Visualização de Dados com R

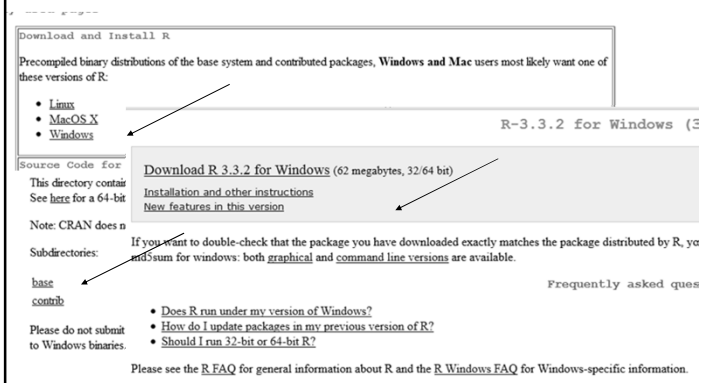
Lupércio França Bessegato
Dep. Estatística/UFJF

Roteiro

1. Análise exploratória de dados
2. Representação de dados univariados
3. Representação de dados bivariados
4. Representação de dados multivariados
5. Exemplos de aplicação
6. Referências

Fundamentos da Linguagem R

Download do Programa



Comandos

- Para solicitar uma tarefa do R podemos digitar uma linha de comando no console

```
R Console

R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em publica

Digite 'demo()' para demonstrações, 'help()' para o sistema on-li
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu na
Digite 'q()' para sair do R.

> sqrt(4)
[1] 2
>
```

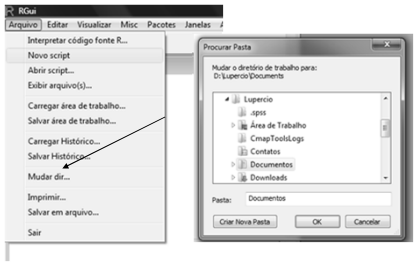

- Todas as funções do R devem ser digitadas em letras **minúsculas**
  - √ O R é sensível a letras maiúsculas e minúsculas.
- Todas as palavras-chaves do R estão em letras minúsculas
- R usa um ponto "." em vez de virgula "," quando há números com casas decimais.

Diretório de Trabalho

- Comando para verificar o diretório de trabalho que o R está usando:
  - √ `getwd()`
- Ideal sempre deixar scripts e dados de trabalho no mesmo diretório!
- Comando para mudar o R para seu diretório
  - √ `setwd("caminho_ate_diretorio")`
  - √ `Ex.: "D:/Lupercio/Documents"`

Mudança Diretório – Barra de Ferramentas

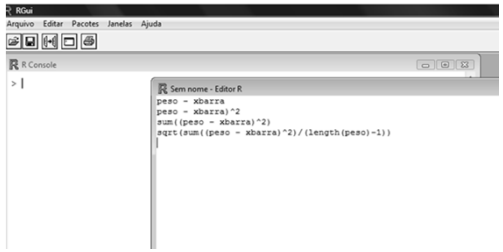
- Sugestão:
  - √ Sempre mude para seu diretório de trabalho quando iniciar a sessão em R
  - √ Guarde nele seus dados, gráficos, scripts, etc



Script

- Facilita para:
  - √ Correção ou expansão de comandos
  - √ Repetição de comandos
  - √ Armazenamento de resultados

Compilação do Script



- Usa-se a tecla F5 para compilar:
  - √ A linha em que se encontra o cursor (no script)
  - √ As linhas selecionadas (no script)
- Resultado compilação no console

Uso do Script

- Vantagens:
  - √ Facilidade para corrigir os comandos ou valores
  - √ Possibilidade de armazenar todos os resultados
  - √ Repetição dos passos corretos de toda a sessão
- Trabalho ‘limpo’
  - √ Para limpar o console usa-se CTRL + L

Carregando Pacotes

- Pacotes:
  - √ Conjuntos de funções específicas do R
  - √ No repositório do R está armazenada uma quantidade muito grande de pacotes que tem funções para um certo conjunto de tarefas
  - √ Para usar um pacote:
    - baixar o pacote (download) do repositório
    - carregar o pacote na sua área de trabalho.

Pacotes

- Quais pacotes estão disponíveis na sua instalação de R?
  - √ `library()`
- Interface hipertexto de ajuda:
  - √ `help.start()`
  - √ Escolher o link “Packages”
  - √ Clique no nome de um dos pacotes
    - Lista todos os objetos que este pacote contém.

- Página do R Project

Package Index

Packages in C:\Program Files\R\R-3.3.1\library

A B C D E F G H I J K L M N O P Q R S T U V X Y Z

abind	Combine Multidimensional Arrays
abundant	Abundant regression and high-dimensional principal fitted components
acepack	ace() and avas() for selecting regression transformations
ade4	Analysis of Ecological Data : Exploratory and Euclidean Methods in Environmental Sciences
asbio	A Collection of Statistical Tools for Biologists
assertthat	Easy pre and post assertions.
base	The R Base Package
base64enc	Tools for base64 encoding
BB	Solving and Optimizing Large-Scale Nonlinear Systems
BH	Boost C++ Header Files
BHH2	Useful Functions for Box, Hunter and Hunter II
BiasedUrn	Biased Urn Model Distributions
BiocGenerics	S4 generic functions for Bioconductor
BiocInstaller	Install/Update Bioconductor, CRAN, and github Packages

- Quais pacotes estão carregados na sua sessão?
  - √ `search()`
- Instalação de pacote direto do R
  - √ `install.packages("vegan")`

Exemplo

- Geração de amostra aleatória:

```
# 15 números aleatórias de uma distribuição normal, com média 1 e
# desvio-padrão=3

x1 <- rnorm(n = 15, mean = 1, sd = 3)
hist(x1)             # histograma de frequência
hist(x1, freq = F)   # histograma de densidade
truehist(x1)  # outro tipo de histograma
```

- Uso de função do pacote MASS:

```
search()             # pacotes disponíveis área de trabalho
library(MASS)        # carrega pacote MASS
truehist(x1)
help(package = MASS) # ajuda sobre o pacote
```

Conjuntos de Dados do R

- R traz vários conjuntos de dados internos, que são geralmente usados em demos ou exemplos
- Comando para ver a lista dos conjuntos de dados carregados:
  √ `data()`

Carregamento Built-in Data Set

- Carregamento do conjunto de dados:
  √ mtcars: Motor Trend Car Road Tests

```
# Conjunto de dados: mtcars: Motor Trend Car Road Tests
data(mtcars)          # carregamento
head(mtcars, 8)       # Print das primeiras 8 linhas
help(mtcars)          # informações sobre o banco
```

- Manipulação do conjunto de dados

```
nrow(mtcars)          # Número de linhas  (observações)
ncol(mtcars)          # Número de colunas (linhas)
str(mtcars)           # estrutura do objeto
```


Importação de Dados

- Importação para o R de dados disponíveis em formato eletrônico
- Comandos
  √ `read.table`
  √ `read.csv`
  √ `read.csv2`
  √ outros

Comando `read.table`

- Importação de dados em formato texto (arquivo do tipo ASCII)

```
# arquivo sem cabeçalho
ex01 <- read.table("gam01.txt")
head(ex01)
# arquivo com cabeçalho na 1a. limha
ex02 <- read.table("exemplo02.txt", head=T)
head(ex02)
# arquivo com campos separados por : e decimais, por vírgula
ex03 <- read.table("dadosfic.csv", head=T, sep=":", dec=",")
head(ex03)
# leitura direta pela web
ex04<-read.table("http://www.leg.ufpr.br/~paulojus/dados/gam01.txt")
head(ex04)
# leitura de arquivo com informações em suas 1as. linhas
teste.data <- read.table("test2.txt", skip=4, header=TRUE, sep="\t")
head(teste.data)
```

Comando `read.csv`

- Importação de dados em formato csv (*Comma Separated Value*)

```
# arquivo csv formato inglês
aereas.data <- read.csv("AirPassengers.csv", header=TRUE)
head(aereas.data)
# arquivo csv - comando read.table
aereas.data2 <- read.table("AirPassengers.csv", header=TRUE,sep=",")
head(aereas.data2)
# arquivo csv - gravado em formato brasileiro
solo <- read.csv("solo.csv", header = TRUE, dec = ",", sep = ";")
head(solo)
# comando read.csv2 (leitura direta de csv em formato brasileiro)
solo2 <- read.csv2("solo.csv", header = TRUE)
head(solo2)
```


Importação do Excel

- Importação direta de planilhas com extensão xlsx

```
# lê a primeira guia da planilha meuexcel.xlsx
# nomes das variáveis na primeira linha
library(xlsx)
dados <- read.xlsx("meuexcel.xlsx", 1)

# lê a guia na planilha chamada minhaguia
dados <- read.xlsx("meuexcel.xlsx", sheetName = “minhaguia")
```


Análise Exploratória de Dados

Compreendendo os Dados

- Questões chave ao iniciar análise de dados:
  - √ O que é a unidade de análise?
  - √ Há quantas observações?
  - √ Quanta variabilidade existe nos dados?
  - √ Quais são as unidades de medida dos dados?
  - √ As variáveis tomam quais valores?
  - √ Quantos são os dados faltantes?
  - √ Como as variáveis estão relacionadas?
    - (Elas estão correlacionados?)

O que é Análise Exploratória de Dados?

- Uma filosofia/abordagem para análise de dados
- Emprega uma variedade de técnicas (a maioria gráficas)…trabalharemos com alguns deles:
  - √ Diagrama de dispersão
  - √ Boxplot
  - √ Gráficos para identificação de outliers
  - √ Curvas de crescimento
  - √ Etc.

- São técnicas que buscam:
  - √ maximizar o "insight" do conjunto de dados;
  - √ perceber a estrutura subjacente;
  - √ extrair variáveis importantes;
  - √ detectar valores atípicos (extremos) e anomalias;
  - √ testar hipóteses fundamentais;
  - √ desenvolver modelos parcimoniosos; e
  - √ determinar conjunto ótimo de fatores

Idéia Básica

- Modelo = Suave + Irregular (tosco)
  - √ Frequentemente, as técnicas gráficas podem separar o "suave" do "irregular" ("ruído")

Clássica vs Exploratória

- Seqüência Clássica:
  - √ Problema > Dados > Modelo> Análise > Conclusões
- Exploratória:
  - √ Problema > Dados > Análise > Modelo > Conclusões

Análise Descritiva

- Inicia-se pela verificação dos tipos disponíveis de variáveis
  - √ Elas podem ser resumidas por:
    - Gráficos
    - Medidas
    - Tabelas

Classificação

- Qualitativas (Categóricas)
  - √ Nominais:
  - √ Ordinais
- Quantitativas:
  - √ Discretas
  - √ Contínuas

Objetivos Primários

- Familiarização com os dados
- Detecção de estruturas interessantes
- Presença de valores atípicos (*outliers*)

Razões para Uso de AED

- √ Identificação de erros e inconsistências
- √ Verificação de pressupostos do modelo
- √ Seleção preliminar de modelos apropriados
- √ Determinação das relações entre as variáveis explicativas
- √ Avaliação da direção e da intensidade das relações entre as variáveis explicativas e as variáveis respostas.

Análise Exploratória Multivariada

- Objetivo:
  - √ Explorar globalmente o conjunto de dados
  - √ Identificar correlação entre múltiplas variáveis:
    - – Há grupos de variáveis que estão sempre correlacionadas
  - √ Identificar correlação entre casos
    - – Os casos têm mesma estrutura?

- Métodos multivariados para redução de dados:
  - √ Resumir as correlações entre variáveis
  - √ Produzir um conjunto menor de variáveis (não correlacionadas) contendo as informações mais importantes
- Para um conjunto de objetos "relacionados"
  - √ Identificar grupos de objetos semelhantes
  - √ Identificar diferenças entre grupos de objetos semelhantes
    - – (e o que faz com que os objetos sejam semelhantes)

- Recomenda-se executar análise exploratória de dados univariados em cada um dos componentes, antes de realizar a AED multivariada.

Roteiro Básico

- Sequência inicial:
  - √ Medidas-resumo e gráficos:
    - – Variabilidade para cada variável
    - – Forma da distribuição de cada variável
  - √ Grupos de observações:
    - – Pré-determinados
    - – (para encontrar diferenças potenciais)
  - √ Diagrama de dispersão/correlações
    - – Associações entre pares de variáveis

Importante

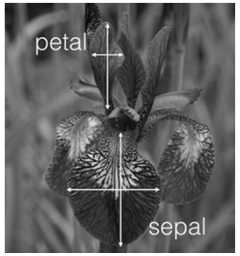
- A Análise Exploratória de Dados é um passo inicial crítico em qualquer análise de dados.

Visualização de Dados Univariados

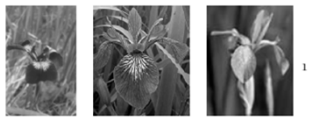
Conjunto de Dados – `iris`

- Anderson (1935) e Fischer (1936)
- Conjunto de dados de flores de íris (gênero de iridácea)
  - √ Medidas morfológicas de 50 flores de cada espécie
  - √ Espécies:
    - Iris setosa (originária do Alasca)
    - Iris versicolor
    - Iris virginica
- Dados: *iris* {*datasets*}

- Morfologia iris:



- Espécies



√ Variáveis:

- Sepal.Length: comprimento da sépala, em cm.
- Sepal.Width: largura da sépala, em cm.
- Petal.Length: comprimento da pétala, em cm.
- Petal.Width: largura da pétala, em cm.
- Species: setosa, versicolor e virginica

- Carregamento do conjunto de dados

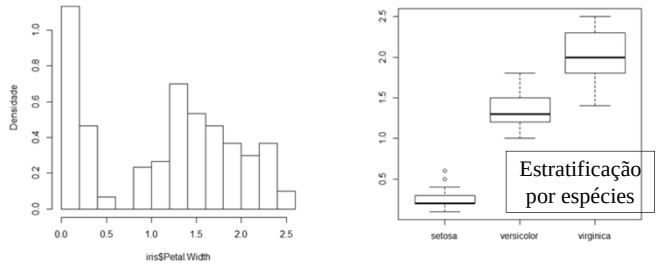
```
> dim(iris)
[1] 150  5
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1  ...
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

- Estratos da categórica

```
> table(iris$Spcies)

    setosa versicolor  virginica
        50         50         50
```


- Histograma da variável `Petal.Width`:

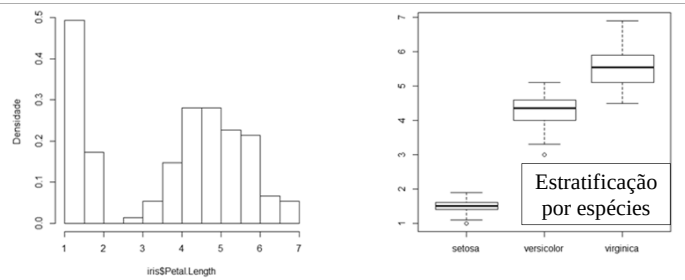
```
> # Petal.Width
> hist(iris$Petal.Width, freq = F, ylab = "Densidade", main = "")
> win.graph()
> boxplot(Petal.Width ~ Species, data = iris)
```



√ Mistura de populações

- Histograma da variável `Petal.Length`:

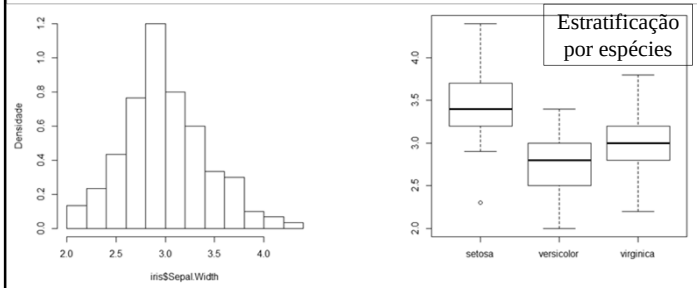
```
> # Petal.Length
> hist(iris$Petal.Length, freq = F, ylab = "Densidade", main = "")
> win.graph()
> boxplot(Petal.Length ~ Species, data = iris)
```



√ Mistura de populações

- Histograma da variável `Sepal.Width`:

```
> # Sepal.Width
> hist(iris$Sepal.Width, freq = F, ylab = "Densidade", main = "")
> win.graph()
> boxplot(Sepal.Width ~ Species, data = iris)
```



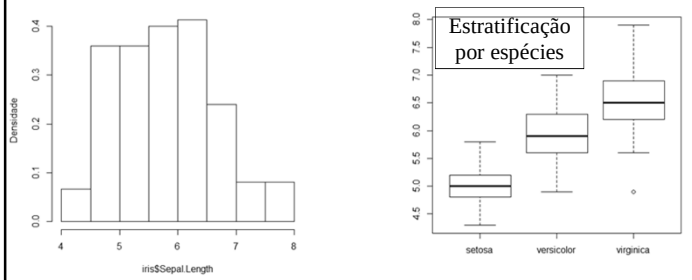
Estratificação por espécies

√ Mistura de populações menos acentuada

50

- Histograma da variável `Sepal.Length`:

```
> # Sepal.Length
> hist(iris$Sepal.Length, freq = F, ylab = "Densidade", main = "")
> win.graph()
> boxplot(Sepal.Length ~ Species, data = iris)
```



Estratificação por espécies

√ Mistura de populações mais próximas

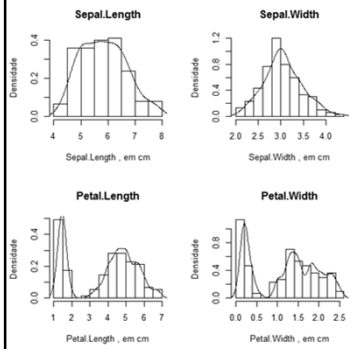
51

- Histogramas com suavizador:
  - √ Comando `density`: núcleo estimador

```
> variaveis <- names(iris[-5])
> par(mfrow = c(2, 2))
> for(i in 1: length(variaveis)) {
+ with(iris, {
+ dados <- eval(parse(text = variaveis[i]))
+ hist(dados, freq = F, main = variaveis[i], ylab = "Densidade",
+ xlab = paste(variaveis[i],", em cm"))
+ d <- density(dados, bw = "sj")
+ lines(d, lty = 1, col = "blue")
+ })
+ }
> par(mfrow = c(1, 1))
```

52

- Histogramas com suavizador:



√ Facilita visualização das misturas

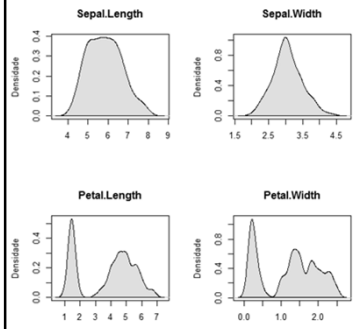
53

- Estimativas das densidades:

```
> variaveis <- names(iris[-5])
> par(mfrow = c(2, 2))
> for(i in 1: length(variaveis)) {
+ with(iris, {
+ dados <- eval(parse(text = variaveis[i]))
+ d <- density(dados, bw = "sj")
+ plot(d, type = "n", main = variaveis[i], ylab = "Densidade",
+ xlab = "")
+ polygon(d, col = "wheat")
+ })
+ }
> par(mfrow = c(1, 1))
```

54

- Estimativas suavizadas das densidades:



√ Todas as variáveis com estratificação por `Species`

55

Visualização de Dados Bivariados

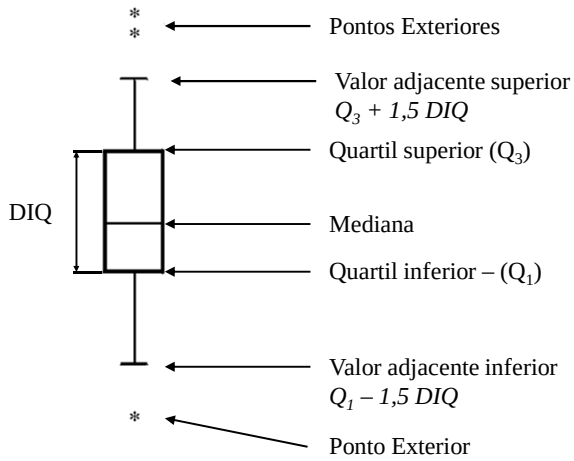
Técnicas Gráficas Bivariadas

- Após a análise univariada, sempre que possível recomenda-se a visualização das variáveis duas a duas

60

Box-plot

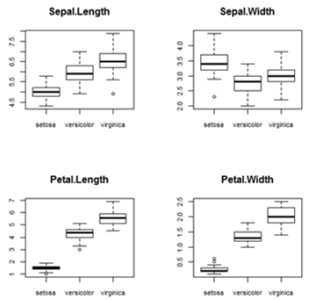
- Muito utilizado para comparações entre diferentes grupos de dados
  - √ Melhor técnica exploratória gráfica para examinar:
    - – Relação entre uma variáveis categórica e quantitativa
    - – Distribuição da variável quantitativa em cada nível da variável categórica



- Comentários:
  - √ São muito bons para apresentar informação sobre a tendência central, forma e peso das caudas
  - √ Podem ser enganosos quanto à multimodalidade
- Outlier:
  - √ Identificação pelo boxplot não indica problema
    - – Boxplot apenas sugere
    - – (técnica exploratória)
  - √ Quantidade depende do tamanho da amostra
    - – Sob normalidade, espera-se a sugestão de 1 em 150

- *Box plots* – `Iris`:

