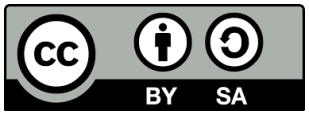
R graphics with ggplot2



Véronique Martin, Sophie Schbath &
Christelle Hennequet-Antier

March 17, 2022

Tour de table

- Qui êtes-vous ? (nom, unité, activité)
- Vos besoins par rapport à votre activité ?
- Vos attentes par rapport à la formation ?

Déroulé de la formation

- Un jour : 17 mars 2022
- Horaires : 9h30 - 17h
- Déjeuner : 12h30 - 14h
- Pauses : 11h et 15h

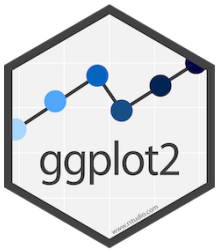
Objectif pédagogique

À l'issue de la formation :

- les stagiaires connaîtront les principales fonctionnalités du package R « ggplot2 » et la démarche sous-jacente pour construire un graphique à partir d’un tableau de données.
- Ils seront capables de réaliser plusieurs types de représentations graphiques, telles que des nuages de points, des courbes, des histogrammes, des diagrammes en bâtons, des boxplots, des heatmaps, etc.

Preliminary

Brief history



R package developed by Hadley Wickham (https://hadley.nz/) is a part of **tidyverse**'s ecosystem.

Creating graphics based on

- Grammar of Graphics (Wilkinson, 2005)
- A layered grammar of graphics (Wickham, 2010)

Citation: H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

One needs to load the corresponding R package

```
library(ggplot2)
```

Here are the datasets we will play with

```
data(diamonds)
names(diamonds)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

```
data(ChickWeight)
names(ChickWeight)
```

```
## [1] "weight" "Time"   "Chick"  "Diet"
```

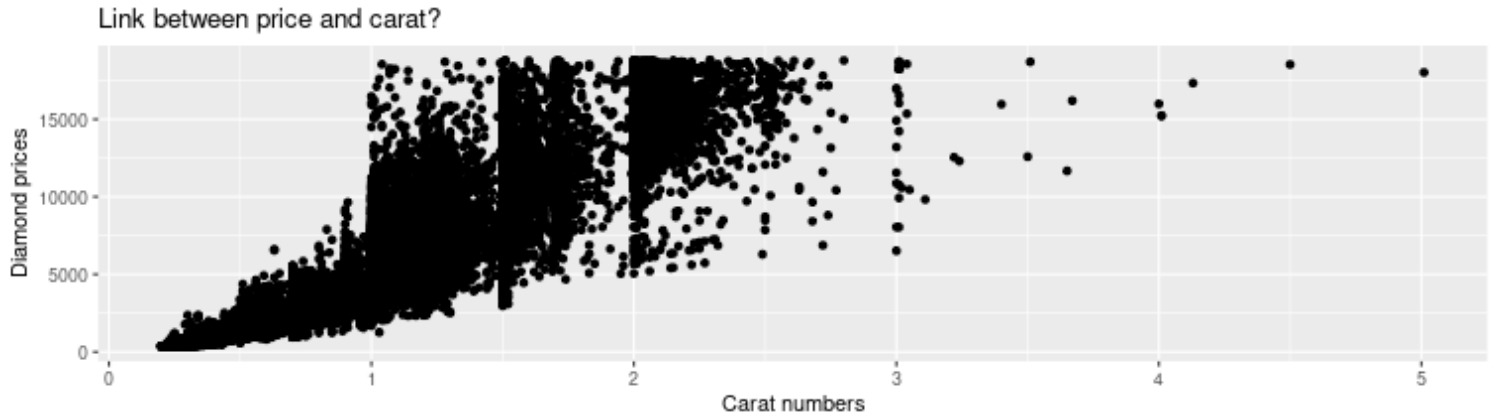
What is a ggplot graphics?

A ggplot graphics is an R object composed of the "sum" of different layers.

These layers define successively the type of plot, the title, the axes labels, the background, the scale, the colors, the size, etc.

A ggplot graphics is attached to a unique data frame containing the data to plot.

```
p <- ggplot(data=diamonds) +
  geom_point(aes(y=price, x=carat)) +
  ggtitle("Link between price and carat?") +
  ylab("Diamond prices") + xlab("Carat numbers")
p
```



Syntax ggplot()

- ggplot(data = DATA, mapping =aes(x,y, ...)) +

  GEOM_FUNCTION()

  all layers use same data and same parameters aes()

- ggplot(data = DATA) +

  GEOM_FUNCTION(mapping =aes(x,y, ...))

  all layers use same data and parameters aes() may vary

- ggplot() +

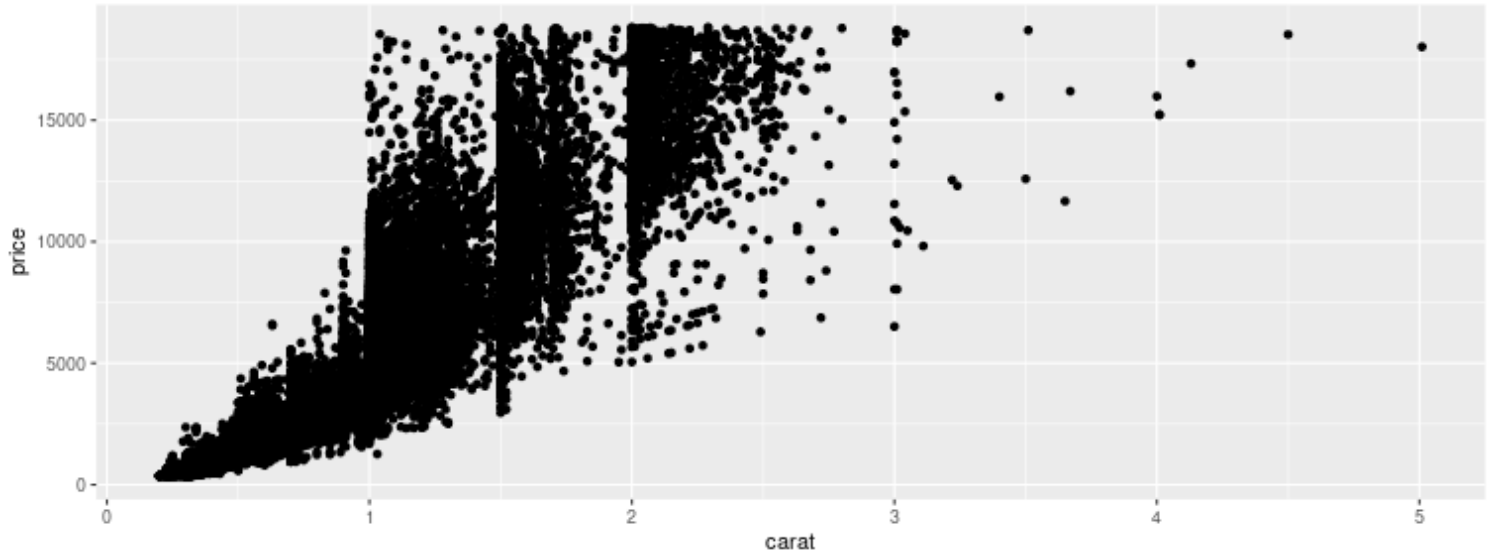
  GEOM_FUNCTION(data = DATA, mapping =aes(x,y, ...))

  layers can use different data and parameters

1. Scatter plot with geom_point()

Basic

```
p <- ggplot(data=diamonds, mapping=aes(y=price, x=carat))
p + geom_point()
```

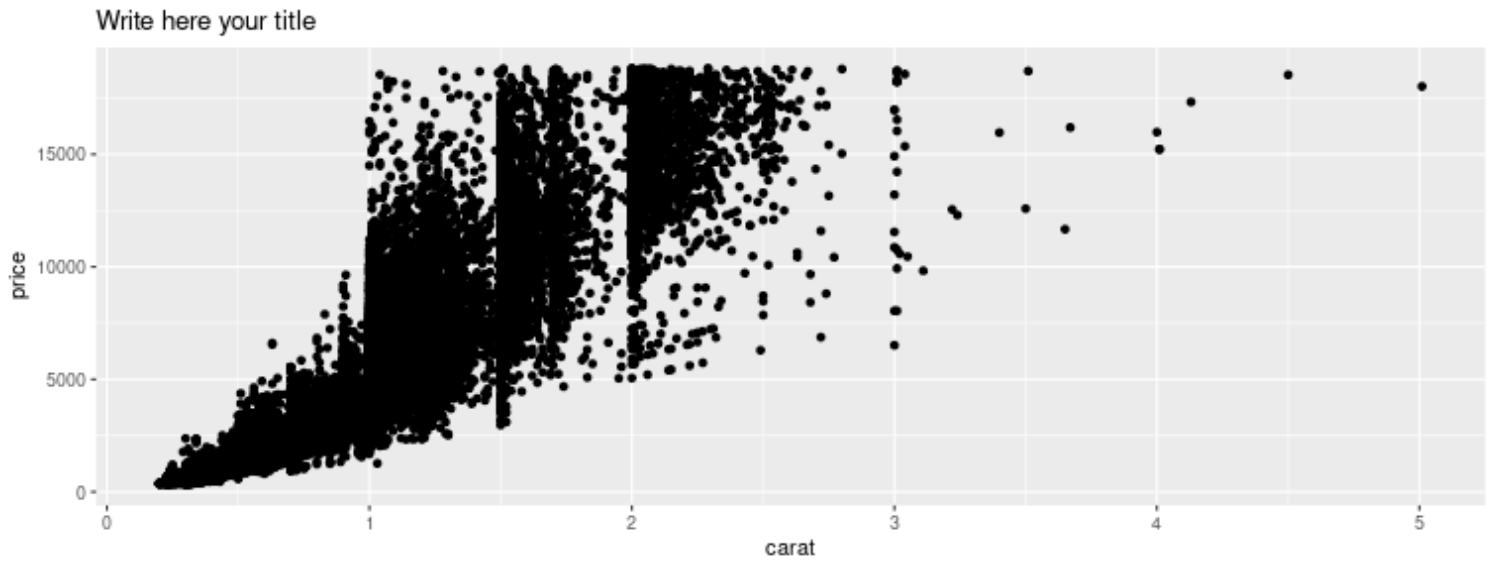


The first line specifies where to find the data and the global aesthetics; here the 'price' variable is mapped to y and the 'carat' variable is mapped to x.

The second line adds the points: x in x-axis and y in y-axis. No local aesthetics parameter --> the global ones are used.

To add a title with ggtitle()

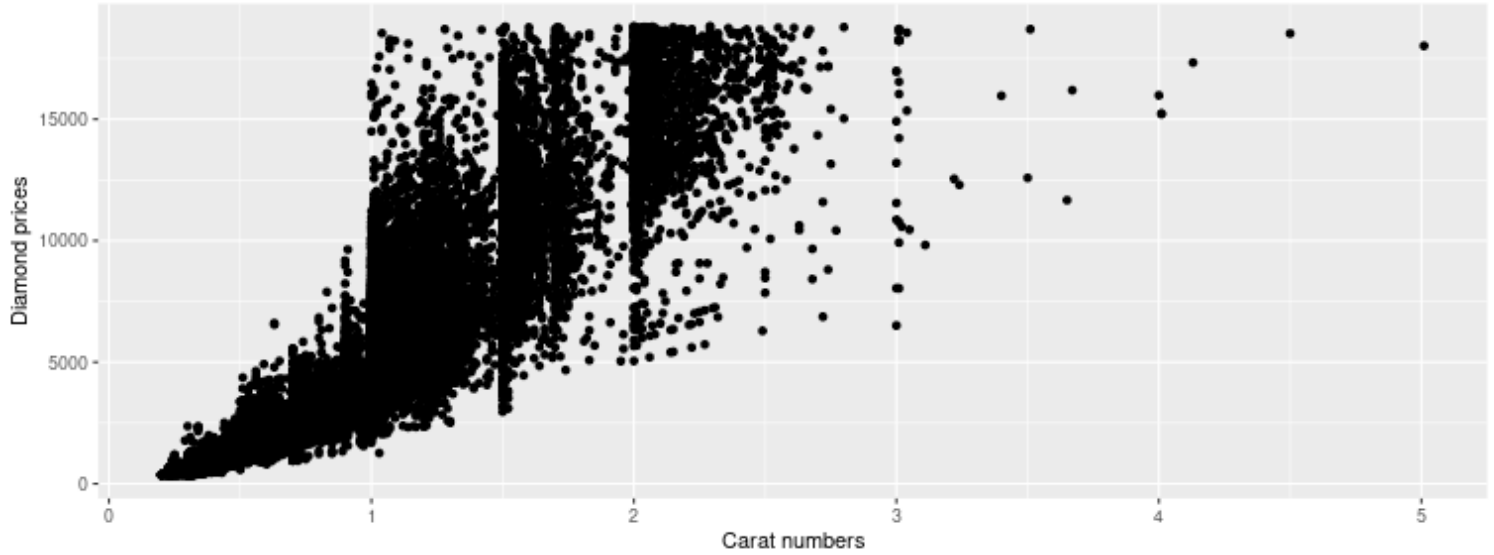
```
p <- ggplot(data = diamonds, mapping = aes(y = price, x = carat))
p <- p + geom_point()
p + ggtitle("Write here your title")
```



We will see later how to change the title's appearance thanks to the 'theme(plot.title=)' function.

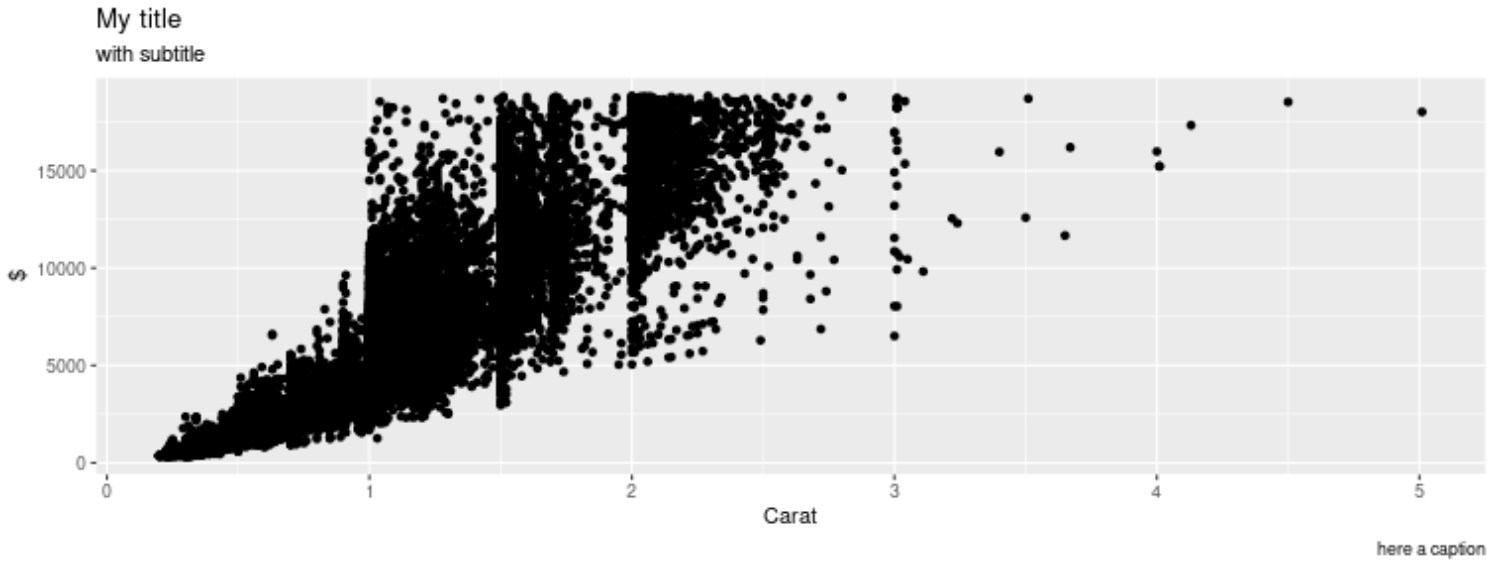
To change x and y labels with xlab(), ylab()

```
p + ylab("Diamond prices") + xlab("Carat numbers")
```



An other way with labs()

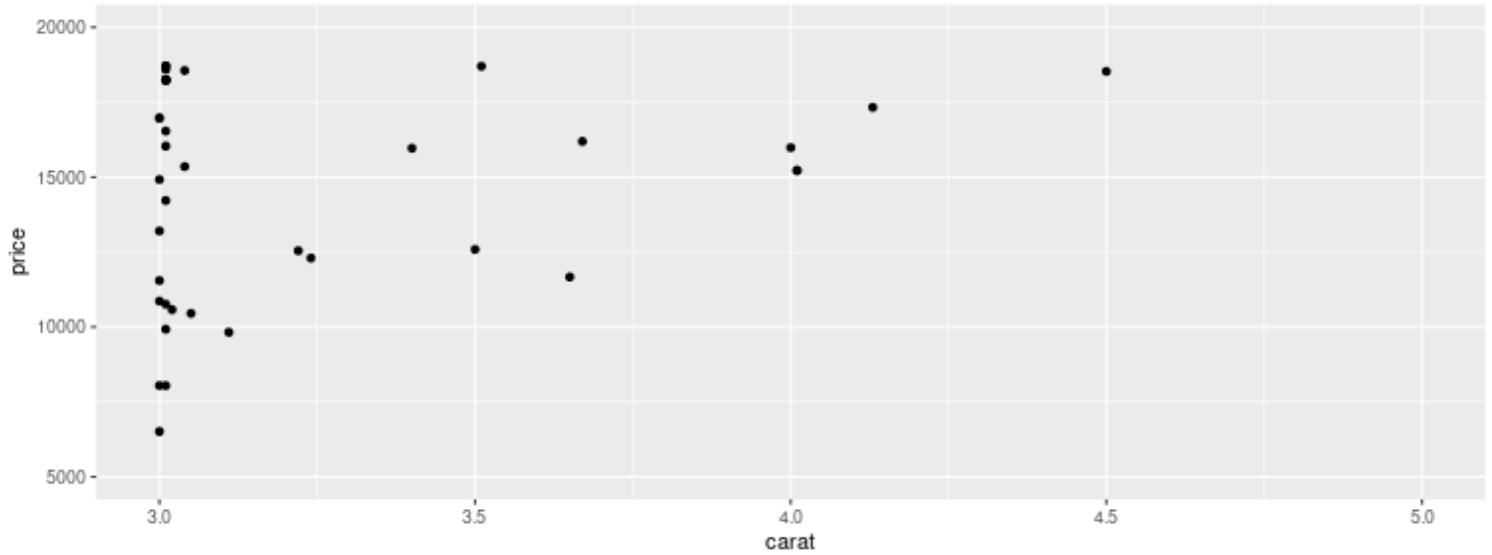
```
p + labs(title="My title", subtitle="with subtitle",
         caption="here a caption", y="$", x="Carat")
```



To change the axes scale

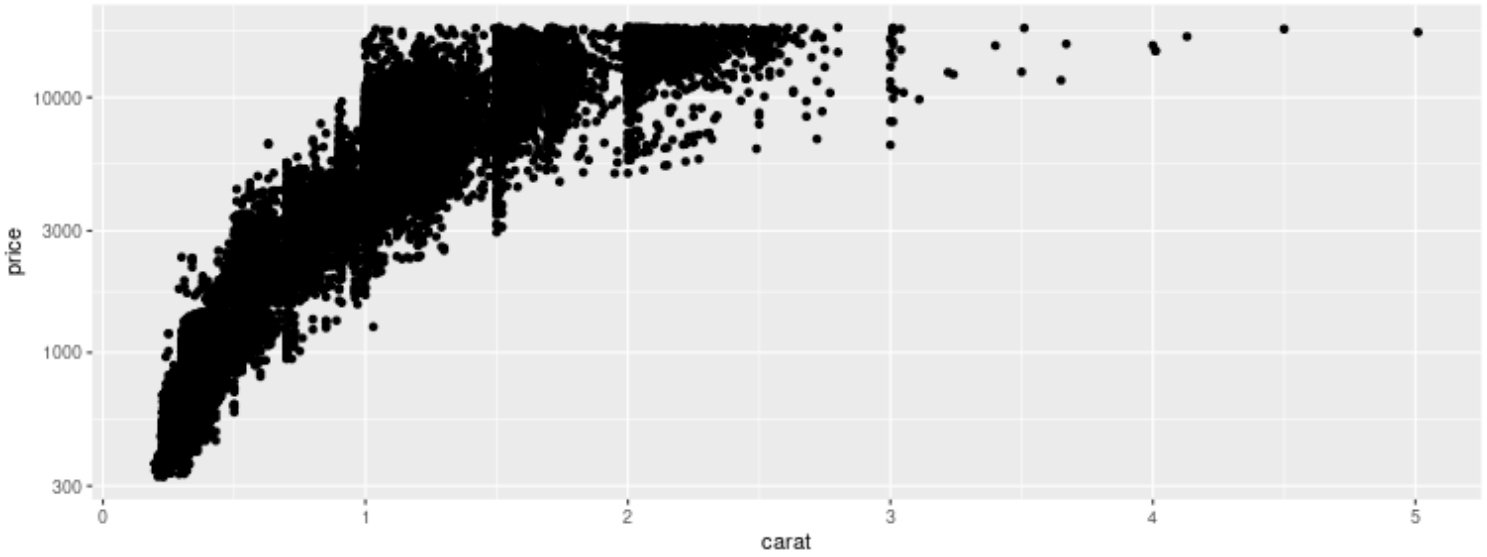
To zoom with `xlim()`, `ylim()`

```
p<-ggplot(data=diamonds, mapping=aes(y=price, x=carat))+
  geom_point()
p + ylim(5000, 20000) + xlim(3,5)
```



For logarithmic scales:

```
p + scale_y_log10()
```

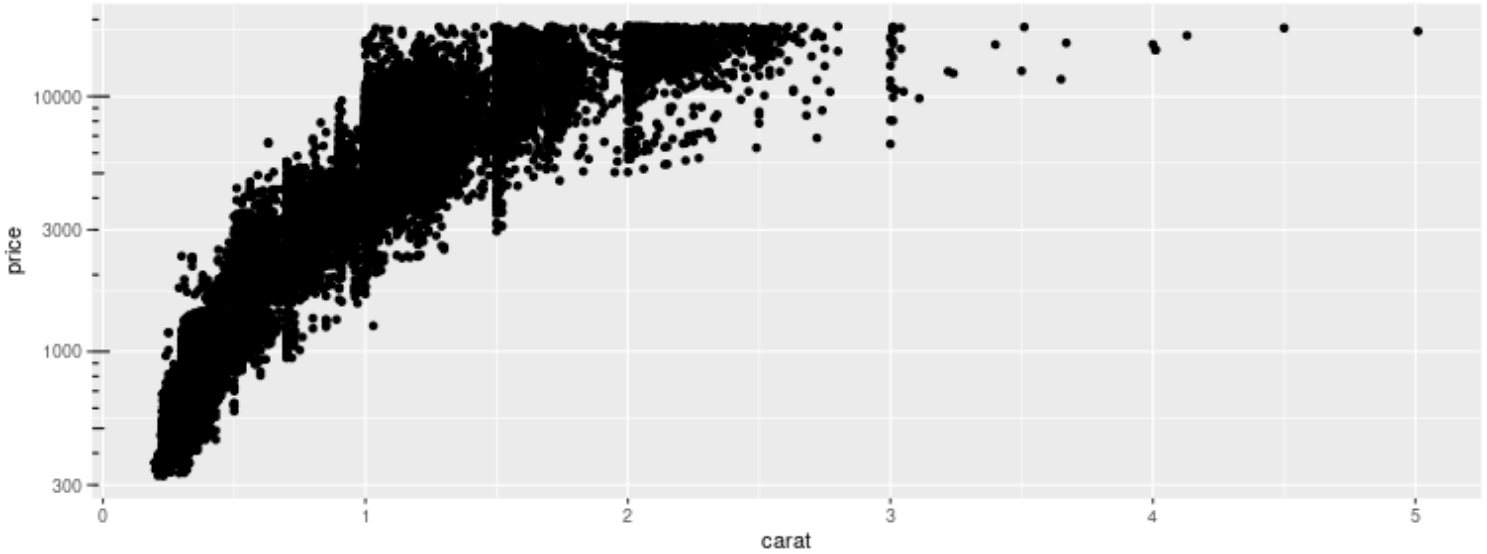


Note: use `scale_x_log10()` for the x-axis.

Note: this is a shortcut for `scale_y_continuous(trans="log10")`; `trans=` can take value from 'log2', 'log', 'log10', 'sqrt'.

Check if you additionaly use `annotation_logticks()`:

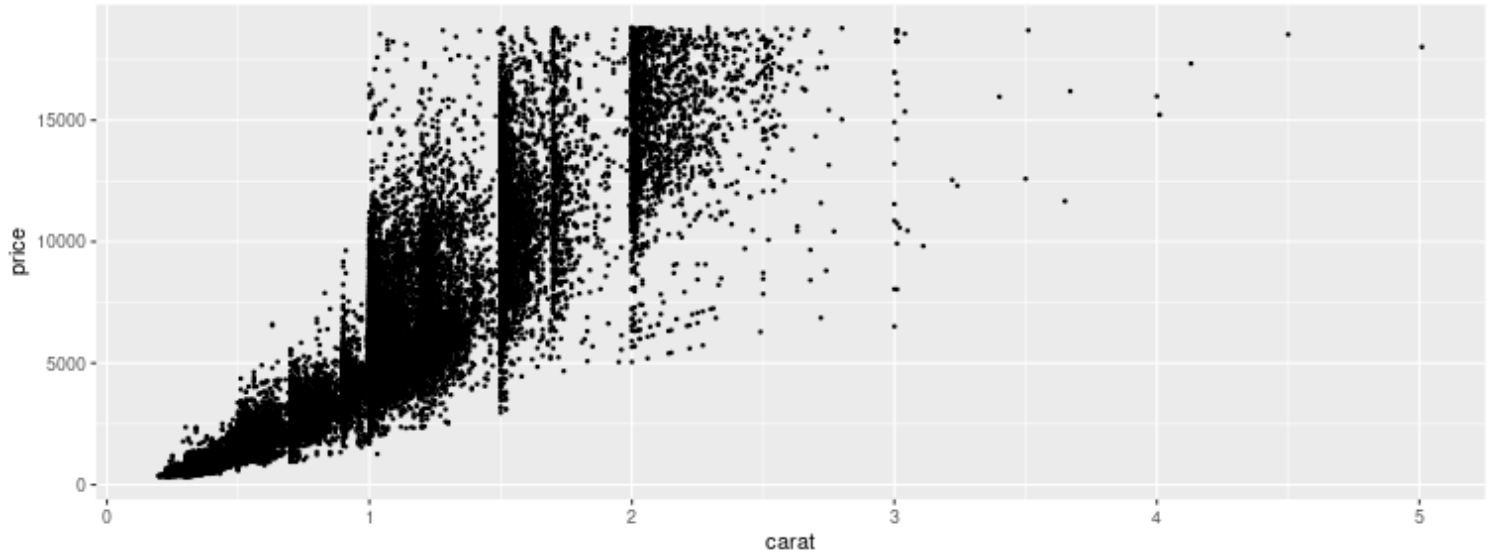
```
p + scale_y_log10() + annotation_logticks(base = 10, sides = "l")
```



Note: `sides` is a string that controls which sides of the plot the log ticks appear on. It can be set to a string containing any of "trbl", for top, right, bottom, and left.

To change the point size

```
p <- ggplot(data = diamonds, mapping = aes(y = price, x = carat))
p + geom_point(size = 0.5)
```

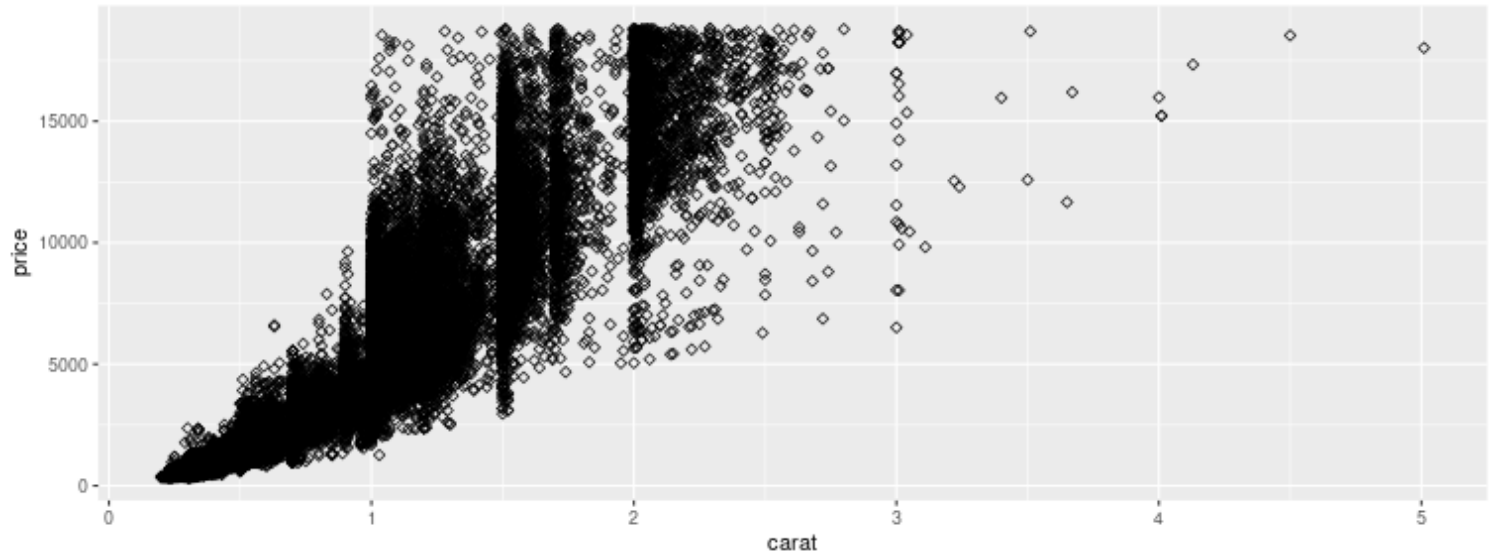


Note: the `alpha=` option would draw the point with some transparency (useful for superimposed points).

To change the point shape

Same shape for all points

```
p + geom_point(shape = 5)  # possible: shape='A'
```



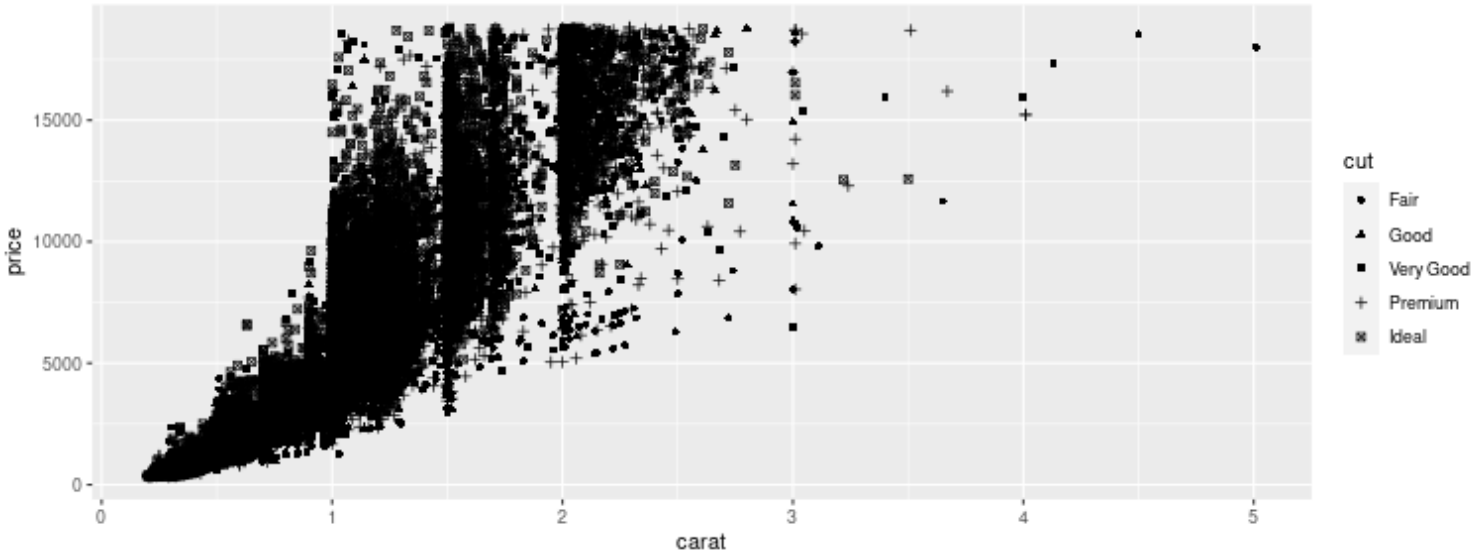
Note: shape can be any integer in [0,25] or a single character.

Different shapes wrt the variable 'cut'

For that, one uses the `shape` option of the aesthetics function:

```
p + geom_point(aes(shape = cut))
```

```
## Warning: Using shapes for an ordinal variable is not advised
```

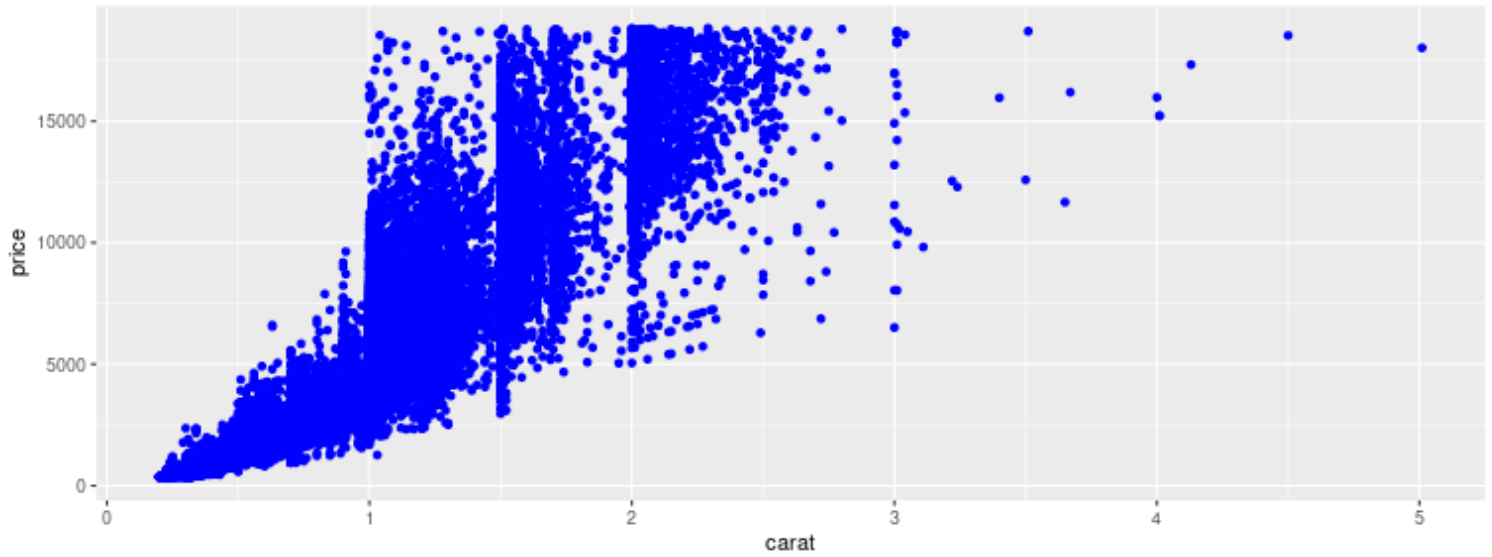


Note: A legend appears, by default on the right; it is possible to move the legend thanks to the function `theme(legend.position="bottom")` (see later).

To change the point color (1/2)

Same color for all points

```
p + geom_point(color = "blue")
```



Note: a large collection of colors can be obtained via `colors()`

Practice

All practices will be made via the 'metacow' dataset:

```
metacow <- read.csv(file = "data/metacow.csv", header = T)
names(metacow)
```

```
##  [1] "OTU"       "Kingdom"   "Phylum"    "Class"     "Order"     "Family"
##  [7] "Genus"     "Species"   "Cow"       "Genotype"  "Location"  "Sample"
## [13] "Abundance"
```

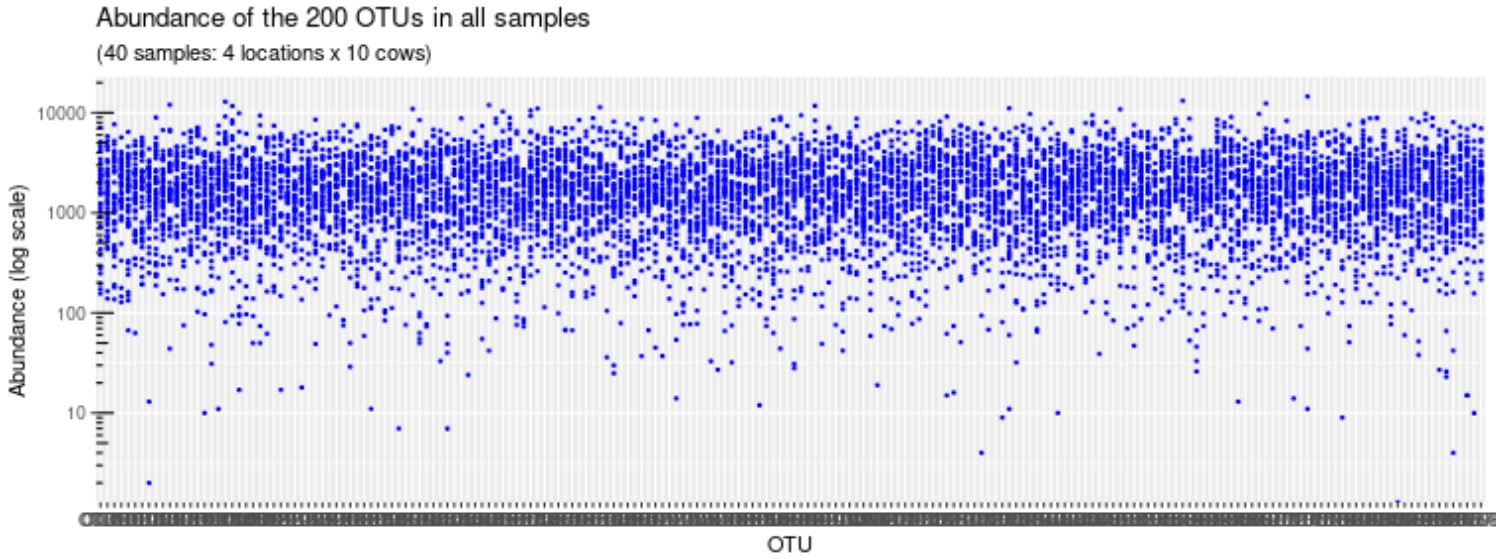
Show 4 entries Search:

	OTU	Kingdom	Phylum	Class	Order	Family	Ge
1	OTU1	Bacteria	Actinobacteria	Actinobacteria	Micrococcales	Promicromonosporaceae	Cellulosi
2	OTU2	Bacteria	Actinobacteria	Actinobacteria_class	Micrococcales	Micrococcaceae	Glutamic
3	OTU3	Bacteria	Firmicutes	Bacilli	Lactobacillales	Lactobacillaceae	Lactobac
4	OTU4	Bacteria	Actinobacteria	Actinobacteria	Propionibacteriales	Propionibacteriaceae	Cutibacte

Showing 1 to 4 of 10 entries Previous 1 2 3 Next

Practice n°1

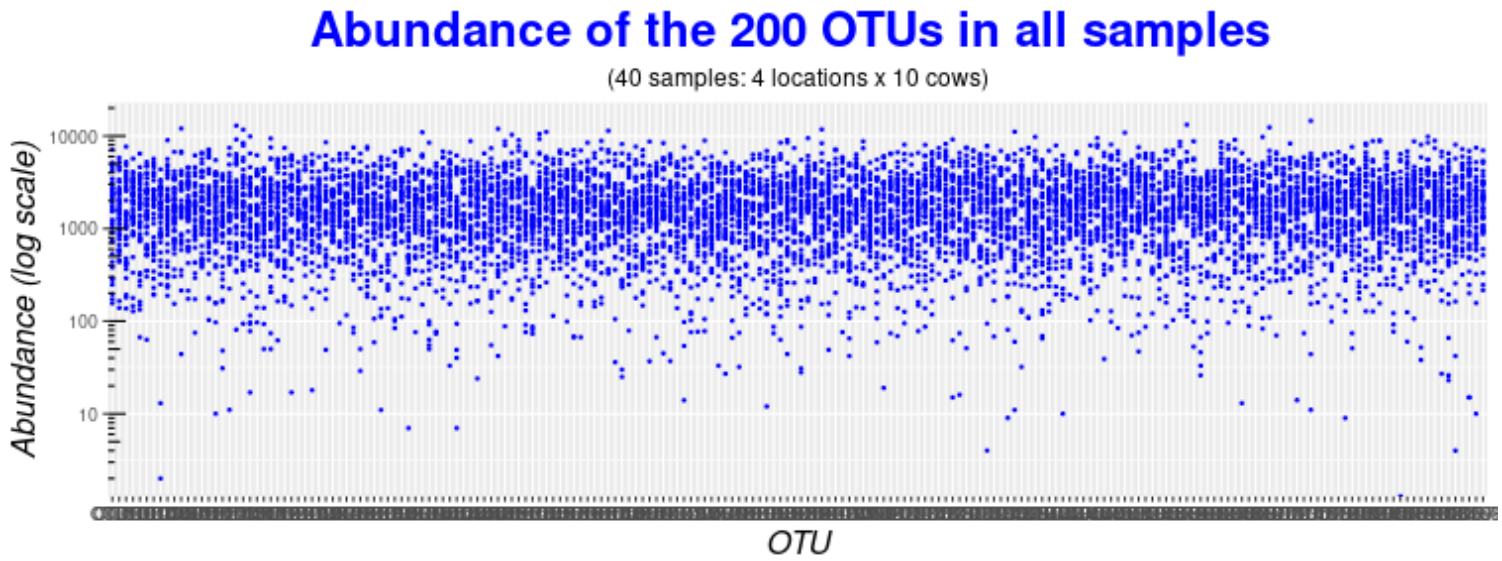
Reproduce the graphics below:



Solution n°1

Practice n°1: bonus

Reproduce the graphics below:



Clues: use the `theme()` function and its options `plot.title`, `plot.subtitle`, `axis.title` which are objects returned by the `element_text()` function.

Solution n°1: bonus

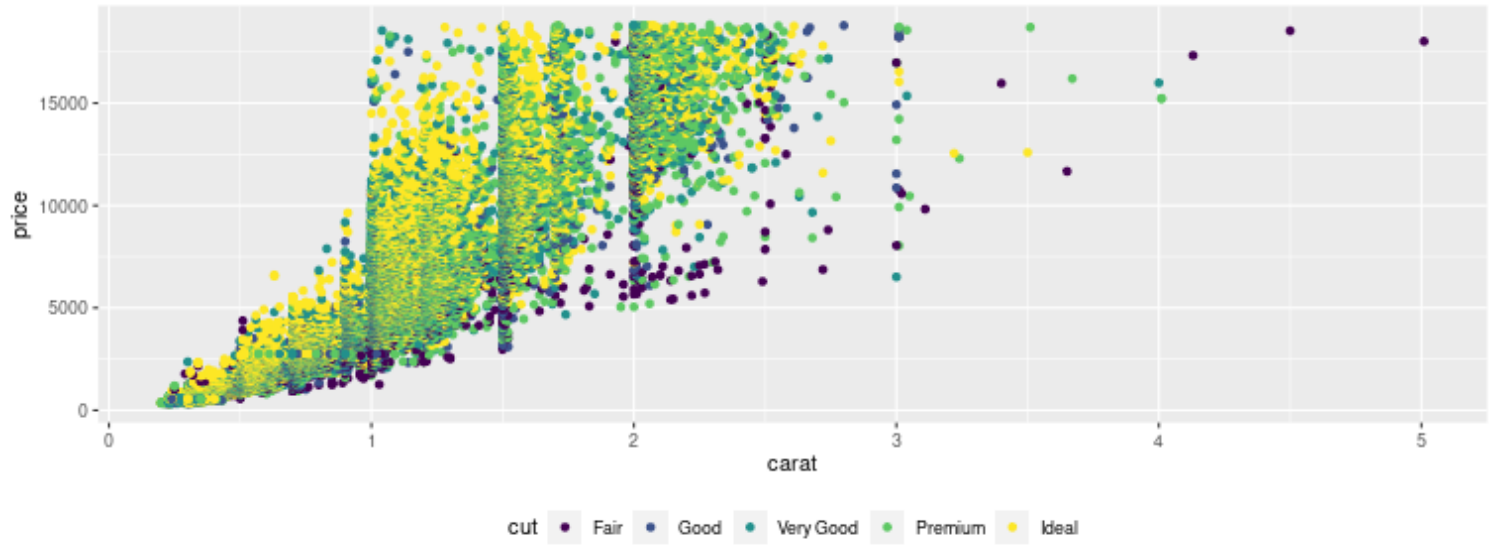
Note: `hjust=` in [0,1], `face` in "plain", "italic", "bold" and "bold.italic"

To change the point color (2/2)

Different colors wrt the variable 'cut'

Check if you rather use the `color` option of the aesthetics function:

```
p + geom_point(aes(color = cut)) + theme(legend.position = "bottom")
```

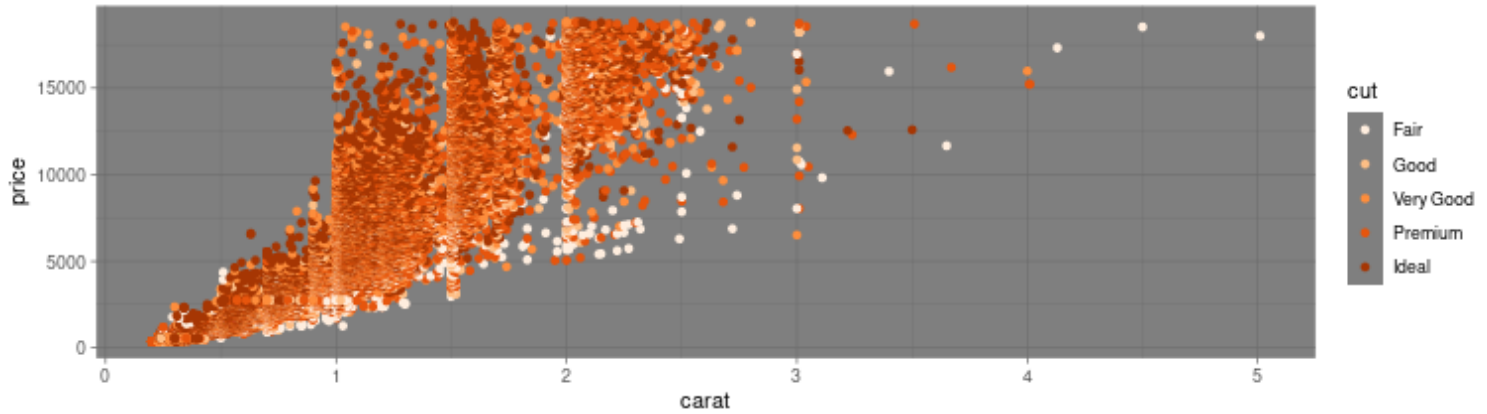


Note: the default color palette is used, it is possible to change it (see later).

To change the color palette ...

By using `scale_color_brewer()`

```
p <- p + geom_point(aes(color = cut))
p + scale_color_brewer(palette = "Oranges") + theme_dark()
```

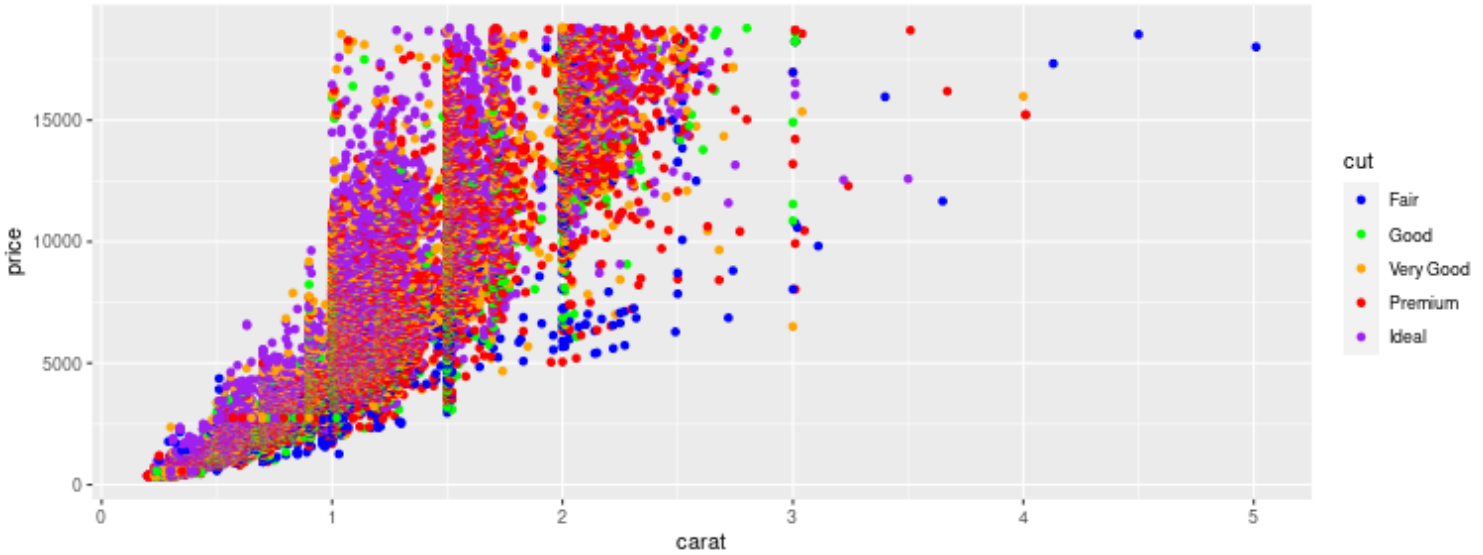


Note: the default palette of `scale_color_brewer()` is a blue palette. See the manual for all other possibilities.

Note: use `theme_bw()`, `theme_classic()` to change the background color.

By using `scale_color_manual()`

