

PLSC 502 – Autumn 2023

Data: Structure, Measurement, and Management

August 28, 2023

Measurement

The quantification of characteristics of a unit, for purposes of comparison.

Importance of:

1. Theory
2. Context
3. Precision

In general:

Conceptualization → Operationalization → Measurement

Validity

- Construct Validity
- Content Validity
- Criterion Validity
- Face Validity

Reliability

- Test-Retest Reliability (“time”)
- Interrater Reliability (“space”)
- Internal Consistency (“tools”)
- Parallel Forms (also “tools”)

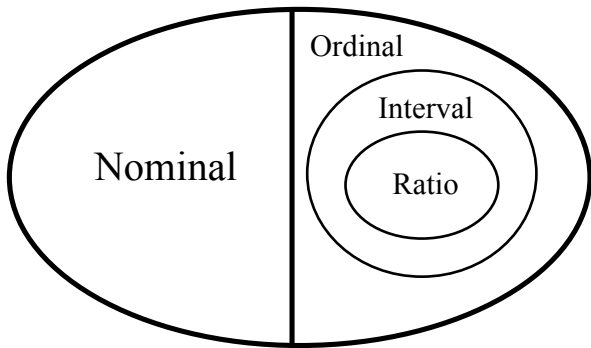
Four levels:

- Nominal (classification)
- Ordinal (order)
- Interval (equal intervals)
- Ratio (“true zero”)

In terms of information:

$$Ratio \geq Interval \geq Ordinal \geq Nominal$$

Levels of Measurement (continued)



Discrete vs. Continuous Indicators

Consider a variable $X \in [a, b] \dots$

For Continuous X

- X can take on *any* value within the range from a to b
- Number of possible values is (theoretically) infinite
- Formally, $\Pr(X = x) = 0 \forall x \dots$
- In general, X is defined over all values from a to b
- *In practice*, the number of possible values X can take on is limited by the precision of the measurement tool

For Discrete X

- X can take on a *finite* or *countably infinite* number of values S between a and b
- $\Pr(X = x) > 0 \forall x \in S$, or 0 otherwise
- That means that $\sum_{x \in S} \Pr(X = x) = 1$

Examples of Measures

Examples of Measures, by Type and Level of Measurement

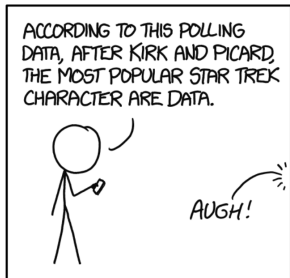
Level of Measurement	Discrete	Continuous
Nominal	{Heels, Boots, Sneakers}	n/a
Ordinal	Social Class (Upper, middle, lower)	(certain rankings)
Interval	Year	Temperature, degrees F
Ratio	Counts of things	Height, weight, distance, etc.

The Key Point About Measurement

Paraphrasing N. R. Campbell (Final Report, p. 340), we may say that measurement, in the broadest sense, is defined as the assignment of numerals to objects or events according to rules. The fact that numerals can be assigned under different rules leads to different kinds of scales and different kinds of measurement. The problem then becomes that of making explicit (a) the various rules for the assignment of numerals, (b) the mathematical properties (or group structure) of the resulting scales, and (c) the statistical operations applicable to measurements made with each type of scale.

– Stevens (1946)

Data



ANNOY GRAMMAR PEDANTS ON ALL SIDES
BY MAKING "DATA" SINGULAR *EXCEPT*
WHEN REFERRING TO THE ANDROID.

Organization:

i	X_1	X_2	...	X_K
1	X_{11}	X_{21}	...	X_{K1}
2	X_{12}	X_{22}	...	X_{K2}
3	X_{13}	X_{23}	...	X_{K3}
\vdots	\vdots	\vdots	\vdots	\vdots
N	X_{1N}	X_{2N}	...	X_{KN}

with indices:

$$i \in \{1, 2, 3, \dots, N\} \text{ (for units / rows)}$$

$$k \in \{1, 2, 3, \dots, K\} \text{ (for variables / columns)}$$

Data can vary...

- ... cross-sectionally:
 - Each line of data is an observation on a unique unit of observation
 - Data represent a single point in time (or unit of time)
 - Example: A [single cross-sectional survey](#) of public opinion
- ... temporally:
 - Each line of data is an observation on a unique unit of observation
 - Data represent a single point in time (or unit of time)
 - Example: Monthly / annual [public mood data](#) for the U.S.
- ... relationally:
 - Each line of data is a “connection” between two or more units of observation
 - Data represent characteristics of the relationship between those two or more units
- Any / all of the above simultaneously...

Cross-Sectional Data: 1997 Baseball Survey

```
> select<-c("respon","age","female","followbaseball","DH_appr")  
> head(DH[select],8)
```

	respon	age	female	followbaseball	DH_appr
1	1	65	Female	0	NA
2	2	63	Male	1	1
3	3	56	Female	1	NA
4	4	24	Female	0	NA
5	5	47	Male	0	NA
6	6	81	Female	1	NA
7	7	28	Male	1	1
8	8	76	Male	1	0

Time Series Data: SCOTUS Clerks

```
> select<-c("Term","female","white","top5law","lcclerk")  
> head(Clerks[select],15)
```

	Term	female	white	top5law	lcclerk
1	1953	0.000	100.00	44.44	12.500
2	1954	0.000	100.00	64.71	44.444
3	1955	0.000	100.00	76.47	41.667
4	1956	0.000	100.00	55.56	20.000
5	1957	0.000	100.00	58.82	30.000
6	1958	0.000	100.00	57.89	27.273
7	1959	0.000	100.00	61.11	44.444
8	1960	0.000	100.00	66.67	7.143
9	1961	0.000	100.00	55.56	21.429
10	1962	0.000	100.00	71.43	21.429
11	1963	0.000	100.00	78.95	25.000
12	1964	0.000	100.00	62.50	8.333
13	1965	0.000	100.00	70.00	43.750
14	1966	5.882	100.00	52.94	33.333
15	1967	0.000	95.24	66.67	44.444

Organization:

$$X_{it} \in X = \begin{pmatrix} X_{11} \\ X_{12} \\ \vdots \\ X_{1T} \\ X_{21} \\ X_{22} \\ \vdots \\ X_{NT-1} \\ X_{NT} \end{pmatrix}$$

... with $i \in \{1, 2, 3, \dots, N\}$ indexing units and $t \in \{1, 2, 3, \dots, T\}$ indexing time points.

Panel/TSCS Data: Countries, 1946-2000

```
> select<-c("country","ccode","year","gdppc","polity","region","coldwar")
> Panel<-Panel[order(Panel$ccode,Panel$year),] # sort
> Panel[1:200,select]
```

	country	ccode	year	gdppc	polity	region	coldwar
9664	US	2	1946	NA	10	1	1
9665	US	2	1947	NA	10	1	1
9666	US	2	1948	NA	10	1	1
9667	US	2	1949	NA	10	1	1
9668	US	2	1950	1915.000	10	1	1
9669	US	2	1951	2196.000	10	1	1
9670	US	2	1952	2300.000	10	1	1
.							
.							
.							
9706	US	2	1988	20848.000	10	1	1
9707	US	2	1989	22192.000	10	1	1
9708	US	2	1990	23218.000	10	1	0
9709	US	2	1991	23639.000	10	1	0
.							
.							
.							
9715	US	2	1997	30468.000	10	1	0
9716	US	2	1998	31776.000	10	1	0
9717	US	2	1999	NA	10	1	0
2676		2	2000	NA	10	1	0
3886	CANADA	20	1946	NA	10	1	1
3887	CANADA	20	1947	NA	10	1	1
3888	CANADA	20	1948	NA	10	1	1
3889	CANADA	20	1949	NA	10	1	1
3890	CANADA	20	1950	1544.000	10	1	1
3891	CANADA	20	1951	1717.000	10	1	1

Organization is into pairs / “dyads”:

- Data are indexed as X_{ij} , where
- $i \in \{1, 2, 3, \dots, N\}$ indexes the “first” unit and
- $j \in \{1, 2, 3, \dots, N\}$ indexes the “second” unit in each “dyad.”

For N units, there are:

$$\frac{N(N-1)}{2} \text{ possible non-directed pairs (‘‘dyads’’)}$$

and

$$N(N-1) \text{ directed pairs (possible ‘‘directed dyads’’)}$$

Note: “Unbalanced” relational data are also possible (with $i \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, M\}$, $M \neq N$)

Relational Data (continued)

Organization (non-directed dyads):

$$X_{ij} \in X = \begin{pmatrix} X_{12} \\ X_{13} \\ \vdots \\ X_{1N} \\ X_{23} \\ X_{24} \\ \vdots \\ X_{2N} \\ X_{34} \\ X_{35} \\ \vdots \\ X_{N-2,N-1} \\ X_{N-2,N} \\ X_{N-1,N} \end{pmatrix}$$

Relational Data: Country "Dyads" (1968)

```
> select<-c("ccode1","ccode2","dyadid","dem1","dem2","allies","distance")
```

```
> Dyads[1:300,select]
```

	ccode1	ccode2	dyadid	dem1	dem2	allies	distance
1	2	20	2020	10	10	1	0
2	2	40	2040	10	-7	0	1135
3	2	41	2041	10	-9	1	1437
4	2	42	2042	10	-3	1	1477
5	2	51	2051	10	10	0	1446
6	2	52	2052	10	8	1	2176
.							
.							
.							
126	2	840	2840	10	5	1	8570
127	2	850	2850	10	-7	0	10172
128	2	900	2900	10	10	1	9916
129	2	920	2920	10	10	1	8759
130	20	40	20040	10	-7	0	1586
131	20	41	20041	10	-9	0	1869
132	20	42	20042	10	-3	0	1893
133	20	51	20051	10	10	0	1897
134	20	52	20052	10	8	0	2547
135	20	53	20053	10	NA	0	2426
.							
.							
.							
259	20	900	20900	10	10	0	10019
260	20	920	20920	10	10	0	9009
261	40	41	40041	-7	-9	0	722
262	40	42	40042	-7	-3	0	868
263	40	51	40051	-7	10	0	506
.							
.							
.							

JSON (JavaScript Object Notation)

- Lightweight, simple, self-describing
- Fields can contain *strings*, *numbers*, *JSON objects*, *arrays*, *Booleans*, or be *null* (empty)
- Widely used in CS + IT; common format for (e.g.) data from APIs
- *In general*, should be converted to rectangular form for analysis

Tabular vs. JSON Formats

Tabular data:

```
> df
```

id	guitar	pickups
1	Stratocaster	3
2	Les Paul	2
3	Telecaster	2

Same data, JSON format:

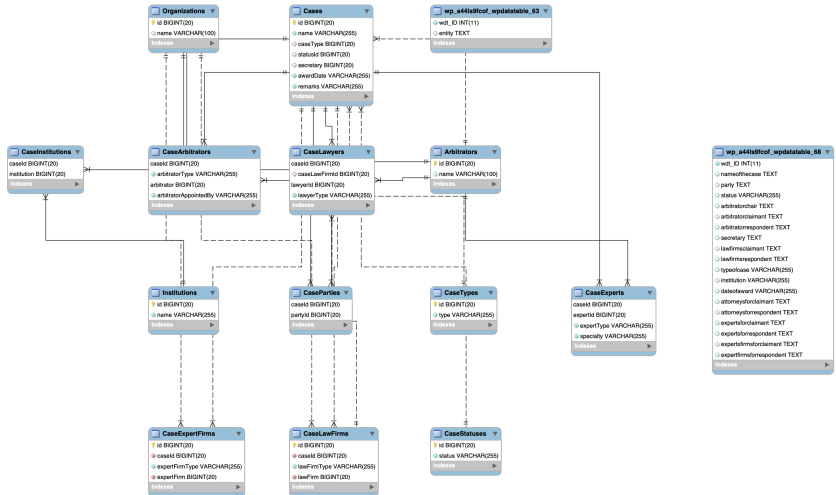
```
> df.JSON<-toJSON(df)
> df.JSON
```

```
[1] "{\"id\": [1,2,3], \"guitar\": [\"Stratocaster\", \"Les Paul\", \"Telecaster\"], \"pickups\": [3,2,2]}\""
```

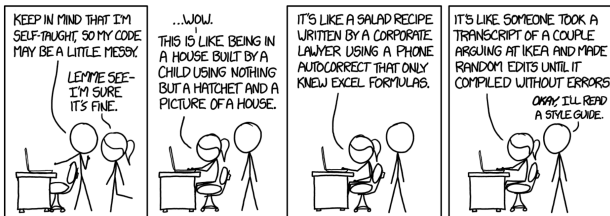
SQL (Structured Query Language)

- Pronounced *ess-kyew-ell* or sometimes *see-quill*
- Format for (*relational*) *databases*
- *De facto* standard database format in CS; applications include MySQL, SQL Server, Oracle, MS Access, PostgreSQL...
- Comprised of rectangular *tables* / arrays of data, that are *related* to each other via *indices* (identifiers)
- Generally one *extracts* tables from SQL databases to analyze them statistically using R...

SQL Example



Data Management



Where Do Data Come From?

Sources:

- Researcher-collected (i.e., *you...*)
- Governments / sovereigns...
- Non-profit organizations (including universities...)
- Business and the private sector:
 - Free / online / etc.
 - Data vendors
 - Disclosed data
- Private individuals
- *Aggregations* of multiple sources...
- “Data exhaust”

Getting Data: The R Version

Options:

1. You can enter data into R/RStudio *by hand* (but you'd rather not).
2. You can read data from a *local file*. For example, in standard `.csv` format:
 - values in each row separated by commas,
 - rows separated by line breaks, and
 - a single row with comma-separated variable names at the top

E.g.:

```
> DF <- read_csv("Data/SCOTUS-votes.csv")
```

Getting Data (continued)