```
> # Box-plots
> variaveis <- names(iris[-5])
> par(mfrow = c(2, 2))
> for(i in 1: length(variaveis)) {
+ with(iris, {
+ dados <- eval(parse(text = variaveis[i]))
+ boxplot(dados ~ Species, data = iris, main = variaveis[i], cex.axis = 0.85)
+ })
+ }
```



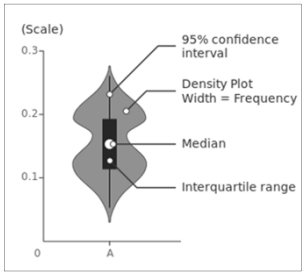
√ Dimensões morfológicas claramente estratificadas por `Species`

Violin Plot

- Usado para visualizar a distribuição dos dados e sua densidade de probabilidade
  - √ Combinação de box plot e gráfico de densidade
  - √ Box plot é limitado na exibição dos dados:
    - – simplicidade visual
    - – tende a esconder detalhes significativos sobre como os valores nos dados são distribuídos.
      - ▪ Ex.: multimodalidade
  - √ Violin plot:
    - – Exibe mais informação
      - ▪ (também mais ruído)

65

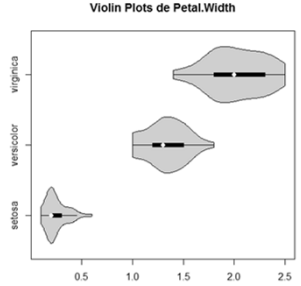
- Apresentação do gráfico:



66

- *Violin plot* de `Petal.Width`:

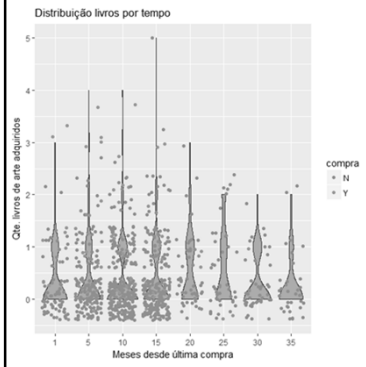
```
> library(vioplot)
> nomes <- levels(iris$Species)
> with(iris,{
+ #for(i in 1:3) assign(paste0("x",i), Petal.Width[Species == nomes[i]])
+ for(i in 1:3) assign(paste("x",i, sep=""), Petal.Width[Species == nomes[i]])
+ vioplot(x1, x2, x3, names = nomes,  col = "lightblue", horizontal = TRUE) # col = "gold"
+ title("Violin Plots de Petal.Width")
+ })
```



- √ Semelhante box plot
- √ Apresenta densidade condicional
- √ Cuidado com o uso em variáveis discretas

67

- *Violin plot* com dados discretos:



- √ Cuidado com o uso:
  - − No caso, a variável é discreta

68

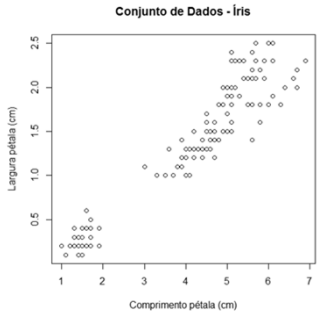
Diagrama de Dispersão

- Técnica exploratória gráfica básica para duas variáveis quantitativas
  - √ Se uma delas é resposta (eixo y)
  - √ Pode-se agregar mais uma ou duas variáveis categóricas
    - – Códigos e cores
  - √ Pode-se agregar uma variável quantitativa
    - – *Bubble plot*

- Objetivo:
  - √ Descrição de relações entre pares de variáveis
    - – Tendências lineares ou não
  - √ Encontrar transformações linearizantes
  - √ Agrupamentos de itens
    - – Há alguma variável categórica que explica?
  - √ Mudanças de variabilidade de uma variável em relação à outra
  - √ Identificação de valores atípicos (‘outliers’)

- Diagrama de Dispersão - Pétalas

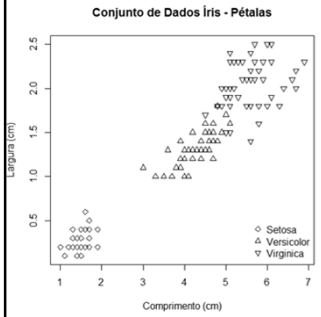
```
> # scatter plot simples
> plot(iris$Petal.Length, iris$Petal.Width, main="Conjunto de Dados - Íris",
+ xlab = "Comprimento pétala (cm)", ylab = "Largura pétala (cm)")
```



- √ Tendência linear
- √ Há dois agrupamentos de dados

- *Scatter plot* – Pétalas por `Species`:

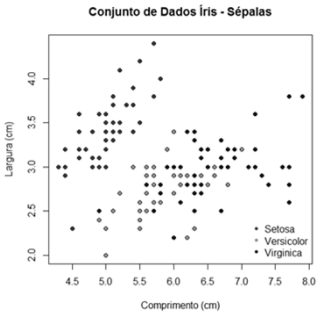
```
> # Scatter plot com o fator 'Species' - Pétalas
> plot(iris$Petal.Length, iris$Petal.Width, pch =
c(23,24,25)[unclass(iris$Species)],
+ main = "Conjunto de Dados Íris - Pétalas", xlab = "Comprimento (cm)",
+ ylab = "Largura (cm)")
> legend("bottomright", legend = c("Setosa", "Versicolor", "Virginica"),
+  pch = c(23,24,25), bty = "n")
```



- √ Tendência linear
- √ Delineia-se a discriminação dos 3 grupos

- *Scatter plot* – Sépalas por `Species`:

```
> # Scatter plot com fator 'Species' - Sépalas
> plot(iris$Sepal.Length, iris$Sepal.Width, pch = 21,
+ bg = c("red","green3","blue")[unclass(iris$Species)],
+ main = "Conjunto de Dados Íris - Sépalas", xlab = "Comprimento (cm)",
+ ylab = "Largura (cm)")
> legend("bottomright", legend = c("Setosa", "Versicolor", "Virginica"),
+   pch = rep(20,3), col = c("red","green3","blue"), cex = 1, bty = "n")
```

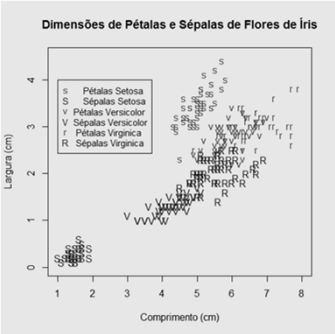


√ Grupo das setosas está bem discriminado
√ Discriminação entre os dois outros grupos não está tão clara

- *Scatter plot* com todas medidas e com o fator `Species`:

```
> # Scatter plot com o fator 'Species' - Todos comprimentos e larguras
> iS <- iris$Species == "setosa"
> iV <- iris$Species == "versicolor"
> iG <- iris$Species == "virginica"
> op <- par(bg = "bisque")
> matplot(c(1, 8), c(0, 4.5), type =  "n",
+   xlab = "Comprimento (cm)", ylab = "Largura (cm)",
+         main = "Dimensões de Pétalas e Sépalas de Flores de Íris")
> matpoints(iris[iS,c(1,3)], iris[iS,c(2,4)], pch = "sS", col = c(2,4))
> matpoints(iris[iV,c(1,3)], iris[iV,c(2,4)], pch = "vV", col = c(2,4))
> matpoints(iris[iG,c(1,3)], iris[iG,c(2,4)], pch = "rR", col = c(2,4))
> legend(1, 4, c("    Pétalas Setosa", "    Sépalas Setosa",
+                "Pétalas Versicolor", "Sépalas Versicolor",
+    " Pétalas Virginica", " Sépalas Virginica"), cex=0.9,
+        pch = "sSvVrR", col = rep(c(2,4), 3))
> par(op)
```


- *Scatter plot* – todos os comprimentos e larguras por `Species`:



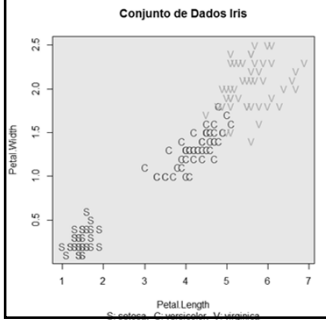
√ Setosa é do Alasca
√ Aparentemente as dimensões das pétalas discriminam melhor

- Construção *scatter plot* com array:

```
> # Criação da array
> nome.var <- colnames(iris)[-5]
> nome.esp <- as.character(unique(iris[,"Species"]))
> # criação array
> iris.S <- array(NA, dim = c(50, 4, 3),
+                 dimnames = list(NULL, nome.var, nome.esp))
> for(i in 1:3) iris.S[, , i] <- data.matrix(iris[1:50 + 50*(i-1), -5])
> # 1a. observação, 2a. camada
> iris.S[1, , 2]
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
         5.1          3.5          1.4          0.2
> # 1a. observação, todas as camadas
> iris.S[1, , ]
             setosa versicolor virginica
Sepal.Length    5.1        7.0       6.3
Sepal.Width     3.5        3.2       3.3
Petal.Length    1.4        4.7       6.0
Petal.Width     0.2        1.4       2.5
```

- *Scatter plot* – `iris`:

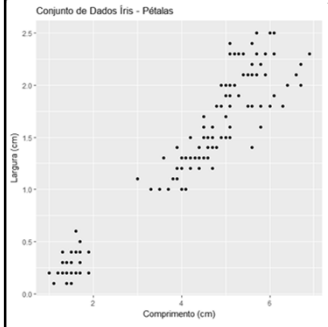
```
> # scatter plot - bg apenas na região do gráfico
> matplot(iris.S[, "Petal.Length", ], iris.S[, "Petal.Width", ], type = "n",
+         sub = paste(c("S: ", "C: ", "V: "), dimnames(iris.S)[[3]],
+                     sep = "", collapse= ",  "),
+   xlab = "Petal.Length", ylab = "Petal.Width", main = "Conjunto de Dados Iris")
> rect(par("usr")[1], par("usr")[3], par("usr")[2], par("usr")[4],
+ col = "bisque")
> matpoints(iris.S[, "Petal.Length", ], iris.S[, "Petal.Width", ], pch = "SCV",
+         col = rainbow(3, start = 0.8, end = 0.1))
```



√ Boa visualização da discriminação entre espécies

- *Scatter plot* com pacote `ggplot2`:

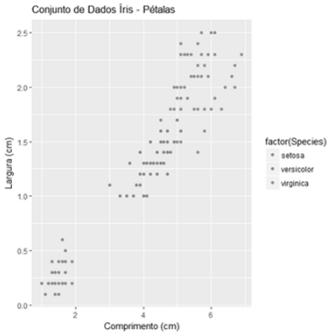
```
> library(ggplot2)
> library(gridExtra)
> # Plot com pontos default
> sp1 <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width))
> sp1 + geom_point() +
+ xlab("Comprimento (cm)") +   ylab("Largura (cm)") +
+ ggtitle("Conjunto de Dados Íris - Pétalas")
```



√ Configuração default

- *Scatter plot* com pacote `ggplot2`:

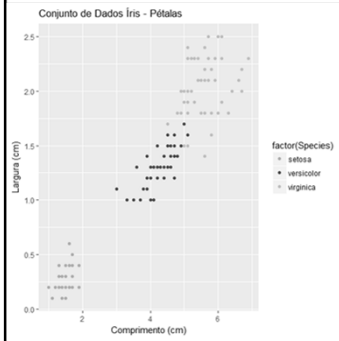
```
> # Mudança de cor dos pontos
> sp2 <- sp1 + geom_point(aes(color = factor(Species))) + # cor p/ nível fator
+  xlab("Comprimento (cm)") +
+    ylab("Largura (cm)") +
+  ggtitle("Conjunto de Dados Íris - Pétalas")
> sp2
```



√ Pontos com códigos de cores

- *Scatter plot* com pacote `ggplot2`:

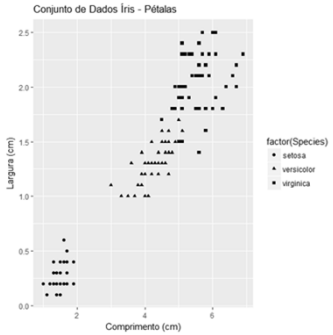
```
> # Cores conforme usuário
> sp3 <- sp1 + geom_point(aes(color=factor(Species))) +
+ scale_color_manual(values = c("orange", "purple", "gray")) +
+ xlab("Comprimento (cm)") +
+   ylab("Largura (cm)") +
+ ggtitle("Conjunto de Dados Íris - Pétalas")
> sp3
```



√ Códigos de cores conforme usuário

- *Scatter plot* com pacote `ggplot2`:

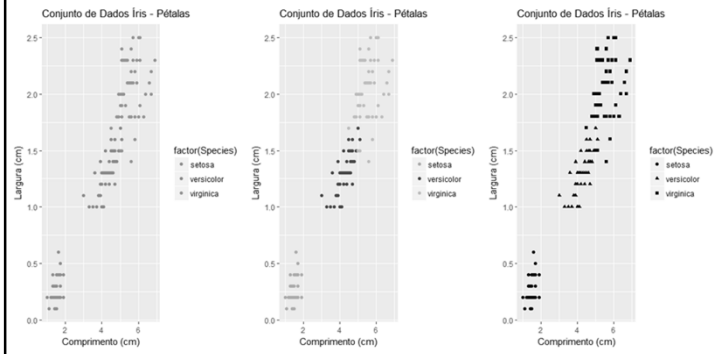
```
> # Mudança forma e tamanho dos pontos
> sp4 <- sp1 + geom_point(aes(shape = factor(Species))) + # forma p/ nível fator
+  xlab("Comprimento (cm)") +
+    ylab("Largura (cm)") +
+  ggtitle("Conjunto de Dados Íris - Pétalas")
> sp4
```



√ Modificação da forma e do tamanho dos pontos

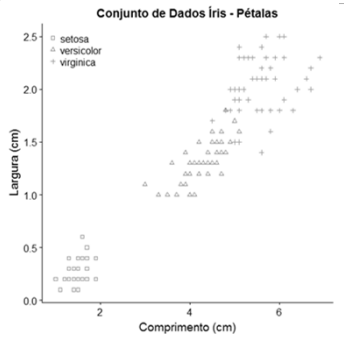
- *Scatter plot* com pacote `ggplot2`:
  - √ Painel com gráficos

```
> # painel com os gráficos
> grid.arrange(sp2, sp3, sp4, nrow=1)
```



- *Scatter plot* com pacote `ggplot2`:

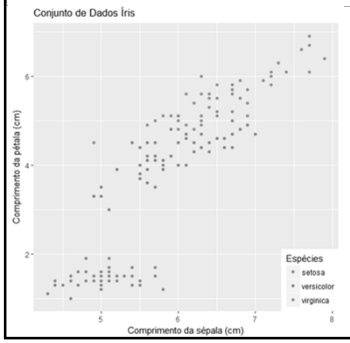
```
> # Formato pontos cfe. usuário
> sp1 + geom_point(aes(shape = factor(Species), color=factor(Species))) +
+ scale_shape_manual(values=c(0,2,3))+
+ theme(legend.position = c(0,1), legend.justification = c(0,1)) +
+ theme(legend.title=element_blank()) +
+ xlab("Comprimento (cm)") +
+   ylab("Largura (cm)") +
+ ggtitle("Conjunto de Dados Íris - Pétalas")
```



√ Visualização mais

- *Scatter plot* com pacote `ggplot2`:

```
> # Scatterplot comprimentos + espécies
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
+ geom_point()+
+ xlab("Comprimento da sépala (cm)") +
+   ylab("Comprimento da pétala (cm)") +
+ ggtitle("Conjunto de Dados Íris") +
+ scale_color_discrete(name ="Espécies") +
+ theme(legend.position = c(1, 0), legend.justification = c(1,0))
```



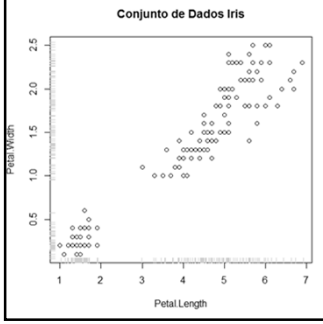
√ Comprimentos de pétala e de sépala oferecem boa discriminação das espécies

Marginal Plot

- Uso:
  - √ Avaliação da relação entre duas variáveis e exame de suas distribuições.
- Gráfico:
  - √ Diagrama de dispersão com histogramas, *dot plots* ou *box-plots* nas margens dos eixos x e y

- *Marginal plot* - comando `rug`:

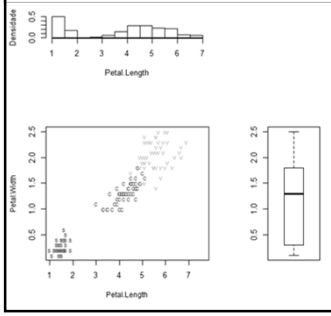
```
> # comando rug
> with(iris, {
+  plot(Petal.Width ~ Petal.Length, main = "Conjunto de Dados Iris",
+ xlab = "Petal.Length", ylab = "Petal.Width")
+ rug(jitter(Petal.Length, amount = 0.05), side = 1, ticksize = 0.02,
+ col = "light blue")
+ rug(jitter(Petal.Width, amount = 0.05), side = 2, ticksize = 0.02,
+ col = "light blue")
+ }) # end with
```



- √ Há concentração nos menores valores das variáveis

- *Marginal plot* - comando `layout`:

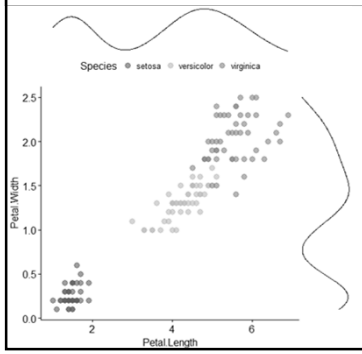
```
> # Dispersão com marginais (histograma e boxplot)
> layout(matrix(c(2,0,1,3), nrow = 2, byrow = T), respect = T,
+   widths = c(2, 1), heights = c(1, 2))
> xlim <- with(iris, range(Petal.Length)) * 1.1
> with(iris, {plot(Petal.Width ~ Petal.Length, data = iris, type = "n", xlim =
+ xlim, cex.lab = 0.9, xlab = "Petal.Length", ylab = "Petal.Width")
+ points(Petal.Length,Petal.Width,cex=0.6, pch=c("S","C","V")[unclass(Species)],
+         col = rainbow(3, start = 0.8, end = 0.1)[unclass(Species)])
+ hist(Petal.Length, main = NULL, freq = F, ylab = "Densidade")
+ boxplot(Petal.Width)})
```



- √ Permite visualização mais completa

- *Marginal plot* - pacote `ggpubr`:

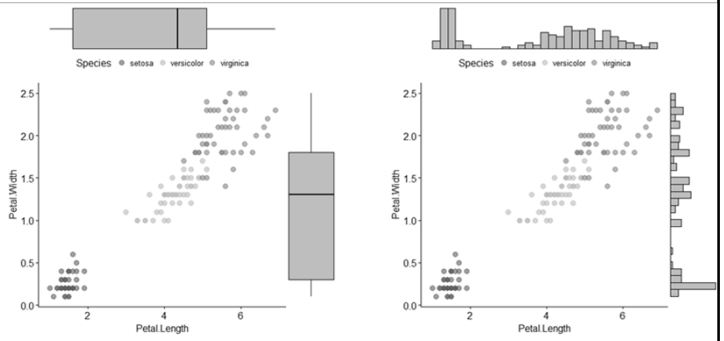
```
> library(ggpubr)# comando ggscatter
> library("ggExtra")# comando ggMarginal
> # objeto scatter plot
> # size: tamanho dos pontos; alpha: transparência cores
> p <- ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
+                color = "Species", palette = "jco",
+                size = 3, alpha = 0.6)
> ggMarginal(p, type = "density")
```



- √ Gráfico com construção mais simples
- √ Densidade estimada oferece visualização mais "limpa" da estrutura subjacente de cada variável

- *Marginal plot* com `ggpubr`:
  - √ box-plot ou histograma

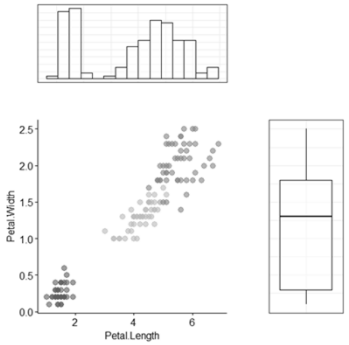
```
> # marginais: box-plot
> ggMarginal(p, type = "boxplot")
> # marginais: histograma
> ggMarginal(p, type = "histogram")
```



- *Marginal plot* - pacote `cowplot`:
  - √ Gráficos diferentes nas marginais
  - √ x: histograma e y: box-plot

