IPS Instituto Politécnico de Setúbal

Session 3

Data Cleaning and EDA in R (Part 2)

Course Name:

Programming for Data Science

By Vala A. Rohani
vala.ali.rohani@estsetubal.ips.pt

Learning goals of this session:

By the end of this session, you will be able to do the following things:

- Being familiar with different techniques of data cleaning
- Know how to build different graphs in R

Data cleaning is one of the most important aspects of data science.

As a data scientist, you can expect to spend up to **80% of your time cleaning data**.

In this post you'll learn how to detect missing values using the tidyr and dplyr packages from the Tidyverse.

NA Stands for not available. NA is a placeholder for a missing value.

Your turn!
Try the following codes and see the results

```
# what is NA?


NA + 1
sum(c(NA, 1, 2))
median(c(NA, 1, 2, 3), na.rm = TRUE)
length(c(NA, 2, 3, 4))
3 == NA
NA == NA
TRUE | NA
```

NULL means no class (its class is NULL) and has length 0 so it does not take up any space in a vector.

Your turn!
Try the following codes and see the results

```
# what is NULL?


length(c(1, 2, NULL, 4))
sum(c(1, 2, NULL, 4))
x <- NULL
c(x, 2)
```

Missing Values

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Statistical analysis error

```
# ---- Statistical analysis error


age <- c(23, 16, NA)
mean(age)

## [1] NA

mean(age, na.rm = TRUE)
## [1] 19.5

```

Identify and remove the NA value

Missing Values

```
# --- Identify the NA value

complete.cases(age)

#or

is.na(age)


# --- Remove NA values

na.omit(age)

age

```

Data Cleaning > Graphs Principles > Graphs in R

recode missing values with the mean

Missing Values

```r
#--- Recode missing values with mean

x <- c(1:5, NA, 9:11, NA)
is.na(x)

df <- data.frame(col1 = c(1:3, NA),
                 col2 = c("this", NA,"is", "text"),
                 col3 = c(TRUE, FALSE, TRUE, TRUE),
                 col4 = c(2.5, 4.2, 3.2, NA),
                 stringsAsFactors = FALSE)
is.na(df)

# -- To identify the location of the NA.


which(is.na(x))
sum(is.na(df))

#or for data frame

colSums(is.na(df))


# recode missing values with the mean
x

x[is.na(x)] <- mean(x, na.rm = TRUE)

# do it for dataframe

df

df$col4[is.na(df$col4)] <- mean(df$col4, na.rm = TRUE)

```

Data Cleaning > Graphs Principles > Graphs in R

showing complete and incomplete observations

Missing Values

```
#-- For complete observations.


df

complete.cases(df)


# subset with complete.cases to get complete cases
df[complete.cases(df), ]

# or use na.omit() to get same as above
na.omit(df)

# or subset with `!` operator to get incomplete cases
df[!complete.cases(df), ]
```

What can we do with missing values in datasets?

Missing Values

1. **Deleting the observations**
   - Have sufficient data points, so the model doesn't lose power.
   - Not to introduce bias (meaning, disproportionate or non-representation of classes).

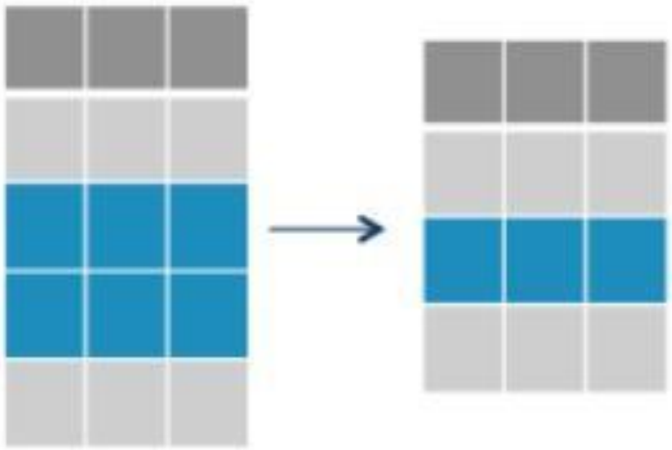
2. **Deleting the variable**
   - When we have lots of missing values for a specific variable

3. **Imputation with mean / median / mode**
   - Replacing the missing values with the mean / median / mode is a crude way of treating missing values. Depending on the context, like if the variation is low or if the variable has low leverage over the response, such

4. **Prediction**
   - Prediction is most advanced method to impute your missing values and includes different approaches such as: kNN Imputation, rpart, and mice.

Duplicate values

Remove Duplicate Data in R



duplicated(): *Identify duplicate elements (R base)*
unique(): *Keep only unique elements (R base)*
distinct(): *Efficient solution to remove duplicate in a data table (dplyr)*

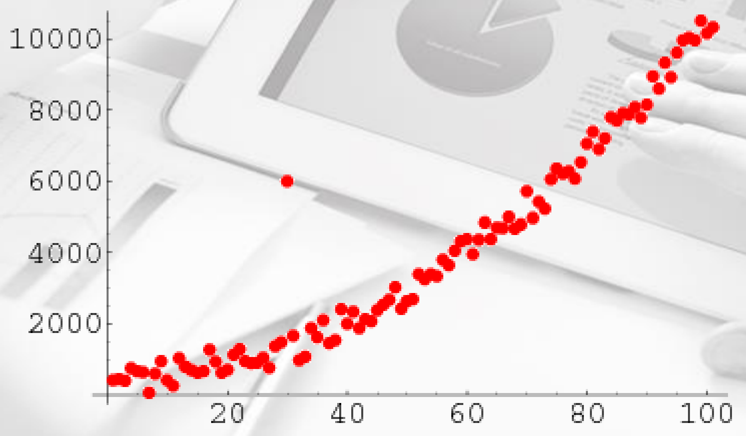
Data Cleaning > Graphs Principles > Graphs in R

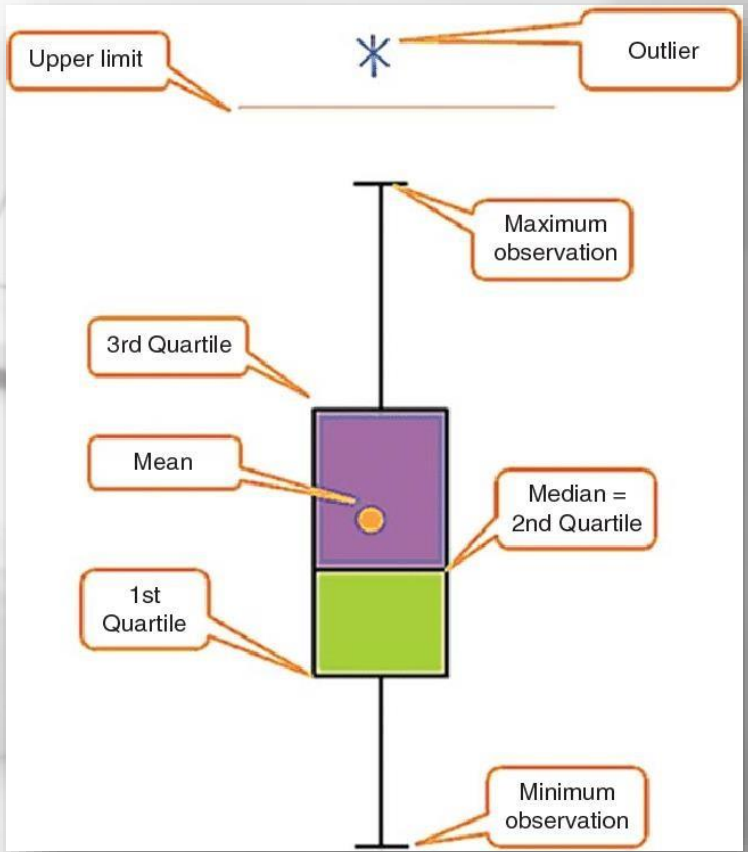
Duplicate values

```r
#########  Removing dublicates ###############
#--- Given the following vector:
x <- c(1, 1, 4, 5, 4, 6)
#--- To find the position of duplicate elements in x, use this:
duplicated(x)
# --- Extract duplicate elements:
x[duplicated(x)]
#--- If you want to remove duplicated elements, use !duplicated()
x[!duplicated(x)]
x <- x[!duplicated(x)]
x

#-- Following this way, you can remove duplicate rows from a data frame based on a column values, as follow:
library(tidyverse)
my_data <- as_tibble(iris)
my_data
# Remove duplicates based on Sepal.Width columns
my_data[!duplicated(my_data$Sepal.Width), ]

# --- Extract unique elements
x <- c(1, 1, 4, 5, 4, 6)
unique(x)
unique(my_data)

#--- Remove duplicate rows in a data frame

my_data %>% distinct()
my_data %>% distinct(Sepal.Length, .keep_all = TRUE)
```

Outliers

In statistics, an **outlier** is a data point that differs significantly from other observations.
An outlier may be due to variability in the measurement or it may indicate experimental error. An outlier can cause serious problems in statistical analyses.





Outliers

In this part, we use mtcars dataset. It comes with the base package, so no need to import anything)

mtcars {datasets} R Documentation

Motor Trend Car Road Tests

Description

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Usage

```
mtcars
```

Format

A data frame with 32 observations on 11 (numeric) variables.

