

-

Tutorial

R Tutorial (R-Tutorial.html)

ggplot2

ggplot2 Short Tutorial (ggplot2-Tutorial-With-R.html)

ggplot2 Tutorial 1 - Intro (Complete-Ggplot2-Tutorial-Part1-With-R-Code.html)

ggplot2 Tutorial 2 - Theme (Complete-Ggplot2-Tutorial-Part2-Customizing-Theme-With-R-Code.html)

ggplot2 Tutorial 3 - Masterlist (Top50-Ggplot2-Visualizations-MasterList-R-Code.html)

ggplot2 Quickref (ggplot2-cheatsheet.html)

Foundations

Linear Regression (Linear-Regression.html)

Statistical Tests (Statistical-Tests-in-R.html)

Missing Value Treatment (Missing-Value-Treatment-With-R.html)

Outlier Analysis (Outlier-Treatment-With-R.html)

Feature Selection (Variable-Selection-and-Importance-With-R.html)

Model Selection (Model-Selection-in-R.html)

Logistic Regression (Logistic-Regression-With-R.html)

Advanced Linear Regression (Environments.html)

Advanced Regression Models

Advanced Regression Models (adv-regression-models.html)

Time Series

Time Series Analysis (Time-Series-Analysis-With-R.html)

Time Series Forecasting (Time-Series-Forecasting-With-R.html)

More Time Series Forecasting ([Time-Series-Forecasting-With-R-part2.html](#))

High Performance Computing

Parallel computing ([Parallel-Computing-With-R.html](#))

Strategies to Speedup R code ([Strategies-To-Improve-And-Speedup-R-Code.html](#))

Useful Techniques

Association Mining ([Association-Mining-With-R.html](#))

Multi Dimensional Scaling ([Multi-Dimensional-Scaling-With-R.html](#))

Optimization ([Profiling.html](#))

InformationValue package ([Information-Value-With-R.html](#))

Stay up-to-date. Subscribe!

(https://docs.google.com/forms/d/1xkMYkLNU9U39Dd8S_2JC0p8B5t6_Yq6zUQjanQQJpY/viewform)

Chat! (<https://docs.google.com/forms/d/13GrkCFcNa-T0IIIQghsz2SIEbc-YqY9eJX02B19l5Ow/viewform>)

Contents

Basic tasks

Legend

Plot text and annotation

Multiple plots

Geom layers

ggplot2 Quickref

Basics tasks

- Basic plot setup (ggplot2-cheatsheet.html#Basic%20plot%20setup)
- Scatterplot (ggplot2-cheatsheet.html#Scatterplot)
- Static - point size, shape, color and boundary thickness (ggplot2-cheatsheet.html#Static%20-%20point%20size,%20shape,%20color%20and%20boundary%20thickness)
- Dynamic - point size, shape, color and boundary thickness (ggplot2-cheatsheet.html#Dynamic%20-%20point%20size,%20shape,%20color%20and%20boundary%20thickness)
- Add title, X and Y axis labels (ggplot2-cheatsheet.html#Add%20Title,%20X%20and%20Y%20axis%20labels)
- Change color of all text (ggplot2-cheatsheet.html#Change%20color%20of%20all%20text)
- Change title, X and Y axis label size (ggplot2-cheatsheet.html#Change%20title,%20X%20and%20Y%20axis%20label%20size)

- Change title face, color, line height (ggplot2-cheatsheet.html#Change%20title%20face,%20color,%20line%20height)
- Change point color (ggplot2-cheatsheet.html#Change%20point%20color)
- Adjust X and Y axis limits (ggplot2-cheatsheet.html#Adjust%20X%20and%20Y%20axis%20limits)
- Change X and Y axis labels (ggplot2-cheatsheet.html#Change%20X%20and%20Y%20axis%20labels)
- Rotate axis text (ggplot2-cheatsheet.html#Rotate%20axis%20text)
- Flip X and Y axis (ggplot2-cheatsheet.html#Flip%20X%20and%20Y%20Axis)
- Grid lines and panel background (ggplot2-cheatsheet.html#Grid%20lines%20and%20panel%20background)
- Plot margin and background (ggplot2-cheatsheet.html#Plot%20margin%20and%20background)
- Colors (ggplot2-cheatsheet.html#Colors)

Legend

- Hide legend (ggplot2-cheatsheet.html#Hide%20legend)
- Change legend title (ggplot2-cheatsheet.html#Change%20legend%20title)
- Change legend and point color (ggplot2-cheatsheet.html#Change%20legend%20and%20point%20color)
- Change legend position (ggplot2-cheatsheet.html#Change%20legend%20position)
- Change order of legend items (ggplot2-cheatsheet.html#Change%20order%20of%20legend%20items)
- Change legend title, text, box, symbol (ggplot2-cheatsheet.html#Legend%20title,%20text,%20box,%20symbol)

Plot text and annotation

- Add text in chart (ggplot2-cheatsheet.html#Add%20text%20in%20chart)
- Annotation (ggplot2-cheatsheet.html#Annotation)

Multiple plots

- Multiple chart panels (ggplot2-cheatsheet.html#Multiple%20chart%20panels)
- Free X and Y axis scales (ggplot2-cheatsheet.html#Free%20X%20and%20Y%20axis%20scales)
- Arrange multiple plots (ggplot2-cheatsheet.html#Arrange%20multiple%20plots)

Geom layers

- Add smoothing line (ggplot2-cheatsheet.html#Add%20smoothing%20line)
- Add horizontal / vertical line (ggplot2-cheatsheet.html#Add%20horizontal%20/%20vertical%20line)
- Add bar chart (ggplot2-cheatsheet.html#Add%20bar%20chart)
- Distinct color for bars (ggplot2-cheatsheet.html#Distinct%20color%20for%20bars)
- Change color and width of bars (ggplot2-cheatsheet.html#Change%20color%20and%20width%20of%20bars)
- Change color palette (ggplot2-cheatsheet.html#Change%20color%20palette)
- Line chart (ggplot2-cheatsheet.html#Line%20chart)
- Line chart from timeseries (ggplot2-cheatsheet.html#Line%20chart%20from%20timeseries)
- Ribbons (ggplot2-cheatsheet.html#Ribbons)
- Area (ggplot2-cheatsheet.html#Area)
- Boxplot and Violin (ggplot2-cheatsheet.html#Boxplot%20and%20Violin)
- Density (ggplot2-cheatsheet.html#Density)
- Tiles (ggplot2-cheatsheet.html#Tiles)

Not finding what you were looking for? Let me know! (<https://docs.google.com/forms/d/13GrkCFcNa-TOIIIQghsz2SIEbc-YqY9eJX02B19l5Ow/viewform>)

Basic tasks

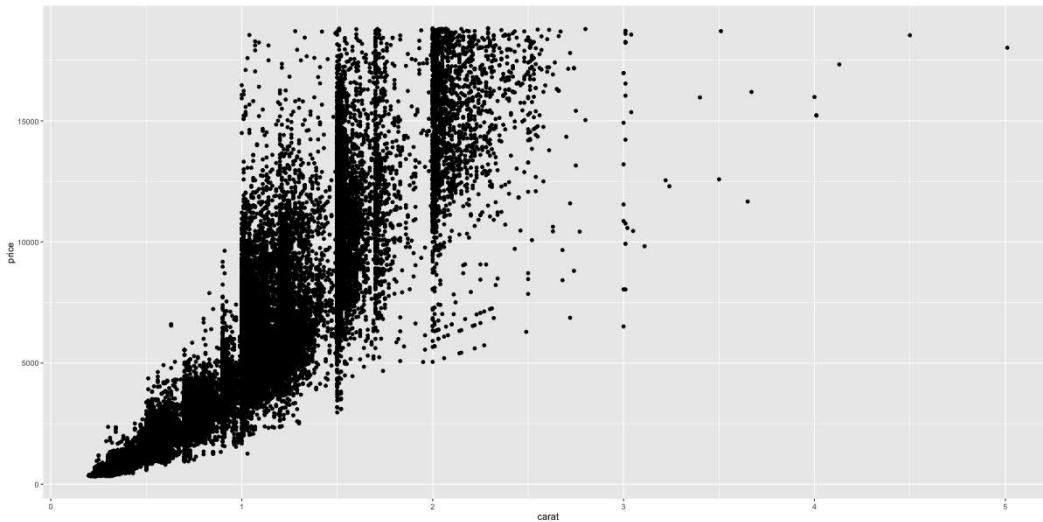
Basic plot setup

```
gg <- ggplot(df, aes(x=xcol, y=ycol))
```

df must be a dataframe that contains all information to make the ggplot. Plot will show up only after adding the geom layers.

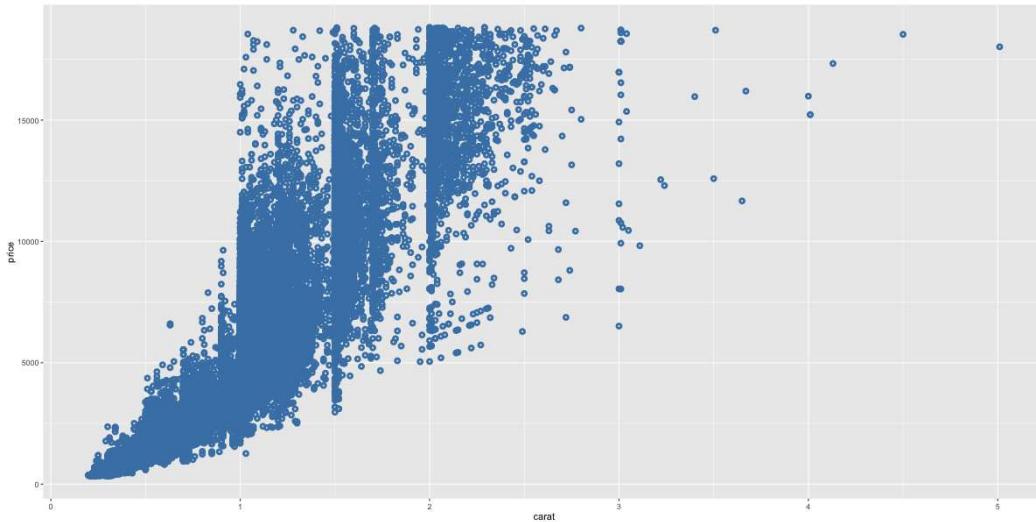
Scatterplot

```
library(ggplot2)
gg <- ggplot(diamonds, aes(x=carat, y=price))
gg + geom_point()
```



Static - point size, shape, color and boundary thickness

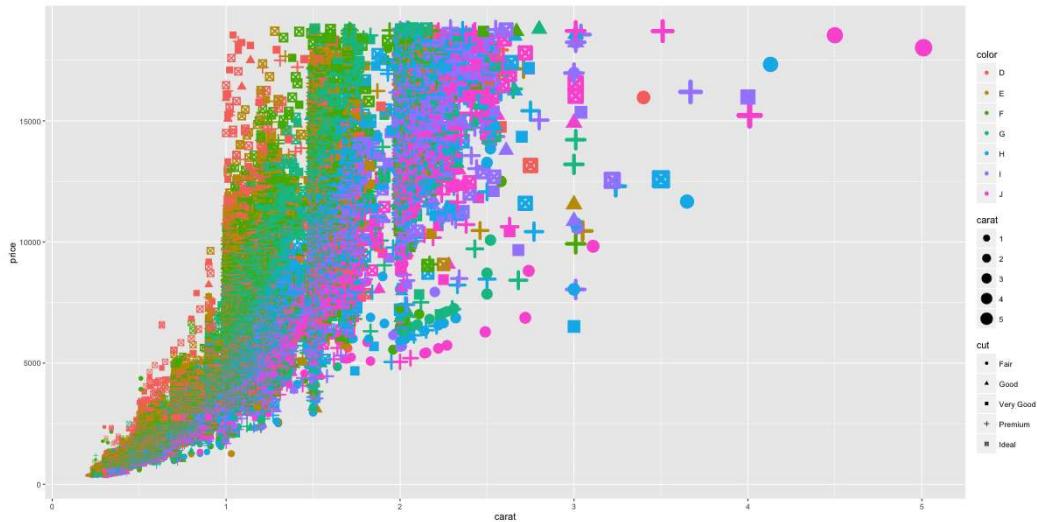
```
gg + geom_point(size=1, shape=1, color="steelblue", stroke=2) # 'stroke' controls the thickness of point boundary
```



