

Storytelling with data

01 | analytics communication challenges; example case study; workflow, a common approach with R or Python libraries

welcome, course overview

Meeting your professor

Education

Doctorate

Jurisprudence

Honors in research and writing

Focus — analysis

Master of Science

Sports Management

Focus — data science analytics

Won, SABR analytics competition

Bachelor of Science

Chemical Engineering

Focus — numerical methods,

statistical process control



Scott Spencer Columbia University

Faculty, Lecturer, Alumnus

Consultant, Data Scientist

Professional sports

Example — Major League Baseball research and development for player performance & manager decision-making

Data for good

Example — Bayesian, generative modeling effects of climate change on perceived expectations of property values

Innovation

Example — whether invented attributes of an edible oil previously existed or was made or sold by competitor

Teaching and Research

Developing generative models

Building Bayesian, generative models to enable decision-making in complex fields such as sports performance.

Communicating uncertainty

Writing monograph on quantitative persuasion amid uncertainty. Developing R packages to tie human perception to graphical representation of data.

Contributing open-source software

Contribute to interfaces to Stan, a probabilistic programming language.

Meeting your Associate

Education

Doctorate
Education

Master of Science
Media Studies

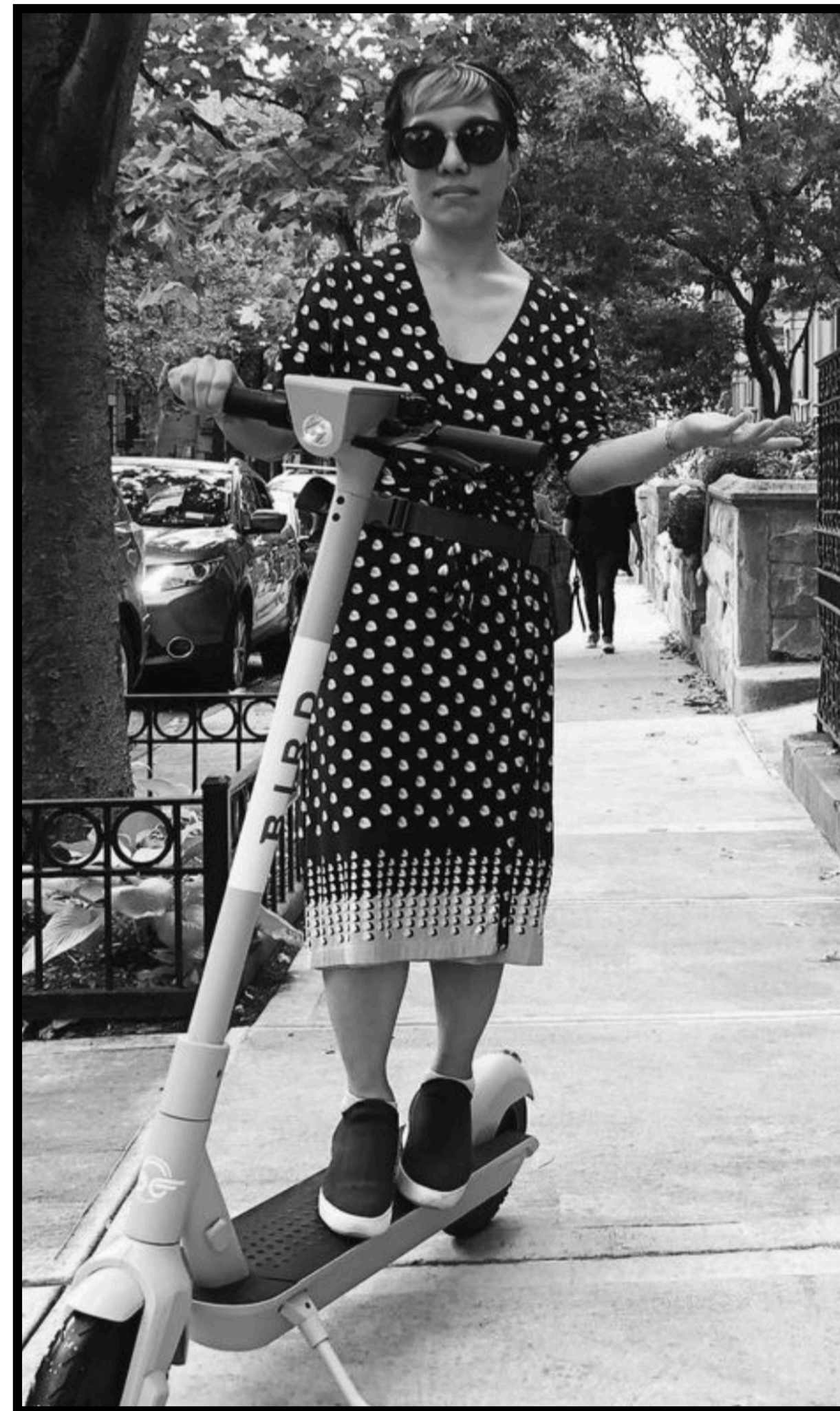
Bachelor of Fine Arts
Design

Background & Interests

Born: *Suwon, Korea*

Languages: *English, Korean, Japanese*

Hobbies: *Founder of the NYC Ramen Enthusiasts, Kickboxing, Travel*



Laura Scherling Columbia University

Director, Associate Faculty SPS & TC

laurascherling.info

Instagram: [laura.skerling](https://www.instagram.com/laura.skerling)

Research & Publications

Areas of research

Management, Marketing Analytics, Ed Tech and Higher Ed Analytics, Information Design, Visual Design, Digital Transformation, Sustainability

Publications

Ethics in Design (Bloomsbury, 2020); Digital Transformation in Design (under review, Bloomsbury 2022, expected)

Research publications for:

Brooking Metro, Design Observer, Urban Activist, Design and Culture, Spark Journal, Interiors: Design/Architecture/Culture, Futures Worth Preserving Cultural Constructions of Nostalgia and Sustainability

Industry Experience

Founder
GreenspaceNYC

Senior Interactive Designer
Marketing @ The New School

Designer & Developer
Marketing @ Housing Works

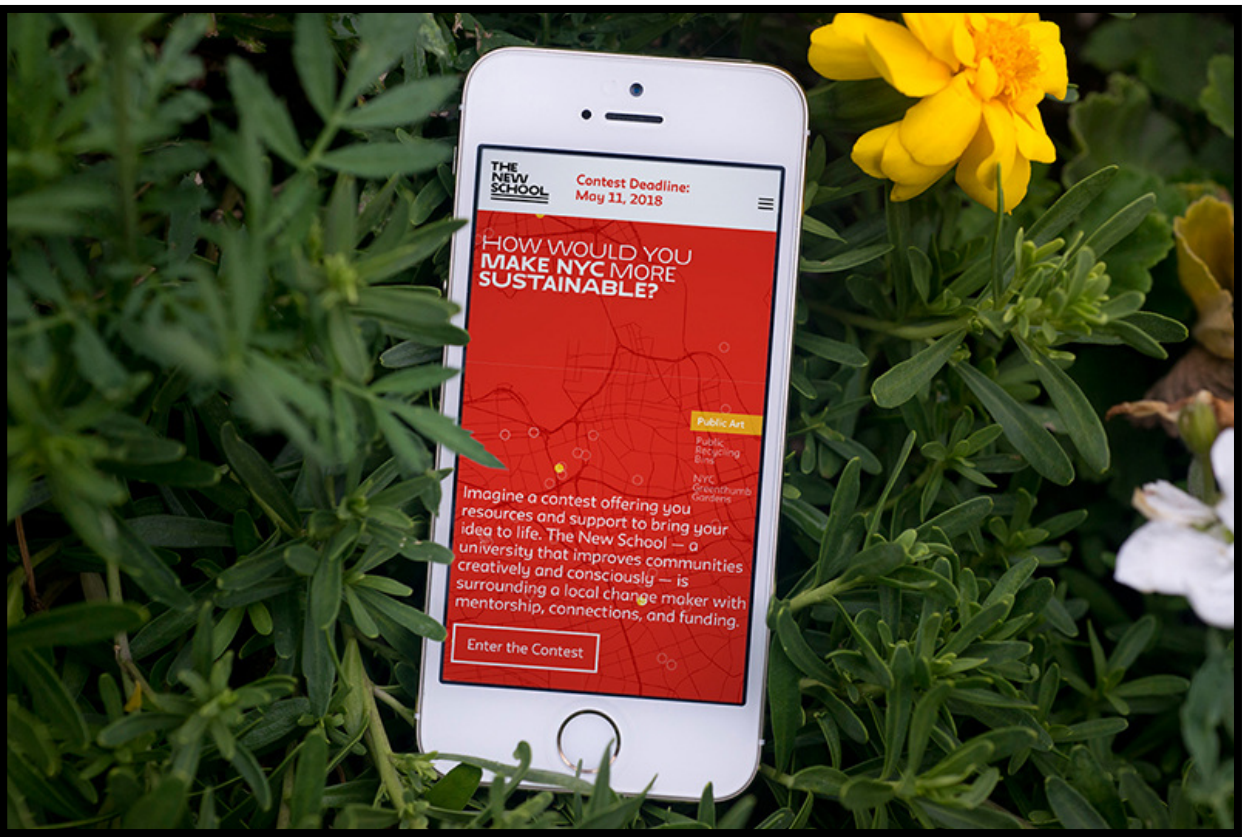
Designer & Developer
Motion Graphics, Advertising @ Guava Studios