[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	carb	Number of carburetors

Data Cleaning > Graphs Principles > Graphs in R

Outliers

```
######### Removing Outliers ###############

# First of all, we insert a couple of outliers to the $disp column of the mtcars dataset
# (mtcars comes with the base package, so no need to import anything)
# In order to have a couple of outliers in this dataset, we simply multiply the values in mtcars$disp that are higher than 420 by *2

mtcars$disp[which(mtcars$disp >420)] <- c(mtcars$disp[which(mtcars$disp >420)]*2)

# (This is just a random way of inserting a couple of outlier values, you could also assign a couple of high values in a milion different

# Now we have a look at $disp column of the mtcars dataset with boxplot

boxplot(mtcars$disp)

# You can get the actual values of the outliers with this

boxplot(mtcars$disp)$out

# Now you can assign the outlier values into a vector

outliers <- boxplot(mtcars$disp, plot=FALSE)$out

# Check the results

print(outliers)


###### Removing the outliers

mtcars[which(mtcars$disp %in% outliers),]


# Now you can remove the rows containing the outliers, one possible option is:

mtcars <- mtcars[-which(mtcars$disp %in% outliers),]

# If you check now with boxplot, you will notice that those pesky outliers are gone

boxplot(mtcars$disp)
```

Lab Activity:

Data cleaning the Singapore Airbnb dataset

You can download the raw data in the shared folder :

/Session 3/data

Source:

https://www.kaggle.com/jojoker/singapore-airbnb

Lab Activity:

Data cleaning the Singapore Airbnb dataset

idroom id
nameroom names
host_idhost id
host_namehost names
neighbourhood_groupSingapore regions
neighbourhoodspecific place
latitudelatitude
longitudelongtitude
room_typeroom type

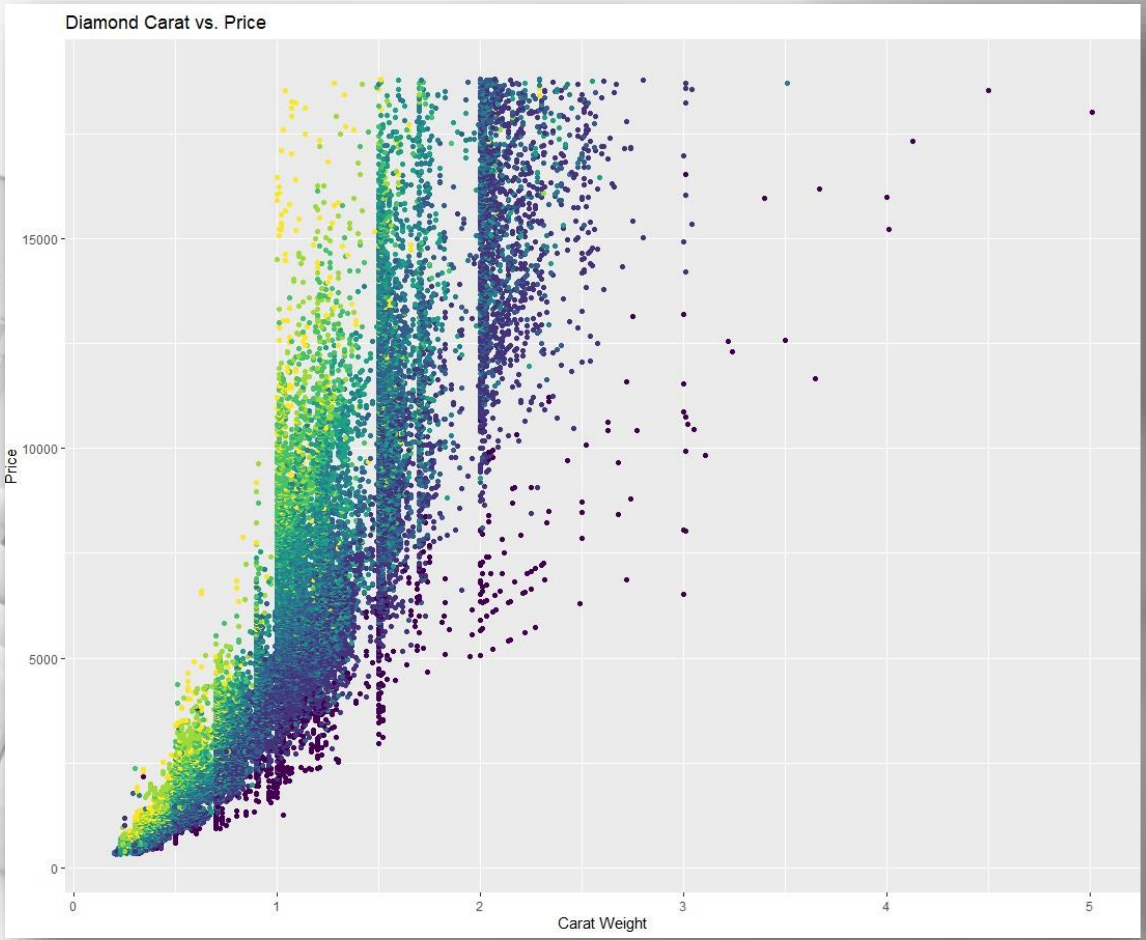
pricesingapore dollar per night
minimum_nightsminimum nights
number_of_reviewsnumber of review
last_reviewlast review
reviews_per_monthI don't know exaclty
calculated_host_listings_counttotal room or house in host catalog on Airbnb
availability_365availability

Lab Activity #1: Data cleaning the Singapore Airbnb dataset

Apply all required data cleaning techniques that you learned in this session to clean the Singapore Airbnb dataset.
Then submit the following files:

- your source code (name: Session3-[your name]) in R
- The cleaned excel file

Graphs in R


Diamond Carat vs. Price
Price
15000
10000
5000
0
0
1
2
3
4
5
Carat Weight

For this chapter, we will use a simple case study to demonstrate the kinds of simple graphs that can be useful in exploratory analyses. The data we will be using come from the U.S. Environmental Protection Agency (EPA), which is the U.S. government agency that sets national air quality standards for outdoor air pollution. One of the national ambient air quality standards in the U.S. concerns the long-term average level of fine particle pollution, also referred to as **PM2.5**.

The file is available is the shared folder

First, let's read the file:

```
rm(list=ls())


library(dplyr)
library("readxl")


setwd("D:/Academia/IPS/2019 Analytics for Master Students/Training Material/Season 3/Data")


 class <- c("numeric", "character", "factor", "numeric", "numeric")
 pollution <- read.csv("avgpm25.csv", colClasses = class)


 head(pollution)

 str(pollution)
```

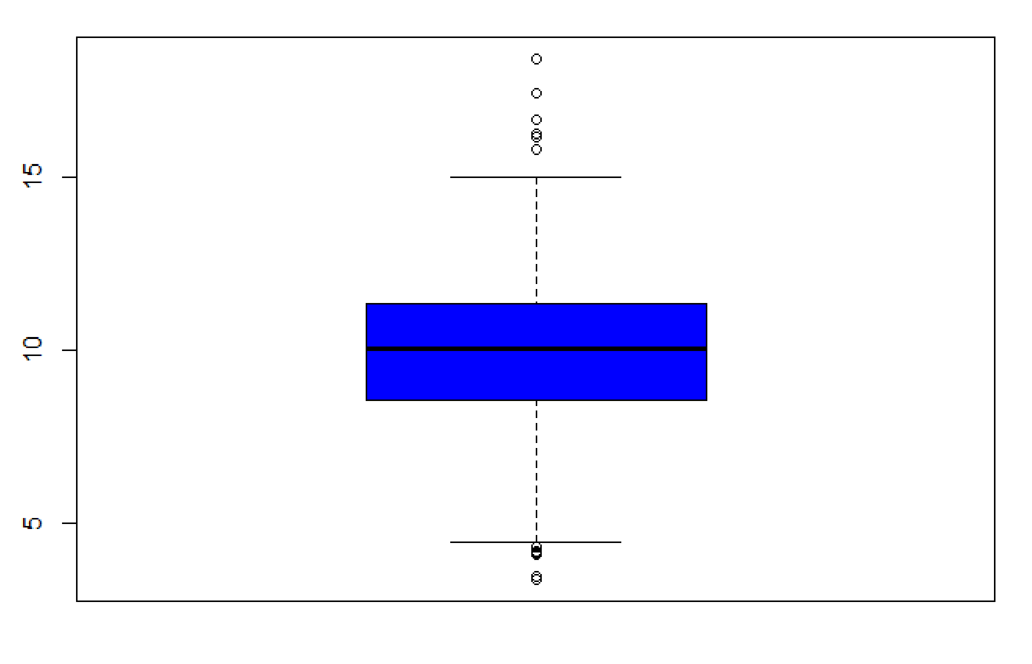
Five Number Summary:

> summary(pollution$pm25)

```
> summary(pollution$pm25)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.383   8.549  10.050   9.836  11.360  18.440 
```