```
my.palette <- c("blue", "green", "orange", "red", "purple")
p + scale_color_manual(values = my.palette)
```

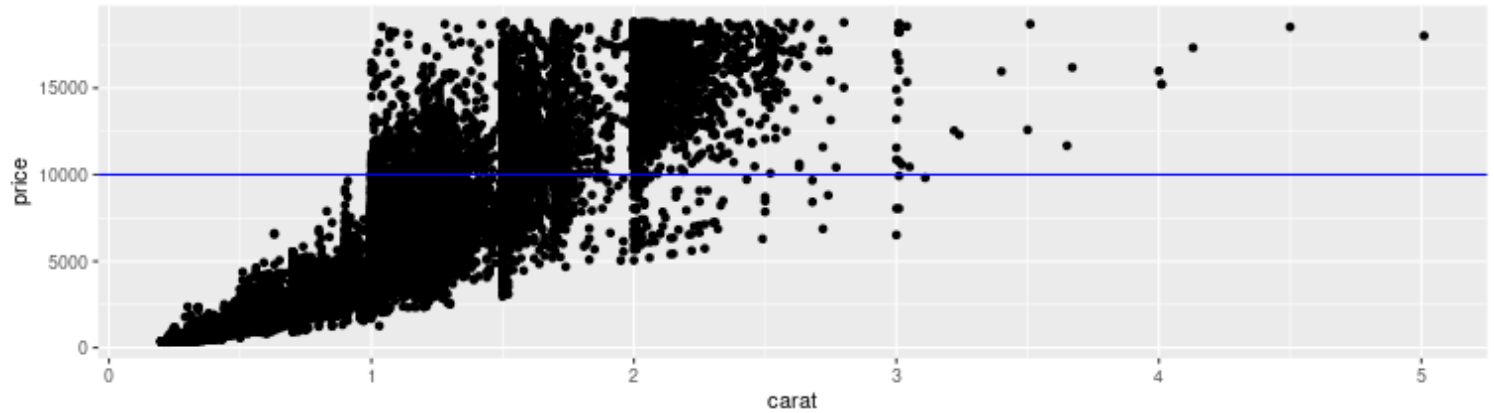


Note: similarly, there is `scale_shape_manual()` for the point shapes.

To add a line

Horizontal line with `geom_hline()`

```
p<-ggplot(data=diamonds, mapping=aes(y=price, x=carat)) +
  geom_point()
p + geom_hline(yintercept=10000, color="blue")
```

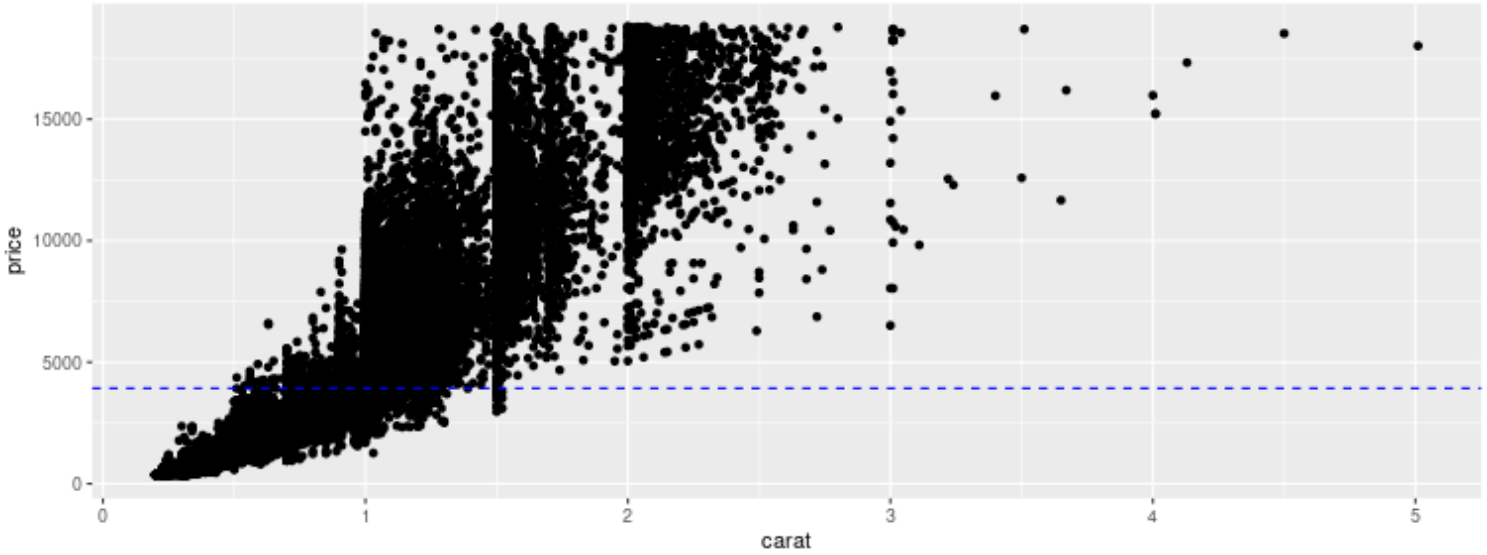


Note: Vertical line with `geom_vline(xintercept=)`

General case with `geom_abline(intercept= , slope= )`

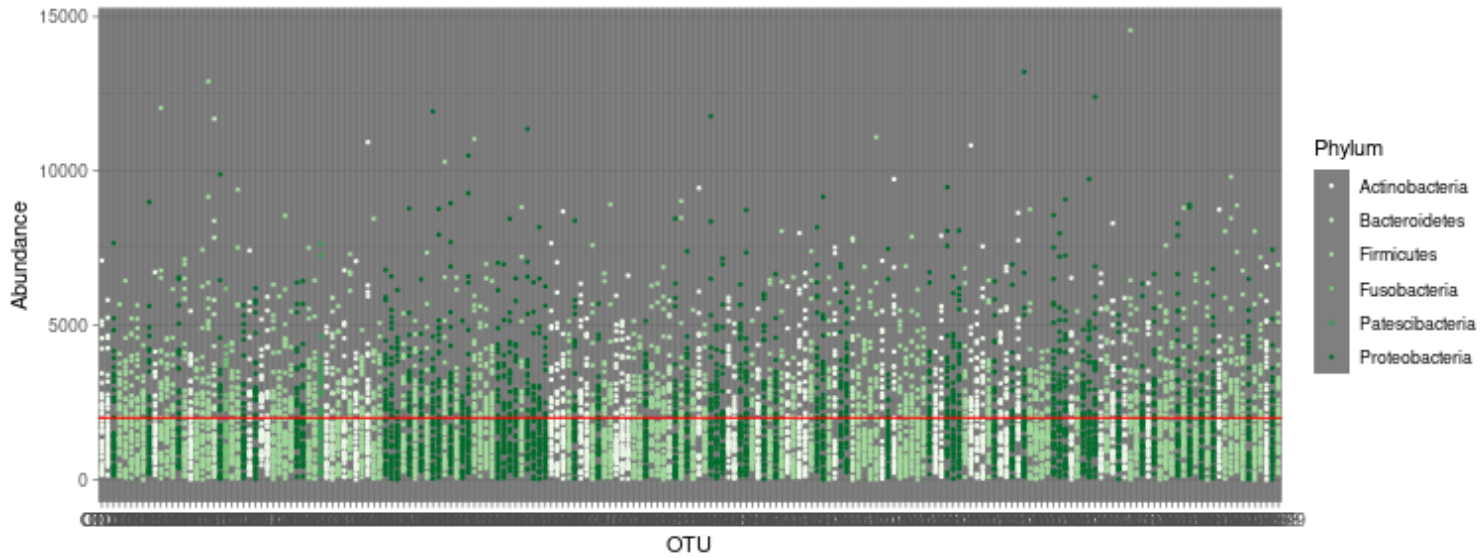
Use the aesthetic function to access to the data:

```
#p + geom_hline(yintercept=10000, color="blue")
p + geom_hline(aes(yintercept=mean(price)), color="blue",
               ,linetype = "dashed")
```



Practice n°2

Reproduce the graphics below:



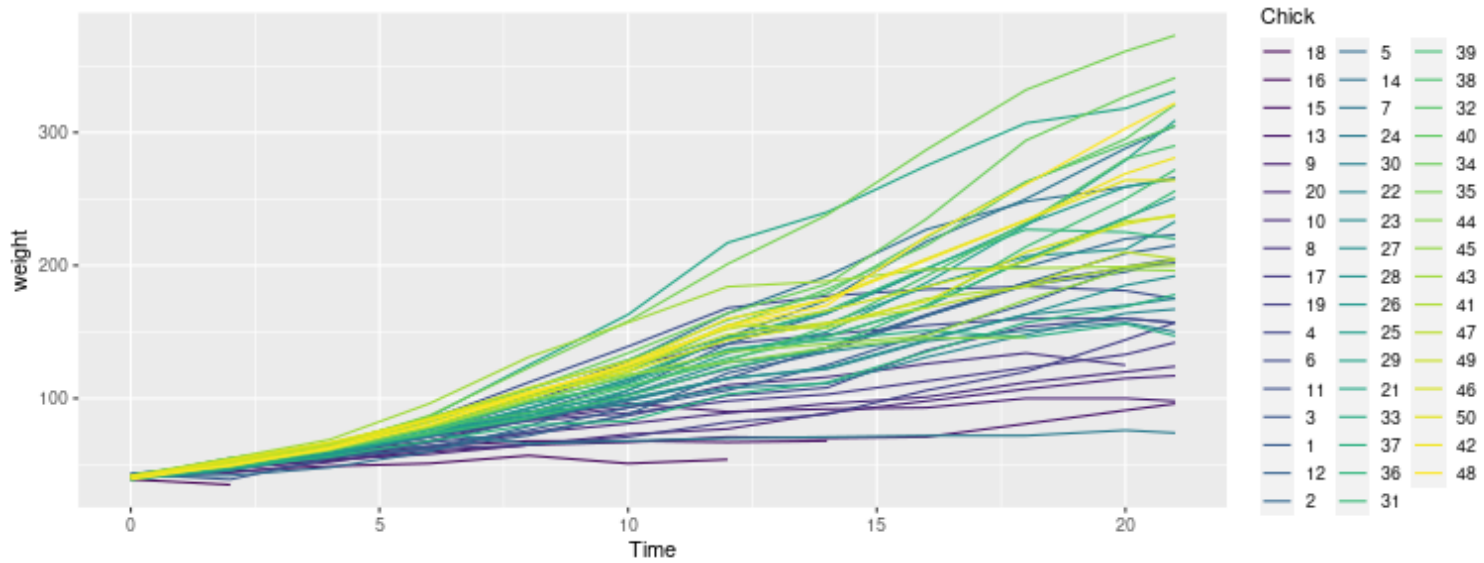
Note: the line indicates the mean abundance.

Solution n°2

2. Curves with `geom_line()`

Basic

```
p<- ggplot(data=ChickWeight,aes(y=weight, x=Time))
p + geom_line(mapping=aes(color=Chick)) +
  theme(legend.key.height=unit(0.5,"cm"))
```



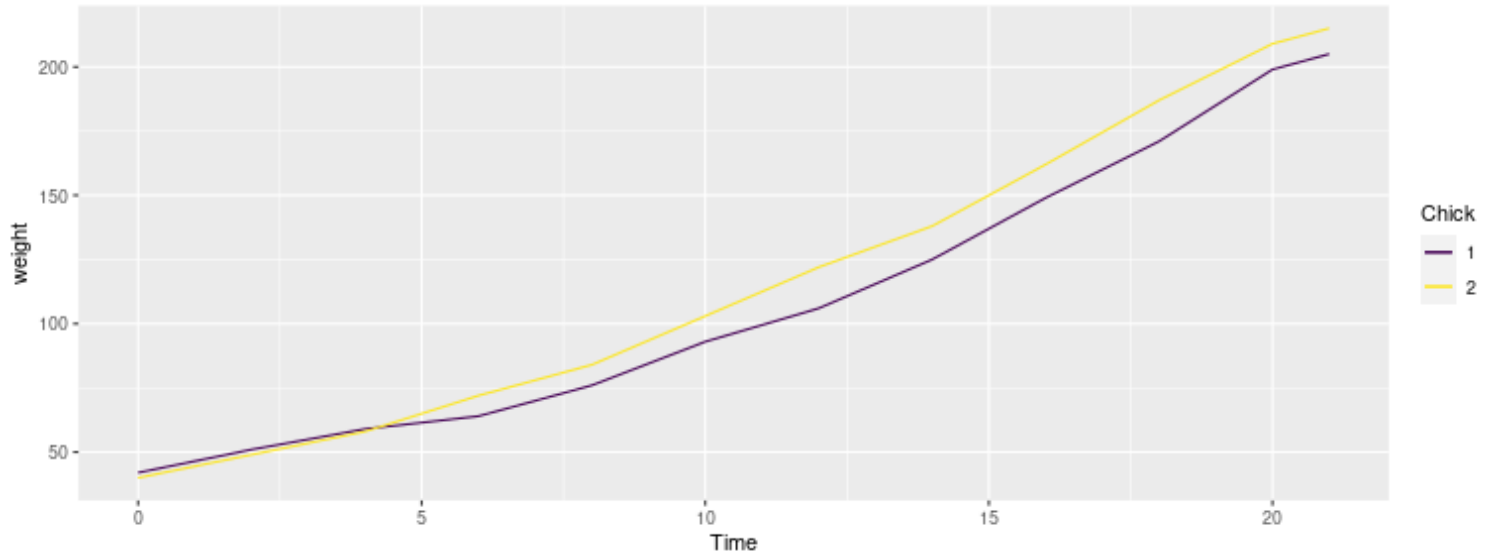

```
# + theme(legend.position = c(0.15, 0.5))
```

To change the label's order in the legend: `ChickWeight$Chick<-factor(ChickWeight$Chick, levels=1:48)`

To keep few curves:

you need to filter specifically your data

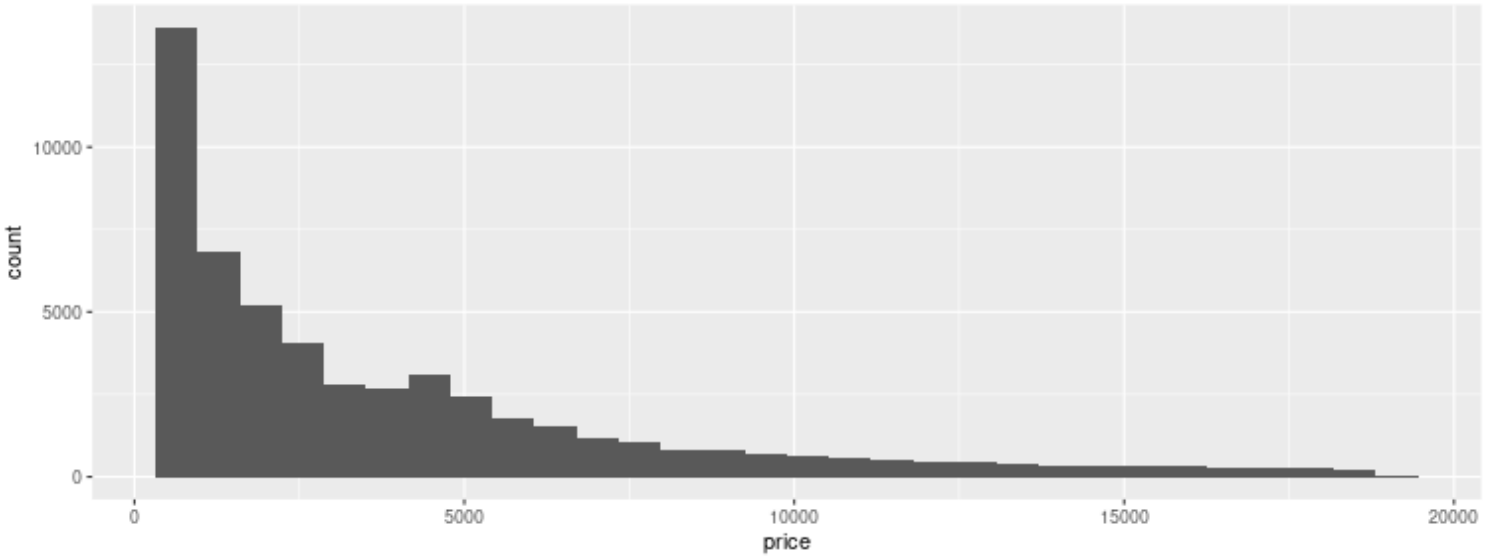
```
ggplot(data=ChickWeight) +
  geom_line(data=subset(ChickWeight, Chick==1),
            mapping=aes(y=weight, x=Time, color= Chick))  +
  geom_line(data=subset(ChickWeight, Chick==2),
            mapping=aes(y=weight, x=Time, color= Chick))
```



3. Histograms with `geom_histogram()`

Basic

