

Evaluation in programming

tidyeval



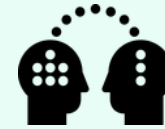
A. Ginolhac | rworkshop | 2021-09-10

▮ **Advanced topic** ✨

Learning objectives

You will learn to:

- Understand how quoted functions work
- Why we need **tidyeval**
- How to apply it for your programming
- Embrace the curly-curly operator and forget the above points



Environments and promises

Functions enclose their variables

```
x <- 1
plus_one <- function(x) {
  x <- x + 1
  x
}
plus_one(x)
```

```
[1] 2
```

```
plus_one(x)
```

```
[1] 2
```


The `x` object in **Global.Env** wasn't modified

Separate assignment and evaluation

```
msg <- "old"
delayedAssign("promise", msg)
msg <- "new!"
promise # new!
```

```
[1] "new!"
```

The promise was created when `msg` was "old"

 is fine with letting user creating variables (`promises`) that will be evaluated only later.

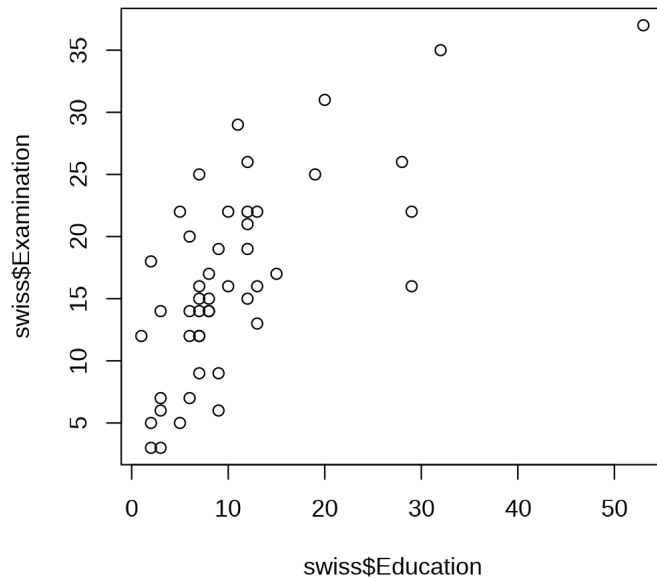
Source: help page of `delayedAssign()`

Standard vs Non-standard evaluation

R base, must refers to known objects

- But quoting (as no evaluation) is used for axis labels

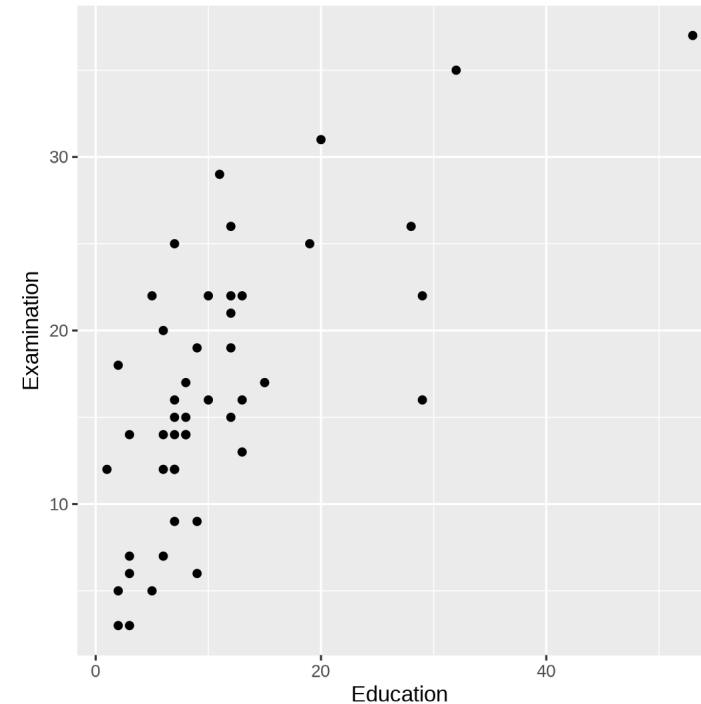
```
plot(swiss$Education, swiss$Examination)
```



ggplot2 or widely in the tidyverse

- Evaluating columns in **data** context.

```
ggplot(swiss, aes(x = Education, y = Examination)) +  
  geom_point()
```



Quoting

! Does not mean adding quotes!

