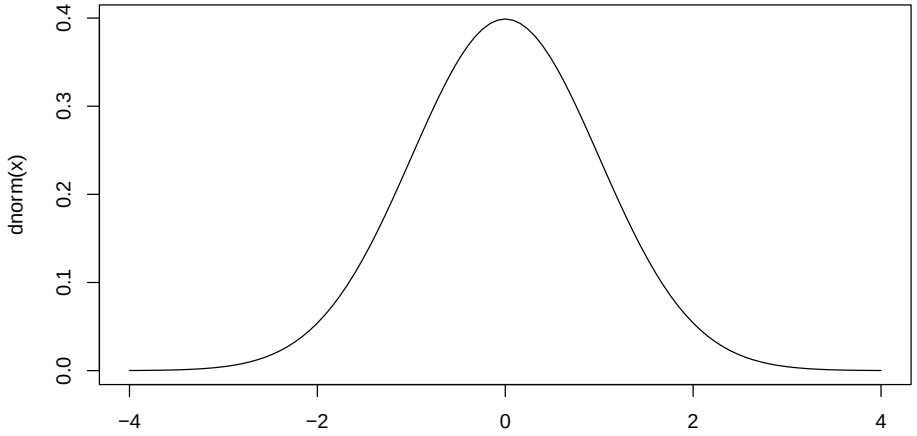
Plot de função

```
plot(function(x) x^2, from = -10, to = 10)
```



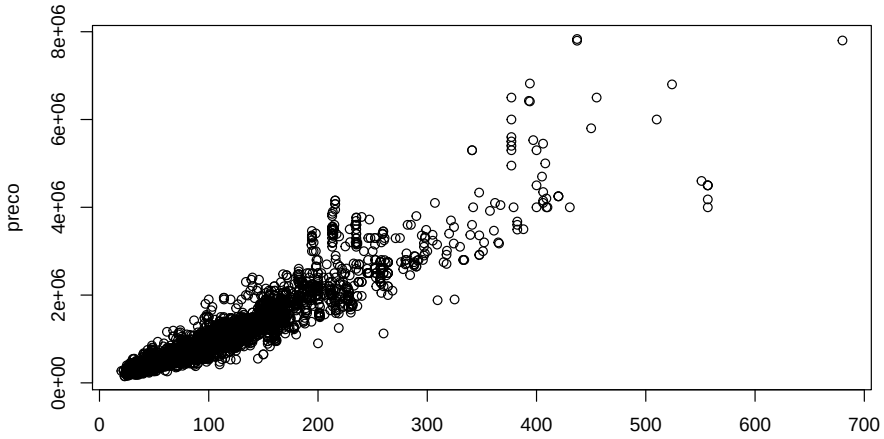
Plot de expressão

```
curve(dnorm(x), -4, 4)
```



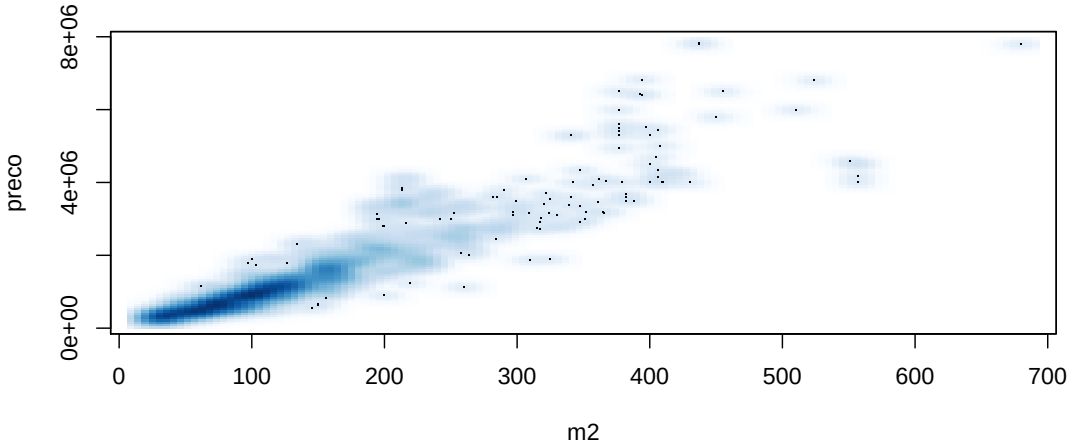
Scatter Plot

```
with(venda, plot(m2, preco))
```



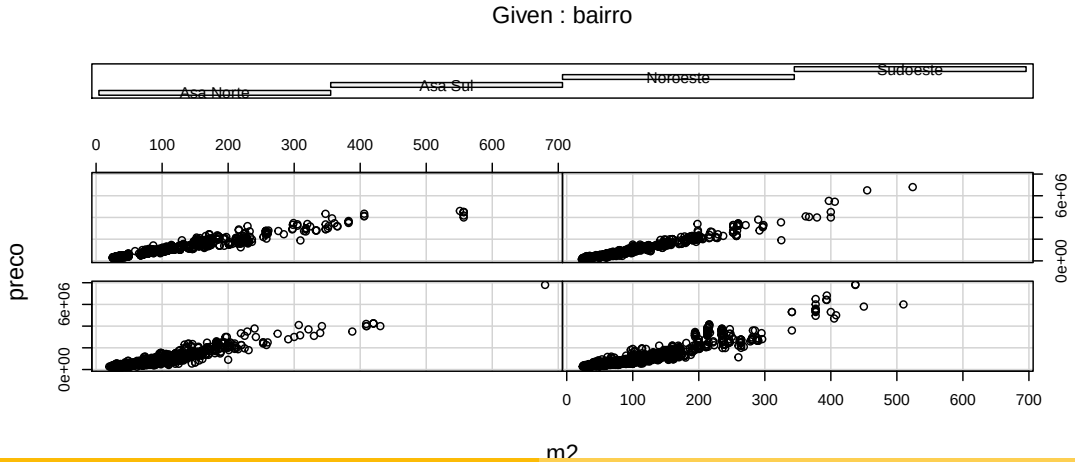
Smooth Scatter Plot

```
with(venda, smoothScatter(m2, preco))
```