```
> # marginais: x = histograma; y = box-plot
> library(cowplot) # ggExtra não lida com vários grupos
>   # marginais plots - x: histograma e y: box-plt
> xplot <- gghistogram(iris, x = "Petal.Length", bins = 15, ylab = "",
+ ggtheme = theme_bw())
> yplot <- ggboxplot(iris, y = "Petal.Width", alpha = 0.5,
+ xlab = "", ggtheme = theme_bw())
>   # limpeza dos gráficos
> p.b <- p + rremove("legend")
> xplot <- xplot + clean_theme() + rremove("legend")
> yplot <- yplot + clean_theme() + rremove("legend")
>   # montado gráfico com pacote cowplot
> plot_grid(xplot, NULL, p.b, yplot, ncol = 2, align = "hv",
+           rel_widths = c(2, 1), rel_heights = c(1, 2))
```


- *Marginal plot* :
  - √ Eixo x: histograma de `Petal.Length`
  - √ Eixo y: box-plot de `Petal.Width`

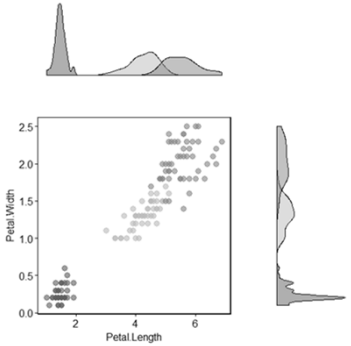


√ Pacote ggExtra não lida com gráficos diferentes nas margens

- *Marginal plot* - pacote `cowplot`:
  - √ Gráficos de densidades marginais por grupos

```
> # Scatter plot e density marginal plots coloridos por grupos
> sp1 <- ggscatter(iris, x = "Petal.Length", y = "Petal.Width", color = "Species",
+ palette = "jco", size = 3, alpha = 0.6) + border()
>   # densidades marginais por grupos
> xplot1 <- ggdensity(iris, "Petal.Length", fill = "Species",
+                     palette = "jco")
> yplot1 <- ggdensity(iris, "Petal.Width", fill = "Species",
+                     palette = "jco") + rotate()
>   # limpeza dos gráficos
> sp1 <- sp1 + rremove("legend")
> yplot1 <- yplot1 + clean_theme() + rremove("legend")
> xplot1 <- xplot1 + clean_theme() + rremove("legend")
>   # Montagem do gráfico com pacote cowplot
> plot_grid(xplot1, NULL, sp1, yplot1, ncol = 2, align = "hv",
+           rel_widths = c(2, 1), rel_heights = c(1, 2))
```

- *Marginal plot* :
  - √ Eixo x: densidade estimada de Petal.Length
  - √ Eixo y: densidade estimada de Petal.Width

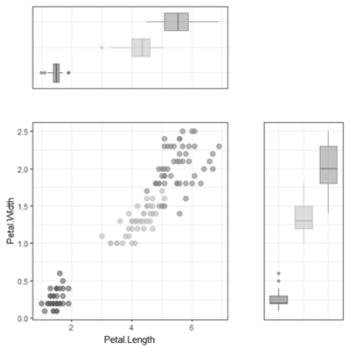


√ Pacote cowplot lida com vários grupos no *scatter plot* e nos *marginal plots*

- *Marginal plot* - pacote `cowplot`:
  - √ Gráficos de densidades marginais por grupos

```
>   # Scatter plot e box-plots por grupos + ggtheme
> sp2 <- ggscatter(iris, x = "Petal.Length", y = "Petal.Width", color= "Species",
+ palette = "jco", size = 3, alpha = 0.6, ggtheme = theme_bw())
>   # box-plots marginais
> xplot2 <- ggboxplot(iris, x = "Species", y = "Petal.Length", color = "Species",
+    fill = "Species", palette = "jco", alpha = 0.5,
+    ggtheme = theme_bw()) + rotate()
> yplot2 <- ggboxplot(iris, x = "Species", y = "Petal.Width", color = "Species",
+    fill = "Species", palette = "jco", alpha = 0.5,
+    ggtheme = theme_bw())
>   # limpeza dos gráficos
> sp2 <- sp2 + rremove("legend")
> yplot2 <- yplot2 + clean_theme() + rremove("legend")
> xplot2 <- xplot2 + clean_theme() + rremove("legend")
>   # montagem do gráfico com pacote cowplot
> plot_grid(xplot2, NULL, sp2, yplot2, ncol = 2, align = "hv",
+           rel_widths = c(2, 1), rel_heights = c(1, 2))
```


- *Marginal plot* :
  - √ Eixo x: box-plot de Petal.Length
  - √ Eixo y: box-plot de Petal.Width

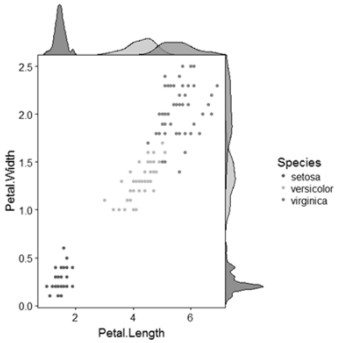


√ Box-plots marginais com vários grupos

- *Marginal plot* - pacote `cowplot`:
  - √ Sem espaçamento entre gráficos

```
> # Scatter plot com marginal plot por grupo (sem espaço entre gráficos)
>
>   # carrega função
> source("funcoes/axis_canvas.R")
>   # gráfico principal
> pp <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species))+
+ geom_point()+ ggpubr::color_palette("jco")
>   # densidades marginais - eixo x
> xdens <- axis_canvas(pp, axis = "x")+
+ geom_density(data = iris, aes(x = Petal.Length, fill = Species),
+   alpha = 0.7, size = 0.2) +
+   ggpubr::fill_palette("jco")
>   # densidades marginais - eixo y
>   # coord_flip = TRUE, se coord_flip() for usada
> ydens <- axis_canvas(pp, axis = "y", coord_flip = TRUE)+
+    geom_density(data = iris, aes(x = Petal.Width, fill = Species),
+    alpha = 0.7, size = 0.2)+
+    coord_flip()+
+    ggpubr::fill_palette("jco")
>   # montagem dos gráficos
> p1 <- insert_xaxis_grob(pp, xdens, grid::unit(.2, "null"), position = "top")
> p2 <- insert_yaxis_grob(p1, ydens, grid::unit(.2, "null"), position = "right")
> ggdraw(p2)
```

- *Marginal plot* :
  - √ Eixo x: box-plot de Petal.Length
  - √ Eixo y: box-plot de Petal.Width



√ `axis_canvas` para eliminar espaço entre gráficos

Linha de Regressão

- Scatterplots são úteis para interpretar tendências no dados
  - √ Regressão linear:
    - Ajustar a melhor reta entre os pontos
    - Linha de tendência.
- Usos de regressão linear:
  - √ Preditivo
  - √ Teste estatístico de significância:
    - Inclinação da reta de regressão é diferente de 0? (Mudança em *x* tem efeito em *y*?)

- Reta de regressão:
  - √ Resposta: `Petal.Width`
  - √ Explicativa: `Petal.Length`

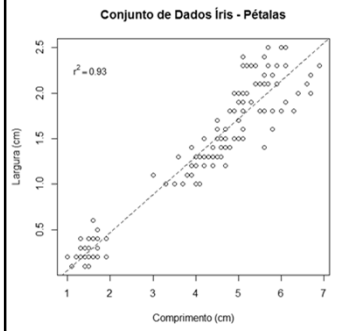
```
>   # reta de regressão y = Petal.Width, x = Petal.Length
> petal.lm <- lm(Petal.Width ~ Petal.Length, data = iris)
> summary(petal.lm)
Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.363076   0.039762  -9.131  4.7e-16 ***
Petal.Length  0.415755   0.009582  43.387  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2065 on 148 degrees of freedom
Multiple R-squared:  0.9271,    Adjusted R-squared:  0.9266
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
> cor(iris$Petal.Length, iris$Petal.Width)
[1] 0.9628654
```

$$\hat{\beta}_0 = -0,363$$
$$\hat{\beta}_1 = 0.416$$

$$r = 0,963$$
$$R^2 = 0.927$$

- *Scatter plot* com linha de regressão:

```
>   # scatter plot com reta de regressão
> r2 <- summary(petal.lm)$r.squared
> plot(Petal.Width ~ Petal.Length, data = iris,
+ main = "Conjunto de Dados Íris - Pétalas", xlab = "Comprimento (cm)",
+ ylab = "Largura (cm)")
> abline(petal.lm, lty = 2, col = "blue")
> text(x = 1, y = 2.25, bquote(r^2 == .(round(r2,2)), pos = 4, cex = 0.90)
```



√ Forte relação linear entre as variáveis
√ Tendência linear crescente

Conjunto de Dados – `cars`

- Conjunto de dados sobre velocidade de carros e distância para parar após freada.
  - √ Amostra: 50 observações
  - √ Dados dos anos 20 do século passado
- Variáveis:
  - `speed`: velocidade, em milhas por hora
  - `dist`: distância para parada, em pés
- Dados: `cars{datasets}`

- Carregamento do conjunto de dados

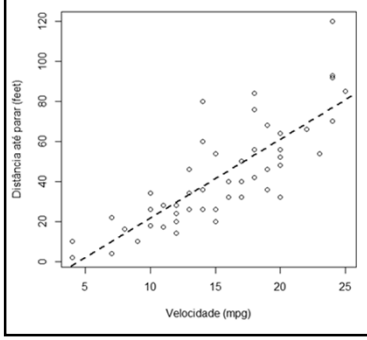
```
> help(cars)
> dim(cars)
[1] 50  2
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

√ Preparação dos dados

```
> # preparação dos dados
> x <- with(cars, speed)
> y <- with(cars, dist)
> tamanho <- dim(cars)[1]
> x.novo <- seq(min(x), max(x), length.out = tamanho)
```


- *Scatter plot* com linha de regressão:

```
> # scatter plot com regressão
> plot(dist ~ speed, data = cars, xlab = "Velocidade (mpg)",
+ ylab = "Distância até parar (feet)")
>   # ajuste reta de regressão
> cars.lm <- lm(dist ~ speed, data = cars)
> abline(cars.lm, lty = 2, lwd = 2, col = "blue")
```




```
> cars.lm
Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

√ Interpretação inclinação?
√ Reta se ajusta bem aos dados?

Ajuste por Polinômio

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \ldots \hat{\beta}_p x^p$$

- Polinômios podem ser uma boa escolha
  - √ Fácil de obter uma solução direta e fechada
  - √ Fáceis de serem implementados
  - √ Podem ser facilmente comparados com outros modelos preditivos.

- Regressão polinomial:
  - √ Resposta: `dist`
  - √ Explicativa: `speed`

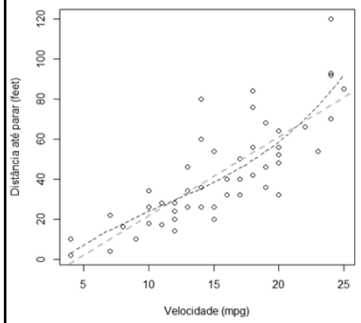
```
>   # ajuste regressão cúbica
> cars.3 = lm(y ~ poly(x, 3))
> cars.3

Call:
lm(formula = y ~ poly(x, 3))

Coefficients:
(Intercept)  poly(x, 3)1  poly(x, 3)2  poly(x, 3)3
      42.98       145.55        23.00        13.80
```

$$\hat{\beta}_0 = 42,98$$
$$\hat{\beta}_1 = 145,55$$
$$\hat{\beta}_2 = 23,00$$
$$\hat{\beta}_3 = 13,80$$

$$\text{dist} = 42,98 + 145,55\,\text{speed} + 23,00\,\text{speed}^2 + 13,80\,\text{speed}^3$$

- *Scatter plot* com ajuste polinomial:

```
>   # scatter plot com ajuste polinomial
> plot(dist ~ speed, data = cars, xlab = "Velocidade (mpg)",
+ ylab = "Distância até parar (feet)")
> abline(cars.lm, lty = 2, lwd = 2, col = "gray")
> lines(x.novo, predict(cars.3, data.frame(x = x.novo)), col = "blue", lty = 2)
```



√ Polinômio de 3º grau aparenta se ajustar melhor aos dados

Suavização

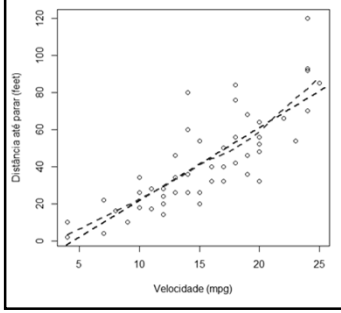
- Usada para descobrir tendências em dados com ruído.
  - √ Tendências não lineares
  - √ Mudanças na estrutura da tendência

- Suavizadores mais utilizados:
  - √ Loess
  - √ Núcleo estimador
  - √ Splines

- Loess e Lowess
- Regressão não paramétrica ponderada localmente
  - √ Abordagem do vizinho mais próximo.
  - √ Span: valor de controle da suavização.
  - √ Pode-se construir região de confiança em torno da tendência
    - Precisão da predição.

- *Scatter plot* com linha de regressão:

```
        # scatter plot com regressão e ajuste loess
x <- with(cars, speed)
y <- with(cars, dist)
tamanho <- dim(cars)[1]
  # ajuste do loess
cars.loess <- loess.smooth(x, y, evaluation = tamanho, family = "gaussian",
                                                span = 0.75, degree = 1)
lines(cars.loess, lty = 2, lwd = 2, col = "red")
```



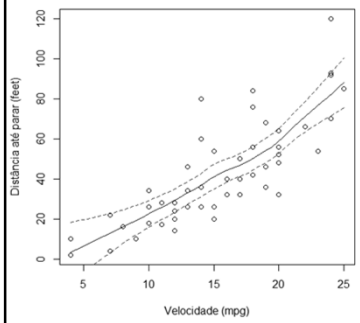
√ Aparentam se afastar nos extremos

- Loess com região com 95% de confiança
  - √ Comandos

```
>   # alternativa para ajuste do loess
> cars.loess2 <- loess(y ~ x, family = "gaussian", span = 0.75, degree = 1)
>
>   # construção de região com 95% de confiança em torno cars.loess
> x.novo <- seq(min(x), max(x), length.out = tamanho)
> ci <- cbind(predict(cars.loess2, data.frame(x = x.novo)),
+ predict(cars.loess2, data.frame(x = x.novo)) +
+ predict(cars.loess2, data.frame(x = x.novo),
+      se = TRUE)$se.fit*qnorm(1 - 0.05/2),
+ predict(cars.loess2, data.frame(x = x.novo)) -
+ predict(cars.loess2, data.frame(x=x.novo),
+   se = TRUE)$se.fit*qnorm(1 - 0.05/2)
+ )
```


- *Loess* com região de confiança:

```
>   # scatter plot dos pontos
> plot(dist ~ speed, data = cars, xlab = "Velocidade (mpg)",
+ ylab = "Distância até parar (feet)")
>   # Loess com região de confiança
> matplot(x.novo, ci, lty = c(1, 2, 2), col = c(1, 2, 2), type = "l", add = T)
```



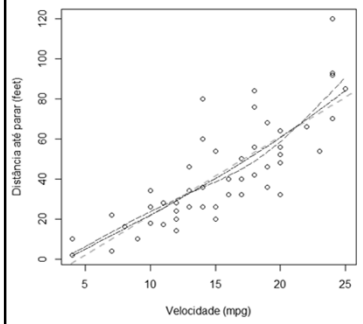
√ Região de confiança mais larga nos extremos

- Suavização por spline
  - √ Usa uma faixa valores de x para determinar sua suavidade.
    - – Valores são denominado nós
    - – Quanto menor a quantidade de nós, mais suave a curva.
    - – (mais nós, mais ruído no modelo)

- Suavização com *spline*
  - √ Comandos

```
> library(splines)
>   # spline natural
> cars.ns.3 <- lm(y ~ ns(x, 3) )
>   # spline suavizadoe
> cars.sp <- smooth.spline(y ~ x, nknots = 15)
>   # scatter plot
> plot(dist ~ speed, data = cars, xlab = "Velocidade (mpg)",
+ ylab = "Distância até parar (feet)")
>   # ajuste da reta de regressão
> abline(cars.lm, lty = 2, lwd = 2, col = "gray")
>   # spline suavizado
> lines(cars.sp, col = "blue", lty = 6)
>   # spline natural
> lines(x.novo, predict(cars.ns.3, data.frame(x = x.novo)), col = "red", lty = 5)
```


- *Loess* com região de confiança:

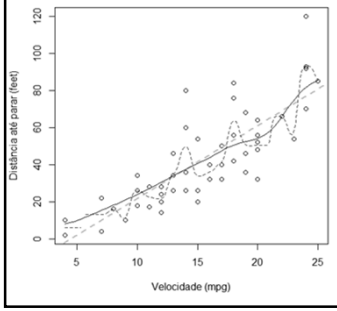


- √ Os ajustes do spline natural (verde) e do spline suavizado (azul) são bastante semelhantes.

- Suavização por núcleo estimador
  - √ Usa uma média ponderada do ponto (x, y) correspondente.
    - – Suavização é baseada na largura da janela (valor predefinido
    - – Quanto maior a largura da janela, maior a suavidade
    - – Larguras da janela menores podem ser muito sensível à variação local
      - Método não lida bem com variação localizada
      - Podem ocorrer mudanças muito dramáticas na curva.

- *Scatter plot* com núcleo estimador

```
> # adicionando ajuste por núcleo estimador no scatter plot
>   # scatter plot
> plot(dist ~ speed, data = cars, xlab = "Velocidade (mpg)",
+ ylab = "Distância até parar (feet)")
>   # ajuste da reta de regressão
> abline(cars.lm, lty = 2, lwd = 2, col = "gray")
>   # suavização por núcleo estimador - largura da janela: 5
> lines(ksmooth(x, y, "normal", bandwidth = 5), col = "blue", lty = 7)
>   # suavização por núcleo estimador - largura da janela: 1
> lines(ksmooth(x, y, "normal", bandwidth = 1), col = "red", lty = 2)
```

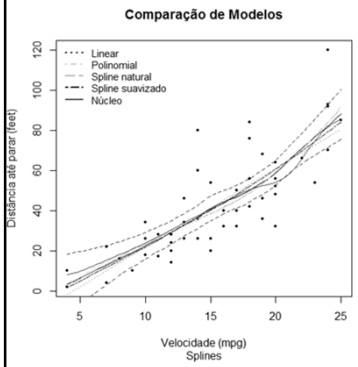


√ A curva mais suave tem a maior largura da janela (5)

- Comparação dos ajustes
  - √ Comandos