But capture an expression

```
quote(Education)
```

```
Education
```

```
quote(swiss$Education)
```

```
swiss$Education
```

Without quoting, we cannot evaluate

```
eval(Education, envir = swiss)
```

```
Error in eval(Education, envir = swiss): object 'Education' not
```

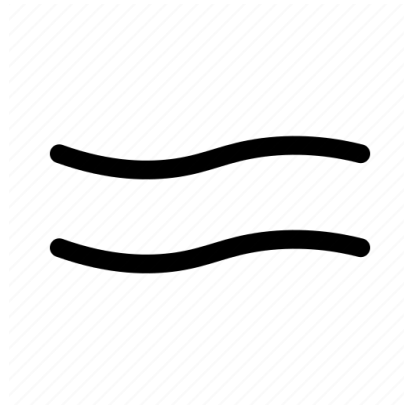
`envir` tells where to find the vector `Education`

Quoting allows to evaluate when needed

```
eval(quote(Education), envir = swiss)
```

```
[1] 12  9  5  7 15  7  7  8  7 13  6 12  7 12  5  2  8 28 20  
[26]  8  3 10 19  8  2  6  2  6  3  9  3 13 12 11 13 32  7  7  5
```

Non-standard evaluation (NSE)



Unquoted variable names

technically WRONG, but useful approximation. - [Jenny Bryan, rstudio::conf2019](#)

Quotation also works for expression

Exist in base

```
subset(swiss, (Education + Examination) > 60)[, 2:4]
```

	Agriculture	Examination	Education
Neuchatel	17.6	35	32
V. De Geneve	1.2	37	53

dplyr

```
filter(swiss, (Education + Examination) > 60) %>%  
  select(2:4)
```

	Agriculture	Examination	Education
Neuchatel	17.6	35	32
V. De Geneve	1.2	37	53

Evaluating expression

Standard evaluation of an expression

To evaluate an **expression**, you search **environments** for name bindings-values and perform the evaluation immediately.

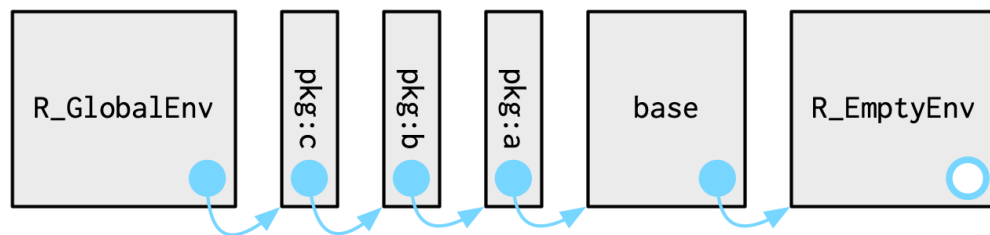
Non-standard evaluation

means you might

- Modify the **expression** or
- Modify the chain of searched **environments** before evaluation.

Why **swiss** is found while being absent in **Global_Env**?

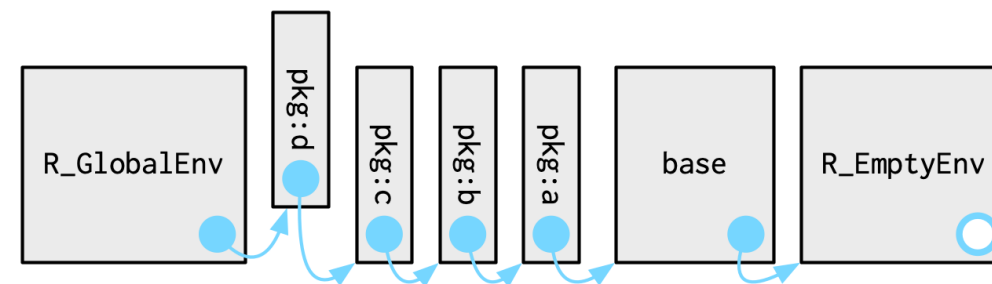
Values in environments are searched in a precise order



```
# callr allows to run a clean R session  
callr::r(function() rlang::search_envs())
```

```
[[1]] $ <env: global>  
[[2]] $ <env: package:stats>  
[[3]] $ <env: package:graphics>  
[[4]] $ <env: package:grDevices>  
[[5]] $ <env: package:utils>  
[[6]] $ <env: package:datasets>  
[[7]] $ <env: package:methods>  
[[8]] $ <env: AutoLoads>  
[[9]] $ <env: tools:callr>  
[[10]] $ <env: package:base>
```

- `swiss` lies in `datasets`, so found after 5 fails



```
callr::r(function() {  
  library(forcats)  
  rlang::search_envs()  
})
```

```
[[1]] $ <env: global>  
[[2]] $ <env: package:forcats>  
[[3]] $ <env: package:stats>  
[[4]] $ <env: package:graphics>  
[[5]] $ <env: package:grDevices>  
[[6]] $ <env: package:utils>  
[[7]] $ <env: package:datasets>  
[[8]] $ <env: package:methods>  
[[9]] $ <env: AutoLoads>  
[[10]] $ <env: tools:callr>  
[[11]] $ <env: package:base>
```

Quoting prevents evaluation, nice, but how does it work?

