Statistics 133:
Getting Started with R

Why R ?

Why R ?

- Some of you may have used statistical software with a GUI, like Minitab or SPSS. You may also be familiar with other programming languages, like C, Java, Python, etc.
- We will use the R programming language and environment as our "home base" for performing many data analytic tasks.

Why R ?

- Allows custom analysis
- High-level scripting language
- Statistical programming language
- Interactive exploratory data analysis

Why R ?

- Easy to replicate analysis
- Sound numerical methods
- Large Community of contributors
- It’s Free!

Let’s Install **R**

Let’s Install R

- Open a Web browser and go to Google: http://www.google.com/
- Search for R (that’s right – the letter R).
- One of the top links will be to The *R* Project for Statistical Computing,

  which takes you to http:// www.r-project.org/
- At http://www.r-project.org click on CRAN (left menu)
- Select a mirror site near us, i.e. there is a mirror site at **http://cran.stat.ucla.edu**

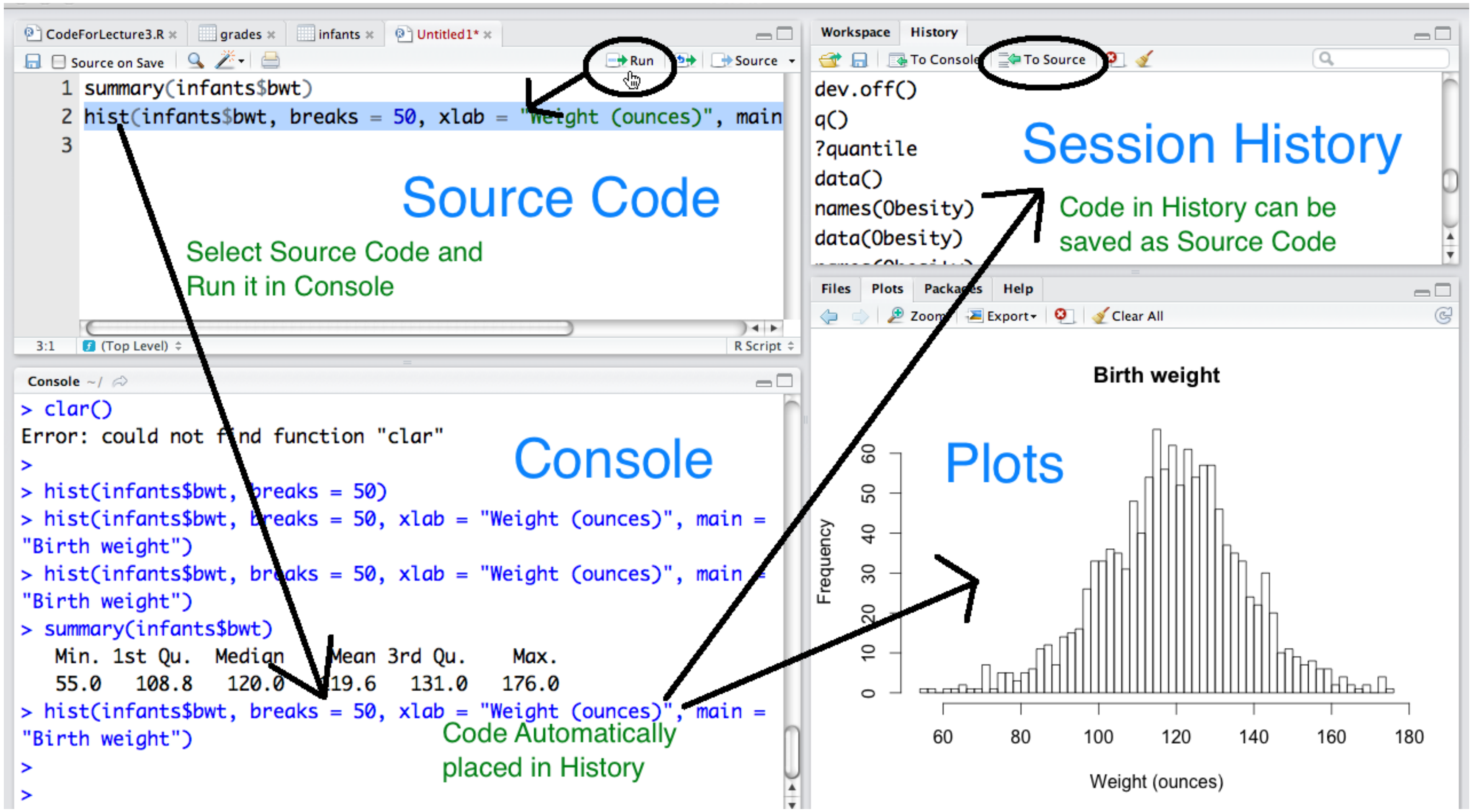
Let's Install RStudio

- Open a Web browser and go to Google: http://www.google.com/
- Search for RStudio
- One of the top links will be

http://www.rstudio.org click on Download tab

- Download the version for running R on your desktop

Let's Try It - RStudio

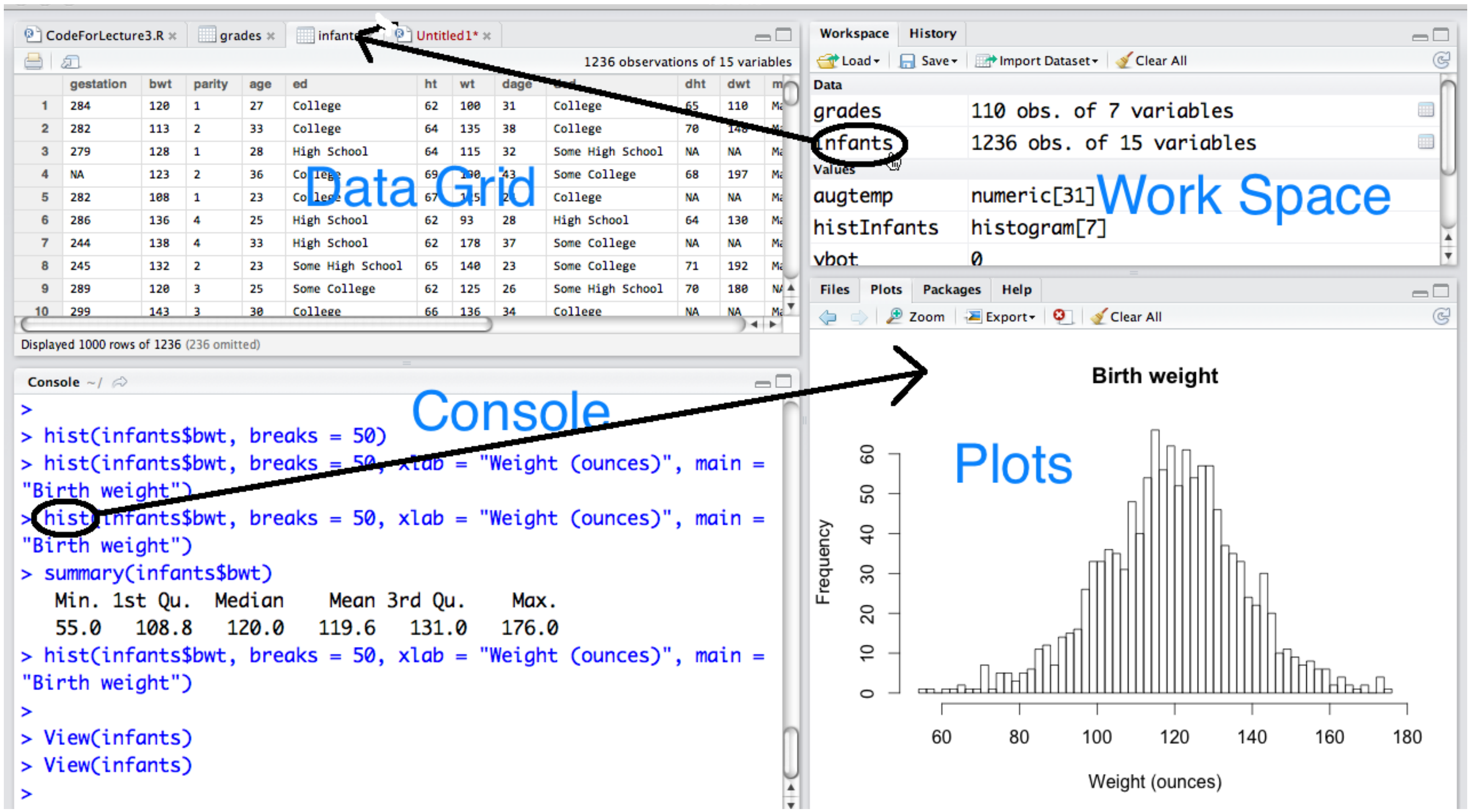


```
summary(infants$bwt)
hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main
```

```
dev.off()
q()
?quantile
data()
names(Obesity)
data(Obesity)
```

```
> clar()
Error: could not find function "clar"
>
> hist(infants$bwt, breaks = 50)
> hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main =
"Birth weight")
> hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main =
"Birth weight")
> summary(infants$bwt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   55.0   108.8   120.0   119.6   131.0   176.0
> hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main =
"Birth weight")
>
>
```

