

Data Visualization Live Class

Tags	Data Visualization	Live Class
Class		
Finished Yet?	<input checked="" type="checkbox"/>	
Knowledge	Live Classes	

Data Viz Slide:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/49e1bf11-d6ac-48ca-93b8-bfe71e98fee9/Data_Viz_Slide.pdf

Chinook:

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/005f84e8-6664-4a0e-b417-8ced557fd17b/chinook.db>

-การใช้ RStudio (Posit Desktop) อย่าลืมโหลดภาษา R ลงเครื่องด้วย

-Library ที่ต้องโหลด

1. tidyverse
2. RSQLite
3. RPostgreSQL
4. lubridate
5. janitor

-เราสามารถใช้ R ในการ Query ข้อมูลใน Database (แบบ SQL) ได้ และเราสามารถเก็บ Table ที่เรา Query ข้อมูลไว้ในตัวแปรได้ด้วย ทำให้สะดวกต่อการ Manipulate ข้อมูลด้วย dplyr

*ใช้ R ควบคู่กับ SQL จะยืดหยุ่นกว่าการใช้ Pure SQL และเราสามารถใช้ function เหล่านี้กับทุก Database

```
library(tidyverse)
library(RSQLite) #DBI
library(RPostgreSQL)
library(lubridate)
library(janitor)

##Connect Database
connection <- dbConnect(SQLite(), "chinook.db")

##List table name
```

```

dbListTables(connection)

##List fields in a table
dbListFields(connection, "customers")

##Query data by writing SQL Queries
df <- dbGetQuery(connection, "SELECT * FROM customers LIMIT 10")

df %>%
  select(FirstName, LastName)

clean_df <- clean_names(df)
View(clean_df)

##Write JOIN syntax
df2 <- dbGetQuery(connection, "SELECT * FROM albums, artists
                                WHERE albums.artistid = artists.artistid") %>%
  clean_names()

##Write a table
dbWriteTable(connection, "cars", mtcars)
dbListTables(connection)

dbGetQuery(connection, "SELECT * FROM cars LIMIT 5")

##Remove a table
dbRemoveTable(connection, "cars")

##Close Connection
dbDisconnect(connection)

```

-PostgreSQL เป็นหนึ่งใน SQL ที่ได้รับความนิยมมากที่สุด

ElephantSQL (PostgreSQL as a Service): <https://www.elephantsql.com/>

-การเลือก Database ควรเลือก Database ที่อยู่ใกล้กับ User มากที่สุด เช่น User ของเรารอยู่ที่ญี่ปุ่น ที่ควรตั้ง Database ที่ Region Tokyo เป็นต้น

Connecting PostgreSQL with R:

```

##Connecting with PostgreSQL
con <- dbConnect(PostgreSQL(),
                 host = "floppy.db.elephantsql.com",
                 port = 5432,
                 user = "ujcgydew",
                 pass = "vvh5YMY6I6i6I8FxuemPK_dvVafio2B8",
                 dbname = "ujcgydew")

#Write table
dbWriteTable(con, "cars", mtcars %>% slice(1:5))

#List table
dbListTables(con)

#Get query
dbGetQuery(con, "SELECT * FROM cars")

#Disconnect
dbDisconnect(con)

```

Working with date

```

#Working With Date
library(lubridate)
library(tidyverse)

#YYYY-MM-DD
date_df <- data.frame(
  x = c(
    "2023-02-25",
    "2023-02-26",
    "2023-02-27",
    "2023-02-28",
    "2023-03-01"))

date_df %>%
  mutate(date_x = ymd(x),

```

```

year = year(date_x),
month = month(date_x, label = TRUE, abbr = FALSE),
day = day(date_x),
wday = wday(date_x, label = TRUE, abbr = FALSE),
week = week(date_x))

#Excel default USA date
date_df <- data.frame(
  x = c(
    "02/25/2023",
    "02/26/2023",
    "02/27/2023",
    "02/28/2023",
    "03/01/2023"))

date_df %>%
  mutate(date_x = mdy(x),
        year = year(date_x),
        month = month(date_x, label = TRUE, abbr = FALSE),
        day = day(date_x),
        wday = wday(date_x, label = TRUE, abbr = FALSE),
        week = week(date_x))

#Messy date format
date_df <- data.frame(
  x = c(
    "Feb 2023 - 25",
    "Feb 2023 - 26",
    "Feb 2023 - 27",
    "Mar 2023 - 9",
    "April 2023 - 1"))

date_df %>%
  mutate(date_x = myd(x),
        year = year(date_x),
        month = month(date_x, label = TRUE, abbr = FALSE),
        day = day(date_x),
        wday = wday(date_x, label = TRUE, abbr = FALSE),
        week = week(date_x))

convert_thai_to_eng_date <- function(year) {
  return(year - 543)
}