`Education` is unknown in the **Global Environment**

```
Education
```

```
Error in eval(expr, envir, enclos): object 'Education' not found
```

Quoting **prevents** evaluation

```
quote(Education)
```

```
Education
```

But how to **force** evaluation then?

 uses `eval()` in *data* context with `envir`

```
Education <- "tic"  
# the Education in the GlobalEnv won't clash  
eval(expr = quote(Education), envir = swiss)
```

```
[1] 12  9  5  7 15  7  7  8  7 13  6 12  7 12  5  2  8 28 20  
[26]  8  3 10 19  8  2  6  2  6  3  9  3 13 12 11 13 32  7  7  5
```

Evaluation in the tidyverse

Works even if an object name is colliding with the **Global Environment**

- `Education` is evaluated in the `swiss` context **only**

```
Education <- 20  
filter(swiss, Education > 40)
```

	Fertility	Agriculture	Examination	Education	Cathol
V. De Geneve	35	1.2	37	53	42.
	Infant.Mortality				
V. De Geneve		18			

- `.data` and `.env` pronouns exist if you need to precise **who** is **where**

```
filter(swiss, .data$Education > .env$Education)
```

	Fertility	Agriculture	Examination	Education	Cathol
Lausanne	55.7	19.4	26	28	12.
Neuchatel	64.4	17.6	35	32	16.
V. De Geneve	35.0	1.2	37	53	42.
Rive Droite	44.7	46.6	16	29	50.
Rive Gauche	42.8	27.7	22	29	58.
	Infant.Mortality				
Lausanne		20.2			
Neuchatel		23.0			
V. De Geneve		18.0			
Rive Droite		18.2			
Rive Gauche		19.3			

But you better avoid names collision

Quotations of expressions

Doing it by hand

```
expr <- quote((Education + Examination) > 60)
expr
```

```
(Education + Examination) > 60
```

```
swiss[eval(expr, envir = swiss), 1:2]
```

	Fertility	Agriculture
Neuchatel	64.4	17.6
V. De Geneve	35.0	1.2

In a function it **fail!** 💣

```
quoted_filter <- function(data, expr) {
  q_expr <- quote(expr)
  data[eval(q_expr, envir = data), 1:2]
}
quoted_filter(swiss, (Education + Examination) > 60)
```

```
Error in eval(q_expr, envir = data): object 'Examination' not f
```

Expressions in functions

`substitute()` is needed

```
quoted_filter_sub <- function(data, expr) {  
  q_expr <- substitute(expr)  
  # returns both the results and substituted expression  
  list(data[eval(q_expr, envir = data), 1:2],  
        q_expr  
  )  
}  
quoted_filter_sub(swiss, (Education + Examination) > 60)
```

```
[[1]]  
      Fertility Agriculture  
Neuchatel      64.4      17.6  
V. De Geneve   35.0       1.2  
  
[[2]]  
(Education + Examination) > 60
```

Substituting and Quoting Expressions help page

`substitute` returns the parse tree for the (unevaluated) expression `expr`, substituting any variables bound in `env`. `quote` simply returns its argument. The argument is not evaluated and can be any R expression.

- Complex but works, so **why** do we need `tidyeval`?

Why do we need `tidyeval`, i. e quasiquotation?

