



CEFET/RJ

# Data Visualization



Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br

<https://eic.cefet-rj.br/~eogasawara>

# Plotting charts

- R comes with basic plot functions
  - They are easy to operate for simple charts, but they are limited
- ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics\*.
- R for Data Science\*\* is designed to give you a comprehensive introduction to ggplot2
  - The Data Visualization and Graphics for communication chapters

```
# The easiest way to get ggplot2 is to install the whole tidyverse
#install.packages("tidyverse")
#install.packages("ggplot2")
```

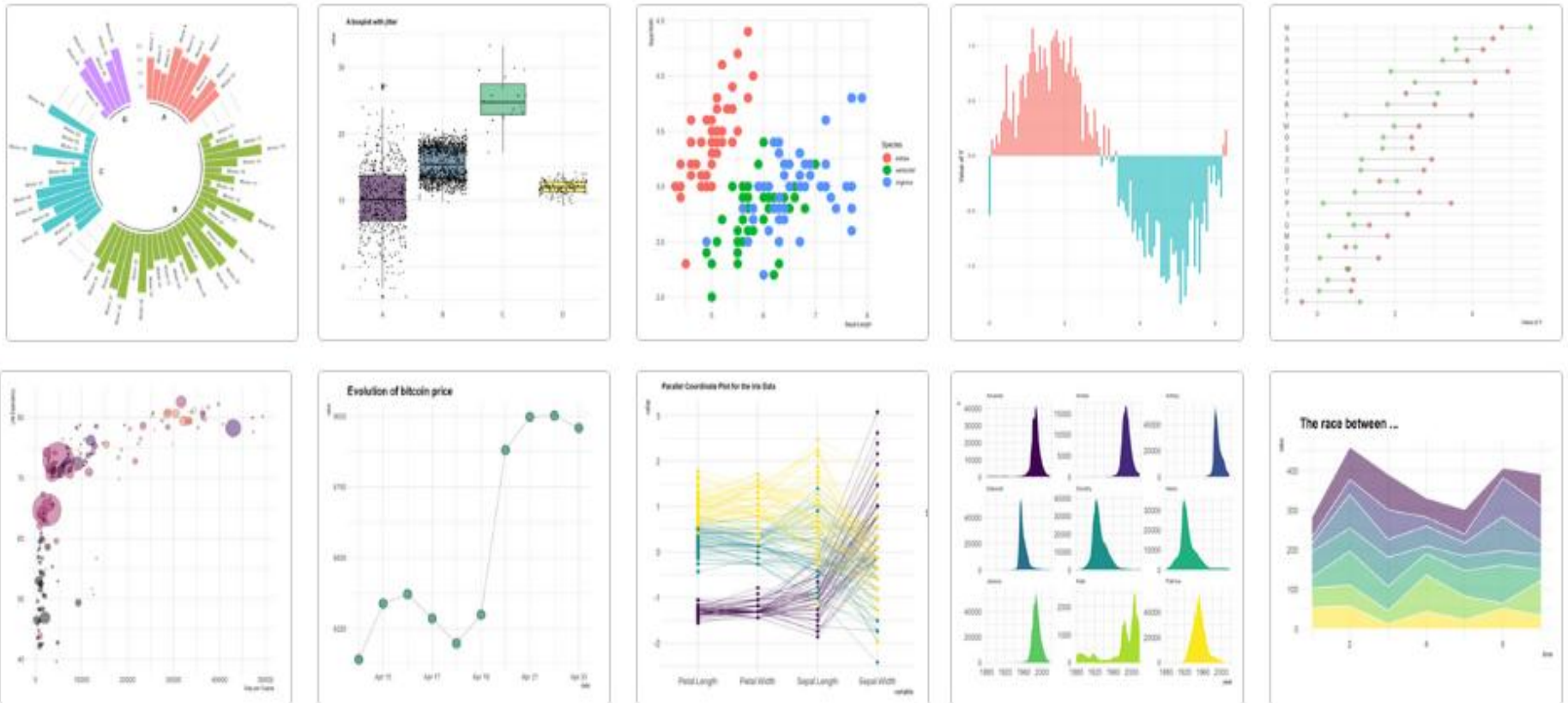


(\*) <https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448>

(\*\*) <https://r4ds.had.co.nz/>

# Examples of ggplot2 charts

- This gallery shows the wide variety of visualizations you can create using ggplot2
- It's a great resource for inspiration and learning how to customize plots