Coplot

```
with(venda, coplot(preco ~ m2 | bairro))
```

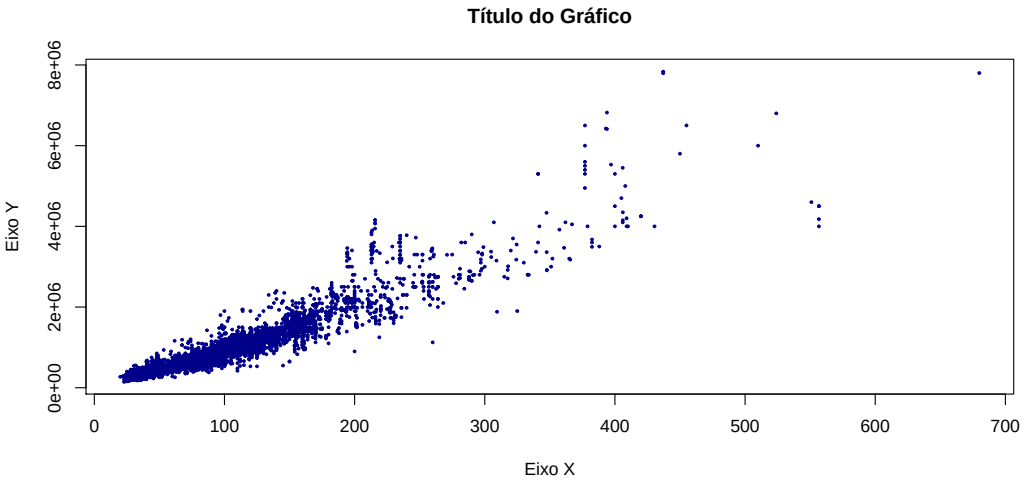


Personalizando parâmetros

Para ver os parâmetros disponíveis, digite ?par:

```
with(venda, plot(m2, preco,
                 col = "darkblue",
                 main = "Título do Gráfico",
                 xlab = "Eixo X",
                 ylab = "Eixo Y",
                 pch = 20,
                 cex = 0.5))
```

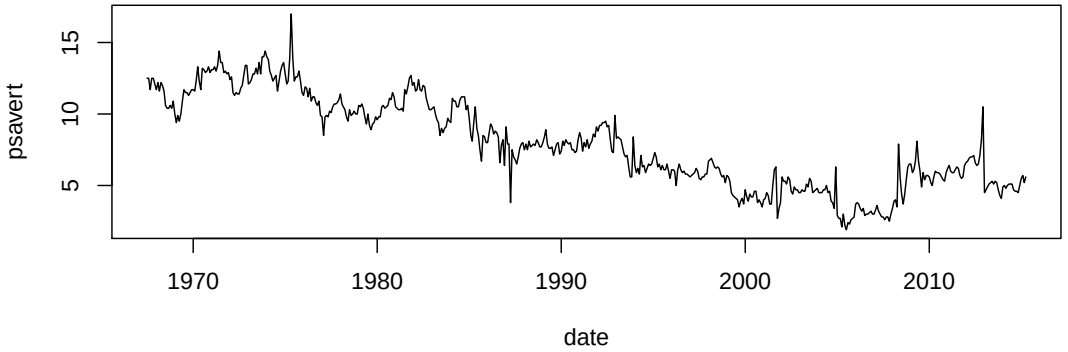
Personalizando parâmetros



Line plot

Basta mudar `type="l"`. Note que o gráfico pode ter notação em formula também:

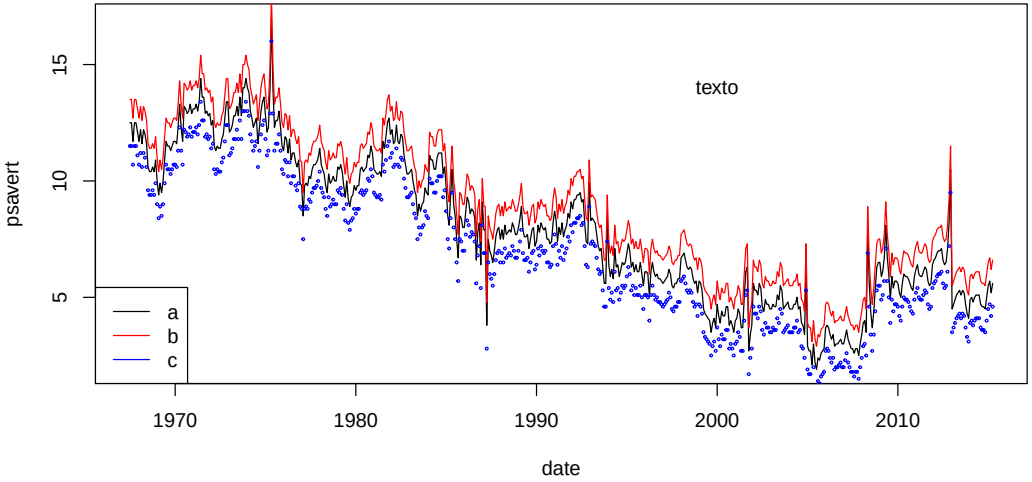
```
library(ggplot2) # para a base de dados economics
plot(psavert ~ date, type = "l", data = economics)
```