```

*ในว่า Date Format จะเลือกให้โดย自动ให้ Lubridate เพื่อแก้ไขและ Extract วันเดือนปีอุปกรณ์ได้ เนื้อหาทำด้วย Google Sheets / Excel

Skill Stacking > Specialization (การบูรณาการความรู้หลายเรื่องสำคัญกว่ารู้ลึกเรื่องเดียว)

ทำให้ได้ทุกรสื่อสารในชีวิตมากกว่าเก่งแค่เรื่องเดียว

Data Visualization

- เราใช้ library ชื่อ ggplot2 เพื่อใช้ในการสร้าง Chart ที่ทำได้ง่ายกว่าและสวยงามกว่าการใช้ Base R
- ประเภทของข้อมูลมีผลต่อการเลือกใช้ Chart
- Histogram = The most common graph to show data distribution
- ggplot ไม่สามารถสร้าง 3D Chart ได้ สร้างได้แต่ 2D
- เราสามารถเก็บ chart ไว้ในตัวแปรได้

```

library(tidyverse)

#Data Viz
#ggplot = grammar of graphics

#Base R (Not beautiful and hard to use):
plot(mtcars$mpg, mtcars$hp, pch = 16, col = 'red')

boxplot(mtcars$mpg)

t1 <- table(mtcars$am)
barplot(t1)

hist(mtcars$mpg)

#ggplot (beautiful and easy to use):
ggplot(data = mtcars,
       mapping = aes(x = mpg)) +
  geom_histogram(bins = 10)

```

```

ggplot(data = mtcars,
       mapping = aes(x = mpg)) +
  geom_density()

ggplot(data = mtcars,
       mapping = aes(x = mpg)) +
  geom_freqpoly()

p1 <- ggplot(mtcars, aes(mpg)) +
  geom_histogram(bins = 5)

p2 <- ggplot(mtcars, aes(hp)) +
  geom_histogram(bins = 10)

mtcars %>%
  filter(hp <= 200) %>%
  count()

#Summary table before creating a bar chart
mtcars <- mtcars %>%
  mutate(am = ifelse(am == 0, "Auto", "Manual"))

View(mtcars)

#Approach 1 - Summary table + geom_col()
t2 <- mtcars %>%
  mutate(am = ifelse(am == 0, "Auto", "Manual")) %>%
  count(am)

ggplot(t2, aes(x = am, y = n)) +
  geom_col()

#Approach 2 - geom_bar()
ggplot(mtcars, aes(am)) +
  geom_bar()

#Two variables (numeric)
#Scatter plot
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(col = 'red', size = 5)

```

*geom_col() ใช้กับ Summary Table ส่วน geom_bar() ใช้กับ Database ได้เลย

Diamonds database

```

library(tidyverse)

#Data frame -> diamonds
#Ordinal factor
temp <- c("high", "med", "low", "high")
temp <- factor(temp, levels = c("low", "med", "high"),
               ordered = TRUE)

#Categorical factor
gender <- c("m", "f", "nb")
gender <- factor(gender)

glimpse(diamonds)

#Frequency table
diamonds %>% count(cut, color, clarity)

#Sample
set.seed(42)
diamonds %>%
  sample_n(5)

diamonds %>%
  sample_frac(0.1)

diamonds %>%
  slice(500:510)

#Relationship (Pattern)
p3 <- ggplot(diamonds %>% sample_n(500), aes(carat, price)) +
  geom_point() +
  geom_smooth(method = "loess") +
  geom_rug()

```

```

#Setting VS Mapping
colors()

#Setting (Manually set options)
ggplot(diamonds, aes(price)) +
  geom_histogram(bins = 100, fill = 'salmon')

ggplot(diamonds %>% sample_n(500),
       aes(carat, price)) +
  geom_point(size = 5, alpha = 0.2, col = 'red')

#Mapping (Occurs in aesthetic function)
ggplot(diamonds %>% sample_n(500),
       mapping = aes(carat, price, col = cut)) +
  geom_point(size = 5, alpha = 0.8) +
  theme_minimal() +
  labs(
    title = "Relationship between carat and price",
    x = "Carat",
    y = "Price (USD)",
    subtitle = "We found a positive relationship",
    caption = "Data source: diamonds (ggplot2)"
  ) +
  scale_color_brewer(type = "qual", palette = 1)

#Map color scale
ggplot(mtcars, aes(hp, mpg, col = wt)) +
  geom_point(size = 5, alpha = 0.7) +
  theme_minimal() +
  scale_color_gradient(low = "green", high = "red")

#Facet
ggplot(diamonds %>% sample_n(500),
       aes(carat, price)) +
  geom_point(alpha = 0.6) +
  geom_smooth(col = "red", fill = "gold") +
  theme_minimal() +
  facet_wrap(~cut)

ggplot(diamonds %>% sample_n(500),
       aes(carat, price)) +
  geom_point(alpha = 0.6) +
  geom_smooth(col = "red", fill = "gold") +
  theme_minimal() +
  facet_grid(cut ~ color)

library(patchwork)
library(ggplot2)

p1 <- qplot(mpg, data = mtcars, geom = "histogram", bins = 10)
p2 <- qplot(hp, mpg, data = mtcars, geom = "point")
p3 <- qplot(hp, data = mtcars, geom = "density")

p1 + p2 + p3
(p1 + p2) / p3
p1 / p2 / p3
p1 / (p2 + p3)

