Basics in R
(part 2)

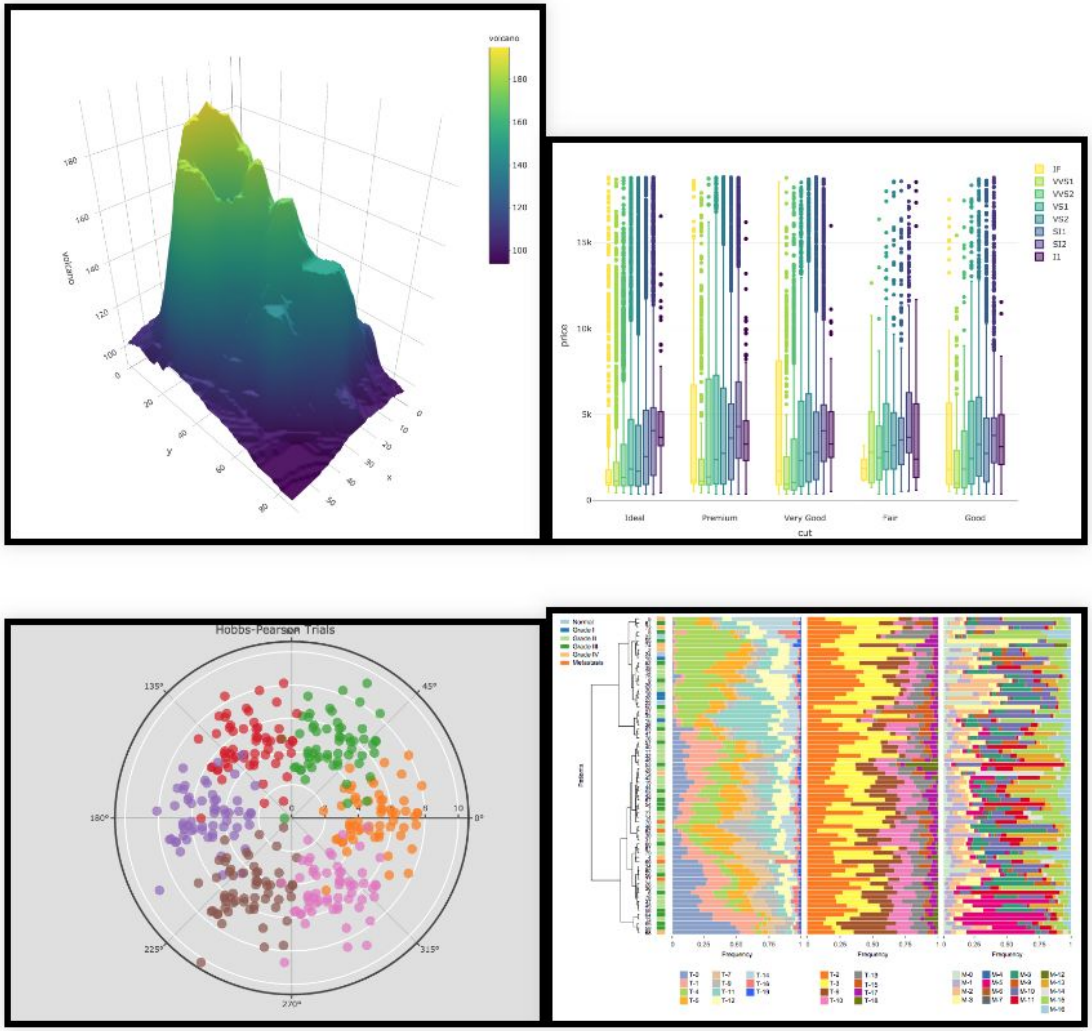
•••

Lokesh

Why use R?

- R is a *statistical* programming language (derived from S)
- Superb data management & graphics capabilities
- You can write your own functions
- Powerful and flexible
- Runs on all computer platforms
- Well established system of packages and documentation
- Active development and dedicated community
- Can use a nice GUI front end such as `Rstudio`
- Reproducibility
  - keep your scripts to see exactly what was done
  - distribute these with your data
  - embed your R analyses in polished R`Markdown` files
- FREE