# ggplot2 cheat sheet

- This cheat sheet is a quick reference for syntax and structure in ggplot2
- It's especially helpful when you're starting out or revisiting unfamiliar features

## Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```

ggplot (data = <DATA>) +
  <GEOM_FUNCTION> (mapping = aes (<MAPPINGS>),
    stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
  
```

**required**

**Not required, sensible defaults supplied**

**ggplot(data = mpg, aes(x = cty, y = hwy))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

**last\_plot()** Returns the last plot.

**ggsave("plot.png", width = 5, height = 5)** Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

## ONE VARIABLE continuous

```

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight
  
```

## TWO VARIABLES both continuous

```

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1,
  nudge_y = 1) x, y, label, alpha, angle, color,
  family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = "lm")
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1,
  nudge_y = 1) x, y, label, alpha, angle, color,
  family, fontface, hjust, lineheight, size, vjust
  
```

## one discrete, one continuous

```

f <- ggplot(mpg, aes(class, hwy))

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, ymin, alpha,
  color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill, group

f + geom_violin(scale = "area")
x, y, alpha, color, fill, group, linetype, size, weight
  
```

## ***ggplot easy encapsulation through daltoolbox***

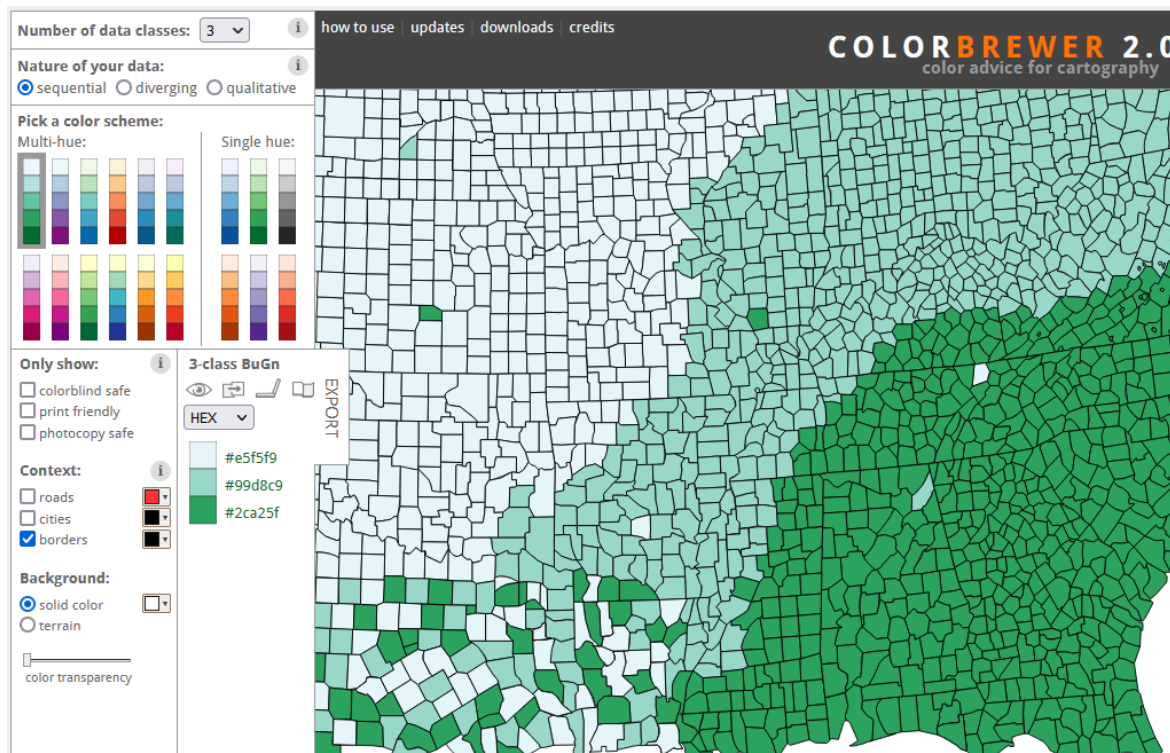
- The DAL Toolbox enables plotting charts encapsulating ggplot2
- It enables an easy startup while learning how to use ggplot2
- Most functions require a data.frame with two basic parameters
  - The first parameter is usually associated with the x-axis: x
  - The second parameter is related to the y-axis: value
  - Sometimes, the third parameter might be a group variable definition
- The DAL Toolbox is loaded using library function

