**Plot One Variable: Frequency Graph, Density Distribution and More**

[*kassambara*](http://www.sthda.com/english/user/profile/1)*|   17/11/2017 |   282230  |*[*Comments (2)*](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#comments-list)*|*[*R Graphics Essentials*](http://www.sthda.com/english/articles/32-r-graphics-essentials/)

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

* For **categorical variables** (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.

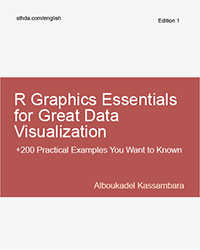
In this R graphics tutorial, you’ll learn how to:

* Visualize the frequency distribution of 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.

Contents:

* [Prerequisites](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#prerequisites)
* [One categorical variable](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#one-categorical-variable)
  + [Bar plot of counts](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#bar-plot-of-counts)
  + [Pie charts](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#pie-charts)
  + [Dot charts](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#dot-charts)
* [One continuous variable](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#one-continuous-variable)
  + [Data format](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#data-format)
  + [Basic plots](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#basic-plots)
  + [Density plots](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#density-plots)
  + [Histogram plots](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#histogram-plots)
  + [Alternative to density and histogram plots](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#alternative-to-density-and-histogram-plots)
  + [Density ridgeline plots](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#density-ridgeline-plots)
  + [Bar plot and modern alternatives](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#bar-plot-and-modern-alternatives)
* [Conclusion](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#conclusion)
* [References](http://www.sthda.com/english/articles/32-r-graphics-essentials/133-plot-one-variable-frequency-graph-density-distribution-and-more/#references)

The Book:

[[](http://www.sthda.com/english/web/5-bookadvisor/52-r-graphics-essentials-for-great-data-visualization-200-practical-examples-you-want-to-know-for-data-science/)  
R Graphics Essentials for Great Data Visualization: +200 Practical Examples You Want to Know for Data Science](http://www.sthda.com/english/web/5-bookadvisor/52-r-graphics-essentials-for-great-data-visualization-200-practical-examples-you-want-to-know-for-data-science/)

**Prerequisites**

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

**library**(ggplot2)

**library**(ggpubr)

theme\_set(theme\_pubr())

**One categorical variable**

**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

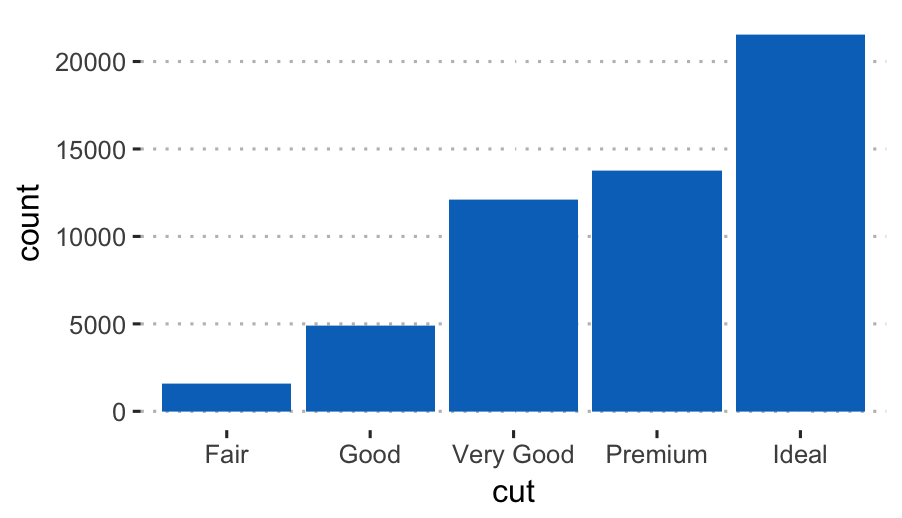
Demo data set: diamonds [in ggplot2]. Contains the prices and other attributes of almost 54000 diamonds. The column cut contains the quality of the diamonds cut (Fair, Good, Very Good, Premium, Ideal).

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

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].

# Compute the frequency

**library**(dplyr)

df <- diamonds %>%

group\_by(cut) %>%

summarise(counts = n())

df

## # A tibble: 5 x 2

## cut counts

##

## 1 Fair 1610

## 2 Good 4906

## 3 Very Good 12082

## 4 Premium 13791

## 5 Ideal 21551

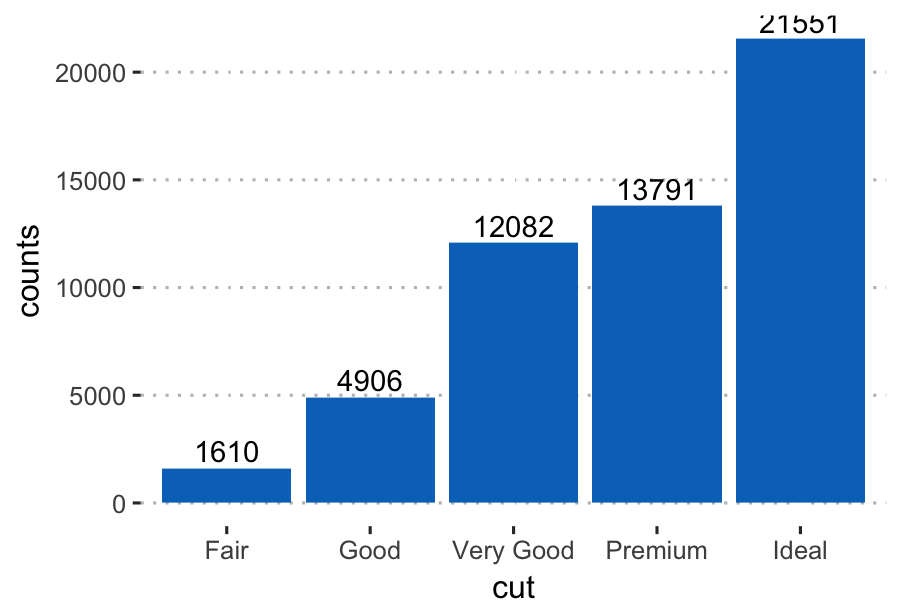
# Create the bar plot. Use theme\_pubclean() [in ggpubr]

ggplot(df, aes(x = cut, y = counts)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = counts), vjust = -0.3) +

theme\_pubclean()



**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.

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

##

## 1 Ideal 21551 40.0 20.0

## 2 Premium 13791 25.6 52.8

## 3 Very Good 12082 22.4 76.8

## 4 Good 4906 9.1 92.5

* Create the pie charts using ggplot2 verbs. Key function: coord\_polar().

ggplot(df, aes(x = "", y = prop, fill = cut)) +

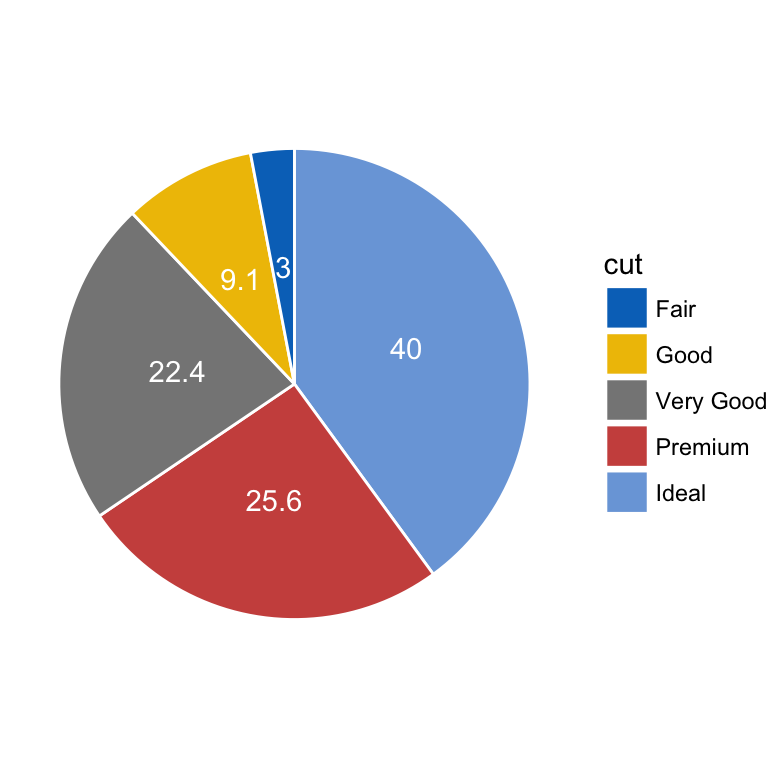
geom\_bar(width = 1, stat = "identity", color = "white") +

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]:

ggpie(

df, x = "prop", label = "prop",

lab.pos = "in", lab.font = list(color = "white"),

fill = "cut", color = "white",

palette = "jco"

)

