```
library(dplyr)

rladies_global %>%
  filter(city == ‘Orlando’)
```

# R-Ladies Orlando Data Visualization with ggplot2

# A Very Special Thank You

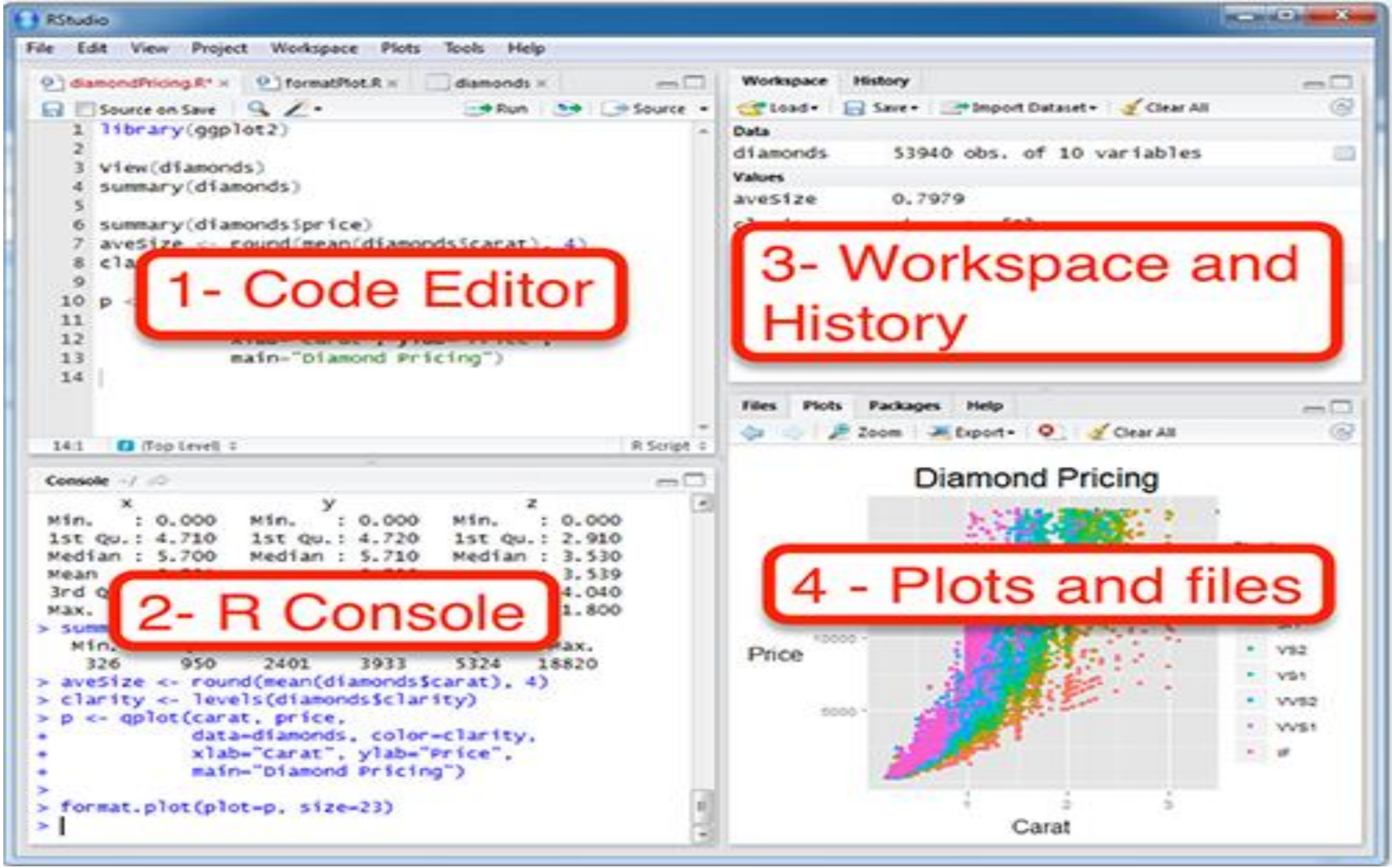
Code School

a Pluralsight company

# Hello!

## I am Kathy Joshi

You can find me at:
Kathy@rladies.org
kathyjoshi1030@gmail.com



```
library(ggplot2)

view(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
cla

p <


                 main="Diamond Pricing")

```

Data

diamonds        53940 obs. of 10 variables

Values

aveSize         0.7979

```
         x                 y                 z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean                                       3.539
 3rd Q                                      4.040
 Max.                                       1.800
> summ
   Min.                                    Max.
    326      950     2401     3933     5324    18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+              data=diamonds, color=clarity,
+              xlab="Carat", ylab="Price",
+              main="Diamond Pricing")
>
> format.plot(plot=p, size=23)
> 
```

1- Code Editor

2- R Console

3- Workspace and History

4 - Plots and files

# Variables: characters & integers

```r
# Create a new script: File -> New file -> R Script
# Now let’s create a variable!
a <- 2

# Look at the Environment section to see variables you’ve made and their current values.
# Change the value of a, then look at the Environment section to see the updated value.
a <- 3 + 5

# c() creates a vector: a sequence of data points of the same type
b <- c(1,3,5,7,9)
b <- 1:12

# Now try this: what happens?
c <- c(4,3,"5",4)

# Let’s explore. Try this:
c <- "5"

# Now try this:
c <- 5
a <- "RStudio"                     # quotation marks make character variables
a <- as.character(5)    # this is another way to make the character variable type
```

