Cast Column Types

WORKING WITH DATA IN THE TIDYVERSE

Alison Hill
Professor & Data Scientist

Why bother?

The readr package

```
library(readr) # once per work session
```

[1] http://readr.tidyverse.org

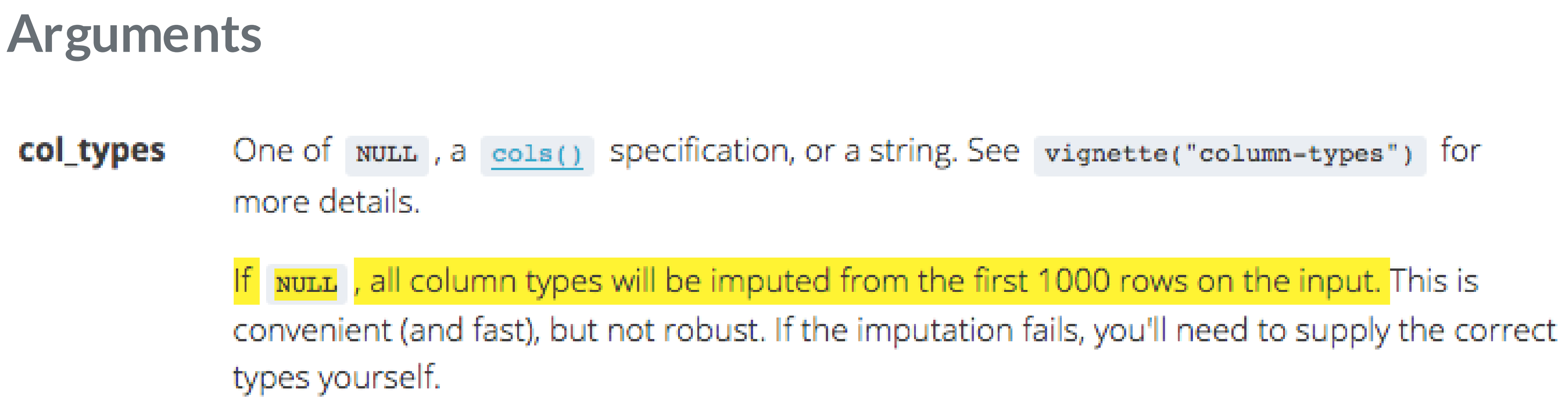
read_csv

```
?read_csv
```

Usage

```
read_csv(file, col_names = TRUE,
              col_types = NULL,
              locale = default_locale(),
              na = c("", "NA"), quoted_na = TRUE,
              quote = "\"", comment = "",
              trim_ws = TRUE,
              skip = 0,
              n_max = Inf,
              guess_max = min(1000, n_max),
              progress = show_progress())
```

The col_types argument



Arguments

col_types One of `NULL`, a `cols()` specification, or a string. See `vignette("column-types")` for more details.

If `NULL`, all column types will be imputed from the first 1000 rows on the input. This is convenient (and fast), but not robust. If the imputation fails, you'll need to supply the correct types yourself.

bakers_tame

```
bakers_tame
```

```
# A tibble: 10 x 6
   series baker        age num_episodes aired_us last_date_uk
    <dbl> <chr>      <dbl>        <dbl> <lgl>    <date>
 1     3. Natasha      36.           1. FALSE    2012-08-14
 2     3. Sarah-Jane   28.           7. FALSE    2012-09-25
 3     3. Cathryn      27.           8. FALSE    2012-10-02
 4     4. Lucy         38.           2. TRUE     2013-08-27
 5     4. Howard       51.           6. TRUE     2013-09-24
 6     4. Beca         31.           9. TRUE     2013-10-15
 7     4. Kimberley    30.          10. TRUE     2013-10-22
 8     5. Enwezor      39.           2. TRUE     2014-08-13
 9     5. Jordan       32.           3. TRUE     2014-08-20
10     5. Iain         31.           4. TRUE     2014-08-27
```

Tame versus raw bakers

```
bakers_tame %>% dplyr::slice(1:4)
```

```
# A tibble: 4 x 6
  series baker        age num_episodes aired_us last_date_uk
   <dbl> <chr>      <dbl>        <dbl> <lgl>    <date>
1     3. Natasha      36.           1. FALSE    2012-08-14
2     3. Sarah-Jane   28.           7. FALSE    2012-09-25
3     3. Cathryn      27.           8. FALSE    2012-10-02
4     4. Lucy         38.           2. TRUE     2013-08-27
```

```
bakers_raw %>% dplyr::slice(1:4)
```

```
# A tibble: 4 x 6
  series baker      age      num_episodes aired_us last_date_uk
   <dbl> <chr>      <chr>           <dbl>    <dbl> <chr>
1     3. Natasha    36 years           1.       0. 14 August 2012
2     3. Sarah-Jane 28 years           7.       0. 25 September 2012
3     3. Cathryn    27 years           8.       0. 2 October 2012
4     4. Lucy       38 years           2.       1. 27 August 2013
```

parse_number

```r
bakers_raw %>% dplyr::slice(1:4)
```

```
# A tibble: 4 x 6
  series baker      age      num_episodes aired_us last_date_uk
   <dbl> <chr>      <chr>           <dbl>    <dbl> <chr>
1     3. Natasha    36 years           1.       0. 14 August 2012
2     3. Sarah-Jane 28 years           7.       0. 25 September 2012
3     3. Cathryn    27 years           8.       0. 2 October 2012
4     4. Lucy       38 years           2.       1. 27 August 2013
```

```r
parse_number("36 years")
```

```
36
```