```

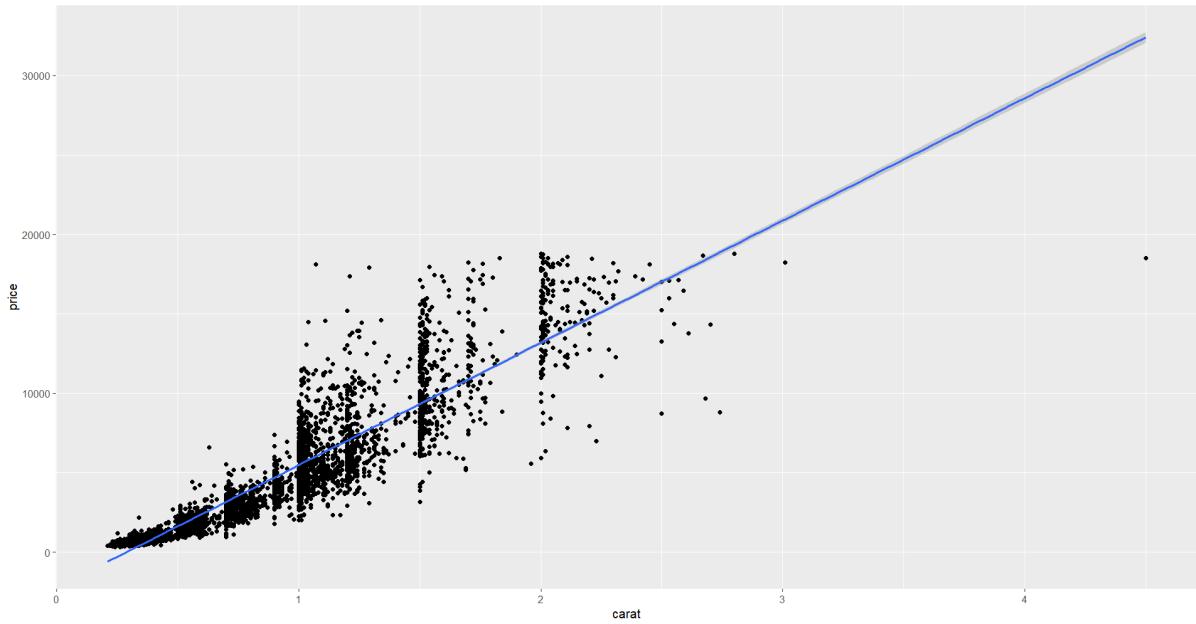
-ถ้าเจอข้อมูลมาก ๆ ให้ sample ข้อมูลมาค่อน เพื่อให้พอเห็นแนวโน้มของข้อมูลและเสียเวลาบ่นอยกว่าการเรียกใช้ข้อมูลทั้งหมด

*ggplot สามารถ plot graph ได้มากกว่า 1 graph ในรูปเดียวคับด้วยการ + กับ

```

ggplot(diamonds %>% sample_n(5000), aes(carat, price)) +
  geom_point() +
  geom_smooth(method = "lm")

```



*การตีความ chart จะต้องเข้าใจ business context เช่น ยิ่งกะรตของเพชรสูง ราคา ก็ควรจะมีแนวโน้มเพิ่มขึ้น ไม่ใช่ลดลง เป็นต้น
-เราสามารถดึง Hex code มาใช้เติมสี chart ใน R ได้

```
ggplot(diamonds %>% sample_n(500),
       mapping = aes(carat, price, col = cut)) +
  geom_point(size = 5, alpha = 0.8) +
  theme_minimal() +
  labs(
    title = "Relationship between carat and price",
    x = "Carat",
    y = "Price (USD)",
    subtitle = "We found a positive relationship",
    caption = "Data source: diamonds (ggplot2)"
  )
```



*Setting = เราต้องเซ็ตอพชั่นต่าง ๆ เอง เช่นสีหรือรูปลักษณ์ แต่ Mapping = การเซ็ตอพชั่นอิงจาก column ใน Data Frame ซึ่งเราจะกำหนดในส่วนของ mapping (aes)