Notable clients

HBO, NHL, Ogilvy, Stanford Medicine, Sesame Workshop, EWG, PBS, Barnes & Noble

Laura Scherling, tools used in example projects

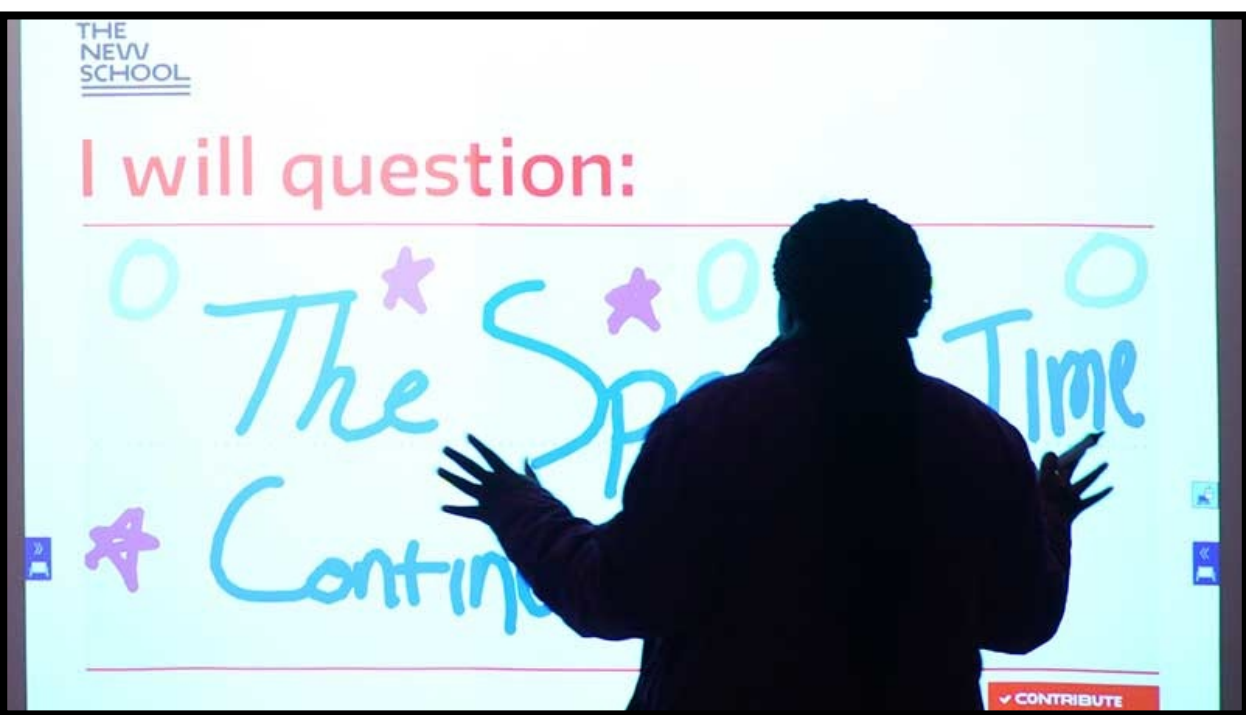
javascript



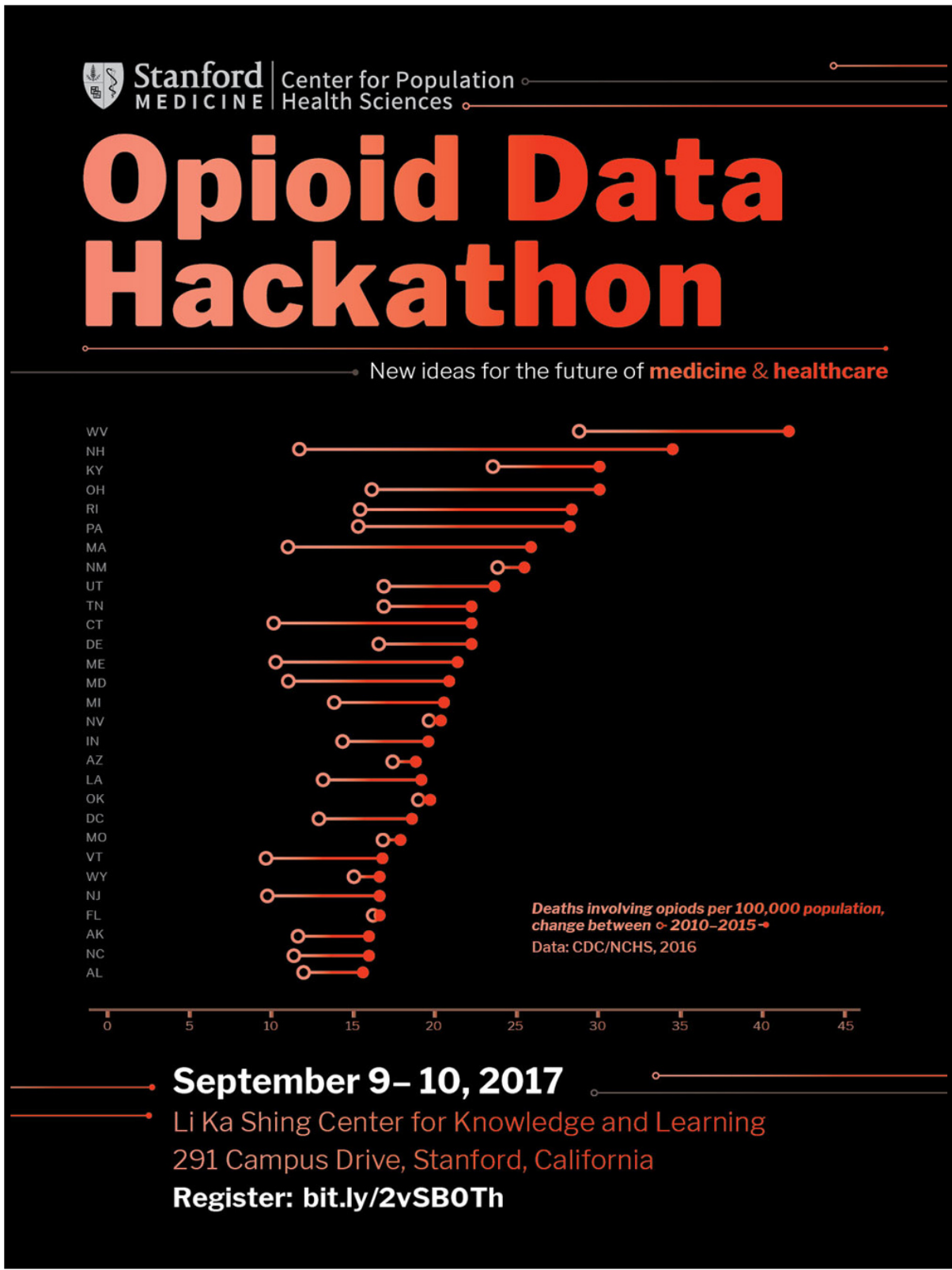
python



Excel



In Design



Illustrator

Media in small and large scale for mapping and infographics projects



photography

introductions, weekly discussion and office hours

weekly, in-person discussion

Class meets Mondays 6:10-8PM
Riverside Church, Assembly Hall

office hours

Professor Scott Spencer
[Click to schedule appointment](#)

Associate Laura Scherling
[Click to schedule appointment](#)

introductions, learning as a team — introductions through *group* resumes

In **groups of six**, appoint **one speaker to summarize:**

tools you (all) have used for data visualization,

previous **education** majors,

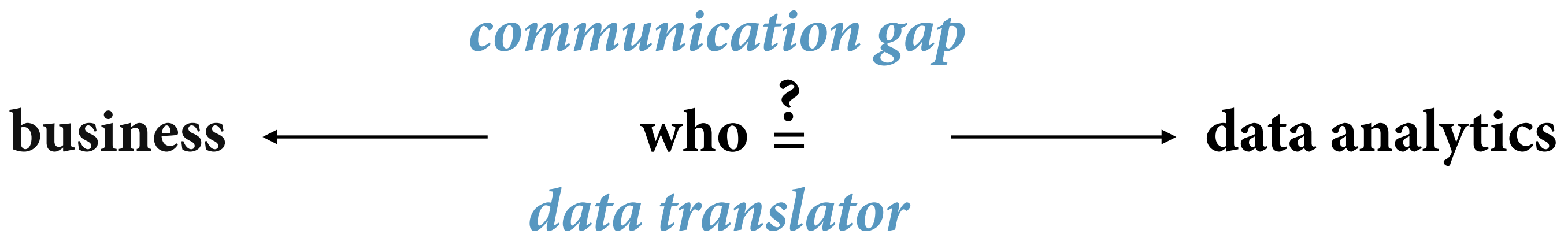
relevant (work) **experience**, and

hobbies.

Do this as quickly as possible. Several groups will have the opportunity to share out in **one minute elevator pitches.**

