PSTAT 10

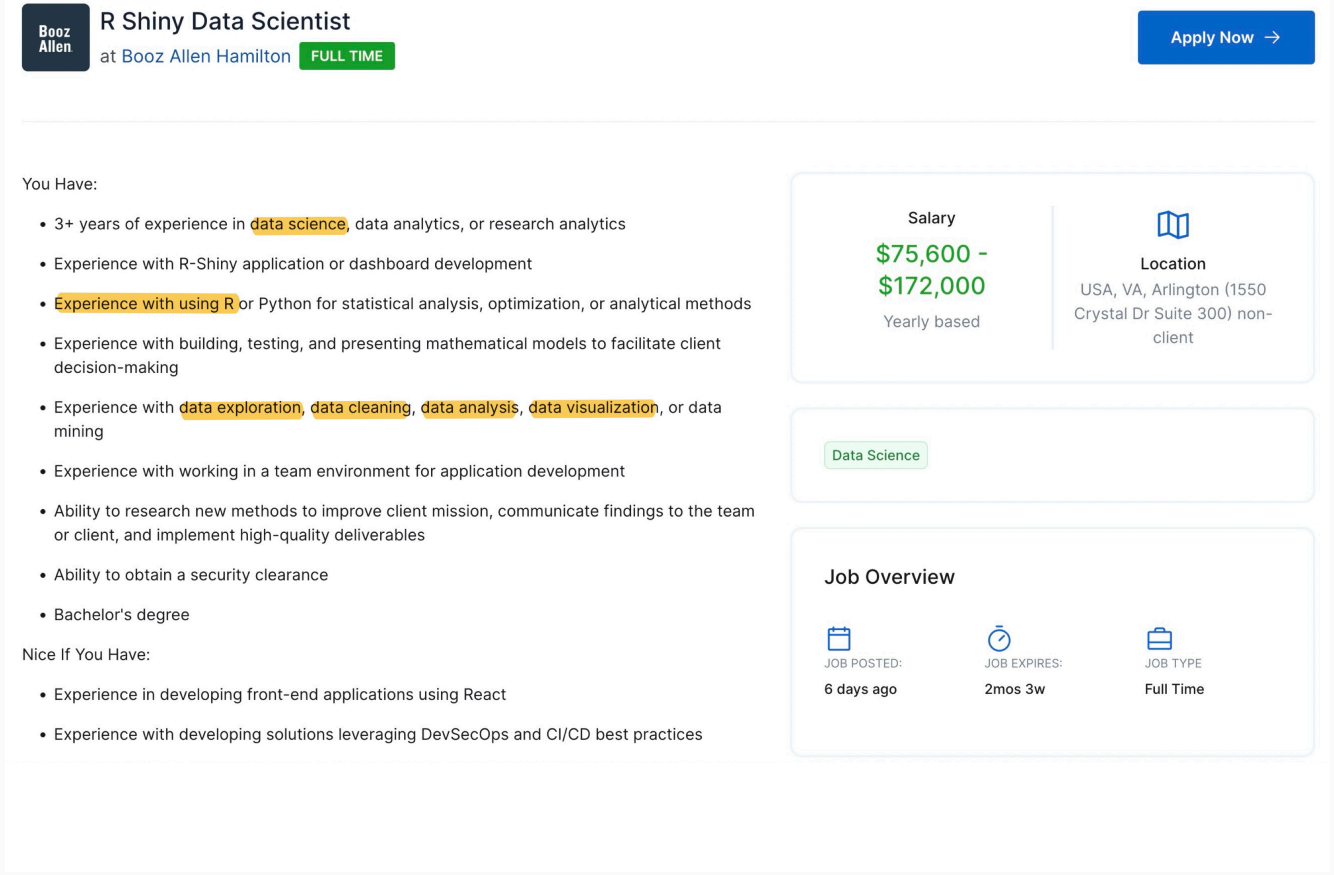
Principles of Data Science

Lecture 1: Course logistics, the R ecosystem, Vectors

Ingmar Sturm
UCSB
2024-06-24

Special thanks to Robin Liu for select course content used with permission.

A Job Ad



Booz Allen

R Shiny Data Scientist

at Booz Allen Hamilton FULL TIME

Apply Now →

You Have:

- 3+ years of experience in data science, data analytics, or research analytics
- Experience with R-Shiny application or dashboard development
- Experience with using R or Python for statistical analysis, optimization, or analytical methods
- Experience with building, testing, and presenting mathematical models to facilitate client decision-making
- Experience with data exploration, data cleaning, data analysis, data visualization, or data mining
- Experience with working in a team environment for application development
- Ability to research new methods to improve client mission, communicate findings to the team or client, and implement high-quality deliverables
- Ability to obtain a security clearance
- Bachelor's degree

Nice If You Have:

- Experience in developing front-end applications using React
- Experience with developing solutions leveraging DevSecOps and CI/CD best practices

Salary

$75,600 - $172,000

Yearly based

Location

USA, VA, Arlington (1550 Crystal Dr Suite 300) non-client

Data Science

Job Overview

JOB POSTED: 6 days ago

JOB EXPIRES: 2mos 3w

JOB TYPE Full Time

About this class

Week 1

- R ecosystem, vectors, control flow
- recycling, filtering, vectorization

Week 2

- Data frames and tibbles
- `dplyr` and the pipe `|>`

Week 3

- Probability through simulation
- Simulating random experiments

Week 4

- Structure of a database system
- SQL select/where/join

Week 5

- Advanced topics
- `ggplot`, the S3 class, SQL window functions
- ...

