**#COMMON DATA PROBLEMS**

-To see data type for each column :

**\*glimpse(nama\_file) or str(nama\_file) or class(nama\_kolom)**

-tipe data dbl (double) : tipe data numeric dengan 2 angka decimal di belakang koma.

-checking data type :

**\*is.numeric(nama\_kolom) : hasilnya akan berupa TRUE or FALSE**

-another check type of data :

**\*library(assertive)**

**\*assert\_is\_numeric(nama\_kolom)** : hasil yg keluar akan membeitahukan tipe data kolom trsbt jika salah. Jika tidak maka akan blank saja

-Character to Number :

**\*library(stringr)**

**\*nama\_kolom\_baru = as.numeric(str\_remove(nama\_kolom, “,”))**

\*str remove untuk menghilangkan koma

-Convert factor to numeric :

**\*as.numeric(as.character(nama\_kolom))**

**-Setting the trimmed exercise :**

bike\_share\_rides <- bike\_share\_rides %>%

  # Remove 'minutes' from duration: duration\_trimmed

  mutate(duration\_trimmed = str\_remove(duration, "minutes"),

         # Convert duration\_trimmed to numeric: duration\_mins

         duration\_mins = as.numeric(duration\_trimmed))

# Glimpse at bike\_share\_rides

glimpse(bike\_share\_rides)

# Assert duration\_mins is numeric

assert\_is\_numeric(bike\_share\_rides$duration\_mins)

# Calculate mean duration

mean(bike\_share\_rides$duration\_mins)

-Finding out of range value using histogram :

**\*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 value using table :

**\*assert\_all\_are\_in\_closed\_range(nama\_kolom, lower = 0, upper = 5)**

-Handling out of range values :

1.Remove Rows (if small proportion)

2.Treas As NA

3.Replace with range limit (e.g value is 6 change to 5 that the limit of number)

4.Replace with other value (e.g mean of the data)

-Removing Rows :

**\*filter(avg\_rating >=0, avg\_rating <=5)**

-Treat as NA :

**\*replace(nama\_kolom, condition, replacement)**

-Date Range Constraint : menguji tanggal yg ada di masa depan

**\*assert\_all\_are\_in\_past(nama\_kolom) or filter(nama\_kolom > today())**

-Ride Duration Constraint :

# Create breaks

breaks <- c(min(bike\_share\_rides$duration\_min), 0, 1440, max(bike\_share\_rides$duration\_min))

# Create a histogram of duration\_min

ggplot(bike\_share\_rides, aes(duration\_min)) +

geom\_histogram(breaks = breaks)

-exercise back to the future :

library(lubridate)

# Convert date to Date type

bike\_share\_rides <- bike\_share\_rides %>%

  mutate(date = as.Date(date))

# Make sure all dates are in the past

assert\_all\_are\_in\_past(bike\_share\_rides$date)

# Filter for rides that occurred before or on today's date

bike\_share\_rides\_past <- bike\_share\_rides %>%

  filter(date <= today())

# Make sure all dates from bike\_share\_rides\_past are in the past

assert\_all\_are\_in\_past(bike\_share\_rides\_past$date)

-Finding full duplicates :

**\*duplicated(nama\_kolom)**

-Mencari jumlah data duplicate :

**\*sum(duplicated(nama\_kolom))**

-mencari row yg memiliki duplicate :

**\*filter(nama\_kolom, duplicated(nama\_kolom))**

-Dropping full duplicates :

**\*credit\_scores\_unique <- distinct (credit\_scores)**

-Finding Partial Duplicate :

**\*dups\_id <- credit\_scores %>%**

**\*count(first\_name, last\_name) %>%**

**\*filter (n>1)**

**\*credit\_scores %>%**

**\*filter(credit\_scores, first\_name %in% dup\_ids$first\_name, last\_name %in% dup\_ids$last\_name)**

**-Dropping Partial Duplicate :**

**\*credit\_scores %>%  
\*distinct(first\_name, last\_name, .keep\_all = TRUE)**

**-Another way handle partial duplicate :**

**\*credit\_score %>%**

**\*group\_by(first\_name, last\_name) %>%**

**\*mutate(mean\_score = mean(credit\_score) %>%**

**\*distinct(first\_name, last\_name, .keep\_all = TRUE)**

**\*select(-credit\_score)**

-Full Duplicate Exercise :

# Count the number of full duplicates

sum(duplicated(bike\_share\_rides))

# Remove duplicates

bike\_share\_rides\_unique <- distinct(bike\_share\_rides)

# Count the full duplicates in bike\_share\_rides\_unique

sum(duplicated(bike\_share\_rides\_unique))

-Removing partial Duplicate exercise :

# Find duplicated ride\_ids

bike\_share\_rides %>%

  count(ride\_id) %>%

  filter(n > 1)