```
library(daltoolbox)  
library(ggplot2)
```



# Using Color Effectively in Charts

- Clear color choices help convey meaning and maintain visual identity
- Use ColorBrewer to select palettes:
  - Sequential: for ordered data
  - Diverging: for deviations from a midpoint
  - Qualitative: for categorical data



# Example of pallets using Color Brewer R Package

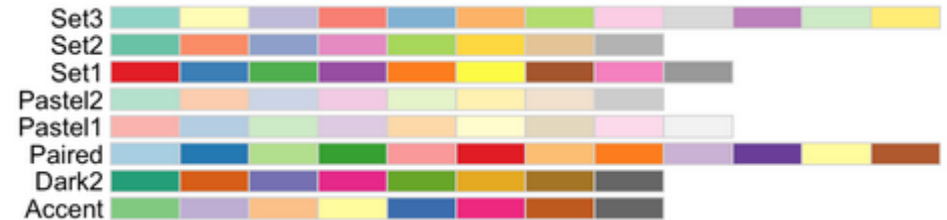
```
#install.packages("RColorBrewer")  
library(RColorBrewer)  
colors <- brewer.pal(4, 'Set1')
```



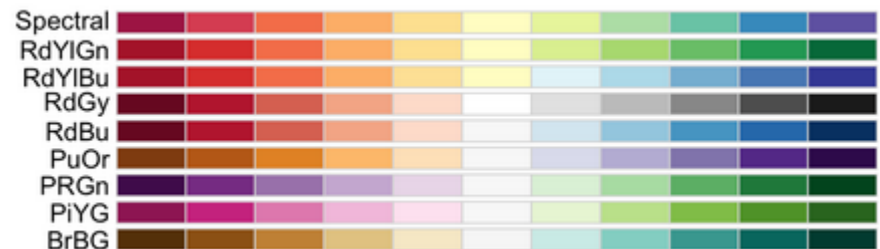
Sequential



Qualitative



Diverging





# Understanding the DAL Toolbox Input Format

- DAL Toolbox simplifies ggplot2 usage for beginners
- Most functions use a data.frame with:
  - x: the independent variable (horizontal axis)
  - value: the dependent variable (vertical axis)
  - group: optional, for grouping bars or lines
- Loaded via: library(daltoolbox)
- Encourages clean and quick visualizations

Iris datasets

