populationdata BC

Data Management in R

Session 1

July 14 th, 2020

@EcoLaurenY

lauren@mapdatascience.com

Lauren Yee is multifaceted researcher and data scientist. She currently works in the consulting industry and designs projects around data visualization, dashboards and ecological, spatial analysis, studied emerging infectious zoonoses and spatial epidemiology.

Ann Greenwood is the Education and Training Lead for Population Data BC. She manages a variety of education and training services for researchers and population health professionals . Services include an online certificate course in Population Health Data Analysis, live webinar-based courses, free online self -paced workshops, colloquium presentations/webinars and in-person workshops.

Data Management

Data in the real world

- Often not what you expect

- There can be significant time delays to getting the data you need that cut into analysis time

- Staff turn-over from the data providers can make it difficult to get questions answered

- Can be hard to access or gain access to data that exists and you need

- Often little to no documentation about what variables mean, how it was collected, or what may have changed during the data collection period

Data documentation is important for assessing data quality and reliability.

Ideally the documentation will: provide concise and clear explanation of the datasets, including variable definitions and value codes; review survey instruments or collection forms and collection protocols

Once you have reviewed data documentation, plan to reach out to several people in different departments with questions and for clarifications. Administrative data may come from large and complex systems, and often no single person understands them all. Even when a description seems clear to you, it may mean something entirely different to the person who wrote it.

https://www.poverty-action.org/sites/default/files/publications/Goldilocks-Deep-Dive-Using-Administrative-Data-for-Monitoring-and-Evaluation_0.pdf

https://www150.statcan.gc.ca/n1/pub/12-539-x/2009001/administrative-administratives-eng.htm

https://www.statcan.gc.ca/eng/about/admin_data_faq

Administrative Data

Data that is collected or created by an agency (government, private sector) for their own purposes, typically used to track a program's implementation, project activities and expenses, indicators for a program outcome or to comply with government reporting regulations.

Examples include: hospital records of patient visits and health outcomes, drug benefit claims, surveillance programs,educational records, financial institutions

Benefits:

- lower the collection costs relative to administering your own survey and supply you with good data
- large collection of longitudinal data

Limitations :

- possible lack of accuracy
- different coding criteria across individuals and institutions
- changing criteria over time
- changing in coding system over time
- difficulties in linkage and merging of different databases
- difficult to obtain or practice on

Administrative Data

Consider asking the data producer about how the data was collected (data entry and standards), and if it is suitable for your analysis/hypothesis.

Why the data was created in the first place (legislation, objectives, needs of the organization, end-use of the data) ?

Consider the time period for any subject being studied (e.g., a person, a family, a child care program) is limited by the period of time that the subject is using the service for which the data are being collected

Can the data be supplemented ?

Health care administrative data can be supplemented through linkages with other data sources such as census data to estimate neighbourhood income, clinical registries, electronic medical records, citizenship data or survey data to add to patient perspectives and lifestyle variables.

https://bmchealthservres.biomedcentral.com/articles/10.1186/s12913-016-1489-0

Making your R work organized and portable

One day you will need to quit R, go do something else and return to your analysis later.

One day you will have multiple analyses going that use R and you want to keep them separate.

Perhaps there is a global pandemic and you have to switch workstations to a home office

Use R to organize all your projects, data, data cleaning scripts, visualization scripts, model scripts, image and data outputs.

Pseudocode

1. The process of writing out "what you want to do" in common terms
2. Then adapting your written words into R code

Load Data into R

Run summary statistics

Remove NA values

Plot Values

This allows you to form a roadmap for your analysis and troubleshoot along the way

Helps with mental fatigue when troubleshooting for hours

What was I doing? What was I trying to do? What is the next step?

Refer back to your pseudocode!

Workflow

Workflow

1. Examine the data, data dictionaries, documentation and realistically...long long email threads about the data
2. Set up your project in R
3. Use R Markdown as a living document or data dictionary
4. Exploratory Data Analysis
5. Generate data cleaning scripts and basic statistics (does this align with expected results from step 1?)
6. Separate clean data, reduced data into separate folder structures
7. Repeat steps above as necessary
8. Functionalize any repeating code or handy scripts you will use often, create loops for automated data processing
9. Generate final reports, visualizations as R Markdown to be shared

Before working with data...