-เราสามารถเปลี่ยน Palette สีของ การ Mapping ได้ด้วย scale เช่น:

```
#Manual  
scale_color_manual(values = c(  
  "red", "blue", "green", "gold", "salmon"  
)  
  
#Auto  
scale_color_brewer(type = "qual", palette = 1)
```

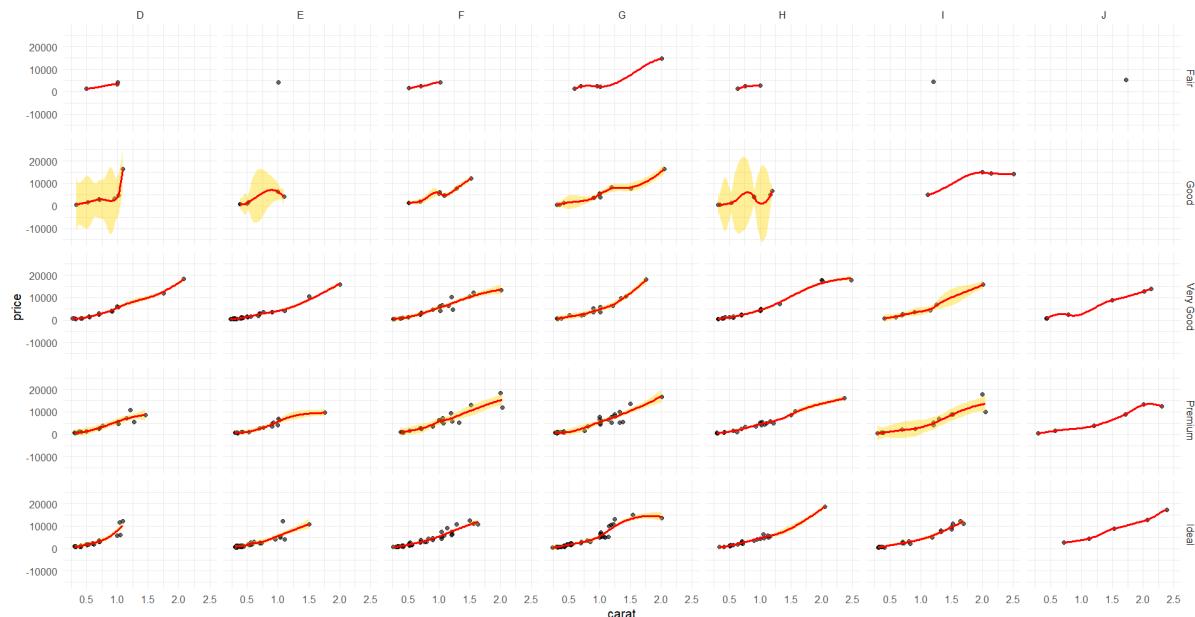
เว็บ Color Palette Generator: <https://coolors.co/>

-การเปลี่ยนสี Chart เป็นสีพาสเทลจะทำให้ Chart ของเราง่ายขึ้น

-facet ใช้แบ่ง Chart ของเรางานค่าใน column ได้ เช่น:

```
facet_wrap( ~cut)
```

```
#facet_grid()  
ggplot(diamonds %>% sample_n(500),  
       aes(carat, price)) +  
  geom_point(alpha = 0.6) +  
  geom_smooth(col = "red", fill = "gold") +  
  theme_minimal() +  
  facet_grid(cut ~ color)
```



-Patchwork ช่วยในการรวม chart เข้าด้วยกัน

-qplot() = Chart แบบย่อ (Quickplot)

*พัฒนาความรู้ อยู่ที่การเลือกใช้เครื่องมือให้เหมาะสมกับงาน

Markdown

-เราสามารถสร้าง Report ได้ด้วย library ชื่อ rmarkdown

-# คือการทำให้เป็น Heading มี 6 ขนาด (ตั้งแต่ # ถึง ##### ยัง # มาก ขนาดยังน้อย)

- Era สามารถ knit Report ของเราระบบเป็น HTML หรือ PDF ได้

- การ Insert R code chunk ให้กดปุ่มสีเขียว และเลือก R

R Markdown cheat sheet:

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/432aed0f-b733-4aa4-8a24-3903331e9e3c/rmarkdown-cheatsheet.pdf>

Homework

Explore Data Frame `diamonds` and create 5 visualizations.

```
---
```

```
title: "Homework Data Viz"
author: "Poorin Pitakratananukool"
date: "`r Sys.Date()`"
output:
  html_document: default
---
```

```
Today we learn a few topics in R:
```

- databases
- working with date
- ggplot2
- R markdown

```
## Homework:
```

```
Explore Data Frame `diamonds` and create 5 visualizations.
```

```
### Chart 1 - Relationship between carat and price
```{r message=FALSE, warning=FALSE}
library(tidyverse)
head(diamonds)
```

Create my first chart!
```{r}
set.seed(69)
ggplot(diamonds %>% sample_n(500),
 aes(carat, price)) +
 geom_point() +
 geom_smooth() +
 theme_minimal()
```

We found a positive relationship between two variables.
```

```
Another chart!
```{r}
ggplot(diamonds, aes(cut, fill = color)) +
 geom_bar(position = "fill") +
 theme_minimal()
```

We found that diamonds with color [J] have the least quantity compared to other colors of diamonds.
```

```
[R Markdown](https://rmarkdown.rstudio.com/)
```

```
Add an image to this report
```

```
![New Image](https://images.unsplash.com/photo-1504868584819-f8e8b4b6d7e3?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8)

### Chart 2 - Relationship between color and price
```{r}
set.seed(69)
ggplot(diamonds %>% sample_frac(0.1),
 aes(color, price)) +
 geom_col() +
 theme_minimal() +
 labs(
 title = "Relationship between color and price",
 x = "Color",
 y = "Price (USD)",
 subtitle = "We found that color G is the most expensive diamond of all colors.",
 caption = "Data source: diamonds (ggplot2)"
)
```

```

```

```{r}
set.seed(69)
ggplot(diamonds %>% sample_frac(0.1),
 aes(cut, fill = clarity)) +
 geom_bar(position = "fill") +
 theme_minimal() +
 labs(
 title = "Cut count for each clarity type",
 x = "Cut",
 y = "Count in Percentage",
 subtitle = "SI1 is the most common clarity for each type of cut, and IF is the least common.",
 caption = "Data source: diamonds (ggplot2)"
)
```

```{r}
set.seed(69)
ggplot(diamonds %>% sample_frac(0.1),
 aes(cut, price, fill = cut)) +
 geom_col() +
 theme_minimal() +
 labs(
 title = "Relationship between cut and price",
 x = "Cut",
 y = "Price (USD)",
 subtitle = "We found that Ideal cut is the most expensive of all cuts of diamonds.",
 caption = "Data source: diamonds (ggplot2)"
)
```

```{r message=FALSE, warning=FALSE}
library(patchwork)
p1 <- qplot(carat, data = diamonds %>% sample_frac(0.1), geom = "histogram", bins = 10)
p2 <- qplot(carat, price, data = diamonds %>% sample_frac(0.1), geom = "point")
p3 <- qplot(cut, price, data = diamonds %>% sample_frac(0.1), geom = "col")

((p1 + p2) / p3)
```

#### We found that:
1. The more carat a diamond has, the rarer it will be.
2. The more carat a diamond has, the more expensive it will be.
3. The cut quality of diamonds affects their price. The better quality diamonds are, the more expensive they will be.

#### In conclusion:
Diamonds with more carats are harder to find and are more expensive. Also, diamonds with better quality are more expensive.

```

C:\Users/asus\Desktop\R Programming Course\Data_Viz_Report.html Open in Browser Find Publish

Homework Data Viz

Poorin Pitakratananukool
2023-03-13

Today we learn a few topics in R:

- databases
- working with date
- ggplot2
- R markdown

Homework:

Explore Data Frame `diamonds` and create 5 visualizations.

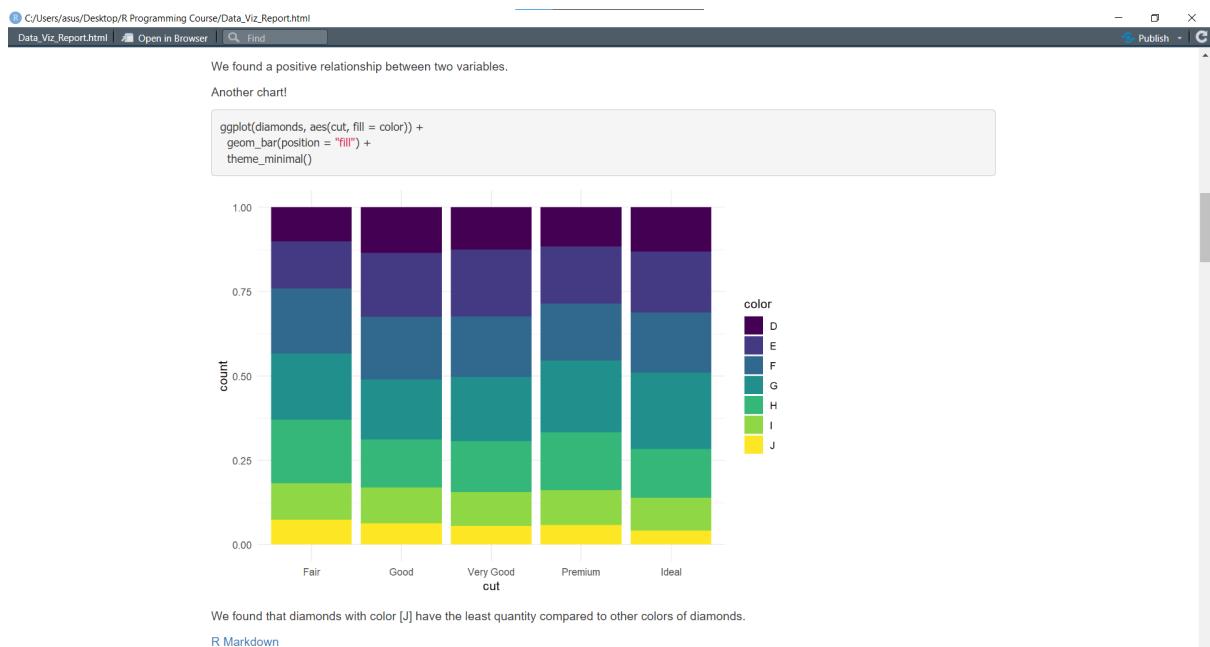
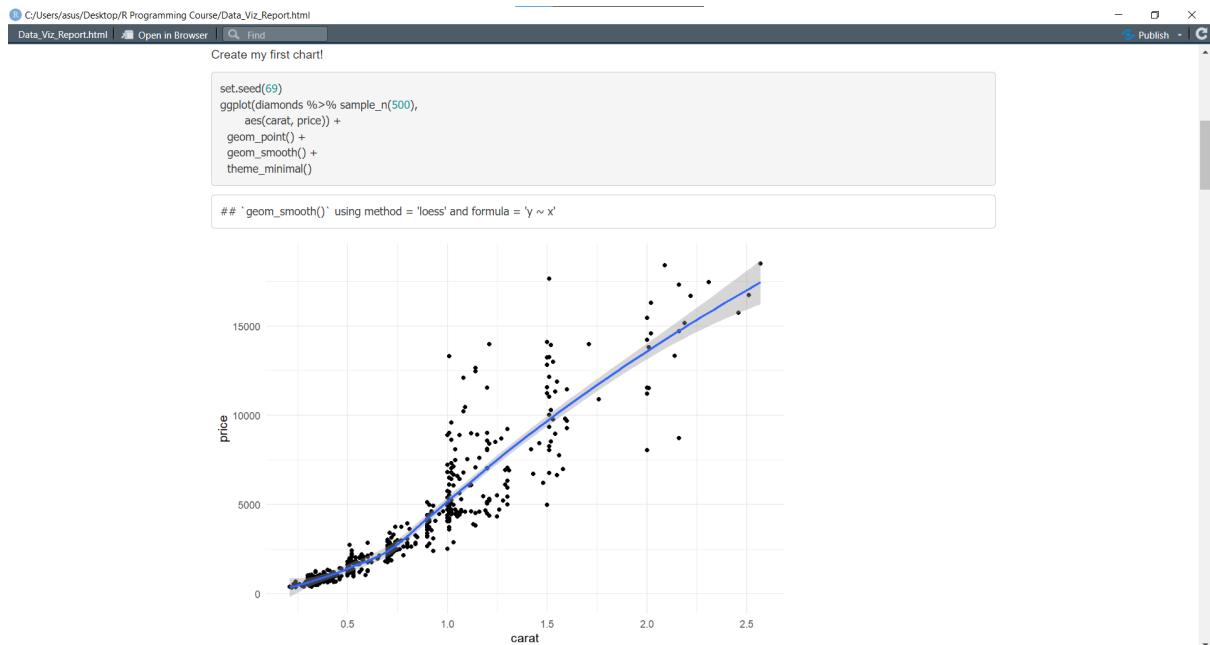
Chart 1 - Relationship between carat and price

```
library(tidyverse)
head(diamonds)
```

```
## # A tibble: 6 × 10
##   carat cut   color clarity depth table price     x     y
##   <dbl> <ord> <ord> <ord> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0.23 Ideal   E    SI2    61.5  55  326  3.95  3.98  2.43
## 2 0.21 Premium E    VS1    59.8  61  326  3.89  3.84  2.31
## 3 0.23 Good    E    VS1    56.9  65  327  4.05  4.07  2.31
## 4 0.29 Premium I    VS2    62.4  58  334  4.2  4.23  2.63
## 5 0.31 Good    J    SI2    63.3  58  335  4.34  4.35  2.75
## 6 0.24 Very Good J    VVS2   62.8  57  336  3.94  3.96  2.48
```

