Lecture 4: Data Management

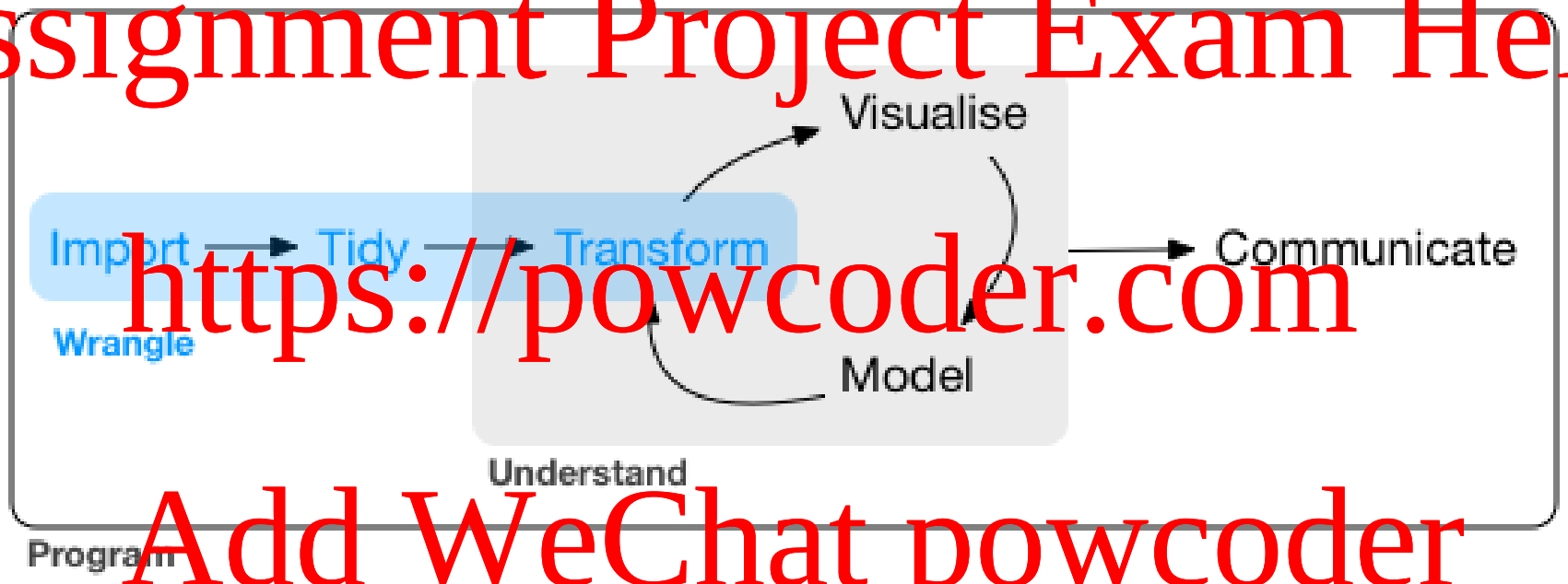
Spatial Data Science II

Dr. Adams

Overview

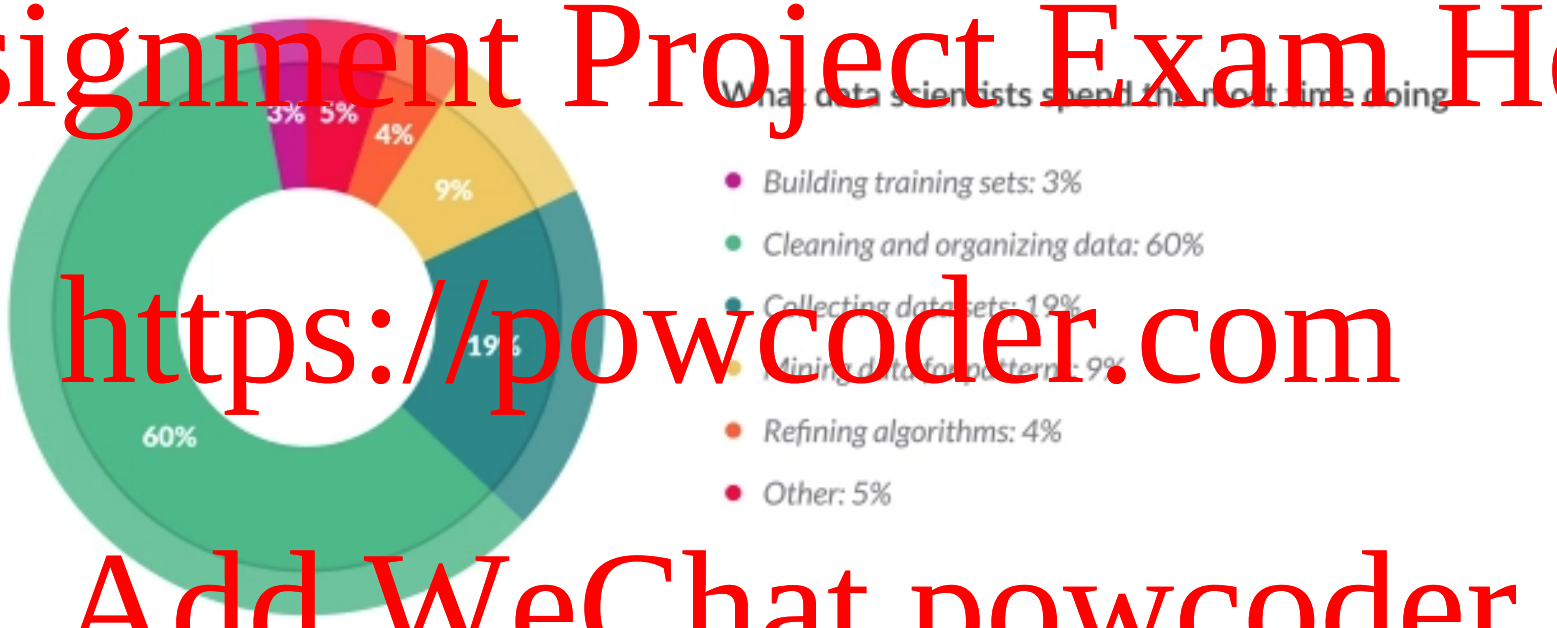