```
p <- ggplot(data = diamonds, mapping = aes(x = price))
p + geom_histogram()
```

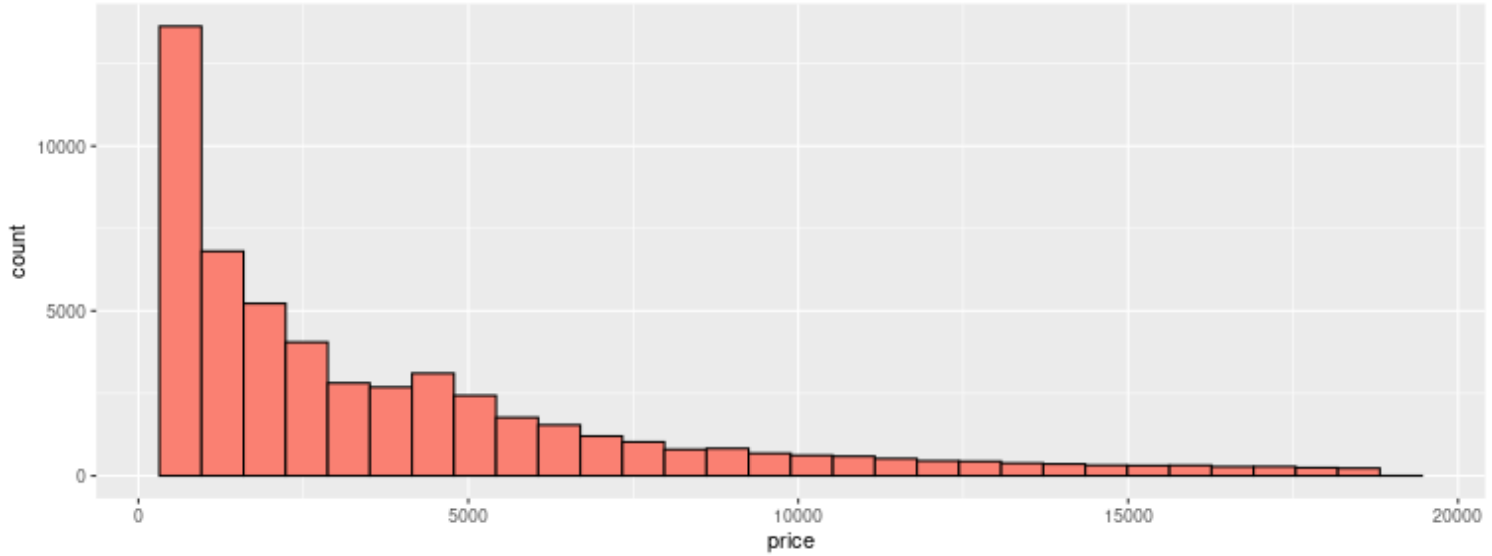



```
# counts by default

# p + geom_histogram(aes(y=..density..)) + geom_density()
```

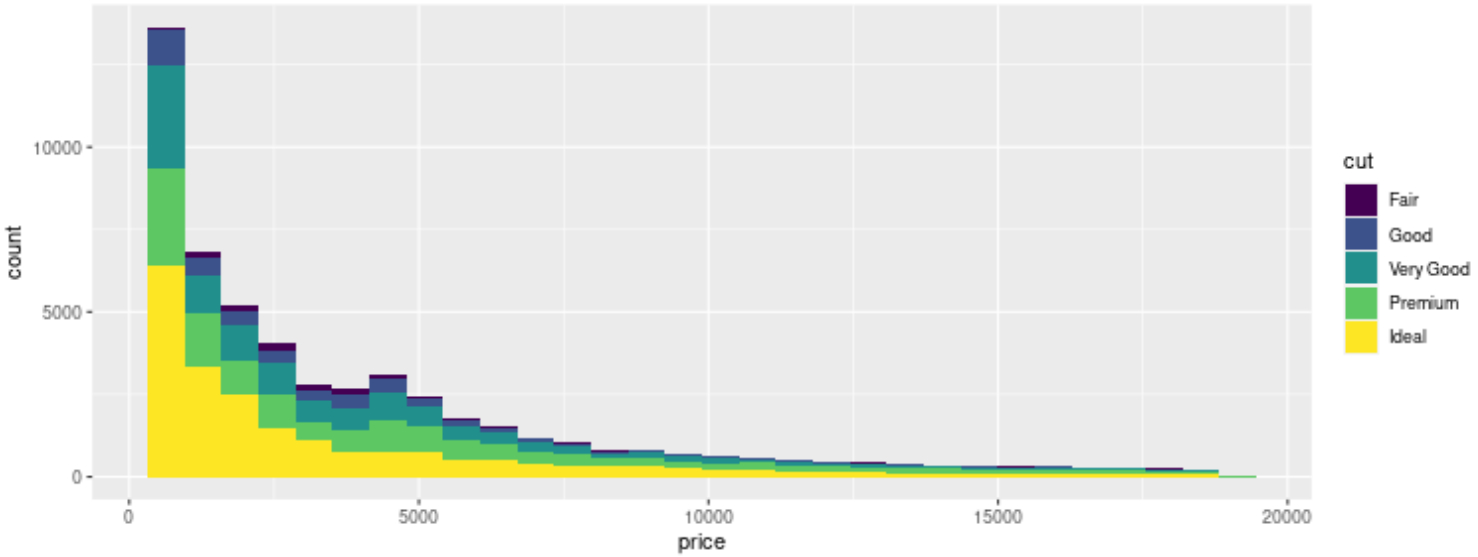
To change the colors

```
p + geom_histogram(fill = "salmon", color = "black")
```



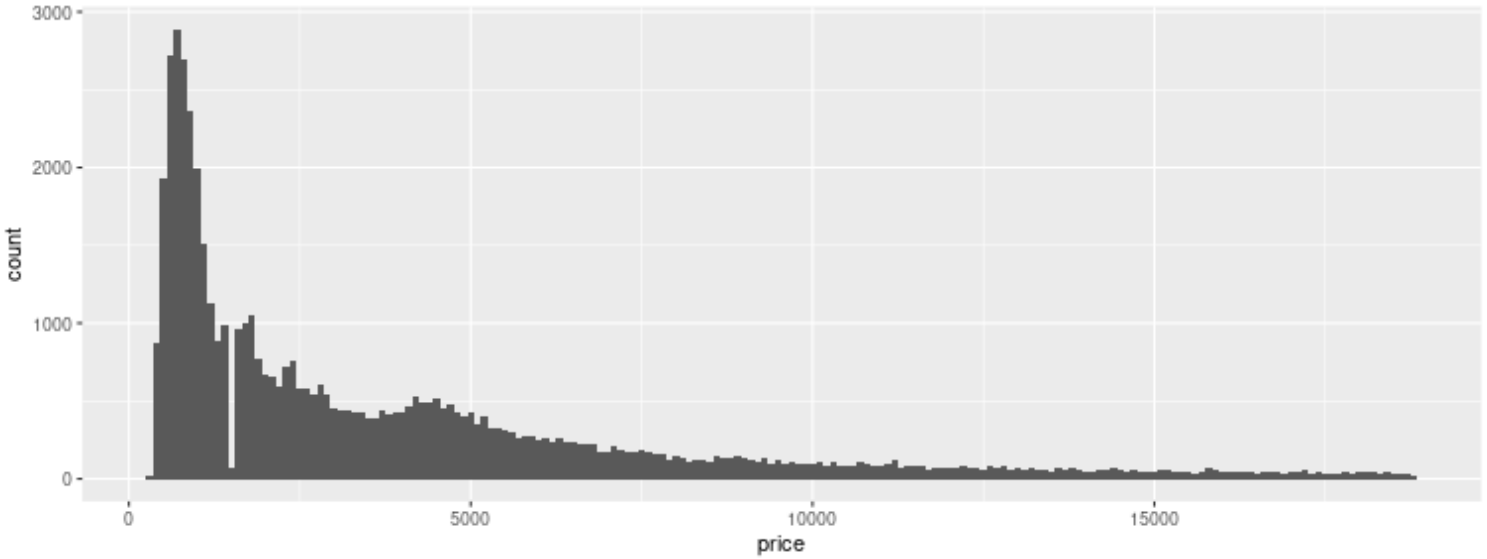
To color wrt variable 'cut'

```
p + geom_histogram(mapping = aes(fill = cut))
```



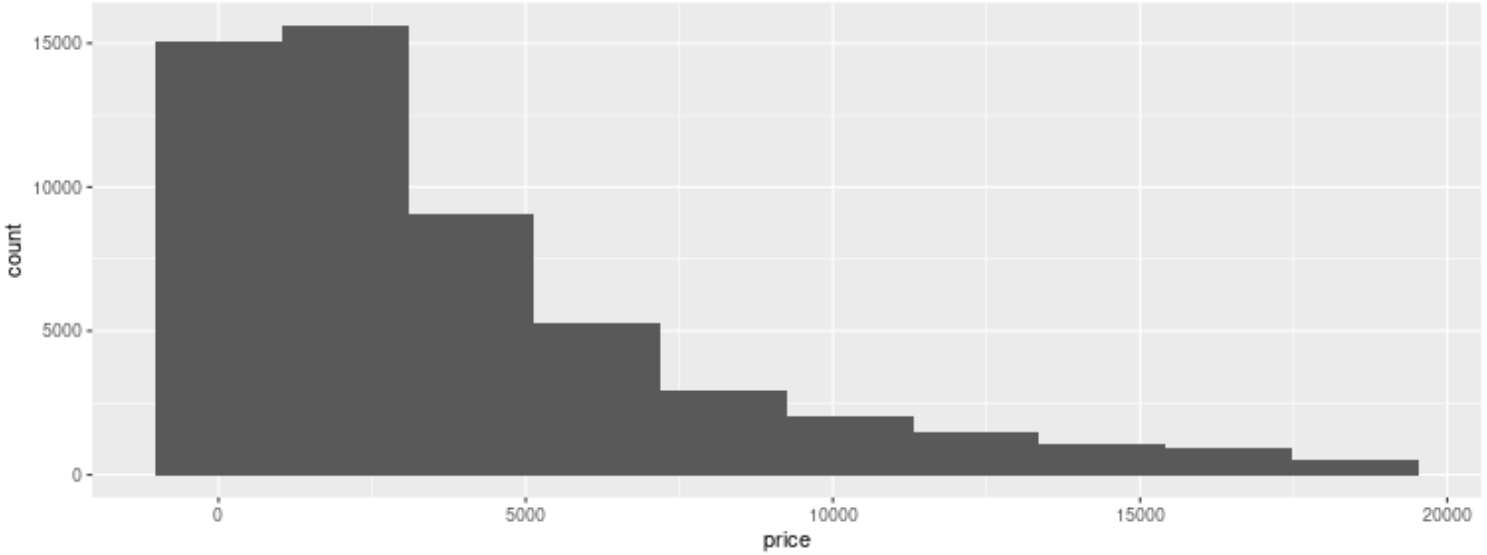
To change the bin width with `binwidth` option

```
p + geom_histogram(binwidth = 100)
```



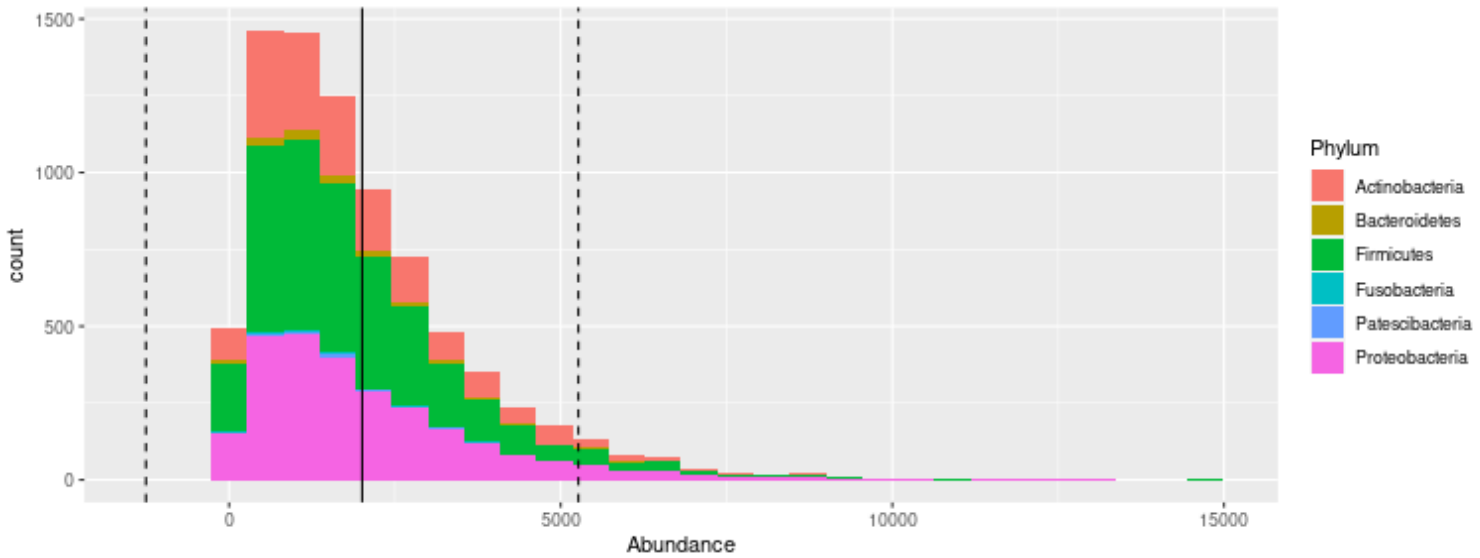
To change the bin number with `bins` option

