

ETC5512: Extended Consultation

Table of contents

Learning Objective

In today's class we'll be looking at two different AI solutions that tidy the data from assignment 1.

- We'll compare two different AI-based approaches to data cleaning.
- Evaluate the strengths and limitations of each AI solution.
- Decide if we think the code smells good
- Apply critical thinking to decide whether these AI solutions were suitable for task

Prompt

Write code in R to produce a tidy dataset for me. Tidy in the sense of tidy data. Keep the code clean and human readable. I also want code accessible for entry level tidy users new to string manipulation

Alongside the prompt a sample of the data was shared.

AI Solution #1

```
library(tidyverse)

# 1. Read the raw file ----
raw_data <- read_csv(
  "your_file.csv",
  skip = 8,           # skip metadata rows at the top
  na = c("-", "", "NA") # treat these as missing values
)
```

```

# 2. Clean column names ----
data <- raw_data %>%
  janitor::clean_names()

# 3. Separate coordinates into numeric columns ----
data <- data %>%
  separate(
    coordinates,
    into = c("longitude", "latitude"),
    sep = ",",
    remove = TRUE
  ) %>%
  mutate(
    longitude = str_remove_all(longitude, "\\[") %>% as.numeric(),
    latitude = str_remove_all(latitude, "\\]") %>% as.numeric()
  )

# 4. Extract suburb and region from location ----
data <- data %>%
  separate(
    location,
    into = c("road", "details"),
    sep = "\\(",
    extra = "merge",
    fill = "right"
  ) %>%
  separate(
    details,
    into = c("suburb", "region"),
    sep = ",",
  ) %>%
  mutate(
    suburb = str_remove(suburb, "Suburb:"),
    region = str_remove(region, "Region:")
  )

# 5. Convert date-time columns ----
data <- data %>%
  mutate(
    created = lubridate::dmy_hm(created),
    updated = lubridate::dmy_hm(updated),
    ended = lubridate::dmy_hm(ended)
  )

```

```

)

# 6. Select and arrange tidy columns ----
tidy_data <- data %>%
  select(
    id,
    title,
    road,
    suburb,
    region,
    created,
    updated,
    traffic_conditions,
    impact,
    advice,
    other_advice,
    longitude,
    latitude
  )

# 7. Final tidy dataset ----
tidy_data

```

AI Solution #2

```

# =====
# Tidy the NSW LiveTraffic Historical Data Export
# =====
# The raw file has two problems to solve before we can analyse it:
# 1. The first 6 rows are metadata (document title, date range, etc.)
#    and row 7 is blank - the real column headers start on row 8.
# 2. Once loaded, many columns need reshaping:
#    - "-" is used as a placeholder for missing values
#    - Location hides suburb and region inside it
#    - Version is "X of Y" but we really want two numbers
#    - Coordinates is a text string like "[153.49, -28.16]"
#    - Several columns are almost entirely empty and can be dropped
#
# Packages used:
#   readr - fast, message-friendly CSV reading

```

```

# dplyr - column selection, filtering, renaming, mutating
# tidyr - separating compound columns into multiple columns
# stringr - all the string manipulation (str_* functions)
# lubridate - parsing dates and times

library(readr)
library(dplyr)
library(tidyr)
library(stringr)
library(lubridate)

# Step 1: Load the raw file
#
# skip = 7 skips the 6 metadata rows + 1 blank row so the first row
# read is the real header row.
#
# name_repair = "universal" converts awkward column names (spaces,
# parentheses) into safe R-friendly names automatically.

raw <- read_csv(
  "your_file.csv", # <-- replace with your actual file path
  skip = 7,
  name_repair = "universal"
)

# Step 2: Replace "-" placeholders with proper NA
#
# The export uses a bare hyphen "-" wherever there is no value.
# We convert those to NA so R treats them as missing, not as text.
# across(everything()) applies the same transformation to every column.

traffic <- raw |>
  mutate(
    across(everything(), ~ if_else(.x == "-", NA_character_, .x))
  )

# Step 3: Drop columns that are entirely or almost entirely empty
#
# Public.transport, Traffic.arrangement, and Download.maps have zero

```

```

# non-missing values in this export - they add no information.
# Geometry.collection is a raw geometry blob not useful for tabular analysis.
# We use select() with the minus sign to remove them.

traffic <- traffic |>
  select(
    -Public.transport,
    -Traffic.arrangement,
    -Download.maps,
    -Geometry.collection
  )

# Step 4: Parse the Version column
#
# Version looks like "3 of 53" - the current update number and the
# total number of updates for that incident.
#
# separate_wider_regex() splits a single column using a regular
# expression pattern into multiple new columns. The pattern below
# captures two groups of digits separated by " of ".
#
# (?P<version_number>\\d+) - one or more digits → version_number
# \\s+of\\s+                - " of " (with flexible spaces)
# (?P<version_total>\\d+)  - one or more digits → version_total

traffic <- traffic |>
  separate_wider_regex(
    cols      = Version,
    patterns  = c(
      version_number = "\\d+",
      "\\s+of\\s+",
      version_total  = "\\d+"
    )
  ) |>
  mutate(
    version_number = as.integer(version_number),
    version_total  = as.integer(version_total)
  )

# Step 5: Parse the Location column

```

```

#
# Location looks like:
# "M1 Pacific Motorway at X (Suburb:Tweed Heads, Region:REG_NORTH)"
#
# We want three separate columns: road, suburb, region.
#
# Plan:
# a) Extract the suburb label using str_extract() with a regex that
# looks for "Suburb:" followed by characters up to the next comma.
# b) Extract the region label the same way.
# c) Strip the "(Suburb:... Region:...)" suffix off the road name
# using str_remove() and trim trailing whitespace with str_trim().
#
# str_extract() returns the matched text or NA if there is no match.
# The regex "(?<=Suburb:)[^,]+" means:
# (?<=Suburb:) - preceded by "Suburb:" (but don't include it)
# [^,]+ - one or more characters that are not a comma or ")"

traffic <- traffic |>
  mutate(
    suburb = str_extract(Location, "(?<=Suburb:)[^,]+" ),
    region = str_extract(Location, "(?<=Region:)[^,]+" ),
    road   = Location |>
      str_remove("\\s*\\(Suburb:.*\\)\\s*$") |>
      str_trim(),
    .keep = "unused" # removes the original Location column
  )

# Step 6: Parse the Coordinates column
#
# Coordinates looks like "[153.4987, -28.1641]"
# We want two numeric columns: longitude and latitude.
#
# str_remove_all() strips the square brackets [ and ].
# Then separate_wider_delim() splits on ", " into two columns.

traffic <- traffic |>
  mutate(
    Coordinates = str_remove_all(Coordinates, "[\\[\\]]")
  ) |>
  separate_wider_delim(

```

```

    cols = Coordinates,
    delim = ", ",
    names = c("longitude", "latitude")
  ) |>
  mutate(
    longitude = as.double(longitude),
    latitude = as.double(latitude)
  )

# Step 7: Parse date-time columns
#
# Dates look like "Tue 19 Oct 2021 08:33AM".
# lubridate's dmy_hm() parses day-month-year + hour-minute.
# The format string "%a %d %b %Y %I:%M%p" matches that pattern:
# %a abbreviated weekday (Tue)
# %d day of month (19)
# %b abbreviated month name (Oct)
# %Y 4-digit year
# %I hour in 12-hour clock
# %M minutes
# %p AM / PM
#
# quiet = TRUE suppresses the "N failed to parse" message for the
# rows where the value was already NA.

parse_livetraffic_date <- function(x) {
  parse_date_time(x, orders = "%a %d %b %Y %I:%M%p", quiet = TRUE)
}

traffic <- traffic |>
  mutate(
    across(c(Created, Ended, Updated), parse_livetraffic_date)
  )

# Step 8: Tidy up remaining column types and names
#
# Queue_length and Reduced.speed.limit arrived as character columns.
# Convert them to numeric. -1 is used to mean "no value" for speed
# limit, so replace those with NA as well.
#

```

```

# rename() gives us snake_case column names throughout.

traffic <- traffic |>
  mutate(
    Queue_length      = as.integer(Queue_length),
    Reduced.speed.limit = as.integer(Reduced.speed.limit),
    Reduced.speed.limit = na_if(Reduced.speed.limit, -1L)
  ) |>
  rename(
    id          = ID,
    title       = Title,
    created     = Created,
    ended       = Ended,
    updated     = Updated,
    date_planned = Date..Planned.Incidents.,
    traffic_conditions = Traffic.conditions,
    queue_length_m = Queue_length,
    impact       = Impact,
    attending    = Attending,
    advice       = Advice,
    other_advice = Other_advice,
    diversions   = Diversions,
    website_links = Website.links,
    speed_limit_kmh = Reduced.speed.limit,
    expected_delay = Expected.delay,
    org_name     = Organisation.name,
    org_email    = Organisation.email,
    org_contact  = Organisation.contact,
    org_website  = Organisation.website
  )

# Step 9: Preview and save

glimpse(traffic)

write_csv(traffic, "livetraffic_tidy.csv")

```