- Tidy data
- Tibble vs. Data.Frame
- Data Import
- tidyr

Data Analysis Pipeline



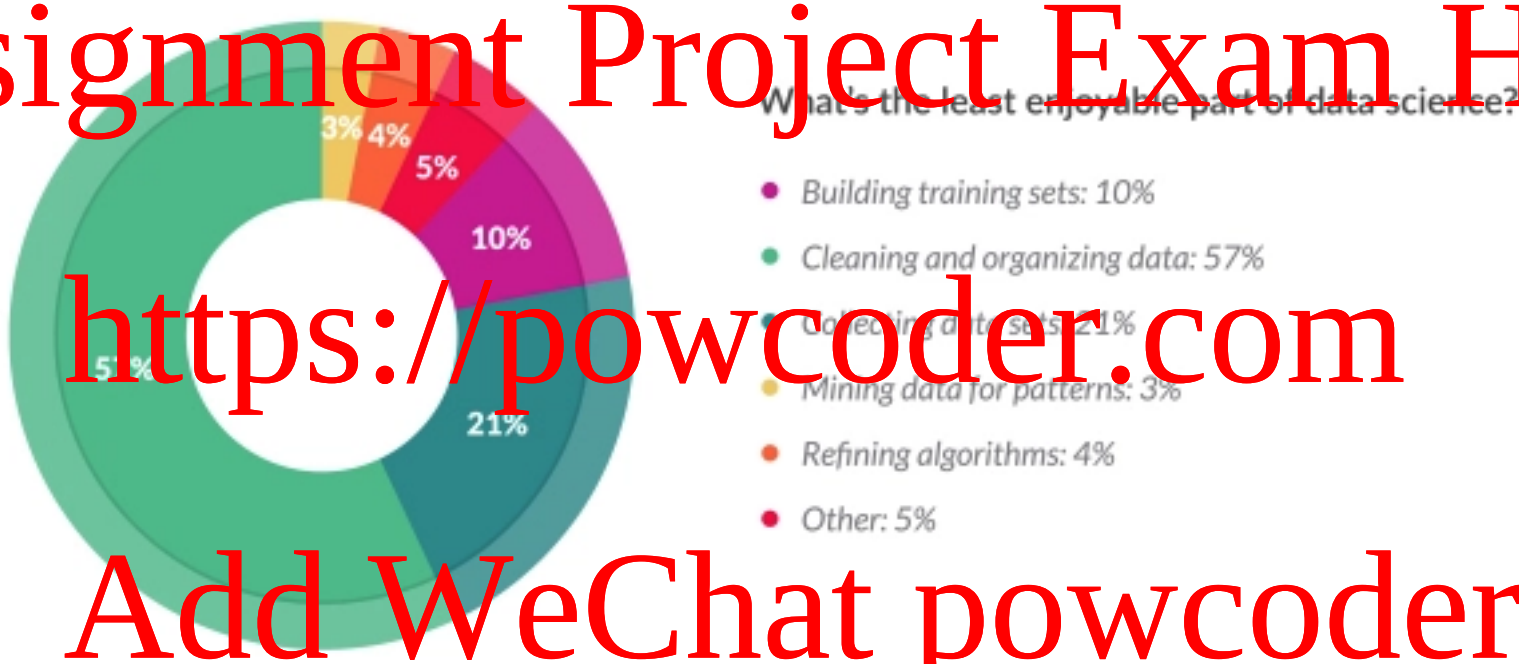
(Wickham and Grolemund 2016)