**analytics communication challenges
and course overview**

challenges, communication gaps



challenges, bridging the gaps with data translators, qualities needed

project management

data wrangling

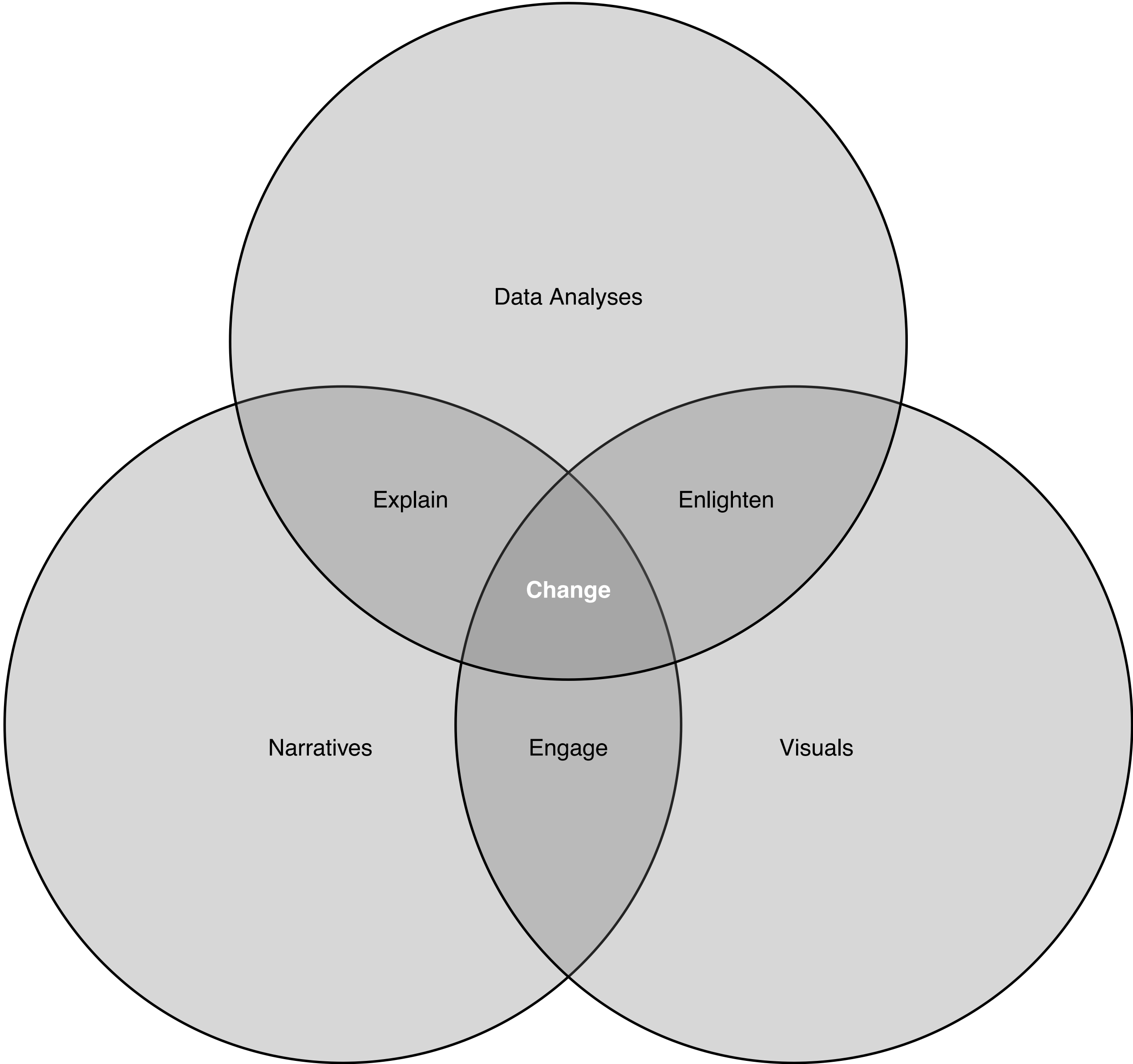
data analysis

subject expertise

design

storytelling

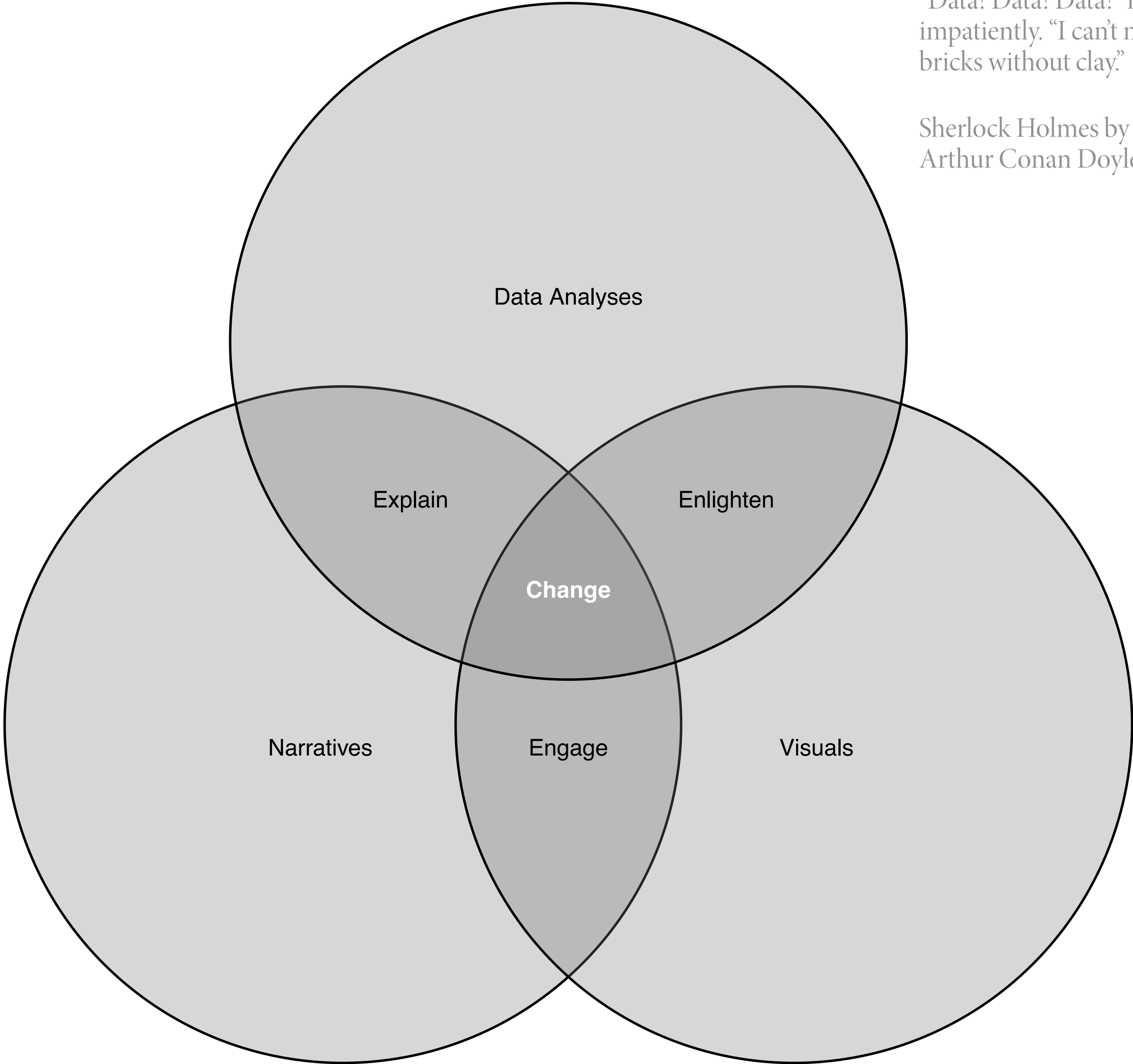
course overview, learn to drive change using data visuals and narrative



course overview, learn to drive change using data visuals and narrative

“Data! Data! Data!” he cried impatiently. “I can’t make bricks without clay.”

Sherlock Holmes by Sir Arthur Conan Doyle, *author*



No one ever made a decision because of a number. They need a story.

Daniel Kahneman, *psychologist, behavioral economist, and author*

The greatest value of a picture is when it forces us to notice what we never expected to see.

John W Tukey, *mathematician*

course overview, course deliverables and structure

Individual Work

For learning data visualization and written narrative techniques

Group work

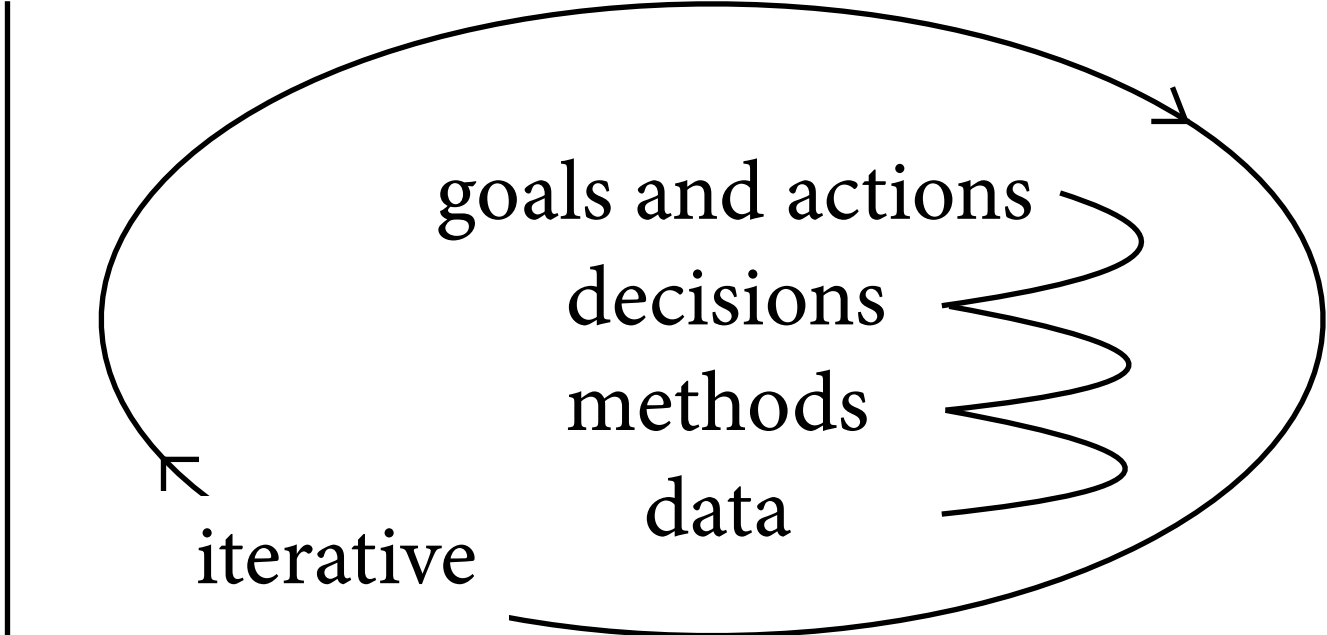
For building graphics and narrative into interactive communications

Sept 30	Oct 14	Oct 28	Nov 18	Nov 18	Dec 11	Dec 13
Homework 1 graphics	Homework 2 graphics	Homework 3 writing	Homework 4 graphics	Proposal	Interactive Communication	Multimodal communication
10%	10%	10%	10%	15%	20%	15%
Participation 10%						

introducing an example case study

example case study, let's begin a project (indeed, *any* data analytics project) with questions

