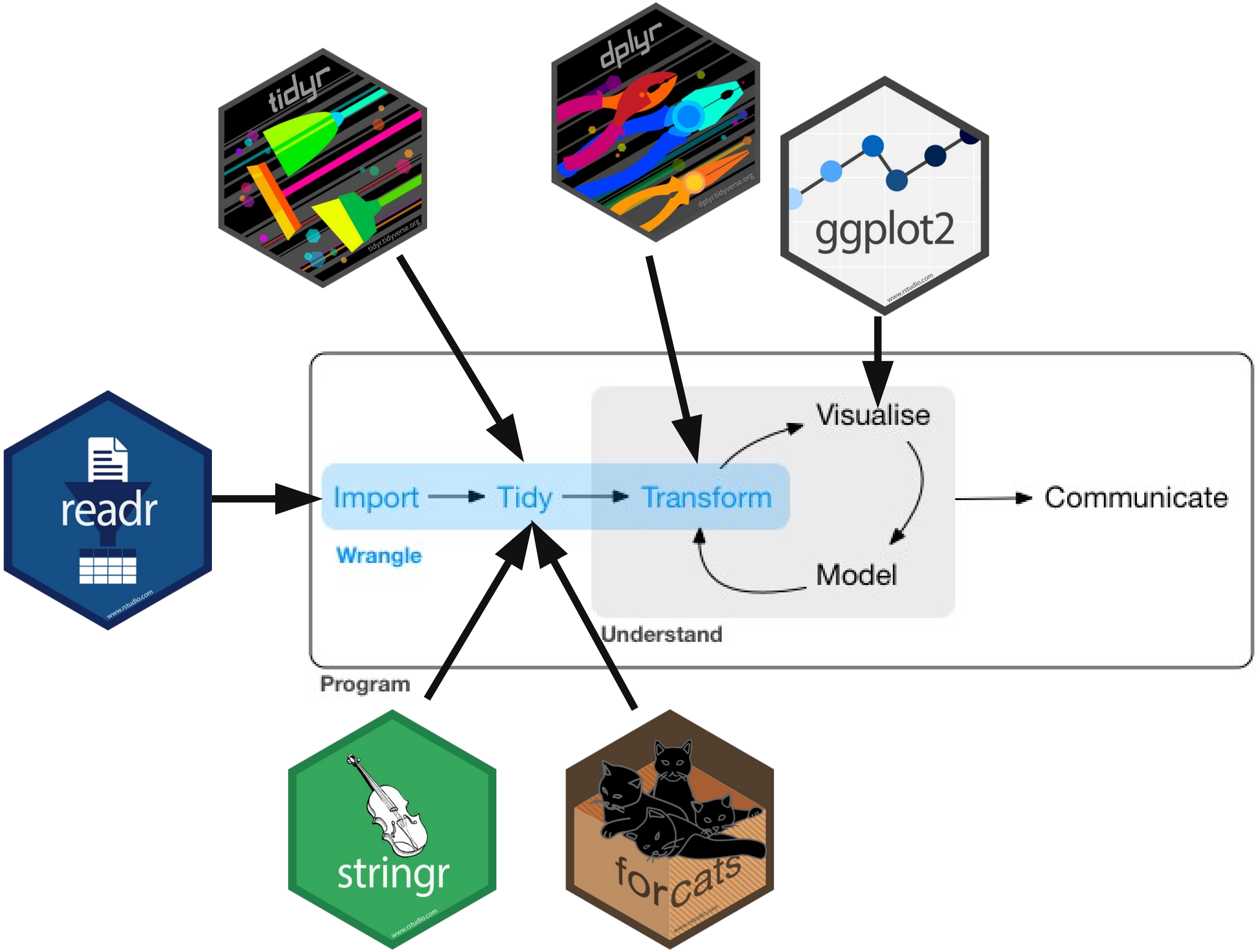
GG606

Factors & Dates



Import → Tidy → Transform

Wrangle

Visualise

Model

Understand

→ Communicate

Program

python: arrow, numpy, siuba, (pycats in pandas)

Factors

- Categorical variables: fixed, known set of possible values
- nb: Base R often converts characters to factors
- Read about history of `stringsAsFactors`

Factors

$\mathbb{N}$

- Look-up table between Natural Numbers table of values
- Example:
  - 1 = ctrl
  - 2 = trtA
  - 3 = trtB

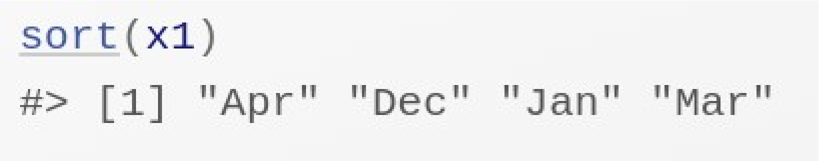
  1 = Apr

  2 = Dec

  3 = Jan

Nice Worked Example

- Two vectors
- `x1 ← c("Dec", "Apr", "Jan", "Mar")`
- `x2 ← c("Dec", "Apr", "Jam", "Mar")`
- `sort(x1)`
- Specify levels or use what’s there?

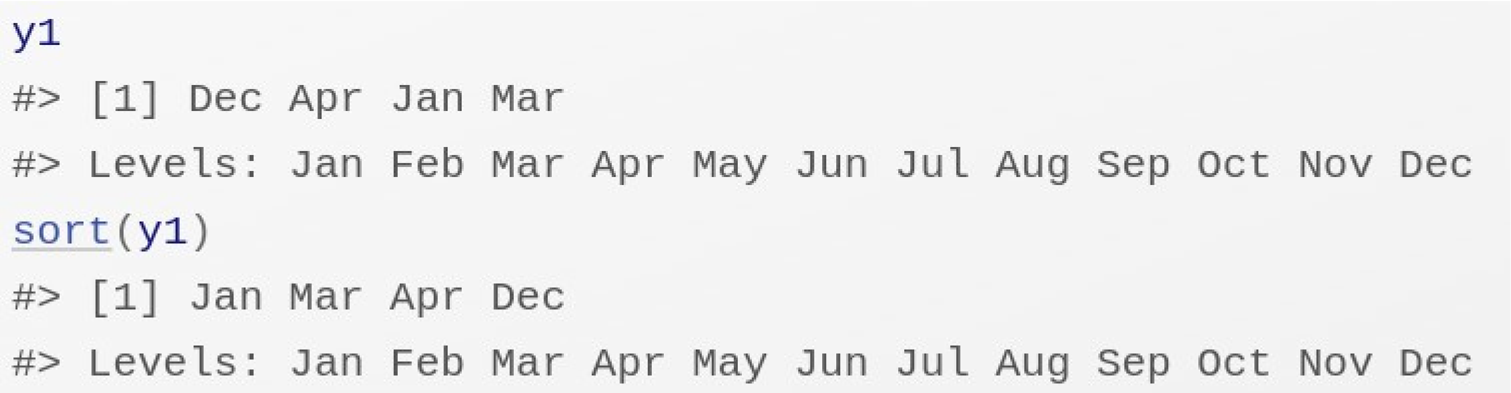


```
sort(x1)
#> [1] "Apr" "Dec" "Jan" "Mar"
```

Nice Worked Example

-
  ```
  month_levels <- c(
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
  ```

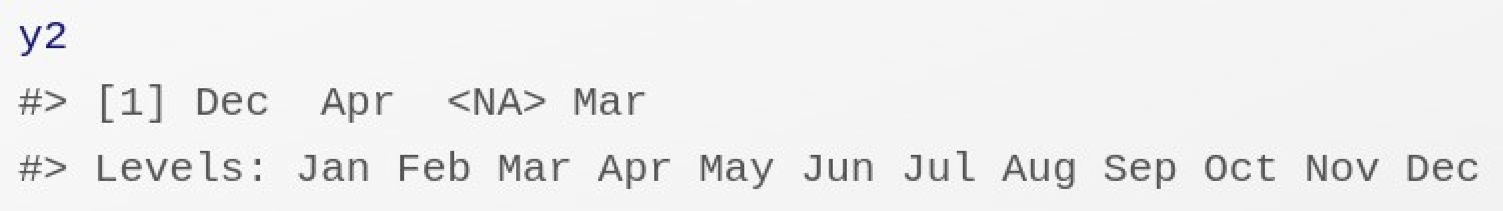
-
  ```
  y1 <- factor(x1, levels = month_levels)
  ```



```
y1
#> [1] Dec Apr Jan Mar
#> Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
sort(y1)
#> [1] Jan Mar Apr Dec
#> Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

Nice Worked Example

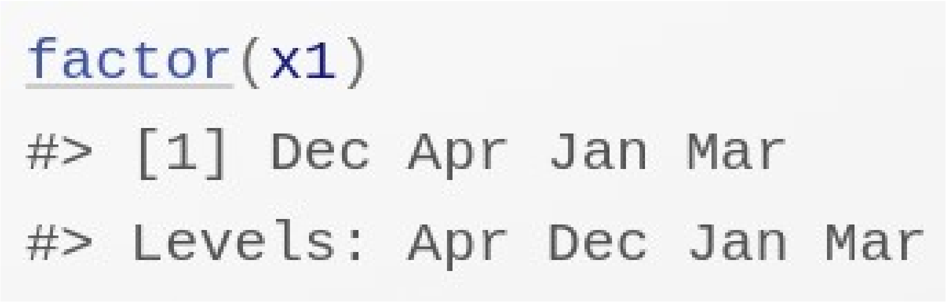
- `month_levels ← c(`
  `"Jan", "Feb", "Mar", "Apr", "May", "Jun",`
  `"Jul", "Aug", "Sep", "Oct", "Nov", "Dec")`
- `y1 ← factor(x1, levels = month_levels)`
- `y2 ← factor(x2, levels = month_levels)`



```
y2
#> [1] Dec  Apr  <NA> Mar
#> Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

Common (less safe?) Approach

- `factor(x1)`
- But only has some levels
- When would this matter or not matter?

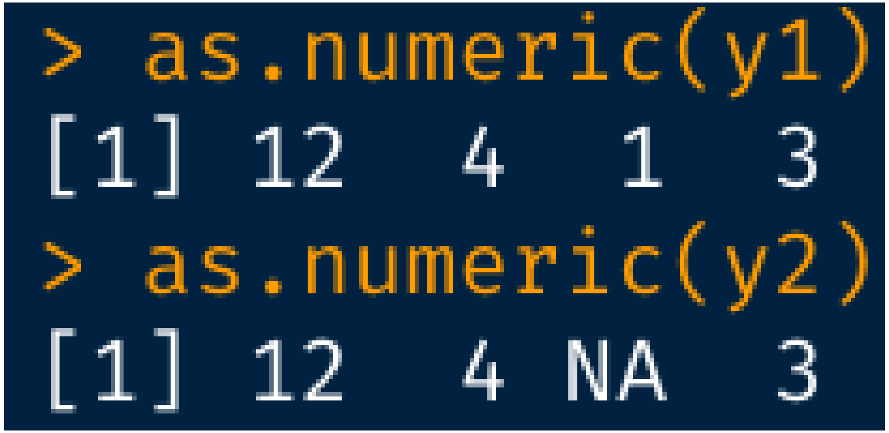


```
factor(x1)
#> [1] Dec Apr Jan Mar
#> Levels: Apr Dec Jan Mar
```

Factors

- Look-up table between Natural Numbers table of values

$\mathbb{N}$



```
> as.numeric(y1)
[1] 12  4  1  3
> as.numeric(y2)
[1] 12  4 NA  3
```

General Social Survey

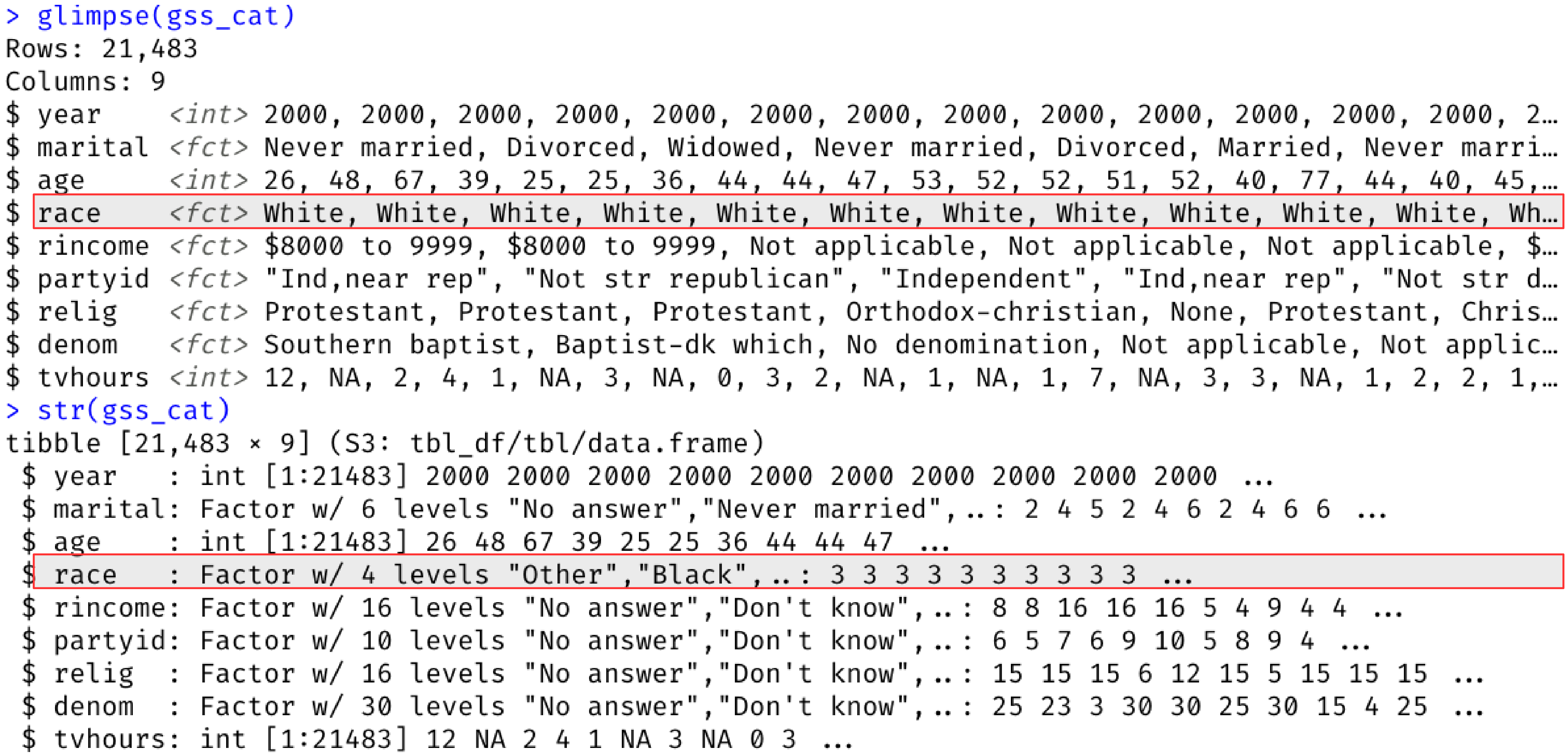
- Long-running US survey
- Good example of problems/challenges
- `forcats::gss_cat`
- Not dissimilar to census data

The General Social Survey
GSS

NORC at the University of Chicago

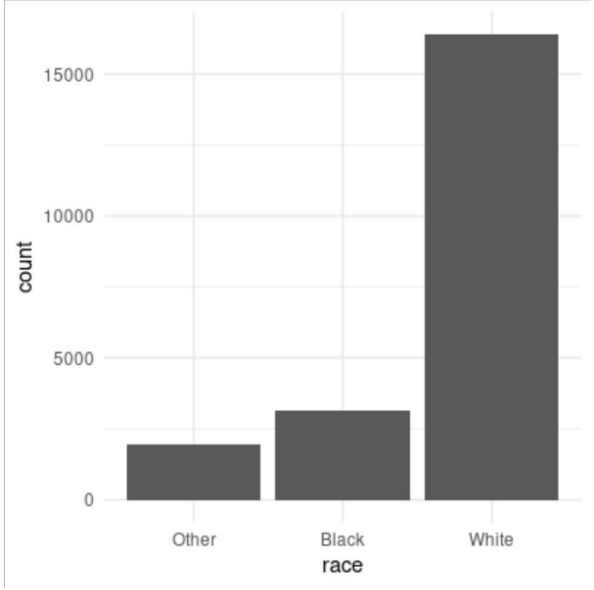
`gss_cat`

- 9 columns
- Integers and factors
- `glimpse(gss_cat)`
- `str(gss_cat)`



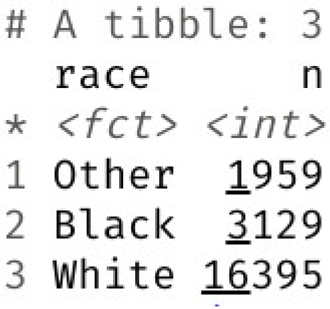
```
> glimpse(gss_cat)
Rows: 21,483
Columns: 9
$ year    <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2…
$ marital <fct> Never married, Divorced, Widowed, Never married, Divorced, Married, Never marri…
$ age     <int> 26, 48, 67, 39, 25, 25, 36, 44, 44, 47, 53, 52, 52, 51, 52, 40, 77, 44, 40, 45,…
$ race    <fct> White, White, White, White, White, White, White, White, White, White, White, Wh…
$ rincome <fct> $8000 to 9999, $8000 to 9999, Not applicable, Not applicable, Not applicable, $…
$ partyid <fct> "Ind,near rep", "Not str republican", "Independent", "Ind,near rep", "Not str d…
$ relig   <fct> Protestant, Protestant, Protestant, Orthodox-christian, None, Protestant, Chris…
$ denom   <fct> Southern baptist, Baptist-dk which, No denomination, Not applicable, Not applic…
$ tvhours <int> 12, NA, 2, 4, 1, NA, 3, NA, 0, 3, 2, NA, 1, NA, 1, 7, NA, 3, 3, NA, 1, 2, 2, 1,…
> str(gss_cat)
tibble [21,483 × 9] (S3: tbl_df/tbl/data.frame)
 $ year   : int [1:21483] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
 $ marital: Factor w/ 6 levels "No answer","Never married",..: 2 4 5 2 4 6 2 4 6 6 ...
 $ age    : int [1:21483] 26 48 67 39 25 25 36 44 44 47 ...
 $ race   : Factor w/ 4 levels "Other","Black",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ rincome: Factor w/ 16 levels "No answer","Don't know",..: 8 8 16 16 16 5 4 9 4 4 ...
 $ partyid: Factor w/ 10 levels "No answer","Don't know",..: 6 5 7 6 9 10 5 8 9 4 ...
 $ relig  : Factor w/ 16 levels "No answer","Don't know",..: 15 15 15 6 12 15 5 15 15 15 ...
 $ denom  : Factor w/ 30 levels "No answer","Don't know",..: 25 23 3 30 30 25 30 15 4 25 ...
 $ tvhours: int [1:21483] 12 NA 2 4 1 NA 3 NA 0 3 ...
```

Check factors



-
  ```
  gss_cat %>%
    count(race)
  ```

-
  ```
  ggplot(gss_cat, aes(race)) +
    geom_bar()
  ```
-



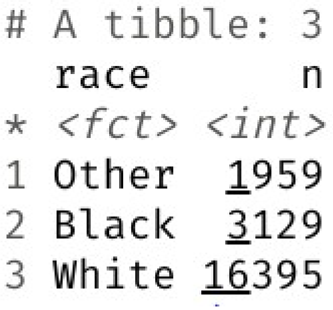
```
# A tibble: 3 x 2
  race        n
* <fct> <int>
1 Other  1959
2 Black  3129
3 White 16395
```

Check factors

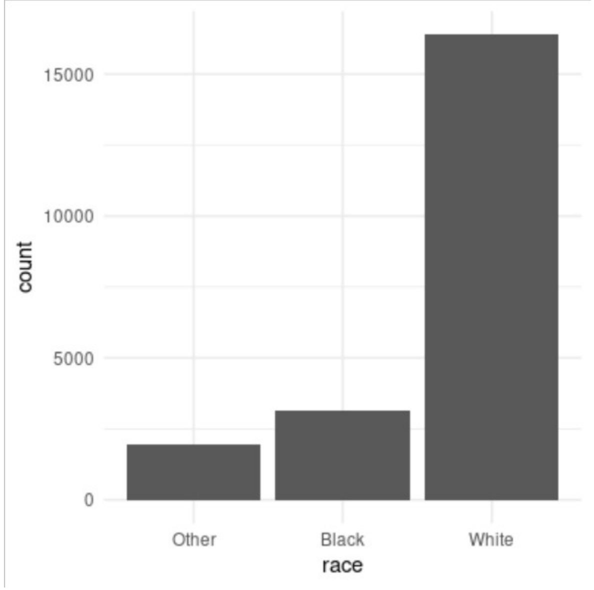
-
  ```
  gss_cat %>%
    count(race)
  ```

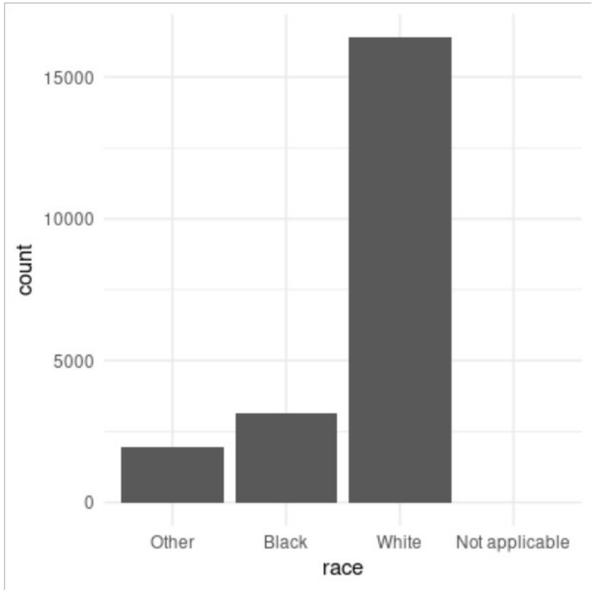
-
  ```
  ggplot(gss_cat, aes(race)) +
    geom_bar()
  ```

-
  ```
  ggplot(gss_cat, aes(race)) +
    geom_bar() +
    scale_x_discrete(drop = FALSE)
  ```



```
# A tibble: 3 x 2
  race      n
* <fct> <int>
1 Other  1959
2 Black  3129
3 White 16395
```





Try this

- Explore the distribution of `rincome` (reported income). What makes the default bar chart hard to understand? How could you improve the plot?
- What is the most common `relig` in this survey? What's the most common `partyid`?

Factor order

- Average hours watching TV per religion
- Hint: use `group_by` & `summarise`

Factor order

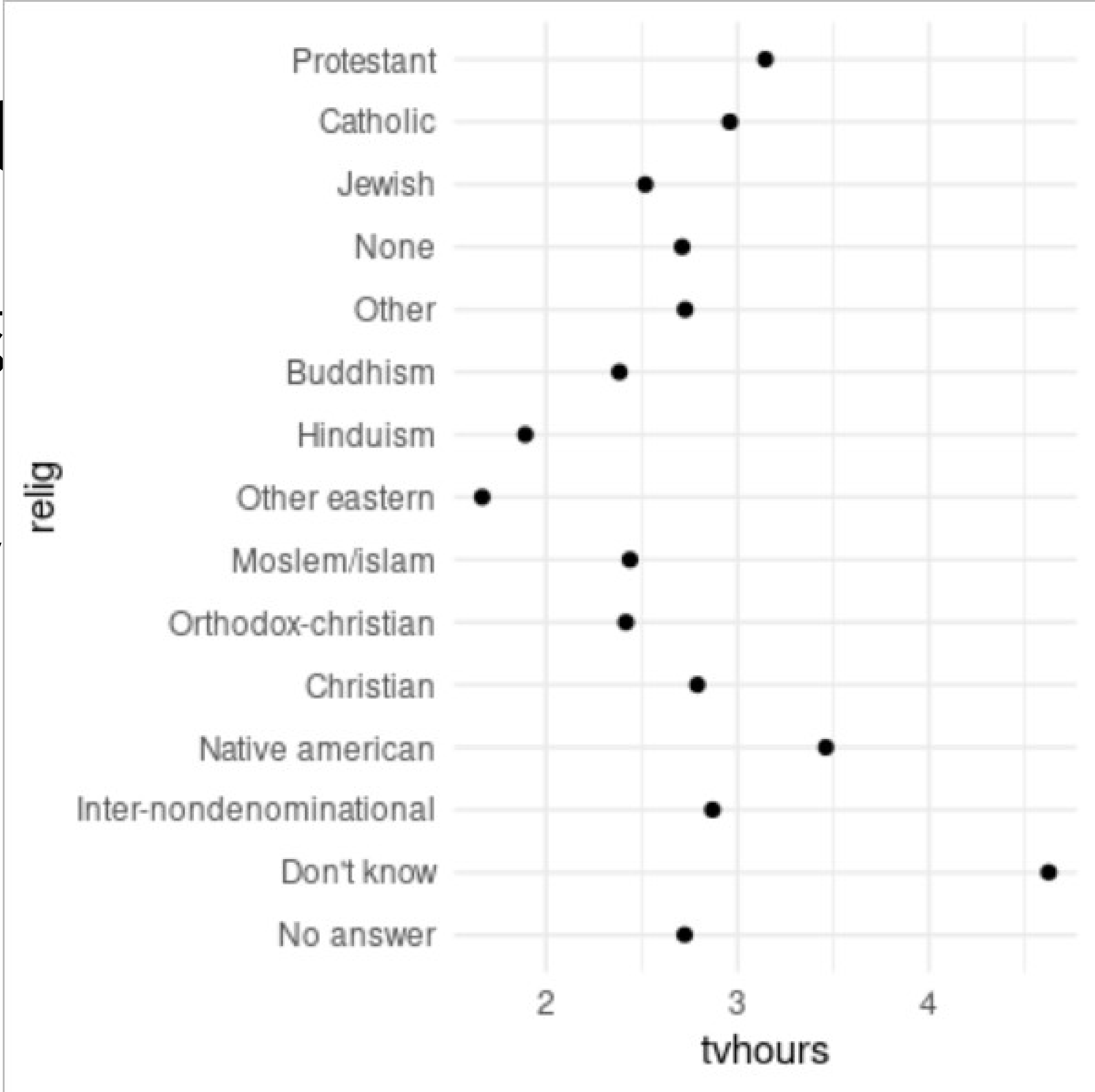
-
  ```
  relig_summary ← gss_cat %>%
    group_by(relig) %>%
    summarise(
      age = mean(age, na.rm = TRUE),
      tvhours = mean(tvhours, na.rm = TRUE),
      n = n()
    )
  ```

-
  ```
  ggplot(relig_summary, aes(tvhours, relig)) +
    geom_point()
  ```

Fact

-
  ```
  relig_summary ← gss_
    group_by(relig) %>%
    summarise(
      age = mean(age, n
      tvhours = mean(tv
      n = n()
    )
  ```

-
  ```
  ggplot(relig_summary,
    geom_point()
  ```



Factor reorder

- `fct_reorder(f, x)`
- Defaults to median, can specify functions
  - Reorder `f` according to `x`

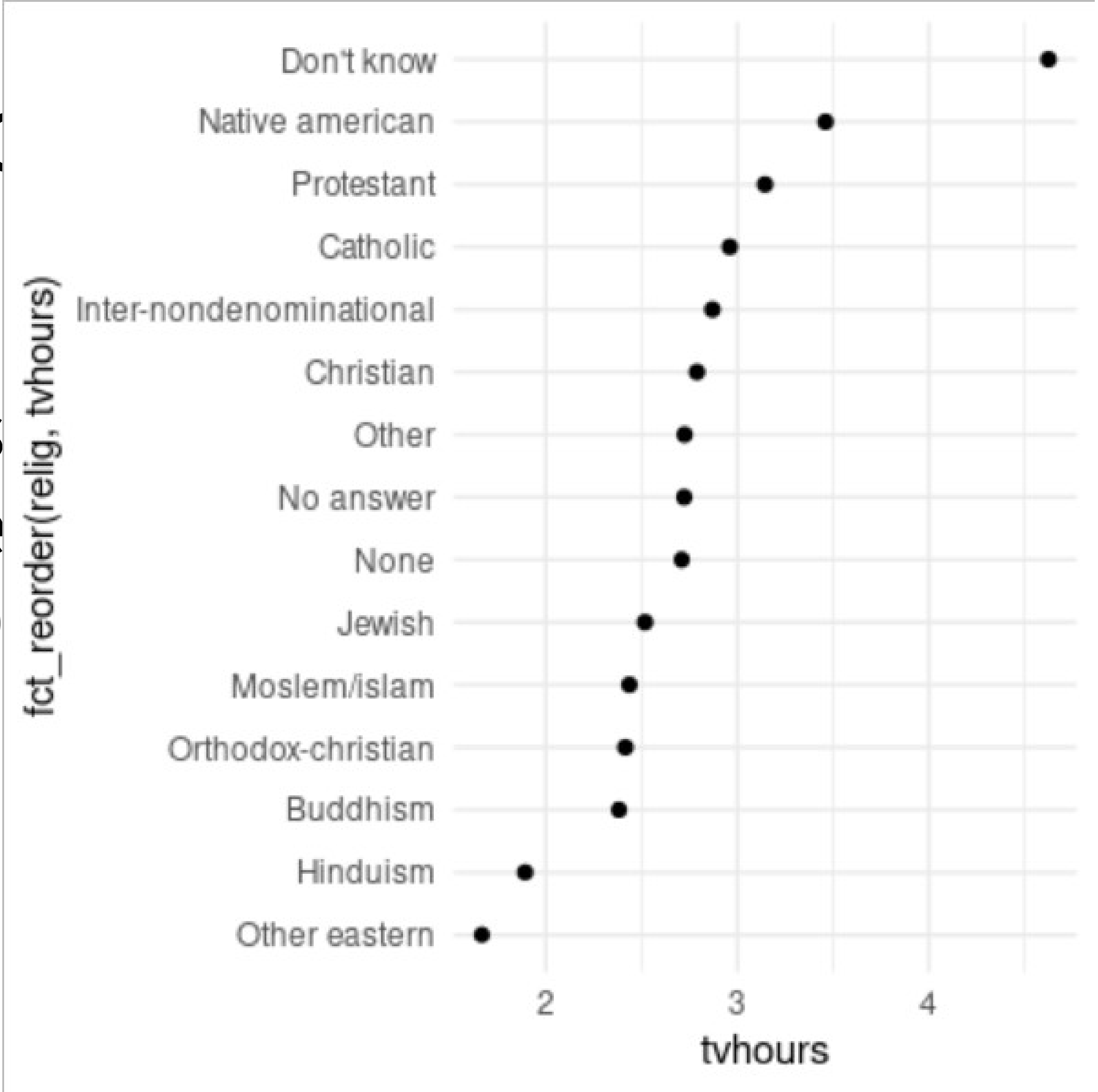
Factor reorder

- `fct_reorder(f, x)`
- Defaults to median, can specify functions

-
  ```
  ggplot(relig_summary, aes(tvhours,
  fct_reorder(relig, tvhours))) +
    geom_point()
  ```
- Better to do it with `mutate` or outside of `ggplot`

Facto

- `fct_reorder(f,`
- `ggplot(relig_s`
  `fct_reorder(re`
  `  geom_point()`



Factor reorder

- Income vs age?

-
  ```
  rincome_summary ← gss_cat %>%
    group_by(rincome) %>%
    summarise(
      age = mean(age, na.rm = TRUE),
      tvhours = mean(tvhours, na.rm = TRUE),
      n = n()
    )
  ```

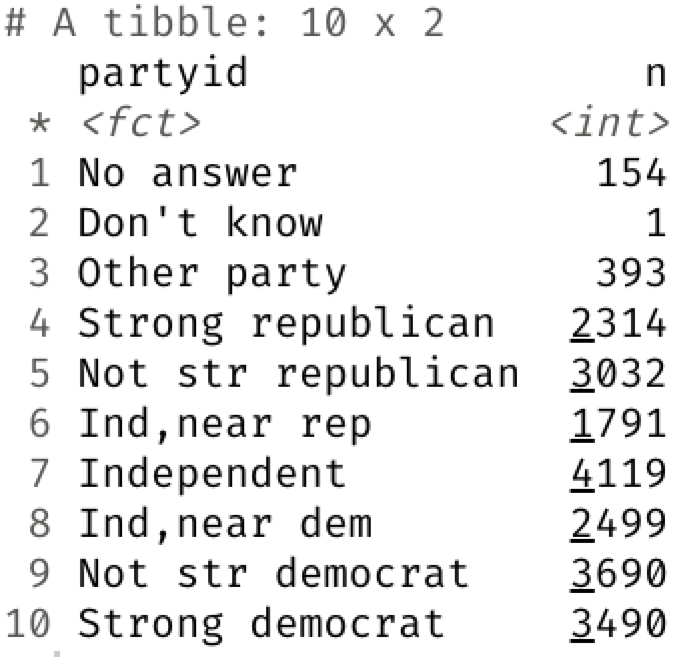
-
  ```
  ggplot(rincome_summary, aes(age, fct_reorder(rincome, age))) +
  geom_point()
  ```
- Does it make any sense?

Other reorder

- `fct_reorder2`: orders by y with largest x (2D)
- `fct_infreq`: orders by frequency
-
  ```
  gss_cat %>%
    mutate(marital = marital %>%
           fct_infreq() %>% fct_rev()) %>%
    ggplot(aes(marital)) +
      geom_bar()
  ```

Modifying levels

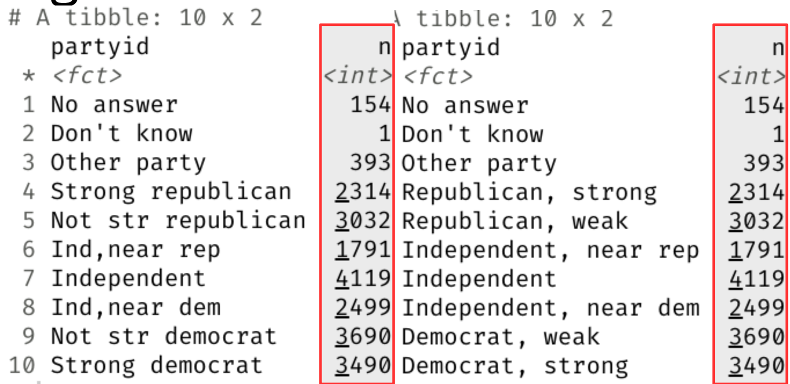
- `fct_recode`: change (recode) values of levels
- `gss_cat %>% count(partyid)`



```
# A tibble: 10 x 2
   partyid                n
 * <fct>              <int>
 1 No answer            154
 2 Don't know             1
 3 Other party          393
 4 Strong republican   2314
 5 Not str republican  3032
 6 Ind,near rep        1791
 7 Independent         4119
 8 Ind,near dem        2499
 9 Not str democrat    3690
10 Strong democrat     3490
```

Modifying levels

-
  ```
  gss_cat %>%
    mutate(partyid = fct_recode(partyid,
      "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"
    )) %>%
    count(partyid)
  ```



```
# A tibble: 10 x 2
   partyid                n
 * <fct>              <int>
 1 No answer            154
 2 Don't know             1
 3 Other party          393
 4 Strong republican   2314
 5 Not str republican  3032
 6 Ind,near rep        1791
 7 Independent         4119
 8 Ind,near dem        2499
 9 Not str democrat    3690
10 Strong democrat     3490
```

```
\ tibble: 10 x 2
partyid                   n
<fct>                 <int>
No answer               154
Don't know                1
Other party             393
Republican, strong     2314
Republican, weak       3032
Independent, near rep  1791
Independent            4119
Independent, near dem  2499
Democrat, weak         3690
Democrat, strong       3490
```

Collapsing levels

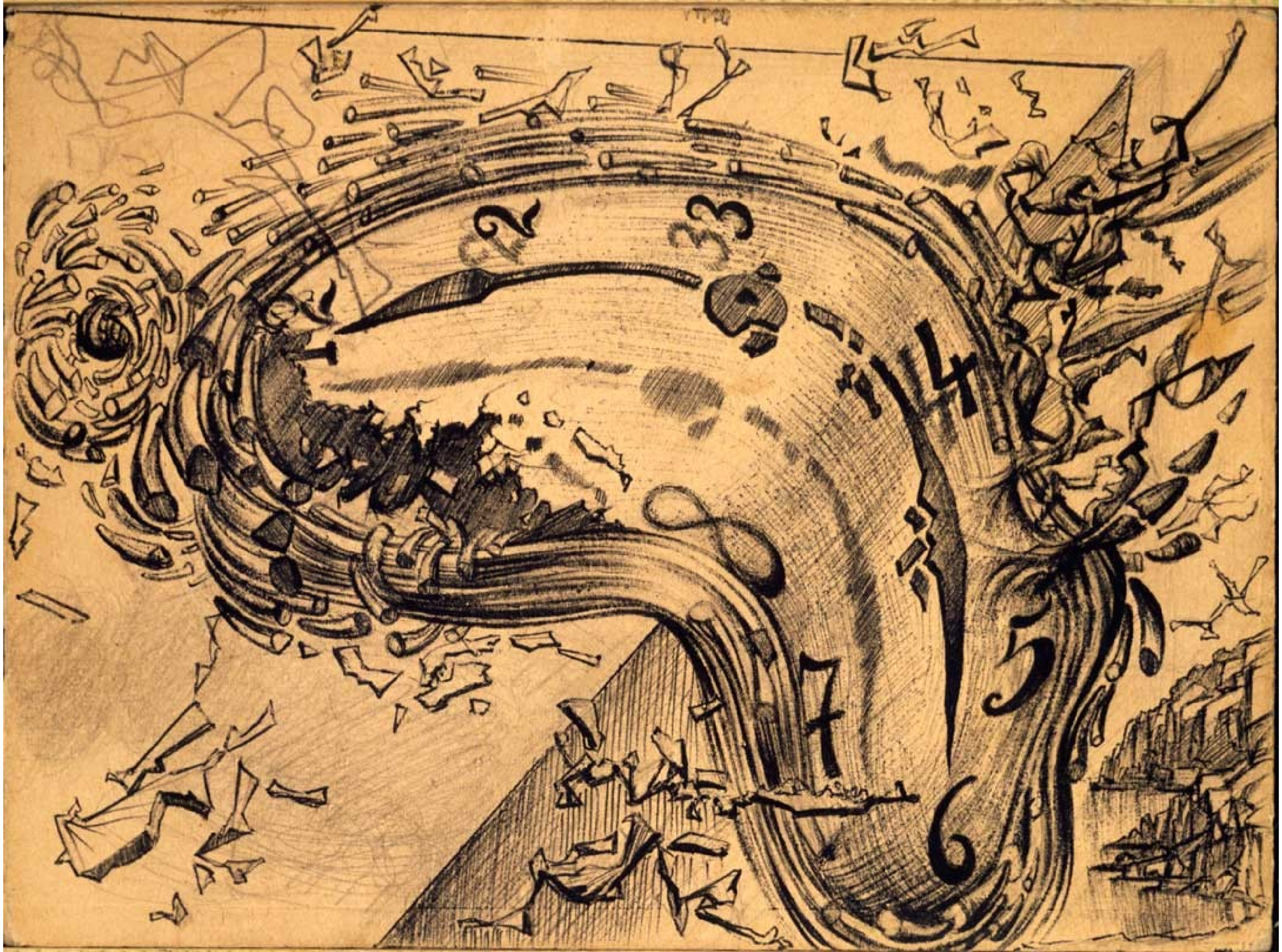
-
  ```
  gss_cat %>%
    mutate(partyid = fct_collapse(partyid,
      other = c("No answer", "Don't know", "Other party"),
      rep = c("Strong republican", "Not str republican"),
      ind = c("Ind,near rep", "Independent", "Ind,near dem"),
      dem = c("Not str democrat", "Strong democrat")
    )) %>%
    count(partyid)
  ```
- See also `?fct_lump`

When does/did time begin?
When is t=0?

Disney · PIXAR
FORKY ASKS A QUESTION:
WHAT IS
TIME?
12
3
6
9

When does/did time begin?
When is t=0?

Open spreadsheet, type 0, change format to YYYY-MM-DD hh:mm:ss



2
3
4
5
6
7

lubridate

- Friendly with `tidyverse`
- `ts`, `xts` & `zoo` for time series
  - time + matrix
  - Common for predictions, some models

How is time counted

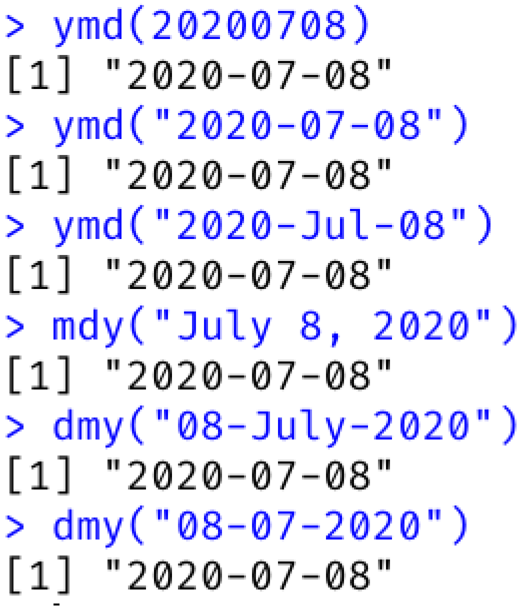
- Excel: days since 1900-01-01 or 1904-01-01
- POSIX: seconds since 1970-01-01 UTC
  - POSIXct: (calendar/continuous time) seconds
  - POSIXlt: (local time) list with elements
- What gets exported to csv?

Types

- date
- time
- date-time <dttm>: POSIXct
- Who types in date-time format in spreadsheets?

Construct Dates

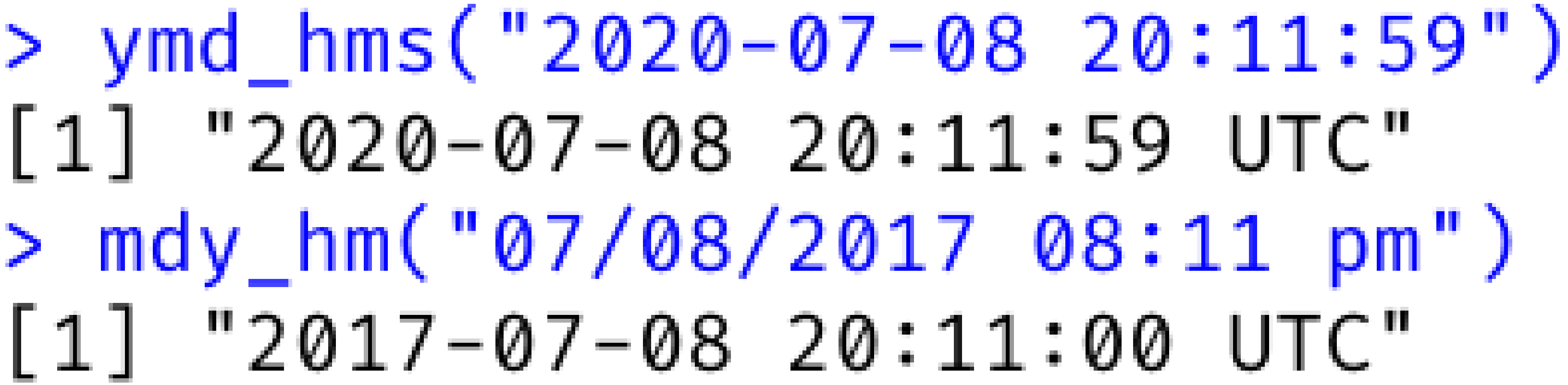
- today() & now()
- Helper functions for numbers & strings:
  - ymd(20200708)
    ymd("2020-07-08")
    ymd("2020-Jul-08")
  - mdy("July 8, 2020")
  - dmy("08-July-2020")
    dmy("08-07-2020")



```
> ymd(20200708)
[1] "2020-07-08"
> ymd("2020-07-08")
[1] "2020-07-08"
> ymd("2020-Jul-08")
[1] "2020-07-08"
> mdy("July 8, 2020")
[1] "2020-07-08"
> dmy("08-July-2020")
[1] "2020-07-08"
> dmy("08-07-2020")
[1] "2020-07-08"
```

Construct Dates & Times

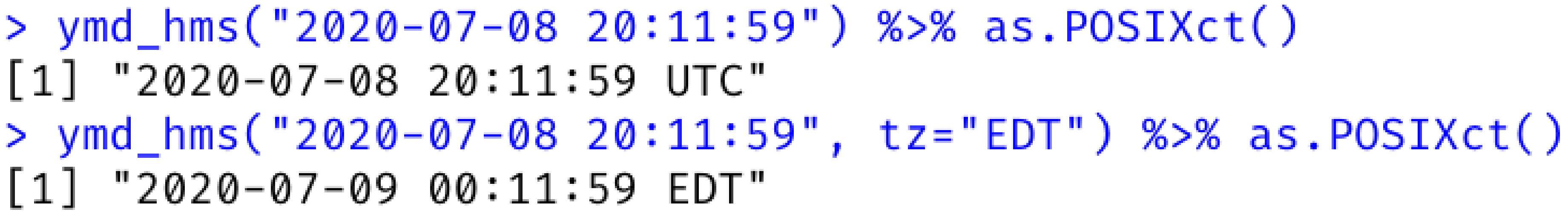
- `ymd_hms("2020-07-08 20:11:59")`
- `mdy_hm("07/08/2017 08:01 pm")`
- Converts to UTC internally



```
> ymd_hms("2020-07-08 20:11:59")
[1] "2020-07-08 20:11:59 UTC"
> mdy_hm("07/08/2017 08:11 pm")
[1] "2017-07-08 20:11:00 UTC"
```

Construct Dates & Times

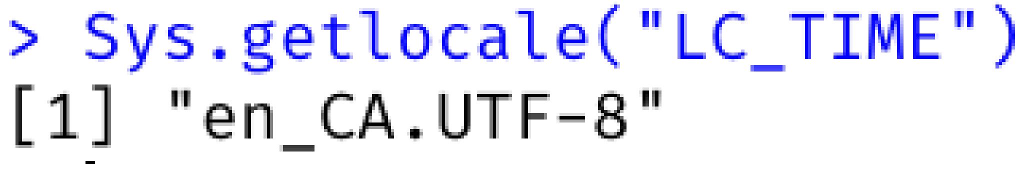
- Converts to UTC internally
- `ymd_hms("2020-07-08 20:11:59") %>% as.POSIXct()`
- `ymd_hms("2020-07-08 20:11:59", tz="EDT") %>% as.POSIXct()`



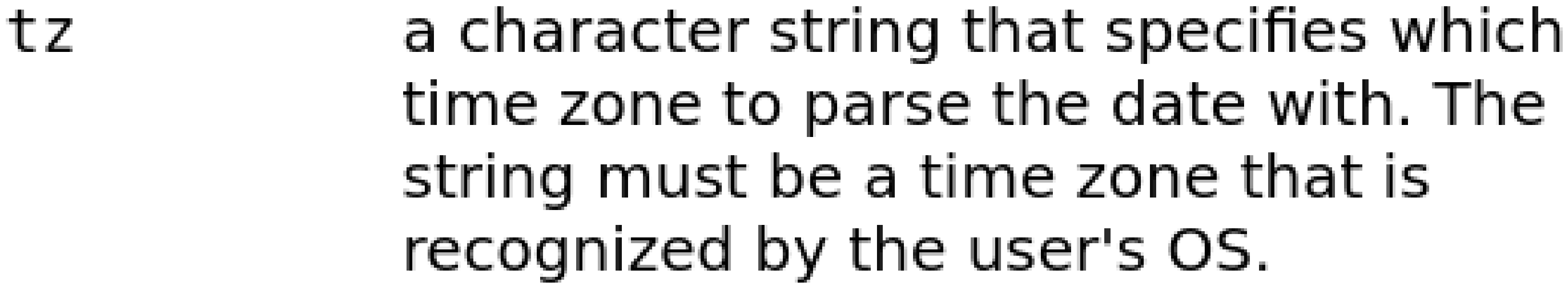
```
> ymd_hms("2020-07-08 20:11:59") %>% as.POSIXct()
[1] "2020-07-08 20:11:59 UTC"
> ymd_hms("2020-07-08 20:11:59", tz="EDT") %>% as.POSIXct()
[1] "2020-07-09 00:11:59 EDT"
```

Time Zones & locales

- `Sys.getlocale("LC_TIME")`



```
> Sys.getlocale("LC_TIME")
[1] "en_CA.UTF-8"
```

