Lecture 2: Data Manipulation

Intro to Data Science for Public Policy
Spring 2017

Jeff Chen + Dan Hammer

Roadmap

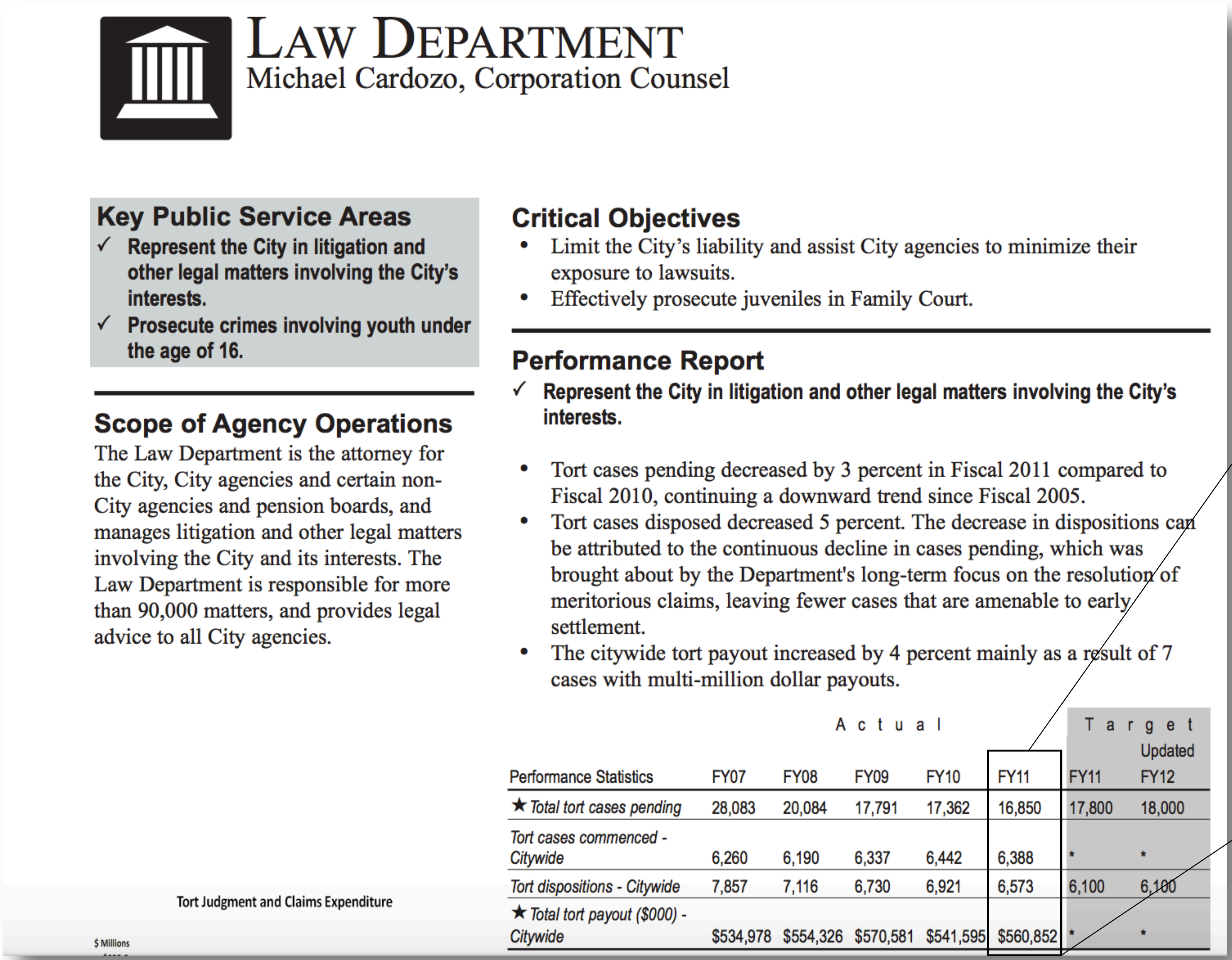
- **Motivating Story**
- Data Manipulation Concepts
- Code-along

Motivation

$560.9 million

lawsuit payouts

New York City, 2011



LAW DEPARTMENT

Michael Cardozo, Corporation Counsel

Key Public Service Areas

- ✓ **Represent the City in litigation and other legal matters involving the City's interests.**
- ✓ **Prosecute crimes involving youth under the age of 16.**

Scope of Agency Operations

The Law Department is the attorney for the City, City agencies and certain non-City agencies and pension boards, and manages litigation and other legal matters involving the City and its interests. The Law Department is responsible for more than 90,000 matters, and provides legal advice to all City agencies.

Critical Objectives

- Limit the City's liability and assist City agencies to minimize their exposure to lawsuits.
- Effectively prosecute juveniles in Family Court.