Adicionando linhas, pontos, textos e legenda

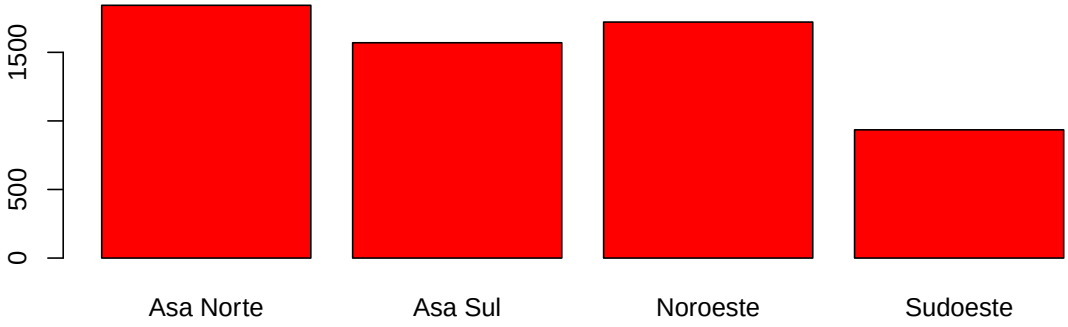
```
lines(economics$date, economics$psavert + 1, col = "red")

points(economics$date, economics$psavert - 1, col = "blue", cex = 0.3)

text(x = as.Date("2000-01-01"), y = 14, "texto")

legend("bottomleft", col=c("black", "red", "blue"),
       legend = c("a", "b", "c"), lty = 1)
```

Adicionando linhas, pontos, textos e legenda



Barplot

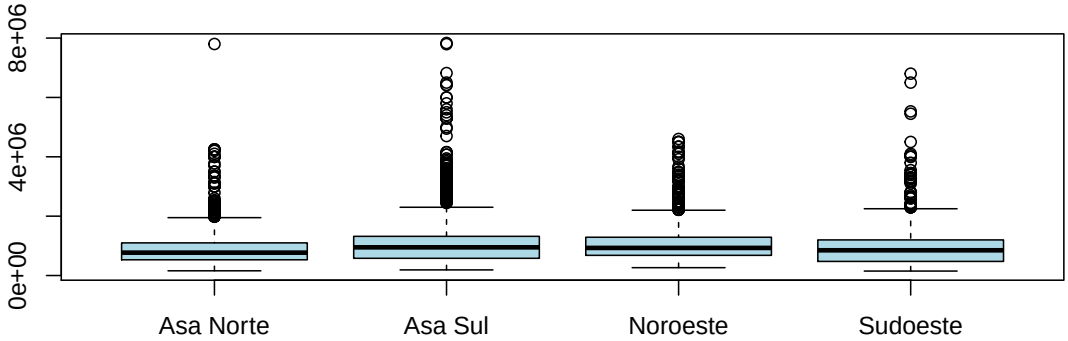
```
barplot(table(venda$bairro), col = "red")
```



Boxplot

Função `boxplot()`:

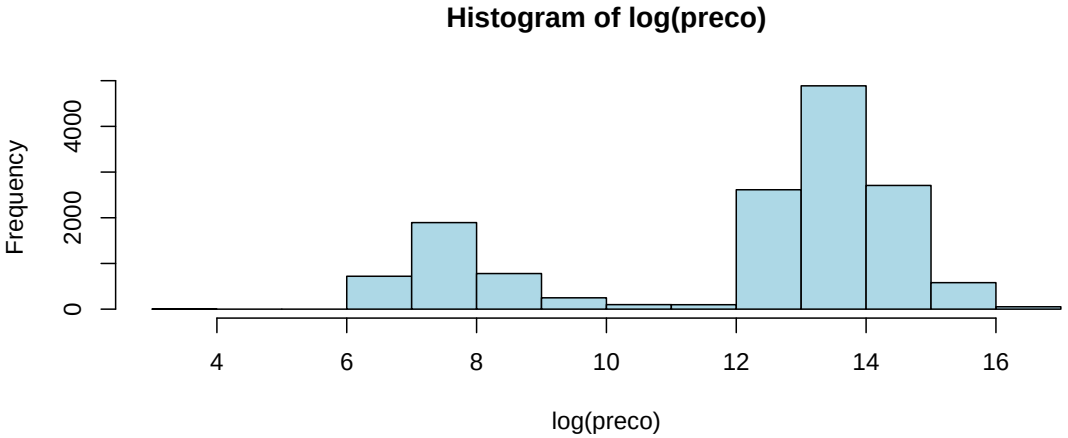
```
boxplot(preco ~ bairro, data = venda, col = "lightblue")
```



Histograma

Função `hist()`:

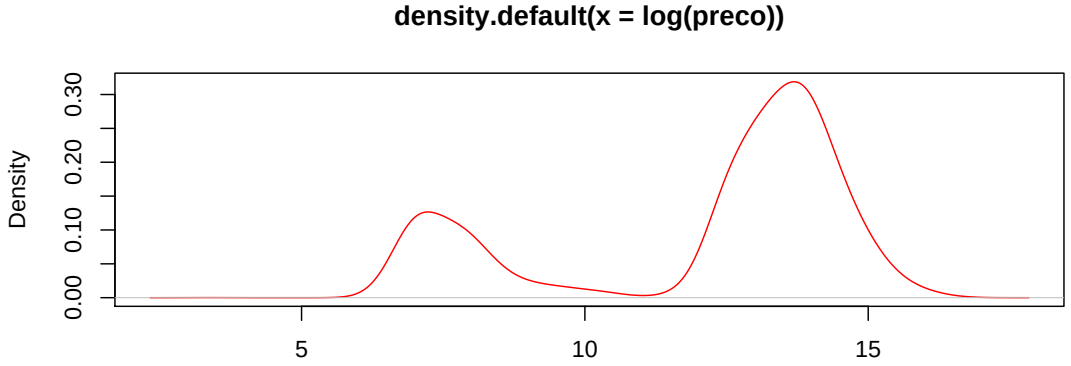
```
histograma <- with(limpos, hist(log(preco), col = "lightblue"))
```