Before opening starting an analysis...

- Ask all the questions about that data!
- Read the documentation
- Make note of contacts for the data
- Make a plan and approach for analysis, ask for feedback from peers or superiors
- Expect to rewrite, abandon or rework your R code many times
- Data management is easier in R with new technologies that are being developed every day...

Data Quality

Never assume you have good data

What Data quality controls were employed during data collection?

Specific data quality standards exist see the International Organization for Standardization (ISO):
https://www.iso.org/obp/ui/#iso:std:iso:8000:-61:ed-1:v1:en

....but are often not universally followed by data producers

Prepare to:

Check for completeness of data, duplicates, duplicate unique ids

Maximum, minimum values

Logic checks (Is age negative or over 100?)

Recode variables

Are dollar values in the current year?

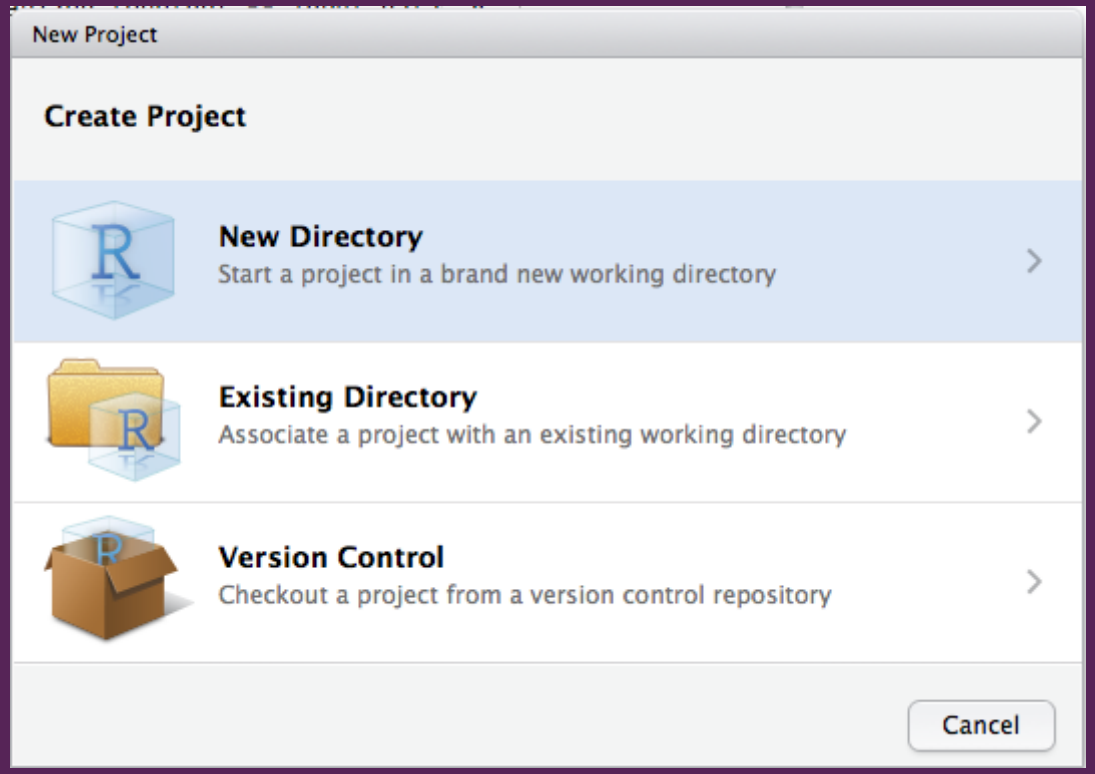
Develop code to reuse in your analysis to check for data quality

This will save you time, money and frustration

Do you use `getwd()` and `setwd()`?

Don't! there are better ways

Setting the directory limits reproducibility



Use Projects!

R uses a "working directory" which is where R will first look for files that you want it to load and where it will save. A recommended structure for R projects is as follows:

- Set up a directory for the project/analysis. Inside this folder create a "Workspace".
- Inside your workspace create two folders, Data and Outputs. Data for where you will be loading raw data from and outputs for saving.

A great tool in R is to "Create a New Project", which will then be mapped to your "Workspace". This is similar to use `setwd()` in R, however all projects by default are mapped to the folder it is saved in.

- All references can be relative`"./Data/"` rather than absolute paths `"C:/Desktop/Projects/ABC/Data/"`

See also: https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

Paths

Paths and directories can differ between Mac/Linux and Windows. There are three chief ways in which they differ:

The most important difference is how you separate the components of the path. Mac and Linux uses slashes (e.g. plots/diamonds.pdf) and Windows uses backslashes (e.g. plots\diamonds.pdf). R can work with either type (no matter what platform you're currently using)

Absolute paths (i.e. paths that point to the same place regardless of your working directory) look different. In Windows they start with a drive letter (e.g. C:) or two backslashes (e.g. `\\servername`) and in Mac/Linux they start with a slash "/" (e.g. /users/hadley).

You should never use absolute paths in your scripts, because they hinder sharing e.g. the user hadley above may be your username

The place that ~ points to. ~ is a convenient shortcut to your home directory. Windows doesn't really a home directory, so it points to your documents directory.

Informative file names

Object names must start with a letter, and can only contain letters, numbers, .codeblock_ and .

You want your object names to be descriptive, so you'll need a convention for multiple words. I recommend snake*case where you separate lowercase words with* `

```
i_use_snake_case

otherPeopleUseCamelCase

some.people.use.periods

And_aFew.People_RENOUNCEconvention
```

RStudio projects give you a solid workflow that will serve you well in the future:

Create an RStudio project for each data analysis project.

Keep data files there; we'll talk about loading them into R in data import.

Keep scripts there; edit them, run them in bits or as a whole.

Save your outputs (plots and cleaned data) there.

Only ever use relative paths, not absolute paths.

Everything you need is in one place, and cleanly separated from all the other projects that you are working on.

R Markdown

- *markdown* is a simplified language for text output that is commonly used to convert text to the HTML language that is meant to be easily understood.
- R Markdown is an authoring format that combines the *markdown* syntax with embedded R code chunks
- allows you to combine your notes, narrative, methods, code and results in one file
- fully reproducible with many output formats: PDF, word, html, dashboards and slideshows
- can use multiple programming languages in one document such as: R, Python, and SQL

R Markdown

Designed to be used in three ways:

1. Communication to decision makers

- High level conclusions and visualizations

2. Collaboration with teams

- Including code, methods and approach

3. Environment to **do** data analysis

- A modern day lab notebook including what you did, your code, and why you did it that way

Use a consistent style

Do not repeat
automate, functions,
loops

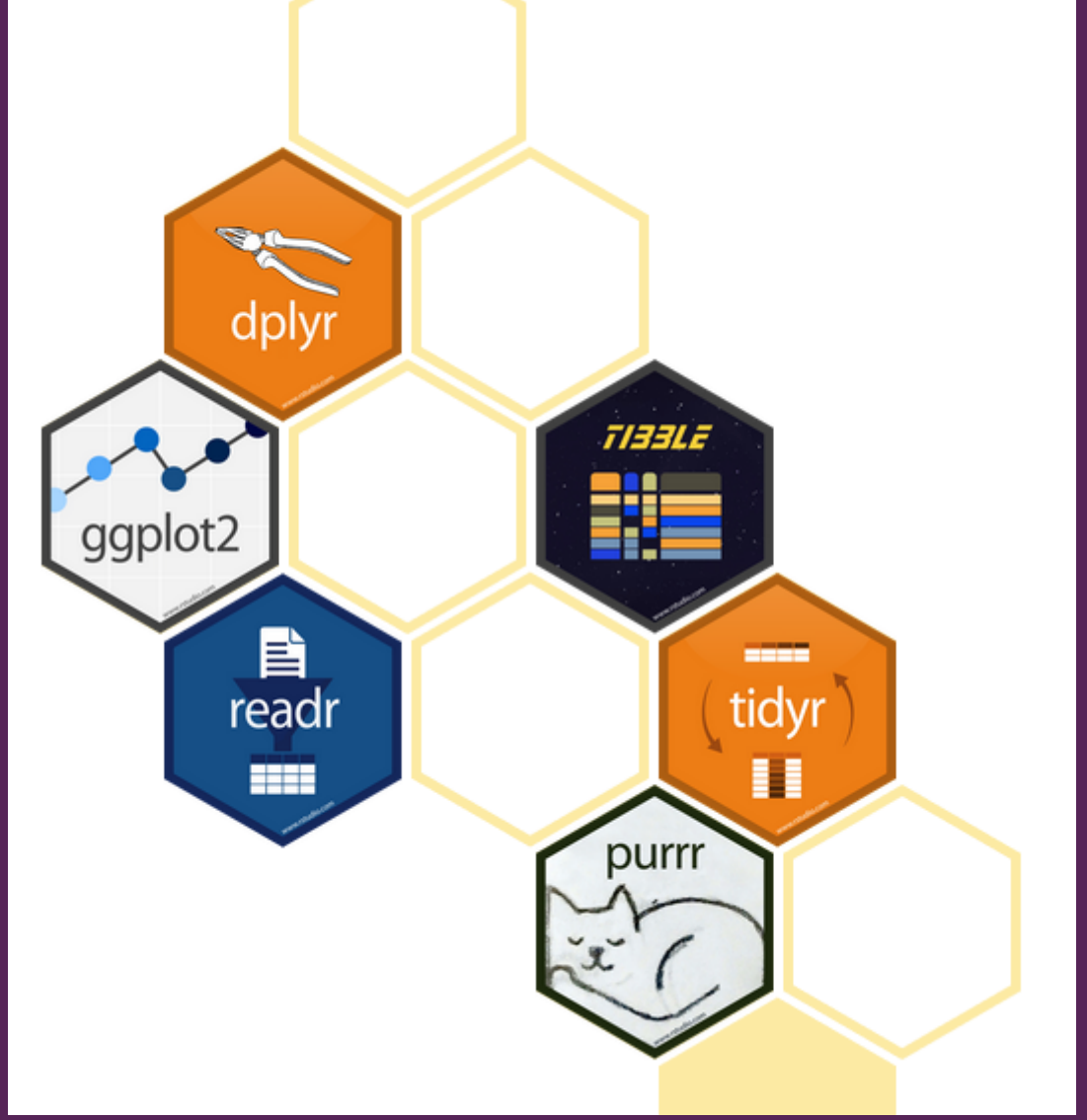
Use relative paths

Tidyverse

What is the tidyverse?

A collection of R packages that make it easier to work with data.

- `dplyr` is used to manipulate, group, and summarize data
- `ggplot2` is a data visualization library that we will be using in the course!
- `readr` loads data into R
- `tibble` data format, similar to data frames
- `purrr` functions which replace for loops with code that is both more succinct and easier to read
- `tidyr` is a library meant for formatting and shaping data into the 'tidy' format...



Tidy Data

It is often said that 80% of data analysis is spent on the cleaning and preparing data.

Data Tidying : structuring datasets to facilitate analysis.

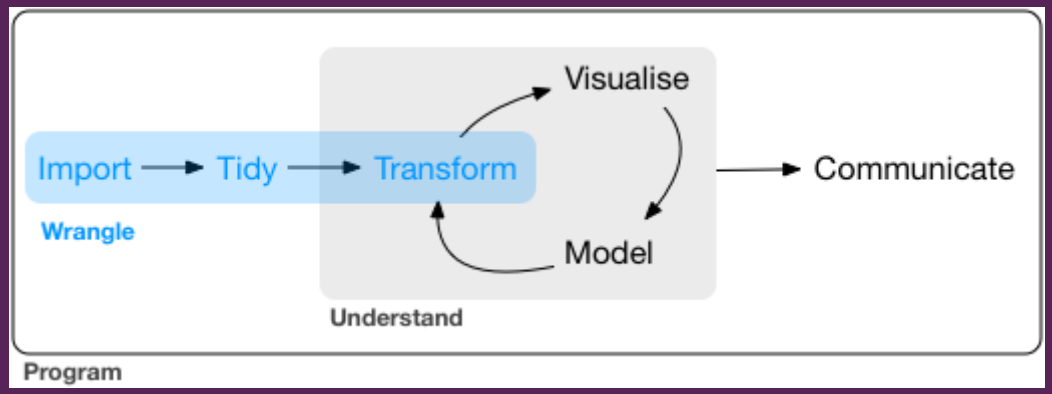
The goal of tidyr is to help you create tidy data. Tidy data is data where:

- Every column is variable
- Every row is an observation
- Every cell is a single value

Tidy data describes a standard way of storing data that is used wherever possible throughout the tidyverse. It is an attempt to standardize data.

All `tidyverse` packages are designed to work with tidy data.

Workflow in the Tidyverse



Import the data into R using `readr`

'Tidy' the data using `dplyr` or `tidyr`

Transform the data for your analysis, recode variables using `dplyr`, `purrr`, `lubridate`,`stringr`

Visualize `ggplot2`

Communicate `RMarkdown`

Workflow in the Tidyverse

Readr

To accurately read a rectangular dataset with readr you combine two pieces: a function that parses the overall file, and a column specification. The column specification describes how each column should be converted from a character vector to the most appropriate data type, and in most cases it's not necessary because readr will guess it for you automatically.

readr supports seven file formats with seven `read_` functions:

- `read_csv()`: comma separated (CSV) files
- `read_tsv()`: tab separated files
- `read_delim()`: general delimited files
- `read_fwf()`: fixed width files
- `read_table()`: tabular files where columns are separated by white-space.
- `read_log()`: web log files

Other Types of Data

Try one of these packages to import other types of files:

haven - SPSS, Stata, and SAS Files **readxl** - Excel files (.xls and .xlsx) **DBI** - databases **jsonlite** - json

A Grammar of Data Manipulation

dplyr is based on the concepts of functions as verbs that manipulate data frames.

- `filter`: pick rows matching criteria
- `slice`: pick rows using index(es)
- `select`: pick columns by name
- `pull`: grab a column as a vector
- `arrange`: reorder rows
- `mutate`: add new variables
- `distinct`: filter for unique rows
- `sample_n` / `sample_frac`: randomly sample rows
- `summarise`: reduce variables to values
- ... (many more)

dplyr rules for functions

First argument is *always* a data frame/tibble

Subsequent arguments say what to do with that data frame

Always return a data frame

Pipes %>%

Easier to read structure

"pipe" the output of the previous line of code as the first input of the next line of code.

The + operator in **ggplot2** functions is used for "layering". This means you create the plot in layers, separated by +.

Example Data

Titanic Data Set

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	NA	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	NA	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	NA	S
6	0	3	Moran, Mr. James	male	NA	0	0	330877	8.4583	NA	Q

`filter` to select a subset of rows passengers who were 35 years old

```
titanic %>%
  filter(Age == 35)
```

```
## # A tibble: 23 x 12
##    PassengerId Survived Pclass Name
##          <dbl>    <dbl>  <dbl> <chr>
##  1           4        1      1 Futrelle, Mrs. Jacques He
##  2           5        0      3 Allen, Mr. William Henry
##  3          21        0      2 Fynney, Mr. Joseph J
##  4         212        1      2 Cameron, Miss. Clear Anni
##  5         231        1      1 Harris, Mrs. Henry Birkha
##  6         259        1      1 Ward, Miss. Anna
##  7         270        1      1 Bissette, Miss. Amelia
##  8         280        1      3 Abbott, Mrs. Stanton (Ros
##  9         364        0      3 Asim, Mr. Adola
## 10         384        1      1 Holverson, Mrs. Alexander
## # ... with 13 more rows
```

`filter` for many conditions at once Age is 35 and Sex is female

```
titanic %>%
  filter(Age == 35, Sex=="female")
```

```
## # A tibble: 11 x 12
##    PassengerId Survived Pclass Name
##          <dbl>    <dbl>  <dbl> <chr>
##  1           4        1      1 Futrelle, Mrs. Jacques Hea
##  2         212        1      2 Cameron, Miss. Clear Annie
##  3         231        1      1 Harris, Mrs. Henry Birkhar
##  4         259        1      1 Ward, Miss. Anna
##  5         270        1      1 Bissette, Miss. Amelia
##  6         280        1      3 Abbott, Mrs. Stanton (Rosa
##  7         384        1      1 Holverson, Mrs. Alexander
##  8         487        1      1 Hoyt, Mrs. Frederick Maxfi
##  9         966       NA      1 Geiger, Miss. Amalie
## 10        1014       NA      1 Schabert, Mrs. Paul (Emma
## 11        1098       NA      3 McGowan, Miss. Katherine
```

Logical operators in R

operator	definition	operator	definition
`<`	less than	`x | y`	`x` OR `y`
`<=`	less than or equal to	`is.na(x)`	test if `x` is `NA`
`>`	greater than	`!is.na(x)`	test if `x` is not `NA`
`>=`	greater than or equal to	`x %in% y`	test if `x` is in `y`
`==`	exactly equal to	`!(x %in% y)`	test if `x` is not in `y`
`!=`	not equal to	`!x`	not `x`
`x & y`	`x` AND `y`		

`select` to keep variables

```
titanic %>%
  filter(Age == 35, Sex=="female")%>%
  select(Name, Sex, Age, Fare)
```

```
## # A tibble: 11 x 4
##    Name                                                Sex      Age   Fare
##    <chr>                                               <chr>  <dbl>  <dbl>
##  1 Futrelle, Mrs. Jacques Heath (Lily May Peel)        female    35  53.1
##  2 Cameron, Miss. Clear Annie                          female    35  21
##  3 Harris, Mrs. Henry Birkhardt (Irene Wallach)        female    35  83.5
##  4 Ward, Miss. Anna                                    female    35 512.
##  5 Bissette, Miss. Amelia                              female    35 136.
##  6 Abbott, Mrs. Stanton (Rosa Hunt)                    female    35  20.2
##  7 Holverson, Mrs. Alexander Oskar (Mary Aline Towner) female    35  52
##  8 Hoyt, Mrs. Frederick Maxfield (Jane Anne Forby)     female    35  90
##  9 Geiger, Miss. Amalie                                female    35 212.
## 10 Schabert, Mrs. Paul (Emma Mock)                     female    35  57.8
## 11 McGowan, Miss. Katherine                            female    35   7.75
```

`select` to exclude variables

```
titanic %>%
  select(-Embarked)
```

```
## # A tibble: 1,309 x 11
##    PassengerId Survived Pclass Name                                          Sex      Age SibSp Parch Ti
##          <dbl>    <dbl>  <dbl> <chr>                                         <chr>  <dbl> <dbl> <dbl> <c
##  1           1        0      3 Braund, Mr. Owen Harris                       male      22     1     0 A/
##  2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female    38     1     0 PC
##  3           3        1      3 Heikkinen, Miss. Laina                        female    26     0     0 ST
##  4           4        1      1 Futrelle, Mrs. Jacques Heath (Lily May Peel)  female    35     1     0 11
##  5           5        0      3 Allen, Mr. William Henry                      male      35     0     0 37
##  6           6        0      3 Moran, Mr. James                              male      NA     0     0 33
##  7           7        0      1 McCarthy, Mr. Timothy J                       male      54     0     0 17
##  8           8        0      3 Palsson, Master. Gosta Leonard                male       2     3     1 34
##  9           9        1      3 Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) female    27     0     2 34
## 10          10        1      2 Nasser, Mrs. Nicholas (Adele Achem)           female    14     1     0 23
## # ... with 1,299 more rows
```

`select` a range of variables

```
titanic %>%
  select(PassengerId:Name)
```

```
## # A tibble: 1,309 x 4
##    PassengerId Survived Pclass Name
##          <dbl>    <dbl>  <dbl> <chr>
##  1           1        0      3 Braund, Mr. Owen Harris
##  2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer)
##  3           3        1      3 Heikkinen, Miss. Laina
##  4           4        1      1 Futrelle, Mrs. Jacques Heath (Lily May Peel)
##  5           5        0      3 Allen, Mr. William Henry
##  6           6        0      3 Moran, Mr. James
##  7           7        0      1 McCarthy, Mr. Timothy J
##  8           8        0      3 Palsson, Master. Gosta Leonard
##  9           9        1      3 Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
## 10          10        1      2 Nasser, Mrs. Nicholas (Adele Achem)
## # ... with 1,299 more rows
```

`slice` for certain row numbers

First five

```
titanic %>%
  slice(1:5)
```

```
## # A tibble: 5 x 12
##   PassengerId Survived Pclass Name                                            Sex      Age SibSp Parch Tic
##         <dbl>    <dbl>  <dbl> <chr>                                           <chr>  <dbl> <dbl> <dbl> <ch
## 1           1        0      3 Braund, Mr. Owen Harris                         male      22     1     0 A/5
## 2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female    38     1     0 PC
## 3           3        1      3 Heikkinen, Miss. Laina                          female    26     0     0 STO
## 4           4        1      1 Futrelle, Mrs. Jacques Heath (Lily May Peel)    female    35     1     0 113
## 5           5        0      3 Allen, Mr. William Henry                        male      35     0     0 373
```

`slice` for certain row numbers

Last five

```
last_row <- nrow(titanic)
titanic %>%
  slice((last_row - 4):last_row)
```

```
## # A tibble: 5 x 12
##   PassengerId Survived Pclass Name                      Sex      Age SibSp Parch Ticket              Fare
##         <dbl>    <dbl>  <dbl> <chr>                     <chr>  <dbl> <dbl> <dbl> <chr>               <dbl>
## 1        1305       NA      3 Spector, Mr. Woolf        male    NA       0     0 A.5. 3236           8.05
## 2        1306       NA      1 Oliva y Ocana, Dona. Fer… female  39       0     0 PC 17758          109.
## 3        1307       NA      3 Saether, Mr. Simon Sive… male    38.5     0     0 SOTON/O.Q. 3101262  7.25
## 4        1308       NA      3 Ware, Mr. Frederick       male    NA       0     0 359309              8.05
## 5        1309       NA      3 Peter, Master. Michael J  male    NA       1     1 2668               22.4
```

`pull` to extract a column as a vector

```
titanic %>%
  slice(1:6) %>%
  pull(Fare)
```

```
## [1]  7.2500 71.2833  7.9250 53.1000  8.0500  8.4583
```

vs.

```
titanic %>%
  slice(1:6) %>%
  select(Fare)
```

```
## # A tibble: 6 x 1
##    Fare
##   <dbl>
## 1  7.25
## 2 71.3
## 3  7.92
## 4 53.1
## 5  8.05
```

`sample_n` / `sample_frac` for a random sample

- `sample_n`: randomly sample 5 observations

```
titanic_n5 <- titanic %>%
  sample_n(5, replace = FALSE)
dim(titanic_n5)
```

```
## [1]  5 12
```

- `sample_frac`: randomly sample 20% of observations

```
titanic_perc20 <-titanic %>%
  sample_frac(0.2, replace = FALSE)
dim(titanic_perc20)
```

```
## [1] 262  12
```

`distinct` to filter for unique rows

And `arrange` to order alphabetically

```
titanic %>%
  select(Pclass, Fare) %>%
  distinct() %>%
  arrange(Fare, Pclass)
```

```
## # A tibble: 289 x 2
##    Pclass  Fare
##     <dbl> <dbl>
##  1      1  0
##  2      2  0
##  3      3  0
##  4      3  3.17
##  5      3  4.01
##  6      1  5
##  7      3  6.24
##  8      3  6.44
##  9      3  6.45
## 10      3  6.50
## # ... with 279 more rows
```

`summarise` to reduce variables to values

```
titanic %>%
  summarise(avg_fare = mean(Fare,na.rm=T))
```

```
## # A tibble: 1 x 1
##   avg_fare
##      <dbl>
## 1     33.3
```

`group_by` to do calculations on groups

```
titanic %>%
  group_by(Sex) %>%
  summarize(avg_fare = mean(Fare,na.rm=T))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   Sex    avg_fare
##   <chr>     <dbl>
## 1 female     46.2
## 2 male       26.2
```

`count` observations in groups

```
titanic %>%
  count(Sex)
```

```
## # A tibble: 2 x 2
##   Sex        n
##   <chr>  <int>
## 1 female   466
## 2 male     843
```

```
titanic %>%
  count(Survived)
```

```
## # A tibble: 3 x 2
##   Survived     n
##      <dbl> <int>
## 1        0   549
## 2        1   342
## 3       NA   418
```

References

Portions of this material are derived from:

RStudio's 'Learning Tidyverse'

Data Carpentry datasciencebox.org

Estrellado, R. A., Bovee, E. A., Motsipak, J., Rosenberg, J. M., & Velásquez, I. C. (in press). Data science in education using R. London, England: Routledge. Nb. All authors contributed equally

https://stat545.com/