```
> # Comparação dos ajustes
> plot(x, y, xlim = c(min(x), max(x)), ylim = c(min(y), max(y)), pch = 16,
+ cex = 0.5, xlab = "Velocidade (mpg)",
+ ylab = "Distância até parar (feet)", main = "Comparação de Modelos",
+ sub = "Splines")
>   # Loess com região de confiança
> matplot(x.novo, ci, lty = c(1, 2, 2), col = c(1, 2, 2), type = "l", add = T)
>   # Regressão linear
> lines(x.novo, predict(cars.lm, data.frame(speed = x.novo)), col = "orange",
+ lty = 3)
>   # ajuste polinomial
> lines(x.novo, predict(cars.3, data.frame( x = x.novo)), col = "light blue",
+ lty=4)
>   # Spline natural
> lines(x.novo, predict(cars.ns.3, data.frame(x = x.novo)), col = "green",
+ lty = 5)
>   # Spline suavizado
> lines(cars.sp, col = "blue", lty = 6)
>   # Suavização por núcleo
> lines(ksmooth(x, y, "normal", bandwidth = 5), col = "purple", lty = 7)
> legend("topleft",c("Linear", "Polinomial", "Spline natural", "Spline suavizado",
+  "Núcleo"), col = c("black", "light blue", "green", "blue", "purple"),
+ lty = c(3, 4, 5, 6, 7), lwd = 2, bty = "n", cex = 0.9)
```


- *Scatter plot* com os ajustes



- √ Não é tarefa fácil tentar determinar qual é o melhor modelo
- √ Decisão pode depender da sensibilidade permitida para a variação local que você deseja permitir.
- √ No exemplo, todas as abordagens levam a soluções semelhantes.
- √ Se é necessária visualização de uma estrutura suave, talvez seja melhor usar a abordagem mais simples.
  - √ Deve-se tomar mais cuidados com ajustes para predição

Estimação de Densidade

- Há situações em que o objetivo é examinar o scatter plot e identificar regiões do plot com maior ou menor densidade de observações (clusters)
  - √ Estimação de densidade por núcleo estimador

Núcleo Estimador da Densidade

$$\hat{f}(x) = \frac{1}{hn}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

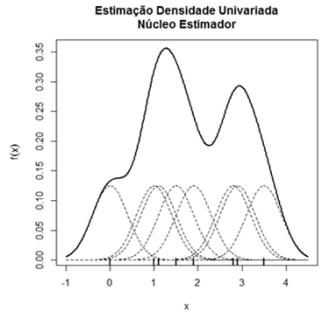
√ K: núcleo estimador é uma densidade
- Pode assumir várias formas

√ Núcleo gaussiano $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$

- Estimativa por núcleo em x ($\hat{f}(x)$):
  √ Soma das contribuições de cada observações

- Estimação da densidade por núcleo:
  √ Exemplo

```
> # densidade univariada - funcionamento do núcleo
>   # observações
> x <- c(0, 1, 1.1, 1.5, 1.9, 2.8, 2.9, 3.5)
> n <- length(x)
> xgrid <- seq(from = min(x) -1, to = max(x) + 1, by = 0.01)
>   # janela
> h <- 0.4
>   # aplicação do núcleo no grid
> cont <-sapply(x, function(a) dnorm((xgrid - a)/h)/(n * h))
>   # gráfico da estimação e das contribuições
> plot(xgrid, rowSums(cont), lwd = 2, xlab = "x",
+ ylab = expression(hat(f)(x)), type = "l",
+ main = "Estimação Densidade Univariada\nNúcleo Estimador" )
>   # observações
> rug(x, lwd = 2)
>   # plot das contribuições
> out <- apply(cont, 2, function(b) lines(xgrid, b, lty = 2))
```

√ Estimação da densidade em um ponto
- Média das observações ponderada pelo núcleo
- Núcleo determina formato das elevações
- Janela determina sua largura

Conjunto de Dados – `iris`

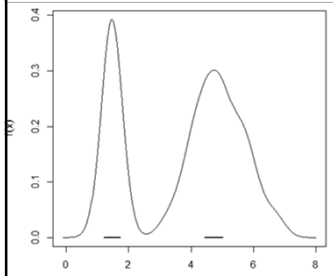
- Conjunto de dados de flores de íris (gênero de iridácea)
  √ 4 medidas morfológicas de 50 flores de cada espécie
  √ Variáveis:
  - `Sepal.Length`: comprimento da sépala, em cm.
  - `Sepal.Width`: largura da sépala, em cm.
  - `Petal.Length`: comprimento da pétala, em cm.
  - `Petal.Width`: largura da pétala, em cm.
  - `Species`: setosa, versicolor e virginica
- Dados: *`iris`* {*`datasets`*}

√ Variáveis:

- Sepal.Length: comprimento da sépala, em cm.
- Sepal.Width: largura da sépala, em cm.
- Petal.Length: comprimento da pétala, em cm.
- Petal.Width: largura da pétala, em cm.
- Species: setosa, versicolor e virginica

- Núcleo estimador densidade – univariada
  √ `Petal.Length`

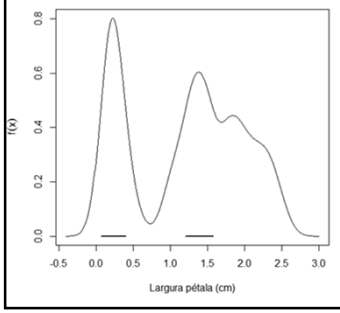
```
> # Exemplo - Estimação densidade univariada
> library(ks)
>  # Petal.Length
> fhat <- kde(x = iris[, 3])
> plot(fhat, cont = 50, col.cont = "blue", cont.lwd = 2,
+ ylab = expression(hat(f)(x)),
+ xlab = "Comprimento pétala (cm)")
```



√ Atributo `cont`:
- Linha horizontal (azul) indica nível (%) de densidade de mais alto definido.

- Núcleo estimador densidade – univariada
  √ `Petal.Width`

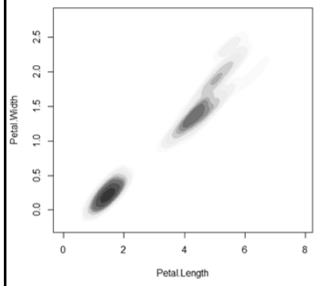
```
>   # Petal.Width
> fhat <- kde(x = iris[, 4])
> plot(fhat, cont = 50, col.cont = "blue", cont.lwd = 2,
+ ylab = expression(hat(f)(x)),
+ xlab = "Largura pétala (cm)")
```

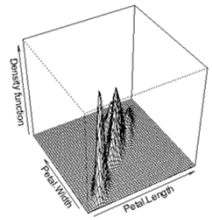


√ Também apresenta bimodalidade
√ Como será a densidade conjunta de ambas as variáveis?

- Núcleo estimador densidade – bivariada
  √ `Petal.Length` e `Petal.Width`

```
> # Scatter plot com densidade bivariada
> >   # Petal.Length e Petal.Width
> fhat <- kde(x = iris[, 3:4])
>   # contour plot
> plot(fhat, display = "filled.contour2", cont = seq(10, 90, by = 10))
>   # perspectiva
> plot(fhat, display = "persp", thin = 3, border = 1, col = "white")
```



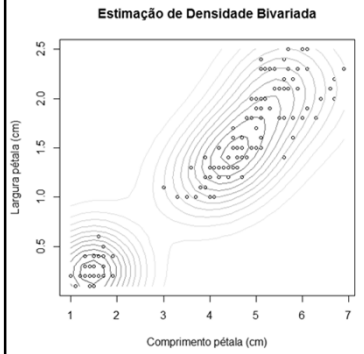


√ Indício de 3 grupos

- *Countour plot* da estimação densidade:
  - √ `Petal.Length` e `Petal.Width`

```
> # estimativas densidade bivariada por núcleo - contour plot
> library(MASS)
> library(colorspace)
>
> plot(Petal.Width ~ Petal.Length, data = iris, cex = 0.75,
+ xlab = "Comprimento pétala (cm)",
+ ylab = "Largura pétala (cm)", main = "Estimação de Densidade Bivariada")
>   # estimação densidade - package: MASS
> k <- kde2d(iris[, 3], iris[, 4])
> cnt <- contourLines(k$x, k$y, k$z)
> n <- length(cnt)
>   # definição de cores - package: colorspace
> cores <- rev(sequential_hcl(n))
>   # curvas de nível - desenho e cores
> for( i in seq_len(n) ) lines(cnt[[i]], col=cores[i])
```

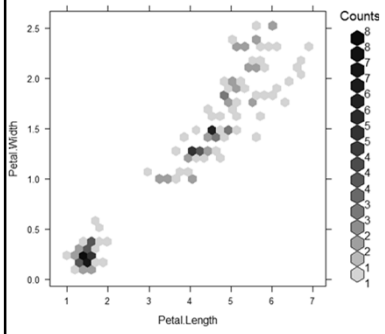

- Curvas de nível densidade bivariada



√ Cores atenuadas facilitam visualização da densidade?
√ Indício de multimodalidade

- Estimativa densidade codificada por cor
  - √ `Petal.Length` e `Petal.Width`

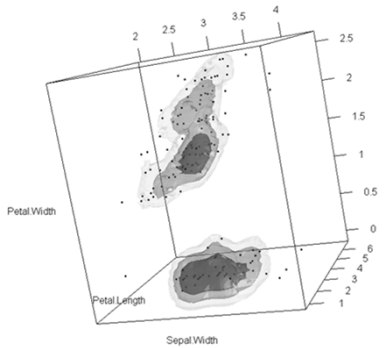
```
> # estimativa densidade codificada por cor
> library(hexbin)
>
> hexbinplot(Petal.Width ~ Petal.Length, iris, aspect = 1)
```



√ Visualização das contagens de observações por cores

- Núcleo estimador densidade – trivariada
  - √ `Sepal.Width`, `Petal.Length` e `Petal.Width`

```
> # Densidade trivariada
> fhat <- kde(x = iris[,2:4])
> plot(fhat, drawpoints = TRUE)
```

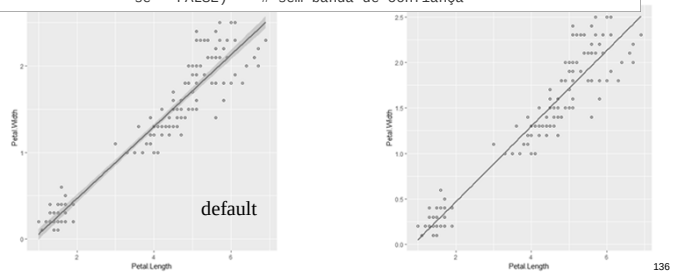


√ Objeto RGL
- Interação facilita visualização em 3 dimensões

- *Scatter plot* com o pacote `ggplot2`
  - √ Pacote para visualização de dados
  - √ Trabalha com componentes semânticos, tais como escalas e camadas

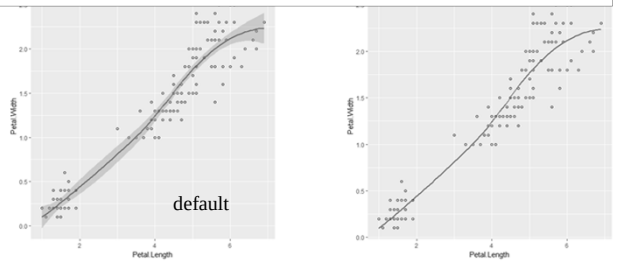
- *Scatter plot* e reta de regressão –`ggplot2`

```
>   # scatter plot com reta de regressão – ggplot2
> library(ggplot2)
>
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
+     geom_point(shape = 1) +    # círculos vazios
+     geom_smooth(method = lm)   # adiciona linha de regressão
> # default: banda de confiança de 95%
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
+     geom_point(shape = 1) +    # círculos vazios
+     geom_smooth(method = lm,   # adiciona linha de regressão
+                 se = FALSE)    # sem banda de confiança
```



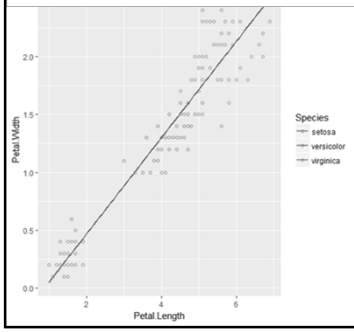
- *Scatter plot* e suavização *loess* –`ggplot2`

```
> # com banda de confiança
>   # scatter plot e loess c/ banda de confiança
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
+     geom_point(shape = 1) +    # círculos vazios
+     geom_smooth()           # adiciona curva suavizada loess
> # default: banda de confiança de 95%
>   # scatter plot e loess s/ banda de confiança
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
+     geom_point(shape = 1) +    # círculos vazios
+     geom_smooth(method = "loess",# adiciona curva suavizada loess
+                 se = FALSE)    # sem banda de confiança
```



- *Scatter plot* c/ pontos coloridos por `Species` – `ggplot2`

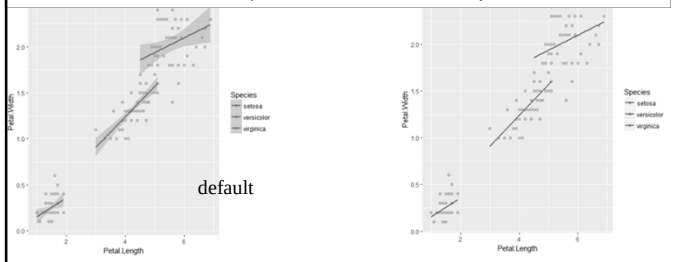
```
>   # scatter plot com cores por Species
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
+ geom_point(shape = 1) + # círculos vazios
+ geom_smooth(method = lm , # adiciona reta de regressão
+ aes(group = 1),# única reta
+ se = FALSE)# sem banda de confiança
```



- √ A linha de regressão é a mesma em todas as espécies?
  - Mesma inclinação por grupo?

• Retas de regressão por Species

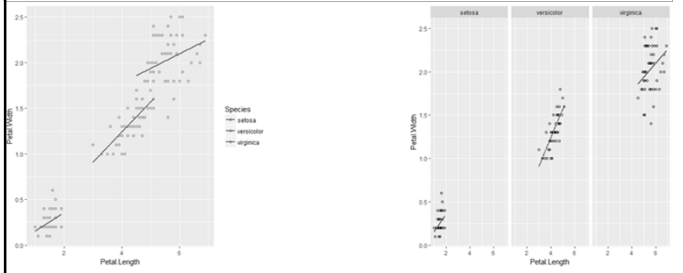
```
>   # scatter plot com cores diferentes e linhas de regressão p/ grupo
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
+     geom_point(shape = 1) +# círculos vazios
+     scale_colour_hue(l = 50) + # paleta um pouco mais escura que a normal
+     geom_smooth(method = lm)   # adiciona retas de regressão
> # com banda de confiança
>   # scatter plot com cores diferentes e linhas de regressão s/ ic
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
+     geom_point(shape = 1) +
+     scale_colour_hue(l = 50) + # paleta um pouco mais escura que a normal
+     geom_smooth(method = lm,   # adiciona retas de regressão
+                 se = FALSE)    # sem banda de confiança
```



default

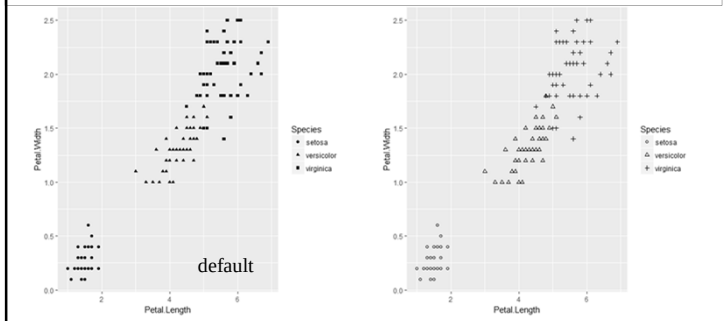
• Retas de regressão por Species

```
>   # scatter plot com cores diferentes e linhas de regressão s/ ic
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
+     geom_point(shape = 1) +
+     scale_colour_hue(l = 50) + # paleta um pouco mais escura que a normal
+     geom_smooth(method = lm,   # adiciona retas de regressão
+                 se = FALSE)    # sem banda de confiança
>   # painéis por Species c/ linhas de regressão
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, group = Species)) +
+        geom_point(shape = 1, size = 1.5)    + # tamanho dos pontos
+        stat_smooth(method = lm, se = FALSE) + # retas regressão s/ ic
+        facet_wrap( ~ Species)     # painéis por Species
```



• Pontos formatados por Species

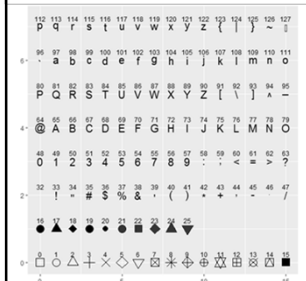
```
>   # ajusta formato ponto por Species
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, shape = Species)) +
+ geom_point()
 # ajusta pontos por Species, com formatos diferentes
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, shape = Species)) +
        geom_point() +
        scale_shape_manual(values = c(1, 2, 3))  # círculo vazio, triângulo e cruz
```



default

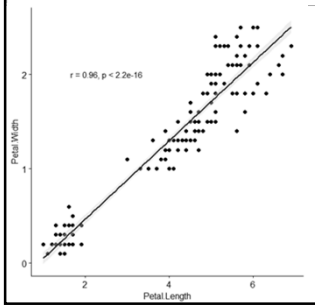
• Formato dos pontos – Species

```
> # Formatos dos ponto - ggplot2
> d <- data.frame(p = c(0:25, 32:127))
> ggplot() +
+ scale_y_continuous(name = "") +
+ scale_x_continuous(name = "") +
+ scale_shape_identity() +
+ geom_point(data = d, mapping = aes(x = p%%16, y = p%/%16, shape = p),
+ size = 5, fill = "red") +
+ geom_text(data = d, mapping = aes(x = p%%16, y = p%/%16 + 0.25,
+ label = p), size = 3)
```



- *Scatter plot* com pacote `ggpubr`:

```
> # Scatter plot com pacote ggpubr
> library(ggpubr)
>   # plot básico
> ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
+           add = "reg.line",# Adiciona reta de regressão
+           conf.int = TRUE,# Adiciona intervalo de confiança
+           add.params = list(color = "blue",
+                             fill = "lightgray")
+           )+
+ stat_cor(method = "pearson", label.x = 2.5, label.y = 2) # Adiciona correlação
```

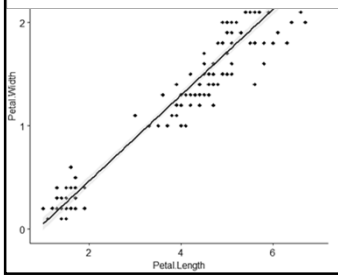


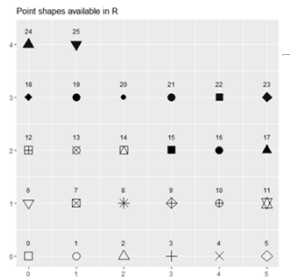
√ Correlação forte entre as variáveis

- *Scatter plot* com pacote `ggpubr`:

√ Formato do ponto customizado

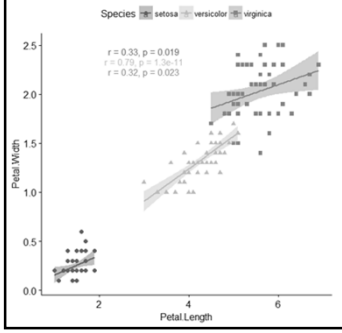
```
>   # Muda formato do ponto
> ggscatter(iris, x = "Petal.Length", y = "Petal.Width", shape = 18,
+           add = "reg.line",# Adiciona reta de regressão
+           conf.int = TRUE,# Adiciona intervalo de confiança
+           add.params = list(color = "blue",
+                             fill = "lightgray")
+           )
>   # formatos de pontos
> show_point_shapes()
```





- *Scatter plot* coloridos por `Species`:

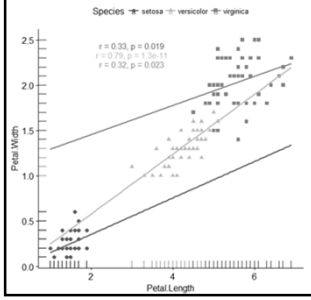
```
>   # gráfico colorido por grupos
> ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
+           add = "reg.line",
+           conf.int = TRUE,
+           color = "Species", palette = "jco",# Cor por Species
+           shape = "Species"# Formato pontos por Species
+          )+
+   stat_cor(aes(color = Species), label.x = 3)# correlação por Species
```



√ Correlações nos grupos são menores que a correlação total

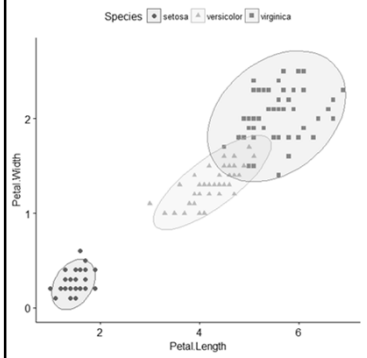
- Retas de regressão estendidas por grupo

