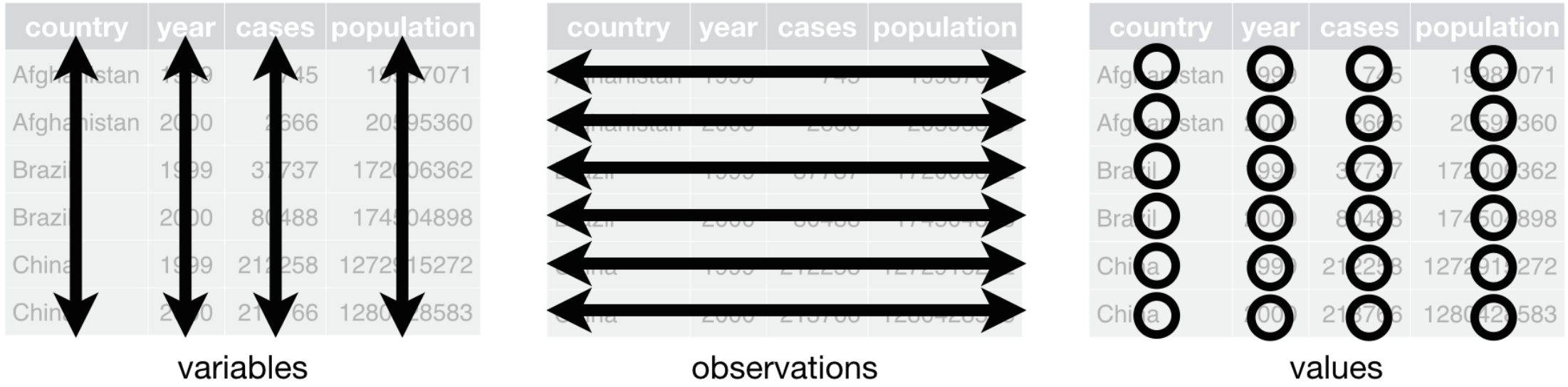
Tidy Data and Dates

Spring 23

April 11 2023

What are tidy data?

1. Each variable forms a column
2. Each observation forms a row
3. Each value has its own cell



Untidy data: example 1

```
untidy_data <- tibble(
  name = c("Ana","Bob","Cara"),
  wt_07_01_2021 = c(100, 150, 140),
  wt_08_01_2021 = c(104, 155, 138),
  wt_09_01_2021 = c(NA, 160, 142)
)
```

```
# A tibble: 3 × 4
  name  wt_07_01_2021 wt_08_01_2021 wt_09_01_2021
  <chr>         <dbl>         <dbl>         <dbl>
1 Ana             100           104            NA
2 Bob             150           155           160
3 Cara            140           138           142
```

Tidy data: example 1

```
library(lubridate)  # for date manipulations
library(stringr) # for string manipulations
untidy_data %>%
  pivot_longer(names_to = "date",
               values_to = "weight",
               cols = -name) %>%
  mutate(date = stringr::str_remove(date,"wt_"),
         date = lubridate::dmy(date))
# A tibble: 9 × 3
  name  date       weight
  <chr> <date>        <dbl>
1 Ana   2021-01-07    100
2 Ana   2021-01-08    104
3 Ana   2021-01-09     NA
4 Bob   2021-01-07    150
5 Bob   2021-01-08    155
6 Bob   2021-01-09    160
7 Cara  2021-01-07    140
8 Cara  2021-01-08    138
9 Cara  2021-01-09    142
```

Wide vs. long data

- **Wide** data has one row per subject, with multiple columns for their repeated measurements
- **Long** data has multiple rows per subject, with one column for the measurement variable and another indicating from when/where the repeated measures are from

id	SBP_visit1	SBP_visit2	SBP_visit3
a	130	110	112
b	120	116	122
c	130	136	138
d	119	106	118

wide

id	visit	SBP
a	1	130
b	1	120
c	1	130
d	1	119
a	2	110
b	2	116
c	2	136
d	2	106
a	3	112
b	3	122
c	3	138
d	3	118