Create my first chart!

```
set.seed(69)
```



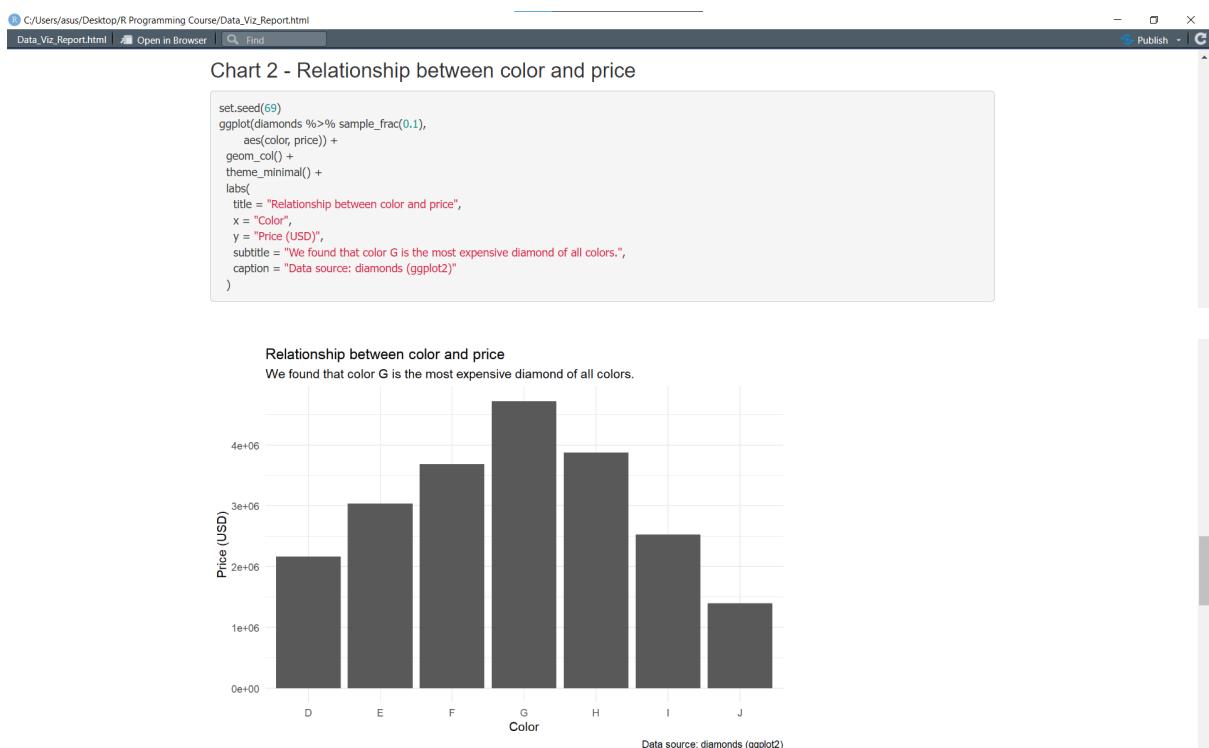
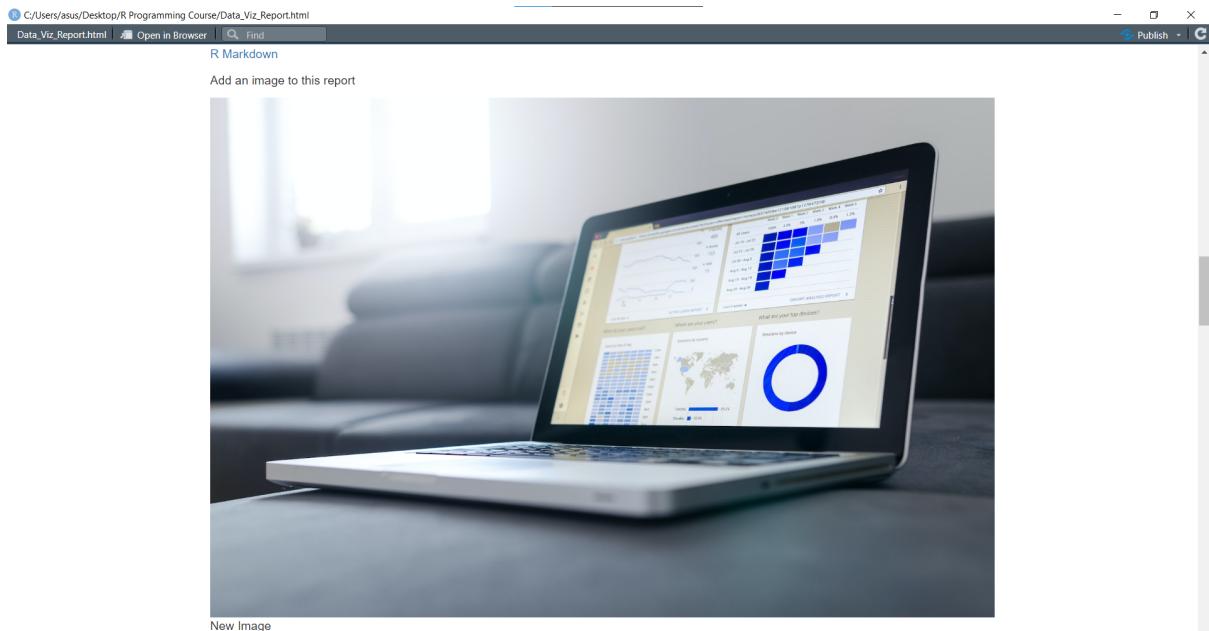


