1-03: Mathematical Operations

0.1 Changes

1 Purpose

- Perform mathematical operations, including powers, on numerical variables
- Explicit use of mathematical symbols in formulas
- Convert algebraic formulas to programming formulas
- Use parentheses to establish the order of operations for formulas

2 Questions about the material...

If you have any questions about the material in this lesson, feel free to email them to the instructor, Charlie Belinsky, at belinsky@msu.edu.

3 Putting a formula in code

Once again, we will calculate ***velocity*** using ***distance*** and ***time,*** except, we will now use the full version of the ***velocity*** formula, which looks at the changes in distance and time as opposed to absolute distance and time.

The full velocity formula is (subscript ***i*** means initial, subscript ***f*** means final):
$v = \frac{d_f - d_i}{t_f - t_i}$.

Note: If $\boldsymbol{t_i}$ and $\boldsymbol{d_i}$ are zero then you get the formula we used in Lesson 1-2: ($\boldsymbol{v} = \frac{d}{t}$)

We are going to code this formula, but there are a couple of issues:

1. fractions are "stacked" – but in scripts, equations can only be read left to right
2. variable names have subscript characters (e.g., $\boldsymbol{t_i}$), but subscript and superscript characters are not allowed in script

3.1 Minding your parentheses

In script, everything goes left-to-right so you cannot write a fraction as you would in Algebra. Instead, we need to be more explicit and put both both the numerator and the denominator in parentheses:

$$v = \frac{(d_f - d_i)}{(t_f - t_i)}$$

Then, pull out the fraction between the numerator and denominator and replace it with a division sign:

$$v = (d_f - d_i)/(t_f - t_i)$$

Now the formula is all on one line, but the symbols need to be replaced with valid variable names that do not have subscripts:

$$\text{velocity} = (\text{finalDist} - \text{initDist})/(\text{finalTime} - \text{initTime})$$

This is now a valid line of code in R, assuming all four variables on the right side have assigned values.

```
velocity = (finalDist - initDist) / (finalTime - initTime);
```

The line of code above says that ***velocity*** will be assigned the value equal to the calculations of the four variables on the right side of the equation.

3.2 Other ways to assign values

In most programming languages the equal sign is used to assign values, and the equal sign always evaluates what is on the right side and assigns it to the variable on the left. In R, you can use arrows to assign values:

```
velocity <- (finalDist - initDist) / (finalTime - initTime); # commonly used
(finalDist - initDist) / (finalTime - initTime) -> velocity; # rarely used
```

The top (**<-**) is the most commonly used in R and the bottom (**->**) works but is rarely used anymore. I prefer using (**=**) to (**<-**) because (**=**) is used in most programming languages whereas (**<-**) is not.

Note: In this case, (**=**) and (**<-**) are functionally the same. There are differences between the two, which we will talk about in the lesson on functions.

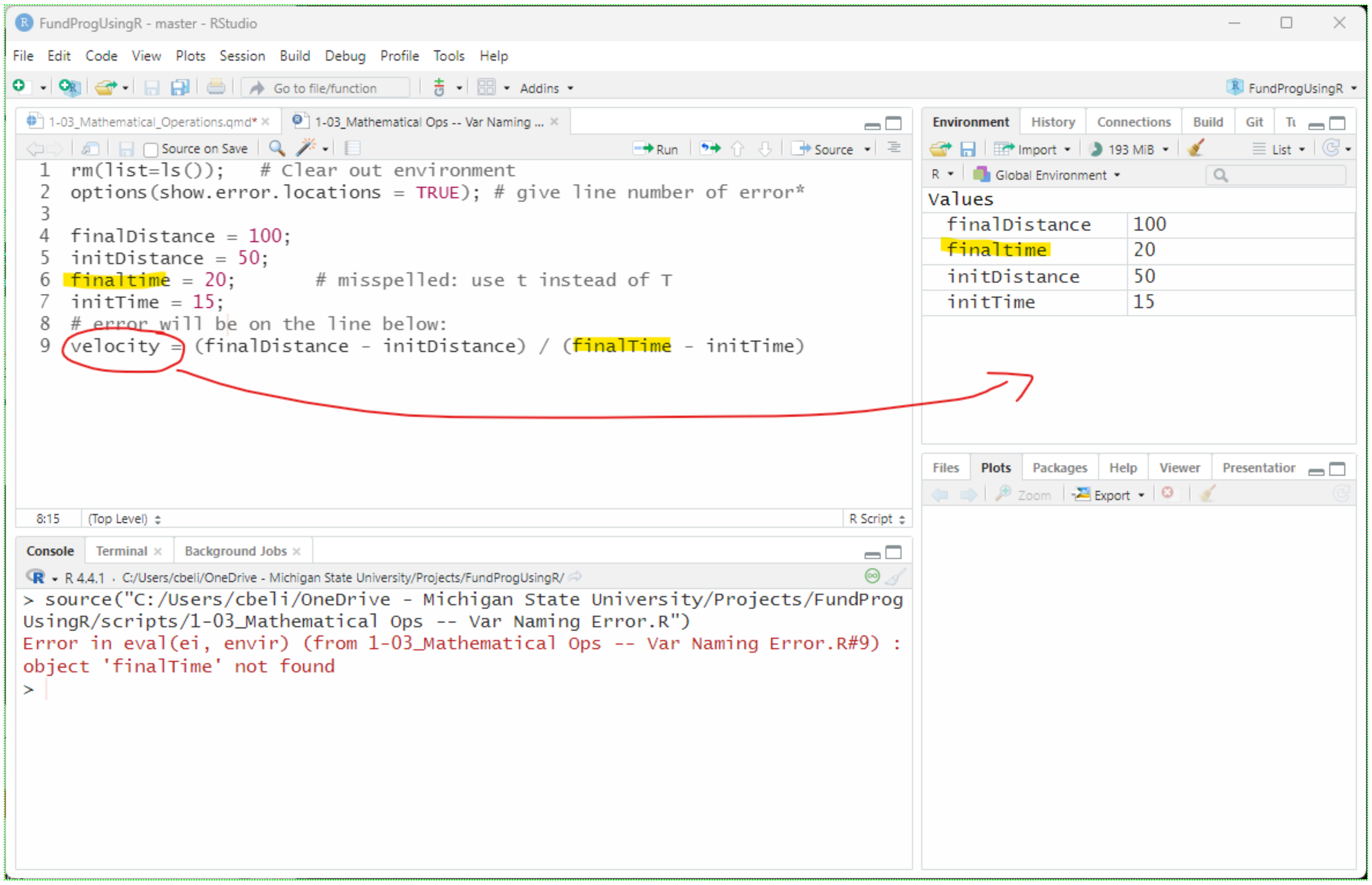
3.3 Variable naming error

Here is the full script with a small error on the line calculating ***velocity***:

```
rm(list=ls());   # Clear out environment
options(show.error.locations = TRUE); # give line number of error*

finalDistance = 100;
initDistance = 50;
finaltime = 20;       # misspelled: use t instead of T
initTime = 15;
# error will be on the line below:
velocity = (finalDistance - initDistance) / (finalTime -
         initTime)
```