long

Wide to long: `pivot_longer()`

```
BP_wide
# A tibble: 4 × 5
  id    sex   SBP_v1 SBP_v2 SBP_v3
  <chr> <chr>  <dbl>  <dbl>  <dbl>
1 a     F        130    110    112
2 b     M        120    116    122
3 c     M        130    136    138
4 d     F        119    106    118
```

```
BP_long <- BP_wide %>%
  pivot_longer(names_to = "visit",
               values_to = "SBP",
               cols = SBP_v1:SBP_v3)
BP_long
# A tibble: 12 × 4
   id    sex   visit    SBP
   <chr> <chr> <chr>  <dbl>
 1 a     F     SBP_v1   130
 2 a     F     SBP_v2   110
 3 a     F     SBP_v3   112
 4 b     M     SBP_v1   120
 5 b     M     SBP_v2   116
 6 b     M     SBP_v3   122
 7 c     M     SBP_v1   130
 8 c     M     SBP_v2   136
 9 c     M     SBP_v3   138
10 d     F     SBP_v1   119
11 d     F     SBP_v2   106
12 d     F     SBP_v3   118
```

Wide to long: `pivot_longer()`

`pivot_longer` lengthens data, increasing the number of rows and decreasing the number of columns.

Need to specify:

- ***new column names***
  - ***`names_to`: stores row names of wide data's columns***
  - ***`values_to`: stores data values***
- ***which columns to pivot***

Long to wide: `pivot_wider()`

```
BP_long
# A tibble: 12 × 4
   id    sex   visit    SBP
   <chr> <chr> <chr>  <dbl>
 1 a     F     SBP_v1   130
 2 a     F     SBP_v2   110
 3 a     F     SBP_v3   112
 4 b     M     SBP_v1   120
 5 b     M     SBP_v2   116
 6 b     M     SBP_v3   122
 7 c     M     SBP_v1   130
 8 c     M     SBP_v2   136
 9 c     M     SBP_v3   138
10 d     F     SBP_v1   119
11 d     F     SBP_v2   106
12 d     F     SBP_v3   118
```

```
BP_wide1 <- BP_long %>%
  pivot_wider(names_from = "visit",
              values_from = "SBP")
BP_wide1
# A tibble: 4 × 5
  id    sex   SBP_v1 SBP_v2 SBP_v3
  <chr> <chr>  <dbl>  <dbl>  <dbl>
1 a     F        130    110    112
2 b     M        120    116    122
3 c     M        130    136    138
4 d     F        119    106    118
```

Long to wide: `pivot_wider()`

`pivot_wider` increases number of columns and decreases the number of rows.

Need to specify:

- ***new column names***
  - ***names_from: get the name of the column***
  - ***values_from: get the cell values from***

Separate Info

```
BP_long
# A tibble: 12 × 4
   id    sex   visit    SBP
   <chr> <chr> <chr>  <dbl>
 1 a     F     SBP_v1   130
 2 a     F     SBP_v2   110
 3 a     F     SBP_v3   112
 4 b     M     SBP_v1   120
 5 b     M     SBP_v2   116
 6 b     M     SBP_v3   122
 7 c     M     SBP_v1   130
 8 c     M     SBP_v2   136
 9 c     M     SBP_v3   138
10 d     F     SBP_v1   119
11 d     F     SBP_v2   106
12 d     F     SBP_v3   118
```

```
BP_long1 <- BP_long %>%
  separate(visit, c("acrnym", "visit"))
BP_long1
# A tibble: 12 × 5
   id    sex   acrnym visit   SBP
   <chr> <chr> <chr>  <chr> <dbl>
 1 a     F     SBP    v1      130
 2 a     F     SBP    v2      110
 3 a     F     SBP    v3      112
 4 b     M     SBP    v1      120
 5 b     M     SBP    v2      116
 6 b     M     SBP    v3      122
 7 c     M     SBP    v1      130
 8 c     M     SBP    v2      136
 9 c     M     SBP    v3      138
10 d     F     SBP    v1      119
11 d     F     SBP    v2      106
12 d     F     SBP    v3      118
```

readr::parse_number()

Goal: Extract first number from the visit column and remove the characters

```
BP_long
# A tibble: 12 × 4
   id    sex   visit    SBP
   <chr> <chr> <chr>  <dbl>
 1 a     F     SBP_v1   130
 2 a     F     SBP_v2   110
 3 a     F     SBP_v3   112
 4 b     M     SBP_v1   120
 5 b     M     SBP_v2   116
 6 b     M     SBP_v3   122
 7 c     M     SBP_v1   130
 8 c     M     SBP_v2   136
 9 c     M     SBP_v3   138
10 d     F     SBP_v1   119
11 d     F     SBP_v2   106
12 d     F     SBP_v3   118
```

```
BP_long2 <- BP_long %>%
  mutate(visit = parse_number(visit))
BP_long2
# A tibble: 12 × 4
   id    sex   visit   SBP
   <chr> <chr> <dbl> <dbl>
 1 a     F         1   130
 2 a     F         2   110
 3 a     F         3   112
 4 b     M         1   120
 5 b     M         2   116
 6 b     M         3   122
 7 c     M         1   130
 8 c     M         2   136
 9 c     M         3   138
10 d     F         1   119
11 d     F         2   106
12 d     F         3   118
```

Make cleaned-up long data wide

```
head(BP_long2, 4)
# A tibble: 4 × 4
  id    sex   visit   SBP
  <chr> <chr> <dbl> <dbl>
1 a     F         1   130
2 a     F         2   110
3 a     F         3   112
4 b     M         1   120
```

Problem: have numbers as column names

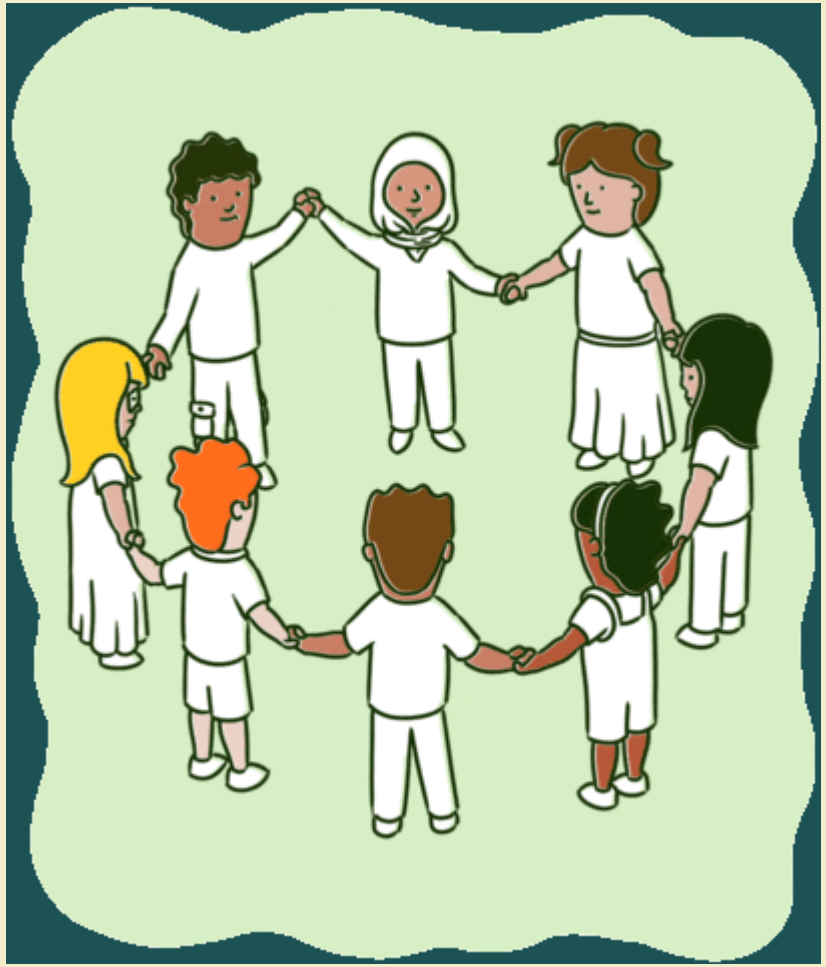
Solution: have row names start with the key column's name separated by a character

```
BP_wide2 <- BP_long2 %>%
  pivot_wider(names_from = "visit",
              values_from = "SBP")
BP_wide2
# A tibble: 4 × 5
  id    sex     `1`   `2`   `3`
  <chr> <chr> <dbl> <dbl> <dbl>
1 a     F       130   110   112
2 b     M       120   116   122
3 c     M       130   136   138
4 d     F       119   106   118
```

```
BP_wide3 <- BP_long2 %>%
  pivot_wider(names_from = "visit",
              values_from = "SBP",
              names_prefix = "value_")
BP_wide3
# A tibble: 4 × 5
  id    sex   value_1 value_2 value_3
  <chr> <chr>   <dbl>   <dbl>   <dbl>
1 a     F         130     110     112
2 b     M         120     116     122
3 c     M         130     136     138
4 d     F         119     106     118
```

GROUP ACTIVITY 1

10:00



- *Let's go over to maize server/ local Rstudio and our class moodle*
- *Get the class activity 8.Rmd file*
- *Work on your turn 1*

Dates with `lubridate`

- *Convert characters to special "Date" type*
- *Easy date magic examples:*
  - *add and subtract dates*
  - *convert to minutes/years/etc*
  - *change timezones*
  - *add 1 month to a date...*

Allison Horst

What kind of date do you have?

PARSE DATE-TIMES (Convert strings or numbers to date-times)

1. Identify the order of the year (**y**), month (**m**), day (**d**), hour (**h**), minute (**m**) and second (**s**) elements in your data.
2. Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00	**ymd_hms**(), **ymd_hm**(), **ymd_h**(). *ymd_hms("2017-11-28T14:02:00")*
2017-22-12 10:00:00	**ydm_hms**(), **ydm_hm**(), **ydm_h**(). *ydm_hms("2017-22-12 10:00:00")*
11/28/2017 1:02:03	**mdy_hms**(), **mdy_hm**(), **mdy_h**(). *mdy_hms("11/28/2017 1:02:03")*
1 Jan 2017 23:59:59	**dmy_hms**(), **dmy_hm**(), **dmy_h**(). *dmy_hms("1 Jan 2017 23:59:59")*
20170131	**ymd**(), **ydm**(). *ymd(20170131)*
July 4th, 2000	**mdy**(), **myd**(). *mdy("July 4th, 2000")*
4th of July '99	**dmy**(), **dym**(). *dmy("4th of July '99")*
2001: Q3	**yq**() Q for quarter. *yq("2001: Q3")*
2:01	hms::**hms**() Also lubridate::**hms**(), **hm**() and **ms**(), which return periods.* *hms::hms(sec = 0, min= 1, hours = 2)*

Parsing Complex Dates

```
complex_dates <- tibble(
  name = c("Yi", "Mo", "Dee"),
  dob = c("31-October-1952 14:30:15",
          "12-Jan-1984 22:15:00",
          "02-Feb-2002 10:45:30")
)

complex_dates %>%
  mutate(
    dob_date = dmy_hms(dob),
  ) %>% knitr::kable()
```

name	dob	dob_date
Yi	31-October-1952 14:30:15	1952-10-31 14:30:15
Mo	12-Jan-1984 22:15:00	1984-01-12 22:15:00
Dee	02-Feb-2002 10:45:30	2002-02-02 10:45:30

Advanced Date Arithmetic

```
complex_dates %>%
  mutate(
    dob_date = dmy_hms(dob),
    dob_year = year(dob_date),
    time_since_birth = interval(dob_date, now()),
    age_exact = as.duration(time_since_birth) / dyears(1),
  ) %>% knitr::kable()
```

name	dob	dob_date	dob_year	time_since_birth	age_exact
Yi	31-October-1952 14:30:15	1952-10-31 14:30:15	1952	1952-10-31 14:30:15 UTC--2023-04-12 03:27:52 UTC	70.44364
Mo	12-Jan-1984 22:15:00	1984-01-12 22:15:00	1984	1984-01-12 22:15:00 UTC--2023-04-12 03:27:52 UTC	39.24495
Dee	02-Feb-2002 10:45:30	2002-02-02 10:45:30	2002	2002-02-02 10:45:30 UTC--2023-04-12 03:27:52 UTC	21.18740

Creating Date-Time Objects with `make_datetime()`

```
# Components of date and time
year_component <- 2023
month_component <- 3
day_component <- 23
hour_component <- 18
minute_component <- 45
second_component <- 30
```

```
# Using make_datetime() to create a date-time object
date_time_object <- make_datetime(
  year = year_component,
  month = month_component,
  day = day_component,
  hour = hour_component,
  min = minute_component,
  sec = second_component
)
```

```
date_time_object %>% knitr::kable()
```

x
2023-03-23 18:45:30

Combining Date and Time Objects

```
date_only <- ymd("2023-03-23")
time_only <- hms("18:45:30")

date_time_combined <- make_datetime(
  year = year(date_only),
  month = month(date_only),
  day = day(date_only),
  hour = hour(time_only),
  min = minute(time_only),
  sec = second(time_only)
)

date_time_combined %>% knitr::kable()
```

x
2023-03-23 18:45:30

Extracting Components from Date-Time Objects

```
date_components <- tibble(
  date_time = ymd_hms("2023-03-23 18:45:30")
)

date_components %>%
  mutate(
    year = year(date_time),
    month = month(date_time, label = TRUE),
    day = day(date_time),
    hour = hour(date_time),
    minute = minute(date_time),
    second = second(date_time)
  ) %>% knitr::kable()
```

date_time	year	month	day	hour	minute	second
2023-03-23 18:45:30	2023	Mar	23	18	45	30

Advanced Date-Time Arithmetic with Periods

```
date_periods <- tibble(
  date = ymd("2023-01-31")
)

date_periods %>%
  mutate(
    add_month = date + month(1),
    add_year = date + years(1),
    subtract_week = date - weeks(1)
  ) %>% knitr::kable()
```

date	add_month	add_year	subtract_week
2023-01-31	2023-02-01	2024-01-31	2023-01-24

Duration and Time Differences

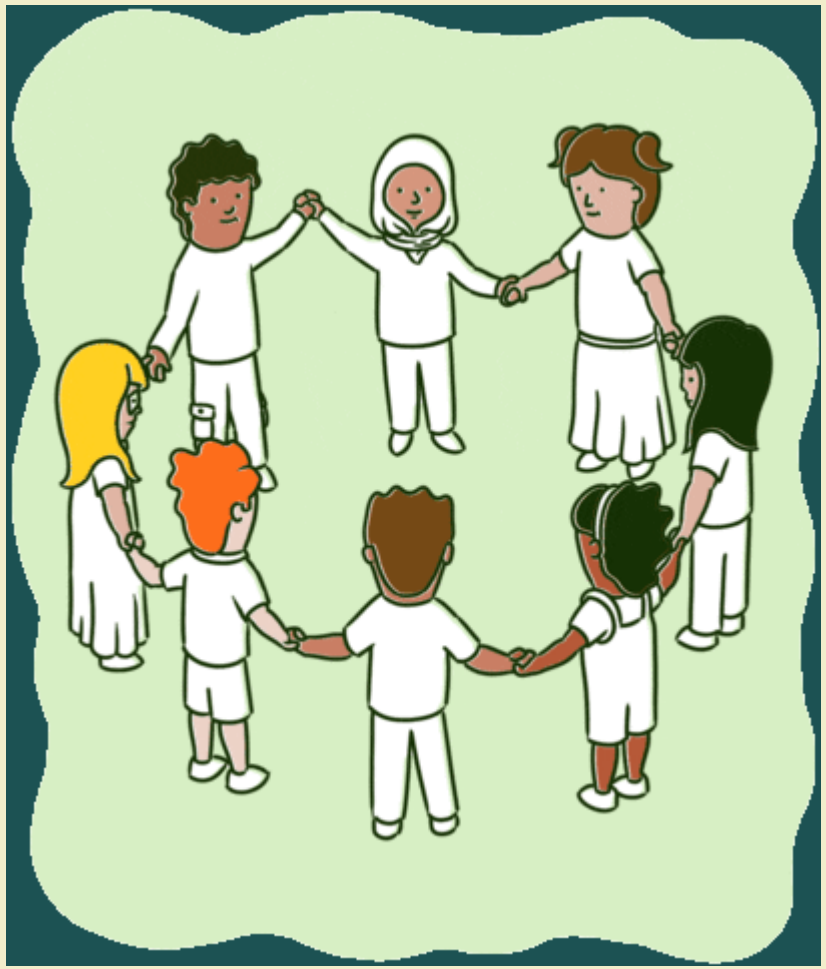
```
time_diff_example <- tibble(
  start_date = ymd("2023-01-01"),
  end_date = ymd("2023-12-31")
)

time_diff_example %>%
  mutate(
    time_diff = end_date - start_date,
    duration_days = as.duration(time_diff),
    duration_weeks = duration_days / dweeks(1),
    duration_months = duration_days / dmonths(1),
    duration_years = duration_days / dyears(1)
  )  %>% knitr::kable()
```

start_date	end_date	time_diff	duration_days	duration_weeks	duration_months	duration_years
2023-01-01	2023-12-31	364 days	31449600s (~52 weeks)	52	11.95893	0.9965777

GROUP ACTIVITY 2

10:00



- *Work on your turn 2*
- *Ask me questions*