Chart 3 - Cut count for each clarity type

```
set.seed(69)
ggplot(diamonds %>% sample_frac(0.1),
       aes(cut, fill = clarity)) +
  geom_bar(position = "fill") +
  theme_minimal() +
  labs(
    title = "Cut count for each clarity type",
    x = "Cut",
    y = "Count In Percentage",
    subtitle = "S1 is the most common clarity for each type of cut, and IF is the least common.",
    caption = "Data source: diamonds (ggplot2)"
  )
```

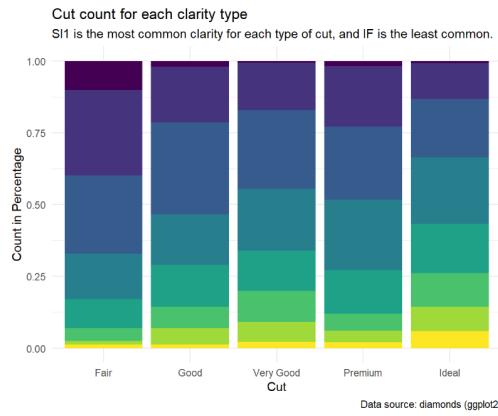


Chart 4 - Relationship between cut and price

```
set.seed(69)
ggplot(diamonds %>% sample_frac(0.1),
       aes(cut, price, fill = cut)) +
  geom_col() +
  theme_minimal() +
  labs(
    title = "Relationship between cut and price",
    x = "Cut",
    y = "Price (USD)",
    subtitle = "We found that Ideal cut is the most expensive of all cuts of diamonds.",
    caption = "Data source: diamonds (ggplot2)"
  )
```

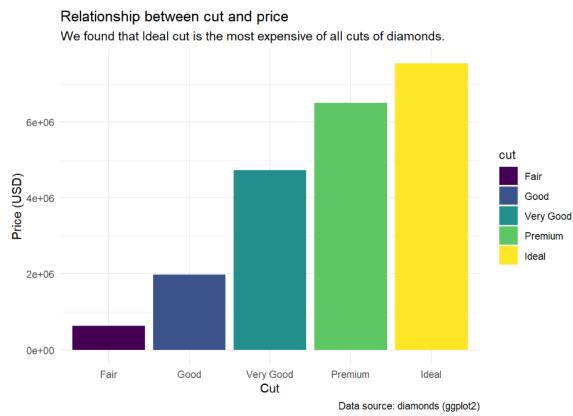
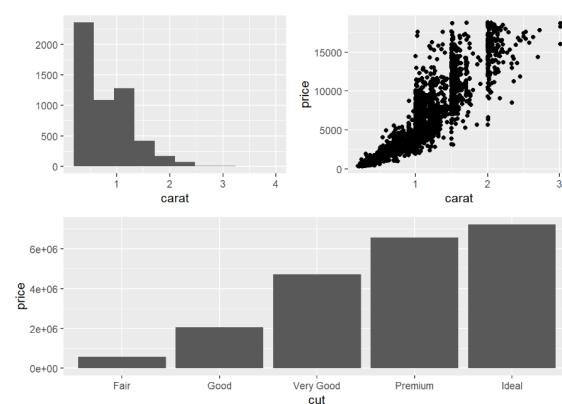


Chart 5 - Multiple charts in one board

```
library(patchwork)
p1 <- qplot(carat, data = diamonds %>% sample_frac(0.1), geom = "histogram", bins = 10)
p2 <- qplot(carat, price, data = diamonds %>% sample_frac(0.1), geom = "point")
p3 <- qplot(cut, price, data = diamonds %>% sample_frac(0.1), geom = "col")

((p1 + p2) / p3)
```



We found that:

1. The more carat a diamond has, the rarer it will be.
2. The more carat a diamond has, the more expensive it will be.
3. The cut quality of diamonds affects their price. The better quality diamonds are, the more expensive they will be.

In conclusion:

Diamonds with more carats are harder to find and are more expensive. Also, diamonds with better quality are more expensive.