Northeastern University

May Institute goes ONLINE!
Computation and statistics for mass spectrometry and proteomics

May 3 – May 14, 2021, Northeastern University, Boston MA
Organizer: Olga Vitek

May 8, 2021

11:00am-2:15pm **Hands-on, Ryan Benz:** Part 2: Introduction to R for beginners

Starting soon

Materials at
https://computationalproteomics.khoury.northeastern.edu/

NIH National Institutes of Health

Chan Zuckerberg Initiative

ASA Boston Chapter

Tidy data & the tidyverse

May Institute 2021: Introduction to R for Beginners

Day 2 – Module 5

Goals for this module

- Understand what tidy data is, and how it differs from messy data
- Learn how to recognize messy data and why it’s hard to work with
- Learn about the tidyverse and why it’s a great ecosystem for working with data in R

Quick note about the terms “messy” & “tidy”

Messy

- Has negative connotations but that’s not the intent here
- Messy data is simply data that’s not yet in a form suitable for analysis
- Messy data doesn’t mean bad data

Tidy

- Has positive connotations but that’s not the intent here either
- Tidy data is data in a consistent format that supports the analysis process
- Tidy data doesn’t mean good data

In the R ecosystem, the term “tidy data” has a very specific meaning
(that we’ll get to soon)

The Anna Karenina principle…

> Happy families are all alike;
> every unhappy family is unhappy in its own way.

– Leo Tolstoy

… applies to data too

Tidy datasets

> ~~Happy families~~ are all alike;
>
> every ~~unhappy family is~~ un~~happy in~~ its own way.

messy dataset *messy*

– Hadley Wickham

Messy data

Most “real world” data starts out messy...

...which makes it hard to work with.

All tidy data has similar structure...

...making it easier to work with since you know what to expect

Data can be messy in all kinds of ways

Excel spreadsheet with complex formatting

Professor Smith: Data Science Grades Sheet, 1st Trimester 2020

University of DS, Department of Biology

Class Date: 1/6/2020 to 3/27/2020

	Quizzes			Tests		
Name	**Score 1**	**Score 2**	**Score 3**	**Test 1**	**Midterm**	**Final**
al-Jafri, Anas	17 / 25	41 / 50	18 / 25	87 / 100	89 / 100	90 / 100
Drum, Anyssa	19 / 25	40 / 50	19 / 25	88 / 100	87 / 100	92 / 100
Gonzales, David	18 / 25	38 / 50	19 / 25	84 / 100	91 / 100	83 / 100
Granger, Shannan	14 / 25	42 / 50	20 / 25	89 / 100	82 / 100	90 / 100
Kwak, Surabhi	16 / 25	41 / 50	21 / 25	85 / 100	88 / 100	93 / 100
Michels, Breana	17 / 25	46 / 50	18 / 25	80 / 100	85 / 100	82 / 100
Reed, Tyler	19 / 25	39 / 50	17 / 25	78 / 100	94 / 100	88 / 100
Smith, Tyler	16 / 25	40 / 50	19 / 25	79 / 100	90 / 100	86 / 100
Smith, Westin	15 / 25	45 / 50	17 / 25	87 / 100	85 / 100	91 / 100
Vang, Seher	19 / 25	43 / 50	19 / 25	79 / 100	82 / 100	87 / 100

Data tables in PDF (or even images!)

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Tidy Data, H. Wickham, J. Stat. Software

Badly formatted text files

```
1, 4, 5, 6, 10
10,000, 900, 874
5,6,8
7473,134187,1398,17
```

Proprietary MS data formats 💾

Data in web pages, plots & figures, ...

