Copyright ©2017 by Alboukadel Kassambara. All rights reserved.

Published by STHDA (http://www.sthda.com), Alboukadel Kassambara

Contact: Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to STHDA (http://www.sthda.com).

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials.

Neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

For general information contact Alboukadel Kassambara <alboukadel.kassambara@gmail.com>.
## Contents

0.1 What you will learn ......................................................... vi
0.2 Book website ...................................................................... vi
0.3 Executing the R codes from the PDF ................................. vi
0.4 Colophon ........................................................................... vii

About the author ................................................................. viii

1 R Basics for Data Visualization ........................................ 1
  1.1 Introduction ...................................................................... 1
  1.2 Install R and RStudio .................................................... 1
  1.3 Install and load required R packages .............................. 2
  1.4 Data format ...................................................................... 2
  1.5 Import your data in R ..................................................... 3
  1.6 Demo data sets .............................................................. 3
  1.7 Data manipulation .......................................................... 4
  1.8 R graphics systems ........................................................ 4
  1.9 Export R graphics ......................................................... 13

2 Plot One Variable .............................................................. 15
  2.1 Introduction ...................................................................... 15
  2.2 Prerequisites .................................................................... 15
  2.3 One categorical variable ................................................. 16
  2.4 One continuous variable ............................................... 19
  2.5 Conclusion ....................................................................... 35

3 Plot Grouped Data .............................................................. 37
  3.1 Introduction ...................................................................... 37
  3.2 Prerequisites .................................................................... 37
  3.3 Grouped categorical variables ........................................ 37
  3.4 Grouped continuous variables ....................................... 43
  3.5 Conclusion ....................................................................... 66
  3.6 See also .......................................................................... 67

4 Plot Two Continuous Variables ........................................... 68
  4.1 Introduction ...................................................................... 68
  4.2 Prerequisites .................................................................... 68
  4.3 Basic scatter plots .......................................................... 69
  4.4 Multiple groups .............................................................. 71
  4.5 Add point text labels ...................................................... 75
## 4. Bubble chart ......................................................... 76
## 4. Color by a continuous variable ................................. 77
## 4. Add marginal density plots ...................................... 78
## 4. Continuous bivariate distribution ............................... 80
## 4. Zoom in a scatter plot ............................................ 81
## 4. Add trend lines and equations ................................... 82
## 4. Conclusion ............................................................ 84
## 4. See also ............................................................... 85

## 5 Plot Multivariate Continuous Data ............................... 86
## 5.1 Introduction ....................................................... 86
## 5. Demo data set and R package .................................... 86
## 5. Create a 3d scatter plot ......................................... 87
## 5.4 Create a scatter plot matrix .................................... 87
## 5. Correlation analysis ................................................. 90
## 5. Principal component analysis .................................... 91
## 5. Cluster analysis .................................................... 92
## 5. Conclusion ............................................................ 93

## 6 Visualizing Multivariate Categorical Data ...................... 95
## 6.1 Introduction ........................................................ 95
## 6. Prerequisites ........................................................ 95
## 6.3 Bar plots of contingency tables .................................. 95
## 6.4 Balloon plot ........................................................ 96
## 6.5 Mosaic plot .......................................................... 98
## 6. Correspondence analysis ........................................... 99

## 7 Plot Time Series Data .................................................. 101
## 7.1 Introduction ........................................................ 101
## 7. Basic ggplot of time series ....................................... 101
## 7.3 Plot multiple time series data .................................... 102
## 7.4 Set date axis limits ................................................. 104
## 7. Format date axis labels ............................................. 104
## 7.6 Add trend smoothed line .......................................... 105
## 7.7 ggplot2 extensions for ts objects ................................ 105

## 8 Facets: Multi-Panels GGPlot ......................................... 108
## 8.1 Introduction ........................................................ 108
## 8.2 Prerequisites ........................................................ 108
## 8.3 Split the plot into a matrix of panels .......................... 109
## 8.4 See also ............................................................... 112

