What's new in dbplyr 2.0.0?

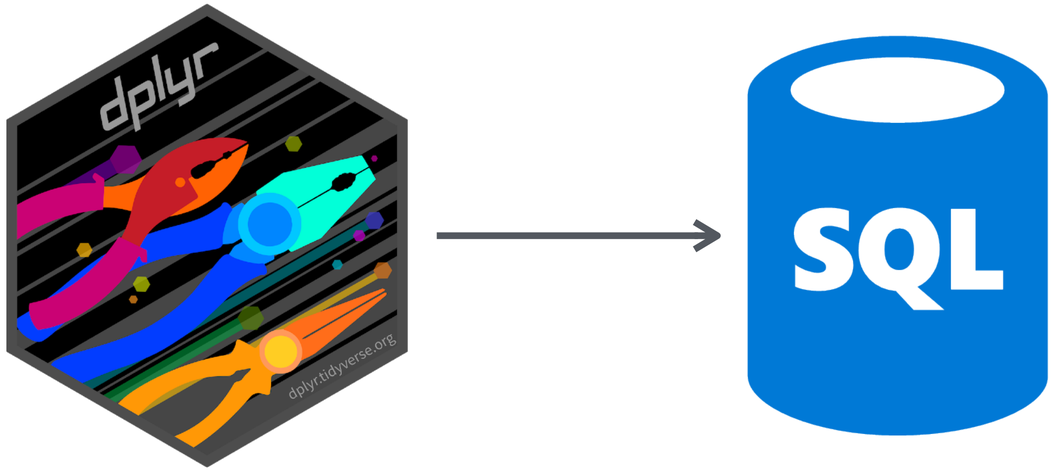
Snazzy new logo by Allison Horst

Mara Averick (@dataandme)
Tidyverse Developer Advocate, RStudio

What is dbplyr?

{dbplyr} is the database backend for **{dplyr}**.

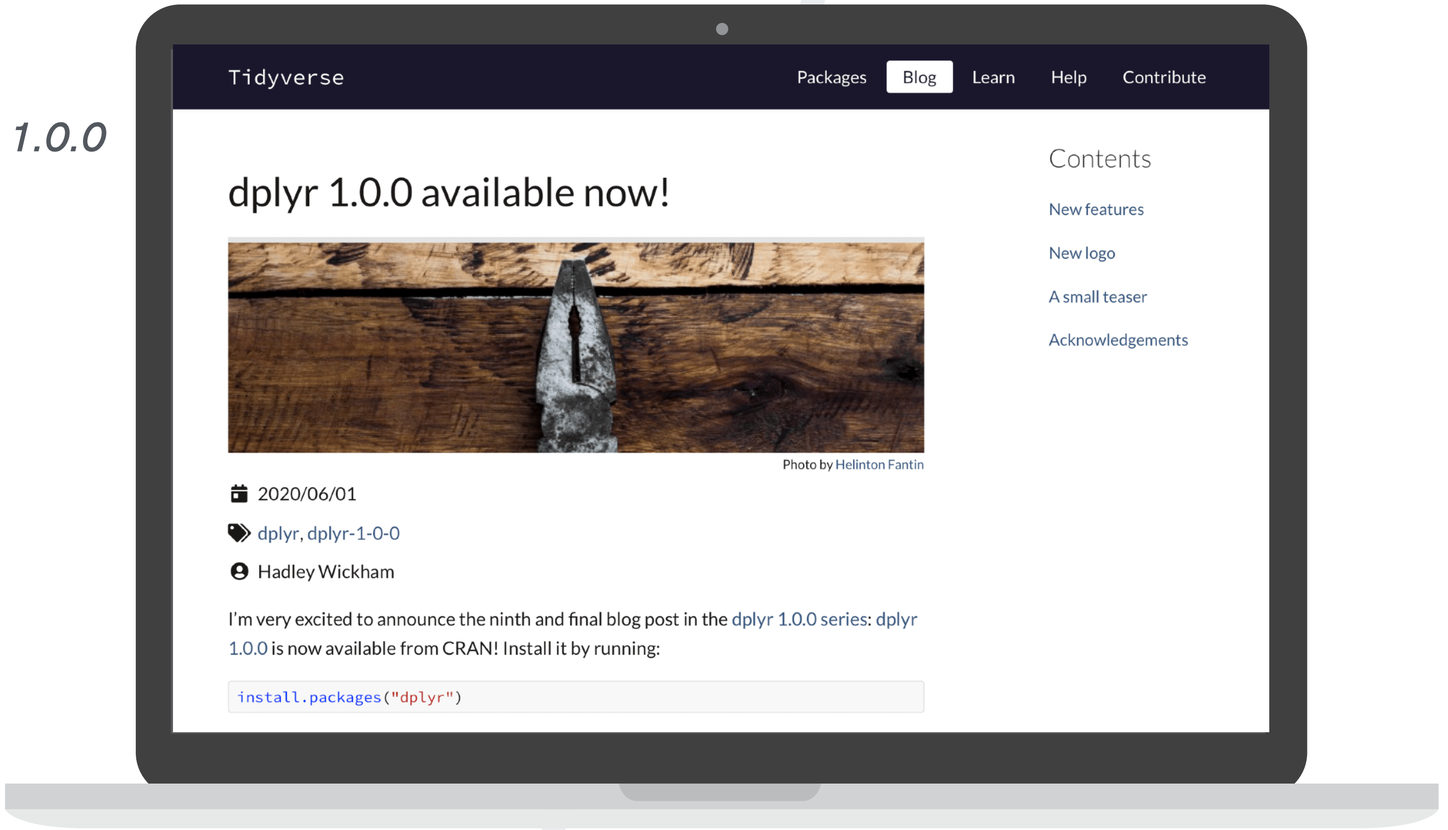
It allows you to use remote database tables as if they are in-memory data frames by converting **dplyr** code into **SQL**.



source: https://dbplyr.tidyverse.org

What's new?

