W1: Intro to Computing and Class Logistics

Welcome!

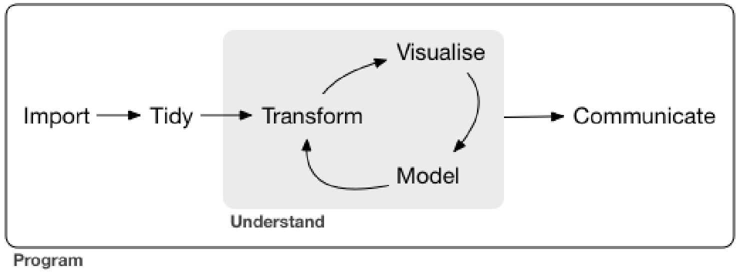
Introductions

- Who am I?
- What is DaSL? And TAs!
- Who are you? (Online people, in Chat)
  - Name, pronouns, group you work in
  - What you want to get out of the class
  - Favorite autumn activity

Goals of the course

- Fundamental concepts in programming languages: *How do programs run, and how do we solve problems effectively using functions and data structures?*

- Data science fundamentals: *How do you translate your scientific question to a data wrangling problem and answer it?*



Data science workflow

Culture of the course

- Challenge: We sometimes struggle with our data science in isolation, unaware that someone two doors down from us has gone through the same struggle.

- *We learn and work better with our peers.*

- *Know that if you have a question, other people will have it.*

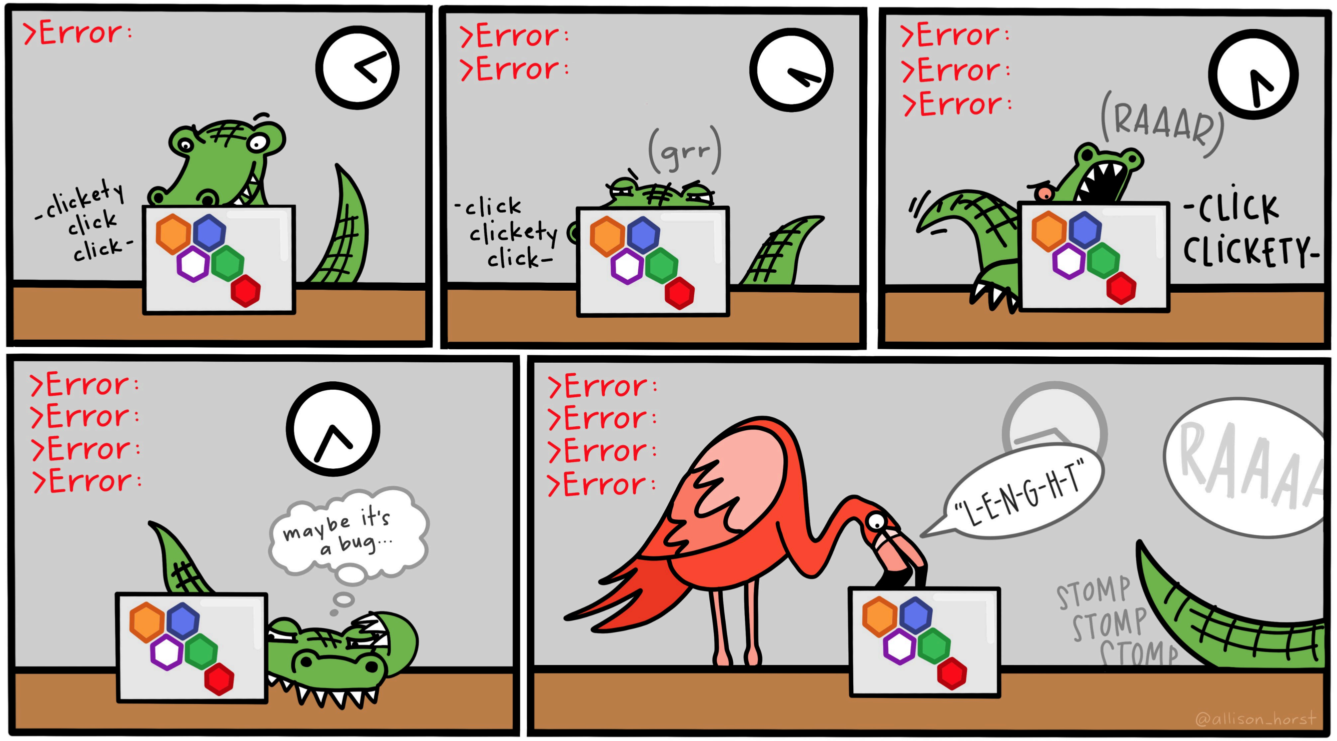
- *Asking questions is our way of taking care of others.*

We ask you to follow Participation Guidelines and Code of Conduct.