Boxplot

```
> boxplot(pollution$pm25, col = "blue")
```

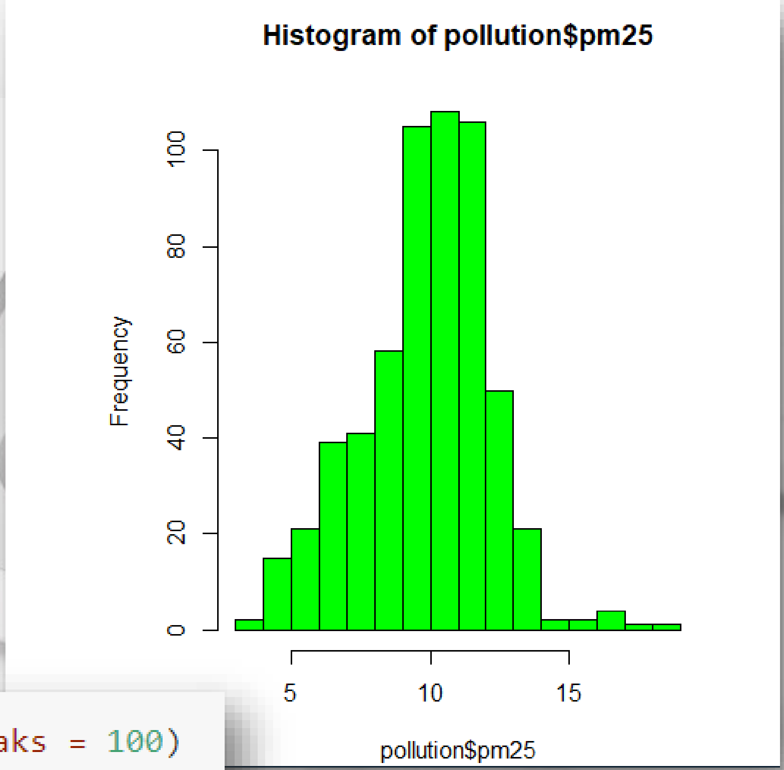


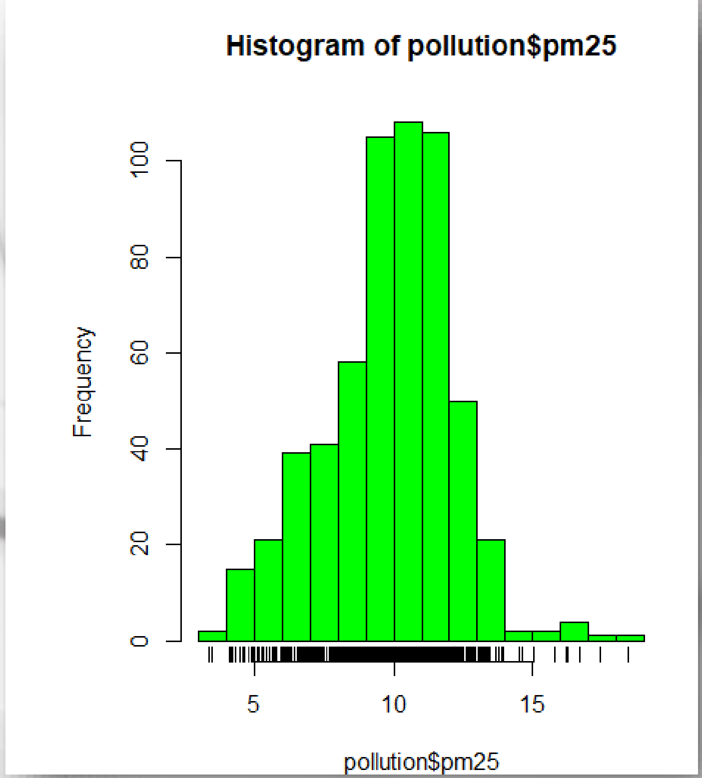
Your turn!

Based on the generated boxplot, show the outliers.

Histogram

```
> hist(pollution$pm25, col = "green")
> rug(pollution$pm25)
```

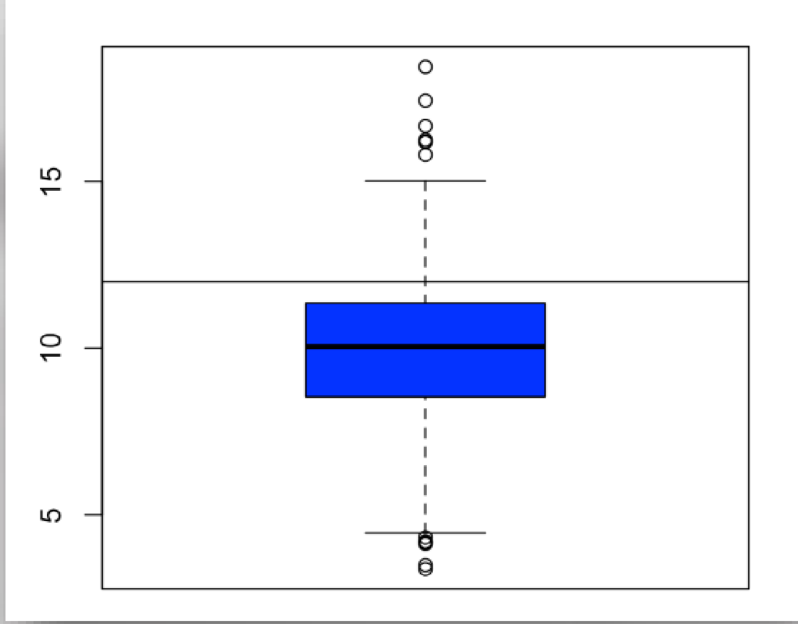




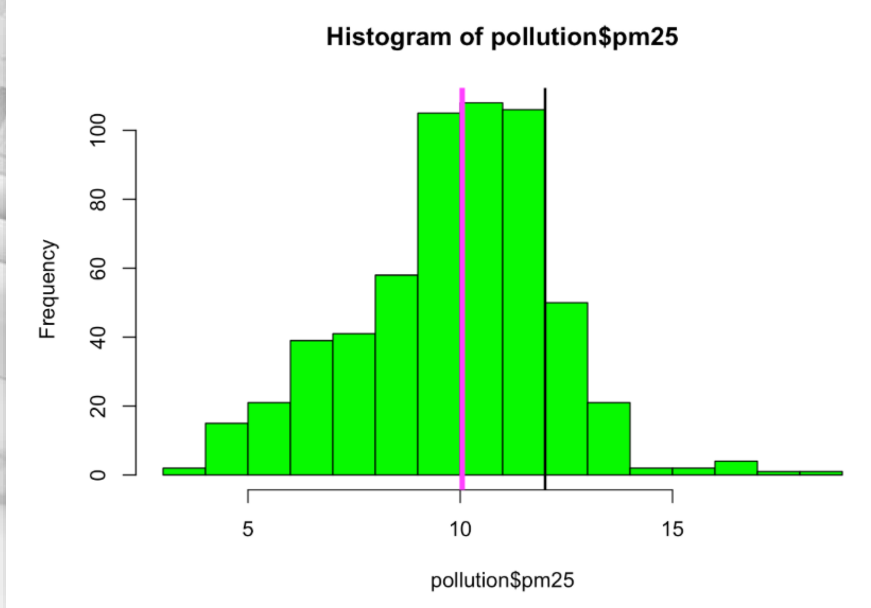

```
> hist(pollution$pm25, col = "green", breaks = 100)
> rug(pollution$pm25)
```

Adding reference lines:

```
> boxplot(pollution$pm25, col = "blue")
> abline(h = 12)
```



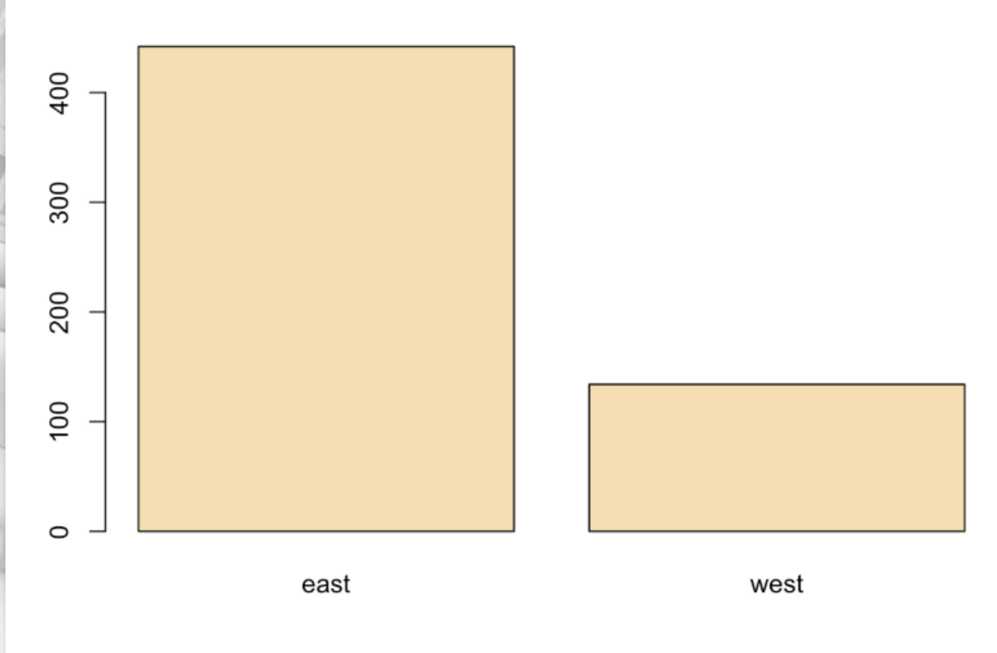

```
> hist(pollution$pm25, col = "green")
> abline(v = 12, lwd = 2)
> abline(v = median(pollution$pm25), col = "magenta", lwd = 4)
```