Where does the time go in data analysis?



https://whatsthebigdata.com/2016/05/01/
data-scientists-spend-most-of-their-time-cleaning-data/

What is the worst part of data analysis?



https://whatsthebigdata.com/2016/05/01/
data-scientists-spend-most-of-their-time-cleaning-data/

Why is it such a pain

- Lots of people work on building new tools for data modelling
- Very-little work has been accomplished on data cleaning tools
- Data formats are very inconsistent
- Disconnect between data entry / data generation and analysis
- Historically smaller datasets

Families and Data

"Happy families are all alike; every unhappy family is unhappy in its own way." - Leo Tolstoy

"Like families, tidy datasets are all alike but every messy dataset is messy in its own way." - (Wickham 2014)

General data structure

- Composed of rows and columns
- Columns are often labelled
- Rows can be labelled

We call this rectangular data (flat file)

- Excel table
- ArcGIS attribute table
- CSV file

Terminology

Dataset

- A collection of values
  - Numbers (Quantitative)
  - Strings (Qualitative)

Values

- Organized as variables and observations

Variable: Values measured for an attribute (e.g. height, population or age)

Observation: Values measured for an entity (e.g. person, country or neighbourhood)

Data Example

Biological Oxygen Demand

```
datasets::BOD
```



```
##   Time demand
## 1    1    8.3
## 2    2   10.3
## 3    3   19.0
## 4    4   16.0
## 5    5   15.6
## 6    7   19.8
```

- datasets::BOD is a dataset
- Time is a variable
- demand is a variable
- There are six observations
- 12 values

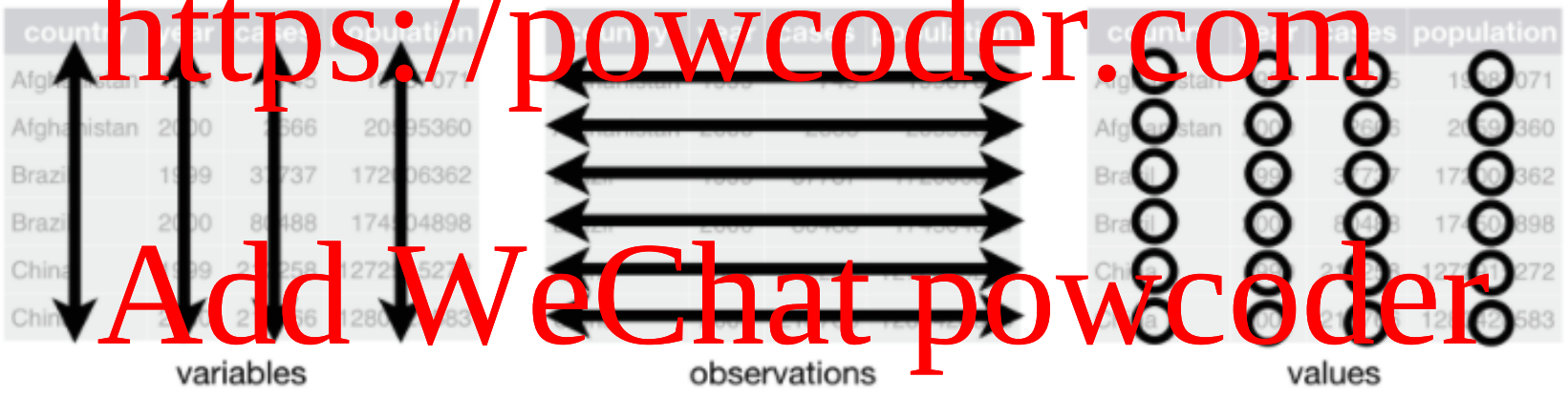
A set of data organization rules

Tidy Data is a set of rules that allows us to maintain our data in a consistent fashion (Wickham 2014)

- Less time on data munging (data preparation)

Tidy Data Rules

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.



(Wickham 2014)

Benefits of tidy data

- It is easier to learn to work with different tools when they share a consistent input.
- R can use vectorized functions, which work well with tidy data.
  - Operations that occur on the entire vector in an efficient way
    - Opposed to using loops, which are slow in R

Data.Frame

- Most people begin with data.frame() in R.
- read.csv() reads data into a data.frame().


```
BOD
```



```
##   Time demand
## 1    1    8.3
## 2    2   10.3
## 3    3   19.0
## 4    4   16.0
## 5    5   15.6
## 6    7   19.8
```

Tibble


```
library(tidyverse)
as_tibble(ED)
```



```
## # A tibble: 6 x 2
##   Time demand
##    <dbl>  <dbl>
## 1     1    8.3
## 2     2   10.3
## 3     3   19
## 4     4   16
## 5     5   15.6
## 6     7   19.8
```

- Prints first 10 rows
- Prints only columns that fit on screen
- Prints column types
- Prints dimensions of data (row, column)

Issues with data.frame


```
colnames(BOD)
```

```
## [1] "Time"   "demand"
```



```
# Partial Matching
BOD$Ti
```



```
## [1] 1 2 3 4 5 7
```

Column Missing Warnings


```
# Convert BOD to tibble
tibbleBOD <- as_tibble(BOD)

# Try partial match
tibbleBOD$Ti
```

```
## Warning: Unknown or uninitialised column: 'Ti'.
```

```
## NULL
```

Select variable in a data.frame

To select a columm (varible) we use the $ operator with a data.frame


```
BOD$Time
```

```
## [1] 1 2 3 4 5 7
```

Select varaible in a tibbel

To select a columm (varible) we use the select() function with a tibble


```
tibbleBOD %>%
  select(Time)
```



```
## # A tibble: 6 x 1
##    Time
##   <dbl>
## 1     1
## 2     2
## 3     3
## 4     4
## 5     5
## 6     7
```

Extracting rows by criteria - data.frame


```
BOD[which(BOD$Time > 2),]
```



```
##   Time demand
## 3    3   19.0
## 4    4   16.0
## 5    5   15.6
## 6    7   19.8
```

Extracting rows by criteria - tibble


```
tibbleBOD %>%
  filter(Time > 2)
```



```
## # A tibble: 4 x 2
##    Time demand
##   <dbl>  <dbl>
## 1     3   19
## 2     4   16
## 3     5   15.6
## 4     7   19.8
```

Subset Observations (Tibble)

Data Wrangling Cheat Sheet

- dplyr::filter()
  - Rows that meet a logical criteria
- dplyr::distinct()
  - Remove duplicate rows
- dplyr::sample_frac()
  - Random sample of fraction of rows
- dplyr::sample_n()
  - Random sample of n rows
- dplyr::slice()
  - Select rows by posistion, e.g. 10:15