Now compatible with features from dplyr 1.0.0

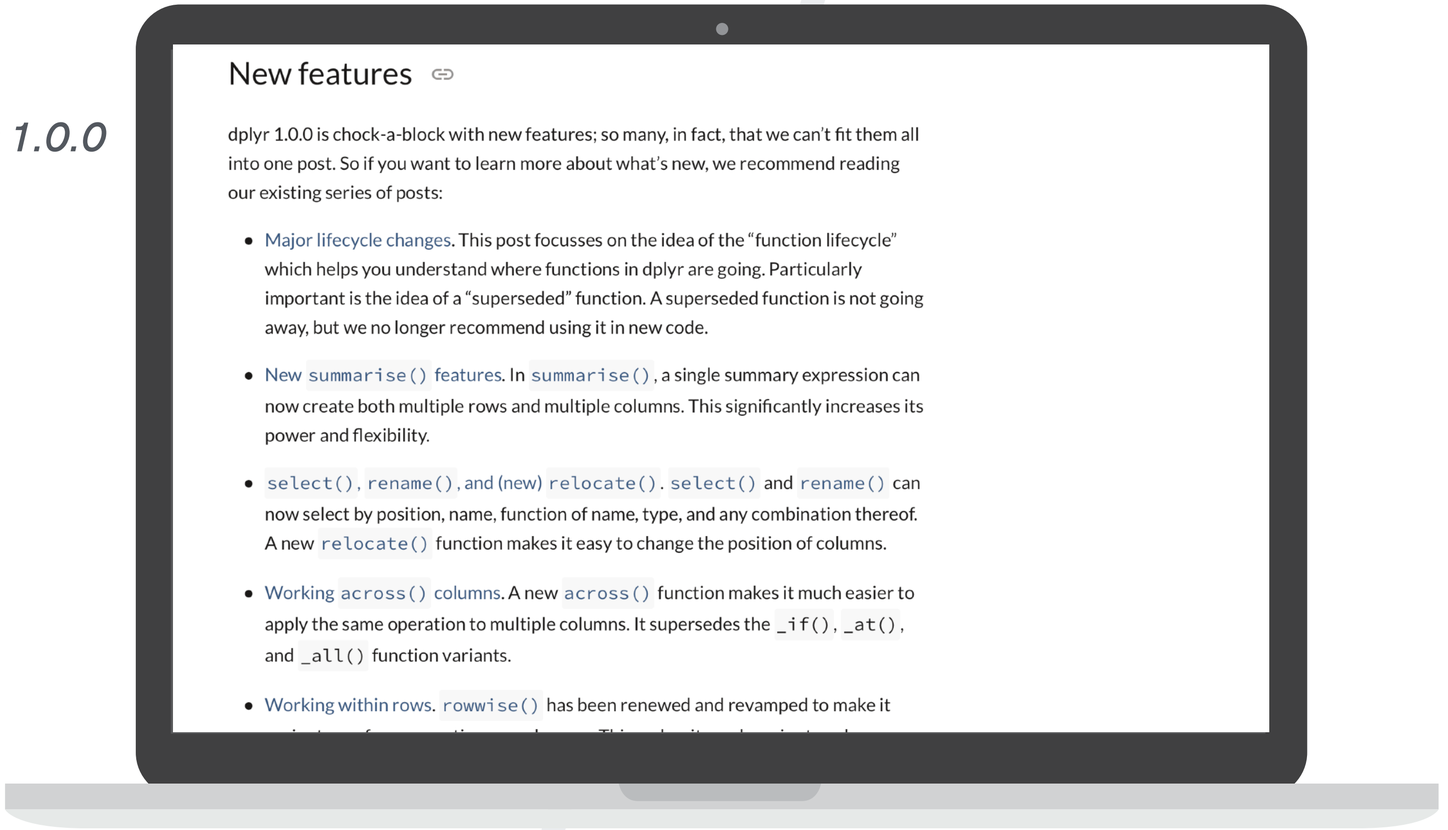


https://www.tidyverse.org/blog/2020/06/dplyr-1-0-0

What's new?

there are a lot of them

Now compatible with features from dplyr 1.0.0



New features

dplyr 1.0.0 is chock-a-block with new features; so many, in fact, that we can't fit them all into one post. So if you want to learn more about what's new, we recommend reading our existing series of posts:

- Major lifecycle changes. This post focusses on the idea of the "function lifecycle" which helps you understand where functions in dplyr are going. Particularly important is the idea of a "superseded" function. A superseded function is not going away, but we no longer recommend using it in new code.
- New `summarise()` features. In `summarise()`, a single summary expression can now create both multiple rows and multiple columns. This significantly increases its power and flexibility.
- `select()`, `rename()`, and (new) `relocate()`. `select()` and `rename()` can now select by position, name, function of name, type, and any combination thereof. A new `relocate()` function makes it easy to change the position of columns.
- Working `across()` columns. A new `across()` function makes it much easier to apply the same operation to multiple columns. It supersedes the `_if()`, `_at()`, and `_all()` function variants.
- Working within rows. `rowwise()` has been renewed and revamped to make it

https://www.tidyverse.org/blog/2020/06/dplyr-1-0-0

dplyr 1.0.0 compatibility

- ✓ Apply operation to multiple cols with **`across()`**
- ✓ Use tidyselect syntax with **`rename()`** and **`select()`**
- ✓ Move columns around with **`relocate()`**

dbplyr 2.0.0 https://www.tidyverse.org/blog/2020/11/dbplyr-2-0-0

dplyr 1.0.0 compatibility

- ✓ Programmatically rename cols with **`rename_with()`**
- ✓ Get rows by position with **`slice_min()`** and **`_max()`**
- ✓ Randomly select rows with **`slice_sample()`**