Performance Report

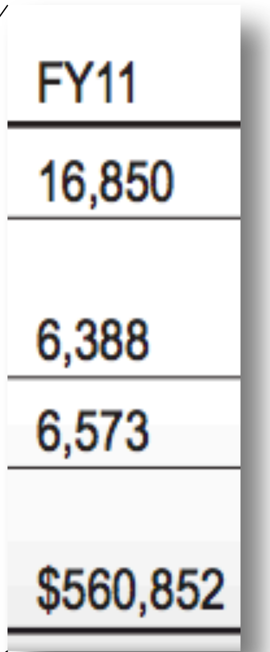
✓ **Represent the City in litigation and other legal matters involving the City's interests.**

- Tort cases pending decreased by 3 percent in Fiscal 2011 compared to Fiscal 2010, continuing a downward trend since Fiscal 2005.
- Tort cases disposed decreased 5 percent. The decrease in dispositions can be attributed to the continuous decline in cases pending, which was brought about by the Department's long-term focus on the resolution of meritorious claims, leaving fewer cases that are amenable to early settlement.
- The citywide tort payout increased by 4 percent mainly as a result of 7 cases with multi-million dollar payouts.

	Actual					Target	
							Updated
Performance Statistics	FY07	FY08	FY09	FY10	FY11	FY11	FY12
★ *Total tort cases pending*	28,083	20,084	17,791	17,362	16,850	17,800	18,000
Tort cases commenced - Citywide	6,260	6,190	6,337	6,442	6,388	*	*
Tort dispositions - Citywide	7,857	7,116	6,730	6,921	6,573	6,100	6,100
★ *Total tort payout ($000) - Citywide*	$534,978	$554,326	$570,581	$541,595	$560,852	*	*

Tort Judgment and Claims Expenditure

$ Millions



FY11
16,850
6,388
6,573
$560,852

Old strategy

- Settle for low and as quick as possible to clear the cases from the docket

Potential Policy Options

- Continue current course
- Ask lawyers to pick and choose cases to litigate
- Use information to moneyball litigation

Use information to moneyball litigation is the same as asking **“which cases are winnable?”** or otherwise stated:

Pr(win | case details)

Example (from similar but not exact data)

Table: Metadata

Case Name. -- Joseph McDermott v. City of N. Y., 50 N.Y.2d 211 (1980)

Case Date. May 1, 1980

Payout Amount: $150,000

Court: Court of Appeals of the State of New York

Other: Type of case, judge, lawyers

Source: https://www.ravellaw.com/opinions/2f75081b7aa9376053a07190a0a38559. Note that some data has been simulated.

Doc: Summary and Notes

The cause of action for indemnification interposed against the manufacturer of an allegedly defective product is independent of the underlying wrong and for the purpose of the Statute of Limitations accrues when the loss is suffered by the party seeking indemnity. Hence, the dismissal of that part of the third-party complaint seeking indemnity, as barred by the four-year Statute of Limitations for breach of warranty measured from the date of tender of delivery (Uniform Commercial Code, § 2-725), was unwarranted. Plaintiff Joseph McDermott, an employee in the New York City Sanitation Department, commenced this action against the city in 1969 after his arm was severed by the hopper mechanism of a sanitation truck. The city, in turn, brought a third-party action in June, 1975 against respondent Heil Company, the manufacturer of the body and hopper of the truck. The city alleged that any injury to the plaintiff was caused solely by Heil's breach of duty, and demanded full indemnification. Specifically, the city claimed that the product was not fit for its intended use and was dangerous to those

Logs: Activities

1. McDermott Case – Request for documents
2. McDermott Case – Five results submitted
3. McDermott Case – New questions submitted
4. McDermott Case – Requests reviewed
5. McDermott Case – New questions submitted
6. McDermott Case – Judge visit
7. McDermott Case – Discovery session

Example (from similar but not exact data)

Table: Metadata

Case Name. -- Joseph McDermott v. City of N. Y., 50 N.Y.2d 211 (1980)

Plaintiff
Plaintiff gender
Number of plaintiffs
Individual or corp

Respondent
Type of respondent

Case Date. May 1, 1980

Age of case
Case load in year

Payout Amount: $150,000

Payout

Court: Court of Appeals of the State of New York

Historical leaning of court

Other: Type of case, judge, lawyers

Past record for case type
Judge's record on similar cases
Lawyer's case load
Lawyer to judge interaction

Example (from similar but not exact data)

Doc: Summary and Notes

The cause of action for indemnification interposed against the manufacturer of an allegedly defective product is independent of the underlying wrong and for the purpose of the Statute of Limitations accrues when the loss is suffered by the party seeking indemnity. Hence, the dismissal of that part of the third-party complaint seeking indemnity, as barred by the four-year Statute of Limitations for breach of warranty measured from the date of tender of delivery (Uniform Commercial Code, § 2-725), was unwarranted. Plaintiff Joseph McDermott, an employee in the New York City Sanitation Department, commenced this action against the city in 1969 after his arm was severed by the hopper mechanism of a sanitation truck. The city, in turn, brought a third-party action in June, 1975 against respondent Heil Company, the manufacturer of the body and hopper of the truck. The city alleged that any injury to the plaintiff was caused solely by Heil's breach of duty, and demanded full indemnification. Specifically, the city claimed that the product was not fit for its intended use and was dangerous to those who used it. In addition, the third-party complaint asserted a claim for negligence. The trial evidence indicated that sanitation truck bearing serial number 252-386 was delivered to the city on February 5, 1969. On the evening of February 24, 1969, plaintiff and two fellow sanitation workers, Richard Mancuso and Joseph Cantelli, were assigned to that truck. When the accident occurred, Mancuso was driving, while McDermott and Cantelli were collecting refuse and loading it into the rear of the truck. Plaintiff