```
head(iris, 3)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1          3.5          1.4          0.2
## 2          4.9          3.0          1.4          0.2
## 3          4.7          3.2          1.3          0.2
## Species
## 1 setosa
## 2 setosa
## 3 setosa
```

Options from graphics: colors and font size

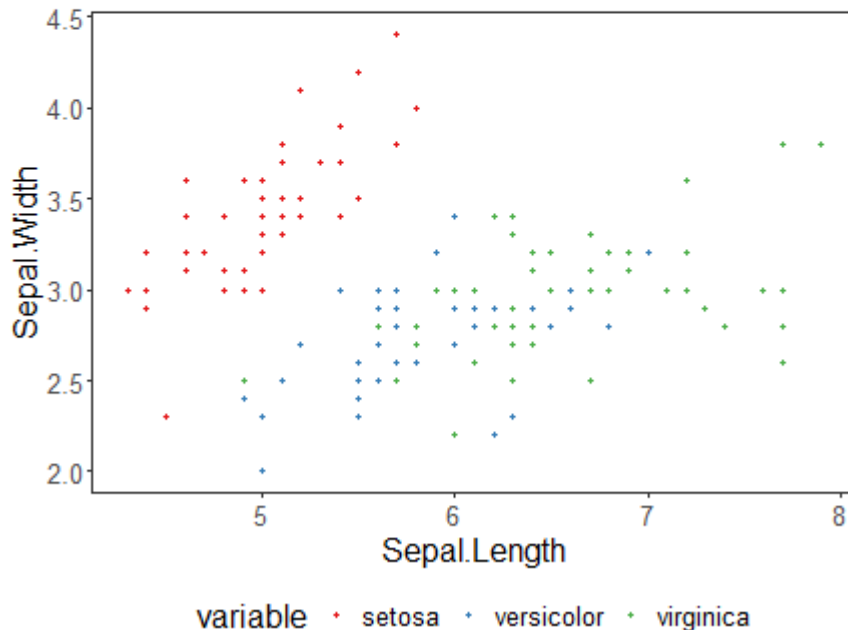
```
colors <- brewer.pal(4, 'Set1')
font <- theme(text = element_text(size=16))
```



## Scatter plot

- Scatter plots are used to show the relationship between two numeric variables, which can help identify trends, clusters, or outliers

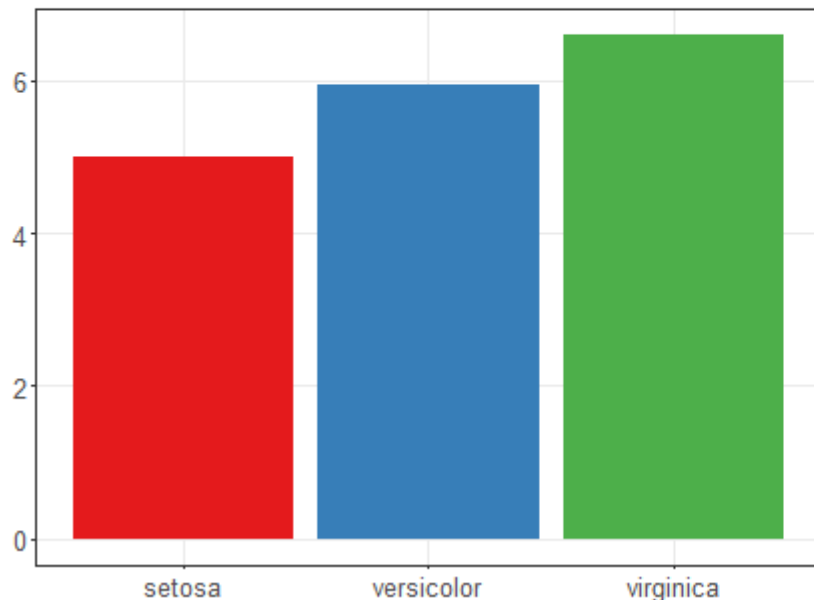
```
library(dplyr)
data <- iris |> select(x = Sepal.Length, value = Sepal.Width, variable = Species)
#head(data)
grf <- plot_scatter(data, label_x = "Sepal.Length", label_y = "Sepal.Width", colors=colors[1:3]) + font
plot(grf)
```



## Bar graph

- Bar graphs are ideal for comparing quantities across categories
- They are simple, effective, and widely used in categorical analysis

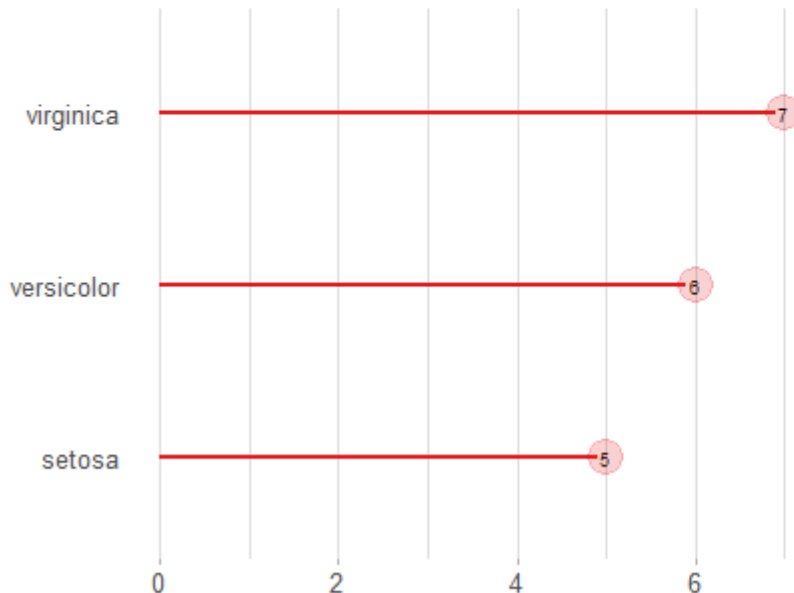
```
library(dplyr)
data <- iris |> group_by(Species) |> summarize(Sepal.Length=mean(Sepal.Length))
#head(data)
grf <- plot_bar(data, colors=colors[1:3]) + font
plot(grf)
```



## Lollipop plot

- A lollipop plot presents the same information as a bar chart, but with a cleaner, lighter visual style — especially effective when there are many categories

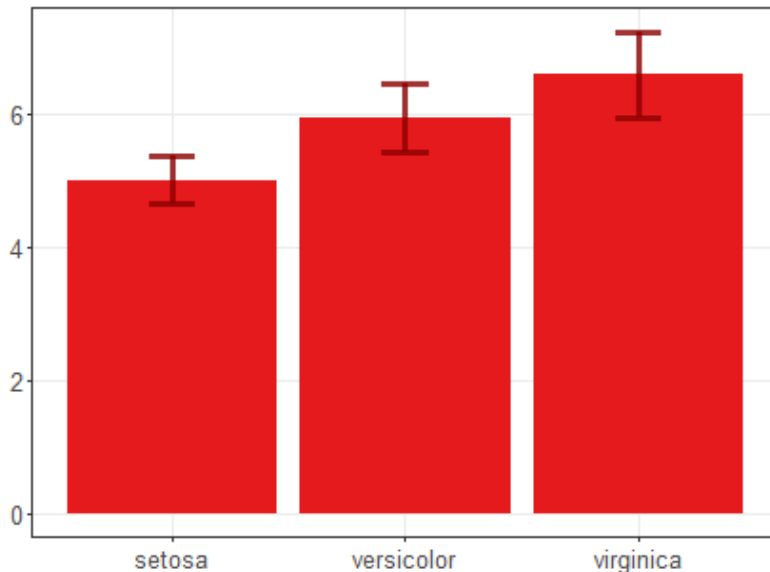
```
library(dplyr)
data <- iris |> group_by(Species) |> summarize(Sepal.Length=mean(Sepal.Length))
#head(data)
grf <- plot_lollipop(data, colors=colors[1], max_value_gap=0.2) + font + coord_flip()
plot(grf)
```



## Bar graph with error bars

- Used to show averages with associated variability, such as confidence intervals or standard deviations

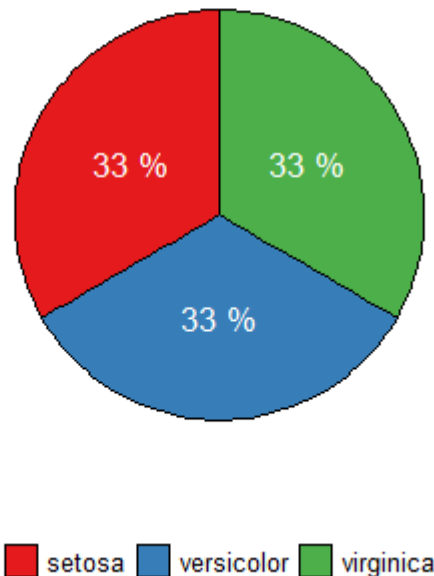
```
library(dplyr)
data <- iris |> group_by(Species) |> summarize(mean=mean(Sepal.Length), sd=sd(Sepal.Length))
#head(data)
grf <- plot_bar(data, colors=colors[1], alpha=1) + font
grf <- grf + geom_errorbar(aes(x=Species, ymin=mean-sd, ymax=mean+sd),
                           width=0.2, colour="darkred", alpha=0.8, size=1.1)
plot(grf)
```



## Pie chart

- Circular statistical graphic to illustrate numerical proportion

