

Lab 03: CS631

Working with Data

Alison Hill



Data for today

We'll use data from [Wordbank](#)- an open source database of children's vocabulary development. The tool used to measure children's language and communicative development in this database is the [MacArthur-Bates Communicative Development Inventories \(MB-CDI\)](#). The MB-CDI is a parent-reported questionnaire.

- R package [wordbankr](#)
- [wordbankr](#) vignette
- More about [Wordbank](#)
- More about [MB-CDI](#)

Get the data

Use this code chunk to import my cleaned CSV file:

```
library(readr)  
sounds ← read_csv("http://bit.ly/cs631-meow")
```

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

<code>as.logical</code>	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
<code>as.numeric</code>	1, 0, 1	Integers or floating point numbers.
<code>as.character</code>	'1', '0', '1'	Character strings. Generally preferred to factors.
<code>as.factor</code>	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

RStudio Base R Cheatsheet

<https://github.com/rstudio/cheatsheets/blob/master/base-r.pdf>

Know your data types

- Numeric (2 subtypes)
 - Integers (1, 50)
 - Double (1.5, 50.25, ?double)
- Character ("hello")
- Factor (grade = "A" | grade = "B")
- Logical (TRUE | FALSE)

```
typeof(sounds$age)
```

```
[1] "double"
```

```
typeof(sounds$sound)
```

```
[1] "character"
```

```
typeof(sounds$sound == "meow")
```

```
[1] "logical"
```

Even better: glimpse

```
glimpse(sounds)
```

```
Observations: 33
```

```
Variables: 4
```

```
$ age          <dbl> 8, 8, 8, 9, 9, 9, 10, 10, 10, 11, 11, 11, 12, 12, 12,  
$ sound        <chr> "cockadoodledoo", "meow", "woof woof", "cockadoodledoo  
$ kids_produce <dbl> 1, 0, 3, 0, 2, 2, 0, 5, 4, 0, 5, 12, 0, 12, 28, 9, 125  
$ kids_respond <dbl> 35, 35, 35, 91, 93, 93, 139, 145, 143, 94, 94, 94, 141
```

sounds (a subset)

- **age**: child age in months
- **sound**: a string describing a type of animal sound
- **kids_produce**: the number of parents who answered "yes, my child produces this animal sound"
- **kids_respond**: the number of parents who responded to this question at all

age	sound	kids_produce	kids_respond
8	cockadoodledoo	1	35
8	meow	0	35
8	woof woof	3	35
9	cockadoodledoo	0	91
9	meow	2	93
9	woof woof	2	93

Data types

Vectors

Atomic vectors

Logical

Numeric

Integer

Double

Character

List

NULL



Let's review

Data wrangling with `dplyr`

From DataCamp Chapter 3

- `group_by`
- `summarize`

Adding onto your arsenal of...

- `filter`
- `arrange`
- `mutate`
- `glimpse`
- `distinct`
- `count`
- `tally`
- `pull`
- `top_n`



More on mutate

3 ways to mutate

1. Create a new variable with a specific value
2. Create a new variable based on other variables
3. Change an existing variable

```
sounds %>%  
  mutate(form = "WS")
```

```
# A tibble: 33 x 5  
   age sound          kids_produce kids_respond form  
   <dbl> <chr>          <dbl>      <dbl> <chr>  
1     8 cockadoodledoo         1        35 WS  
2     8 meow                0        35 WS  
3     8 woof woof           3        35 WS  
4     9 cockadoodledoo         0        91 WS  
5     9 meow                2        93 WS  
6     9 woof woof           2        93 WS  
7    10 cockadoodledoo         0       139 WS  
8    10 meow                5       145 WS  
9    10 woof woof           4       143 WS
```

3 ways to mutate

1. Create a new variable with a specific value
2. Create a new variable based on other variables
3. Change an existing variable

```
sounds %>%  
  mutate(prop_produce = kids_produce / kids_respond)
```

```
# A tibble: 33 x 5
```

	age <dbl>	sound <chr>	kids_produce <dbl>	kids_respond <dbl>	prop_produce <dbl>
1	8	cockadoodledoo	1	35	0.0286
2	8	meow	0	35	0
3	8	woof woof	3	35	0.0857
4	9	cockadoodledoo	0	91	0
5	9	meow	2	93	0.0215
6	9	woof woof	2	93	0.0215
7	10	cockadoodledoo	0	139	0
8	10	meow	5	145	0.0345
9	10	woof woof	4	143	0.0280

3 ways to mutate

1. Create a new variable with a specific value
2. Create a new variable based on other variables
3. **Change an existing variable**

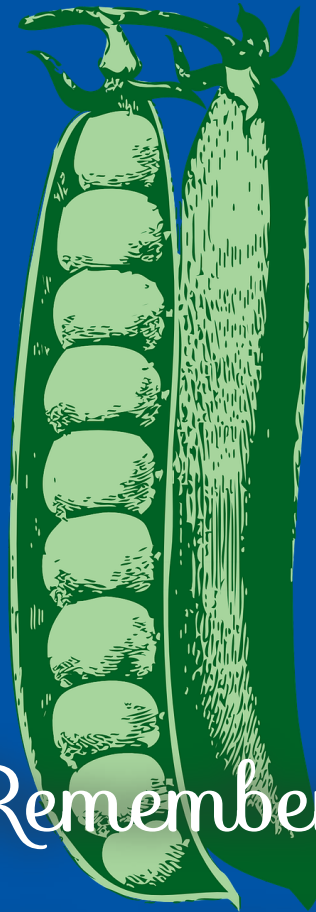
```
sounds %>%  
  mutate(prop_produce = prop_produce * 100)
```

```
# A tibble: 33 x 5
```

	age <dbl>	sound <chr>	kids_produce <dbl>	kids_respond <dbl>	prop_produce <dbl>
1	8	cockadoodledoo	1	35	2.86
2	8	meow	0	35	0
3	8	woof woof	3	35	8.57
4	9	cockadoodledoo	0	91	0
5	9	meow	2	93	2.15
6	9	woof woof	2	93	2.15
7	10	cockadoodledoo	0	139	0
8	10	meow	5	145	3.45
9	10	woof woof	4	143	2.80



Let's review some helpful functions for
mutate + summarize



Remember:

Base R + Tidyverse



First:

Arithmetic

especially useful for mutate

See:

<http://r4ds.had.co.nz/transform.html#mutate-funs>

?Arithmetic

Operator	Description	Usage
+	addition	$x + y$
-	subtraction	$x - y$
*	multiplication	$x * y$
/	division	x / y
^	raised to the power of	$x ^ y$
abs	absolute value	abs(x)
%%/%	integer division	$x \text{ } \%/\% y$
%%	remainder after division	$x \% \% y$

```
5 %/% 2 # 2 goes into 5 two times with ...
```

```
[1] 2
```

```
5 %% 2 # 1 left over
```

```
[1] 1
```



Second: Summaries

*especially useful for summarize
even more useful after a group_by*

See:

<http://r4ds.had.co.nz/transform.html#summarise-funs>

Description	Usage
sum	sum(x)
minimum	min(x)
maximum	max(x)
mean	mean(x)
median	median(x)
standard deviation	sd(x)
variance	var(x)
rank	rank(x)

- All allow for `na.rm` argument to remove `NA` values before summarizing. The default setting for this argument is *always* `na.rm = FALSE`, so if there is one `NA` value the summary will be `NA`.
- See "Maths Functions" in the RStudio Base R Cheatsheet:
<https://github.com/rstudio/cheatsheets/blob/master/base-r.pdf>



"Spent day pondering grayscale vs colourscale
using ggplot"

photo and caption courtesy @alice-data

Today's lab: COLORS

Specifically, discrete colors.

At the end of today's lab, you'll see an extra section on continuous colors.

But first: shape



1



2



3



4



5



6



7



8



9



10



11



12



13



14



15



16



17



18



19



20



21



22



23



24



25

Shapes with color = "hotpink"



1



2



3



4



5



6



7



8



9



10



11



12



13



14



15



16



17



18



19



20



21



22



23



24



25

Shapes with `fill = "gold"`



1



2



3



4



5



6



7



8



9



10



11



12



13



14



15



16



17



18



19



20



21



22



23



24



25

Default shape for `geom_point`

🕵️ Requires spelunking into the dark corners of the `ggplot2` code on [GitHub](#):

```
default_aes = aes(  
  shape = 19, colour = "black", size = 1.5, fill = NA,  
  alpha = NA, stroke = 0.5  
)
```

So, the default for `geom_point(shape = 19)`! This is important to remember: this shape only "understands" the *color* aesthetic, but not the *fill* aesthetic.



R Markdown:

<https://www.markdowntutorial.com>

<https://andrewbtran.github.io/NICAR/2018/workflow/docs/02-rmarkdown.html>

<https://yihui.name/tinytex/> (*install!*)

<https://github.com/rstudio/cheatsheets/blob/master/rmarkdown-2.0.pdf>

https://rmarkdown.rstudio.com/html_document_format.html

https://rmarkdown.rstudio.com/pdf_document_format.html