Messy data can be hard to work with

- Unclear structure & organization can make it hard to understand what's there
- Data might be optimized for data entry or visual consumption, not computer consumption
- Might be represented in a strange ways (e.g. numbers combined with words, unclear coding variables)
- Might be in multiple places (e.g. in different files)
- Might be in strange formats (html, pdf, png 🙀)

Messy data isn’t necessarily bad data

- Not all data is created or presented with data analysis in mind
- The people who curate the data might not understand the needs of someone who needs to work with it
- The goals for the data might not align directly with data analysis needs (presenting data on a slide, performance or storage requirements)
- ... but messy data might be a sign of lurking data problems too 🕵️

Tidy data in the R ecosystem

Tidy data has consistent structure, arranged in a rectangular table

Variables (columns)
the “things” you are measuring

Observations (rows)
the “things” you are making measurements on

Values (cells)
the values of the measurements

figure: https://r4ds.had.co.nz/tidy-data.html

Example Tidy Data Table

Run	Condition	BioReplicate
JD_06232014_sample1-A.raw	Condition1	1
JD_06232014_sample1_B.raw	Condition1	1
JD_06232014_sample1_C.raw	Condition1	1
JD_06232014_sample2_A.raw	Condition2	2
JD_06232014_sample2_B.raw	Condition2	2
JD_06232014_sample2_C.raw	Condition2	2
JD_06232014_sample3_A.raw	Condition3	3
JD_06232014_sample3_B.raw	Condition3	3
JD_06232014_sample3_C.raw	Condition3	3
JD_06232014_sample4-A.raw	Condition4	4
JD_06232014_sample4_B.raw	Condition4	4
JD_06232014_sample4_C.raw	Condition4	4

What is this table about?

What does each row represent?

What is being measured
(what are the columns)?

A few notes about tidy data

- Tidy data is not the only way
  - Other structures might be needed to optimize for performance or storage
  - Certain fields might follow other data conventions
  - Some types of data might not naturally fit into a rectangular table

- BUT, if your data *can* fit into rectangular structure, tidy data is usually the way to go

- Sometimes the differences between observations and variables is not always clear, and you might swap them depending on the context

- Data tidiness isn't necessarily black & white, different circumstances might require different levels of tidiness

Be on the lookout for signs of messy data!

- Overall structure of the data is unclear or inconsistent
- Multiple pieces of information are stored in a single cell, e.g. `Male_age16, Female_age42`
- A variable is spread across multiple columns
- An observation is spread across multiple rows

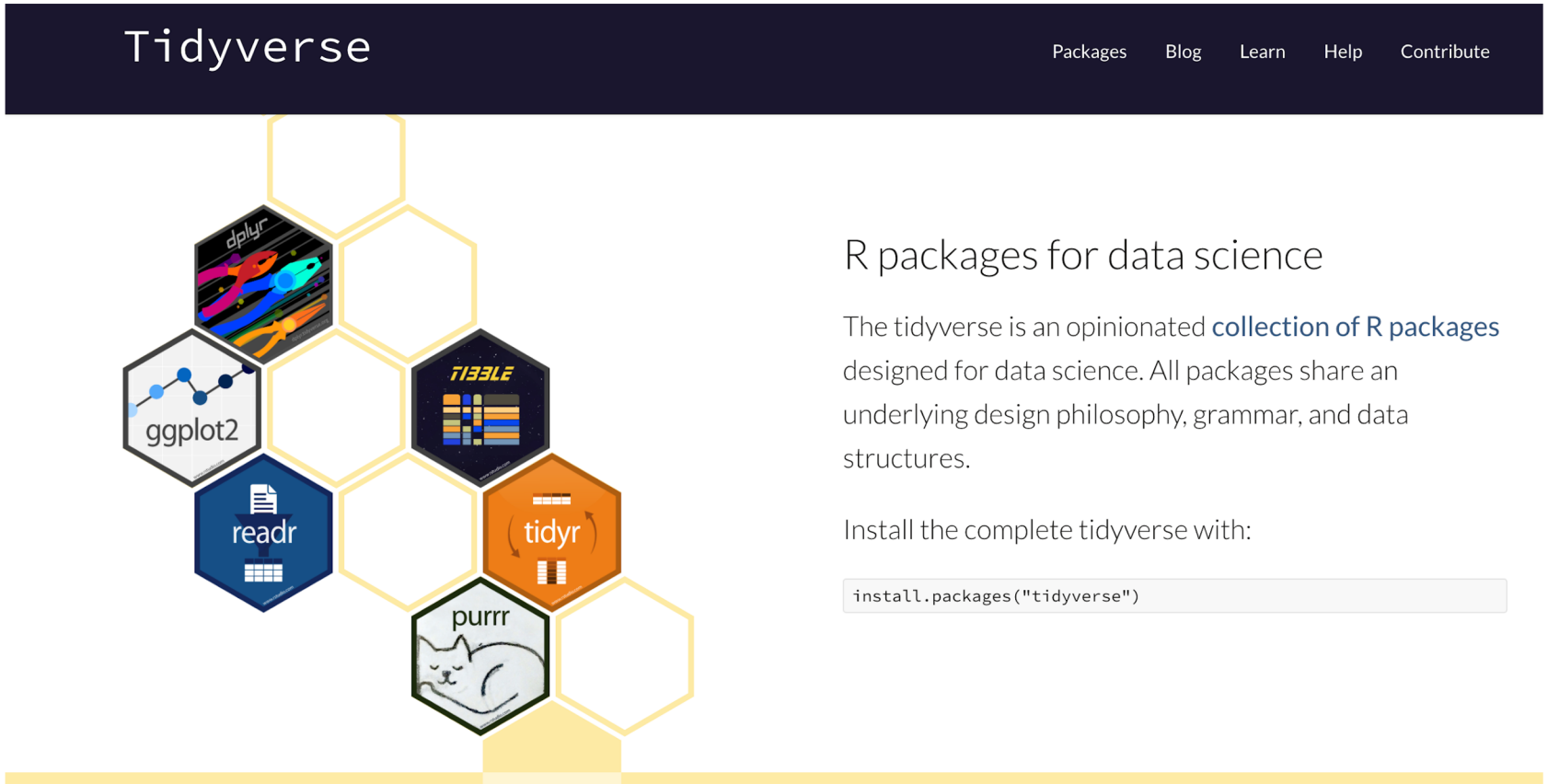
As you get more experience recognizing messy data, you can better communicate to others (i.e. collaborators) how to produce well structured data in order to make your job easier as a data analyst!

How Might We Further Tidy This Table?

Run	Condition	BioReplicate
JD_06232014_sample1-A.raw	Condition1	1
JD_06232014_sample1_B.raw	Condition1	1
JD_06232014_sample1_C.raw	Condition1	1
JD_06232014_sample2_A.raw	Condition2	2
JD_06232014_sample2_B.raw	Condition2	2
JD_06232014_sample2_C.raw	Condition2	2
JD_06232014_sample3_A.raw	Condition3	3
JD_06232014_sample3_B.raw	Condition3	3
JD_06232014_sample3_C.raw	Condition3	3
JD_06232014_sample4-A.raw	Condition4	4
JD_06232014_sample4_B.raw	Condition4	4
JD_06232014_sample4_C.raw	Condition4	4

The Tidyverse

An opinionated collection of R packages for data science



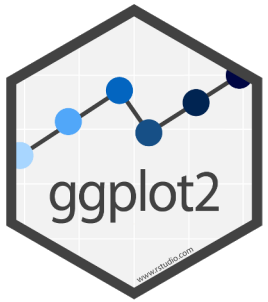
https://www.tidyverse.org

Once you know the general structure of the input data (i.e. tidy data), you can build all kinds of tools to work with it

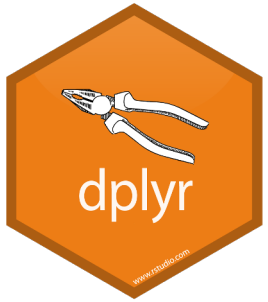
That's the tidyverse!

The tidyverse covers the fundamental components of the data analysis workflow

Core tidyverse Packages



Data Visualization



Fundamental Data Manipulation & Pipelines

Read Data
Tidy Data

Manipulate Specific Types of Data

https://www.tidyverse.org

And LOTS of other packages developed by others...

Recap

- Tidy data is well structured data that's ready for analysis
- In a tidy data table
  - each row is an observation
  - each column is a variable
  - each (individual) piece of data goes in its own cell
- Messy data is everywhere – as you get more proficient with R, you'll learn how to better deal with messy data
- The tidyverse leverages the well-structured nature of tidy data to provide awesome tools for working with data

Basic Data Manipulation with dplyr

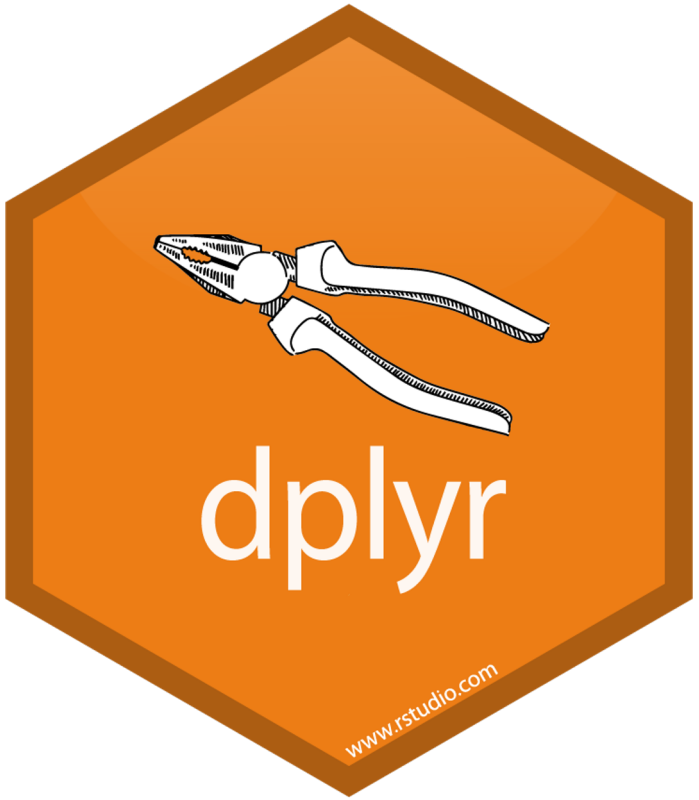
May Institute 2021: Introduction to R for Beginners

Day 2 – Module 6

Goals of the Module

- Understand fundamental operations that underlie most common data manipulation challenges
- Learn about the `dplyr` R package and how it helps you perform these data manipulation operations
- Learn about the pipe `%>%` operator
- Learn how to build data manipulation and transformation pipelines with `dplyr`

`dplyr` is a tidyverse R Package for Data Manipulation

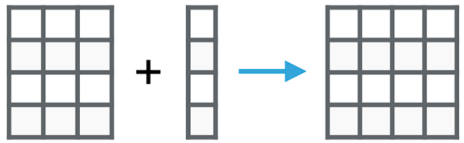
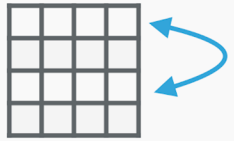
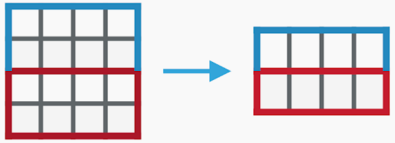


- Defines the fundamental operations that encompass most data analysis tasks
- Assumes you already have tidy data
- Uses a pipeline coding structure to perform complex operations a straight-forward, natural way

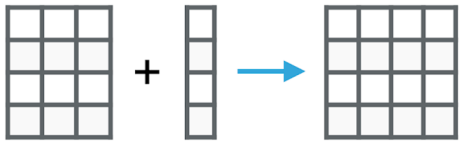
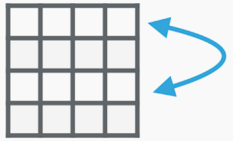
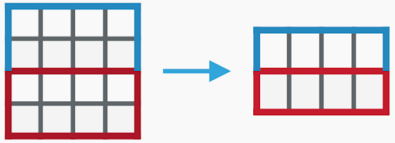
`dplyr` Defines Data Manipulation *Verbs*

- dplyr formalizes the fundamental operations that occur when working with data into a set of “verbs”
- These verbs are represented as functions that you can use to manipulate data in R
- There are a small number of these verbs, which makes them easy(er) to remember and work with
- Helps you to focus on the question you want to answer rather than the mechanics of how to answer the question

Fundamental Data Manipulation Operations

Operation		Use case
Add a column		compute new data from existing variables, join new data
Pick specific columns		focus in on specific variables
Subset to specific rows		focus on a specific sub-group in the data
Reorder/sort rows		understand the order of the data, find top/bottom observations
Group subsets		want to analyze sub-groups in your data
Summarize rows		compute summary values across multiple rows, useful with grouping

Fundamental Data Manipulation Operations

Operation		Use case	dplyr verb
Add a column		compute new data from existing variables, join new data	`mutate`
Pick specific columns		focus in on specific variables	`select`
Subset to specific rows		focus on a specific sub-group in the data	`filter`
Reorder/sort rows		understand the order of the data, find top/bottom observations	`arrange`
Group subsets		want to analyze sub-groups in your data	`group_by`
Summarize rows		compute summary values across multiple rows, useful with grouping	`summarize`

Using dplyr Verbs (functions)

- The first argument of a dplyr function is *always* the data frame that you want to operate on, e.g.

  `mutate(my_df, …)` – *adds a column to my_df*

  `select(my_df, …)` – *subsets to specific columns of my_df*

- The … above are additional arguments that further define what the operation should do

- dplyr verbs always give back a data frame

  ```
  my_df2 <- mutate(my_df, …)
  ```

  my_df2 is a data frame that has a new column

dplyr Verb Examples: mutate

- You use mutate when you want to add a new column to your data frame, often based upon existing columns

```
# load the tidyverse package
library(tidyverse)

# load the data
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")

# add a new column that is 10 times the value of the BioReplicate column
dat <- mutate(dat, rep_times_ten = BioReplicate * 10)
```

It's common to re-assign the data frame

name of the new column that will be created

the expression that will be used to fill in the values

Note: you don't use quotes around the column name

dplyr Verb Examples: select

- You use select when you to to get specific columns, or get rid of ones you don't want

```
# load the tidyverse package
library(tidyverse)

# load the data
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")

# get only the Condition and BioReplicate columns
dat <- select(dat, c("Condition", "BioReplicate"))
```


give a vector of column name
that you want to choose

dplyr Verb Examples: filter

- You use filter when you to to get specific rows, very often specified using a conditional expression

```
# load the tidyverse package
library(tidyverse)

# load the data
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")

# get only the rows for Condition4
dat <- filter(dat, Condition == "Condition4")
```

this is a conditional expression
that tests values in the
Condition column

dplyr Verb Examples: arrange

- You use arrange when you want to re-order the rows of a data frame

```
# load the tidyverse package
library(tidyverse)

# load the data
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")

# sort the rows by Condition
dat <- arrange(dat, Condition)

# use desc to reverse the sorting
dat <- arrange(dat, desc(Condition))
```

orders the rows based on the (alphabetical) sorting of Condition

dplyr Verb Examples: group_by + summarize

- These two verbs are usually used together, when you want to group by a particular column and compute summaries for each group

```
# load the tidyverse package
library(tidyverse)

# load the data
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")

# count the number of runs per condition
dat_grouped <- group_by(dat, Condition)
summarize(dat_grouped, n_runs = length(Run))
```

makes 4 groups, one for each Condition

for each individual group…

compute a new column…

that has the number of Run values

Practice Time – Using dplyr Verbs

1. Use the following code to read the exercise data set:

```
library(tidyverse)
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")
```

2. Run the `select` and `filter` examples from previous slides. Do these operations look familiar? Can you accomplish the same things using alternative code you learned from Day 1?

3. Run the group_by + summarize example and pay close attention to the output. What determines how many rows the final result has?

The Pipe Operator: %>%

- The pipe operator takes an input data frame and “feeds” it into a function
  … the function performs an operation on the input data
- In code, the pipe operator is three individual characters
- Think of %>% as a pipe, funneling the input data into the function to do some work

tidy data frame %>% dplyr verb *produces a* modified data frame

The Pipe Operator: %>%

Template

tidy data frame %>% dplyr verb *produces a* modified data frame

---

tidy data frame %>% `mutate` *produces a* modified data frame with a new column

tidy data frame %>% `select` *produces a* modified data frame with selected columns

tidy data frame %>% `filter` *produces a* modified data frame with filtered rows

Practice with Pipes

R Code without pipes

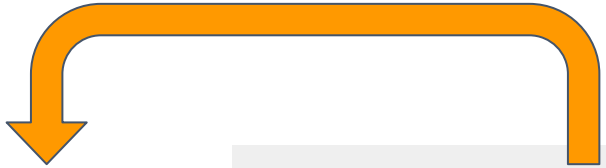
```
filter(dat, Condition == "Condition4")
```

R Code *with* pipes

```
dat %>% filter(Condition == "Condition4")
```

Both lines of code do exactly the same thing, but are written differently
Can you spot the differences?

Practice with Pipes



```
filter(dat, Condition == "Condition4")
```

```
dat %>% filter(Condition == "Condition4")
```

Two different ways to code the same thing

Take the first argument (the input data frame),
move it to the front and add a pipe, %>%

Practice Time – Practice with Pipes

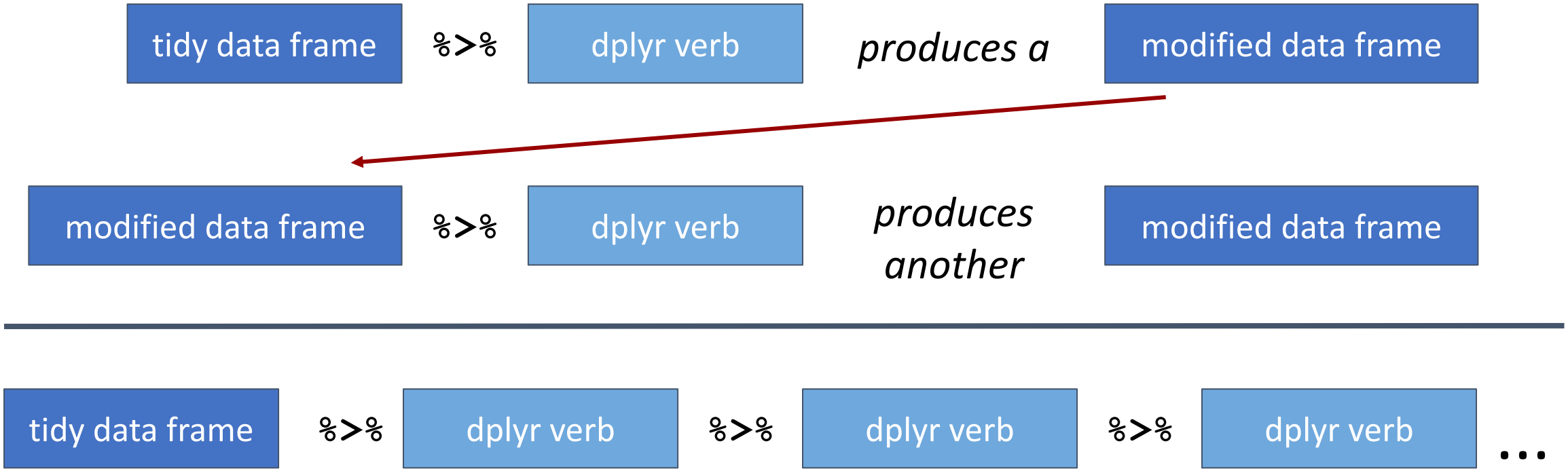
1. Use the following code to read the exercise data set:
```
library(tidyverse)
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")
```

2. Rewrite the `arrange` example using pipe syntax
```
dat <- arrange(dat, Condition)
```

3. Make a new vector with the values 1 to 5 and assign it to a new variable. Write the code to get the maximum value of the vector.

4. Try re-writing the code in 3. using pipe syntax
Hint: follow the pattern on the previous slide

The Pipe Operator Can Chain Multiple Verbs (pipeline!)



You can chain (pipe) together dplyr verbs to produce pipelines that embody complex manipulations

dplyr Pipeline Example

Do the following:

1. only get BioReplicate values of 2 or higher, then…
2. count the number of Run values per Condition, then…
3. order the results in descending Condition order

```
# load the tidyverse package
library(tidyverse)

# load the data
dat <- read_csv("Choi2017_DDA_Skyline_annotation.csv")

# accomplish the above steps in a dplyr pipeline
dat %>%
  filter(BioReplicate >= 2) %>%
  group_by(Condition) %>%
  summarize(n_runs = length(Run)) %>%
  arrange(desc(Condition))
```

"only get BioReplicate values of 2 or higher"

"count the number of Run values per Condition"

"order the results in descending Condition order"

dplyr pipelines can (and often are) placed on multiple lines (make sure %>% is at the end of a line)

Recap

- dplyr provides a general framework for manipulating tidy data
- The fundamental manipulations are specified as verbs (functions)
- These verbs can be combined (piped) together to create data processing pipelines
- Often, these pipelines can often be translated from “human-form” to R code in a straight-forward, natural way
- dplyr helps you to focus on answering the question rather than the mechanics of how to answer the question

Recap & Next Steps

May Institute 2021: Introduction to R for Beginners

Wrap-up

What we Learned

- What is R & RStudio
- Basic R Syntax:
  - variable assignment
  - working with functions
  - subsetting
  - conditional expressions
- How to work with vectors and data frames
- How to read data into R
- What tidy data is
- How to use dplyr to operate on data frames

Are You Now Able To...

- start-up RStudio, make an RStudio project & write/execute code?
- create variables and vectors, and perform calculations with them?
- read a formatted text file of data into R?
- understand basic properties about data tables?
- tell someone what tidy data is and why it’s important?
- perform basic data manipulations and operations on data tables?

Resources

Websites

- Rstudio's Learning References: https://education.rstudio.com/learn/beginner/
- The tidyverse: https://www.tidyverse.org/

Books

- Hands on Programming with R: https://rstudio-education.github.io/hopr/
- R for Data Science: https://r4ds.had.co.nz/

People/Orgs to Follow

- Prof. Olga Vitek and Team
- Bioconductor (https://www.bioconductor.org)
- RStudio
- Hadley Wickham (@hadleywickham)
- Jenny Bryan (@JennyBryan)
- Julia Silge (@juliasilge)
- Mara Averick (@dataandme)
- David Robinson (@drob)
- Twitter hash-tags: #rstats, #DataScience

Where to get help

- Google is a good place to start
  - copy error messages and search with quotes: "*error message here*"
  - try to use R specific/centric terms when possible in your search, e.g. data frame, tidyverse, package names, etc.
- Google will often link you to Stack Overflow https://stackoverflow.com/, you can often get good help here, but it can be a mixed bag
- RStudio Community https://community.rstudio.com/ is a great, friendly place to search for help and solutions, but not as extensive as Stack Overflow (yet?); worth it to just browser the topics
- Join a local R Users group (highly recommended!)

Thank you to the teaching assistants!

Sai Srikanth Lakkimsetty

Ritwik Anand

Vartika Tewari

Sara Taheri

Devon Kohler

Thank you!

Question time…