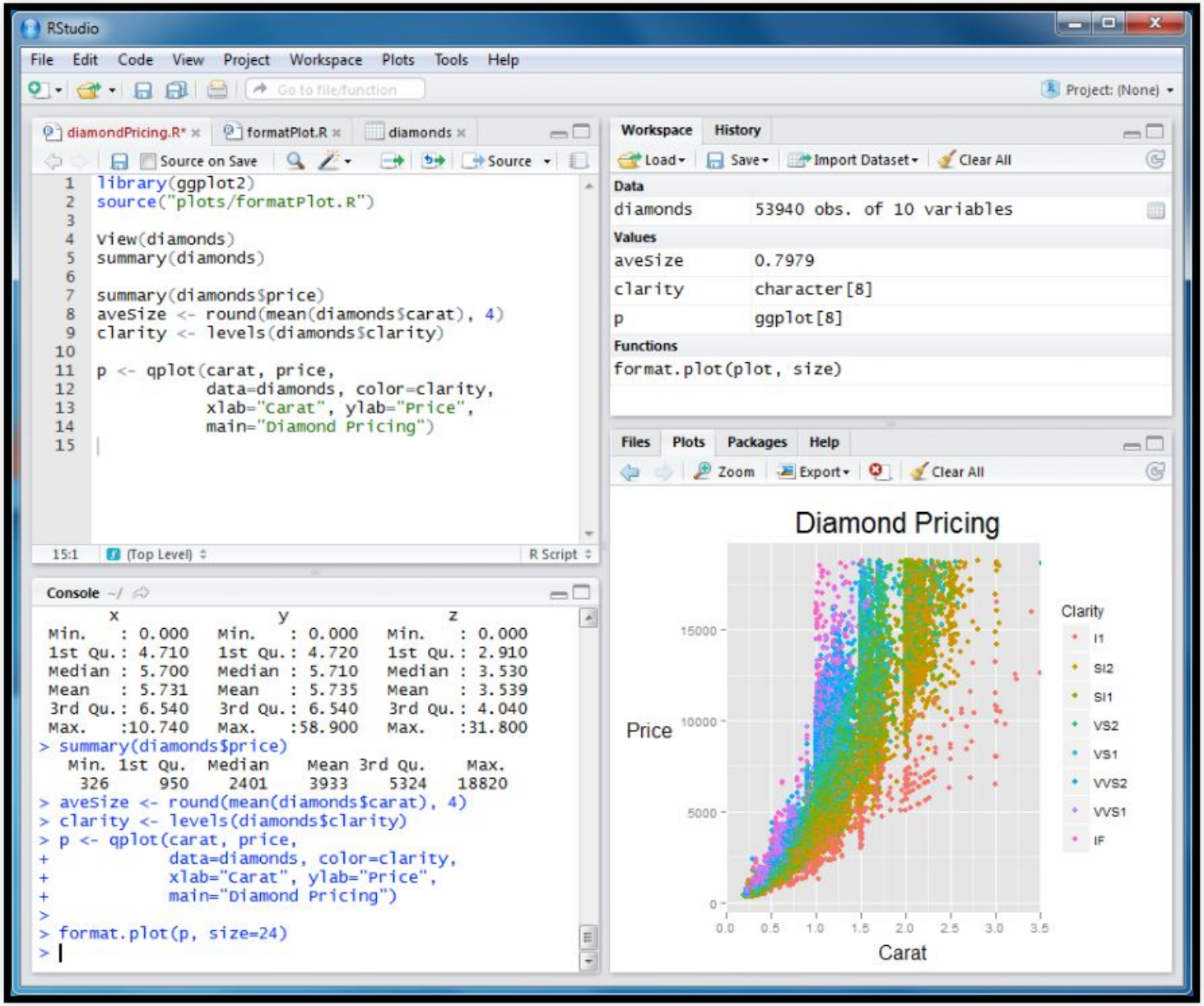
Why use R?


volcano
price
cut
Ideal
Premium
Very Good
Fair
Good
Hobbs-Pearson Trials
Normal
Grade I
Grade II
Grade III
Grade IV
Veterans
Frequency

R resources

- The R Project Homepage: http://www.r-project.org
- Quick R Homepage: http://www.statmethods.net
- Bioconductor: http://www.bioconductor.org
- An Introduction to R (long!): http://cran.r-project.org/doc/manuals/R-intro.html
- R for Data Science: https://r4ds.had.co.nz
- Google - tutorials, guides, demos, packages and more

RStudio


RStudio
File Edit Code View Project Workspace Plots Tools Help
Project: (None)
diamondPricing.R*
formatPlot.R
diamonds
Source on Save
Source
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12 data=diamonds, color=clarity,
13 xlab="Carat", ylab="Price",
14 main="Diamond Pricing")
15
15:1 (Top Level) R Script
Console ~/
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. 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(p, size=24)
>
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
0
5000
10000
15000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

Rstudio Demo

•••

CRAN and Bioconductor

? and ??

Assigning Variables

- A better way to do this is to assign variables
- Variables are assigned values using the <- operator.
- Variable names must begin with a letter, but other than that, just about anything goes.
- Do keep in mind that R is **case sensitive**.

STRINGS

```r
x <- "I Love"
print(x)
```

```
## [1] "I Love"
```

```r
y <- "Biostatistics"
print(y)
```

```
## [1] "Biostatistics"
```

```r
z <- c(x, y)
print(z)
```

```
## [1] "I Love"        "Biostatistics"
```

STRINGS

- Operations can be performed on *character* variables as well
- Note that “characters” need to be set off by quotation marks to differentiate them from numbers
- The c stands for `concatenate`
- Note that we are using the same variable names as we did previously, which means that we’re overwriting our previous assignment
- A good rule of thumb is to use new names for each variable, and make them short but still descriptive

VECTORS

- In general R thinks in terms of vectors
  - a list of characters, factors or numerical values (“I Love”)
  - it will benefit any R user to try to write scripts with that in mind
  - it will simplify most things
- Vectors can be assigned directly using the ‘c()’ function and then entering the exact values.