```
library(dplyr)
data <- iris |> group_by(Species) |> summarize(n = n())
#head(data)
grf <- plot_pieplot(data, colors=colors[1:3]) + font
plot(grf)
```

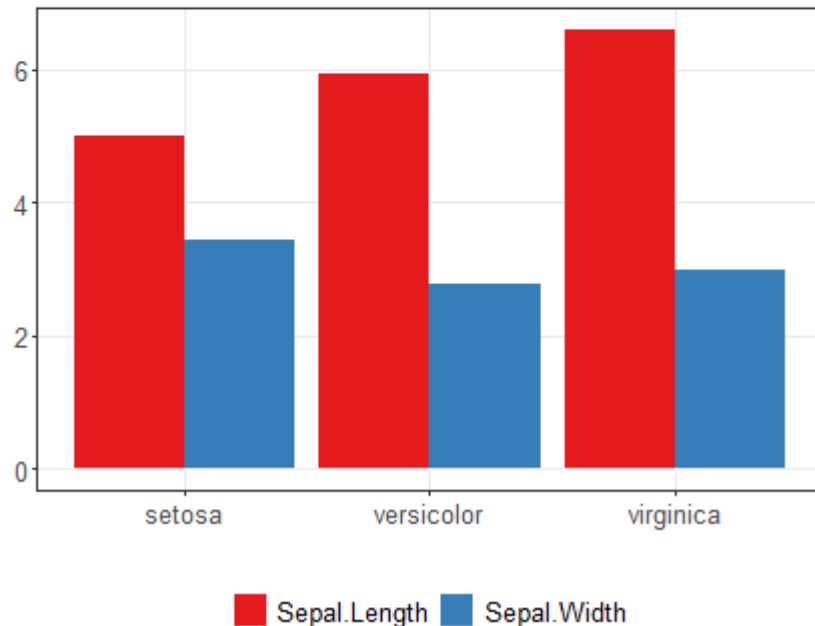


## Grouped bar

- Grouped bar charts let you compare sub-categories within main categories. Each group shows side-by-side bars, one per sub-category