From parsing to casting

```
parse_number("36 years")
```

```
36
```

```
bakers_tame <- read_csv(file = "bakers.csv",
                        col_types = cols(age = col_number()))
bakers_tame %>% slice(1:4)
```

```
# A tibble: 4 x 6
  series baker        age num_episodes aired_us last_date_uk
   <dbl> <chr>      <dbl>        <dbl> <lgl>    <chr>
1     3. Natasha      36.           1. FALSE    14 August 2012
2     3. Sarah-Jane   28.           7. FALSE    25 September 2012
3     3. Cathryn      27.           8. FALSE    2 October 2012
4     4. Lucy         38.           2. TRUE     27 August 2013
```

parse_date

```
bakers_tame %>% dplyr::slice(1:4)
```

```
# A tibble: 4 x 6
  series baker          age num_episodes aired_us last_date_uk
   <dbl> <chr>        <dbl>        <dbl> <lgl>    <chr>
1      3. Natasha       36.           1. FALSE    14 August 2012
2      3. Sarah-Jane    28.           7. FALSE    25 September 2012
3      3. Cathryn       27.           8. FALSE    2 October 2012
4      4. Lucy          38.           2. TRUE     27 August 2013
```

```
?parse_date
```

Format the day

`parse_datetime()` recognises the following format specifications:

- Year: `"%Y"` (4 digits). `"%y"` (2 digits); 00-69 -> 2000-2069, 70-99 -> 1970-1999.
- Month: `"%m"` (2 digits), `"%b"` (abbreviated name in current locale), `"%B"` (full name in current locale).
- Day: `"%d"` (2 digits), `"%e"` (optional leading space)

```
parse_date("14 August 2012", format = "%d ___ ___")
```

Format the month

`parse_datetime()` recognises the following format specifications:

- Year: `"%Y"` (4 digits). `"%y"` (2 digits); 00-69 -> 2000-2069, 70-99 -> 1970-1999.
- Month: `"%m"` (2 digits), `"%b"` (abbreviated name in current locale), `"%B"` (full name in current locale).
- Day: `"%d"` (2 digits), `"%e"` (optional leading space)

```
parse_date("14 August 2012", format = "%d %B ___")
```

Format the year

```
parse_date("14 August 2012", format = "%d %B %Y")
```

```
"2012-08-14"
```

`parse_datetime()` recognises the following format specifications:

- Year: `"%Y"` (4 digits). `"%y"` (2 digits); 00-69 -> 2000-2069, 70-99 -> 1970-1999.
- Month: `"%m"` (2 digits), `"%b"` (abbreviated name in current locale), `"%B"` (full name in current locale).
- Day: `"%d"` (2 digits), `"%e"` (optional leading space)

Parse & cast `last_date_uk`

```
bakers <- read_csv("bakers.csv",
                   col_types = cols(
                     last_date_uk = col_date(format = "%d %B %Y")))
```

```
# A tibble: 10 x 6
   series baker        age num_episodes aired_us last_date_uk
    <dbl> <chr>      <dbl>        <dbl> <lgl>    <date>
 1     3. Natasha      36.           1. FALSE    2012-08-14
 2     3. Sarah-Jane   28.           7. FALSE    2012-09-25
 3     3. Cathryn      27.           8. FALSE    2012-10-02
 4     4. Lucy         38.           2. TRUE     2013-08-27
 5     4. Howard       51.           6. TRUE     2013-09-24
 6     4. Beca         31.           9. TRUE     2013-10-15
 7     4. Kimberley    30.          10. TRUE     2013-10-22
 8     5. Enwezor      39.           2. TRUE     2014-08-13
 9     5. Jordan       32.           3. TRUE     2014-08-20
10     5. Iain         31.           4. TRUE     2014-08-27
```

Parse functions in readr

Type	dplyr::glimpse()	readr::parse_*()	readr::col_*()
Logical	<lgl>	parse_logical()	col_logical()
Numeric	<int> *or* <dbl>	parse_number()	col_number()
Character	<chr>	parse_character()	col_character()
Factor	<fct>	parse_factor(levels)	col_factor(levels)
Date	<date>	parse_date(format)	col_date(format)

Let's get to work!

WORKING WITH DATA IN THE TIDYVERSE

Recode Values

WORKING WITH DATA IN THE TIDYVERSE

Alison Hill
Professor & Data Scientist

Find-and-replace

```
bakeoff %>%
    distinct(result)
```

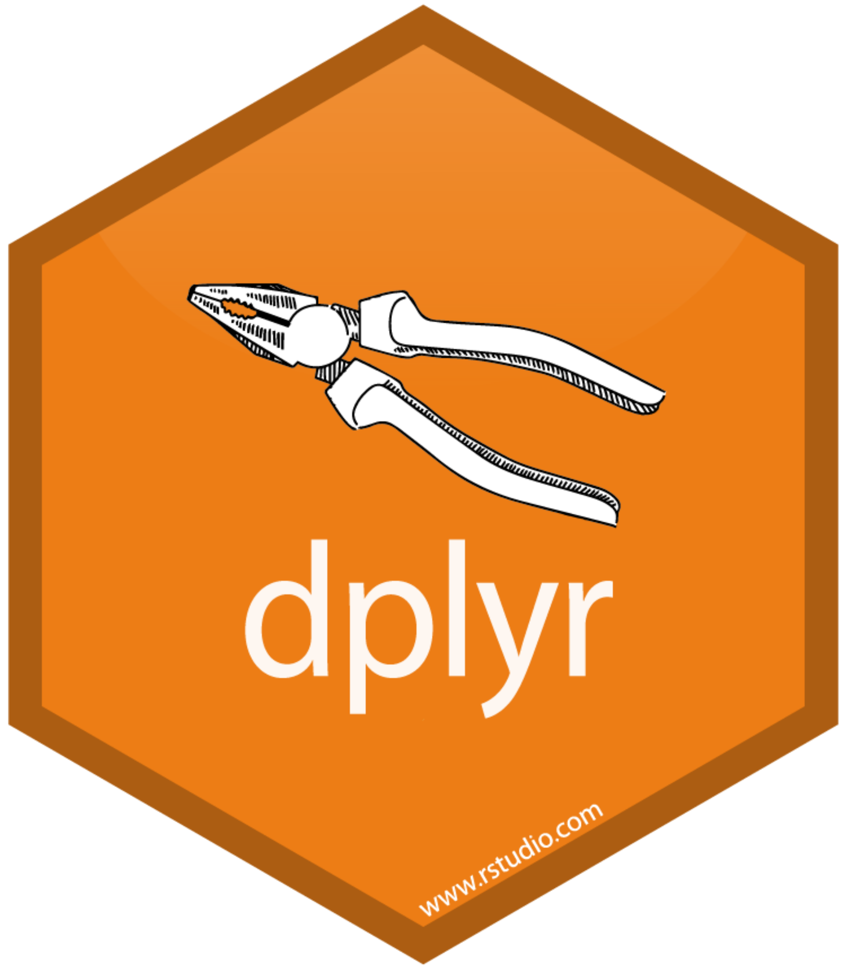
```
# A tibble: 6 x 1
  result
  <fct>
1 IN
2 OUT
3 RUNNER UP
4 WINNER
5 SB
6 LEFT
```

```
bakeoff %>%
    distinct(result)
```

```
# A tibble: 6 x 1
  result
  <fct>
1 IN
2 OUT
3 RUNNER UP
4 WINNER
5 STAR BAKER
6 LEFT
```

The `dplyr` package

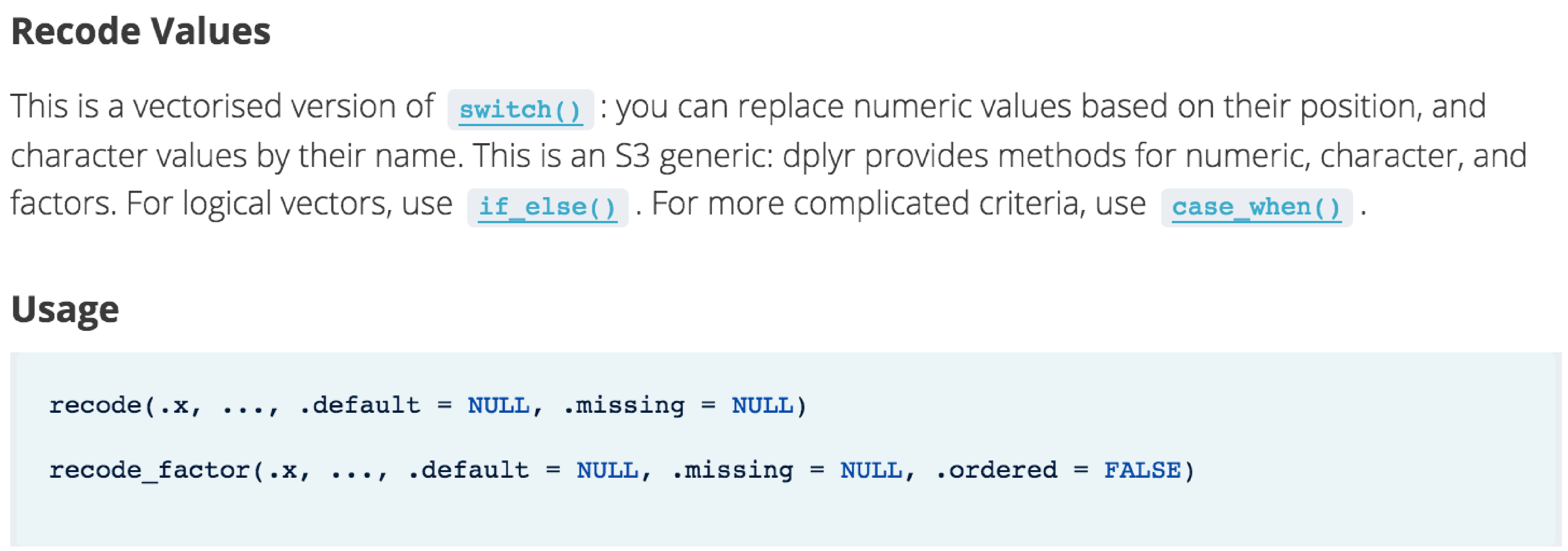
```
library(dplyr) # once per work session
```



[1] http://dplyr.tidyverse.org

Recode: usage

```
?recode
```



Recode Values

This is a vectorised version of `switch()` : you can replace numeric values based on their position, and character values by their name. This is an S3 generic: dplyr provides methods for numeric, character, and factors. For logical vectors, use `if_else()` . For more complicated criteria, use `case_when()` .

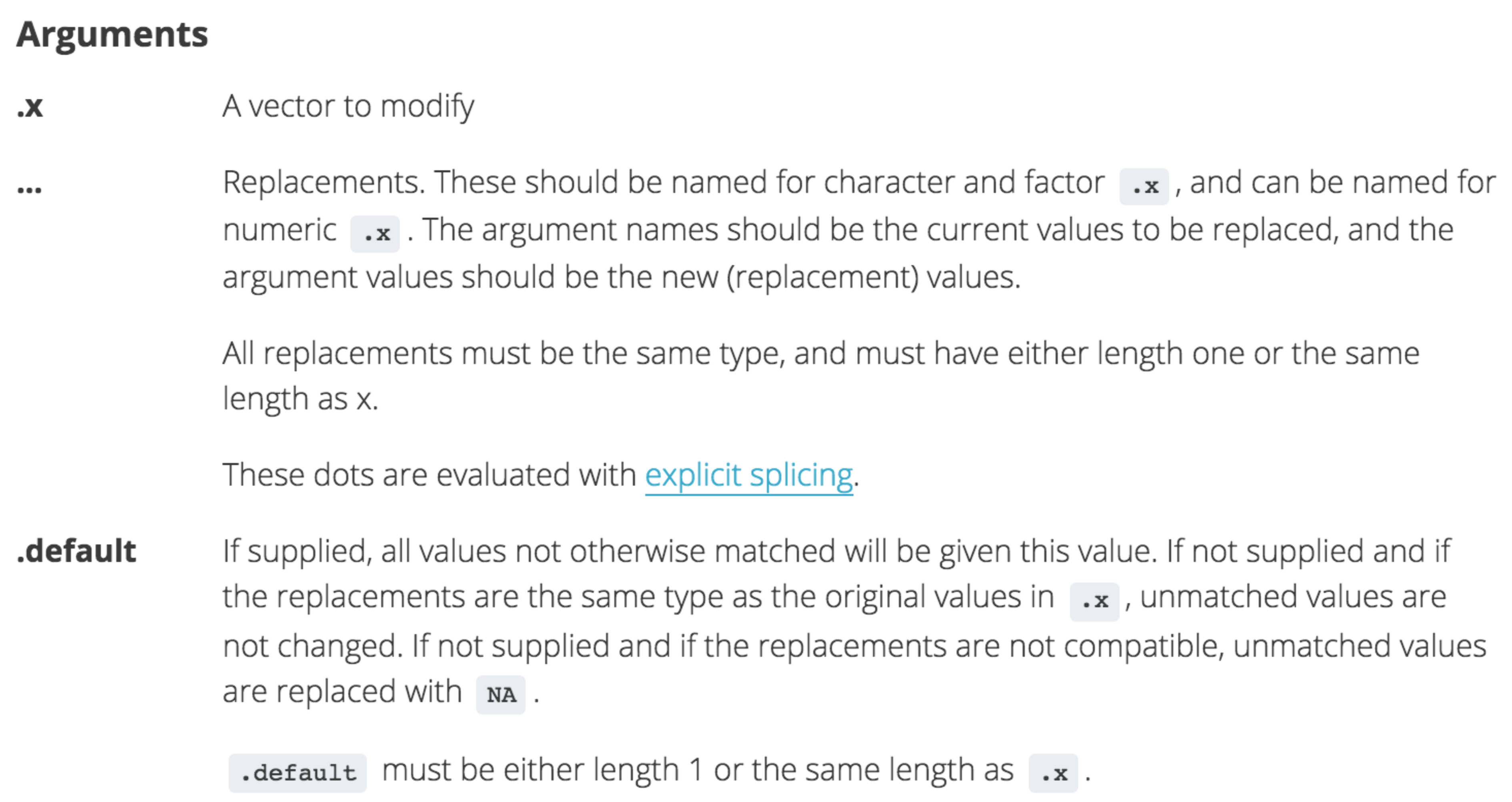
Usage

```
recode(.x, ..., .default = NULL, .missing = NULL)

recode_factor(.x, ..., .default = NULL, .missing = NULL, .ordered = FALSE)
```

Recode: arguments

```
?recode
```



Arguments

.x	A vector to modify
...	Replacements. These should be named for character and factor `.x`, and can be named for numeric `.x`. The argument names should be the current values to be replaced, and the argument values should be the new (replacement) values. All replacements must be the same type, and must have either length one or the same length as x. These dots are evaluated with explicit splicing.
.default	If supplied, all values not otherwise matched will be given this value. If not supplied and if the replacements are the same type as the original values in `.x`, unmatched values are not changed. If not supplied and if the replacements are not compatible, unmatched values are replaced with `NA`. `.default` must be either length 1 or the same length as `.x`.

Youngest bakers

```
young_bakers
```

```
# A tibble: 10 x 4
   baker       age occupation                            student
   <chr>     <dbl> <chr>                                   <dbl>
 1 Flora       19. art gallery assistant                      0.
 2 Julia       21. aviation broker                            0.
 3 Benjamina   23. teaching assistant                         0.
 4 Martha      17. student                                    1.
 5 Jason       19. civil engineering student                  1.
 6 Liam        19. student                                    1.
 7 Ruby        20. history of art and philosophy student      1.
 8 Michael     20. student                                    1.
 9 James       21. medical student                            2.
10 John        23. law student                                2.
```

Recode student

```r
young_bakers %>%
  mutate(stu_label = recode(student, `0` = "other",
                                .default = "student"))
```

```
# A tibble: 10 x 5
   baker       age occupation                            student stu_label
   <chr>     <dbl> <chr>                                   <dbl> <chr>
 1 Flora       19. art gallery assistant                      0. other
 2 Julia       21. aviation broker                            0. other
 3 Benjamina   23. teaching assistant                         0. other
 4 Martha      17. student                                    1. student
 5 Jason       19. civil engineering student                  1. student
 6 Liam        19. student                                    1. student
 7 Ruby        20. history of art and philosophy student      1. student
 8 Michael     20. student                                    1. student
 9 James       21. medical student                            2. student
10 John        23. law student                                2. student
```

Recode with NA

```
young_bakers %>%
  mutate(stu_label = recode(student, `0` = NA_character_,
                                .default = "student"))
```

```
# A tibble: 10 x 5
   baker       age occupation                           student stu_label
   <chr>     <dbl> <chr>                                  <dbl> <chr>
 1 Flora       19. art gallery assistant                     0. NA
 2 Julia       21. aviation broker                           0. NA
 3 Benjamina   23. teaching assistant                        0. NA
 4 Martha      17. student                                   1. student
 5 Jason       19. civil engineering student                 1. student
 6 Liam        19. student                                   1. student
 7 Ruby        20. history of art and philosophy student     1. student
 8 Michael     20. student                                   1. student
 9 James       21. medical student                           2. student
10 John        23. law student                               2. student
```

Recode multiple values

```r
young_bakers %>%
  mutate(stu_label = recode(student, `0` = NA_character_,
                           `2` = "law/med",
                           .default = "student"))
```

```
# A tibble: 10 x 5
   baker       age occupation                          student stu_label
   <chr>     <dbl> <chr>                                 <dbl> <chr>
 1 Flora       19. art gallery assistant                    0. NA
 2 Julia       21. aviation broker                          0. NA
 3 Benjamina   23. teaching assistant                       0. NA
 4 Martha      17. student                                  1. student
 5 Jason       19. civil engineering student                1. student
 6 Liam        19. student                                  1. student
 7 Ruby        20. history of art and philosophy student    1. student
 8 Michael     20. student                                  1. student
 9 James       21. medical student                          2. law/med
10 John        23. law student                              2. law/med
```

Convert to NA only

```
young_bakers %>%
  mutate(student = na_if(student, 0))
```

```
# A tibble: 10 x 4
   baker       age occupation                            student
   <chr>     <dbl> <chr>                                   <dbl>
 1 Flora       19. art gallery assistant                     NA
 2 Julia       21. aviation broker                           NA
 3 Benjamina   23. teaching assistant                        NA
 4 Martha      17. student                                    1.
 5 Jason       19. civil engineering student                  1.
 6 Liam        19. student                                    1.
 7 Ruby        20. history of art and philosophy student      1.
 8 Michael     20. student                                    1.
 9 James       21. medical student                            2.
10 John        23. law student                                2.
```

Let's practice!

WORKING WITH DATA IN THE TIDYVERSE

Select Variables

WORKING WITH DATA IN THE TIDYVERSE

Alison Hill
Professor & Data Scientist

Youngest bakers

```
young_bakers2
```

```
# A tibble: 5 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
4 Ruby           3.               2.            0.               1.
5 John           1.               1.            1.               0.
```

Select

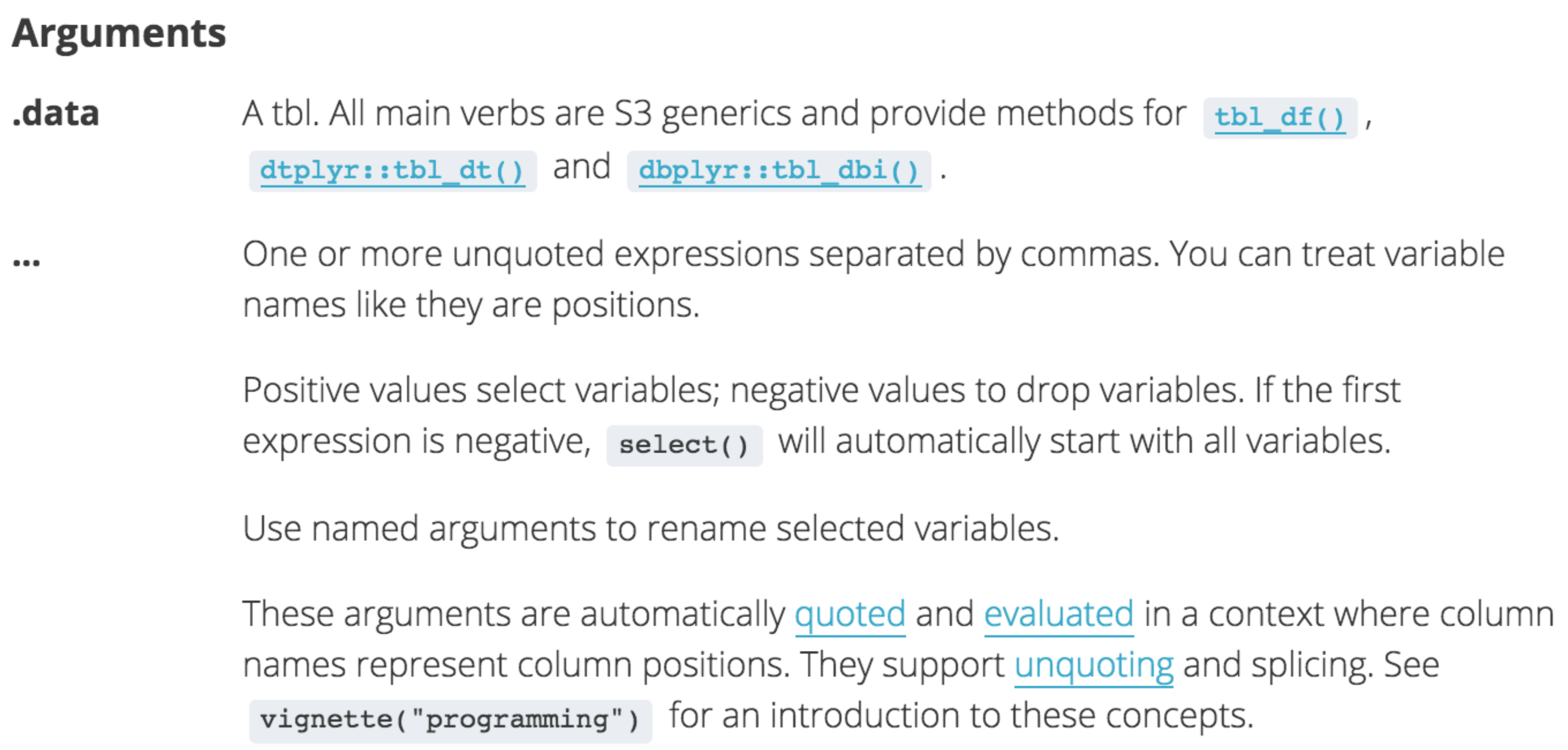
```
?select
```

Usage

```
select(.data, ...)
```

Select: arguments

```
?select
```



Arguments

.data	A tbl. All main verbs are S3 generics and provide methods for `tbl_df()`, `dtplyr::tbl_dt()` and `dbplyr::tbl_dbi()`.
...	One or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, `select()` will automatically start with all variables. Use named arguments to rename selected variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing. See `vignette("programming")` for an introduction to these concepts.

```
young_bakers2
```

```
# A tibble: 5 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
4 Ruby           3.               2.            0.               1.
5 John           1.               1.            1.               0.
```

```
young_bakers2 %>%
   select(baker, series_winner)
```

```
# A tibble: 5 x 2
  baker  series_winner
  <chr>          <dbl>
1 Martha            0.
2 Flora             0.
3 Jason             0.
4 Ruby              0.
5 John              1.
```

Select a range of variables

```
young_bakers2
```

```
# A tibble: 3 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
```

```
young_bakers2 %>%
   select(baker:technical_winner)
```

```
# A tibble: 3 x 3
  baker  star_baker technical_winner
  <chr>       <dbl>            <dbl>
1 Martha         0.               2.
2 Flora          0.               1.
3 Jason          2.               1.
```

Drop variables

```
young_bakers2
```

```
# A tibble: 3 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
```

```
young_bakers2 %>%
   select(-technical_winner)
```

```
# A tibble: 3 x 4
  baker  star_baker series_winner series_runner_up
  <chr>       <dbl>         <dbl>            <dbl>
1 Martha         0.            0.               0.
2 Flora          0.            0.               0.
3 Jason          2.            0.               0.
```

Select helpers: starts_with()

```
young_bakers2
```

```
# A tibble: 3 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
```

```
young_bakers2 %>%
   select(baker, starts_with("series"))
```

```
# A tibble: 3 x 3
  baker  series_winner series_runner_up
  <chr>          <dbl>            <dbl>
1 Martha            0.               0.
2 Flora             0.               0.
3 Jason             0.               0.
```

Select helper: ends_with()

```
young_bakers2
```

```
# A tibble: 3 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
```

```
young_bakers2 %>%
   select(ends_with("winner"), baker)
```

```
# A tibble: 3 x 3
  technical_winner series_winner baker
             <dbl>         <dbl> <chr>
1               2.            0. Martha
2               1.            0. Flora
3               1.            0. Jason
```

Select helper: contains()

```
young_bakers2
```

```
# A tibble: 3 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
```

```
young_bakers2 %>%
   select(contains("bake"))
```

```
# A tibble: 3 x 2
  baker  star_baker
  <chr>       <dbl>
1 Martha         0.
2 Flora          0.
3 Jason          2.
```

Combine helper functions

```
young_bakers2
```

```
# A tibble: 3 x 5
  baker  star_baker technical_winner series_winner series_runner_up
  <chr>       <dbl>            <dbl>         <dbl>            <dbl>
1 Martha         0.               2.            0.               0.
2 Flora          0.               1.            0.               0.
3 Jason          2.               1.            0.               0.
```

```
young_bakers2 %>%
   select(contains("bake"), starts_with("series"))
```

```
# A tibble: 3 x 4
  baker  star_baker series_winner series_runner_up
  <chr>       <dbl>         <dbl>            <dbl>
1 Martha         0.            0.               0.
2 Flora          0.            0.               0.
3 Jason          2.            0.               0.
```

Filter versus select

```
young_bakers2 %>%
   filter(series_winner == 1 | series_runner_up == 1)
```

```
# A tibble: 2 x 5
  baker star_baker technical_winner series_winner series_runner_up
  <chr>      <dbl>            <dbl>         <dbl>            <dbl>
1 Ruby          3.               2.            0.               1.
2 John          1.               1.            1.               0.
```

```
young_bakers2 %>%
   select(baker, starts_with("series"))
```

```
# A tibble: 2 x 3
  baker  series_winner series_runner_up
    <chr>        <dbl>            <dbl>
1 Martha            0.               0.
2 Flora             0.               0.
```

Let's practice!

WORKING WITH DATA IN THE TIDYVERSE

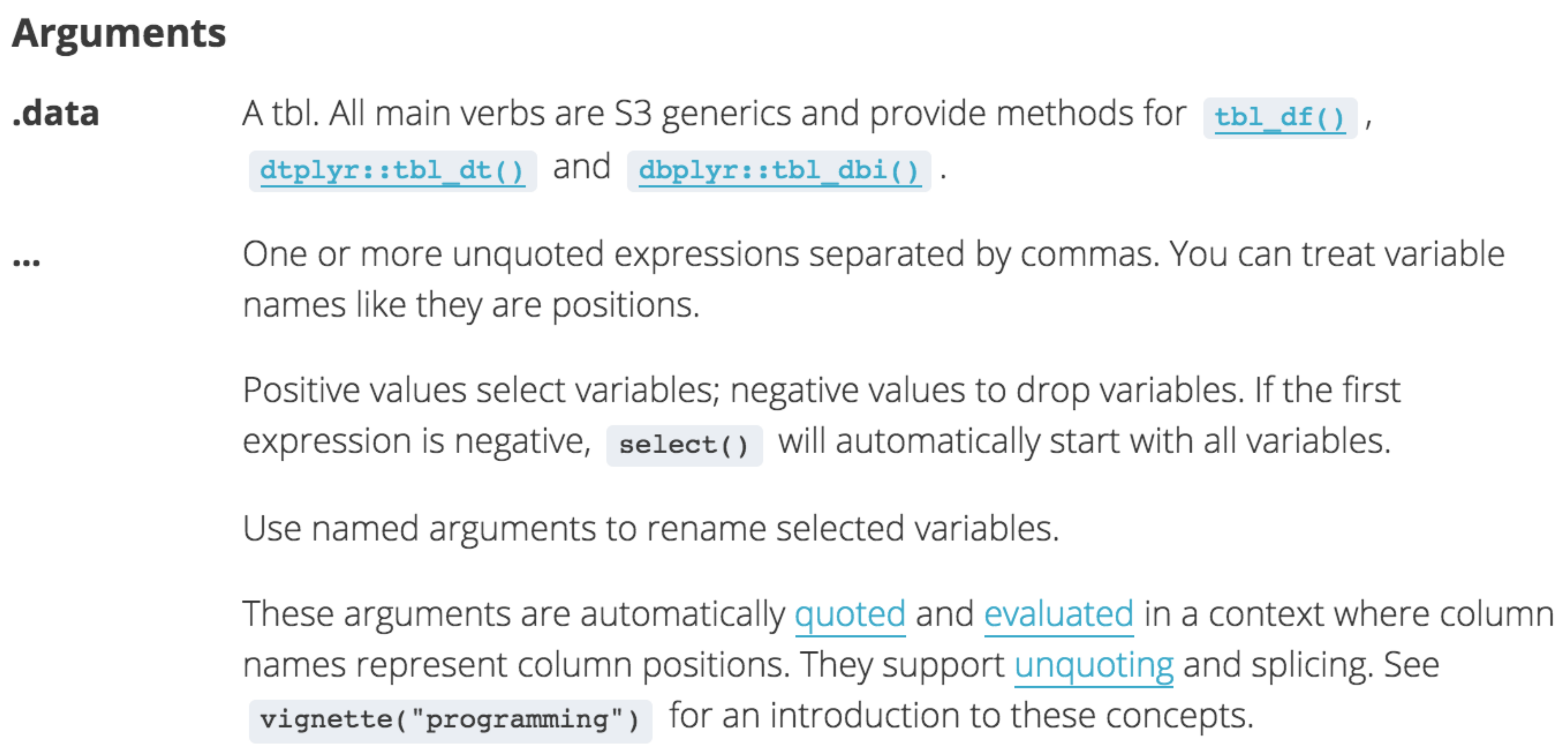
Tame Variable Names

WORKING WITH DATA IN THE TIDYVERSE

Alison Hill
Professor & Data Scientist

Select: arguments

```
?select
```



Arguments

.data	A tbl. All main verbs are S3 generics and provide methods for `tbl_df()`, `dtplyr::tbl_dt()` and `dbplyr::tbl_dbi()`.
...	One or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, `select()` will automatically start with all variables. Use named arguments to rename selected variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing. See `vignette("programming")` for an introduction to these concepts.

Select & change variable names

```
young_bakers3
```

```
# A tibble: 3 x 6
  baker     student   age  tre1  tre2  tre3
  <chr>       <dbl> <dbl> <dbl> <dbl> <dbl>
1 Ruby           1.   20.   12.    3.    3.
2 Julia          0.   21.    3.    4.    2.
3 Benjamina      0.   23.    6.    3.    6.
```

```
young_bakers3 %>%
   select(baker, tech_1 = tre1)
```

```
# A tibble: 3 x 2
  baker     tech_1
  <chr>      <dbl>
1 Ruby         12.
2 Julia         3.
3 Benjamina     6.
```

Select & change variable names

```
young_bakers3
```

```
# A tibble: 3 x 6
  baker     student   age  tre1  tre2  tre3
  <chr>       <dbl> <dbl> <dbl> <dbl> <dbl>
1 Ruby           1.   20.   12.    3.    3.
2 Julia          0.   21.    3.    4.    2.
3 Benjamina      0.   23.    6.    3.    6.
```

```
young_bakers3 %>%
   select(baker, tech_ = tre1:tre3)
```

```
# A tibble: 3 x 4
  baker     tech_1 tech_2 tech_3
  <chr>      <dbl>  <dbl>  <dbl>
1 Ruby         12.     3.     3.
2 Julia         3.     4.     2.
3 Benjamina     6.     3.     6.
```

Change names for a variable range

```r
young_bakers3
```

```
# A tibble: 3 x 9
  baker       age student  tre1 rse1   tre2 rse2   tre3 rse3
  <chr>     <dbl>   <dbl> <dbl> <chr> <dbl> <chr> <dbl> <chr>
1 Ruby        20.      1.   12. IN       3. SB       3. IN
2 Julia       21.      0.    3. IN       4. IN       2. SB
3 Benjamina   23.      0.    6. IN       3. IN       6. IN
```

```r
young_bakers3 %>%
   select(baker, tech_ = starts_with("tr"),
          result_ = starts_with("rs"))
```

```
# A tibble: 3 x 7
  baker     tech_1 tech_2 tech_3 result_1 result_2 result_3
  <chr>      <dbl>  <dbl>  <dbl> <chr>    <chr>    <chr>
1 Ruby         12.     3.     3. IN       SB       IN
2 Julia         3.     4.     2. IN       IN       SB
3 Benjamina     6.     3.     6. IN       IN       IN
```

Change names without reordering

```
young_bakers3
```

```
# A tibble: 3 x 9
  baker       age student  tre1 rse1   tre2 rse2   tre3 rse3
  <chr>     <dbl>   <dbl> <dbl> <chr> <dbl> <chr> <dbl> <chr>
1 Ruby        20.      1.   12. IN       3. SB       3. IN
2 Julia       21.      0.    3. IN       4. IN       2. SB
3 Benjamina   23.      0.    6. IN       3. IN       6. IN
```

```
young_bakers3 %>%
   rename(tech_1 = t_first, result_1 = r_first)
```

```
# A tibble: 3 x 9
  baker       age student tech_1 result_1  tre2 rse2   tre3 rse3
  <chr>     <dbl>   <dbl>  <dbl> <chr>    <dbl> <chr> <dbl> <chr>
1 Ruby        20.      1.    12. IN          3. SB       3. IN
2 Julia       21.      0.     3. IN          4. IN       2. SB
3 Benjamina   23.      0.     6. IN          3. IN       6. IN
```

Select & change names without reordering

```
young_bakers3
```

```
# A tibble: 3 x 9
  baker       age student  tre1 rse1   tre2 rse2   tre3 rse3
  <chr>     <dbl>   <dbl> <dbl> <chr> <dbl> <chr> <dbl> <chr>
1 Ruby        20.      1.   12. IN       3. SB       3. IN
2 Julia       21.      0.    3. IN       4. IN       2. SB
3 Benjamina   23.      0.    6. IN       3. IN       6. IN
```

```
young_bakers3 %>%
   select(everything(), tech_ = starts_with("tr"),
          result_ = starts_with("rs"))
```

```
# A tibble: 3 x 9
  baker       age student tech_1 result_1 tech_2 result_2 tech_3 result_3
  <chr>     <dbl>   <dbl>  <dbl> <chr>     <dbl> <chr>     <dbl> <chr>
1 Ruby        20.      1.    12. IN           3. SB           3. IN
2 Julia       21.      0.     3. IN           4. IN           2. SB
3 Benjamina   23.      0.     6. IN           3. IN           6. IN
```

What's in a name?

```
i_use_snake_case
```

```
otherPeopleUseCamelCase
```

```
some.people.use.periods
```

```
And_aFew.People_RENOUNCEconvention
```

[1] R for Data Science (http://r4ds.had.co.nz/workflow [2] basics.html#whats [3] in [4] a [5] name)

Clean all variable names

```
young_bakers3
```

```
# A tibble: 4 x 9
  Baker       Age `Student #` `Tr E1` `Rs E1` `Tr E2` `Rs E2` `Tr E3` `Rs E3`
  <chr>     <dbl>       <dbl>   <dbl> <chr>     <dbl> <chr>     <dbl> <chr>
1 Ruby        20.          1.     12. IN           3. SB           3. IN
2 Julia       21.          0.      3. IN           4. IN           2. SB
3 Benjamina   23.          0.      6. IN           3. IN           6. IN
```

```
library(janitor)
young_bakers3 %>%
   clean_names()
```

```
# A tibble: 4 x 9
  baker       age student_number tr_e1 rs_e1 tr_e2 rs_e2 tr_e3 rs_e3
  <chr>     <dbl>          <dbl> <dbl> <chr> <dbl> <chr> <dbl> <chr>
1 Ruby        20.             1.   12. IN       3. SB       3. IN
2 Julia       21.             0.    3. IN       4. IN       2. SB
3 Benjamina   23.             0.    6. IN       3. IN       6. IN
```

Let's practice!

WORKING WITH DATA IN THE TIDYVERSE