Barplots

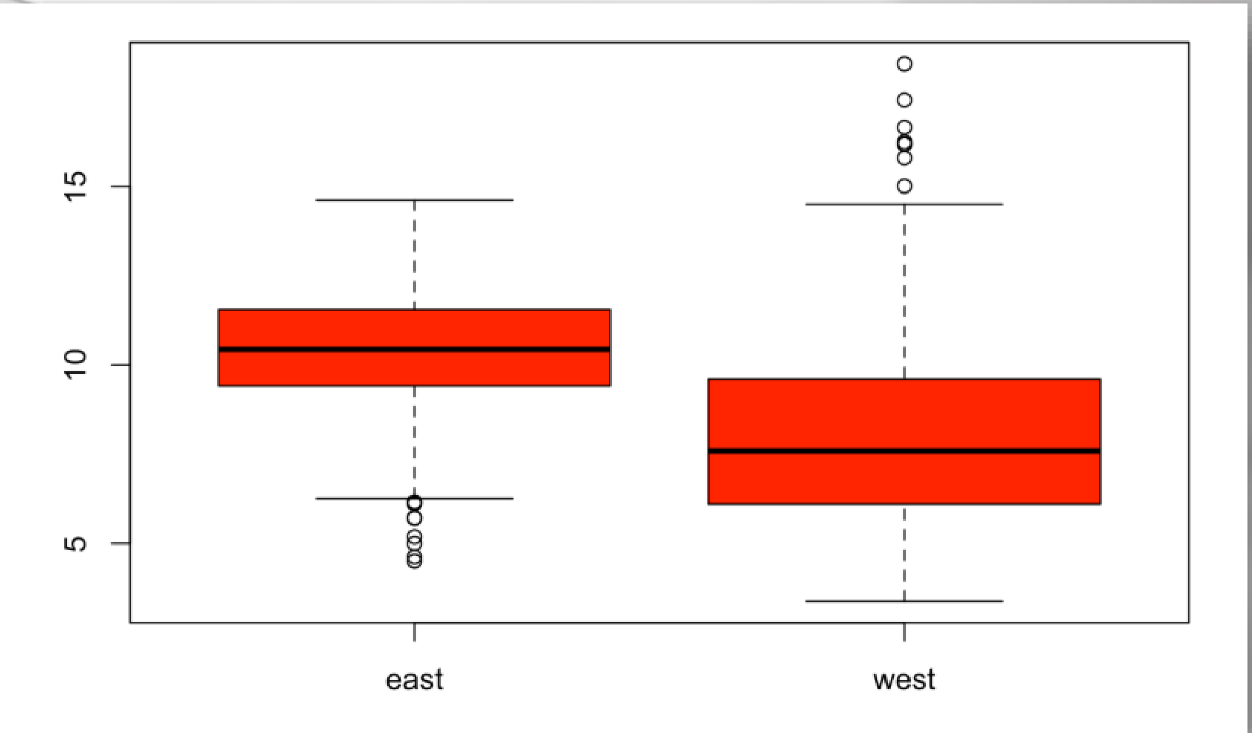
```
> library(dplyr)
> table(pollution$region) %>% barplot(col = "wheat")
```

The barplot is useful for summarizing categorical data. Here we have one categorical variable, the region in which a county resides (east or west). We can see how many western and eastern counties there are with barplot(). We use the table() function to do the actual tabulation of how many counties there are in each region.



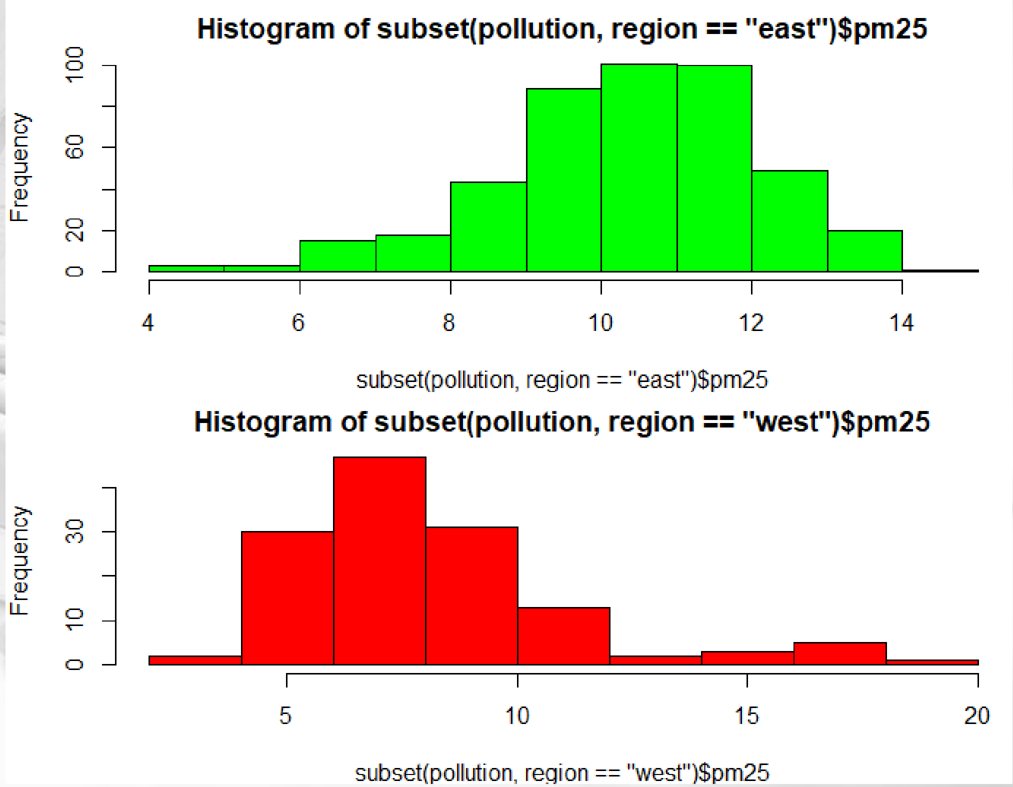
Multiple Boxplot

```
> boxplot(pm25 ~ region, data = pollution, col = "red")
```



Multiple Histograms

```
> hist(subset(pollution, region == "east")$pm25, col = "green")
> hist(subset(pollution, region == "west")$pm25, col = "green")
```

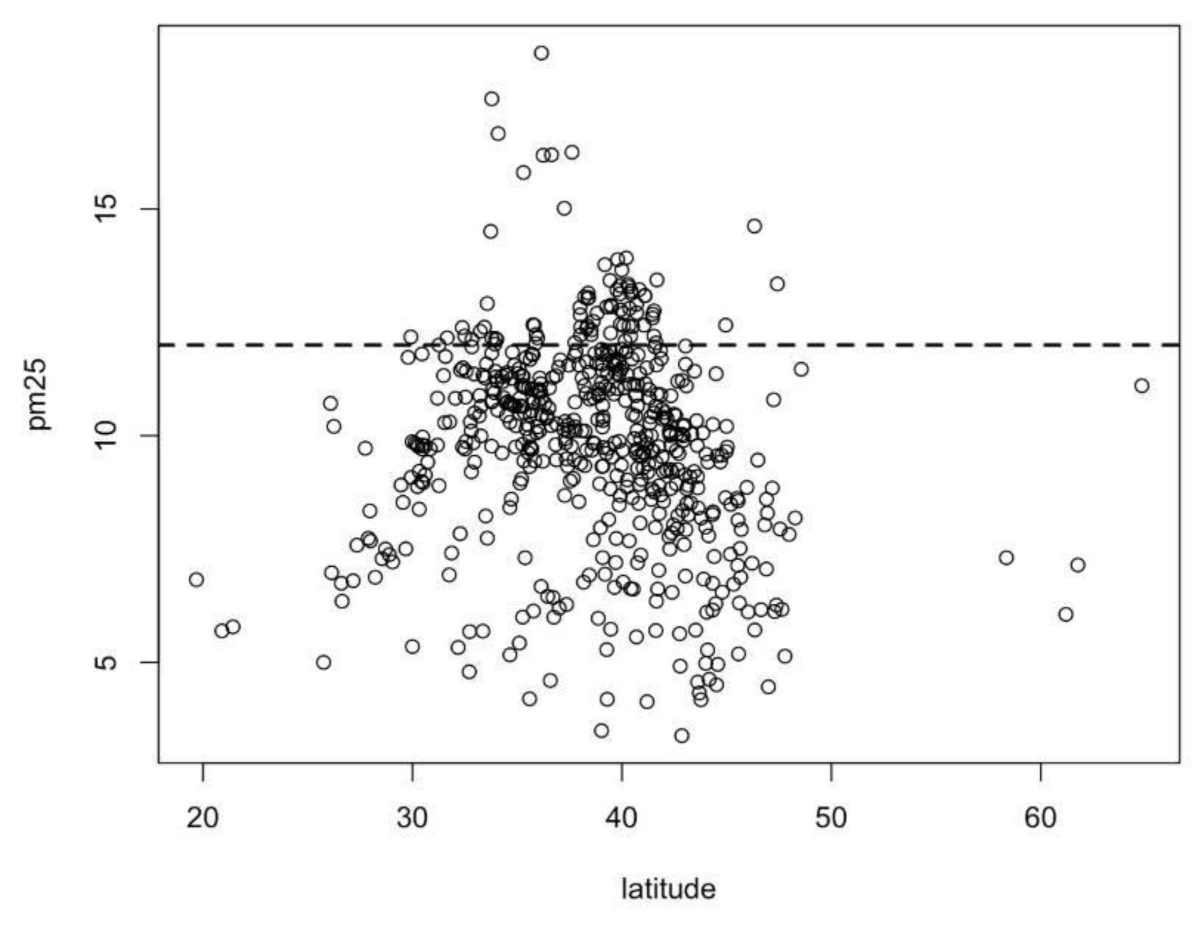


Your turn!