```
library(dplyr)
data <- iris |> group_by(Species) |> summarize(Sepal.Length=mean(Sepal.Length), Sepal.Width=mean(Sepal.Width))

#head(data)
grf <- plot_groupedbar(data, colors=colors[1:2]) + font
plot(grf)
```

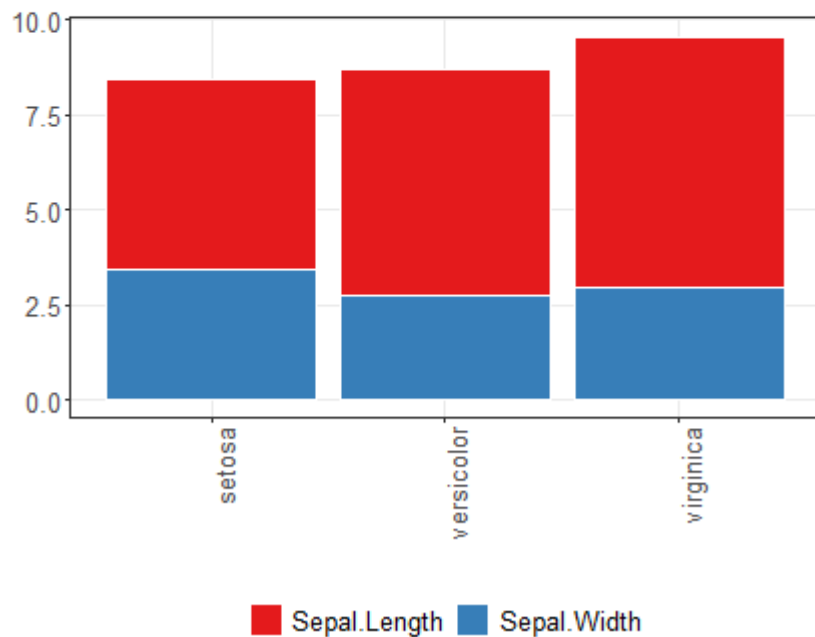


## Stacked-bars

- The height of the bar shows the combined result of the groups

```
library(dplyr)
data <- iris |> group_by(Species) |> summarize(Sepal.Length=mean(Sepal.Length), Sepal.Width=mean(Sepal.Width))

#head(data)
grf <- plot_stackedbar(data, colors=colors[1:2]) + font
grf <- grf + theme(axis.text.x = element_text(angle=90, hjust=1))
plot(grf)
```





# Distribution Visualization Overview

## Examples using data distribution

- The following examples use random variables so that different data distribution can be better viewed

```
example <- data.frame(exponencial = rexp(10000, rate = 1),  
                      uniform = runif(10000, min=2.5, max = 3.5),  
                      normal = rnorm(10000, mean = 5))  
  
head(example)
```

```
##  exponencial  uniform   normal  
## 1  0.65926515 2.577149 3.380406  
## 2  0.96877025 2.937234 5.967816  
## 3  0.05059315 2.758117 3.307216  
## 4  0.12027304 2.565390 4.780961  
## 5  5.31988220 2.641392 4.713785  
## 6  2.17472525 2.591428 3.539582
```

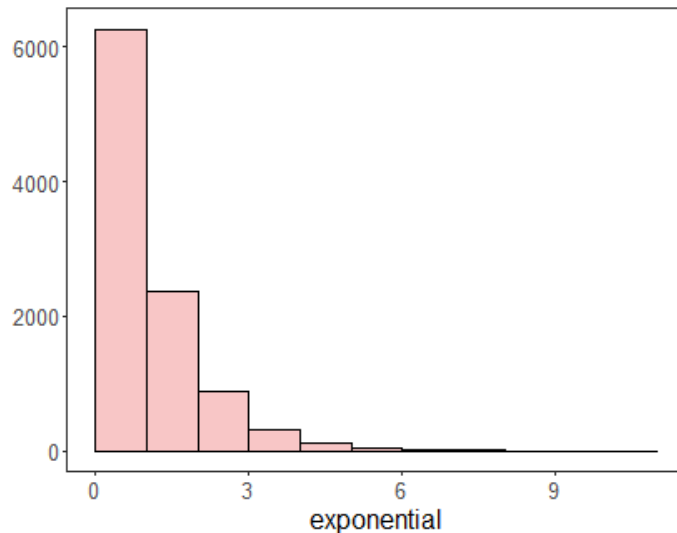
# Histogram

- Counting of a single continuous variable organized into bins