Let’s Try It - RStudio



1236 observations of 15 variables

	gestation	bwt	parity	age	ed	ht	wt	dage	[illegible]	dht	dwt	m
1	284	120	1	27	College	62	100	31	College	65	110	Ma
2	282	113	2	33	College	64	135	38	College	70	140	Ma
3	279	128	1	28	High School	64	115	32	Some High School	NA	NA	Ma
4	NA	123	2	36	College	69	[illegible]	43	Some College	68	197	Ma
5	282	108	1	23	College	67	[illegible]	[illegible]	College	NA	NA	Ma
6	286	136	4	25	High School	62	93	28	High School	64	130	Ma
7	244	138	4	33	High School	62	178	37	Some College	NA	NA	Ma
8	245	132	2	23	Some High School	65	140	23	Some College	71	192	Ma
9	289	120	3	25	Some College	62	125	26	Some High School	70	180	NA
10	299	143	3	30	College	66	136	34	College	NA	NA	Ma

Displayed 1000 rows of 1236 (236 omitted)

Data	
grades	110 obs. of 7 variables
infants	1236 obs. of 15 variables
Values	
augtemp	numeric[31]
histInfants	histogram[7]
vbot	0

Console ~/

```
>
> hist(infants$bwt, breaks = 50)
> hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main =
"Birth weight")
> hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main =
"Birth weight")
> summary(infants$bwt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   55.0   108.8   120.0   119.6   131.0   176.0
> hist(infants$bwt, breaks = 50, xlab = "Weight (ounces)", main =
"Birth weight")
>
> View(infants)
> View(infants)
>
```

Expressions in **R**

Expressions in R

- The R prompt is: >
- At the prompt, type an ***expression***
- Hit the return/enter key
- R ***evaluates*** the expression (performs a ***computation***)
- R returns a value

```
> 2 + 3
```

Returns 5

```
> rnorm(3)
```

Returns 3 random normal values

```
0.1603903 -0.2925857
-0.8274805
```

```
> hist(x)
```

Returns nothing and makes a plot as a side effect

What do expressions look like?

```
2 + 3
9 - 8
4 * 5
10 / 3
7 ^ 2
9 %/% 2
11 %% 7
```

These are simple arithmetic expressions

Similar to what you have with a calculator

Parsing Expressions

- How does R know what computation to perform?
- It breaks down an expression into parts, called tokens
- From these pieces it can figure out what computation to perform

Parsing English

hatheads…

- The above letters are the beginning of something that I am writing.
- Can you figure out what it is?
- What would make it easier for you to do this?

How do we parse English?

- Punctuation: . , ! ? : ;
- Capitalization
- Blank spaces

So What Does hatheads… mean?

Ha! The ad's finished!

"Hat!" He ads on his way out the door.

How does **R** parse expressions?

• White space:	**`22`** vs **`2 2`**
• Atomic Tokens	**`+ - * / ^`**
	`;` end of line
• Quotation Marks	**`#`** comment
	`“Hi”` or **`‘Bye’`** not
• Naming Conventions	**`“My’`**
	`x2` not **`2x`**
• New Line	

How does R parse expressions?