Every variable on the right side of the ***velocity*** equation must be given a value beforehand – otherwise, you will get the pesky Object not found error as shown in the image below (Figure 1)


FundProgUsingR - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
FundProgUsingR
1-03_Mathematical_Operations.qmd*
1-03_Mathematical Ops -- Var Naming ...
Source on Save
Run
Source
8:15
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.4.1 · C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/
Environment
History
Connections
Build
Git
Import
193 MiB
List
R
Global Environment
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export

```r
rm(list=ls());   # Clear out environment
options(show.error.locations = TRUE); # give line number of error*

finalDistance = 100;
initDistance = 50;
finaltime = 20;        # misspelled: use t instead of T
initTime = 15;
# error will be on the line below:
velocity = (finalDistance - initDistance) / (finalTime - initTime)
```

Values

finalDistance	100
finaltime	20
initDistance	50
initTime	15

```
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProg
UsingR/scripts/1-03_Mathematical Ops -- Var Naming Error.R")
Error in eval(ei, envir) (from 1-03_Mathematical Ops -- Var Naming Error.R#9) :
object 'finalTime' not found
>
```

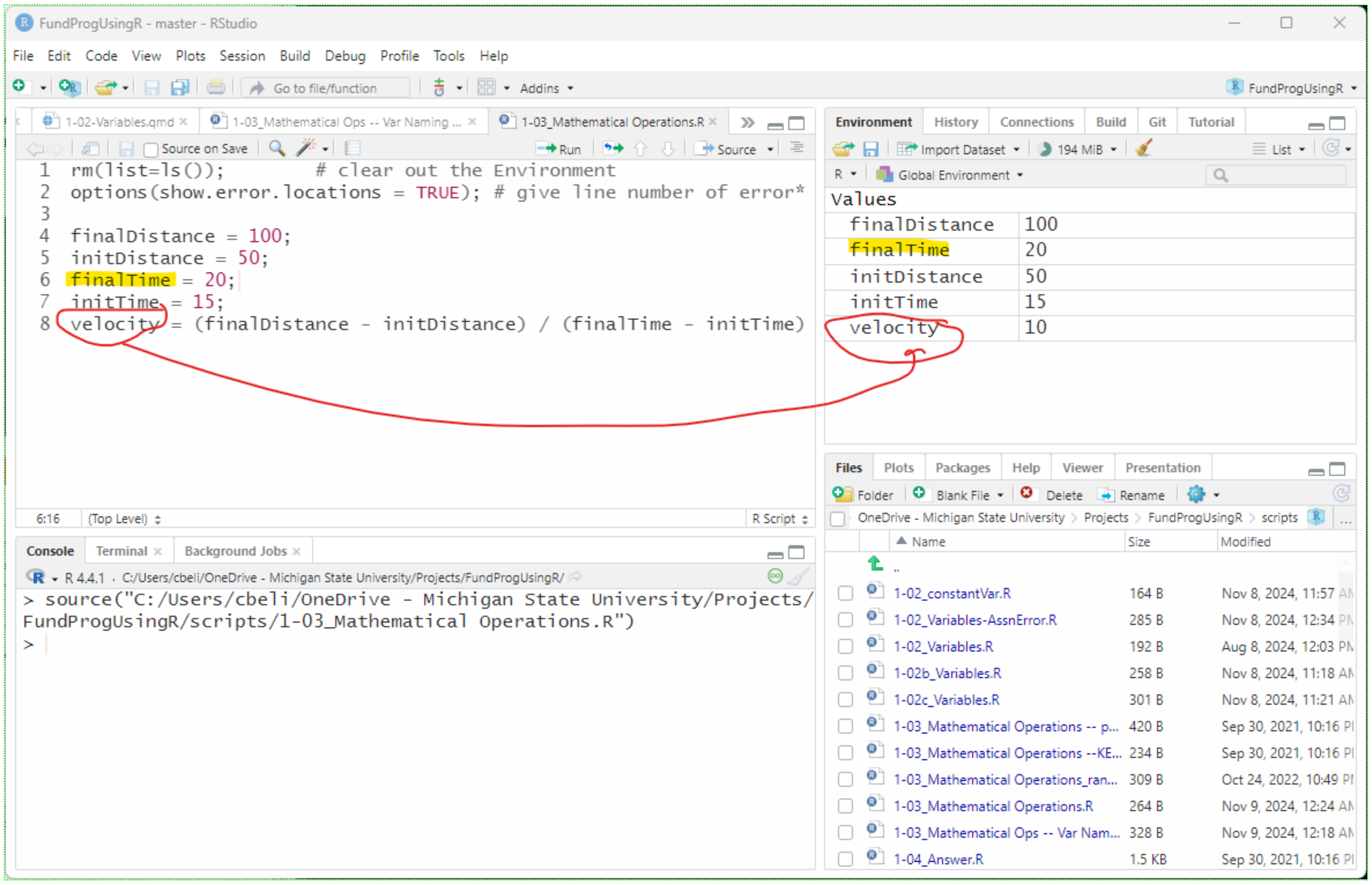
Figure 1: Object not found because the variable name does not exist (it is misspelled).

Extension: The show.error.location line

Note: **Object** is almost synonymous with **Variable** in R. The error is basically saying that there is no variable with that name. Any spelling error will cause the Object not found error. In this case I "spelled" the variable name wrong by changing the case of the ***T***. **finaltime** is not the same as **finalTime.**

```r
rm(list=ls());            # clean out the environment
options(show.error.locations = TRUE); # give line number of error*

finalDistance = 100;
initDistance = 50;
finalTime = 20;
initTime = 15;
velocity = (finalDistance - initDistance) / (finalTime - initTime);
```


FundProgUsingR - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
1-02-Variables.qmd
1-03_Mathematical Ops -- Var Naming ...
1-03_Mathematical Operations.R
Environment History Connections Build Git Tutorial
Import Dataset
194 MiB
Global Environment
Files Plots Packages Help Viewer Presentation
Console Terminal Background Jobs
6:16 (Top Level) R Script

```
rm(list=ls());        # clear out the Environment
options(show.error.locations = TRUE); # give line number of error*

finalDistance = 100;
initDistance = 50;
finalTime = 20;
initTime = 15;
velocity = (finalDistance - initDistance) / (finalTime - initTime)
```

