

ID 543: Day 3

Introduction to R

Homework review

Today's goals

- Understand how to use `summarize()` and `group_by()` to get summary statistics
- Learn how to read in data from different file types
- Understand how to use the `here` package to refer to files
- Learn some tools for dealing with missing data
- Learn how to join datasets using `left_join()`, `right_join()`, `full_join()`, and `inner_join()`

Summary statistics

We can get certain summary statistics about our data with `summary()`, which we can use either on an entire dataframe or on a single variable

```
1 nlsy_sleep <- nlsy |>
2   select(id, contains("sleep"), age_bir, sex)
3 summary(nlsy_sleep$age_bir)
```

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
13.00  19.00   22.00   23.45  27.00   52.00
```

```
1 summary(nlsy_sleep)
```

```
      id      sleep_wkdy      sleep_wknd      age_bir
Min.   :    3  Min.     : 0.000  Min.     : 0.000  Min.     :13.00
1st Qu.: 2317  1st Qu.: 6.000  1st Qu.: 6.000  1st Qu.:19.00
Median : 4744  Median : 7.000  Median : 7.000  Median :22.00
Mean   : 5229  Mean   : 6.643  Mean   : 7.267  Mean   :23.45
3rd Qu.: 7937  3rd Qu.: 8.000  3rd Qu.: 8.000  3rd Qu.:27.00
Max.   :12667  Max.   :13.000  Max.   :14.000  Max.   :52.00

      sex
Min.   :1.000
1st Qu.:1.000
```

Summary statistics

We can also apply certain functions to a variable(s) to get a single statistic: `mean()`, `median()`, `var()`, `sd()`, `cov()`, `cor()`, `min()`, `max()`, `quantile()`, etc.

```
1 median(nlsy$age_bir)
```

```
[1] 22
```

```
1 cor(nlsy$sleep_wkdy, nlsy$sleep_wknd)
```

```
[1] 0.7101579
```

```
1 quantile(nlsy$income, probs = c(0.1, 0.9))
```

```
10%    90%  
3177.2 33024.0
```

New function: `summarize()`

But what if we want a lot of summary statistics – just not those that come with the `summary()` function?

- For example, it doesn't give us a standard deviation!

We can use `summarize()`

```
1 summarize(nlsy,  
2           sd_age_bir = sd(age_bir),  
3           cor_sleep = cor(sleep_wkdy, sleep_wknd),  
4           ten_pctle_inc = quantile(income, probs = 0.1),  
5           ninety_pctle_inc = quantile(income, probs = 0.9))
```

```
# A tibble: 1 × 4  
  sd_age_bir cor_sleep ten_pctle_inc ninety_pctle_inc  
  <dbl>      <dbl>      <dbl>          <dbl>  
1     5.99    0.710     3177.         33024
```

summarize() specifics

Important to note:

- Takes a dataframe as its first argument. That means we can use pipes!
- Returns a tibble – helpful if you want to use those values in a figure or table.
- Can give the summary statistics names.
- Can ask for any type of function of the variables (including one you make up yourself).

Combining with other functions

Because we can pipe, we can also look at statistics of variables that we make using `mutate()`, in a dataset we've subsetted with `filter()`.

```
1 nlsy |>
2   mutate(age_bir_stand = (age_bir - mean(age_bir)) / sd(age_bir))
3   filter(sex == 1) |>
4   summarize(mean_men = mean(age_bir_stand))
```

```
# A tibble: 1 × 1
  mean_men
  <dbl>
1    0.283
```

Note

Note that we're standardizing the data *before* filtering. Or else the mean would be 0!

Exercise

What if we want both groups at once?

```
1 nlsy |>
2   filter(sex == 1) |>
3   summarize(age_bir_men = mean(age_bir))
```

```
# A tibble: 1 × 1
  age_bir_men
  <dbl>
1      25.1
```

```
1 nlsy |>
2   filter(sex == 2) |>
3   summarize(age_bir_women = mean(age_bir))
```

```
# A tibble: 1 × 1
  age_bir_women
  <dbl>
1      22.2
```

We can “group” tibbles using `group_by()`

We can tell it’s “grouped” and how many groups there are by printing out the data.

The data itself won’t look (much) different, but we’ll be able to perform grouped functions on it.

```
1 nlsy_by_region <- group_by(nlsy, region)
2 nlsy_by_region
```

```
# A tibble: 1,205 × 15
# Groups:   region [4]
   id glasses eyesight sleep_wkdy sleep_wknd nsibs race_eth sex region
  <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl> <dbl>
1     3     0     1     5     7     3     3     2     1
2     6     1     2     6     7     1     3     1     1
3     8     0     2     7     9     7     3     2     1
4    16     1     3     6     7     3     3     2     1
5    18     0     3    10    10     2     3     1     3
6    20     1     2     7     8     2     3     2     1
7    27     0     1     8     8     1     3     2     1
8    49     1     1     8     8     6     3     2     1
9    57     1     2     7     8     1     3     2     1
10   67     0     1     8     8     1     3     1     1
```

group_by() and summarize()

This function is especially important when calculating summary statistics, which we often want to be stratified.

```
1 nlsy_by_region <- group_by(nlsy, region)
2
3 summarize(nlsy_by_region,
4   mean_inc = mean(income))
```

```
# A tibble: 4 × 2
  region mean_inc
  <dbl>   <dbl>
1     1  17771.
2     2  16698.
3     3  14101.
4     4  13360.
```

Stratify with `group_by() |> summarize()`

Like the other functions we've seen, we can use pipes:

```
1 nlsy |>
2   mutate(income_stand = (income - mean(income))/sd(income)) |>
3   group_by(region) |>
4   summarize(mean_inc = mean(income_stand),
5             sd_inc = sd(income_stand))
```

```
# A tibble: 4 × 3
  region mean_inc sd_inc
  <dbl>   <dbl> <dbl>
1     1    0.186  1.17
2     2    0.106  0.958
3     3   -0.0891 1.03
4     4   -0.145  0.810
```

Counting groups

Sometimes we just want to know how many observations are in a group. We already saw how to do that with `count()`, but we can also do it with `group_by() |> summarize()`:

```
1 nlsy |>
2   count(sex)
```

```
# A tibble: 2 × 2
  sex      n
<dbl> <int>
1     1  501
2     2  704
```

```
1 nlsy |>
2   group_by(sex) |>
3   summarize(n = n())
```

```
# A tibble: 2 × 2
  sex      n
<dbl> <int>
1     1  501
2     2  704
```

Exercise

Getting *other* data into R

We have been reading in data as an `.rds` file:

```
1 nlsy_rds <- read_rds("https://github.com/louisahsmith/data/raw/main/ma")
```

We could also read it in as a `.csv` file:

```
1 nlsy_csv <- read_csv("https://github.com/louisahsmith/data/raw/main/ma")
```


What do you notice about the differences?

