Strings & Factors

PSY 410: Data Science for Psychology

Dr. Sara Weston

2026-05-06

Setup

Strings: The basics

How many genders are in this dataset?

id	gender
1	Male
2	male
3	M
4	MALE
5	Female
6	female
7	F

How many genders are in this dataset?

R says **7**. You meant **2**.

The problem is inconsistent text — different cases, abbreviations, trailing spaces. Today we learn to fix that.

What are strings?

Strings are text data — anything in quotes:

```
participant_name <- "Jane Doe"
diagnosis <- "Major Depressive Disorder"
feedback <- "The task was confusing"
```

The `stringr` package (part of tidyverse) gives you tools for working with strings. All `stringr` functions start with `str_`.

Creating and combining strings

```
first <- "Jane"
last <- "Doe"

# Combine strings
str_c(first, last)           # No space
```

```
[1] "JaneDoe"
```

```
str_c(first, last, sep = " ")  # With space
```

```
[1] "Jane Doe"
```

. . .

```
# Combine with other text
str_c("Participant: ", first, " ", last)
```

```
[1] "Participant: Jane Doe"
```

```
# Glue is even easier (from the glue package)
library(glue)
glue("Participant: {first} {last}")
```

```
Participant: Jane Doe
```

String length

```
responses <- c("Yes", "No", "Maybe", "I don't know")

str_length(responses)
```

```
[1]  3  2  5 12
```

. . .

Useful for checking free-response data quality:

```
# Flag very short responses
tibble(responses) |>
  mutate(too_short = str_length(responses) < 3)
```

```
# A tibble: 4 x 2
  responses    too_short
  <chr>        <lgl>
1 Yes          FALSE
2 No           TRUE
3 Maybe        FALSE
4 I don't know FALSE
```

Changing case

```
messy_data <- c("MALE", "Female", "male", "FEMALE", "Male")

str_to_lower(messy_data)
```

```
[1] "male"   "female" "male"   "female" "male"
```

```
str_to_upper(messy_data)
```

```
[1] "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"
```

```
str_to_title(messy_data)
```

```
[1] "Male"   "Female" "Male"   "Female" "Male"
```

. . .

Essential for cleaning demographic data!

Trimming whitespace

Survey data often has extra spaces:

```
messy_responses <- c("  Yes", "No  ", "  Maybe  ")

str_trim(messy_responses)         # Remove leading/trailing
```

```
[1] "Yes"   "No"    "Maybe"
```

```
str_squish(messy_responses)       # Also reduce internal spaces
```

```
[1] "Yes"   "No"    "Maybe"
```

Psychology example: Cleaning demographics

```
survey <- tibble(
  gender = c("  Female", "MALE  ", "female", "Male", "  non-binary")
)

survey |>
  mutate(
    gender_clean = str_to_lower(str_trim(gender))
  )
```

```
# A tibble: 5 x 2
  gender         gender_clean
  <chr>          <chr>
1 "  Female"     female
```

```
2 "MALE  "       male
3 "female"       female
4 "Male"         male
5 "  non-binary" non-binary
```

Detecting patterns

`str_detect()` checks if a pattern is present:

```
feedback <- c(
  "The task was clear",
  "I found it confusing",
  "Very clear instructions",
  "Somewhat confusing"
)

str_detect(feedback, "clear")
```

```
[1]  TRUE FALSE  TRUE FALSE
```

. . .

Use in `filter()`:

```
tibble(feedback) |>
  filter(str_detect(feedback, "clear"))
```

```
# A tibble: 2 x 1
  feedback
  <chr>
1 The task was clear
2 Very clear instructions
```

Case-insensitive detection

Patterns are case-sensitive by default:

```
str_detect("Clear instructions", "clear")  # FALSE
```

```
[1] FALSE
```

. . .

Make them case-insensitive with `regex(ignore_case = TRUE)`:

```
str_detect("Clear instructions", regex("clear", ignore_case = TRUE))
```

```
[1] TRUE
```

Replacing text

```
responses <- c("Strongly Disagree", "Disagree", "Strongly Agree", "Agree")

# Replace first match
str_replace(responses, "Strongly ", "Very ")
```

```
[1] "Very Disagree" "Disagree"      "Very Agree"    "Agree"
```

```
# Replace all matches
str_replace_all(responses, "Strongly ", "Very ")
```

```
[1] "Very Disagree" "Disagree"      "Very Agree"    "Agree"
```

Multiple replacements

```
diagnosis <- c("MDD", "GAD", "MDD", "OCD", "GAD")

# Replace multiple patterns at once
str_replace_all(diagnosis, c(
  "MDD" = "Major Depressive Disorder",
  "GAD" = "Generalized Anxiety Disorder",
  "OCD" = "Obsessive-Compulsive Disorder"
))
```

```
[1] "Major Depressive Disorder"     "Generalized Anxiety Disorder"
[3] "Major Depressive Disorder"     "Obsessive-Compulsive Disorder"
[5] "Generalized Anxiety Disorder"
```

Extracting parts of strings

```
participant_ids <- c("PSY001", "PSY002", "PSY010", "PSY123")

# Extract substring by position
str_sub(participant_ids, start = 4)  # Get everything after position 3
```

```
[1] "001" "002" "010" "123"
```

. . .

```
# Extract numbers
str_extract(participant_ids, "\\d+")  # \d+ means "one or more digits"
```

```
[1] "001" "002" "010" "123"
```

When you need more: Regular expressions

Regular expressions (regex) are powerful pattern matching tools.

. . .

Examples:

- `\\d` matches digits
- `\\s` matches whitespace
- `.` matches any character
- `+` means "one or more"
- `*` means "zero or more"

. . .

> **Note**
>
> Regex is powerful but complex. For this course, stick to simple patterns. When you need more, check R4DS Ch 14 or regex101.com.

Pair coding break

Your turn: Clean text data

You have messy survey responses:

```
messy_survey <- tibble(
  id = 1:5,
  gender = c("  FEMALE", "male  ", "Female", "MALE", "non-binary  "),
  comment = c(
    "Great study!",
    "too long",
    "  Very interesting  ",
    "CONFUSING INSTRUCTIONS",
    "I enjoyed this"
  )
)
```

1. Clean `gender` to lowercase with no extra spaces
2. Clean `comment` to title case with no extra spaces
3. Create a logical column `is_negative` that is TRUE if the comment contains "long" or "confusing" (case-insensitive)
4. Filter to only negative comments

Time: 10 minutes

Factors

What are factors?

Factors are R's way of representing categorical data with a fixed set of possible values.

. . .

```
# A character vector
gender_char <- c("Male", "Female", "Female", "Male")

# A factor
gender_fct <- factor(gender_char)
gender_fct
```

```
[1] Male   Female Female Male
Levels: Female Male
```

. . .

Notice the **Levels** line — those are the possible categories.

Why use factors?

1. **Memory efficient** — R stores categories once, not repeatedly
2. **Prevent typos** — Can't accidentally add invalid categories
3. **Control order** — Specify the order for plots and tables
4. **Model requirements** — Many statistical models require factors

Creating factors

```r
condition <- c("Control", "Treatment", "Treatment", "Control")

# Let R choose levels (alphabetical)
factor(condition)
```

```
[1] Control   Treatment Treatment Control
Levels: Control Treatment
```

. . .

```r
# Specify levels explicitly
factor(condition, levels = c("Control", "Treatment"))
```

```
[1] Control   Treatment Treatment Control
Levels: Control Treatment
```

Order matters!

```
likert <- c("Agree", "Disagree", "Strongly Agree", "Agree", "Strongly Disagree")

# Alphabetical order (default)
factor(likert)
```

```
[1] Agree             Disagree          Strongly Agree    Agree
[5] Strongly Disagree
Levels: Agree Disagree Strongly Agree Strongly Disagree
```

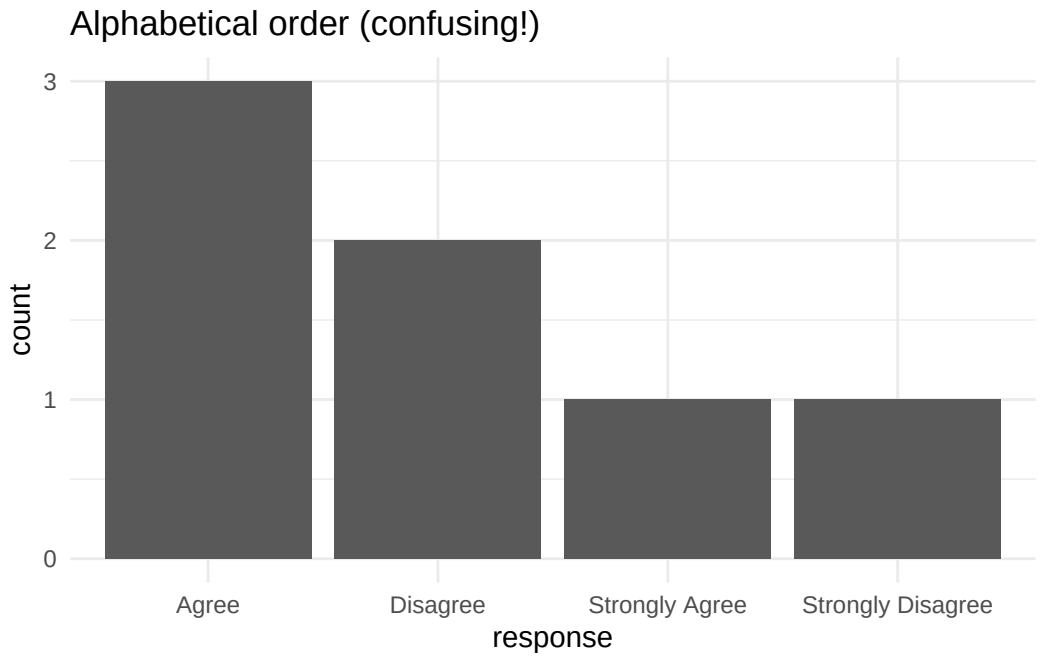
. . .

```
# Logical order
factor(likert, levels = c(
  "Strongly Disagree", "Disagree", "Agree", "Strongly Agree"
))
```

```
[1] Agree             Disagree          Strongly Agree    Agree
[5] Strongly Disagree
Levels: Strongly Disagree Disagree Agree Strongly Agree
```

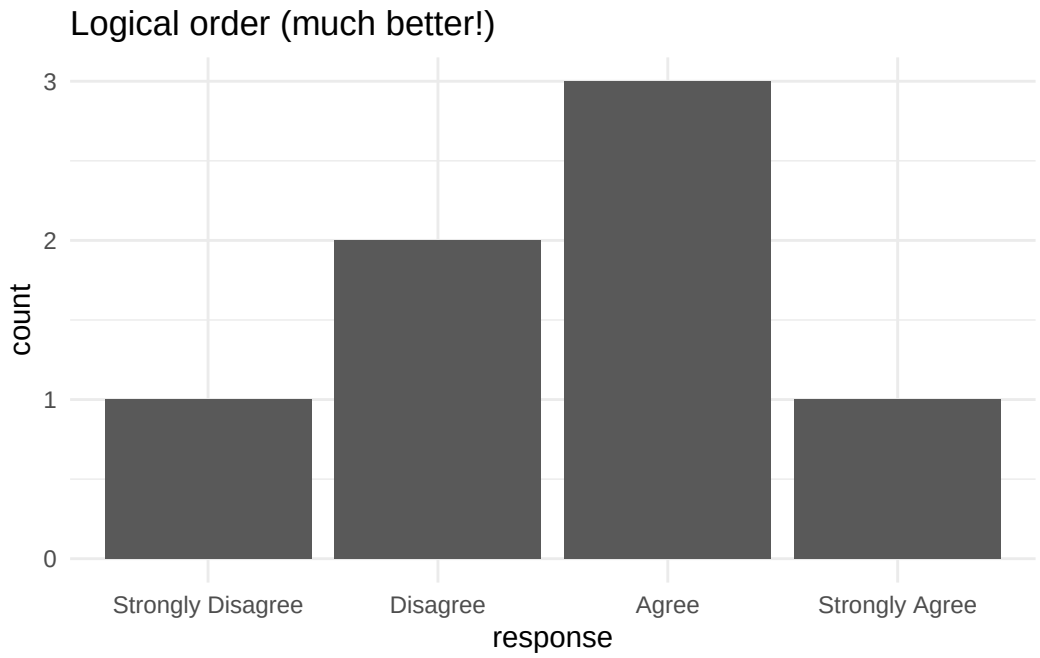
Why order matters: Example

```
survey_data <- tibble(
  response = c("Agree", "Disagree", "Strongly Agree", "Agree",
               "Strongly Disagree", "Agree", "Disagree")
)

# Without factor ordering
ggplot(survey_data, aes(x = response)) +
  geom_bar() +
  labs(title = "Alphabetical order (confusing!)") +
  theme_minimal()
```



Fixed with factor ordering

```
survey_data <- survey_data |>
  mutate(
    response = factor(response, levels = c(
      "Strongly Disagree", "Disagree", "Agree", "Strongly Agree"
    ))
  )

ggplot(survey_data, aes(x = response)) +
  geom_bar() +
  labs(title = "Logical order (much better!)") +
  theme_minimal()
```



Forcats: Factor tools

The `forcats` package (part of tidyverse) provides functions for working with factors.

All `forcats` functions start with `fct_`.

Key functions:

- `fct_relevel()` — manually reorder levels
- `fct_reorder()` — reorder by another variable
- `fct_infreq()` — order by frequency
- `fct_recode()` — rename levels
- `fct_collapse()` — combine levels

fct_relevel(): Manual reordering

```
education <- factor(c("High School", "Bachelor's", "Master's", "High School"))
education
```

```
[1] High School Bachelor's  Master's    High School
Levels: Bachelor's High School Master's
```

. . .

```
# Put in logical order
fct_relevel(education, "High School", "Bachelor's", "Master's")
```

```
[1] High School Bachelor's  Master's    High School
Levels: High School Bachelor's Master's
```

fct_reorder(): Order by another variable

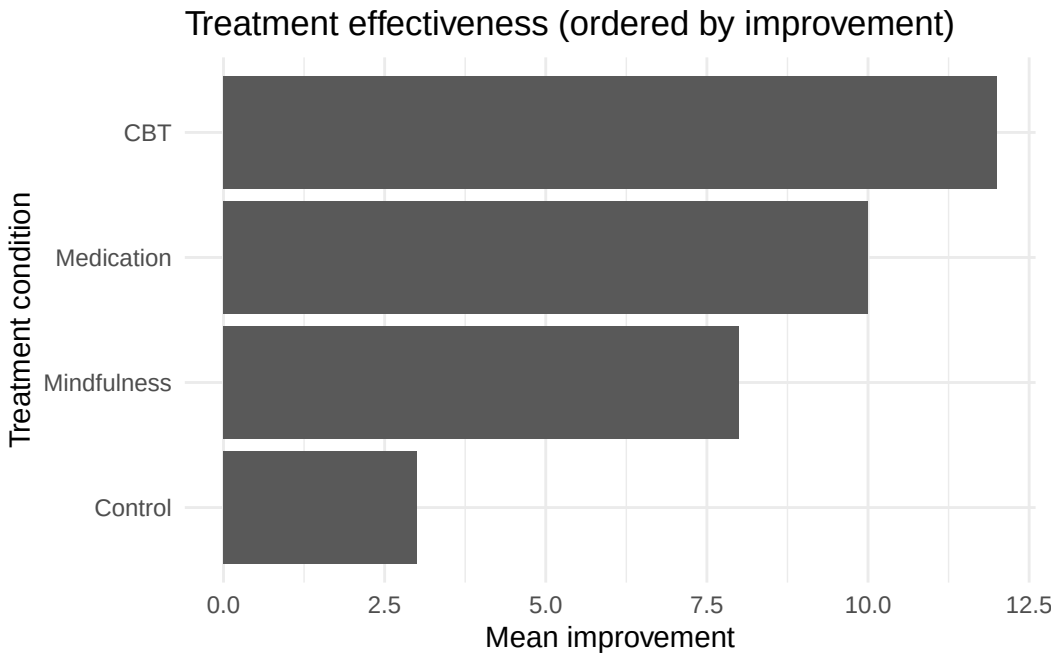
Extremely useful for plots!

```
therapy <- tibble(
  condition = c("CBT", "Control", "Mindfulness", "Medication"),
  mean_improvement = c(12, 3, 8, 10)
)

therapy |>
  mutate(condition = fct_reorder(condition, mean_improvement))
```

```
# A tibble: 4 x 2
  condition   mean_improvement
  <fct>                  <dbl>
1 CBT                       12
2 Control                    3
3 Mindfulness                8
4 Medication                10
```

fct_reorder() in action

```
therapy |>
  mutate(condition = fct_reorder(condition, mean_improvement)) |>
  ggplot(aes(x = mean_improvement, y = condition)) +
  geom_col() +
  labs(
    title = "Treatment effectiveness (ordered by improvement)",
    x = "Mean improvement",
    y = "Treatment condition"
  ) +
  theme_minimal()
```



fct_infreq(): Order by frequency

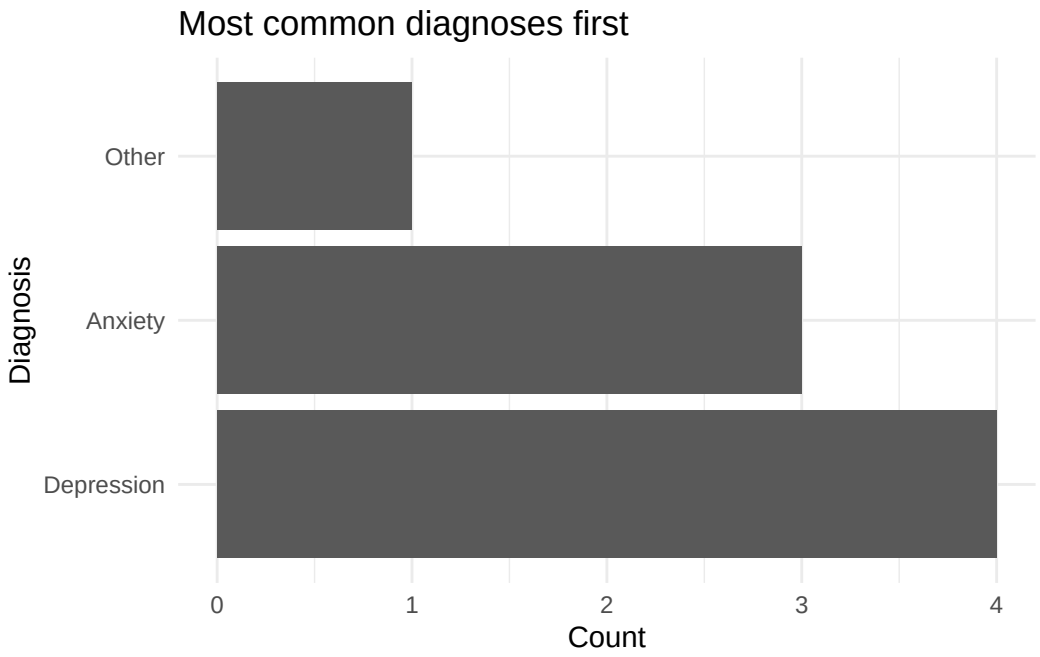
```
diagnosis <- c("Depression", "Anxiety", "Depression", "Other",
               "Depression", "Anxiety", "Anxiety", "Depression")

# Order from most to least common
fct_infreq(factor(diagnosis))
```

```
[1] Depression Anxiety    Depression Other      Depression Anxiety    Anxiety
[8] Depression
Levels: Depression Anxiety Other
```

fct_infreq() in plots

```
tibble(diagnosis = factor(diagnosis)) |>
  mutate(diagnosis = fct_infreq(diagnosis)) |>
  ggplot(aes(y = diagnosis)) +  # Note: y instead of x to read easily
  geom_bar() +
  labs(
```

```
    title = "Most common diagnoses first",
    x = "Count",
    y = "Diagnosis"
  ) +
  theme_minimal()
```



fct_recode(): Rename levels

```
gender <- factor(c("M", "F", "F", "M", "NB"))

fct_recode(gender,
  "Male" = "M",
  "Female" = "F",
  "Non-binary" = "NB"
)
```

```
[1] Male       Female     Female     Male       Non-binary
Levels: Female Male Non-binary
```

fct_collapse(): Combine levels

Useful for grouping rare categories:

```
diagnosis <- factor(c("MDD", "GAD", "OCD", "PTSD", "Panic Disorder",
                      "MDD", "GAD", "Social Anxiety"))

fct_collapse(diagnosis,
  Depression = "MDD",
  Anxiety = c("GAD", "OCD", "PTSD", "Panic Disorder", "Social Anxiety")
)
```

```
[1] Depression Anxiety    Anxiety    Anxiety    Anxiety    Depression Anxiety
[8] Anxiety
Levels: Anxiety Depression
```

Psychology example: Recoding demographics

```
demo_data <- tibble(
  age_group = c("18-25", "26-35", "18-25", "36-45", "26-35", "46+"),
  education = c("HS", "BA", "BA", "MA", "HS", "PhD")
)

demo_data |>
  mutate(
    # Order age groups logically
    age_group = factor(age_group, levels = c("18-25", "26-35", "36-45", "46+")),

    # Expand education codes
    education = fct_recode(factor(education),
      "High School" = "HS",
      "Bachelor's" = "BA",
      "Master's" = "MA",
      "Doctorate" = "PhD"
    )
  )
```

```
# A tibble: 6 x 2
  age_group education
  <fct>     <fct>
```

```
1 18-25     High School
2 26-35     Bachelor's
3 18-25     Bachelor's
4 36-45     Master's
5 26-35     High School
6 46+       Doctorate
```

Dropping unused levels

After filtering, factors keep old levels:

```
all_diagnoses <- factor(c("Depression", "Anxiety", "Other"))
just_depression <- all_diagnoses[all_diagnoses == "Depression"]
just_depression
```

```
[1] Depression
Levels: Anxiety Depression Other
```

. . .

Use `fct_drop()` to remove them:

```
fct_drop(just_depression)
```

```
[1] Depression
Levels: Depression
```

Common factor issues

Problem 1: Factors created from numbers

```
age_factor <- factor(c(25, 30, 25, 40))
mean(age_factor)  # Error! It's not numeric anymore
```

```
[1] NA
```

. . .

Solution: Convert back to numeric carefully:

```r
as.numeric(as.character(age_factor))  # Correct way
```

```
[1] 25 30 25 40
```

Common factor issues

Problem 2: Factors behave differently than strings

```r
colors <- factor(c("red", "blue"))

# Can't just add new values
colors[3] <- "green"  # This creates NA!
colors
```

```
[1] red  blue <NA>
Levels: blue red
```

. . .

Solution: Convert to character first, or use `fct_expand()` to add levels.

End-of-deck exercise

Practice: Clean and visualize survey data

You have messy Likert scale data:

```r
likert_data <- tibble(
  question = rep(c("Q1", "Q2", "Q3"), each = 10),
  response = sample(c("strongly agree", "Agree", "NEUTRAL",
                      "disagree", "Strongly Disagree"), 30, replace = TRUE)
)
```

1. Clean the `response` variable to title case
2. Convert `response` to a factor with logical ordering
3. Create a bar chart showing response counts for each question
4. Use `facet_wrap()` to make separate panels for each question
5. Bonus: Use `fct_infreq()` to order responses by overall frequency

Wrapping up

String functions cheat sheet

Function	Purpose
`str_c()`	Combine strings
`str_length()`	Get string length
`str_to_lower()`, `str_to_upper()`	Change case
`str_trim()`, `str_squish()`	Remove whitespace
`str_detect()`	Find pattern
`str_replace()`, `str_replace_all()`	Replace pattern
`str_sub()`	Extract substring

Factor functions cheat sheet

Function	Purpose
`factor()`	Create a factor
`fct_relevel()`	Manually reorder levels
`fct_reorder()`	Order by another variable
`fct_infreq()`	Order by frequency
`fct_recode()`	Rename levels
`fct_collapse()`	Combine levels
`fct_drop()`	Remove unused levels

Key takeaways

1. **Strings** are text — use `stringr::str_*()` functions to manipulate
2. **Always clean string data** — case, whitespace, typos
3. **Factors** are categorical data with fixed levels
4. **Factor order matters** for plots and tables
5. **Use forcats** (`fct_*()`) to manipulate factors
6. **Order factors logically** — not alphabetically
7. When in doubt, **check the data type** with `class()` or `glimpse()`

Before next class

Read:

- R4DS Ch 19: Joins

Do:

- Submit Assignment 6
- Check your final project data for string/factor issues
- Practice cleaning demographic variables

The one thing to remember

Messy categories turn into messy results. `str_to_lower()` and `factor()` are your first line of defense.

See you Monday for joins!