```
p + geom_histogram(bins = 10)
```



Practice n°3

Reproduce the graphics below:



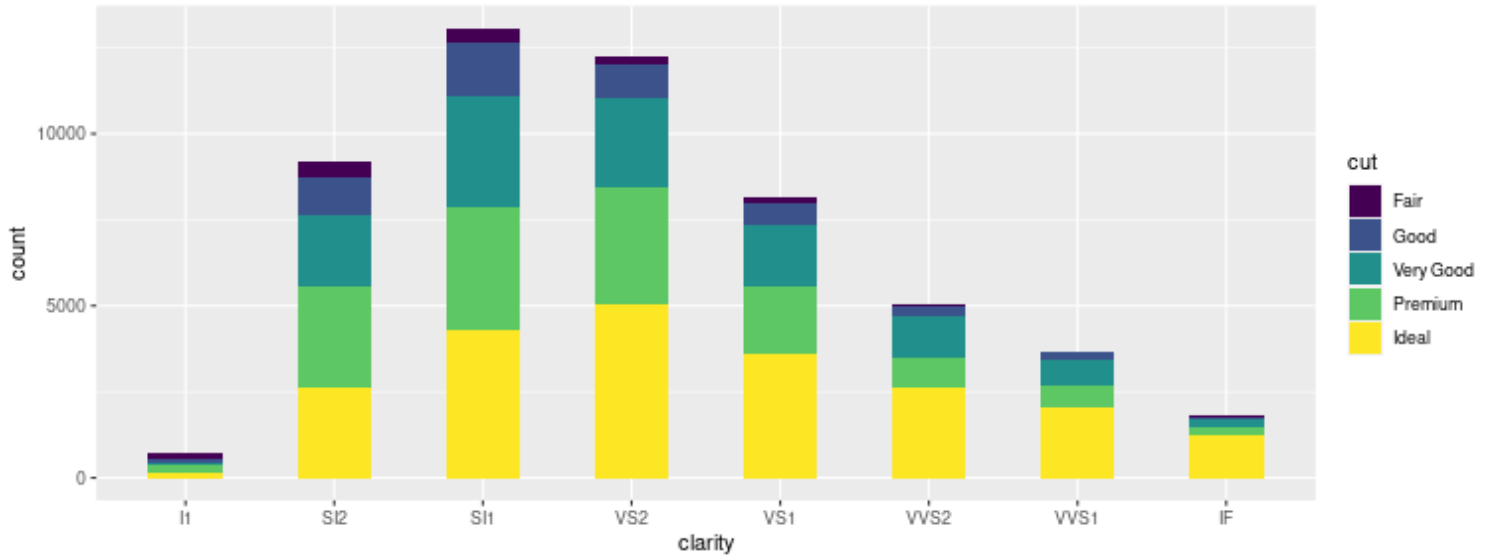
Note: the plain line indicates the mean abundance whereas the dashed lines represent twice the standard deviation.

Solution n°3

4. Barplots with `geom_bar()`

Basic (counts)

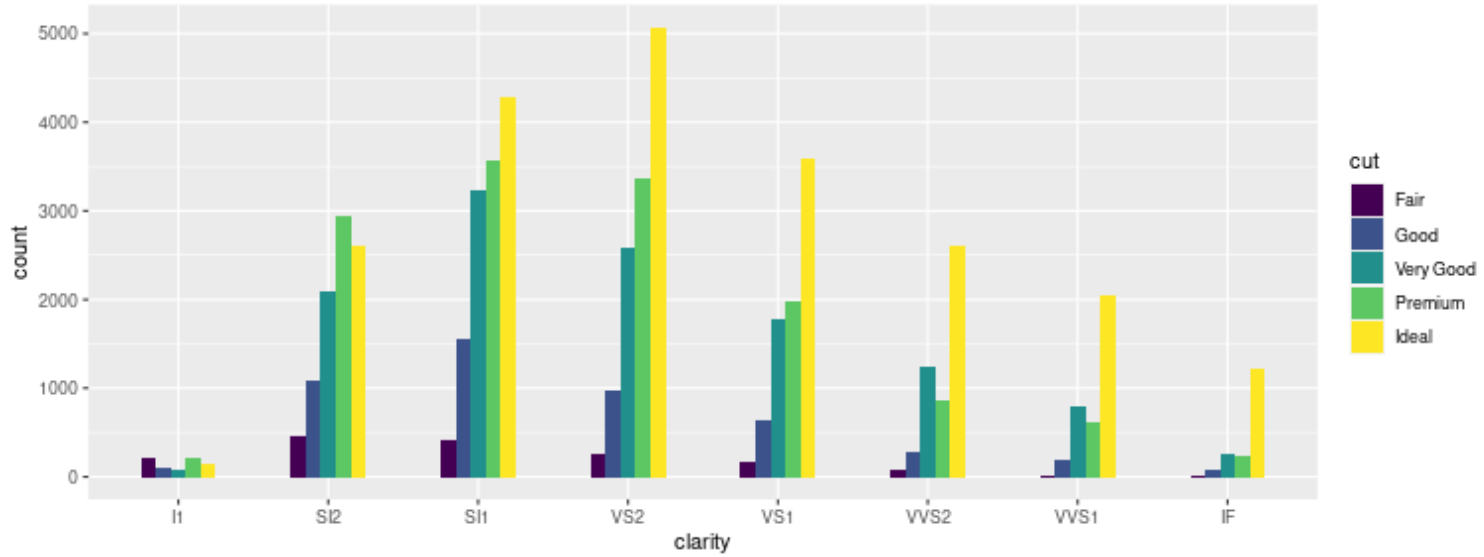
```
p <- ggplot(data = diamonds, mapping = aes(x = clarity))
p + geom_bar(mapping = aes(fill = cut), width = 0.5)
```




```
# p + geom_bar(fill='green') # Coloriage global
```

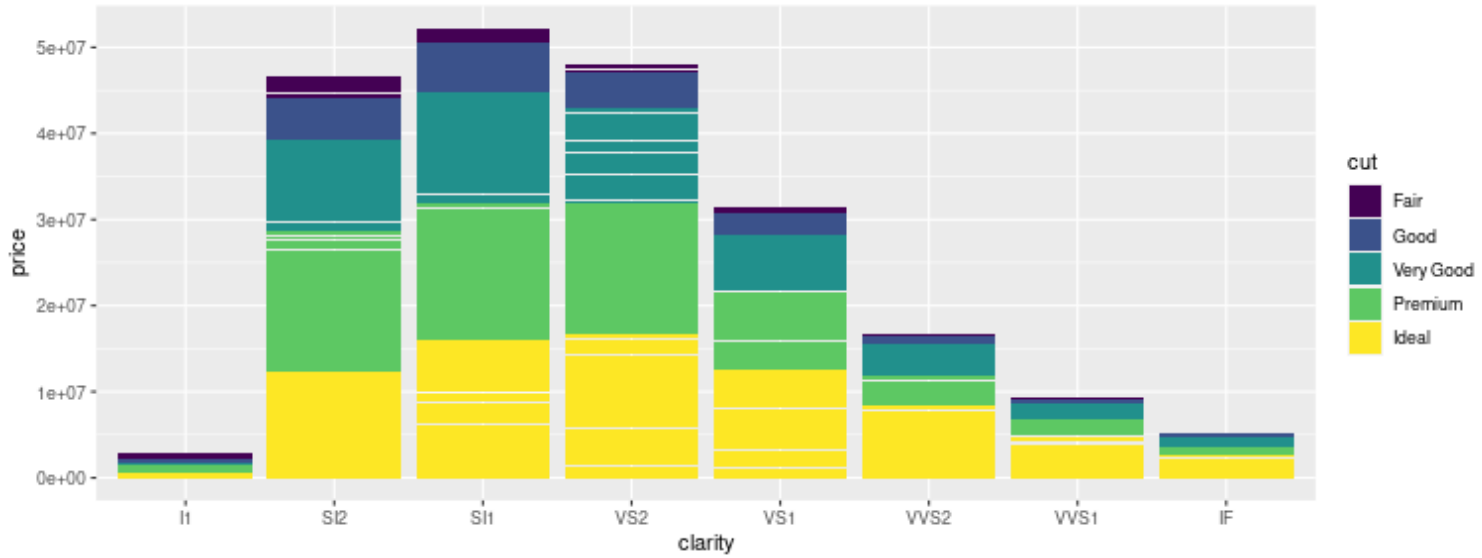
Basic (counts) side by side

```
p<-ggplot(data=diamonds, mapping=aes(x=clarity) )
p + geom_bar(mapping=aes(fill=cut), width=0.5,
             position="dodge")
```



Cumulated values

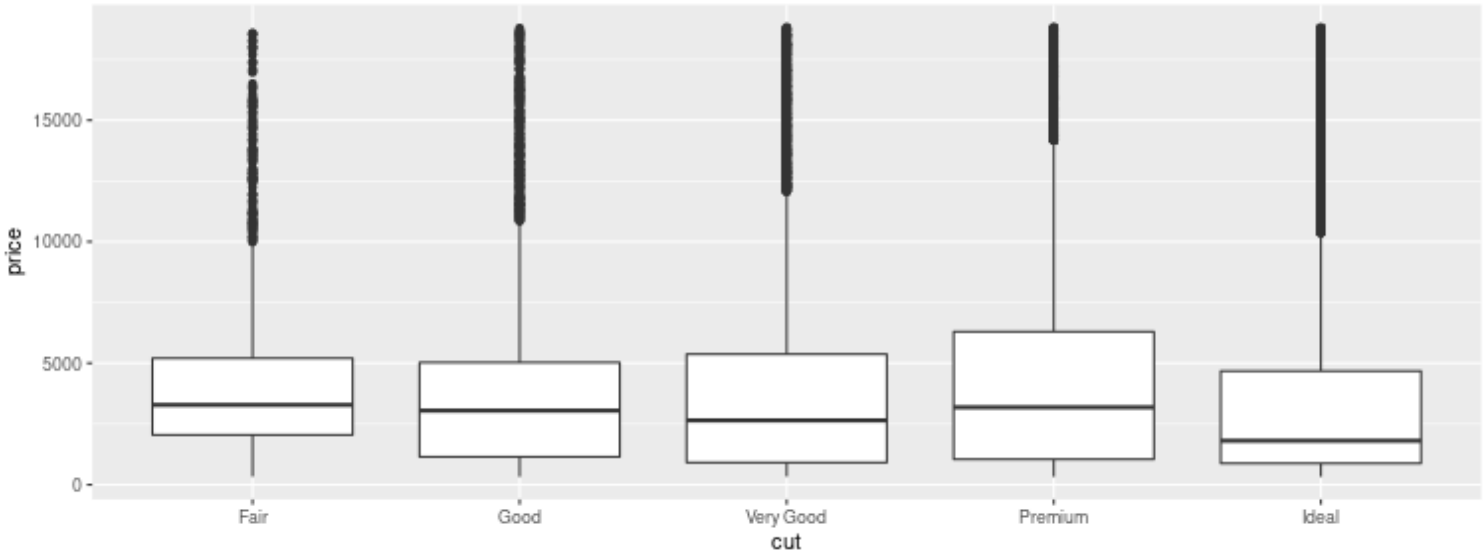
```
p <- ggplot(data = diamonds, mapping = aes(x = clarity, y = price))
p + geom_bar(mapping = aes(fill = cut), stat = "identity")
```



5. Boxplots with `geom_boxplot()`

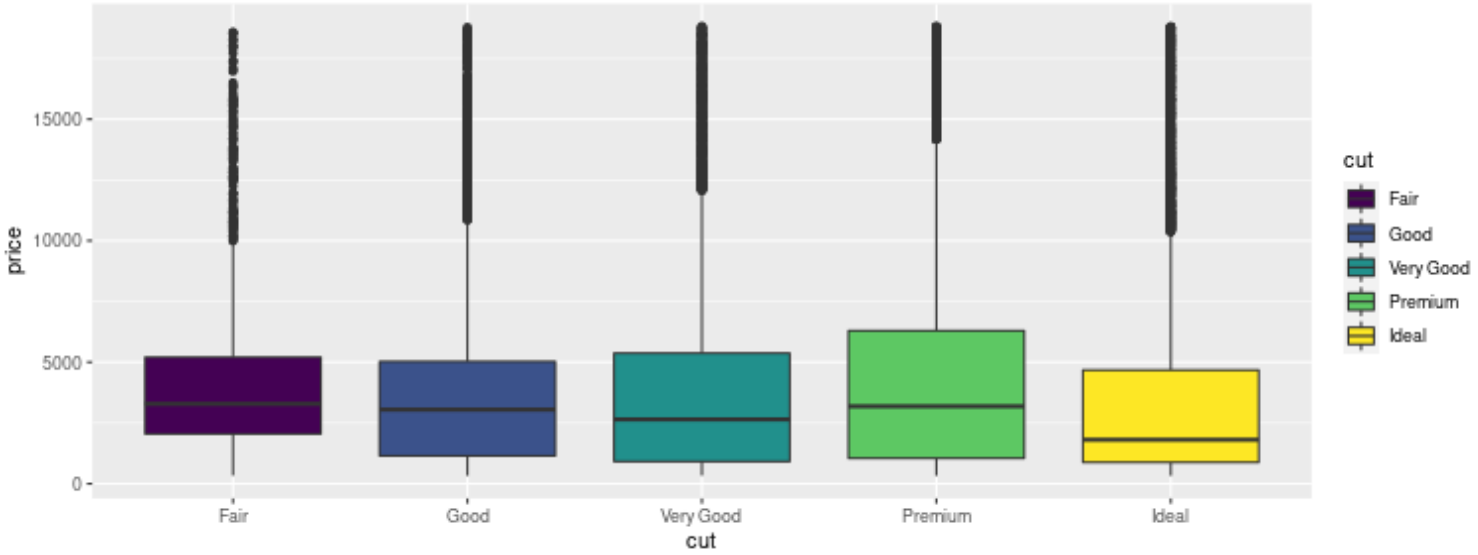
Basic

```
p <- ggplot(data = diamonds, mapping = aes(x = cut, y = price))
p + geom_boxplot()
```



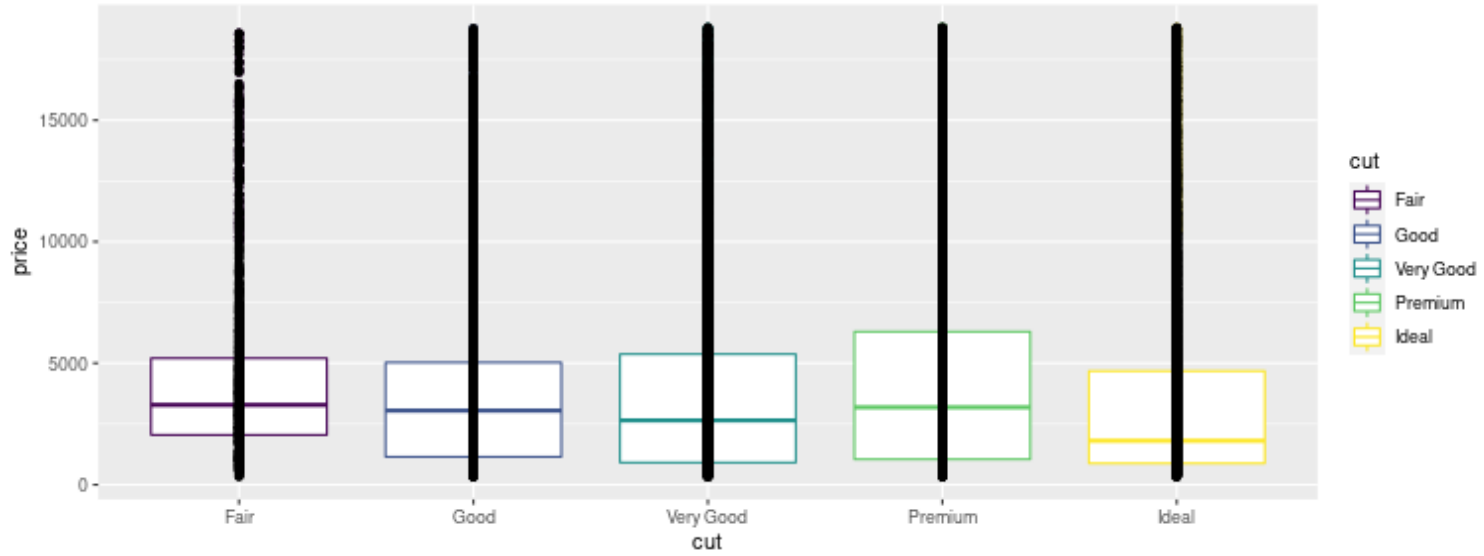
With colors

```
p + geom_boxplot(mapping = aes(fill = cut))
```



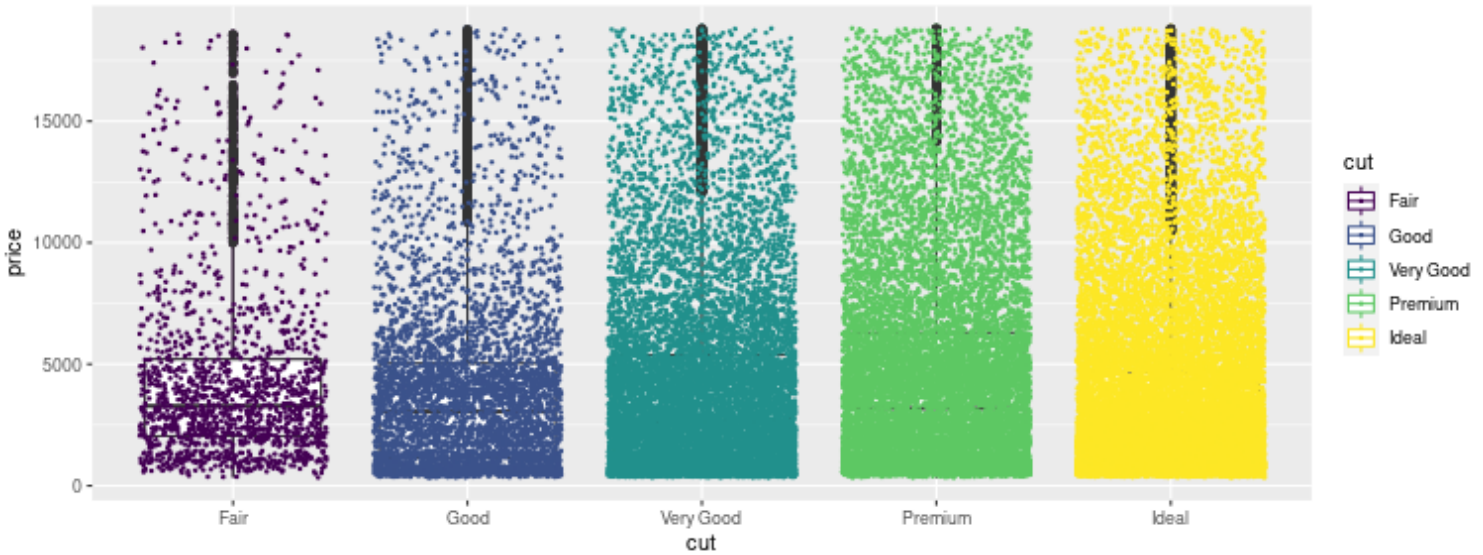
Boxplots with points