```
1 nlsy_rds |> select(id, contain
```

```
# A tibble: 1,205 × 5
  id eyesight_cat glasses_cat
race_eth_cat sex_cat
  <dbl> <fct>      <fct>
<fct>      <fct>
1     3 Excellent Doesn't wear glasses
Non-Black, Non-Hispanic Female
2     6 Very Good  Wears glasses/contacts
Non-Black, Non-Hispanic Male
3     8 Very Good  Doesn't wear glasses
Non-Black, Non-Hispanic Female
4    16 Good       Wears glasses/contacts
Non-Black, Non-Hispanic Female
5    18 Good       Doesn't wear glasses
Non-Black, Non-Hispanic Male
6    20 Very Good  Wears glasses/contacts
Non-Black, Non-Hispanic Female
7    27 Excellent Doesn't wear glasses
Non-Black, Non-Hispanic Female
8    49 Excellent Wears glasses/contacts
Non-Black, Non-Hispanic Female
9    57 Very Good  Wears glasses/contacts
Non-Black, Non-Hispanic Female
10   67 Excellent Doesn't wear glasses
Non-Black, Non-Hispanic Male
```

```
1 nlsy_csv |> select(id, contain
```

```
# A tibble: 1,205 × 6
  id eyesight_cat glasses_cat
race_eth_cat sex_cat slp_cat_wkdy
  <dbl> <chr>      <chr>
<chr>      <chr>      <chr>
1     3 Excellent Doesn't wear glasses
Non-Black, No... Female some
2     6 Very Good  Wears glasses/contacts
Non-Black, No... Male some
3     8 Very Good  Doesn't wear glasses
Non-Black, No... Female ideal
4    16 Good       Wears glasses/contacts
Non-Black, No... Female some
5    18 Good       Doesn't wear glasses
Non-Black, No... Male lots
6    20 Very Good  Wears glasses/contacts
Non-Black, No... Female ideal
7    27 Excellent Doesn't wear glasses
Non-Black, No... Female ideal
8    49 Excellent Wears glasses/contacts
Non-Black, No... Female ideal
9    57 Very Good  Wears glasses/contacts
Non-Black, No... Female ideal
10   67 Excellent Doesn't wear glasses
Non-Black, No... Male ideal
```

`.rds` is an R-specific file for a single object

It will be the exact same object when you read it back in.

```
1 write_rds(nlsy_rds, "nlsy.rds")
```

You can save any object, not just a dataframe:

```
1 x <- c(4, 5, 6)
2 write_rds(x, "numbers.rds")
```

What is `y` going to print?

```
1 y <- read_rds("numbers.rds")
2 y
```

.csv files are much more general but don't maintain things like factors

The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	glasses	eyesight	sleep_wkdy	sleep_wknd	id	nsibs	samp	race_eth	sex	region	income	res_1980	res_2002	age_bir	eyesight_cat	glasses_cat	race_eth_cat	sex_cat	slp_cat_wkdy			
2	0	1	5	7	3	3	5	3	2	1	22390	11	11	19	Excellent	Doesn't wea	Non-Black, N	Female	some			
3	1	2	6	7	6	1	1	3	1	1	35000	3	11	30	Very Good	Wears glass	Non-Black, N	Male	some			
4	0	2	7	9	8	7	6	3	2	1	7227	11	11	17	Very Good	Doesn't wea	Non-Black, N	Female	ideal			
5	1	3	6	7	16	3	5	3	2	1	48000	11	11	31	Good	Wears glass	Non-Black, N	Female	some			
6	0	3	10	10	18	2	1	3	1	3	4510	11	11	19	Good	Doesn't wea	Non-Black, N	Male	lots			
7	1	2	7	8	20	2	5	3	2	1	50000	3	11	30	Very Good	Wears glass	Non-Black, N	Female	ideal			
8	0	1	8	8	27	1	5	3	2	1	20000	11	11	27	Excellent	Doesn't wea	Non-Black, N	Female	ideal			
9	1	1	8	8	49	6	5	3	2	1	23900	11	11	24	Excellent	Wears glass	Non-Black, N	Female	ideal			
10	1	2	7	8	57	1	5	3	2	1	23289	11	11	21	Very Good	Wears glass	Non-Black, N	Female	ideal			
11	0	1	8	8	67	1	1	3	1	1	35000	3	11	36	Excellent	Doesn't wea	Non-Black, N	Male	ideal			
12	0	3	8	8	86	7	7	2	2	1	1688	11	19	17	Good	Doesn't wea	Black	Female	ideal			
13	1	5	7	7	96	2	6	3	2	1	3000	11	11	19	Poor	Wears glass	Non-Black, N	Female	ideal			
14	1	1	7	8	97	7	5	3	2	1	8000	11	11	29	Excellent	Wears glass	Non-Black, N	Female	ideal			
15	0	1	7	7	98	2	6	3	2	1	6618	11	11	30	Excellent	Doesn't wea	Non-Black, N	Female	ideal			
16	0	1	8	8	117	2	1	3	1	1	52300	6	11	26	Excellent	Doesn't wea	Non-Black, N	Male	ideal			
17	0	1	7	7	137	4	5	3	2	1	40000	3	11	26	Excellent	Doesn't wea	Non-Black, N	Female	ideal			
18	0	3	7	4	172	9	6	3	2	1	3162	11	11	35	Good	Doesn't wea	Non-Black, N	Female	ideal			
19	1	2	8	8	179	2	5	3	2	1	10000	11	11	22	Very Good	Wears glass	Non-Black, N	Female	ideal			
20	1	3	8	8	186	2	5	3	2	1	30000	3	11	31	Good	Wears glass	Non-Black, N	Female	ideal			
21	1	3	8	9	200	2	5	3	2	1	25200	11	11	24	Good	Wears glass	Non-Black, N	Female	ideal			
22	0	4	7	7	205	4	8	1	2	1	11960	11	11	21	Fair	Doesn't wea	Hispanic	Female	ideal			
23	1	2	6	10	218	2	1	3	1	1	15000	11	11	31	Very Good	Wears glass	Non-Black, N	Male	some			
24	0	2	8	8	227	4	7	2	2	1	5184	11	11	17	Very Good	Doesn't wea	Black	Female	ideal			
25	0	5	7	7	237	4	5	3	2	3	5710	11	11	23	Poor	Doesn't wea	Non-Black, N	Female	ideal			
26	0	1	8	8	242	2	5	3	2	1	7236	11	11	27	Excellent	Doesn't wea	Non-Black, N	Female	ideal			
27	0	1	7	7	243	3	1	3	1	1	12600	11	11	26	Excellent	Doesn't wea	Non-Black, N	Male	ideal			
28	1	2	7	8	244	4	5	3	2	1	7910	11	11	18	Very Good	Wears glass	Non-Black, N	Female	ideal			
29	0	4	7	7	247	4	2	3	1	1	6000	3	11	27	Fair	Doesn't wea	Non-Black, N	Male	ideal			
30	0	1	6	10	250	1	5	3	2	1	10600	11	11	24	Excellent	Doesn't wea	Non-Black, N	Female	some			
31	1	2	6	6	256	2	5	3	2	1	18000	11	11	18	Very Good	Wears glass	Non-Black, N	Female	some			
32	0	2	6	6	259	2	1	3	1	1	19700	11	11	21	Very Good	Doesn't wea	Non-Black, N	Male	some			
33	1	3	6	6	274	6	2	3	1	1	5800	16	11	19	Good	Wears glass	Non-Black, N	Male	some			
34	1	2	7	7	281	3	1	3	1	1	25000	3	11	34	Very Good	Wears glass	Non-Black, N	Male	ideal			
35	0	2	7	7	290	11	8	1	2	1	17900	11	11	19	Very Good	Doesn't wea	Hispanic	Female	ideal			
36	1	3	8	8	297	2	5	3	2	1	16705	11	11	21	Good	Wears glass	Non-Black, N	Female	ideal			
37	1	2	7	7	317	2	1	3	1	1	29000	3	11	33	Very Good	Wears glass	Non-Black, N	Male	ideal			
38	0	2	6	10	333	5	7	2	2	1	10600	11	11	23	Very Good	Doesn't wea	Black	Female	some			
39	0	4	6	6	335	7	7	2	2	1	800	11	11	32	Fair	Doesn't wea	Black	Female	some			
40	0	3	6	5	337	2	3	2	1	1	5000	1	11	20	Good	Doesn't wea	Black	Male	some			

.csv files might need a little more specification to read in

```
1 read_csv("https://github.com/louisahsmith/data/raw/main/nlsy/nlsy.csv") |> print(n = 2)
```

```
# A tibble: 12,686 × 14  
  H0012400 H0012500 H0022300 H0022500 R0000100 R0009100 R0173600 R0214700  
    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>  
1      -4      -4      -4      -4         1         1         5         3  
2         0         1         4         3         2         8         5         3  
# i 12,684 more rows  
# i 6 more variables: R0214800 <dbl>, R0216400 <dbl>, R0217900 <dbl>,  
#   R0402800 <dbl>, R7090700 <dbl>, T4120500 <dbl>
```

```
1 nlsy_full <- read_csv(  
2   "https://github.com/louisahsmith/data/raw/main/nlsy/nlsy.csv", skip = 1,  
3   col_names = c("glasses", "eyesight", "sleep_wkdy", "sleep_wknd",  
4                 "id", "nsibs", "samp", "race_eth", "sex", "region",  
5                 "income", "res_1980", "res_2002", "age_bir"),  
6                 na = c("-1", "-2", "-3", "-4", "-5", "-998"))  
7 print(nlsy_full, n = 2)
```

```
# A tibble: 12,686 × 14  
  glasses eyesight sleep_wkdy sleep_wknd   id nsibs  samp race_eth  sex region  
    <dbl>    <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  
1     NA     NA     NA     NA     1     1     5     3     2     1  
2      0      1      4      3     2     8     5     3     2     1  
# i 12,684 more rows  
# i 4 more variables: income <dbl>, res_1980 <dbl>, res_2002 <dbl>,  
#   age_bir <dbl>
```

Corresponding `write_()` function

If you are sharing data with collaborators who don't use R, or you want to look at it in Excel, you can save a dataframe as a `.csv` file:

```
1 nlsy_rds <- read_rds("https://github.com/louisahsmith/data/raw/main/nlsy_rds.rds")
2 write_csv(nlsy_rds, "nlsy.csv", na = "")
```

The data will be saved in your “working directory” (see the top of your console)

Note

We'll talk about directories in a little bit!

Other functions come from the `{haven}` package

```
1 library(haven)
2 medical_dta <- read_dta("http://www.principlesofeconometrics.com/s
3 medical_sas <- read_sas("http://www.principlesofeconometrics.com/s
```

```
1 glimpse(medical_dta)
```

```
Rows: 1,000
Columns: 6
$ id      <dbl> 1, 1, 1, 1, 1, 2, 2, 2, 2, 2,
3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, ...
$ year    <dbl> 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, ...
$ medexp  <dbl> 9, 9, 9, 10, 11, 6, 7, 7, 7, 7,
4, 3, 5, 4, 4, 5, 3, 6, 6, 3, 4...
$ inc     <dbl> 49, 51, 55, 58, 61, 48, 48, 58,
59, 63, 46, 51, 55, 58, 63, 68,...
$ age     <dbl> 51, 52, 53, 54, 55, 62, 63, 64,
65, 66, 57, 58, 59, 60, 61, 48,...
$ insur   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, ...
```

```
1 glimpse(medical_sas)
```

```
Rows: 1,000
Columns: 6
$ ID      <dbl> 1, 1, 1, 1, 1, 2, 2, 2, 2, 2,
3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, ...
$ YEAR    <dbl> 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, ...
$ MEDEXP  <dbl> 9, 9, 9, 10, 11, 6, 7, 7, 7, 7,
4, 3, 5, 4, 4, 5, 3, 6, 6, 3, 4...
$ INC     <dbl> 49, 51, 55, 58, 61, 48, 48, 58,
59, 63, 46, 51, 55, 58, 63, 68,...
$ AGE     <dbl> 51, 52, 53, 54, 55, 62, 63, 64,
65, 66, 57, 58, 59, 60, 61, 48,...
$ INSUR   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, ...
```

Or `{readxl}`

```
1 library(readxl)
2 dat <- read_excel("excel_data.xlsx")
```

All these functions take arguments, but `read_excel()` takes *a ton* of arguments – which sheet, how many rows to read, whether there are column names, a specific range to read in, etc....

- See `help(read_excel)` for details!

Exercise

Where are these files? File paths

```
1 list.files()
```

```
[1] "_extensions"      "_freeze"          "_publish.yml"     "_quarto.yml"  
[5] "_site"           "cheat_sheet.html" "cheat_sheet.qmd" "data"  
[9] "data.html"       "data.qmd"         "decktape calls"  "exercises"  
[13] "homeworks"      "ID543 2024.Rproj" "img"              "index.html"  
[17] "index.qmd"      "pages"            "site_libs"       "slides"  
[21] "www"
```

```
1 getwd()
```

```
[1] "/Users/l.smith/Documents/Teaching/Harvard/ID543 2024"
```

```
1 file.path("data", "my_dataset.csv")
```

```
[1] "data/my_dataset.csv"
```

```
1 file.path("~", "Downloads", "my_dataset.csv")
```

```
[1] "~/Downloads/my_dataset.csv"
```

```
1 file.path("C:", "Users", "Downloads", "my_dataset.csv")
```

```
[1] "C:/Users/Downloads/my_dataset.csv"
```

The problem with `setwd()`

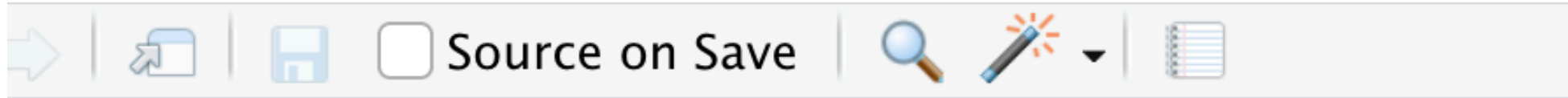
- `setwd()` changes the working directory, leading to potential issues in collaboration and reproducibility
- You and I don't have the same file structure!
- For example, my current working directory is

```
1 getwd()
```

```
[1] "/Users/l.smith/Documents/Teaching/Harvard/ID543 2024"
```

- It's also really annoying to change your working directory when you move around files and folders, even if it's just you using them

Do you think this code from 2015 still works?



```
1 #Create results table for poster
2 rm(list=ls())
3 setwd("~/Box Sync/Behavior Outcomes/")
4
5 #Load data
6 load("./Data/behavior_datasets.Rdata")
7
```

R Projects

```
my-project/  
├── my-project.Rproj  
├── README  
├── data/  
│   ├── raw/  
│   └── processed/  
├── R/  
├── results/  
│   ├── tables/  
│   ├── figures/  
│   └── output/  
└── docs/
```

- An `.Rproj` file is mostly just a placeholder. It remembers various options, and makes it easy to open a new RStudio session that starts up in the correct working directory. You never need to edit it directly.
- A README file can just be a text file that includes notes for yourself or future users.
- I like to have a folder for raw data – which I never touch – and a folder(s) for datasets that I create along the way.

R Projects

Demo

Referring to files with the `here` package

```
1 source(here::here("R", "functions.R"))
2
3 dat <- read_csv(here::here("data", "raw", "data.csv"))
4
5 p <- ggplot(dat) + geom_point(aes(x, y))
6
7 ggsave(plot = p,
8         filename = here::here("results", "figures", "fig.pdf"))
```

- The `here` package lets you refer to files without worrying too much about relative file paths.
- Construct file paths with reference to the top directory holding your `.Rproj` file.
- `here::here("data", "raw", "data.csv")` for me, here, becomes `"/Users/l.smith/Documents/Teaching/Harvard/ID543_2024/data/raw/data.csv"`
- But if I send you my code to run, it will become whatever file path *you* need it to be, as long as you're running it within the R Project.

Referring to the `here` package

```
1 here::here()
```

is equivalent to

```
1 library(here)
2 here()
```

I just prefer to write out the package name whenever I need it, but you can load the package for your entire session if you want.

Note

Note that you can refer to any function without loading the whole package this way, e.g. `haven::read_dta()`

Exercise

Missing values

- R uses `NA` for missing values
- Unlike some other statistical software, it will return `NA` to any logical statement
- This makes it somewhat harder to deal with but also harder to make mistakes

```
1 3 < NA
```

```
[1] NA
```

```
1 mean(c(1, 2, NA))
```

```
[1] NA
```

```
1 mean(c(1, 2, NA), na.rm = TRUE)
```

```
[1] 1.5
```

Special NA functions

Certain functions deal with missing values explicitly

```
1 vals <- c(1, 2, NA)
2 is.na(vals)
```

```
[1] FALSE FALSE TRUE
```

```
1 anyNA(vals)
```

```
[1] TRUE
```

```
1 na.omit(vals)
```

```
[1] 1 2
```

Specific missingness

You know some value is implausible, whether for everyone or for a specific observation

```
1 nlsy <- nlsy |>
2   mutate(sleep_wknd = ifelse(sleep_wknd > 24, NA, sleep_wknd),
3         # OR
4         sleep_wknd = case_when(
5           sleep_wknd > 24 ~ NA,
6           .default = sleep_wknd
7         ),
8         income = ifelse(id == 283, NA, income),
9         nsibs = na_if(nsibs, 99))
```

`na_if(x, y)` will replace values in `x` that are equal to `y` with NA

Read in NA's directly

In NLSY, -1 = Refused, -2 = Don't know, -3 = Invalid missing, -4 = Valid missing, -5 = Non-interview

Other files might have `.` for missing, or `999`.

```
1 nlsy_cols <- c("glasses", "eyesight", "sleep_wkdy", "sleep_wknd",
2               "id", "nsibs", "samp", "race_eth", "sex", "region",
3               "income", "res_1980", "res_2002", "age_bir")
4 nlsy <- read_csv("https://github.com/louisahsmith/data/raw/main/nlsy.csv",
5                 na = c("-1", "-2", "-3", "-4", "-5", "-998"),
6                 skip = 1, col_names = nlsy_cols)
```

- You have to write the values as strings, even if they're numbers

Reasons for missingness

Caveat: This previous way, you lose the info about the reason for missingness. If that's important, read in the data first, create a variable for missingness reason (e.g., use `fct_recode()`), then changes the values to `NA`.

```
1 nlsy <- read_csv("https://github.com/louisahsmith/data/raw/main/nlsy.csv")
2           skip = 1, col_names = nlsy_cols) |>
3   mutate(age_bir_missing = ifelse(age_bir > 0, NA, age_bir),
4          age_bir_missing = fct_recode(
5            factor(age_bir_missing), "Refused" = "-1",
6            "Don't know" = "-2", "Invalid missing" = "-3",
7            "Valid missing" = "-4", "Non-interview" = "-5",
8            "Other missing" = "-998"))
9   summary(nlsy$age_bir_missing)
```

Other missing	Non-interview	Invalid missing	NA's
1343	5385	15	5943

Complete cases

Sometimes you may just want to get rid of all the rows with missing values.

```
1 nrow(nlsy)
```

```
[1] 12686
```

```
1 nlsy_cc <- nlsy |> filter(complete.cases(nlsy))  
2 nrow(nlsy_cc)
```

```
[1] 6743
```

```
1 nlsy2 <- nlsy |> na.omit() # same
```

Caution

Don't do this without good reason! It will exclude rows with *any* missing values, even in variables you're not using.

Exercise

Joins

There are multiple functions in the `tidyverse` (specifically, the `dplyr` package) for joining/merging data

Mutating joins merge two datasets based on matching variable(s), adding together the new columns from the joined dataframe

```
1 left_join(x, y, by = join_by(xcol == ycol))
```

Note

We will also refer to the `x` dataframe as the left-hand side (LHS) and the `y` dataframe as the right-hand side (RHS)

Merging with kids

The NLSY also included the kids of the moms in the NLSY79 survey that we're using.

```
1 nlsy_kids
```

```
# A tibble: 11,551 × 6
  id_kid id_mom sex_kid dob_kid agebir_mom bwt_kid
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     201     2     2   1993     34    139
2     202     2     2   1994     35     NA
3     301     3     2   1981     19    162
4     302     3     2   1983     22    144
5     303     3     2   1986     24    112
6     401     4     1   1980     18    107
7     403     4     2   1997     34     NA
8     801     8     2   1976     17    119
9     802     8     1   1979     20    107
10    803     8     2   1982     24    146
# i 11,541 more rows
```

Note

Birthweight is in ounces

Left join

```
1 left_join(nlsy_sleep, nlsy_kids)
```

```
Error in `left_join()`:  
! `by` must be supplied when `x` and `y` have no common variables.  
i Use `cross_join()` to perform a cross-join.
```

It will automatically look for matching columns (can be dangerous!) but if none, need to specify:

```
1 left_join(nlsy_sleep, nlsy_kids,  
2           by = join_by(id == id_mom))
```

```
# A tibble: 2,284 × 10  
  id sleep_wkdy sleep_wknd age_bir sex id_kid sex_kid dob_kid agebir_mom  
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 3 5 7 19 2 301 2 1981 19  
2 3 5 7 19 2 302 2 1983 22  
3 3 5 7 19 2 303 2 1986 24  
4 6 6 7 30 1 NA NA NA  
5 8 7 9 17 2 801 2 1976 17  
6 8 7 9 17 2 802 1 1979 20  
7 8 7 9 17 2 803 2 1982 24  
8 16 6 7 31 2 1601 1 1990 31
```

Left join

LHS rows are duplicated if we have multiple matches, but we lose any rows in the RHS dataset that don't have a match

```
1 n_distinct(nlsy_kids$id_kid)
```

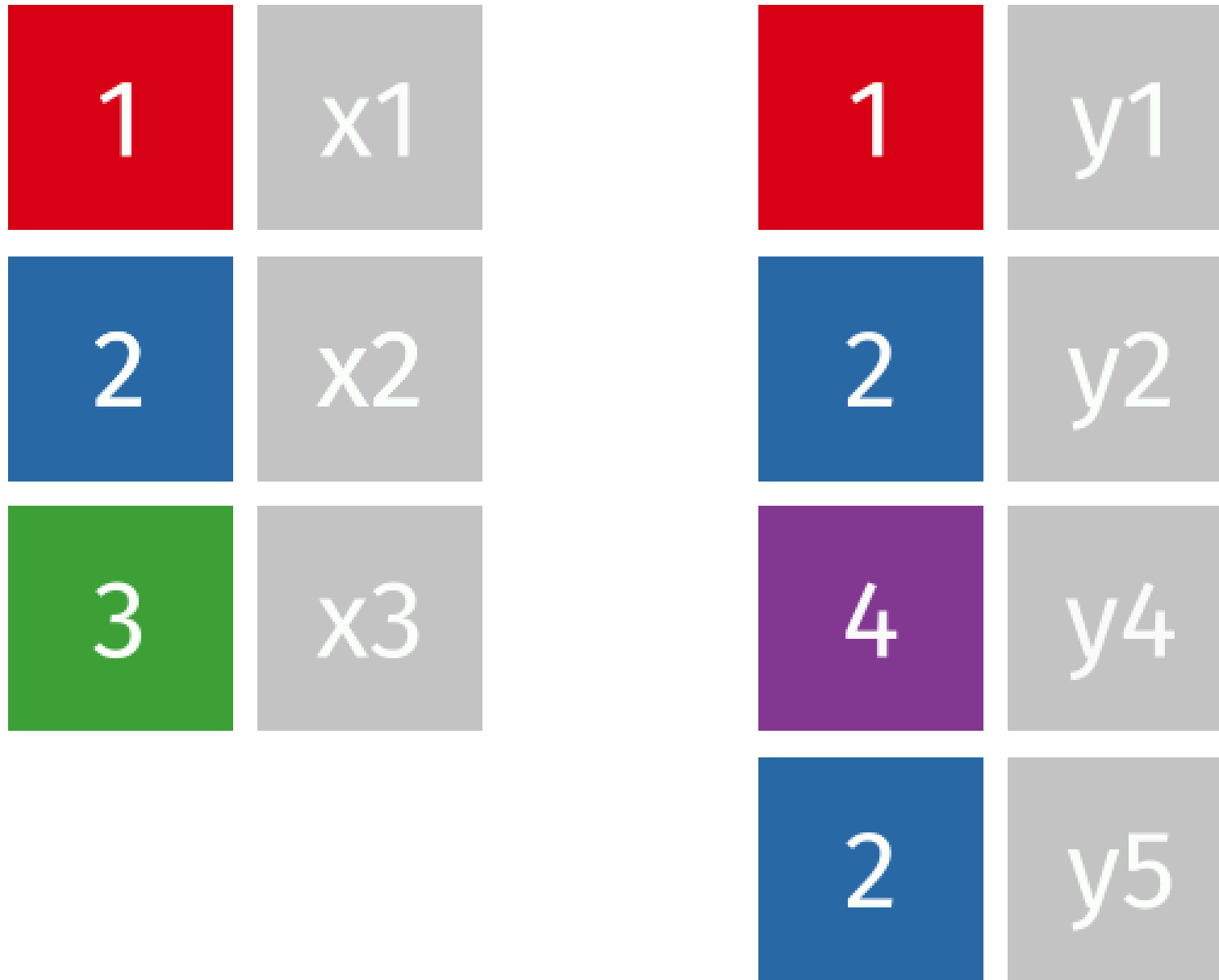
```
[1] 11551
```

```
1 nlsy_left <- left_join(nlsy_sleep, nlsy_kids,  
2                       by = join_by(id == id_mom))  
3  
4 n_distinct(nlsy_left$id_kid)
```

```
[1] 1784
```

In this case, the moms of some of the kids aren't in the `nlsy_sleep` dataset, so kids without moms are lost

left_join(x, y)



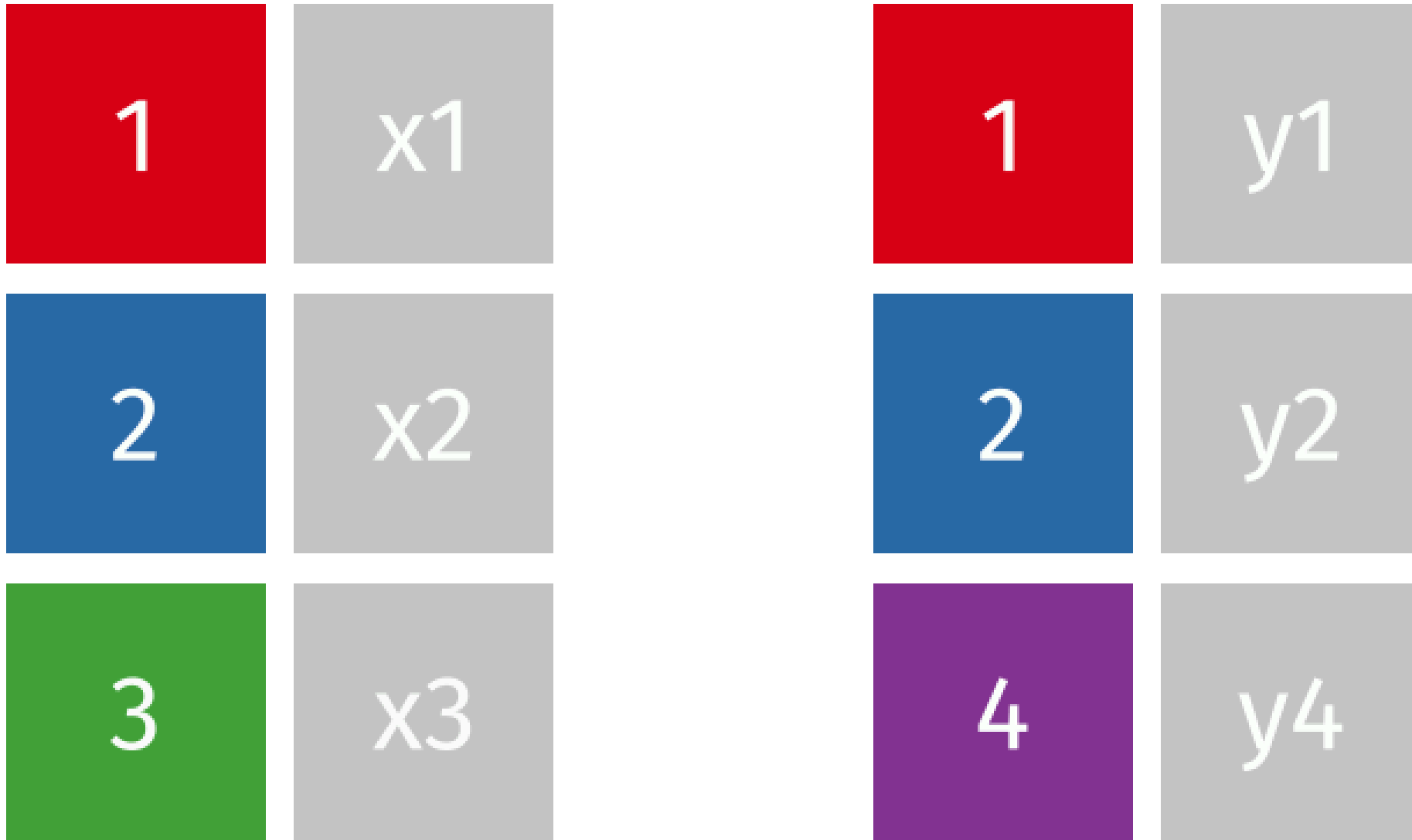
Right join

```
1 right_join(nlsy_sleep, nlsy_kids, by = join_by(id == id_mom))
```

```
# A tibble: 11,551 × 10
  id sleep_wkdy sleep_wknd age_bir sex id_kid sex_kid dob_kid agebir_mom
  <dbl>      <dbl>      <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>
1     3         5         7     19     2    301     2    1981        19
2     3         5         7     19     2    302     2    1983        22
3     3         5         7     19     2    303     2    1986        24
4     8         7         9     17     2    801     2    1976        17
5     8         7         9     17     2    802     1    1979        20
6     8         7         9     17     2    803     2    1982        24
7    16         6         7     31     2   1601     1    1990        31
8    16         6         7     31     2   1602     1    1993        34
9    16         6         7     31     2   1603     2    1996        37
10   20         7         8     30     2   2001     2    1990        30
# i 11,541 more rows
# i 1 more variable: bwt_kid <dbl>
```

- Now we don't have the dads, because there are no matching ids in the RHS dataset
- But we do keep all the kids, even those without moms in the LHS

`right_join(x, y)`



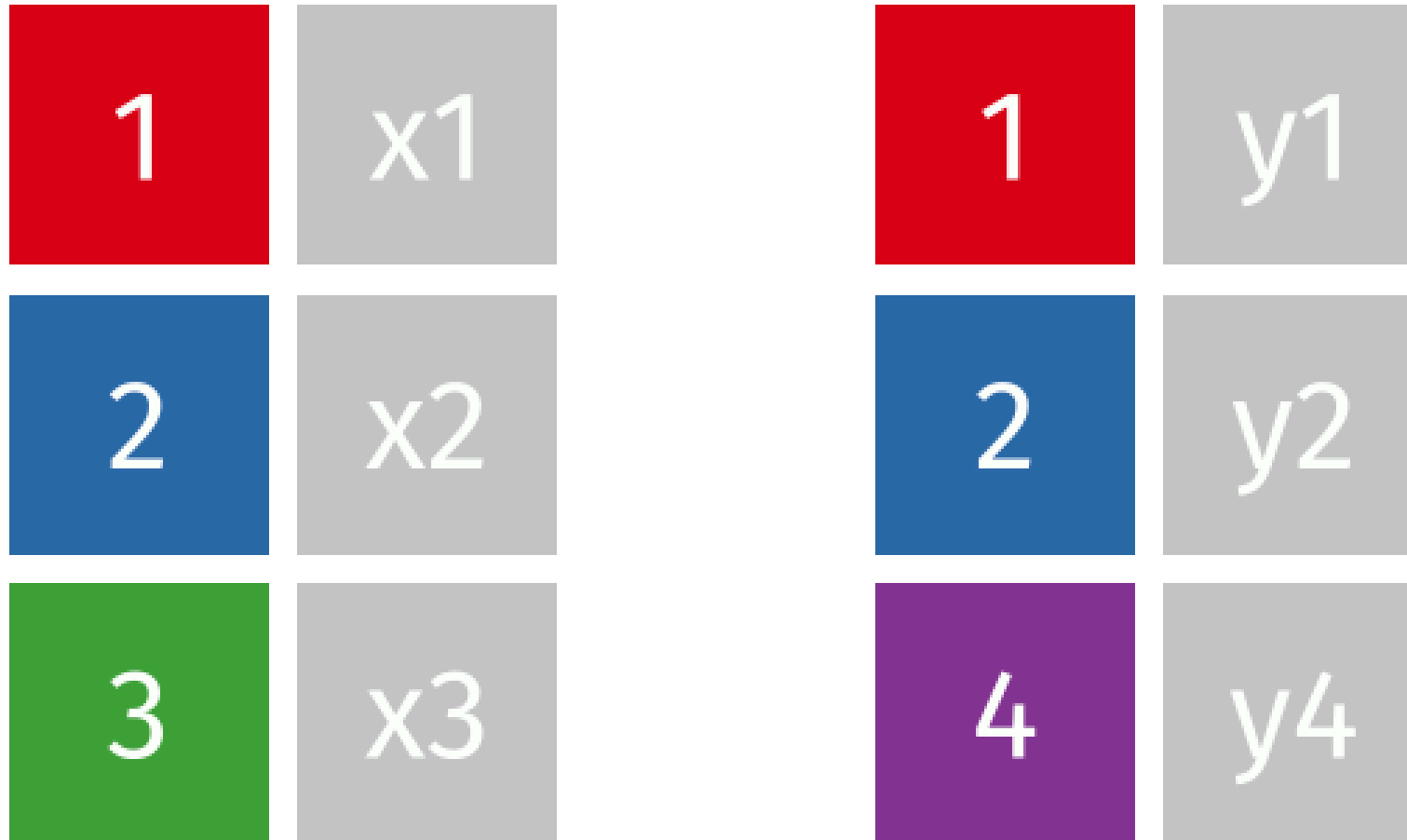
Full join: we want everything!

```
1 full_join(nlsy_sleep, nlsy_kids, by = join_by(id == id_mom))
```

```
# A tibble: 12,052 × 10
  id sleep_wkdy sleep_wknd age_bir sex id_kid sex_kid dob_kid agebir_mom
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 3 5 7 19 2 301 2 1981 19
2 3 5 7 19 2 302 2 1983 22
3 3 5 7 19 2 303 2 1986 24
4 6 6 7 30 1 NA NA NA
5 8 7 9 17 2 801 2 1976 17
6 8 7 9 17 2 802 1 1979 20
7 8 7 9 17 2 803 2 1982 24
8 16 6 7 31 2 1601 1 1990 31
9 16 6 7 31 2 1602 1 1993 34
10 16 6 7 31 2 1603 2 1996 37
# i 12,042 more rows
# i 1 more variable: bwt_kid <dbl>
```

This dataset is larger than either of the initial datasets alone: it has the dads without kids and the kids without moms

full_join(x, y)



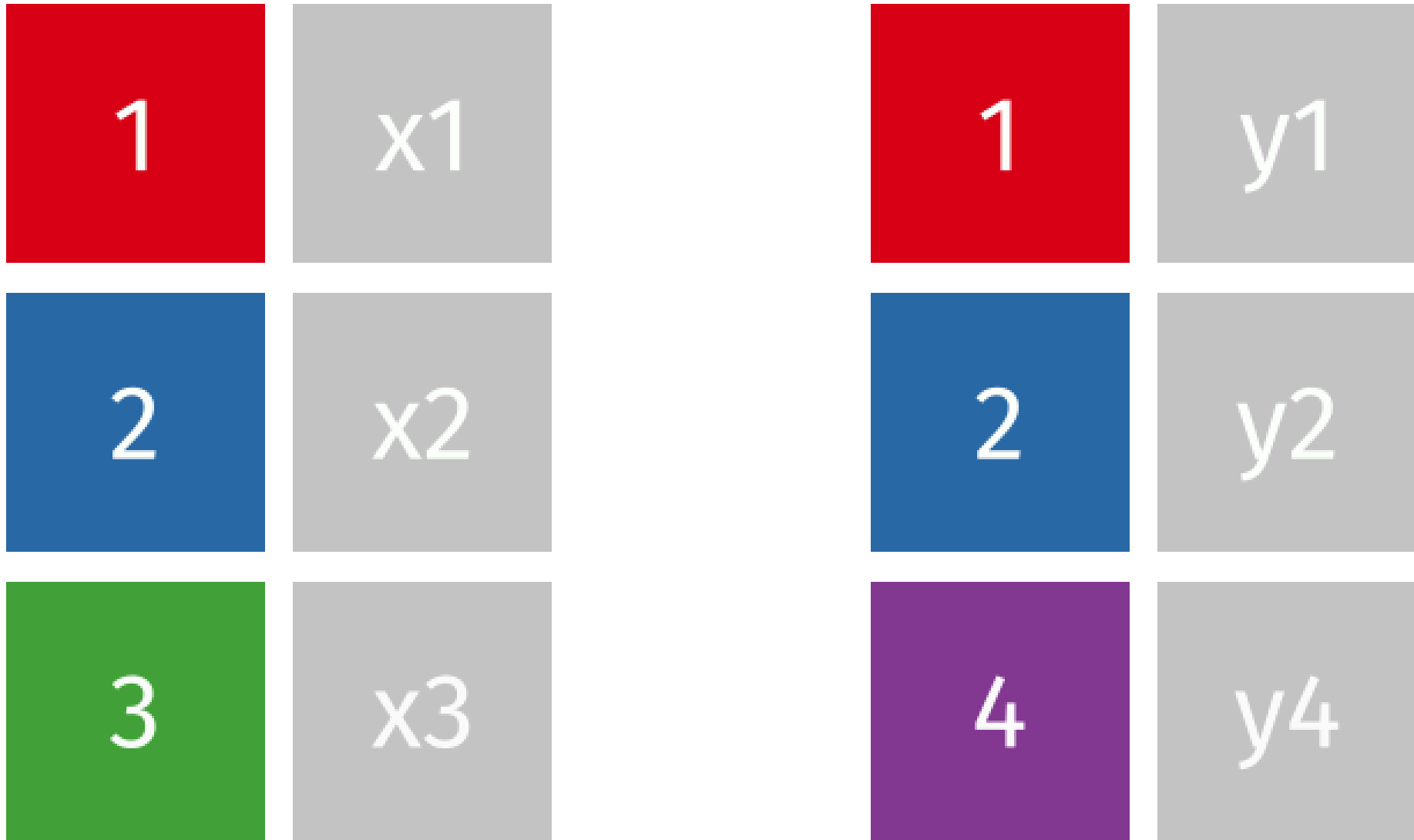
Inner join: we *only* want matches

```
1 inner_join(nlsy_sleep, nlsy_kids, by = join_by(id == id_mom))
```

```
# A tibble: 1,783 × 10
  id sleep_wkdy sleep_wknd age_bir sex id_kid sex_kid dob_kid agebir_mom
  <dbl>      <dbl>      <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>
1     3         5         7     19     2   301     2   1981         19
2     3         5         7     19     2   302     2   1983         22
3     3         5         7     19     2   303     2   1986         24
4     8         7         9     17     2   801     2   1976         17
5     8         7         9     17     2   802     1   1979         20
6     8         7         9     17     2   803     2   1982         24
7    16         6         7     31     2  1601     1   1990         31
8    16         6         7     31     2  1602     1   1993         34
9    16         6         7     31     2  1603     2   1996         37
10   20         7         8     30     2  2001     2   1990         30
# i 1,773 more rows
# i 1 more variable: bwt_kid <dbl>
```

- This dataset has only the moms with kids (no dads) and the kids with moms
- It still has multiple rows per mom – one for each kid

inner_join(x, y)



Join by multiple variables

- I only want the kid that was the mom's first
- I'm going to match on the age at first birth on the RHS

```
1 first_births <- inner_join(nlsy_sleep, nlsy_kids,  
2                             by = join_by(id == id_mom,  
3                                           age_bir == agebir_mom))  
4  
5 first_births
```

```
# A tibble: 708 × 9  
  id sleep_wkdy sleep_wknd age_bir sex id_kid sex_kid dob_kid bwt_kid  
  <dbl>      <dbl>      <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  
1     3         5         7     19     2    301     2    1981    162  
2     8         7         9     17     2    801     2    1976    119  
3    16         6         7     31     2   1601     1    1990    109  
4    20         7         8     30     2   2001     2    1990    129  
5    27         8         8     27     2   2701     2    1988    117  
6    49         8         8     24     2   4901     1    1982    139  
7    57         7         8     21     2   5701     1    1979    148  
8    86         8         8     17     2   8601     2    1977     97  
9    96         7         7     19     2   9601     2    1980    124  
10   97         7         8     29     2   9701     1    1987     48
```

Exercise

Today's summary

- We can use `summarize()` to get summary statistics of our data
- We can use `group_by()` to group our data and then get summary statistics within those groups
- Missing values in R are `NA`
- R projects are a good way to keep your files organized
- We can use the `here` package to refer to files in a project
- We can use `left_join()`, `right_join()`, `full_join()`, and `inner_join()` to merge datasets

Today's functions

- `summarize()`: calculate summary statistics
- `group_by()`: group data
- `here::here()`: refer to files in a project
- `read_rds()`, `read_csv()`, `read_dta()`, `read_sas()`, `read_excel()`: read in data
- `complete.cases()`, `na.omit()`: remove missing values
- `is.na()`, `anyNA()`: check for missing values
- `na_if()`: replace values with `NA`
- `left_join()`, `right_join()`, `full_join()`, `inner_join()`: merge datasets