Values

finalDistance	100
finalTime	20
initDistance	50
initTime	15
velocity	10

```
R 4.4.1 · C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/
FundProgUsingR/scripts/1-03_Mathematical Operations.R")
>
```

OneDrive - Michigan State University > Projects > FundProgUsingR > scripts

Name	Size	Modified
..		
1-02_constantVar.R	164 B	Nov 8, 2024, 11:57 A
1-02_Variables-AssnError.R	285 B	Nov 8, 2024, 12:34 P
1-02_Variables.R	192 B	Aug 8, 2024, 12:03 P
1-02b_Variables.R	258 B	Nov 8, 2024, 11:18 A
1-02c_Variables.R	301 B	Nov 8, 2024, 11:21 A
1-03_Mathematical Operations -- p...	420 B	Sep 30, 2021, 10:16 P
1-03_Mathematical Operations --KE...	234 B	Sep 30, 2021, 10:16 P
1-03_Mathematical Operations_ran...	309 B	Oct 24, 2022, 10:49 P
1-03_Mathematical Operations.R	264 B	Nov 9, 2024, 12:24 A
1-03_Mathematical Ops -- Var Nam...	328 B	Nov 9, 2024, 12:18 A
1-04_Answer.R	1.5 KB	Sep 30, 2021, 10:16 P

Figure 2: Misspelled variable name corrected – script now calculates velocity.

4 Powers and multiplication

We will look at one more formula that relates ***kinetic energy*** to ***mass*** and ***velocity***:

$$E_k = \frac{1}{2}mv^2$$

There are two new issues with coding this formula:

1. the square function is a superscript – you cannot use superscript characters in R
2. the implicit multiplication – we know that mass (**m**) and velocity (**v**) are being multiplied, but there is no multiplication sign

4.1 Dealing with parenthesis and multiplication

So let's first pull the ***one-half*** out of fraction form and into division form:

$$E_k = 1/2mv^2$$

We need to be more explicit because this formula could be misinterpreted by the reader as $E_k = 1/\left(2mv^2\right)$, so we need to put the one-half in parenthesis:

$$E_k = (1/2)mv^2$$

Next, we will explicitly put in the multiplication symbols – a necessity in programming:

$$E_k = (1/2)^* m^* v^2$$

Traps: Forgetting Multiplication Symbol

And then change the symbols to script-friendly variable names:

$$\text{kineticEnergy } = (1/2)^* \text{ mass }^* \text{ velocity }^2$$

4.2 Dealing with square power

In R the (^) is the power operator. So ***^2*** means raise to the power of 2 (i.e., square):

```
# this formula works...
kineticEnergy = (1/2)*mass*velocity^2;
```

While the above works correctly, it is often helpful to be explicit and add parenthesis around the value or values that are getting raised to the power:

```
# more explicit solution
kineticEnergy = (1/2)*mass*(velocity)^2;
```

5 The power operator (^)

The (**^**) operator works for all powers including square roots, cubed roots, and mixed powers (e.g., raising to the 3/2 or 5/3).

Let's rearrange the ***kinetic energy*** formula to solve for ***velocity***, which requires a square root

$$v = \sqrt{\frac{2E_k}{m}}$$

To put the above formula into a script form, we need to:
1) Put the numerator and denominator on one line by taking out the fraction and replacing it with a division sign.

$$v = \sqrt{(2E_k/m)}$$

2) Be explicit and put in multiplication symbols.

$$v = \sqrt{(2^* E_k/m)}$$

3) Spell the formula out using script-friendly variable names:

$$\text{velocity} \;=\; \sqrt{(2^{*}\ \text{kineticEnergy}\ /\ \text{mass}\ )}$$

4) Use the power operator (**^**) to square root the whole formula. Square rooting something is the same as saying raise it to the 1/2 power. Since we are square rooting the whole formula, we need to put the whole formula in parenthesis.

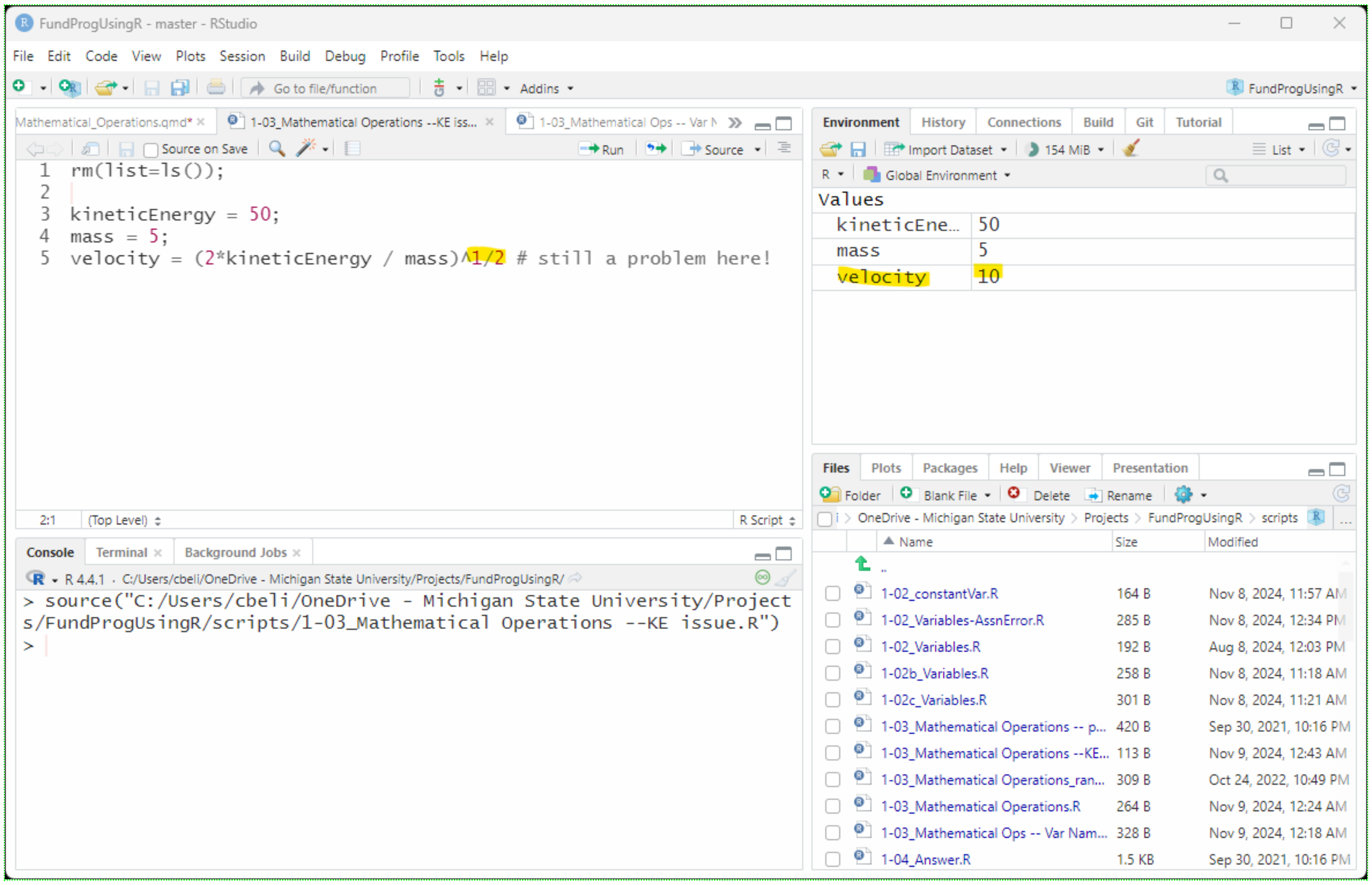
$$\text{velocity} \;=\; (2^{*}\ \text{kineticEnergy}\ /\ \text{mass}\ )^{1/2}$$