Add one variable in GlobalEnv

```
threshold <- 60  
quoted_filter_sub(swiss, (Education + Examination) > threshold)
```

```
[[1]]  
      Fertility Agriculture  
Neuchatel      64.4      17.6  
V. De Geneve   35.0       1.2  
  
[[2]]  
(Education + Examination) > threshold
```

Error, names clash 💣

`Fertility` is **also** a data column

```
Fertility <- 60  
quoted_filter_sub(swiss, (Education + Examination) > Fertility)
```

```
[[1]]  
      Fertility Agriculture  
Neuchatel      64.4      17.6  
V. De Geneve   35.0       1.2  
Rive Droite    44.7      46.6  
Rive Gauche    42.8      27.7
```

```
[[2]]  
(Education + Examination) > Fertility
```

Quasiquotation, bang bang !! operator

- When we need to unquote **part** of the expression: !!
- `rlang::qq_show()` helper to check

Demonstration

```
quo((Education + Examination) > Fertility)
```

```
<quosure>  
expr: ^(Education + Examination) > Fertility  
env:  0x5641f6598958
```

```
rlang::qq_show(quo((Education + Examination) > !! Fertility))
```

```
quo((Education + Examination) > 60)
```

The bang bang operator !!

In dplyr, unquote only Fertility

- Keep the rest of the expression as before

```
filter(swiss, (Education + Examination) > !! Fertility)
```

	Fertility	Agriculture	Examination	Education	Catholic
Neuchatel	64.4	17.6	35	32	16.
V. De Geneve	35.0	1.2	37	53	42.

	Infant.Mortality
Neuchatel	23
V. De Geneve	18

New, the curly-curly operator

```
filter(swiss, (Education + Examination) > {{Fertility}})
```

	Fertility	Agriculture	Examination	Education	Catholic
Neuchatel	64.4	17.6	35	32	16.
V. De Geneve	35.0	1.2	37	53	42.

	Infant.Mortality
Neuchatel	23
V. De Geneve	18



BrodieG

@BrodieGaslam




New in-depth blogpost on NSE with [#rlang](#)'s quosures:

brodieg.com/2020/08/11/quosures/

Jump down the rabbit hole with me to see how they work, and of course, how to implement them in base [#rstats](#).



rlang is providing the toolkit

- Really advanced
- Most the time, you don't need it
- Already exposed in main tidyverse packages
- Animation done in  with [rayshader](#) by **Brodie Gaslam**

tidyeval, what to remember

Use unquoted names in your own functions

- `Fertility` as a **promise** (unknown in `Global Env`)
- Evaluation is delayed by `enquo()`
- User decide the evaluation with `!!`
- Both can be abstracted with `{{}}`

```
select_head <- function(.data, column, n = 5) {  
  .data %>%  
    select({{column}}) %>%  
    slice_head(n = n)  
}  
select_head(swiss, column = Fertility, n = 2)
```

	Fertility
Courtelary	80.2
Delemont	83.1

```
select_head(swiss, column = Fertility, n = 2)
```

	Fertility
Courtelary	80.2
Delemont	83.1

```
select_head(swiss, column = c(Fertility, Catholic), n = 2)
```

	Fertility	Catholic
Courtelary	80.2	9.96
Delemont	83.1	84.84

```
select_head(swiss, column = Fertility:Examination, n = 3)
```

	Fertility	Agriculture	Examination
Courtelary	80.2	17.0	15
Delemont	83.1	45.1	6
Franches-Mnt	92.5	39.7	5

All you need to remember, **unquoted** arguments in functions: use `{{arg}}`

New column name, the operator :=

- To turn a **quosure** into a **name** that could be pasted
- Must use a specific assignment **:=** (**walrus**, from **data.table**)

```
my_summarise <- function(.data, out_name, expr) {  
  summarise(.data,  
    {{out_name}} := mean({{expr}})  
  )  
}
```

```
my_summarise(swiss, m_fer, Fertility)
```

```
      m_fer  
1 70.14255
```

```
my_summarise(swiss, mean_exam, Examination)
```

```
    mean_exam  
1  16.48936
```

Before we stop

You learned to:

- grasp non-standard evaluation
- use it with `tidyeval`
- Pass arguments as promises, not strings

I can categorically say if you're pasting strings to program with dplyr, there is always better way.

— [Hadley Wickham](#)

Acknowledgments ☐ ☐

- Lionel Henry
- Jenny Bryan
- Clause O. Wilke
- Hadley Wickham

Further reading 📖

- [5min talk](#) by Hadley Wickham
- [RStudio::conf 2019, Lazy evaluation](#) by Jenny Bryan
- [1h Rstudio webinar](#) by Lionel Henry
- [RStudio::conf 2017 slides](#) by Lionel Henry and Hadley Wickham
- [quasiquote](#) adv in R by Hadley Wickham
- [reduce, map to generate code, lm example](#) adv in R by Hadley Wickham
- [tidyeval ressources clustered](#) by Mara Averick
- [do it by quasiquote](#) by Jamel Alsalam

Thank you for your
attention!