```
>   # Estendendo reta de regressão --> fullrange = TRUE
>   # Adicionando densidade marginal (rug) ---> rug = TRUE
> ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
+           add = "reg.line",
+           color = "Species", palette = "jco",
+           shape = "Species",
+           fullrange = TRUE,# Estendendo a reta de regressão
+           rug = TRUE# Adicionando rug marginal
+           )+
+   stat_cor(aes(color = Species), label.x = 3)
```



- *Scatter plot* com elipses de concentração:

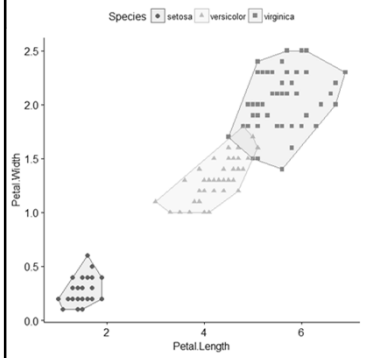
```
 # elipses de concentração
ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
          color = "Species", palette = "jco",
          shape = "Species",
          ellipse = TRUE, ellipse.level = 0.95)
```



√ `ellipse.level`:
- Tamanho da elipse de concentração em condições de normalidade
  - Default é 0,95

- *Scatter plot* com polígonos convexos:

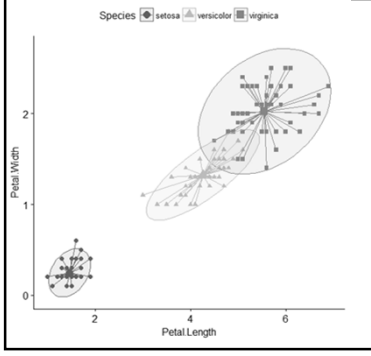
```
>   # Adiciona média dos pontos por grupos e estrelas
> ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
+           color = "Species", palette = "jco",
+           shape = "Species",
+           ellipse = TRUE,
+           mean.point = TRUE
```



√ `ellipse.type`:
- Tipos possíveis: ‘convex’, ‘confidence’ ou os tipos do comando stat_ellipse (ggplot2): ‘t’, ‘norm’, ‘euclid’. Default is ‘norm’.
  - Default é ‘norm’

- *Scatter plot* com médias:

```
>   # Adiciona média dos pontos por grupos e estrelas
> ggscatter(iris, x = "Petal.Length", y = "Petal.Width",
+           color = "Species", palette = "jco",
+           shape = "Species",
+           ellipse = TRUE,
+           mean.point = TRUE,
+           star.plot = TRUE)
```



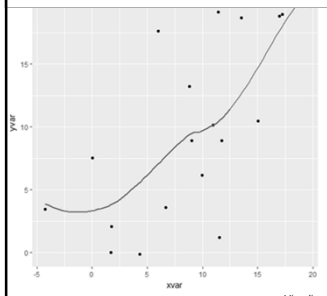
√ `ellipse.type`:
- Tipos possíveis: ‘convex’, ‘confidence’ ou os tipos do comando stat_ellipse (ggplot2): ‘t’, ‘norm’, ‘euclid’. Default is ‘norm’.
  - Default é ‘norm’

Sobreposição de Pontos

- Pontos pode se sobrepor:
  - √ Muitos pontos de dados
  - √ Escalas de dados discretas
- Impossível visualizar se há muitos pontos no mesmo local.

- Geração de dados com ruído

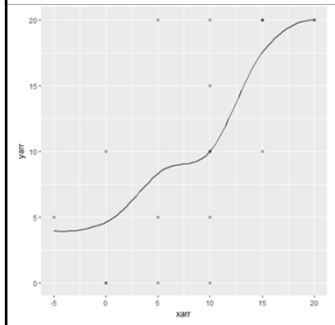
```
> set.seed(955)
> # geração dados crescente com ruído (n = 20)
> dat <- data.frame(cond = rep(c("A", "B"), each = 10),
+                   xvar = 1:20 + rnorm(20, sd = 3),
+                   yvar = 1:20 + rnorm(20, sd = 3))
>  # scatter plot com loess dos dados originais
> ggplot(dat, aes(x = xvar, y = yvar)) +
+ geom_point(shape = 19) +# círculos sólidos
+ geom_smooth(se = FALSE)# suavização por loess
```



√ Dados não apresentam sobreposição?

- Discretizando os dados

```
> # arredondando xvar and yvar para o 5 mais próximo
> dat$xarr <- round(dat$xvar/5)*5
> dat$yarr <- round(dat$yvar/5)*5
> # pontos parcialmente transparente, c/ opacidade 1/4
> ggplot(dat, aes(x = xarr, y = yarr)) +
+ geom_point(shape = 19,# círculos sólidos
+               alpha = 1/4)  +   # opacidade 1/4
+ geom_smooth(method = "loess", se = FALSE)
```

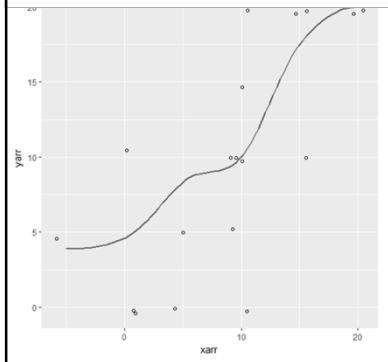


√ Pontos sobrepostos
- Pontos discretizados

√ Para forte overplotting, tente usar valores menores de opacidade

- Scatter plot dos dados com perturbação

```
 # Perturbação dos pontos
ggplot(dat, aes(x = xarr, y = yarr)) +
        geom_point(shape = 1,      # Use hollow circles
        position = position_jitter(width = 1, height = 0.5)) +
        geom_smooth(method = "loess", se = FALSE)
```



√ Intervalo de *jitter*:
- Eixo x: 1
- Eixo y: 0,5

Bubble Plot

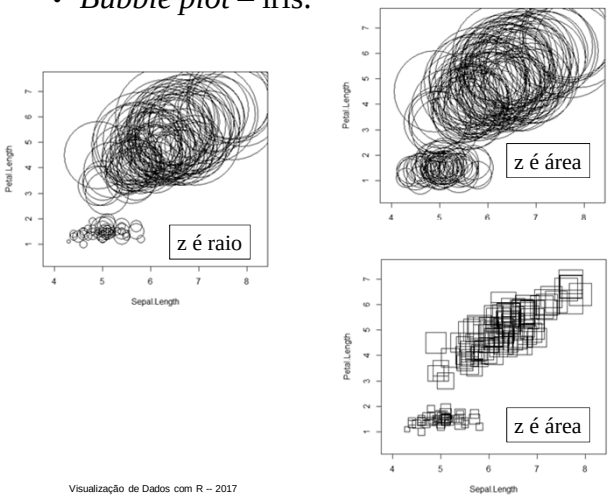
- Gráfico que exibe 3 dimensões de dados
  - √ Extensão do diagrama de dispersão:
  - √ Usa dimensão adicional dos dados para determinar tamanho dos símbolos
- Olho humano percebe diferença entre áreas
  - √ Area de um disco é proporcional ao quadrado do raio
- Há dificuldades para representar 3ª variável com valores negativos ou zero

- *Bubble plot*:
  - √ `Sepal.Width`, `Petal.Length` e `Petal.Width`
  - √ Comandos

```
> # variável z é raio
> with(iris, symbols(Sepal.Length, Petal.Length, circles = Petal.Width))
> # variável z é área
> raio <- sqrt(iris$Petal.Width/pi)
> with(iris, symbols(Sepal.Length, Petal.Length, circles = raio))
> # x = S.L; y = P.L, z = P.W
> with(iris, symbols(Sepal.Length, Petal.Length, circles = raio, inches=0.35,
+ fg = "white", bg = "darkgray", xlab = "Comprimento de sépala",
+ ylab = "Comprimento de pétala"))
> # quadrado com área Petal.Width
> with(iris, symbols(Sepal.Length, Petal.Length, squares = sqrt(Petal.Width),
+   inches=0.5))
```

159

- *Bubble plot* – iris:



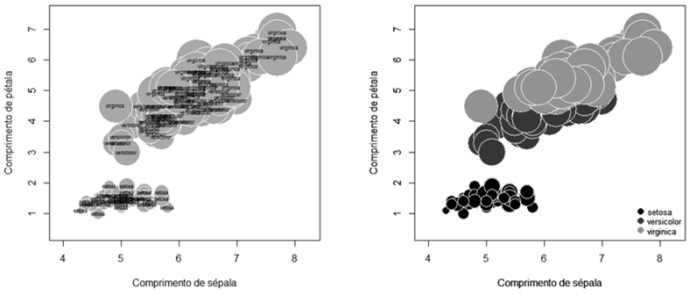
160

- *Bubble plot* – iris:
  - √ x = `Sepal.Length`.
  - √ y = `Petal.Length`.
  - √ z = `Petal.Width`.

```
> # x = S.L; y = P.L, z = P.W e fator
> with(iris, symbols(Sepal.Length, Petal.Length, circles = raio, inches=0.35,
+ fg = "white", bg = "darkgray", xlab = "Comprimento de sépala",
+ ylab = "Comprimento de pétala"))
> text(iris$Sepal.Length, iris$Petal.Length, iris$Species, cex=0.5)
> # x = S.L; y = P.L, z = P.W e fator em 3 cores
> with(iris, {
+ symbols(Sepal.Length, Petal.Length, circles = raio, inches=0.35,
+   fg = "white", bg = unclass(Species),
+   xlab = "Comprimento de sépala", ylab = "Comprimento de pétala")
+ legend("bottomright", levels(iris$Species), pch = rep(20, 3),
+   pt.cex = 2, bg = unique(unclass(iris$Species)),
+   col = unique(unclass(iris$Species)), bty = "n", cex = 0.8)
+ })
```

161

- *Bubble plot* – iris:

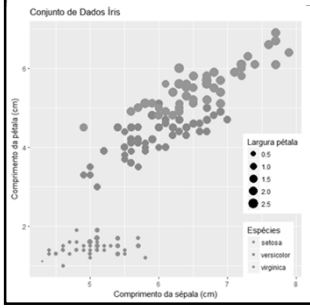


√ Espécie setosa têm pétalas mais estreitas

162

- *Bubble plot* com pacote `ggplot2`:

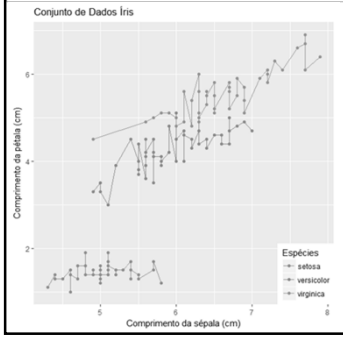
```
> # Scatterplot comprimentos vs. espécies vs.largura pétala (bubble)
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species,
+  size = Petal.Width), alpha = I(0.7)) +# alpha: reduz overplotting
+ geom_point() +
+ xlab("Comprimento da sépala (cm)") +
+   ylab("Comprimento da pétala (cm)") +
+ ggtitle("Conjunto de Dados Íris") +
+ scale_color_discrete(name ="Espécies") +
+ scale_size_continuous(name = "Largura pétala") +
+ theme(legend.position = c(1, 0), legend.justification = c(1,0))
```



√ Flores da espécie setosa têm as pétalas mais estreitas

- *Scatter plot* em linhas:

```
>   # Scatter plot em linhas
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
+ geom_line()+ geom_point() +
+ xlab("Comprimento da sépala (cm)") +
+   ylab("Comprimento da pétala (cm)") +
+ ggtitle("Conjunto de Dados Íris") +
+ scale_color_discrete(name ="Espécies") +
+ theme(legend.position = c(1, 0), legend.justification = c(1,0))
```

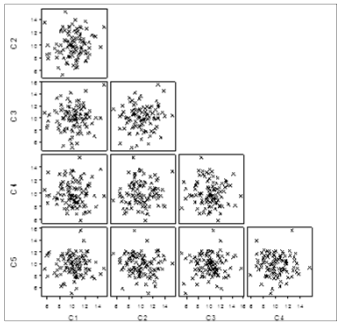


√ Gráfico não faz muito sentido, mas pode ajudar a "enxergar" grupos

Scatter Plot Matrix

- Uso com dados multivariados:
  - √ Principalmente com variáveis métricas
- Com uma matriz de scatter plots podemos:
  - √ Visualizar todas as relações entre pares de variáveis em um único gráfico
  - √ Descrever as relações entre 3 ou mais variáveis

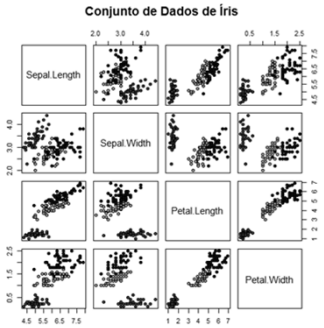
- Diagramas de dispersão bivariados
  - √ Apresentação em forma matricial
  - √ Calcular coeficiente de correlação de cada par de variáveis



- √ Há associação entre as variáveis? Qual sua forma?
- √ A variabilidade é homogênea?
- √ Observam-se clusters?
- √ Há indicação de outliers?

- *Scatter plot matrix* – iris:

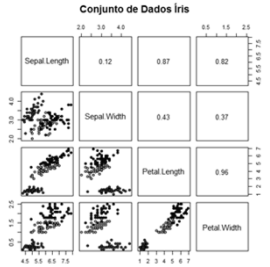
```
> pairs(iris[1:4], main = "Conjunto de Dados de Íris", pch = 21,
+ bg = c("red", "green3", "blue")[unclass(iris$Species)])
```



√ Quais gráficos aparentam discriminar melhor os grupos?
√ Há relações entre as medidas morfológicas?

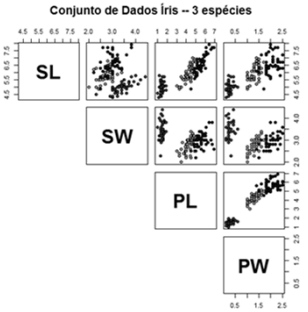
- *Scatter plot matrix* com correlações:

```
> # função para personalização do painel
> painel.pearson <- function(x, y, ...) {
+ horizontal <- (par("usr")[1] + par("usr")[2]) / 2;
+ vertical <- (par("usr")[3] + par("usr")[4]) / 2;
+ text(horizontal, vertical, format(abs(cor(x,y)), digits=2), cex = 1.2,
+ font = 1)
+ }
> # Scatter plot matrix com correlações
> pairs(iris[1:4], main = "Conjunto de Dados Íris", pch = 21,
+ bg = c("red","green3","blue")[unclass(iris$Species)],
+ upper.panel = painel.pearson)
```



- *Scatter plot matrix* com diagonal modificada:

```
> # Scatterplot matrix com diagonal modificada
> pairs(iris[1:4], main = "Conjunto de Dados Íris -- 3 espécies", pch = 21,
+ bg = c("red", "green3", "blue")[unclass(iris$Species)],
+ lower.panel = NULL, labels = c("SL", "SW", "PL", "PW"), font.labels = 2,
+ cex.labels = 3.0
```

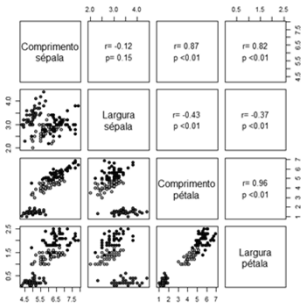


- *Scatter plot matrix* com correlação e p-valor:
  - √ Função para personalização do painel

```
> # Scatterplot matrix com correlação e p-valor
>
> # função para personalização do painel
> painel.cor <- function(x, y, digits = 2, cex.cor, ...){
+   usr <- par("usr"); on.exit(par(usr))
+   par(usr = c(0, 1, 0, 1))
+   # coeficiente de correlação
+  r <- cor(x, y)
+   txt <- format(c(r, 0.123456789), digits = digits)[1]
+   txt <- paste("r= ", txt, sep = "")
+   text(0.5, 0.6, txt, cex = 1.2)
+   # cálculo do p-valor
+   p <- cor.test(x, y)$p.value
+   txt2 <- format(c(p, 0.123456789), digits = digits)[1]
+  txt2 <- paste("p= ", txt2, sep = "")
+   if(p < 0.01) txt2 <- paste("p ", "<0.01", sep = "")
+   text(0.5, 0.4, txt2, cex = 1.2)
+ }
```

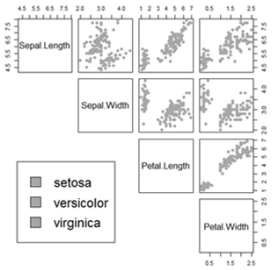
√ *Scatter plot matrix* com correlação e p-valor:

```
> # scatter plot matrix
> pairs(iris[,1:4], pch = 21,
+ bg = c("red", "green3", "blue")[unclass(iris$Species)],
+ upper.panel = painel.cor,
+ labels = c("Comprimento\nsépala", "Largura\nsépala",
+ "Comprimento\npétala", "Largura\npétala"))
```



• *Scatter plot matrix* com legenda:

```
> library(colorspace) # cores melhores
> species_labels <- iris[,5]
> species.cor <- rev(rainbow_hcl(3))[as.numeric(iris$Species)]
> # Plot um SPloM:
> pairs(iris[-5], col = species.cor, lower.panel = NULL,
+ cex.labels = 1.7, pch = 19, cex = 1.2)
> par(xpd = TRUE)
> legend(x = 0.05, y = 0.4, cex = 1.5, legend = as.character(levels(iris$Species)),
+      fill = unique(species.cor))
> par(xpd = NA)
```



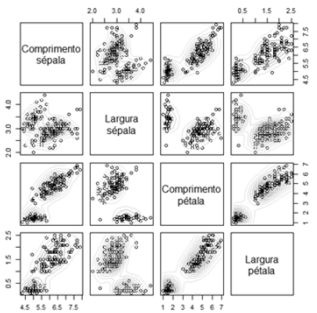
• *Scatter plot matrix* com estimativas de densidade bivariada:

√ Função para estimativas de densidade bivariada

```
> library(MASS)
> library(colorspace)
>
> # função para estimativas densidade bivariada por núcleo
> painel.dens <- function(x,y) {
+      points(x,y)
+      k <- kde2d(x,y)# package: MASS
+      cnt <- contourLines(k$x, k$y, k$z)
+      n <- length(cnt)
+      cols <- rev(sequential_hcl(n))# package: colorspace
+      for( i in seq_len(n) ) lines(cnt[[i]], col=cols[i])
+ }
```


√ *Scatter plot matrix* com densidade bivariada:

```
> # Scatter plot matrix
> pairs(iris[,1:4], panel = painel.dens,
+ labels = c("Comprimento\nsépala", "Largura\nsépala",
+ "Comprimento\npétala", "Largura\npétala"))
```

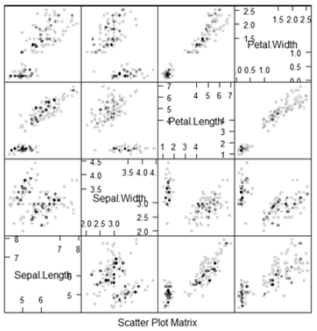


√ Estimativa densidade está codificada por cor

√ Pode ser conveniente quando houver muitos empates

√ *Scatter plot matrix* com densidade bivariada:

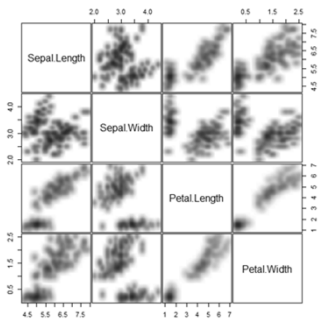
```
> # Scatterplot Matrix com estimativas de densidades bivariadas
> # estimativa densidade codificada por cor
>
> library(hexbin)
> hexplom(iris[,1:4])
```



- √ Estimativas de densidade codificadas por áreas hexagonais
- √ Facilita percepção das densidades

√ *Scatter plot matrix* com densidade bivariada em intensidade de cor:

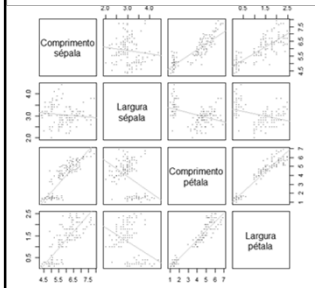
```
> # Scatter plot matrix com densidade em intensidade de cores
> pairs(iris[, 1:4], panel = function(...) smoothScatter(..., nrpoints = 0,
+ add = TRUE), gap = 0.2)
```



- √ Estimativas de densidade codificadas por intensidade de cor

• *Scatter plot matrix* com ajuste linear:

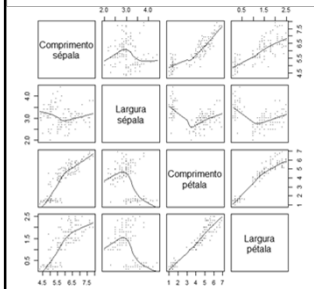
```
> # Scatterplot matrix com ajuste linear
>   # função painel
> panel.reg <- function(x, y, ...){
+ points(x, y, ...)
+ abline(lm(y ~x), col = "grey")
+ }
>   # scatter plot matrix
> pairs(iris[,1:4], panel = panel.reg, pch = ".", cex = 1.5,
+ labels = c("Comprimento\nsépala", "Largura\nsépala",
+ "Comprimento\npétala", "Largura\npétala"))
```



- √ Permite verificar:
  - − Associação linear entre pares de variáveis
  - − Influência de pontos ou grupos no ajuste linear

• *Scatter plot matrix* com suavização por núcleo:

```
> # Scatterplot matrix com suavização por núcleo
>
> pairs(iris[,1:4], panel = panel.smooth, pch = ".", cex = 1.5,
+ labels = c("Comprimento\nsépala", "Largura\nsépala",
+ "Comprimento\npétala", "Largura\npétala"))
```

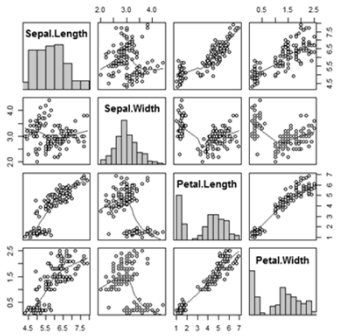


- √ Permite verificar:
  - − Associação não lineares entre pares de variáveis
  - − Presença de grupos

- *Scatter plot matrix* com suavização por núcleo e histograma na diagonal:
  - √ Comandos

```
> # Scatterplot matrix - suavização p/ núcleo e digonal c/ histograma
>
>   # função painel
> panel.hist <- function(x, ...){
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(usr[1:2], 0, 1.5) )
+     h <- hist(x, plot = FALSE)
+     breaks <- h$breaks; nB <- length(breaks)
+     y <- h$counts; y <- y/max(y)
+     rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
+ }
>   # scatter plot matrix
> pairs(iris[1:4], panel = panel.smooth,
+       cex = 1, pch = 21, bg = "light blue",
+       diag.panel = panel.hist, cex.labels = 1.4, font.labels = 2)
```

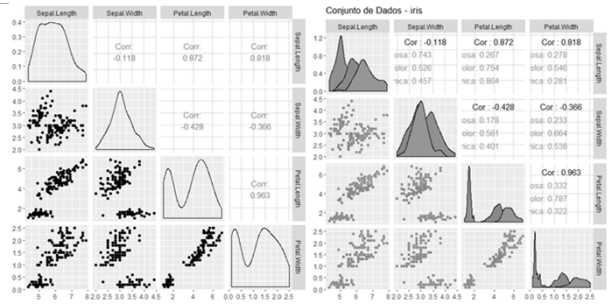

√ *Scatter plot matrix*
- Diagonal modificada pelo usuário



√ Histograma para visualização das distribuições univariadas

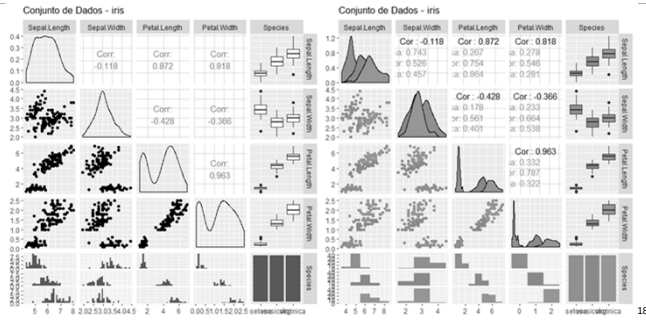
- *Scatter plot matrix* com pacote `GGally`:

```
> library(GGally)
> # default
> ggpairs(iris[,1:4])
>   # SPloM - cores por Species
> ggpairs(data = iris, # data.frame com as variáveis
+         columns = 1:4, # colunas a serem plotadas, default: todas
+         title = "Conjunto de Dados - iris",
+   mapping = ggplot2::aes(colour = Species))
```



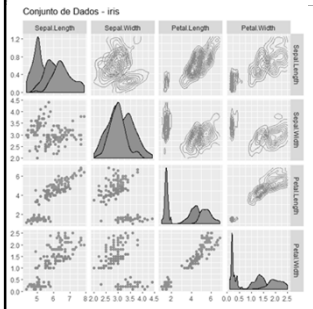
- *Scatter plot matrix* com pacote `GGally`:

```
>   # Todas as variáveis
> ggpairs(data = iris, title = "Conjunto de Dados - iris")
>   # Todas as variáveis - cores por Species
> ggpairs(data = iris, title = "Conjunto de Dados - iris",
+   mapping = ggplot2::aes(colour = Species),
+     lower = list(combo = wrap("facethist", binwidth = 1))
+   )
```



• Customizando com `GGally` – Plot 1

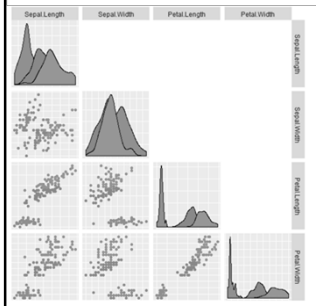
```
>   # controlando tipos de plot - plot 1
> p1 <- ggpairs(data = iris,
+   columns = 1:4,
+   upper = list(continuous = "density"),
+   lower = list(combo = "facetdensity"),
+   title = "Conjunto de Dados - iris",
+   mapping = ggplot2::aes(colour = Species))
> print(p1)
```



√ Inferior: pontos
√ Superior: densidade bivariada
√ Diagonal: densidades por Species

• Customizando com `GGally` – Plot 2

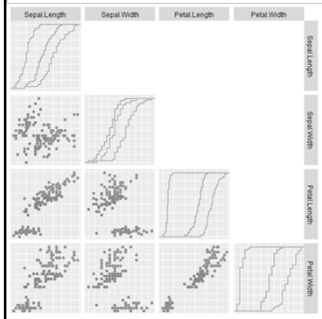
```
>   # controlando tipo de plot - plot 2
> p2 <- ggpairs(data = iris,
+       columns = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
+   mapping = ggplot2::aes(colour = Species),
+       lower = list(continuous='points'),
+   axisLabels = 'none',
+   upper = list(continuous = 'blank'))
> print(p2)
```



√ Inferior: pontos
√ Superior: Vazio
√ Diagonal: densidades por Species

• Inserção de plot customizado no Plot 2

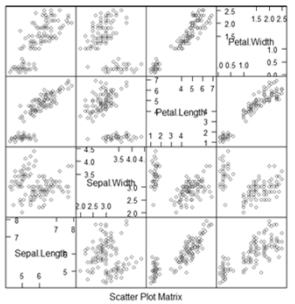
```
>   # inserção de plot no scatter plot matrix
> for (i in 1:4) {
+   p2 <- putPlot(p2, ggplot(data = data.frame(x = iris[,i], Species =
iris$Species),
+                            aes(x = x, colour = Species)) + stat_ecdf(), i,i)
+ }
> print(p2)
```



√ Inferior: pontos
√ Superior: Vazio
√ Diagonal: função de distribuição acumulada empírica

• *Scatter plot matrix* com pacote `lattice`

```
> # Scatterplot matrices com lattice
>
> library(lattice)
>
>   # default
> splom(~iris[1:4])
```



- *Scatter plot matrix* com pacote `car`:

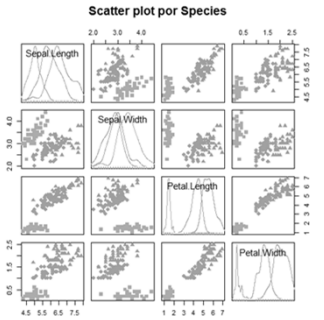
√ Plot 1

```
> # Scatterplot matrices com car
> library(car)
> library(RColorBrewer)
>   # ajuste das cores
> minhas.cores <- brewer.pal(nlevels(as.factor(iris$Species)), "Set2")
>   # scatter plot matrix – plot 1
> scatterplotMatrix(~ .|Species, data = iris, reg.line = "", legend.plot = FALSE,
+ smoother = "", col = minhas.cores , smoother.args = list(col = "grey"),
+ cex = 1.2, pch = c(15, 16, 17),
+ main = "Scatter plot por Species")
```

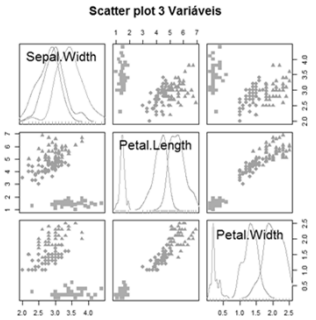
√ Plot 2

```
> # scatter plot matrix - plot 2
> scatterplotMatrix(~ Sepal.Width + Petal.Length + Petal.Width|Species,
+ data = iris, reg.line = "", legend.plot = FALSE,
+ smoother = "", col = minhas.cores , smoother.args = list(col = "grey"),
+ cex = 1.2, pch = c(15, 16, 17),
+ main = "Scatter plot 3 Variáveis")
```


Plot 1

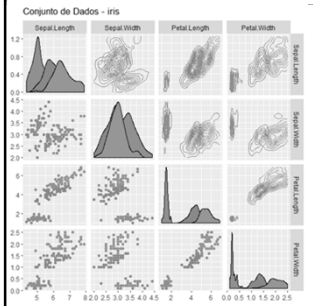


Plot 2



- Customizando com `GGally` – Plot 1

```
>   # controlando tipos de plot - plot 1
> p1 <- ggpairs(data = iris,
+   columns = 1:4,
+   upper = list(continuous = "density"),
+   lower = list(combo = "facetdensity"),
+   title = "Conjunto de Dados - iris",
+   mapping = ggplot2::aes(colour = Species))
> print(p1)
```



√ Inferior: pontos
√ Superior: densidade bivariada
√ Diagonal: densidades por Species

Plots Bivariados

- Pacote xda:
  √ `library(devtools)`
  √ `install_github("ujjwalkarn/xda")`
- Plots de variáveis contra uma variável (resposta)

- *Plot* bivariado – pacote `xda`:

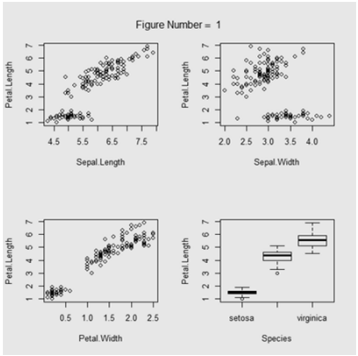
```
> library(devtools)
> install_github("ujjwalkarn/xda")
> library(xda)
> # resumo de todas as variáveis quantitativas
> numSummary(iris)
              n mean    sd max min range nunique nzeros  iqr lowerbound
Sepal.Length 150 5.84 0.828 7.9 4.3   3.6      35      0 1.30       3.15
Sepal.Width  150 3.06 0.436 4.4 2.0   2.4      23      0 0.50       2.05
Petal.Length 150 3.76 1.765 6.9 1.0   5.9      43      0 3.55      -3.72
Petal.Width  150 1.20 0.762 2.5 0.1   2.4      22      0 1.50      -1.95
             upperbound noutlier kurtosis skewness mode miss miss%   1%   5%
Sepal.Length       8.35        0   -0.606    0.309  5.0    0     0 4.40 4.60
Sepal.Width        4.05        4    0.139    0.313  3.0    0     0 2.20 2.34
Petal.Length      10.42        0   -1.417   -0.269  1.4    0     0 1.15 1.30
Petal.Width        4.05        0   -1.358   -0.101  0.2    0     0 0.10 0.20
             25%  50% 75%  95%  99%
Sepal.Length 5.1 5.80 6.4 7.25 7.70
Sepal.Width  2.8 3.00 3.3 3.80 4.15
Petal.Length 1.6 4.35 5.1 6.10 6.70
Petal.Width  0.3 1.30 1.8 2.30 2.50
> # resumo de todas as variáveis qualitativas
> charSummary(iris)
         n miss miss% unique                       top5levels:count
Species 150    0     0      3 setosa:50, versicolor:50, virginica:50
```


- *Plot* Tabela de dupla entrada entre `Sepal.Length` e `Species`:

```
> # análise bivariada entre 'Species' e 'Sepal.Length'
> bivariate(iris,'Species','Sepal.Length')
  bin_Sepal.Length setosa versicolor virginica
1       (4.3,5.2]     39          5         1
2       (5.2,6.1]     11         29        10
3         (6.1,7]      0         16        27
4         (7,7.9]      0          0        12
```


- *Plot* de todas as variáveis vs. `Petal.Length`:

```
> # plot de todas as variáveis contra Petal.Length
> Plot(iris,'Petal.Length')
```

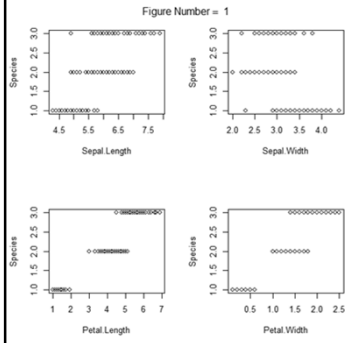


√ Gráficos bivariados com `Petal.Length` aparentam discriminar estratos de `Species`

√ Padrões interessantes podem ser vistos e usados para modelo preditivo

- *Plot* de todas as variáveis vs. `Species`:

```
> # plot de todas as variáveis contra Species
> Plot(iris,'Species')
```

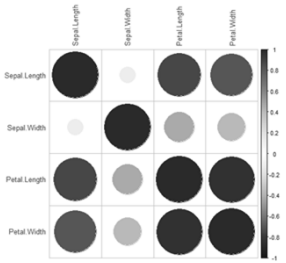


√ Resposta é categórica

Visualização de Dados Multivariados

- Correlograma:

```
        # Matriz de correlações
(iris.cor <- cor(iris[-5]))
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
> library(corrplot)
> # correlograma - círculo
> corrplot(iris.cor, method = "circle")
```



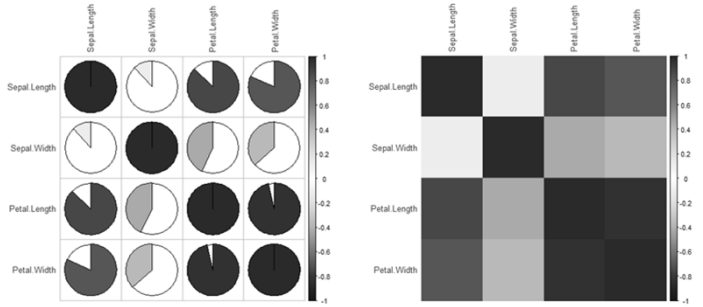
√ círculos

205

Visualização de Dados com R -- 2017

- Correlograma:

```
> # correlograma - pizza
> corrplot(iris.cor, method = "pie")
> # coorelograma - cor
> corrplot(iris.cor, method = "color")
```

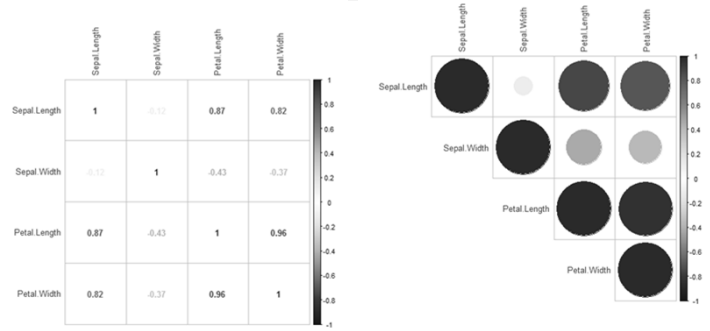


206

Visualização de Dados com R -- 2017

- Correlograma:

```
> # correlograma - valores
> corrplot(iris.cor, method = "number")
> # correlograma - superior
> corrplot(iris.cor, type = "upper")
```

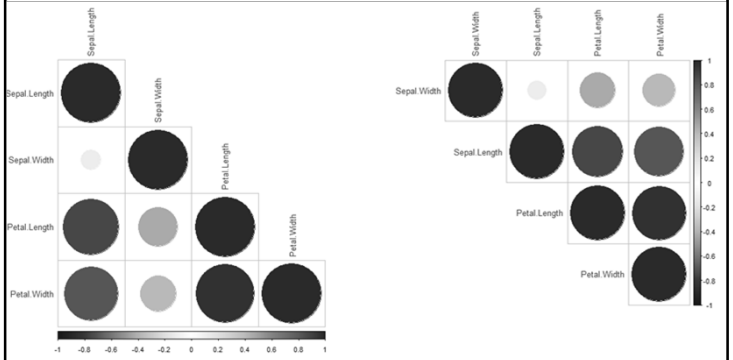


207

Visualização de Dados com R -- 2017

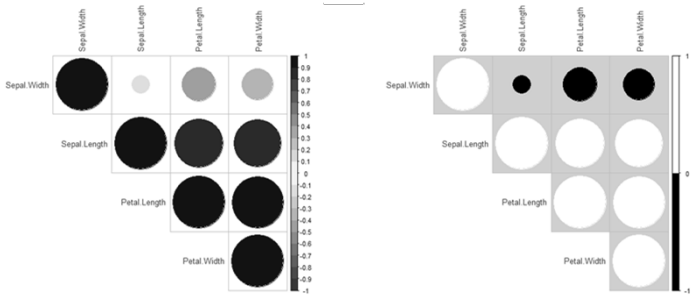
- Correlograma:

```
> # correlograma - inferior
> corrplot(iris.cor, type = "lower")
> # correlograma c/ reordenação por hclust
> corrplot(iris.cor, type="upper", order = "hclust")
```



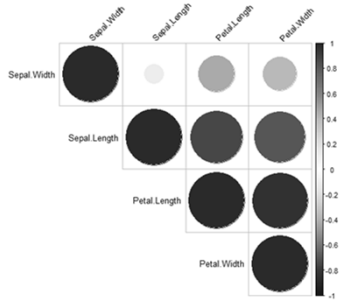
- Correlograma:

```
> # usando espectro de cores diferente
> col <- colorRampPalette(c("red", "white", "blue"))(20)
> corrplot(iris.cor, type = "upper", order = "hclust", col = col)
> # Mudando cor de fundo para lightblue
> corrplot(iris.cor, type = "upper", order = "hclust", col = c("black", "white"),
+          bg = "lightblue")
```



- Correlograma:

```
> # Mudando a cor e a rotação dos rótulos
> corrplot(iris.cor, type = "upper", order = "hclust", tl.col = "black",
+ tl.srt = 45)
> #tl.col (cor do texto) e tl.srt (rotação texto)
```



- Correlograma:
  - √ Função para cálculo de p-valor

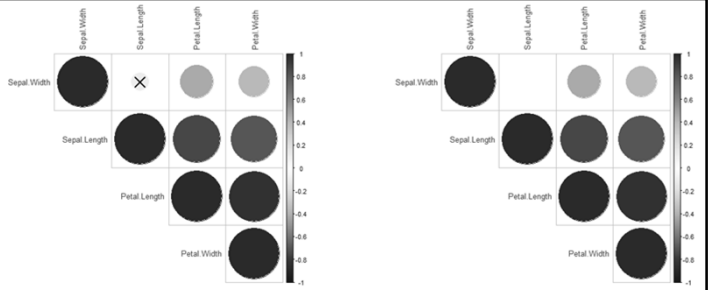
```
> # Função para cálculo do p-valor das correlações
> cor.mteste <- function(mat, ...) {
+     mat <- as.matrix(mat)
+     n <- ncol(mat)
+     p.mat<- matrix(NA, n, n)
+     diag(p.mat) <- 0
+     for (i in 1:(n - 1)) {
+         for (j in (i + 1):n) {
+             tmp <- cor.test(mat[, i], mat[, j], ...)
+             p.mat[i, j] <- p.mat[j, i] <- tmp$p.value
+         }
+     }
+   colnames(p.mat) <- rownames(p.mat) <- colnames(mat)
+   p.mat
+ }
```

  - √ Matriz dos p-valores das correlações

