2025 CCADD Winter Internship - Book Reading Session

Chapter 16: Factors

- https://r4ds.hadley.nz/factors

0. Introduction

What is factor

Categorical data, often stored as "factors" in R, represents variables with a fixed set of possible values.

- Factors are **R's data structure** for **categorical** data.
- They have **defined** set of levels, which helps enforce data integrity and allows for efficient storage
  - Example: days of the week, survey responses
- They may be ordered or unordered

Why is it important?

- By restricting variables to predefined levels
  - **Data integrity**: Prevents invalid categories.
- By leveraging the levels' structure,
  - **Accurate analysis**: Enables proper statistical modeling and summarization.
  - **Effective visualization**: Controls the order and grouping in plots.

Objectives:

1. **Create** `factor` variables
2. Explore the **General Social Survey** dataset via `forcats::gss_cat`
3. **Reorder** factor levels
4. **Modify** factor levels

In [1]:
```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.
0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts
() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all c
onflicts to become errors
```

1. Create factors

`factor()`: create factor from a character or numeric vector

If the set is fixed and we **already know the possible values for it**, which are not in alphabetical order, it is useful to display it as a factor.

In [2]:
```
### Example: Month

# Create a list of the valid levels
month_levels <- c(
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
```

In [3]:
```
x1 <- c("Dec", "Apr", "Jan", "Mar")
x2 <- c("Dec", "Apr", "Jam", "Mar") # Issue #1: typo problem
```

In [4]:
```
print(class(x1))
print(x1)
sort(x1) # Issue #2: Alphabetic order does not match with month level
```

```
[1] "character"
[1] "Dec" "Apr" "Jan" "Mar"
```

'Apr' · 'Dec' · 'Jan' · 'Mar'

Using factors can fix the issues with strings

In [5]:
```r
# Create a factor >> follow specific order
y1 <- factor(x1, levels = month_levels)
print(class(y1))
print(y1) #Before sort: same with chr
sort(y1)
```

```
[1] "factor"
[1] Dec Apr Jan Mar
Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

Jan · Mar · Apr · Dec

▶ **Levels**:

In [6]:
```r
y2 <- factor(x2, levels = month_levels)
print(class(y2))
#**WARNING**: Value not in the level (typo) is automatically converted to NA
y2
```

```
[1] "factor"
```

Dec · Apr · <NA> · Mar

▶ **Levels**:

In [7]:
```r
print(factor(x1)) #not defining specific level -> alphabetic order
```

```
[1] Dec Apr Jan Mar
Levels: Apr Dec Jan Mar
```

In [8]:
```r
levels(y1)
```

'Jan' · 'Feb' · 'Mar' · 'Apr' · 'May' · 'Jun' · 'Jul' · 'Aug' · 'Sep' · 'Oct' · 'Nov' · 'Dec'

`forcats::fct()`: similar with `factor()` but...

Error in `fct()`:

- All values of `x` must appear in `levels` or `na`
- Missing level: "Jam"

In [9]:
```r
y2 <- fct(x2, levels = month_levels)
```

```
Error in `fct()`:
! All values of `x` must appear in `levels` or `na`
i Missing level: "Jam"
Traceback:

1. cli::cli_abort(c("All values of {.arg x} must appear in {.arg levels} or
{.arg na}",
 .     i = "Missing level{?s}: {.str {invalid}}"))
2. rlang::abort(message, ..., call = call, use_cli_format = TRUE,
 .     .frame = .frame)
3. signal_abort(cnd, .file)
4. signalCondition(cnd)
```

`col_factor()`: Define a column as factor when you read a file

- df <- read_csv(csv, col_types = cols(month = `col_factor(month_levels)`))

In [10]:
```
csv <- "
month,value
Jan,12
Feb,56
Mar,12
"
```

In [11]:
```
df <- read_csv(csv, col_types = cols(month = col_factor(month_levels)))
df$month
```

Jan · Feb · Mar

▶ **Levels**:

[Tip 💡] If you want to order the levels to match the order of the first appearance in the data:

1. `unique()`
2. `fct_inorder()`

In [12]:
```
x1 <- c('Dec','Apr','Jan','Mar',"Dec")
f1 <- factor(x1, levels = unique(x1))
f1
```

Dec · Apr · Jan · Mar · Dec

▶ **Levels**:

In [13]:
```
f2 <- x1 |> factor() |>
fct_inorder()
sort(f2)
```

Dec · Dec · Apr · Jan · Mar

▶ **Levels**:

---

2. General Social Survey (`gss_cat`) Dataset

Dataset introduction

- A tibble with 21483 obervations & 9 columns

In [14]:
```
head(gss_cat) #General Social Survey, a long-running US survey
```

A tibble: 6 × 9

year	marital	age	race	rincome	partyid	relig	denom	tvhours
<int>	<fct>	<int>	<fct>	<fct>	<fct>	<fct>	<fct>	<int>
2000	Never married	26	White	$8000 to 9999	Ind,near rep	Protestant	Southern baptist	12
2000	Divorced	48	White	$8000 to 9999	Not str republican	Protestant	Baptist-dk which	NA
2000	Widowed	67	White	Not applicable	Independent	Protestant	No denomination	2
2000	Never married	39	White	Not applicable	Ind,near rep	Orthodox-christian	Not applicable	4
2000	Divorced	25	White	Not applicable	Not str democrat	None	Not applicable	1
2000	Married	25	White	$20000 - 24999	Strong democrat	Protestant	Southern baptist	NA

In [ ]:
```
?gss_cat
```

In [16]:
```
gss_cat |>
    count(race)
```

A tibble: 3 × 2

race	n
<fct>	<int>
Other	1959
Black	3129
White	16395

When working with factors, the two most common operations are changing the order of the levels, and changing the values of the levels. Those operations are described in the sections below.

gss_cat |> group_by(relig) |> summarise(n = n())

In [17]:
```
# What is the most common relig in this survey?
gss_cat |>
    count(relig) |>
    arrange(desc(n))
```

A tibble: 15 × 2

relig	n
<fct>	<int>
Protestant	10846
Catholic	5124
None	3523
Christian	689
Jewish	388
Other	224
Buddhism	147
Inter-nondenominational	109
Moslem/islam	104
Orthodox-christian	95
No answer	93
Hinduism	71
Other eastern	32
Native american	23
Don't know	15

In [18]:
```
# What's the most common partyid?
gss_cat |>
    count(partyid) |>
    arrange(desc(n))
```

A tibble: 10 × 2

partyid	n
<fct>	<int>
Independent	4119
Not str democrat	3690
Strong democrat	3490
Not str republican	3032
Ind,near dem	2499
Strong republican	2314
Ind,near rep	1791
Other party	393
No answer	154
Don't know	1

In [19]:
```
levels(gss_cat$partyid)
```

'No answer' · 'Don\'t know' · 'Other party' · 'Strong republican' · 'Not str republican' · 'Ind,near rep' · 'Independent' · 'Ind,near dem' · 'Not str democrat' · 'Strong democrat'

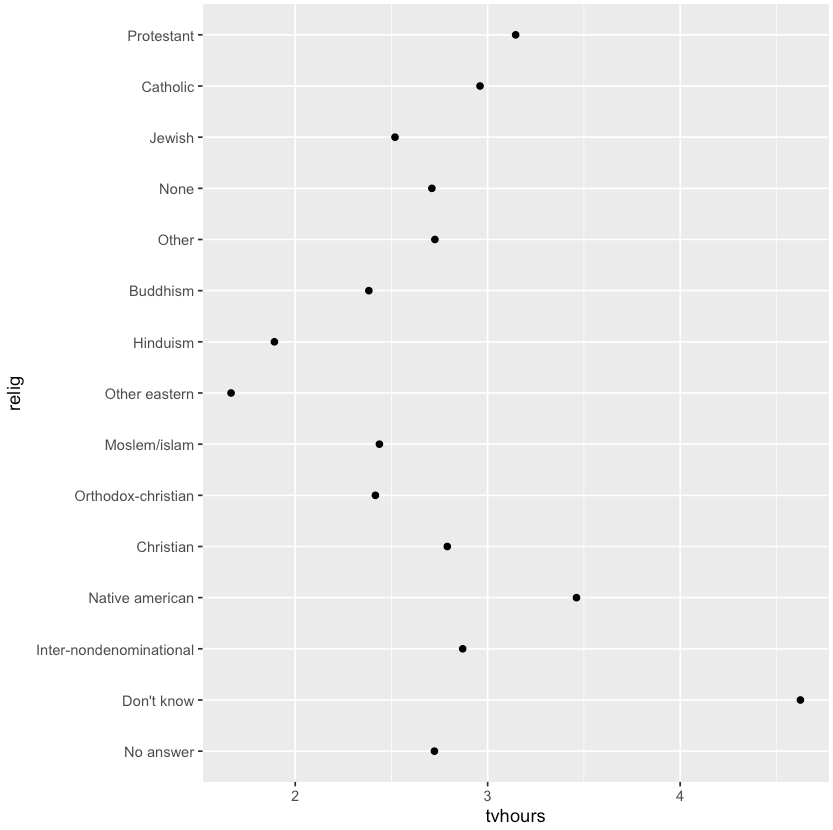
In [20]:
```
relig_summary <- gss_cat |>
  group_by(relig) |>
  summarize(
    tvhours = mean(tvhours, na.rm = TRUE),
    n = n()
  ) |>
    arrange(desc(tvhours))

relig_summary
```

A tibble: 15 × 3

relig	tvhours	n
<fct>	<dbl>	<int>
Don't know	4.625000	15
Native american	3.461538	23
Protestant	3.145487	10846
Catholic	2.960297	5124
Inter-nondenominational	2.870370	109
Christian	2.790281	689
Other	2.725806	224
No answer	2.723404	93
None	2.710227	3523
Jewish	2.518519	388
Moslem/islam	2.437500	104
Orthodox-christian	2.416667	95
Buddhism	2.382716	147
Hinduism	1.891892	71
Other eastern	1.666667	32

In [21]:
```
ggplot(relig_summary, aes(x = tvhours, y = relig)) +
  geom_point()
```



In [22]:
```
levels(gss_cat$relig)
```

'No answer' · 'Don\'t know' · 'Inter-nondenominational' · 'Native american' · 'Christian' · 'Orthodox-christian' · 'Moslem/islam' · 'Other eastern' · 'Hinduism' · 'Buddhism' · 'Other' · 'None' · 'Jewish' · 'Catholic' · 'Protestant' · 'Not applicable'

3. Reordering Factors

Factors can be reordered based on logical or statistical criteria:

A. Use `fct_reorder()` to reorder levels based on another variable.

```
fct_reorder(factor_var, numeric_var)
```

In [23]:
```
reordered_reli_tv <- relig_summary |>
    mutate(
        relig = fct_reorder(relig, tvhours) #relig(factor) is reorganized by
    ) |> arrange(relig)
    #ggplot(aes(x = tvhours, y = relig)) +
    #geom_point()
reordered_reli_tv
```

A tibble: 15 × 3

relig	tvhours	n
<fct>	<dbl>	<int>
Other eastern	1.666667	32
Hinduism	1.891892	71
Buddhism	2.382716	147
Orthodox-christian	2.416667	95
Moslem/islam	2.437500	104
Jewish	2.518519	388
None	2.710227	3523
No answer	2.723404	93
Other	2.725806	224
Christian	2.790281	689
Inter-nondenominational	2.870370	109
Catholic	2.960297	5124
Protestant	3.145487	10846
Native american	3.461538	23
Don't know	4.625000	15

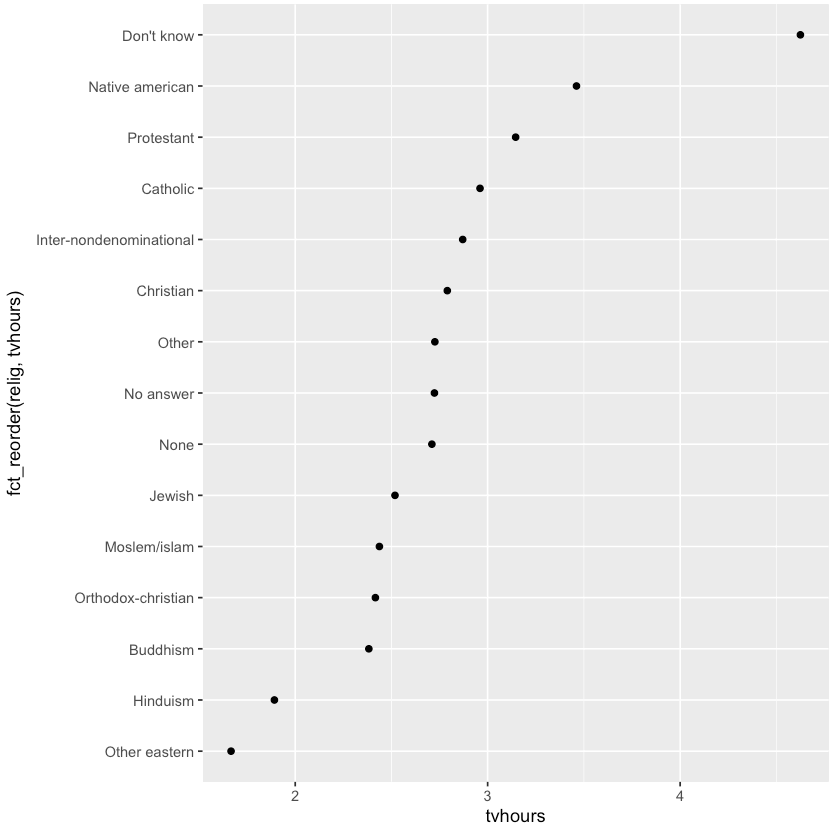
In [24]:
```
levels(reordered_reli_tv$relig)
#print("==============")
levels(relig_summary$relig)
```

'Other eastern' · 'Hinduism' · 'Buddhism' · 'Orthodox-christian' · 'Moslem/islam' · 'Jewish' · 'None' · 'No answer' · 'Other' · 'Christian' · 'Inter-nondenominational' · 'Catholic' · 'Protestant' · 'Native american' · 'Don\'t know' · 'Not applicable'

'No answer' · 'Don\'t know' · 'Inter-nondenominational' · 'Native american' · 'Christian' · 'Orthodox-christian' · 'Moslem/islam' · 'Other eastern' · 'Hinduism' · 'Buddhism' · 'Other' · 'None' · 'Jewish' · 'Catholic' · 'Protestant' · 'Not applicable'

relig_summary |> mutate(relig = fct_reorder(relig,tvhours)) |> ggplot(aes(x = tvhours, y = relig)) + geom_point()

In [25]:
```
ggplot(relig_summary, aes(x = tvhours, y = fct_reorder(relig, tvhours))) +
  geom_point()
```



B. Use `fct_relevel()` to move a level to the front or specify the exact order.

```
fct_relevel(factor, levels) # any levels not mentioned will be left in their existing order
```

In [26]:
```
f <- factor(c("a", "b", "c", "d"), levels = c("b", "c", "d", "a"))
levels(fct_relevel(f))
levels(fct_relevel(f, "a"))
levels(fct_relevel(f, "b", "a"))
```

'b' · 'c' · 'd' · 'a'

'a' · 'b' · 'c' · 'd'

'b' · 'a' · 'c' · 'd'

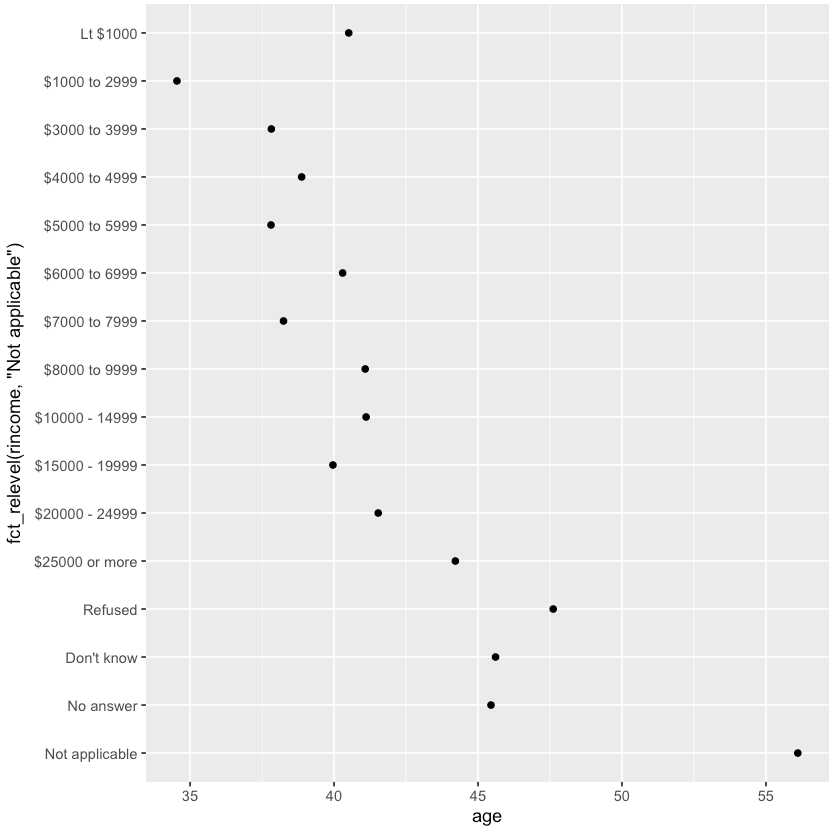
In [27]:
```
rincome_summary <- gss_cat |>
    group_by(rincome) |>
    summarize(
        age = mean(age, na.rm = TRUE),
        tvhours = mean(tvhours, na.rm = TRUE),
        n = n()
    )

rincome_summary
```

A tibble: 16 × 4

rincome	age	tvhours	n
<fct>	<dbl>	<dbl>	<int>
No answer	45.45029	2.904762	183
Don't know	45.60902	3.411290	267
Refused	47.61082	2.481973	975
$25000 or more	44.21217	2.234208	7363
$20000 - 24999	41.53365	2.784753	1283
$15000 - 19999	39.96180	2.912245	1048
$10000 - 14999	41.11301	3.016541	1168
$8000 to 9999	41.08235	3.148571	340
$7000 to 7999	38.24468	2.645455	188
$6000 to 6999	40.29907	3.174312	215