## 9 Arrange Multiple GGPlot on One Page ........................... 113
## 9.1 Introduction ........................................................ 113
## 9.2 Prerequisites ........................................................ 113
## 9.3 Arrange on one page ................................................ 113
## 9.4 Annotate the arranged figure ..................................... 114
## 9.5 Change column and row span of a plot ......................... 115
## 9.6 Use shared legend for combined ggplots ....................... 116
9.7 Mix table, text and ggplot2 graphs .......................... 117
9.8 Arrange over multiple pages ........................................ 118
9.9 Export the arranged plots ........................................... 119
9.10 See also ............................................................... 119

10 Customize GGPlot ....................................................... 120
10.1 Prerequisites .......................................................... 120
10.2 Titles and axis labels ................................................ 120
10.3 Axes: Limits, Ticks and Log ...................................... 122
10.4 Legends: Title, Position and Appearance .................... 127
10.5 Themes gallery ....................................................... 130
10.6 Background color and grid lines ................................. 131
10.7 Add background image to ggplot2 graphs ..................... 132
10.8 Colors ................................................................. 133
10.9 Points shape, color and size ...................................... 138
10.10 Line types ............................................................. 139
10.11 Rotate a ggplot .................................................... 141
10.12 Plot annotation ...................................................... 142
Preface

0.1 What you will learn

Data visualization is one of the most important part of data science. Many books and courses present a catalogue of graphics but they don’t teach you which charts to use according to the type of the data.

In this book, we start by presenting the key graphic systems and packages available in R, including R base graphs, lattice and ggplot2 plotting systems.

Next, we provide practical examples to create great graphics for the right data using either the ggplot2 package and extensions or the traditional R graphics.

With this book, you’ll learn:

• How to quickly create beautiful graphics using ggplot2 packages
• How to properly customize and annotate the plots
• Type of graphics for visualizing categorical and continuous variables
• How to add automatically p-values to box plots, bar plots and alternatives
• How to add marginal density plots and correlation coefficients to scatter plots
• Key methods for analyzing and visualizing multivariate data
• R functions and packages for plotting time series data
• How to combine multiple plots on one page to create production-quality figures.

0.2 Book website

The website for this book is located at: http://www.sthda.com/english/. It contains number of resources.

0.3 Executing the R codes from the PDF

For a single line R code, you can just copy the code from the PDF to the R console.
For a multiple-line R codes, an error is generated, sometimes, when you copy and paste directly the R code from the PDF to the R console. If this happens, a solution is to:

- Paste firstly the code in your R code editor or in your text editor
- Copy the code from your text/code editor to the R console

### 0.4 Colophon

This book was built with R 3.3.2 and the following packages:

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>version</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bookdown</td>
<td>0.5</td>
<td>Github:rstudio/bookdown</td>
</tr>
<tr>
<td>2</td>
<td>changepoint</td>
<td>2.2.2</td>
<td>CRAN</td>
</tr>
<tr>
<td>3</td>
<td>cowplot</td>
<td>0.8.0.9000</td>
<td>Github:wilkelab/cowplot</td>
</tr>
<tr>
<td>4</td>
<td>dplyr</td>
<td>0.7.4</td>
<td>cran</td>
</tr>
<tr>
<td>5</td>
<td>factoextra</td>
<td>1.0.5.999</td>
<td>local:kassambara/factoextra</td>
</tr>
<tr>
<td>6</td>
<td>FactoMineR</td>
<td>1.38</td>
<td>CRAN</td>
</tr>
<tr>
<td>7</td>
<td>GGally</td>
<td>1.3.0</td>
<td>CRAN</td>
</tr>
<tr>
<td>8</td>
<td>ggcorrplot</td>
<td>0.1.1.9000</td>
<td>Github:kassambara/ggcorrplot</td>
</tr>
<tr>
<td>9</td>
<td>ggforce</td>
<td>0.1.1</td>
<td>Github:thomasp85/ggforce</td>
</tr>
<tr>
<td>10</td>
<td>ggformula</td>
<td>0.6</td>
<td>CRAN</td>
</tr>
<tr>
<td>11</td>
<td>ggfortify</td>
<td>0.4.1</td>
<td>CRAN</td>
</tr>
<tr>
<td>12</td>
<td>ggpmisc</td>
<td>0.2.15</td>
<td>CRAN</td>
</tr>
<tr>
<td>13</td>
<td>ggpubr</td>
<td>0.1.5.999</td>
<td>Github:kassambara/ggpubr</td>
</tr>
<tr>
<td>14</td>
<td>lattice</td>
<td>0.20-34</td>
<td>CRAN</td>
</tr>
<tr>
<td>15</td>
<td>readr</td>
<td>1.1.1</td>
<td>cran</td>
</tr>
<tr>
<td>16</td>
<td>scatterplot3d</td>
<td>0.3-40</td>
<td>cran</td>
</tr>
<tr>
<td>17</td>
<td>strucchange</td>
<td>1.5-1</td>
<td>CRAN</td>
</tr>
<tr>
<td>18</td>
<td>tidyr</td>
<td>0.7.2</td>
<td>cran</td>
</tr>
</tbody>
</table>
About the author