Densidade

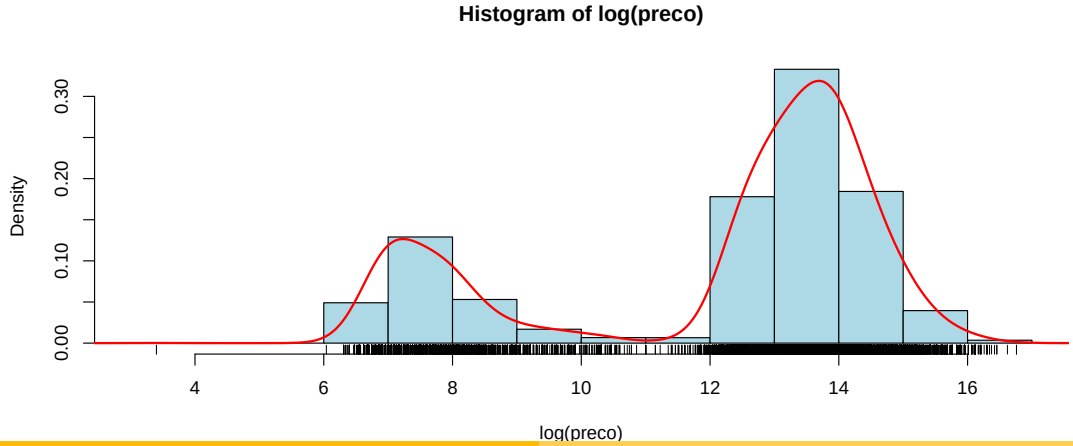
Calculando e plotando a densidade:

```
densidade <- with(limpos, density(log(preco)))
plot(densidade, col = "red")
```



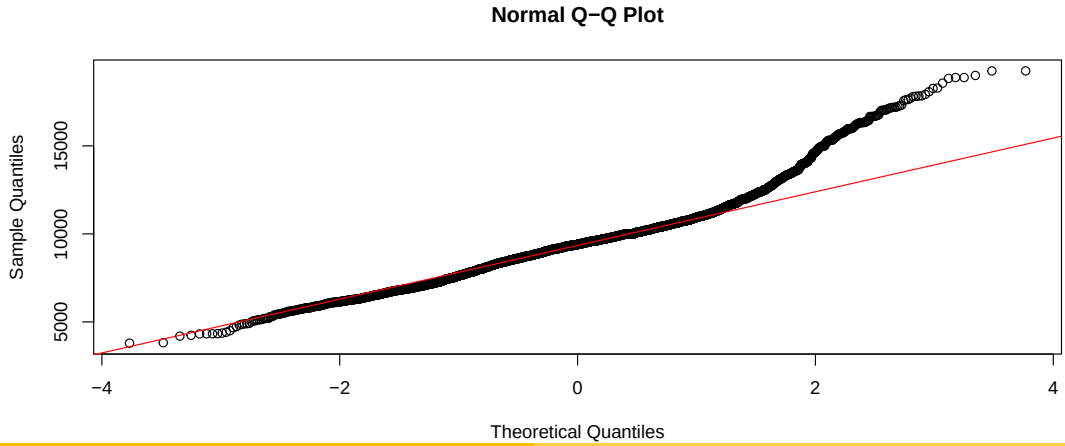
Histograma mais densidade e rug

```
plot(histograma, col = "lightblue", freq = FALSE)
lines(densidade, col = "red", lwd = 2); rug(log(limpos$preco))
```



Quantile plots

```
qqnorm(venda$pm2)
qqline(venda$pm2, col="red")
```



Vários gráficos ao mesmo tempo

Para plotar vários gráficos ao mesmo tempo, utilizar `par(mfrow=c(n,m))` ou `par(mfcol=c(n,m))`

ggplot2: Uma gramática de gráficos

Utilizando gráficos para explorar sua base de dados

Os gráficos base do R são bastante poderosos e com eles é possível fazer muita coisa. Entretanto, eles podem ser um pouco demorados para explorar dinamicamente sua base de dados. O pacote `ggplot2` é uma alternativa atraente para resolver este problema. O `ggplot2` é um pouco diferente de outros pacotes gráficos pois não segue a lógica de desenhar elementos na tela; ao invés disso, a sintaxe do `ggplot2` segue uma "gramática de gráficos estatísticos" baseada no Grammar of Graphics de Wilkinson (2005).

Utilizando gráficos para explorar sua base de dados

No começo, pode parecer um pouco diferente essa forma de construir gráficos. Todavia, uma aprendidos os conceitos básicos da gramática, você vai pensar em gráficos da mesma forma que pensa numa análise de dados, construindo seu gráfico iterativamente, com visualizações que ajudem a revelar padrões e informações interessantes gastando poucas linhas de código. É um investimento que vale a pena.

Utilizando gráficos para explorar sua base de dados

Antes de continuar, você precisa instalar e carregar os pacotes que vamos utilizar nesta seção. Além do próprio `ggplot2`, vamos utilizar também os pacotes `ggthemes` e `gridExtra`.

```
# Instalando os pacotes (caso não os tenha instalados)
install.packages(c("ggplot2", "ggthemes", "gridExtra"))

# Carregando os pacotes
library(ggplot2)
library(ggthemes)
library(gridExtra)
```

A "gramática dos gráficos"

Mas o que seria essa gramática de gráficos estatísticos? Podemos dizer que um gráfico estatístico é um **mapeamento** dos dados para propriedades **estéticas** (cor, forma, tamanho) e **geométricas** (pontos, linhas, barras) da tela. O gráfico também pode conter **transformações estatísticas** e múltiplas **facetas** para diferentes subconjuntos dos dados. É a combinação de todas essas **camadas** que forma seu gráfico estatístico. Deste modo, os gráficos no `ggplot2` são construídos por meio da **adição de camadas**. Cada gráfico, *grosso modo*, é composto de:

- Uma base de dados (um data.frame, preferencialmente no formato **long**);
- Atributos estéticos (**aes**thetics);
- Camadas, contendo
  - Objetos **geom**étricos;
  - Transformações e**stat**ísticas;
- **Facet**as; e,
- Demais ajustes.

