

Making it **easier** to be **tidy**

Data Wrangling: Session 8

Kieran Healy

Statistical Horizons, December 2022

Load the packages, as always




















```
library(here)      # manage file paths
library(socviz)    # data and some useful functions
library(tidyverse) # your friend and mine
library(haven)     # for Stata, SAS, and SPSS files

library(broom)     # tidy model summaries
```

Moving ahead

Some helpful things

The RStudio Community

Topic		Replies	Views	Activity
   Welcome to the RStudio Community! meta Welcome to community.rstudio.com — we're glad to have you! This welcome page will give you some advice on how to get the most out of the site if you're getting or giving help. We want this to be a friendly, inclusive com... read more		0	8.8k	2018-07-22
 what does that mean %in% in this code? • General	  	2	72	19m
 "William" in the Northeast ggplot Help • tidyverse ggplot2	  	5	69	28m
 Error while running RSelenium • shiny	 	4	27	28m
 Too slow to repsond •		0	16	40m
 Relative risk (RR) using categorical variables •		0	18	1h

The **reprex** package



reprex

part of the [tidyverse](#)
2.0.0.9000

Reference

Articles ▾

News ▾



Overview

Prepare reprexes for posting to [GitHub issues](#), [StackOverflow](#), in Slack [messages](#) or [snippets](#), or even to paste into PowerPoint or Keynote slides. What is a `reprex`? It's a **re**producible **ex**ample, as coined by [Romain Francois](#).

Given R code on the clipboard, selected in RStudio, as an expression (quoted or not), or in a file ...

- run it via `rmarkdown::render()`,
- with deliberate choices re: `render()` arguments, knitr options, and Pandoc options.

Get resulting runnable code + output as

- Markdown, suitable for GitHub or Stack Overflow or Slack, or as
- R code, augmented with commented output, or as
- Plain HTML or (experimental) Rich Text

The result is returned invisibly, written to a file and, if possible, placed on the clipboard. Preview an HTML version in RStudio viewer or default browser.

Installation



Links

Download from CRAN at

[https://cloud.r-project.org/
package=reprex](https://cloud.r-project.org/package=reprex)

Browse source code at

<https://github.com/tidyverse/reprex/>

Report a bug at

[https://github.com/tidyverse/reprex/
issues](https://github.com/tidyverse/reprex/issues)

License

[Full license](#)

MIT + file [LICENSE](#)

Community

[Contributing guide](#)

[Code of conduct](#)

Developers

Best demonstrated live

When asking for help, make a **reproducible example**

```
library(reprex)
```

```
library(tidyverse)
```

```
starwars |>  
  count(homeworld, species) |>  
  mutate(pct = n / sum(n) * 100) |>  
  arrange(desc(pct))
```

```
## # A tibble: 58 × 4  
##   homeworld species      n    pct  
##   <chr>      <chr>   <int> <dbl>  
## 1 Tatooine   Human         8  9.20  
## 2 Naboo     Human         5  5.75  
## 3 <NA>      Human         5  5.75  
## 4 Alderaan  Human         3  3.45  
## 5 Naboo     Gungan        3  3.45  
## 6 <NA>      Droid         3  3.45  
## 7 Corellia  Human         2  2.30  
## 8 Coruscant Human         2  2.30  
## 9 Kamino    Kaminoan      2  2.30  
## 10 Kashyyyk Wookiee       2  2.30  
## # ... with 48 more rows
```

The **usethis** package

usethis **2.0.1.9000**



Setup

Reference

Articles ▾

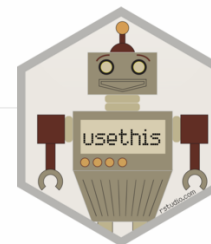
News ▾

Search...



usethis

usethis is a workflow package: it automates repetitive tasks that arise during project setup and development, both for R packages and non-package projects.