Words of Encouragement

- Be gentle with yourself
- Learning data science and programming can be inherently difficult
- It’s not you, it’s the subject

Breaks Are Super Helpful



Format of the course

- Hybrid, and recordings will be available.
- 1-2 hour exercises after each session are encouraged for practice.
- Office Hours Fridays 10am - 11am PT.
- Online discussion via Slack / Teams (use office hours teams meeting).

Quick Look at the Website

https://hutchdatascience.org/Intro_to_R

Content of the course (by week, roughly)

1. *Intro to Computing
2. Data structures
3. *Data visualization
4. Learning Community Session (optional)
5. *Data wrangling 1
6. Data wrangling 2
7. *Learning Community Session (optional)
8. Wrap up / Loading your own data / Code-a-thon prep

In person: I will be on-campus on the * dates. Other dates, you are free to attend in the DaSL lounge.

Full info is here: https://hutchdatascience.org/Intro_to_R/index.html#class-schedule

Ask Me Two Questions

What is a computer program?

- A sequence of instructions to manipulate data for the computer to execute.

- A series of translations: English <-> Programming Code for Interpreter <-> Machine Code for Central Processing Unit (CPU)

We will focus on English <-> Programming Code for R Interpreter in this class.

Another way of putting it: **How we organize ideas <-> Instructing a computer to do something**.

Posit Cloud tour and trying out your first analysis!

Break

A pre-course survey:

https://forms.gle/Hr59ZbAan1JTumCa7

Grammar Structure 1: Evaluation of Expressions

- **Expressions** are built out of **operations** or **functions**.

- Operations and functions take in **data types** and return another data type.

- We can combine multiple expressions together to form more complex expressions: an expression can have other expressions nested inside it.

Examples

```r
18 + 21
```

```
[1] 39
```

```r
max(18, 21)
```

```
[1] 21
```

```r
max(18 + 21, 65)
```

```
[1] 65
```

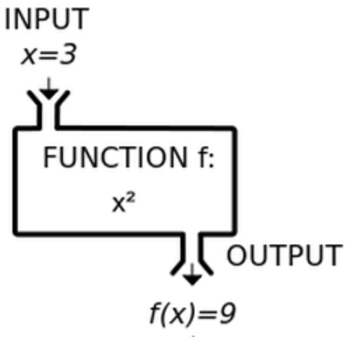
```r
18 + (21 + 65)
```

```
[1] 104
```

```r
nchar("ATCG")
```

```
[1] 4
```

Function machine from algebra class



Operations are functions. We could have written:

```r
sum(18, 21)
```

```
[1] 39
```

```r
sum(18, sum(21, 65))
```

```
[1] 104
```

Data types

- **Numeric**: 18, -21, 65, 1.25
- **Character**: “ATCG”, “Whatever”, “948-293-0000”
- **Logical**: TRUE, FALSE

Grammar Structure 2: Storing data types in the environment

To build up a computer program, we need to store our returned data type from our expression somewhere for downstream use.

```r
x = 18 + 21
```

Execution rule for variable assignment

Evaluate the expression to the right of `=`.

Bind variable to the left of `=` to the resulting value.

The variable is stored in the environment.

`<-` is okay too!

Remember: Case (capitalization) Matters!

x is not equal to X

(If you’re coming from SAS)

Downstream

Look, now `x` can be reused downstream:

```r
x - 2
```

```
[1] 37
```

```r
y = x * 2
y
```

```
[1] 78
```

Grammar Structure 3: Evaluation of Functions

A function has a **function name**, **arguments**, and **returns** a data type.

Execution rule for functions:

Evaluate the function by its arguments, and if the arguments contain expressions, evaluate those expressions first.

The output of functions is called the **returned value**.

```r
sqrt(nchar("hello"))
```

```
[1] 2.236068
```

```r
(nchar("hello") + 4) * 2
```

```
[1] 18
```

A programming language has following features:

- Grammar structure to construct expressions
- Combining expressions to create more complex expressions
- Encapsulate complex expressions via functions to create modular and reusable tasks
- Encapsulate complex data via data structures to allow efficient manipulation of data

Tips on writing your first code

`Computer = powerful + stupid`

Even the smallest spelling and formatting changes will cause unexpected output and errors!

- Write incrementally, test often
- Check your assumptions, especially using new functions, operations, and new data types.
- Live environments are great for testing, but not great for reproducibility.
- **Ask for help!**

Taking the Temperature

Please take our weekly check in survey: https://forms.gle/vGMcWC5hZN3AcYYW8

- Clearest Point
- Muddiest Points: example here
- Anything else you want to share

We will have a brief discussion before we start each class and respones will also be posted on the course website.

In Summary

- We will use Posit Cloud for the course
- Notebooks help you keep track of your work
- Computers evaluate expressions, store variables, and use functions to do things
- All materials will be posted here: https://hutchdatascience.org/Intro_to_R
- Friday 10 - 11 am PT to practice together!