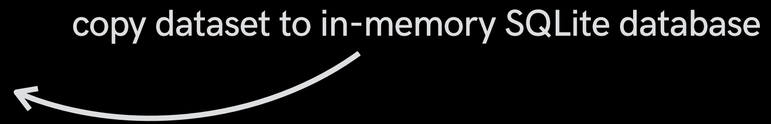
dbplyr 2.0.0 https://www.tidyverse.org/blog/2020/11/dbplyr-2-0-0

```r
library(dbplyr)
library(dplyr, warn.conflicts = FALSE)
data("gtcars", package = "gt")
glimpse(gtcars)
```

```
#> Rows: 47
#> Columns: 15
#> $ mfr         <chr> "Ford", "Ferrari", "Ferrari", "Ferrari", "Ferrari", "Ferrari", "Ferr…
#> $ model       <chr> "GT", "458 Speciale", "458 Spider", "458 Italia", "488 GTB", "Califo…
#> $ year        <dbl> 2017, 2015, 2015, 2014, 2016, 2015, 2017, 2015, 2015, 2015, 2017, 20…
#> $ trim        <chr> "Base Coupe", "Base Coupe", "Base", "Base Coupe", "Base Coupe", "Bas…
#> $ bdy_style   <chr> "coupe", "coupe", "convertible", "coupe", "coupe", "convertible", "c…
#> $ hp          <dbl> 647, 597, 562, 562, 661, 553, 680, 652, 731, 949, 573, 545, 700, 610…
#> $ hp_rpm      <dbl> 6250, 9000, 9000, 9000, 8000, 7500, 8250, 8000, 8250, 9000, 6500, 64…
#> $ trq         <dbl> 550, 398, 398, 398, 561, 557, 514, 504, 509, 664, 476, 436, 507, 413…
#> $ trq_rpm     <dbl> 5900, 6000, 6000, 6000, 3000, 4750, 5750, 6000, 6000, 6750, 2000, 32…
#> $ mpg_c       <dbl> 11, 13, 13, 13, 15, 16, 12, 11, 11, 12, 21, 16, 11, 16, 12, 15, 13, …
#> $ mpg_h       <dbl> 18, 17, 17, 17, 22, 23, 17, 16, 16, 16, 22, 22, 18, 20, 20, 25, 21, …
#> $ drivetrain  <chr> "rwd", "rwd", "rwd", "rwd", "rwd", "rwd", "awd", "awd", "rwd", "rwd"…
#> $ trsmn       <chr> "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a", "9a", "6…
#> $ ctry_origin <chr> "United States", "Italy", "Italy", "Italy", "Italy", "Italy", "Italy…
#> $ msrp        <dbl> 447000, 291744, 263553, 233509, 245400, 198973, 298000, 295000, 3199…
```

https://dbplyr.tidyverse.org/articles/dbplyr.html

```
con ← DBI::dbConnect(RSQLite::SQLite(), ":memory:")
copy_to(con, gtcars)
```



https://dbplyr.tidyverse.org/articles/dbplyr.html

```r
con ← DBI::dbConnect(RSQLite::SQLite(), ":memory:")
copy_to(con, gtcars)

gtcars2 ← tbl(con, "gtcars")
gtcars2
```

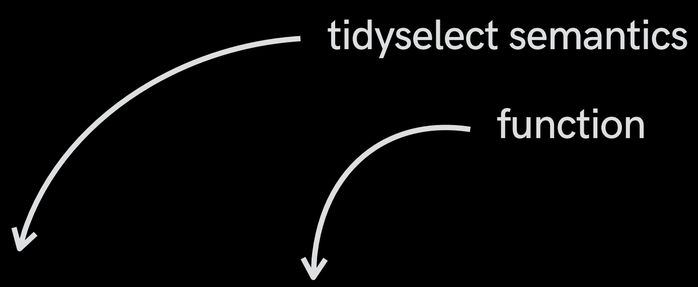
copy dataset to in-memory SQLite database

see the first few rows of the remote table

```
#> # Source:   table<gtcars> [?? x 15]
#> # Database: sqlite 3.33.0 [:memory:]
#>    mfr   model  year trim  bdy_style    hp hp_rpm   trq trq_rpm mpg_c mpg_h
#>    <chr> <chr> <dbl> <chr> <chr>     <dbl>  <dbl> <dbl>   <dbl> <dbl> <dbl>
#>  1 Ford  GT     2017 Base… coupe       647   6250   550    5900    11    18
#>  2 Ferr… 458 …  2015 Base… coupe       597   9000   398    6000    13    17
#>  3 Ferr… 458 …  2015 Base  converti…   562   9000   398    6000    13    17
#>  4 Ferr… 458 …  2014 Base… coupe       562   9000   398    6000    13    17
#>  5 Ferr… 488 …  2016 Base… coupe       661   8000   561    3000    15    22
#>  6 Ferr… Cali…  2015 Base… converti…   553   7500   557    4750    16    23
#>  7 Ferr… GTC4…  2017 Base… coupe       680   8250   514    5750    12    17
#>  8 Ferr… FF     2015 Base… coupe       652   8000   504    6000    11    16
#>  9 Ferr… F12B…  2015 Base… coupe       731   8250   509    6000    11    16
#> 10 Ferr… LaFe…  2015 Base… coupe       949   9000   664    6750    12    16
#> # … with more rows, and 4 more variables: drivetrain <chr>, trsmn <chr>,
#> #   ctry_origin <chr>, msrp <dbl>
```

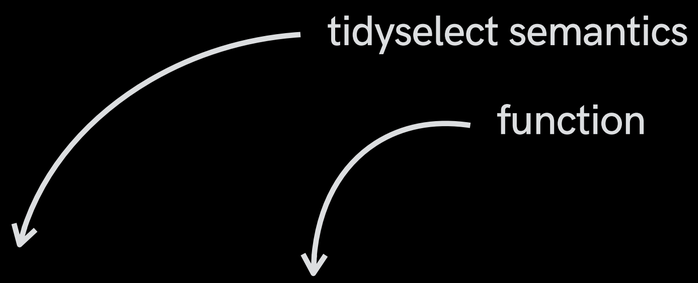
https://dbplyr.tidyverse.org/articles/dbplyr.html

across()




```
# generate the sql code
mpg_summary ← gtcars2 %>%
  group_by(mfr) %>%
  summarise(across(starts_with("mpg"), mean, na.rm = TRUE)) %>%
  arrange(desc(mpg_h))
```