Dynamic - point size, shape, color and boundary thickness

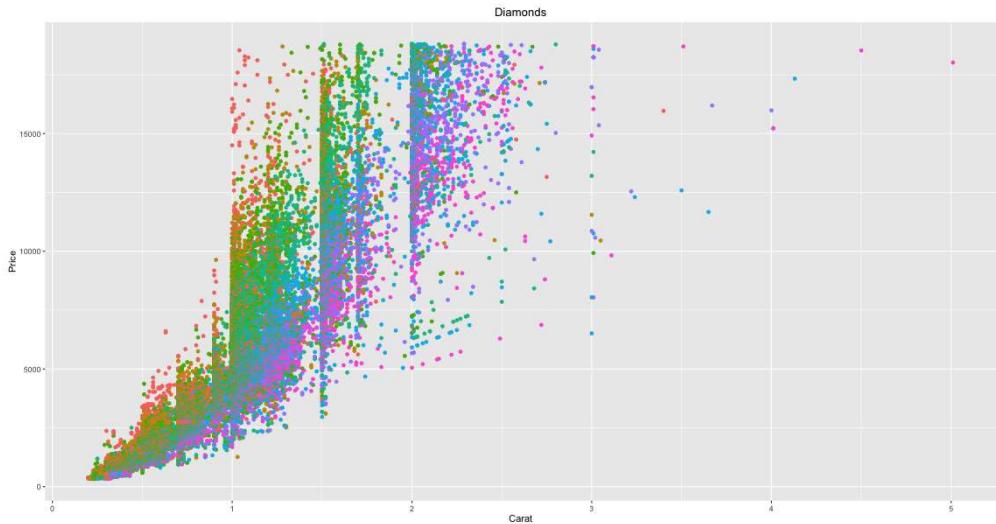
Make the aesthetics vary based on a variable in df.

```
gg + geom_point(aes(size=carat, shape=cut, color=color, stroke=carat)) # carat, cut and
color are variables in 'diamonds'
```



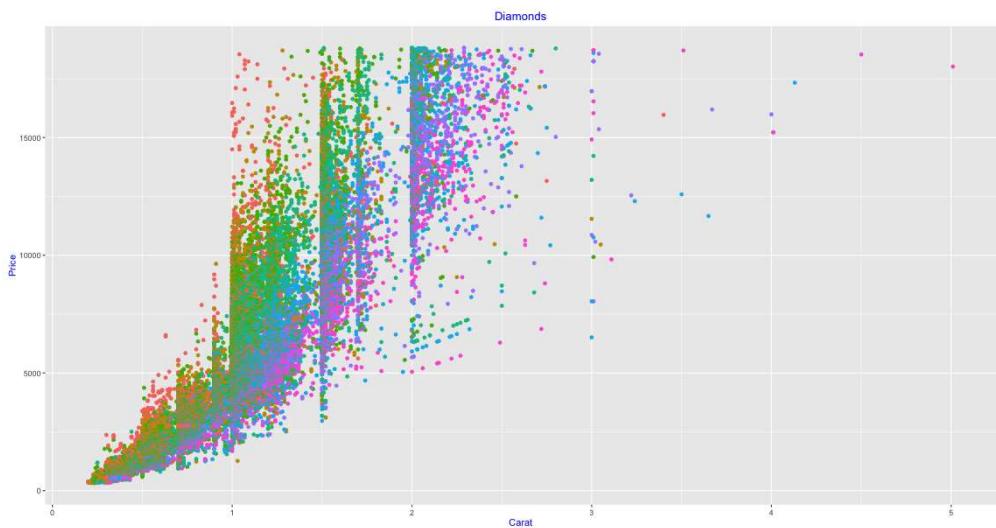
Add Title, X and Y axis labels

```
gg1 <- gg + geom_point(aes(color=color))
gg2 <- gg1 + labs(title="Diamonds", x="Carat", y="Price") # ggttitle("title") also changes the title.
print(gg2)
```



Change color of all text

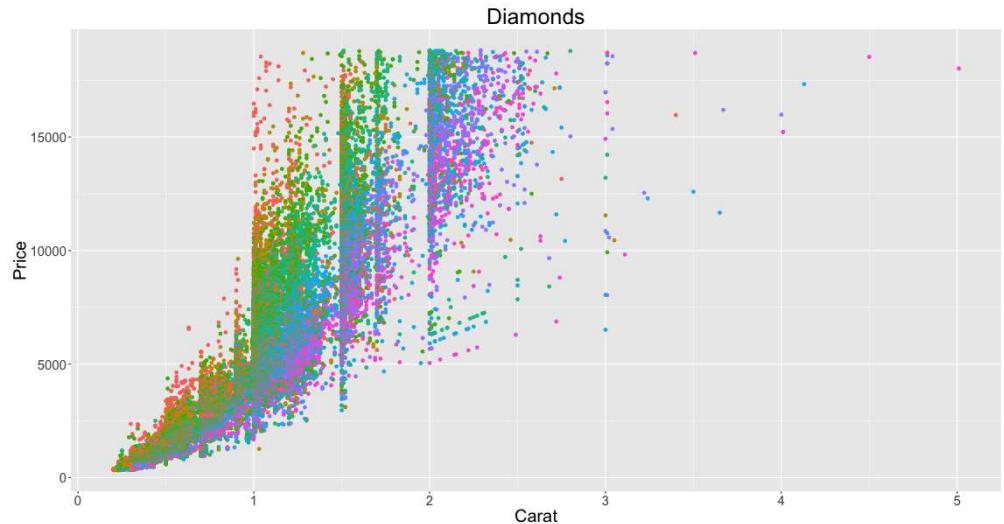
```
gg2 + theme(text=element_text(color="blue")) # all text turns blue.
```



Change title, X and Y axis label and text size

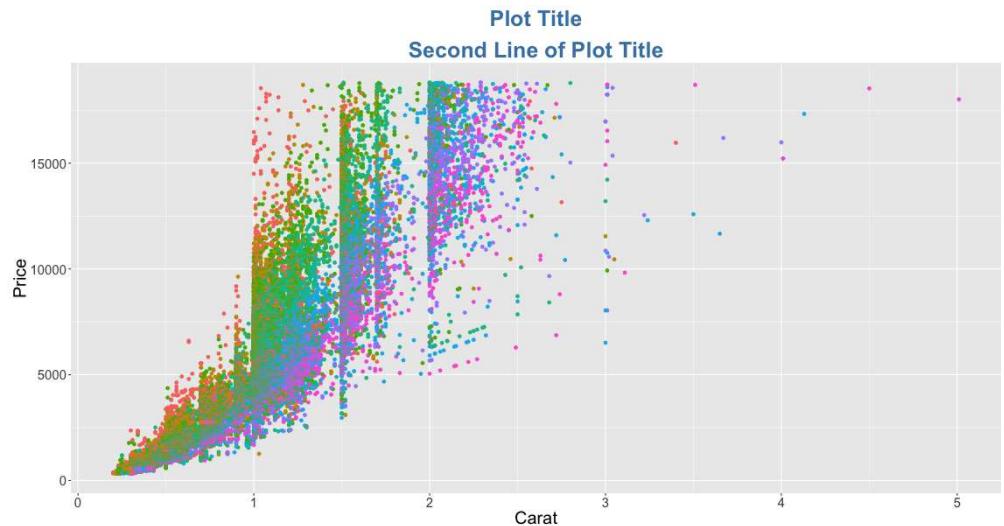
`plot.title`: Controls plot title. `axis.title.x`: Controls X axis title `axis.title.y`: Controls Y axis title
`axis.text.x`: Controls X axis text `axis.text.y`: Controls y axis text

```
gg3 <- gg2 + theme(plot.title=element_text(size=25), axis.title.x=element_text(size=20),
axis.title.y=element_text(size=20), axis.text.x=element_text(size=15), axis.text.y=element_text(size=15))
print(gg3)
```



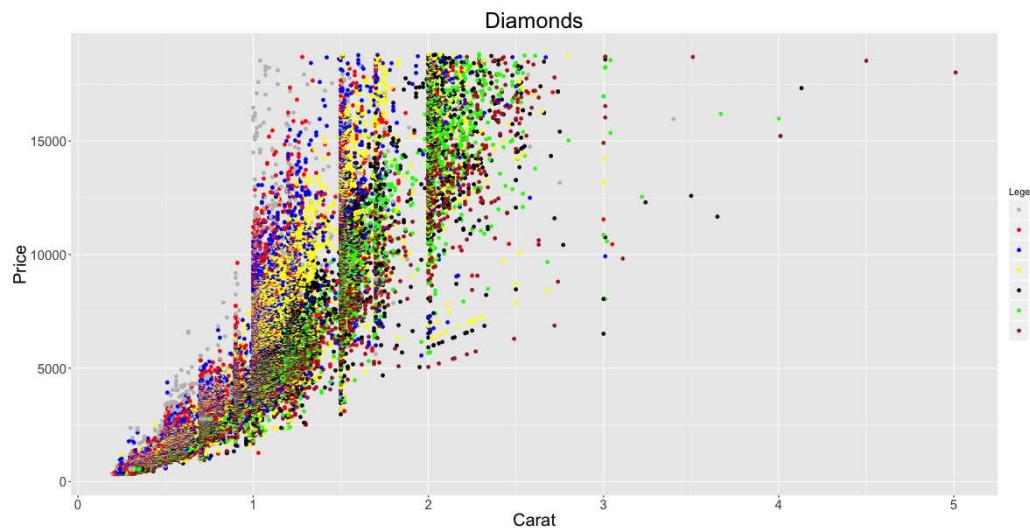
Change title face, color, line height

```
gg3 + labs(title="Plot Title\nSecond Line of Plot Title") + theme(plot.title=element_text(face="bold", color="steelblue", lineheight=1.2))
```



Change point color

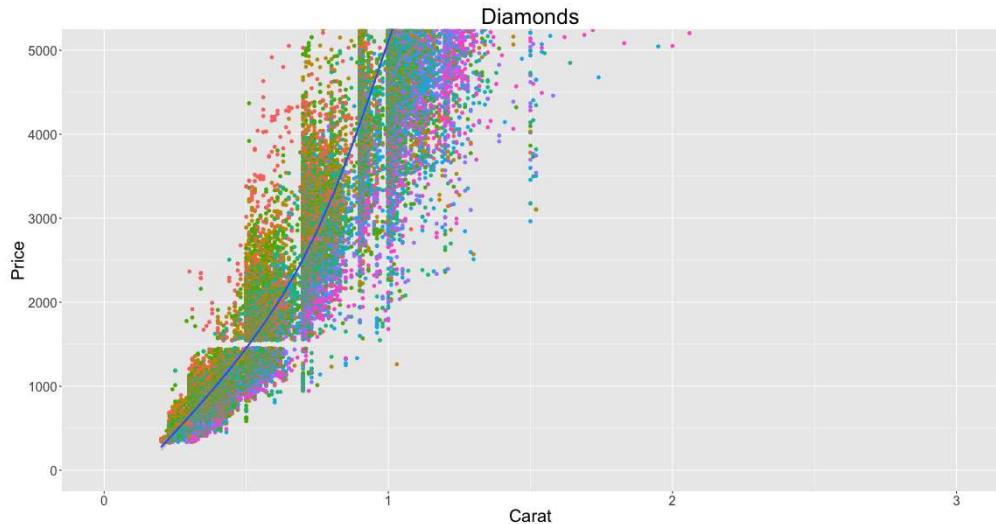
```
gg3 + scale_colour_manual(name='Legend', values=c('D'='grey', 'E'='red', 'F'='blue', 'G'='yellow', 'H'='black', 'I'='green', 'J'='firebrick'))
```