process



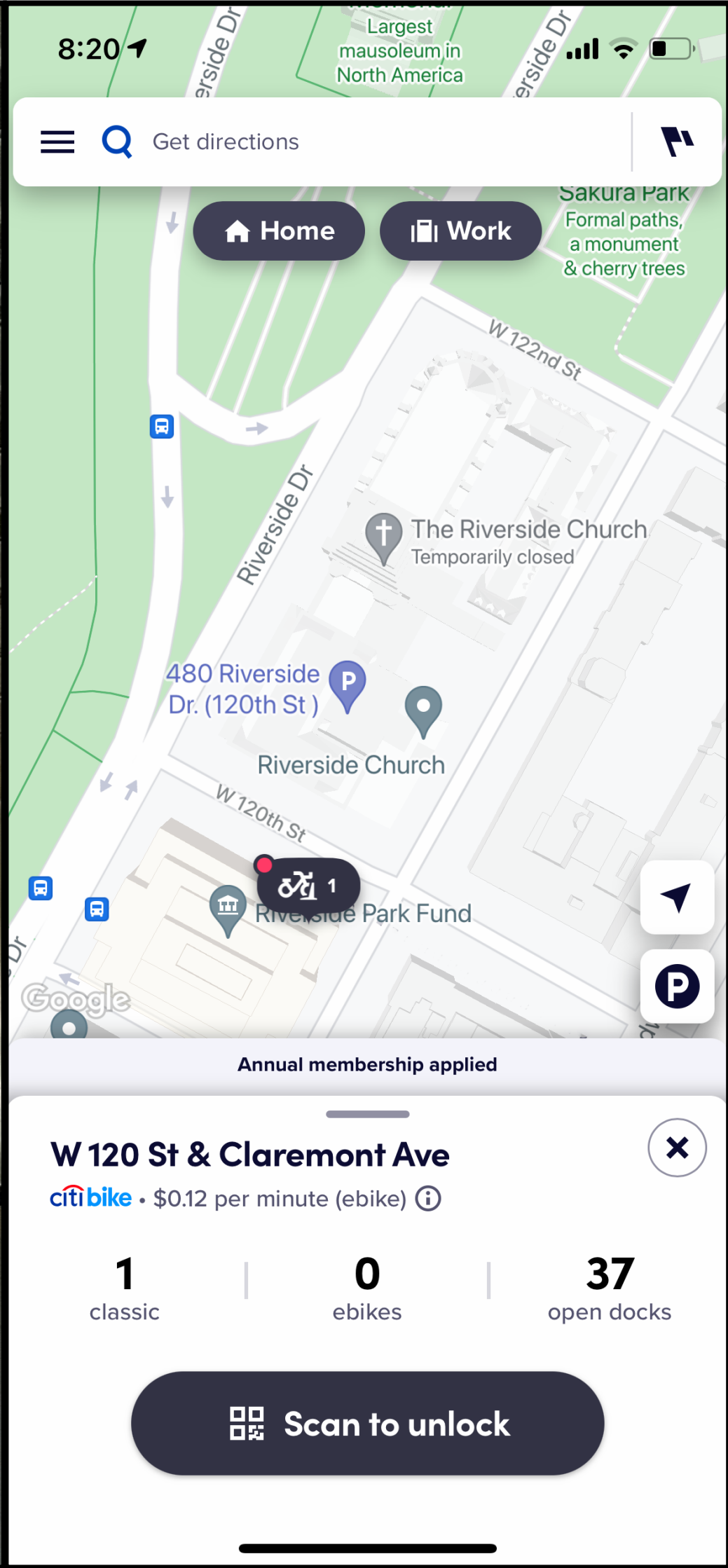
initial questions

- What **problem** is to be solved?
- Is the problem **important**?
- Could an answer have **impact**?
- Do **data** have a role in **solving the problem**?
- Are the right **data available**?
- In what **contexts** may the data be generated?
- Is the **organization** ready to **tackle the problem** and **take actions from insights**?

example case study, exploring changes in usage of Citi Bike, a bike share in New York City



example case study, docking station outside our classroom and related information on iPhone app



example case study, rebalancing as a challenge



Rebalancing is one of the **biggest challenges** of any bike share system, especially in ... New York where residents don't all work a traditional 9-5 schedule, and ... people work in a variety of other neighborhoods.

— Simmons, Dani. Citi Bike spokeswoman. 2016

example case study, changing conditions, how might the pandemic affect *current* rebalancing efforts?

The New York Times | PLAY THE CROSSWORD | Account

How Coronavirus Has Changed New York City Transit, in One Chart

By Veronica Penney | March 8, 2021

At 8:30 on weekday mornings, there is now enough space in the Main Concourse at Grand Central Terminal for travelers to walk at least six feet apart. Most move at a stroll rather than the New York City speedwalk. Early last year, about 160,000 people passed through the station each weekday. Ridership is now less than a quarter of that.

The pandemic has profoundly disrupted the largest public transit system in America, throwing it into financial turmoil. But getting more people on public transportation will be a crucial component of New York City's plan to become carbon neutral by 2050. The system needs to grow — right at a time when it is facing a sharp decline in ridership and revenue.

Month	Bridge and tunnel traffic	Buses	Subways	LIRR	Metro-North
March 2020	0	0	0	0	0
May 2020	-40	-60	-70	-80	-85
July 2020	-25	-45	-65	-75	-80
September 2020	-20	-40	-60	-70	-75
November 2020	-15	-35	-55	-65	-70
March 2021	-10	-30	-50	-60	-65

By Veronica Penney - Source: The M.T.A.'s day-by-day ridership numbers. | Percent change is calculated as a comparison with the preceding-year equivalent day, with the exception of the commuter rail systems, which are compared with the 2019 monthly weekday/Saturday/Sunday average. The M.T.A. began reporting data for the LIRR and Metro-North on April 1, 2020.

Subway rides, bus rides and car trips in New York City fell drastically last March as coronavirus cases surged and the city entered a mandatory lockdown. Some residents who could afford to [left the city](#) for second homes or rentals in the suburbs. Many employees switched to remote work and have not yet returned to their offices.

Keeping the city's buses and subways moving has been crucial for transporting medical and essential workers, but, with fewer riders,

For New York City to hit its climate goals, it **will be critical for more people to use public transit, bikes or walking to commute than before the pandemic.** When offices and businesses begin to reopen, more flexible remote options for workers could also be friendly for the planet.

Transit experts also say that existing tools and policies could encourage commuters to embrace low-emissions modes of transportation. **Bike shares and bike sales are experiencing a boom in the city,** which could help reduce transit emissions, but cycling advocates say continued investment in bike paths and protected lanes will be key for keeping people on their bikes as commuting returns to its post-pandemic normal.

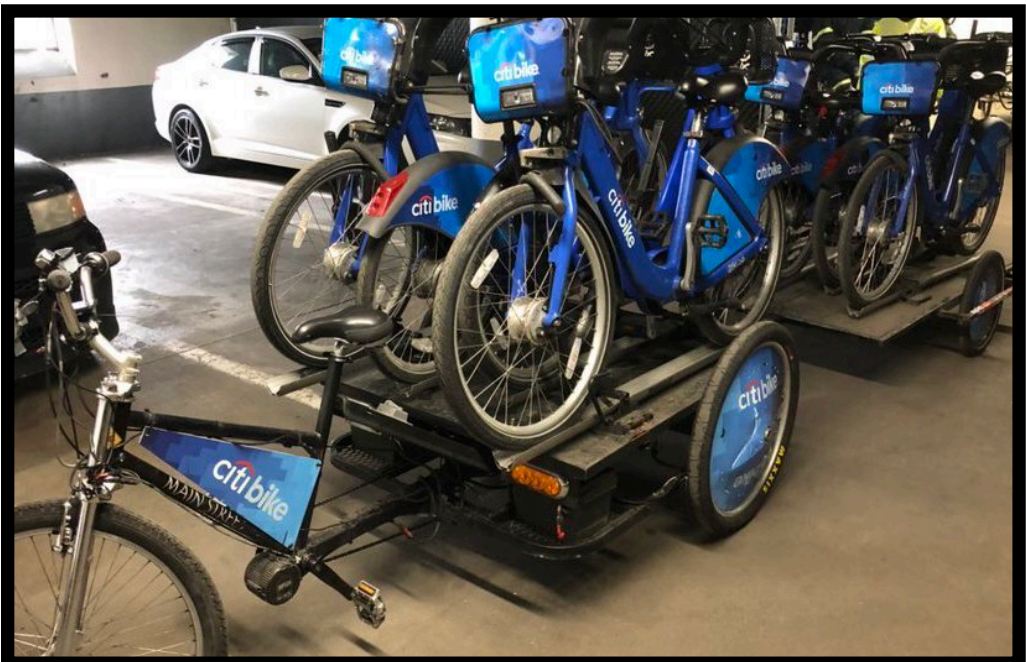
— Penney, Veronica. New York Times. 2021

example case study, Citi Bike's approaches to rebalancing

Bike Angels



Bike trains



Valet schedule

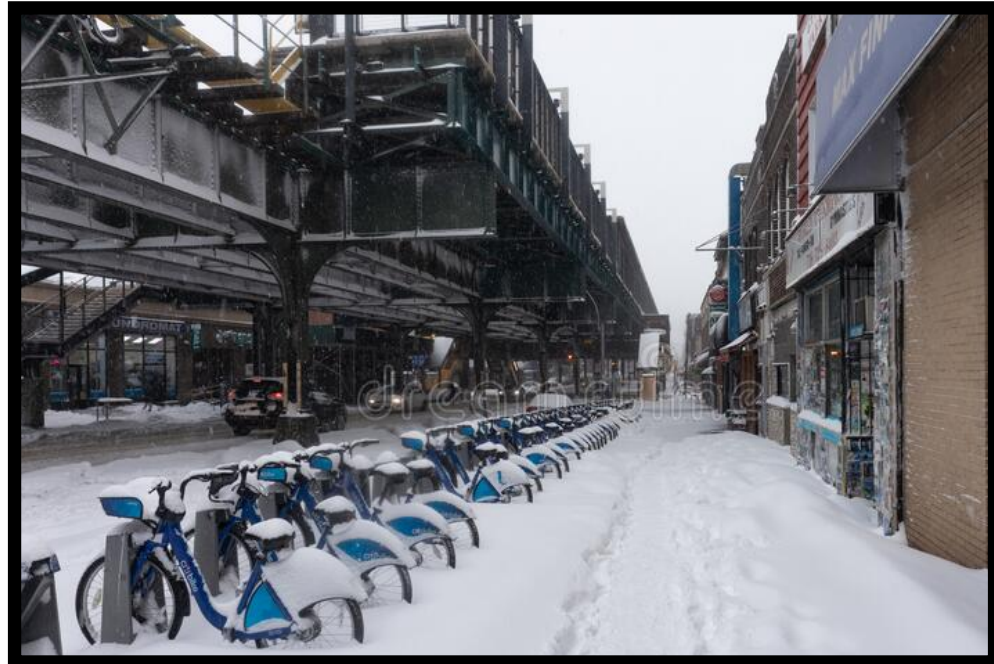


Motorized vehicles



— Citi Bike. “How We’re Rebalancing the Citi Bike System.” *Citi Bike NYC* (blog), August 14, 2020. <https://www.citibikenyc.com/blog/rebalancing-the-citi-bike-system>.

example case study, think about the tangible world, then how relevant events may be measured and recorded



Identifying events and user behavior

What **events** may be correlated with or cause empty or full bike docking stations?

What potential **user behaviors** or **preferences** may lead to these events?

From what **analogous** things could we draw **comparisons** to provide **context**?

Measurements of events and behaviors

How may these events and behaviors have been **measured and recorded**?

What **data are available**? Where?
What form?

May these data be **sufficient to find insights** through analysis, useful for decisions and goals?

data, a basic taxonomy

data for analytics projects, defining **datum** and **data set**

DATUM | an abstraction of a real-world entity (person, object, or event). The terms *variable*, *feature*, and *attribute* are often used interchangeably to denote an individual abstraction. **Data** are the plural of datum.

DATA SET | consists of the **data** relating to a collection of entities, with each entity described in terms of a set of attributes. In its most basic form, a data set is organized in an $n \cdot m$ data matrix called the analytics record, where n is the number of entities (rows) and m is the number of attributes (columns).

NOMINAL types are *names* for categories, classes, or states of things.

ORDINAL types are similar to nominal types, except it is possible to *rank or order* categories of an ordinal type.

NUMERIC types are *measurable* quantities we can represent using integer or real values. Numeric types can be measured on an *interval* scale or a *ratio* scale.

data for analytics projects, **structured** and **unstructured** data

STRUCTURED DATA | data that can be stored in a table, and every instance in the table has the same structure (i.e., set of attributes).

UNSTRUCTURED DATA | data where each instance in the data set may have its own internal structure, and this structure is not necessarily the same in every instance.

example case study, research and explore possible sources of available data

Examples of publicly available data sources



Bike and dock: data are recorded of each bike unlocked and docked, along with remaining dock capacities at the locations, dates, and times of each event: <https://www.citibikenyc.com/system-data>

Taxi pickup and drop-off locations and times: <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

Subway lines entrance/exit locations: <https://data.cityofnewyork.us/Transportation/Subway-Stations/arq3-7z49>

MTA usage data (and change since coronavirus): <https://new.mta.info/coronavirus/ridership>

Historical weather: National Weather Service. *API Web Service*. <https://www.weather.gov/documentation/services-web-api>; Yip, Stan. "Weatherr: Tools for Handling and Scraping Instant Weather Forecast Feeds." *Manual*, 2020. <https://CRAN.R-project.org/package=weatherr>.

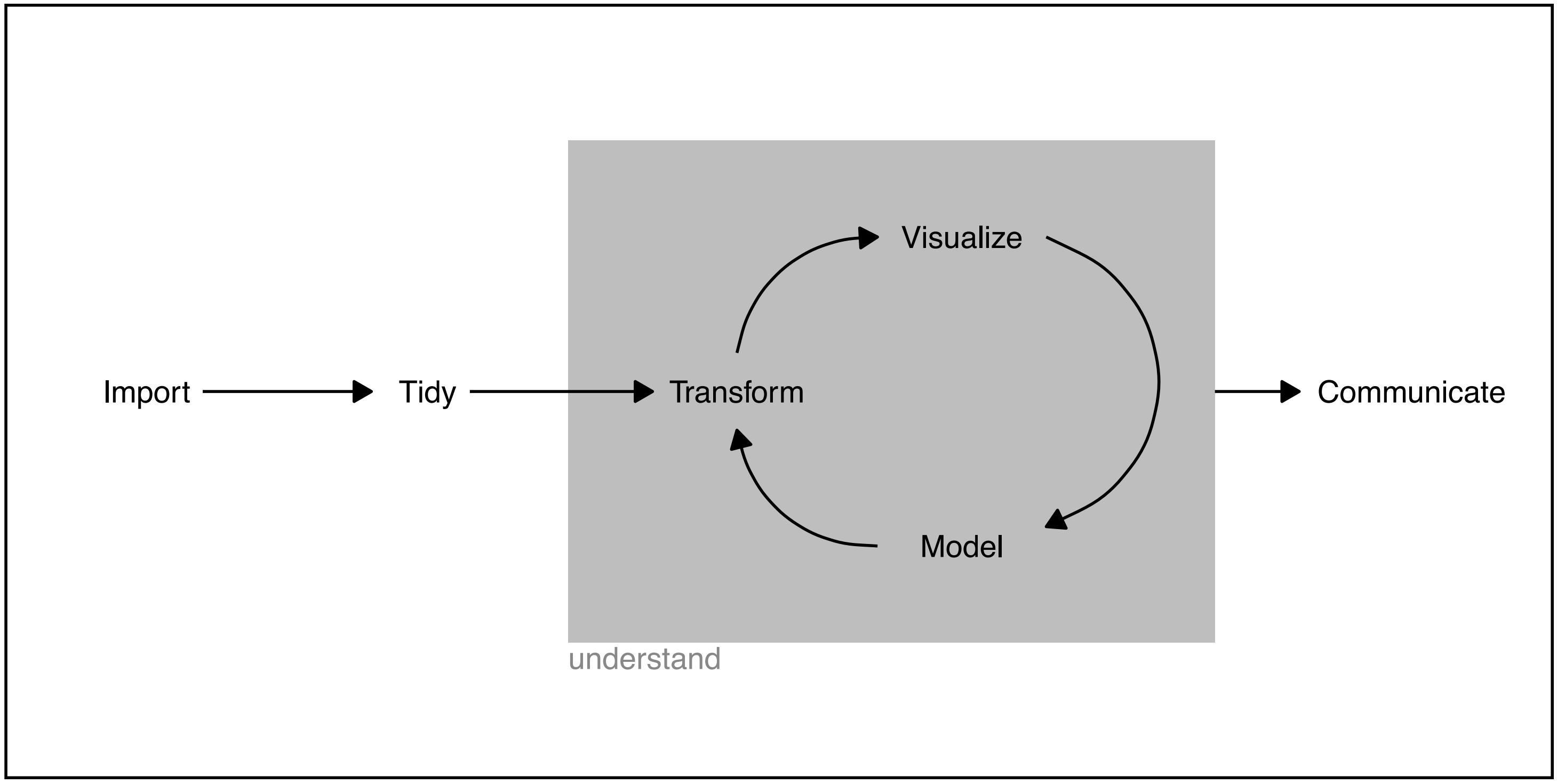
Geography (elevation): USGS. *Elevation Point Query Service*. <https://ned.usgs.gov/epqs/>; Hollister, Jeffrey, et al. "Elevatr: Access Elevation Data from Various APIs." *Manual*, 2021. <https://doi.org/10.5281/zenodo.5119662>.

Traffic data and more: <http://www.nyc.gov/html/dot/html/about/datafeeds.shtml#realtime>

Twitter: <https://twitter.com/CitiBikeNYC>; Gentry, Jeff. "Twitter: R Based Twitter Client." *Manual*, 2015. <https://CRAN.R-project.org/package=twitter>.

**workflow, software tools for
data exploration and analysis**

explore & analyze, a basic, general workflow



program

— Adapted from Wickham, Hadley, and Garrett Golemund. *R for Data Science*. <https://r4ds.had.co.nz>

software tools, demonstrations in R, can be mimicked in Python

R / tidyverse

Python / datar & plotnine

software tools, demonstrations in R, can be mimicked in Python — loading libraries

R / tidyverse

Python / datar & plotnine

```
library(package_name)
```

```
from package_name import *
```

software tools, demonstrations in R, can be mimicked in Python — importing data

R / tidyverse

Python / datar & plotnine

```
df_r <- read_csv("filename.csv")
```

```
df_py = pd.read_csv("filename.csv")
```

software tools, demonstrations in R, can be mimicked in Python — data structures

R

Python

```
1; 1L; TRUE; "foo"
```

single-element vector



scalar

```
c(1.0, 2.0, 3.0); c(1L, 2L, 3L)
```

multi-element vector



list

```
list(1L, TRUE, "foo")
```

list of multiple types



tuple

```
list(a = 1L, b = TRUE, c = "foo")
```

named list



dict

```
matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)
```

matrix / array



numpy ndarray

```
data.frame(x = 1:3, y = c("a", "b", "c"))
```

data frame



pandas DataFrame

```
function(x) x + 1
```

function



python function

```
NULL; TRUE; FALSE
```

NULL, TRUE, FALSE



None, True, False

software tools, demonstrations in R, can be mimicked in Python — how functions work

R / tidyverse

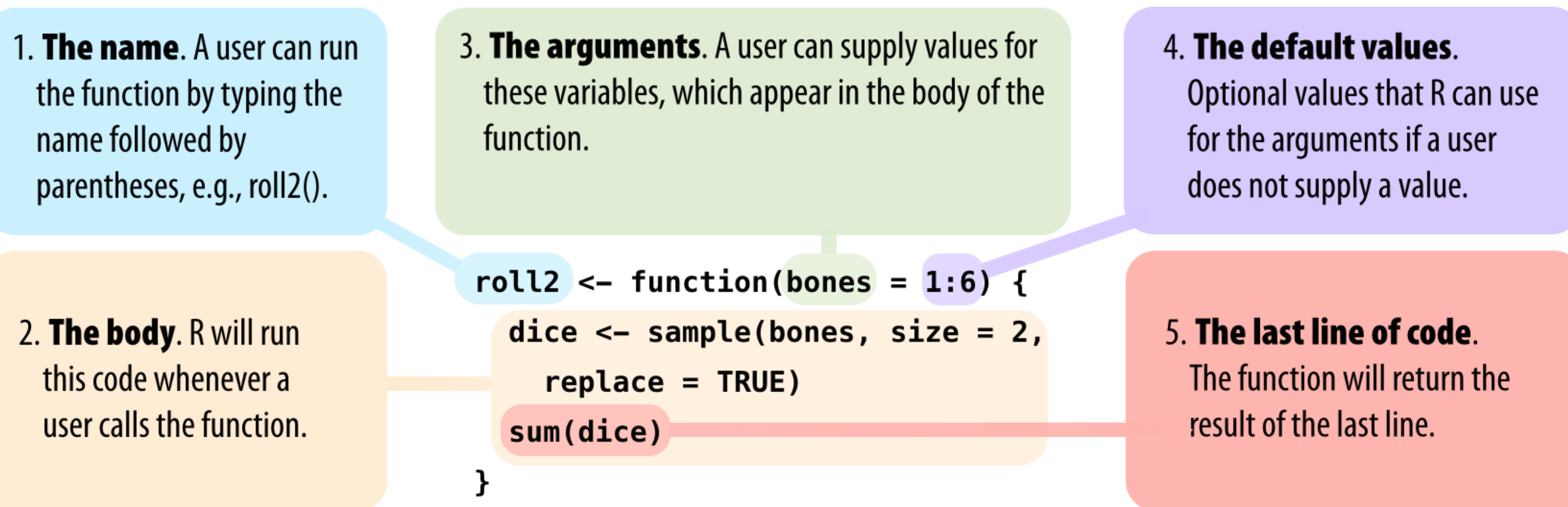
Python / datar & plotnine

```
# basic function
returned_object <- function_name(parameter1, parameter2)

# object-oriented member function
final_object <- initial_object$member_function()
```

```
#basic function
returned_object = function_name(parameter1, parameter2)

# object-oriented member function
final_object = initial_object.member_function()
```



software tools, demonstrations in R, can be mimicked in Python — applying successive functions to data frame

R / tidyverse
pipe operator %>%

Python / datar & plotnine
pipe operator >>

```
final_object <- initial_object %>%  
  f() %>%  
  g() %>%  
  h()
```

```
final_object = initial_object >> \  
  f() >> \  
  g() >> \  
  h()
```

software tools, demonstrations in R, can be mimicked in Python — Python library ports of R's dplyr & ggplot2