**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.

ggplot(df, aes(cut, prop)) +

geom\_linerange(

aes(x = cut, ymin = 0, ymax = prop),

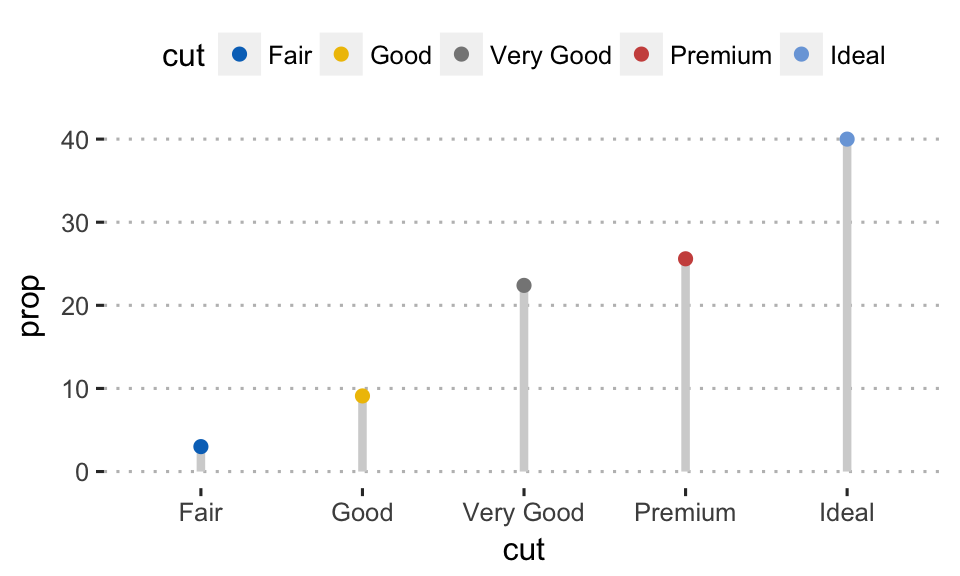
color = "lightgray", size = 1.5

)+

geom\_point(aes(color = cut), size = 2)+

ggpubr::color\_palette("jco")+

theme\_pubclean()



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

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()

)

**One continuous variable**

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

**Data format**

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

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)

## sex weight

## 1 F 53.8

## 2 F 55.3

## 3 F 56.1

## 4 F 52.7

Compute the mean weight by sex using the dplyr package. First, the data is grouped by sex and then summarized by computing the mean weight by groups. The operator %>% is used to combine multiple operations:

**library**("dplyr")

mu <- wdata %>%

group\_by(sex) %>%

summarise(grp.mean = mean(weight))

mu

## # A tibble: 2 x 2

## sex grp.mean

##

## 1 F 54.9

## 2 M 58.1

**Basic plots**

We start by creating a plot, named a, that we’ll finish in the next section by adding a layer.

a <- ggplot(wdata, aes(x = weight))

Possible layers include: geom\_density() (for density plots) and geom\_histogram() (for histogram plots).

Key arguments to customize the plots:

* color, size, linetype: change the line color, size and type, respectively
* fill: change the areas fill color (for bar plots, histograms and density plots)
* alpha: create a semi-transparent color.

**Density plots**

Key function: geom\_density().

1. **Create basic density plots**. Add a vertical line corresponding to the mean value of the weight variable (geom\_vline()):

# y axis scale = ..density.. (default behaviour)

a + geom\_density() +

geom\_vline(aes(xintercept = mean(weight)),

linetype = "dashed", size = 0.6)

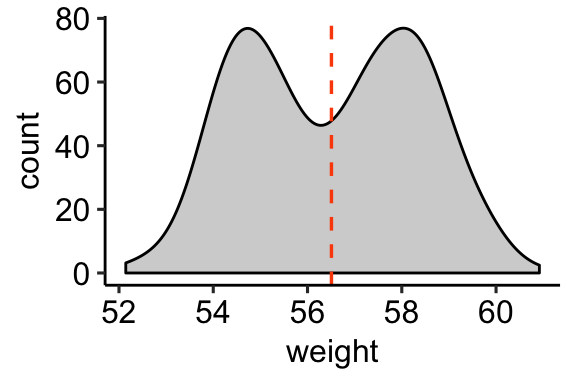
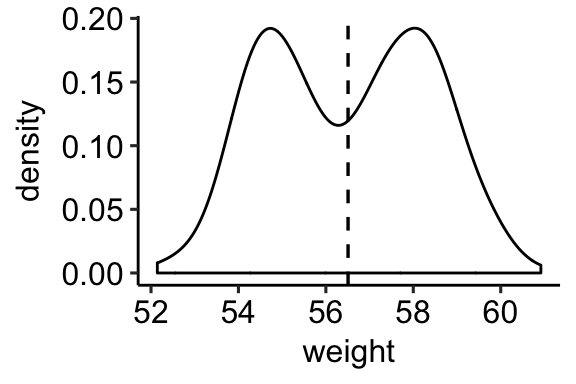
# Change y axis to count instead of density

a + geom\_density(aes(y = ..count..), fill = "lightgray") +

geom\_vline(aes(xintercept = mean(weight)),

linetype = "dashed", size = 0.6,

color = "#FC4E07")



1. **Change areas fill and add line color by groups** (sex):

* Add vertical mean lines using geom\_vline(). Data: mu, which contains the mean values of weights by sex (computed in the previous section).
* Change color manually:
  + use scale\_color\_manual() or scale\_colour\_manual() for changing line color
  + use scale\_fill\_manual() for changing area fill colors.

# Change line color by sex

a + geom\_density(aes(color = sex)) +

scale\_color\_manual(values = c("#868686FF", "#EFC000FF"))

# Change fill color by sex and add mean line

# Use semi-transparent fill: alpha = 0.4

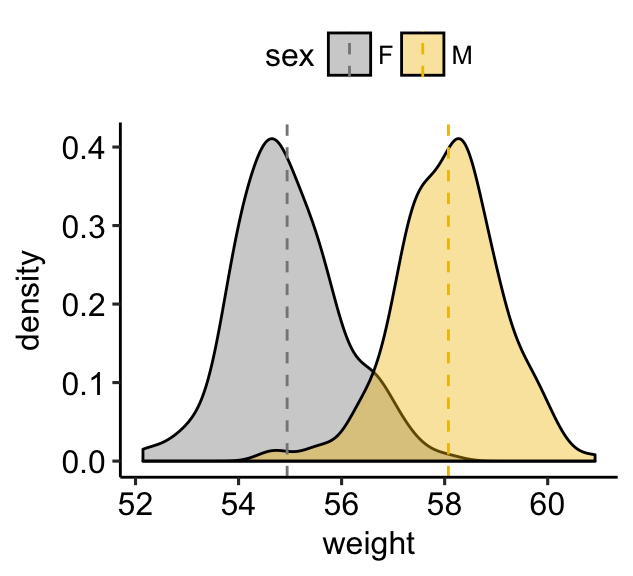
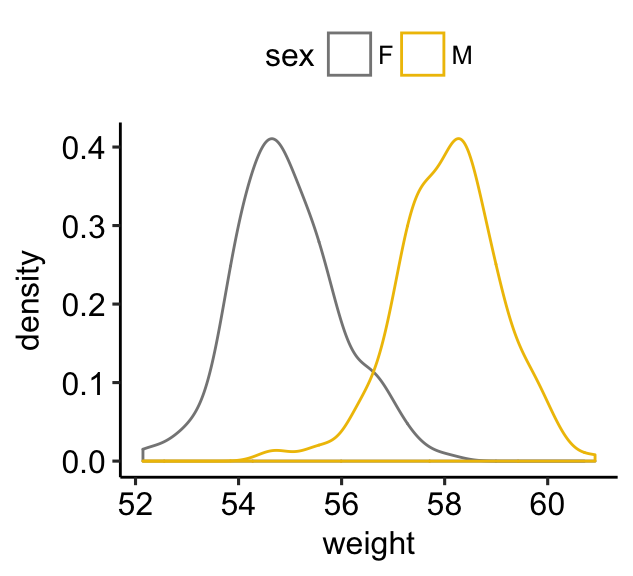
a + geom\_density(aes(fill = sex), alpha = 0.4) +

geom\_vline(aes(xintercept = grp.mean, color = sex),

data = mu, linetype = "dashed") +

scale\_color\_manual(values = c("#868686FF", "#EFC000FF"))+

scale\_fill\_manual(values = c("#868686FF", "#EFC000FF"))



1. **Simple solution to create a ggplot2-based density plots**: use ggboxplot() [in ggpubr].