```
library(dplyr)
data <- example |> select(exponential)
#head(data)
grf <- plot_hist(data, label_x = "exponential", color=colors[1]) + font
```

```
## Using as id variables
```

```
plot(grf)
```

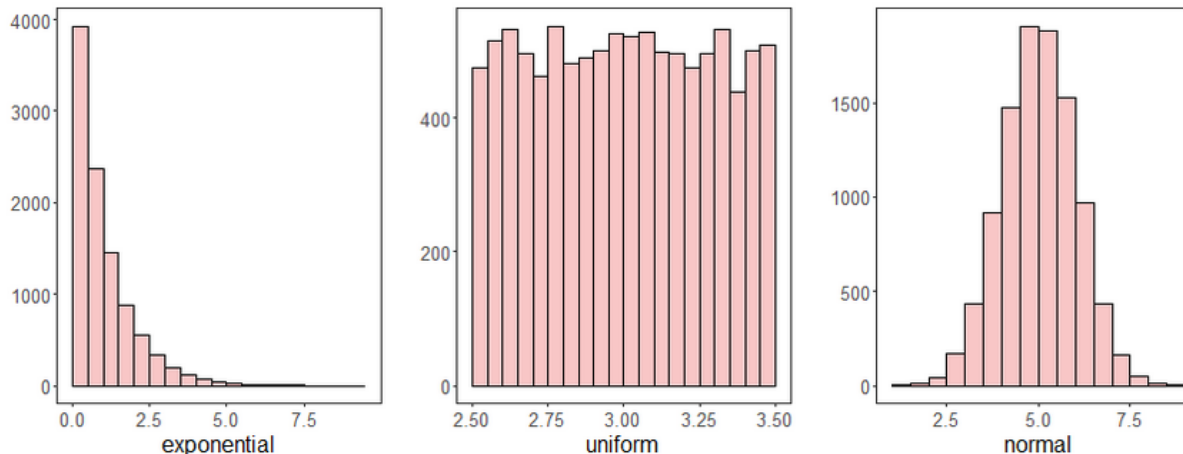


# Multiple histograms

- Multiple histograms allow you to compare distributions across different groups or variables
- Each group can be shown using color or faceting

```
{  
  library(gridExtra)  
  grfe <- plot_hist(example |> select(exponential), label_x = "exponential", color=colors[1]) + font  
  grfu <- plot_hist(example |> select(uniform), label_x = "uniform", color=colors[1]) + font  
  grfn <- plot_hist(example |> select(normal), label_x = "normal", color=colors[1]) + font  
  grid.arrange(grfe, grfu, grfn, ncol=3)  
}
```

```
## Using as id variables  
## Using as id variables  
## Using as id variables
```



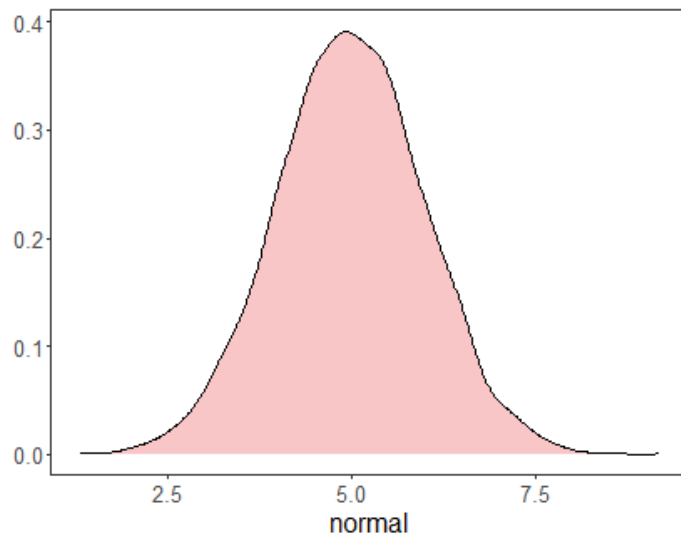
# Density plot

- Draws kernel density estimate