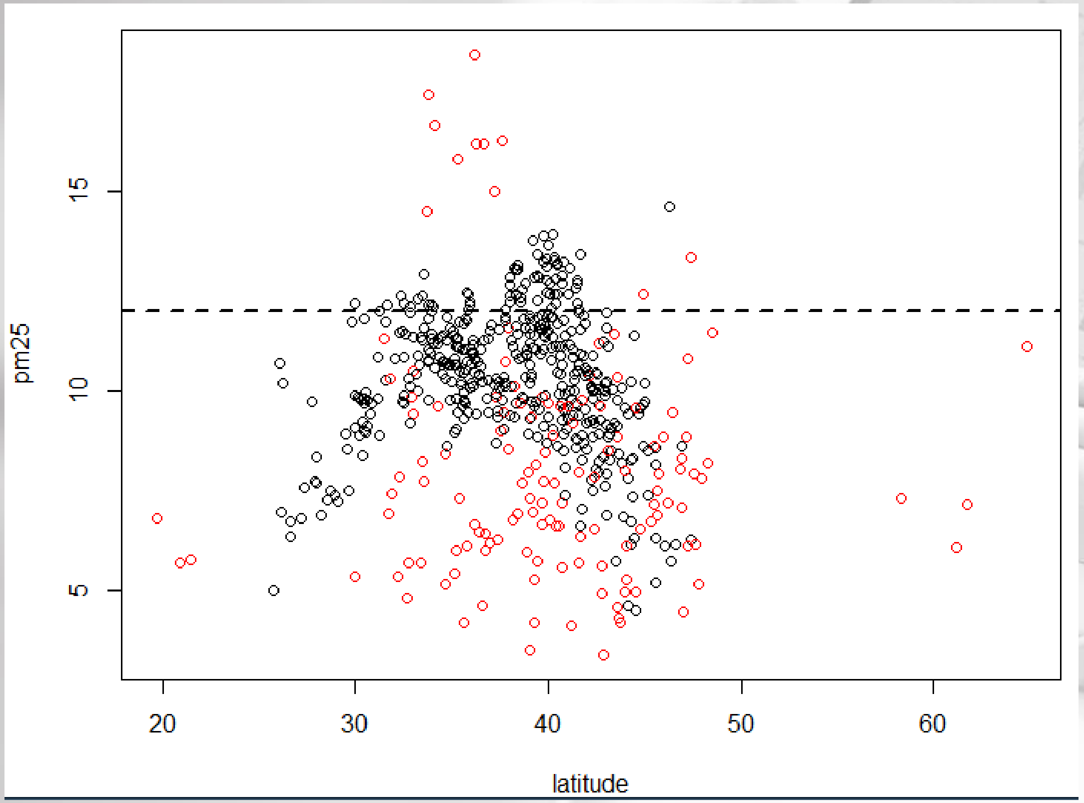
How to have two or more graphs in one picture?

Scatterplot

```
> with(pollution, plot(latitude, pm25))
> abline(h = 12, lwd = 2, lty = 2)
```



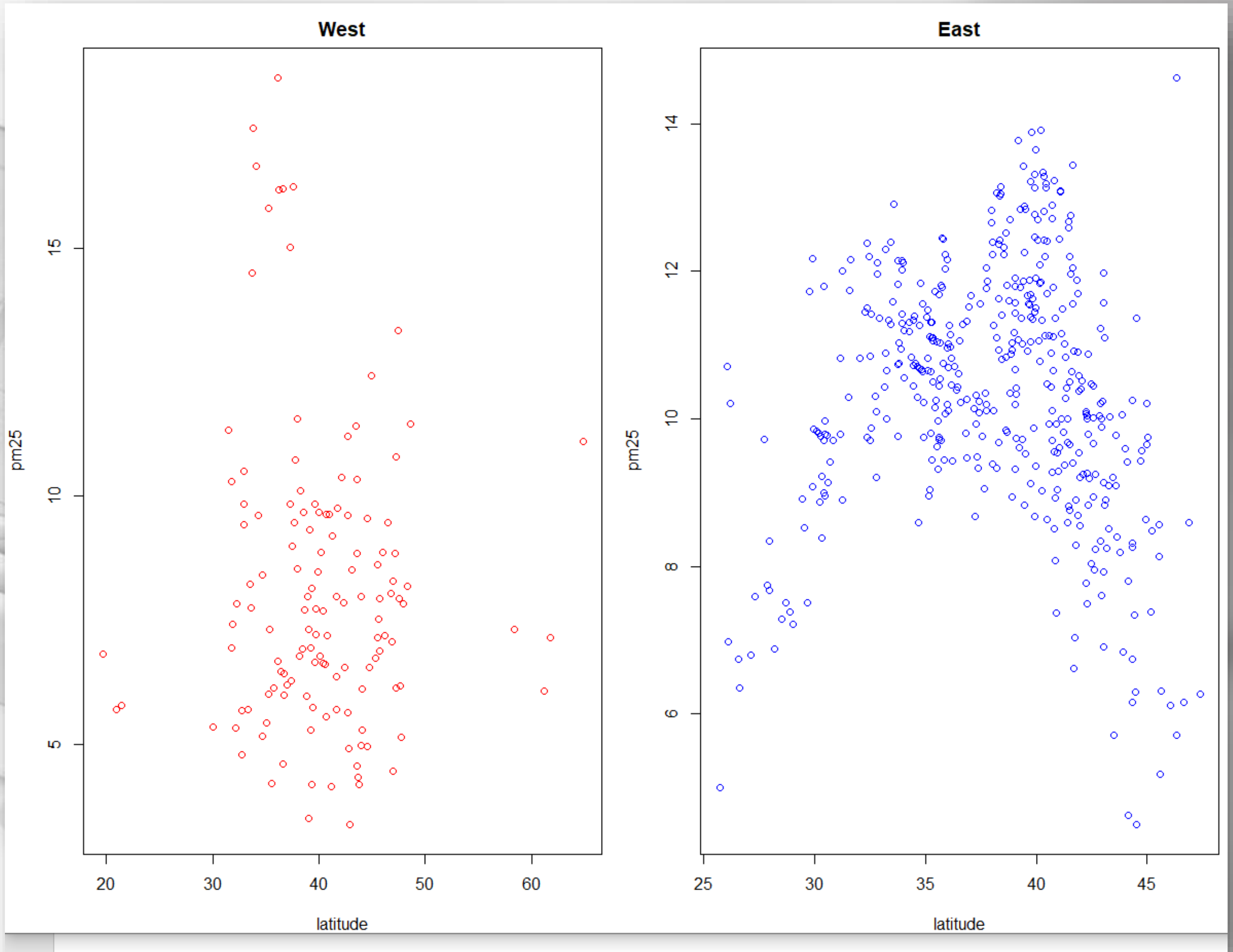
Scatterplot



Your turn!

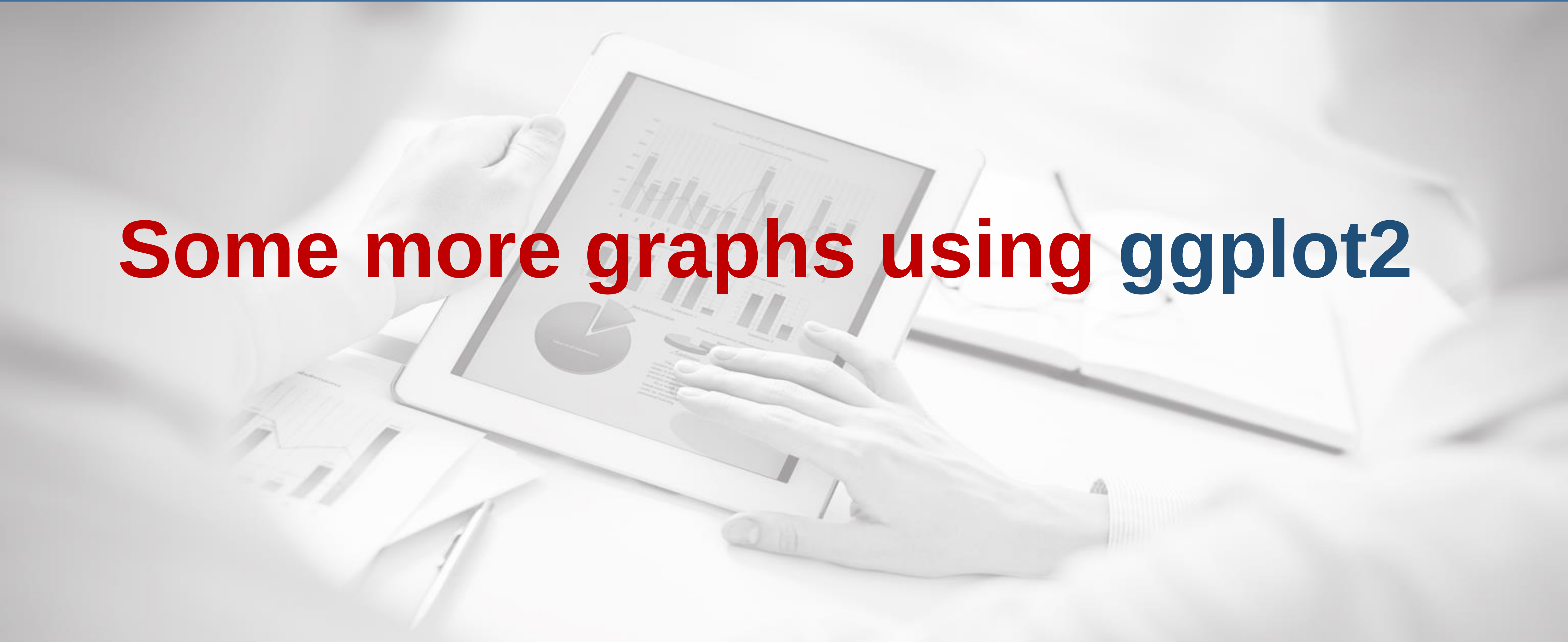
Add color to the Scatterplot based on the **region** attribute.