aes: x e y - geom_point()

Vejamos um exemplo simples de **diagrama de dispersão** com os dados de preço e metro quadrado dos imóveis da nossa base de dados.

```
ggplot(data = venda, aes(x = m2, y = preco)) + geom_point()
```

Traduzindo o comando acima do `ggplot2`, nós começamos chamando a função `ggplot()` que inicializa o gráfico com os seguintes parâmetros:

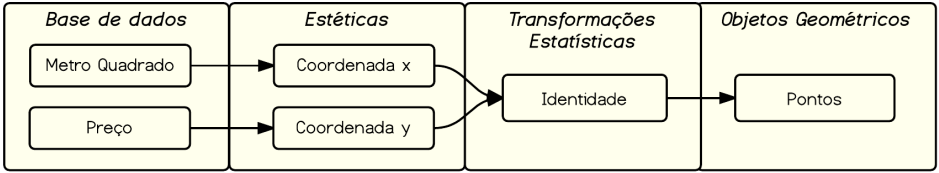
- **data**: aqui indicamos que estamos usando a base de dados `venda`;
- **aes**: aqui indicamos as **estéticas** que estamos mapeando. Mais especificamente, estamos dizendo que vamos mapear o eixo x na variável `m2` e o eixo y na variável `preco`.

Em seguida, adicionamos um objeto geométrico:

- **geom_point()**: estamos falando ao `ggplot` que queremos adicionar o ponto como objeto geométrico.

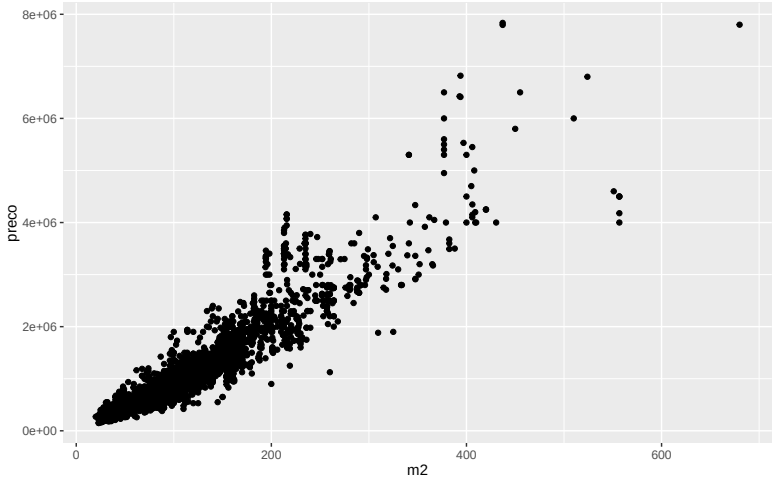
aes: x e y - geom_point()

Com relação às transformações estatísticas, neste caso não estamos realizando nenhuma. Isto é, estamos plotando os dado sem quaisquer modificações. Em termos esquemáticos, nós estamos fazendo o seguinte mapeamento:



aes: x e y - geom_point()

Como resultado, temos:



Outros geoms: pontos, retas, boxplots, regressões

Vimos como exemplo o `geom_poin()`, mas o `ggplot2` vem com vários **geoms** diferentes e abaixo listamos os mais utilizados:

Tipo de Gráfico	**geom**
scatterplot (gráfico de dispersão)	`geom_point()`
barchart (gráfico de barras)	`geom_bar()`
boxplot	`geom_boxplot()`
line chart (gráfico de linhas)	`geom_line()`
histogram (histograma)	`geom_histogram()`
density (densidade)	`geom_density()`
smooth (aplica modelo estatístico)	`geom_smooth()`

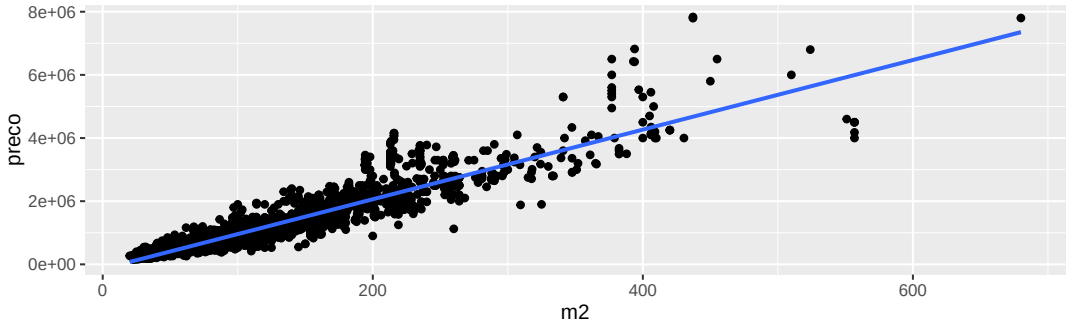
Outros geoms - experimente

```
# Scatter plot
ggplot(data = venda, aes(x = m2, y = preco)) + geom_point()

# Line plot
ggplot(data = venda, aes(x = m2, y = preco)) + geom_line()

# Histogram
ggplot(data = venda, aes(x = preco)) + geom_histogram()

# Density
ggplot(data = venda, aes(x = preco)) + geom_density()

# Boxplot
ggplot(data = venda, aes(x = bairro, y = preco)) + geom_boxplot()

# Smoother (lm, loess, gam)
ggplot(data = venda, aes(x = m2, y = preco)) + geom_smooth(method = "lm")
```