Alboukadel Kassambara is a PhD in Bioinformatics and Cancer Biology. He works since many years on genomic data analysis and visualization (read more: http://www.alboukadel.com/).

He has work experiences in statistical and computational methods to identify prognostic and predictive biomarker signatures through integrative analysis of large-scale genomic and clinical data sets.

He created a bioinformatics web-tool named GenomicScape (www.genomicscape.com) which is an easy-to-use web tool for gene expression data analysis and visualization.

He developed also a training website on data science, named STHDA (Statistical Tools for High-throughput Data Analysis, www.sthda.com/english), which contains many tutorials on data analysis and visualization using R software and packages.

He is the author of many popular R packages for:

- survival analysis (survminer, http://www.sthda.com/english/rpkgs/survminer/),

Recently, he published three books on data analysis and visualization:

Chapter 1

R Basics for Data Visualization

1.1 Introduction

R is a free and powerful statistical software for analyzing and visualizing data.

In this chapter, you’ll learn:

- the basics of R programming for importing and manipulating your data:
  - filtering and ordering rows,
  - renaming and adding columns,
  - computing summary statistics
- R graphics systems and packages for data visualization:
  - R traditional base plots
  - Lattice plotting system that aims to improve on R base graphics
  - ggplot2 package, a powerful and a flexible R package, for producing elegant graphics piece by piece.
  - ggpubr package, which facilitates the creation of beautiful ggplot2-based graphs for researcher with non-advanced programming backgrounds.
  - ggformula package, an extension of ggplot2, based on formula interfaces (much like the lattice interface)

1.2 Install R and RStudio

RStudio is an integrated development environment for R that makes using R easier. R and RStudio can be installed on Windows, MAC OSX and Linux platforms.

1. R can be downloaded and installed from the Comprehensive R Archive Network (CRAN) webpage (http://cran.r-project.org/)
3. Launch RStudio and start use R inside R studio.
1.3 Install and load required R packages

An R package is a collection of functionalities that extends the capabilities of base R. To use the R code provide in this book, you should install the following R packages:

- **tidyverse** packages, which are a collection of R packages that share the same programming philosophy. These packages include:
  - `readr`: for importing data into R
  - `dplyr`: for data manipulation
  - `ggplot2` and `ggpubr` for data visualization.
- **ggpubr** package, which makes it easy, for beginner, to create publication ready plots.

1. **Install the tidyverse package.** Installing tidyverse will install automatically `readr`, `dplyr`, `ggplot2` and more. Type the following code in the R console:

```r
install.packages("tidyverse")
```

2. **Install the ggpubr package.**
   - We recommend to install the latest developmental version of ggpubr as follow:

```r
if(!require(devtools)) install.packages("devtools")
devtools::install_github("kassambara/ggpubr")
```

   - If the above R code fails, you can install the latest stable version on CRAN:

```r
install.packages("ggpubr")
```

3. **Load required packages.** After installation, you must first load the package for using the functions in the package. The function `library()` is used for this task. An alternative function is `require()`. For example, to load `ggplot2` and `ggpubr` packages, type this:

```r
library("ggplot2")
library("ggpubr")
```

Now, we can use R functions, such as `ggscatter()` [in the ggpubr package] for creating a scatter plot.

If you want to learn more about a given function, say `ggscatter()`, type this in R console: `?ggscatter`.

1.4 Data format

Your data should be in rectangular format, where columns are variables and rows are observations (individuals or samples).

- Column names should be compatible with R naming conventions. Avoid column with blank space and special characters. Good column names: `long_jump` or `long_jump`. Bad column name: `long jump`. 
1.5 Import your data in R

First, save your data into txt or csv file formats and import it as follow (you will be asked to choose the file):

```r
library("readr")

# Reads tab delimited files (.txt tab)
my_data <- read_tsv(file.choose())

# Reads comma (,) delimited files (.csv)
my_data <- read_csv(file.choose())

# Reads semicolon(;) separated files(.csv)
my_data <- read_csv2(file.choose())
```

Read more about how to import data into R at this link: [http://www.sthda.com/english/wiki/importing-data-into-r](http://www.sthda.com/english/wiki/importing-data-into-r)

1.6 Demo data sets

R comes with several demo data sets for playing with R functions. The most used R demo data sets include: USArrests, iris and mtcars. To load a demo data set, use the function `data()` as follow. The function `head()` is used to inspect the data.

```r
data("iris")  # Loading
head(iris, n = 3)  # Print the first n = 3 rows

## 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
```

---

To learn more about iris data sets, type this:

```r
?iris
```

After typing the above R code, you will see the description of `iris` data set: this iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

### 1.7 Data manipulation

After importing your data in R, you can easily manipulate it using the `dplyr` package (Wickham et al., 2017), which can be installed using the R code: `install.packages("dplyr")`.

After loading `dplyr`, you can use the following R functions:

- `filter()`: Pick rows (observations/samples) based on their values.
- `distinct()`: Remove duplicate rows.
- `arrange()`: Reorder the rows.
- `select()`: Select columns (variables) by their names.
- `rename()`: Rename columns.
- `mutate()`: Add/create new variables.
- `summarise()`: Compute statistical summaries (e.g., computing the mean or the sum)
- `group_by()`: Operate on subsets of the data set.

Note that, `dplyr` package allows to use the forward-pipe chaining operator (`%>%`) for combining multiple operations. For example, `x %>% f` is equivalent to `f(x)`. Using the pipe (`%>%`), the output of each operation is passed to the next operation. This makes R programming easy.

We’ll show you how these functions work in the different chapters of this book.

### 1.8 R graphics systems

There are different graphic packages available in R\(^2\) for visualizing your data: 1) R base graphs, 2) Lattice Graphs (Sarkar, 2016) and 3) `ggplot2` (Wickham and Chang, 2017).

In this section, we start by providing a quick overview of R base and lattice plots, and then we move to `ggplot2` graphic system. The vast majority of plots generated in this book is based on the modern and flexible `ggplot2` R package.

\(^2\)http://www.sthda.com/english/wiki/data-visualization
1.8. R GRAPHICS SYSTEMS

1.8.1 R base graphs

R comes with simple functions to create many types of graphs. For example:

<table>
<thead>
<tr>
<th>Plot Types</th>
<th>R base function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scatter plot</td>
<td>plot()</td>
</tr>
<tr>
<td>Scatter plot matrix</td>
<td>pairs()</td>
</tr>
<tr>
<td>Box plot</td>
<td>boxplot()</td>
</tr>
<tr>
<td>Strip chart</td>
<td>stripchart()</td>
</tr>
<tr>
<td>Histogram plot</td>
<td>hist()</td>
</tr>
<tr>
<td>density plot</td>
<td>density()</td>
</tr>
<tr>
<td>Bar plot</td>
<td>barplot()</td>
</tr>
<tr>
<td>Line plot</td>
<td>plot() and line()</td>
</tr>
<tr>
<td>Pie charts</td>
<td>pie()</td>
</tr>
<tr>
<td>Dot charts</td>
<td>dotchart()</td>
</tr>
<tr>
<td>Add text to a plot</td>
<td>text()</td>
</tr>
</tbody>
</table>

In the most cases, you can use the following arguments to customize the plot:

- **pch**: change point shapes. Allowed values comprise number from 1 to 25.
- **cex**: change point size. Example: `cex = 0.8`.
- **col**: change point color. Example: `col = "blue"`.
- **frame**: logical value. `frame = FALSE` removes the plot panel border frame.
- **main, xlab, ylab**: Specify the main title and the x/y axis labels, respectively.
- **las**: For a vertical x axis text, use `las = 2`.