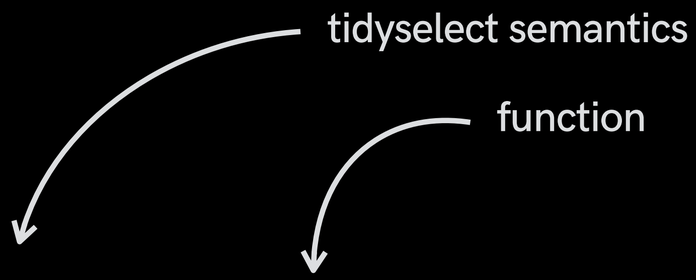
across()




```
# generate the sql code
mpg_summary ← gtcars2 %>%
  group_by(mfr) %>%
  summarise(across(starts_with("mpg"), mean, na.rm = TRUE)) %>%
  arrange(desc(mpg_h))

# preview the query with `show_query()
mpg_summary %>% show_query()
#> <SQL>
#> SELECT `mfr`, AVG(`mpg_c`) AS `mpg_c`, AVG(`mpg_h`) AS `mpg_h`
#> FROM `gtcars`
#> GROUP BY `mfr`
#> ORDER BY `mpg_h` DESC
```

across()

```
# generate the sql code
mpg_summary ← gtcars2 %>%
  group_by(mfr) %>%
  summarise(across(starts_with("mpg"), mean, na.rm = TRUE)) %>%
  arrange(desc(mpg_h))

# preview the query with `show_query()
mpg_summary %>% show_query()
#> <SQL>
#> SELECT `mfr`, AVG(`mpg_c`) AS `mpg_c`, AVG(`mpg_h`) AS `mpg_h`
#> FROM `gtcars`
#> GROUP BY `mfr`
#> ORDER BY `mpg_h` DESC
```



```
# execute with collect()
mpg_summary %>% collect()
  #> # A tibble: 19 x 3
  #>    mfr           mpg_c mpg_h
  #>    <chr>         <dbl> <dbl>
  #>  1 Porsche        19.8  28.2
  #>  2 BMW            19    25.4
  #>  3 Bentley        15    25
  #>  4 Audi           15.2  24.8
  #>  5 Mercedes-Benz  18    24.5
  #>  6 Lotus          16    24
  #>  7 Jaguar         16    24
  #>  8 McLaren        16    23
  #>  9 Maserati       15.3  22.7
  #> 10 Nissan         16    22
  #> 11 Chevrolet      15    22
  #> 12 Acura          21    22
  #> 13 Aston Martin   13.8  20.5
  #> 14 Rolls-Royce    12.5  20
  #> 15 Lamborghini    13    19.3
  #> 16 Dodge          12    19
  #> 17 Ford           11    18
  #> 18 Ferrari        12.9  17.9
  #> 19 Tesla          NA    NA
```

`rename()` & `select()`

With **tidyselect** syntax you can select by:

- Position: `df %>% select(1:3)`
- Name: `df %>% select(c(a, b, j)`
- Function of name: `df %>% select(starts_with("x"))`
- Combinations using Boolean operators:

```
df %>% select(starts_with("x") | ends_with("2"))
```

Learn more: https://www.tidyverse.org/blog/2020/03/dplyr-1-0-0-select-rename-relocate/

rename(), select(), relocate()

```
# lazy frame is an easy way to just preview the SQL queries
lf ← lazy_frame(gtcars)

# relocate lets you move columns around
lf %>% relocate("year") %>%
  select(1:3) # select the first three cols
#> <SQL>
#> SELECT `year`, `mfr`, `model`
#> FROM `df`
```

Learn more: https://www.tidyverse.org/blog/2020/03/dplyr-1-0-0-select-rename-relocate/

rename_with()

```
# programmatically rename columns
lf %>% rename_with(toupper)
#> <SQL>
#> SELECT `mfr` AS `MFR`, `model` AS `MODEL`, `year` AS `YEAR`, `trim` AS `TRIM`,
`bdy_style` AS `BDY_STYLE`, `hp` AS `HP`, `hp_rpm` AS `HP_RPM`, `trq` AS `TRQ`,
`trq_rpm` AS `TRQ_RPM`, `mpg_c` AS `MPG_C`, `mpg_h` AS `MPG_H`, `drivetrain` AS
`DRIVETRAIN`, `trsmn` AS `TRSMN`, `ctry_origin` AS `CTRY_ORIGIN`, `msrp` AS
`MSRP`
#> FROM `df`
```

slice_*()

```
# slice_* functions ungrouped

lf %>% #lf to view the SQL
  select(c(1:5, "msrp")) %>%
  slice_max(msrp, n = 5)
#> <SQL>
#> SELECT `mfr`, `model`, `year`, `trim`, `bdy_style`, `msrp`
#> FROM (SELECT `mfr`, `model`, `year`, `trim`, `bdy_style`, `msrp`, RANK() OVER
(ORDER BY `msrp` DESC) AS `q01`
#> FROM (SELECT `mfr`, `model`, `year`, `trim`, `bdy_style`, `msrp`
#> FROM `df`) `q01`) `q02`
#> WHERE (`q01` ⩽ 5)
```

slice_*()

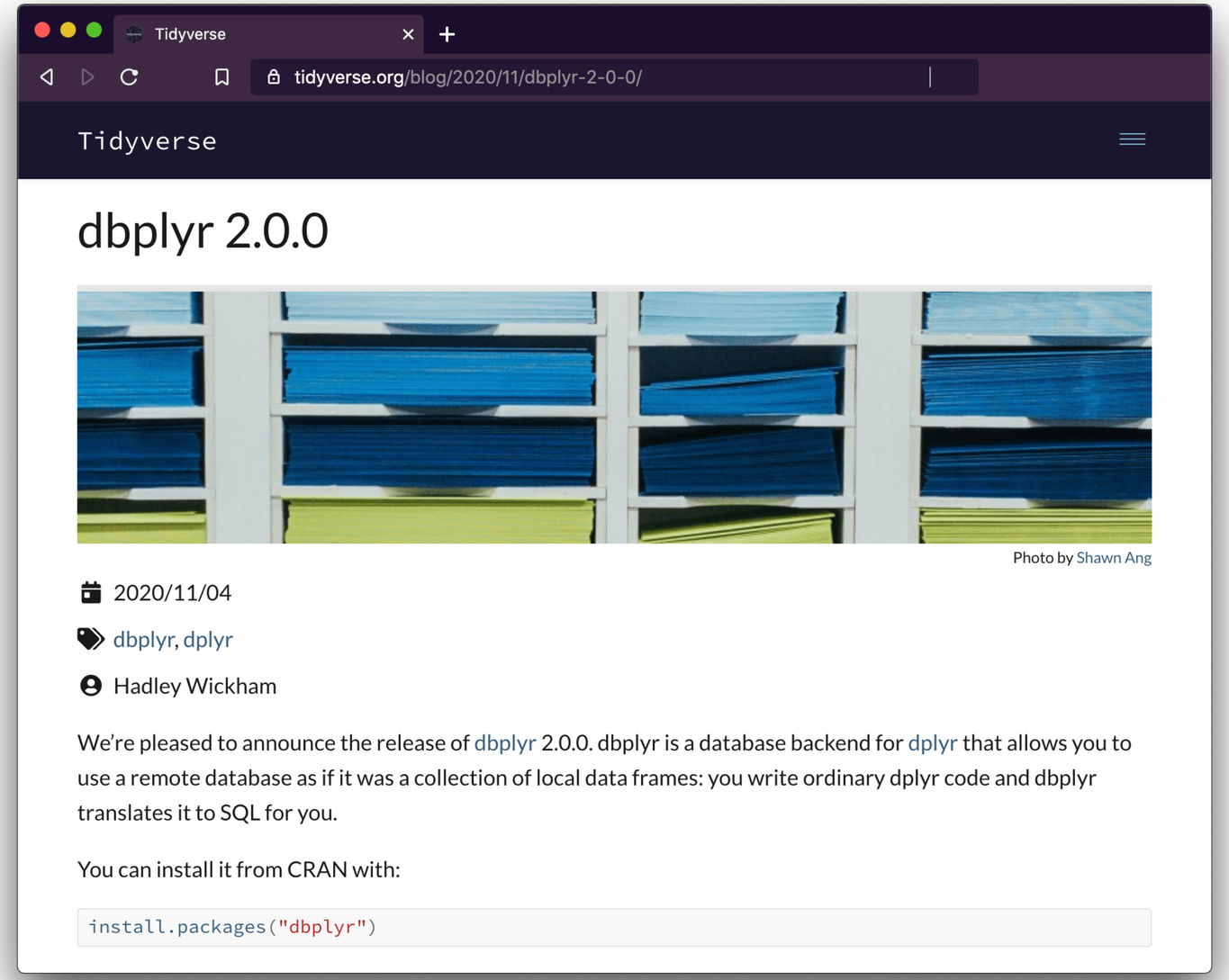
```
# slice_* functions ungrouped

gtcars2 %>%
  select(c(1:5, "msrp")) %>%
  slice_max(msrp, n = 5)
#> # Source:     lazy query [?? x 6]
#> # Database:   sqlite 3.33.0 [:memory:]
#> # Ordered by: desc(msrp)
#>   mfr         model          year trim             bdy_style      msrp
#>   <chr>       <chr>         <dbl> <chr>            <chr>         <dbl>
#> 1 Ferrari     LaFerrari      2015 Base Coupe       coupe       1416362
#> 2 Ford        GT             2017 Base Coupe       coupe        447000
#> 3 Lamborghini Aventador      2015 LP 700-4 Coupe   coupe        397500
#> 4 Rolls-Royce Dawn           2016 Base Convertible convertible  335000
#> 5 Ferrari     F12Berlinetta  2015 Base Coupe       coupe        319995
```

slice_*()

```
# slice_* functions grouped
gtcars2 %>%
  select(c(1:5, "msrp")) %>%
  group_by(bdy_style) %>%
  slice_max(msrp, n = 5)
#> # Source:     lazy query [?? x 6]
#> # Database:   sqlite 3.33.0 [:memory:]
#> # Groups:     bdy_style
#> # Ordered by: desc(msrp)
#>    mfr           model          year trim              bdy_style      msrp
#>    <chr>         <chr>         <dbl> <chr>             <chr>         <dbl>
#>  1 Rolls-Royce   Dawn           2016 Base Convertible  convertible  335000
#>  2 Ferrari       458 Spider     2015 Base              convertible  263553
#>  3 Ferrari       California     2015 Base Convertible  convertible  198973
#>  4 Mercedes-Benz SL-Class       2016 SL400 Convertible convertible   85050
#>  5 Porsche       718 Boxster    2017 Base Convertible  convertible   56000
#>  6 Ferrari       LaFerrari      2015 Base Coupe        coupe       1416362
#>  7 Ford          GT             2017 Base Coupe        coupe        447000
#>  8 Lamborghini   Aventador      2015 LP 700-4 Coupe    coupe        397500
#>  9 Ferrari       F12Berlinetta  2015 Base Coupe        coupe        319995
#> 10 Rolls-Royce   Wraith         2016 Base Coupe        coupe        304350
#> # … with more rows
```

