Week 2: *Functions & Packages*

EMSE 4574: Intro to Programming for Analytics

John Paul Helveston

September 08, 2020

Quiz 1

Go to `#links` channel in Slack for link

06:00

Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types
3. Packages
4. Polya’s Problem Solving Technique

Funtions take this form:

`name(argument)`

Example:

```r
sqrt(225)
```

```
## [1] 15
```

Not every function has an argument:

```r
date()
```

```
## [1] "Tue Sep  8 11:08:20 2020"
```

Some functions have multiple arguments:

```r
round(3.1415, 2)
```

```
## [1] 3.14
```

Arguments have names too:

```r
round(x = 3.1415, digits = 2)
```

```
## [1] 3.14
```

If you don't include all arguments, default values will be used:

```r
round(x = 3.1415)
```

```
## [1] 3
```

For arguments, use "=" , not "<-"

=

Arguments are "local" to the function

```
round(x = 3.1415, digits = 2)
```

```
## [1] 3.14
```

```
x
```

```
Error: object 'x' not found
```

<-

Arguments also get created "globally"

```
round(x <- 3.1415, digits <- 2)
```

```
## [1] 3.14
```

```
x
```

```
## [1] 3.1415
```

```
digits
```

```
## [1] 2
```

Use ? to get help

```
?round()
```

```
Rounding of Numbers
Description
ceiling takes a single numeric argument x and returns a numeric vector containing the smallest
integers not less than the corresponding elements of x.

floor takes a single numeric argument x and returns a numeric vector containing the largest
integers not greater than the corresponding elements of x.

trunc takes a single numeric argument x and returns a numeric vector containing the integers
formed by truncating the values in x toward 0.

round rounds the values in its first argument to the specified number of decimal places (default
0). See ‘Details’ about “round to even” when rounding off a 5.

signif rounds the values in its first argument to the specified number of significant digits.

Usage
ceiling(x)
floor(x)
trunc(x, ...)

round(x, digits = 0)
```

Combining functions

You can use functions as arguments to other functions:

```r
round(sqrt(7), digits = 2)
```

```
## [1] 2.65
```

This is the same as doing this:

```r
temp <- sqrt(7)
round(temp, digits = 2)
```

```
## [1] 2.65
```

Combining functions

Ex: What do you think this would return:

```r
sqrt(1 + abs(-8))
```

```
## [1] 3
```

Frequently used **math** functions

Function	Description	Example input	Example output
`sqrt()`	Square root	`sqrt(64)`	8
`round(x, digits=0)`	Round `x` to the `digits` decimal place	`round(3.1415, digits=2)`	3.14
`floor(x)`	Round `x` **down** the nearest integer	`floor(3.1415)`	3
`ceiling(x)`	Round `x` **up** the nearest integer	`ceiling(3.1415)`	4
`abs()`	Absolute value	`abs(-42)`	42
`min()`	Minimum value	`min(1, 2, 3)`	1
`max()`	Maximum value	`max(1, 2, 3)`	3

Think-Pair-Share

08:00

Consider the following code (don't run it):

```r
val <- sqrt(y = abs(-10))
val <- abs(x <- log(10))
result <- round(x, digits <- sqrt(abs(-4)))
result*digits
```

NOTE: This is just for practice - you should always use "=" for function arguments.

Now follow these steps:

1. Type out what you expect R will return when all the lines are run at once.
2. Compare your expectations with each other.
3. Run the code and compare the results with your expectations.

Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types
3. Packages
4. Polya’s Problem Solving Technique

Use these patterns:

Convert type of `x`:

`as.______(x)`

Check type of `x`:

`is.______(x)`

Replace "______" with:

- `character`
- `logical`
- `numeric` / `double` / `integer`

Convert type with `as.______(x)`

Convert **numeric** types:

```
as.numeric("3.1415")
```

```
## [1] 3.1415
```

```
as.double("3.1415")
```

```
## [1] 3.1415
```

```
as.integer("3.1415")
```

```
## [1] 3
```

Convert **non-numeric** types:

```
as.character(3.1415)
```

```
## [1] "3.1415"
```

```
as.logical(3.1415)
```

```
## [1] TRUE
```

A few notes on converting types

Converting any number to a logical returns `TRUE` except for `0`

```r
as.logical(7)
```

```
## [1] TRUE
```

```r
as.logical(0)
```

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

`TRUE = 1`, `FALSE = 0`:

```r
as.numeric(TRUE)
```

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

```r
as.numeric(FALSE)
```

```
## [1] 0
```