```
p <- p + geom_boxplot(mapping = aes(color = cut))
p + geom_point()
```



`geom_jitter` to add a small amount of random variation to the location of each point

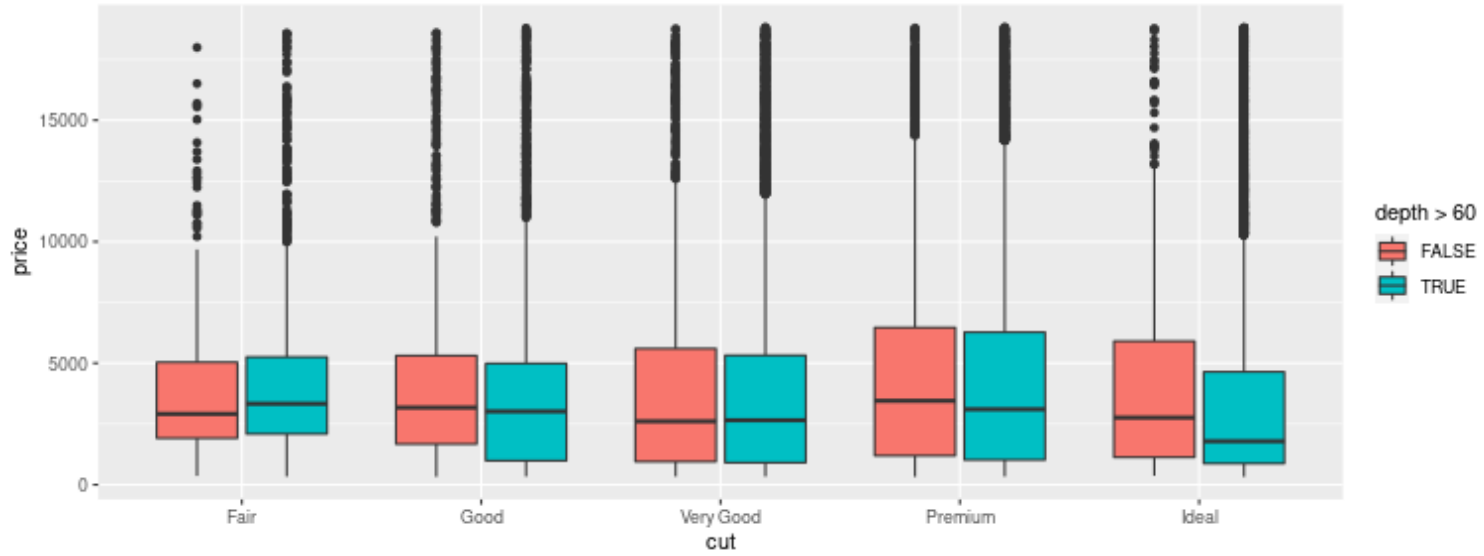
```
p <- p + geom_boxplot()
p + geom_jitter(size = 0.5, mapping = aes(color = cut))
```




```
# geom_jitter is a shortcut for geom_point(position='jitter')
```

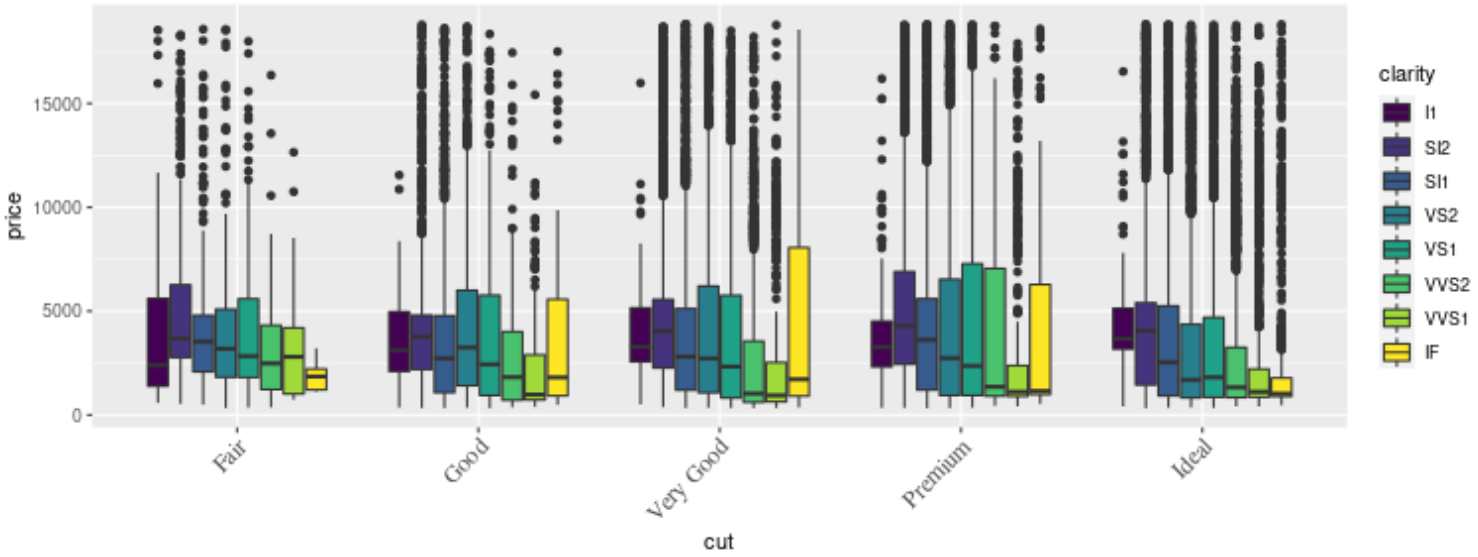
Boxplots wrt 2 factors

```
p <- ggplot(data = diamonds, mapping = aes(x = cut, y = price))
p + geom_boxplot(aes(fill = depth > 60))
```



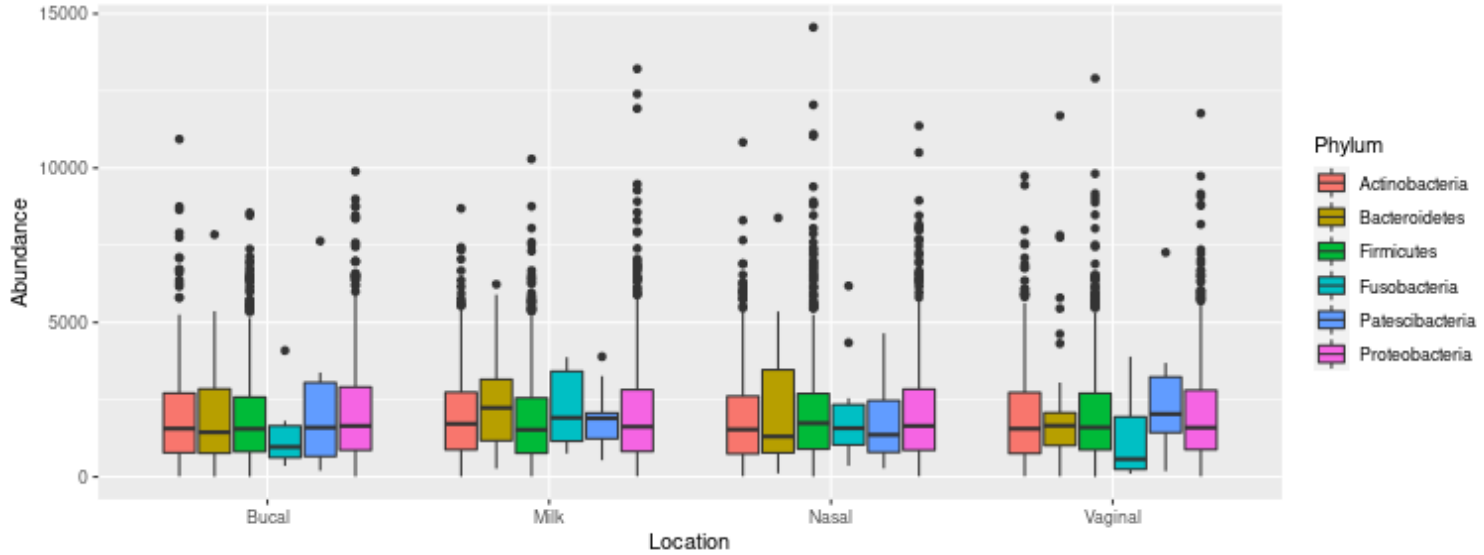
Another example:

```
p<- p+ geom_boxplot(mapping=aes(fill=clarity))
p + theme(axis.text.x = element_text(
  family="Times", angle = 45, size = 12, hjust=1))
```



Practice n°4

Reproduce the graphics below:



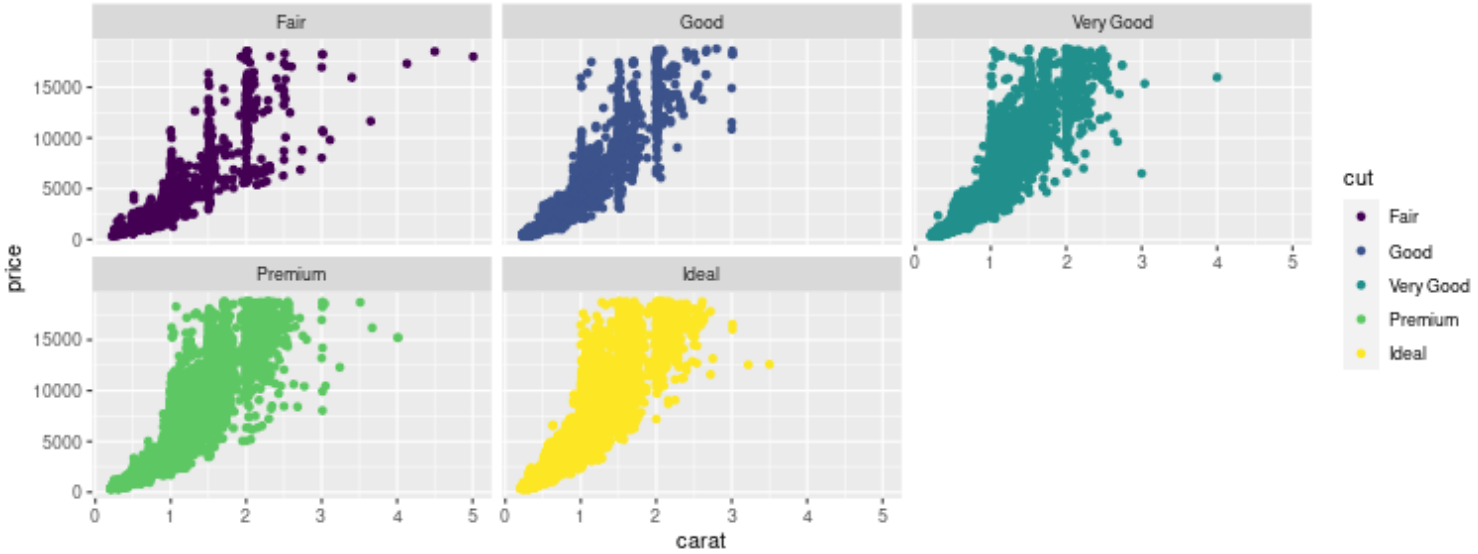
Clue: to change the text of the x-axis scale, use `scale_x_discrete(labels =)`

Solution n°4

6. Having several panels

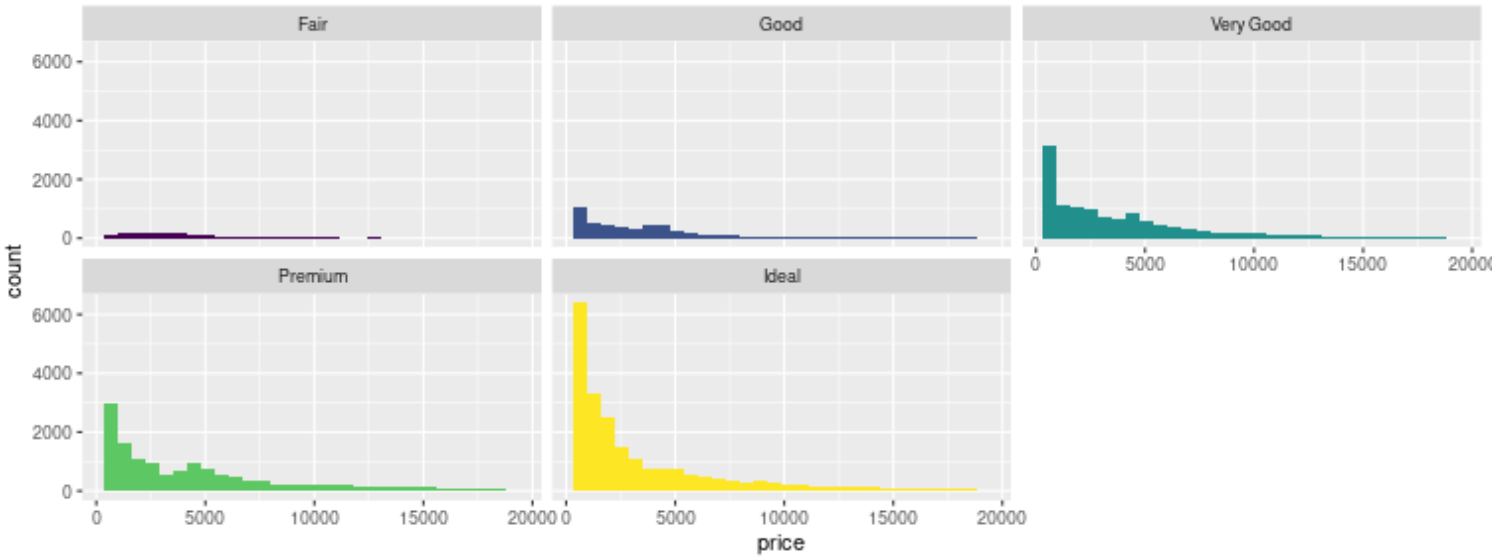
`facet_wrap()` for one variable