Combinando geoms

Os geoms podem sem combinados:

```
# pontos mais reta de regressão
ggplot(data = venda, aes(x = m2, y = preco)) + geom_point() +
  geom_smooth(method = "lm")
```



aes: mapeando cor, tamanho, forma etc

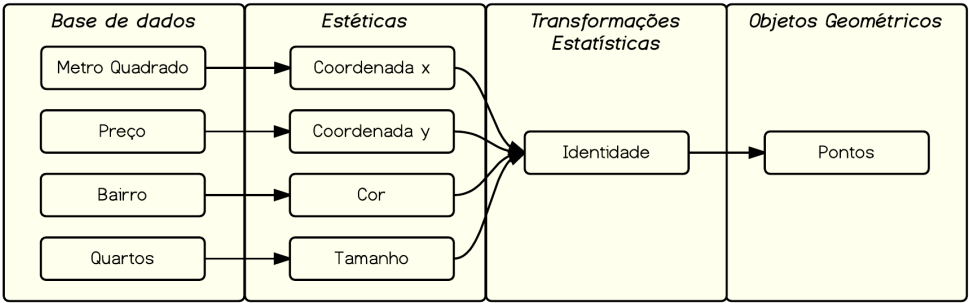
Um gráfico no plano tem apenas duas coordenadas, `x` e `y`, mas nossa base de dados tem, em geral, vários colunas... como podemos representá-las? Uma forma de fazer isso é mapear variáveis em outras propriedades estéticas do gráfico, tais como **cor**, **tamanho** e **forma**. Isto é, vamos expandir as variáveis que estamos meapeando nos **aes**thetics.

aes: mapeando cor, tamanho, forma etc

Para exemplificar, vamos mapear cada bairro em uma cor diferente e o número de quartos no tamanho dos pontos.

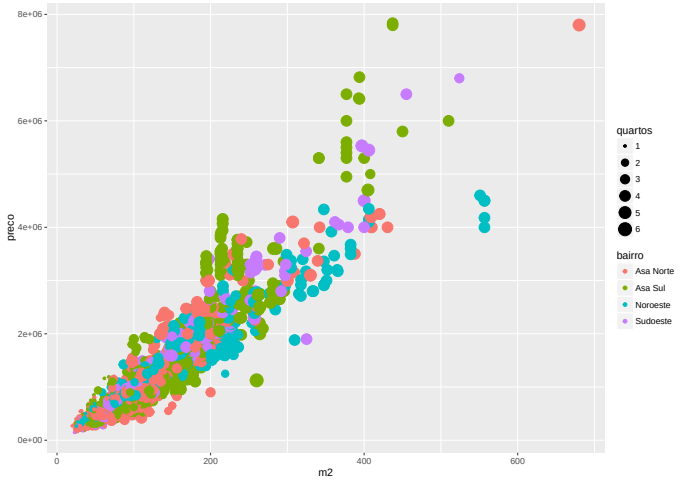
```
ggplot(data = venda, aes(x = m2, y = preco, color = bairro, size = quartos)) +
  geom_point()
```

Nosso esquema ficaria da seguinte forma.



aes: mapeando cor, tamanho, forma etc

E o gráfico resultante:



aes: mapear é diferente de determinar!

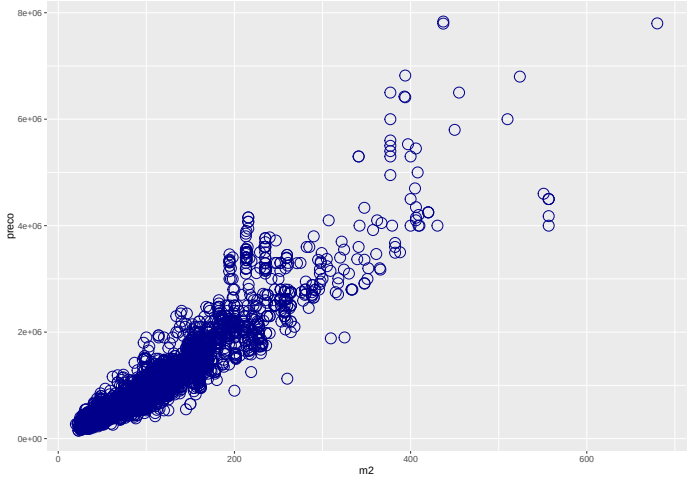
Uma dúvida bastante comum quando as pessoas começam a aprender o `ggplot2` é a diferença entre mapear variáveis em certo atributo estético e determinar certo atributo estético. Quando estamos mapeando variáveis, fazemos isso **dentro** do comando `aes()`. Quando estamos apenas mudando a estética do gráfico, sem vincular isso a alguma variável, fazemos isso **fora** do comando `aes()`.

aes: mapear é diferente de determinar!

Por exemplo, no comando abaixo mudamos a cor, o tamanho e a forma dos pontos do scatter plot. Entretanto, essas mudanças foram apenas cosméticas e não representam informações de variáveis da base de dados e, portanto, não possuem legenda.

```
# muda o tamanho, a cor e a forma dos pontos
# note que não há legenda, pois não estamos
# mapeando os dados a atributos estéticos
ggplot(data = venda, aes(x = m2, y = preco)) +
  geom_point(color = "darkblue", shape = 21, size = 5)
```

aes: mapear é diferente de determinar!



Combinando aes e geom

Os gráficos do `ggplot2` são construídos em etapas e podemos combinar uma série de camadas compostas de **aes** e **geoms** diferentes, adicionando informações ao gráfico iterativamente. Toda informação que você passa dentro do comando inicial `ggplot()` é repassada para os `geoms()` seguintes. Assim, as estéticas que você mapeia dentro do comando `ggplot()` valem para todas as comadas subsequentes; por outro lado, as estéticas que você mapeia dentro dos **geoms** valem apenas para aquele **geom** especificamente. Vejamos um exemplo.