Learn more: from the blog

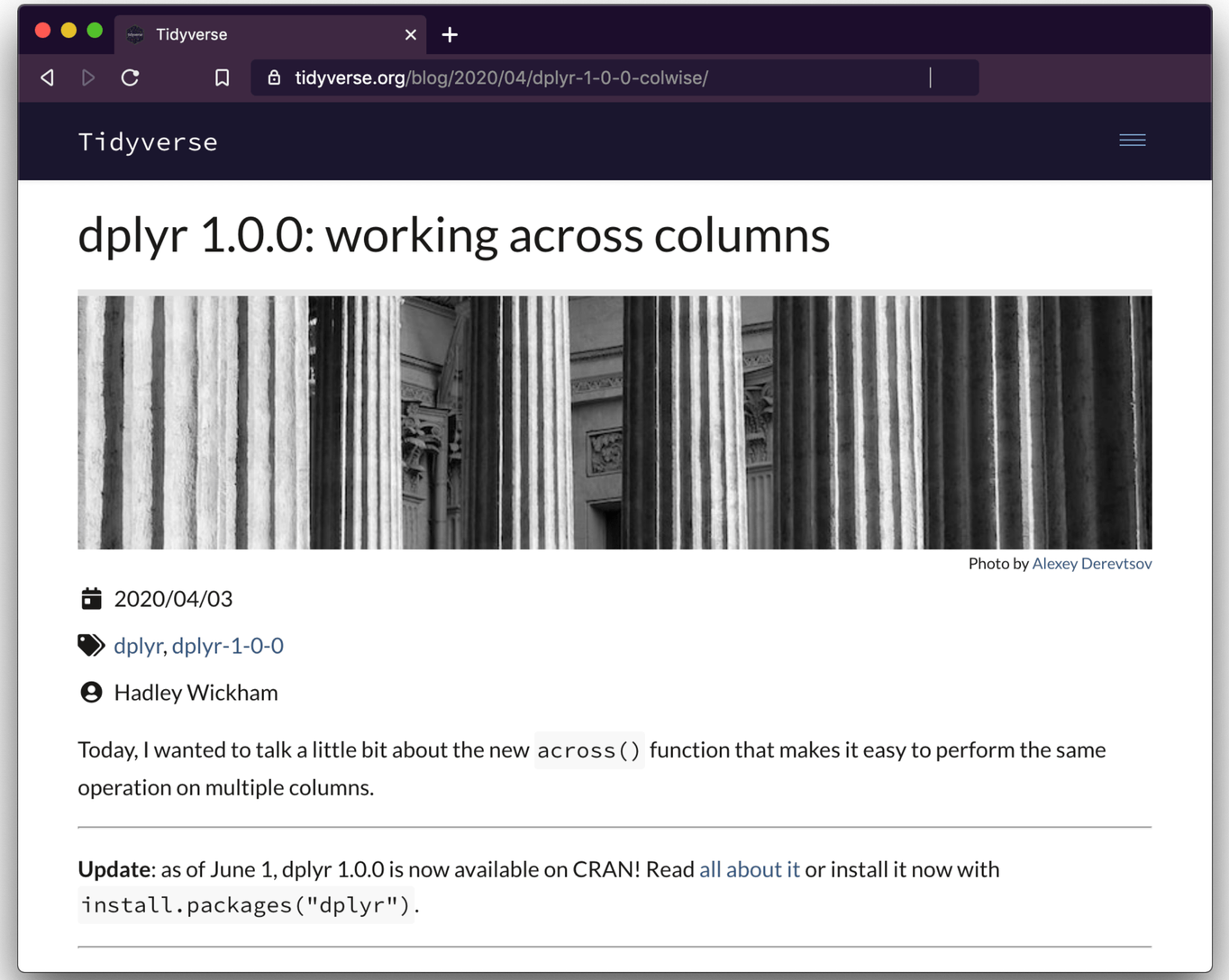
- dplyr 1.0.0 compatibility
- SQL translation improvements
- Extensibility



https://www.tidyverse.org/blog/2020/11/dbplyr-2-0-0/

Learn more: from the blog

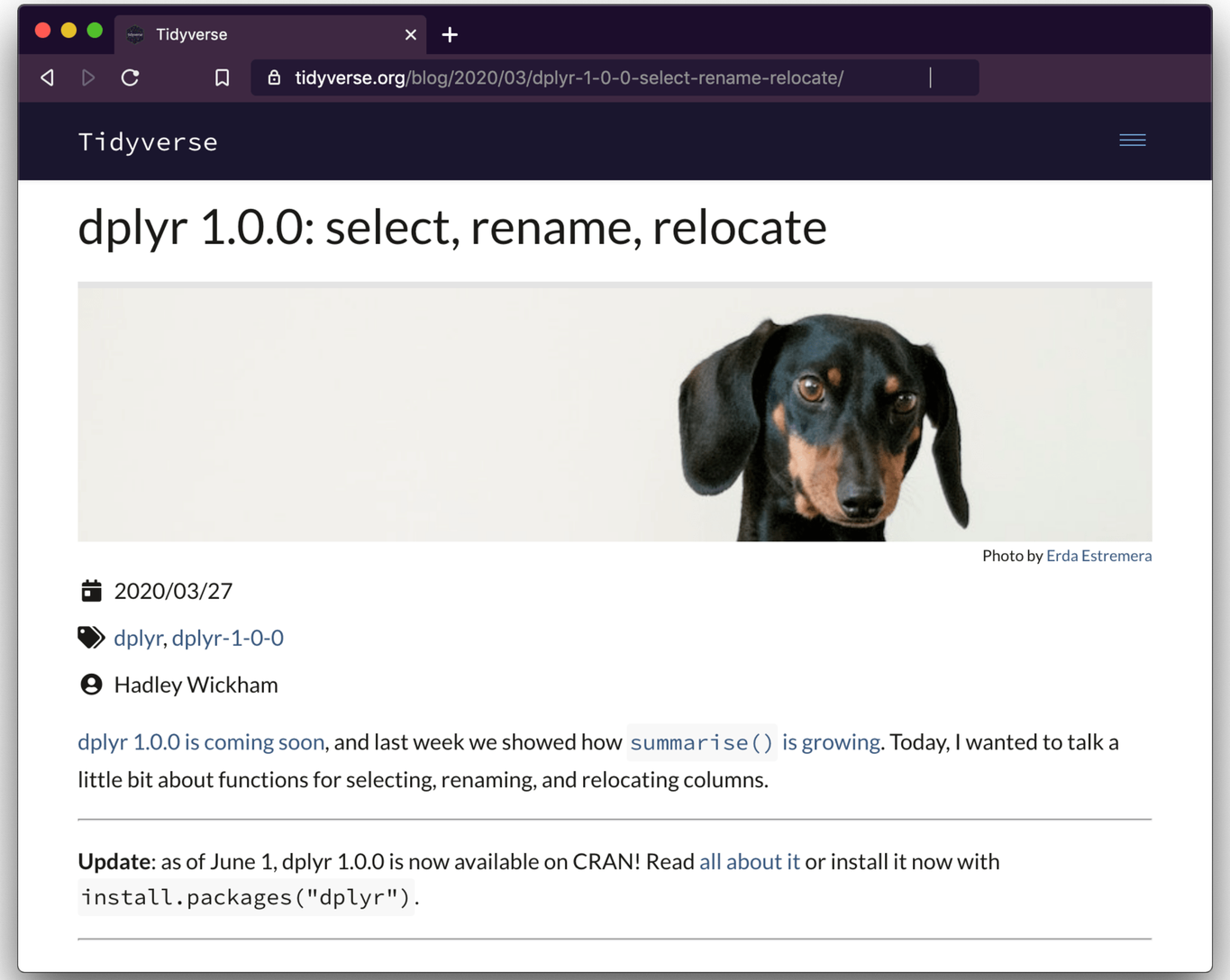
The dplyr 1.0.0 release posts...