# Remove full and partial duplicates

bike\_share\_rides\_unique <- bike\_share\_rides %>%

  # Only based on ride\_id instead of all cols

  distinct(ride\_id, .keep\_all = TRUE)

# Find duplicated ride\_ids in bike\_share\_rides\_unique

bike\_share\_rides\_unique %>%

  # Count the number of occurrences of each ride\_id

  count(ride\_id) %>%

  # Filter for rows with a count > 1

  filter(n>1)

-Aggregating Partial Duplicate Exercise :

bike\_share\_rides %>%

  # Group by ride\_id and date

  group\_by(ride\_id, date) %>%

  # Add duration\_min\_avg column

  mutate(duration\_min\_avg = mean(duration\_min) ) %>%

  # Remove duplicates based on ride\_id and date, keep all cols

  distinct(ride\_id, date, .keep\_all=TRUE) %>%

  # Remove duration\_min column

  select(-duration\_min)

**#CATEGORICAL AND TEXT DATA**

-checking membership :

Using anti join

**\*study\_data %>%**

**\*anti\_join (blood\_types, by = “blood\_type”)**

to filter good data wih join :

**\*study\_data %>%**

**\*semi\_join (blood\_types, by =”blood\_type”)**

-Anti Join :

# Find bad dest\_size rows

sfo\_survey %>%

  # Join with dest\_sizes data frame to get bad dest\_size rows

  anti\_join(dest\_sizes, by="dest\_size") %>%

  # Select id, airline, destination, and dest\_size cols

  select(id, airline, destination, dest\_size)

-Semi Join :

# Remove bad dest\_size rows

sfo\_survey %>%

  # Join with dest\_sizes

  semi\_join(dest\_sizes, by="dest\_size") %>%

  # Count the number of each dest\_size

  count(dest\_size)

-Data Categorical Problems :

Mengubah tulisan menjadi huruf kecil semua

**Library(stringr)**

**Library(dplyr)**

**\*animals**

**\*mutate(type\_lower = str\_to\_lower(type)) %>%**

**\*count(type\_lower)** -> to check

Mengubah menjadi capital semua

**\*animals**

**\*mutate(type\_upper = str\_to\_upper(type)) %>%**

**\*count(type\_upper)** -> to check

-Menghapus spasi di awal dan akhir, tapi tdk bisa di tengah :

**\*library(stringr)**

**\*animals %>%  
\*mutate(type\_trimmed = str\_trim(type\_lower))**

-Count and sort :

**\*animals %>%**

**\*count(type\_trimmed, sort = TRUE)**

-Membuat list menjadi others untuk yg tdak termasuk dalam category tersebut :

**\*other\_categories = c(“amphibian”, “fish”, “bug”, “invertebrate”, “reptile”)**

**\*library(forcats)**

**\*animals %>%  
\*mutate(type\_collapsed=fct\_collapse(type\_trimmed, other = other\_categories)**

-correcting inconsistency exercise :

# Add new columns to sfo\_survey

sfo\_survey <- sfo\_survey %>%

  # dest\_size\_trimmed: dest\_size without whitespace

  mutate(dest\_size\_trimmed = str\_trim(dest\_size),

         # cleanliness\_lower: cleanliness converted to lowercase

         cleanliness\_lower = str\_to\_lower(cleanliness))

# Count values of dest\_size\_trimmed

sfo\_survey %>%

  count(dest\_size\_trimmed)

# Count values of cleanliness\_lower

sfo\_survey %>%

  count(cleanliness\_lower)

-Collapsing Cattegories Exercise :

# Count categories of dest\_region

sfo\_survey %>%

  count(dest\_region)

# Categories to map to Europe

europe\_categories <- c("EU", "eur", "Europ")

# Add a new col dest\_region\_collapsed

sfo\_survey %>%

  # Map all categories in europe\_categories to Europe

  mutate(dest\_region\_collapsed = fct\_collapse(dest\_region,

                                              Europe = europe\_categories)) %>%

  # Count categories of dest\_region\_collapsed

  count(dest\_region\_collapsed)

-Cleaning Text Data

Detecting symbol :

**\*str\_detect(customers$credit\_card, “-“) or you want filter it :**

**\*customers %>%**

**\*filter(str\_detect(credit\_card, “-“))**

-Replace All :

**\*customers %>%  
\*mutate(credit\_card\_spaces = str\_replace\_all(credit\_card, “-“, “ “))** -> replace “-“ with blank

-Remove All :

**\*str\_remove\_all(“-“) : Remove all hyphens**

**\*str\_remove\_all(“ “) : Remove all spaces**

-Count Character :

**\*str\_length(nama\_kolom)**

-finding invalid count character :

**\*customers %>%**

**\*filter(str\_length(nama\_kolom) != 16)**

-Regex :

Use stringr package

Searching the character

**\*str\_detect(nama\_kolom, fixed(“$”))**

-Detecting Inconsistent text data

# Filter for rows with "(" or ")" in the phone column

sfo\_survey %>%

  filter(str\_detect(phone, fixed("(")) | str\_detect(phone, fixed(")")))

-Replacing and Removing Exercise :

# Remove parentheses from phone column

phone\_no\_parens <- sfo\_survey$phone %>%

  # Remove "("s

  str\_remove\_all(fixed("(")) %>%

  # Remove ")"s

  str\_remove\_all(fixed(")"))

# Add phone\_no\_parens as column

sfo\_survey %>%

  mutate(phone\_no\_parens = phone\_no\_parens,

  # Replace all hyphens in phone\_no\_parens with spaces

         phone\_clean = str\_replace\_all(phone\_no\_parens, "-", " "))

-Invalid phone numbers exercise :

# Check out the invalid numbers

sfo\_survey %>%

  filter(str\_length(phone) != 12)

# Remove rows with invalid numbers

sfo\_survey %>%

  filter(str\_length(phone) == 12)

**#ADVANCED DATA PROBLEMS**

-Uniformity : using if else

**\*mutate(temp\_c = ifelse(temp >50, (temp-32)\*5/9, temp))**

-parsing multiple date format :

**\*library(lubridate)**

**\*parse\_date\_time(nyc\_temps$date, orders =c(“%Y-%m-%d”, “%m/%d/%y”, “%B %d, %Y”))**

Format date akan berubah sama semua, jika format date sebelumnya ada di dalam list tersebut. Jika tidak ada dalam list tersebut maka akan menjadi NA.

-Date Uniformity Exercise :

# Check out the accounts data frame

head(accounts)

# Define the date formats

formats <- c("%Y-%m-%d", "%B %d, %Y")

# Convert dates to the same format

accounts %>%

  mutate(date\_opened\_clean = parse\_date\_time(date\_opened, orders= formats))

-Currency uniformity exercise :

# Scatter plot of opening date and total amount

accounts %>%

  ggplot(aes(x = date\_opened, y = total)) +

  geom\_point()

# Left join accounts to account\_offices by id

accounts %>%

  left\_join(account\_offices, by = "id") %>%

  # Convert totals from the Tokyo office to USD

  mutate(total\_usd = ifelse(office == "Tokyo", total / 104, total)) %>%

  # Scatter plot of opening date vs total\_usd

  ggplot(aes(x = date\_opened, y = total\_usd)) +

    geom\_point()

-Cross Field Validation

Validating Numbers

**\*credit\_cards %>%**

**\*mutate(theoretical\_total = dining\_cb +groceries\_cb + gas\_cb) %>%  
\*filter(theoretical\_cb != total\_cb) %>%**

**\* select(dining\_cb:theoretical\_cb)**

Calculating age

**\*library(lubridate)**

**\*date\_difference <- as.Date(“2015-09-04”) %--% today()**

**\*floor(as.numeric(date\_difference, “years”) )** -> merubah ke dalam bentuk tahun

Validaitng Age

**\*credit\_cards %>%**

**\*mutate(theor\_age = floor(as.numeric(date\_opened %--% today(), “years”))) %>%**

**\*filter(theor\_age !=acct\_age)**

-Validating total exercise :

# Find invalid totals

accounts %>%

  # theoretical\_total: sum of the three funds

  mutate(theoretical\_total=fund\_A+fund\_B+fund\_C) %>%

  # Find accounts where total doesn't match theoretical\_total

  filter(total != theoretical\_total)

-Validating Age Exercise :

# Find invalid acct\_age

accounts %>%

  # theoretical\_age: age of acct based on date\_opened

  mutate(theoretical\_age = floor(as.numeric(date\_opened %--% today(), "years"))) %>%

  # Filter for rows where acct\_age is different from theoretical\_age

  filter(acct\_age != theoretical\_age)

-Completeness

Finding missing value

**\*is.na(airquality) : True if missing, False if not missing**

Jumlah missing value

**\*sum(is.na(airquality))**

Visualizing missing value

**\*library(visdate)**

**\*vis\_miss(airquality)**

Investigasi missingness

**\*airquality %>%**

**\*mutate(miss\_ozone = is.na(Ozone) %>%  
\*group\_by(miss\_ozone) %>%  
\*summarize\_all(median, na.rm = TRUE) -> Melihat nilai mediannya dengan data kosong atau tidak, pengaruhnya besar gak**

**\*airquality %>%**

**\*arrange(Temp) %>%**

**\*vis\_miss() -> Melihat secara grafik**

Dropping missing value

**\*airquality %>%**

**\*filter(!is.na(Ozone), !is.na(Solar.R))**

Replacing missing value

**\*airquality %>%**

**\*mutate(ozone\_filled = ifelse(is.na(Ozone), mean(Ozone, na.rm =TRUE), Ozone))**

-Visualizing missing data Exercise

# Visualize the missing values by column

vis\_miss(accounts)

accounts %>%

  # missing\_inv: Is inv\_amount missing?

  mutate(missing\_inv = is.na(inv\_amount)) %>%

  # Group by missing\_inv

  group\_by(missing\_inv) %>%

  # Calculate mean age for each missing\_inv group

  summarize(avg\_age = mean(age))

# Sort by age and visualize missing vals

accounts %>%

  arrange(age) %>%

  vis\_miss()

-Treating Missing Data Exercise

# Create accounts\_clean

accounts\_clean <- accounts %>%

  # Filter to remove rows with missing cust\_id

  filter(!is.na(cust\_id)) %>%

  # Add new col acct\_amount\_filled with replaced NAs

  mutate(acct\_amount\_filled = ifelse(is.na(acct\_amount), inv\_amount \* 5, acct\_amount))

# Assert that cust\_id has no missing vals

assert\_all\_are\_not\_na(accounts\_clean$cust\_id)

# Assert that acct\_amount\_filled has no missing vals

assert\_all\_are\_not\_na(accounts\_clean$acct\_amount\_filled)

**#RECORD LINKAGE**

**-How to link data woth linkage : data A and data B -> Generate Pairs -> Compare Pairs -> Score Pairs -> Link Data**

Type of edit distance

-Damerau-Levensthien

-Levenshtein

-LCS (Longest Common Subsequence)

-Jaro winkler

-Jaccard

- Damerau-Levensthien method :

**\*library(stringdist)**

**\*stringdist(“baboo”, “typhoon”,**

**\*method = “dl”) -> another method : lcs, jaccard**

-Remapping using string distance :

**\*library(fuzzyjoin)**

**\*stringdist\_left\_join(survey, cities, by = “city”, method = “dl”)**

**\*stringdist\_left\_join(survey, cities, by = “city”, method = “dl”, max\_dist =1)**

-Fixing Typos with string distance :

# Count the number of each city variation

zagat %>%

  count(city)

# Join zagat and cities and look at results

zagat %>%

  # Left join based on stringdist using city and city\_actual cols

  stringdist\_left\_join(cities, by = c("city" = "city\_actual")) %>%

  # Select the name, city, and city\_actual cols

  select(name, city, city\_actual)

Linkage doesn’t need exact matces . join need exact matches

Generating pairs in R

**\*library(reclin)**

**\*pair\_blocking(df\_A, df\_B)**

Pair Blocking in R  
**\*pair\_blocking(df\_A, df\_B, blocking\_var = “state”)**

Comparing Pairs

**\*pair\_blocking(df\_A, df\_B, blocking\_var =”state”) %>%  
\*compare\_pairs(by=”name”, default\_comparator = lcs())**

Comparing Multiple Column

**\*pair\_blocking(df\_A, df\_B, blocking\_var =”state”) %>%  
\*compare\_pairs(by= c(”name”, “zip”), default\_comparator = lcs())**

-Pair Blocking

# Load reclin

library(reclin)

# Generate pairs with same city

pair\_blocking(zagat, fodors, blocking\_var="city")

-Comparing Pairs

# Generate pairs

pair\_blocking(zagat, fodors, blocking\_var = "city") %>%

  # Compare pairs by name using lcs()

  compare\_pairs(by = "name" ,

      default\_comparator =lcs())

# Generate pairs

pair\_blocking(zagat, fodors, blocking\_var = "city") %>%

  # Compare pairs by name, phone, addr

  compare\_pairs(by=c("name","phone","addr"),

      default\_comparator=jaro\_winkler())

Scoring and Linking :

**\*pair\_blocking (df\_A, df\_B, blocking\_var =”state”) %>%**

**\*copare\_pairs(by =c(“name”, “zip”) , default\_comparator = lcs()) %>%  
\*score\_simsum()**

Scoring and linking using scoring probabilistically

**\*pair\_blocking (df\_A, df\_B, blocking\_var =”state”) %>%**

**\*copare\_pairs(by =c(“name”, “zip”) , default\_comparator = lcs()) %>%**

**\*score\_problink() %>%**

**\*select\_n\_to\_m() %>% -> memilih nilai terbesar**

**\*link()**

-Putting it together

# Create pairs

pair\_blocking(zagat, fodors, blocking\_var = "city") %>%

  # Compare pairs

  compare\_pairs(by = "name", default\_comparator = jaro\_winkler()) %>%

  # Score pairs

  score\_problink() %>%

  # Select pairs

  select\_n\_to\_m() %>%

  # Link data

  link()