- New Line

```
(2 + 34)
[1] 36
(2 + 3
4)
Error: unexpected
numeric constant in:
"(2 + 3
4"
(2 + 34
)
[1] 36
```

Order of Operations

Order of operations is what you expect:

- exponentiation first, followed by multiplication and division, then addition and subtraction;
- left to right;
- parentheses override order

Try It

- Write the following as an R expression:

power(10, subtract(divide(15,3), 2))

```
10^(15 / 3 – 2)
```

- Write the following as an R expression $\frac{\sqrt{6-2}}{3^2}$

```
(6-2)^0.5 / 3^2
```

- Circle the tokens in the following R expression

```
cat = (1 + x2)^24
```

```
cat = (1 + x2)^24
 ^  ^ ^^ ^ ^ ^^ ^
```

We all make mistakes in writing code

Understanding how R parses expressions will help you fix them

Variables in **R**

Output and Assignment

When we evaluate an expression, R prints the results to the screen as output

- How do we save the result?
- How do we use the output as input to another expression?

Output and Assignment

- We can assign the result of the computation to a variable, e.g., named x:

```
> x = 10^(15 / 3 – 2)
```

- We can use **`x`** as an input in another expression

```
> sqrt(x)
[1] 31.62278
```

- To see the value of a variable type the variable name at the prompt and hit return

```
> x
[1] 1000
```

Output and Assignment

- `=` and `<-` are both valid assignment operators

```
x <- 10^(15 / 3 – 2)
```

- Choose one and use it consistently
- As we'll see `==` means something completely different.

Variables

- Variables have a name and a value
- To access the value we use the name
- Variables allow us to:
  - Store a value without needing to recompute it
  - Write a general expression, e.g. `sqrt(a^2 + b^2)`
  - Reduce redundancy (and mistakes)

Rules for Variable Names

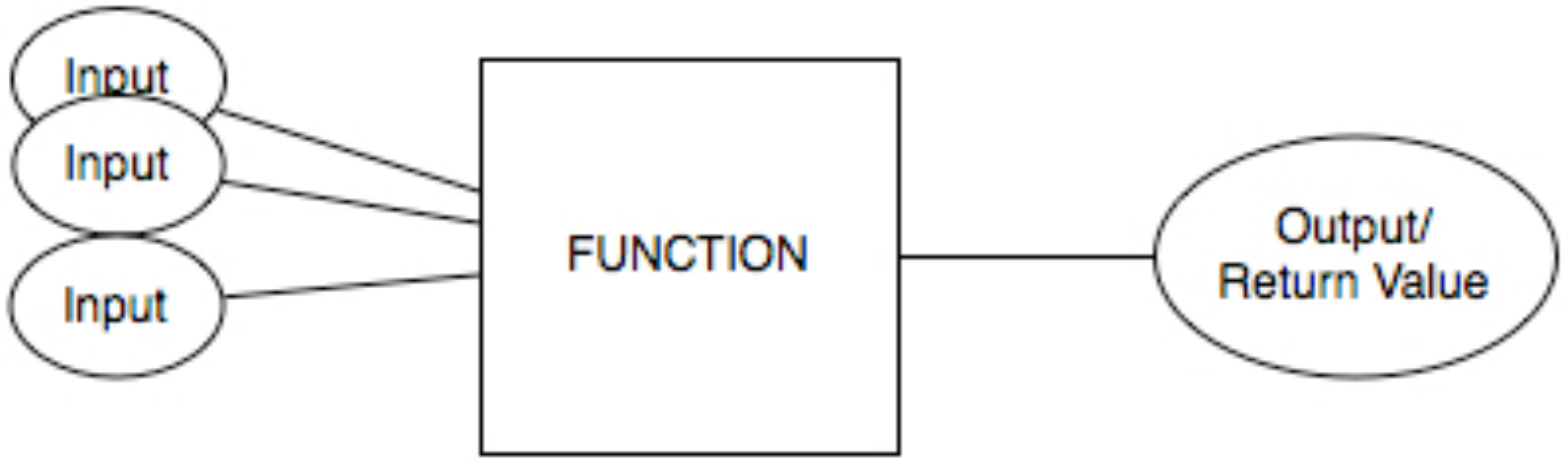
- Variable names must follow some rules
  - May not start with a digit or underscore (_)
  - May contain numbers, characters, and some punctuation - period and underscore are ok, but most others are not
  - Case-sensitive, so x and X are different
- Advice on variable names:
  - Use meaningful names
  - Avoid names that have meaning in R, e.g., function names. If in doubt, check:

```
> exists("pi")
[1] TRUE
```

Function Style Expressions

Function Style Expressions

- Functions contain code (expressions) that perform a specific task.