In the following R code, we’ll use the iris data set to create a:

- (1) Scatter plot of Sepal.Length (on x-axis) and Sepal.Width (on y-axis).
- (2) Box plot of Sepal.length (y-axis) by Species (x-axis)

```r
# (1) Create a scatter plot
plot(
  x = iris$Sepal.Length, y = iris$Sepal.Width,
  pch = 19, cex = 0.8, frame = FALSE,
  xlab = "Sepal Length", ylab = "Sepal Width"
)

# (2) Create a box plot
boxplot(Sepal.Length ~ Species, data = iris,
  ylab = "Sepal.Length",
  frame = FALSE, col = "lightgray")
```
1.8.2 Lattice graphics

The **lattice** R package provides a plotting system that aims to improve on R base graphs. After installing the package, with the R command `install.packages("lattice")`, you can test the following functions.

- Main functions in the lattice package:

<table>
<thead>
<tr>
<th>Plot types</th>
<th>Lattice functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scatter plot</td>
<td><code>xyplot()</code></td>
</tr>
<tr>
<td>Scatter plot matrix</td>
<td><code>splom()</code></td>
</tr>
<tr>
<td>3D scatter plot</td>
<td><code>cloud()</code></td>
</tr>
<tr>
<td>Box plot</td>
<td><code>bwplot()</code></td>
</tr>
<tr>
<td>strip plots (1-D scatter plots)</td>
<td><code>stripplot()</code></td>
</tr>
<tr>
<td>Dot plot</td>
<td><code>dotplot()</code></td>
</tr>
<tr>
<td>Bar chart</td>
<td><code>barchart()</code></td>
</tr>
<tr>
<td>Histogram</td>
<td><code>histogram()</code></td>
</tr>
<tr>
<td>Density plot</td>
<td><code>densityplot()</code></td>
</tr>
<tr>
<td>Theoretical quantile plot</td>
<td><code>qqmath()</code></td>
</tr>
<tr>
<td>Two-sample quantile plot</td>
<td><code>qq()</code></td>
</tr>
<tr>
<td>3D contour plot of surfaces</td>
<td><code>contourplot()</code></td>
</tr>
<tr>
<td>False color level plot of surfaces</td>
<td><code>levelplot()</code></td>
</tr>
<tr>
<td>Parallel coordinates plot</td>
<td><code>parallel()</code></td>
</tr>
<tr>
<td>3D wireframe graph</td>
<td><code>wireframe()</code></td>
</tr>
</tbody>
</table>

The `lattice` package uses formula interface. For example, in lattice terminology, the formula `y ~ x | group`, means that we want to plot the `y` variable according to the `x` variable, splitting the plot into multiple panels by the variable `group`.

• Create a basic scatter plot of y by x. Syntax: y ~ x. Change the color by groups and use auto.key = TRUE to show legends:

```r
library("lattice")
xyplot(
  Sepal.Length ~ Petal.Length, group = Species,
  data = iris, auto.key = TRUE, pch = 19, cex = 0.5
)
```

```
setosa ○
versicolor ○
virginica ○
```

• Multiple panel plots by groups. Syntax: y ~ x | group.

```r
xyplot(
  Sepal.Length ~ Petal.Length | Species,
  layout = c(3, 1),
  group = Species, data = iris,
  type = c("p", "smooth"),
  scales = "free"
)
```

Read more examples at: Lattice Graphics on STHDA®

®http://www.sthda.com/english/wiki/lattice-graphs
1.8.3 ggplot2 graphics

GGPlot2 is a powerful and a flexible R package, implemented by Hadley Wickham, for producing elegant graphics piece by piece. The gg in ggplot2 means Grammar of Graphics, a graphic concept which describes plots by using a “grammar”. According to the ggplot2 concept, a plot can be divided into different fundamental parts: **Plot = data + Aesthetics + Geometry**

- **data**: a data frame
- **aesthetics**: used to indicate the x and y variables. It can be also used to control the color, the size and the shape of points, etc....
- **geometry**: corresponds to the type of graphics (histogram, box plot, line plot, ...)

The ggplot2 syntax might seem opaque for beginners, but once you understand the basics, you can create and customize any kind of plots you want.