Indemnification sought

Statute of Limitations

Time lapse since event

Respondent detail

Potential misuse

Type of employment

Bodily injury

Disability

Third-parties

Witnesses

Example (from similar but not exact data)

Logs: Activities

Intensity of research

1. McDermott Case – Request for documents
2. McDermott Case – Five results submitted
3. McDermott Case – New questions submitted
4. McDermott Case – Requests reviewed
5. McDermott Case – New questions submitted
6. McDermott Case – Judge visit
7. McDermott Case – Discovery session

# of activities on case = 7

Types of activities

To turn the data into usable information, we need to standardize values, extract concise information, transform values, and merge it together.

Roadmap

- Motivating Story
- Data Manipulation Concepts
- Code-along

Extract-Transform-Load (ETL)

- Data is almost never provided in clean, usable form. ETL is the process that makes data usable.
- **Extract** = obtain data from a database or multiple database of consistent or variable formats
- **Transform** = data is transformed into a usable format for either storage, analysis or other use
- **Load** = the output of the transform stage gets loaded into a database, software, or algorithm for use

Extract-Transform-Load (ETL)

Extract

Subset

Transform

Value

- Clean
- Reformat
- De-dupe
- Extract information

Structural

- Subset
- Order
- Collapse
- Reshape
- Merge

Load

Extract/Subset

Data manipulation: Structural

Exercise!

Reformat/Clean

Data manipulation: Value

- Standardization of values – get values to be consistent
- Reformatting variables

Cleansing: Example

What's wrong with this vector?

```
salary <- c("$100k ","$10,000","None")
```

Cleansing: Example

What's wrong with this vector?

```
salary <- c("$100k ","$10,000","None")
```

- Remove $
- Replace k with 000

- Remove $

- Replace with NA

- Convert to numeric

Cleansing: Example

What's wrong with this vector?

```
salary <- c("$100k ","$10,000","None")
```

```
gsub()
```

```
as.numeric()
```

Cleansing: Example

```
gsub("[pattern]", "[new pattern]", obj)
```

gsub() = “find and replace”

- Pattern and new pattern are the find and replace text respectively
- obj is an R-object

Cleansing: Example

```
salary <- c("$100k ","$10,000","None")
salary <- gsub("k", "000", salary)
salary <- gsub("None", NA, salary)
```

How do we remove $?

Cleansing: Example

```
salary <- c("$100k ","$10,000","None")
salary <- gsub("k", "000", salary)
salary <- gsub("None", NA, salary)
```

How do we remove $ and comma?

```
gsub("[[:punct:]]","", salary)
gsub("[$,]","", salary)
```

Cleansing: Strings & RegEx

String methods. Functions designed to manipulate and work with string values.

```
gsub("[[:punct:]]","", salary)
```

Regular Expressions (RegEx). Special characters that represent string patterns

Cleansing: Strings & RegEx

String methods. Functions designed to manipulate and work with string values.

`grep()`	Returns index position for matched pattern.
`gsub()`	Find and replace string.
`regexpr()`	Returns character position of match.
`substr()`	Extracts values based on character positions.
`regmatches()`	Extracts matched pattern.
`trimws()`	Removes white space around strings.
`tolower()`	Converts all characters to lower case.

Cleansing: String methods

```
#examples
   x <- c("Where's waldo?","Not here")
   grep("waldo",x)
   regexpr("waldo",x)
   substr(x, 8,14)
   regmatches(x,regexpr("waldo",x))
   gsub("Not here", "...found him!",x)
```

Cleansing: RegEx

Positions

`^`	Beginning of string
`$`	End of string

Examples of escaped characters

`\\.`	period
`\\$`	dollar sign
`\\"`	quotation mark

Quantifiers

`*`	Wildcard, match at least 0
`{n}`	Match pattern n times

Extended character classes for R

`[[:punct:]]`	punctuation
`[[:alpha:]] or [a-zA-Z]`	alphabetic
`[[:digit:]] or \\d`	numbers
`[[:alnum:]] or [a-zA-Z0-9]`	alphanumeric
`[[:space:]] or \\s`	spaces
`\\w`	word characters
`\\W`	Not word

More on this at: https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html

Cleansing: String methods

```
#Censor the phone number and the time
a <- ”Please call me at 212-353-4213 at 12pm"
```

De-duplication

Data manipulation

- De-duplication ensures that certain records are not over-represented.
- Ensures more accurate count of unique records and minimizes erroneous ‘cartesian product’ matches during merging

De-Dupe Method

```
duplicated(obj)
```

Accepts an R-object (e.g. dataframe, vector), returns boolean vector indicating if a record is a duplicate.

```
#example
  vec <- c(1, 2, 3, 4, 1, 2, 3, 4)
  duplicated(vec)
  vec[!duplicated(vec)]
```

Parsing

Data manipulation

Data can be stored in nested formats:

```
a <- "1,2,4,10"
```

And data that is delimited in any way can be parsed:

```
b <- "duck, duck, goose"
c <- "The Dow is up!"
```

Parsing

`strsplit(x, delim)`	Splits string based on delimiter.
`unlist(x)`	Converts list object into vectors.

```
#Convert following into three column matrix
   b <- "duck, duck, goose"
   c <- strsplit(b, ",")
   matrix(trimws(unlist(c)), nrow=1, byrow=T)
```

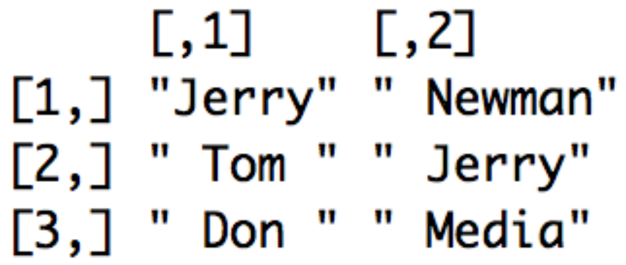
Parsing: Example

```
#Convert following into three column matrix
   b <- "duck, duck, goose"
   c <- strsplit(b, ",")
   matrix(trimws(unlist(c)), nrow=1, byrow=T)
```

Parsing: Exercise

```
#Epic battles: Parse into two column matrix
case <-  c(  "Jerry v. Newman",
              " Tom vs. Jerry",
              " Donald versus Media")
```

Goal →



```
     [,1]    [,2]
[1,] "Jerry" " Newman"
[2,] " Tom " " Jerry"
[3,] " Don " " Media"
```

```
Hint: clean up delimiter first
```

Parsing: Exercise - Answer

```
case <-  c(  "Jerry v. Newman", " Tom vs. Jerry",
             " Don versus Media")
case <- gsub("vs\\.| v\\.", "versus",case)
b <- strsplit(case,"versus")
matrix(unlist(b), nrow=3, byrow=T)
```

Reshape

Data manipulation

Tabular data often is found in one of two formats:

Long form

Loc_id	Type	Value
1	Lat	-40.72
1	Lon	50.1
2	Lat	-43.5
2	Lon	52.45

Wide

Loc_id	Lat	Lon
1	-40.72	50.1
2	-43.5	52.45

Reshape

Data manipulation

Tabular data often is found in one of two formats:

Long form

Loc_id	Type	Value
1	Lat	-40.72
1	Lon	50.1
2	Lat	-43.5
2	Lon	52.45

Wide

Loc_id	Lat	Lon
1	-40.72	50.1
2	-43.5	52.45

Reshape: Example Uses

Long form

- Panel series
- Compact data storage
- Server Logs
- Sensor recordings

Wide form

- Satellite imagery
- Economic time series
- Most tabular analysis-ready data

Reshape: Long vs. Wide

```
x <- data.frame( id = c(1,1,2,2),
                 t = c(1,2,1,2),
                 income = c(50,55,101,123),
                 vote = c(8,7,4,3))
wide <- reshape(x,
                 idvar="id",
                 timevar="t",
                 direction="wide")
```

Reshape: Wide to Long (simple case)

```
#simple case
long <- reshape(     wide,
                     idvar = "id",
                     timevar = "t",
                     direction = "long")
```

Reshape: Wide to Long (complex)

```
#complex case
wide <- data.frame( id = c(1,2,3),
                         income = rnorm(3,100,5),
                         debt = rnorm(3,-10,20))
long <- reshape(wide,
               varying = c("income", "debt"),
               v.names = "amount",
               timevar = "financials",
               time = c("income", "debt"),
               direction = "long")
```

Collapse/Aggregate

Data manipulation

- Collapse methods convert raw transactional data into summaries of discrete units
- Aggregates = summary stats by some group identifier

Collapse: Example Uses

Often times more signal in aggregates and patterns begin to emerge

- Uber rides → Calculate number of uber rides by origin and destination
- Call center logs → Call volume by hour, day, type for staff planning
- Legal documents → Frequency of word combinations that are associated with themes or issues

Collapse

```
aggregate(obj, by = list(x), FUN = fun)
```

Key

obj = an R object like a data frame

by = group by or list of variables to be

FUN = a statistical function (e.g. mean,

Collapse: Example

```
x <- data.frame(var1 =    round(rnorm(100,1000,100)),
                group1 = round(runif(100)*5))

aggregate(x$var1, by = list(x$group1), FUN = length)
aggregate(x$var1, by = list(x$group1), FUN = summary)
```

Merges / Joins

Data manipulation

Combine two or more data using common attributes

Why joining is necessary?

- Tables are often "normalized" – data is stored in separate tables to reduce redundancy
- Allows for identifying commonalities and gaps
- Practical cases:
  - What is the market conversion rate in a twitter campaign?
  - Add variables to improve predictions

Service User

<u>ID</u>	Name
1	James
2	Jane
3	John
4	Jana
5	Jorgenson
6	Janette

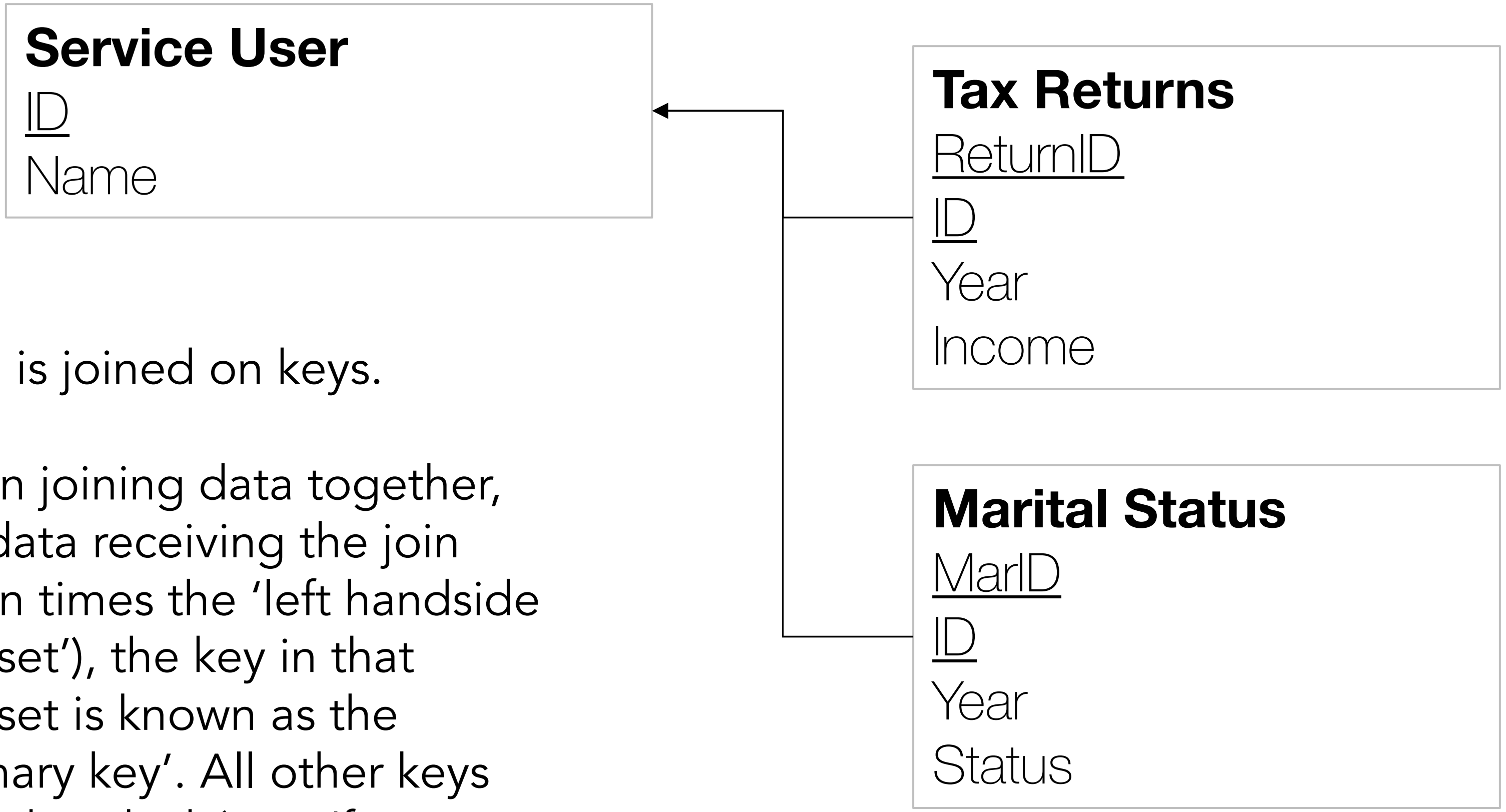
Tax Returns

<u>ReturnID</u>	<u>ID</u>	Year	Income
1	1	2010	$100,000
21	1	2011	$100,000
34	1	2012	$90,000
300	2	2011	$90,000
405	2	2013	$96,000
1524	2	2014	$100,000

Each table has a '**primary key**', which is a column or combination of columns that uniquely identifies a given row

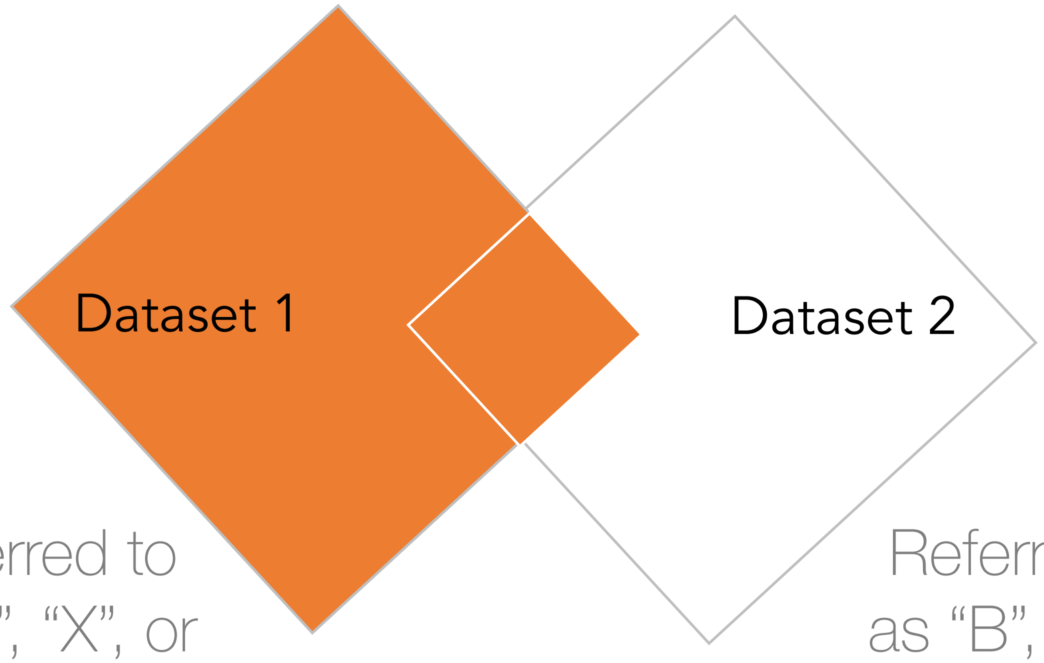
Marital Status

<u>MarID</u>	<u>ID</u>	Year	Status
142	1	2010	Married
4	2	1990	Married
30	3	2009	Single
531	3	2012	Married
953	3	2013	Divorce



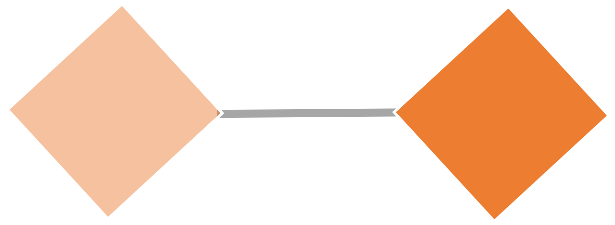
Data is joined on keys.

When joining data together, the data receiving the join (often times the 'left handside dataset'), the key in that dataset is known as the 'primary key'. All other keys (right handside) are 'foreign keys'



Referred to as “A”, “X”, or “left”

Referred to as “B”, “Y” or “right”

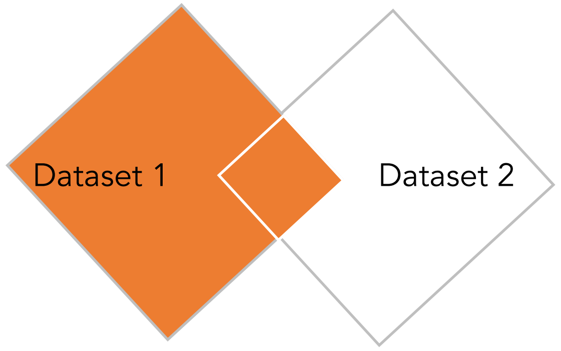


Datasets

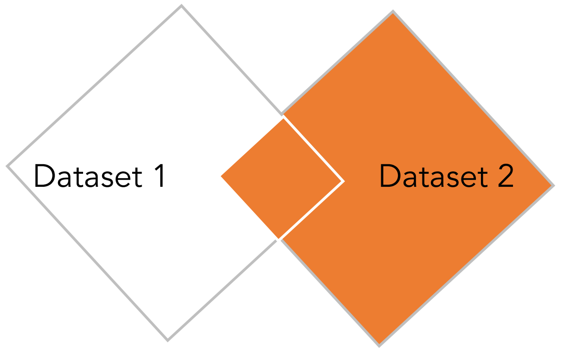
Records

Types of Joins/Merges

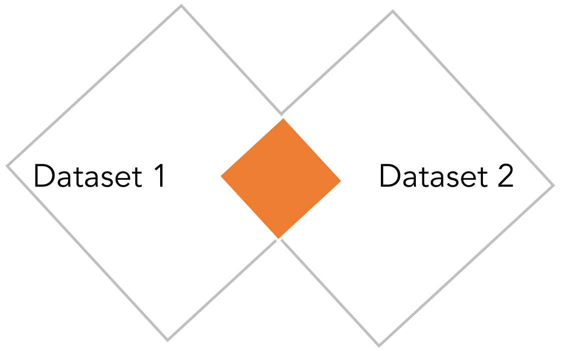
Table-level



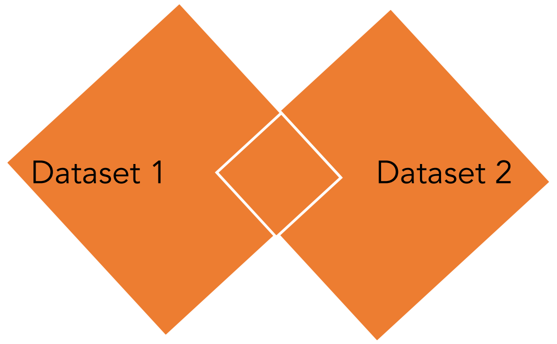
Left [Outer] Join
A U (A ∩ B)



Right [Outer] Join
B U (A ∩ B)

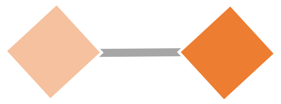


Inner Join
Intersection = A ∩ B

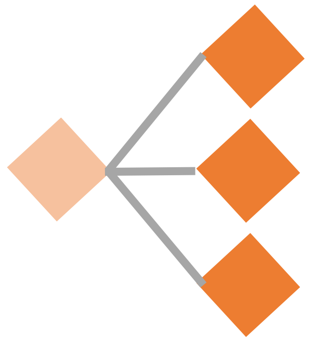


[Full] Outer Join
Union = A U B

Record-level



1:1



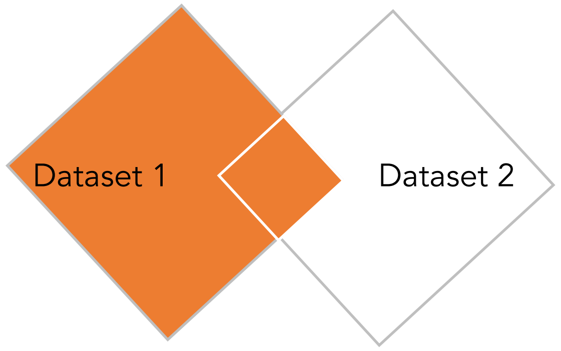
1:m

merge()

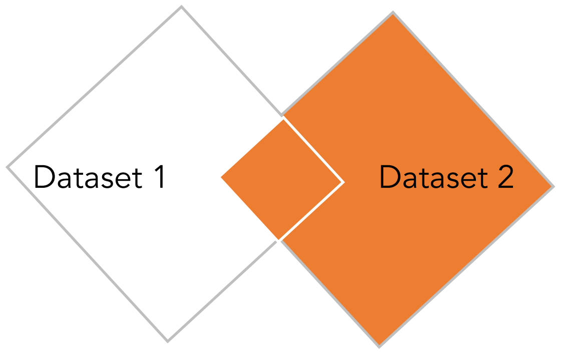
```
merge(df1, df2,
      by = c("match_var"),
      all.x = T, all.y = F)
```

Key
df1 = data frame #1
df2 = data frame #2
by = vector of column names to be used for matching
all.x = boolean to keep all df1
all.y = boolean to keep all df2

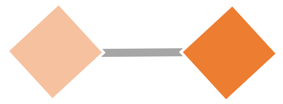
What should the merge command be if the merge variable is "id"?



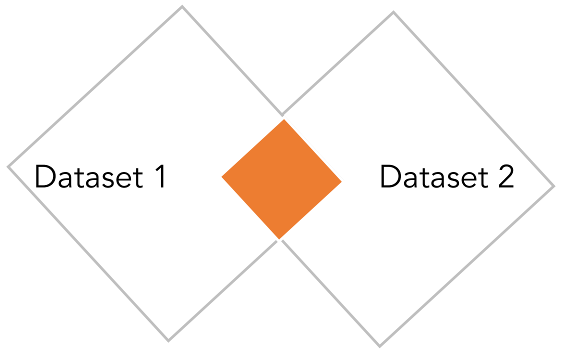
[ ]



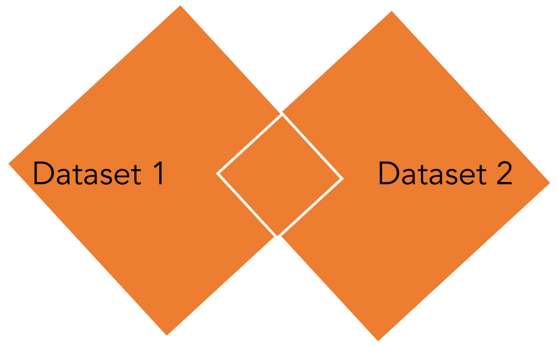
[ ]



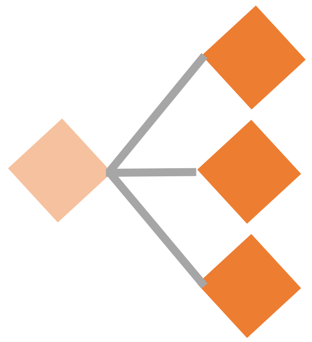
[ ]



[ ]