Adjust X and Y axis limits

Method 1: Zoom in

```
gg3 + coord_cartesian(xlim=c(0,3), ylim=c(0, 5000)) + geom_smooth() # zoom in
```

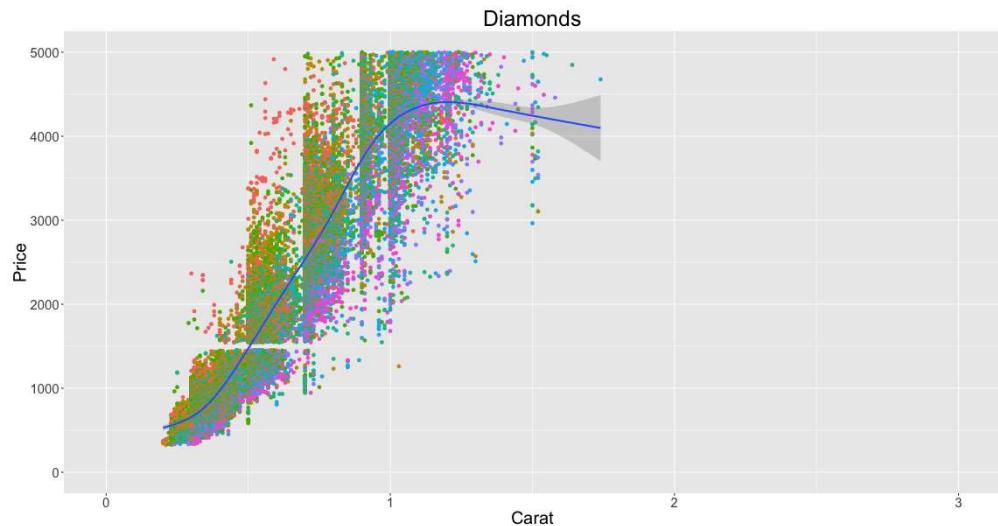


Method 2: Deletes the points outside limits

```
gg3 + xlim(c(0,3)) + ylim(c(0, 5000)) + geom_smooth() # deletes the points
#> Warning messages:
#> 1: Removed 14714 rows containing non-finite values (stat_smooth).
#> 2: Removed 14714 rows containing missing values (geom_point).
```

Method 3: Deletes the points outside limits

```
gg3 + scale_x_continuous(limits=c(0,3)) + scale_y_continuous(limits=c(0, 5000)) + geom_smooth() # deletes the points outside limits
#> Warning message:
#> Removed 14714 rows containing missing values (geom_point).
```

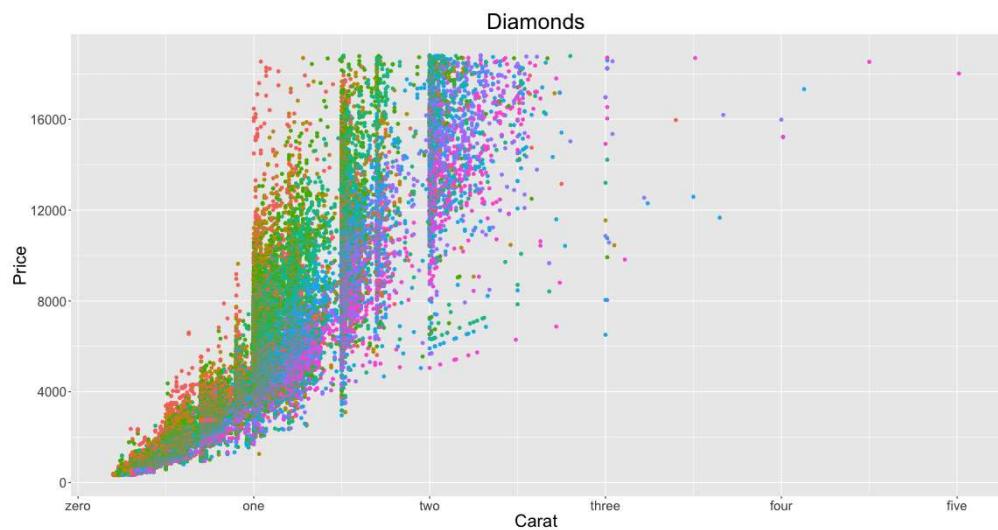


Notice the change in smoothing line because of deleted points. This could sometimes be misleading in your analysis.

Change X and Y axis labels

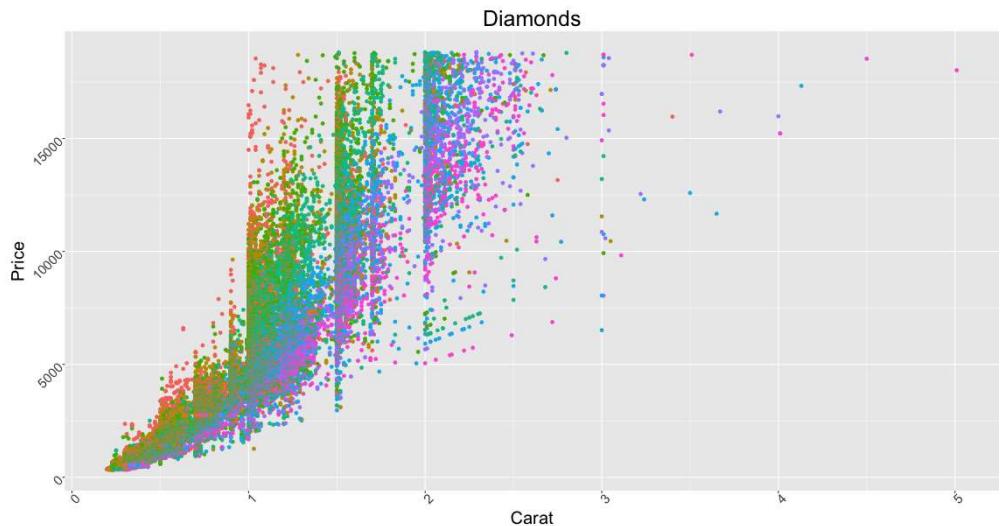
```
gg3 + scale_x_continuous(labels=c("zero", "one", "two", "three", "four", "five")) + scale_y_continuous(breaks=seq(0, 20000, 4000)) # if Y is continuous, if X is a factor
```

Use `scale_x_discrete` instead, if X variable is a factor.



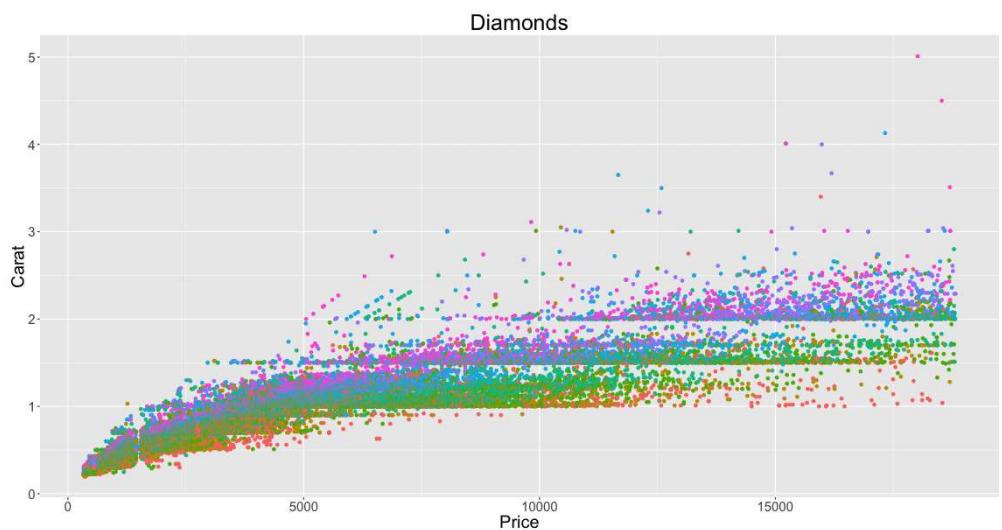
Rotate axis text

```
gg3 + theme(axis.text.x=element_text(angle=45), axis.text.y=element_text(angle=45))
```



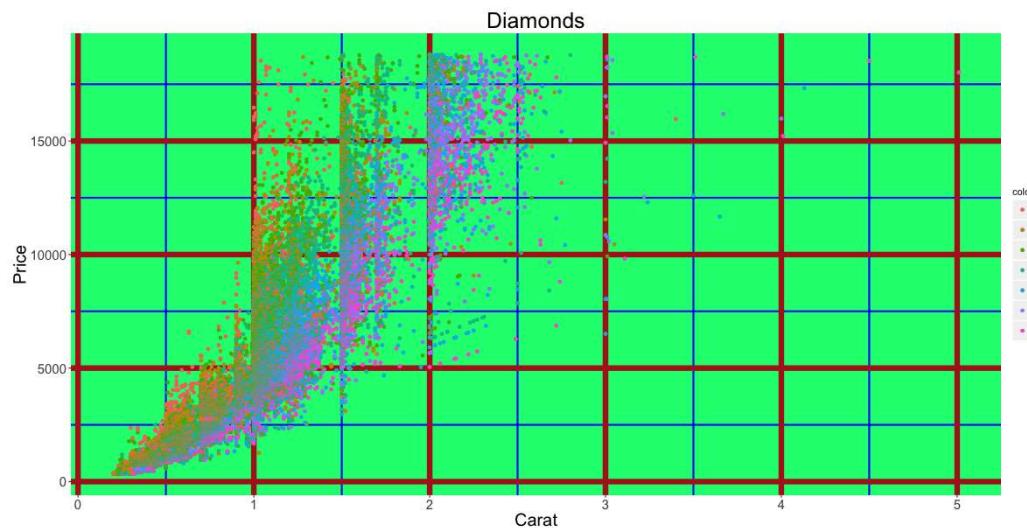
Flip X and Y Axis

```
gg3 + coord_flip() # flips X and Y axis.
```



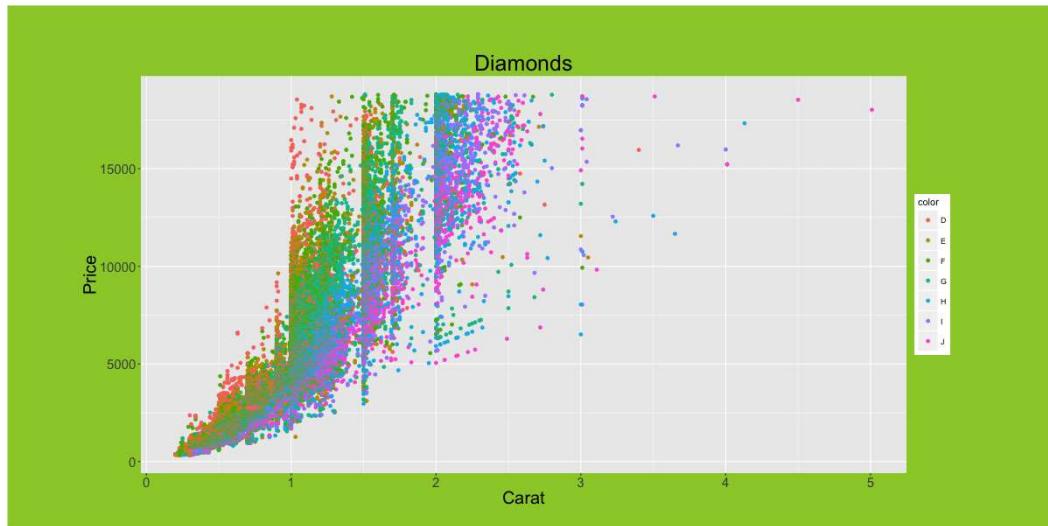
Grid lines and panel background

```
gg3 + theme(panel.background = element_rect(fill = 'springgreen'),
  panel.grid.major = element_line(colour = "firebrick", size=3),
  panel.grid.minor = element_line(colour = "blue", size=1))
```



Plot margin and background

```
gg3 + theme(plot.background=element_rect(fill="yellowgreen"), plot.margin = unit(c(2, 4, 1, 3), "cm")) # top, right, bottom, left
```



