Data type constraints

CLEANING DATA IN R

Maggie Matsui
Content Developer @ DataCamp

Checking data types

```
is.numeric(sales$revenue)
```

```
FALSE
```

```
library(assertive)
assert_is_numeric(sales$revenue)
```

```
Error: is_numeric : sales$revenue is not of class 'numeric'; it has class 'character'.
```

```
assert_is_numeric(sales$quantity)
```

```

```

Checking data types

Logical checking - returns `TRUE` / `FALSE`

- `is.character()`
- `is.numeric()`
- `is.logical()`
- `is.factor()`
- `is.Date()`
- ...

`assertive` checking - errors when `FALSE`

- `assert_is_character()`
- `assert_is_numeric()`
- `assert_is_logical()`
- `assert_is_factor()`
- `assert_is_date()`
- ...

Converting data types

- `as.character()`
- `as.numeric()`
- `as.logical()`
- `as.factor()`
- `as.Date()`
- ...

Watch out: factor to numeric

```
product_type
```

```
1000 1000 3000 2000 3000
Levels: 1000 2000 3000
```

```
class(product_type)
```

```
"factor"
```

```
as.numeric(product_type)
```

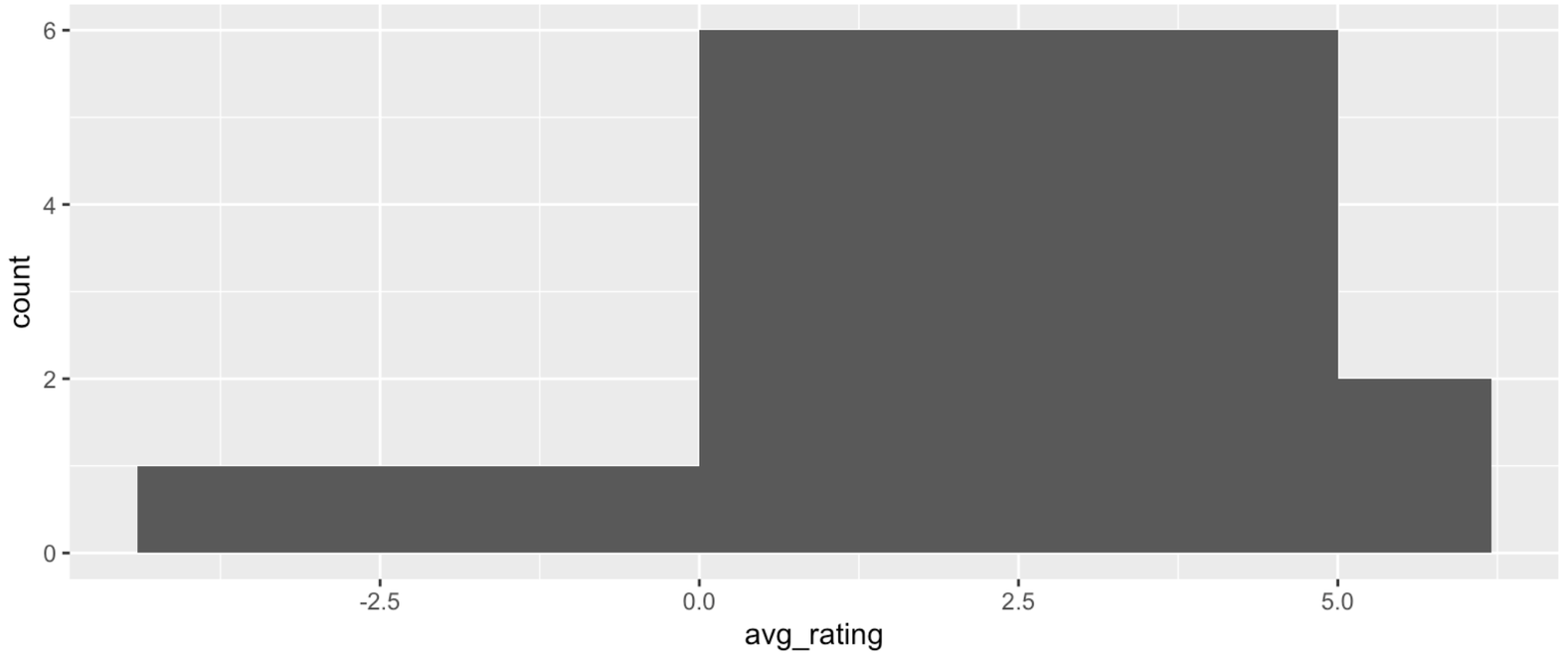
```
1 1 3 2 3
```

```
as.numeric(as.character(product_type))
```

```
1000 1000 3000 2000 3000
```

Finding out of range values

```
breaks <- c(min(movies$avg_rating), 0, 5, max(movies$avg_rating))
ggplot(movies, aes(avg_rating)) +
    geom_histogram(breaks = breaks)
```



Finding out of range values

```
library(assertive)
assert_all_are_in_closed_range(movies$avg_rating, lower = 0, upper = 5)
```

```
Error: is_in_closed_range : movies$avg_rating are not all in the range [0,5].
There were 3 failures:
  Position Value    Cause
1        5   5.8 too high
2        8   6.2 too high
3        9  -4.4  too low
```

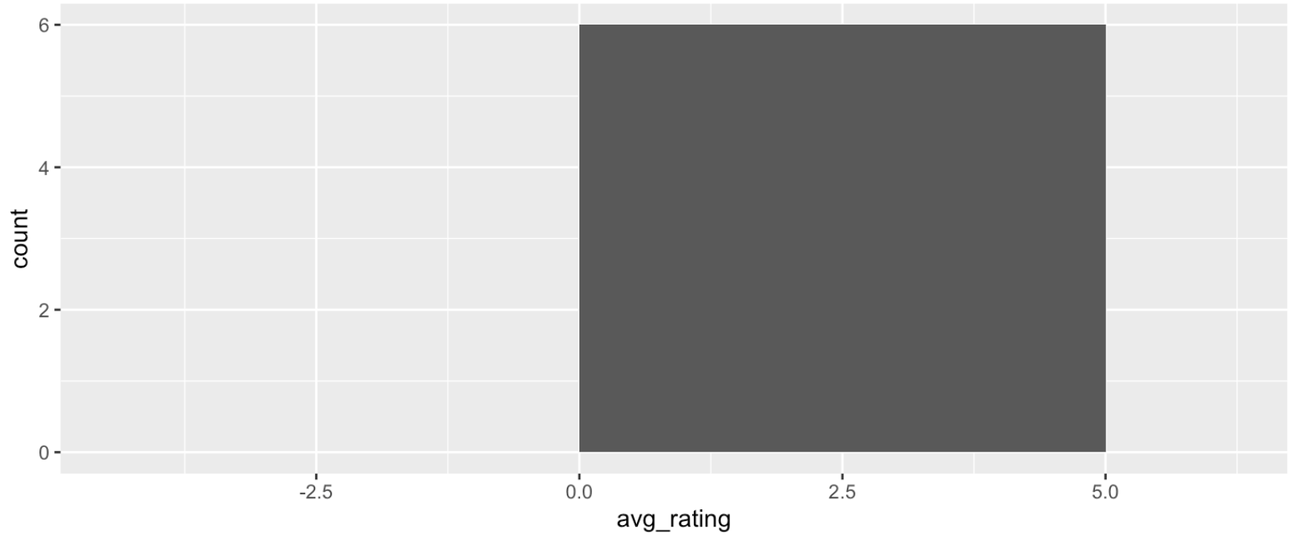
Handling out of range values

- Remove rows
- Treat as missing (`NA`)
- Replace with range limit
- Replace with other value based on domain knowledge and/or knowledge of dataset

Removing rows

```
movies %>%
  filter(avg_rating >= 0, avg_rating <= 5) %>%

  ggplot(aes(avg_rating)) +
  geom_histogram(breaks = c(min(movies$avg_rating), 0, 5, max(movies$avg_rating)))
```



Treat as missing

```
movies
```

```
  title                    avg_rating
  <chr>                         <dbl>
1 A Beautiful Mind                4.1
2 La Vita e Bella                 4.3
3 Amelie                          4.2
4 Meet the Parents                3.5
5 Unbreakable                     5.8
6 Gone in Sixty Seconds           3.3
...
```

```
replace(col, condition, replacement)
```

```
movies %>%
  mutate(rating_miss =
    replace(avg_rating, avg_rating > 5, NA))
```

```
  title                    rating_miss
  <chr>                         <dbl>
1 A Beautiful Mind                4.1
2 La Vita e Bella                 4.3
3 Amelie                          4.2
4 Meet the Parents                3.5
5 Unbreakable                      NA
6 Gone in Sixty Seconds           3.3
...
```

Replacing out of range values

```
heart_rate %>%
  mutate(rating_const =
           replace(avg_rating, avg_rating > 5, 5))
```

```
  title                rating_const
  <chr>                       <dbl>
1 A Beautiful Mind              4.1
2 La Vita e Bella               4.3
3 Amelie                        4.2
4 Meet the Parents              3.5
5 Unbreakable                   5.0
6 Gone in Sixty Seconds         3.3
...
```

Date range constraints

```
assert_all_are_in_past(movies$date_recorded)
```

```
Error: is_in_past : movies$date_recorded are not all in the past.
There was 1 failure:
  Position               Value     Cause
1        3 2064-09-22 20:00:00 in future
```

```
library(lubridate)
heart_rate %>%
  filter(date_recorded > today())
```

```
    title  avg_rating  date_recorded
1  Amelie         4.2  2064-09-23
```

Removing out-of-range dates

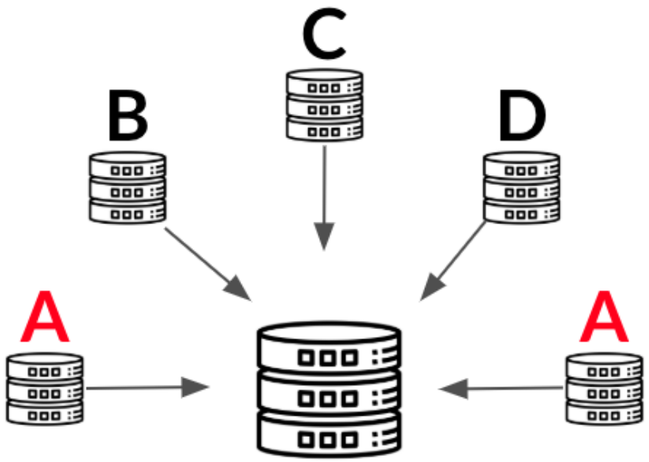
```
library(lubridate)
movies <- movies %>%
  filter(date_recorded <= today())
```

```
library(assertive)
assert_all_are_in_past(movies$date_recorded)
```

Remember, no output = passed!

Why do duplicates occur?

Data Entry &
Human Error



Join or merge
Errors

Bugs and design
errors

Finding full duplicates

```
duplicated(credit_scores)
```

```
FALSE FALSE FALSE TRUE FALSE ...
```

```
sum(duplicated(credit_scores))
```

```
2
```

Finding full duplicates

```
filter(credit_scores, duplicated(credit_scores))
```

```
  first_name last_name                  address credit_score
1     Miriam       Day 6042 Sollicitudin Avenue          313
2     Katell       Roy      Ap #434-4081 Mi Av.          455
```

Factors

- In a `factor`, each category is stored as a number number and has a corresponding label

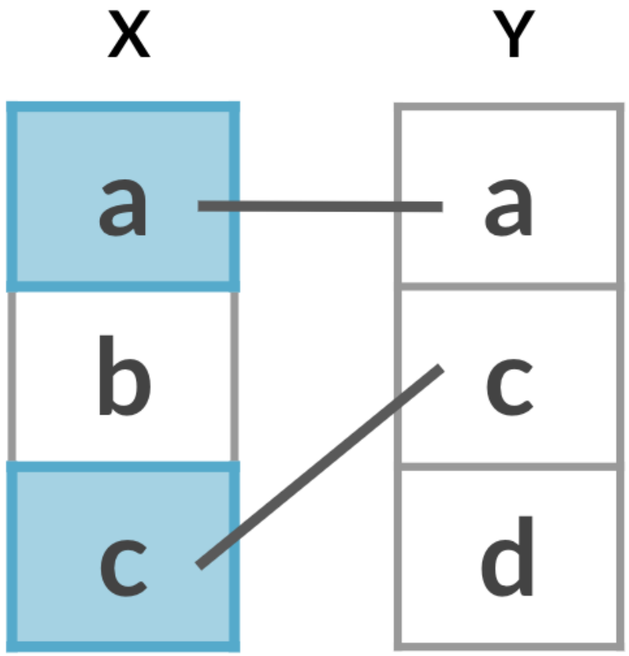
Data	Labels	Numeric representation
Marriage status	`unmarried`, `married`	`1`, `2`
Household income category	`0-20K`, `20-40K`, ...	`1`, `2`, ...
T-shirt size	`S`, `M`, `L`, `XL`	`1`, `2`, `3`, `4`

Filtering joins: a quick review

- Keeps or removes observations from the first table without adding columns

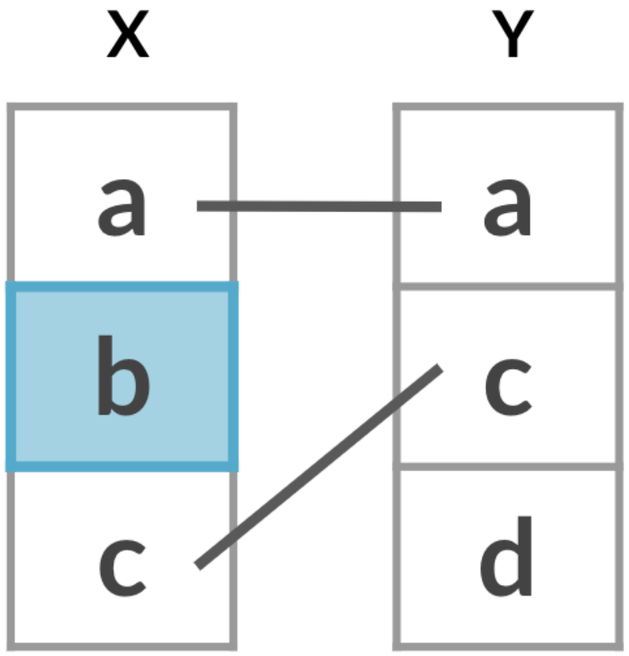
Semi-join

*What observations of X are **also** in Y?*

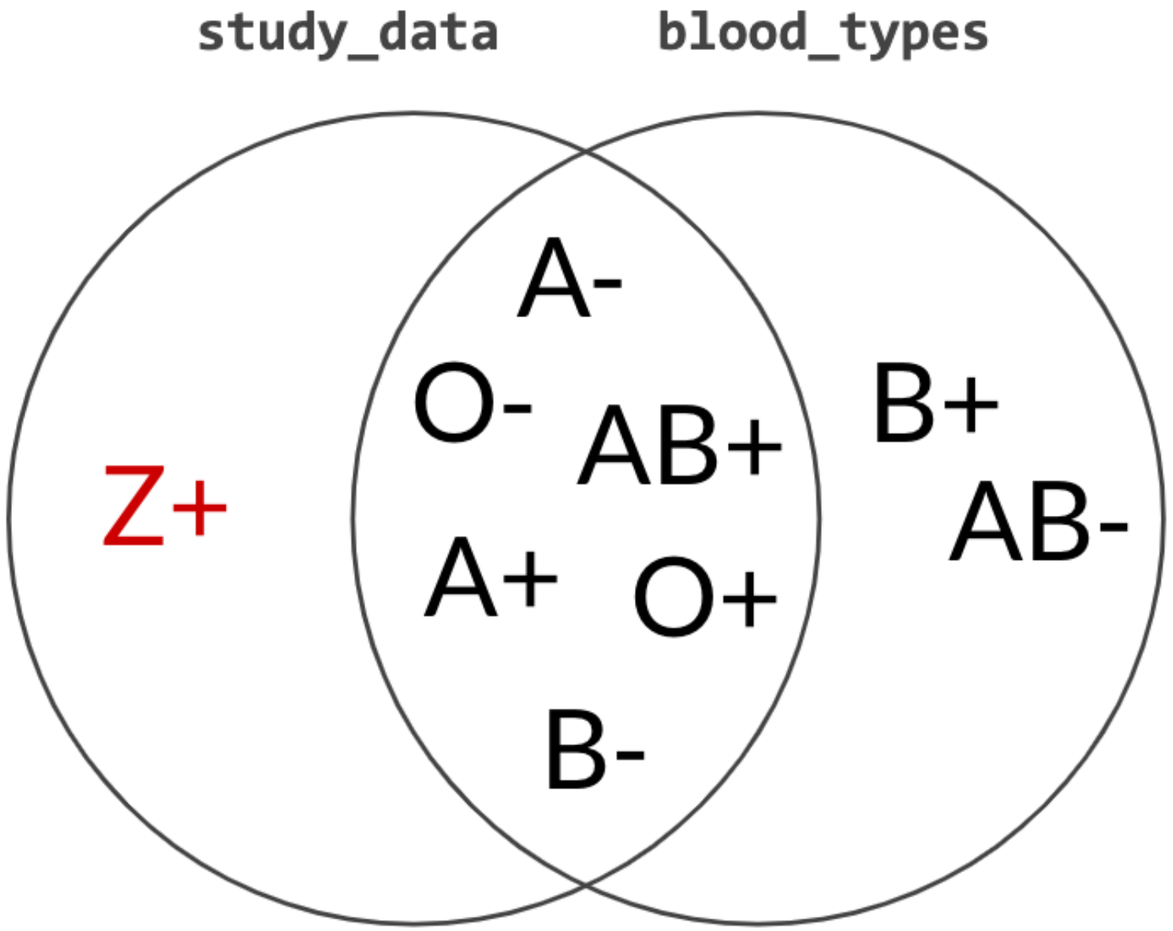


Anti-join

*What observations of X are **not** in Y?*



Finding non-members

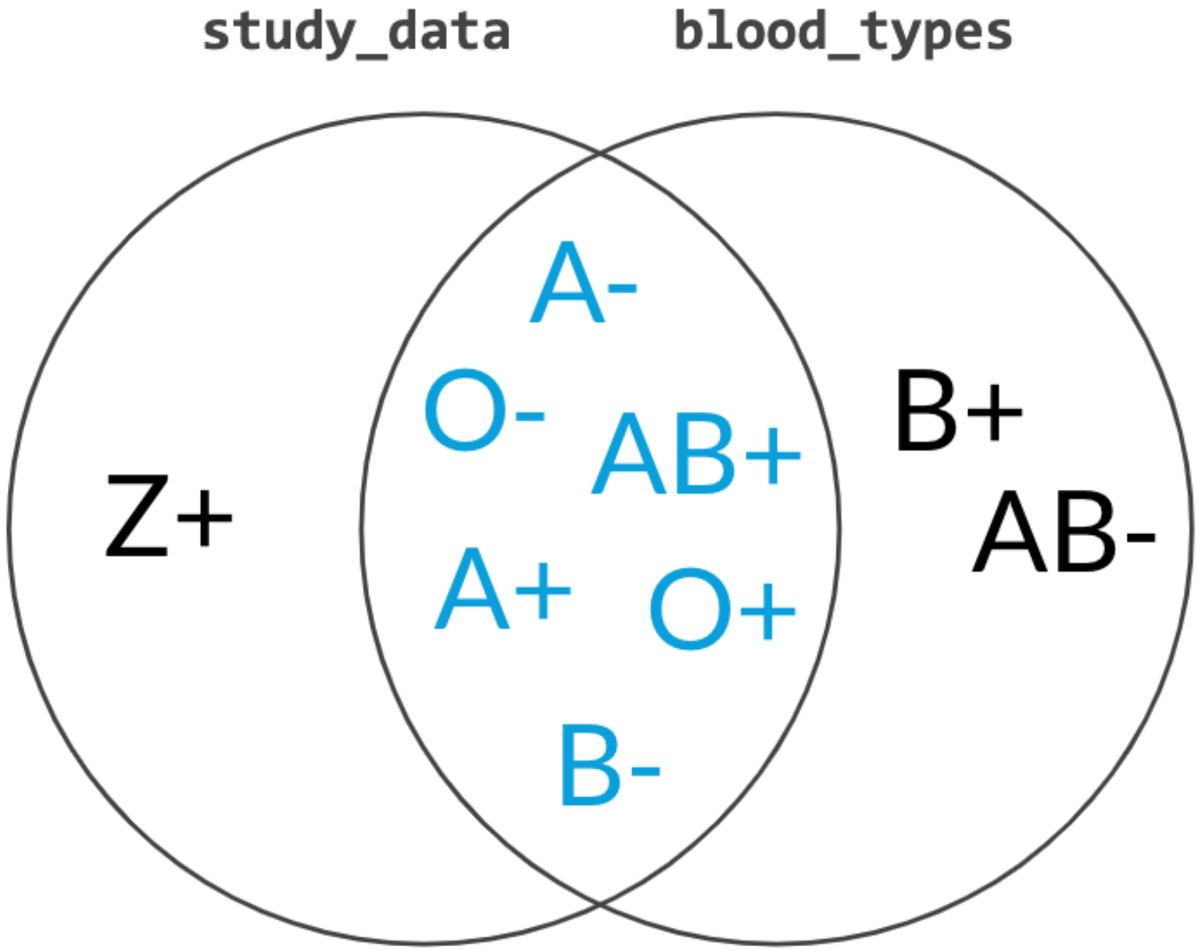


Anti-join

```
study_data %>%
  anti_join(blood_types, by = "blood_type")
```

```
      name   birthday blood_type
1 Jennifer 2019-12-17         Z+
```

Removing non-members



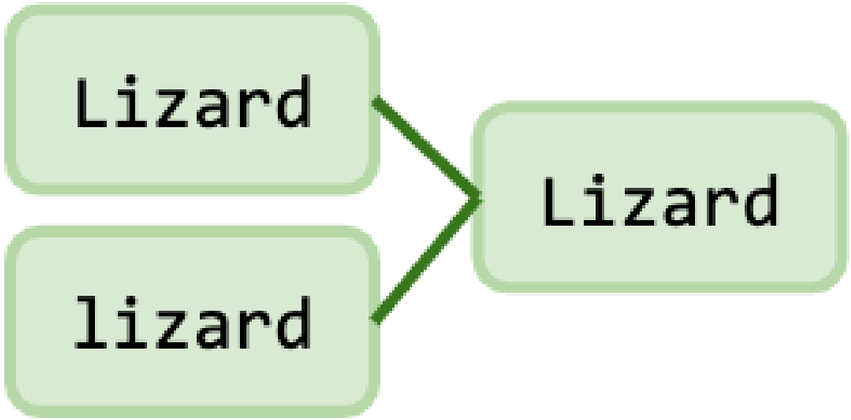
Semi-join

```
study_data %>%
  semi_join(blood_types, by = "blood_type")
```

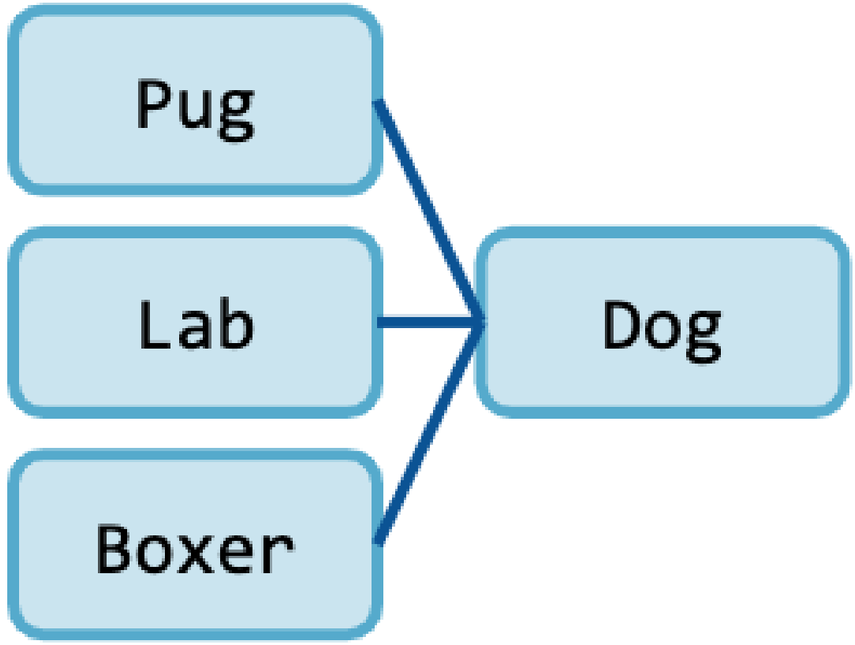
```
      name   birthday blood_type
1     Beth 2019-10-20         B-
2 Ignatius 2020-07-08         A-
3     Paul 2019-08-12         O+
4    Helen 2019-03-17         O-
5  Kennedy 2020-04-27         A+
6    Keith 2019-04-19        AB+
```

Categorical data problems

Inconsistency within a category



Too many categories



Collapsing categories

```
other_categories =  c("amphibian", "fish", "bug", "invertebrate", "reptile")
```

```
library(forcats)
animals %>%
  mutate(type_collapsed = fct_collapse(type_trimmed, other = other_categories))
```

```
  animal_name  hair  eggs  fins  legs  tail  type_trimmed  type_collapsed
  <fct>       <int> <int>  <int> <int> <int> <chr>         <chr>
1 mole            1     0     0     4     1  mammal        mammal
2 chicken         0     1     0     2     1  bird          bird
3 capybara        1     0     0     2     1  mammal        mammal
4 tuna            0     1     1     0     1  fish          other
5 ostrich         0     1     0     2     1  bird          bird
```

More complex text problems

- A ***regular expression*** is a sequence of characters that allows for robust searching within a string.
- Certain characters are treated differently in a regular expression:
  - `(`, `)`, `[`, `]`, `$`, `.`, `+`, `*`, and others
- `stringr` functions use regular expressions
- Searching for these characters requires using `fixed()`:
  - `str_detect(column, fixed("$"))`

Learn more in String Manipulation with stringr in R & Intermediate Regular Expressions in R

Uniformity

CLEANING DATA IN R

Maggie Matsui
Content Developer @ DataCamp

Uniformity

- Different units or formats
  - **Temperature:** `°C` vs. `°F`
  - **Weight:** `kg` vs. `g` vs. `lb`
  - **Money:** USD `$` vs. GBP `£` vs. JPY `¥`
  - **Date:** `DD-MM-YYYY` vs. `MM-DD-YYYY` vs. `YYYY-MM-DD`

Parsing multiple formats

```
library(lubridate)
parse_date_time(nyc_temps$date,
                orders = c("%Y-%m-%d", "%m/%d/%y", "%B %d, %Y"))
```

```
"2019-11-23 UTC" "2019-01-15 UTC" "2019-04-24 UTC" "2019-08-30 UTC"
"2019-10-03 UTC" "2019-03-17 UTC"
```

```
parse_date_time("Monday, January 3",
                orders = c("%Y-%m-%d", "%m/%d/%y", "%B %d, %Y"))
```

```
NA
```

Validating numbers

```
credit_cards %>%
  mutate(theoretical_total = dining_cb + groceries_cb + gas_cb) %>%
  filter(theoretical_total != total_cb) %>%
  select(dining_cb:theoretical_total)
```

```
  dining_cb groceries_cb  gas_cb total_cb theoretical_total
1     98.50        283.68  281.70   788.33            663.88
2   3387.53        363.85 2706.42  4502.94           6457.80
```

Calculating age

```
library(lubridate)
date_difference <- as.Date("2015-09-04") %--% today()
date_difference
```

```
2015-09-04 UTC--2020-03-09 UTC
```

```
as.numeric(date_difference, "years")
```

```
4.511978
```

```
floor(as.numeric(date_difference, "years"))
```

```
4
```

Validating age

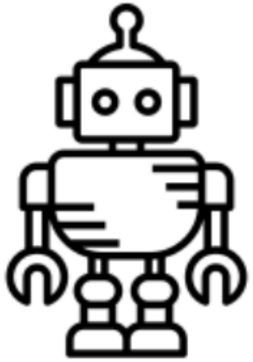
```
credit_cards %>%
  mutate(theor_age = floor(as.numeric(date_opened %--% today(), "years"))) %>%
  filter(theor_age != acct_age)
```

```
  date_opened acct_age dining_cb groceries_cb  gas_cb total_cb theor_age
1  2016-03-25        4    814.34       471.58 3167.41  4453.33         3
2  2018-03-06        3    238.48       186.05  213.84   638.37         2
```

What is missing data?

Occurs when no data value is stored for a variable in an observation

Can be represented as `NA`, `nan`, `0`, `99`, `.` ...

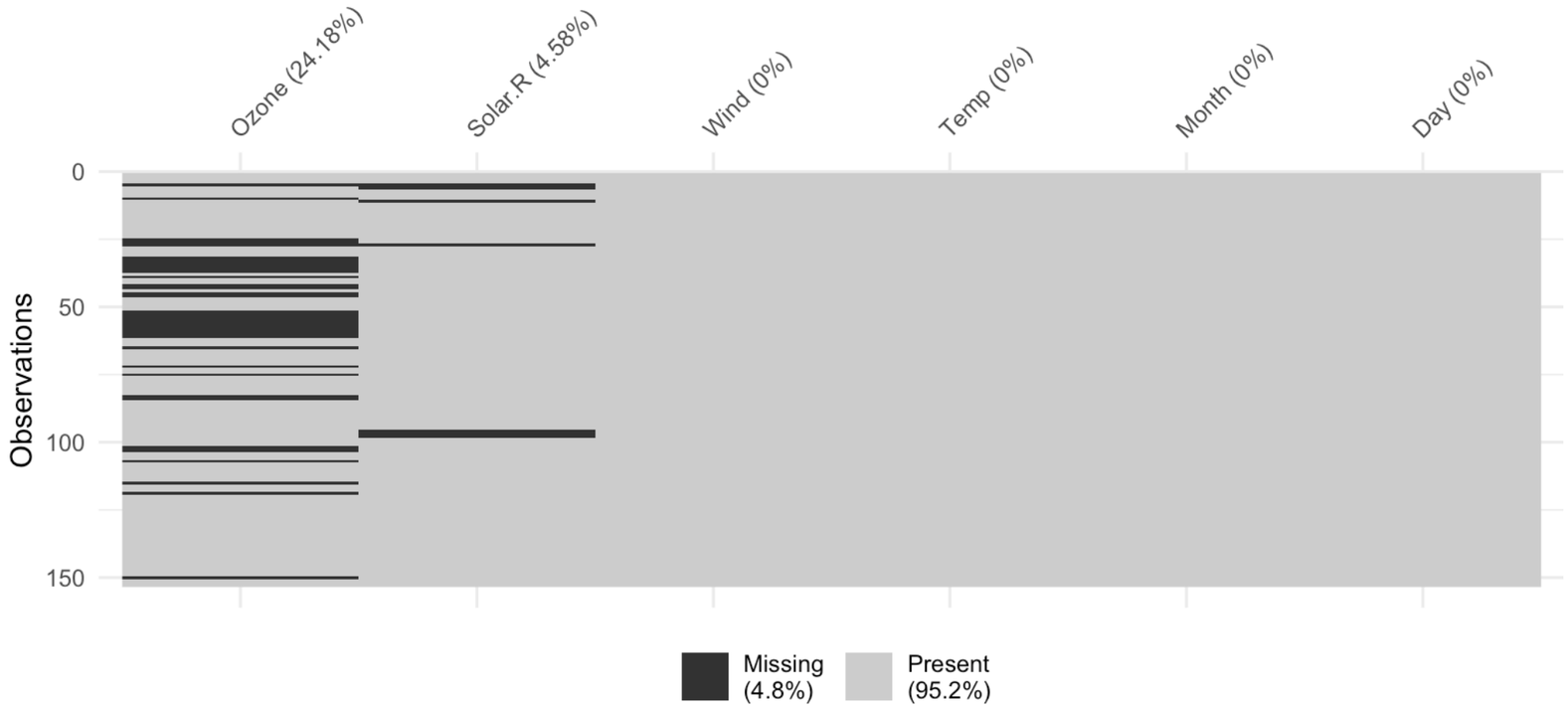


Technical error

Human error

Visualizing missing values

```
library(visdat)
vis_miss(airquality)
```



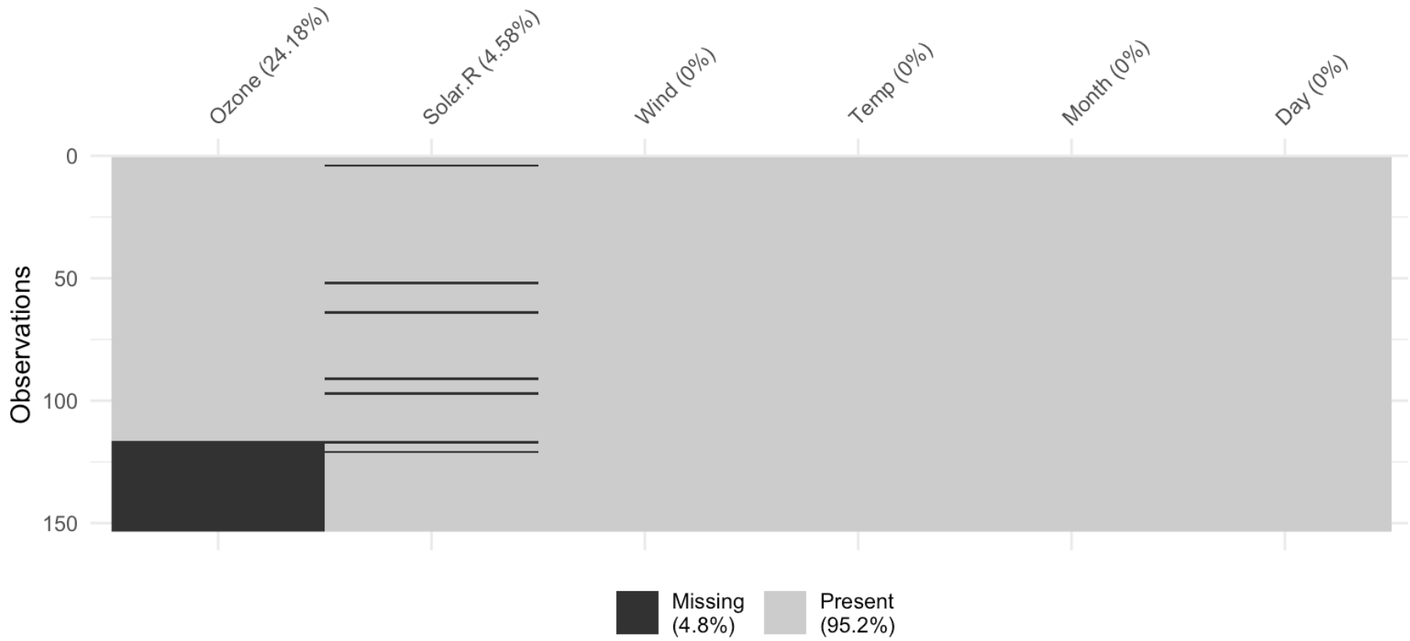
Investigating missingness

```
airquality %>%
  mutate(miss_ozone = is.na(Ozone)) %>%
  group_by(miss_ozone) %>%
  summarize_all(median, na.rm = TRUE)
```

```
  miss_ozone Ozone Solar.R  Wind  Temp Month   Day
  <lgl>      <dbl>   <int> <dbl> <dbl> <dbl> <dbl>
1 FALSE       31.5     207   9.7    65     7    16
2 TRUE        NA       194   9.7    99     6    15
```

Investigating missingness

```
airquality %>%
  arrange(Temp) %>%
  vis_miss()
```



Types of missingness

Missing Completely at Random

(MCAR)

No systematic relationship between missing data and other values

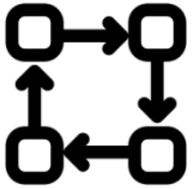
Data entry errors when inputting data

Missing at Random

(MAR)

Systematic relationship between missing data and other observed values

Missing ozone data for high temperatures



Missing Not at Random

(MNAR)

Systematic relationship between missing data and unobserved values

Missing temperature values for high temperatures

Dealing with missing data

Simple approaches:

1. Drop missing data
2. Impute (fill in) with statistical measures *(mean, median, mode..)* or domain knowledge

More complex approaches:

1. Impute using an algorithmic approach
2. Impute with machine learning models

Learn more in ***Dealing with Missing Data in R***

Dropping missing values

```
airquality %>%
  filter(!is.na(Ozone), !is.na(Solar.R))
```

```
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    41     190   7.4    67     5     1
 2    36     118     8    72     5     2
 3    12     149  12.6    74     5     3
 4    18     313  11.5    62     5     4
 5    23     299   8.6    65     5     7
 6    19      99  13.8    59     5     8
```

Replacing missing values

```
airquality %>%
  mutate(ozone_filled = ifelse(is.na(Ozone), mean(Ozone, na.rm = TRUE), Ozone))
```

```
   Ozone Solar.R  Wind  Temp Month   Day ozone_filled
   <int>   <int> <dbl> <int> <int> <int>         <dbl>
 1    41     190   7.4    67     5     1          41
 2    36     118   8      72     5     2          36
 3    12     149  12.6    74     5     3          12
 4    18     313  11.5    62     5     4          18
 5    NA      NA  14.3    56     5     5          42.1
```

Minimum edit distance

How many typos are needed to get from one string to another?

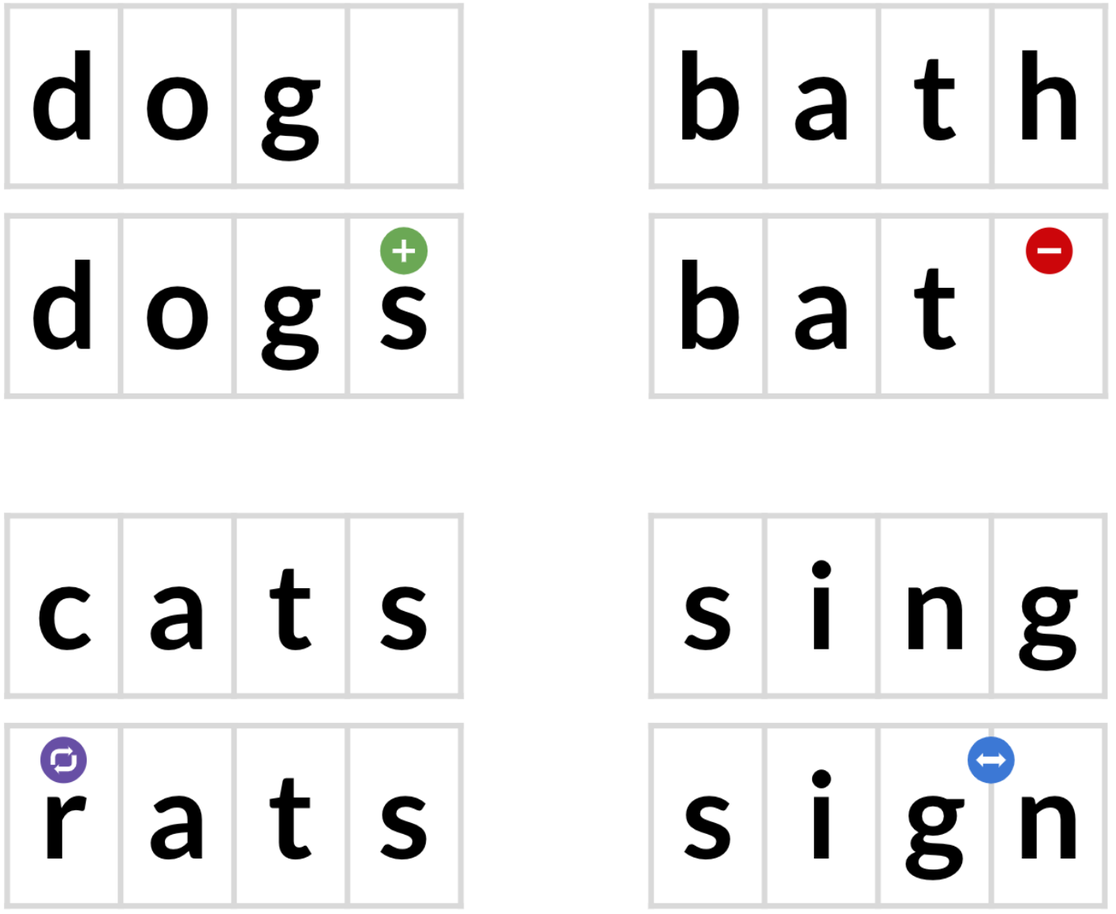
Insertion

Deletion

Substitution

Transposition

Edit distance = 1

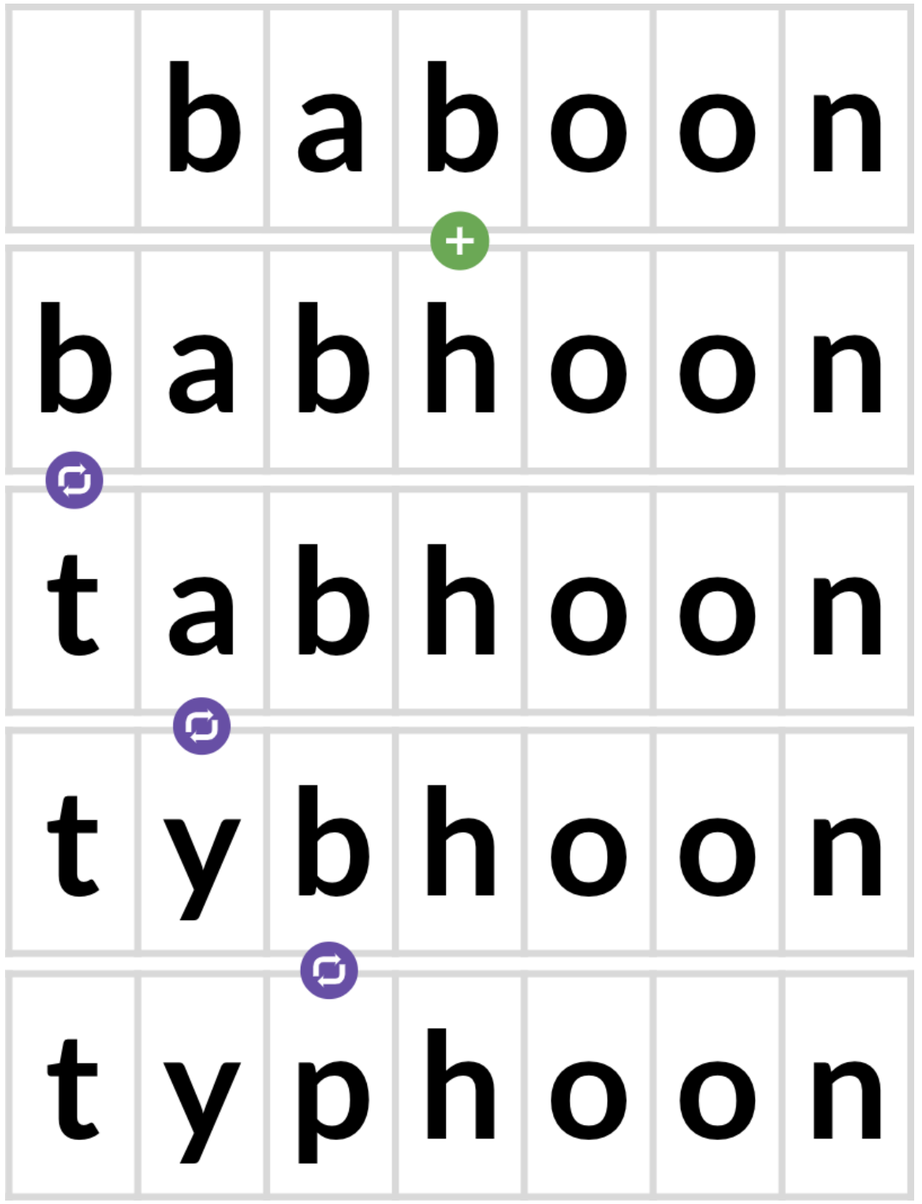


A more complex example

baboon* → *typhoon

- Insert **h**
- Substitute **b** → **t**
- Substitute **a** → **y**
- Substitute **b** → **p**

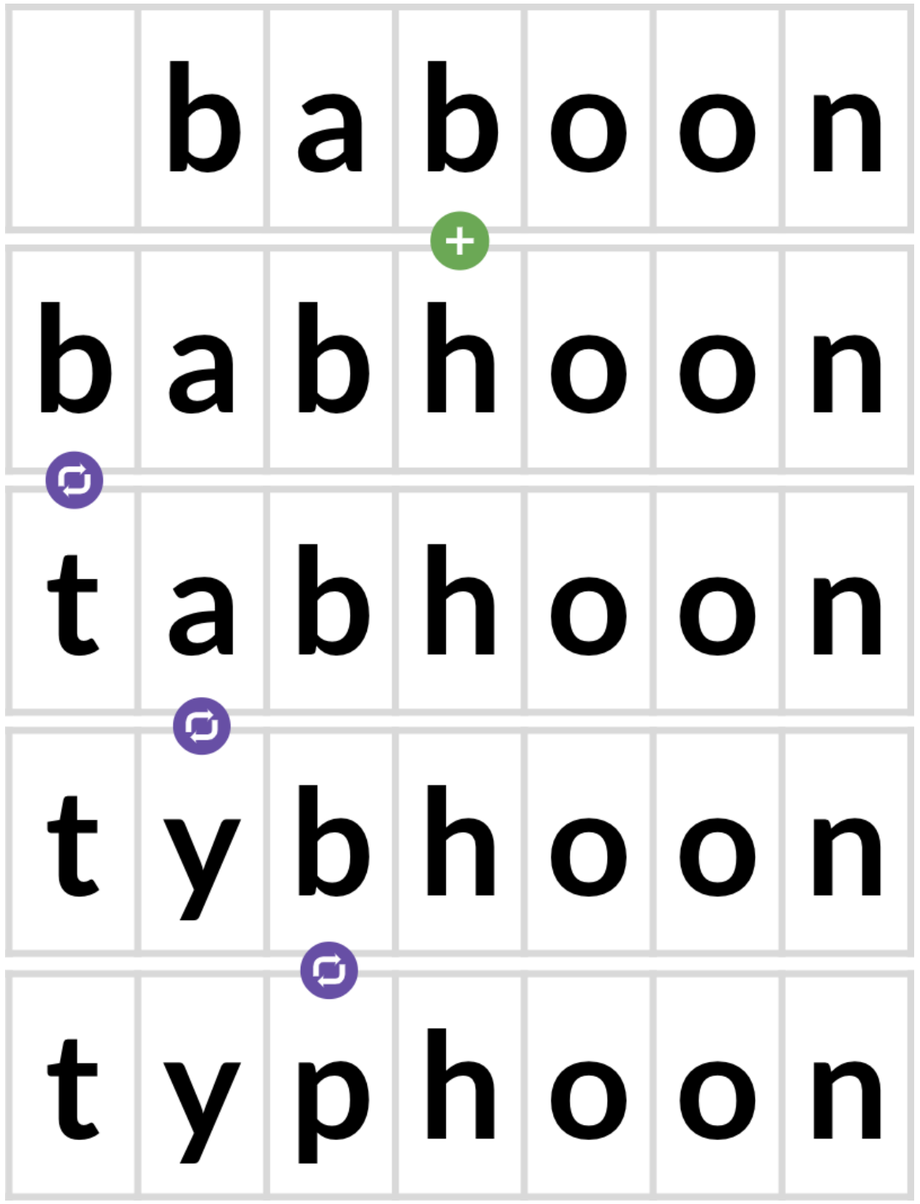
Total: 4



String distance in R

```
library(stringdist)
stringdist("baboon",
           "typhoon",
           method = "dl")
```

```
4
```



Other methods

```
# LCS
stringdist("baboon", "typhoon",
           method = "lcs")
```

```
7
```

```
# Jaccard
stringdist("baboon", "typhoon",
           method = "jaccard")
```

```
0.75
```

Remapping using string distance

```
library(fuzzyjoin)
stringdist_left_join(survey, cities, by = "city", method = "dl")
```

```
        city.x move_score      city.y
1       chicgo          4     chicago
2   los angles          4 los angeles
3      chicogo          5     chicago
4      new yrk          5    new york
5     new yoork         2    new york
6      seatttle         3     seattle
7   losangeles          4 los angeles
8      seeatle          2     seattle
9      siattle          1     seattle
...
```

Remapping using string distance

```
stringdist_left_join(survey, cities, by = "city", method = "dl", max_dist = 1)
```

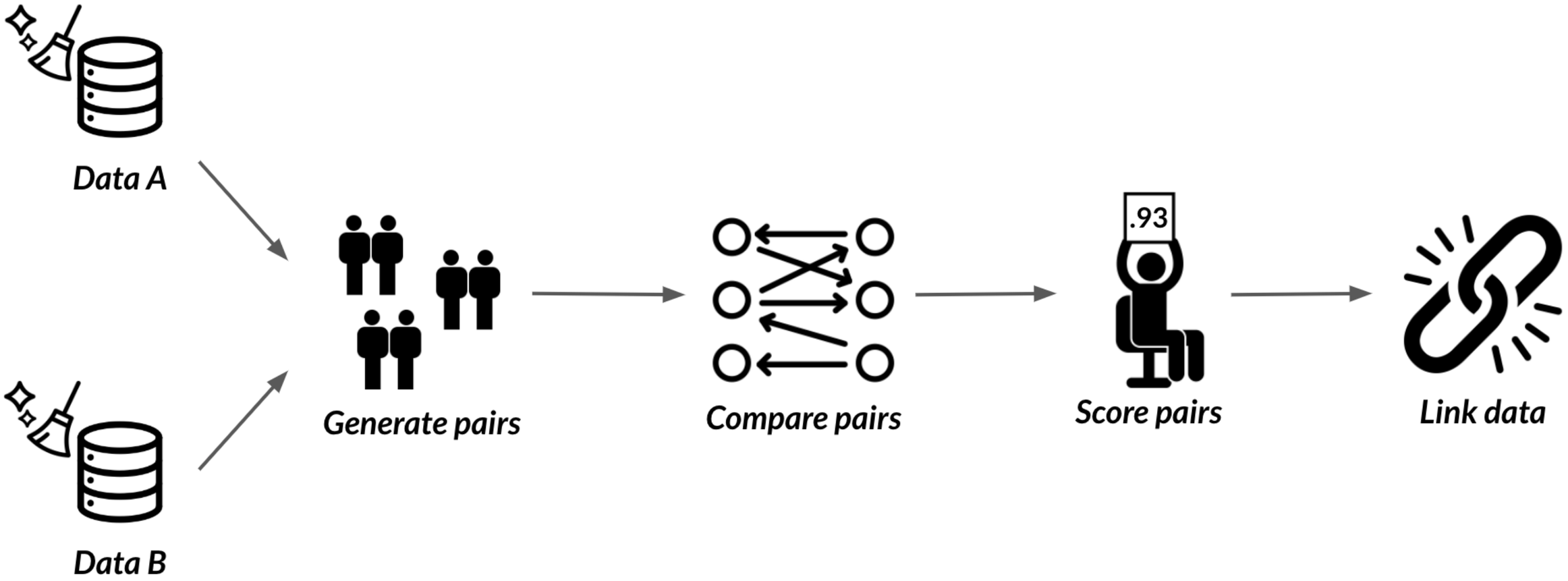
```
        city.x move_score      city.y
1       chicgo          4     chicago
2   los angles          4 los angeles
3      chicogo          5     chicago
4      new yrk          5    new york
5     new yoork         2    new york
6      seatttle         3     seattle
7   losangeles          4 los angeles
8      seeatle          2        <NA>
9      siattle          1     seattle
...
```

When joins won't work

Event	Time
Houston Rockets vs Chicago Bulls	19:00
Miami Heat vs Los Angeles Lakers	19:00
Brooklyn Nets vs Orlando Magic	20:00
Denver Nuggets vs Miami Heat	21:00
San Antonio Spurs vs Atlanta Hawks	21:00

Event	Time
NBA: Nets vs Magic	8pm
NBA: Bulls vs Rockets	9pm
NBA: Heat vs Lakers	7pm
NBA: Grizzlies vs Heat	10pm ✓
NBA: Heat vs Cavaliers	9pm ✓

What is record linkage?