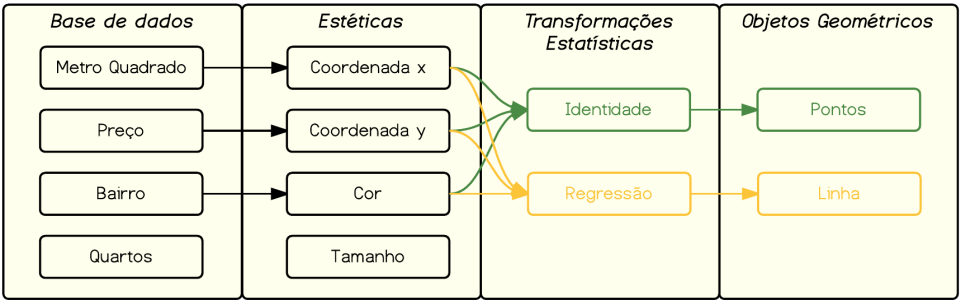
Combinando aes e geom

O comando abaixo mapeia o bairro como cor dentro do comando `ggplot()`. Dessa forma, tanto nos pontos `geom_point()`, quanto nas regressões `geom_smooth()` temos cores mapeando bairros, resultando em várias regressões diferentes.

```
# aes(color) compartilhado
ggplot(venda, aes(m2, preco, color = bairro)) +
  geom_point() +
  geom_smooth(method = "lm")
```

Combinando aes e geom

Vejamos no esquema:



Combinando aes e geom

Resultado:

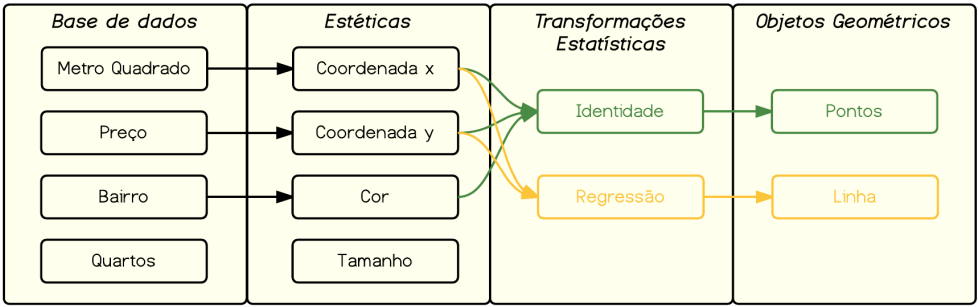
Combinando aes e geom

Mas e se você quisesse manter os pontos com cores diferentes com apenas uma regressão para todas observações? Neste caso, temos que mapear os bairros nas cores **apenas** para os pontos. Note que no comando a seguir passamos a estética `color = bairro` apenas para `geom_poin()`.

```
# aes(color) apenas nos pontos
ggplot(venda, aes(m2, preco)) +
  geom_point(aes(color = bairro)) +
  geom_smooth(method = "lm")
```

Combinando aes e geom

Vejamos no esquema:



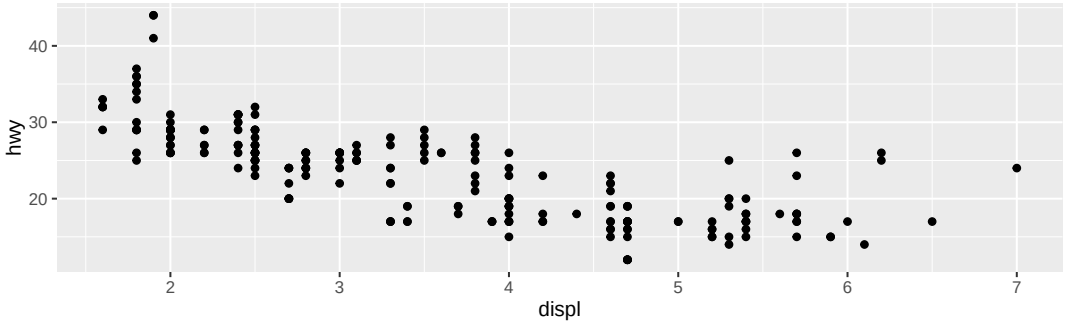
Combinando aes e geom

Resultado:

Cilindradas, cilindros e Milhas por Galão

Um exemplo legal de como um gráfico pode revelar informações. Gráfico simples:

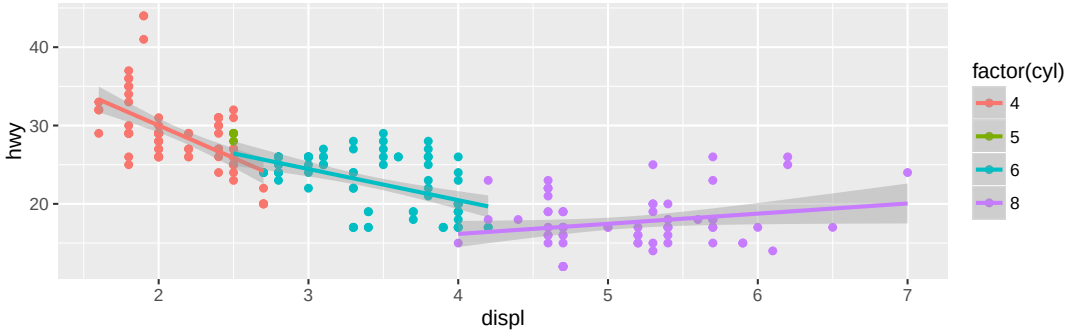
```
ggplot(mpg, aes(displ, hwy)) + geom_point()
```



Cilindradas, cilindros e Milhas por Galão

Gráfico com cor e regressão por cilindro:

```
ggplot(mpg, aes(displ, hwy, col = factor(cyl))) + geom_point() +
  geom_smooth(method = "lm")
```



Adicionando facetas

Você pode dividir o gráfico por variáveis categóricas, usando o `facet_wrap()`.