```
> # matriz dos p-valores das correlações
> p.mat <- cor.mteste(iris[-5])
> head(p.mat)
 Sepal.Length  Sepal.Width Petal.Length  Petal.Width
Sepal.Length 0.000000e+00 1.518983e-01 1.038667e-47 2.325498e-37
Sepal.Width  1.518983e-01 0.000000e+00 4.513314e-08 4.073229e-06
Petal.Length 1.038667e-47 4.513314e-08 0.000000e+00 4.675004e-86
Petal.Width  2.325498e-37 4.073229e-06 4.675004e-86 0.000000e+00
```

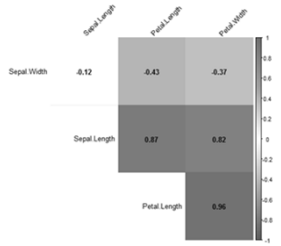
- Correlograma:

```
> # Agregando nível de significância ao correlograma
> corrplot(iris.cor, type="upper", order="hclust", p.mat = p.mat,
+ sig.level = 0.01)
> # Deixando em branco coeficiente não significante
> corrplot(iris.cor, type = "upper", order = "hclust", p.mat = p.mat,
+ sig.level = 0.01, insig = "blank")
```



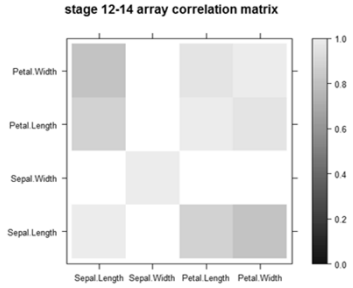
- Correlograma:

```
> # Customizando o correlograma
> col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD",
+ "#4477AA"))
> corrplot(iris.cor, method="color", col=col(200), type="upper", order="hclust",
+          addCoef.col = "black", # Adiciona coeficiente de correlação
+          tl.col="black", tl.srt=45, # Rotação e cor de texto rótulo
+          # Combinação com significância
+          p.mat = p.mat, sig.level = 0.01, insig = "blank",
+          diag=FALSE # elimina valores da diagonal principal
+          )
```



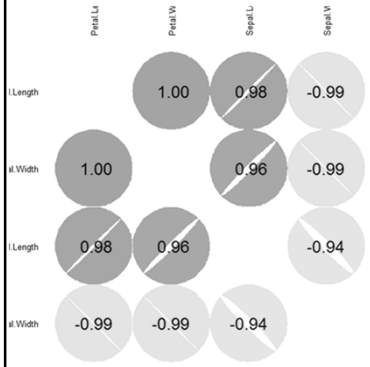
- Matriz de Correlações – pacote `lattice`:

```
> library(lattice)
> rgb.palette <- colorRampPalette(c("blue", "yellow"), space = "rgb")
> levelplot(iris.cor, main = "stage 12-14 array correlation matrix",
+ xlab = "", ylab = "", col.regions = rgb.palette(120),
+ cuts = 100, at = seq(0, 1, 0.01)
+ )
```



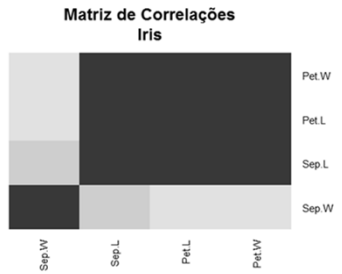
- Matriz de Correlações – pacote `lattice`:

```
> source("https://github.com/JVAdams/jvamisc/blob/master/R/plotcor.R")
> library(plotrix)
> library(seriation)
> library(MASS)
> plotcor(cor(iris.cor), mar = c(0.1, 4, 4, 0.1))
```



- Mapa de Calor:

```
> library(gplots)
> library(RColorBrewer)
> heatmap.2(iris.cor, col = brewer.pal(9, "GnBu"), trace ="none",
+     key = FALSE, dend = "none", cexCol = 1.1, cexRow = 1.1, srtCol = 90,
+     labRow = c("Sep.L", "Sep.W", "Pet.L", "Pet.W"),
+     labCol = c("Sep.L", "Sep.W", "Pet.L", "Pet.W"),
+     main = "\n\n\nMatriz de Correlações\nIris")
```



- Gráfico em html:
  - √ Gráfico 1:

```
> library(plotly)
> p1 <- plot_ly(data = iris, x =~Sepal.Length, y =~Sepal.Width, split =~Species,
+   showlegend = F)
> p2 <- plot_ly(data = iris, x =~Sepal.Length, y =~Sepal.Width, split =~Species,
+   showlegend = T)
> subplot(p1,p2)
```

  - √ Gráfico 2:

```
> p1 <-
+   iris %>%
+   group_by(Species) %>%
+   plot_ly(x =~Sepal.Length, color = ~Species) %>%
+   add_markers(y = ~Sepal.Width)
> p2 <-
+   iris %>%
+   group_by(Species) %>%
+   plot_ly(x=~Sepal.Length, color= ~Species) %>%
+   add_markers(y =~Sepal.Width, showlegend = F)
> subplot(p1,p2)
```


Exemplos de Aplicação

Conjunto de Dados – honolulu

- Doenças cardiovasculares
  - √ 7.683 casos coletados no Havaí em 1969
  - √ Fator de exposição: fumante
- Universo:
  - √ Homens doentes com idade entre 45 e 67 anos
  - √ Média de Idade da população: 54,36 anos
- Tamanho da amostra: 100
- Dados: *honolulu.txt*

√ Variáveis codificadas:

- educacao : nível de instrução (1 = nenhuma, 2 = primeiro grau incompleto; 3 = primeiro grau completo; 4 = segundo grau completo; 5 = curso técnico; 6 = curso superior)
- peso, em Kg
- altura, em cm
- Idade, em anos
- fumante : status de fumante (0 = não; 1 = sim)
- atividade : atividade física em casa (1 = sedentário; 2 = moderada; 3 = alta)
- glicose: nível de glicose no sangue em mg percentuais
- colesterol: nível de colesterol sérico em miligramas percentuais
- pressao: pressão sanguínea sistólica, em mmHg

271

- Importação do conjunto de dados:

```
> dados <- read.table("honolulu.txt", head = TRUE)
> honolulu <- dados[-1]
> dim(honolulu)
[1] 100  9
> str(honolulu)
'data.frame':   100 obs. of  9 variables:
 $ educacao  : int  2 1 1 2 2 4 1 3 5 2 ...
 $ peso      : int  70 60 62 66 70 59 47 66 56 62 ...
 $ altura    : int  165 162 150 165 162 165 160 170 155 167 ...
 $ idade     : int  61 52 52 51 51 53 61 48 54 48 ...
 $ fumante   : int  1 0 1 1 0 0 0 1 0 0 ...
 $ atividade : int  1 2 1 1 1 2 1 1 2 1 ...
 $ glicose   : int  107 145 237 91 185 106 177 120 116 105 ...
 $ colesterol: int  199 267 272 166 239 189 238 223 279 190 ...
 $ pressao   : int  102 138 190 122 128 112 128 116 134 104 ...
> head(honolulu)
  educacao peso altura idade fumante atividade glicose colesterol pressao
1        2   70    165    61       1         1     107        199     102
2        1   60    162    52       0         2     145        267     138
3        1   62    150    52       1         1     237        272     190
4        2   66    165    51       1         1      91        166     122
5        2   70    162    51       0         1     185        239     128
6        4   59    165    53       0         2     106        189     112
```

272

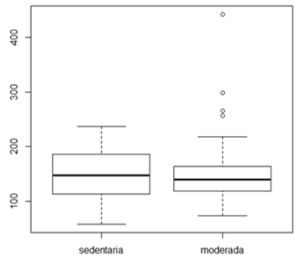
- Preparação do conjunto de dados:

```
> # Transformação variáveis em fatores
> nomes.col <- c("educacao", "fumante", "atividade")
> honolulu[nomes.col] <- lapply(honolulu[nomes.col] , factor)
>
> # Renomeação níveis dos fatores - educacao
> edu.niveis <- c("N", "1I", "1C", "2C", "T", "S")
> levels(honolulu$educacao) <- edu.niveis
> # Renomeação níveis dos fatores - atividade
> ativ.niveis <- c("sedentaria", "moderada", "alta")
> levels(honolulu$atividade) <- ativ.niveis
> # Renomeacao níveis e reordenação níveis - fumante
> fuma.niveis <- c("N", "S")
> levels(honolulu$fumante) <- fuma.niveis
> # transformacao em fator ordenado
> honolulu$educacao <- ordered(honolulu$educacao)
> honolulu$atividade <- ordered(honolulu$atividade)
> head(honolulu)
  educacao peso altura idade fumante  atividade glicose colesterol pressao
1       1I   70    165    61       S sedentaria     107        199     102
2        N   60    162    52       N   moderada     145        267     138
3        N   62    150    52       S sedentaria     237        272     190
4       1I   66    165    51       S sedentaria      91        166     122
5       1I   70    162    51       N sedentaria     185        239     128
6       2C   59    165    53       N   moderada     106        189     112
```

273

- *Boxplot*: `glicose` vs. `atividade`:

```
> # Default
> boxplot(glicose ~ atividade, data = honolulu)
```

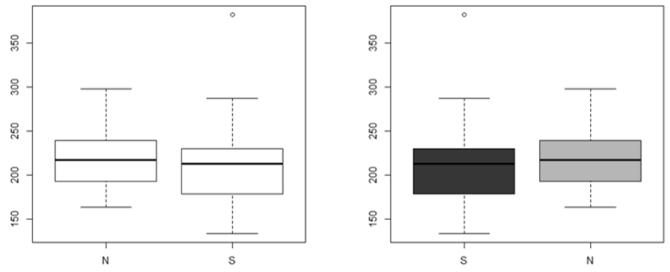


√ Há diferença no nível médio de glicose entre os grupos
√ Há diferença das variabilidades dos grupos?
√ Há *outliers*?

274

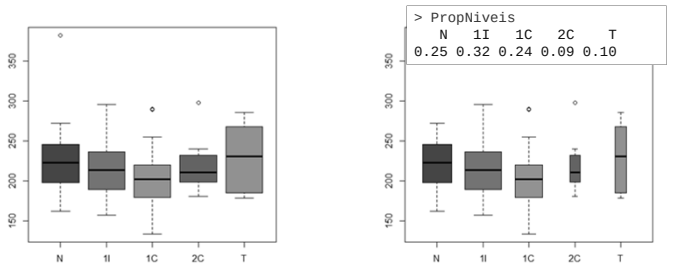
- *Boxplot*: `colesterol` vs. `fumo`:

```
> # Default
> boxplot(colesterol ~ fumante, data = honolulu)
> # Boxplot com caixas coloridas (inversão ordem fatores)
> fator <- relevel(honolulu$fumante, "S")
> boxplot(colesterol ~ fator, data = honolulu, col = c("red", "green"))
```



- *Boxplot*: `colesterol` vs. `educacao`:

```
> # Boxplot com caixas coloridas - qualquer qte. categorias
> library(RColorBrewer)
> QteNiveis <- length(levels(honolulu$educacao))
> cores <- brewer.pal(n = QteNiveis, name = "Set1")
> boxplot(colesterol ~ educacao, data = honolulu, col = cores)
>  # Boxplot com caixas proporcionais
> PropNiveis <- prop.table(table(honolulu$educacao))
> boxplot(colesterol ~ educacao, data = honolulu, width = PropNiveis, col = cores)
```

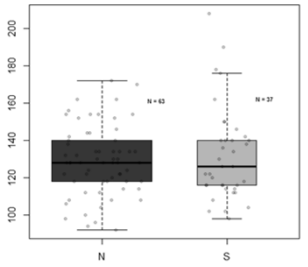



```
> PropNiveis
    N   1I   1C   2C    T
0.25 0.32 0.24 0.09 0.10
```


- *Boxplot*: `pressao` vs. `fumante`:
  - √ *Boxplot* customizado

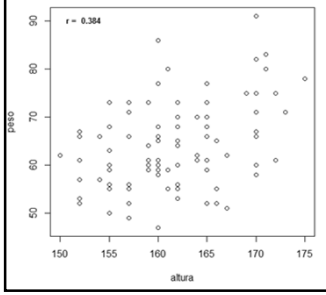
```
> # Boxplot customizado - pontos e caixas proporcionais
> PropNiveis <- prop.table(table(honolulu$fumante))
> boxplot(pressao ~ fumante, data = honolulu, col = c("red", "green"),
+ width = PropNiveis, outpch = NA)
>   # Acrescenta pontos
> niveis <- levels(honolulu$fumante)
> for(i in 1:length(niveis))
+ {
+   este.nivel <- niveis[i]
+   valores <- honolulu[honolulu$fumante == este.nivel, "pressao"]
+      # Adiciona perturbação, proporcional à N em cada nível (eixo X)
+   perturbacao <- jitter(rep(i, length(valores)), amount = PropNiveis[i]/2)
+   points(perturbacao, valores, pch = 20, col = rgb(0, 0, 0, 0.2))
+      # Adiciona texto do tamanho dos grupos
+   tipica <- min(max(valores), median(valores) + IQR(valores)*1.5)
+   text(i + PropNiveis[i]/2, tipica, cex = 0.6, font = 2, pos = 4,
+ labels = paste("N = ", length(valores), sep=""))
+ }
```


- *Boxplot* customizado:
  - √ Pontos, caixas proporcionais ao tamanho dos grupos, cores



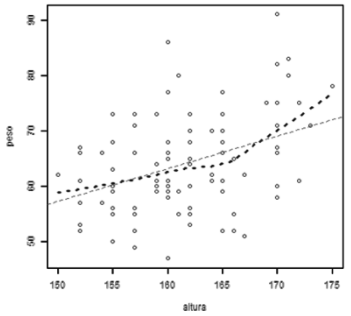
- *Scatter plot* – Doença cardiovascular:

```
> # Box-plots
> variaveis <- names(iris[-5])
> par(mfrow = c(2, 2))
> for(i in 1: length(variaveis)) {
+ with(iris, {
+ dados <- eval(parse(text = variaveis[i]))
+ boxplot(dados ~ Species, data = iris, main = variaveis[i], cex.axis = 0.85)
+ })
+ }
```



√ Qual a relação entre o peso e a altura das pessoas?
√ Percebem-se 'clusters'?
√ Há diferenças na variabilidade de uma variável, considerados os valores da outra?
√ Há valores atípicos?

- Relação entre as variáveis:
  √ Reta de regressão e suavização

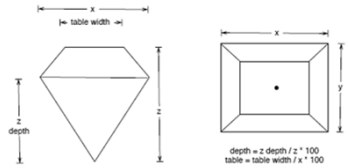


Conjunto de Dados – `diamonds`

- Preços e outros atributos de diamantes
  √ Conjunto de dados com informações (preços e outros 9 atributos) sobre 53.940 diamantes
  √ Fonte não informada
- Dados: `diamonds{ggplot2}`

√ Variáveis codificadas:
- price: preço, em US$ ($326 a $18.823)
- carat: peso do diamante, em quilates (0,2 a 5,01)
- cut: qualidade do corte (Fair, Good, Very Good, Premium, Ideal)
- colour: cor do diamante (de J = pior para D = melhor)
- clarity: medida de quão claro o diamante é (I1 = pior, SI1, SI2, VS1, VS2, VVS1, VVS2, IF = melhor)
- x: comprimento, em mm (0 a 10,74)
- y: largura, em mm (0 a 58,9)
- z: espessura, em mm (0 a 31,8)
- depth: espessura total percentual (43 a 79) $\frac{z}{\frac{x+y}{2}} = \frac{2\,z}{x+y}$
- table: largura do topo do diamante em relação ao ponto mais largo (43 a 95)

- Desenho esquemático diamante

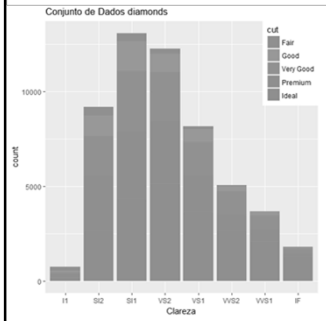


- Importação do conjunto de dados:

```
> library(ggplot2)
> dim(diamonds)
[1] 53940    10
> str(diamonds)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       53940 obs. of  10 variables:
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
> head(diamonds)
# A tibble: 6 x 10
  carat       cut color clarity depth table price     x     y     z
  <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23     Ideal     E     SI2  61.5    55   326  3.95  3.98  2.43
2  0.21   Premium     E     SI1  59.8    61   326  3.89  3.84  2.31
3  0.23      Good     E     VS1  56.9    65   327  4.05  4.07  2.31
4  0.29   Premium     I     VS2  62.4    58   334  4.20  4.23  2.63
5  0.31      Good     J     SI2  63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good     J    VVS2  62.8    57   336  3.94  3.96  2.48
> head(diamonds)
```


- Gráfico de barras – `clarity` *versus* `cut`

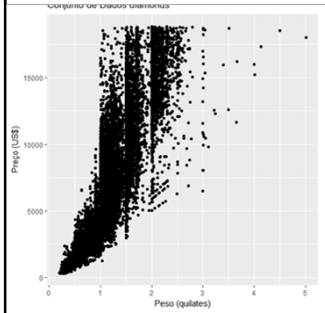
```
> # Diagrama de barras de 'clareza' categorizado com 'corte'
> ggplot(diamonds, aes(clarity, fill = cut)) + geom_bar() +
+ xlab("Clareza") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_color_discrete(name = "Corte") +
+ theme(legend.position = c(1, 1), legend.justification = c(1,1))
```



√ Qual a relação entre as duas variáveis?
√ Frequências de `cut` mudam nos níveis de `clarity`?

- *Scatter plot* – `price` *versus* `carat`

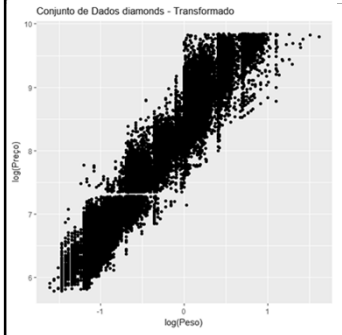
```
> # Peso vs. Preço
> p <- ggplot(data = diamonds, aes(x = carat, y = price)) +
+ geom_point() +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("Conjunto de Dados diamonds")
> p
```



√ Gráfico aponta forte relação entere as variáveis
  − Há outliers importantes
  − Estrias verticais interessante
√ Relação aparenta ser exponencial
  − Interessante transformar os dados

- *Scatter plot* – `log price` *versus* `carat`

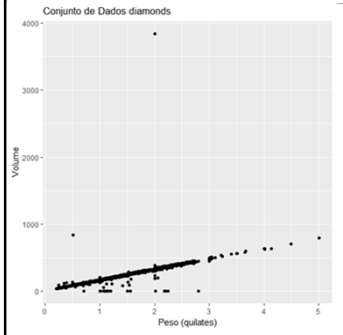
```
> # Transformação dos dados
> ggplot(data = diamonds, aes(x = log(carat), y = log(price))) +
+ geom_point() +
+ xlab("log(Peso)") +
+ ylab("log(Preço)") +
+ ggtitle("Conjunto de Dados diamonds - Transformado")
```



√ Relação agora parece ser linear
- Necessária cautela devido a *overplotting*

- Relação entre volume e peso

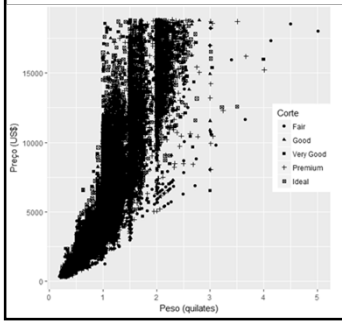
```
> # Relação entre volume (x*y*z) e peso do diamante
> ggplot(data = diamonds, aes( x = carat, y = x * y * z)) +
+ geom_point() +
+ xlab("Peso (quilates)") +
+ ylab("Volume") +
+ ggtitle("Conjunto de Dados diamonds")
```



√ Espera-se que densidade seja constante
- Relação linear entre volume e peso
- Maioria dos pontos parece situar-se em uma linha
- Há *outliers* grandes

- *Scatter plot* – `price`, `carat` e `cut`

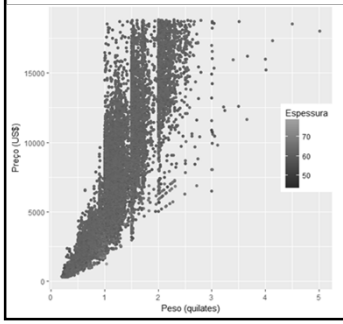
```
> # Peso vs. Preço, com caracter diferentes em cut
> ggplot(data = diamonds, aes(x = carat, y = price, shape = cut)) +
+ geom_point() +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_shape_discrete(name = "Corte") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5)
```



√ Difícil a leitura devido ao overplotting

- *Scatter plot* – `price`, `carat` e `depth`

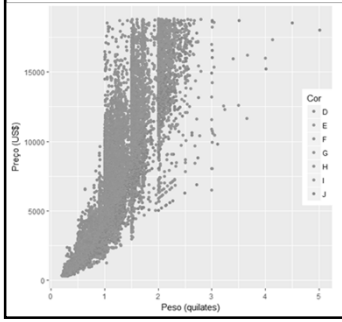
```
> # Preço vs. peso com codificação de cor para depth
> ggplot(data = diamonds, aes(x = carat, y = price, colour = depth)) +
+ geom_point() +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_color_continuous(name = "Espessura") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5))
```



√ Difícil a leitura devido ao overplotting

- *Scatter plot* – `price`, `carat` e `color`