The Inputs are called
arguments
The output is the ***return value***

Function Style Expressions

- When you use a function with a particular set of arguments, you are said to be ***calling*** the function
- R ***evaluates*** the function call and returns the output
- For now, we will work with R’s built-in functions

Examples Built-in functions

- log()
- exp()
- sqrt()
- abs()
- sin()   cos()

- mean()
- sd()
- median()
- min() max()
  range()
- sum()

Example

Suppose we want to generate 3 random values from a normal curve with center 0 and spread 5.

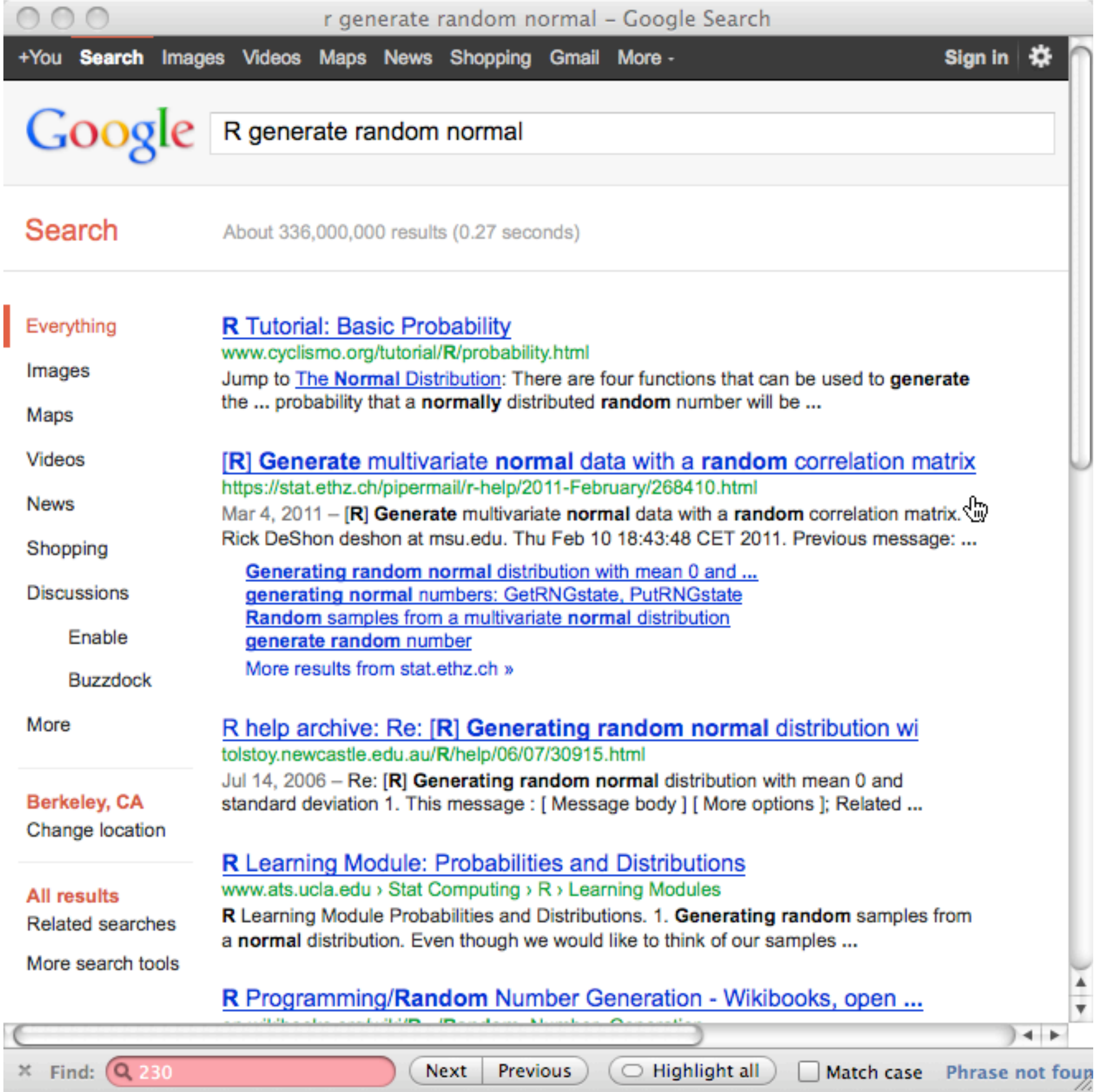
- Is there a function in R that can do this?
- How do we find it?
- How do we call it?

