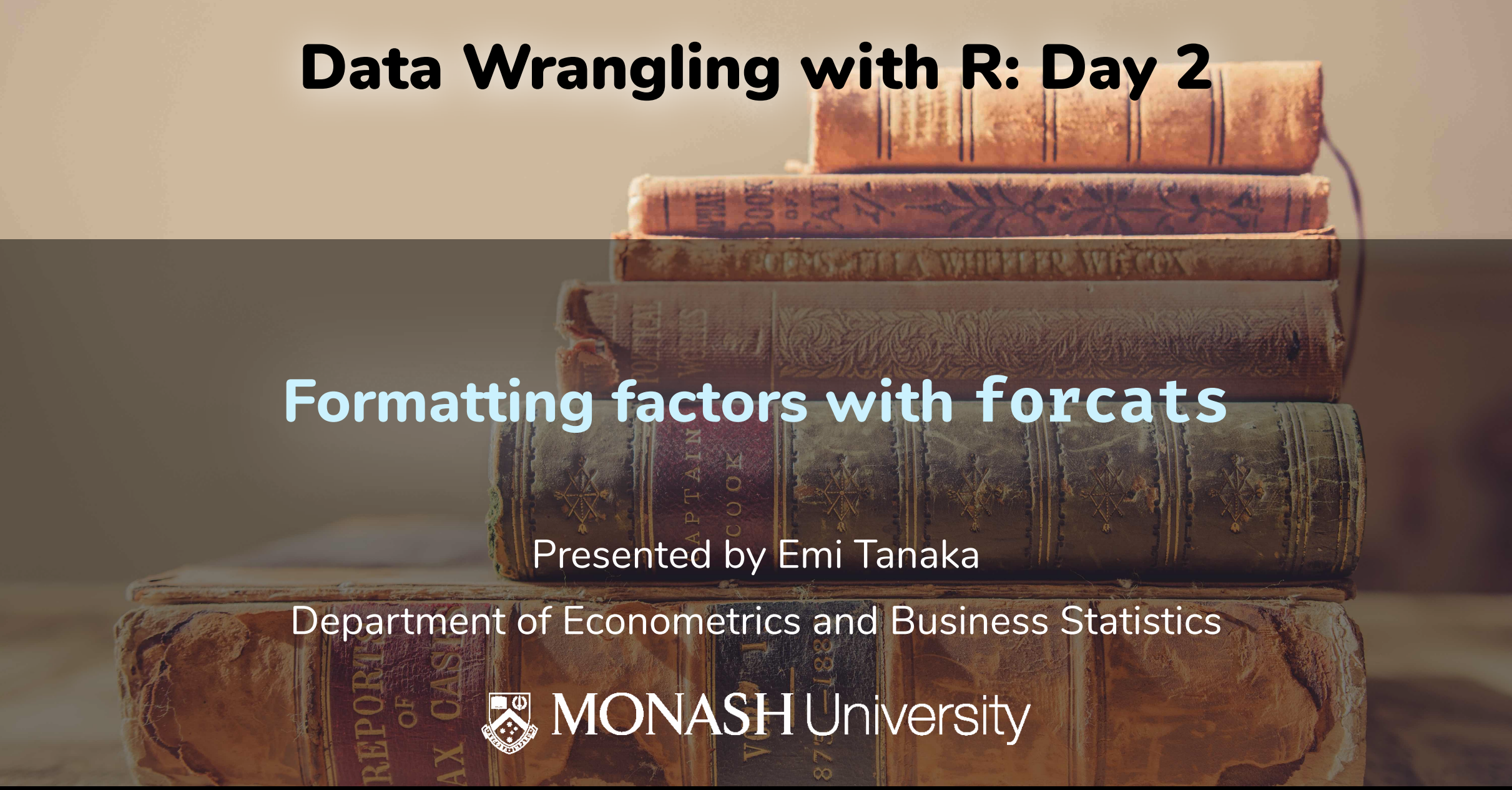
Data Wrangling with R: Day 2



Formatting factors with `forcats`

Presented by Emi Tanaka

Department of Econometrics and Business Statistics

MONASH University

2nd December 2020 @ Statistical Society of Australia | Zoom

There are two types of categorical variables

Nominal where there is no intrinsic ordering to the categories

E.g. blue, grey, black, white.

Ordinal where there is a clear order to the categories

E.g. Strongly disagree, disagree, neutral, agree, strongly agree.

Categorical variables in R Part 1

- In R, categorical variables may be encoded in various ways.

```
cat_chr <- c("red", "white", "blue")
cat_fct <- factor(c("red", "white", "blue"))

class(cat_chr)

## [1] "character"

class(cat_fct)

## [1] "factor"
```

- Then you have categorical variables that look like a numerical variable (e.g. coded variables like say 1=male, 2=female)
- And also those that have fixed levels of numerical values (e.g. `ToothGrowth$dose`: 0.5, 1.0 and 2.0)

So why encode as `factor` instead of `character`?

In some cases, characters are converted to factors (or vice-versa) in functions so they can be similar.

The main idea of a factor is that the variable has a *fixed number of levels*

Categorical variables in R Part 2

- When a variable is encoded as a **factor** then there is an attribute with the levels

```
data <- c(2, 2, 1, 1, 3, 3, 3, 1)
factor(data)

## [1] 2 2 1 1 3 3 3 1
## Levels: 1 2 3
```

- You can easily change the labels of the variables:

```
factor(data,
       labels = c("I", "II", "III"))

## [1] II  II  I   I   III III III I
## Levels: I II III
```

Categorical variables in R Part 3

- Order of the factors are determined by the input:

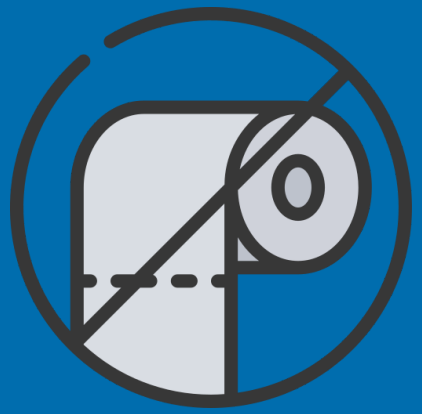
```
# numerical input are ordered in increasing order
factor(c(1, 3, 10))

## [1] 1  3  10
## Levels: 1 3 10

# character input are ordered alphabetically
factor(c("1", "3", "10"))

## [1] 1  3  10
## Levels: 1 10 3

# you can specify order of levels explicitly
factor(c("1", "3", "10"),  levels = c("1", "3", "10"))

## [1] 1  3  10
```

Why would the order of the levels matter?

- Some downstream analysis may use it

```
data("population", package = "tidyr")
population %>%
  filter(year == 2013) %>%
  # just choose 5 countries
  slice(c(1, 11, 21, 31, 41)) %>%
  ggplot(aes(population, country)) +
  geom_col()
```

```
population %>%
  filter(year == 2013) %>%
  slice(c(1, 11, 21, 31, 41)) %>%
  mutate(country =
      reorder(country, population)) %>%
  ggplot(aes(population, country)) +
  geom_col()
```



Cautionary tales of working with factors

Numerical factors in R

```
x <- factor(c(10, 20, 30, 10, 20))
mean(x)

## Warning in mean.default(x): argument is not numeric
## or logical: returning NA

## [1] NA
```

⚠ `as.numeric` function returns the internal integer values of the factor

```
mean(as.numeric(x))

## [1] 1.8
```

You probably want to use:

```
mean(as.numeric(levels(x)[x]))

## [1] 18
```

or

```
mean(as.numeric(as.character(x)))

## [1] 18
```

Defining levels explicitly Part 1

- If the variable contain values that are not in the levels of the factors, then those values will become a missing value

```
factor(c("Yes", "No", "Maybe"), levels = c("Yes", "No"))
```

```
## [1] Yes  No   <NA>
## Levels: Yes No
```

- This can be useful at times, but it's a good idea to check the values before it is transformed as NA

```
factor(c("Yes", "No", "No", "Yess"), levels = c("Yes", "No"))
```

```
## [1] Yes  No   No   <NA>
## Levels: Yes No
```

Defining levels explicitly Part 2

- You can have levels that are not observed

```
f <- factor(c("Yes", "Yes", "Yes", "No"), levels = c("Yes", "Maybe", "No"))
f

## [1] Yes Yes Yes No
## Levels: Yes Maybe No
```

- This can be useful at times downstream, e.g.

```
table(f)

## f
##   Yes Maybe    No
##     3     0     1
```

Combining factors as vectors

```
f1 <- factor(c("F", "M", "F"))
f2 <- factor(c("F", "F"))
```

- What do you think the output will be for below?

```
c(f1, f2)

## [1] 1 2 1 1 1
```

- Was that expected?
- The c function strips the class when you combine factors

```
unclass(f1)

## [1] 1 2 1
## attr(,"levels")
## [1] "F" "M"
```

Combining factors in a data frame

```
df1 <- data.frame(f = factor(c("a", "b")))
df2 <- data.frame(f = factor(c("c", "b")))
```

- What do you think the output below will be?

```
rbind(df1, df2)
```

```
##   f
## 1 a
## 2 b
## 3 c
## 4 b
```

```
rbind(df1, df2)$f
```

```
## [1] a b c b
## Levels: a b c
```

Working with factors with `forcats`

Formatting factors

- The `forcats` package is part of `tidyverse`
- Like the `stringr` package the main functions in `forcats` **prefix with `fct_` or `lvls_`** and the **first argument is a factor (or a character) vector**

  some functions do not allow character as input, e.g. `fct_c`

- The list of available commands are:

  - `fct_anon`
  - `fct_c`
  - `fct_collapse`
  - `fct_count`
  - `fct_cross`
  - `fct_drop`
  - `fct_expand`
  - `fct_explicit_na`
  - `fct_infreq`
  - `fct_inorder`
  - `fct_inseq`
  - `fct_lump`
  - `fct_lump_lowfreq`
  - `fct_lump_min`
  - `fct_lump_n`
  - `fct_lump_prop`
  - `fct_match`
  - `fct_other`
  - `fct_recode`
  - `fct_relabel`
  - `fct_relevel`
  - `fct_reorder`
  - `fct_reorder2`
  - `fct_rev`
  - `fct_shift`
  - `fct_shuffle`
  - `fct_unify`
  - `fct_unique`
  - `lvls_expand`
  - `lvls_reorder`
  - `lvls_revalue`
  - `lvls_union`

Combining factors as vectors with forcats

```
f1 <- factor(c("F", "M", "F"))
f2 <- factor(c("F", "F"))

c(f1, f2)

## [1] 1 2 1 1 1

fct_c(f1, f2)

## [1] F M F F F
## Levels: F M

c1 <- c("F", "M", "F")

fct_c(c1, f2)

## Error: All elements of `...` must be factors
```

Count levels in a factor

```
data("gss_cat", package = "forcats")
table(gss_cat$race)

##
##         Other         Black         White
##          1959          3129         16395
## Not applicable
##               0
```

- `table` in Base R is useful but you may want the output as a data frame

```
fct_count(gss_cat$race, sort = TRUE, prop = TRUE)

## # A tibble: 4 x 3
##   f                 n      p
##   <fct>         <int>  <dbl>
## 1 White         16395 0.763
```

Collapse levels in a factor

```
levels(gss_cat$marital)

## [1] "No answer"     "Never married" "Separated"
## [4] "Divorced"      "Widowed"       "Married"

gss_cat$marital %>%
  fct_collapse(Single = c("Never married", "Separated", "Divorced")) %>%
  fct_relevel("No answer", after = Inf) %>% # move to last place
  fct_count()

## # A tibble: 4 x 2
##   f              n
##   <fct>      <int>
## 1 Single      9542
## 2 Widowed     1807
## 3 Married    10117
## 4 No answer     17
```

Lumping factor levels Part 1

- Sometimes you have a lot of levels and you'd prefer to lump some of them together to the "Other" category
- What criterion do you use to lump levels together?
- There are four main criterion to lump levels using `fct_lump*` functions:
  - `fct_lump_n`: lump all levels except the `n` most frequent
  - `fct_lump_min`: lump together those less than `min` counts
  - `fct_lump_prop`: lump together those less than proportion of `prop`
  - `fct_lump_lowfreq`: lump up least frequent levels such that the Other level is still the smallest level
  - `fct_lump` lifecycle superseded, it is better to use one of the above functions instead

Lumping factor levels Part 2

```
levels(gss_cat$relig)
```

```
##  [1] "No answer"
##  [2] "Don't know"
##  [3] "Inter-nondenominational"
##  [4] "Native american"
##  [5] "Christian"
##  [6] "Orthodox-christian"
##  [7] "Moslem/islam"
##  [8] "Other eastern"
##  [9] "Hinduism"
## [10] "Buddhism"
## [11] "Other"
## [12] "None"
## [13] "Jewish"
## [14] "Catholic"
```

```
fct_lump_n(gss_cat$relig, n = 2) %>%
  fct_count(sort = TRUE, prop = TRUE)
```

```
## # A tibble: 3 x 3
##   f              n     p
##   <fct>      <int> <dbl>
## 1 Protestant 10846 0.505
## 2 Other       5513 0.257
## 3 Catholic    5124 0.239
```

```
fct_lump_lowfreq(gss_cat$relig) %>%
  fct_count(sort = TRUE, prop = TRUE)
```

```
## # A tibble: 2 x 3
##   f              n     p
##   <fct>      <int> <dbl>
## 1 Protestant 10846 0.505
```

</> If you installed the `dwexercise` package, run below in your R console

```
learnr::run_tutorial("day2-exercise-02", package = "dwexercise")
```

🔗 If the above doesn't work for you, go here.

? Questions or issues, let us know!

15:00

Session Information

```
devtools::session_info()
```

```
## ─ Session info ─────────────────────────────────────
##  setting  value
##  version  R version 4.0.1 (2020-06-06)
##  os       macOS Catalina 10.15.7
##  system   x86_64, darwin17.0
##  ui       RStudio
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2020-12-01
##
## ─ Packages ─────────────────────────────────────────
##  package     * version    date       lib
##  anicon        0.1.0      2020-06-21 [1]
##  assertthat    0.2.1      2019-03-21 [2]
```

These slides are licensed under