https://www.tidyverse.org/blog/2020/04/dplyr-1-0-0-colwise/

Learn more: from the blog

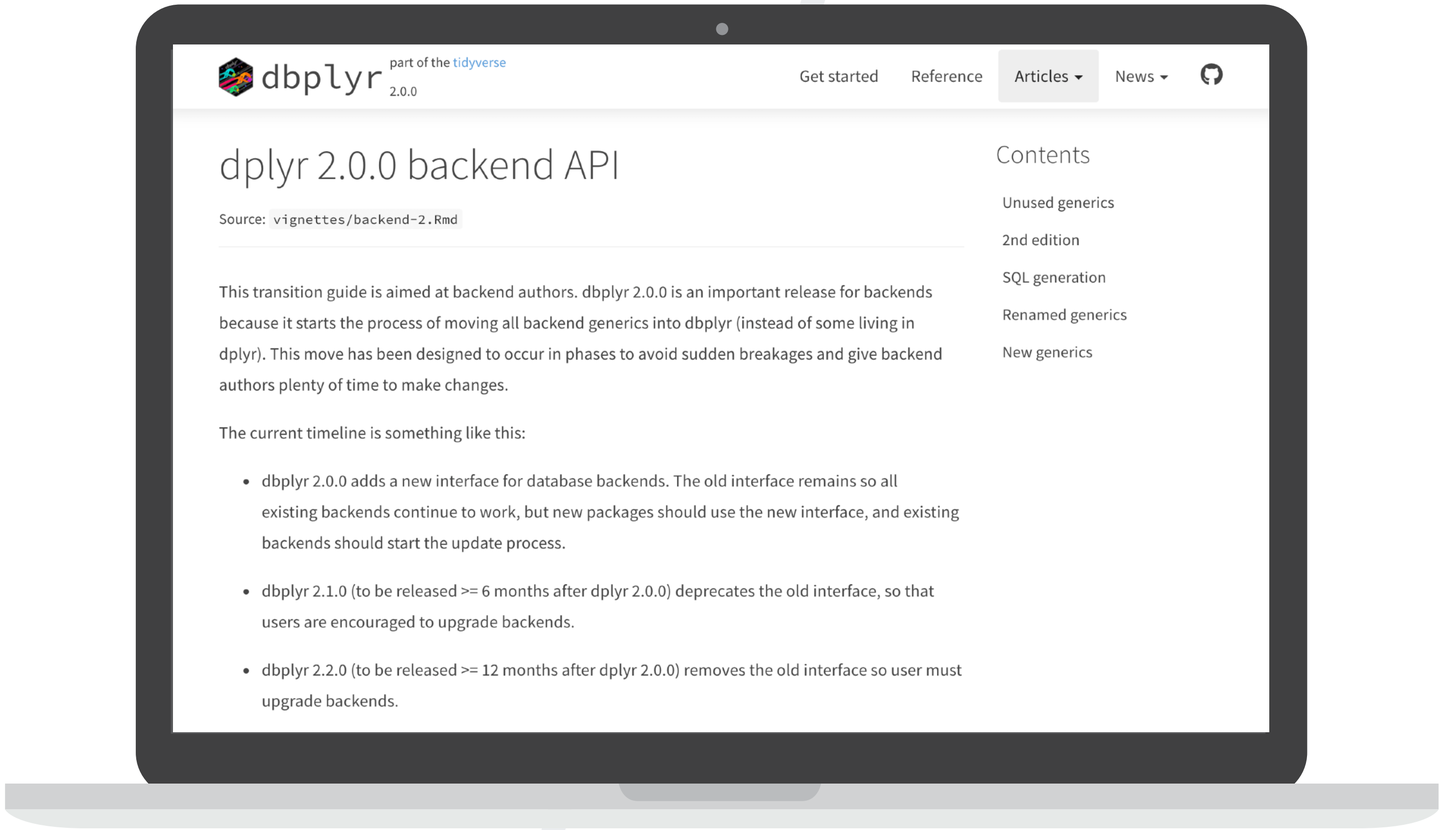
The dplyr 1.0.0 release posts...