Note that, to reduce this opacity, we recently created an R package, named ggpubr (ggplot2 Based Publication Ready Plots), for making ggplot simpler for students and researchers with non-advanced programming backgrounds. We’ll present ggpubr in the next section.

After installing and loading the ggplot2 package, you can use the following key functions:

<table>
<thead>
<tr>
<th>Plot types</th>
<th>GGPlot2 functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize a ggplot</td>
<td>ggplot()</td>
</tr>
<tr>
<td>Scatter plot</td>
<td>geom_point()</td>
</tr>
<tr>
<td>Box plot</td>
<td>geom_boxplot()</td>
</tr>
<tr>
<td>Violin plot</td>
<td>geom_violin()</td>
</tr>
<tr>
<td>strip chart</td>
<td>geom_jitter()</td>
</tr>
<tr>
<td>Dot plot</td>
<td>geom_dotplot()</td>
</tr>
<tr>
<td>Bar chart</td>
<td>geom_bar()</td>
</tr>
<tr>
<td>Line plot</td>
<td>geom_line()</td>
</tr>
<tr>
<td>Histogram</td>
<td>geom_histogram()</td>
</tr>
<tr>
<td>Density plot</td>
<td>geom_density()</td>
</tr>
<tr>
<td>Error bars</td>
<td>geom_errorbar()</td>
</tr>
<tr>
<td>QQ plot</td>
<td>stat_qq()</td>
</tr>
<tr>
<td>ECDF plot</td>
<td>stat_ecdf()</td>
</tr>
<tr>
<td>Title and axis labels</td>
<td>labs()</td>
</tr>
</tbody>
</table>

The main function in the ggplot2 package is **ggplot()**, which can be used to initialize the plotting system with data and x/y variables.

For example, the following R code takes the iris data set to initialize the ggplot and then a layer (**geom_point()**) is added onto the ggplot to create a scatter plot of x = Sepal.Length by y = Sepal.Width:

```r
library(ggplot2)
pplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_point()
```
# Change point size, color and shape

ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(size = 1.2, color = "steelblue", shape = 21)

Note that, in the code above, the shape of points is specified as number. To display the different point shape available in R, type this:

```r
ggpubr::show_point_shapes()
```

It’s also possible to control points shape and color by a grouping variable (here, `Species`). For example, in the code below, we map points color and shape to the `Species` grouping variable.

```r
# Control points color by groups

ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(color = Species, shape = Species))

# Change the default color manually.
# Use the scale_color_manual() function

ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(color = Species, shape = Species)) +
  scale_color_manual(values = c("#0AFBB", "#E7B800", "#FC4E07"))
```
You can also split the plot into multiple panels according to a grouping variable. R function: `facet_wrap()`. Another interesting feature of ggplot2, is the possibility to combine multiple layers on the same plot. For example, with the following R code, we'll:

- Add points with `geom_point()`, colored by groups.
- Add the fitted smoothed regression line using `geom_smooth()`. By default the function `geom_smooth()` add the regression line and the confidence area. You can control the line color and confidence area fill color by groups.
- Facet the plot into multiple panels by groups
- Change color and fill manually using the function `scale_color_manual()` and `scale_fill_manual()`

```r
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom_point(aes(color = Species)) + geom_smooth(aes(color = Species, fill = Species)) + facet_wrap(~Species, ncol = 3, nrow = 1) + scale_color_manual(values = c("#00AFBB", "#E7B800", "#FC4E07")) + scale_fill_manual(values = c("#00AFBB", "#E7B800", "#FC4E07"))
```

Note that, the default theme of ggplots is `theme_gray()` (or `theme_grey()`), which is theme with grey background and white grid lines. More themes are available for professional presentations or publications. These include: `theme_bw()`, `theme_classic()` and
1.8. R GRAPHICS SYSTEMS

theme_minimal().

To change the theme of a given ggplot (p), use this: `p + theme_classic()`. To change the default theme to `theme_classic()` for all the future ggplots during your entire R session, type the following R code:

```r
theme_set(  
    theme_classic()
)
```

Now you can create ggplots with `theme_classic()` as default theme:

```r
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
    geom_point()
```

![Graph showing the relationship between Sepal.Length and Sepal.Width](image)

### 1.8.4 ggpubr for publication ready plots

The **ggpubr** R package facilitates the creation of beautiful ggplot2-based graphs for researcher with non-advanced programming backgrounds (Kassambara, 2017).