# Variables: Boolean Data

```r
# Boolean or logical variables only have true and false options.
# Computers see “true” as 1 and “false” as 0.

# Make this variable and then look at the Environment section to look at the variable type.
a <- TRUE
a <- c(TRUE, FALSE, FALSE, TRUE)
```

```r
# There are also questions you can ask R that will give boolean answers.
# Ask R whether a is a numeric (non-character) variable.
is.numeric(a)
```

```
## [1] FALSE
```

```r
# a is a boolean variable, not a numeric variable

# Even though the computer sees true as 1 and false as 0, it can tell that you made a boolean
# variable because you typed TRUE and FALSE when you created it.
```

```r
# Now try these questions:

is.logical(a)

is.character(a)
```

# Functions

```r
# We have been using functions to ask R questions and to give it tasks.
# The functions we have used so far are:
c()
as.character(5)
is.numeric(a)
is.logical(a)
is.character(a)
# Functions have a function name (c, as.character, is.numeric etc)
# The function name is followed by curved brackets:()
# Functions always have these brackets, even if there’s nothing inside.
# The area inside the brackets can be used to give the function information
as.character(5)

# Here we are giving the function as.character() the information 5.
# This means that it checks whether 5 is a character


# If we wanted to store the result that the function gives us, we can make the answer into a
# stored variable like this:

b <- as.character(5)


# What variable type would b be, and what would its value be?
```

# Data Frames and Tibbles

```
# The data.frame() function makes data frames out of your data
# A data frame is like a table: it stores your data neatly
# The data frame structure is used in most applications of R.


# Create three vectors: a, b, c.
# (Reminder: a vector is a sequence of data that has the same variable type)
a <- c(1,2,3,4,10,11,12)
b <- c(5,6,7,8,13,14,15)
c <- c("yes","no","no","yes","no","yes","yes”)

 # Combine these vectors to make and store a data frame called myData
 # (or call it something else if you want!)
 myData <- data.frame(a,b,c)

 # Now run this function:
 head(myData,3)

 # In Tidyverse will be using the term tibble. Tibbles are data frames, but they tweak some older
 # behaviors to make life a little easier.

 as.tibble()
```

# Data Frames and Tibbles

```r
# Data frames use the [row, column] access structure.
# To access the element in the 4th row, 2nd column of the data frame:
myData[4,2]


# To access a whole column of the data frame, use the operator $.
# This allows you to find information by name: let’s find all the values with column name a.
myData$a


# To access a whole column of the data frame, use the column number in the [row,column] format.
myData[,1]


# To access a whole row of the data frame, use the row number in the [row,column] format.
myData[1,]
```

Phote credit: https://blog.musicteachershelper.com/wp-content/images/90.jpg

# Recap Basic Functions

- Load tidyverse, how do we do that??
- Create a new project or new script
- Load a dataset , today will work on the “diamonds” dataset already part of tidyverse
- getwd()
- setwd()
- head()
- tail()
- str()
- View()
- summary()

# Why we should always visualize our data:

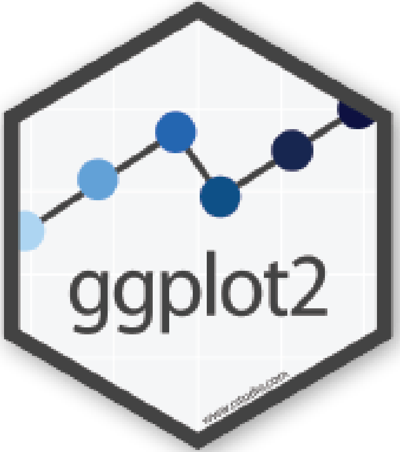
- https://www.autodeskresearch.com/publications/samestats

- https://fivethirtyeight.com/features/al-gores-new-movie-exposes-the-big-flaw-in-online-movie-ratings/

# Diamonds dataset

```
A tibble with 53940 rows and 10 variables:
price           price in US dollars (\$326--\$18,823)
carat           weight of the diamond (0.2--5.01)
cut             quality of the cut (Fair,Good,Very,Good,Premium,Ideal)
color           diamond color, from J (worst) to D (best)
clarity         a measurement of how clear the diamond is (I1
                (worst), SI2, SI1, VS2, VS1, VVS2, VVS1,
                IF (best))
x               length in mm (0--10.74)
y               width in mm (0--58.9)
z               depth in mm (0--31.8)
depth           total depth percentage = z / mean(x, y) = 2 * z /
                (x + y) (43--79)
table           width of top of diamond relative to widest point
                (43--95)
```

# Let’s go to Rstudio:



рахмат
danke
謝謝
ngiyabonga
spas
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
blagodaram
dank je
misaotra
paldies
grazzi
matondo
спасибі
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
thank you
gracias
хвала
asante
manana
bedankt
bayarlalaa
gràçie
hvala
köszönöm
dankon
ačiū
akun
obrigada
murakoze
tenki
enkosi
dziękuję
djiere dieuf
tau
mochchakkeram
дякую
mamnun
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
go raibh maith agat
sobodi
děkuji
obrigado
sagolun
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
kam sah hamnida
rahmat
terima kasih
rahmet
ευχαριστώ
diolch
dhanyavadagalu
shukriya
mercé
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci