QUIRKS OF R

BAY AREA USER GROUP MEETUP

SEPTEMBER 13, 2016

Ankur Gupta | @ankurio

WHICH PACKAGE LOADS?

```
> library(ggplot2)

> # ggplot2 loads
```

WHICH PACKAGE LOADS?

```
> library("ggplot2")

> # ggplot2 loads
```

WHICH PACKAGE LOADS?

```
dplyr <- "ggplot2"
> library(dplyr)

Attaching package: ‘dplyr’

The following object is masked from ‘package:stats’:

    filter

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
> # dplyr loads!
```

R parses unquoted text.

This is non-standard evaluation or NSE .

NON-STANDARD EVALUATION (NSE) HAPPENS EVERYWHERE

base	rm, ls, subset
utils	demo, example
graphics	plot
ggplot2	aes
plyr	summarize
dplyr	filter, select, arrange, summarize

We will focus on `subset()`.

`subset()` USES NSE

```
> pop.df
  state year pop
1    CA 2005  37
2    WI 2005   6
3    CA 2015  39
4    WI 2015   6

> subset(pop.df, year == 2015)
  state year pop
3    CA 2015  39
4    WI 2015   6
```

`year` is the column name (in dataframe scope)

`subset()` USES NSE WITH MIXED SCOPING

```
> pop.df
  state year pop
1    CA 2005  37
2    WI 2005   6
3    CA 2015  39
4    WI 2015   6

> x <- 2015
> subset(pop.df, year == x)
  state year pop
3    CA 2015  39
4    WI 2015   6
```

`year` is the column name (in dataframe scope)

`x` is the global variable (global scope)

`subset()` CAN'T DIFFERENTIATE SCOPES

```
> pop.df
  state year pop
1    CA 2005  37
2    WI 2005   6
3    CA 2015  39
4    WI 2015   6

results.by.year.1 <- function(df, year) {
    df.subset <- subset(df, year == year)
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.1(pop.df, 2015)
[1] 4
> # Incorrect result. Silent mistake! Correct result is 2.
```

Using `year` as an argument name is a bad idea.

Let's change it.

APPARENT FIX

```
> pop.df
  state year pop
1    CA 2005  37
2    WI 2005   6
3    CA 2015  39
4    WI 2015   6

results.by.year.2 <- function(df, yr) {
    df.subset <- subset(df, year == yr)
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.2(pop.df, 2015)
[1] 2
> # Correct result
```

This doesn't solve the problem.

It only hides the problem.

HIDDEN PROBLEM WITH `subset()`

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

results.by.year.2 <- function(df, yr) {
    df.subset <- subset(df, year == yr)
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.2(pop.df, 2015)
[1] 1
> # Incorrect result. Silent mistake! Correct result is 2.
```

How do we choose the second argument name for `results.by.year.x()` ?

DIRTY FIX

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

results.by.year.3 <- function(df, yr) {
    testthat::expect_false("yr" %in% names(df))
    df.subset <- subset(df, year == yr)
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.3(pop.df, 2015)
Error: "yr" %in% names(df) is not false
> # Throws error. At least it is not a silent mistake.
```

This will stop code execution but it won't make a silent mistake.

BETTER FIX

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

results.by.year.4 <- function(df, yr) {
    df.subset <- df[df[["year"]] == yr, , drop = FALSE]
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.4(pop.df, 2015)
[1] 2
> # Correct result
```

Don't use `subset()`. Use logical indexing.

(Remember to use `drop = FALSE`)

WHAT ABOUT `filter` FUNCTIONS IN `dplyr`?

`dplyr` contains two functions that do the job of `subset()`:

- `filter()` - uses NSE
- `filter_()` - uses SE

Do these solve the problem?

`filter()` WITH SINGLE SCOPE

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

> # NSE function, single scope
> dplyr::filter(pop.df, year == 2015)
  state year pop   yr rate
1    CA 2015  39 3000    6
2    WI 2015   6 5000   10
> # Correct result
```

`filter()` WITH MIXED SCOPE

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

> # NSE function, mixed scope
> year <- 2015
> dplyr::filter(pop.df, year == year)
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00
> # Incorrect result
```

`filter_()` WITH SINGLE SCOPE

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

> # SE function, single scope
> dplyr::filter_(pop.df, "year == 2015")
  state year pop   yr rate
1    CA 2015  39 3000    6
2    WI 2015   6 5000   10
> # Correct result
```

`filter_()` WITH MIXED SCOPE

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

> # SE function, mixed scope
> year <- 2015
> dplyr::filter_(pop.df, "year == year")
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00
> # Incorrect result
```

Using a SE function by itself does not solve the problem of mixed scope.

PROBLEM OF MIXED SCOPE REMAINS

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

results.by.year.5 <- function(df, yr) {
    df.subset <- dplyr::filter_(pop.df, "year == yr")
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.5(pop.df, 2015)
[1] 1
> # Incorrect result. Silent mistake! Correct result is 2.
```

`year` is the column name (in dataframe scope)

`yr` is in both in dataframe scope and function scope

FIX USING CONSTRUCTED CONDITION

```
> pop.df
  state year pop   yr  rate
1    CA 2005  37 2005  4.01
2    WI 2005   6 1000  2.00
3    CA 2015  39 3000  6.00
4    WI 2015   6 5000 10.00

results.by.year.6 <- function(df, yr) {
    condition <- lazyeval::interp(~year == x, x = yr)
    df.subset <- dplyr::filter_(pop.df, .dots = condition)
    # Do some computation
    return_value <- nrow(df.subset)
    return(return_value)
}

> results.by.year.6(pop.df, 2015)
[1] 2
> # Correct result
```

(can be slow; possible precision issues with `doubles`)

NSE + MIXED SCOPE = SILENT MISTAKES

```r
# Dirty Fix - Put in assert-like statements
testthat::expect_false("yr" %in% names(df))
df.subset <- subset(df, year == yr)

# Fix 1 - Use SE with constructed condition
condition <- lazyeval::interp(~year == x, x = yr)
df.subset <- dplyr::filter_(pop.df, .dots = condition)

# Fix 2 - Use logical indexing
df.subset <- df[df[["year"]] == yr, , drop = FALSE]
```

TAKEAWAYS

- NSE can cause problems with scope
- Even with SE, mixed scope can cause problems
- Add assert-like statements, especially with NSE
- For non-interactive code, package writing, use logical indexing instead

THANK YOU

- Slides: tiny.cc/quirksofr

Ankur Gupta | @ankurio