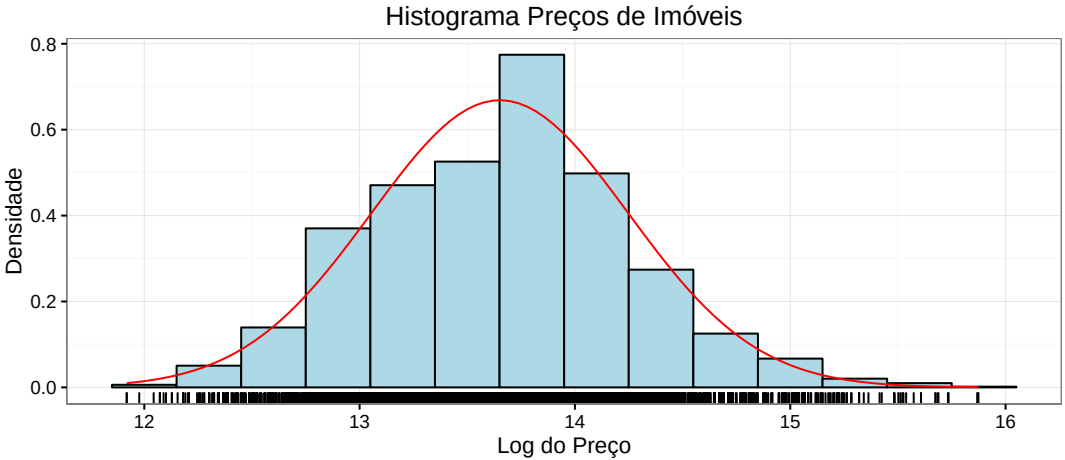
```
ggplot(venda, aes(m2, preco)) +
  geom_point(aes(col = factor(quartos))) +
  geom_smooth(method = "lm") +
  facet_wrap(~bairro)
```

Personalizando mais o gráfico

Colocando *labels*, *títulos*, e mudando o *fundo* para branco:

```
media <- mean(log(venda$preco))
dp <- sd(log(venda$preco))
ggplot(data = venda, aes(x = log(preco))) +
  geom_histogram(aes(y = ..density..), binwidth = 0.3,
                 fill = "lightblue", col = "black") +
  geom_rug() +
  stat_function(fun = dnorm, args = list(mean = media, sd = dp),
                color = "red") +
  xlab("Log do Preço") +
  ylab("Densidade") +
  ggtitle("Histograma Preços de Imóveis") +
  theme_bw()
```

Personalizando mais o gráfico

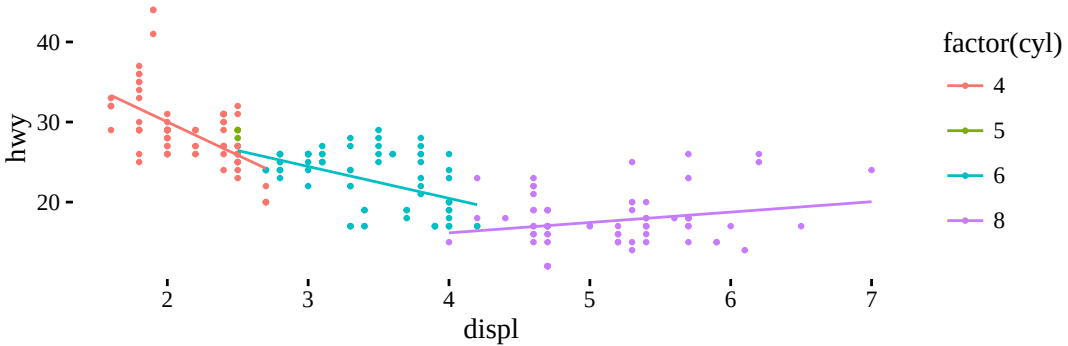


Temas pré-prontos

O pacote `ggthemes` já vem com vários temas pré-programados, replicando formatações de sites como The Economist, The Wall Street Journal, FiveThirtyEight, ou de outros aplicativos como o Stata, Excel entre outros. Esta é uma forma rápida e fácil de adicionar um estilo diferente ao seu gráfico. Experimente com os temas!

Temas pré-prontos

```
library(ggthemes)
ggplot(mpg, aes(displ, hwy, col = factor(cyl))) +
  geom_point() +
  geom_smooth(method = "lm", se = F) +
  theme_tufte()
```



Exercícios

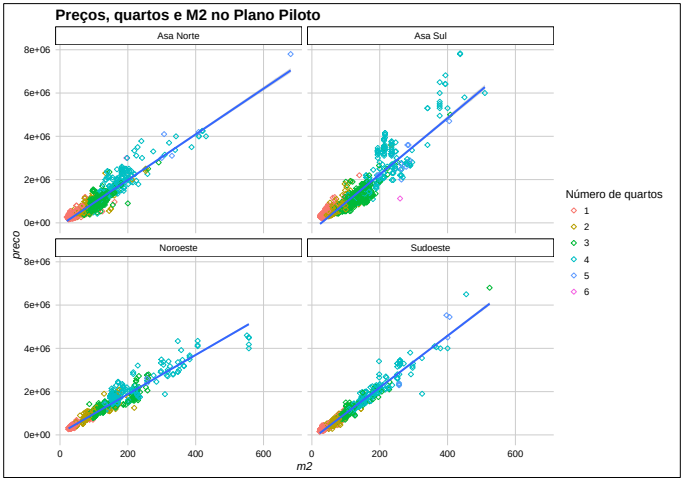
Sua vez.

- Faça um gráfico similar ao apresentado no primeiro dia de aula: "gráfico de dispersão de preço contra metro quadrado por bairro, cor dos pontos de acordo com número de quartos e linha de regressão".
- Adicione um tema do `ggthemes` a seu gráfico.

Soluções

```
ggplot(venda, aes(m2, preco)) +
  geom_point(shape = 5, aes(color = factor(quartos))) +
  geom_smooth(method = "lm") +
  facet_wrap(~bairro) +
  ggtitle("Preços, quartos e M2 no Plano Piloto") +
  scale_color_discrete("Número de quartos") +
  theme_gdocs()
```

Soluções


Preços, quartos e M2 no Plano Piloto
Asa Norte
Asa Sul
Noroeste
Sudoeste
preco
m2
0e+00
2e+06
4e+06
6e+06
8e+06
0
200
400
600
Número de quartos
1
2
3
4
5
6