**library**(ggpubr)

# Basic density plot with mean line and marginal rug

ggdensity(wdata, x = "weight",

fill = "#0073C2FF", color = "#0073C2FF",

add = "mean", rug = TRUE)

# Change outline and fill colors by groups ("sex")

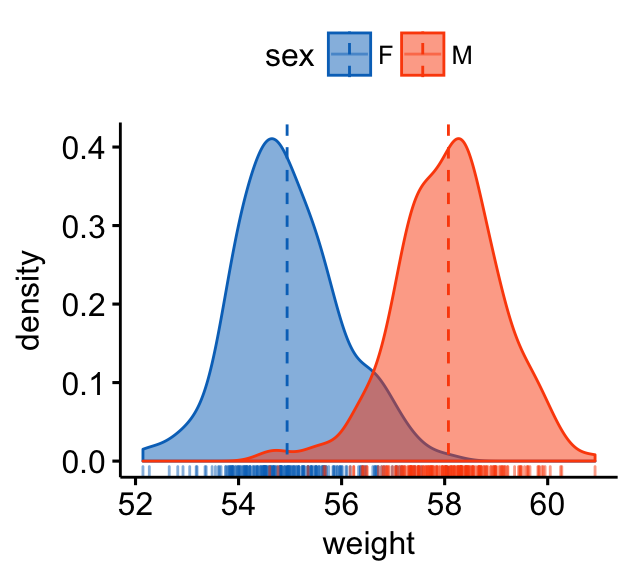
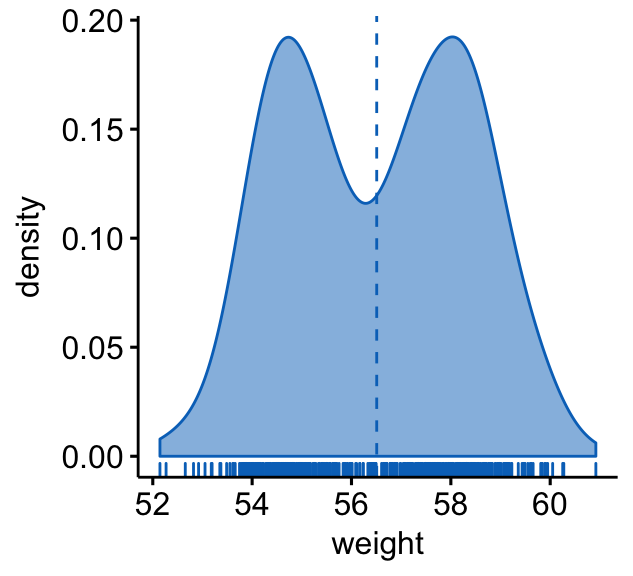
# Use a custom palette

ggdensity(wdata, x = "weight",

add = "mean", rug = TRUE,

color = "sex", fill = "sex",

palette = c("#0073C2FF", "#FC4E07"))



**Histogram plots**

An alternative to density plots is histograms, which represents the distribution of a continuous variable by dividing into bins and counting the number of observations in each bin.

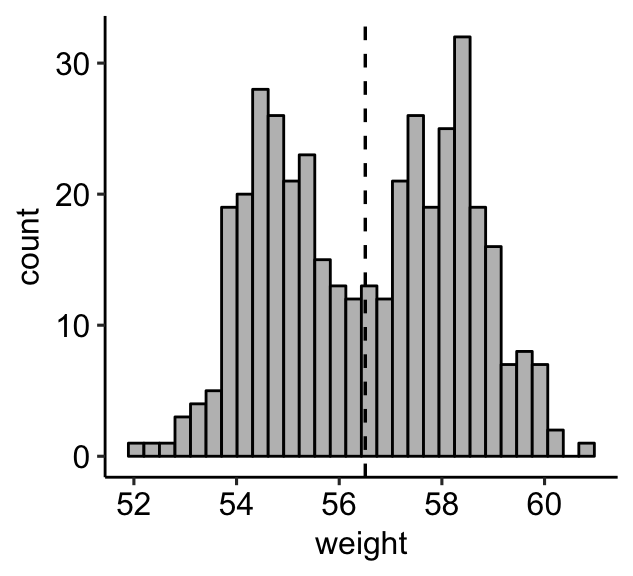
Key function: geom\_histogram(). The basic usage is quite similar to geom\_density().

1. **Create a basic plots**. Add a vertical line corresponding to the mean value of the weight variable:

a + geom\_histogram(bins = 30, color = "black", fill = "gray") +

geom\_vline(aes(xintercept = mean(weight)),

linetype = "dashed", size = 0.6)



Note that, by default:

* By default, geom\_histogram() uses 30 bins - this might not be good default. You can change the number of bins (e.g.: bins = 50) or the bin width (e.g.: binwidth = 0.5)
* The y axis corresponds to the count of weight values. If you want to change the plot in order to have the density on y axis, specify the argument y = ..density.. in aes().

1. **Change areas fill and add line color by groups** (sex):

* Add vertical mean lines using geom\_vline(). Data: mu, which contains the mean values of weights by sex.
* Change color manually:
  + use scale\_color\_manual() or scale\_colour\_manual() for changing line color
  + use scale\_fill\_manual() for changing area fill colors.
* Adjust the position of histogram bars by using the argument position. Allowed values: “identity”, “stack”, “dodge”. Default value is “stack”.

# Change line color by sex

a + geom\_histogram(aes(color = sex), fill = "white",

position = "identity") +

scale\_color\_manual(values = c("#00AFBB", "#E7B800"))

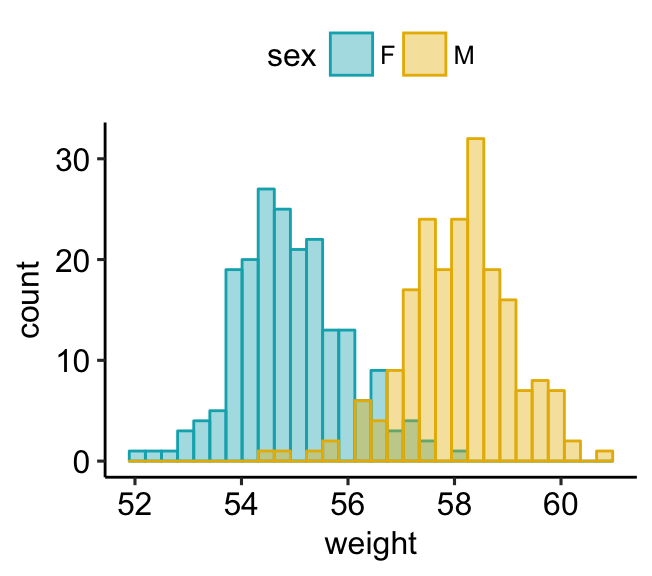
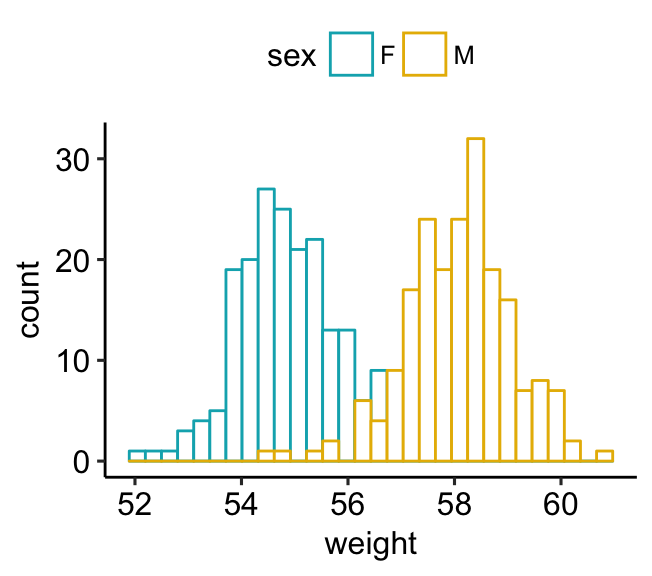
# change fill and outline color manually

a + geom\_histogram(aes(color = sex, fill = sex),

alpha = 0.4, position = "identity") +

scale\_fill\_manual(values = c("#00AFBB", "#E7B800")) +

scale\_color\_manual(values = c("#00AFBB", "#E7B800"))



1. **Combine histogram and density plots**:

* Plot histogram with density values on y-axis (instead of count values).
* Add density plot with transparent density plot

# Histogram with density plot

a + geom\_histogram(aes(y = ..density..),

colour="black", fill="white") +

geom\_density(alpha = 0.2, fill = "#FF6666")

# Color by groups

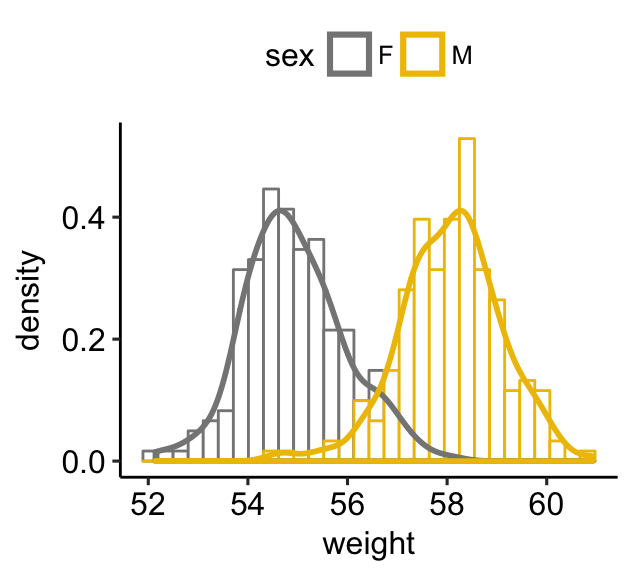
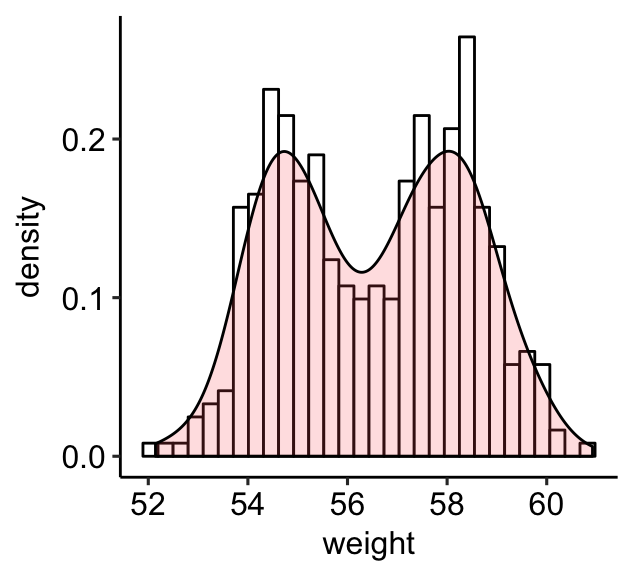
a + geom\_histogram(aes(y = ..density.., color = sex),

fill = "white",

position = "identity")+

geom\_density(aes(color = sex), size = 1) +

scale\_color\_manual(values = c("#868686FF", "#EFC000FF"))



1. **Simple solution to create a ggplot2-based histogram plots**: use gghistogram() [in ggpubr].

**library**(ggpubr)

# Basic histogram plot with mean line and marginal rug

gghistogram(wdata, x = "weight", bins = 30,

fill = "#0073C2FF", color = "#0073C2FF",

add = "mean", rug = TRUE)

# Change outline and fill colors by groups ("sex")

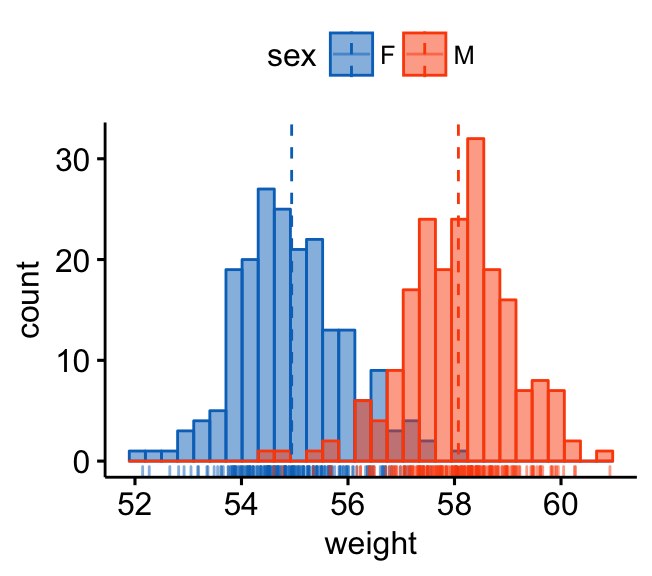
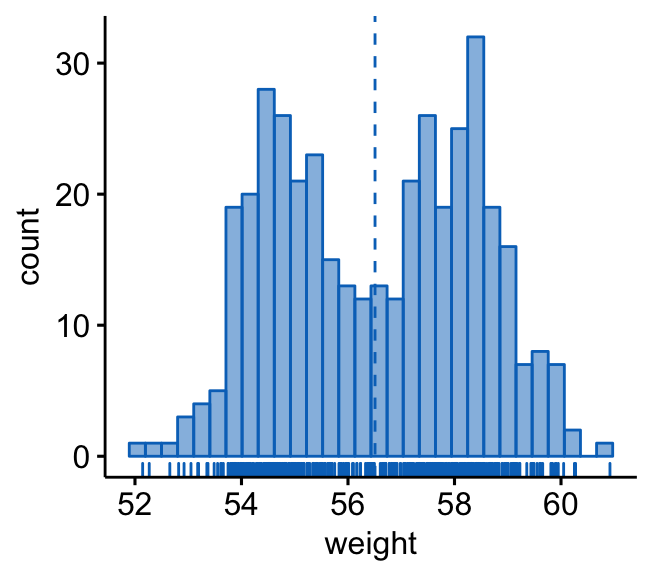
# Use a custom palette

gghistogram(wdata, x = "weight", bins = 30,

add = "mean", rug = TRUE,

color = "sex", fill = "sex",

palette = c("#0073C2FF", "#FC4E07"))



**Alternative to density and histogram plots**

1. **Frequency polygon**. Very close to histogram plots, but it uses lines instead of bars.
   * Key function: geom\_freqpoly().
   * Key arguments: color, size, linetype: change, respectively, line color, size and type.
2. **Area plots**. This is a continuous analog of a stacked bar plot.
   * Key function: geom\_area().
   * Key arguments:
     + color, size, linetype: change, respectively, line color, size and type.
     + fill: change area fill color.

In this section, we’ll use the theme theme\_pubclean() [in ggpubr]. This is a theme without axis lines, to direct more attention to the data. Type this to use the theme:

theme\_set(theme\_pubclean())

* Create a basic frequency polygon and basic area plots:

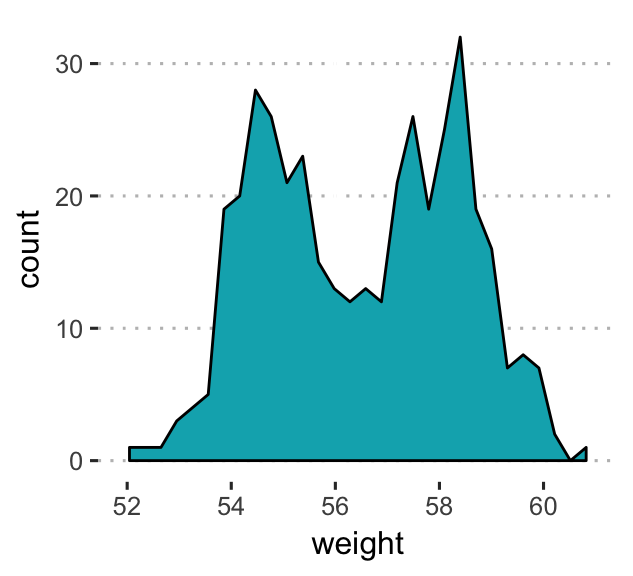
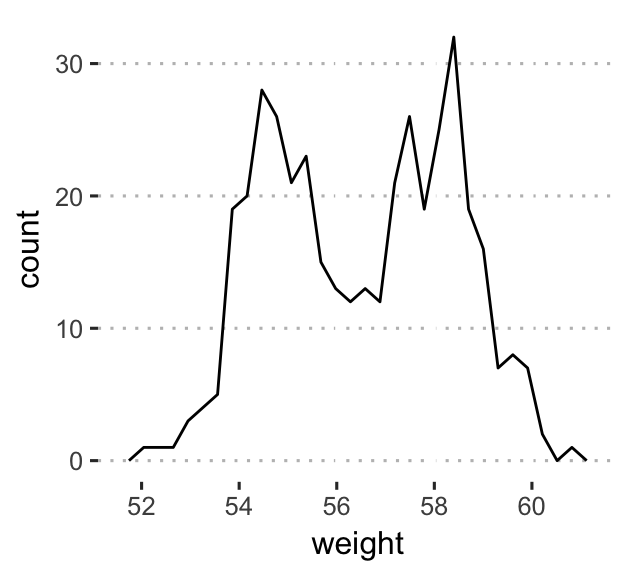
# Basic frequency polygon

a + geom\_freqpoly(bins = 30)

# Basic area plots, which can be filled by color

a + geom\_area( stat = "bin", bins = 30,

color = "black", fill = "#00AFBB")



* Change colors by groups (sex):

# Frequency polygon:

# Change line colors and types by groups

a + geom\_freqpoly( aes(color = sex, linetype = sex),

bins = 30, size = 1.5) +

scale\_color\_manual(values = c("#00AFBB", "#E7B800"))

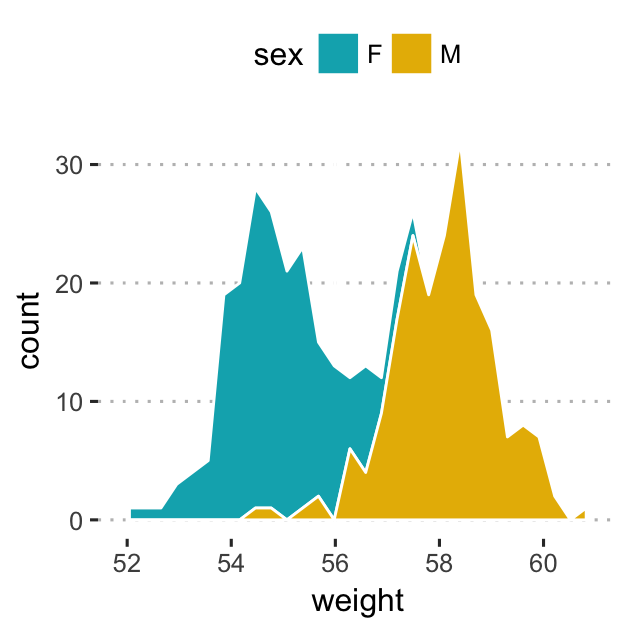
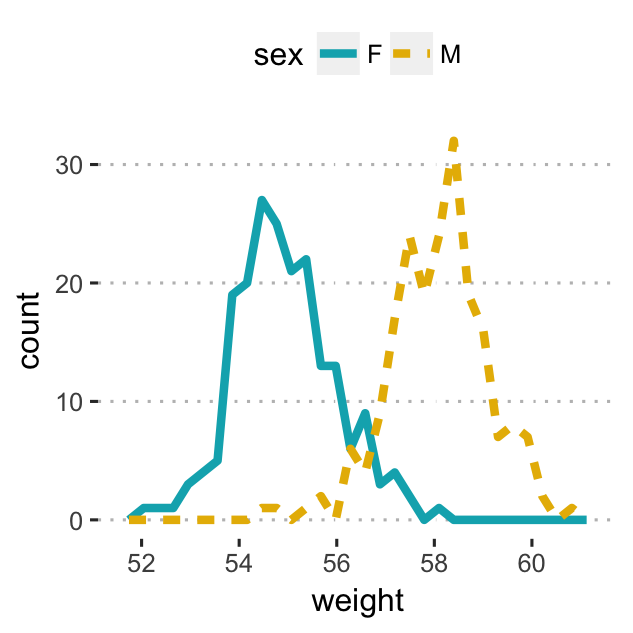
# Area plots: change fill colors by sex

# Create a stacked area plots

a + geom\_area(aes(fill = sex), color = "white",

stat ="bin", bins = 30) +

scale\_fill\_manual(values = c("#00AFBB", "#E7B800"))



As in histogram plots, the default y values is count. To have density values on y axis, specify y = ..density.. in aes().

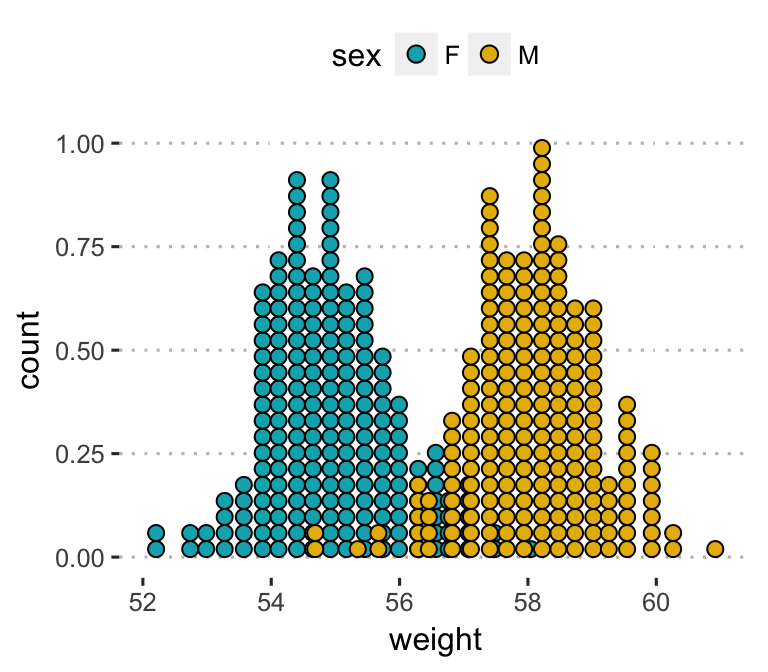
1. **Dot plots**. Represents another alternative to histograms and density plots, that can be used to visualize a continuous variable. Dots are stacked with each dot representing one observation. The width of a dot corresponds to the bin width.

* Key function: geom\_dotplot().
* Key arguments: alpha, color, fill and dotsize.

Create a dot plot colored by groups (sex):

a + geom\_dotplot(aes(fill = sex), binwidth = 1/4) +

scale\_fill\_manual(values = c("#00AFBB", "#E7B800"))



1. **Box plot**:
   * Create a box plot of one continuous variable: geom\_boxplot()
   * Add jittered points, where each point corresponds to an individual observation: geom\_jitter(). Change the color and the shape of points by groups (sex)

ggplot(wdata, aes(x = factor(1), y = weight)) +

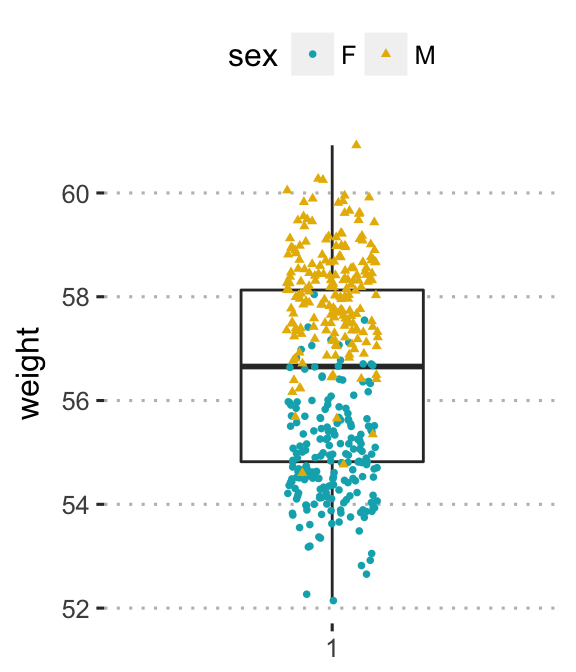
geom\_boxplot(width = 0.4, fill = "white") +

geom\_jitter(aes(color = sex, shape = sex),

width = 0.1, size = 1) +

scale\_color\_manual(values = c("#00AFBB", "#E7B800")) +

labs(x = NULL) # Remove x axis label



1. **Empirical cumulative distribution function (ECDF)**. Provides another alternative visualization of distribution. It reports for any given number the percent of individuals that are below that threshold.

For example, in the following plots, you can see that:

* about 25% of our females are shorter than 50 inches
* about 50% of males are shorter than 58 inches

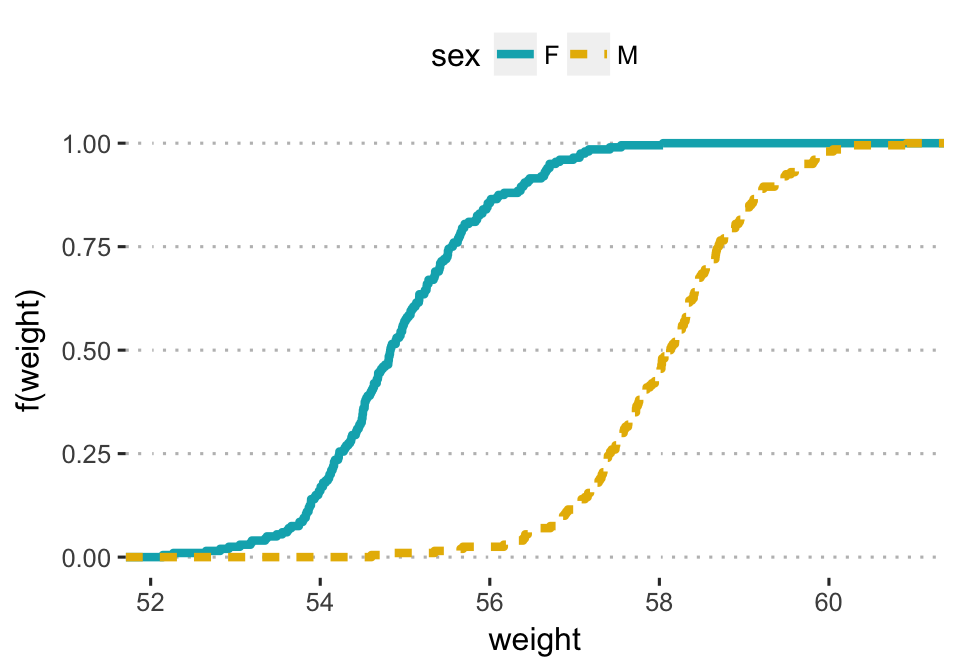
# Another option for geom = "point"

a + stat\_ecdf(aes(color = sex,linetype = sex),

geom = "step", size = 1.5) +

scale\_color\_manual(values = c("#00AFBB", "#E7B800"))+

labs(y = "f(weight)")



1. **Quantile-quantile plot** (QQ plots). Used to check whether a given data follows normal distribution.