5.1 Coding the power

So we have this in R:

```r
rm(list=ls());          # clean out the environment

kineticEnergy = 50;
mass = 5;
velocity = (2*kineticEnergy / mass)^1/2; # still a problem here!
```



```r
rm(list=ls());

kineticEnergy = 50;
mass = 5;
velocity = (2*kineticEnergy / mass)^1/2 # still a problem here!
```

Values

kineticEne...	50
mass	5
velocity	10

```
> source("C:/Users/cbeli/OneDrive - Michigan State University/Project
s/FundProgUsingR/scripts/1-03_Mathematical Operations --KE issue.R")
>
```

Name	Size	Modified
..		
1-02_constantVar.R	164 B	Nov 8, 2024, 11:57 AM
1-02_Variables-AssnError.R	285 B	Nov 8, 2024, 12:34 PM
1-02_Variables.R	192 B	Aug 8, 2024, 12:03 PM
1-02b_Variables.R	258 B	Nov 8, 2024, 11:18 AM
1-02c_Variables.R	301 B	Nov 8, 2024, 11:21 AM
1-03_Mathematical Operations -- p...	420 B	Sep 30, 2021, 10:16 PM
1-03_Mathematical Operations --KE...	113 B	Nov 9, 2024, 12:43 AM
1-03_Mathematical Operations_ran...	309 B	Oct 24, 2022, 10:49 PM
1-03_Mathematical Operations.R	264 B	Nov 9, 2024, 12:24 AM
1-03_Mathematical Ops -- Var Nam...	328 B	Nov 9, 2024, 12:18 AM
1-04_Answer.R	1.5 KB	Sep 30, 2021, 10:16 PM

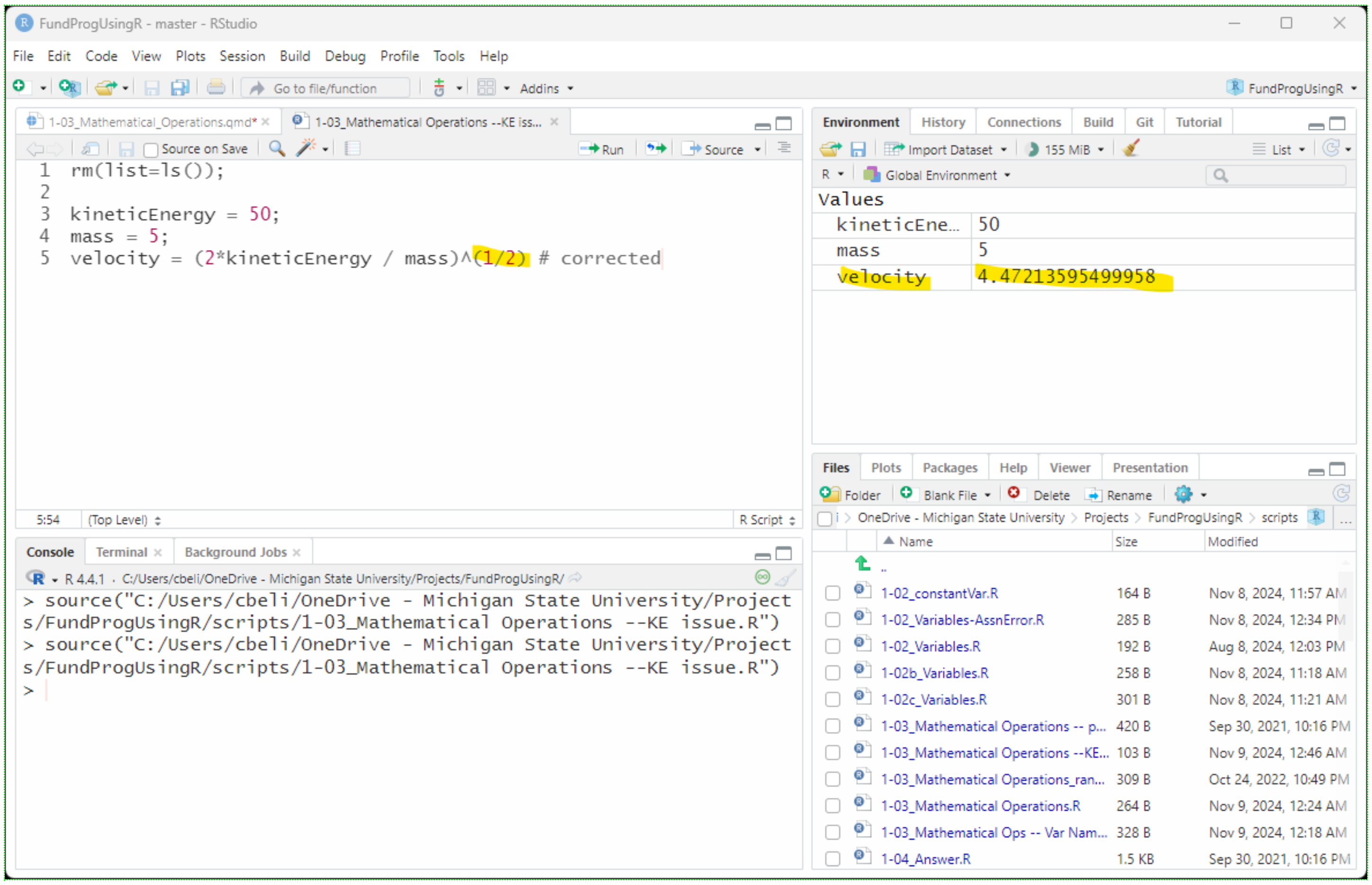
Figure 3: Incorrect answer for velocity because the power is missing parentheses

