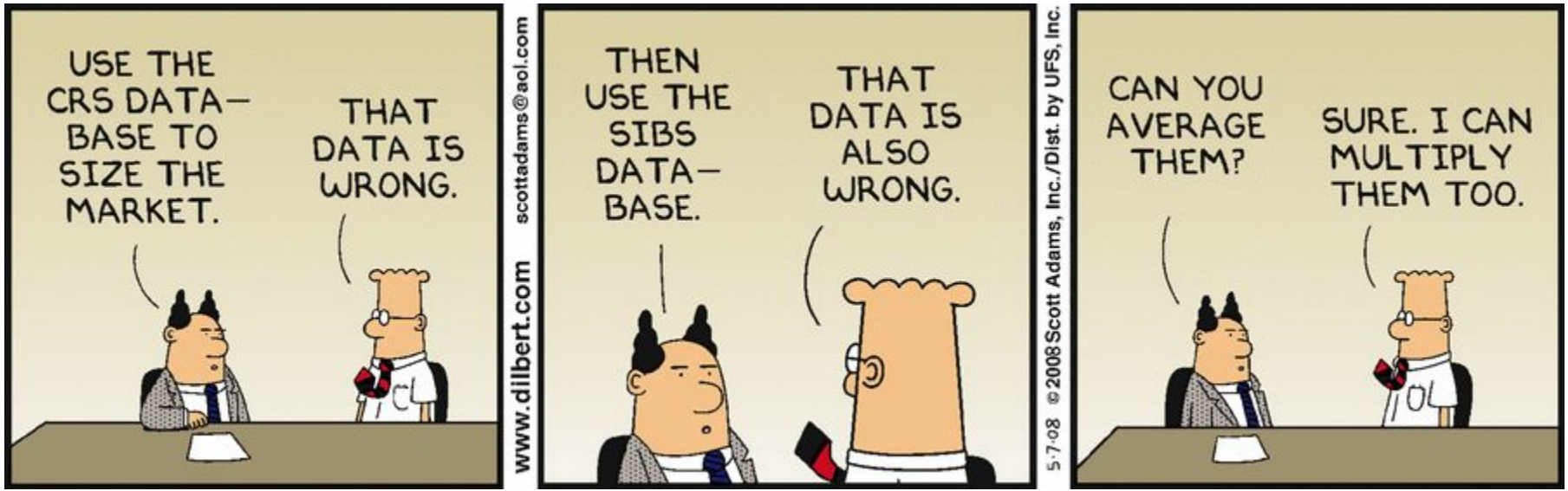
Data Cleaning

"This is not what I meant when I said 'we need better data cleansing!'"

Garbage in, garbage out!



Spring Weekend Poll Results

- Let’s poll Brown students to find out which artists they voted for to perform during Spring Weekend.
- Our poll asks students to write in an artist’s name.
  - Pros: candidates are not limited to a fixed set of artists (anyone is fair game)
  - Cons: it’s hard to tally the results
    - How many people voted for Waka Flocka Flame?
    - Some people might write Waka Flocka; some might waka flacka, or waka floka, etc.
- We can easily change text to all uppercase or all lowercase.
- Parsing out extra words and correcting spelling is more difficult, especially in a large data set.

What is data cleaning?

To produce technically correct data:

1. Type checking: verify that data values are stored correctly: e.g., numbers as numerics not characters; categories as factors; etc.
2. Normalizing: are data values comparable: e.g., is a gender value `M` or `m` or `Male` or `male`, etc.; are infant ages recorded in months or years?

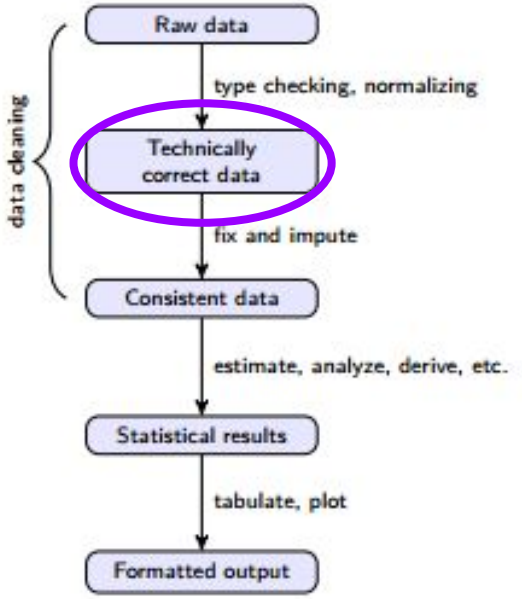


Figure 1: Statistical analysis value chain

Image source

What is data cleaning?

To produce consistent data:

1. Correcting incorrect values (e.g., negative ages, pregnant males, etc.)
   - But what if a very young child is reported married? Which is wrong, status or age?
2. Handling extreme and missing values:
   - Detect outliers, and possibly remove them
   - Possibly impute (i.e., infer) missing values
   - Use sound judgment, and always document and explain your decisions!

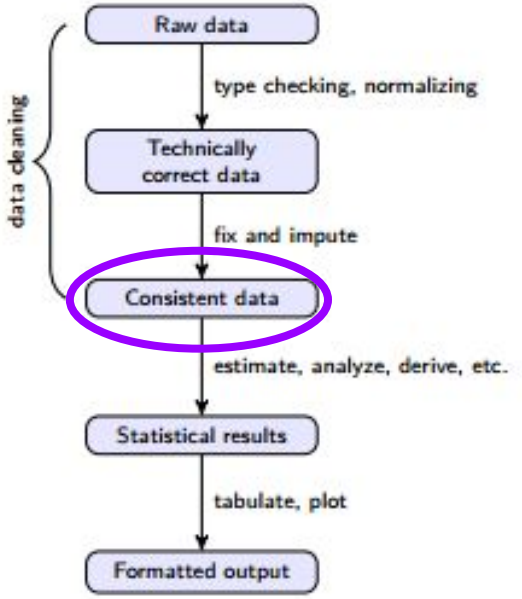


Figure 1: Statistical analysis value chain

Image source

Type checking

Use type coercion functions:

```
> as.character(2017)
“2017”

> as.numeric(TRUE)
1

> as.logical(0)
FALSE

> as.factor("Male")
Male
Levels: Male
```

String Manipulation

- `toupper` & `tolower`: changes the case of strings
- Good style and makes processing easier
  - E.g., == is case sensitive

```
> toupper(c("Green", "Red"))
"GREEN" "RED"

> tolower(c("Green", "Red"))
"green" "red"
```

Normalization: strings

Use `stringr` library:

```
> str_trim("     Hello World!")
"Hello World!"

> str_pad("Hello World!", 15, "left")
"   Hello World!"

> str_detect("Hello World!", "ello")
TRUE

> str_replace("Hello World!", "Hello", "Yellow")
"Yellow World!"
```

Normalization: dates

Use `lubridate` library:

```
> ymd("20110630")
"2011-06-30 UTC"

> dmy("30/06/2011")
"2011-06-30 UTC"

> mdy_hms("063020111111111")
"2011-06-30 11:11:11 UTC"
```

```
> day(ymd("20110630"))
30

> year(dmy("30/06/2011"))
2011

> hour(mdy_hms("063021100111111"))
11
```

Libraries for cleaning data

- `stringr`: `str_detect`, `str_replace`, ...
- `lubridate`: `ymd`, `mdy`, `dmy`, `hms`, `ymd_hms`, ...
- `tidyr`: `gather`, `spread`, `unite`, `separate`, ...
- ...

Ways to clean data

- Edit text value for a single variable to be consistent (spelling, capitalization, spacing)
- Standardize units of measurement for a single variable
- Remove duplicate rows (common) or columns (less frequent)