Week 6

- ... (TBD)
- Final exam (last day of class)

Why come to class?

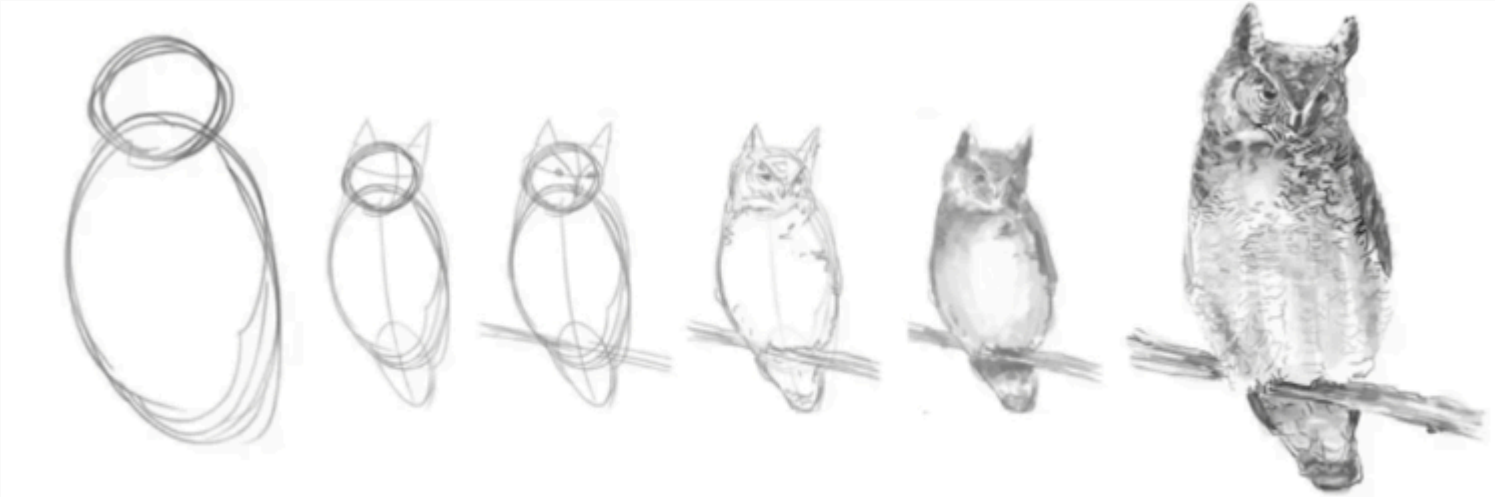
How to draw an owl

1. Draw some circles

2. Draw the rest of the owl

Why come to class?

How to draw an owl



1. Draw some circles

2. Draw the rest of the owl

Grading

- Lab worksheets: 25%
- Homework: 25%
- Midterm exam: 20%
- Final exam: 30%
- Discussion forum: (positive) tie-breaker
- Attendance and participation: (negative) tie-breaker

Lab worksheets are graded pass/fail based on completion.

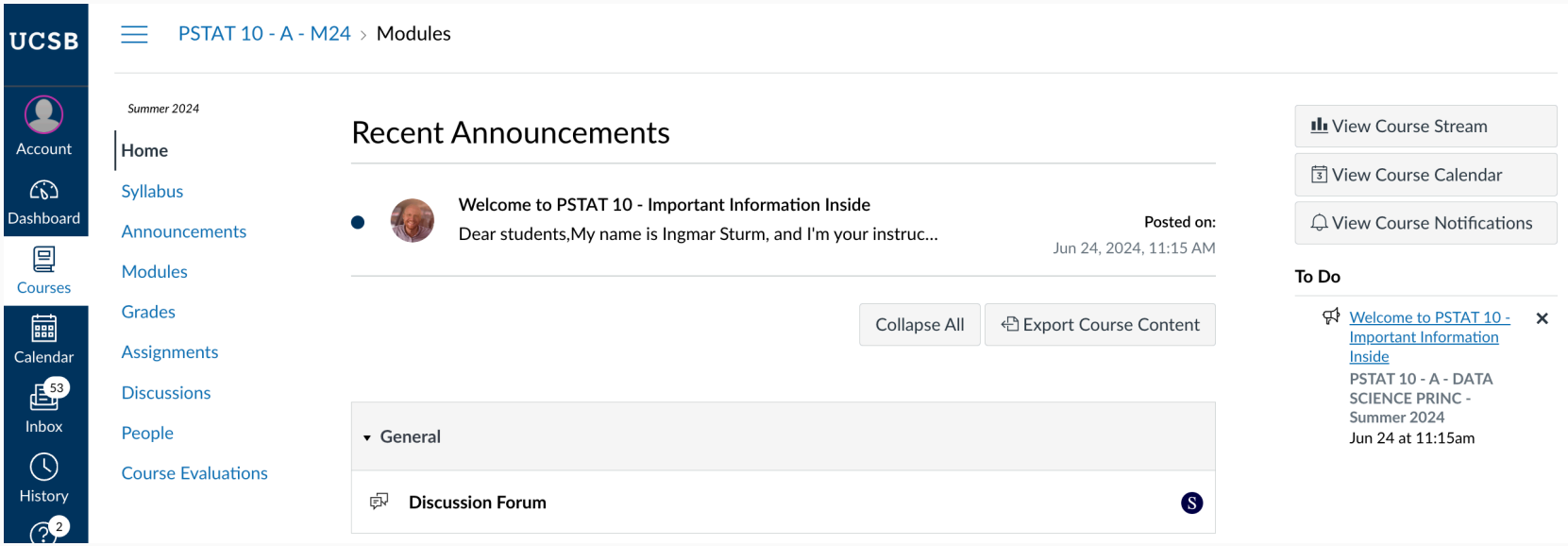
Don't fall behind!!

I have designed this class so that contents will build on top of each other. This means the lectures may refer to material in the lab worksheets or homework. Skipping an assignment will mean missing a large portion of the material.

If you need help keeping up, please use office hours.

Discussion Forum, Office Hours, and Email Policy

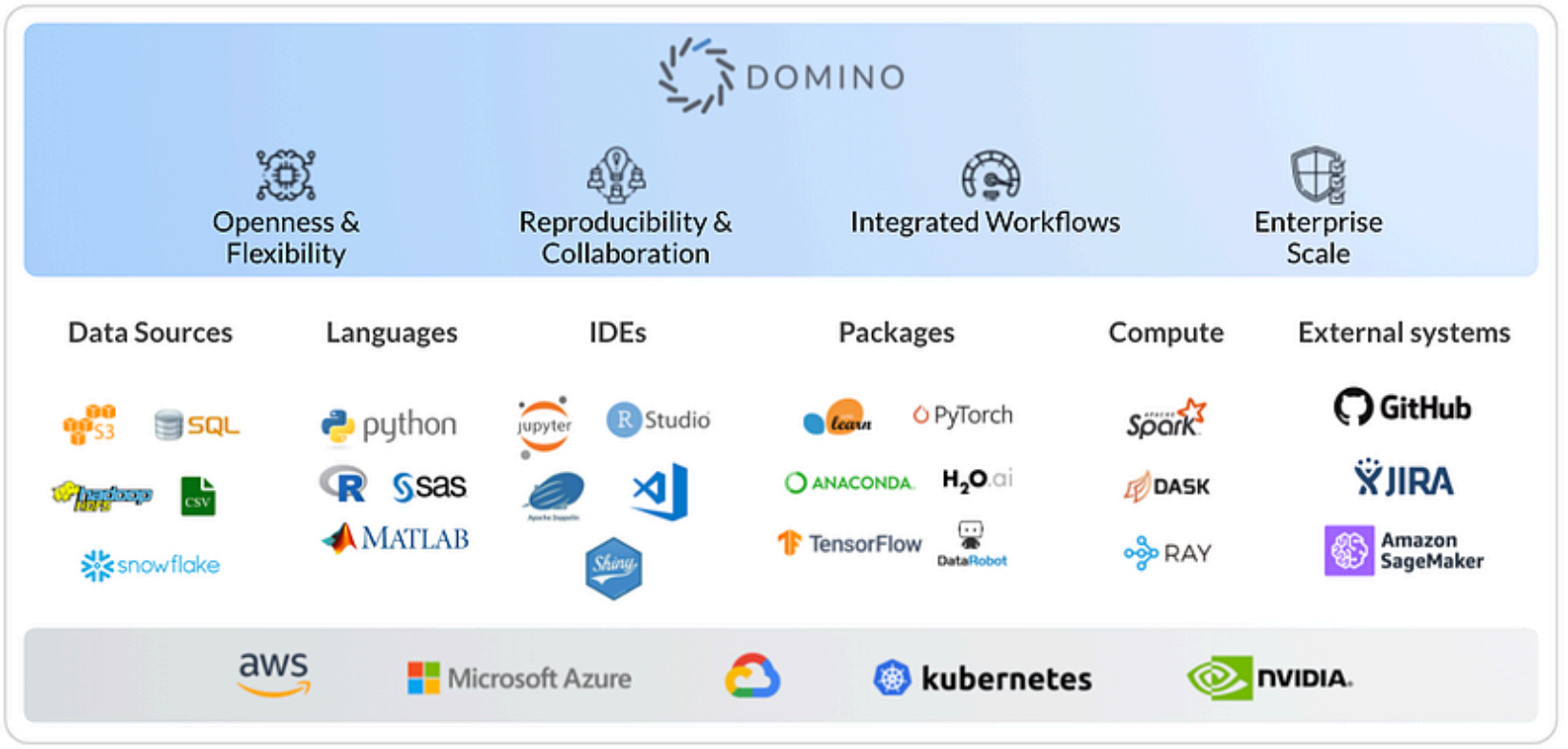
- It is normal to have questions and we are happy to answer them
- Email overload is a thing
- So please try to ask your questions either during or after class or during our office hours
- If you have questions about course rules or content that might be of interest to others, please ask them in the discussion forum
- Reserve email for all other questions



About me

- PhD Student in the Political Science department since 2017
- MA Statistics (Data Science) in 2021
- Coding in R since 2013
- Developed R package `fbsamplR`
- Fun fact: My dog's name is Barbara

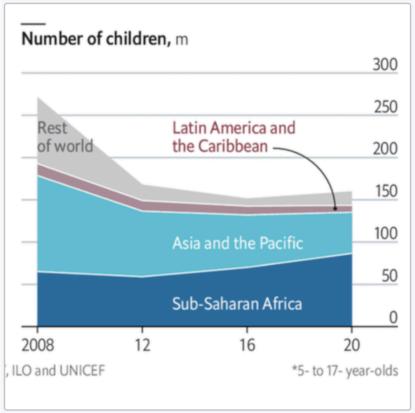
Programming for data science

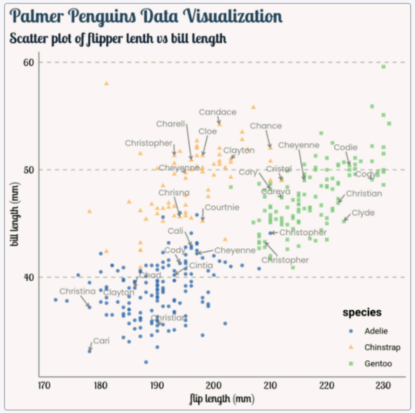


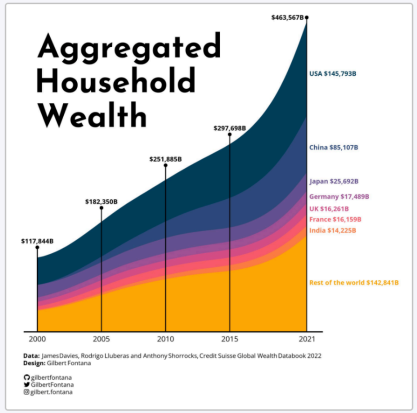
Why R?

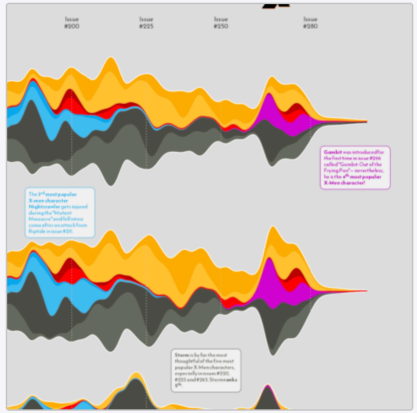
- Designed for statistical research
- R is open source
- Large ecosystem (a lot of people use it)
- Easy to simulate random experiments
- Intuitive data exploration, manipulation, and plotting

The R Graph Gallery









The R Ecosystem

What does it mean to be **open source**?

Not only is R free to use, you can **download its source code**

- https://mirror.las.iastate.edu/CRAN/sources.html

A community of researchers continually add functionality in the form of **packages**
These packages are also open source

- https://mirror.las.iastate.edu/CRAN/

Many of our faculty at PSTAT have developed their own packages as part of their research

Exercise: find out what packages and by whom.

The difficulty of teaching an intro class

R contains **a lot** of functionality. I am still constantly finding out new things about R.

It was hard to pick what to include and exclude from this class.

I encourage you to look at functionality beyond the course material.

Generally speaking, if your code produces correct output, it is correct. Unless the question asks for a particular method.

Use the help

The `?` operator searches the documentation for the given function. This is displayed in the RStudio help window.

What does the `seq_len` function do?

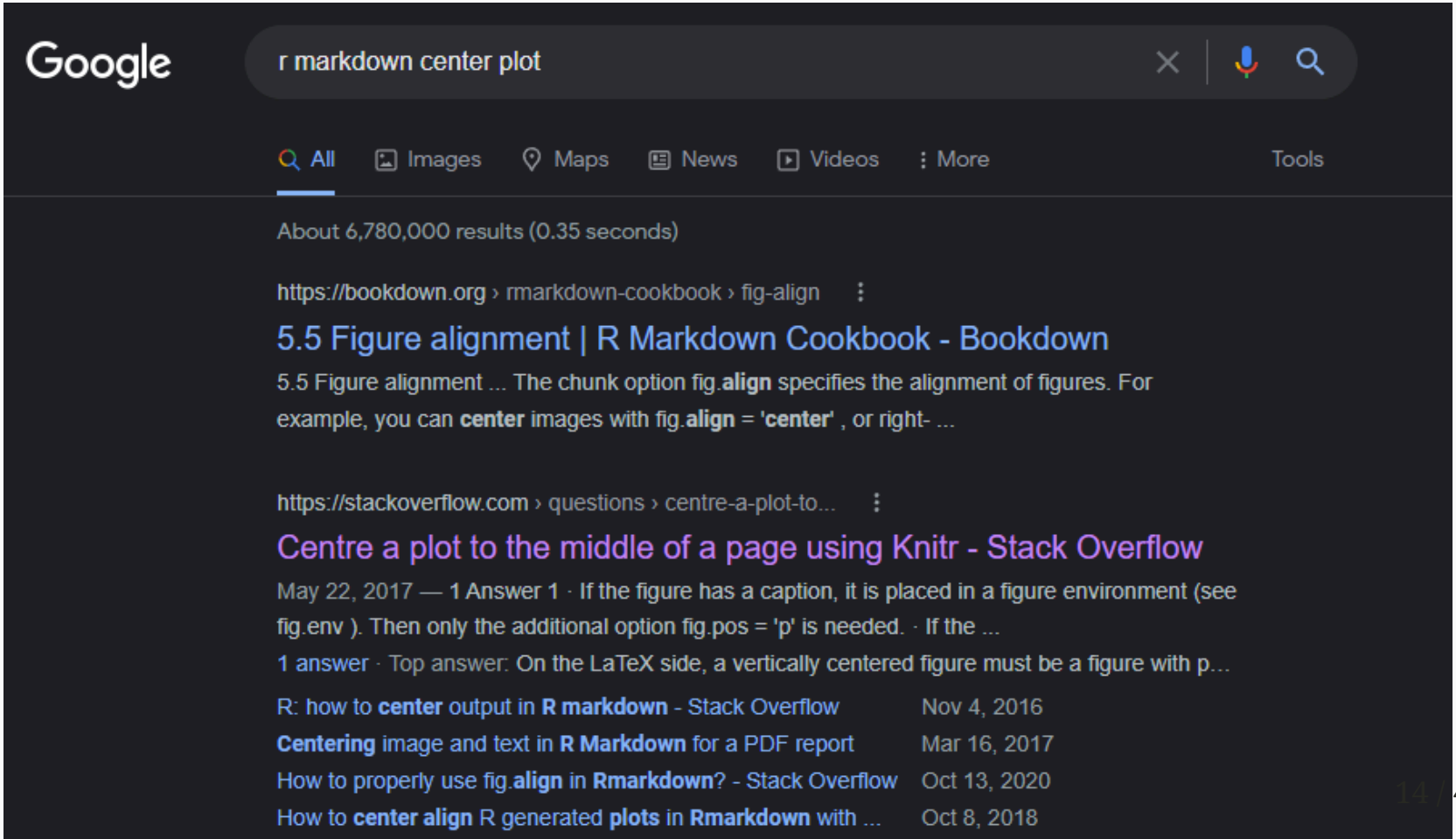
```
?seq_len
```

01:00

Use Google and StackOverflow

These websites are your friends. Especially StackOverflow.

Searching for coding solutions online is a *skill*.



Avoid copy pasting code

I encourage you to work together and to search online for help.

But avoid simply copy/pasting code. If you find some code you want to use, you must *type it in manually*.

Typing the code character-for-character will teach you more than copy pasting.

AI — ChatGPT, Google Gemini, Copilot, etc.

AI is a powerful tool. Use it to your advantage by asking questions and learning.

Abusing AI is a form of self-harm. It will hurt you in the long run.

Abusing AI is a form of academic dishonesty. If discovered, you may fail the class.

Dos and Don'ts with AI

- **Do** use AI to help you learn (e.g. "Explain in different words: ...")
- **Do** use AI to help you debug (e.g. "R error: object not found", "What's wrong with this code?")
- **Do** use AI to help you find solutions (e.g. "R how to sort a vector")
- **Do** use AI to help you understand (e.g. "ELI5 R for loops")

- **Don't** use AI to do your work for you
- **Don't** copy-paste AI answers
- **Don't** use AI to cheat
- **Don't** use AI to avoid learning
- **Don't** use AI to avoid thinking

General recommendation: Use AI to learn/reinforce what we cover in this course. We will notice if you suddenly start using advanced techniques we haven't covered yet.

Tools of the trade

- **R console**: a program that *interprets* R code, one line at a time
- **RStudio**: an *integrated development environment* (IDE)

We will primarily use RStudio, but do note that the above are different things.

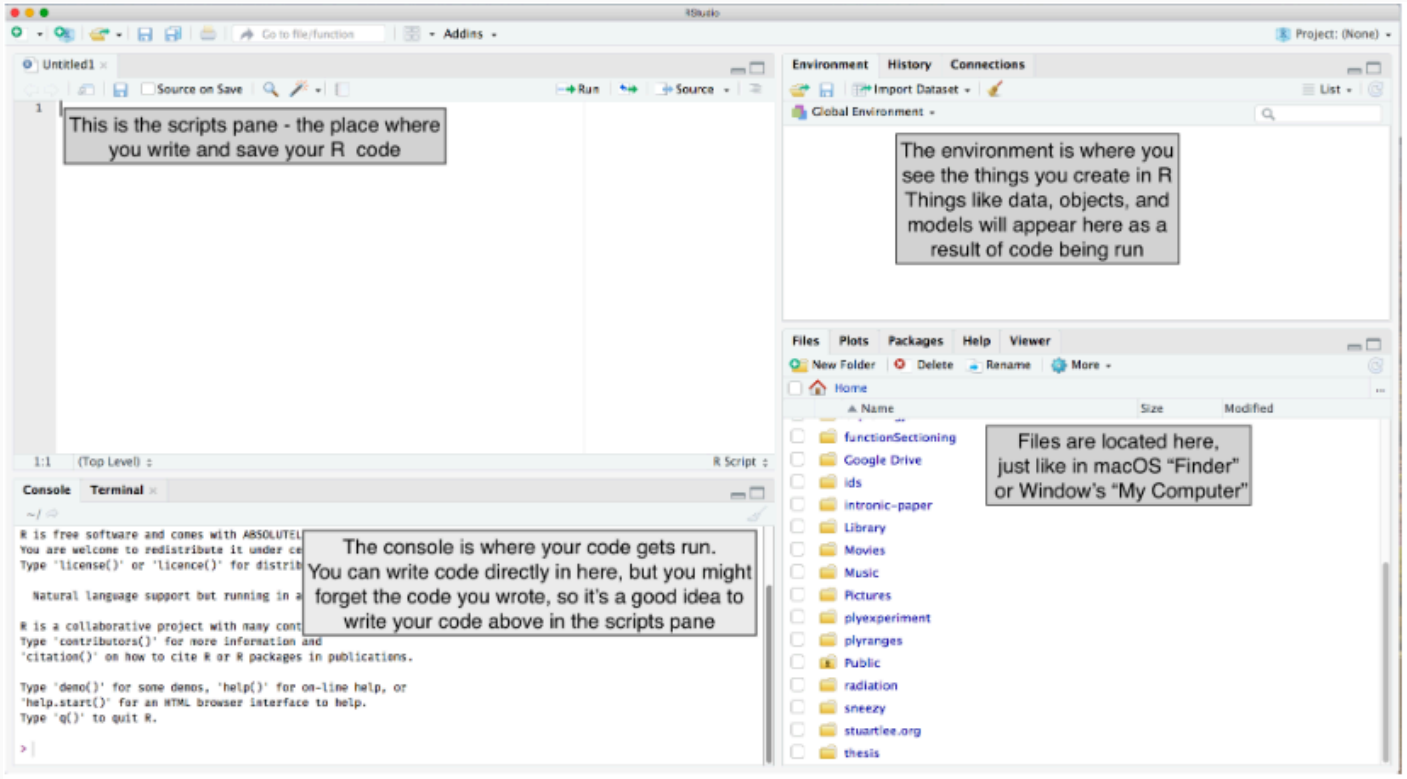
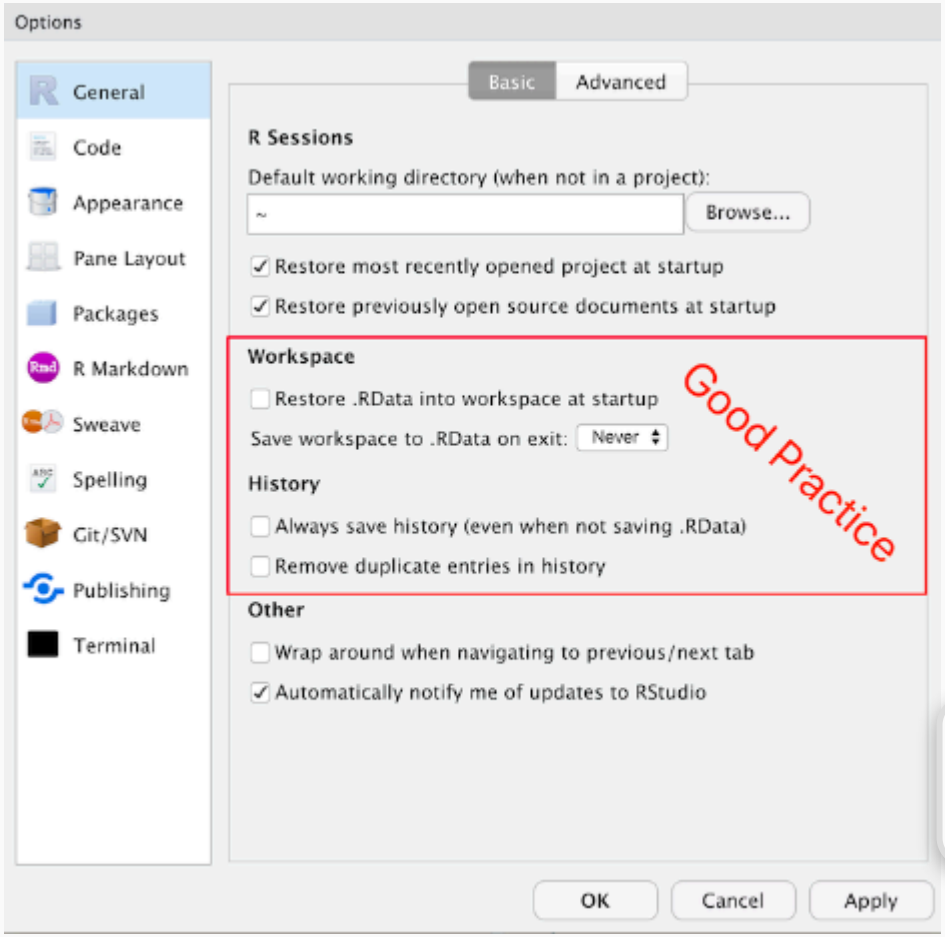


image credit

Customize RStudio

In RStudio, navigate to Tools > Global Options > Appearance and choose a theme that you like.

Also adjust these settings:



05:00

Installing a Package

```
install.packages("cowsay")
```

If you don't like this package...

```
remove.packages("cowsay")
```

Exercise: Where on your computer are these packages actually installed?

Week 1: Programming Concepts

https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf

R as a calculator

```
2 + 3
```

```
## [1] 5
```

```
2 * 3
```

```
## [1] 6
```

```
2^3
```

```
## [1] 8
```

```
exp(4.2) # raise 4.2 to the power of e
```

```
## [1] 66.68633
```

```
log(exp(4.2))
```

```
## [1] 4.2
```

Calculator

Evaluate

$$\frac{5^7 - 2\sqrt{4}}{\log_2(100)}$$

Hint: Type `?log` and `?sqrt` in the console to access the help.

05:00

Assignment operator

- Assignment operator: we usually want to save the result of an expression using `<-`
- Three datatypes: Numeric, Character, Logical

```
hello <- "Hello world!"
print(hello)
```

```
# [1] "Hello world!"
```

```
result <- 55 + 77
result # print function not needed if executing in the console
```

```
# [1] 132
```

```
istrue <- 10 == 10
print(istrue)
```

```
# [1] TRUE
```

Assignment operator

A trick I will sometimes use

```
hello <- "Hello world!"
hello
```

```
## [1] "Hello world!"
```

```
(hello <- "Hello world!")
```

```
## [1] "Hello world!"
```

The dreaded '+' in the console

Run the following

```
hello <- "Hello world!
```

What happens?

Assignment operator

This may be surprising...

```
my_vector <- c(5, 3, 7, 1, 0)
sort(my_vector)
```

```
## [1] 0 1 3 5 7
```

What is the new value of `my_vector`?

```
my_vector
```

```
## [1] 5 3 7 1 0
```

`my_vector` was unchanged by `sort`! We must change it by hand:

```
my_vector <- sort(my_vector)
```

Vectors

The most fundamental data structure in R

Vectors

Vectors are called *atomic* because they can contain only one data type.

Ways to create (atomic) vectors: combine function, colon operator, seq, and rep

```
vec_comb <- c(1, 2, 3, 4, 5, 6)
print(vec_comb)
```

```
## [1] 1 2 3 4 5 6
```

```
vec_colon <- 1:6
print(vec_colon)
```

```
## [1] 1 2 3 4 5 6
```

```
vec_seq <- seq(1, 6, 2)
print(vec_seq)
```

```
## [1] 1 3 5
```

```
vec_seq2 <- seq_len(6)
print(vec_seq2)
```

```
## [1] 1 2 3 4 5 6
```

```
vec_rep <- rep(1:3, 3)
print(vec_rep)
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

```
vec_rep_each <- rep(1:3, each = 3)
print(vec_rep_each)
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

Vectors

Create a vector `x` containing values (1, 2, 6) and a vector `y` containing values (1, 1, 1). What is the result of the following?

```
x + y
```

02:00

Vectors

length and typeof

```
length(0:89)
```

```
## [1] 90
```

```
typeof(0:89)
```

```
## [1] "integer"
```

```
length(c(TRUE, FALSE))
```

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

```
typeof(c(TRUE, FALSE))
```

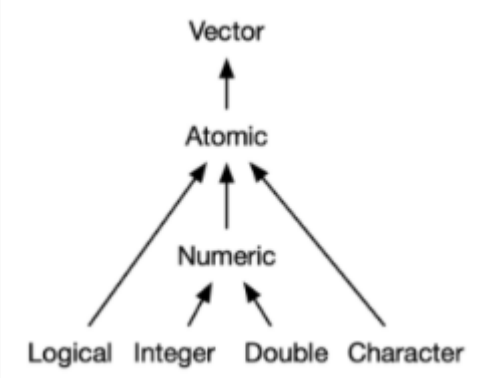
```
## [1] "logical"
```

```
length(100)
```

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

Scalars are length-one vectors

Vector datatype hierarchy



Working with logicals

A logical statement is either TRUE or FALSE

Comparing numerics

```
10 > 10
```

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

```
10 >= 10
```

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

```
5 == 10
```

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

```
5 != 10
```

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

Comparing strings

```
"cat" == "dog"
```

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

```
"cat" != "dog"
```

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

```
"cat" < "dog" # ?? best to avoid
```

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

Logicals

Combining logical expressions

```
5 < 10 & "cat" == "dog" # logical and
```

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

```
5 < 10 | "cat" == "dog" # logical or
```

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

Logicals

Weird but useful facts

- TRUE and FALSE can be abbreviated T and F
- FALSE has numeric value 0
- TRUE has numeric value 1
- What is TRUE + TRUE * FALSE + FALSE?

```
T + T * F + F
```

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

Some built-in functions

```
x <- 11:99
sum(x)
```

```
## [1] 4895
```

```
mean(x)
```

```
## [1] 55
```

```
median(x)
```

```
## [1] 55
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      11      33      55      55      77      99
```

```
y <- c(T, F, F, T, T)
sum(y)
```

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

Subsetting

```
x <- c(1, 3, 5, 6)
x[c(1, 3)] # using a numeric vector
```

```
## [1] 1 5
```

```
x[c(T, F, T, T)] # using a logical vector
```

```
## [1] 1 5 6
```

```
x[-3] # using a negative index
```

```
## [1] 1 3 6
```

What is the result of `x[-c(1, 3)]`?

01:00

Named Vectors

We can name each element of a vector

```
x <- c(1, 3, 5, 6)
names(x) <- c("a", "b", "c", "d")
x
```

```
## a b c d
## 1 3 5 6
```

```
x[c("b", "d")] # subsetting by name
```

```
## b d
## 3 6
```

Vectors

Suppose we have test scores for 5 students: Bob, Alice, Alex, Juan and Amy.

Their scores are 8, 7, 8, 10, and 5 respectively.

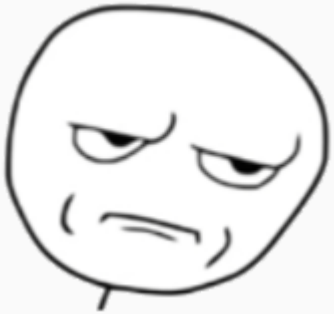
1. Create a vector of these scores.
2. Find the mean score in two ways (using `mean` and using `sum`).
3. Find the median score.
4. Assign the name of each student to their test score.
5. Retrieve Alice's score in two ways.
6. Retrieve Amy's and Alice's score, in that order.
7. Retrieve all except Amy's score.

10:00

Style

Typical R code :(

```
for (i in 1:7) {
for (j in 1:10) {
  tmp <- lm(Life.Exp~X[,rs$which[i,]]-1, subset=fold!=j)
  pred <- X[fold==j,rs$which[i,]]%*%coef(tmp)
  pse.cv[i,j] <- mean((Life.Exp[fold==j]-pred)^2)
}}
```



Follow the tidyverse style guide

A major part of coding is communicating with other developers. It is **very** important to adhere to a style convention.

This is what I use and suggest. Take some time to look at chapters 1 - 3.
https://style.tidyverse.org/

I thought about grading your code style but decided against it. But if your code is unreadable points will be deducted

Consider using the styler addin for RStudio.

Style

In RStudio you should see a faint line at the 80 character mark. It is widespread coding practice across all languages to keep your lines of code under 80 characters per line.

```
x <- 1000 # Make sure you don't go past the 80 character mark or else your code might look
```

A note about comments

Explain tricky code with comments, but do not overuse them.

In my experience, a bad comment is worse than no comment at all.

In this class, comments will sometimes be required for grading purposes. Assignment will say "Comment with the answer".

```
# assign to result the value of 45 plus 64
result <- 45 + 64
print(result) # print the result
```

```
## [1] 109
```

```
# Were the above comments really necessary?
```

Summary

- R Ecosystem
- Vectors
  - numeric, character (strings), logical
- Subsetting Vectors