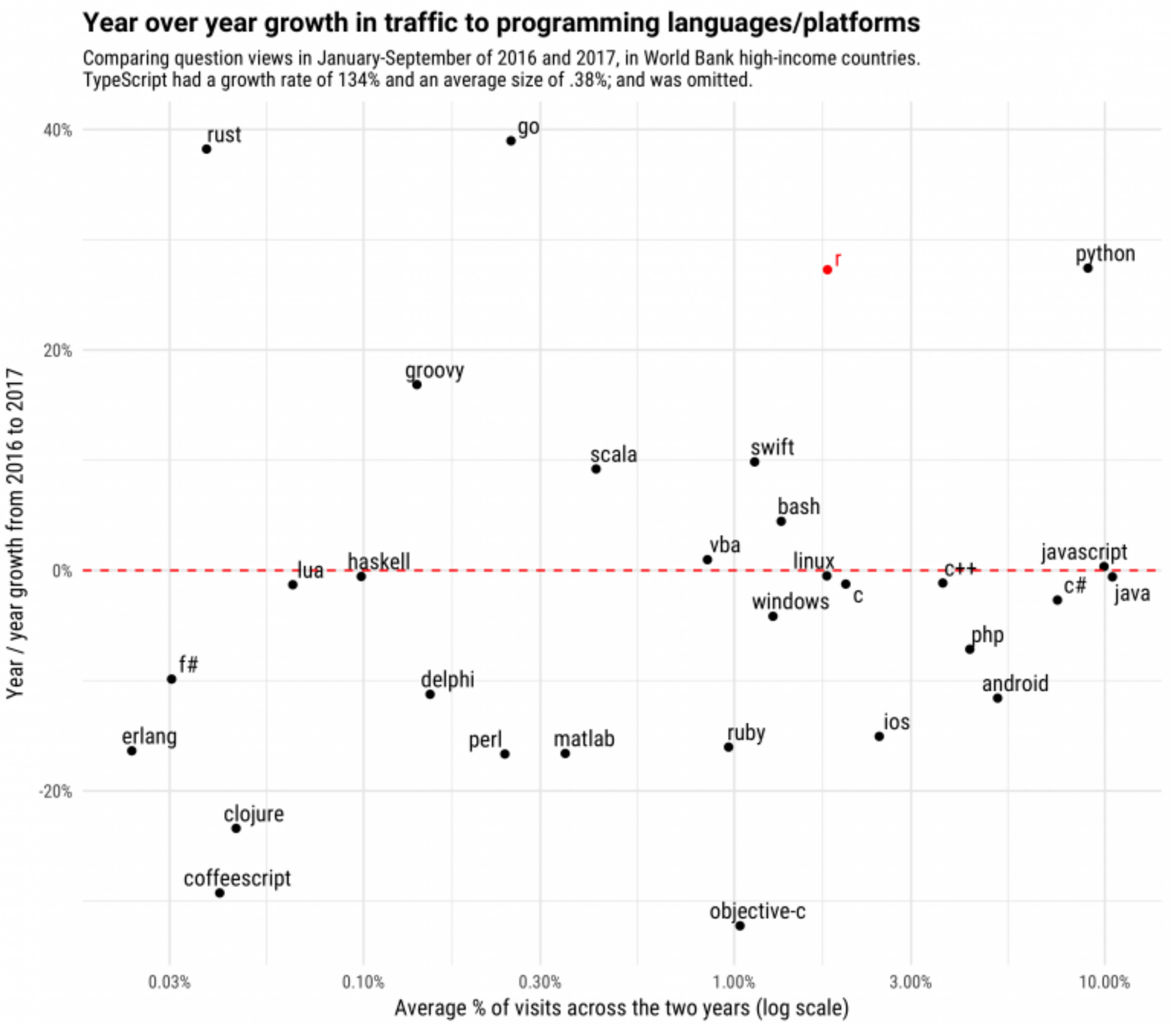
Extending R Syntax

in package space

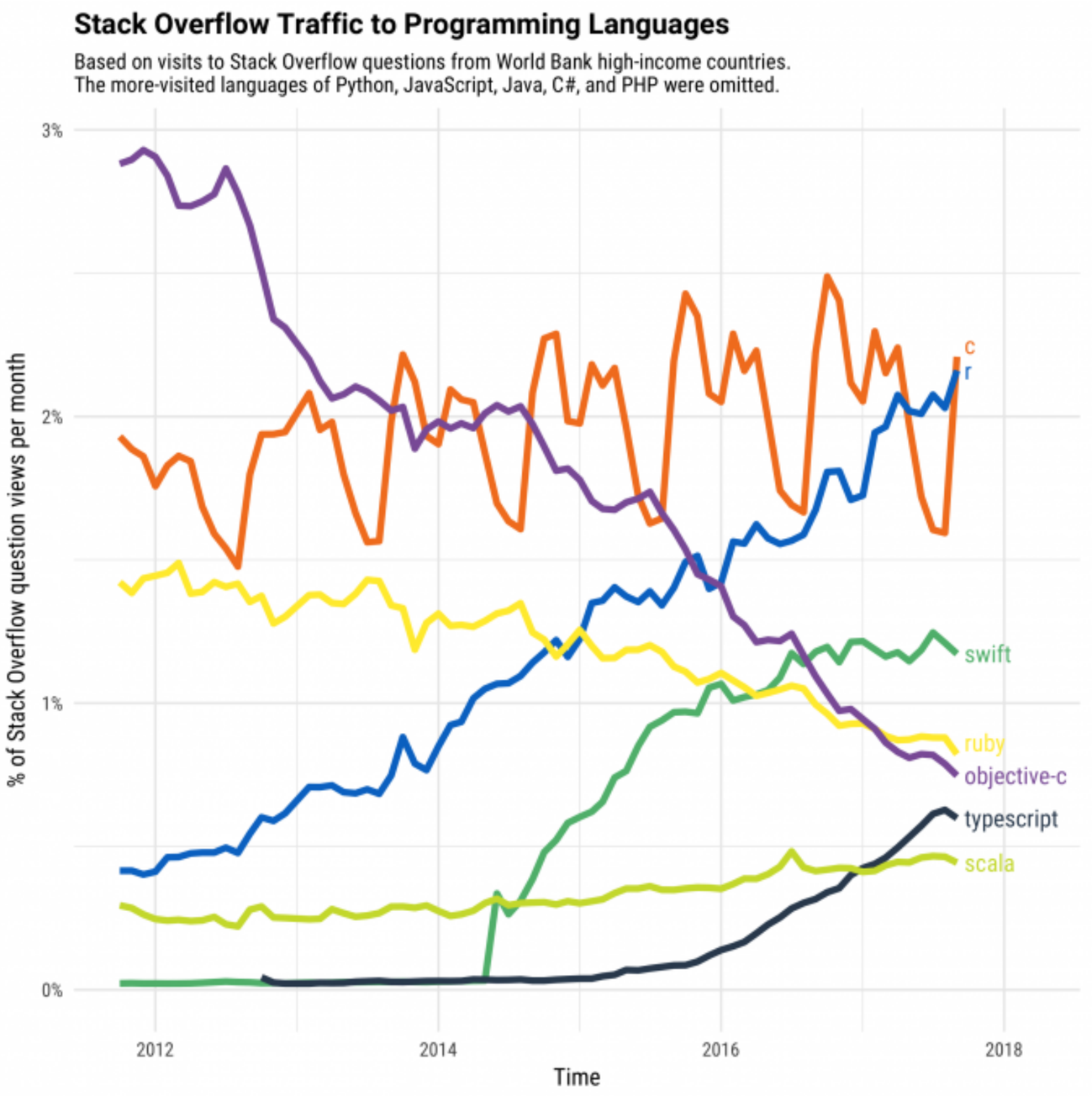
November 07th 2018

Jim Hester
🔗 bit.ly/ext-r-syntax
RStudio®

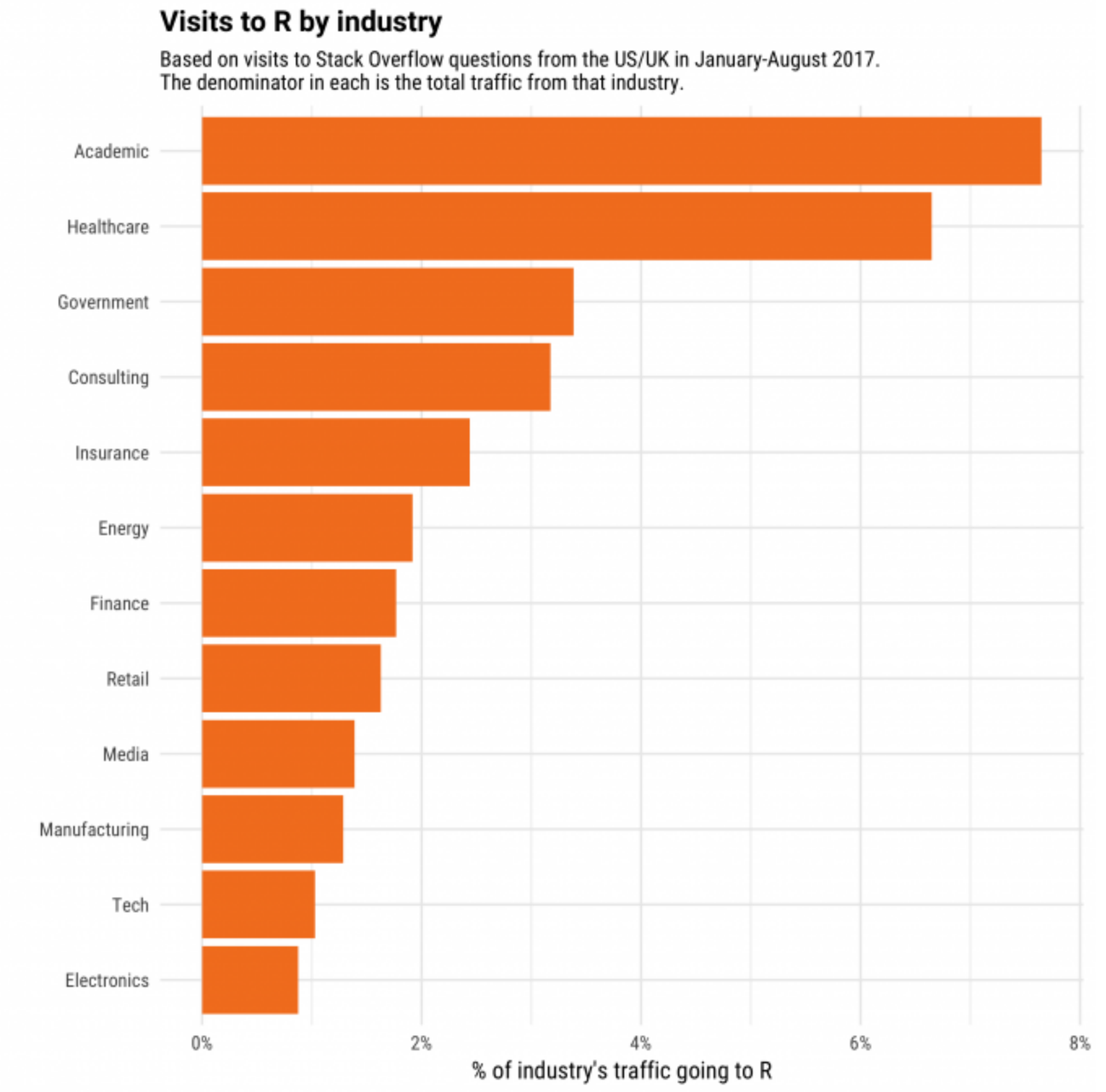
Who uses R?



David Robinson - The Impressive Growth of R
https://stackoverflow.blog/2017/10/10/impressive-growth-r/



David Robinson - The Impressive Growth of R
https://stackoverflow.blog/2017/10/10/impressive-growth-r/