VECTORS

```r
n <- c(2, 3, 4, 2, 1, 2, 4, 5, 10, 8, 9)
print(n)
```

```
##  [1]  2  3  4  2  1  2  4  5 10  8  9
```

```r
z <- n + 3
print(z)
```

```
##  [1]  5  6  7  5  4  5  7  8 13 11 12
```

Basic Statistics

Many functions exist to operate on vectors.

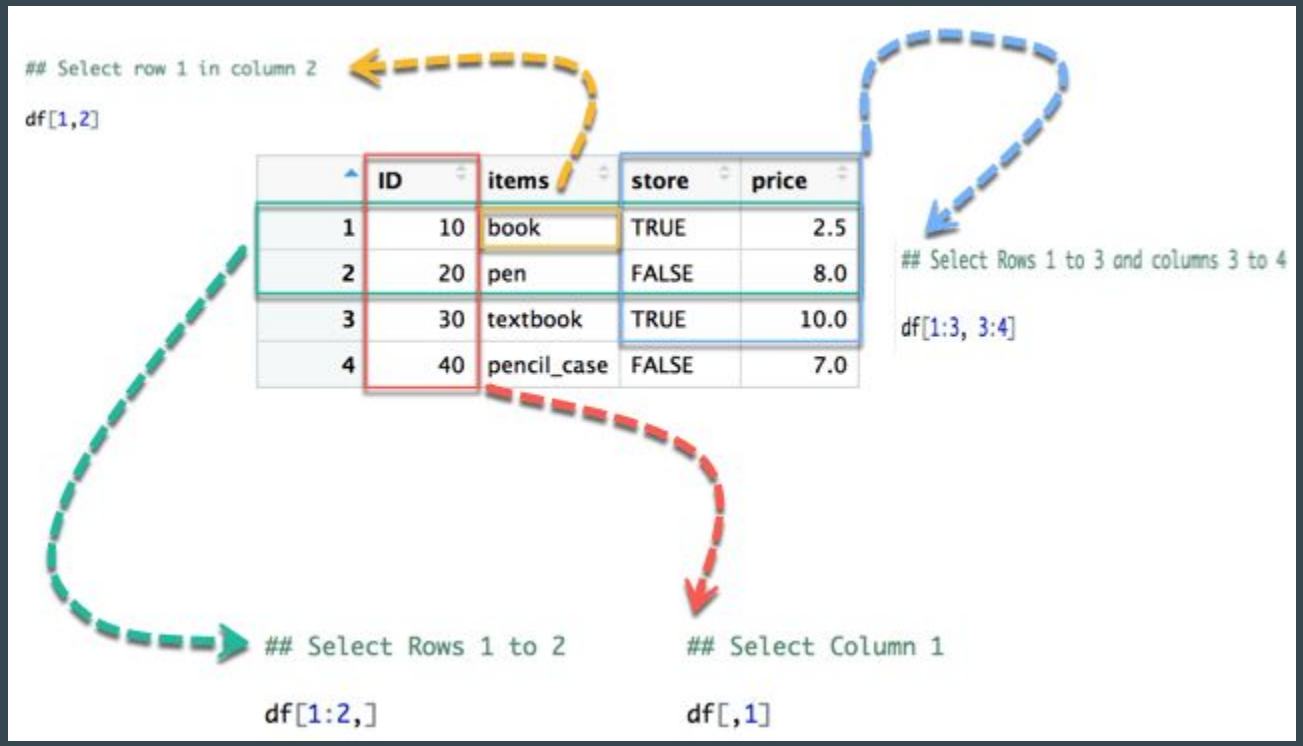
```
mean(n)
median(n)
var(n)
log(n)
exp(n)
sqrt(n)
sum(n)
length(n)
sample(n, replace = T)  #has an additional argument (replace=T)
```

- Arguments modify or direct the function in some way
  - There are many arguments for each function, some of which are defaults
  - Tab complete is helpful to view argument options

Session II

DataFrames

- TSV format best suited for having tables to be imported
- Dataframes are basically structured matrices with specific column and rownames
- Probably most used datatype in R for biology



FACTORS

- The vector `x` is now what is called a list of character values ("I Love").
- Sometimes we would like to treat the characters as if they were units for subsequent calculations.
- These are called `factors`, and we can redefine our character variables as factors.
- This might seem a bit strange, but it's important for statistical analyses where we might want to see the mean or variance for two different treatments.

Types of vectors of data

- `int` stands for integers
- `dbl` stands for doubles, or real numbers
- `chr` stands for character vectors, or strings
- `dttm` stands for date-times (a date + a time)
- `lgl` stands for logical, vectors that contain only TRUE or FALSE
- `fctr` stands for factors, which R uses to represent categorical variables with fixed possible values
- `date` stands for dates

Types of vectors of data

- Logical vectors can take only three possible values:
  - `FALSE`
  - `TRUE`
  - `NA` which is 'not available'.
- Integer and double vectors are known collectively as numeric vectors.
  - In R numbers are doubles by default.
- Integers have one special value: NA, while doubles have four:
  - `NA`
  - `NaN` which is 'not a number'
  - `Inf`
  - `-Inf`