# Introduction to Tidy Data & Data Manipulation with the tidyverse

*OCRUG Hackathon – April 10, 2021*

# Goals for this module

- Understand what tidy data is and why it's important for data analysis
- Learn about the tidyverse and why it’s a great ecosystem for working with data in R
- Learn about data manipulation operations using dplyr

# Tidy Data

# 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 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

| make_model | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21 | 6 | 160 | 110 | 3.9 | 2.62 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21 | 6 | 160 | 110 | 3.9 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.32 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.44 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.46 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360 | 245 | 3.21 | 3.57 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.19 | 20 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.15 | 22.9 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.44 | 18.3 | 1 | 0 | 4 | 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!*

# Going from messy to tidy data takes work!

- The "dirty" secret of being a data scientist:
  *a non-trivial amount of time is spent cleaning up data*

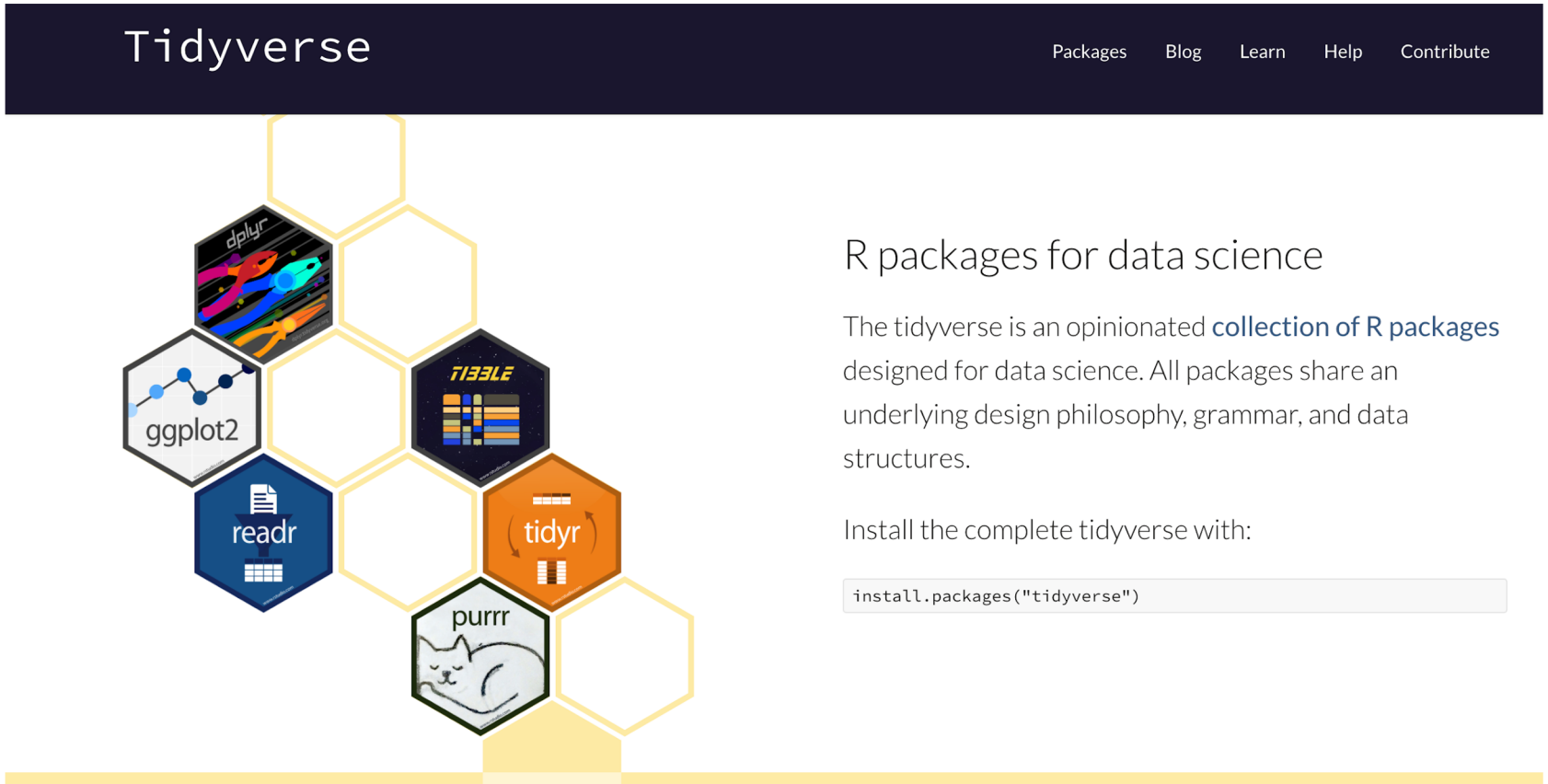
- Sometimes messy data can be cleaned-up by hand, but this is not optimal: time consuming, error-prone, tedious

- Successful data scientists assemble a "toolkit" of methods/techniques for manipulating data using code

- In R, the tidyr package provides useful functions for to help make messy data tidy

# The Tidyverse

## An opinionated collection of R packages for data science



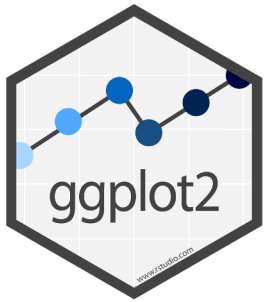

https://www.tidyverse.org

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

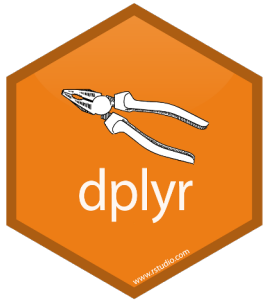
That's the tidyverse!

# The tidyverse covers the fundamental components of the data analysis workflow

## Core tidyverse Packages




Data Visualization




Fundamental Data Manipulation & Pipelines


Read Data
Tidy Data

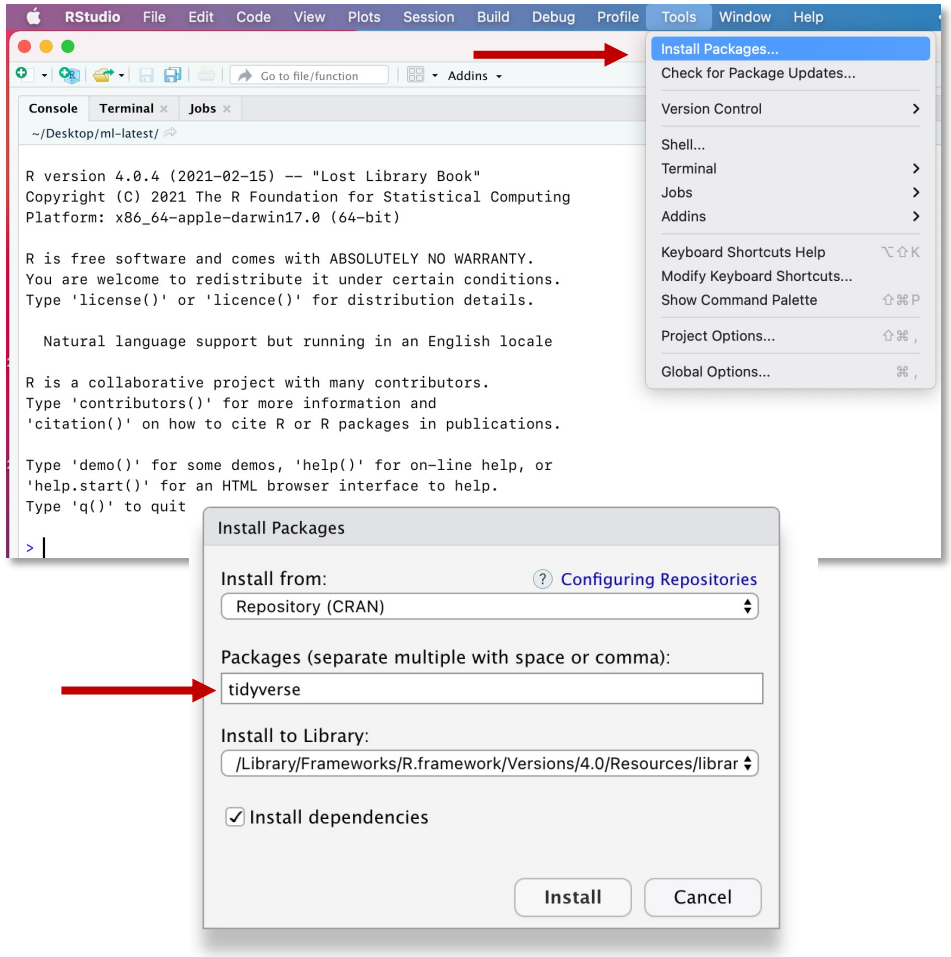

Manipulate Specific Types of Data

https://www.tidyverse.org

*And LOTS of other packages developed by others...*

# Installing the core tidyverse packages is easy

*From RStudio*




*From the R Console*

```
install.packages("tidyverse")
```

*The tidyverse package is just a wrapper*

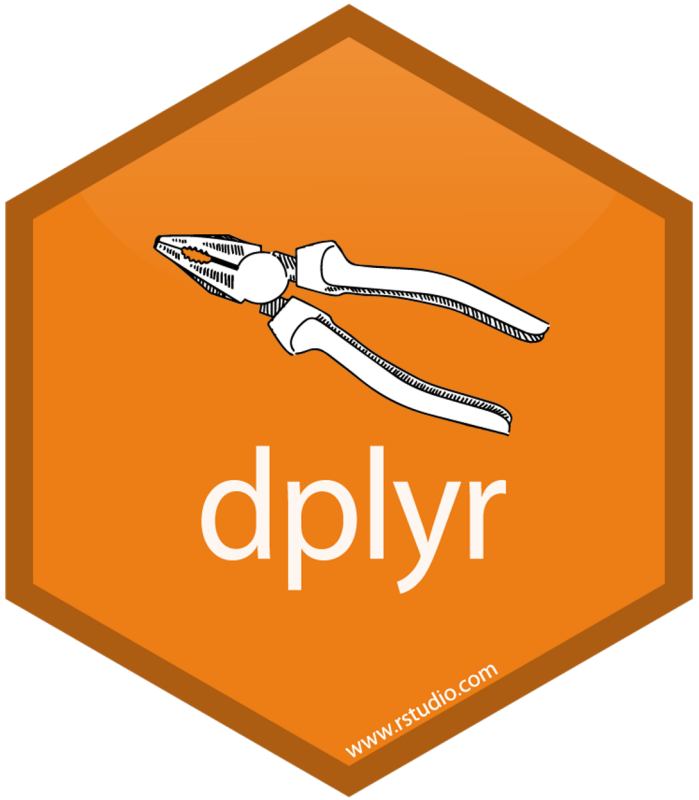
*It will install the core tidyverse packages for you*

# 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

# `dplyr` is a tidyverse R package for data manipulation

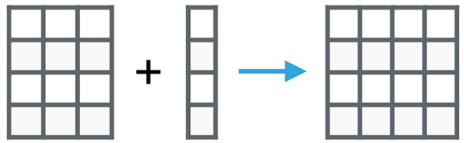
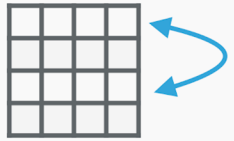
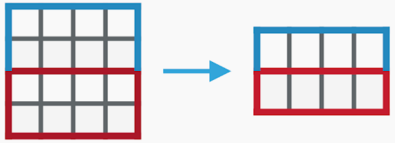



- Defines the fundamental operations that encompass most data analysis tasks
- Assumes you already have tidy data
- Uses a pipeline coding structure to perform complex operations a straight-forward, natural way

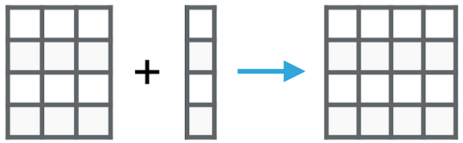
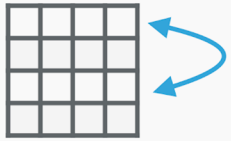
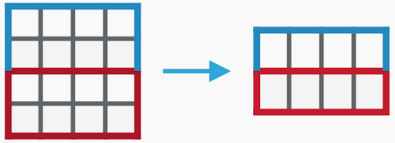
# `dplyr` defines data manipulation *verbs*

- 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` |

# We'll use this table for the following examples

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21 | 6 | 160 | 110 | 3.9 | 2.62 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21 | 6 | 160 | 110 | 3.9 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.32 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.44 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.46 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360 | 245 | 3.21 | 3.57 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.19 | 20 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.15 | 22.9 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.44 | 18.3 | 1 | 0 | 4 | 4 |

… many more rows

This is the mtcars data set, part of the standard R installation

You can access the data table from the `mtcars` variable

# 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(mtcars, …)` – *adds a column to* `mtcars`

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

- The … above are additional arguments that further define what the operation is going to do

- dplyr verbs always give back a data frame

  `mtcars2 <- mutate(mtcars, …)`

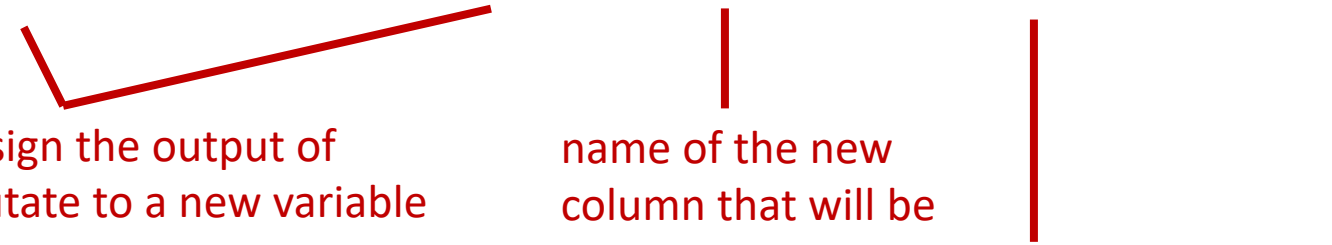
  `mtcars2` *is a data frame that has a new column*

# 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)


# add a new column that is the quarter-mile time in minutes
mtcars2 <- mutate(mtcars, qmin = qsec / 60)
```



Assign the output of mutate to a new variable

It's also common to reassign to the same variable

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 names

# dplyr verb examples: select

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

```
# get only the miles per gallon and horsepower columns
mtcars2 <- select(mtcars, mpg, hp)
```

input table always comes first

the subsequent arguments specify the columns to keep

```
# remove the weight column
mtcars2 <- select(mtcars, -wt)
```

This will give you all the columns in mtcars *except* for wt

# dplyr verb examples: filter

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

```
# only get the cars with 4 cylinders
mtcars2 <- filter(mtcars, cyl == 4)
```

Use conditional expressions to specify which rows you want to keep

```
# only get the cars with horsepower above 200
mtcars2 <- filter(mtcars, hp > 200)

# only get the cars with 8 cylinders AND horsepower above 200
mtcars2 <- filter(mtcars, cyl == 8, hp > 200)
```

Multiple expression are combined with the AND operator (use | for or)

# dplyr verb examples: arrange

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

```
# sort the rows by miles/gallon, from low to high
mtcars2 <- arrange(mtcars, mpg)

# use desc to reverse the sorting (high to low)
mtcars2 <- arrange(mtcars, desc(mpg))
```

sorting will occur using R's sorting rules
depending on the data type being used

# dplyr verb examples: group_by + summarize

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

```
# compute the average miles/gallon for each cylinder type (count)
mtcars_grouped <- group_by(mtcars, cyl)

summarize(mtcars_grouped, mean_mpg = mean(mpg))
```

makes 3 groups, one for each cylinder type (4, 6, 8)

for each individual group…

compute a new column…

that's the mean mpg for each cylinder group

How many rows will this expression return?

# 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
- Yes -- it looks strange
- Think of %>% as a pipe, funneling the input data into the function to do some work

# 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(mtcars, cyl == 4)
```

R Code *with* pipes

```
mtcars %>% filter(cyl == 4)
```

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

# Practice with pipes

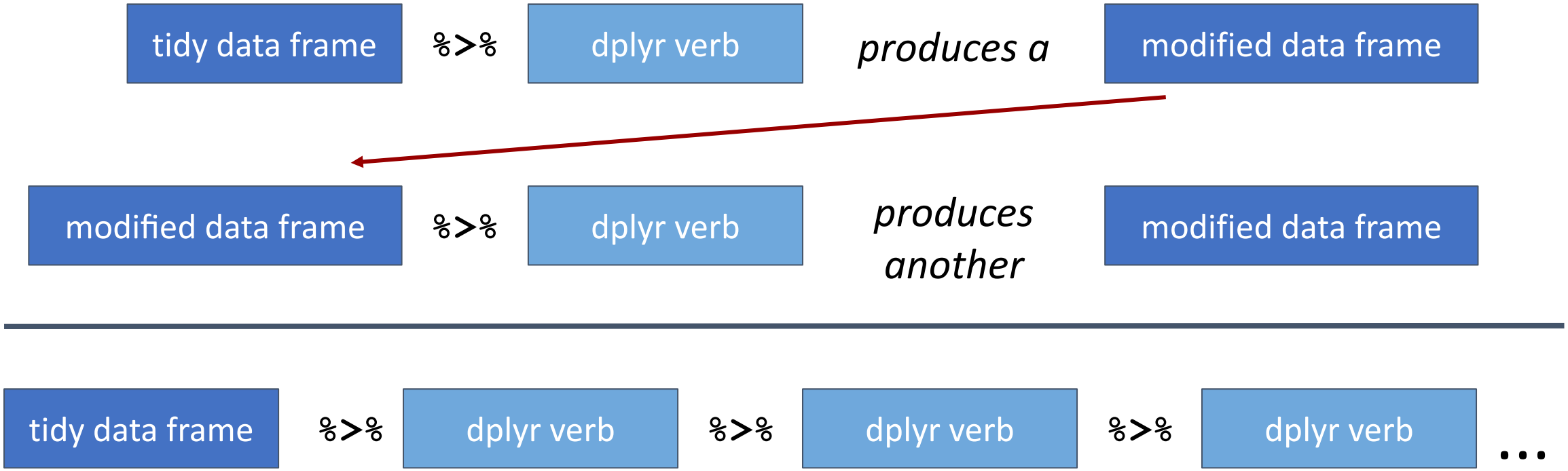
```
filter(mtcars, cyl == 4)
```

```
mtcars %>% filter(cyl == 4)
```

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, %>%

# 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 with the mtcars data set:

1. remove cars with 6 cylinders, then…
2. compute the mean horsepower for cars group by the number of gears they have, then…
3. order the results by descending mean horsepower

```
# accomplish the above steps in a dplyr pipeline
mtcars %>%
  filter(cyl != 6) %>%
  group_by(gear) %>%
  summarize(mean_hp = mean(hp)) %>%
  arrange(desc(mean_hp))
```

Using the mtcars data table…
filter out cars with 6 cylinders…
then group by the number of gears…
and compute the mean horsepower…
then order the results by descending mean horsepower.

The code reads very similar to how you'd describe the process in words!

*Note: 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

# Resources

- Paper on tidy data by H. Wickham
  https://vita.had.co.nz/papers/tidy-data.pdf
- R For Data Science, section on tidy data
  https://r4ds.had.co.nz/tidy-data.html
- tidyr package website
  https://tidyr.tidyverse.org
- Data importing cheat sheet
  https://github.com/rstudio/cheatsheets/blob/master/data-import.pdf