```
> # Peso vs. Preço, com cores em color
> ggplot(data = diamonds, aes(x = carat, y = price, color = color)) +
+ geom_point() +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_color_discrete(name = "Cor") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5))
```



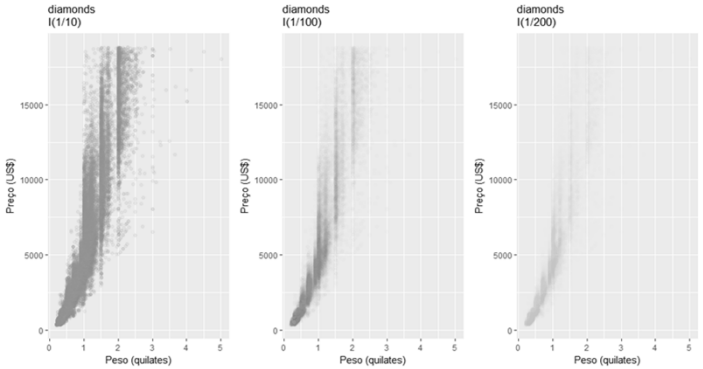
√ Aparentemente há diferentes relações entre `price` e `carat`, de acordo com os níveis de `color`
  – Leitura ainda prejudicada pelo overplotting

- *Scatter plot* – `price`, `carat` e `color`
  √ Redução do overplotting

```
> # Peso vs. Preço, com cores em color - Redução overplotting
> dp1 <- ggplot(data = diamonds, aes(x = carat, y = price, color = color)) +
+ geom_point(alpha = I(1/10)) +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("diamonds\nI(1/10)") +
+ guides(colour=FALSE)
> dp2 <- ggplot(data = diamonds, aes(x = carat, y = price, color = color)) +
+ geom_point(alpha = I(1/100)) +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("diamonds\nI(1/100)") +
+ guides(colour=FALSE)
> dp3 <- ggplot(data = diamonds, aes(x = carat, y = price, color = color)) +
+ geom_point(alpha = I(1/200)) +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("diamonds\nI(1/200)") +
+ guides(colour=FALSE)
> library(gridExtra)
> grid.arrange(dp1, dp2, dp3, nrow=1
```

denominador especifica a quantidade de pontos que devem se sobrepor para se obter uma cor completamente opaca

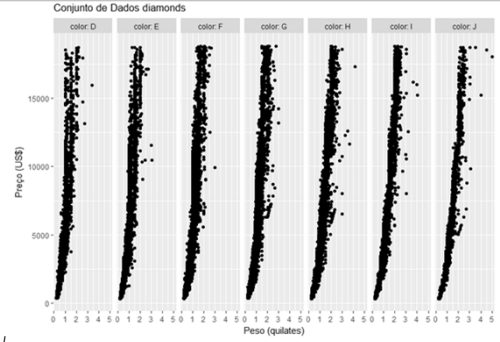
- *Scatter plot* – Redução overplotting



√ Relações diferentes para os níveis de color.

- *Scatter plot* – `price`, `carat` e `color`

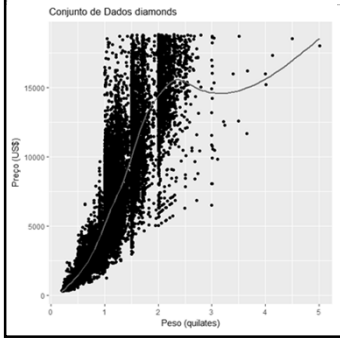
```
> # Preço vs. Peso para cada nível de color
> p + geom_point() + facet_grid(. ~ color, labeller = label_both)
```



√ Diferentes tendências de crescimento por cor

- Relação entre `price` e `carat`

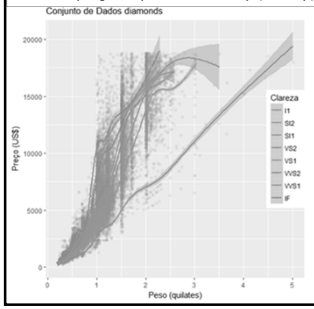
```
> # Exploração relação entre peso e preço
> ggplot(data = diamonds, aes(x = carat, y = price)) +
+ geom_point() +
+ geom_smooth(se = FALSE) +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("Conjunto de Dados diamonds")
```



√ Suavização
- Conjunto de dados pequenos `method = loess`
- Conjunto de dados grandes: `method = gam`

- Relação `price`, `carat` e `clarity`

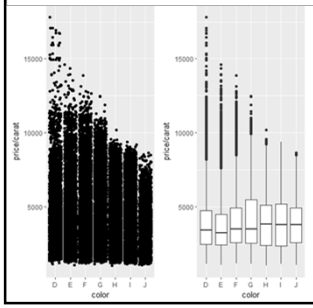
```
> # Preço vs. peso, com suavização para clarity
> ggplot(data = diamonds, aes(x = carat, y = price, colour = clarity)) +
+ geom_point(alpha = 0.1) +
+ geom_smooth() +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_color_discrete(name = "Clareza") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5))
```



√ Relação entre peso e preço aparentam ser diferentes com relação aos níveis de clareza

- Comparação preço unitário por cor

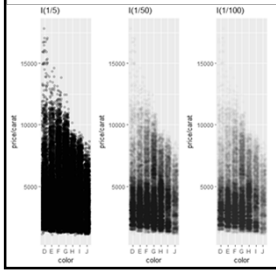
```
> # Comparação preço unitário para níveis de color
>   # gráfico de pontos
> bp1 <- ggplot(data = diamonds, aes(x = color, y = price/carat)) +
+ geom_point(position = position_jitter(width = 0.4))
>   # box-plot
> bp2 <- ggplot(data = diamonds, aes(x = color, y = price/carat)) +
+ geom_boxplot()
>   # painel
> grid.arrange(bp1, bp2, nrow=1)
```



√ À medida em que a cor melhora (da esquerda para a direita):
- Diminui a dispersão dos valores
- Há pouca alteração no centro da distribuição

- Comparação preço unitário por cor

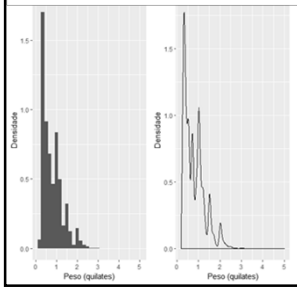
```
> # Comparação preço unitário para níveis de color - redução overplotting
> bp1 <- ggplot(data = diamonds, aes(x = color, y = price/carat)) +
+ geom_point(position=position_jitter(width=0.4), alpha = I(1/5))+
+ ggtitle("I(1/5)")
> bp2 <- ggplot(data = diamonds, aes(x = color, y = price/carat)) +
+ geom_point(position=position_jitter(width=0.4), alpha = I(1/50))+
+ ggtitle("I(1/50)")
> bp3 <- ggplot(data = diamonds, aes(x = color, y = price/carat)) +
+ geom_point(position=position_jitter(width=0.4), alpha = I(1/100))+
+ ggtitle("I(1/100)")
> grid.arrange(bp1, bp2, bp3, nrow=1)
```



√ À medida em que a opacidade diminui começamos visualiza-se onde se situa a maior parte dos dados
- Boxplot efetua melhor esta tarefa

- Distribuição do peso

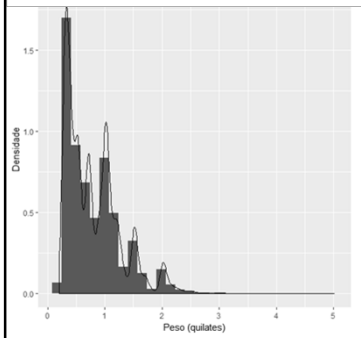
```
> # Distribuição do Peso
> hs1 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..)) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")
> hs2 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_density() +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")
> grid.arrange(hs1, hs2, nrow=1)
```



√ Aparentemente à grupos de preços
- Diferentes distribuições de preços

- Distribuição do peso – suavização

```
> hs3 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..)) +
+ geom_density() +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")
> hs3
```

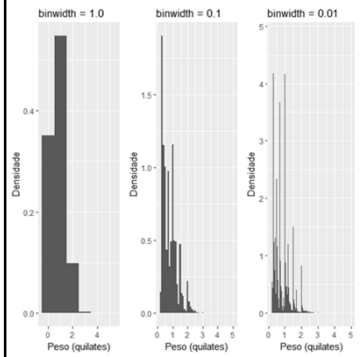


√ Aparência de multimodalidade dos dados
√ Importante tentar vários graus de suavização
- No histograma: `binwidth` controla a quantidade de suavização

- Distribuição do peso
  - √ Pesquisa da quantidade de suavização

```
> # Distribuição do peso - pesquisa da qte de suavização
>
> hs1 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..), binwidth = 1.0) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")+
+ ggtitle("binwidth = 1.0")
> hs2 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..), binwidth = 0.1) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade") +
+ ggtitle("binwidth = 0.1")
> hs3 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..), binwidth = 0.01) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade") +
+ ggtitle("binwidth = 0.01")
> grid.arrange(hs1, hs2, hs3, nrow=1)
```

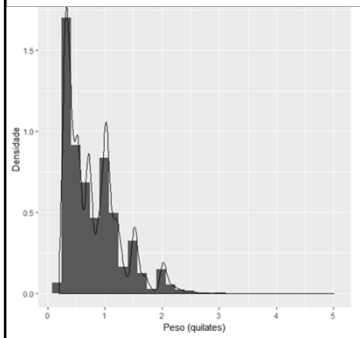

- Estimativa de densidade do preço



√ Gráfico com menor intervalo de classe apresenta as estrias percebidas
- No scatterplot a maioria ocorre em númeors “bonitos”

• Distribuição do peso – suavização

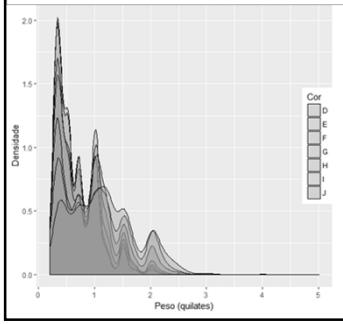
```
> hs3 <- ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..)) +
+ geom_density() +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")
> hs3
```



√ Aparência de multimodalidade dos dados
√ Importante tentar vários graus de suavização
- No histograma: `binwidth` controla a quantidade de suavização

• Densidades do peso por nível de cor

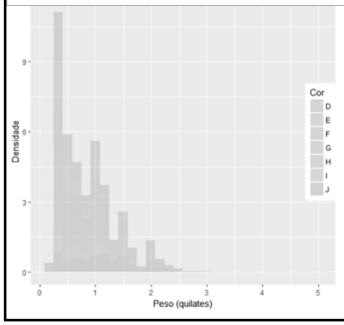
```
> # Comparação distribuição de peso entre níveis de cor
> ggplot(data = diamonds, aes(x = carat, fill = color)) +
+  geom_density(alpha = 0.3) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_fill_discrete(name = "Cor") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5))
```



√ As densidades parecem fáceis de ser lida na comparação as diferentes curvas.
√ Importante tentar vários graus de suavização
√ Supõem hipóteses que podem não ser verdade para os dados:
- Densidade contínua, suave e ilimitada

• Histogramas do peso por nível de cor

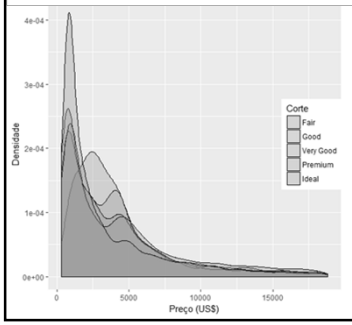
```
> # Comparação histogramas de peso entre níveis de cor
> ggplot(data = diamonds, aes(x = carat, fill = color)) +
+  geom_histogram(aes(y = ..density..), alpha = 0.3) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_fill_discrete(name = "Cor") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5))
```



√ Interpretação do gráfico é mais difícil

• Densidades de preço por nível de corte

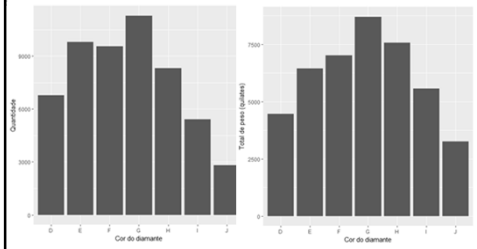
```
> # Comparação distribuição de preço entre níveis de corte
>  ggplot(data = diamonds, aes(x = price, fill = cut)) +
+  geom_density(alpha = 0.3) +
+ xlab("Preço (US$)") +
+ ylab("Densidade") +
+ ggtitle("Conjunto de Dados diamonds") +
+ scale_fill_discrete(name = "Corte") +
+ theme(legend.position = c(1, 0.5), legend.justification = c(1,0.5)
```



√ Distribuições diferentes por nível de corte
- Forte assimetria

- Distribuição da variável `color`:

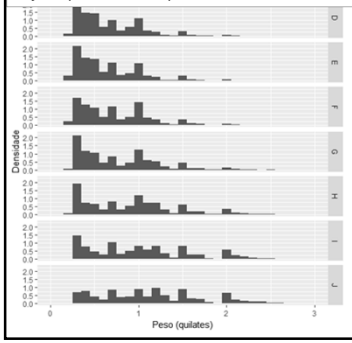
```
> # Distribuição variável color
> ggplot(data = diamonds, aes(x = color)) +
+ geom_bar() +
+ xlab("Cor do diamante") +
+ ylab("Quantidade")
> ggplot(data = diamonds, aes(x = color)) +
+ geom_bar(aes(weight = carat)) +
+ xlab("Cor do diamante") +
+ ylab("Total de peso (quilates)")
```



√ Cor dos diamantes ponderado por carat
- Peso total dos diamantes por nível de cor

- Histogramas de peso por cor:

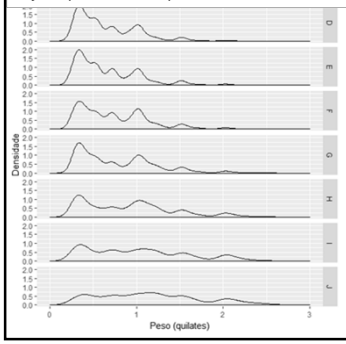
```
> # Histogramas de Peso condicionado a cor
> ggplot(data = diamonds, aes(x = carat)) +
+ geom_histogram(aes(y = ..density..), binwidth = 0.1) +
+ xlim(0, 3) +
+ facet_grid(color ~ .) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")
```



√ Assimetria em direção aos menores valores para diamantes de alta qualidade (cor D)
- Distribuição torna-se mais plana à medida em que a qualidade diminui

- Histogramas de peso por cor:

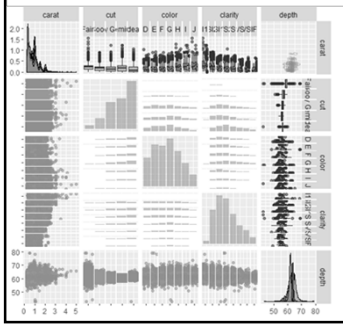
```
> # Density plot de Peso condicionado a cor
> ggplot(data = diamonds, aes(x = carat)) +
+ geom_density() +
+ xlim(0, 3) +
+ facet_grid(color ~ .) +
+ xlab("Peso (quilates)") +
+ ylab("Densidade")
```



√ Distribuições dos diamantes são mais fáceis de serem comparadas
- Ignora a quantidade de diamantes em cada nível de cor

- *Scatter plot matrix*:
  - √ `Carat`, `cut`, `color`, `clarity`, `depth`

```
> library(GGally)
> ggpairs(diamonds[,1:5], upper = list(continuous = "density", combo = "box"),
+      lower = list(continuous = "points", combo = "dot"),
+     mapping = ggplot2::aes(colour = cut, alpha = 0.4),
+    title = "Conjunto de Dados - diamonds"
+  )
```



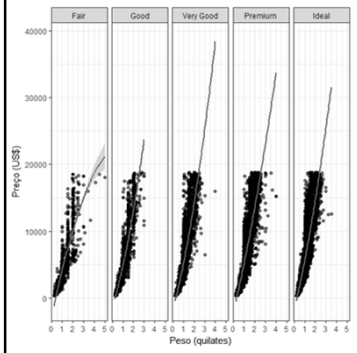
√ Visualização de pares de variáveis
- Métricas ou não métricas

- Relação entre preço e peso:
  - √ Gráficos com suavização por corte

```
> library(dplyr)
>
> diamonds %>%
+ ggplot(aes(x = carat,y = price)) +
+ geom_point(alpha = 0.5) +
+ facet_grid(~ cut) +
+ stat_smooth(method = lm, formula = y ~ poly(x,2)) +
+ xlab("Peso (quilates)") +
+ ylab("Preço (US$)") +
+ theme_bw()
```

  - √ Operador %>% passa a saída do operador da esquerda como o primeiro argumento para o operador da direita

- Relação entre preço e peso:
  - √ Gráficos com suavização por corte



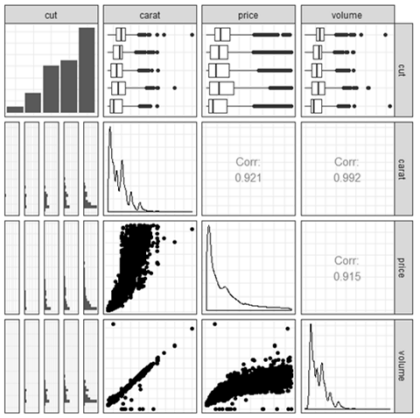
- √ Preço aumenta com tamanho do diamante
- √ Relacionamento é não-linear
- √ Há alguns outliers
- √ Relacionamento com corte não é forte

- *Scatter plot matrix*:

```
diamonds %>%
        mutate(volume = x*y*z) %>%
        select(cut, carat, price, volume) %>%
        sample_frac(0.5, replace = TRUE) %>%
        ggpairs(axisLabels = "none") +
        theme_bw()
```

  - √ Gráfico informativo:
    - Podemos aprender muito sobre a estrutura de covariância dos dados
    - Scatterplots (contínua vs. contínua) ou histogramas por grupos (contínua vs. categórica)
    - Diagonal: estimativas densidades (dados contínuos), histogramas (categóricos)
    - upper: correlação (dados contínuos) ou boxplots por grupos (contínuos vs. categóricos)

- *Scatter plot matrix*:



- √ Facilita visualização dos dados

Referências

Bibliografia Recomendada

- DALGAARD, P. *Introductory statistics with R.* Springer, 2002.
- MURRELL, P. *R graphics*. Chapman & Hall, 2006.
- EVERITT, B.; HOTHORN, T. *An Introduction to Applied Multivariate Analysis with R*. Springer, 2011.
- ZELTERMAN, D. *Applied Multivariate Statistics with R*. Springer, 2015.