$5000 to 5999	37.81057	3.163793	227
$4000 to 4999	38.87500	3.145299	226
$3000 to 3999	37.82182	3.312102	276
$1000 to 2999	34.54430	3.004525	395
Lt $1000	40.51049	3.361842	286
Not applicable	56.10628	3.791468	7043

In [28]:
```
rincome_summary|>
    #ggplot(aes(x = age, y = rincome)) +
    ggplot(aes(x = age, y = fct_relevel(rincome,"Not applicable"))) +
        geom_point()
```



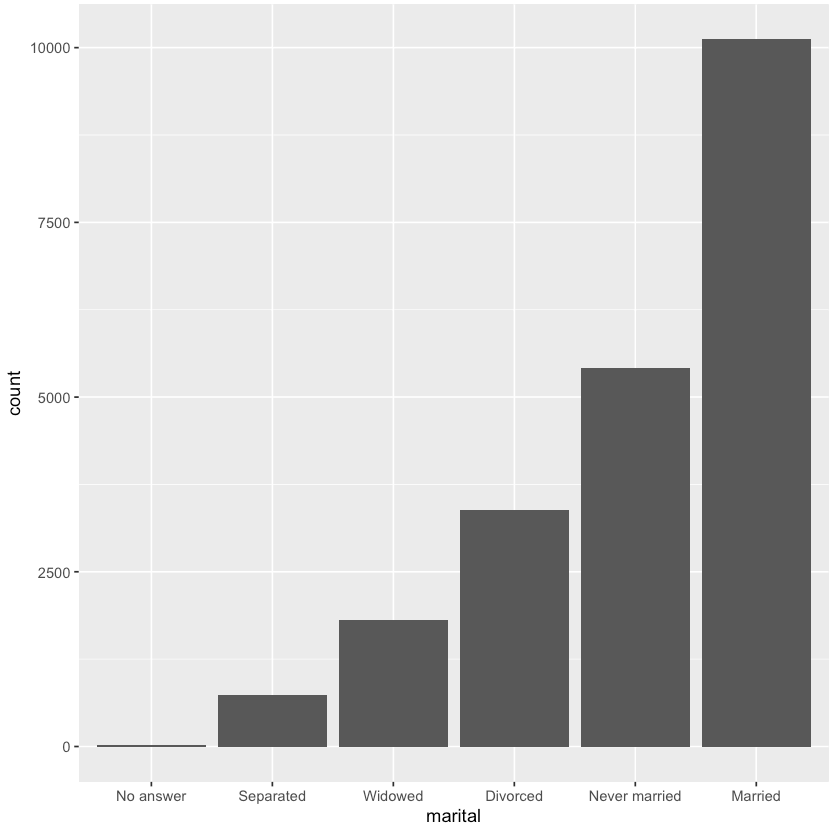
C. For changing the order of a bar plot,

1. by number of observations with each level (largest first, decreasing manner): `fct_infreq()`
2. Sort in reverse order: `fct_rev()`

In [29]:
```
levels(gss_cat$marital)
```

'No answer' · 'Never married' · 'Separated' · 'Divorced' · 'Widowed' · 'Married'

In [30]:
```
gss_cat |>
    mutate(
        marital = marital |>
            fct_infreq() |>
            fct_rev()
          ) |>
    ggplot(aes(x = marital)) +
        geom_bar()
```



4. Modifying Factor Levels

We can change the values of the levels!

A. **Rename** levels with `fct_recode()`:

```
fct_recode(factor_var, "new_level" = "old_level" )
```

In [31]:
```
levels(gss_cat$partyid)
```

'No answer' · 'Don\'t know' · 'Other party' · 'Strong republican' · 'Not str republican' · 'Ind,near rep' · 'Independent' · 'Ind,near dem' · 'Not str democrat' · 'Strong democrat'

In [32]:
```
gss_cat |>
    mutate(
        partyid = fct_recode(partyid,
            # Changing name: <new name> = <old name>
            "Republican, strong" = "Strong republican",
            "Republican, weak" = "Not str republican",
            "Independent, near rep" = "Ind,near rep",
            "Independent, near dem" = "Ind,near dem",
            "Democrat, weak" = "Not str democrat",
            "Democrat, strong" = "Strong democrat",
            # Combine groups by recycling the name
            "Other" = "No answer",
            "Other" = "Don\'t know",
            "Other" = "Other party"
```

```
        )
    ) |>
    count(partyid)
```

A tibble: 8 × 2

partyid	n
<fct>	<int>
Other	548
Republican, strong	2314
Republican, weak	3032
Independent, near rep	1791
Independent	4119
Independent, near dem	2499
Democrat, weak	3690
Democrat, strong	3490

B. **Collapse** levels with `fct_collapse()`:

```
fct_collapse(factor_var, new_level = c("old_level1", "old_level2"))
```

In [33]:
```
# If you have too many levels: `fct_collapse()` is much useful!
gss_cat |>
    mutate(
        partyid = fct_collapse(partyid,
            "other" = c("No answer","Don\'t know","Other party"),
            "rep" = c("Strong republican","Not str republican"),
            "ind" = c("Ind,near rep", "Independent", "Ind,near dem"),
            "dem" = c("Strong democrat","Not str democrat")
        )
    ) |>
    count(partyid)
```

A tibble: 4 × 2

partyid	n
<fct>	<int>
other	548
rep	5346
ind	8409
dem	7180

C. **Lump** uncommon factor **together** with `fct_lump*()`:

```
fct_lump_lowfreq(factor) #lumps together the least frequent levels
into "Other"
fct_lump_min(factor, min = <input_minimum>) #lumps levels that
appear fewer than minimum times
fct_lump_prop(factor, prop = p) #lumps levels that appear in fewer
than prop * n times
```

In [ ]:
```
?fct_lump_lowfreq
```

In [34]:
```
gss_cat |>
    mutate(relig= fct_lump_lowfreq(relig) ) |>
    count(relig)
# 'No answer''Don\'t know''Other party' lumped together
```

A tibble: 2 × 2

relig	n
<fct>	<int>
Protestant	10846
Other	10637