Installation

Install the released version of usethis from CRAN:

```
install.packages("usethis")
```

Or install the development version from GitHub with:

```
# install.packages("devtools")
devtools::install_github("r-lib/usethis")
```

Usage

Most `use_*()` functions operate on the *active project*: literally, a directory on your computer. If you've just used usethis to create a new package or project, that will be the active project. Otherwise, usethis verifies that current working directory is or is below a valid project directory and that becomes the active project. Use `proj_get()` or `proj_sitrep()` to manually query the project and [read more in the docs](#).

A few usethis functions have no strong connections to projects and will expect you to provide a path.

Links

Download from CRAN at
<https://cloud.r-project.org/package=usethis>

Browse source code at
<https://github.com/r-lib/usethis/>

Report a bug at
<https://github.com/r-lib/usethis/issues>

License

[Full license](#)

MIT + file [LICENSE](#)

Community

[Contributing guide](#)

[Code of conduct](#)

Developers

[Hadley Wickham](#)

Author

[Jennifer Bryan](#)

Author, maintainer

[Malcolm Barrett](#)

Author

The packages that made these slides



Never **paste** tables into a slide again!

```
---  
  
# Never .kjh-orange[copy and paste] code to a slide again!  
  
.pull-left[  
  
![:scale 100%](img/xaringan-sample.png)  
  
]  
  
.pull-right[  
  
```{r}  
Oh no, its the GSS
gss_sm %>%
 count(bigregion, religion)
```,  
  
]
```

religion	Northeast	Midwest	South	West
Protestant	158	325	650	238
Catholic	162	172	160	155
Jewish	27	3	11	10
None	112	157	170	180
Other	28	33	50	48
NA	1	5	11	1

Tables, tables, tables

The **gtsummary** package is very powerful. There are a number of other very good tidy table-making options too.

```
library(gtsummary)
```

```
trial
```

```
## # A tibble: 200 × 8
##   trt      age marker stage grade response death ttdeath
##   <chr>  <dbl>  <dbl> <fct> <fct>    <int> <int>   <dbl>
## 1 Drug A    23  0.16  T1    II         0     0     24
## 2 Drug B     9  1.11  T2     I         1     0     24
## 3 Drug A    31  0.277 T1    II         0     0     24
## 4 Drug A    NA  2.07  T3   III         1     1    17.6
## 5 Drug A    51  2.77  T4   III         1     1    16.4
## 6 Drug B    39  0.613 T4     I         0     1    15.6
## 7 Drug A    37  0.354 T1    II         0     0     24
## 8 Drug A    32  1.74  T1     I         0     1    18.4
## 9 Drug A    31  0.144 T1    II         0     0     24
## 10 Drug B   34  0.205 T3     I         0     1    10.5
## # ... with 190 more rows
```

Tables, tables, tables

The **gtsummary** package is very powerful. There are a number of other very good tidy table-making options too.

```
trial |>
  tbl_summary(
    by = trt, # split table by group
    missing = "no" # don't list missing data separately
  ) |>
  add_n() |> # add column with total number of non-missing observations
  add_p() |> # test for a difference between groups
  modify_header(label = "**Variable**") |> # update the column header
  bold_labels()
```

Tables, tables, tables

Variable	N	Drug A, N = 98 ¹	Drug B, N = 102 ¹	p-value ²
Age	189	46 (37, 59)	48 (39, 56)	0.7
Marker Level (ng/mL)	190	0.84 (0.24, 1.57)	0.52 (0.19, 1.20)	0.085
T Stage	200			0.9
T1		28 (29%)	25 (25%)	
T2		25 (26%)	29 (28%)	
T3		22 (22%)	21 (21%)	
T4		23 (23%)	27 (26%)	
Grade	200			0.9
I		35 (36%)	33 (32%)	
II		32 (33%)	36 (35%)	
III		31 (32%)	33 (32%)	
Tumor Response	193	28 (29%)	33 (34%)	0.5
Patient Died	200	52 (53%)	60 (59%)	0.4
Months to Death/Censor	200	23.5 (17.4, 24.0)	21.2 (14.6, 24.0)	0.14
¹ Median (IQR); n (%)				
² Wilcoxon rank sum test; Pearson's Chi-squared test				

Tables, tables, tables

gtsummary() straight out of the box:

```
gss_sm |>
  select(race, degree, marital) |>
  drop_na() |>
  tbl_summary(
    by = race, # split table by group
    missing = "no" # don't list missing data separately
  ) |>
  add_n() |> # add column with total number of non-missing observations
# modify_header(label = "**Variable**") |> # update the column header
bold_labels()
```

Tables, tables, tables

Variable	N	White, N = 2,096 ¹	Black, N = 487 ¹	Other, N = 276 ¹
degree	2,859			
Lt High School		197 (9.4%)	60 (12%)	71 (26%)
High School		1,057 (50%)	292 (60%)	112 (41%)
Junior College		166 (7.9%)	33 (6.8%)	17 (6.2%)
Bachelor		426 (20%)	71 (15%)	39 (14%)
Graduate		250 (12%)	31 (6.4%)	37 (13%)
marital	2,859			
Married		979 (47%)	121 (25%)	110 (40%)
Widowed		196 (9.4%)	35 (7.2%)	18 (6.5%)
Divorced		363 (17%)	93 (19%)	39 (14%)
Separated		55 (2.6%)	27 (5.5%)	20 (7.2%)
Never Married		503 (24%)	211 (43%)	89 (32%)
¹ n (%)				

Tables, tables, tables

With a bit more work ...

```
trial |>
  select(trt, age, marker) |>
  tbl_summary(
    by = trt,
    type = all_continuous() ~ "continuous2",
    statistic = all_continuous() ~ c("{N_nonmiss}",
                                      "{mean} ({sd})",
                                      "{median} ({p25}, {p75})",
                                      "{min}, {max}"),
    missing = "no"
  ) |>
  italicize_levels()
```


Tables, tables, tables

Characteristic	Drug A, N = 98	Drug B, N = 102
Age		
<i>N</i>	91	98
<i>Mean (SD)</i>	47 (15)	47 (14)
<i>Median (IQR)</i>	46 (37, 59)	48 (39, 56)
<i>Range</i>	6, 78	9, 83
Marker Level (ng/mL)		
<i>N</i>	92	98
<i>Mean (SD)</i>	1.02 (0.89)	0.82 (0.83)
<i>Median (IQR)</i>	0.84 (0.24, 1.57)	0.52 (0.19, 1.20)
<i>Range</i>	0.00, 3.87	0.01, 3.64

The power of lists

Similar to earlier, but simpler:

```
library(gapminder)

## Fit as a function, for clarity
fit_ols <- function(df) {
  lm(lifeExp ~ log(gdpPercap), data = df)
}

out_le <- gapminder |>
  filter(continent %in% "Oceania") |>
  group_by(continent) |>
  nest() |>
  mutate(model = map(data, fit_ols),
         mod_sum = map(model, glance),
         mod_terms = map(model, tidy, conf.int = TRUE),
         ) |>
  unnest(cols = c(mod_terms))
```

The power of lists

```
out_le
```

```
## # A tibble: 8 × 11
## # Groups:   continent [4]
##   continent data      model  mod_sum term      estim...1 std.e...2 stati...3 p.value
##   <fct>      <list>   <list> <list>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 Asia      <tibble> <lm>   <tibble> (Interce... 9.58       2.71       3.54 4.46e- 4
## 2 Asia      <tibble> <lm>   <tibble> log(gdpP... 6.25       0.331      18.9 3.73e- 57
## 3 Europe    <tibble> <lm>   <tibble> (Interce... 13.0       1.92       6.76 5.52e- 11
## 4 Europe    <tibble> <lm>   <tibble> log(gdpP... 6.31       0.205      30.8 8.06e-103
## 5 Africa    <tibble> <lm>   <tibble> (Interce... 7.60       2.63       2.89 4.03e- 3
## 6 Africa    <tibble> <lm>   <tibble> log(gdpP... 5.69       0.361      15.8 1.86e- 47
## 7 Americas  <tibble> <lm>   <tibble> (Interce... -19.1      4.82      -3.95 9.65e- 5
## 8 Americas  <tibble> <lm>   <tibble> log(gdpP... 9.72       0.558      17.4 2.51e- 47
## # ... with 2 more variables: conf.low <dbl>, conf.high <dbl>, and abbreviated
## #   variable names 1estimate, 2std.error, 3statistic
```

The power of lists

```
## Nice formatting of the numbers
## There are many convenience packages
## like this; it's not too hard to write your own, either
# remotes::install_github("tjmahr/printy")

text_ready <- out_le |>
  mutate(
    across(c(estimate, conf.low, conf.high),
           printy::fmt_fix_digits, 2),
    se = printy::fmt_fix_digits(std.error, 3),
    ci = glue::glue("[{conf.low}, {conf.high}]")
  ) |>
  select(continent, term, estimate, se, ci)
```

The printy package is by [T.J. Mahr](#)

The power of lists

```
text_ready
```

```
## # A tibble: 8 × 5
## # Groups:   continent [4]
##   continent term      estimate se      ci
##   <fct>      <chr>      <chr>    <chr> <glue>
## 1 Asia      (Intercept)    9.58    2.706 [4.26, 14.90]
## 2 Asia      log(gdpPercap) 6.25     0.331 [5.60, 6.90]
## 3 Europe    (Intercept)   12.97    1.917 [9.19, 16.74]
## 4 Europe    log(gdpPercap) 6.31     0.205 [5.91, 6.71]
## 5 Africa    (Intercept)    7.60    2.632 [2.43, 12.77]
## 6 Africa    log(gdpPercap) 5.69     0.361 [4.98, 6.40]
## 7 Americas  (Intercept)   -19.07    4.824 [-28.56, -9.58]
## 8 Americas  log(gdpPercap) 9.72     0.558 [8.62, 10.82]
```

The power of **lists**

Now ...

```
stats <- text_ready |>
  mutate(term = janitor::make_clean_names(term)) |>
  printy::super_split(continent, term) # Thanks again, TJ Mahr
```

The power of lists

Why are we doing this?

```
stats
```

```
## $Africa
## $Africa$intercept
## # A tibble: 1 × 5
## # Groups:   continent [1]
##   continent term      estimate se      ci
##   <fct>      <chr>      <chr>   <chr> <glue>
## 1 Africa    intercept  7.60    2.632 [2.43, 12.77]
##
## $Africa$log_gdp_percap
## # A tibble: 1 × 5
## # Groups:   continent [1]
##   continent term      estimate se      ci
##   <fct>      <chr>      <chr>   <chr> <glue>
## 1 Africa    log_gdp_percap 5.69    0.361 [4.98, 6.40]
##
##
## $Americas
## $Americas$intercept
## # A tibble: 1 × 5
## # Groups:   continent [1]
##   continent term      estimate se      ci
##   <fct>      <chr>      <chr>   <chr> <glue>
## 1 Americas  intercept -19.07   4.824 [-28.56, -9.58]
##
## $Americas$log_gdp_percap
## # A tibble: 1 × 5
```

The power of **lists**

The Intercept term for Africa was `'r stats$Africa$intercept$estimate' 'r stats$Africa$intercept$ci'`.

For Europe it was `'r stats$Europe$intercept$estimate' 'r stats$Europe$intercept$ci'`

The Intercept term for Africa was 7.60 [2.43, 12.77].

For Europe it was 12.97 [9.19, 16.74].

For more, see [this post by TJ Mahr](https://www.tjmahr.com/lists-knitr-secret-weapon/):

<https://www.tjmahr.com/lists-knitr-secret-weapon/>

Test the output of your functions

```
countries <- read_csv(here("data", "countries.csv"))
```

```
countries
```

```
## # A tibble: 213 × 4
##   cname      iso3 iso2 continent
##   <chr>      <chr> <chr> <chr>
## 1 Afghanistan AFG  AF   Asia
## 2 Algeria      DZA  DZ   Africa
## 3 Armenia      ARM  AM   Asia
## 4 Australia    AUS  AU   Oceania
## 5 Austria      AUT  AT   Europe
## 6 Azerbaijan   AZE  AZ   Asia
## 7 Bahrain      BHR  BH   Asia
## 8 Belarus      BLR  BY   Europe
## 9 Belgium      BEL  BE   Europe
## 10 Brazil       BRA  BR   South America
## # ... with 203 more rows
```

Test the output of your functions

```
get_stmf <- function(url = "https://www.mortality.org/File/GetDocument/Public/STMF/Outputs",
  fname = "stmf",
  date = lubridate::today(),
  ext = "csv",
  dest = "data-raw/data",
  save_file = c("n", "y"),
  ...) {
  save_file <- match.arg(save_file)
  target <- fs::path(url, fname, ext = ext)
  message("target: ", target)

  destination <- fs::path(here::here("data-raw/data"),
    paste0(fname, "_", date), ext = ext)

  tf <- tempfile(fileext = ext)
  curl::curl_download(target, tf)

  switch(save_file,
    y = fs::file_copy(tf, destination),
    n = NULL)

  janitor::clean_names(read_csv(tf, ...))
}
```

Test the output of your functions

```
stmf_raw <- read_csv(here("data", "stmf.csv"), skip = 2) |>
  janitor::clean_names() |>
  rename(deaths_total = d_total, rate_total = r_total) |>
  select(country_code:sex, deaths_total, rate_total, split:forecast, everything()) |>
  pivot_longer(
    cols = d0_14:r85p,
    names_to = c("measure", "age_group"),
    names_pattern = "(r|d)(.*)"
  ) |>
  pivot_wider(names_from = measure,
              values_from = value) |>
  mutate(age_group = stringr::str_replace(age_group, "_", "-"),
         age_group = stringr::str_replace(age_group, "p", "+")) |>
  rename(death_count = d, death_rate = r) |>
  mutate(approx_date = paste0(year, "-", "W",
                              stringr::str_pad(week, width = 2, pad = "0"), "-", "7"),
         approx_date = ISOweek::ISOweek2date(approx_date)) |>
  select(country_code:sex, split:forecast, approx_date,
         age_group:death_rate, deaths_total, rate_total) |>
  mutate(country_code = replace(country_code, country_code == "AUS2", "AUS"),
         country_code = replace(country_code, country_code == "NZL_NP", "NZL"))
```

Test the output of your functions

```
stmf_raw
```

```
## # A tibble: 576,840 × 13
##   country_...1 year week sex split split...2 forec...3 approx_d...4 age_g...5 death...6
##   <chr>      <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <date> <chr> <dbl>
## 1 AUS      2015     1 m     1     0     0 2015-01-04 0-14     17.2
## 2 AUS      2015     1 m     1     0     0 2015-01-04 15-64    359.
## 3 AUS      2015     1 m     1     0     0 2015-01-04 65-74    250
## 4 AUS      2015     1 m     1     0     0 2015-01-04 75-84    436
## 5 AUS      2015     1 m     1     0     0 2015-01-04 85+     413
## 6 AUS      2015     1 f     1     0     0 2015-01-04 0-14     14.0
## 7 AUS      2015     1 f     1     0     0 2015-01-04 15-64    199.
## 8 AUS      2015     1 f     1     0     0 2015-01-04 65-74    183
## 9 AUS      2015     1 f     1     0     0 2015-01-04 75-84    350
## 10 AUS     2015     1 f     1     0     0 2015-01-04 85+     704
## # ... with 576,830 more rows, 3 more variables: death_rate <dbl>,
## #   deaths_total <dbl>, rate_total <dbl>, and abbreviated variable names
## #   1country_code, 2split_sex, 3forecast, 4approx_date, 5age_group,
## #   6death_count
```

Test the output of your functions

```
md_ccodes <- tibble(country_code = unique(stmf_raw$country_code)) |>
  left_join(countries, by = c("country_code" = "iso3")) |>
  mutate(cname = replace(cname, country_code == "DEUTNP", "Germany"),
         iso2 = replace(iso2, country_code == "DEUTNP", "DE"),
         continent = replace(continent, country_code == "DEU", "Europe"),
         cname = replace(cname, country_code == "FRATNP", "France"),
         iso2 = replace(iso2, country_code == "FRATNP", "FR"),
         continent = replace(continent, country_code == "FRA", "Europe"),
         cname = replace(cname, country_code == "GBRTENW", "England and Wales"),
         cname = replace(cname, country_code == "GBR_SCO", "Scotland"),
         cname = replace(cname, country_code == "GBR_NIR", "Northern Ireland"),
         continent = replace(continent, country_code %in% c("GBRTENW", "GBR_SCO", "GBR_NIR"), "Europe")
  ) |>
  left_join(countries)

stmf <- left_join(stmf_raw, md_ccodes) |>
  select(country_code, cname:iso3, everything()) |>
  mutate(iso3 = replace(iso3, iso2 == "DE", "DEU"),
         iso3 = replace(iso3, iso2 == "FR", "FRA"))
```

Test the output of your functions

```
stmf
```

```
## # A tibble: 576,840 × 17
##   country_c...1 cname iso2 conti...2 iso3   year  week sex   split split...3 forec...4
##   <chr>         <chr> <chr> <chr>   <chr> <dbl> <dbl> <chr> <dbl>   <dbl>   <dbl>
## 1 AUS          Aust... AU   Oceania AUS    2015     1 m     1       0       0
## 2 AUS          Aust... AU   Oceania AUS    2015     1 m     1       0       0
## 3 AUS          Aust... AU   Oceania AUS    2015     1 m     1       0       0
## 4 AUS          Aust... AU   Oceania AUS    2015     1 m     1       0       0
## 5 AUS          Aust... AU   Oceania AUS    2015     1 m     1       0       0
## 6 AUS          Aust... AU   Oceania AUS    2015     1 f     1       0       0
## 7 AUS          Aust... AU   Oceania AUS    2015     1 f     1       0       0
## 8 AUS          Aust... AU   Oceania AUS    2015     1 f     1       0       0
## 9 AUS          Aust... AU   Oceania AUS    2015     1 f     1       0       0
## 10 AUS         Aust... AU   Oceania AUS    2015     1 f     1       0       0
## # ... with 576,830 more rows, 6 more variables: approx_date <date>,
## #   age_group <chr>, death_count <dbl>, death_rate <dbl>, deaths_total <dbl>,
## #   rate_total <dbl>, and abbreviated variable names 1country_code, 2continent,
## #   3split_sex, 4forecast
```

For example, manually

```
## stmf
stmf_colnames <- c("country_code", "cname", "iso2", "continent", "iso3", "year",
                  "week", "sex", "split", "split_sex", "forecast", "approx_date",
                  "age_group", "death_count", "death_rate", "deaths_total", "rate_total")

all.equal(colnames(stmf), stmf_colnames)
```

```
## [1] TRUE
```

For example, manually

```
## stmf
stmf_colnames <- c("country_code", "cname", "iso2", "continent", "iso3", "year",
                  "week", "sex", "split", "split_sex", "forecast", "approx_date",
                  "age_group", "death_count", "death_rate", "deaths_total", "rate_total")

all.equal(colnames(stmf), stmf_colnames)
```

```
## [1] TRUE
```

Imagine how you might build up a set of tests and checks

For example, manually

```
## stmf
stmf_colnames <- c("country_code", "cname", "iso2", "continent", "iso3", "year",
                  "week", "sex", "split", "split_sex", "forecast", "approx_date",
                  "age_group", "death_count", "death_rate", "deaths_total", "rate_total")

all.equal(colnames(stmf), stmf_colnames)
```

```
## [1] TRUE
```

Imagine how you might build up a set of tests and checks

But you don't have to manage this manually

Use **testthat** to check things

```
## countries
test_that("countries conforms to spec", {
  countries_colnames <- c("cname", "iso3", "iso2", "continent")
  expect_equal(colnames(countries), countries_colnames)
})

## stmf
test_that("stmf conforms to spec", {
  stmf_colnames <- c("country_code", "cname", "iso2", "continent", "iso3", "year",
                    "week", "sex", "split", "split_sex", "forecast", "approx_date",
                    "age_group", "death_count", "death_rate", "deaths_total", "rate_total")
  expect_equal(colnames(stmf), stmf_colnames)
})
```

Use **testthat** to check things

```
testthat::test_dir(here("tests", "testthat"))
```

```
## ✓ | OK F W S | Context
```

```
##
```

```
## - | 0 | stmf
```

```
## - | 0 | Validating package data objects
```

```
## ✓ | 2 | Validating package data objects
```

```
##
```

```
## == Results ==
```

```
## [ FAIL 0 | WARN 0 | SKIP 0 | PASS 2 ]
```

testthat in practice

Oriented towards package development

Consider packaging your datasets! Benefits to documentation/codebooks etc

One-table example: **uscenpops**

More extensive: **covdata**

How R packages work: **Wickham & Bryan**

Summarizing your wrangling with **skimr**

We might want to make a **codebook** of our data

```
library(skimr)
```

Summarizing your wrangling with **skimr**

We might want to make a **codebook** of our data

```
library(skimr)
organdata <- read_csv(here("data", "organdonation.csv"))
```

Summarizing your wrangling with **skimr**

```
organdata |> skim(where(is.numeric)) |> partition()
```

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
year	34	0.86	1996.50	3.46	1991.00	1993.75	1996.50	1999.25	2002.00	
donors	34	0.86	16.48	5.11	5.20	13.00	15.10	19.60	33.90	
pop	17	0.93	39921.29	62219.22	3514.00	6938.00	15531.00	57301.00	288369.00	
pop.dens	17	0.93	12.00	11.09	0.22	1.94	9.49	19.11	38.89	
gdp	17	0.93	22986.18	4665.92	12917.00	19546.00	22756.00	26180.00	36554.00	
gdp.lag	0	1.00	22574.92	4790.71	11434.00	19034.25	22158.00	25886.50	36554.00	
health	0	1.00	2073.75	733.59	791.00	1581.00	1956.00	2407.50	5665.00	
health.lag	0	1.00	1972.99	699.24	727.00	1542.00	1850.50	2290.25	5267.00	
pubhealth	21	0.91	6.19	0.92	4.30	5.50	6.00	6.90	8.80	
roads	17	0.93	113.04	36.33	58.21	83.46	111.22	139.57	232.48	
cerebvas	17	0.93	610.80	144.45	300.00	500.00	604.00	698.00	957.00	
assault	17	0.93	16.53	17.33	4.00	9.00	11.00	16.00	103.00	
external	17	0.93	450.06	118.19	258.00	367.00	421.00	534.00	853.00	
txp.pop	17	0.93	0.72	0.20	0.22	0.63	0.71	0.83	1.12	

Summarizing your wrangling with **skimr**

```
organdata |> skim(!where(is.numeric)) |> partition()
```

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
country	0	1.00	5	14	0	17	0
world	14	0.94	6	11	0	3	0
opt	28	0.88	2	3	0	2	0
consent.law	0	1.00	8	8	0	2	0
consent.practice	0	1.00	8	8	0	2	0
consistent	0	1.00	2	3	0	2	0
ccode	0	1.00	2	4	0	17	0

Custom Summaries

```
stmf_country_years <- function(df = stmf) {  
  df |>  
    dplyr::select(cname, year) |>  
    dplyr::group_by(cname, year) |>  
    dplyr::tally() |>  
    dplyr::mutate(n = as.character(n),  
                 n = dplyr::recode(n, "0" = "-", .default = "Y")) |>  
    dplyr::group_by(year, cname) |>  
    dplyr::arrange(year) |>  
    tidyr::pivot_wider(names_from = year, values_from = n) |>  
    dplyr::mutate(dplyr::across(where(is.character), dplyr::recode, .missing = "-")) |>  
    dplyr::arrange(cname)  
}
```

Custom Summaries

knitr::kable(stmf_country_years())

cname	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Australia	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y
Austria	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Belgium	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Bulgaria	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Canada	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Chile	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y
Croatia	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Czech Republic	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Denmark	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
England and Wales	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Estonia	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Finland	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
France	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Germany	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Greece	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y
Hungary	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Iceland	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Israel	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

