Redefining rows: Wide vs narrow data organization

Code

Instructor - Soumya Mukherjee

Content Credit- Dr. Matthew Beckman and Olivia Beck

July 20, 2023

Cases, Variables, and Values

A data table is comprised of *cases* and *variables*.

Each *variable* comprises *values* (or levels).

There is no hard distinction between a variable and a value. What's a variable in one situation may be a value in another, and vice versa.

Here are 2 tables with the same information.

The question is what IS different and what is NOT different.

Table 1:

Hide

```
example_eagle_nests
```

region <chr>	2007 <dbl>	2009 <dbl>
Pacific	1039	2587
Southwest	51	176
Rocky Mountains and Plains	200	338

3 rows

Table 2:

Hide

```
example_eagle_nests_tidy
```

region <chr>	year <chr>	num_nests <dbl>
Pacific	2007	1039
Pacific	2009	2587
Southwest	2007	51
Southwest	2009	176
Rocky Mountains and Plains	2007	200
Rocky Mountains and Plains	2009	338

6 rows

Cases, Variables, and Values

For Table 1 ….

- Variables: `2007` , `2009`
  - Values:
    - `2007` : count of nests
    - `2009` : count of nests
- Cases: Regions {Pacific, Southwest, Rocky Mountains and Plains}

region <chr>	2007 <dbl>	2009 <dbl>
Pacific	1039	2587
Southwest	51	176
Rocky Mountains and Plains	200	338

3 rows

For Table 2

- Variables: `year` , `num_nests`
    - Values:
        - `year` : 2007 and 2009
        - `num_nests` : count of nests
- Cases: Regions and Year

region <chr>	**year** <chr>	**num_nests** <dbl>
Pacific	2007	1039
Pacific	2009	2587
Southwest	2007	51
Southwest	2009	176
Rocky Mountains and Plains	2007	200
Rocky Mountains and Plains	2009	338

6 rows

Neither of the tables are wrong, but sometimes one form is more helpful than the other.

Two formats

- Data in Key/Value format are **narrow**
    - possible to get *too* narrow if the meaning of case becomes awkward

ID	**Key**	**Value**
ID1	Key1	value(1,1)
ID1	Key2	value(1,2)
…	…	…
ID2	Key1	value(2,1)

ID	Key	Value
ID2	Key2	value(2,2)
…	…	…

- The corresponding **wide** format has
  - separate variables for each level in `key`
  - sets the values for those variables from the info in `value`

ID	Key1	Key2	….
ID1	value(1,1)	value(1,2)	….
ID2	value(2,1)	value(2,2)	….
…	…	…	

Narrow

Let's remember our original table:

Hide

```
example_eagle_nests
```

region <chr>	2007 <dbl>	2009 <dbl>
Pacific	1039	2587
Southwest	51	176
Rocky Mountains and Plains	200	338

3 rows

Let's make it narrow

Hide

```
narrow_table <-
  example_eagle_nests %>%
  pivot_longer(
    cols = c(`2007`, `2009`),  #names of columns we want to become a new variable
    names_to = "year",         #what you want to call the new column of the data in the line above
    values_to = "num_nests"    #what you want to call the variable that stores the values
  )

narrow_table
```

region <chr>	year <chr>	num_nests <dbl>
Pacific	2007	1039
Pacific	2009	2587
Southwest	2007	51
Southwest	2009	176
Rocky Mountains and Plains	2007	200
Rocky Mountains and Plains	2009	338

6 rows

This is a good narrow table (and a tidy table!). Each case (region, year, quarter combination) has a row and each variable has a column.

Be careful you don't make your tables too narrow where we loose the definition of a case (example on next slide).

Wide

Let's remember our narrow table.

Hide

```
narrow_table
```

region <chr>	year <chr>	num_nests <dbl>
Pacific	2007	1039
Pacific	2009	2587
Southwest	2007	51
Southwest	2009	176
Rocky Mountains and Plains	2007	200
Rocky Mountains and Plains	2009	338

6 rows

Hide

```
NA
```

Now we can make our narrow table wide again.

Hide

```
wide_table <-
  narrow_table %>%
  pivot_wider(
    names_from = year,       # the column in the narrow table with the column names for the wide table
    values_from = num_nests  # the column in the narrow table with the values in the narrow wide
  )

wide_table
```

region <chr>	2007 <dbl>	2009 <dbl>
Pacific	1039	2587
Southwest	51	176

region <chr>	2007 <dbl>	2009 <dbl>
Rocky Mountains and Plains	200	338

3 rows

We can compare this to the original table to make sure its the same thing. Yep!

Hide

```
example_eagle_nests
```

region <chr>	2007 <dbl>	2009 <dbl>
Pacific	1039	2587
Southwest	51	176
Rocky Mountains and Plains	200	338

3 rows

(Too) Narrow

Say this is the data set we have:

Hide

```
class_example_Table
```

region <chr>	2007_Q1 <dbl>	2007_Q2 <dbl>	2007_Q3 <dbl>	2007_Q4 <dbl>	2009_Q1 <dbl>	2009_Q2 <dbl>	2009_Q3 <dbl>	2009_Q4 <dbl>
Pacific	306	244	183	306	761	609	457	761
Southwest	15	12	9	15	52	41	31	52

region <chr>	2007_Q1 <dbl>	2007_Q2 <dbl>	2007_Q3 <dbl>	2007_Q4 <dbl>	2009_Q1 <dbl>	2009_Q2 <dbl>	2009_Q3 <dbl>	2009_Q4 <dbl>
Rocky Mountains and Plains	59	47	35	59	99	80	60	99

3 rows

Hide

```
NA
```

We can make it (appropriately) narrow like this:

Hide

```
class_example_Table %>%
  pivot_longer(!region,                          #everything but region
               names_to = c("Year", "Quarter"),
               names_sep = "_",                   #how year and quarter are seperated
               values_to = "NestCount")
```

region <chr>	Year <chr>	Quarter <chr>	NestCount <dbl>
Pacific	2007	Q1	306
Pacific	2007	Q2	244
Pacific	2007	Q3	183
Pacific	2007	Q4	306
Pacific	2009	Q1	761
Pacific	2009	Q2	609
Pacific	2009	Q3	457
Pacific	2009	Q4	761

region <chr>	Year <chr>	Quarter <chr>	NestCount <dbl>
Southwest	2007	Q1	15
Southwest	2007	Q2	12

1-10 of 24 rows Previous **1** 2 3 Next

Hide

```
NA
```

But we could have made it too narrow. None of the data has been lost here, but it's not a helpful form since there isn't a useful definition of "case".

Hide

```
names <- colnames(class_example_Table)

class_example_Table %>%
  mutate_all(as.character) %>%
  pivot_longer(cols = names,
               names_to = "key",
               values_to = "value")
```

```
Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
Please use `all_of()` or `any_of()` instead.
# Was:
data %>% select(names)

# Now:
data %>% select(all_of(names))

See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

key <chr>	value <chr>
region	Pacific
2007_Q1	306
2007_Q2	244
2007_Q3	183
2007_Q4	306
2009_Q1	761
2009_Q2	609
2009_Q3	457
2009_Q4	761
region	Southwest

1-10 of 27 rows Previous **1** 2 3 Next

So what?

- This allows us to easily redefine how rows are presented in the data
    - Possibly motivated by the research question
    - Possibly motivated by desire to join two data tables with different case definitions
    - Possibly motivated by a data visualization
- Also, some operations are easy in wide format, but hard in narrow and *vice versa*
- We need tools that make it easy to switch back and forth

Example from `BabyNames`

name <chr>	sex <chr>	count <int>	year <int>
Eden	F	1927	2012
Eden	M	348	2012
Eden	F	2022	2013
Eden	M	377	2013
Hazel	F	1780	2012
Hazel	F	2039	2013
Hazel	M	6	2013
Jack	F	10	2012
Jack	M	7915	2012
Jack	F	6	2013

Questions:

- Research Question 1. How many babies of each name and sex?
- Research Question 2. For each name, is it primarily given to girls or boys? Which names are gender neutral?

In narrow format

Hide

```
data("BabyNames", package = "dcData")

BabyNames <-
  BabyNames %>%
  filter( name %in% c("Eden", "Jack", "Hazel"))
```

RQ 1. How many babies of each name and sex?

Hide

```
BabyTotals <-
  BabyNames %>%
  group_by(name, sex) %>%
  summarise(total = sum(count))
```

```
`summarise()` has grouped output by 'name'. You can override using the `.groups` argument.
```

name <chr>	**sex** <chr>	**total** <int>
Eden	F	23892
Eden	M	3640
Hazel	F	238522
Hazel	M	2644
Jack	F	2611
Jack	M	650847

6 rows

Easy!

In Wide format

RQ 2. Which names are most gender neutral?

```
WideOutput <-
  NarrowInput %>%
  pivot_wider(names_from = var1, values_from = var2, values_fill = 0)
```

- we want a new column for each category of `sex` , so `names_from = sex`
  - we will “unstack” each available category as a new variable (a.k.a. cast, spread, unfold)
  - categories of `sex` were “F” and “M” in this example
- the values/entries for our new variables are coming from `total` , so `values_from = total`
  - `values_fill = 0` specifies a default value to fill when missing

Hide

```
BabyTotalsWide <-
  BabyTotals %>%
  pivot_wider(names_from = sex, values_from = total, values_fill = 0)

BabyTotalsWide
```

name <chr>	**F** <int>	**M** <int>
Eden	23892	3640
Hazel	238522	2644
Jack	2611	650847

3 rows

With sexes side by side…

We can easily calculate balance associated with names

Hide

```
BabyTotalsWide <-
  BabyTotalsWide %>%
  rename(fem = F, male = M) %>%         # `F` is a terrible variable name (why?)
  mutate(prop_fem  = fem  / (male + fem),
         prop_male = male / (male + fem),
         name_specificity = pmax(prop_fem, prop_male))    # what does `pmax()` do?

BabyTotalsWide
```

name <chr>	fem <int>	male <int>	prop_fem <dbl>	prop_male <dbl>	name_specificity <dbl>
Eden	23892	3640	0.867790208	0.1322098	0.8677902
Hazel	238522	2644	0.989036597	0.0109634	0.9890366
Jack	2611	650847	0.003995666	0.9960043	0.9960043

3 rows

`pivot_longer( )`—when you have “Wide” and want “Narrow”

Syntax:

```
NarrowOutput <-
  WideInput %>%
  pivot_longer(cols = c(wide_var1, wide_var2, ...), names_to = "long_var1", values_to "long_var2")
```

- The `cols` are the variables we want to combine (a.k.a. melt, stack, fold, gather)
- e.g. `prop_fem` and `prop_male` in this case

Hide

```
BabyTotalsNarrow <-
  BabyTotalsWide %>%
  select(prop_fem, prop_male) %>%
  pivot_longer(cols = c(prop_fem, prop_male), names_to = "sex", values_to = "proportion")
```

```
Adding missing grouping variables: `name`
```

Hide

```
BabyTotalsNarrow
```

name <chr>	sex <chr>	proportion <dbl>
Eden	prop_fem	0.867790208
Eden	prop_male	0.132209792
Hazel	prop_fem	0.989036597
Hazel	prop_male	0.010963403
Jack	prop_fem	0.003995666
Jack	prop_male	0.996004334

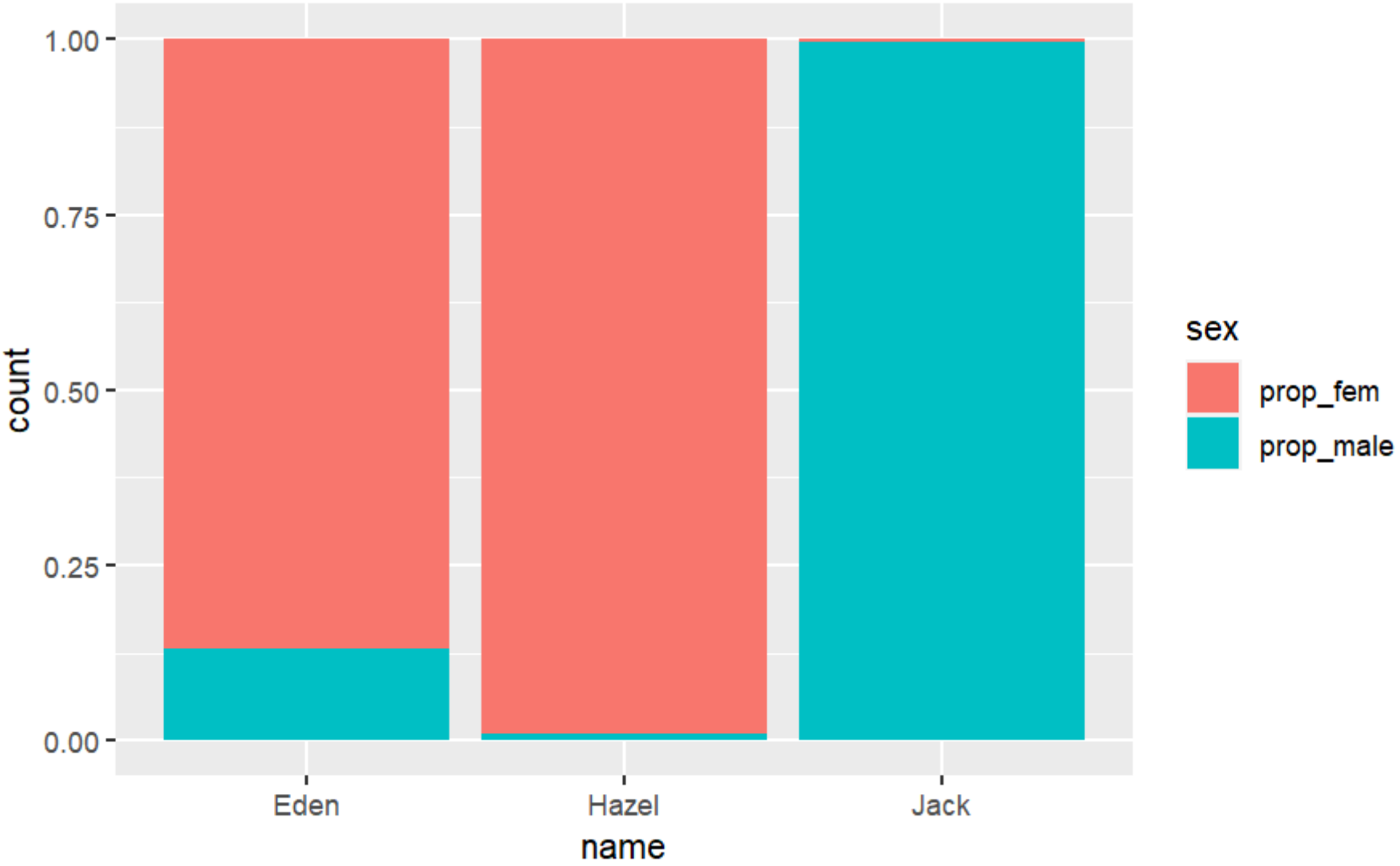
6 rows

With sexes stacked again…

We can make an intuitive bar chart (though some clean up is needed…)

Hide

```
BabyTotalsNarrow %>%
  ggplot() +
  geom_bar(aes(x = name, fill = sex, weight = proportion))
```



Hide

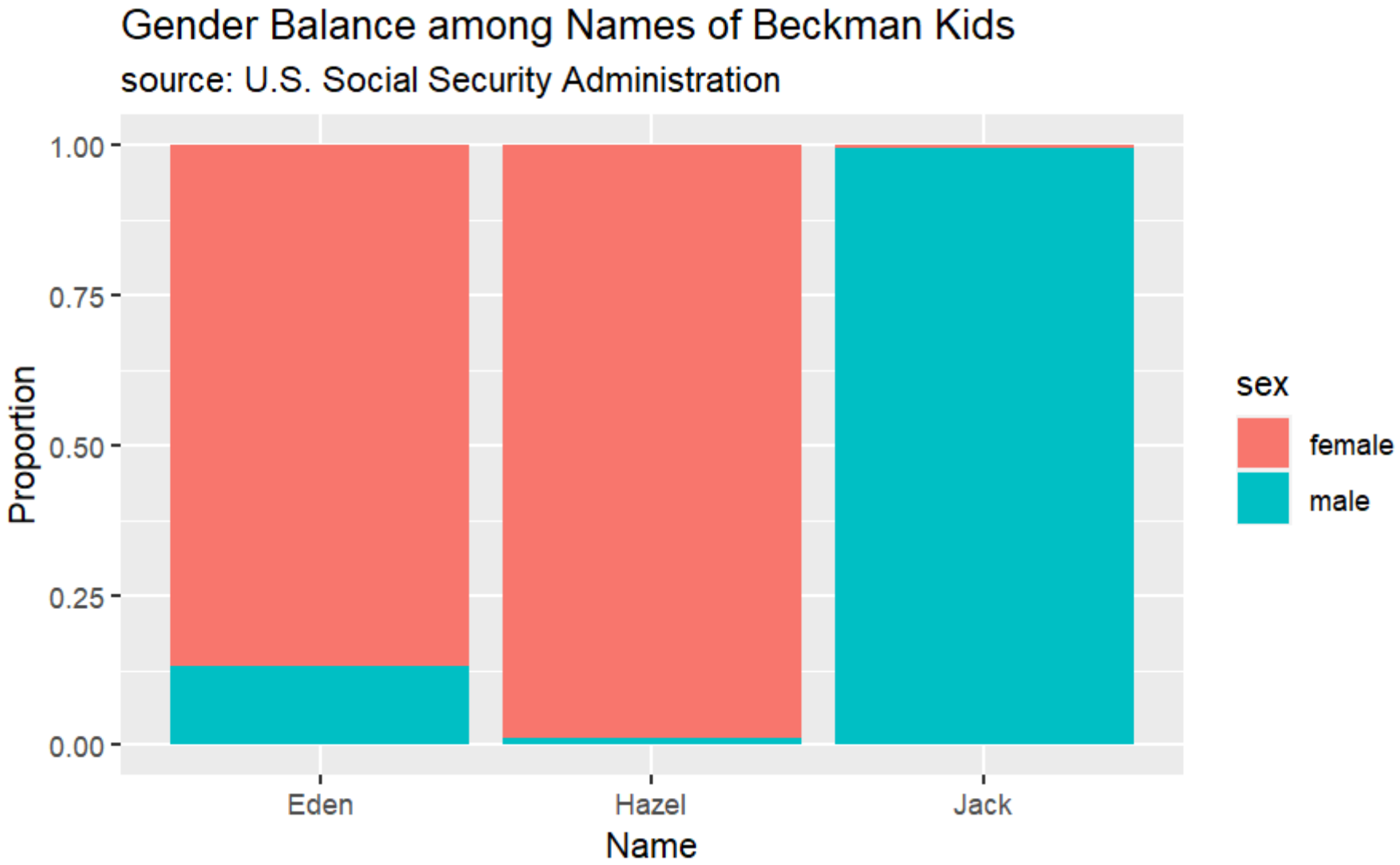
NA

With some improvements

- clean up labels of sexes
- add title, source, & better axis labels (y-axis label had been flat wrong)

Hide

```
# first, clean up the labels in `sex` for plotting
BabyTotalsNarrow %>%
  mutate(sex = if_else(sex == "prop_fem",
                       true = "female",
                       false = if_else(sex == "prop_male",
                                       true = "male",
                                       false = "unk")  # end of "inner" if_else()
                       )                               # ends the "outer" if_else()
         ) %>%                                         # ends the mutate()
  ggplot() +
  geom_bar(aes(x = name, fill = sex, weight = proportion)) +
  ggtitle("Gender Balance among Names of Beckman Kids",
          subtitle = "source: U.S. Social Security Administration") +
  xlab("Name") +
  ylab("Proportion")
```



Hide

```
NA
```

Assignment Reminders

- Activity: PopularNames (due Friday July 21, 9:59 am)

- Reading quiz DataComputing Ebook Chapters 10 and 11 (due Friday July 21, 9:59 am)
- Reading quiz DataComputing Ebook Chapter 12 (due Monday July 24, 9:59 am)
- Activity: STAT184-Bird-Species (due Tuesday July 24, 9:59 am)
- Activity: STAT184-Bicycle-Sharing (due Tuesday July 24, 9:59 am)