Generating pairs in R

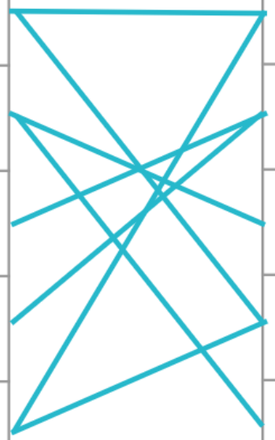
```
library(reclin)
pair_blocking(df_A, df_B)
```

```
Simple blocking
  No blocking used.
  First data set:  5 records
  Second data set: 5 records
  Total number of pairs: 25 pairs
ldat with 25 rows and 2 columns
   x y
1  1 1
2  2 1
3  3 1
...
```

Blocking

`df_A`

Name	Zip	State
Christine M. Conner	10456	NY
Keaton Z Snyder	15020	PA
Arthur Potts	07799	NJ
Maia Collier	07960	NJ
Atkins, Alice W.	10603	NY
...		...



`df_B`

Name	Zip	State
Jerome A. Yates	11743	NY
Garrison, Brenda	08611	NJ
Keaton Snyder	15020	PA
Stuart, Bert F	12211	NY
Hayley Peck	19134	PA
...		...

Only consider pairs when they agree on the blocking variable (State)

Pair blocking in R

```
pair_blocking(df_A, df_B, blocking_var = "state")
```

```
Simple blocking
  Blocking variable(s): state
  First data set:  5 records
  Second data set: 5 records
  Total number of pairs: 8 pairs
```

```
ldat with 8 rows and 2 columns
  x y
1 1 1
2 1 4
3 2 3
4 2 5
5 3 2
6 4 2
7 5 1
8 5 4
```

Comparing multiple columns

```
pair_blocking(df_A, df_B, blocking_var = "state") %>%
  compare_pairs(by = c("name", "zip"), default_comparator = lcs())
```

```
Compare                                  ldat with 8 rows and 4 columns
  By: name, zip                            x y      name zip
                                         1 1 1 0.3529412 0.4
Simple blocking                          2 1 4 0.3030303 0.2
  Blocking variable(s): state            3 2 3 0.9285714 1.0
  First data set:  5 records             4 2 5 0.2962963 0.2
  Second data set: 5 records             ...
  Total number of pairs: 8 pairs         8 5 4 0.3333333 0.2
```

Different comparators

- `default_comparator = lcs()`
- `default_comparator = jaccard()`
- `default_comparator = jaro_winkler()`

```
pair_blocking(df_A, df_B, blocking_var = "state") %>%
  compare_pairs(by = c("name", "zip"), default_comparator = lcs()) %>%
  score_simsum()
```

```
  x y       name zip    simsum
1 1 1 0.3529412 0.4 0.7529412
2 1 4 0.3030303 0.2 0.5030303
3 2 3 0.9285714 1.0 1.9285714
4 2 5 0.2307692 0.2 0.4307692
5 3 2 0.2142857 0.2 0.4142857
6 4 2 0.2857143 0.4 0.6857143
7 5 1 0.1935484 0.4 0.5935484
8 5 4 0.3333333 0.2 0.5333333
```

```
pair_blocking(df_A, df_B, blocking_var = "state") %>%
  compare_pairs(by = c("name", "zip"), default_comparator = lcs()) %>%
  score_simsum()
```

```
  x y       name zip    simsum
1 1 1 0.3529412 0.4 0.7529412
2 1 4 0.3030303 0.2 0.5030303
3 2 3 0.9285714 1.0 1.9285714  <--
4 2 5 0.2307692 0.2 0.4307692
5 3 2 0.2142857 0.2 0.4142857
6 4 2 0.2857143 0.4 0.6857143
7 5 1 0.1935484 0.4 0.5935484
8 5 4 0.3333333 0.2 0.5333333
```

Disadvantages of summing

- 2 records with a similar name (Keaton Z Snyder & Keaton Snyder) are more likely to be a match
- 2 records with the same sex (Male & Male) are not as likely to be a match
- Use probabilistic scoring!

Scoring probabilistically

```
pair_blocking(df_A, df_B, blocking_var = "state") %>%
  compare_pairs(by = c("name", "zip"), default_comparator = lcs()) %>%
  score_problink()
```

```
  x y      name zip    weight
1 1 1 0.3529412 0.4 -1.011599
2 1 4 0.3030303 0.2 -2.219198
3 2 3 0.9285714 1.0 16.019278
4 2 5 0.2307692 0.2 -2.590260
5 3 2 0.2142857 0.2 -2.685570
6 4 2 0.2857143 0.4 -1.321753
7 5 1 0.1935484 0.4 -1.832576
8 5 4 0.3333333 0.2 -2.079436
```

Selecting matches

```
pair_blocking(df_A, df_B, blocking_var = "state") %>%
  compare_pairs(by = c("name", "zip"), default_comparator = lcs()) %>%
  score_problink() %>%
  select_n_to_m()
```

```
  x y      name zip    weight select
1 1 1 0.3529412 0.4 -1.011599  FALSE
2 1 4 0.3030303 0.2 -2.219198  FALSE
3 2 3 0.9285714 1.0 16.019278   TRUE
4 2 5 0.2307692 0.2 -2.590260  FALSE
5 3 2 0.2142857 0.2 -2.685570  FALSE
6 4 2 0.2857143 0.4 -1.321753  FALSE
...
```

Linking the data

```
pair_blocking(df_A, df_B, blocking_var = "state") %>%
  compare_pairs(by = c("name", "zip"), default_comparator = lcs()) %>%
  score_problink() %>%
  select_n_to_m() %>%
  link()
```

Linked data

	name.x	zip.x	state.x	name.y	zip.y	state.y
1	Keaton Z Snyder	15020	PA	Keaton Snyder	15020	PA
2	Christine M. Conner	10456	NY	<NA>	<NA>	<NA>
3	Arthur Potts	07799	NJ	<NA>	<NA>	<NA>
4	Maia Collier	07960	NJ	<NA>	<NA>	<NA>
5	Atkins, Alice W.	10603	NY	<NA>	<NA>	<NA>
6	<NA>	<NA>	<NA>	Jerome A. Yates	11743	NY
7	<NA>	<NA>	<NA>	Garrison, Brenda	08611	NJ
8	<NA>	<NA>	<NA>	Stuart, Bert F	12211	NY
9	<NA>	<NA>	<NA>	Hayley Peck	19134	PA

Expand and build upon your new skills

- Categorical Data
  - Categorical Data in the Tidyverse
- Text Data
  - String Manipulation with stringr in R
  - Intermediate Regular Expressions in R
- Writing Clean Code
  - Defensive R Programming