Scatterplot



Your turn!

Try to generate these double scatterplots



Some more graphs using ggplot2

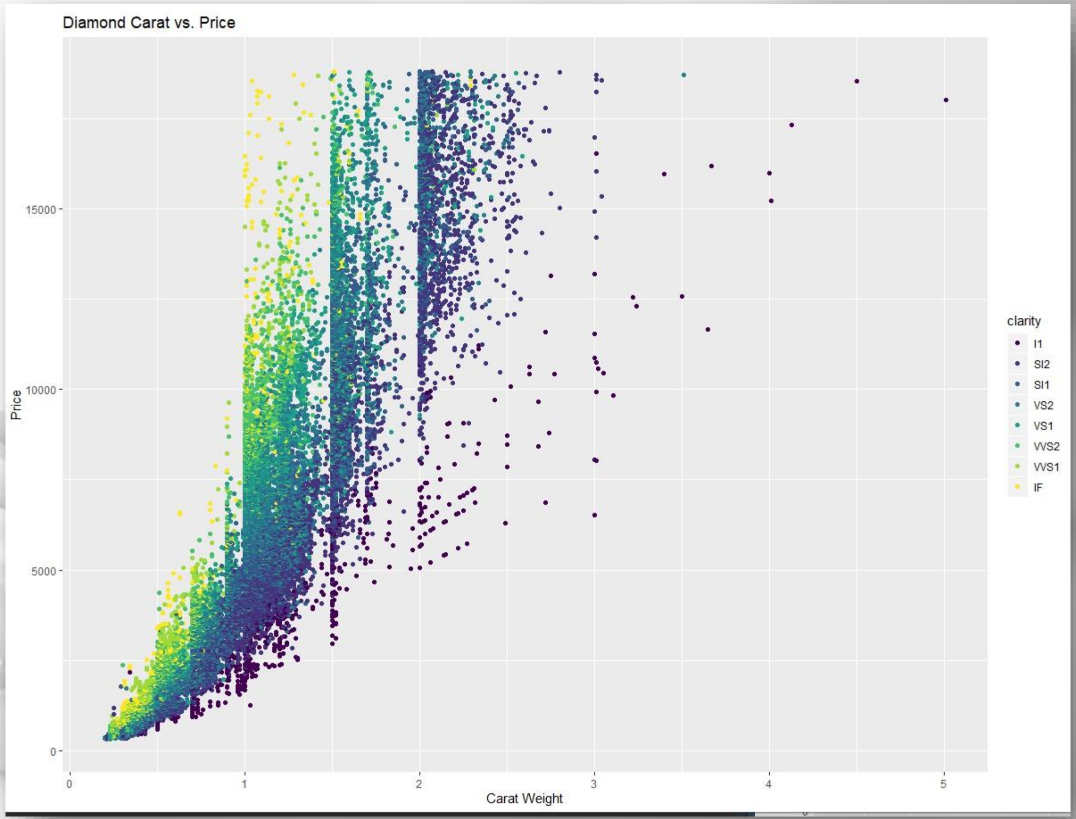
ggplot2 Basics and qplot()

The ggplot2 package is based on the principle that all plots consist of a few basic components: data, a coordinate system and a visual representation of the data. In ggplot2, you built plots incrementally, starting with the data and coordinates you want to use and then specifying the graphical features: lines, points, bars, color, etc.
The ggplot 2 package has two plotting functions qplot() (quick plot) and ggplot() (grammar of graphics plot.). The qplot() function is similar to the base R plot() function in that it only requires a single function call and it can create several different types of plots. qplot() can be useful for quick plotting, but it doesn't allow for as much flexibility as ggplot().

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Data Cleaning > Graphs Principles > Graphs in R

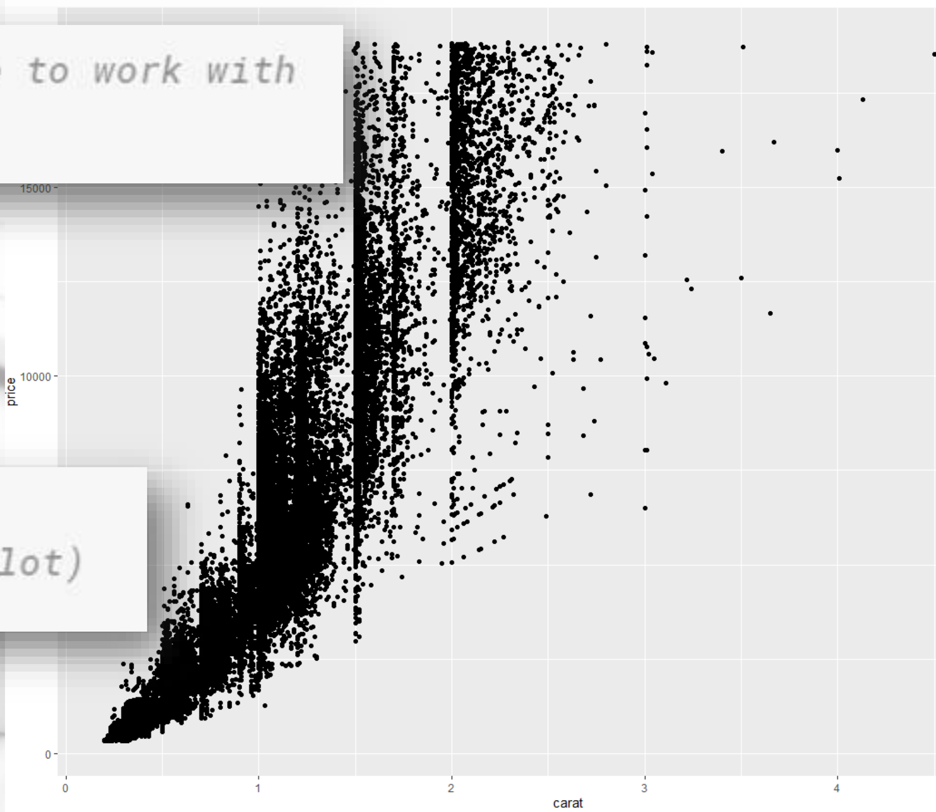
```r
library(ggplot2)

qplot(x = carat,                          # x variable
      y = price,                          # y variable
      data = diamonds,                    # Data set
      geom = "point",                     # Plot type
      color = clarity,                    # Color points by variable clarity
      xlab = "Carat Weight",              # x label
      ylab = "Price",                     # y label
      main = "Diamond Carat vs. Price");  # Title
```



ggplot2 >> Using ggplot()

```
ggplot(data=diamonds,                # call to ggplot() and data frame to work with
       aes(x=carat, y=price))        # aesthetics to assign
```

```
ggplot(data=diamonds, aes(x=carat, y=price)) +  # Initialize plot*
     geom_point()                # Add a layer of points (make scatterplot)
```



ggplot2 >> Using ggplot()