5.2 Correcting the power with parenthses

The ***Environment*** tab (Figure 3) shows that ***v*** is, unexpectedly, ***10***. This is because of the order-of-operations. Instead of raising the ***(2*kineticEnergy/mass)*** to the ***1/2*** power, the above code raised ***(2*kineticEnergy/mass)*** to the ***first (1)*** power and then divided everything by ***2***. We need to be more explicit and put the ***1/2*** in parenthesis.

```
rm(list=ls());          # clean out the environment

kineticEnergy = 50;
mass = 5;
velocity = (2*kineticEnergy / mass)^(1/2); # now we are good!
```


FundProgUsingR - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
FundProgUsingR
1-03_Mathematical_Operations.qmd*
1-03_Mathematical Operations --KE iss...
Source on Save
Run
Source
Environment History Connections Build Git Tutorial
Import Dataset
155 MiB
List
R
Global Environment
5:54
(Top Level)
R Script
Console Terminal Background Jobs
Files Plots Packages Help Viewer Presentation
Folder
Blank File
Delete
Rename

```
rm(list=ls());

kineticEnergy = 50;
mass = 5;
velocity = (2*kineticEnergy / mass)^(1/2) # corrected
```

```
Values
kineticEne...  50
mass           5
velocity       4.47213595499958
```

```
R 4.4.1 · C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/
> source("C:/Users/cbeli/OneDrive - Michigan State University/Project
s/FundProgUsingR/scripts/1-03_Mathematical Operations --KE issue.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Project
s/FundProgUsingR/scripts/1-03_Mathematical Operations --KE issue.R")
>
```

OneDrive - Michigan State University > Projects > FundProgUsingR > scripts

Name	Size	Modified
..		
1-02_constantVar.R	164 B	Nov 8, 2024, 11:57 AM
1-02_Variables-AssnError.R	285 B	Nov 8, 2024, 12:34 PM
1-02_Variables.R	192 B	Aug 8, 2024, 12:03 PM
1-02b_Variables.R	258 B	Nov 8, 2024, 11:18 AM
1-02c_Variables.R	301 B	Nov 8, 2024, 11:21 AM
1-03_Mathematical Operations -- p...	420 B	Sep 30, 2021, 10:16 PM
1-03_Mathematical Operations --KE...	103 B	Nov 9, 2024, 12:46 AM
1-03_Mathematical Operations_ran...	309 B	Oct 24, 2022, 10:49 PM
1-03_Mathematical Operations.R	264 B	Nov 9, 2024, 12:24 AM
1-03_Mathematical Ops -- Var Nam...	328 B	Nov 9, 2024, 12:18 AM
1-04_Answer.R	1.5 KB	Sep 30, 2021, 10:16 PM

Figure 4: Correct answer for velocity

5.3 More power and roots

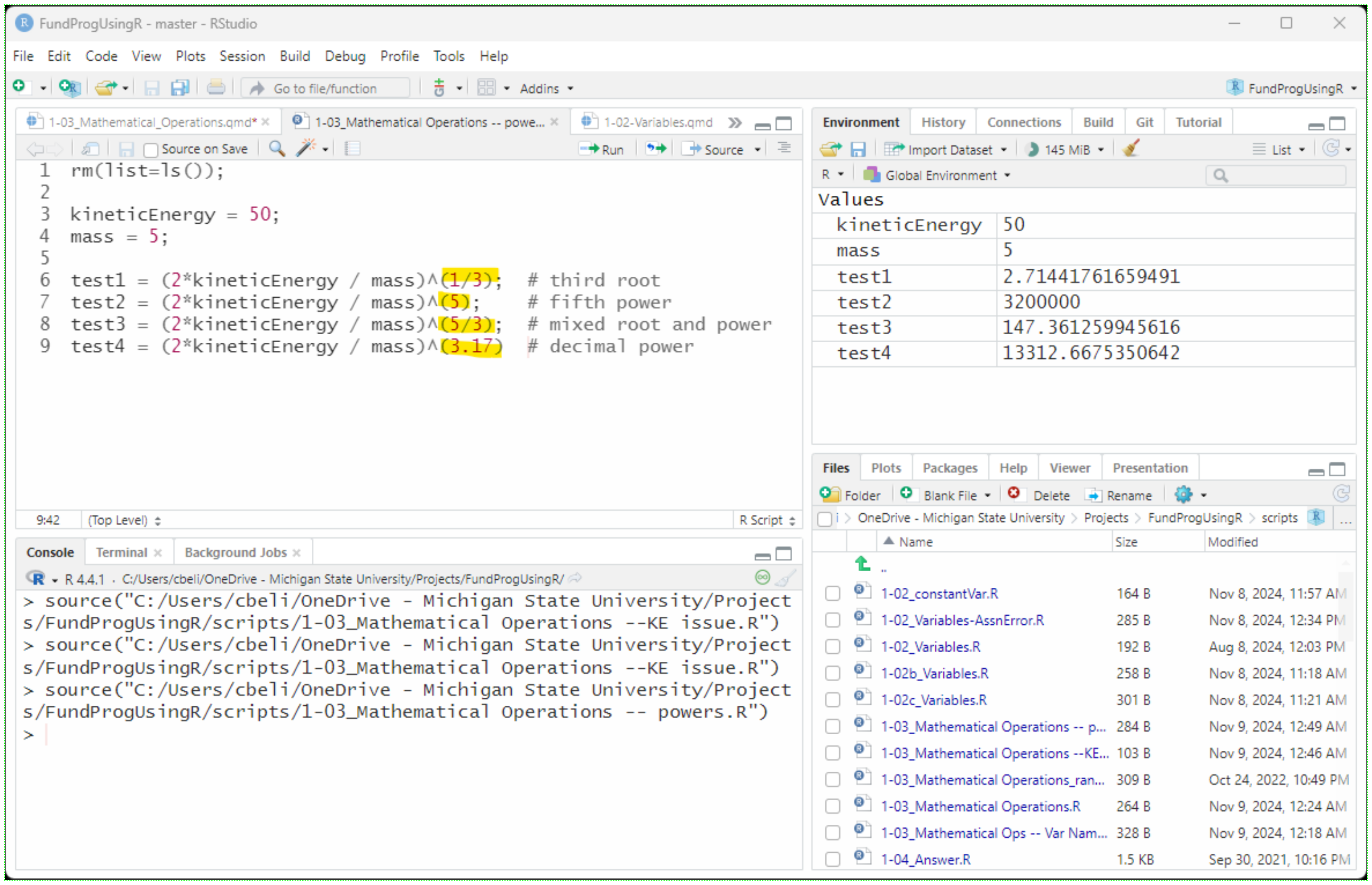
This style will work for all powers and roots:

```
rm(list=ls());          # clean out the environment

kineticEnergy = 50;
mass = 5;

test1 = (2*kineticEnergy / mass)^(1/3);  # third root
test2 = (2*kineticEnergy / mass)^(5);    # fifth power
```

```
test3 = (2*kineticEnergy / mass)^(5/3);  # mixed root and power
test4 = (2*kineticEnergy / mass)^(3.17); # decimal power
```