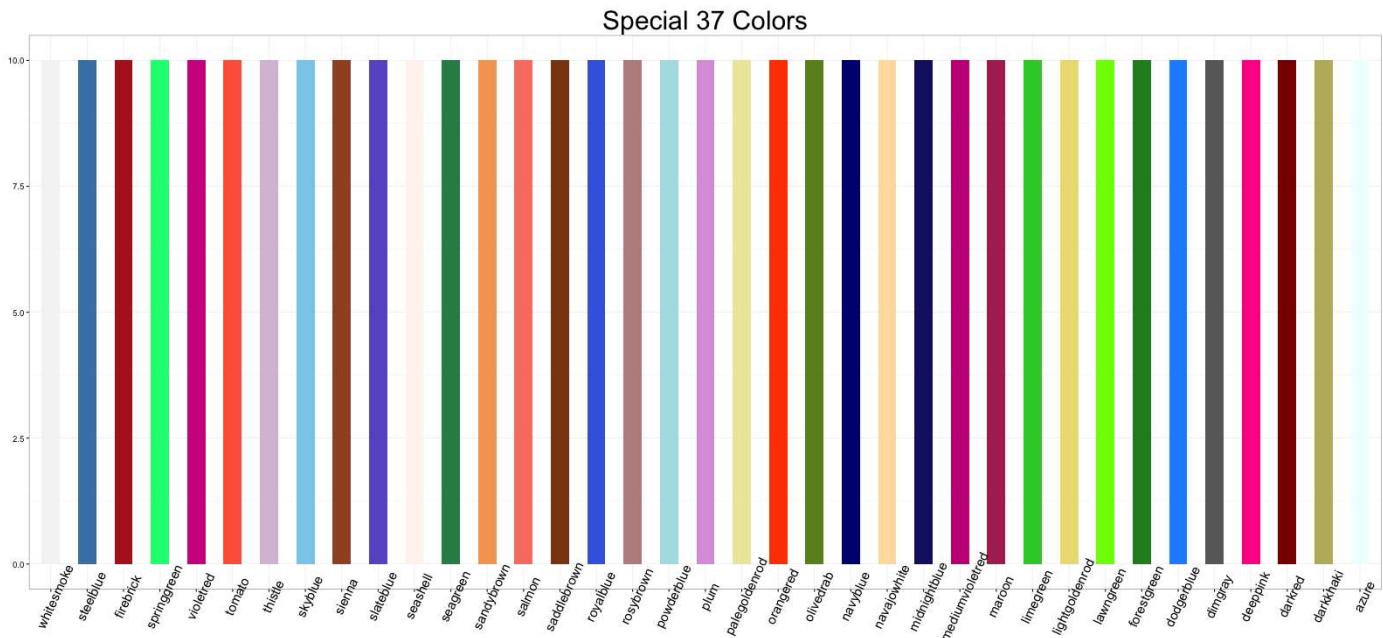
Colors

The whole list of colors are displayed at your R console in the `color()` function. Here are few of my suggestions for nice looking colors and backgrounds:

- steelblue (points and lines)
- firebrick (point and lines)
- springgreen (fills)
- violetred (fills)
- tomato (fills)
- skyblue (bg)
- sienna (points, lines)

- slateblue (fills)
- seagreen (points, lines, fills)
- sandybrown (fills)
- salmon (fills)
- saddlebrown (lines)
- royalblue (fills)
- orangered (point, lines, fills)
- olivedrab (points, lines, fills)
- midnightblue (lines)
- mediumvioletred (points, lines, fills)
- maroon (points, lines, fills)
- limegreen (fills)
- lawngreen (fills)
- forestgreen (lines, fills)
- dodgerblue (fills, bg)
- dimgray (grids, secondary bg)
- deeppink (fills)
- darkred (lines, points)

If you are looking for consistent colors, the RColorBrewer package has predefined color palettes
(<http://127.0.0.1:4000/ggplot2-cheatsheet.html#Change%20color%20palette>)



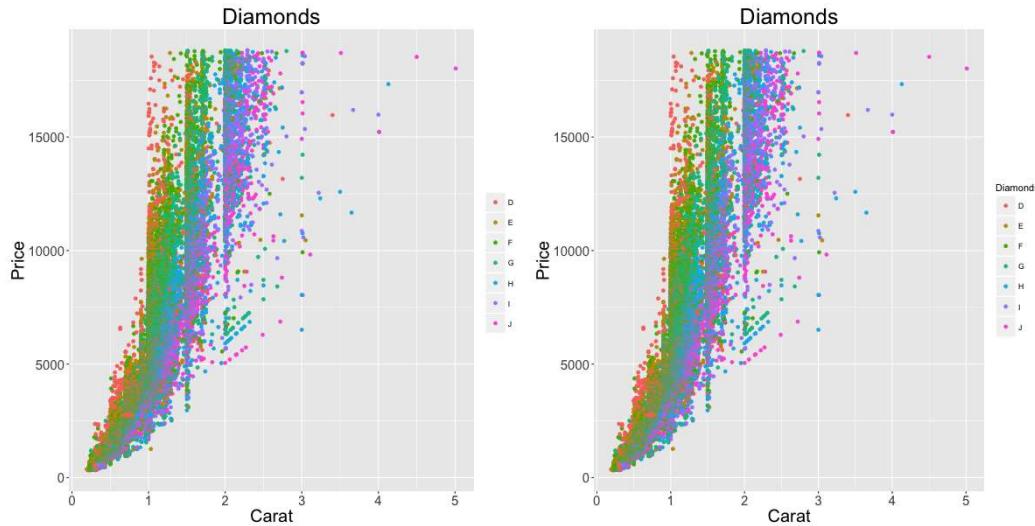
Legend

Hide legend

```
gg3 + theme(legend.position="none") # hides the legend
```

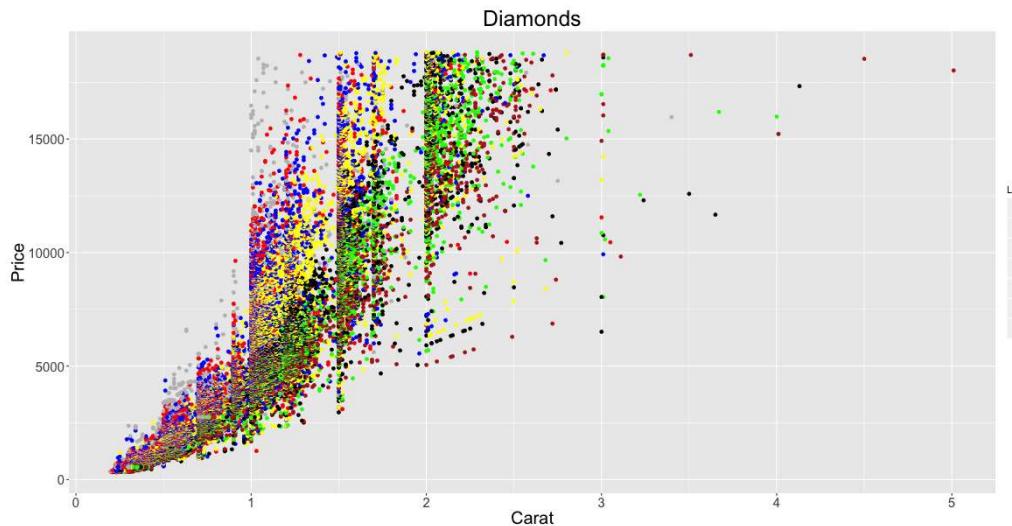
Change legend title

```
gg3 + scale_color_discrete(name "") # Remove legend title (method1)
p1 <- gg3 + theme(legend.title=element_blank()) # Remove legend title (method)
p2 <- gg3 + scale_color_discrete(name="Diamonds") # Change legend title
library(gridExtra)
grid.arrange(p1, p2, ncol=2) # arrange
```



Change legend and point color

```
gg3 + scale_colour_manual(name='Legend', values=c('D'='grey', 'E'='red', 'F'='blue', 'G'='yellow', 'H'='black', 'I'='green', 'J'='firebrick'))
```



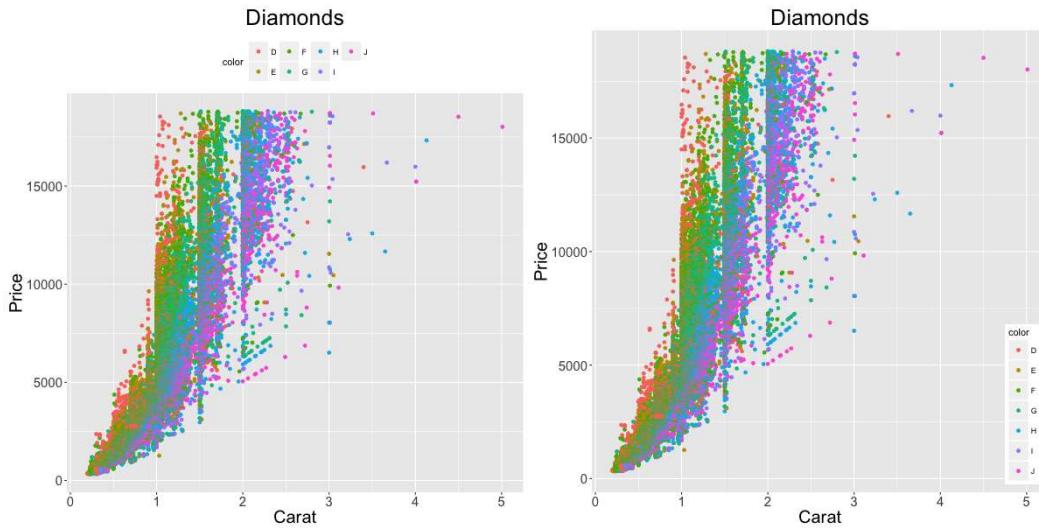
Change legend position

Outside plot

```
p1 <- gg3 + theme(legend.position="top") # top / bottom / left / right
```

Inside plot

```
p2 <- gg3 + theme(legend.justification=c(1,0), legend.position=c(1,0)) # legend justification is the anchor point on the legend, considering the bottom left of legend as (0,0)
gridExtra::grid.arrange(p1, p2, ncol=2)
```



Change order of legend items

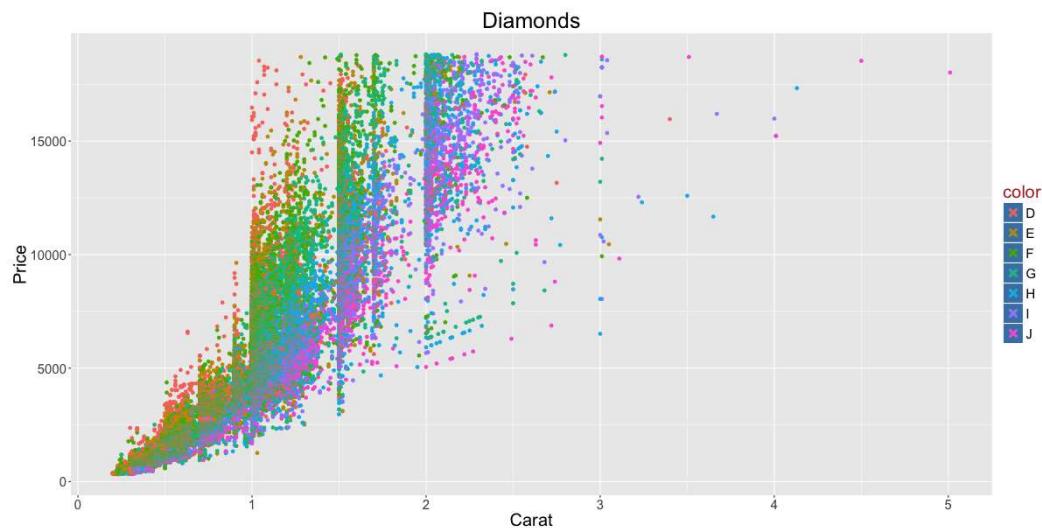
```
df$newLegendColumn <- factor(df$legendcolumn, levels=c(new_order_of_legend_items), ordered = TRUE)
```

Create a new factor variable used in the legend, ordered as you need. Then use this variable instead in the plot.