* Key function: stat\_qq().
* Key arguments: color, shape and size to change point color, shape and size.

Create a qq-plot of weight. Change color by groups (sex)

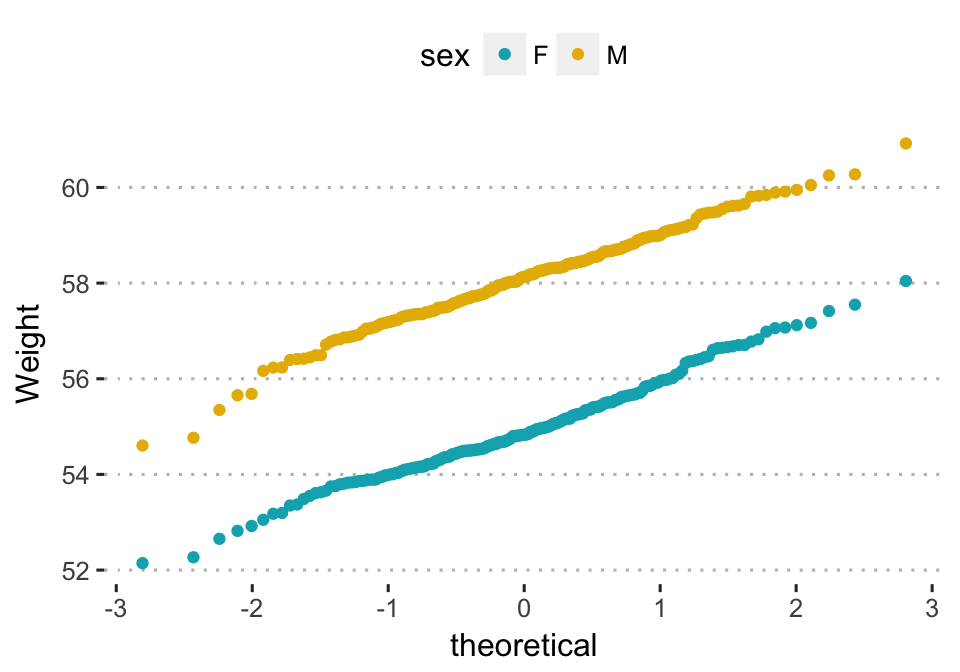
# Change point shapes by groups

ggplot(wdata, aes(sample = weight)) +

stat\_qq(aes(color = sex)) +

scale\_color\_manual(values = c("#00AFBB", "#E7B800"))+

labs(y = "Weight")



Alternative plot using the function ggqqplot() [in ggpubr]. The 95% confidence band is shown by default.

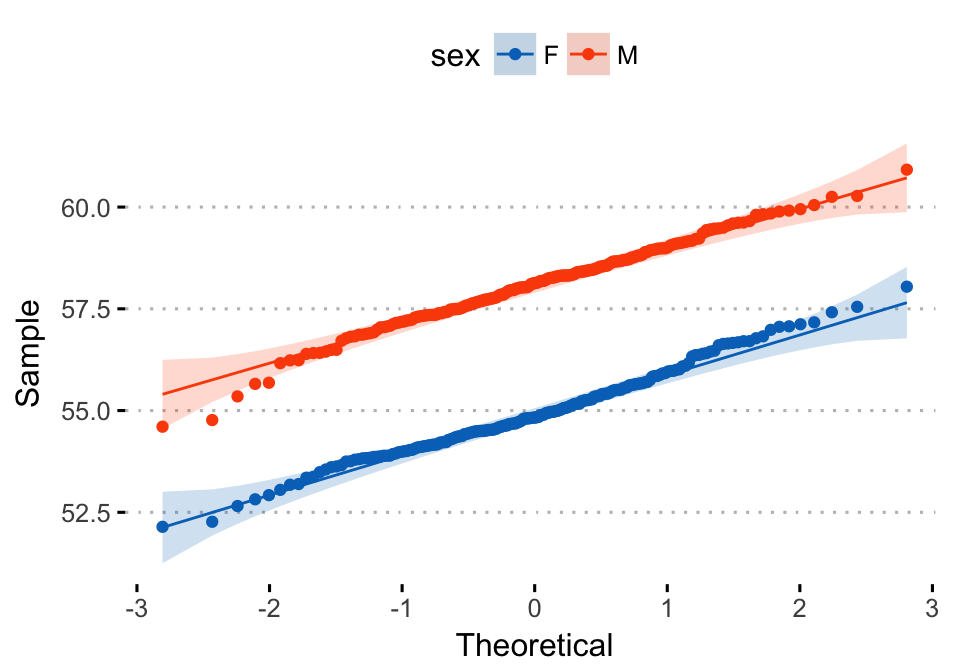
**library**(ggpubr)

ggqqplot(wdata, x = "weight",

color = "sex",

palette = c("#0073C2FF", "#FC4E07"),

ggtheme = theme\_pubclean())



**Density ridgeline plots**

The density ridgeline plot is an alternative to the standard geom\_density() function that can be useful for visualizing changes in distributions, of a continuous variable, over time or space. Ridgeline plots are partially overlapping line plots that create the impression of a mountain range.

This functionality is provided in the R package ggridges (Wilke 2017).

1. **Installation**:

install.packages("ggridges")

1. **Load and set the default theme** to theme\_ridges() [in ggridges]:

**library**(ggplot2)

**library**(ggridges)

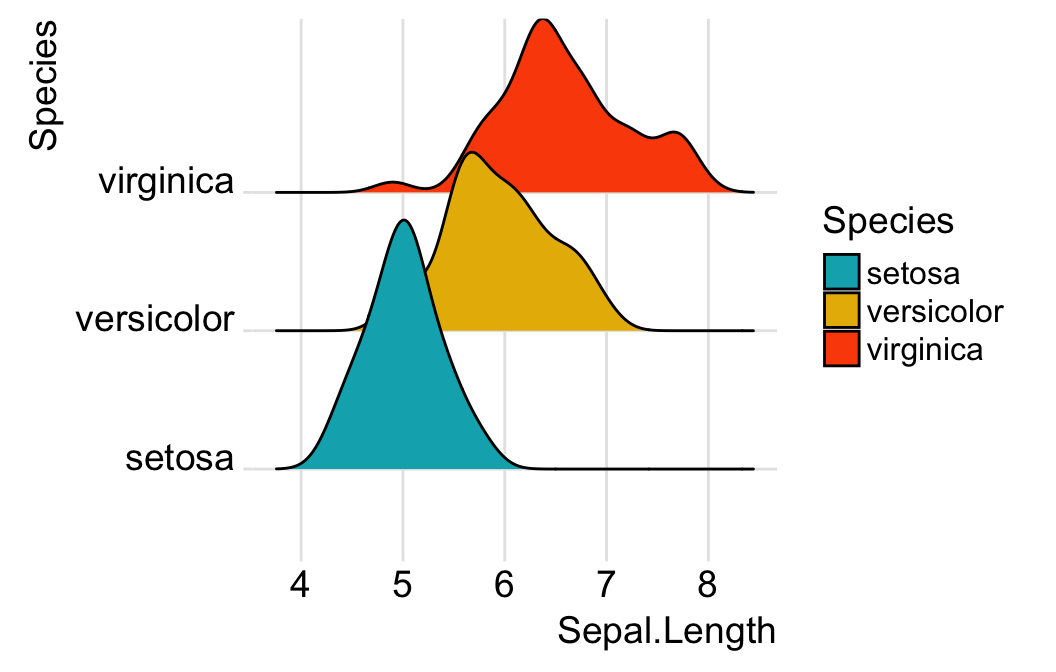
theme\_set(theme\_ridges())

1. **Example 1: Simple distribution plots by groups**. Distribution of Sepal.Length by Species using the iris data set. The grouping variable Species will be mapped to the y-axis:

ggplot(iris, aes(x = Sepal.Length, y = Species)) +

geom\_density\_ridges(aes(fill = Species)) +

scale\_fill\_manual(values = c("#00AFBB", "#E7B800", "#FC4E07"))



You can control the overlap between the different densities using the scale option. Default value is 1. Smaller values create a separation between the curves, and larger values create more overlap.

ggplot(iris, aes(x = Sepal.Length, y = Species)) +

geom\_density\_ridges(scale = 0.9)

1. **Example 4: Visualize temperature data**.