Data in libraries

Data can be stored and accessed with libraries (packages) in R


```
# EPA Dataset
ggplot::mpg

# Gapminder
gapminder::gapminder

# Biological Oxygen Demand
datasets::BOD
```

data.frame to tibble

Many datasets are data.frames


```
# data.frame to tibble as_tibble()
as_tibble(BOD)
```


Functions outside of the tidyverse in other packages may not accept a tibble.


```
# tibble to data.frame with as.data.frame()
as.data.frame(mpg)
```

Note: Base R functions typically use dots between words, and tidyverse functions use underscores.

carat	cut	colour
0.23	Ideal	E
0.25	Good	J
1.2	Fair	I

Comma Separated Values

CSV files are very common and likely your best choice for data.

1. They are human readable
   - It can be opened in a text editor
2. Ideal for rectangular data
3. Compatible with most software

CSV Structure

If you were to open a csv file in a text editor:

```
carat,cut,colour
0.23,deal,E
0.25,Good,J
1.2,Fair,I
```

Reading and Writing CSV

R works well with csv files

- read.csv()
  - Stores input as a data.frame
- write.csv()
  - Writes a data.frame to an output file

Working with tibbles (Note the **underscore**)

- read_csv()
- write_csv()

readr::read_csv()


```
library(readr)

readr::read_csv(
  file, # File Path
  col_names = TRUE, # First row column names?
  na = c("", "NA"), # Character for missing values
  skip = 0, # Skip N rows
  )

data <- read_csv("C:/somedata.csv", na = "-999")
```

Reading Other Files with readr

readr can be used for other flat files (rectangular files)

- read_csv2()
  - semicolon separated files (common in countries where , is used as the decimal place)
- read_tsv()
  - reads tab delimited files
- read_delim()
  - reads in files with any delimiter

Outside of the tidy verse

- Haven: reading SPSS, Stata and SAS files
- readxl: reading excel files .xls & .xlsx
- DBI: accessing databases in-conjunction with RMySQL / RSQLite / RPostgreSQL

DEMO: Read in and prepare an Air Pollution csv file

Lecture 4 files: air_pollution.csv

Dealing with wide data

When a variable is spread across multiple columns

- This is often a time variable
  - As seen in our air pollution data


```
tidyr::gather()
```

Wide Data


```
# Demo table
table4a
```



```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

tidyr::gather()

```
table4a %>%
  gather(`1999`, `2000`, key = "year", value = "cases")
```

```
## # A tibble: 6 x 3
##   country     year   cases
##   <chr>       <chr>  <int>
## 1 Afghanistan 1999     745
## 2 Brazil      1999   37737
## 3 China       1999  212258
## 4 Afghanistan 2000    2666
## 5 Brazil      2000   80488
## 6 China       2000  213766
```

gather() notes

- Unlike a data.frame column names in a tibble can begin with numbers
  - To access these column names, you use back ticks `var`
    - Below the tilde ~ on the keyboard
- key, gathering the column names
- cases, the values from the columns

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Long Data

One observation might be scattered across multiple rows.

```
# Demo Long Table
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population    19987071
##  3 Afghanistan  2000 cases             2666
##  4 Afghanistan  2000 population    20595360
##  5 Brazil       1999 cases             37737
##  6 Brazil       1999 population   172006362
##  7 Brazil       2000 cases             80488
##  8 Brazil       2000 population   174504898
##  9 China        1999 cases            212258
## 10 China        1999 population  1272915272
```

tidyr::spread()

```
table2 %>%
  spread(key = type, value = count)
```

```
## # A tibble: 6 x 4
##       country  year  cases population
##         <chr> <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

spread() notes

- key: column containing variable names
- value: where to obtain values for each column created by the key

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Tidy Verse Demonstration

Demonstration of the tidy-verse for data management

Hadley Wickham Data Analysis, https://youtu.be/go5Au01Jrvs

Reading: Poole, M. A., & O'Farrell, P. N. (1971). The assumptions of the linear regression model. Transactions of the Institute of British Geographers, 145-158.

References

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10). doi:10.18637/jss.v059.i10.

Wickham, Hadley, and Garrett Grolemund. 2016. "R for Data Science."