FundProgUsingR - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
FundProgUsingR
1-03_Mathematical_Operations.qmd*
1-03_Mathematical Operations -- powe...
1-02-Variables.qmd
Source on Save
Run
Source
Environment History Connections Build Git Tutorial
Import Dataset
145 MiB
List
R
Global Environment
9:42
(Top Level)
R Script
Console Terminal Background Jobs
R 4.4.1 · C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/
Files Plots Packages Help Viewer Presentation
Folder
Blank File
Delete
Rename
OneDrive - Michigan State University > Projects > FundProgUsingR > scripts

```
rm(list=ls());

kineticEnergy = 50;
mass = 5;

test1 = (2*kineticEnergy / mass)^(1/3);  # third root
test2 = (2*kineticEnergy / mass)^(5);    # fifth power
test3 = (2*kineticEnergy / mass)^(5/3);  # mixed root and power
test4 = (2*kineticEnergy / mass)^(3.17)  # decimal power
```

Values

kineticEnergy	50
mass	5
test1	2.71441761659491
test2	3200000
test3	147.361259945616
test4	13312.6675350642

```
> source("C:/Users/cbeli/OneDrive - Michigan State University/Project
s/FundProgUsingR/scripts/1-03_Mathematical Operations --KE issue.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Project
s/FundProgUsingR/scripts/1-03_Mathematical Operations --KE issue.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Project
s/FundProgUsingR/scripts/1-03_Mathematical Operations -- powers.R")
>
```

Name	Size	Modified
..		
1-02_constantVar.R	164 B	Nov 8, 2024, 11:57 AM
1-02_Variables-AssnError.R	285 B	Nov 8, 2024, 12:34 PM
1-02_Variables.R	192 B	Aug 8, 2024, 12:03 PM
1-02b_Variables.R	258 B	Nov 8, 2024, 11:18 AM
1-02c_Variables.R	301 B	Nov 8, 2024, 11:21 AM
1-03_Mathematical Operations -- p...	284 B	Nov 9, 2024, 12:49 AM
1-03_Mathematical Operations --KE...	103 B	Nov 9, 2024, 12:46 AM
1-03_Mathematical Operations_ran...	309 B	Oct 24, 2022, 10:49 PM
1-03_Mathematical Operations.R	264 B	Nov 9, 2024, 12:24 AM
1-03_Mathematical Ops -- Var Nam...	328 B	Nov 9, 2024, 12:18 AM
1-04_Answer.R	1.5 KB	Sep 30, 2021, 10:16 PM

Figure 5: Testing different powers in R

6 The square root function

In R, you will usually see square roots done using the ***sqrt()*** function:

```
velocity = sqrt(2*kineticEnergy / mass);  # how square roots are usually done
```

sqrt() works just fine but there is no equivalent for all the other types of powers and roots. That is why I prefer to use the (**^**) operator – it is easy to remember and you can use it for every power and root situation – you just need to attach the appropriate number.

```
velocity = (2*kineticEnergy / mass)^(1/2); # how I prefer to do them
```

7 Random values

In Figure 5 we hardcoded the values for ***kineticEnergy*** and ***mass***, which just means we directly provided a values for the two variables. We can also randomly pick values for variables using ***sample()***.

sample() requires two arguments:

- the range of numbers you want to randomly sample from
- the number of values you want to randomly sample (for now we will just do one)

The code to pick one random number between 20 and 100 is:

```
randomNum1 = sample(20:100, size=1);
```

note: 20:100 is inclusive of the numbers on both ends – so, 20 and 100 are both possibilities meaning there are 81 possible numbers to choose randomly from

The code to pick a random number between -100 and -50 is:

```
randomNum2 = sample(-100:-50, size=1);
```

sample() always returns a whole number.

7.1 normal values

sample() treats every number the same. If there are 100 numbers in the range then every number has a 1% chance of being picked. If you want to pick a random value, but weigh the value (e.g., a normally distributed random values) then you can use ***rnorm()***.

rnorm() requires three arguments:

- the number of values you want to randomly pick (for now, we will just pick 1 at a time)
- the mean of the normal distribution you want to pick randomly from
- the standard deviation of the normal distribution

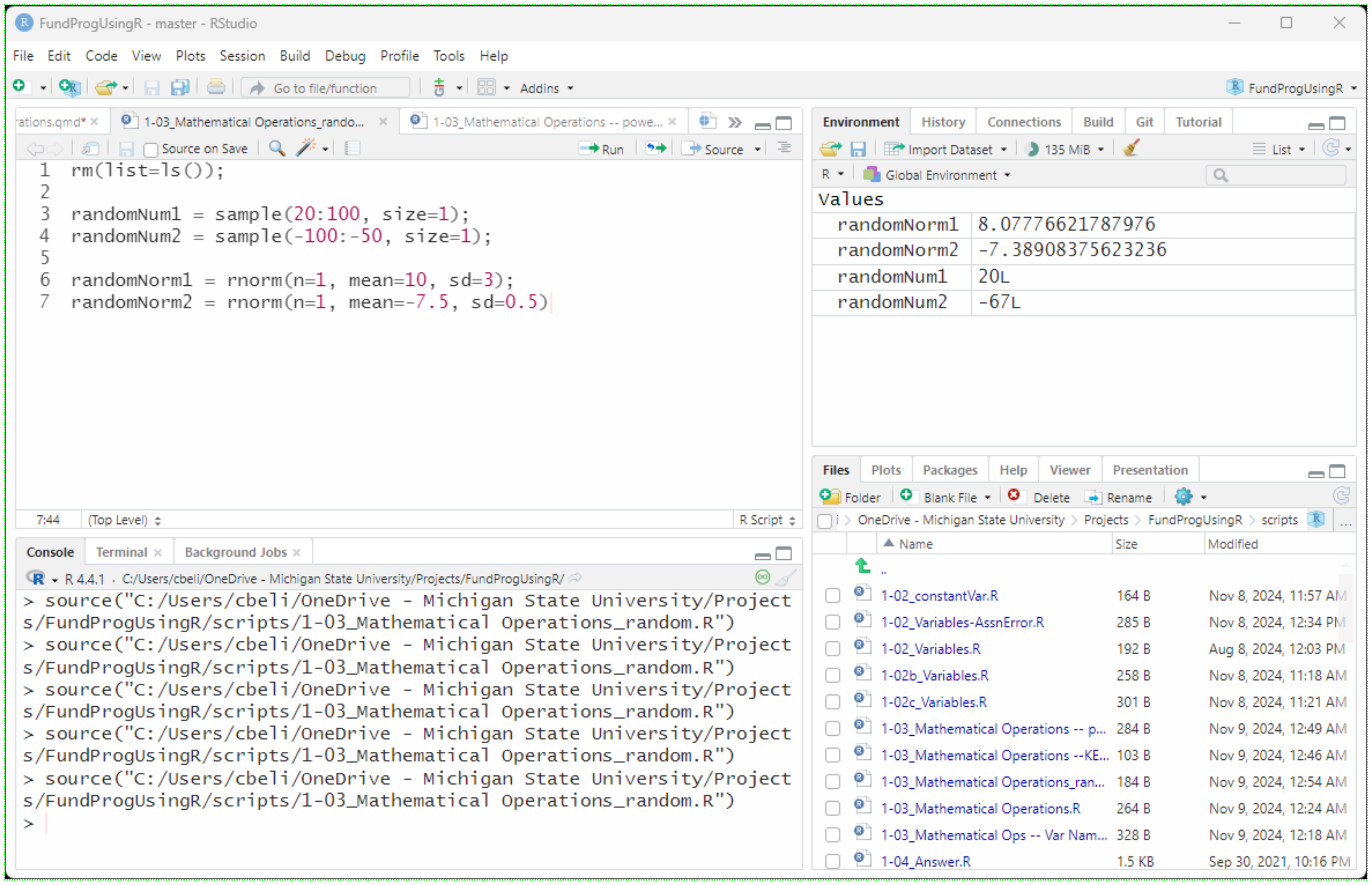
The code to pick one random number from a normal distribution with mean 10 and standard deviation 3 is:

```
randomNorm1 = rnorm(n=1, mean=10, sd=3);
```