R / tidyverse

Python / datar & plotnine

shows data frame variables & types

`glimpse`



`info`

creates or modifies a variable

`mutate`



`mutate`

specifies variables (columns) to keep

`select`



`select`

renames variables

`rename`



`rename`

specifies observations (rows) to keep

`filter`



`filter`

specifies ordering of observations

`arrange`



`arrange`

specifies grouping of observations

`group_by`



`group_by`

specifies some summary of the data

`summarise`



`summarise`

convert form wide to long format

`pivot_longer`



`pivot_longer`

convert form long to wide format

`pivot_wider`



`pivot_wider`

map data to graphical elements

`ggplot`



`ggplot`

example case study, exploring data with software

R / tidyverse

```
library(tidyverse)
```

Python / datar & plotnine

```
import pandas as pd
from plotnine import *
from datar.all import *
from pipda import options
options.assume_all_piping = True
```

R / tidyverse

Python / datar & plotnine

```
df_r <- read_csv("data/201901-citibike-tripdata.csv")
df_r %>% glimpse()
```

```
Rows: 967,287
Columns: 15
$ tripduration      <dbl> 320, 316, 591, 2719, 303, 535, 280...
$ starttime         <dtm> 2019-01-01 00:01:47, 2019-01-01 0...
$ stoptime          <dtm> 2019-01-01 00:07:07, 2019-01-01 0...
$ `start station id` <dbl> 3160, 519, 3171, 504, 229, 3630, 3...
$ `start station name` <chr> "Central Park West & W 76 St", "Pe...
$ `start station latitude` <dbl> 40.77897, 40.75187, 40.78525, 40.7...
$ `start station longitude` <dbl> -73.97375, -73.97771, -73.97667, -...
$ `end station id` <dbl> 3283, 518, 3154, 3709, 503, 3529, ...
$ `end station name` <chr> "W 89 St & Columbus Ave", "E 39 St...
$ `end station latitude` <dbl> 40.78822, 40.74780, 40.77314, 40.7...
$ `end station longitude` <dbl> -73.97042, -73.97344, -73.95856, -...
$ bikeid            <dbl> 15839, 32723, 27451, 21579, 35379,...
$ usertype          <chr> "Subscriber", "Subscriber", "Subsc...
$ `birth year`     <dbl> 1971, 1964, 1987, 1990, 1979, 1989...
$ gender            <dbl> 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2...
```

```
df_py = pd.read_csv("data/201901-citibike-tripdata.csv")
df_py.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 967287 entries, 0 to 967286
Data columns (total 15 columns):
#   Column                Non-Null Count  Dtype
---  -
0   tripduration          967287 non-null  int64
1   starttime              967287 non-null  object
2   stoptime               967287 non-null  object
3   start station id      967269 non-null  float64
4   start station name    967269 non-null  object
5   start station latitude 967287 non-null  float64
6   start station longitude 967287 non-null  float64
7   end station id        967269 non-null  float64
8   end station name      967269 non-null  object
9   end station latitude  967287 non-null  float64
10  end station longitude  967287 non-null  float64
11  bikeid                 967287 non-null  int64
12  usertype               967287 non-null  object
13  birth year            967287 non-null  int64
14  gender                 967287 non-null  int64
dtypes: float64(6), int64(4), object(5)
memory usage: 110.7+ MB
```

R / tidyverse

```
df_r <- df_r %>% rename_all(function(x) gsub(" ", "_", x))

df_r <- df_r %>%
  filter(!is.na(start_station_id)) %>%
  arrange(starttime) %>%
  group_by(bikeid) %>%
  mutate(
    rebalanced =
      if_else(row_number() > 1 &
              start_station_id != lag(end_station_id),
              TRUE, FALSE)
  ) %>%
  ungroup()

df_r %>% pull(rebalanced) %>% table()
```

Python / datar & plotnine

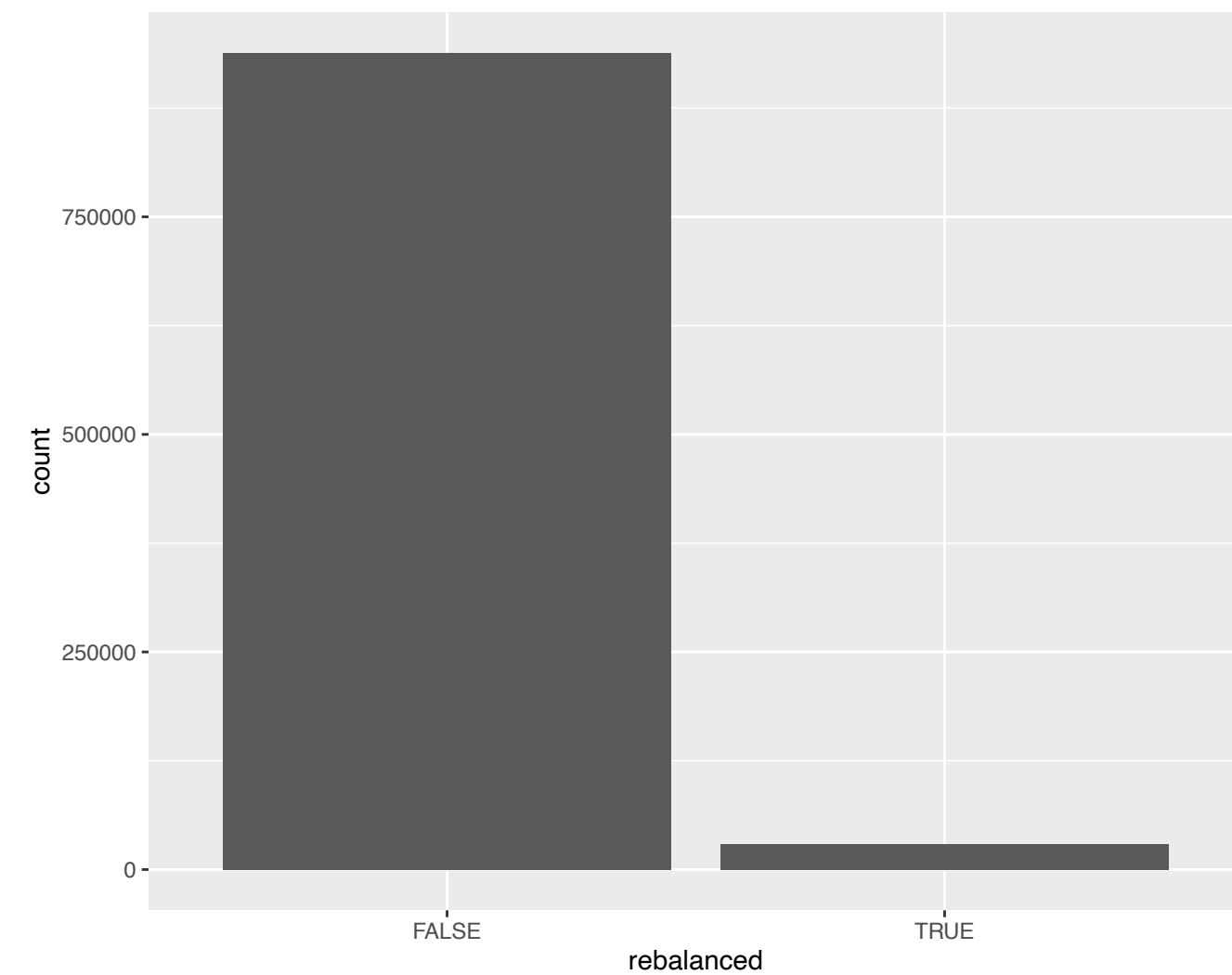
```
df_py = df_py.rename(lambda x: x.replace(' ', '_'), axis = 1)

df_py = df_py >> \
  filter( f.start_station_id.notnull() ) >> \
  arrange(f.starttime) >> \
  group_by(f.bikeid) >> \
  mutate(
    rebalanced =
      if_else((row_number() > 1) &
              (f.start_station_id != lag(f.end_station_id)),
              True, False)
  ) >> \
  ungroup()

df_py.rebalanced.value_counts()
```

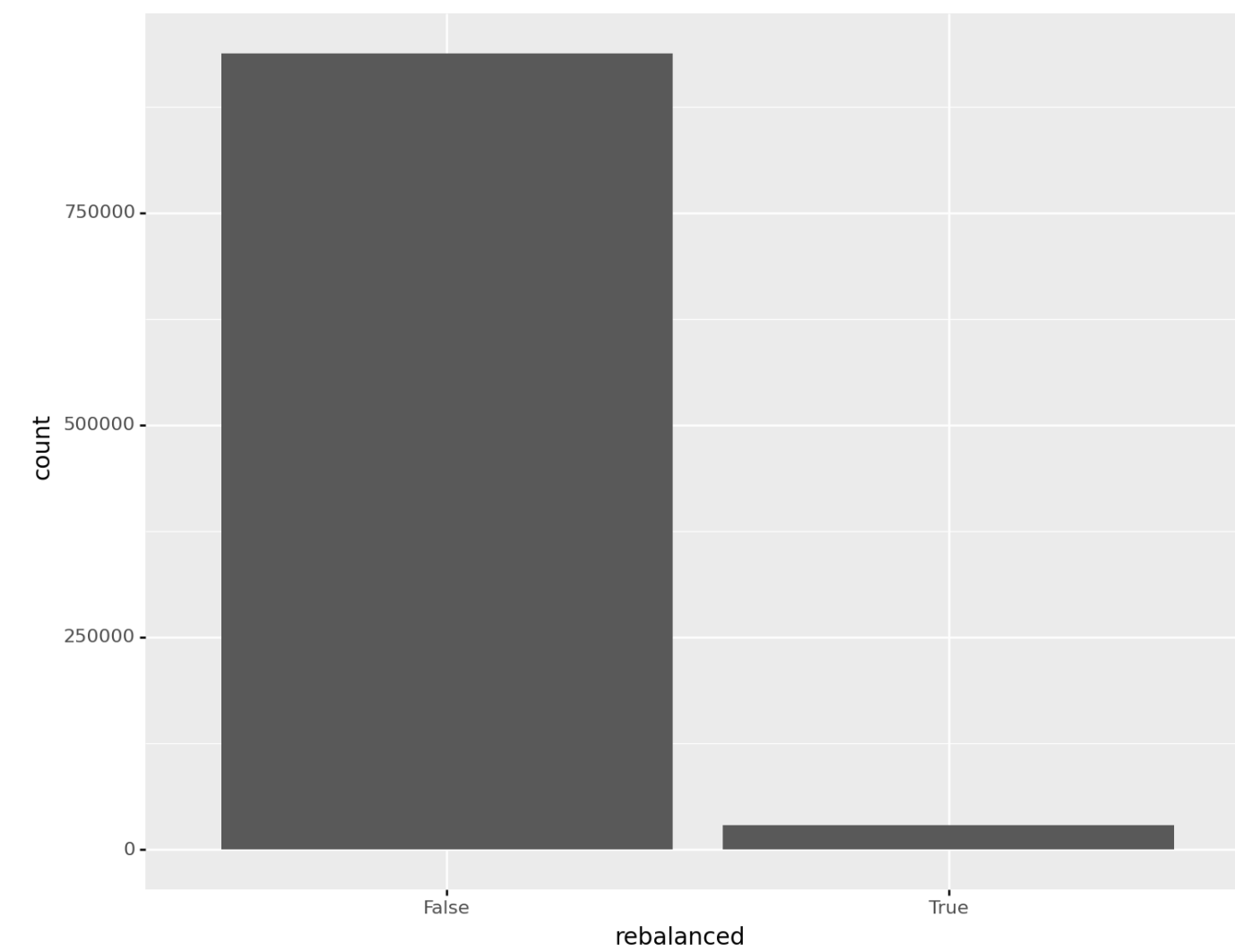
R / tidyverse

```
ggplot(data = df_r) +  
  geom_bar(  
    mapping = aes(x = rebalanced),  
    stat = 'count'  
  )
```

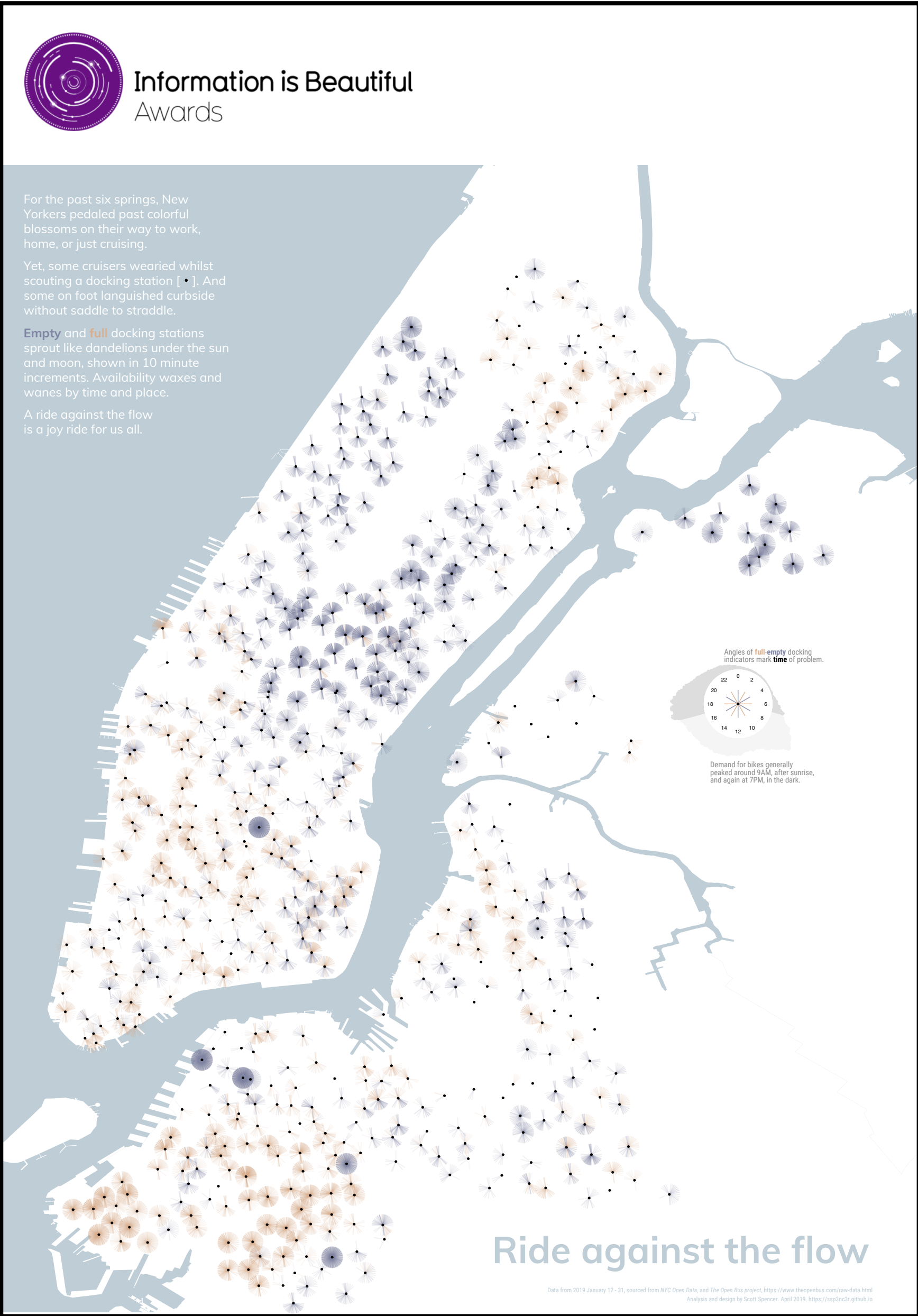


Python / datar & plotnine

```
ggplot(data = df_py) + \  
  geom_bar(  
    mapping = aes(x = 'rebalanced'),  
    stat = 'count'  
  )
```



later in the course, example of an *interactive information graphic* with our Citi Bike case study



click for interactive version

later in the course, example combining tools to create part of an *interactive communication* ...

Explorable differences between umpire calls and modeled probabilities of strikes

Below, we can explore the percentage difference between the umpire calls and modeled probabilities of those calls, with corresponding video footage. Circles \circ in the **top row** represent called pitches (**ball**, **strike**) and lightness of the color represent how close δ our modeled strike probability was to the *actual* call $\{0, 1\}$ for each handedness (throw-stand) matchup $\{LL, LR, RL, RR\}$.

Circles \circ in the **bottom row** represent the corresponding estimate of *runs saved* from that variation.

Hovering a pointer over a circle \circ links \circ — \circ pitches in top and bottom rows, and provides more details of the play in a tooltip. **Clicking** a pitch loads its game video:

Of note, the graphics below only show pitches where the differences of called **ball** or **strike** from modeled probability of strikes exceed ± 0.10 .

Clicking one of the above observed pitches loads its video:

Explorable differences between umpire calls and modeled probabilities of strikes

Below, we can explore the percentage difference between the umpire calls and modeled probabilities of those calls, with corresponding video footage. Circles \circ in the **top row** represent called pitches (**ball**, **strike**) and lightness of the color represent how close δ our modeled strike probability was to the *actual* call $\{0, 1\}$ for each handedness (throw-stand) matchup $\{LL, LR, RL, RR\}$.

Circles \circ in the **bottom row** represent the corresponding estimate of *runs saved* from that variation.

Hovering a pointer over a circle \circ links \circ — \circ pitches in top and bottom rows, and provides more details of the play in a tooltip. **Clicking** a pitch loads its game video:

Of note, the graphics below only show pitches where the differences of called **ball** or **strike** from modeled probability of strikes exceed ± 0.10 .

Clicking one of the above observed pitches loads its video:

Explorable differences between umpire calls and modeled probabilities of strikes

Below, we can explore the percentage difference between the umpire calls and modeled probabilities of those calls, with corresponding video footage. Circles \circ in the **top row** represent called pitches (**ball**, **strike**) and lightness of the color represent how close δ our modeled strike probability was to the *actual* call $\{0, 1\}$ for each handedness (throw-stand) matchup $\{LL, LR, RL, RR\}$.

Circles \circ in the **bottom row** represent the corresponding estimate of *runs saved* from that variation.

Hovering a pointer over a circle \circ links \circ — \circ pitches in top and bottom rows, and provides more details of the play in a tooltip. **Clicking** a pitch loads its game video:

Of note, the graphics below only show pitches where the differences of called **ball** or **strike** from modeled probability of strikes exceed ± 0.10 .

Clicking one of the above observed pitches loads its video:

Software: SQL, R (+ ggplot2 & other packages), Stan, HTML, CSS, javascript, D3.js, htmlwidgets

next deliverable, homework one

Individual Work

For learning data visualization and written narrative techniques

Group work

For building graphics and narrative into interactive communications

Sept 30	Oct 14	Oct 28	Nov 18	Nov 18	Dec 11	Dec 13
Homework 1 graphics	Homework 2 graphics	Homework 3 writing	Homework 4 graphics	Proposal	Interactive Communication	Multimodal communication
10%	10%	10%	10%	15%	20%	15%
Participation 10%						

resources

References

Brady, Chris, Mike Forde, and Simon Chadwick. “*Why Your Company Needs Data Translators.*” MIT Sloan Management Review, March 2017, 1–6.

Berinato, Scott. “*Data Science & the Art of Persuasion.*” Harvard Business Review, December 2018, 1–13.

Citi Bike. “How We’re Rebalancing the Citi Bike System.” *Citi Bike NYC* (blog), August 14, 2020. <https://www.citibikenyc.com/blog/rebalancing-the-citi-bike-system>.

Friedman, Matthew. “*Citi Bike Racks Continue to Go Empty Just When Upper West Siders Need Them.*” News. West Side Rag (blog), August 19, 2017. <https://www.westsiderag.com/2017/08/19/citi-bike-racks-continue-to-go-empty-just-when-upper-west-siders-need-them>.

Kibirige, Hassan, et al. *Plotnine: A Grammar of Graphics for Python* (version v0.8.0), Last accessed 2021 September 18. <https://plotnine.readthedocs.io/en/stable/>.

Kelleher, John D, and Brendan Tierney. “What Are Data, and What Is a Data Set?” In *Data Science*. MIT Press, 2018.

Mailund, Thomas. *Beginning Data Science in R*. Apress, 2017.

Maynard-Atem, Louise, and Ben Ludford. “*The Rise of the Data Translator.*” *Impact* 2020, no. 1 (January 2, 2020): 12–14. <https://doi.org/10.1080/2058802X.2020.1735794>.

Penney, Veronica. *How Coronavirus Has Changed New York City Transit, in One Chart*. New York Times, March 8, 2021. <https://www.nytimes.com/interactive/2021/03/08/climate/nyc-transit-covid.html>

Spencer, Scott. “Analytics Communication Scopes” and “Audiences and Challenges.” In *Data in Wonderland*. 2021. https://ssp3nc3r.github.io/data_in_wonderland.

Spencer, Scott. “Similarities between R and Python for data analysis.” Last updated 2020 Feb. 2. <https://ssp3nc3r.github.io/publications/Spencer-2020-r-python-similarities.html>

Wang, P.W. *datar: port of dplyr and other related R packages in python, using pipda*. (version 0.5.1), Last Accessed 2021 September 18. <https://pwwang.github.io/datar/>.

Wickham, Hadley, and Garrett Grolemund. *R for Data Science*. <https://r4ds.had.co.nz>

Wickham, Hadley, Danielle Navarro, and Thomas Lin Pedersen. *ggplot2: Elegant Graphics for Data Analysis*. Third Edition (in progress). <https://ggplot2-book.org>

supplemental

R / tidyverse

saving *each* step

```
Temp1 <- f(initial_object)
Temp2 <- g(Temp1)
final_object <- h(Temp2)
```

Python / pandas

saving *each* step

```
Temp1 = initial_object.f()
Temp2 = Temp1.g()
final_object = Temp2.h()
```

method chaining

```
final_object = (
    initial_object
    .f()
    .g()
    .h()
)
```

pipe operator %>%

```
final_object <-
  initial_object %>%
  f() %>%
  g() %>%
  h()
```

pipe operator >> from pipda

```
final_object = initial_object >> \
  f() >> \
  g() >> \
  h()
```

explore & analyze, similarities between software languages — functions operating on data frames

R / tidyverse

Python / pandas

shows data frame variables & types

`glimpse`



`info`

creates or modifies a variable

`mutate`



`assign`

specifies variables (columns) to keep

`select`



`filter`

renames variables

`rename`



`rename`

specifies observations (rows) to keep

`filter`



`query`

specifies ordering of observations

`arrange`



`sort_values`

specifies grouping of observations

`group_by`



`groupby`

specifies some summary of the data

`summarise`



`agg`

convert form wide to long format

`pivot_longer`



`melt`

convert form long to wide format

`pivot_wider`



`pivot`

R / tidyverse

Python / pandas

Data Wrangling with dplyr and tidyr Cheat Sheet



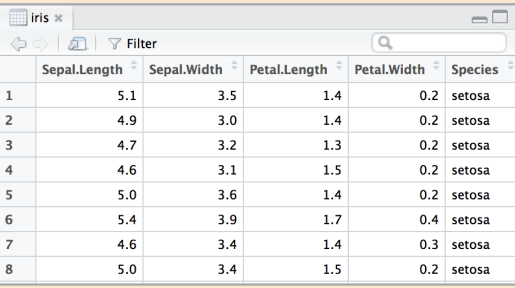
Syntax - Helpful conventions for wrangling

dplyr::tbl_df(iris)
Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1             5.1           3.5          1.4
2             4.9           3.0          1.4
3             4.7           3.2          1.3
4             4.6           3.1          1.5
5             5.0           3.6          1.4
...
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

dplyr::glimpse(iris)
Information dense summary of tbl data.

utils::View(iris)
View data set in spreadsheet-like display (note capital V).



dplyr::%>%
Passes object on left hand side as first argument (or argument) of function on righthand side.

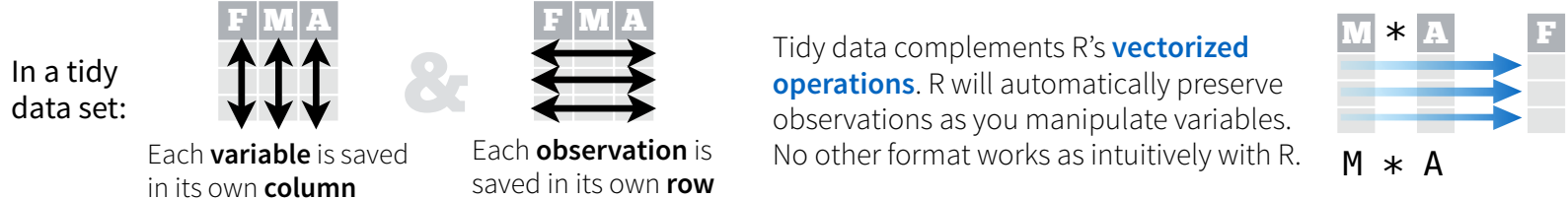
`x %>% f(y)` is the same as `f(x, y)`
`y %>% f(x, ., z)` is the same as `f(x, y, z)`

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

RStudio is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

Tidy Data - A foundation for wrangling in R



Reshaping Data - Change the layout of a data set

<p>tidyr::gather(cases, "year", "n", 2:4) Gather columns into rows.</p>	<p>tidyr::spread(pollution, size, amount) Spread rows into columns.</p>
<p>tidyr::separate(storms, date, c("y", "m", "d")) Separate one column into several.</p>	<p>tidyr::unite(data, col, ..., sep) Unite several columns into one.</p>

dplyr::data_frame(a = 1:3, b = 4:6)
Combine vectors into data frame (optimized).

dplyr::arrange(mtcars, mpg)
Order rows by values of a column (low to high).

dplyr::arrange(mtcars, desc(mpg))
Order rows by values of a column (high to low).

dplyr::rename(tb, y = year)
Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::filter(iris, Sepal.Length > 7)
Extract rows that meet logical criteria.

dplyr::distinct(iris)
Remove duplicate rows.

dplyr::sample_frac(iris, 0.5, replace = TRUE)
Randomly select fraction of rows.

dplyr::sample_n(iris, 10, replace = TRUE)
Randomly select n rows.

dplyr::slice(iris, 10:15)
Select rows by position.

dplyr::top_n(storms, 2, date)
Select and order top n entries (by group if grouped data).

Subset Variables (Columns)

dplyr::select(iris, Sepal.Width, Petal.Length, Species)
Select columns by name or helper function.

Helper functions for select - ?select

- select(iris, contains(" "))** - Select columns whose name contains a character string.
- select(iris, ends_with("Length"))** - Select columns whose name ends with a character string.
- select(iris, everything())** - Select every column.
- select(iris, matches("t"))** - Select columns whose name matches a regular expression.
- select(iris, num_range("x", 1:5))** - Select columns named x1, x2, x3, x4, x5.
- select(iris, one_of(c("Species", "Genus")))** - Select columns whose names are in a group of names.
- select(iris, starts_with("Sepal"))** - Select columns whose name starts with a character string.
- select(iris, Sepal.Length:Petal.Width)** - Select all columns between Sepal.Length and Petal.Width (inclusive).
- select(iris, -Species)** - Select all columns except Species.

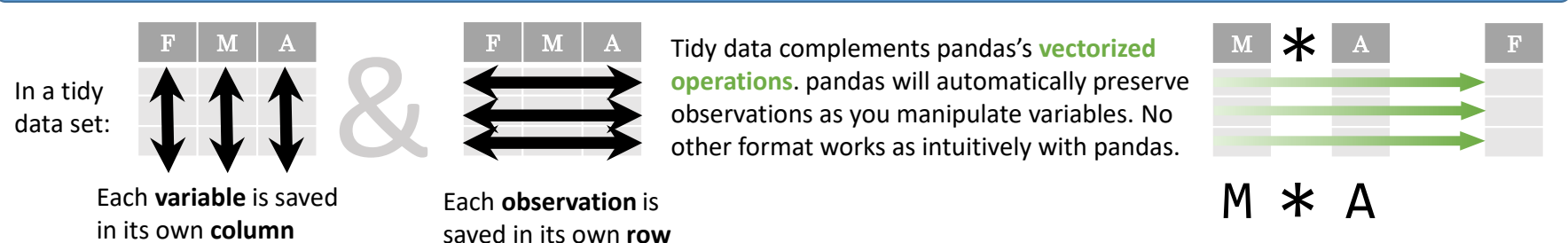
Logic in R - ?Comparison, ?base::Logic		
<	Less than	!=
>	Greater than	%in%
==	Equal to	is.na
<=	Less than or equal to	!is.na
>=	Greater than or equal to	&, , !, xor, any, all
		Boolean operators

devtools::install_github("rstudio/EDAWR") for data sets Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0 • tidyr 0.2.0 • Updated: 1/15

Data Wrangling with pandas Cheat Sheet

<http://pandas.pydata.org>

Tidy Data - A foundation for wrangling in pandas



Syntax - Creating DataFrames

```
df = pd.DataFrame(
  {"a" : [4, 5, 6],
   "b" : [7, 8, 9],
   "c" : [10, 11, 12]},
  index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
  [[4, 7, 10],
   [5, 8, 11],
   [6, 9, 12]],
  index=[1, 2, 3],
  columns=['a', 'b', 'c'])
```

Specify values for each row.

Method Chaining

```
df = pd.DataFrame(
  {"a" : [4, 5, 6],
   "b" : [7, 8, 9],
   "c" : [10, 11, 12]},
  index = pd.MultiIndex.from_tuples(
    [('d', 1), ('d', 2), ('e', 2)],
    names=['n', 'v']))
```

Create DataFrame with a MultiIndex

Logic in Python (and pandas)

Logic in Python (and pandas)		
<	Less than	!=
>	Greater than	df.column.isin(values)
==	Equals	pd.isnull(obj)
<=	Less than or equals	pd.notnull(obj)
>=	Greater than or equals	&, ~, ^, df.any(), df.all()

Reshaping Data - Change the layout of a data set

<p>pd.melt(df) Gather columns into rows.</p>	<p>df.pivot(columns='var', values='val') Spread rows into columns.</p>
<p>pd.concat([df1, df2]) Append rows of DataFrames</p>	<p>pd.concat([df1, df2], axis=1) Append columns of DataFrames</p>

df.sort_values('mpg')
Order rows by values of a column (low to high).

df.sort_values('mpg', ascending=False)
Order rows by values of a column (high to low).

df.rename(columns = {'y': 'year'})
Rename the columns of a DataFrame

df.sort_index()
Sort the index of a DataFrame

df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df.drop(columns=['Length', 'Height'])
Drop columns from DataFrame

Subset Observations (Rows)

df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.

df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

df.iloc[10:20]
Select rows by position.

df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.

Subset Variables (Columns)

df[['width', 'length', 'species']]
Select multiple columns with specific names.

df['width'] or **df.width**
Select single column with specific name.

df.filter(regex='regex')
Select columns whose name matches regular expression regex.

regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Lengths'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]\$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species\$).*'	Matches strings except the string 'Species'

df.loc[:, 'x2': 'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:, [1, 2, 5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a', 'c']]
Select rows meeting logical condition, and only the specific columns.