Week 2: *Functions & Packages*

EMSE 4571 / 6571: Intro to Programming for Analytics

John Paul Helveston

January 23, 2025

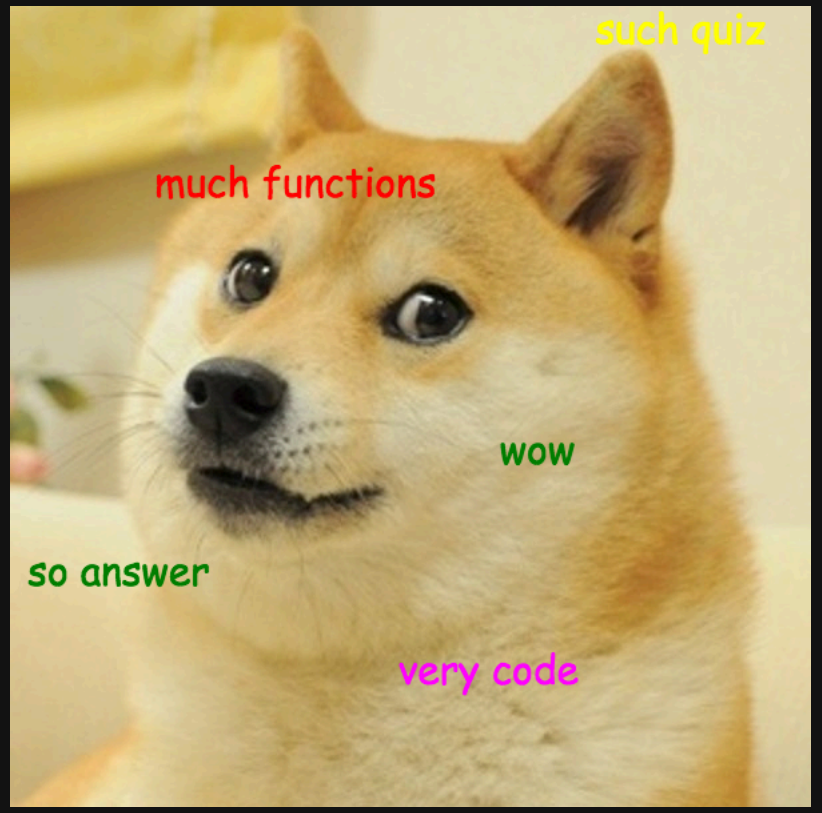
Quiz 1

10:00

Write your name on the quiz!

Rules:

- Work alone; no outside help of any kind is allowed.
- No calculators, no notes, no books, no computers, no phones.



Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types

BREAK

3. External packages 📦
4. Polya's problem solving technique

Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types

BREAK

3. External packages 📦
4. Polya's problem solving technique

Funtions take this form:

`name(argument)`

```
sqrt(225)
```

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

Not every function has an argument:

```
date()
```

```
#> [1] "Thu Jan 23 08:44:37 2025"
```

Some functions have multiple arguments:

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

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

Arguments have names too:

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

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

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

```
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

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

round(x, digits = 0)
signif(x, digits = 6)
Arguments
x          a numeric vector. Or, for round and signif, a complex vector.
digits    integer indicating the number of decimal places (round) or significant digits
(signif) to be used. Negative values are allowed (see 'Details').
```

Combining functions

You can use functions as arguments to other functions:

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

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

What do you think this will return:

```
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(3.9)`	3
`ceiling(x)`	Round `x` **up** the nearest integer	`ceiling(3.1)`	4
`abs()`	Absolute value	`abs(-42)`	42
`min()`	Minimum value	`min(1, 2, 3)`	1
`max()`	Maximum value	`max(1, 2, 3)`	3

Your turn

05:00

Consider the following code blocks:

Block 1:

```
val <- abs(x <- sqrt(10))
result <- round(val, digits <- sqrt(10))
answer <- x*digits
answer
```

Block 2:

```
val <- sqrt(abs(min(-42, -64, 81)))
result <- floor(y = min(val, log(10)))
answer <- result*val
answer
```

Now follow these steps:

1. Don't run the code (yet)!
2. Write down out what you expect R will return when these lines are run in sequence.
3. Compare your expectations with each other.
4. Run the code and compare the results with your expectations.

Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types

BREAK

3. External 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`

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

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

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

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

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

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

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

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

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

A few notes on converting types

Not everything can be converted.

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

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

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

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

A few notes on converting types

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

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

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

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

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

Check type with `is._____(x)`

Checking **numeric** types:

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

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

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

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

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

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

Checking **non-numeric** types:

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

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

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

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

Integers are weird

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

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

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

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

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

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

Your turn

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. Don't run the code (yet)!
2. Write down out what you expect R will return when these lines are run in sequence.
3. Compare your expectations with each other.
4. Run the code and compare the results with your expectations.

Intermission

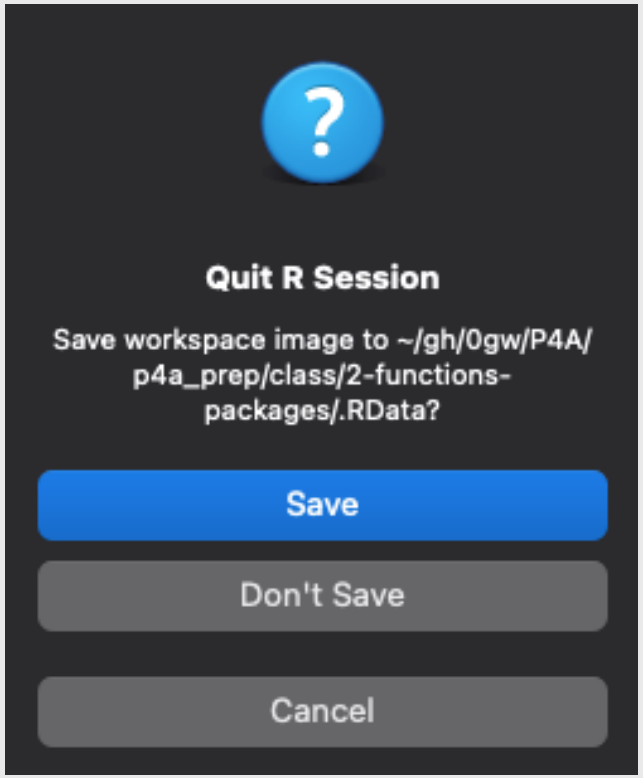
Stand up, stretch, move around!

05:00

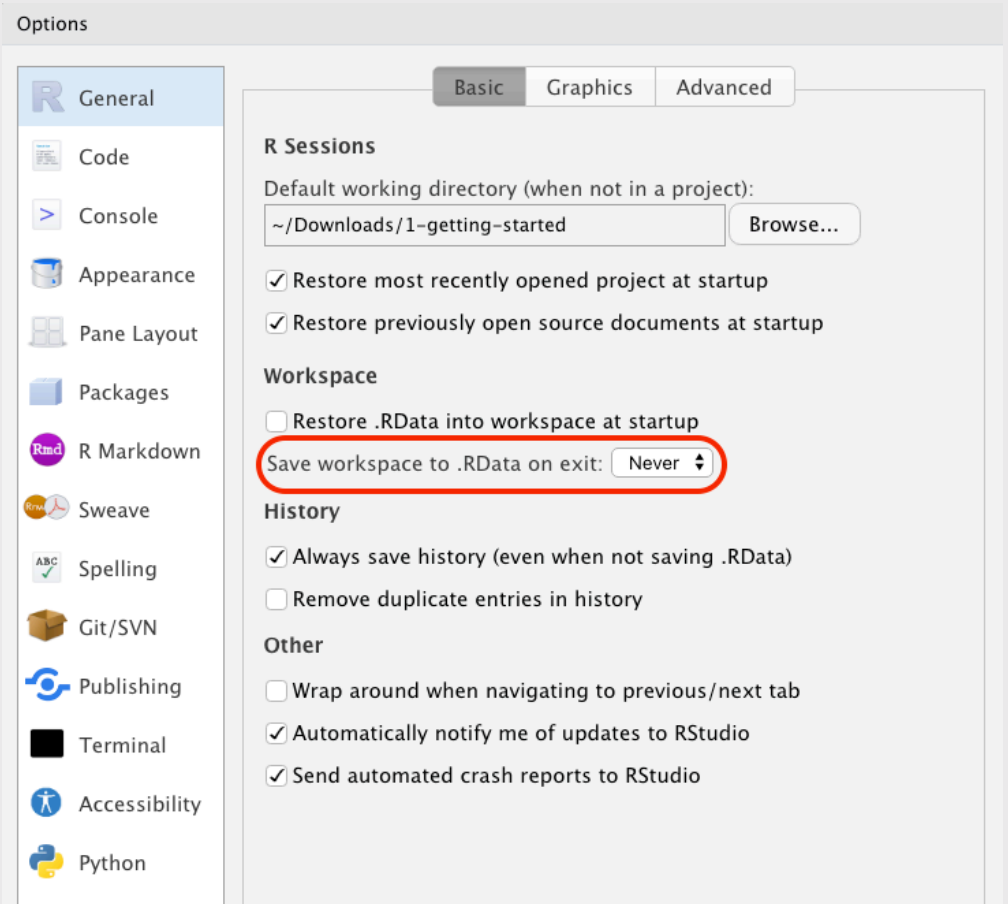
Tip of the week

What's with that .RData file?

Don't save the `.RData` file on exit



Tools → Global Options



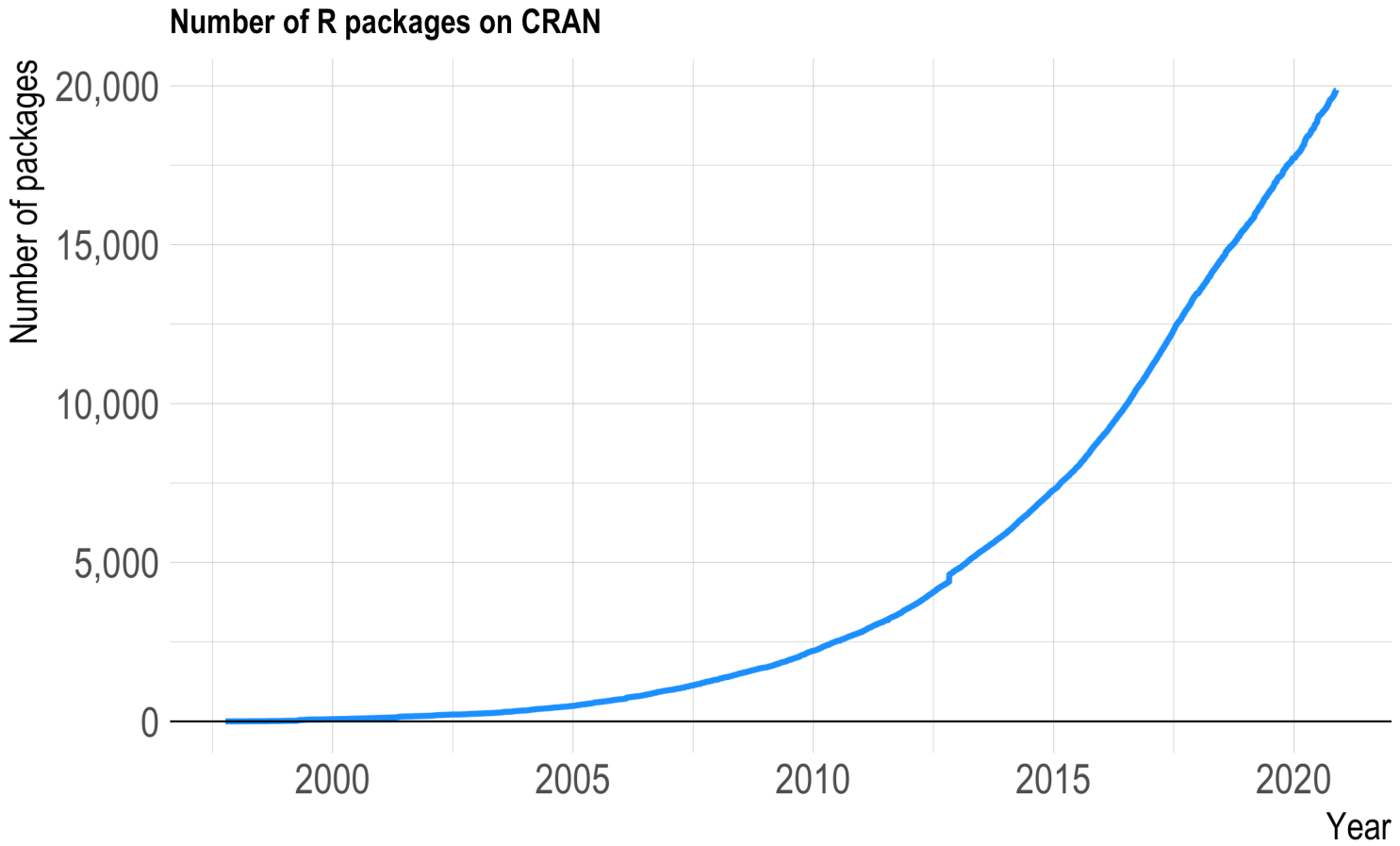
Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types

BREAK

3. External packages 📦
4. Polya's problem solving technique

>20,000 packages on the CRAN



Installing: `install.packages("packagename")`

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: `library(packagename)`

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] "Did you know that the party leader of the new Dutch youth political party
LEF&amp;nbsp;– For the New Generation tattooed his party's program points on his forearm?
(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

Functions can be accessed with this pattern:

```
packagename::functionname()
```

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

```
#> [1] "Did you know that on February 12 in 1502 – Isabella I (portrait shown) issued an
edict outlawing Islam in the Crown of Castile, forcing virtually all her Muslim subjects
to convert to Christianity. (Courtesy of Wikipedia)"
```

Learn more about a package:

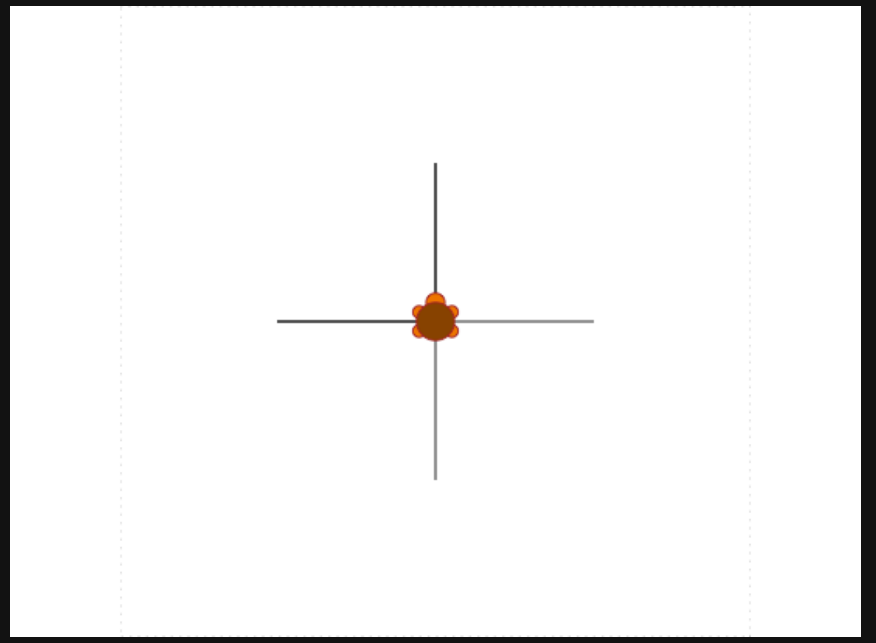
```
help(package = 'packagename')
```

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

Your turn

10:00

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



Week 2: *Functions & Packages*

1. Functions
2. Manipulating data types

BREAK

3. External 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
- Can you derive the solution differently?

Polya practice: What's your degree worth?

1. Understand the problem
2. Devise a plan
3. Carry out the plan
4. Check your work

In the U.S., the average annual salary of a high school graduate is $35,256, 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?

(NOTE: This is a *very* rough estimate - we're assuming away interest rates, inflation, promotions, etc.)

10:00

Polya practice: Should you buy a Hybrid car?

1. Understand the problem
2. Devise a plan
3. Carry out the plan
4. Check your work

Kevin is deciding between purchasing a Toyota Prius, which sells for $27,600, and a Toyota Camry, which sells for $24,000. He knows he can get an average fuel economy of 55 miles per gallon (mpg) 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 $3.00 / 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?

10:00

Preview of HW2