How to find it?

- Use Google search
  - I find searches like the following helpful

  “R random normal”
- R has built-in search

  ```
  help.search(“topic”)
  ```
- Post a question on the class forum



r generate random normal – Google Search

+You Search Images Videos Maps News Shopping Gmail More - Sign in

R generate random normal

Search

About 336,000,000 results (0.27 seconds)

Everything
Images
Maps
Videos
News
Shopping
Discussions
Enable
Buzzdock
More

Berkeley, CA
Change location

All results
Related searches
More search tools

R Tutorial: Basic Probability
www.cyclismo.org/tutorial/R/probability.html
Jump to The Normal Distribution: There are four functions that can be used to generate the ... probability that a normally distributed random number will be ...

[R] Generate multivariate normal data with a random correlation matrix
https://stat.ethz.ch/pipermail/r-help/2011-February/268410.html
Mar 4, 2011 – [R] Generate multivariate normal data with a random correlation matrix. Rick DeShon deshon at msu.edu. Thu Feb 10 18:43:48 CET 2011. Previous message: ...
Generating random normal distribution with mean 0 and ...
generating normal numbers: GetRNGstate, PutRNGstate
Random samples from a multivariate normal distribution
generate random number
More results from stat.ethz.ch »

R help archive: Re: [R] Generating random normal distribution wi
tolstoy.newcastle.edu.au/R/help/06/07/30915.html
Jul 14, 2006 – Re: [R] Generating random normal distribution with mean 0 and standard deviation 1. This message : [Message body] [More options]; Related ...

R Learning Module: Probabilities and Distributions
www.ats.ucla.edu › Stat Computing › R › Learning Modules
R Learning Module Probabilities and Distributions. 1. Generating random samples from a normal distribution. Even though we would like to think of our samples ...

R Programming/Random Number Generation - Wikibooks, open ...

Find: 230 Next Previous Highlight all Match case Phrase not foun

How to call a function

- We call a function as follows:

FunctionName(argument, …, argument)

- Functions can have one or more inputs
- Some arguments are required.
- Other arguments are optional. They have default values so you don't have to specify them

How to call rnorm

- We can find out the arguments to rnorm:

```
> args(rnorm)
function (n, mean = 0, sd = 1)
```

- We see it has 3 arguments: `n`, `mean` and `sd`
- `mean` and `sd` are optional. – they have default values (0 and 1, respectively)
- `n` must be specified – it has no default
- We can learn more with the help function:

```
> ?rnorm
> help(“rnorm”)
```

Generate 3 random values from a normal with center 0 and spread 5

- Generate 3 normals with mean 0 and sd 1.

```
rnorm(3)
```

- Arguments can be identified by position:

```
rnorm(3, 0, 5)
```

- Arguments can be identified by name:

```
rnorm(n = 3, sd = 5)
```

- Use position and name:

`rnorm(3, sd = 5)` (OK for first argument but otherwise be careful)

Simple and Compound Expressions

- Simple Expression: **`rnorm(3, sd = 5)`**
- Compound Expression: **`mean(rnorm(3))`**
- Ill-formed Expressions: **`mean(rnorm(3)]`**

Can you spot what's wrong?

```
Error: unexpected ']' in "mean(rnorm(3)]"
```

Let’s try out what we have learned