https://www.tidyverse.org/blog/2020/03/dplyr-1-0-0-select-rename-relocate

Learn more: vignettes

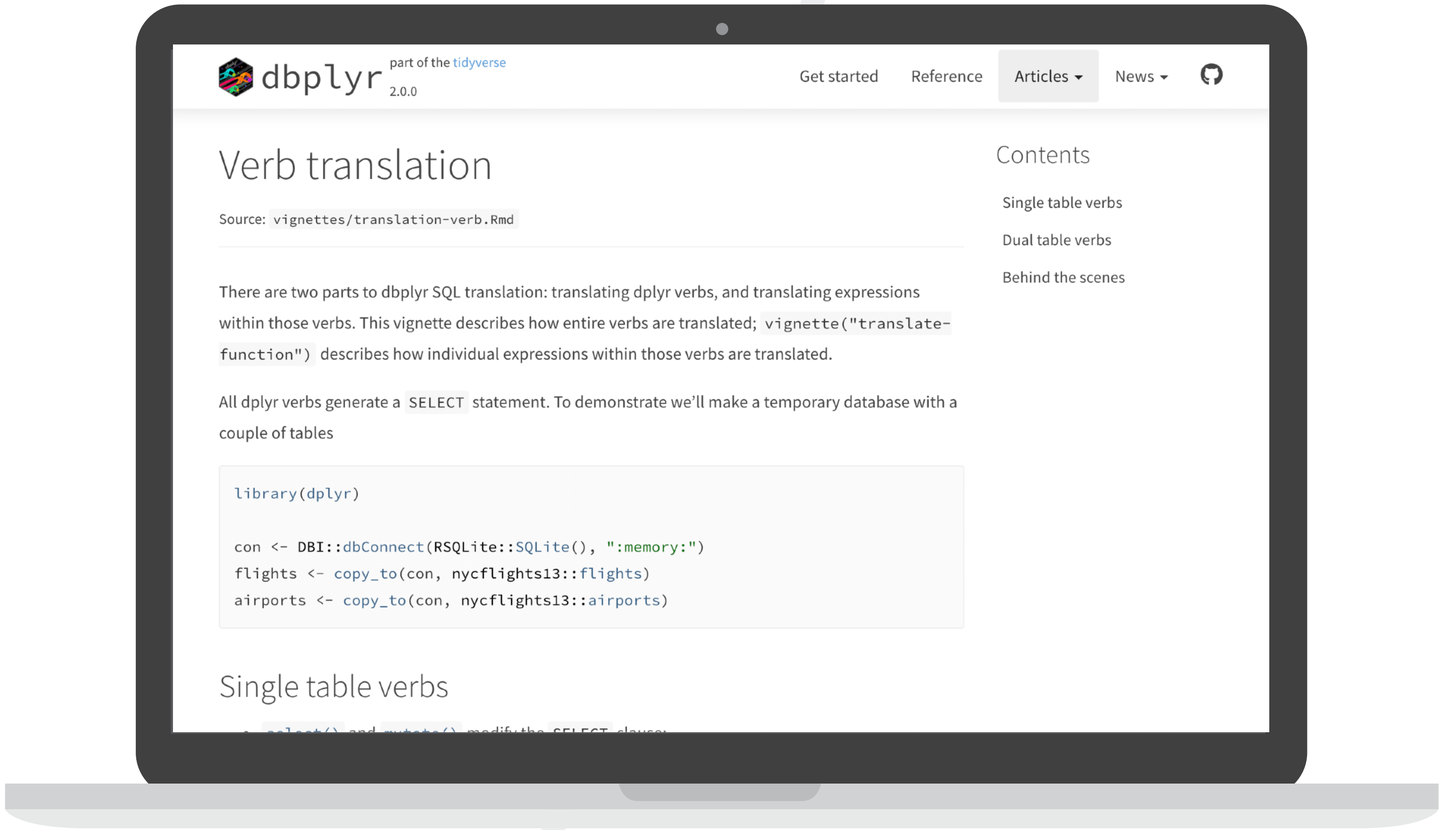
dbplyr 2.0.0 backend API



https://dbplyr.tidyverse.org/articles/backend-2.html

Learn more: vignettes

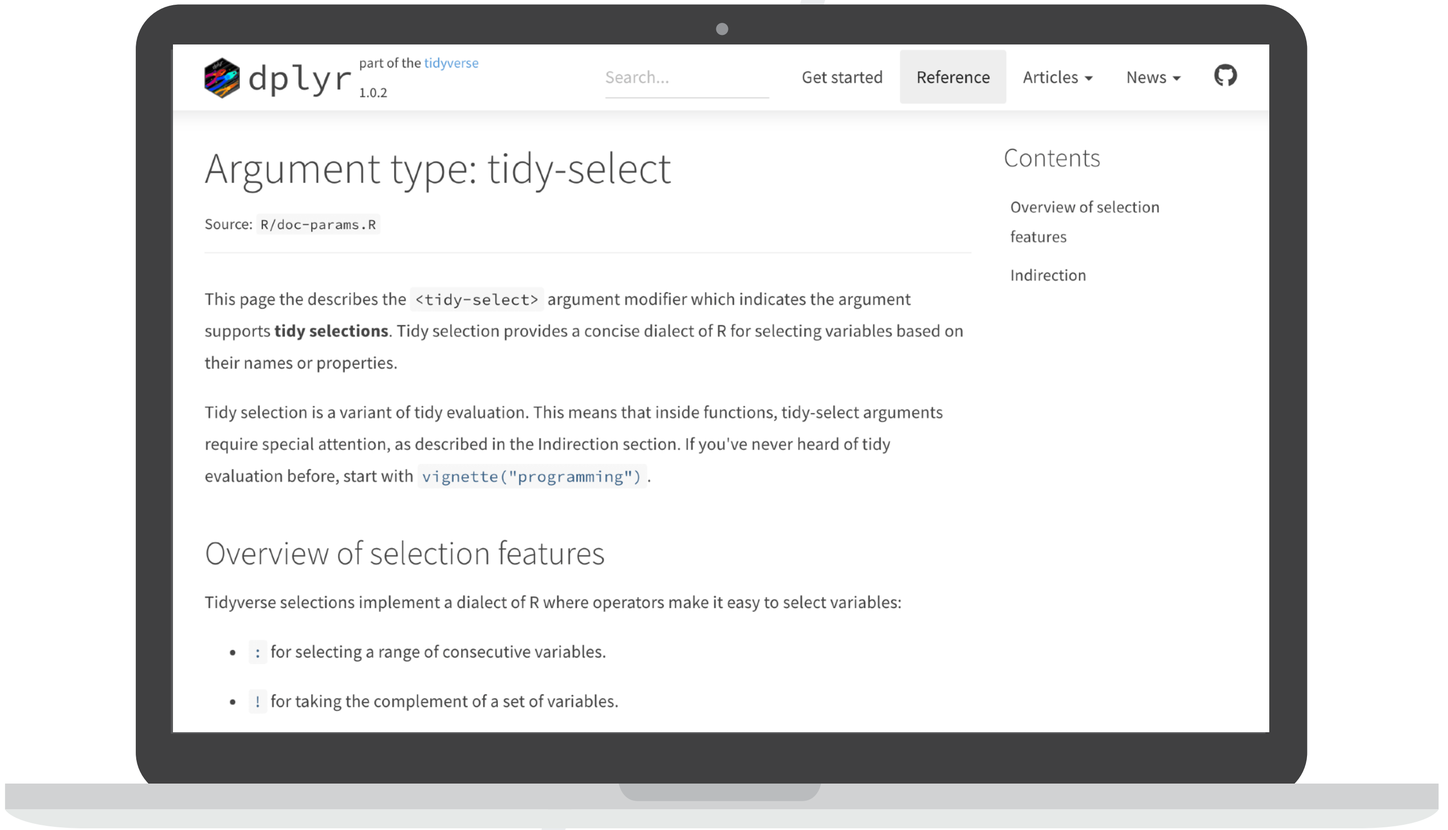
Verb translation



https://dbplyr.tidyverse.org/articles/translation-verb.html

Learn more: vignettes

All things tidy-select



https://dplyr.tidyverse.org/reference/dplyr_tidy_select.html

THANKS