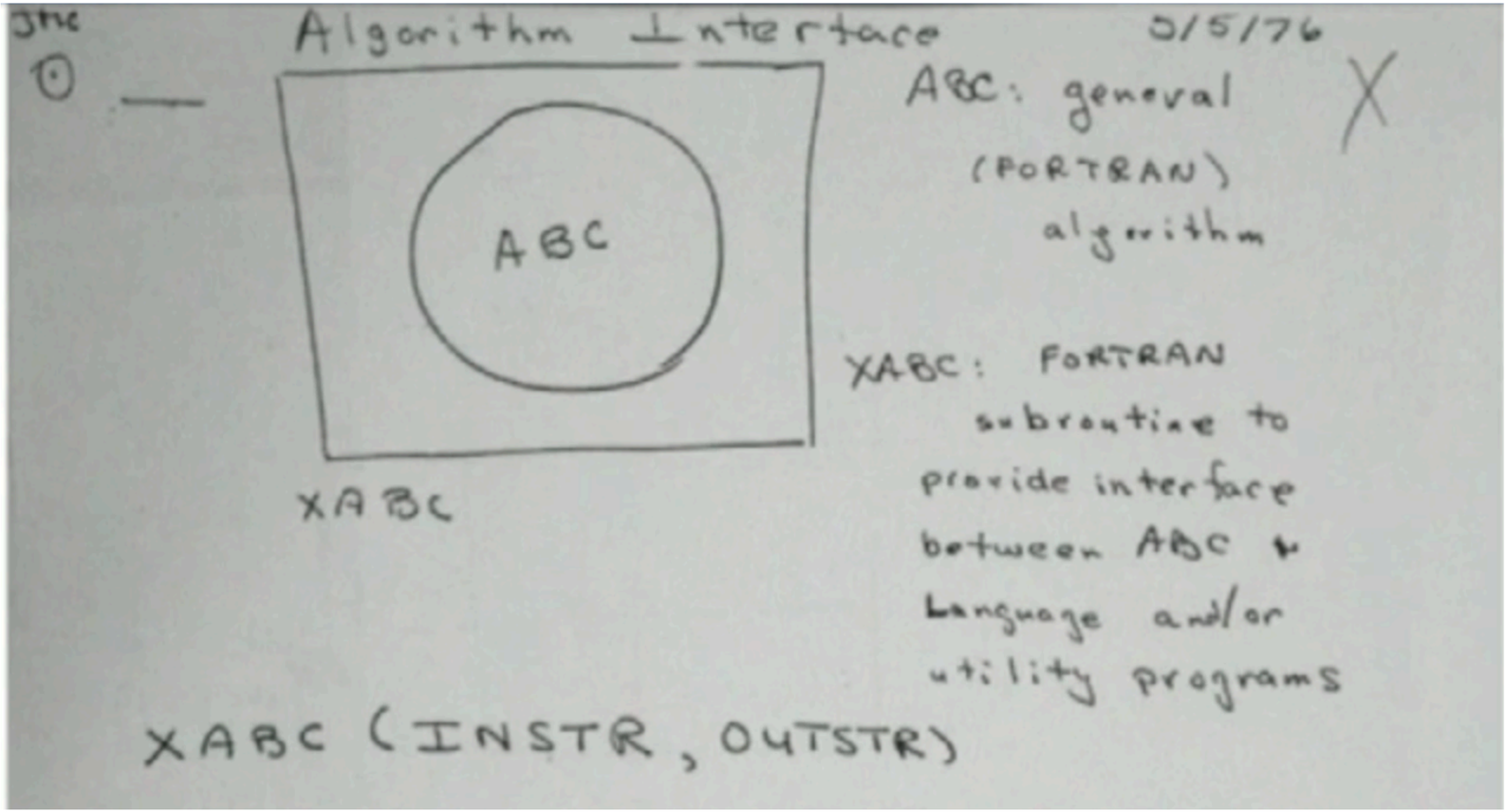


David Robinson - The Impressive Growth of R
https://stackoverflow.blog/2017/10/10/impressive-growth-r/

Where did R come from?

1966			APL
1968			SPSS
1972			C
1976	S	macro processor	SAS
1978			‘K&R’ C
1984			MATLAB
1986			Mathematica
1988	’New’ S	first class functions	S-Plus
1991			Python
1993	‘Modern’ S	S3, formula, data.frame	
1995	R	lexical scope, in-memory data	
1997	GNU R		
2000	R 1.0		Python 2.0
2005	R 2.1	UTF-8 encoding, internationalization	
2008			Python 3.0
2012			Julia
2013	R 3.0	vectors > 2^31	
2018	R 3.5	default byte compilation / JIT	

Interactive Interface language



What makes R different?

wrathematics
@wrathematics

Follow

I feel like people who compare #rstats to other languages often forget that our "hello world" is fitting a linear model. Different crowd.