Legend title, text, box, symbol

- `legend.title` - Change legend title
- `legend.text` - Change legend text
- `legend.key` - Change legend box
- `guides` - Change legend symbols

```
gg3 + theme(legend.title = element_text(size=20, color = "firebrick"), legend.text = element_text(size=15), legend.key=element_rect(fill='steelblue')) + guides(colour = guide_legend(override.aes = list(size=2, shape=4, stroke=2))) # legend title color and size, box color, symbol color, size and shape.
```

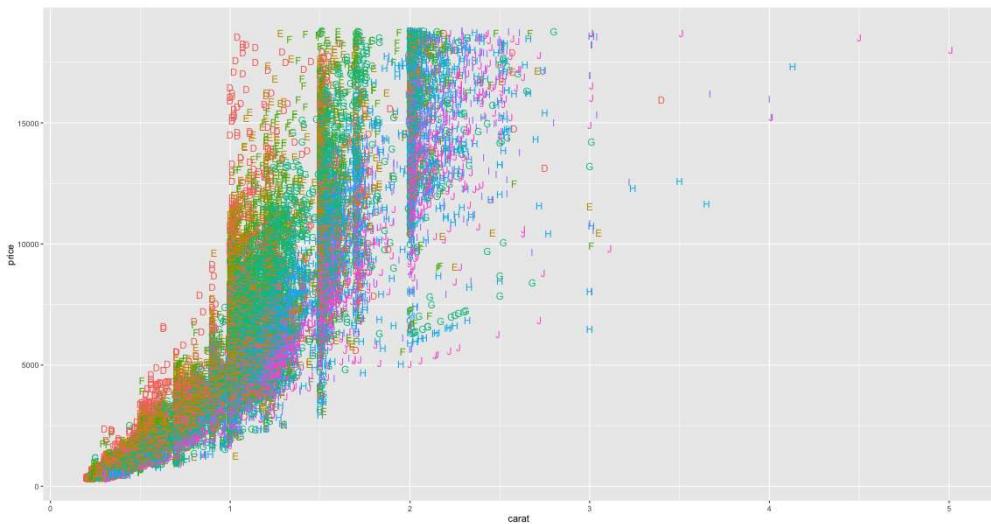


0: □	10: ⊕	20: ●	A: A
1: ○	11: ⊖	21: ●	a: a
2: △	12: ▨	22: ■	B: B
3: +	13: ⊗	23: ♦	b: b
4: ✕	14: □	24: ▲	S: S
5: ◊	15: ■	25: ▼	*: *
6: ▽	16: ●	@: @	: :
7: ☐	17: ▲	+: +	-: -
8: ✩	18: ♦	%: %	?: ?
9: ✧	19: ●	#: #	*: *

Plot text and annotation

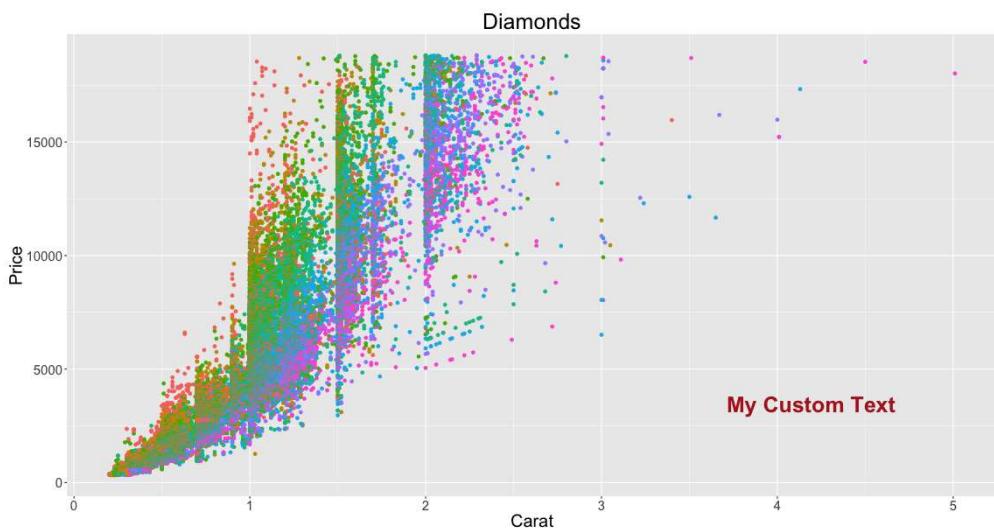
Add text in chart

```
#> Not Run: gg + geom_text(aes(xcol, ycol, label=round(labelCol), size=3)) # general format
gg + geom_text(aes(label=color, color=color), size=4)
```



Annotation

```
#> gg3 + annotate("mytext", x=xpos, y=ypos, label="My text") # Not run: General Format
library(grid)
my_grob = grobTree(textGrob("My Custom Text", x=0.8, y=0.2, gp=gpar(col="firebrick", fontsize=25, fontface="bold")))
gg3 + annotation_custom(my_grob)
```



Multiple plots

Multiple chart panels

```
p1 <- gg1 + facet_grid(color ~ cut) # arrange in a grid. More space for plots.
```

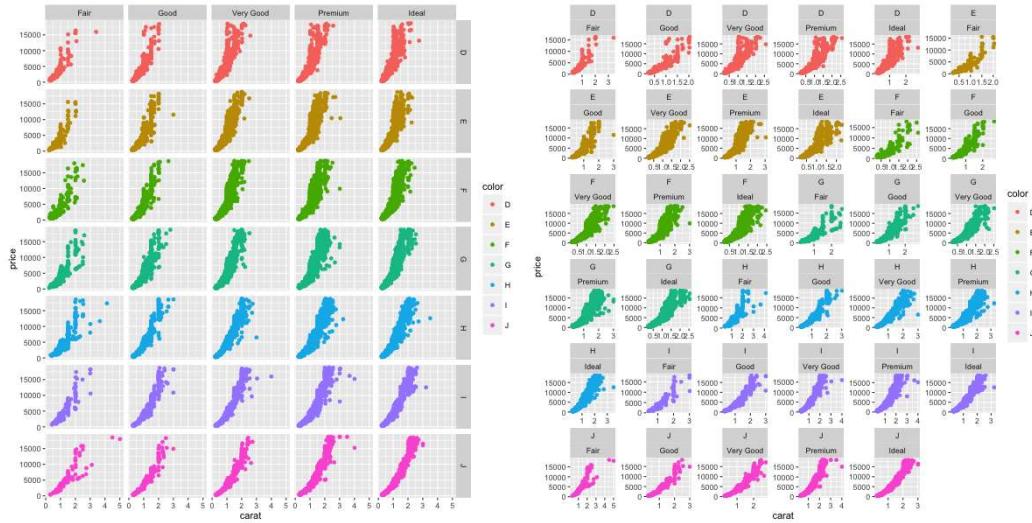
Free X and Y axis scales

By setting `scales='free'`, the scales of both X and Y axis is freed. Use `scales='free_x'` to free only X-axis and `scales='free_y'` to free only Y-axis.

```
p2 <- gg1 + facet_wrap(color ~ cut, scales="free") # free the x and y axis scales.
```

Arrange multiple plots

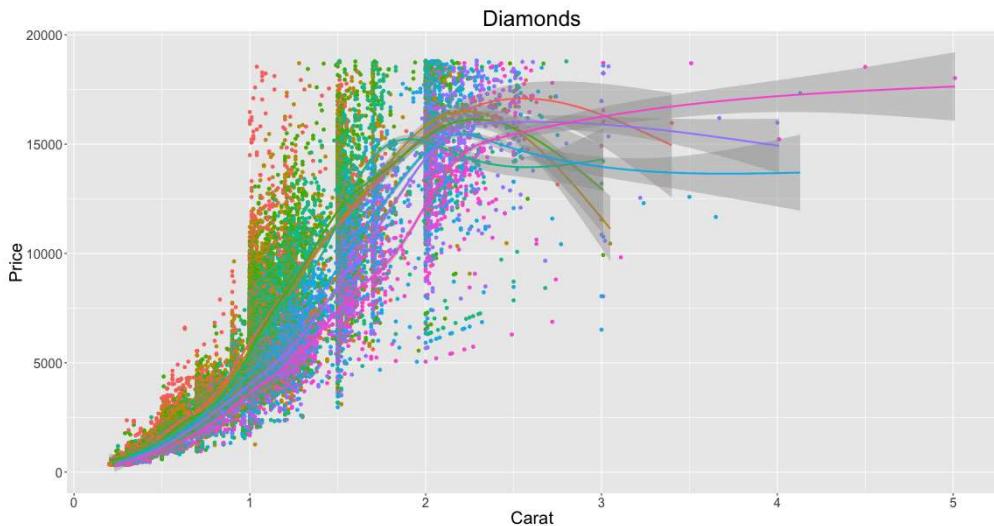
```
library(gridExtra)
grid.arrange(p1, p2, ncol=2)
```



Geom layers

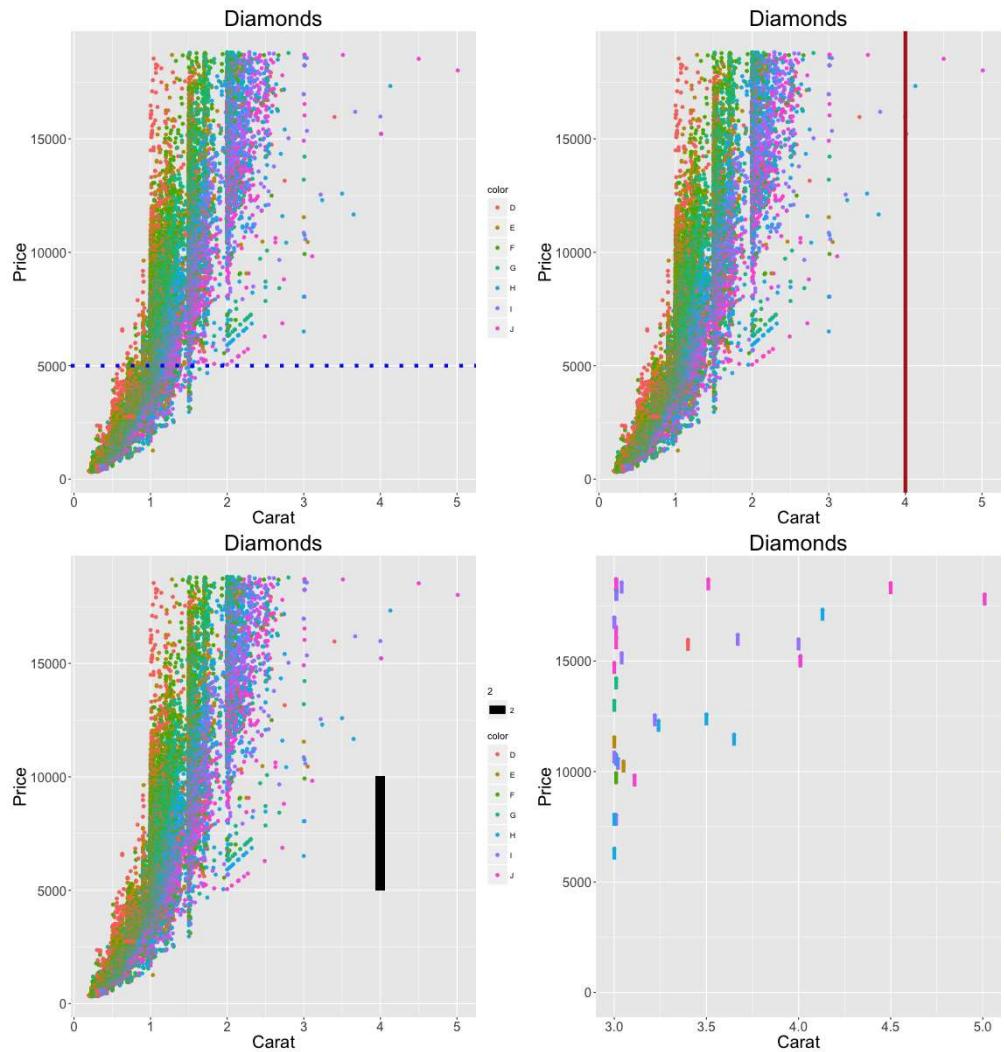
Add smoothing line

```
gg3 + geom_smooth(aes(color=color)) # method could be - 'lm', 'loess', 'gam'
```