A few notes on converting types

Not everything can be converted.

```r
as.numeric('7')   # Works
```

```
## [1] 7
```

```r
as.numeric('foo') # Doesn't work
```

```
## [1] NA
```

A few notes on converting types

`as.integer()` is the same as `floor()`:

```r
as.integer(3.14)
```

```
## [1] 3
```

```r
as.integer(3.99)
```

```
## [1] 3
```

Check type with `is._______(x)`

Checking **numeric** types:

```r
is.numeric(3.1415)
```

```
## [1] TRUE
```

```r
is.double(3.1415)
```

```
## [1] TRUE
```

```r
is.integer(3.1415)
```

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

Checking **non-numeric** types:

```r
is.character(3.1415)
```

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

```r
is.logical(3.1415)
```

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

Integers are weird

```r
is.integer(7)
```

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

...because R thinks `7` is really `7.0`

To check if a number is an integer *in value*:

```r
7 == as.integer(7)
```

```
## [1] TRUE
```

Think-Pair-Share

08:00

Consider the following code (don't run it):

```
number    <- as.logical(as.numeric('3'))
character <- is.character(typeof(7))
true      <- as.logical("FALSE")
false     <- as.logical(as.numeric(TRUE))

! (number == character) & (true | false) | (number & false)
```

Now follow these steps:

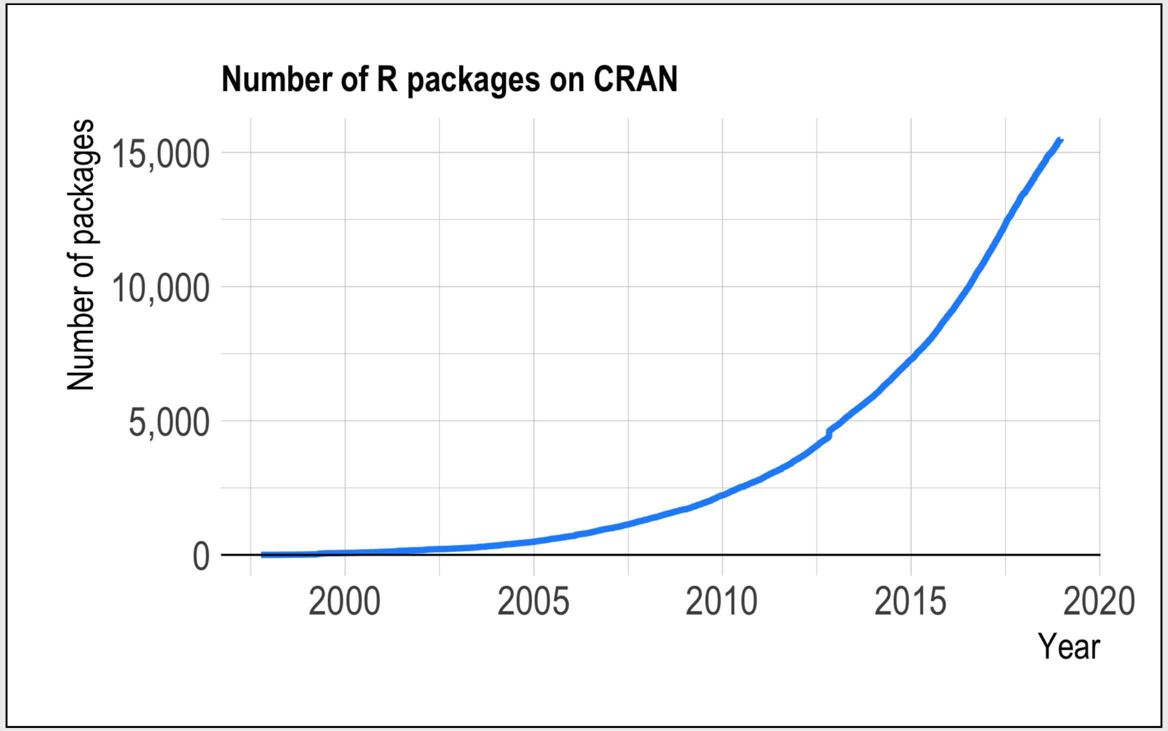
1. Type out what you expect R will return when all the lines are run at once.
2. Compare your expectations with each other.
3. Run the code and compare the results with your expectations.

05:00

Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types
3. Packages
4. Polya's Problem Solving Technique

>15,000 packages on the CRAN



CRAN = The Comprehensive R Archive Network

List of packages

Installing packages

`install.packages("packagename")`

(The package name **must** be in quotes)

```
install.packages("packagename") # This works
install.packages(packagename)   # This doesn't work
```

You only need to install a package once!

Loading packages

`library(packagename)`: Loads all the functions in a package

(The package name *doesn't* need to be in quotes)

```
library("packagename") # This works
library(packagename)   # This also works
```

You need to *load* the package every time you use it!

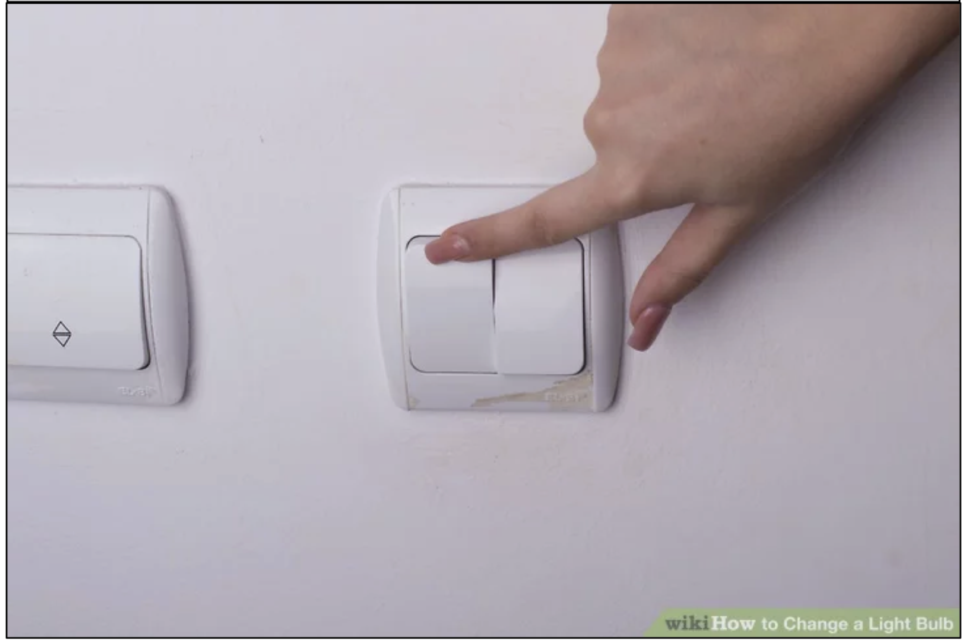
Installing vs. Loading

INSTALL ONCE:

```
install.packages("light")
```

USE MANY TIMES:

```
library("light")
```



Example: **wikifacts**

Install the Wikifacts package, by Keith McNulty:

```
install.packages("wikifacts")
```

Load the package:

```
library(wikifacts) # Load the library
```

Use one of the package functions

```
wiki_randomfact()
```

```
## [1] "Here's some news from 01 August 2015. In cycling, Chris Froome wins the Tour de France for
a second time. (Courtesy of Wikipedia)"
```

Example: **wikifacts**

Now, restart your RStudio session:

> Session -> Restart R

Try using the package function again:

```
wiki_randomfact()
```

```
## Error in wiki_randomfact(): could not find function "wiki_randomfact"
```

Using only *some* package functions

You don't always have to load the whole library.

Functions can be accessed with this pattern:

`packagename::functionname()`

```
wikifacts::wiki_randomfact()
```

```
## [1] "Did you know that on October 17 in 1940 – The body of Willi Münzenberg, a French communist
who was the leading propagandist for the Communist Party of Germany, was found near Saint-
Marcellin. (Courtesy of Wikipedia)"
```

Learn more about a package:

`help(package = 'packagename')`

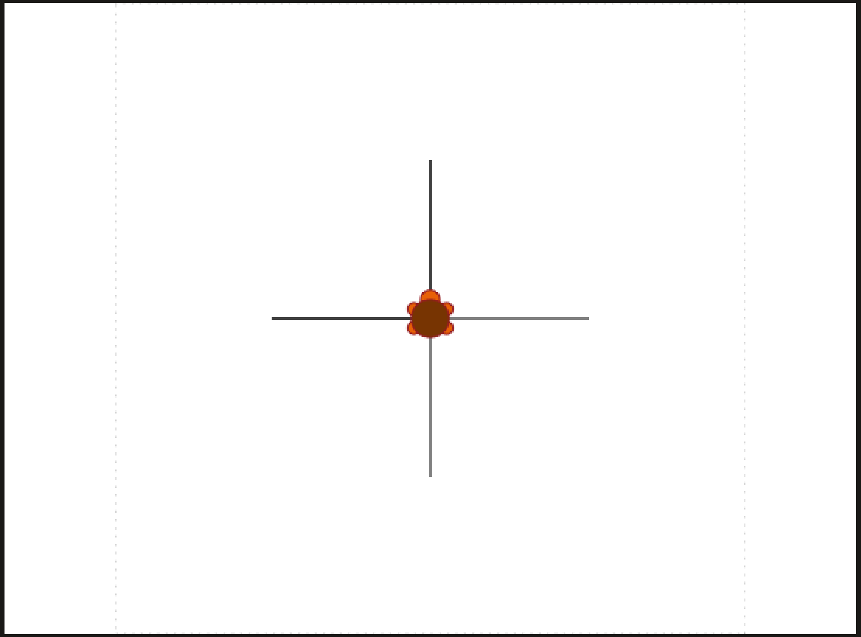
Example:

```r
help(package = 'wikifacts')
```

Think-Pair-Share

10:00

1. Install the `TurtleGraphics` package.
2. Load the package.
3. Use the `turtle_init()` function to create a turtle.
4. Use `help(package = 'TurtleGraphics')` to learn about other functions to control your turtle.
5. Try drawing this shape with your turtle (hint: the length of each line is 50 units).
6. Compare your results and code with each other.



Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types
3. Packages
4. Polya’s Problem Solving Technique

Polya's Problem Solving Technique

Step 1: Understand the problem

Step 2: Devise a plan

Step 3: Carry out the plan

Step 4: Check your work

Polya's Problem Solving Technique

Step 1: Understand the problem

Step 2: Devise a plan

Step 3: Carry out the plan

Step 4: Check your work

- Seems obvious (easy to overlook)
- Restate the problem in your own words
- Draw a figure
- What information do you *have*?
- What information do you *need*?

Polya's Problem Solving Technique

Step 1: Understand the problem

Step 2: Devise a plan

Step 3: Carry out the plan

Step 4: Check your work

- Do you know a related problem?
- Look at the unknown!
- Guess and check
- Eliminate possibilities
- Consider special cases
- Work backwards

Polya's Problem Solving Technique

Step 1: Understand the problem

Step 2: Devise a plan

Step 3: Carry out the plan

Step 4: Check your work

- (this is where you write code)
- **Be patient**
- Stick to the plan…until the plan fails
- Then change your plan
- Error message != plan has failed

Polya's Problem Solving Technique

Step 1: Understand the problem

Step 2: Devise a plan

Step 3: Carry out the plan

Step 4: Check your work

- Seems obvious (easy to overlook)
- Check intermediate values
- *Examine* the solution obtained
- Can you derive the solution differently?

Polya practice: Adding a number sequence

How might you add up the numbers in a sequence from 1 to 10?

Step 1: Understand the problem

Step 2: Devise a plan

Step 3: Carry out the plan

Step 4: Check your work

Polya practice: What's your degree worth?

In the U.S., the average salary of a high school graduate is $35,256 / year, and the average salary of a GW graduate is $76,151. However, GW grads pay an average of $70,000 / year (tuition + fees + housing) for 4 years for their degree, and high school grads are working that entire time. Assuming immediate employment after graduation, how many years after graduating will the GW grad need to work until their net income (salary minus cost of education) surpasses that of the average high school graduate? (This is a *very* rough estimate - you can assume away interest rates, inflation, promotions / salary raises, etc.)

Polya practice: What's your degree worth?

In the U.S., the average salary of a high school graduate is $35,256 / year, and the average salary of a GW graduate is $76,151. However, GW grads pay an average of $70,000 / year (tuition + fees + housing) for 4 years for their degree, and high school grads are working that entire time. Assuming immediate employment after graduation, how many years after graduating will the GW grad need to work until their net income (salary minus cost of education) surpasses that of the average high school graduate? (This is a *very* rough estimate - you can assume away interest rates, inflation, promotions / salary raises, etc.)

Take 5 minutes to restate the problem & devise a plan (don't carry it out!)

05:00

Think-Pair-Share

Kevin is deciding between purchasing a Toyota Prius, which sells for $27,600, and a Toyota Camry, which sells for $24,000. He knows that based on his driving patterns he can get an average fuel economy of 55 miles per gallon (mpg) of gasoline in the Prius but only 28 mpg in the Camry on average. He also knows that he typically drives 12,000 miles each year, and the average price of gasoline is $2.20 / gallon.

How long (in years) would Kevin have to drive the Prius for the money he saves in fuel savings to be greater than the price premium compared to the Camry?

03:00