For example, to create the density distribution of “Sepal.Length”, colored by groups (“Species”), type this:

```r
library(ggpubr)
# Density plot with mean lines and marginal rug
ggdensity(iris, x = "Sepal.Length",  
    add = "mean", rug = TRUE,  
    color = "Species", fill = "Species",  
    palette = "jco")  
```

# Add mean line and marginal rugs
# Color by groups
# use jco journal color palette
Note that the argument `palette` can take also a custom color palette. For example
`palette = c("#00AFBB", "#E7B800", "#FC4E07").`

- Create a box plot with p-values comparing groups:

```r
# Groups that we want to compare
my_comparisons <- list(
  c("setosa", "versicolor"),
  c("versicolor", "virginica"),
  c("setosa", "virginica")
)

# Create the box plot. Change colors by groups: Species
# Add jitter points and change the shape by groups
ggboxplot(iris, x = "Species", y = "Sepal.Length",
          color = "Species", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
          add = "jitter"
) +
stat_compare_means(comparisons = my_comparisons, method = "t.test")
```
1.9 Export R graphics

You can export R graphics to many file formats, including: PDF, PostScript, SVG vector files, Windows MetaFile (WMF), PNG, TIFF, JPEG, etc.

The standard procedure to save any graphics from R is as follow:

1. **Open a graphic device** using one of the following functions:
   - `pdf("r-graphics.pdf")`,
   - `postscript("r-graphics.ps")`,
   - `svg("r-graphics.svg")`,
   - `png("r-graphics.png")`,
   - `tiff("r-graphics.tif")`,
   - `jpeg("r-graphics.jpg")`,
   - `win.metafile("r-graphics.wmf")`,
   - and so on.

   Additional arguments indicating the width and the height (in inches) of the graphics region can be also specified in the mentioned function.

2. **Create a plot**

3. **Close the graphic device** using the function `dev.off()`

For example, you can export R base plots to a pdf file as follow:

Learn more on STHDA at: ggpubr: Publication Ready Plots

http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/
To export ggplot2 graphs, the R code looks like this:

```r
# Create some plots
library(ggplot2)
myplot1 <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) + geom_point()
myplot2 <- ggplot(iris, aes(Species, Sepal.Length)) + geom_boxplot()

# Print plots to a pdf file
pdf("ggplot.pdf")
print(myplot1) # Plot 1 --> in the first page of PDF
print(myplot2) # Plot 2 ---> in the second page of the PDF
dev.off()
```

Note that for a ggplot, you can also use the following functions to export the graphic:

- `ggsave()` [in ggplot2]. Makes it easy to save a ggplot. It guesses the type of graphics device from the file extension.
- `ggexport()` [in ggpubr]. Makes it easy to arrange and export multiple ggplots at once.

See also the following blog post to save high-resolution ggplots:

Chapter 2

Plot One Variable

2.1 Introduction

To visualize one variable, the type of graphs to be used depends on the type of the variable:

- For **categorical variable** or grouping variables. You can visualize the count of categories using a bar plot or using a pie chart to show the proportion of each category.
- For **continuous variable**, you can visualize the distribution of the variable using density plots, histograms and alternatives.

In this R graphics tutorial, you’ll learn how to:

- Visualize a categorical variable using bar plots, dot charts and pie charts
- Visualize the distribution of a continuous variable using:
  - density and histogram plots,
  - other alternatives, such as frequency polygon, area plots, dot plots, box plots, Empirical cumulative distribution function (ECDF) and Quantile-quantile plot (QQ plots).
  - Density ridgeline plots, which are useful for visualizing changes in distributions, of a continuous variable, over time or space.
  - Bar plot and modern alternatives, including lollipop charts and cleveland’s dot plots.

2.2 Prerequisites

Load required packages and set the theme function `theme_pubr()` [in ggpubr] as the default theme:

```r
library(ggplot2)
library(ggpubr)
theme_set(theme_pubr())
```
2.3 One categorical variable

2.3.1 Bar plot of counts

- Plot types: Bar plot of the count of group levels
- Key function: `geom_bar()`
- Key arguments: `alpha`, `color`, `fill`, `linetype` and `size`

Demo data set: `diamonds` [in ggplot2]. Contains the prices and other attributes of almost 54,000 diamonds. The column `cut` contains the quality of the diamonds cut (Fair, Good, Very Good, Premium, Ideal).