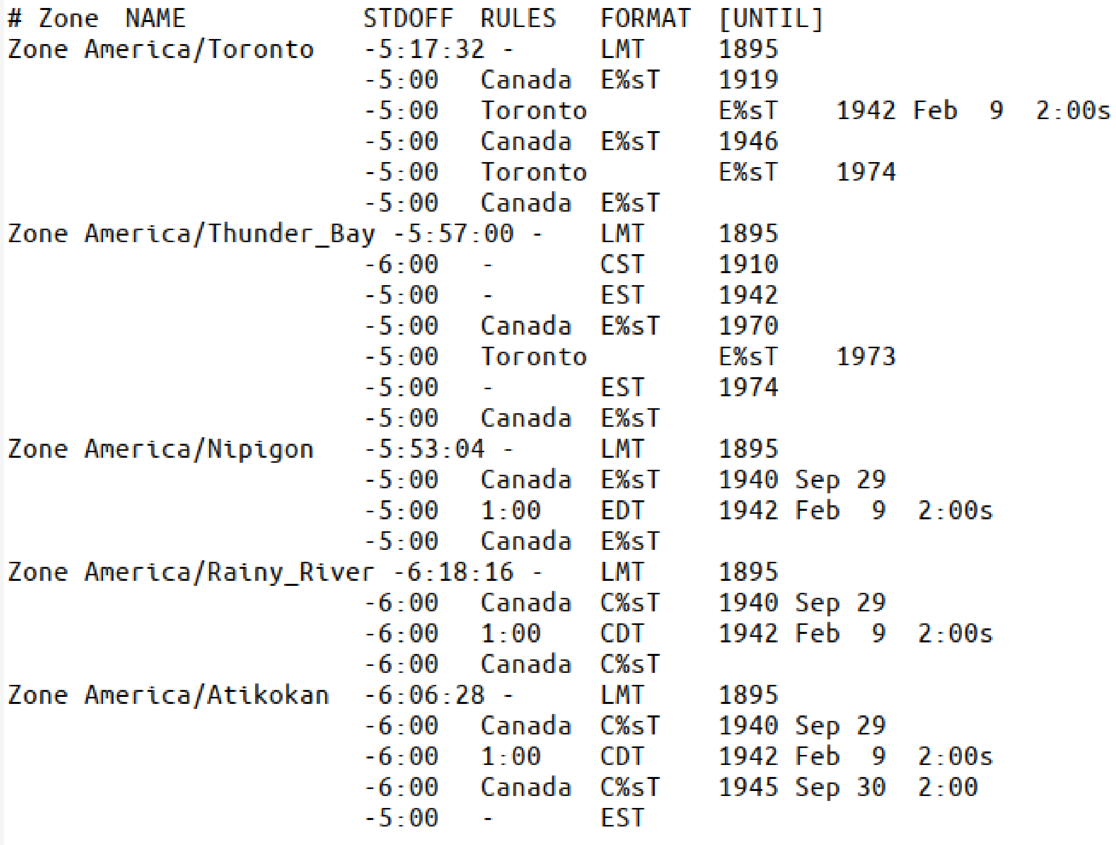


tz	a character string that specifies which time zone to parse the date with. The string must be a time zone that is recognized by the user's OS.

Another problem is that name needs to reflect not only to the current behaviour, but also the *complete history*. For example, there are time zones for both “America/New_York” and “America/Detroit”. These cities both currently use Eastern Standard Time but in *1969-1972 Michigan (the state in which Detroit is located), did not follow DST*, so it needs a different name.

Time Zones

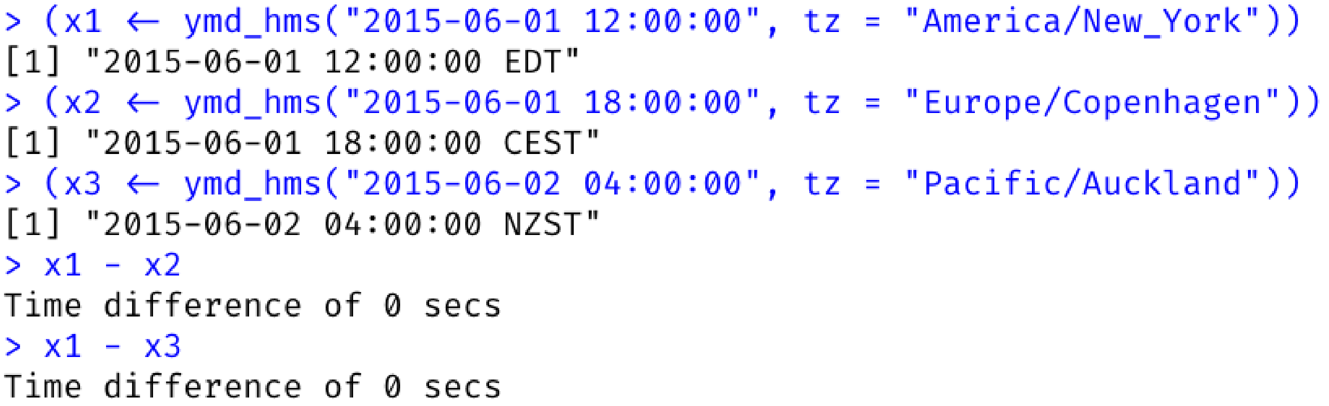
- Complicated
- IANA time zones: <continent>/<city>
  - "Canada/Eastern"
  - "America/Toronto"
  - "America/Nipigon"
  - "America/Thunder_Bay"
- `Sys.timezone()`
- `OlsonNames()`



```
# Zone  NAME            STDOFF  RULES   FORMAT  [UNTIL]
Zone America/Toronto    -5:17:32 -      LMT     1895
                        -5:00   Canada  E%sT    1919
                        -5:00   Toronto         E%sT    1942 Feb  9  2:00s
                        -5:00   Canada  E%sT    1946
                        -5:00   Toronto         E%sT    1974
                        -5:00   Canada  E%sT
Zone America/Thunder_Bay -5:57:00 -     LMT     1895
                        -6:00   -       CST     1910
                        -5:00   -       EST     1942
                        -5:00   Canada  E%sT    1970
                        -5:00   Toronto         E%sT    1973
                        -5:00   -       EST     1974
                        -5:00   Canada  E%sT
Zone America/Nipigon    -5:53:04 -      LMT     1895
                        -5:00   Canada  E%sT    1940 Sep 29
                        -5:00   1:00    EDT     1942 Feb  9  2:00s
                        -5:00   Canada  E%sT
Zone America/Rainy_River -6:18:16 -     LMT     1895
                        -6:00   Canada  C%sT    1940 Sep 29
                        -6:00   1:00    CDT     1942 Feb  9  2:00s
                        -6:00   Canada  C%sT
Zone America/Atikokan   -6:06:28 -      LMT     1895
                        -6:00   Canada  C%sT    1940 Sep 29
                        -6:00   1:00    CDT     1942 Feb  9  2:00s
                        -6:00   Canada  C%sT    1945 Sep 30  2:00
                        -5:00   -       EST
```

Time Zones Control Printing

- (x1 ← ymd_hms("2015-06-01 12:00:00", tz = "America/New_York"))
- (x2 ← ymd_hms("2015-06-01 18:00:00", tz = "Europe/Copenhagen"))
- (x3 ← ymd_hms("2015-06-02 04:00:00", tz = "Pacific/Auckland"))
- x1 - x2
- x1 - x3



```
> (x1 ← ymd_hms("2015-06-01 12:00:00", tz = "America/New_York"))
[1] "2015-06-01 12:00:00 EDT"
> (x2 ← ymd_hms("2015-06-01 18:00:00", tz = "Europe/Copenhagen"))
[1] "2015-06-01 18:00:00 CEST"
> (x3 ← ymd_hms("2015-06-02 04:00:00", tz = "Pacific/Auckland"))
[1] "2015-06-02 04:00:00 NZST"
> x1 - x2
Time difference of 0 secs
> x1 - x3
Time difference of 0 secs
```