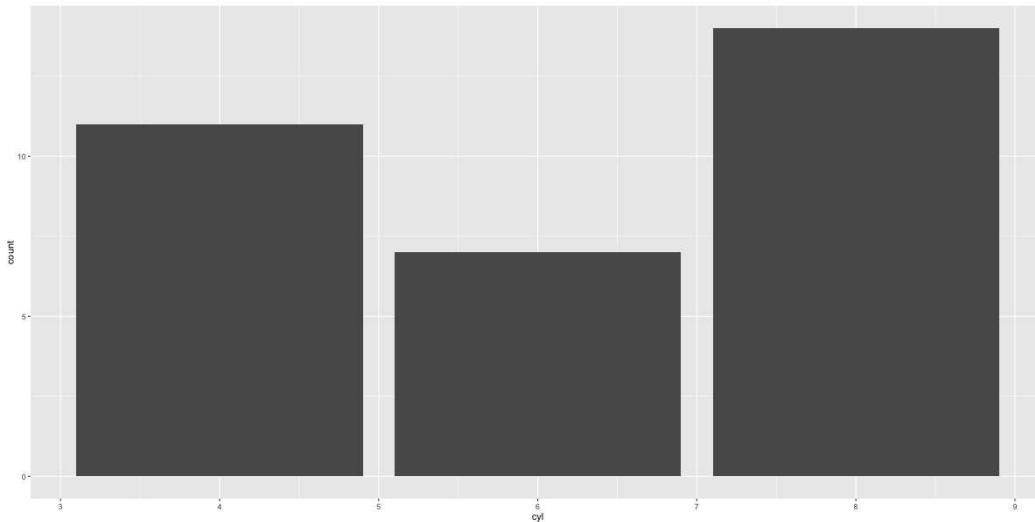
Add horizontal / vertical line

```
p1 <- gg3 + geom_hline(yintercept=5000, size=2, linetype="dotted", color="blue") # line types: solid, dashed, dotted, dotdash, longdash and twodash
p2 <- gg3 + geom_vline(xintercept=4, size=2, color="firebrick")
p3 <- gg3 + geom_segment(aes(x=4, y=5000, xend=4, yend=10000, size=2, lineend="round"))
p4 <- gg3 + geom_segment(aes(x=carat, y=price, xend=carat, yend=price-500, color=color),
size=2) + coord_cartesian(xlim=c(3, 5)) # x, y: start points. xend, yend: end points
gridExtra::grid.arrange(p1,p2,p3,p4, ncol=2)
```

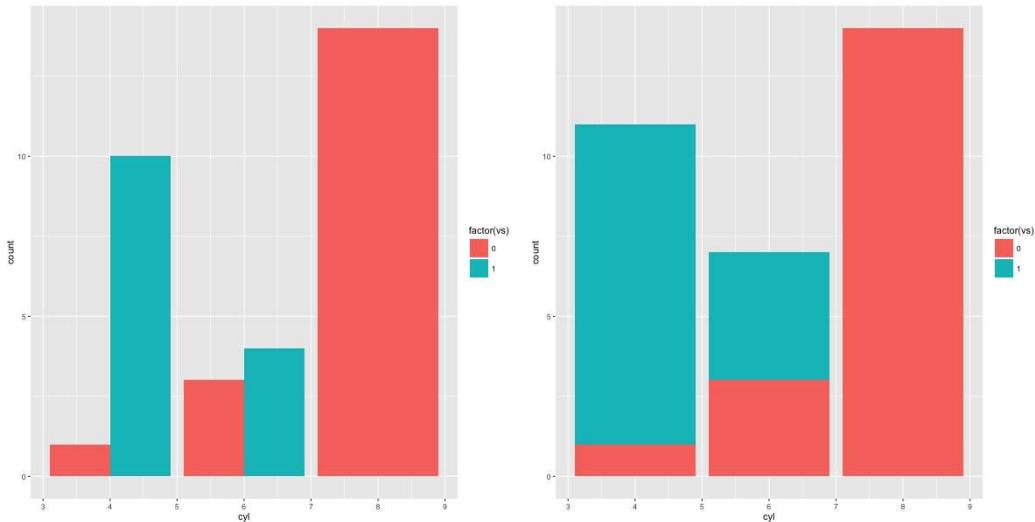


Add bar chart

```
# Frequency bar chart: Specify only X axis.
gg <- ggplot(mtcars, aes(x=cyl))
gg + geom_bar() # frequency table
```

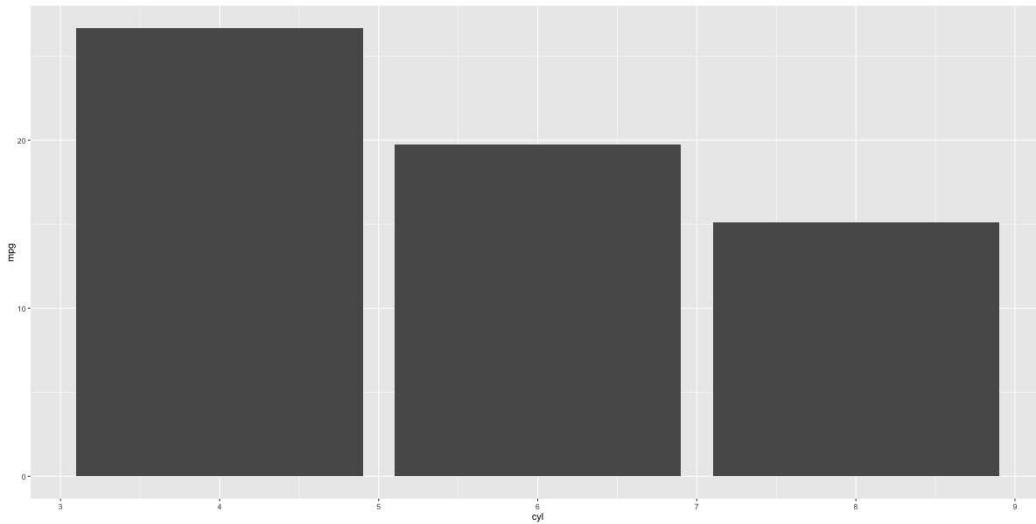


```
gg <- ggplot(mtcars, aes(x=cyl))
p1 <- gg + geom_bar(position="dodge", aes(fill=factor(vs))) # side-by-side
p2 <- gg + geom_bar(aes(fill=factor(vs))) # stacked
gridExtra::grid.arrange(p1, p2, ncol=2)
```



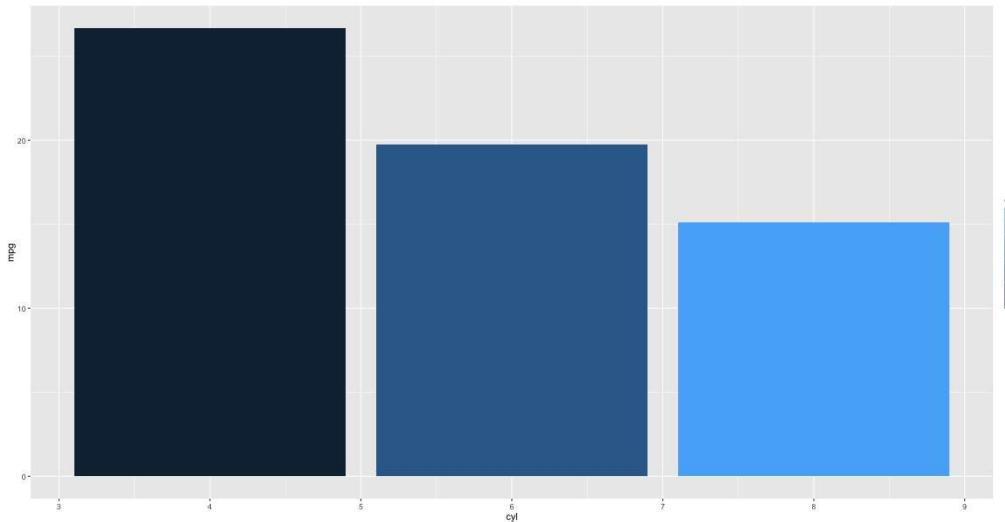
```
# Absolute bar chart: Specify both X adn Y axis. Set stat="identity"
df <- aggregate(mtcars$mpg, by=list(mtcars$cyl), FUN=mean) # mean of mpg for every 'cyl'
names(df) <- c("cyl", "mpg")
head(df)
#>   cyl     mpg
#> 1   4 26.66
#> 2   6 19.74
#> 3   8 15.10

gg_bar <- ggplot(df, aes(x=cyl, y=mpg)) + geom_bar(stat = "identity") # Y axis is explicit. 'stat=identity'
print(gg_bar)
```



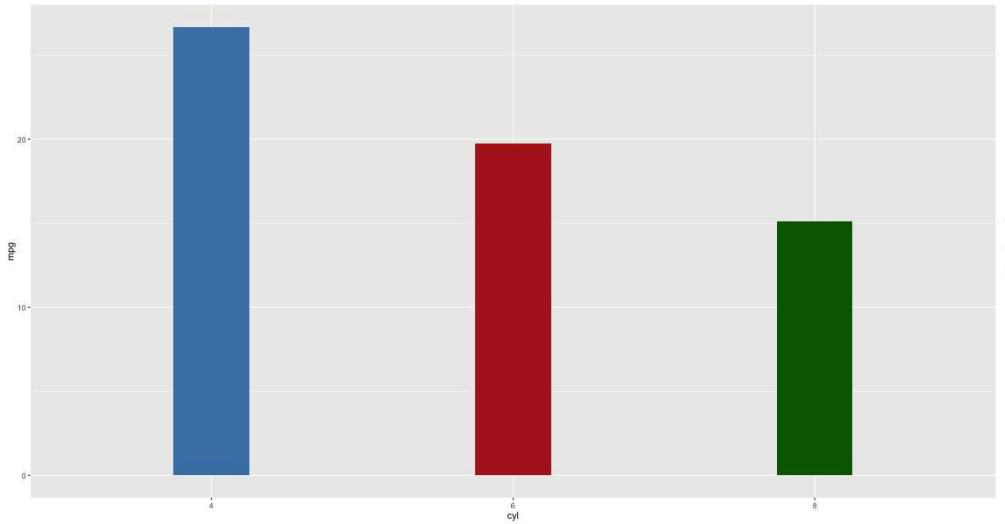
Distinct color for bars

```
gg_bar <- ggplot(df, aes(x=cyl, y=mpg)) + geom_bar(stat = "identity", aes(fill=cyl))
print(gg_bar)
```