* Data set: lincoln\_weather [in ggridges]. Weather in Lincoln, Nebraska in 2016.
* Create the density ridge plots of the Mean Temperature by Month and change the fill color according to the temperature value (on x axis). A gradient color is created using the function geom\_density\_ridges\_gradient()

ggplot(

lincoln\_weather,

aes(x = `Mean Temperature [F]`, y = `Month`)

) +

geom\_density\_ridges\_gradient(

aes(fill = ..x..), scale = 3, size = 0.3

) +

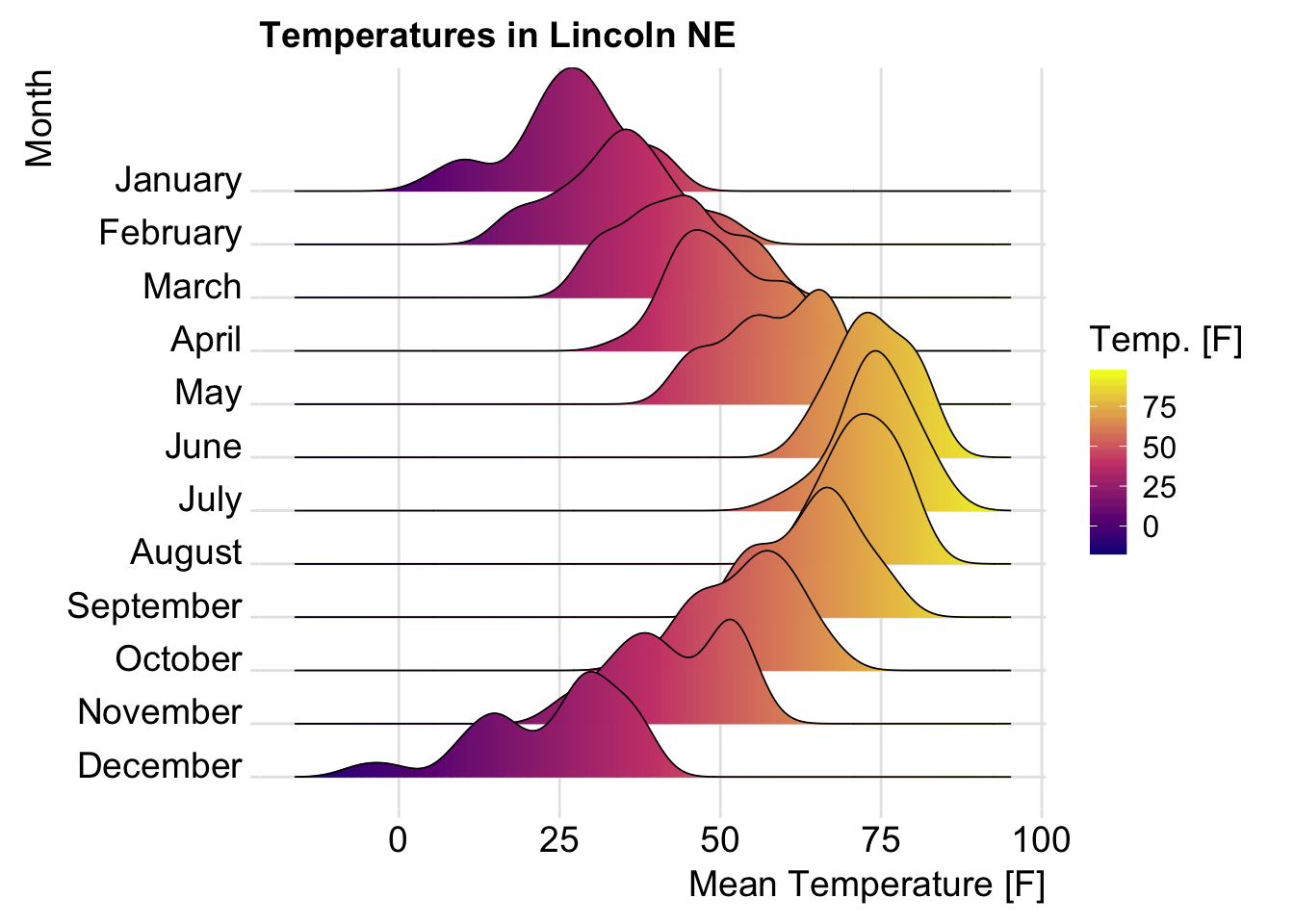
scale\_fill\_gradientn(

colours = c("#0D0887FF", "#CC4678FF", "#F0F921FF"),

name = "Temp. [F]"

)+

labs(title = 'Temperatures in Lincoln NE')



For more examples, type the following R code:

browseVignettes("ggridges")

**Bar plot and modern alternatives**

In this section, we’ll describe how to create easily basic and ordered bar plots using ggplot2 based helper functions available in the ggpubr R package. We’ll also present some modern alternatives to bar plots, including lollipop charts and cleveland’s dot plots.

* Load required packages:

**library**(ggpubr)

* Load and prepare data:

# Load data

dfm <- mtcars

# Convert the cyl variable to a factor

dfm$cyl <- as.factor(dfm$cyl)

# Add the name colums

dfm$name <- rownames(dfm)

# Inspect the data

**head**(dfm[, c("name", "wt", "mpg", "cyl")])

## name wt mpg cyl

## Mazda RX4 Mazda RX4 2.62 21.0 6

## Mazda RX4 Wag Mazda RX4 Wag 2.88 21.0 6

## Datsun 710 Datsun 710 2.32 22.8 4

## Hornet 4 Drive Hornet 4 Drive 3.21 21.4 6

## Hornet Sportabout Hornet Sportabout 3.44 18.7 8

## Valiant Valiant 3.46 18.1 6

* Create an ordered bar plot of the mpg variable. Change the fill color by the grouping variable “cyl”. Sorting will be done globally, but not by groups.

ggbarplot(dfm, x = "name", y = "mpg",

fill = "cyl", # change fill color by cyl

color = "white", # Set bar border colors to white

palette = "jco", # jco journal color palett. see ?ggpar

sort.val = "asc", # Sort the value in dscending order

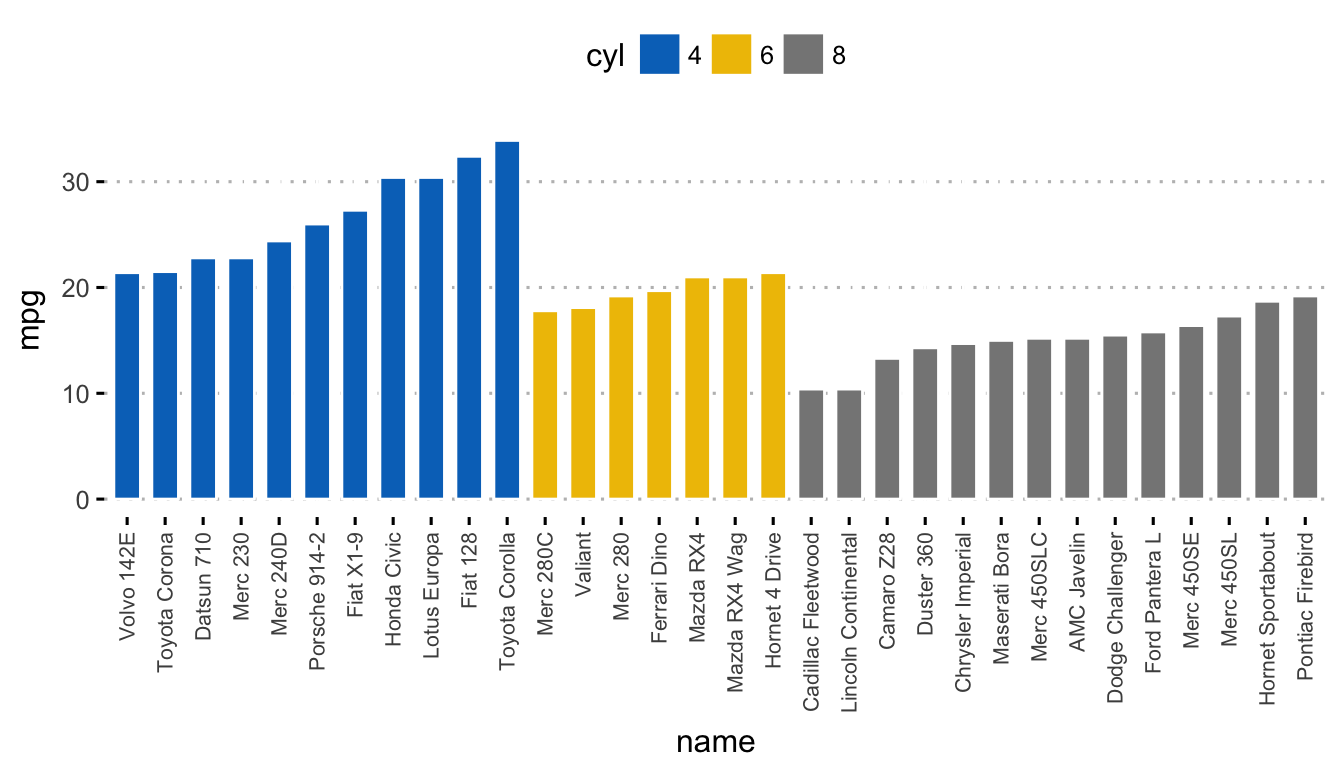
sort.by.groups = TRUE, # Don't sort inside each group

x.text.angle = 90, # Rotate vertically x axis texts

ggtheme = theme\_pubclean()