```
p <- ggplot(data = diamonds, mapping = aes(y = price, x = carat))
p <- p + geom_point(aes(color = cut))
p + facet_wrap(facets = ~cut)
```



Another example

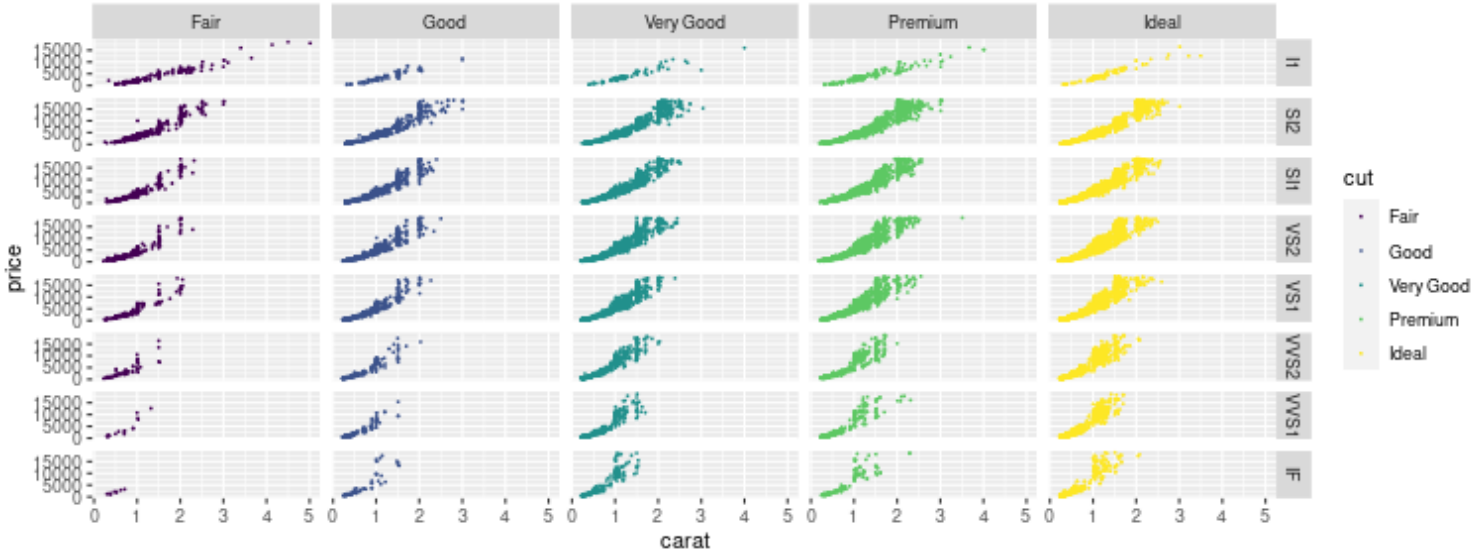
```
p <- ggplot(data = diamonds, mapping = aes(x = price, fill = cut))
p <- p + geom_histogram()
p + facet_wrap(facets = ~cut) + theme(legend.position = "none")
```



Note: `theme(legend.position="none")` to remove the legend.

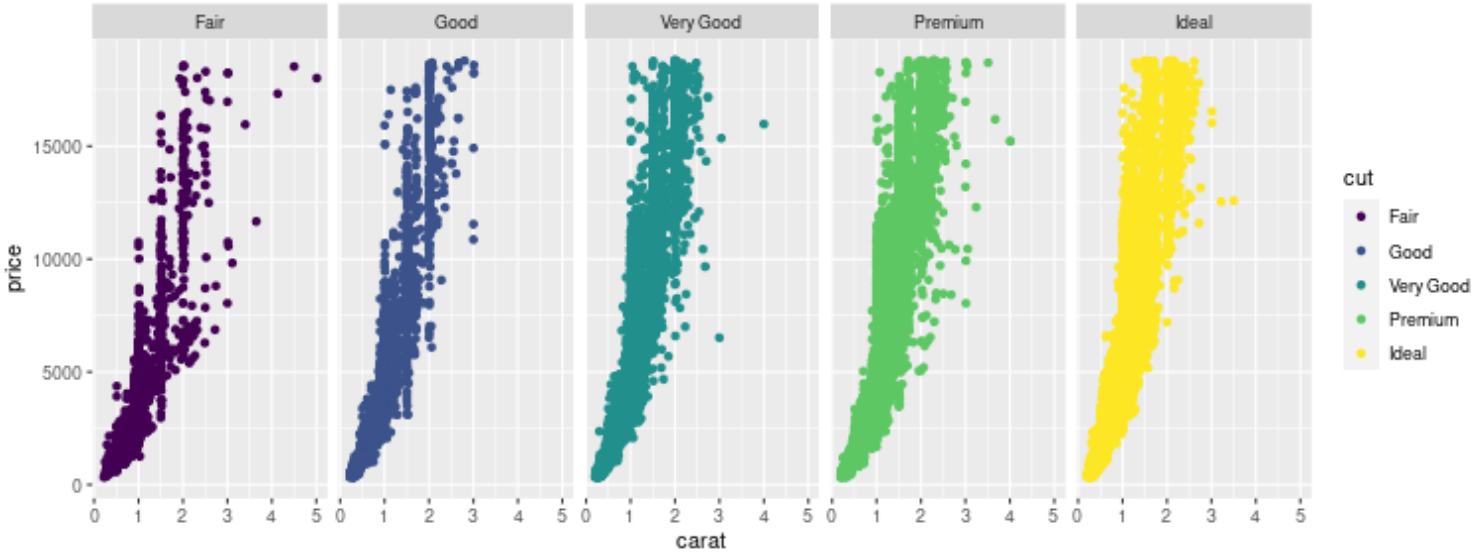
`facet_grid()` for 2 variables

```
p <- ggplot(data = diamonds, mapping = aes(y = price, x = carat))
p <- p + geom_point(aes(color = cut), size = 0.2)
p + facet_grid(facets = clarity ~ cut)
```



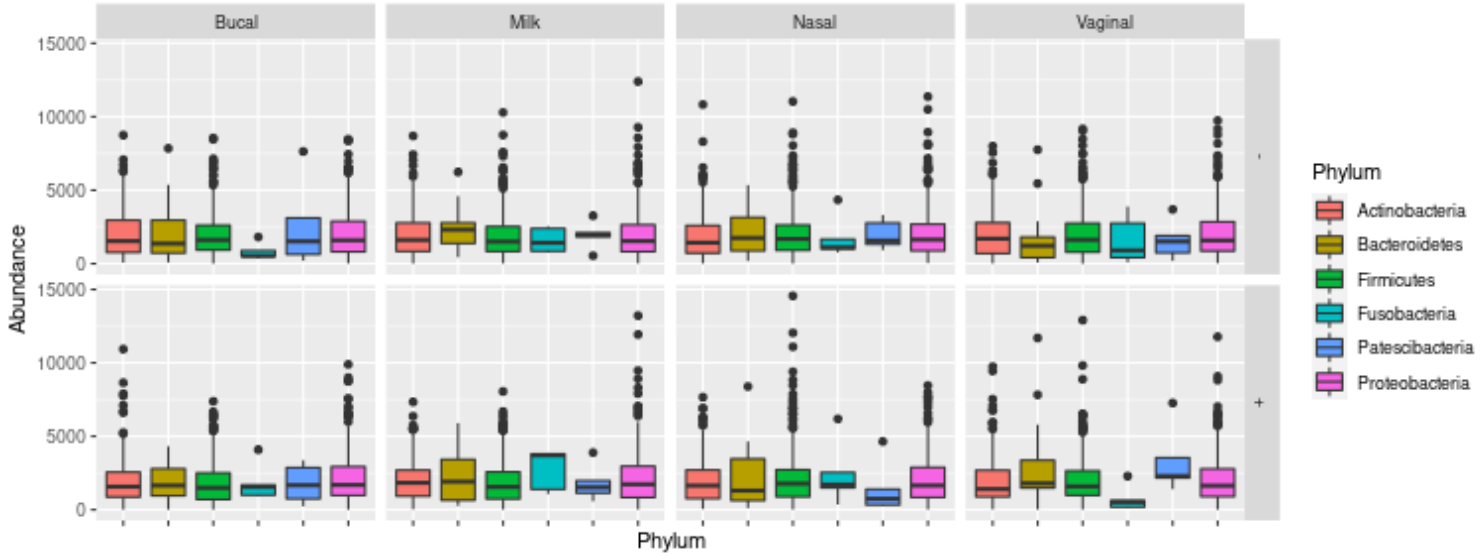
`facet_grid()` for one variable

```
p <- ggplot(data = diamonds, mapping = aes(y = price, x = carat))
p <- p + geom_point(aes(color = cut))
p + facet_grid(facets = ~cut)
```



Practice n°5

Reproduce the graphics below:



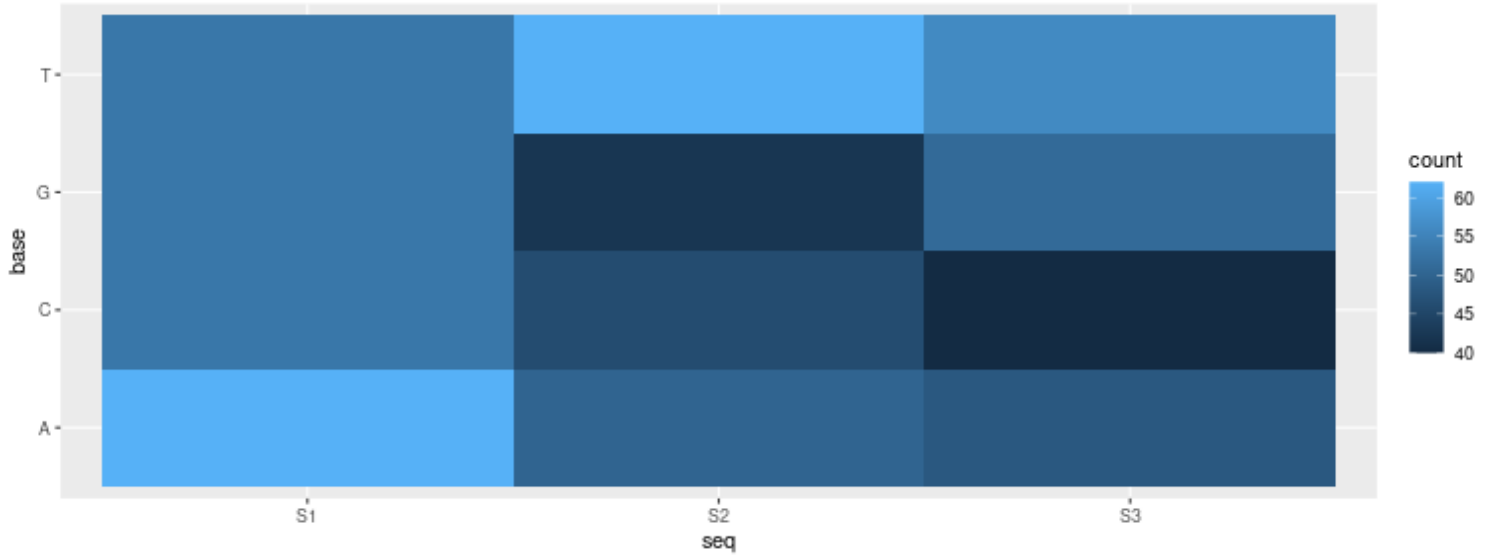
Clue: to change the text of the upper labels, use the option `labeller=labeller()` of `facet_grid`.

Solution n°5

7. Heatmap with `geom_tile()`

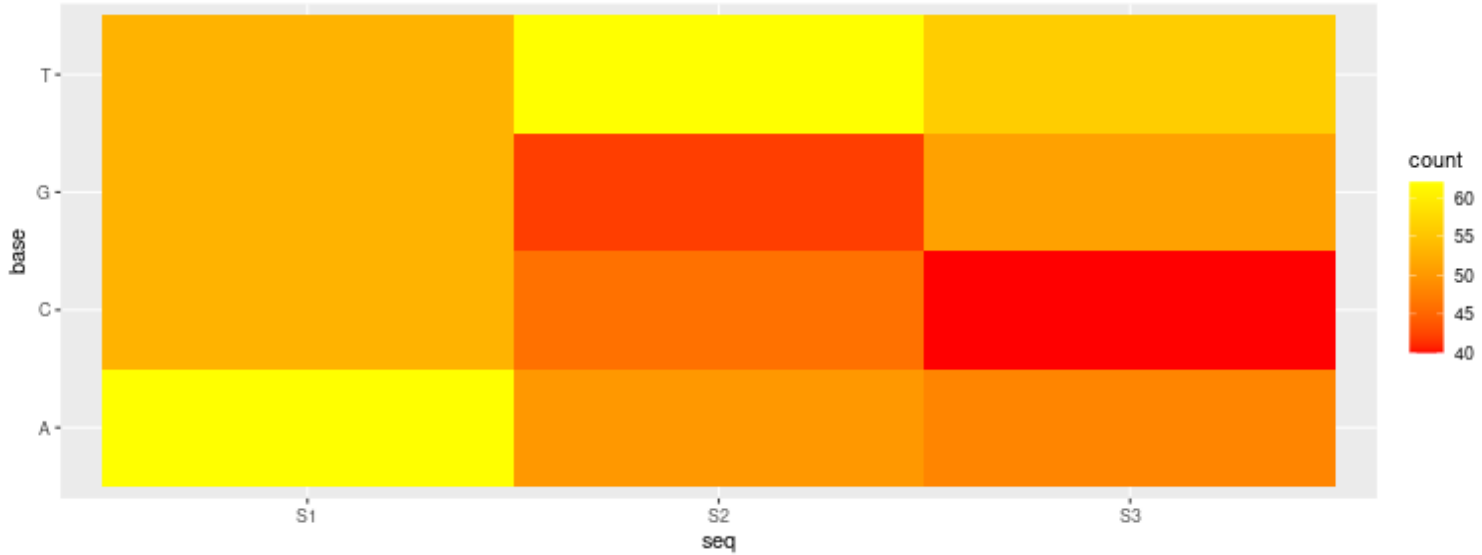
Basic

```
df <-data.frame(base=rep(c('A','C','G','T'), times=3),
                seq=rep(c('S1','S2', 'S3'), each=4),
                count=rpois(n=12,lambda=50))
p <- ggplot(data=df)
p <- p + geom_tile(aes(x=seq, y=base, fill=count))
p
```



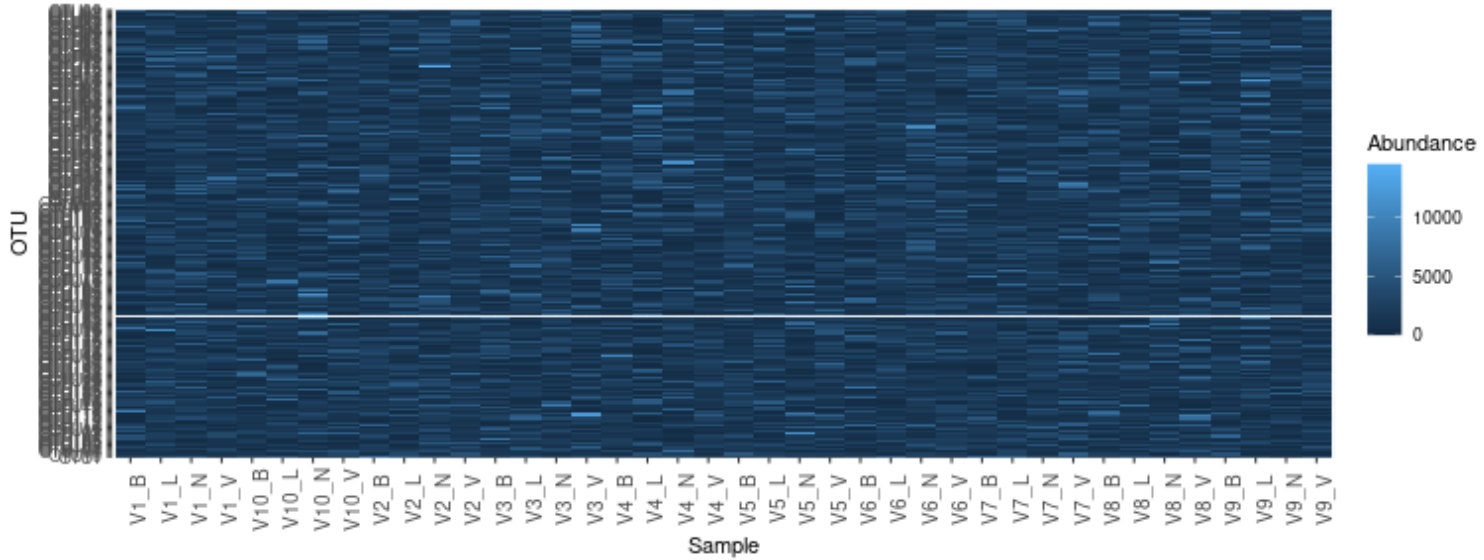
To change the gradient color:

```
p + scale_fill_gradient(low="red", high="yellow")
```



Practice n°6

Reproduce the graphics below:



Solution n°6

Save a graph

with ggplot2

```
ggsave(p, "mygraph.pdf", width = 5, height = 5, units = "cm")

ggsave(p, "mygraph.png", width = 5, height = 5, units = "cm")
```

without ggplot2

```
png("mygraph2.png")
print(p)
dev.off()
```

8. Appendix:

To change the scale on x-axis:

scale_x_continuous(breaks=)

Palette viridis

provides colour maps that are perceptually uniform in both colour and black-and-white. They are also designed to be perceived by viewers with common forms of colour blindness.

library(viridis)
scale_colour_viridis, scale_fill_viridis

Package ggpubr

facilitates the publication of beautiful ggplot2-based graphs.

References

- ggplot2 part of tidyverse: https://ggplot2.tidyverse.org/
- R for Data Science: https://r4ds.had.co.nz/
- Guide de démarrage ggplot2: https://bioinfo-fr.net/guide-de-demarrage-pour-ggplot2-un-package-graphique-pour-r

CHEAT SHEET : https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf