STREAMLINING WITH R

Meghan Hall
NEAIR
July 12, 2022

HOUSEKEEPING

- Intro 👋
- Workshop materials ⬇️
- Break 🕘
- By the end of today ✔️
- Today's plan 📋

TODAY'S PLAN

1. What is R? How can it ease the burden of repeated reporting?
2. Basic functions for manipulating data
3. Using R effectively
4. More data manipulation
5. Visualizing data
6. A peek at advanced topics

WHAT IS R?

WHAT IS R?

R is an open-source (**free**!) scripting language for working with data

THE BENEFITS OF R

My personal Excel nightmare

The magic of R is that it's **reproducible** (by someone else *or* by yourself in six months)

Keeps data separate from code (data preparation steps)

GETTING R

You need the R language

And also the software

NAVIGATING RSTUDIO

NAVIGATING RSTUDIO

sample - RStudio

NEAIR_code.R

```r
library(tidyverse)

faculty <- read_csv("faculty.csv")
courses <- read_csv("courses.csv")

# filter----

# the `==` operator tests for equality

faculty %>%
  filter(dept1 == "Sociology")

# the `|` operator signifies "or"

faculty %>%
  filter(dept1 == "Sociology" |
         dept1 == "Physics")

# the `%in%` operator allows for multiple options in a list
```

code can also go here

Console

```
chr (4): year, rank, dept1, dept2
dbl (1): id

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> courses <- read_csv("courses.csv")
Rows: 104 Columns: 6
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): dept, level
dbl (4): semester, course_id, faculty_id, enrollment

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

Environment

Data	
courses	104 obs. of 6 variables
faculty	392 obs. of 5 variables

imported data shows up here

Files

Name	Size	Modified
..		
sample.Rproj	258 B	Jul 10, 2022, 4:45 PM
NEAIR_code.R	373 B	Jul 10, 2022, 4:49 PM
courses.csv	3.8 KB	Jul 7, 2022, 11:59 AM
faculty.csv	16.6 KB	Jul 7, 2022, 11:27 AM

project files are here

USING R

You use R via *packages*

...which contain *functions*

...which are just *verbs*

TODAY'S DATA

faculty

year	id	rank	dept1	dept2
2021-22	1005	Lecturer	Chemistry	
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1079	Lecturer	Music	
2021-22	1086	Assistant Professor	Music	
2021-22	1095	Adjunct Instructor	Sociology	

TODAY'S DATA

courses

semester	course_id	faculty_id	dept	enrollment	level
20212202	10605	1772	Physics	7	UG
20212202	10605	1772	Physics	32	GR
20212202	11426	1820	Political Science	8	UG
20212202	12048	1914	English	24	UG
20212202	13269	1095	Sociology	48	UG
20212202	13517	1086	Music	17	UG

BASIC DATA MANIPULATION

USEFUL OPERATORS

`<-`	"save as"	`opt + -`
`%>%`	"and then"	`Cmd + shift + m`

COMMON FUNCTIONS

`filter` keeps or discards rows (aka observations)

`select` keeps or discards columns (aka variables)

`arrange` sorts data set by certain variable(s)

`count` tallies data set by certain variable(s)

`mutate` creates new variables

`group_by`/`summarize` aggregates data (pivot tables!)

`str_*` functions work easily with text

SYNTAX OF A FUNCTION


```
function(data, argument(s))
```

is the same as

```
data %>%
  function(argument(s))
```

FILTER

`filter` keeps or discards rows (aka observations)

the `==` operator tests for equality

```
faculty %>%
  filter(dept1 == "Sociology")
```

year	id	rank	dept1	dept2
2021-22	1095	Adjunct Instructor	Sociology	
2021-22	1118	Assistant Professor	Sociology	
2021-22	1161	Assistant Professor	Sociology	
2021-22	1191	Professor	Sociology	
2021-22	1216	Associate Professor	Sociology	American Studies
2021-22	1273	Assistant Professor	Sociology	

FILTER

the | operator signifies "or"

```
faculty %>%
  filter(dept1 == "Sociology" |
         dept1 == "Physics")
```

year	id	rank	dept1	dept2
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1095	Adjunct Instructor	Sociology	
2021-22	1118	Assistant Professor	Sociology	
2021-22	1161	Assistant Professor	Sociology	
2021-22	1191	Professor	Sociology	

FILTER

the `%in%` operator allows for multiple options in a list

```
faculty %>%
  filter(dept1 %in% c("Sociology",
                      "Physics",
                      "Music"))
```

year	id	rank	dept1	dept2
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1079	Lecturer	Music	
2021-22	1086	Assistant Professor	Music	
2021-22	1095	Adjunct Instructor	Sociology	
2021-22	1118	Assistant Professor	Sociology	

FILTER

the & operator combines conditions

```
faculty %>%
  filter(dept1 %in% c("Sociology",
                      "Physics",
                      "Music") &
         rank == "Professor")
```

year	id	rank	dept1	dept2
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1191	Professor	Sociology	
2021-22	1201	Professor	Physics	
2021-22	1209	Professor	Music	
2021-22	1421	Professor	Physics	Engineering

SELECT

`select` keeps or discards columns (aka variables)

```
faculty %>%
  select(id, dept1, rank)
```

id	dept1	rank
1005	Chemistry	Lecturer
1022	Physics	Professor
1059	Physics	Professor
1079	Music	Lecturer
1086	Music	Assistant Professor
1095	Sociology	Adjunct Instructor

SELECT

can drop columns with `-column`

```
faculty %>%
  select(-dept2)
```

year	id	rank	dept1
2021-22	1005	Lecturer	Chemistry
2021-22	1022	Professor	Physics
2021-22	1059	Professor	Physics
2021-22	1079	Lecturer	Music
2021-22	1086	Assistant Professor	Music
2021-22	1095	Adjunct Instructor	Sociology

SELECT

the pipe %>% chains multiple functions together

```
faculty %>%
  select(id, dept1, rank) %>%
  filter(rank == "Professor")
```

id	dept1	rank
1022	Physics	Professor
1059	Physics	Professor
1191	Sociology	Professor
1201	Physics	Professor
1209	Music	Professor
1407	English	Professor

ARRANGE

`arrange` sorts data set by certain variable(s)

use `desc()` to get descending order

```
courses %>%
  arrange(desc(enrollment))
```

semester	course_id	faculty_id	dept	enrollment
20212201	10511	1005	Chemistry	50
20212201	15934	1421	Physics	50
20192002	13850	1105	Chemistry	50
20181901	17773	1942	Music	50
20212202	13269	1095	Sociology	48
20202101	16202	1816	Political Science	48

ARRANGE

can sort by multiple variables

```
courses %>%
  arrange(dept, desc(enrollment))
```

semester	course_id	faculty_id	dept	enrollment
20212201	10511	1005	Chemistry	50
20192002	13850	1105	Chemistry	50
20202102	13850	1258	Chemistry	39
20202102	16606	1393	Chemistry	38
20202101	16540	1784	Chemistry	38
20181901	10511	1829	Chemistry	36

COUNT

`count` tallies data set by certain variable(s) (very useful for familiarizing yourself with data)

```
courses %>%
  count(dept)
```

dept	n
Chemistry	16
English	18
Music	17
Physics	19
Political Science	17
Sociology	17

COUNT

can use `sort = TRUE` to order results

```
courses %>%
  count(dept, level, sort = TRUE)
```

dept	level	n
Chemistry	UG	16
English	UG	16
Music	UG	16
Physics	UG	16
Political Science	UG	16
Sociology	UG	16
Physics	GR	3
English	GR	2
Music	GR	1
Political Science	GR	1
Sociology	GR	1

MUTATE

`mutate` creates new variables (with a single `=`)

```
faculty %>%
  mutate(new = "hello!")
```

year	id	rank	dept1	dept2	new
2021-22	1005	Lecturer	Chemistry		hello!
2021-22	1022	Professor	Physics	Engineering	hello!
2021-22	1059	Professor	Physics		hello!
2021-22	1079	Lecturer	Music		hello!
2021-22	1086	Assistant Professor	Music		hello!
2021-22	1095	Adjunct Instructor	Sociology		hello!

MUTATE

much more useful with a conditional such as `ifelse()`, which has three arguments:

condition, value if true, value if false

```
faculty %>%
  mutate(prof = ifelse(rank == "Professor",
                       1, 0)) %>%
  select(rank, prof)
```

rank	prof
Lecturer	0
Professor	1
Professor	1
Lecturer	0
Assistant Professor	0
Adjunct Instructor	0

MUTATE

the `!` operator means *not*

`is.na()` identifies null values

```
faculty %>%
  mutate(joint = ifelse(!is.na(dept2),
                        "joint", NA)) %>%
  select(dept1, dept2, joint)
```

dept1	dept2	joint
Chemistry		
Physics	Engineering	joint
Physics		
Music		
Music		
Sociology		

MUTATE

with multiple conditions, `case_when()` is much easier!

```
faculty %>%
  mutate(division = case_when(dept1 %in% c("Sociology","Political Science")
                                "Social Sciences",
                              dept1 %in% c("Music","English") ~
                                "Humanities",
                              dept1 %in% c("Chemistry","Physics") ~
                                "Sciences")) %>%
  select(dept1, division)
```

dept1	division
Chemistry	Sciences
Physics	Sciences
Physics	Sciences
Music	Humanities
Music	Humanities
Sociology	Social Sciences

GROUP BY / SUMMARIZE

`group_by`/`summarize` aggregates data (pivot tables!)

`group_by()` identifies the grouping variable(s) and `summarize()` specifies the aggregation

```
courses %>%
  group_by(dept, semester) %>%
  summarize(enr = sum(enrollment))
```

dept	semester	enr
Chemistry	20181901	59
Chemistry	20181902	44
Chemistry	20192001	47
Chemistry	20192002	68
Chemistry	20202101	69
Chemistry	20202102	77

GROUP BY / SUMMARIZE

useful arguments within `summarize`:

`mean`, `median`, `sd`, `min`, `max`, `n`

```
courses %>%
  group_by(dept, semester) %>%
  summarize(enr = sum(enrollment),
            count = n_distinct(course_id))
```

dept	semester	enr	courses
Chemistry	20181901	59	2
Chemistry	20181902	44	2
Chemistry	20192001	47	2
Chemistry	20192002	68	2
Chemistry	20202101	69	2
Chemistry	20202102	77	2

USING R EFFECTIVELY

WORKING IN RSTUDIO

sample - RStudio

NEAIR_code.R

```r
library(tidyverse)

faculty <- read_csv("faculty.csv")
courses <- read_csv("courses.csv")

# filter----

# the `==` operator tests for equality

faculty %>%
  filter(dept1 == "Sociology")

# the `|` operator signifies "or"

faculty %>%
  filter(dept1 == "Sociology" |
         dept1 == "Physics")

# the `%in%` operator allows for multiple options in a list
```

code can also go here

```
chr (4): year, rank, dept1, dept2
dbl (1): id

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> courses <- read_csv("courses.csv")
Rows: 104 Columns: 6
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): dept, level
dbl (4): semester, course_id, faculty_id, enrollment

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

Data	
courses	104 obs. of 6 variables
faculty	392 obs. of 5 variables

imported data shows up here

Name	Size	Modified
sample.Rproj	258 B	Jul 10, 2022, 4:45 PM
NEAIR_code.R	373 B	Jul 10, 2022, 4:49 PM
courses.csv	3.8 KB	Jul 7, 2022, 11:59 AM
faculty.csv	16.6 KB	Jul 7, 2022, 11:27 AM

project files are here

WORKING IN RSTUDIO

Typing in the console

- think of it like a post-it: useful for quick notes but disposable
- actions are saved but code is not
- one chunk of code is run at a time (`Return`)

Typing in a code file

- script files have a `.R` extension
- code is saved and sections of any size can be run (`Cmd + Return`)
- do ~95% of your typing in a code file instead of the console!

WORKING WITH PACKAGES

packages need to be installed on each computer you use

```
# only need to do this once (per computer)
install.packages("tidyverse")
```

packages need to be loaded/attached with `library()` at the beginning of every session

```
# always put the necessary packages at the top of a code file
library(tidyverse)
```

can access help files by typing `??tidyverse` or `??mutate` in the console

ORGANIZING WITH PROJECTS

highly recommend using *projects* to stay organized

keeps code files and data files together, allowing for easier file path navigation and better reproducible work habits

`File -> New Project`

more guidance: here and here

ORGANIZING WITH PROJECTS

sample - RStudio

NEAIR_code.R

```r
library(tidyverse)

faculty <- read_csv("faculty.csv")
courses <- read_csv("courses.csv")

# filter----

# the `==` operator tests for equality

faculty %>%
  filter(dept1 == "Sociology")

# the `|` operator signifies "or"

faculty %>%
  filter(dept1 == "Sociology" |
         dept1 == "Physics")

# the `%in%` operator allows for multiple options in a list
```

code can also go here

Console

```
chr (4): year, rank, dept1, dept2
dbl (1): id

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> courses <- read_csv("courses.csv")
Rows: 104 Columns: 6
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): dept, level
dbl (4): semester, course_id, faculty_id, enrollment

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

Environment

Data	
courses	104 obs. of 6 variables
faculty	392 obs. of 5 variables

imported data shows up here

Files

Name	Size	Modified
sample.Rproj	258 B	Jul 10, 2022, 4:45 PM
NEAIR_code.R	373 B	Jul 10, 2022, 4:49 PM
courses.csv	3.8 KB	Jul 7, 2022, 11:59 AM
faculty.csv	16.6 KB	Jul 7, 2022, 11:27 AM

project files are here

ACCESSING WORKSHOP MATERIALS

click big green Code button and select “Download ZIP”, then open `neair.Rproj`

github.com/meghall06/neair

ACCESSING DATA

use `read_csv()` to import a csv file

```
# the file path is this simple if you use projects!
# ?read_csv() in the console will bring up the help file with more options
faculty <- read_csv("faculty.csv")
```

the readxl package is helpful for Excel files

```
# needs to be loaded but not installed as it's part of the tidyverse
library(readxl)
faculty <- read_excel("faculty.xlsx", sheet = 2)
```

view the data with `View(faculty)` or by clicking on the data name in the Environment pane

MORE DATA MANIPULATION

STRINGR FUNCTIONS

functions from `stringr` (which all start with `str_`) are useful for working with text data

```
faculty %>%
  filter(str_detect(rank, "Professor"))
```

year	id	rank	dept1	dept2
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1086	Assistant Professor	Music	
2021-22	1118	Assistant Professor	Sociology	
2021-22	1158	Assistant Professor	Political Science	
2021-22	1161	Assistant Professor	Sociology	

STRINGR FUNCTIONS

cheat sheet of functions is here

```
courses %>%
  mutate(year = str_c(str_sub(semester, 1, 4),
                      "-",
                      str_sub(semester, 5, 6))) %>%
  select(semester, year) %>%
  unique()
```

semester	year
20212202	2021-22
20212201	2021-22
20202102	2020-21
20202101	2020-21
20192002	2019-20
20192001	2019-20
20181902	2018-19
20181901	2018-19

PIVOTING DATA

existing `faculty` data has one row per faculty, some with multiple departments (sometimes known as *wide* data)

year	id	rank	dept1	dept2
2021-22	1005	Lecturer	Chemistry	
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1079	Lecturer	Music	
2021-22	1086	Assistant Professor	Music	
2021-22	1095	Adjunct Instructor	Sociology	

PIVOTING DATA

what if you instead want one row per faculty *per department*? (sometimes known as *long* data)

year	id	rank	dept_no	dept
2021-22	1005	Lecturer	dept1	Chemistry
2021-22	1022	Professor	dept1	Physics
2021-22	1022	Professor	dept2	Engineering
2021-22	1059	Professor	dept1	Physics
2021-22	1079	Lecturer	dept1	Music
2021-22	1086	Assistant Professor	dept1	Music

PIVOTING DATA

the `pivot_longer` function lengthens data

```
faculty %>%
  pivot_longer(dept1:dept2,
               names_to = "dept_no",
               values_to = "dept",
               values_drop_na = TRUE) %>%
  select(-year, -rank)
```

id	dept_no	dept
1005	dept1	Chemistry
1022	dept1	Physics
1022	dept2	Engineering
1059	dept1	Physics
1079	dept1	Music
1086	dept1	Music

PIVOTING DATA

and `pivot_wider` does the opposite!

semester	course_id	faculty_id	dept	enrollment	level
20212202	10605	1772	Physics	7	UG
20212202	10605	1772	Physics	32	GR

```
courses %>%
  pivot_wider(names_from = "level",
              values_from = "enrollment")
```

semester	course_id	faculty_id	dept	UG	GR
20212202	10605	1772	Physics	7	32
20212202	11426	1820	Political Science	8	
20212202	12048	1914	English	24	
20212202	13269	1095	Sociology	48	

JOINING DATA

R has many useful functions for handling relational data

all you need is at least one key variable that connects data sets

`left_join` is most common, but there are more

JOINING DATA

what's the average UG enrollment per year, per faculty rank?

faculty

year	id	rank	dept1	dept2
2021-22	1005	Lecturer	Chemistry	
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1079	Lecturer	Music	

courses

semester	course_id	faculty_id	dept	enrollment	level
20212202	10605	1772	Physics	7	UG
20212202	10605	1772	Physics	32	GR
20212202	11426	1820	Political Science	8	UG
20212202	12048	1914	English	24	UG

`faculty$id` is the same as `courses$faculty_id`

JOINING DATA

what's the average UG enrollment per year, per faculty rank?

semester	course_id	faculty_id	dept	enrollment	level
20212202	10605	1772	Physics	7	UG
20212202	10605	1772	Physics	32	GR
20212202	11426	1820	Political Science	8	UG
20212202	12048	1914	English	24	UG
20212202	13269	1095	Sociology	48	UG

- filter to UG courses only
- create our `year` variable again
- summarize `enrollment` by `year` and `faculty_id`

JOINING DATA

use the <– operator to create a new data frame

`courses_UG`

```
courses_UG <- courses %>%
  filter(level == "UG") %>%
  mutate(year = str_c(str_sub(semester, 1, 4),
                      "-",
                      str_sub(semester, 5, 6)))
```

JOINING DATA

`filter` to undergraduate courses only and `mutate` a new academic year variable

```
courses_UG <- courses %>%
  filter(level == "UG") %>%
  mutate(year = str_c(str_sub(semester, 1, 4),
                      "-",
                      str_sub(semester, 5, 6)))
```

JOINING DATA

group_by year and faculty member; summarize enrollment

```
courses_UG <- courses %>%
  filter(level == "UG") %>%
  mutate(year = str_c(str_sub(semester, 1, 4),
                      "-",
                      str_sub(semester, 5, 6))) %>%
  group_by(year, faculty_id) %>%
  summarize(enr = sum(enrollment))
```

year	faculty_id	enr
2018-19	1059	35
2018-19	1086	14
2018-19	1102	37
2018-19	1203	25

JOINING DATA

what's the average UG enrollment per year, per faculty rank?

faculty

year	id	rank	dept1	dept2
2021-22	1005	Lecturer	Chemistry	
2021-22	1022	Professor	Physics	Engineering
2021-22	1059	Professor	Physics	
2021-22	1079	Lecturer	Music	
2021-22	1086	Assistant Professor	Music	
2021-22	1095	Adjunct Instructor	Sociology	

courses_UG

year	faculty_id	enr
2021-22	1005	50
2021-22	1086	17
2021-22	1095	48
2021-22	1128	32
2021-22	1147	32
2021-22	1191	7

JOINING DATA

1 2

```
fac_enr <- faculty %>%
  left_join(courses_UG, by = c("id" = "faculty_id",
                               "year" = "year"))
```

3

1. new data frame
2. data frame you're adding data to
3. data frame where the new data is coming from

year	id	rank	dept1	dept2	enr
2021-22	1005	Lecturer	Chemistry		50
2021-22	1022	Professor	Physics	Engineering	
2021-22	1059	Professor	Physics		
2021-22	1079	Lecturer	Music		
2021-22	1086	Assistant Professor	Music		17
2021-22	1095	Adjunct Instructor	Sociology		48

JOINING DATA

what's the average UG enrollment per year, per faculty rank?

```
fac_enr <- faculty %>%
  left_join(courses_UG, by = c("id" = "faculty_id",
                               "year" = "year")) %>%
  group_by(year, rank) %>%
  summarize(avg_enr = mean(enr, na.rm = TRUE))
```

year	rank	avg_enr
2021-22	Adjunct Instructor	34.66667
2021-22	Assistant Professor	23.60000
2021-22	Associate Professor	17.25000
2021-22	Lecturer	31.83333
2021-22	Professor	32.16667
2021-22	Visiting Researcher	

DATA VISUALIZATION

GGPLOT2

`ggplot2` is the data visualization package that is loaded with the `tidyverse`

the grammar of graphics maps data to the aesthetic attributes of geometric points

encoding data into visual cues (e.g., length, color, position, size) is how we signify changes and comparisons

BAR CHART


```
faculty %>%
  count(rank) %>%
  ggplot(aes(x = rank, y = n)) +
  geom_bar(stat = "identity")
```

to combine lines into one code chunk, use + instead of %>%

BAR CHART

can create a prettier plot pretty easily

- expand for full code

LINE GRAPH


```
fac_enr %>%
  filter(!is.na(avg_enr)) %>%
  ggplot(aes(x = year, y = avg_enr, group = rank, color = rank)) +
  geom_line()
```

LINE GRAPH

- expand for full code

GGPLOT2 RESOURCES

from R for Data Science

Data Visualization: a practical introduction

creating custom themes

the ggplot2 book

the R graph gallery

PUTTING IT ALL TOGETHER

with what we've done so far, your `.R` file could:

- import your data files
- document all data cleaning and preparation steps and decisions
- produce a PPT-ready graphic summarizing your results

and that file would make it extremely easy for you or someone else to reproduce this analysis with new data in six months

ADVANCED TOPICS

R MARKDOWN

using RStudio, create `.Rmd` documents that combine text, code, and graphics

many output formats: html, pdf, Word, slides

exceedingly useful for parameterized reporting: can create an R-based PDF report and generate it automatically for, say, each department

INTERNAL PACKAGES

you can also create your *own* packages!

your package can hold:

- common data sets that are used across projects
- custom `ggplot2` themes
- common functions and calculations (and their definitions!)

can be stored on a shared drive to facilitate collaboration

R MARKDOWN AND PACKAGE RESOURCES

R Markdown

the official R Markdown website

R Markdown: The Definitive Guide

internal packages

a comprehensive theoretical explainer

a talk I gave earlier this year on the topic

LEARN MORE ABOUT R

RESOURCES

R for Data Science: the ultimate guide

R for Excel users: a very useful workshop

STAT 545: an online book on reproducible data analysis in R

the RStudio Education site

the Learn `tidyverse` site