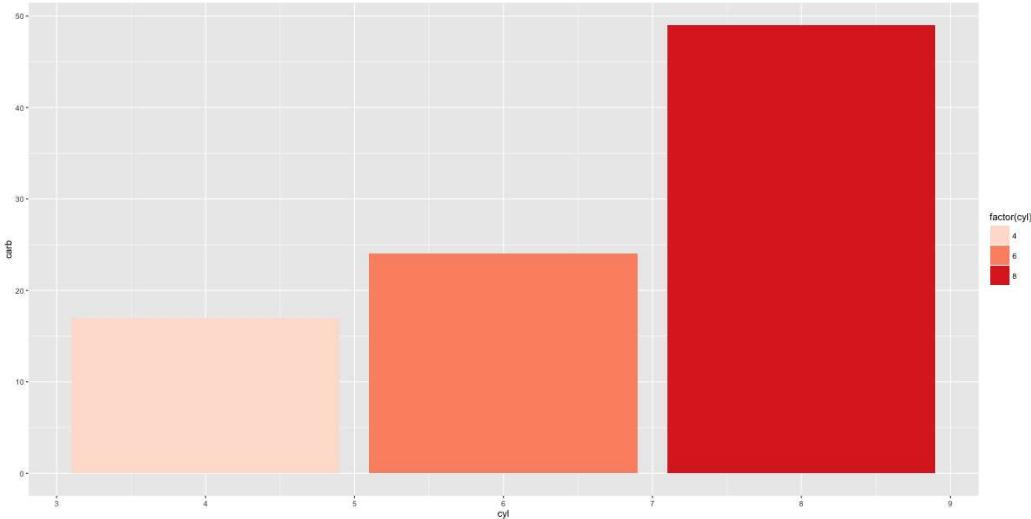
Change color and width of bars

```
df$cyl <- as.factor(df$cyl)
gg_bar <- ggplot(df, aes(x=cyl, y=mpg)) + geom_bar(stat = "identity", aes(fill=cyl), width = 0.25)
gg_bar + scale_fill_manual(values=c("4"="steelblue", "6"="firebrick", "8"="darkgreen"))
```



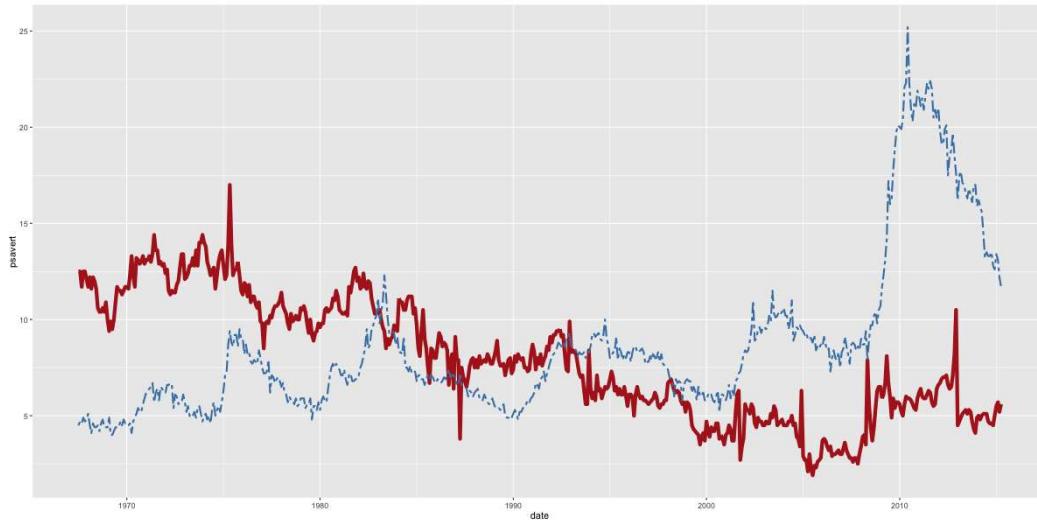
Change color palette

```
library(RColorBrewer)
display.brewer.all(n=20, exact.n=FALSE) # display available color palettes
ggplot(mtcars, aes(x=cyl, y=carb, fill=factor(cyl))) + geom_bar(stat="identity") + scale_fill_brewer(palette="Reds") # "Reds" is palette name
```

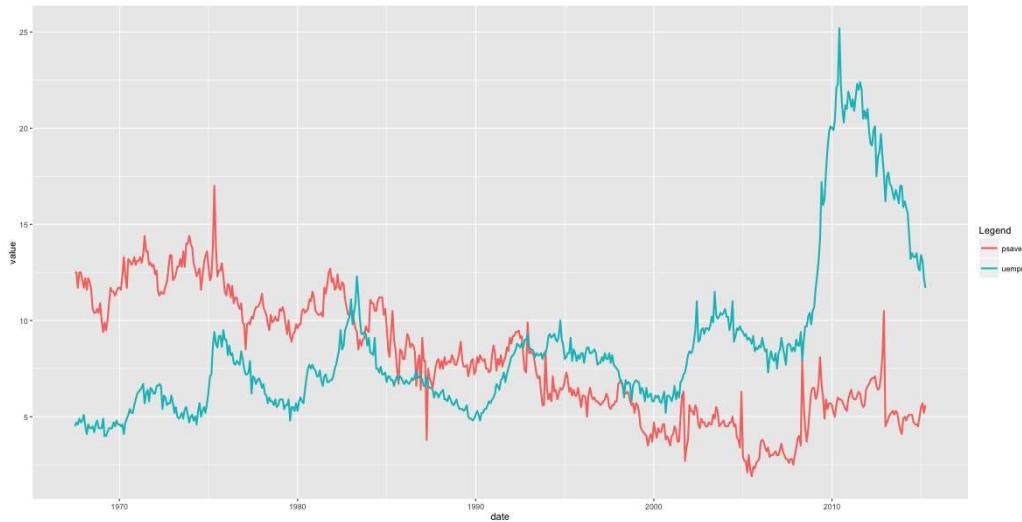


Line chart

```
# Method 1:
gg <- ggplot(economics, aes(x=date)) # setup
gg + geom_line(aes(y=psavert), size=2, color="firebrick") + geom_line(aes(y=uempmed), si
ze=1, color="steelblue", linetype="twodash") # No legend
# available linetypes: solid, dashed, dotted, dotdash, longdash and twodash
```

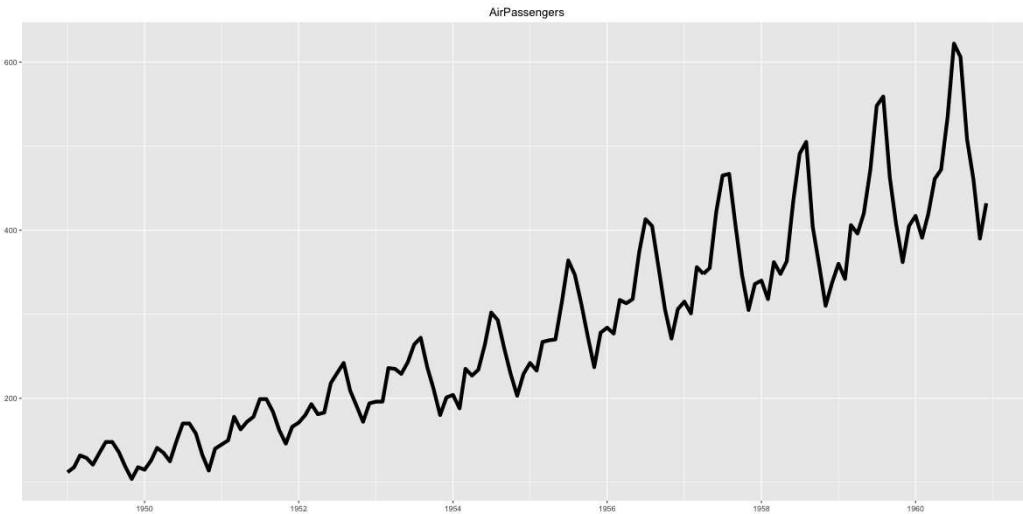


```
# Method 2:
library(reshape2)
df_melt <- melt(economics[, c("date", "psavert", "uempmed")], id="date") # melt by date.
gg <- ggplot(df_melt, aes(x=date)) # setup
gg + geom_line(aes(y=value, color=variable), size=1) + scale_color_discrete(name="Legend") # gets legend.
```



Line chart from timeseries

```
# One step method.
library(ggfortify)
autoplot(AirPassengers, size=2) + labs(title="AirPassengers")
```



Ribbons

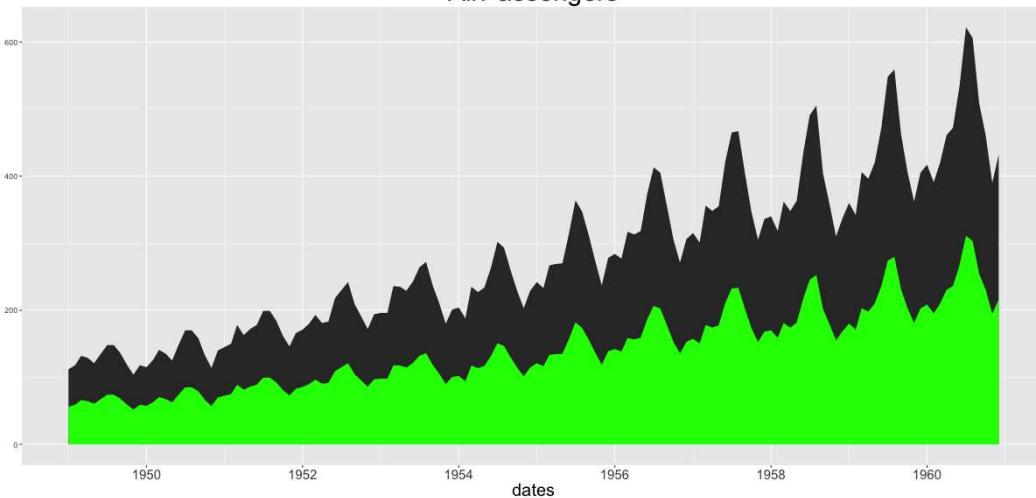
Filled time series can be plotted using `geom_ribbon()`. It takes two compulsory arguments `ymin` and `ymax`.

```
# Prepare the dataframe
st_year <- start(AirPassengers)[1]
st_month <- "01"
st_date <- as.Date(paste(st_year, st_month, "01", sep="-"))
dates <- seq.Date(st_date, length=length(AirPassengers), by="month")
df <- data.frame(dates, AirPassengers, AirPassengers/2)
head(df)

#>           dates AirPassengers AirPassengers.2
#> 1 1949-01-01      112          56.0
#> 2 1949-02-01      118          59.0
#> 3 1949-03-01      132          66.0
#> 4 1949-04-01      129          64.5
#> 5 1949-05-01      121          60.5
#> 6 1949-06-01      135          67.5

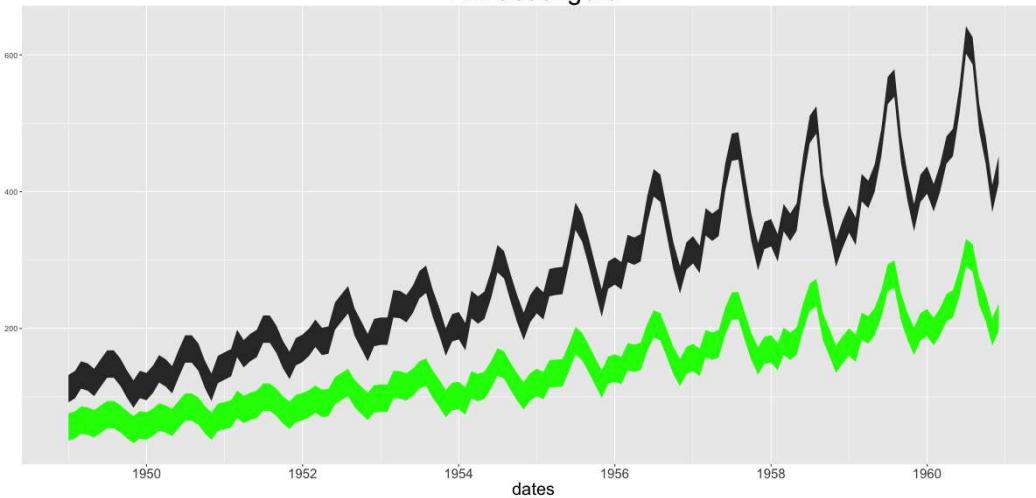
# Plot ribbon with ymin=0
gg <- ggplot(df, aes(x=dates)) + labs(title="AirPassengers") + theme(plot.title=element_text(size=30), axis.title.x=element_text(size=20), axis.text.x=element_text(size=15))
gg + geom_ribbon(aes(ymin=0, ymax=AirPassengers)) + geom_ribbon(aes(ymin=0, ymax=AirPassengers.2), fill="green")
```