The R code below creates a bar plot visualizing the number of elements in each category of diamonds cut.

```r
ggplot(diamonds, aes(cut)) +
  geom_bar(fill = "#0073C2FF") +
  theme_pubclean()
```

Compute the frequency of each category and add the labels on the bar plot:

- `dplyr` package used to summarise the data
- `geom_bar()` with option `stat = "identity"` is used to create the bar plot of the summary output as it is.
- `geom_text()` used to add text labels. Adjust the position of the labels by using `hjust` (horizontal justification) and `vjust` (vertical justification). Values should be in [0, 1].

```r
# Compute the frequency
library(dplyr)
df <- diamonds %>%
  group_by(cut) %>%
  summarise(counts = n())
df
```

```r
## A tibble: 5 x 2
## cut  counts
## <ord> <int>
##   Fair       4
##   Good      10
## Very Good  15
##   Premium  20
##     Ideal  25
```
2.3. ONE CATEGORICAL VARIABLE

## Fair 1610
## Good 4906
## Very Good 12082
## Premium 13791
## Ideal 21551

# Create the bar plot. Use theme_pubclean() [in ggpubr]
```r
ggplot(df, aes(x = cut, y = counts)) +
  geom_bar(fill = "#0073C2FF", stat = "identity") +
  geom_text(aes(label = counts), vjust = -0.3) +
  theme_pubclean()
```

2.3.2 Pie charts

Pie chart is just a stacked bar chart in polar coordinates.

First,

- Arrange the grouping variable (cut) in descending order. This important to compute the y coordinates of labels.
- compute the proportion (counts/total) of each category
- compute the position of the text labels as the cumulative sum of the proportion. To put the labels in the center of pies, we’ll use `cumsum(prop) - 0.5*prop` as label position.

```r
df <- df %>%
  arrange(desc(cut)) %>%
  mutate(prop = round(counts*100/sum(counts), 1),
         lab.ypos = cumsum(prop) - 0.5*prop)
head(df, 4)
```

## A tibble: 4 x 4
## cut counts  prop lab.ypos
Create the pie charts using ggplot2 verbs. Key function: coord_polar().

```r
ggplot(df, aes(x = "", y = prop, fill = cut)) +
geom_bar(width = 1, stat = "identity", color = "white") +
geom_text(aes(y = lab.ypos, label = prop), color = "white") +
coord_polar("y", start = 0) +
ggpubr::fill_palette("jco") +
theme_void()
```

Alternative solution to easily create a pie chart: use the function `ggpie()` in ggpubr:

```r
ggpie(
  df, x = "prop", label = "prop",
  lab.pos = "in", lab.font = list(color = "white"),
  fill = "cut", color = "white",
  palette = "jco"
)
```

### 2.3.3 Dot charts

Dot chart is an alternative to bar plots. Key functions:

- `geom_linerange()`: Creates line segments from x to ymax
- `geom_point()`: adds dots
- `ggpubr::color_palette()`: changes color palette.

```r
ggplot(df, aes(cut, prop)) +
geom_linerange(
  aes(x = cut, ymin = 0, ymax = prop),
  color = "lightgray", size = 1.5
)```
2.4. **ONE CONTINUOUS VARIABLE**

Different types of graphs can be used to visualize the distribution of a continuous variable, including: density and histogram plots.

### 2.4.1 Data format

Create some data (wdata) containing the weights by sex (M for male; F for female):

```r
set.seed(1234)
wdata = data.frame(
  sex = factor(rep(c("F", "M"), each=200)),
  weight = c(rnorm(200, 55), rnorm(200, 58))
)

head(wdata, 4)
```

Easy alternative to create a dot chart. Use `ggdotchart()` [ggpubr]:

```r
ggdotchart(
  df, x = "cut", y = "prop",
  color = "cut", size = 3, # Points color and size
  add = "segment", # Add line segments
  add.params = list(size = 2),
  palette = "jco",
  ggtheme = theme_pubclean()
)
```