)+

font("x.text", size = 8, vjust = 0.5)



To sort bars inside each group, use the argument **sort.by.groups = TRUE**

* Create a Lollipop chart:
  + Color by groups and set a custom color palette.
  + Sort values in ascending order.
  + Add segments from y = 0 to dots. Change segment color and size.

ggdotchart(dfm, x = "name", y = "mpg",

color = "cyl",

palette = c("#00AFBB", "#E7B800", "#FC4E07"),

sorting = "asc", sort.by.groups = TRUE,

add = "segments",

add.params = list(color = "lightgray", size = 2),

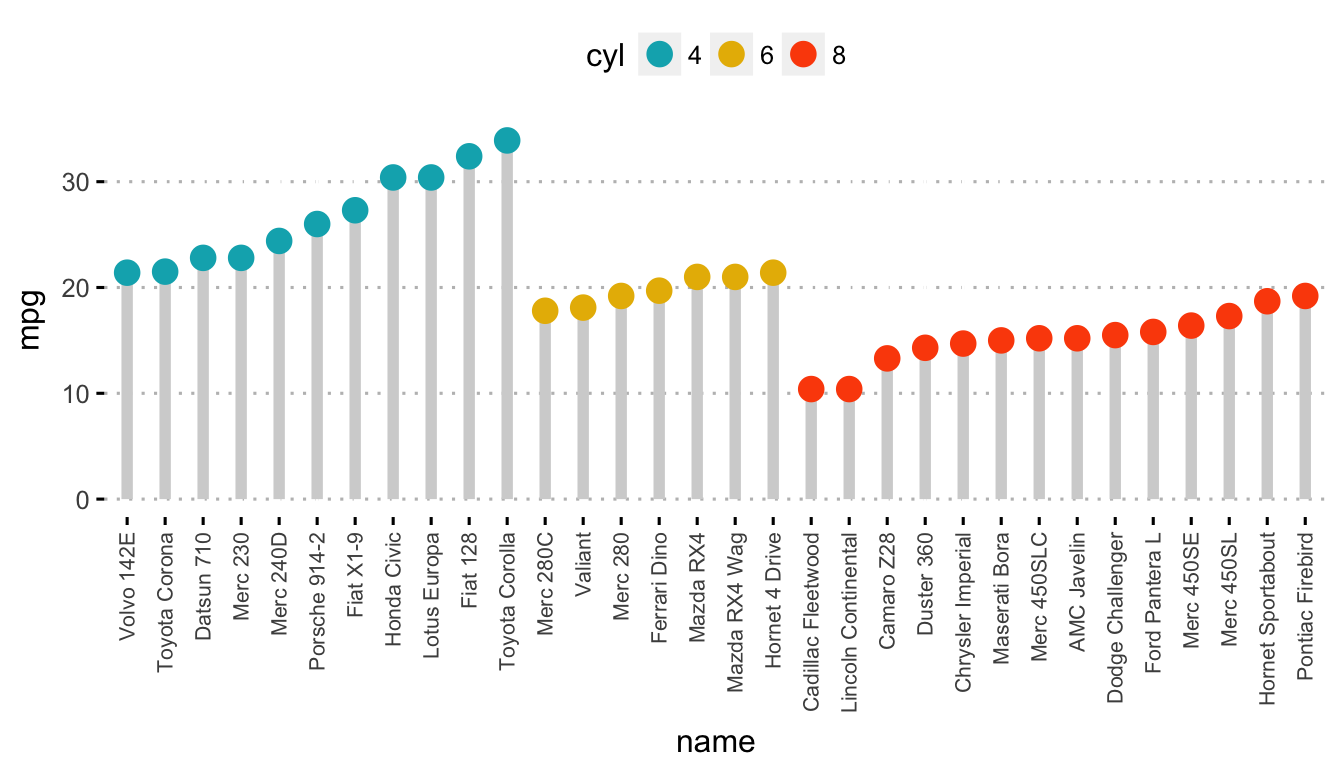
group = "cyl",

dot.size = 4,

ggtheme = theme\_pubclean()

)+

font("x.text", size = 8, vjust = 0.5)



Read more: [Bar Plots and Modern Alternatives](https://goo.gl/eSggcW)

**Conclusion**

* Create a bar plot of a grouping variable:

ggplot(diamonds, aes(cut)) +

geom\_bar(fill = "#0073C2FF") +

theme\_minimal()

* Visualize a continuous variable:

Start by creating a plot, named a, that we’ll be finished by adding a layer.

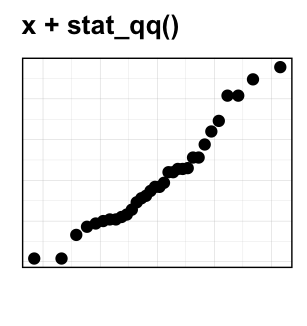
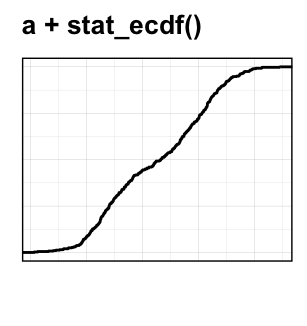
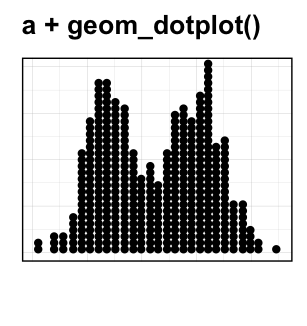
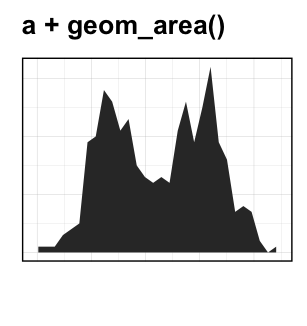
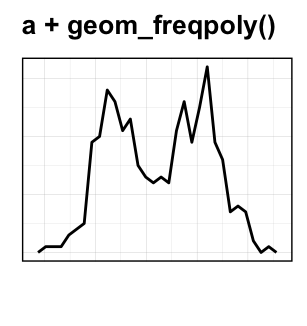
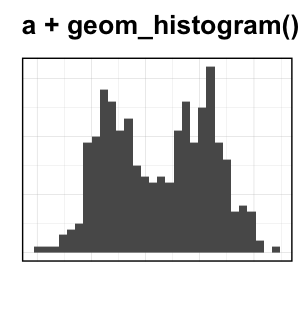
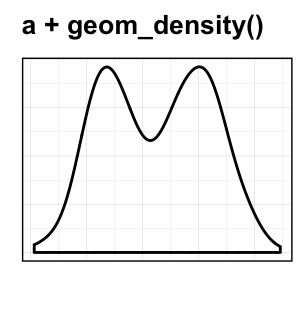
a <- ggplot(wdata, aes(x = weight))

Possible layers include:

* **geom\_density()**: density plot
* **geom\_histogram()**: histogram plot
* **geom\_freqpoly()**: frequency polygon
* **geom\_area()**: area plot
* **geom\_dotplot()**: dot plot
* **stat\_ecdf()**: empirical cumulative density function
* **stat\_qq()**: quantile - quantile plot

Key arguments to customize the plots:

* color, size, linetype: change the line color, size and type, respectively
* fill: change the areas fill color (for bar plots, histograms and density plots)
* alpha: create a semi-transparent color.



**References**

Wilke, Claus O. 2017. *Ggridges: Ridgeline Plots in ’Ggplot2’*. [https://CRAN.R-project.org/package=ggridges](https://cran.r-project.org/package=ggridges).

     0 Note

Enjoyed this article? Give us 5 stars    (just above this text block)! Reader needs to be STHDA member for voting. I’d be very grateful if you’d help it spread by emailing it to a friend, or sharing it on Twitter, Facebook or Linked In.  
  
Show me some love with the like buttons below... Thank you and please don't forget to share and comment below!!