The code to pick one random number from a normal distribution with mean -7.5 and standard deviation 0.5 is:

```
randomNorm2 = rnorm(n=1, mean=-7.5, sd=0.5);
```

rnorm() always returns a decimal number


FundProgUsingR - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Environment History Connections Build Git Tutorial
Files Plots Packages Help Viewer Presentation
Console Terminal Background Jobs

```
rm(list=ls());

randomNum1 = sample(20:100, size=1);
randomNum2 = sample(-100:-50, size=1);

randomNorm1 = rnorm(n=1, mean=10, sd=3);
randomNorm2 = rnorm(n=1, mean=-7.5, sd=0.5)
```

```
Values
randomNorm1   8.07776621787976
randomNorm2   -7.38908375623236
randomNum1    20L
randomNum2    -67L
```

```
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R")
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R")
>
```

Name	Size	Modified
1-02_constantVar.R	164 B	Nov 8, 2024, 11:57 AM
1-02_Variables-AssnError.R	285 B	Nov 8, 2024, 12:34 PM
1-02_Variables.R	192 B	Aug 8, 2024, 12:03 PM
1-02b_Variables.R	258 B	Nov 8, 2024, 11:18 AM
1-02c_Variables.R	301 B	Nov 8, 2024, 11:21 AM
1-03_Mathematical Operations -- p...	284 B	Nov 9, 2024, 12:49 AM
1-03_Mathematical Operations --KE...	103 B	Nov 9, 2024, 12:46 AM
1-03_Mathematical Operations_ran...	184 B	Nov 9, 2024, 12:54 AM
1-03_Mathematical Operations.R	264 B	Nov 9, 2024, 12:24 AM
1-03_Mathematical Ops -- Var Nam...	328 B	Nov 9, 2024, 12:18 AM
1-04_Answer.R	1.5 KB	Sep 30, 2021, 10:16 PM

Figure 6: Picking random numbers – the random numbers will be different each time you execute the code

8 Application

1) Add this code right after the ***rm(list=ls())*** lines:

```
set.seed(5);
```

The above line will make sure that you get the same “random” number every time you execute your script by creating a seed value. Seed values are covered in a much later lesson.

2) Create six variables that all hold length values:

- 1st and 2nd are assigned the values: 25, 30
- 3rd and 4th are randomly picked between 20 and 30 (each number has an equal chance)
- 5th and 6th are randomly picked from a normal distribution with mean of 25 and standard deviation of 2

3) Calculate the (a) mean, (b) variance, and (c) standard deviation of the six values.

- Visit this page if you need a reminder about how to calculate mean, variance, and standard deviation
- you can use R functions like ***mean(), var(),*** and ***sd()*** to check these values, but I want you to manually solve these – it is important to learn how to properly code the mathematics because you will often not have functions to do it for you.

- Extension: Mathematical operations across multiple lines

4) Make sure the 6 values, their mean, their variance, and their standard deviation appear in the ***Environment*** tab after the script is executed.

5) Challenge: Pick a random two-digit decimal number between 0 and 1 (e.g., 0.23, 0.89, 0.10)
- you will need to use ***sample()*** and then manipulate the number

Save the script as ***app1-03.r*** in your ***scripts*** folder and email your Project Folder to Charlie Belinsky at belinsky@msu.edu.

Instructions for zipping the Project Folder are here.

If you have any questions regarding this application, feel free to email them to Charlie Belinsky at belinsky@msu.edu.

8.1 Questions to answer

Answer the following in comments inside your application script:
1. What was your level of comfort with the lesson/application?
2. What areas of the lesson/application confused or still confuses you?
3. What are some things you would like to know more about that is related to, but not covered in, this lesson?

9 Extension: Mathematical operations across multiple lines

If you have a long mathematical formula to execute in code, there is a good chance that you will want to break the code up into multiple lines.

To keep it simple, let's add 5 values together across multiple lines:

```
c = 10 + 10 + 10 +
    10 + 10;
```

If you put the above code in you script, then you will get ***c = 50*** in the ***Environment.***

However, this code:

```
d = 10 + 10 + 10
    + 10 + 10;
```

will put ***d=30*** in the ***Environment***.

This is because R did not know the continue the equation for ***d*** to the second line. R treated the first line as the complete command/equation. By putting the (+) at the end of the first line for ***c***, R knew it needed to continue the command to the next line.