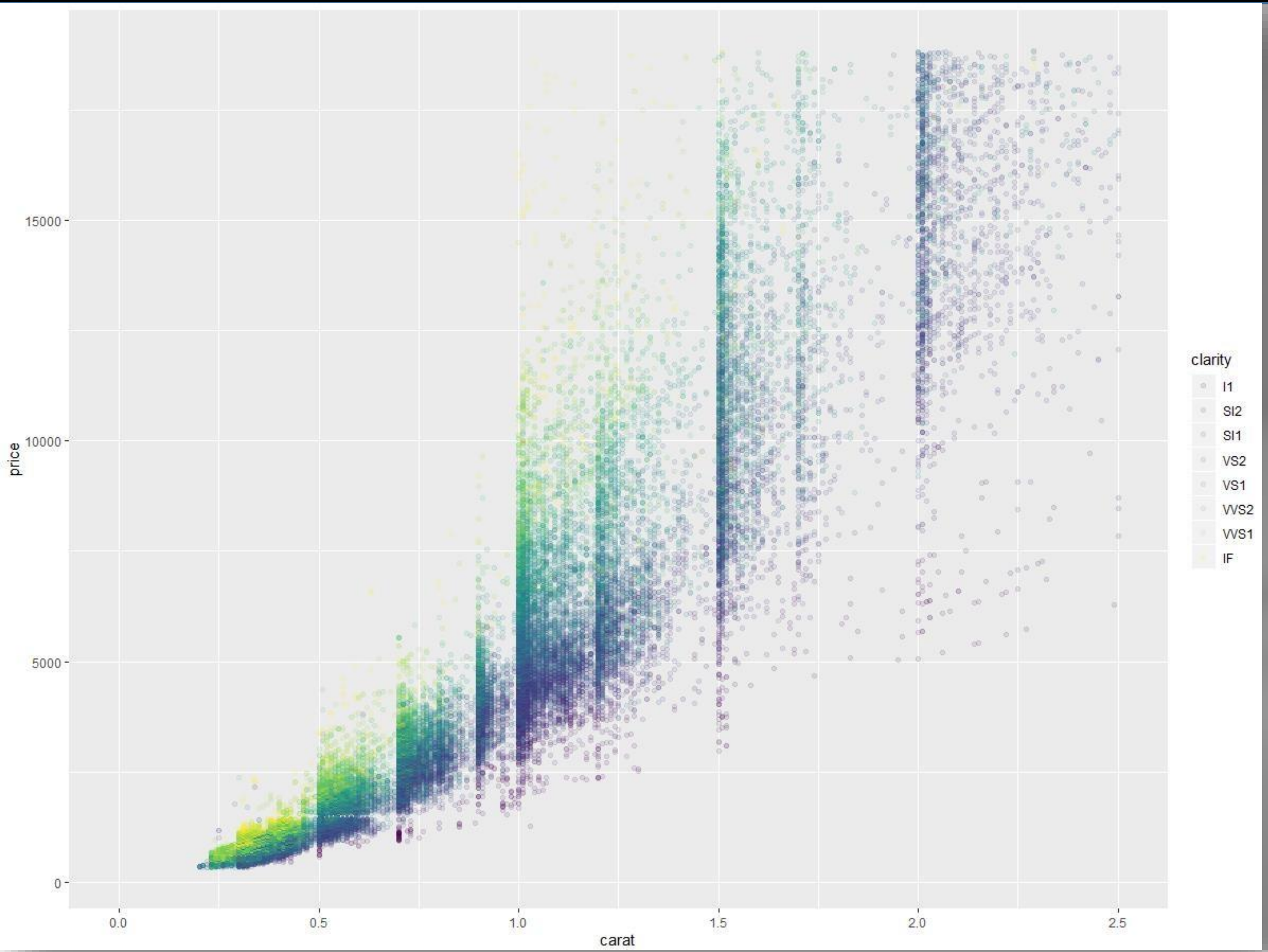
Different functions that can be used in ggplot()

```
geom_histogram()   # histogram
geom_density()     # density plot
geom_boxplot()     # boxplot
geom_violin()      # violin plot (combination of boxplot and density plot)
geom_bar()         # bar graph
geom_point()       # scatterplot
geom_jitter()      # scatterplot with points randomly perturbed to reduce overlap
geom_line()        # line graph
geom_errorbar()    # Add error bar
geom_smooth()      # Add a best-fit line
geom_abline()      # Add a line with specified slope and intercept
```

ggplot2 >> Using ggplot()

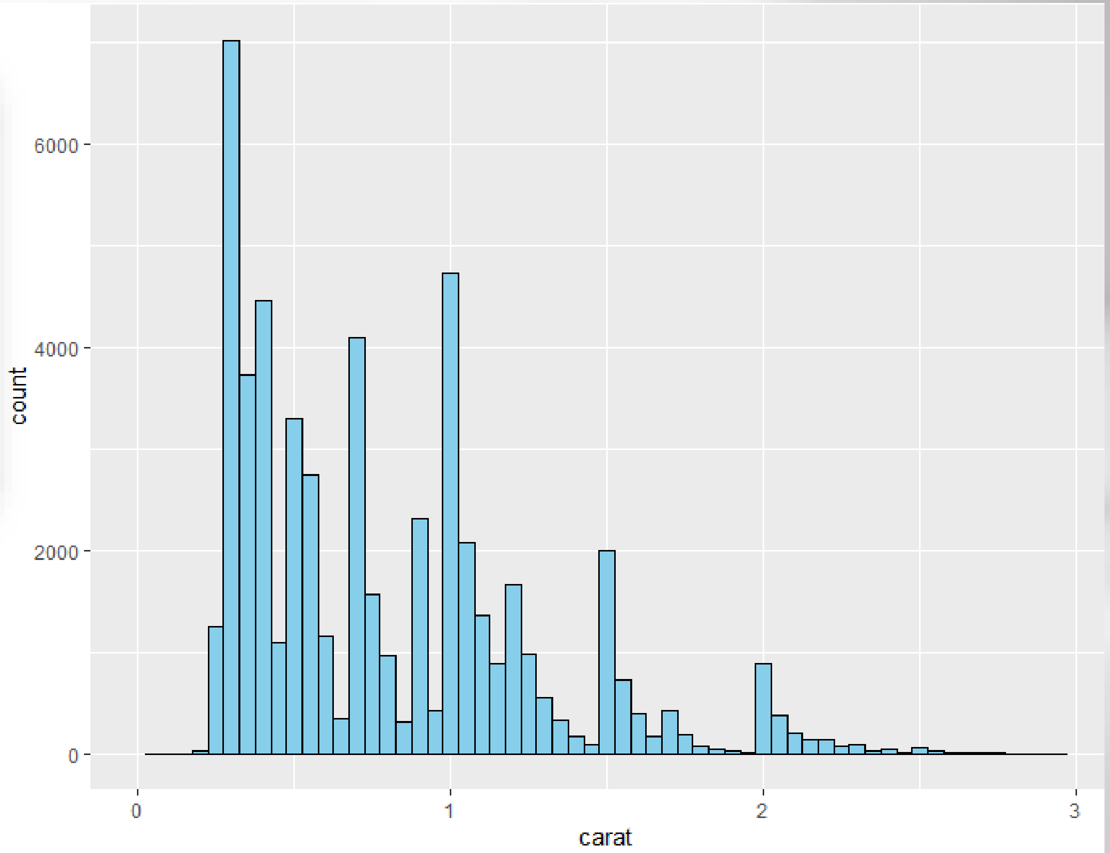
```
ggplot(data=diamonds, aes(x=carat, y=price)) +  # Initialize plot
       geom_point(aes(color = clarity), alpha = 0.1)  +  # Add transparency
       xlim(0,2.5)                                       # Specify x-axis range
```

ggplot2 >> Using ggplot()



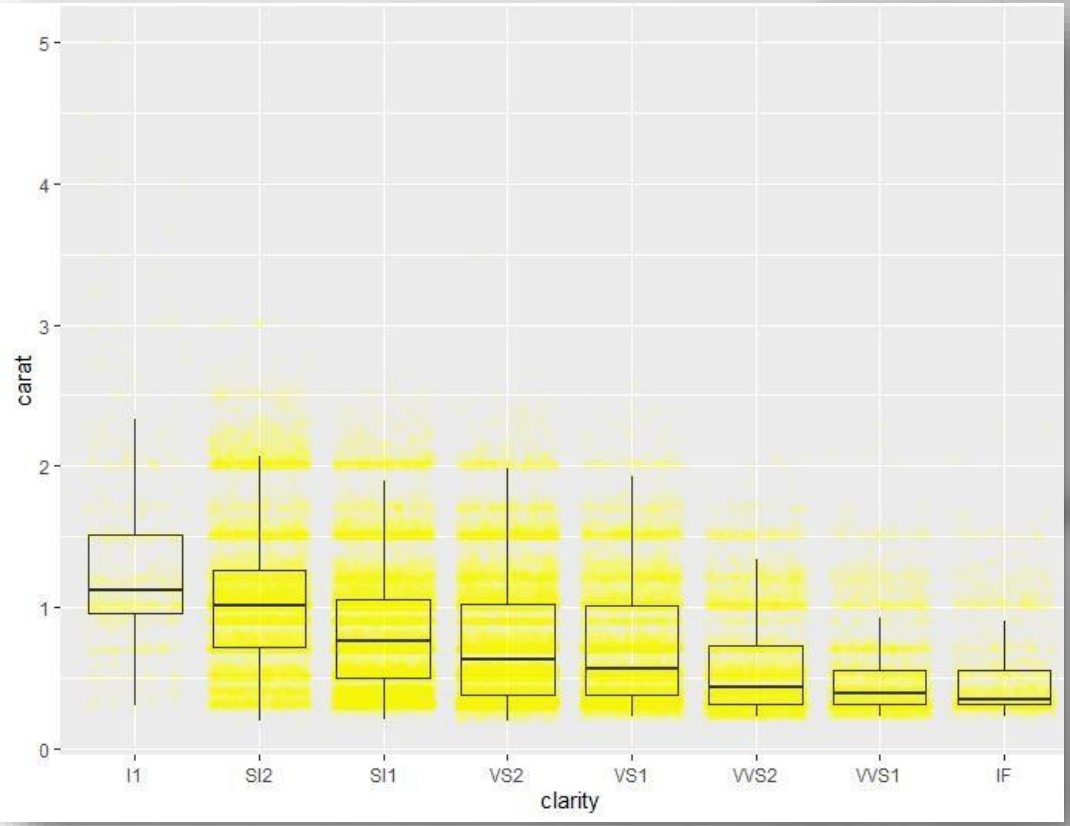
ggplot2 >> Using ggplot()

```
# Create a histogram of carat

ggplot(data=diamonds, aes(x=carat)) +       # Initialize plot

      geom_histogram(fill="skyblue",        # Create histogram with blue bars
                     col="black",           # Set bar outline color to black
                     binwidth = 0.05) +     # Set bin width

      xlim(0,3)                             # Add x-axis limits
```