```
library(dplyr)
data <- example |> select(normal)
#head(data)
grf <- plot_density(data, label_x = "normal", color=colors[1]) + font
```

```
## Using as id variables
```

```
plot(grf)
```



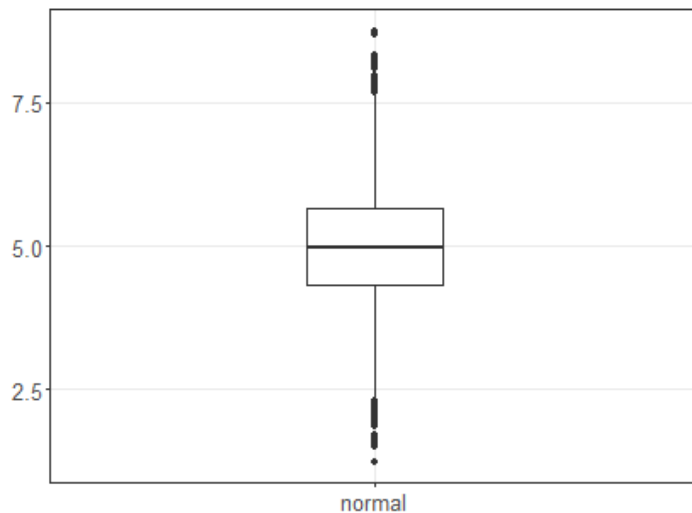
# Box-plot

- Groups numerical data through their quartiles

```
library(dplyr)
data <- example |> select(normal)
#head(data)
grf <- plot_boxplot(data, color="white") + font
```

```
## Using as id variables
```

```
plot(grf)
```



## *Choosing the Right Visualization*








- Selecting the appropriate chart helps communicate your message effectively
- This slide helps match your analysis goal with the most suitable chart type

Goal	Recommended Chart
Compare values across categories	Bar chart, grouped/stacked bars
Show part of a whole	Pie chart, stacked bars
Show distribution	Histogram, boxplot, density plot
Show relationships	Scatter plot, lollipop
Show time trends	Line plot (if added later)
Compare groups with variability	Bar chart with error bars

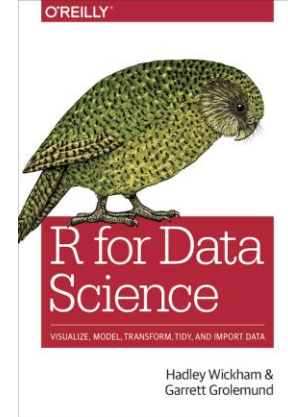
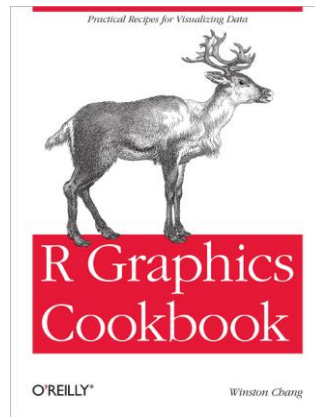
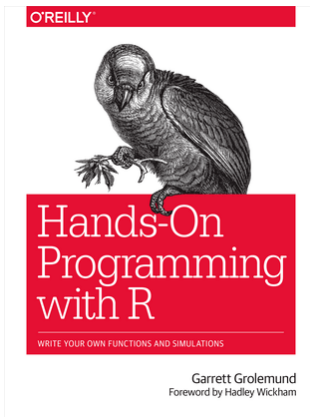


## ***Charting Best Practices***

---

-  Use clear labels on axes and legends
-  Match chart type to your data structure and question
-  Use accessible color palettes (e.g., ColorBrewer)
-  Avoid chartjunk (unnecessary decoration)
-  Use tooltips or annotations to guide interpretation
-  Consistent scales and units
-  The goal of visualization is communication - not decoration.

# Referências



Hands-on Programming with R: <https://rstudio-education.github.io/hopr/index.html>

R Graphics Cookbook: <https://r-graphics.org>

R Packages: <https://r-pkgs.org/index.html>

R for Data Science: <https://r4ds.had.co.nz>

<https://rstudio-education.github.io/hopr/basics.html>