AirPassengers



```
gg + geom_ribbon(aes(ymin=AirPassengers-20, ymax=AirPassengers+20)) + geom_ribbon(aes(ym  
in=AirPassengers.2-20, ymax=AirPassengers.2+20), fill="green")
```

AirPassengers

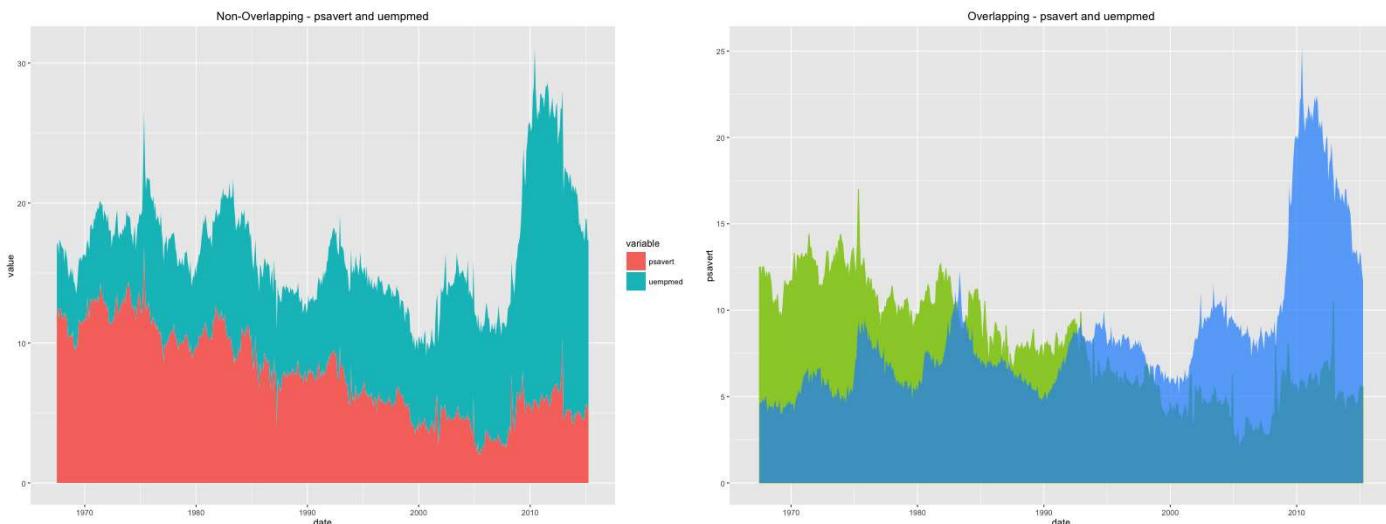


Area

`geom_area` is similar to `geom_ribbon`, except that the `ymin` is set to 0. If you want to make overlapping area plot, use the `alpha` aesthetic to make the top layer translucent.

```
# Method1: Non-Overlapping Area
df <- reshape2::melt(economics[, c("date", "psavert", "uempmed")], id="date")
head(df, 3)
#>       date variable value
#> 1 1967-07-01 psavert 12.5
#> 2 1967-08-01 psavert 12.5
#> 3 1967-09-01 psavert 11.7
p1 <- ggplot(df, aes(x=date)) + geom_area(aes(y=value, fill=variable)) + labs(title="Non-Overlapping - psavert and uempmed")

# Method2: Overlapping Area
p2 <- ggplot(economics, aes(x=date)) + geom_area(aes(y=psavert), fill="yellowgreen", color="yellowgreen") + geom_area(aes(y=uempmed), fill="dodgerblue", alpha=0.7, linetype="dotted") + labs(title="Overlapping - psavert and uempmed")
gridExtra::grid.arrange(p1, p2, ncol=2)
```

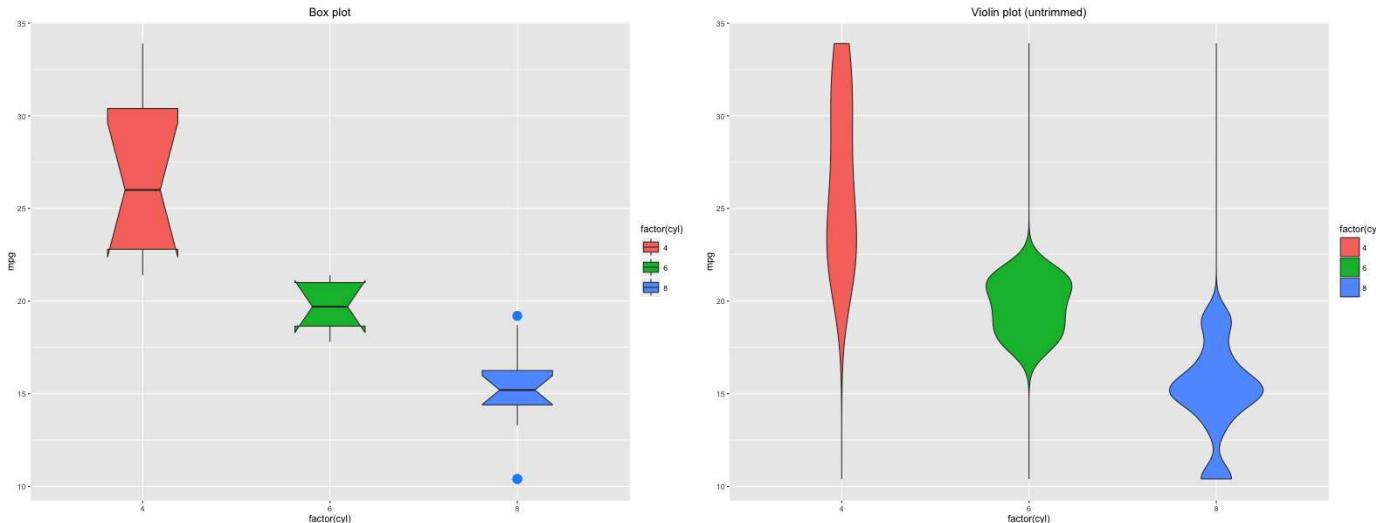


Boxplot and Violin

The outlier points are controlled by the following aesthetics: * outlier.shape * outlier.stroke * outlier.size * outlier.colour

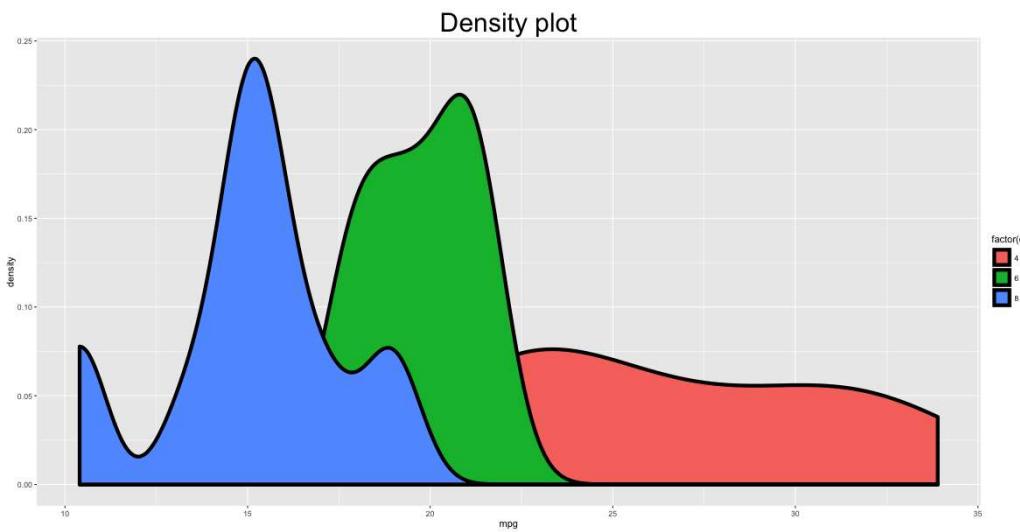
If the notch is turned on (by setting it TRUE), the below boxplot is produced. Else, you would get the standard rectangular boxplots.

```
p1 <- ggplot(mtcars, aes(factor(cyl), mpg)) + geom_boxplot(aes(fill = factor(cyl)), width=0.5, outlier.colour = "dodgerblue", outlier.size = 4, outlier.shape = 16, outlier.stroke = 2, notch=T) + labs(title="Box plot") # boxplot
p2 <- ggplot(mtcars, aes(factor(cyl), mpg)) + geom_violin(aes(fill = factor(cyl)), width=0.5, trim=F) + labs(title="Violin plot (untrimmed)") # violin plot
gridExtra::grid.arrange(p1, p2, ncol=2)
```



Density

```
ggplot(mtcars, aes(mpg)) + geom_density(aes(fill = factor(cyl)), size=2) + labs(title="Density plot") # Density plot
```



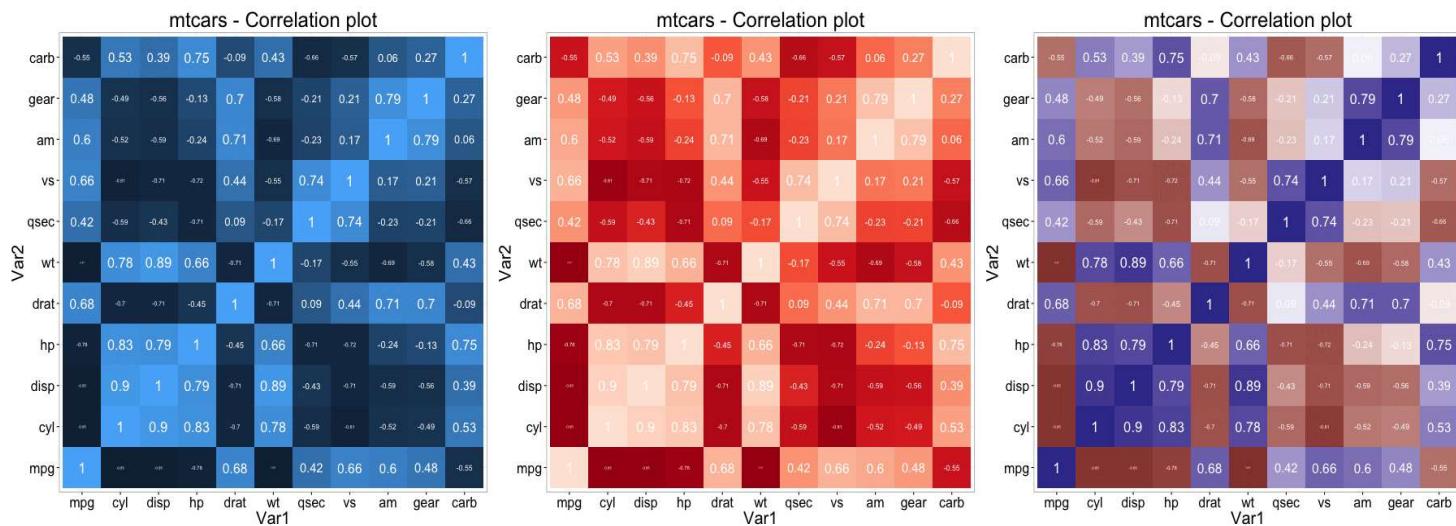
Tiles

```

corr <- round(cor(mtcars), 2)
df <- reshape2::melt(corr)
gg <- ggplot(df, aes(x=Var1, y=Var2, fill=value, label=value)) + geom_tile() + theme_bw()
() + geom_text(aes(label=value, size=value), color="white") + labs(title="mtcars - Correlation plot") + theme(text=element_text(size=20), legend.position="none")

library(RColorBrewer)
p2 <- gg + scale_fill_distiller(palette="Reds")
p3 <- gg + scale_fill_gradient2()
gridExtra::grid.arrange(gg, p2, p3, ncol=3)

```



© 2016-17 Selva Prabhakaran. Powered by jekyll (<http://jekyllrb.com/>), knitr (<http://yihui.name/knitr/>), and pandoc (<http://johnmacfarlane.net/pandoc/>). This work is licensed under the Creative Commons License. (<http://creativecommons.org/licenses/by-nc/3.0/>)