By the way, R does do something with the second line for ***d*** – it prints the answer to ***Console*** if you click ***Run***:

```
> d = 10 + 10 + 10
>     + 10 + 10;
[1] 20
```

In other words:

```
+ 10 + 10
```

is a command that evaluates to **20**

10 Traps: Forgetting Multiplication Symbol

Let's say you are solving for kinetic energy:

$$E_k = \frac{1}{2}mv^2$$

And you have a value for velocity (***v***) and mass (***m***)

```
rm(list=ls());          # clean out the environment

m = 100;
v = 10;
KE = ????;    # should be: KE = 
         (1/2)*m*v^2;
```

If you code ***KE*** like this::

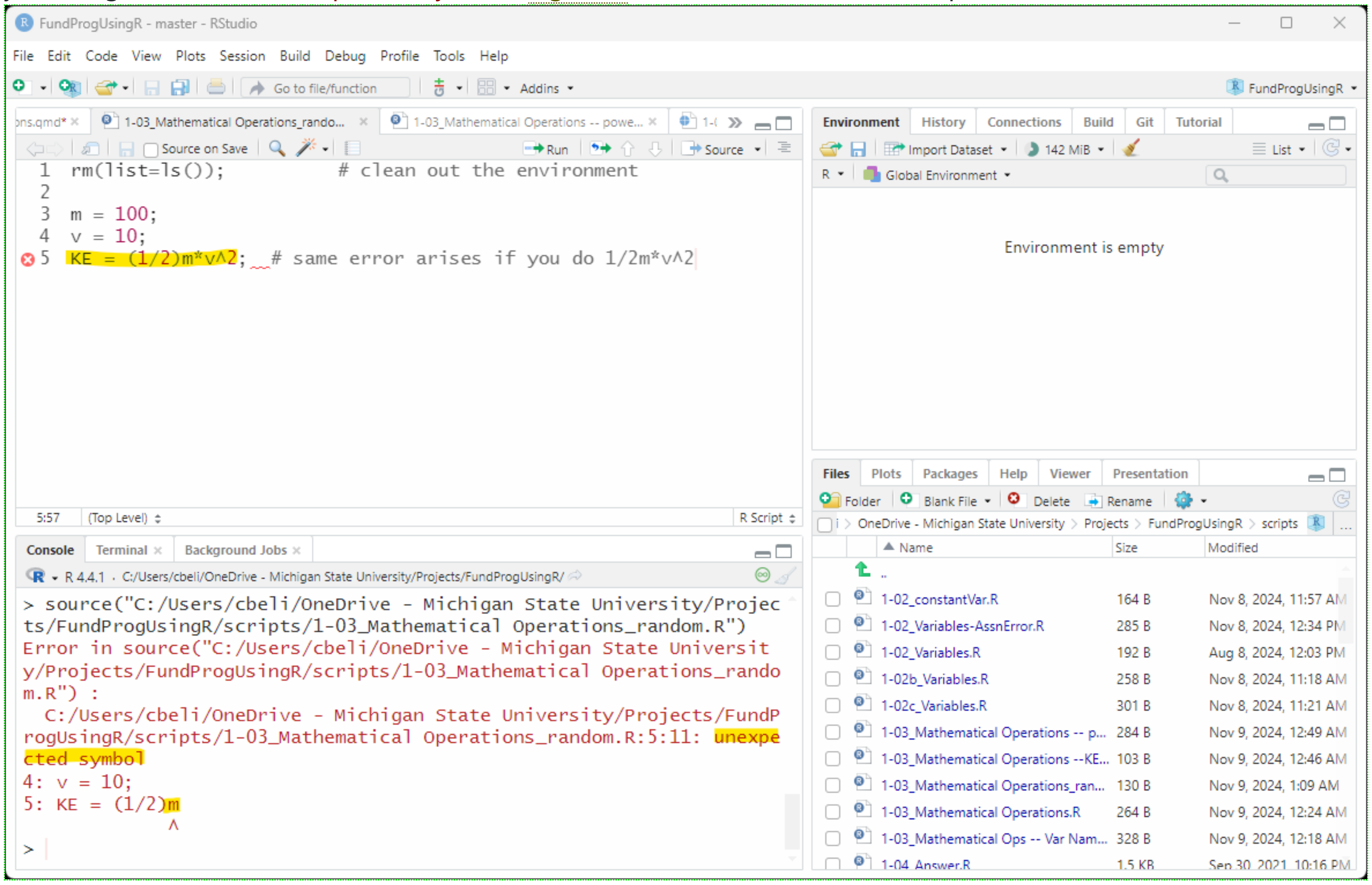
```
KE = 1/2*mv^2;
```

Then you will get the error: object 'mv' not found in the ***Console*** tab because R does not realize you want to multiply the variables ***m*** and ***v***, it thinks you are trying to use a variable named ***mv***, and the variable ***mv*** does not exist.

If you code ***KE*** like this::

```
KE = (1/2)m*v^2;  # same error arises if you do 1/2m*v^2
```

you will get the error: unexpected symbol (Figure 7) and the ***Console*** tab will point to the ***m***



```
rm(list=ls());          # clean out the environment

m = 100;
v = 10;
KE = (1/2)m*v^2; # same error arises if you do 1/2m*v^2
```

```
> source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R")
Error in source("C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R") :
  C:/Users/cbeli/OneDrive - Michigan State University/Projects/FundProgUsingR/scripts/1-03_Mathematical Operations_random.R:5:11: unexpected symbol
4: v = 10;
5: KE = (1/2)m
             ^
>
```

Figure 7: Unexpected symbol error

In this case, R is expecting an operation (symbol) after **(1/2)**. ***m*** is a variable, not a operation, hence the error.

11 Trap: Units, or lack thereof, in programming

One problem that crops up quite often in programming is that none of the numbers used in calculations have units. So we often have lines of code without any mention of units like this:

```
# find an average of the following three weights
weight1 = 175;
weight2 = 200;
weight3 = 210;

aveWeight = (weight1 + weight2 + weight3) / 3;
```

And if we add units to the number…

```
# find an average of the following three weights
weight1 = 175lb;  # causes "unexpected symbol"
          error
```

```
weight2 = 200lb;
weight3 = 210lb;

aveWeight = (weight1 + weight2 + weight3) / 3;
```

We get the error unexpected symbol because R is expecting some sort of operation after the number **175** and **lb** is not a valid operation.

Note: Lines 3 and 4 would also cause an unexpected symbol error but R ceases executing at the first error.

It is best to mention the units somewhere in the comments especially if your script is large or others are using your script.

```
# find an average of the following three weights (all in pounds)
weight1 = 175;
weight2 = 200;
weight3 = 210;

aveWeight = (weight1 + weight2 + weight3) / 3;
```

Otherwise, you risk a situation like the Mars Climate Orbiter crash, which could have easily been avoided with proper comments.

12 Extension: The show.error.location line

The second line of code in Figure 1 tells R to provide a line number where an error occurs:

```
options(show.error.locations = TRUE);
```

In Figure 1, in the ***Console*** tab the error includes the line number, 9, because of the line above. This seems like a great idea except that this line only works in a few limited situations. As you get into more complex code, this line will not work and R will not show you the line number for the error.

Unfortunately, R is one of the harder programming languages to debug. We will get to some debugging strategies in later lessons. For now, you can include the ***show.error.location*** line in your code. It does not hurt – it just has limited utility.