An Introduction to the R Programming Language

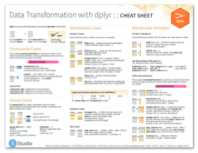
Lesson 1
Intro to R and R Studio

Course Introduction

Objectives

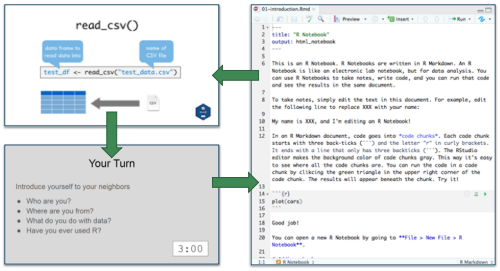
- Use R to perform analyses of laboratory operational data
- Adopt a 'tidy' approach to data analysis within the framework of R
- Leverage R as a means of improving reproducibility in your clinical data analysis

Course Materials



https://github.com/pcmathias/AACC-2019-Introduction-to-R

Lectures



Tips for Success

- **Remember that computers are not actually that smart:** R will do exactly what you tell it to do, even if that's not what you want! Your code can't have errors because R can't figure out what you mean.
- **Take the "copy, paste, and tweak" approach:** It is often much easier to taking existing code that you know works and modify it to suit your ends, rather than trying to write new code from scratch. Exercise ctrl-C/ctrl-V!

Tips for Success

- **The best way to learn to code is by doing:** Find a project, make a goal, and push yourself to use R!
- **Practice is key**!

Getting Oriented to R

Lesson Goals

1. Get oriented to R and RStudio
2. Learn some fundamentals of coding

Lesson Objectives

1. Log in and tour RStudio Cloud
2. Execute code at the console
3. Define and use functions
4. Define and create objects in the environment
5. Interact with a dataframe

What is R?

- R is a statistical programming language.
- Using R you can load, analyze, and visualize data.
- R also provides an environment in which we can conduct reproducible data analysis.
  - Documented
  - Revisable
  - Shareable

RStudio: The Portal to R

- RStudio is an integrated development environment (IDE)
- Using RStudio we can interact with the R programming language to:
  - Write and execute code interactively
  - View data
  - Debug and fix errors
  - Author our code

RStudio: In the Cloud… In Your Home

- RStudio Cloud: An online hosted version of RStudio that we will use for these course sessions
- RStudio Desktop: A locally installed version of RStudio that you will use when you get home to continue your learning

Note: Use Rstudio Cloud only for this course. Do not upload protected health information to the cloud!

Your Turn #1

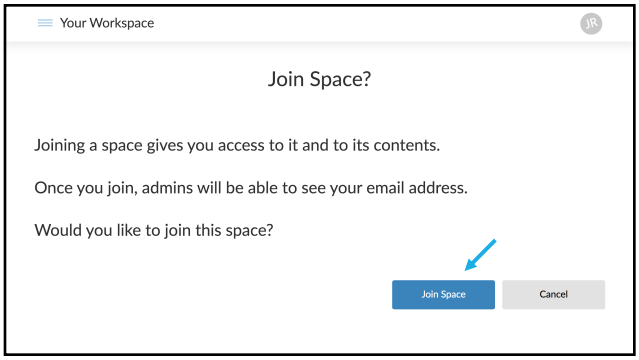
Navigate to: http://bit.ly/AACC-R-2019

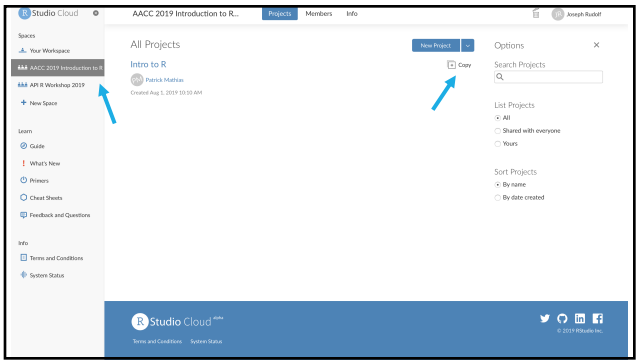
Enter your log in credentials

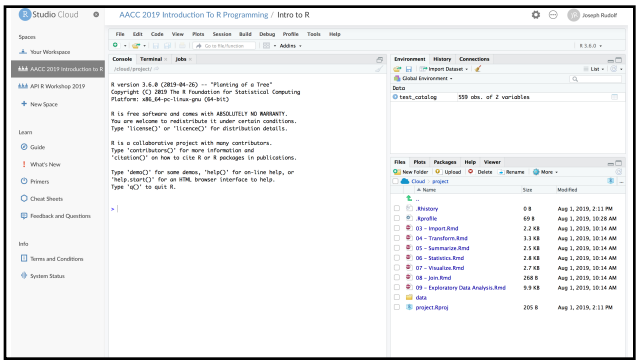
Join Space

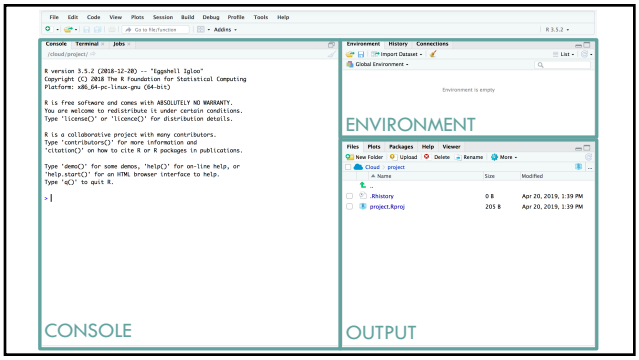
Make a copy of the Core Exercises for yourself

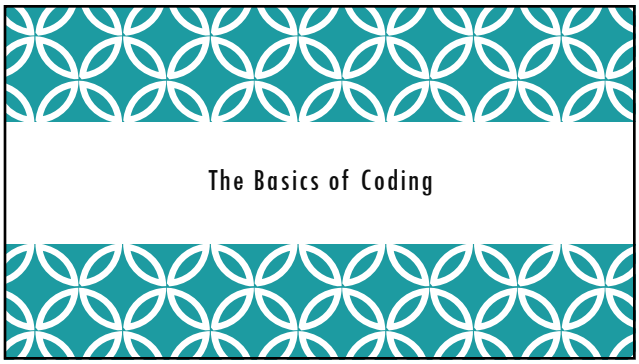
05:00





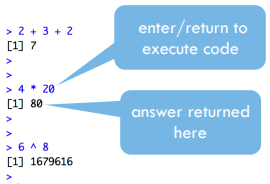

Studio Cloud
AACC 2019 Introduction To R Programming / Intro to R
Spaces
Your Workspace
AACC 2019 Introduction to R
API R Workshop 2019
New Space
Learn
Guide
What's New
Primers
Cheat Sheets
Feedback and Questions
Info
Terms and Conditions
System Status
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Jobs
/cloud/project/
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Environment History Connections
Import Dataset
Global Environment
Data
test_catalog 559 obs. of 2 variables
Files Plots Packages Help Viewer
New Folder Upload Delete Rename More
Cloud project
.Rhistory 0 B Aug 1, 2019, 2:11 PM
.Rprofile 69 B Aug 1, 2019, 10:28 AM
03 – Import.Rmd 2.2 KB Aug 1, 2019, 10:14 AM
04 – Transform.Rmd 3.3 KB Aug 1, 2019, 10:14 AM
05 – Summarize.Rmd 2.5 KB Aug 1, 2019, 10:14 AM
06 – Statistics.Rmd 2.8 KB Aug 1, 2019, 10:14 AM
07 – Visualize.Rmd 2.7 KB Aug 1, 2019, 10:14 AM
08 – Join.Rmd 268 B Aug 1, 2019, 10:14 AM
09 – Exploratory Data Analysis.Rmd 9.9 KB Aug 1, 2019, 10:14 AM
data
project.Rproj 205 B Aug 1, 2019, 2:11 PM


File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Jobs
/cloud/project/
R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
CONSOLE
Environment History Connections
Import Dataset
Global Environment
Environment is empty
ENVIRONMENT
Files Plots Packages Help Viewer
New Folder Upload Delete Rename More
Cloud project
.Rhistory 0 B Apr 20, 2019, 1:39 PM
project.Rproj 205 B Apr 20, 2019, 1:39 PM
OUTPUT


The Basics of Coding

The Basics of Coding: Calculation

- R is a calculator!



Your Turn #2

Place your cursor at the console and click to enter the console.

Complete the following calculation:
-For the date 12-29-1974
-Take the four digit year
-Subtract the month then multiply by the day

05:00

What did you get?

- A four digit number? A five digit number?

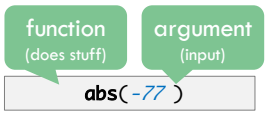
```
> 1974 - 12 * 29
[1] 1626
>
>
> (1974 - 12) * 29
[1] 56898
```

- Order of operations matters!

The Basics of Coding: Functions

- Code that extends our reach beyond the basic operators

```
> abs(-77)
[1] 77
>
```



- What if I don't know what a function does?

```
>
> ?abs()
>
```

Your Turn #3

Launch the help function for seq() using help (?)

Read about the seq() function in the output window

Can you describe what the seq() function does?

05:00

Putting Functions to Work

- We can use functions to do more than simple math, we can make things!

- We can create a series of integers (a vector) using the seq() function

```
>
> seq(from=5, to=150, by=10)
 [1]   5  15  25  35  45  55  65  75  85  95 105 115 125 135 145
```

The Basics of Coding: Objects

- Objects are the vessel for your output

object (stores output) function (does stuff) arguments (input)

```
sequence_of_10s <- seq( from=5, to=150, by=10 )
```

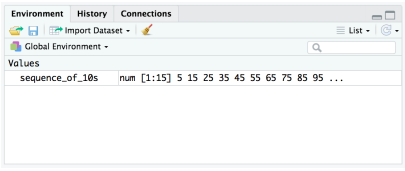
Checking the Contents of an Object

- Entering the object name at the console allows us to output the contents of an object.

```
>
> sequence_of_10s
 [1]   5  15  25  35  45  55  65  75  85  95 105 115 125 135 145
```

Checking the contents of an object

- The environment tab shows us the objects we have created.



Bending objects to your will

- Once we have created an object we can start to interact with it.
- This includes passing our objects to other functions… Whoa!

```
>
> min(sequence_of_10s)
[1] 5
>
> max(sequence_of_10s)
[1] 145
>
```

Your Turn #4

Generate a sequence, store it to an object, and ply your object*

-use the seq() function to create a vector from 0 to 500 by 25.

-assign your sequence to an object called sequence_of_25s

-check the contents of your object (at the console and in the environment tab)

-use the **median()** function to find the median value of your vector

*Use the help (?) feature or your neighbor if you get stuck

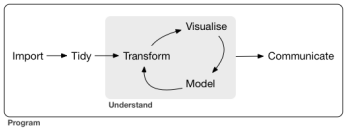
05:00

The Basics of Coding: Packages

- A package is a collection of functions.
- Packages extend the capabilities of the base R programming language.
- The **tidyverse** includes functions for reading data into the R environment, cleaning and manipulating data, and plotting our results.

Getting Tidy in the Verse

- Being "Tidy" with your data is important!



Installing and Loading Packages

- Installing a package

function (does stuff)

arguments (input)

```
install.packages("tidyverse", dependencies = TRUE)
```

- Loading into your environment

```
library(tidyverse)
```

Your Turn #5

Install a package and load it into the environment

-Leverage the **install.packages()** function to install a package called **"janitor"**

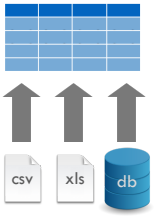
-Load the package into your environment using the **library()** function.

05:00

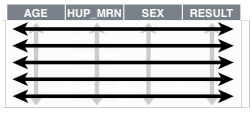
Working with Dataframes (aka Useful Data)

Dataframes: Beyond the Vector

- Dataframe is the term for a table
- Dataframes are composed:
  Columns (Variables)
  Rows (Observations)
- Dataframes are objects and can be acted on like other objects



What is a "Tidy" Dataframe?



A data set is **tidy** if:

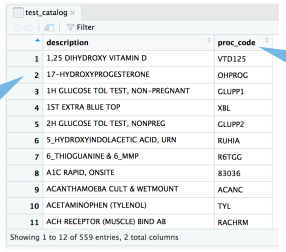
1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

Viewing the Contents of a Dataframe

- We have already loaded a data frame called **test_catalog.**

single click to explore the data

Viewing the Contents of a Dataframe



test_catalog

Filter

	description	proc_code
1	1,25 DIHYDROXY VITAMIN D	VTD125
2	17-HYDROXYPROGESTERONE	OHPROG
3	1H GLUCOSE TOL TEST, NON-PREGNANT	GLUPP1
4	1ST EXTRA BLUE TOP	XBL
5	2H GLUCOSE TOL TEST, NONPREG	GLUPP2
6	5_HYDROXYINDOLACETIC ACID, URN	RUHIA
7	6_THIOGUANINE & 6_MMP	R6TGG
8	A1C RAPID, ONSITE	83036
9	ACANTHAMOEBA CULT & WETMOUNT	ACANC
10	ACETAMINOPHEN (TYLENOL)	TYL
11	ACH RECEPTOR (MUSCLE) BIND AB	RACHRM

Showing 1 to 12 of 559 entries, 2 total columns

559 Observations (Rows)

2 Attributes (Columns)

Working with Dataframes at the Console

- The summary() function provides you with some abbreviated metadata about your dataframe.

summary(*object_name*)

```
>
> summary(test_catalog)
 description         proc_code
Length:559         Length:559
Class :character   Class :character
Mode  :character   Mode  :character
>
>
```

Number of Observations (Rows)

Data Type

Attributes (Columns)

Working with Dataframes at the Console

- The head() function is helpful for displaying a snippet of your dataframe

head(*object_name, n=number of rows to view*)

Sample of Data in Your Object

```
>
> head(test_catalog, n=5)
# A tibble: 5 x 2
  description                      proc_code
  <chr>                            <chr>
1 1,25 DIHYDROXY VITAMIN D         VTD125
2 17-HYDROXYPROGESTERONE           OHPROG
3 1H GLUCOSE TOL TEST, NON-PREGNANT GLUPP1
4 1ST EXTRA BLUE TOP               XBL
5 2H GLUCOSE TOL TEST, NONPREG     GLUPP2
>
```

Your Turn #6

Summarizing Dataframes and Understanding Object Contents

-Use the **summary()** function on our object **test_catalog**

-Leverage the **tail()** function to review the **last 10 rows** of the data frame.

-What is the 10th test from the end of the table?

05:00

Taking the Leap to Visualization

Even Simple Dataframes Can Be Interesting

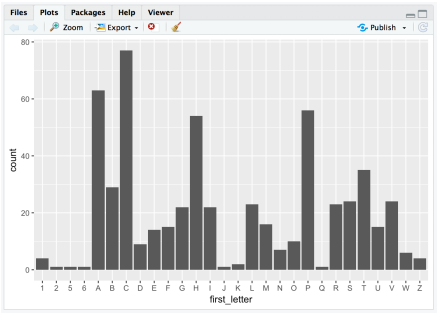
- With a few lines of code we can turn our test catalog…

Identify First Letter in the Test Name

```
> alphabet_analysis <- test_catalog %>%
+     mutate(first_letter = str_sub(description, 1, 1))
>
> ggplot(data = alphabet_analysis) +
+     geom_histogram(mapping = aes(x = first_letter), stat="count")
```

Make a histogram of the Data

- Into an insightful visualization



Console… I only just knew you

- As the complexity of our coding increases, the ability to iterate our lines of code as we write becomes apparent
- We find it increasing difficult to manage the names and states of objects in our environment
- And the console shows us its limitations

A Hero is Born

An Introduction to the R Programming Language

Lesson 2
Reproducible Reporting

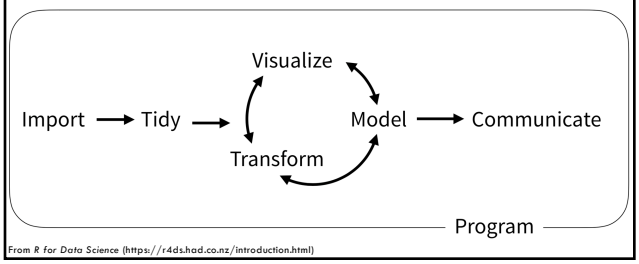
Lesson Goals

1. Practice working with the different components of a R Markdown file

Lesson Objectives

1. Understand why reproducible reporting is important
2. Create an R Markdown document and generate different types of output files
3. Practice modifying each component of a R Markdown file

Typical Data Science Pipeline



From *R for Data Science* (https://r4ds.had.co.nz/introduction.html)

Your Turn #1

1. Break into groups of 3-4
2. List 3 benefits of doing data analysis using spreadsheet software like Microsoft Excel
3. List 3 drawbacks/problems with doing data analysis using spreadsheet software like Microsoft Excel

05:00

Why is reproducibility important?

Replication vs Reproduction

- Replication: other people collect new data
  - Scientific gold standard
  - Difficult and time-consuming
- Reproduction: other people analyze the same data
  - Does not by itself validate the analysis …
  - Has been proposed as a minimal standard

The Duke Cancer Scandal

- Chemo sensitivity from microarrays
- Errors first, then misconduct
- Clinical trials based on flawed models
- Papers retracted, lawsuits settled

"Common errors are simple, Simple errors are common"

Theirs	Ours
"1881_at"	"1882_g_at"
"31321_at"	"31322_at"
"31725_s_at"	"31726_at"
"32307_r_at"	"32308_r_at"
...	

https://youtu.be/tN31vZi0LZY

Point-and-Click Is Not Reproducible

- Interactive tools do not record user actions
- Manual documentation is error-prone
- Manual analyses cannot be repeated on new data sets or shared with collaborators

Computer code can precisely document each step of the analysis

Why YOU Should Do Data Analysis Reproducibly

"Can we redo the analysis with this month's data?"

"Why do the data in Table 1 not seem to agree with Figure 2?"

"Why did I decide to omit these six samples?"

YOUR CLOSEST COLLABORATOR IS **YOU** FROM 6 MONTHS AGO
(BUT YOU DON'T ANSWER E-MAILS)

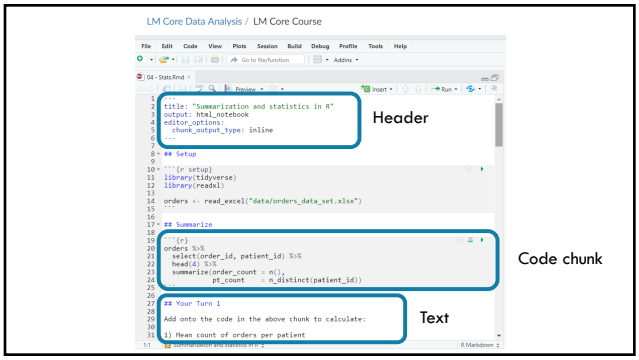
Using R for Reproducibility

Programming in R (or another language) allows one to reproduce analysis steps exactly or perform same analysis on new data

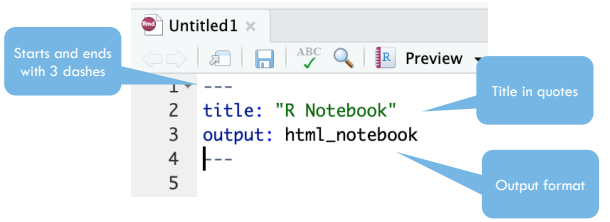
Better practice is to create documentation about analysis to accompany and explain code

Best practice is include documentation and code in one place

Using R Markdown to Support Reproducibility



Header



Text

Include hyperlinks with [text](link) format

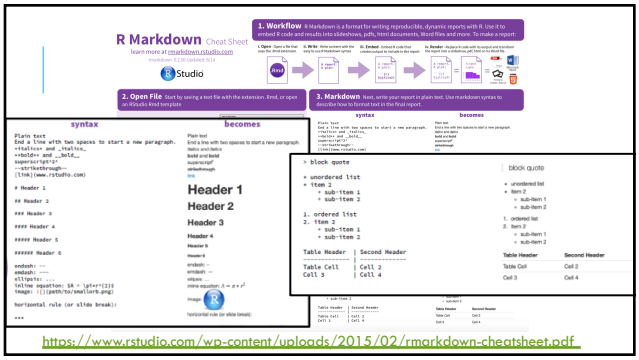
```
5
6  This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute
   code within the notebook, the results appear beneath the code.
7
8  Try executing this chunk by clicking the *Run* button within the chunk or by
   placing your cursor inside it and pressing *Cmd+Shift+Enter*.
9
```

1 asterisk for *italics* (*italics*)

2 asterisks for **bold** (**bold**)

Hyphens (-bullet 1) for bullet points

1 asterisk for *italics* (*italics*)



https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf

Code chunks

Open/close with 3 backticks

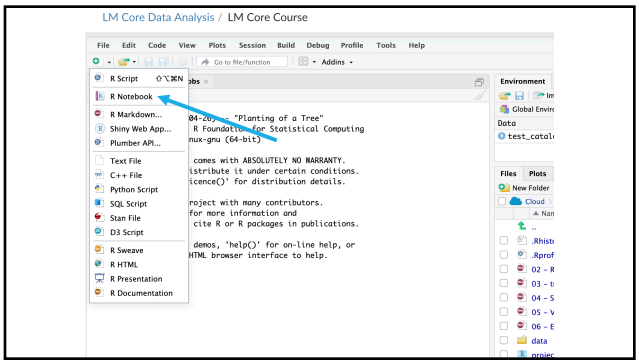
Language

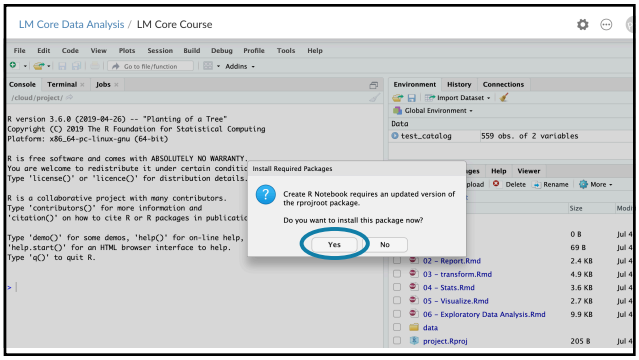
Chunk name

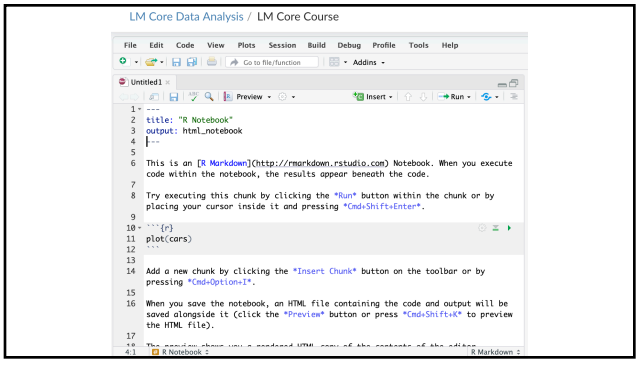
Run chunk

```
18 ```{r cars}
19 summary(cars)
20 ```
21
```

Code in body of chunk


LM Core Data Analysis / LM Core Course
File Edit Code View Plots Session Build Debug Profile Tools Help
R Script
R Notebook
R Markdown...
Shiny Web App...
Plumber API...
Text File
C++ File
Python Script
SQL Script
Stan File
D3 Script
R Sweave
R HTML
R Presentation
R Documentation
Environment
Files Plots


LM Core Data Analysis / LM Core Course
Console Terminal Jobs
Environment History Connections
Install Required Packages
Create R Notebook requires an updated version of the rprojroot package.
Do you want to install this package now?
Yes
No
test_catalog 559 obs. of 2 variables
02 – Report.Rmd
03 – transform.Rmd
04 – Stats.Rmd
05 – Visualize.Rmd
06 – Exploratory Data Analysis.Rmd
data
project.Rproj

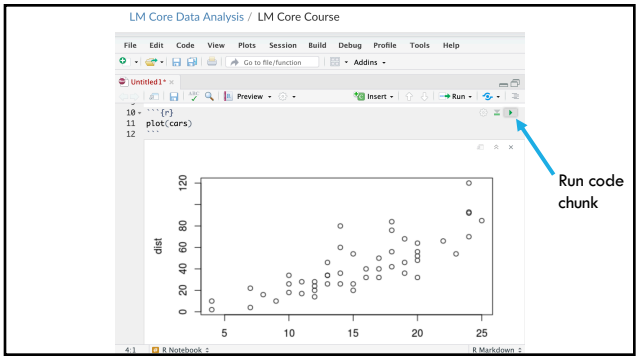

LM Core Data Analysis / LM Core Course
Untitled1
Preview
Insert
Run
---
title: "R Notebook"
output: html_notebook

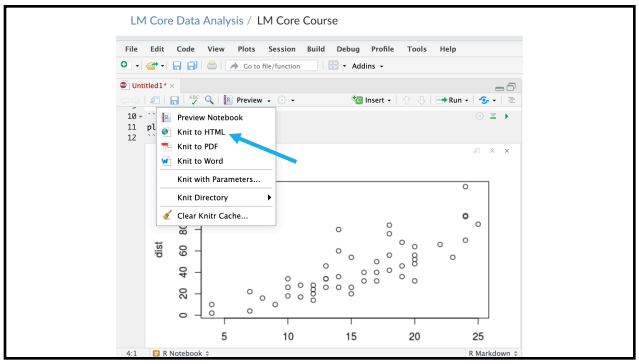
This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.
Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.
```{r}
plot(cars)
```
Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.
When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).
R Notebook
R Markdown

Your Turn #2

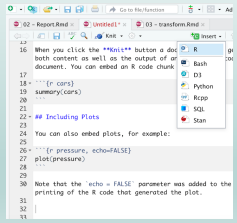
1. Change the title of your R Notebook to "My First R Notebook" by modifying the header.
2. Add your name as the author by adding another line to the header:
   Author: "Your Name"
3. Add a second level heading (##) at the end of the notebook called "My Calculation"

03:00



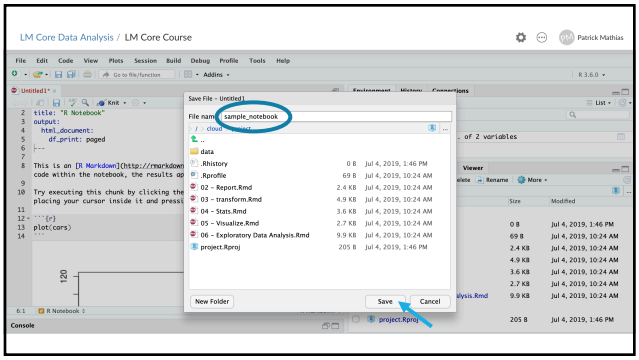


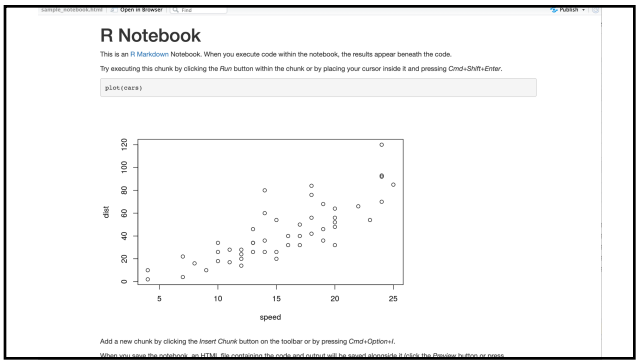
Your Turn #3



1. Under your new heading insert a code chunk into white space using Insert button on top right of code window
2. Add the following to your new code chunk:
   `mean(c(10, 20, 30))`
3. Execute code chunk by pressing Run button on top right of code chunk

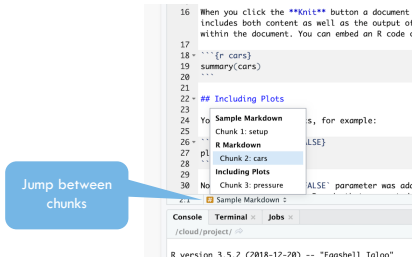
03:00



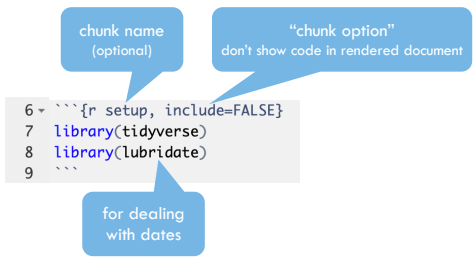


What Else?

Why name Code Chunks?



"Setup" Chunk & Chunk Options



Chunk options

option	default	effect
`eval`	TRUE	Whether to evaluate the code and include its results
`echo`	TRUE	Whether to display code along with its results
`warning`	TRUE	Whether to display warnings
`error`	FALSE	Whether to display errors
`message`	TRUE	Whether to display messages
`tidy`	FALSE	Whether to reformat code in a tidy way when displaying it
`results`	"markup"	"markup", "asis", "hold", or "hide"
`cache`	FALSE	Whether to cache results for future renders
`comment`	"##"	Comment character to preface results with
`fig.width`	7	Width in inches for plots created in chunk
`fig.height`	7	Height in inches for plots created in chunk

For more details visit yihui.name/knitr/

Keyboard Shortcuts

Insert a code chunk into white space within your open R Markdown document using:

- Windows: CTRL+ALT+i
- Mac: COMMAND+OPTION+I

Execute using shortcuts:

- Windows: CTRL+SHIFT+ENTER
- Mac: COMMMAND+SHIFT+ENTER

Multiple Output Formats are Available

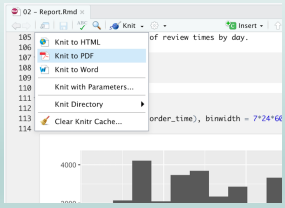
Pandoc universal document converter can create multiple document types:

- html
- pdf
- docx

Can also create presentations and dashboards

- Including Powerpoint (most recent version of RStudio)

Your Turn #4



Create a pdf of your R Notebook using "Knit to pdf"

01:00

Note: if non-executable code is present in code chunks, R will not be able to process it

Lesson Goals

1. Practice working with the different components of a R Markdown file

Lesson Objectives

1. Understand why reproducible reporting is important
2. Create an R Markdown document and generate different types of output files
3. Practice modifying each component of a R Markdown file

An Introduction to the R Programming Language | Lesson 3 **Importing Data**

Lesson Goals

1. Learn functions to import data
2. Practice using tools that help in reviewing imported data

Lesson Objectives

1. Import laboratory data from a common flat file format
2. Inspect imported data with the built-in RStudio viewer

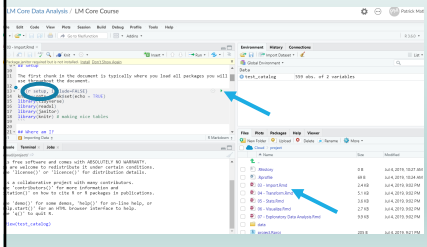
Working with R Markdown for this course

Each lesson has an R Markdown file
- Executable examples
- Exercises

Files used as "notebooks": can document, execute, and iterate

Best practice: work from notebook rather than console

Your Turn #1



1. Open "03 – Import.Rmd"
2. Run the setup chunk

01:00

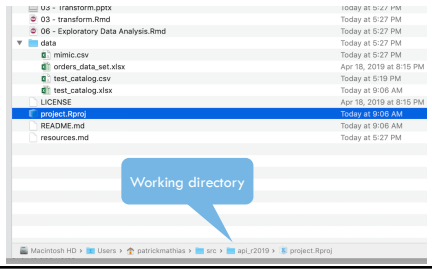
Orders Data Set

- Data set of outpatient laboratory orders
- 45,000 rows x 17 columns
- Queried from Epic orders data (Clarity)
- Deidentified and time shifted
- Includes timestamps for lab order, result, and provider review

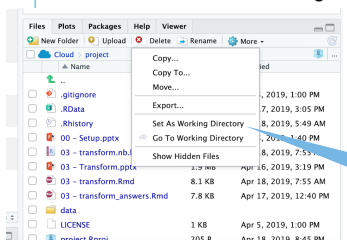
Variable	Description
order_id	Key for order
patient_id	Key for patient
description	Text description of lab test
proc_code	Procedure code for lab test
order_class_c_descr	Setting test is intended to be performed in (eg. Normal = regular blood draw)
lab_status_c	Code for status of laboratory result
lab_status_c_descr	Status of laboratory result
order_status_c	Code for status of order
order_status_c_descr	Status of order
reason_for_canc_c	Code for cancellation reason (if applicable)
reason_for_canc_c_descr	Cancellation reason (if applicable)
order_time	Timestamp for time of original test order
result_time	Timestamp for most recent result in the record
review_time	Timestamp for provider acknowledgment of review of result
department	Clinic associated with test order
ordering_route	Structure/menu in health record from which order was placed
pref_list_type	Category of preference list (if applicable)

Importing Files

Working with your file system



Where am I? Your working directory



Navigate folder and file structure on your computer from Rstudio

getwd() function will tell you which folder you're in

setwd() to set a new working directory

Navigate folders and set working directory

Reading comma separated or tab delimited files

```
read_csv("data/test_catalog.csv")
```

File type (csv, tsv, delim for non-standard delimiters)

File path (if file in a folder within working directory)

File name

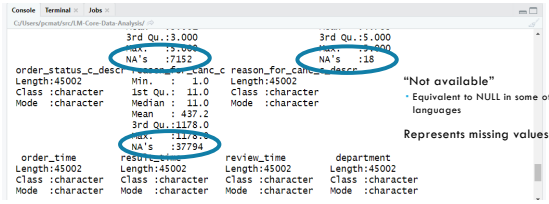
Your Turn #2

order_id	patient_id	description	proc_code	order_class_c_descr	lab_status_c	lab_status_c_descr	order_status_c	order_status_c_descr
19766	511388	PROTHROMBIN TIME	PRO	Normal			4	Canceled
88444	511388	BASIC METABOLIC PANEL	BMP	Normal			4	Canceled
40477	508061	THYROID STIMULATING HORMONE	TSH	Normal	3	Final result	5	Completed
97641	508061	T4, FREE	T4FR	Normal	3	Final result	5	Completed
99868	505646	COMPREHENSIVE METABOLIC PANEL	COMP	Normal	3	Final result	5	Completed

1. Find and import the orders data set and store in an object called "orders"
2. View the data by clicking on the object name in the environment tab
3. Run the summary function on the data

03:00

What is NA?

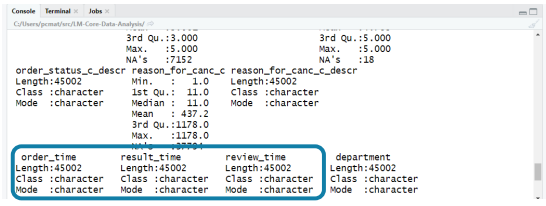


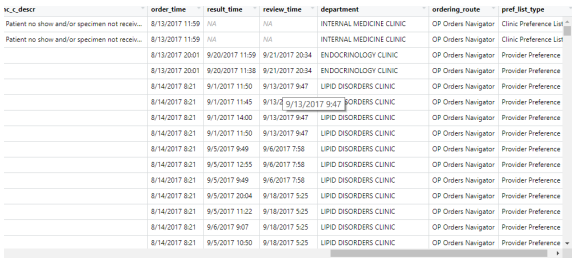
"Not available"

- Equivalent to NULL in some other languages

Represents missing values

Why are the time variables characters?





c_descr	order_time	result_time	review_time	department	ordering_route	pref_list_type
Patient no show and/or specimen not receiv...	8/13/2017 11:59	NA	NA	INTERNAL MEDICINE CLINIC	OP Orders Navigator	Clinic Preference List
Patient no show and/or specimen not receiv...	8/13/2017 11:59	NA	NA	INTERNAL MEDICINE CLINIC	OP Orders Navigator	Clinic Preference List
	8/13/2017 20:01	9/20/2017 11:59	9/21/2017 20:34	ENDOCRINOLOGY CLINIC	OP Orders Navigator	Provider Preference
	8/13/2017 20:01	9/20/2017 11:38	9/21/2017 20:34	ENDOCRINOLOGY CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/1/2017 11:50	9/13/2017 9:47	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/1/2017 11:45	9/13/2017 9:47	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/1/2017 14:00	9/13/2017 9:47	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/1/2017 11:50	9/13/2017 9:47	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/5/2017 9:49	9/6/2017 7:58	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/5/2017 12:55	9/6/2017 7:58	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/5/2017 9:49	9/6/2017 7:58	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/5/2017 20:04	9/18/2017 5:25	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/5/2017 11:22	9/18/2017 5:25	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/6/2017 9:07	9/18/2017 5:25	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference
	8/14/2017 8:21	9/5/2017 10:50	9/18/2017 5:25	LIPID DISORDERS CLINIC	OP Orders Navigator	Provider Preference

Data types/formats can be specified on input

```
orders <- read_csv("data/orders_data_set.csv",
                   col_types = cols(
                     order_time = col_datetime("%m/%d/%Y %H:%M")
                     )
                   )
```

Column types argument to force data type

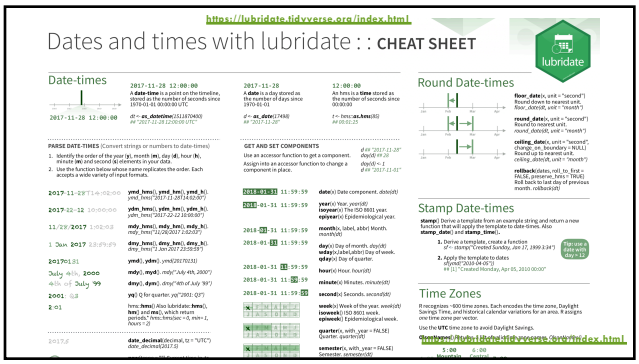
Datetime format

Your Turn #3

1. Import the result_time and review_time columns so they are also datetimes
2. Review the imported data with summary()

03:00

What Else?



Reading Excel files

readxl

```
read_excel("data/excel_file.xlsx",
           sheet = 1)
```

File name

Specify sheet by number or "name"

Can also extract arbitrary rows and columns using "range = " argument

Writing files

readr

```
write_csv(catalog,
"output/test_catalog.csv")
```

File type

Data to write out

Output file name (separate output from raw data!)

Tabulating Data

Create a one variable table

Data frame

Row variable

```
tabyl(orders, department)
```

Part of janitor package: must run library(janitor) to make tabyl function is available

Two variable tables can help answer simple questions quickly

Which clinic cancelled the highest number of lab orders?

Data frame

Row variable

```
tabyl(orders, department,
  order_status_c_descr)
```

Column variable

Prettifying Table Output

```
kable(tabyl(orders, department, pref_list_type))
```

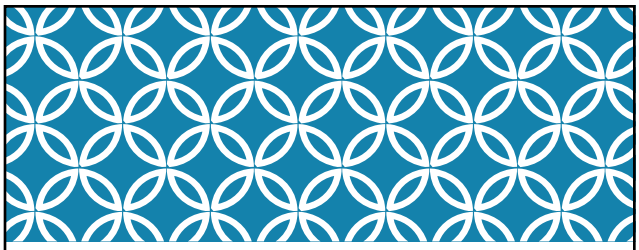
department	Clinic Preference List	None	Provider Preference List
BEHAVIORAL HEALTH CLINIC	40	463	0
CARDIOLOGY CLINIC	888	33	105
ENDOCRINOLOGY CLINIC	613	94	779
FAMILY MEDICINE CLINIC	3135	365	1
GASTROENTEROLOGY CLINIC	618	89	74

Lesson Goals

1. Learn functions to import data
2. Practice using tools that help in reviewing imported data

Lesson Objectives

1. Import laboratory data from a common flat file format
2. Inspect imported data with the built-in RStudio viewer



An Introduction to the R Programming Language

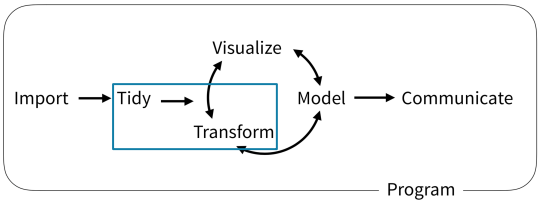
Lesson 4
Data Wrangling

Goals

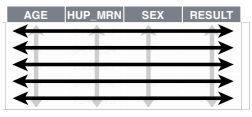
1. Learn how to reduce a data frame to selected data of interest
2. Learn how to expand your data set using data from existing columns

Objectives

1. Select columns and filter rows of interest in a data frame
2. Sort data by specific columns
3. Add columns to your data frame using data from existing columns



What is a "Tidy" Data Frame



A data set is **tidy** if:

1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

Transform Data with

dplyr

dplyr implements a *grammar* for transforming tabular data.

common syntax

Each function takes a data frame as its first argument and returns a data frame as its output.

```
filter(data, ...)
```

dplyr function

data frame to transform

specific arguments

Isolating data

Isolating data

Extract columns with **select()**

Extract rows with **filter()**

Arrange rows, with **arrange()**.

Your Turn 1

Open "**04-Transform.Rmd**"

Run the setup chunk

```
```{r setup}
library(tidyverse)  # Provides functions used throughout this session
library(readxl) # Provides function for reading in excel workbooks
orders <- read_excel("data/orders_data_set.xlsx")
```
```

01:00

select()

select()

Extract columns from a data frame

select()

Extract columns from a data frame

```
select(data,... )
```

dplyr function

data frame to transform

name(s) of columns to extract
(or a select helper)

select()

Extract columns by name.

```
select(orders, description, department)
```

data frame to transform

name(s) of columns to extract (or a select helper)

select()

Extract columns by name.

```
select(orders, description, department)
```

orders

order_id	patient_id	description	proc_code	department
19766	511388	PROTHROMBIN TIME	PRO	INTERNAL MEDICINE CLINIC
88444	511388	BASIC METABOLIC PANEL	BMP	INTERNAL MEDICINE CLINIC
40477	508061	THYROID STIMULATING HORMONE	TSH	ENDOCRINOLOGY CLINIC
97641	508061	T4, FREE	T4FR	ENDOCRINOLOGY CLINIC

description	department
PROTHROMBIN TIME	INTERNAL MEDICINE CLINIC
BASIC METABOLIC PANEL	INTERNAL MEDICINE CLINIC
THYROID STIMULATING HORMONE	ENDOCRINOLOGY CLINIC
T4, FREE	ENDOCRINOLOGY CLINIC

select()

c() is a little function in R that combines two or more values into a vector

Extract columns by index.

```
select(orders, c(1,4))
```

orders

order_id	patient_id	description	proc_code	department
19766	511388	PROTHROMBIN TIME	PRO	INTERNAL MEDICINE CLINIC
88444	511388	BASIC METABOLIC PANEL	BMP	INTERNAL MEDICINE CLINIC
40477	508061	THYROID STIMULATING HORMONE	TSH	ENDOCRINOLOGY CLINIC
97641	508061	T4, FREE	T4FR	ENDOCRINOLOGY CLINIC

order_id	proc_code
19766	PRO
88444	BMP
40477	TSH
97641	T4FR

filter()

filter()

Extract rows that meet logical criteria.

filter()

Extract rows that meet logical criteria.

```
filter(data,... )
```

data frame to transform

one or more logical tests (filter returns each row for which the test is TRUE)

filter()

Extract rows that meet logical criteria.

```
filter(orders, patient_id == 508061)
```

order_id	patient_id	description	proc_code
19766	511388	PROTHROMBIN TIME	PRO
88444	511388	BASIC METABOLIC PANEL	BMP
40477	508061	THYROID STIMULATING HORMONE	TSH
97641	508061	T4, FREE	T4FR

→

order_id	patient_id	description	proc_code
40477	508061	THYROID STIMULATING HORMONE	TSH
97641	508061	T4, FREE	T4FR

filter()

Extract rows that meet logical criteria.

```
filter(orders, patient_id == 508061)
```

order_id	patient_id	description	proc_code
19766	511388	PROTHROMBIN TIME	PRO
88444	511388	BASIC METABOLIC PANEL	BMP
40477	508061	THYROID STIMULATING HORMONE	TSH
97641	508061	T4, FREE	T4FR

= sets
(returns nothing)
== tests if equal
(returns TRUE or FALSE)

filter()

Values coded as character strings must be surrounded by quotes

Extract rows that meet logical criteria.

```
filter(orders, proc_code == "BMP")
```

order_id	patient_id	description	proc_code
19766	511388	PROTHROMBIN TIME	PRO
88444	511388	BASIC METABOLIC PANEL	BMP
40477	508061	THYROID STIMULATING HORMONE	TSH
97641	508061	T4, FREE	T4FR

→

order_id	patient_id	description	proc_code
88444	511388	BASIC METABOLIC PANEL	BMP
55526	511303	BASIC METABOLIC PANEL	BMP
69809	509686	BASIC METABOLIC PANEL	BMP
24316	503847	BASIC METABOLIC PANEL	BMP

Logical tests

`x < y`	Less than
`x > y`	Greater than
`x == y`	Equal to
`x <= y`	Less than or equal to
`x >= y`	Greater than or equal to
`x != y`	Not equal to
`x %in% y`	Group membership
`is.na(x)`	Is NA
`!is.na(x)`	Is not NA

Your Turn 2

Use filter() with the logical operators to find:

a) Every **order_id** that is greater than 100000

b) All of the orders where **lab_status_c_descr** is equal to "Final result"

Common mistakes

1. Using **=** instead of **==**

```
filter(orders, proc_code = "BMP")
filter(orders, proc_code == "BMP")
```

2. Forgetting quotes

```
filter(orders, proc_code == COMP)
filter(orders, proc_code == "COMP")
```

Common mistakes

3. Capitalization matters

```
Filter(Orders, proc_code == "BMP")
filter(orders, proc_code == "BMP")
```

4. Spelling

```
fitler(orders, proc_code == "COMP")
filter(orders, proc_code == "COMP")
```

filter()

Extract rows that meet *multiple* logical criteria.

```
filter(orders, patient_id == 508061, description=="T4, FREE")
```

orders

order_id	patient_id	description	proc_code
19766	511388	PROTHROMBIN TIME	PRO
88444	511388	BASIC METABOLIC PANEL	BMP
40477	508061	THYROID STIMULATING HORMONE	TSH
97641	508061	T4, FREE	T4FR

→

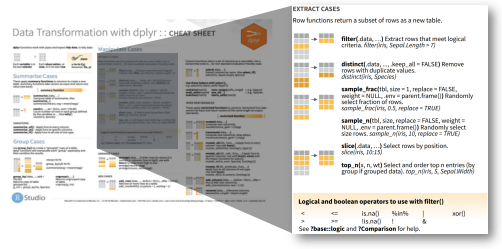
order_id	patient_id	description	proc_code
97641	508061	T4, FREE	T4FR

Boolean operators

?base::Logic

`a & b`	and
`a | b`	or
`!a`	not
`a %in% c(a, b)`	one of (in)

filter() variants



arrange( )

arrange()

Order rows from smallest to largest values

```
arrange(data, ... )
```

data frame to transform

name(s) of columns to arrange by

arrange()

Order rows from smallest to largest values

```
arrange(orders, result_time)
```

order_id	patient_id	description	result_time
19766	511388	PROTHROMBIN TIME	2017-09-20
88444	511388	BASIC METABOLIC PANEL	2017-09-01
40477	508061	THYROID STIMULATING HORMONE	2017-09-28
97641	508061	T4, FREE	2017-09-04

→

order_id	patient_id	description	result_time
88444	511388	BASIC METABOLIC PANEL	2017-09-01
97641	508061	T4, FREE	2017-09-04
19766	511388	PROTHROMBIN TIME	2017-09-20
40477	508061	THYROID STIMULATING HORMONE	2017-09-28

arrange()

Order rows from smallest to largest values

```
arrange(orders, desc(result_time))
```

order_id	patient_id	description	result_time
19766	511388	PROTHROMBIN TIME	2017-09-20
88444	511388	BASIC METABOLIC PANEL	2017-09-01
40477	508061	THYROID STIMULATING HORMONE	2017-09-28
97641	508061	T4, FREE	2017-09-04

→

order_id	patient_id	description	result_time
40477	508061	THYROID STIMULATING HORMONE	2017-09-28
19766	511388	PROTHROMBIN TIME	2017-09-20
97641	508061	T4, FREE	2017-09-04
88444	511388	BASIC METABOLIC PANEL	2017-09-01

arrange()

Order rows from smallest to largest values

```
arrange(orders, patient_id, result_time)
```

order_id	patient_id	description	result_time
19766	511388	PROTHROMBIN TIME	2017-09-20
88444	511388	BASIC METABOLIC PANEL	2017-09-01
40477	508061	THYROID STIMULATING HORMONE	2017-09-28
97641	508061	T4, FREE	2017-09-04

→

order_id	patient_id	description	result_time
97641	508061	T4, FREE	2017-09-04
40477	508061	THYROID STIMULATING HORMONE	2017-09-28
88444	511388	BASIC METABOLIC PANEL	2017-09-01
19766	511388	PROTHROMBIN TIME	2017-09-20

%>%

Data Analysis Steps

```
VITD <- filter(orders, description == "1,25 DIHYDROXY VITAMIN D")

VITD <- select(VITD, department, ordering_route, pref_list_type)

VITD <- arrange(VITD, department)
```

1. Filter orders to 1, 25 Vitamin D
2. Select the relevant columns that contain ordering information
3. Arrange those columns by department

Data Analysis Steps

```
VITD <- arrange(
           select(
             filter(
               orders,
               description == "1,25 DIHYDROXY VITAMIN D"
             ),
             department,
             ordering_route,
             pref_list_type
           ),
           pref_list_type
         )
```

The Pipe Operator %>%

```
orders  %>%  filter(____, patient_id == 508061)
```

Passes result on left into first argument of function on right.

```
filter(orders, patient_id == 508061)
orders %>% filter(patient_id == 508061)
```

Data Analysis Steps

```
125VITD <- arrange(
            select(
              filter(
                orders,
                description == "1,25 DIHYDROXY VITAMIN D"
              ),
              department,
              ordering_route,
              pref_list_type
            ),
            pref_list_type
          )
```

Data Analysis Steps

```
VITD <- orders %>%
            filter(description == "1,25 DIHYDROXY VITAMIN D") %>%
            select(department, ordering_route, pref_list_type) %>%
            arrange(pref_list_type)
```

Shortcut to type %>%

Cmd + Shift + M (Mac)

Ctrl + Shift + M (Windows)

Your Turn 3

Use %>% to write a sequence of three functions that:

- Filters to orders coming from the "BEHAVIORAL HEALTH CLINIC"
- Selects the **description**, **ordering_route**, and **pref_list_type**
- Arrange the dataset by the **description** and **ordering_route** columns

Using <-, assign the result to a new variable.
- CHALLENGE- Use your mouse to select the name of the new data frame from the list of data sets in the upper-right pane of Rstudio. Do you notice any ordering patterns?

05:00

Isolating data

Extract variables with **select()**

Extract rows with **filter()**

Arrange rows, with **arrange()**.

Deriving Data

What are the top 3 tests ordered on weekends?

Breaking down the analytical question

1. Day of the week for each order
2. Count of each test on weekends
3. Ranking of tests by count

Deriving data

Make new variables with **mutate()**

Make summaries of data with **summarize()**

mutate()

mutate()

Creating new columns

mutate()

Creating new columns

```
orders %>%
    mutate(new_column = calculation)
```

name for new column

equals

function whose results will populate columns

mutate()

Creating new columns

calculation can involve another column in the data frame

```
orders %>%
    mutate(coded_order_id = order_id/2)
```

order_id	patient_id
19766	511388
88444	511388
40477	508061
97641	508061

→

order_id	patient_id	coded_order_id
19766	511388	9883
88444	511388	44222
40477	508061	20238
97641	508061	48820

Your Turn 4

The weekdays() function will return the weekday for any date.

1. Use the weekdays() function with mutate() to make a new column which contains the day of the week that each order was placed
2. Then select this column and the **order_time** column

05:00

What Else?

select()
also helpful for renaming

```
select(orders,
       desc = description,
       dept = department)
```

orders

order_id	patient_id	description	proc_code	department
19766	511388	PROTHROMBIN TIME	PRO	INTERNAL MEDICINE CLINIC
88444	511388	BASIC METABOLIC PANEL	BMP	INTERNAL MEDICINE CLINIC
40477	508061	THYROID STIMULATING HORMONE	TSH	ENDOCRINOLOGY CLINIC
97641	508061	T4, FREE	T4FR	ENDOCRINOLOGY CLINIC

→

desc	dept
PROTHROMBIN TIME	INTERNAL MEDICINE CLINIC
BASIC METABOLIC PANEL	INTERNAL MEDICINE CLINIC
THYROID STIMULATING HORMONE	ENDOCRINOLOGY CLINIC
T4, FREE	ENDOCRINOLOGY CLINIC

dplyr

rename()

```
rename(orders,
       desc = description,
       dept = department)
```

orders

order_id	patient_id	description	proc_code	department
19766	511388	PROTHROMBIN TIME	PRO	INTERNAL MEDICINE CLINIC
88444	511388	BASIC METABOLIC PANEL	BMP	INTERNAL MEDICINE CLINIC
40477	508061	THYROID STIMULATING HORMONE	TSH	ENDOCRINOLOGY CLINIC
97641	508061	T4, FREE	T4FR	ENDOCRINOLOGY CLINIC

→

order_id	patient_id	desc	proc_code	dept
19766	511388	PROTHROMBIN TIME	PRO	INTERNAL MEDICINE CLINIC
88444	511388	BASIC METABOLIC PANEL	BMP	INTERNAL MEDICINE CLINIC
40477	508061	THYROID STIMULATING HORMONE	TSH	ENDOCRINOLOGY CLINIC
97641	508061	T4, FREE	T4FR	ENDOCRINOLOGY CLINIC

dplyr

select() helpers

: Select range of columns

```
select(orders, order_id:lab_status_c)
```

– Select every column but

```
select(orders, -c(description, order_status_c))
```

starts_with() Select columns that start with…

```
select(orders, starts_with("order"))
```

ends_with() Select columns that end with…

```
select(orders, ends_with("descr"))
```

select() helpers

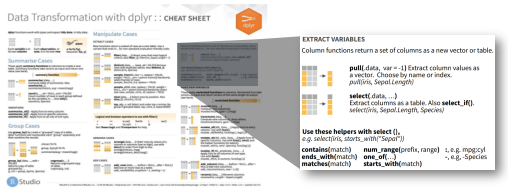
contains() Select columns whose names contain…

```
select(orders, contains("time"))
```

matches() Select columns whose names match regular expression

```
select(orders, matches("^.{4}$"))
```

select() helpers



mutate()

Replacing columns

Function to "coerce" one type of data into another type of data

```
orders %>%
    mutate(order_id = as.character(order_id))
```

order_id	patient_id	description
19766	511388	PROTHROMBIN TIME
88444	511388	BASIC METABOLIC PANEL
40477	508061	THYROID STIMULATING HORMONE
97641	508061	T4, FREE

data types in R

Type of Data	Function	Result of Function
Character	as.character(1)	"1"
Numeric	as.numeric("123")	123
Logical	as.logical(1)	TRUE
Date	as.Date("05-06-2019")	NA

mutate()

Conditionally replace values in a column

```
orders %>%
    mutate(proc_code =
        ifelse(proc_code %in% c("CBC","CBD"),
               "CBC",
               proc_code))
```

logical statement

value if true

value if false

mutate() + case_when()

Apply multiple conditions

Logical statement

Output

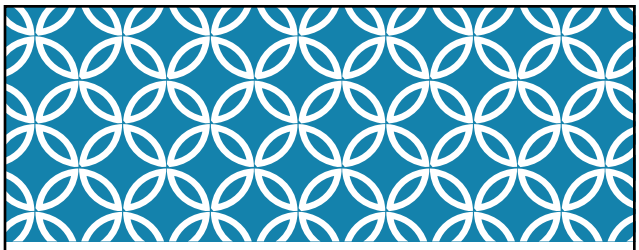
```
orders <- orders %>%
  mutate(pref_list_cat = case_when(
    pref_list_type == "Clinic Preference List" ~ "clinic",
    pref_list_type == "Provider Preference List" ~ "provider",
    pref_list_type == "None" ~ NA_character_
  ))
```

Goals

1. Learn how to reduce a data frame to selected data of interest
2. Learn how to expand your data set using data from existing columns

Objectives

1. Select columns and filter rows of interest in a data frame
2. Sort data by specific columns
3. Add columns to your data frame using data from existing columns



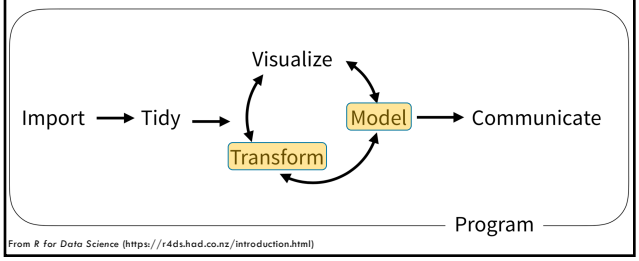
An Introduction to the R Programming Language | Lesson 5 **Summarize Data**

Objectives

Be able to…

1. Summarize variables, including calculation of summary statistics
2. Break apart data frames observations into groups
3. Combine grouping and summarization to quickly calculate basic statistics for subgroups

Typical Data Science Pipeline

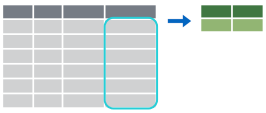


From R for Data Science (https://r4ds.had.co.nz/introduction.html)

Summarize()

summarize()

- Make summaries of your data



summarize()

- Make summaries of your data

```
orders %>%
    summarize(new_variable = calculation)
```

name for new variable

Value or function

summarize()

- Make summaries of your data

function that returns number of observations

```
orders %>%
    select(order_id, patient_id) %>%
    head(4) %>%
    summarize(order_count = n())
```

order_id	patient_id
19766	511388
88444	511388
40477	508061
97641	508061

→

order_count
4

summarize()

- Make summaries of your data

function that returns number of distinct values

```
orders %>%
    select(order_id, patient_id) %>%
    head(4) %>%
    summarize(order_count = n(),
              pt_count = n_distinct(patient_id))
```

order_id	patient_id
19766	511388
88444	511388
40477	508061
97641	508061

→

order_count	pt_count
4	2

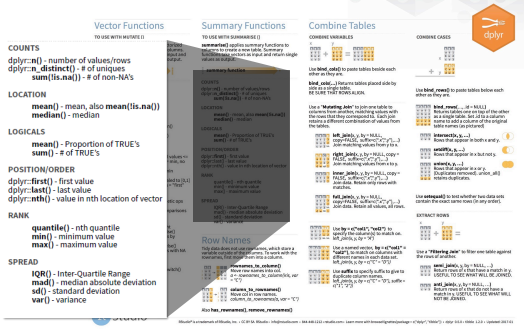
Your Turn 1

Add onto the code in the above chunk to calculate:

a) Mean count of orders per patient

b) Mean count of orders per department

03:00



Your Turn 2

Use *summarize()* to calculate:

a) The date of the first (or minimum) order

b) The median time difference between *order_time* and *result_time*

Hint Refer to help for *NA* handling 05:00

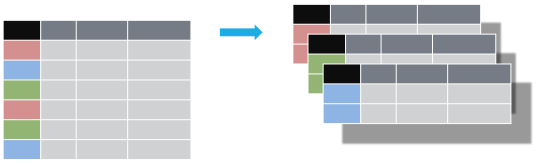
Your Turn 3

Consider:

How would you calculate the median number of orders for each patient?

group_by()

group_by()



group_by()

- *Grouping observations based on a specific variable's values*

```
orders %>%
    group_by(variable)
```

name of variable to group by

group_by()

- *Group observations by patient_id*

```
orders %>%
    group_by(patient_id)
```

```
# A tibble: 45,002 x 17
# Groups:   patient_id [9,406]
   order_id patient_id description proc_code
      <dbl>      <dbl> <chr>       <chr>
 1    19766     511388 PROTHROMBI… PRO
 2    88444     511388 BASIC META… BMP
 3    40477     508061 THYROID ST… TSH
 4    97641     508061 T4, FREE    T4FR
```

group_by()

- *Group observations by patient_id and department*

```
orders %>%
    group_by(patient_id, department)
```

```
# A tibble: 45,002 x 17
# Groups:   patient_id, department [10,102]
   order_id patient_id description proc_code
      <dbl>      <dbl> <chr>       <chr>
 1    19766     511388 PROTHROMBI… PRO
 2    88444     511388 BASIC META… BMP
 3    40477     508061 THYROID ST… TSH
 4    97641     508061 T4, FREE    T4FR
```

group_by() %>% summarize()

group_by() %>% summarize()

Make summaries of your data *by group*

Re-calculate the mean order count per patient

- ***Recall*** *that we previously used summarize() and mutate() like this:*

```
orders %>%
     summarize(order_count = n(),
               pt_count = n_distinct(patient_id),
               pt_order_count_mean = order_count / pt_count)
```

```
# A tibble: 1 x 3
  order_count pt_count pt_order_count_mean
        <int>    <int>               <dbl>
1       45002     9406            4.784393
```

Calculate the mean order count per patient

- *Let's now use group_by() and summarize()…*

```
orders %>%
     group_by(patient_id) %>%
     summarize(order_count = n()) %>%
     summarize(pt_order_count_mean = mean(order_count))
```

1

2

```
# A tibble: 9,406 x 2
   patient_id order_count
        <dbl>       <int>
1      500001           1
2      500002           7
3      500003           5
4      500005           1
5      500006           5
```

```
# A tibble: 1 x 1
  pt_order_count_mean
                <dbl>
1            4.784393
```

Your Turn 4

Calculate:

a) The median number of orders per patient

b) The maximum number of TSH orders per patient
- Hint: `proc_code` == "TSH"

c) (*Bonus*) The 5th and 95th percentile of the number of orders per patient
- *Hint: look up function `quantile()`*

05:00

Objectives

Be able to…

1. Summarize variables, including calculation of summary statistics
2. Break apart data frames observations into groups
3. Combine grouping and summarization to quickly calculate basic statistics for subgroups

What Else?

Your Turn 5

Calculate for each department the mean order count per patient

Hint: summarize() rolls up a single grouping variable at a time

department <chr>	pt_order_count_mean <dbl>
BEHAVIORAL HEALTH CLINIC	5.651685
CARDIOLOGY CLINIC	3.916031
ENDOCRINOLOGY CLINIC	3.416092
FAMILY MEDICINE CLINIC	3.609278
GASTROENTEROLOGY CLINIC	4.540698

05:00

An Introduction to the R Programming Language

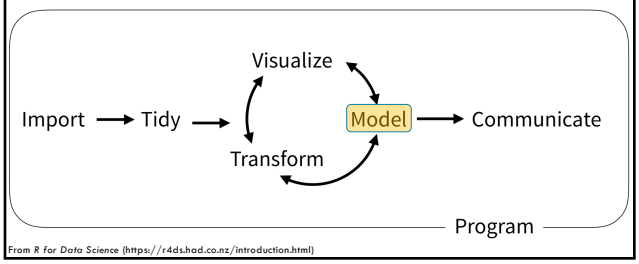
Lesson 6
Statistics

Objectives

Be able to…

1. Calculate basic statistical calculations for groups within a data set
2. Use standard statistical modeling functions
3. Compare variables' distributions

Typical Data Science Pipeline



From *R for Data Science* (https://r4ds.had.co.nz/introduction.html)

Q: Are the number of orders per patient different in Internal Medicine and Family Medicine?

Describe the central tendency of the order count

```
orders %>%
       group_by(department, patient_id) %>%
       summarize(order_count = n()) %>%
       summarize(order_count_median = median(order_count),
                 order_count_mean = mean(order_count)) %>%^
       filter(department %in% c("INTERNAL MEDICINE CLINIC",
                                "FAMILY MEDICINE CLINIC"))
```

1

```
# A tibble: 10,102 x 3
   department              patient_id order_count
   <chr>                        <int>       <int>
 1 BEHAVIORAL HEALTH CLINIC     500126           1
 2 BEHAVIORAL HEALTH CLINIC     500251           9
 3 BEHAVIORAL HEALTH CLINIC     500385          13
 4 BEHAVIORAL HEALTH CLINIC     500395           1
 5 BEHAVIORAL HEALTH CLINIC     500519           1
```

```
# A tibble: 2 x 3
  department              order_count_median order_count_mean
  <chr>                                <dbl>            <dbl>
1 FAMILY MEDICINE CLINIC                   3         3.609278
2 INTERNAL MEDICINE CLINIC                 3         3.888664
```

2

Basic statistics functions

Central Tendency and Variability

Function	What it Calculates
mean(x)	Mean of the numbers in vector x.
median(x)	Median of the numbers in vector x
var(x)	Estimated variance of the population from which the numbers in vector x are sampled
sd(x)	Estimated standard deviation of the population from which the numbers in vector x are sampled
scale(x)	Standard scores (z-scores) for the numbers in vector x

https://www.dummies.com/education/math/statistics/base-r-statistical-functions/

Prep: Order count per patient for each department

```
orders_per_pt_dept <- orders %>%
       group_by(department, patient_id) %>%
       summarize(order_count = n()) %>%
       ungroup() %>%
       filter(department %in% c("INTERNAL MEDICINE CLINIC",
                                "FAMILY MEDICINE CLINIC"))
```

1

```
# A tibble: 10,102 x 3
  department                patient_id order_count
  <chr>                     <int>      <int>
1 BEHAVIORAL HEALTH CLINIC  500126         1
2 BEHAVIORAL HEALTH CLINIC  500251         9
3 BEHAVIORAL HEALTH CLINIC  500385        13
4 BEHAVIORAL HEALTH CLINIC  500395         1
5 BEHAVIORAL HEALTH CLINIC  500519         1
```

2

```
# A tibble: 2,632 x 3
  department                patient_id order_count
  <chr>                     <int>      <int>
1 FAMILY MEDICINE CLINIC    500003         5
2 FAMILY MEDICINE CLINIC    500022         4
3 FAMILY MEDICINE CLINIC    500024        14
4 FAMILY MEDICINE CLINIC    500034         2
5 FAMILY MEDICINE CLINIC    500042         4
6 FAMILY MEDICINE CLINIC    500043         9
```

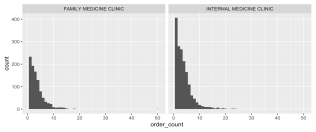
Visualize the distribution of order counts

```
orders_per_pt_dept %>%
       <insert ggplot code>
```

orders_per_pt_dept

department	patient_id	order_count
FAMILY MEDICINE CLINIC	500003	5
FAMILY MEDICINE CLINIC	500022	4
FAMILY MEDICINE CLINIC	500024	14
FAMILY MEDICINE CLINIC	500034	2
FAMILY MEDICINE CLINIC	500042	4
FAMILY MEDICINE CLINIC	500043	9

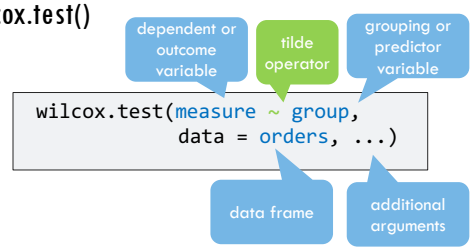
6 rows



What test can compare count distributions?

Populations	Parametric	Non-parametric
Two populations	t-test	Mann-Whitney U
Many populations	ANOVA	Kruskal Wallis / one-way anova
Populations across several treatments/times	repeated measures ANOVA	Friedman test

wilcox.test()



Compare distributions by rank order

Compare order count distributions between 2 departments

- *Wilcoxon rank-sum test*

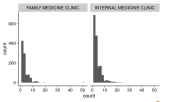
```
wilcox.test(order_count ~ department,
            data = orders_per_pt_dept,
            alternative = "two.sided",
            paired = FALSE,
            conf.int = TRUE)
```

```
        Wilcoxon rank sum test with continuity correction

data:  count by department
W = 779510, p-value = 0.1521
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -3.210939e-06  1.571466e-05
sample estimates:
difference in location
         -4.511812e-05
```

orders_per_pt_dept

```
# A tibble: 6 x 3
                department patient_id count
                     <chr>      <dbl> <int>
1 FAMILY MEDICINE CLINIC      500003     5
2 FAMILY MEDICINE CLINIC      500022     4
3 FAMILY MEDICINE CLINIC      500024    14
4 FAMILY MEDICINE CLINIC      500034     2
5 FAMILY MEDICINE CLINIC      500042     4
6 FAMILY MEDICINE CLINIC      500043     9
```



Objectives

Be able to…

1. Calculate basic statistical calculations for groups within a data set
2. Use standard statistical modeling functions
3. Compare variables' distributions

What Else?

Q: Can we predict the number of orders for each department in September?

Prepare: Order per month for each department

```
orders_per_pt_dept <- orders %>%
       mutate(order_month = month(order_time, label = TRUE, abbr = TRUE)) %>%
       group_by(department, order_month) %>%
       summarize(order_count = n()) %>%
       ungroup() %>%
       spread(key = order_month, value = order_count)
```

1

```
# A tibble: 45,002 x 2
   department              order_month
   <chr>                   <ord>
 1 INTERNAL MEDICINE CLINIC Aug
 2 INTERNAL MEDICINE CLINIC Aug
 3 ENDOCRINOLOGY CLINIC     Aug
 4 ENDOCRINOLOGY CLINIC     Aug
 5 LIPID DISORDERS CLINIC   Aug
 6 LIPID DISORDERS CLINIC   Aug
```

2

department	Aug	Sep	Oct	Nov
<chr>	<int>	<int>	<int>	<int>
1 BEHAVIORAL HEALTH CLINIC	97	160	177	69
2 CARDIOLOGY CLINIC	281	305	332	108
3 ENDOCRINOLOGY CLINIC	255	551	559	121
4 FAMILY MEDICINE CLINIC	683	1162	1274	382
5 GASTROENTEROLOGY CLINIC	202	278	239	62

Simple linear regression modeling

```
model <- lm(formula = y_data ~ x_data, data = data_set)
summary(model)
coef(model)
```

Simple Output

```
Call:
lm(formula = Sales ~ Spend, data = dataset)

Residuals:
    Min      1Q  Median      3Q     Max
 -3385  -2097    258   1726   3034

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1383.4714  1255.2404   1.102    0.296
Spend         10.6222     0.1625  65.378 1.71e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2313 on 10 degrees of freedom
Multiple R-squared:  0.9977,  Adjusted R-squared:  0.9974
F-statistic:  4274 on 1 and 10 DF,  p-value: 1.707e-14
```

Multiple Regression Output

```
Call:
lm(formula = Sales ~ Spend + Month, data = dataset)

Residuals:
      Min        1Q    Median        3Q       Max
-1793.73 -1558.33    -1.73  1374.19  1911.58

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -567.6098  1041.8836  -0.545  0.59913
Spend         10.3825     0.1328  78.159 4.65e-14 ***
Month        541.3736   158.1660   3.423  0.00759 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1607 on 9 degrees of freedom
Multiple R-squared:  0.999,  Adjusted R-squared:  0.9988
F-statistic:  4433 on 2 and 9 DF,  p-value: 3.368e-14
```

Your Turn 1

a) Use linear regression to predict departments' October order counts using September order counts

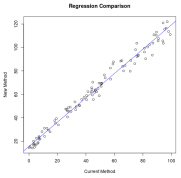
b) (*Bonus*) Use deming regression to compare Oct and Sep order counts

- *HINT*: Look at library "mcr"

c) (*Bonus*) Try to improve the linear regression prediction by adding more predictors

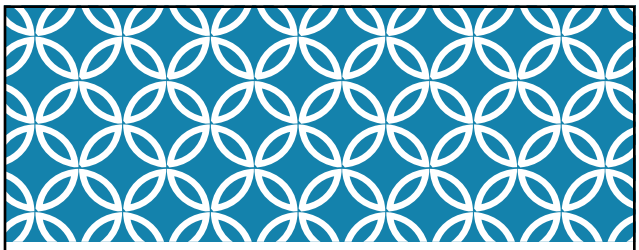
05:00

Method comparison regression package for more advanced regression options




```
install.packages("mcr")
library(mcr)
model <- mcreg(x_data, y_data, method.reg = "Deming")
printSummary(model)
getCoefficients(model)
```

https://www.r-bloggers.com/deming-and-passing-bablok-regression-in-r/



An Introduction to the R Programming Language

Lesson 7
Visualizing Data

Goals

1. Appreciate the importance of visualization for understanding data
2. Learn how to use ggplot2 to visualize data

Objectives

1. Know the 3-step approach to make any kind of graph
2. Define "aesthetic" and explain how aesthetic mappings relate variables of a data frame to graphic markings on a graph
3. Define "geom" and explain how geom functions create layers of a graph
4. Distinguish between setting vs mapping aesthetics
5. Distinguish between global vs local settings

Your Turn 1

Consider the `orders` data frame. How do you think `order_time` and `result_time` are related to each other?

Pair up and discuss.

01:00

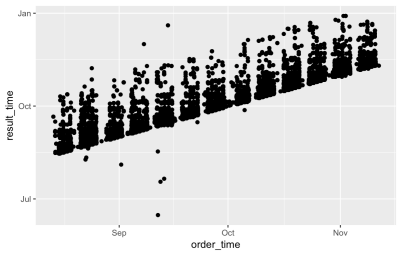
Your Turn 2

Type the following code in the console to make a graph.

Pay attention to the spelling, capitalization, and parentheses!

```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time))
```

01:00




```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time))
```

When you run this code, you will get what looks like an error but is actually just a message. R lets you know that some of the rows did not have data for order_time and/or result_time so those points were not plotted.

```
Warning message:
Removed 7152 rows containing missing values (geom_point).
```

```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time))
```

ggplot()

initialize a plot with **ggplot()**

data frame

+ sign before new line

```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time))
```

type of layer

mappings inside **aes()** function

x variable

y variable

To make **any** kind of graph:

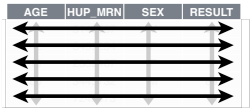
1. Choose a "tidy" **data** frame

```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings))
```

2. Pick a "**geom**" function

3. Write aesthetic **mappings**

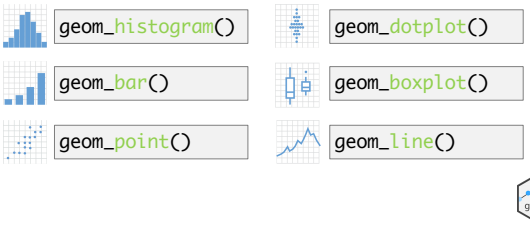
1. Pick a "Tidy" Data Frame

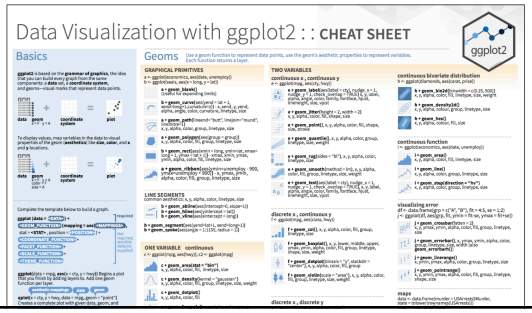


A data set is **tidy** if:

1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

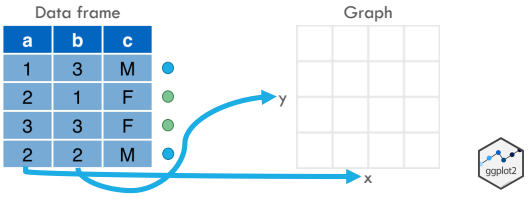
2. Choose a "Geom" Function



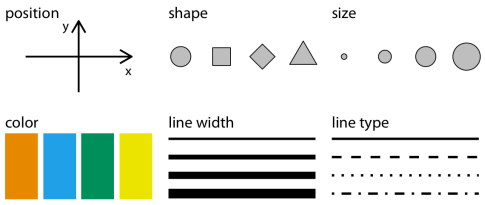


3. Write Aesthetic Mappings

```
aes(x = a, y = b, color = c)
```



Aesthetics



Your Turn 3

Consider the 6 aesthetics shown just previously. All aesthetics can be used for discrete variables, but only some can be used for continuous variables.

Which aesthetics can be used only for discrete variables?

A. Color
B. Line type
C. Shape
D. B + C
E. A + B + C

01:00

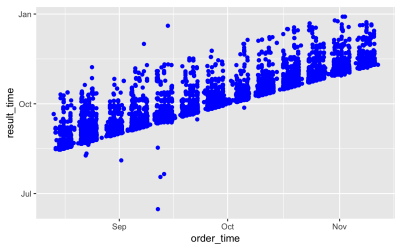
Your Turn 4

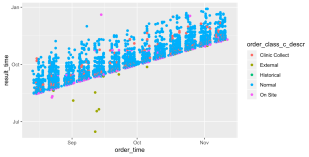
Open `07-visualize.Rmd`. Work through the exercises of the section titled "Your Turn 4."

05:00

Setting vs Mapping Aesthetics

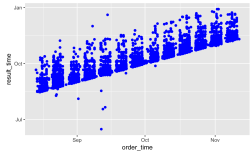
How would you make this plot?





Inside of aes():
map an aesthetic
to a variable

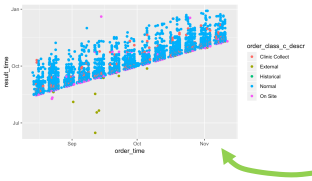
```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time, color = order_class_c_descr))
```

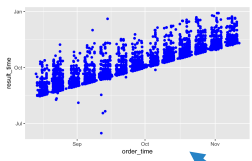


Outside of aes():
set an aesthetic to
a value

color name in
"quotes"

```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time), color = "blue")
```



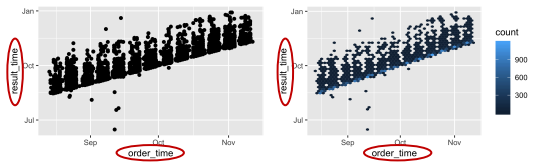



```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time, color = order_class_c_descr))
```

```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time), color = "blue")
```

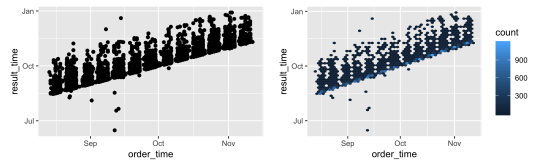
Geom Functions

How are these plots similar?



Same: x axis, y axis, data

How are these plots different?



Different **geometric object** ("geom") used to represent the data

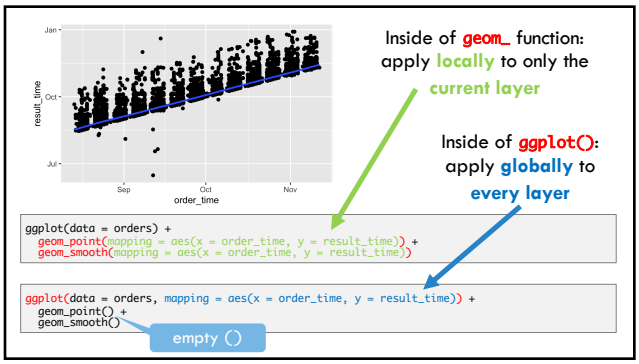
```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings))
```

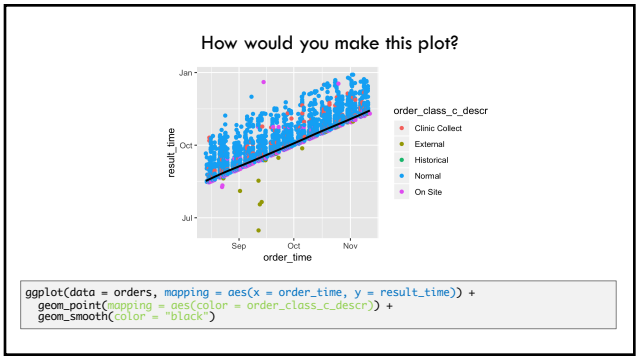
Your Turn 5

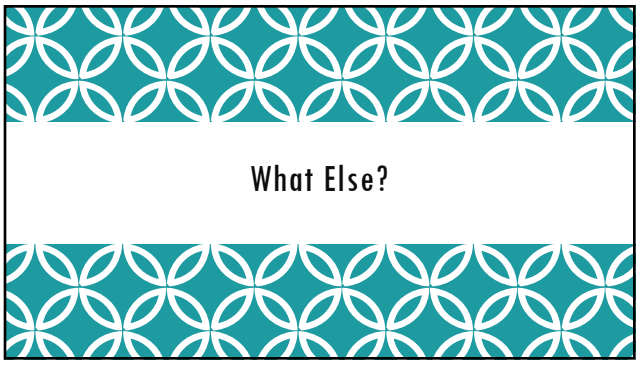
Open `07-visualize.Rmd`. Work through the exercises of the section titled "Your Turn 5."

05:00

Global vs Local Settings

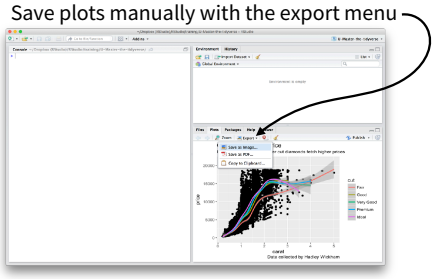

Jan
Oct
Jul
result_time
Sep
Oct
Nov
order_time
Inside of geom_ function: apply locally to only the current layer
Inside of ggplot(): apply globally to every layer
ggplot(data = orders) +
geom_point(mapping = aes(x = order_time, y = result_time)) +
geom_smooth(mapping = aes(x = order_time, y = result_time))
ggplot(data = orders, mapping = aes(x = order_time, y = result_time)) +
geom_point() +
geom_smooth()
empty ()


How would you make this plot?
Jan
Oct
Jul
result_time
Sep
Oct
Nov
order_time
order_class_c_descr
Clinic Collect
External
Historical
Normal
On Site
ggplot(data = orders, mapping = aes(x = order_time, y = result_time)) +
geom_point(mapping = aes(color = order_class_c_descr)) +
geom_smooth(color = "black")


What Else?

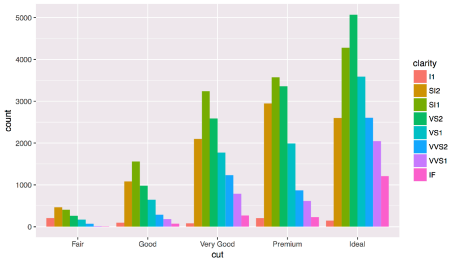
Manually saving plots

Save plots manually with the export menu



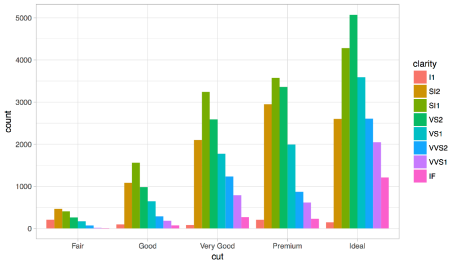
Position Adjustments

How overlapping objects are arranged



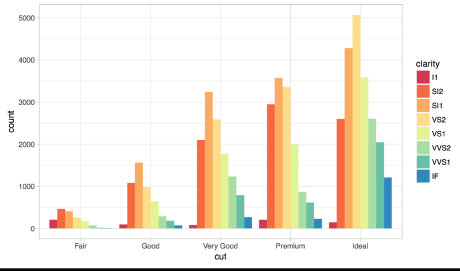
Themes

Visual appearance of non-data elements



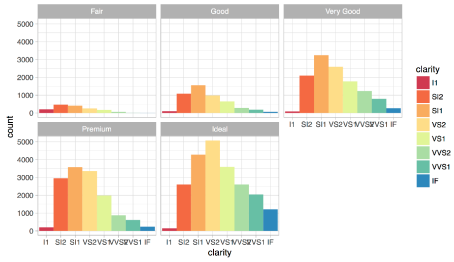
Scales

Customize color scales, other mappings



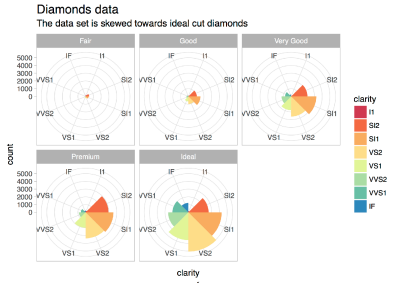
Facets

Subplots that display subsets of the data.



Coordinate systems

Titles and captions




```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings)) +
  theme_function +
  scale_function +
  facet_function +
  coordinate_function +
  ...
```

Required

Optional

Goals

1. Appreciate the importance of visualization for understanding data
2. Learn how to use ggplot2 to visualize data

Objectives

1. Know the 3-step approach to make any kind of graph
2. Define "aesthetic" and explain how aesthetic mappings relate variables of a data frame to graphic markings on a graph
3. Define "geom" and explain how geom functions create layers of a graph
4. Distinguish between setting vs mapping aesthetics
5. Distinguish between global vs local settings