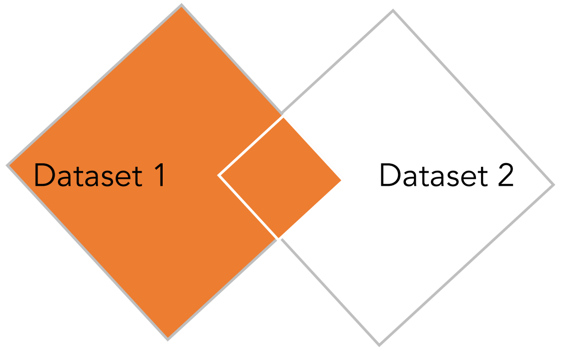


[ ]

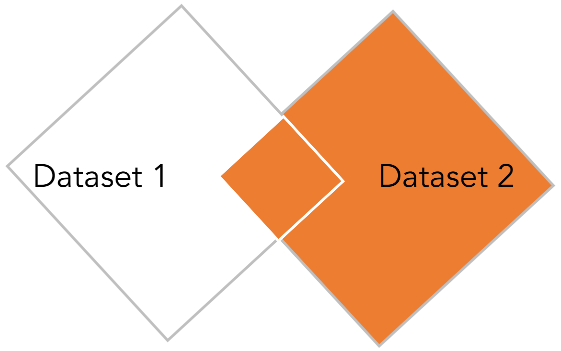


[ ]

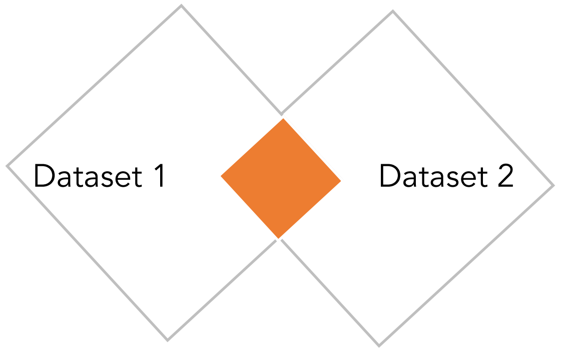
What should the merge command be if the merge variable is “id”?



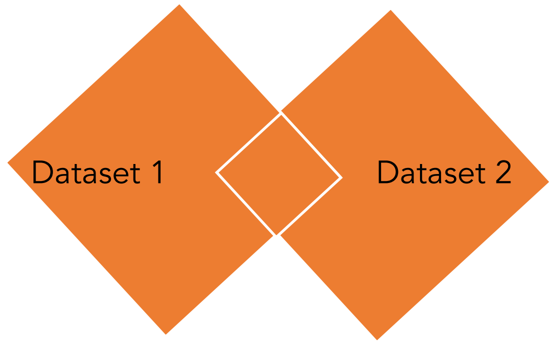
```
merge(df1, df2, by =
"id", all.x = T)
```



```
merge(df1, df2, by =
"id", all.y = T)
```

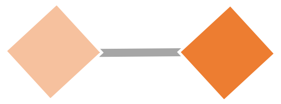


```
merge(df1, df2, by =
"id")
```

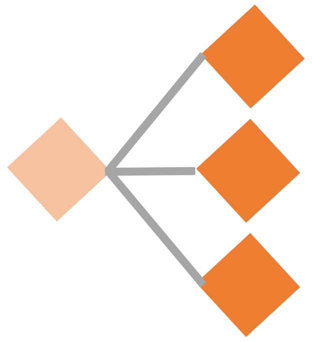


```
merge(df1, df2, by =
"id", all.x = T,
all.y = T)
```

1:1 and 1:m are not dependent on merge but deduplication



Any



Any

Merges: Universes and backbones

Merges are only as good as the completeness of records of the underlying tables. Need ‘complete frame’ or universe to get full picture

Ex: Missing days in these two sequences

```
days1 <- data.frame(time = c( 1, 2, 3, 10, 11, 12,
    20), flag1 = c(1))
days2 <-  data.frame(time = c( 1, 2, 4, 8, 9, 10,
    12, 21), flag2 = c(1))
```

Merges: Universes and backbones

Why gaps matter:

- Knowledge of reliability of data collection
- Ability to score data across the whole potential universe rather than the known universe

Merges: Universes and backbones

Backbone = the most expansive known list of potential records

```
days1 <- data.frame(time = c( 1, 2, 3, 10, 11, 12,
   20),flag1 = c(1))
days2 <-  data.frame(time = c( 1, 2, 4, 8, 9, 10, 12,
   21),flag2 = c(1))
df <- data.frame(time = seq(1,21, 1))
df <- merge(df, days1, by="time", all.x=T)
df <- merge(df, days2, by="time", all.x=T)
```

Merges: Universes and backbones

time	flag1	flag2
1	1	1
2	1	1
3	1	NA
4	NA	1
5	NA	NA
6	NA	NA
7	NA	NA
8	NA	1

Roadmap

- Motivating Story
- Data Manipulation Concepts
- Code-along
  - SOTU data cleansing