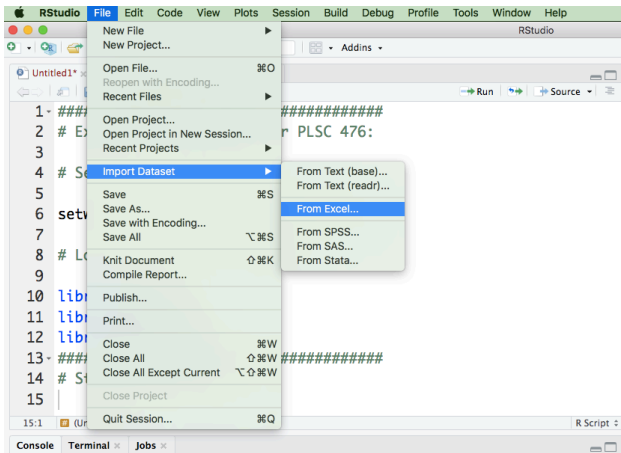
3. You can read a .csv (or, for that matter, a .xls, or other format file) directly from the web, via the file's URL, using routines in the readr package:

```
> install.packages("readr")  
> library(readr)  
> MQScores <- read_csv("http://mqscores.lsa.umich.edu/media/2020/justices.csv")
```

(we'll use this a lot for class)

Getting Data (continued)

4. In RStudio, you can read data interactively, via “File → Import Dataset”



Getting Data (continued)

5. You can import data dynamically via RESTful (and other) APIs...

- Simplest via the `httr` and `jsonlite` packages
- Usually requires knowledge of JSON-formatted data
- A basic tutorial is [here](#).

Summary:

Data Formats and Packages

Format	Useful Command(s) / Package(s)
Plain text (.txt, etc.)	<code>read.table</code> , <code>readr</code>
Comma-separated (.csv)	<code>read_csv</code> , <code>read.csv</code>
Excel (.xls, etc.)	<code>readxl</code> , <code>xlsx</code>
SPSS (.sav, etc.)	<code>haven</code> , <code>foreign</code>
Stata (.dta)	<code>haven</code> , <code>foreign</code>
SAS (.ssd, etc.)	<code>haven</code> , <code>foreign</code>
JSON (.json)	<code>jsonlite</code>
Databases (.adb, .sql, etc.)	<code>DBI</code> , <code>odbc</code> , <code>dbplyr</code>

Why?

- Observation doesn't exist
- Data don't exist for that observation
- Data exist, but are *impossible* to measure
- Data exist, but were not measured

Three types:

- Missing completely at random (“MCAR”) – easy to deal with
- Missing at random (“MAR”) – harder to deal with
- Informatively (or “non-ignorably”) missing – very hard to deal with

Missing Data: What To Do?

- Listwise deletion / “complete cases analysis”
- Interpolation / replacement values
- *Imputation*-based approaches

Things¹ We Do With Data

Inter alia:

- *Sort* data (by values in one or more columns)
- *Add* rows / columns
- *Remove* / *subset* rows / columns
- *Combine* rows / columns (*aggregation*)
- *Separate* rows / columns (*disaggregation*)
- *Transform* rows / columns (including recoding, handling missingness, etc.)
- *Reshaping* data (“wide” vs. “long” formats)
- *Merging* (a/k/a “joining”: 1-to-1, 1-to-many, many-to-1, many-to-many, “inner,” “outer,” “left” / “right,” etc.)

¹(That Aren't Statistics Or Graphics)

- **Use descriptive variable names.**
 - Spell it out.
 - Use “directional” names.
- **Be consistent in naming variables.**
- **Label everything.**
- **Never overwrite anything you can't recreate.**
- **Make everything recreateable.**
- **Annotate your code.**
- **Write reproducible code.**

3 Rs: Readable, reusable, robust.

What does that mean?

- Pull data directly from the source whenever possible.
- Query for packages, and install as necessary.
- Use `set.seed()` when conducting simulations (and use a consistent seed).
- Annotate code to clarify purpose and content.
- Use line breaks and whitespace strategically for clarity.
- Strongly consider consulting a *style guide*.

For more, see (e.g.) [here](#).

From PLSC502-DayTwo-2023.R:

```
>#-----  
># PLSC 502 -- Fall 2023  
.  
.  
.  
> library(readr)  
>  
> # DH data:  
>  
> DH<-as.data.frame(read_csv("https://raw.githubusercontent.com/PrisonRodeo/  
>                               PLSC502-2023-git/master/Data/DH.csv"))  
>  
> select<-c("respon", "age", "female", "followbaseball", "DH_appr") # select variables  
> head(DH[select],8) # prints the first 8 lines + 5 variables of DH  
.  
.  
.
```