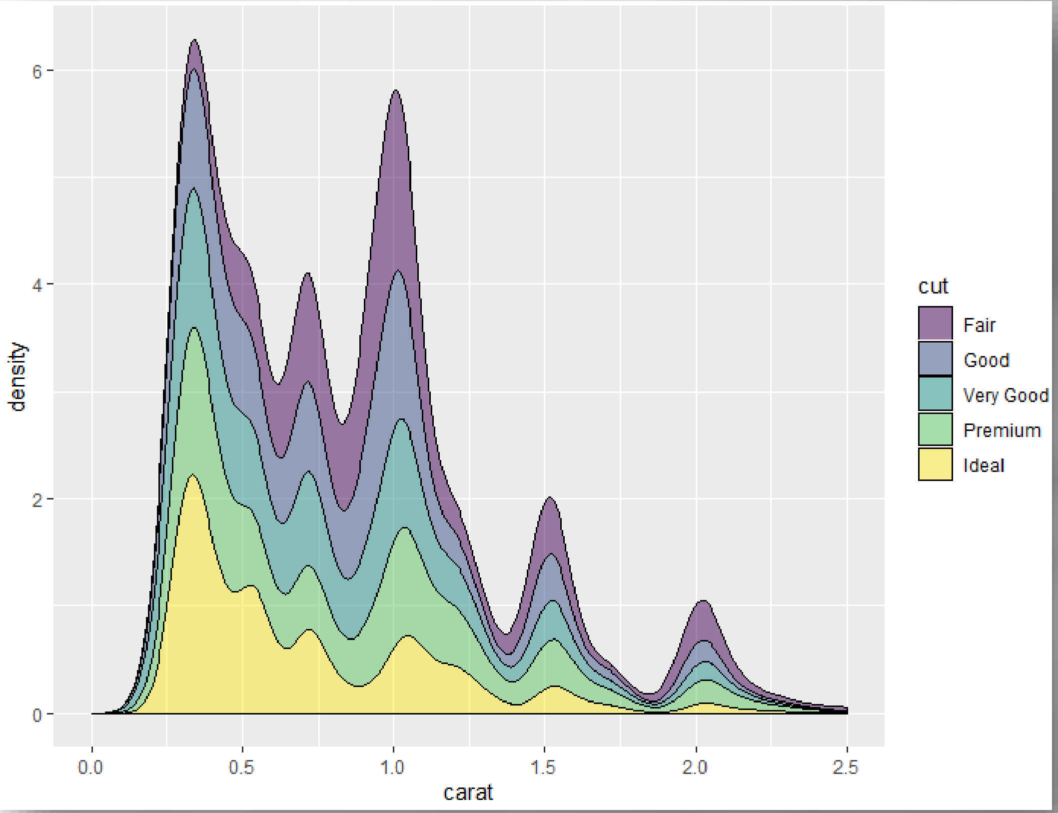
ggplot2 >> Using ggplot()

```
# Create boxplot of carat split on clarity with points added
ggplot(data=diamonds, aes(x=clarity, y=carat)) +  # Initialize plot

      geom_jitter(alpha=0.05,            # Add jittered data points
                  color="yellow") +      # Set data point color

      geom_boxplot(outlier.shape=1,        # Create boxplot and set outlier shape
                   alpha = 0  )            # Make inner boxplot area transparent
```



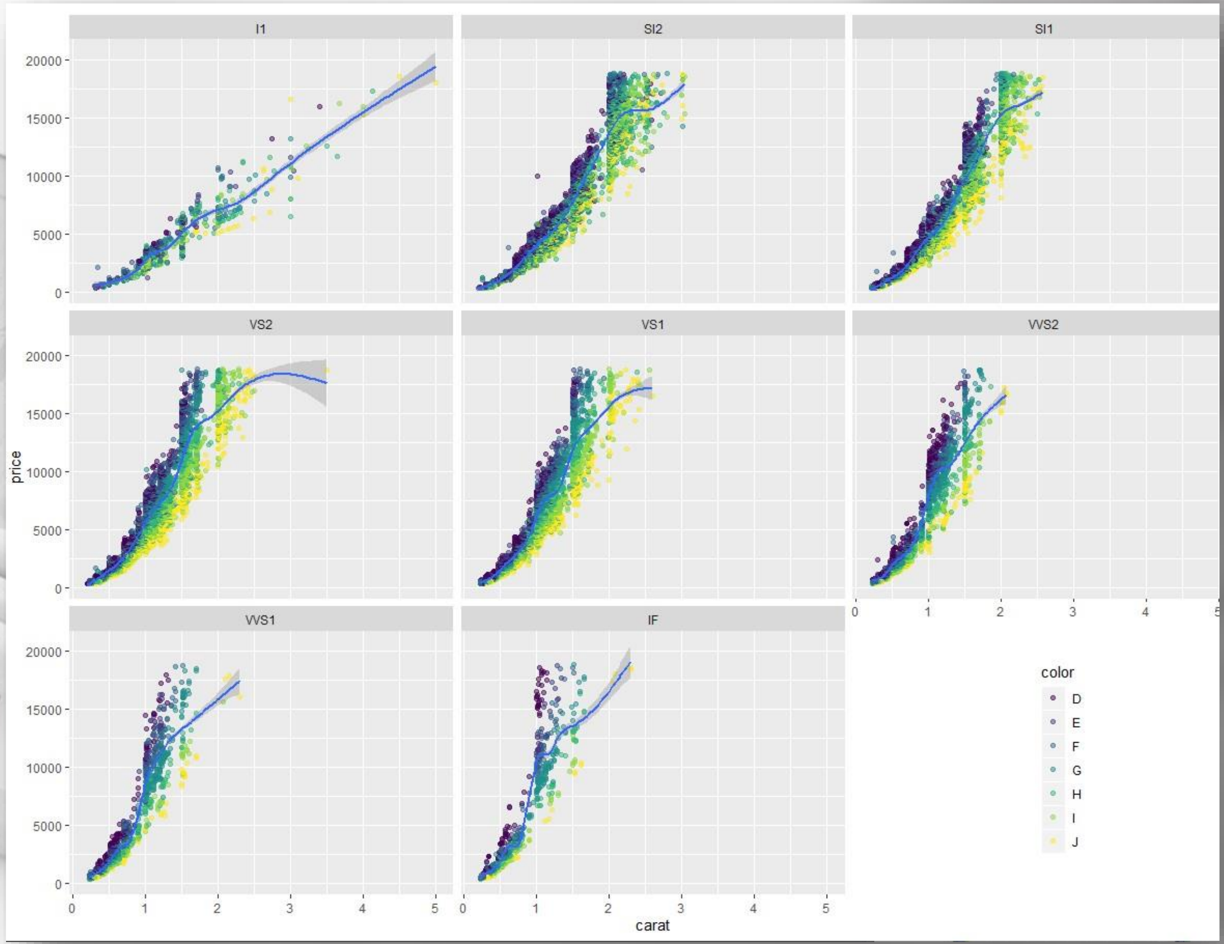
ggplot2 >> Using ggplot()

```
ggplot(data=diamonds, aes(x=carat)) +        # Initialize plot
       xlim(0,2.5)                   +        # Limit the x-axis*

       geom_density(position="stack",         # Create a stacked density chart
                    aes(fill=cut),            # Fill based on cut
                    alpha = 0.5)              # Set transparency
```



ggplot2 >> Using ggplot()

```
ggplot(data=diamonds, aes(x=carat, y=price)) +  # Initialize plot

     geom_point(aes(color=color),               # Color based on diamond color
                alpha=0.5)      +

     facet_wrap(~clarity)       +               # Facet on clarity

     geom_smooth()              +               # Add an estimated fit line*

     theme(legend.position=c(0.85,0.16))        # Set legend position
```



Lab Activity #2: Exploratory Analysis of the Singapore Airbnb dataset

Apply all required exploratory analysis techniques that you learned in this session to the **cleaned** Singapore Airbnb dataset and build the required graphs. [from Basic to advanced]

Then email me the following files:

- your source code (name: Session3-[your name]-EDA) in R
- The exploratory analysis report

Some useful sources for further reading:

1. Edward Tufte (2006). Beautiful Evidence, Graphics Press LLC. www.edwardtufte.com
2. https://bookdown.org/rdpeng/exdata/exploratory-graphs.html#scatterplots
3. https://www.r-graph-gallery.com/
4. https://www.kaggle.com/hamelg/intro-to-r-part-20-plotting-with-ggplot2

Any questions?

Vala@data-corner.com