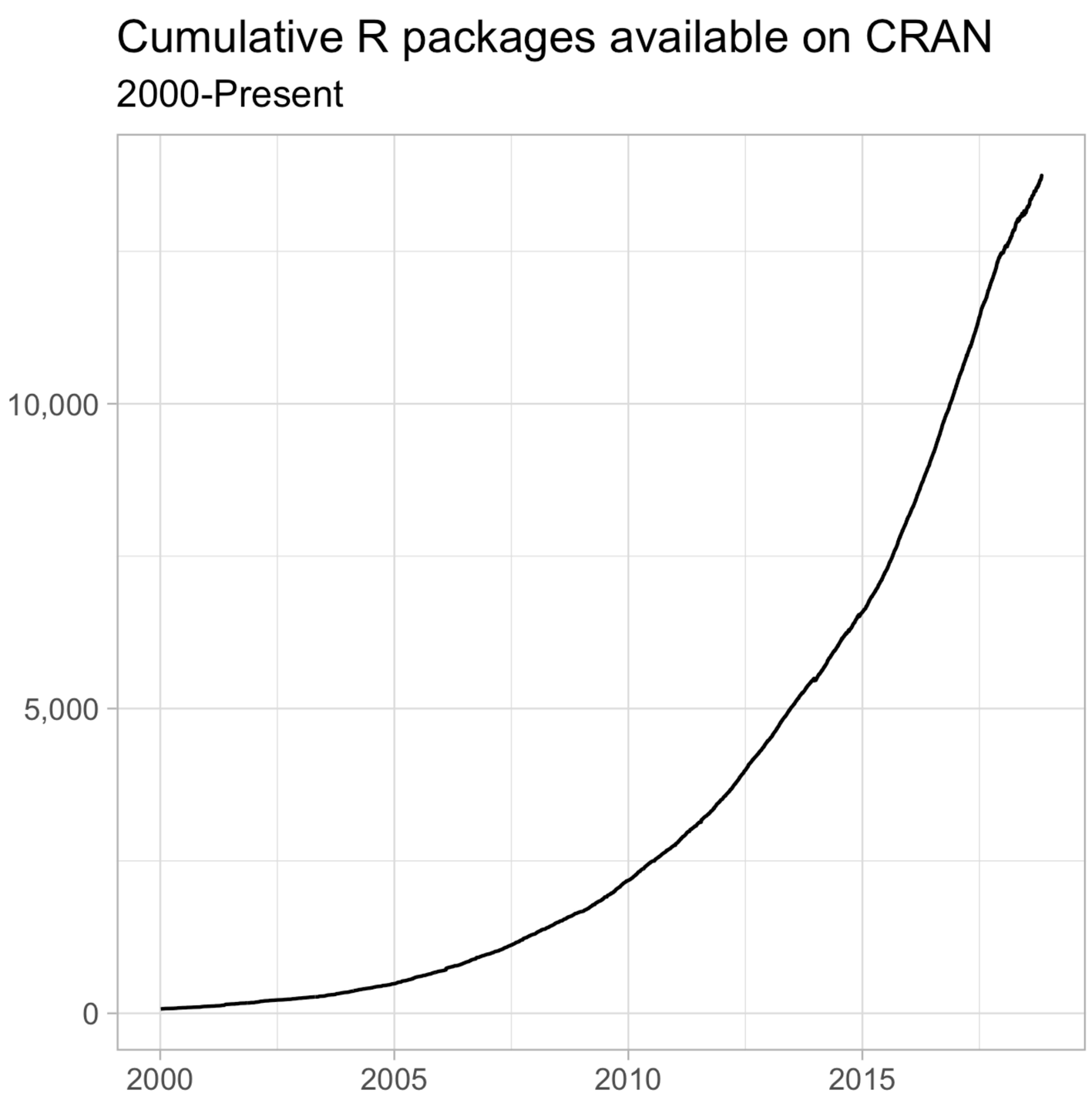
12:51 PM - 21 Oct 2014

"Everything that exists is an object. Everything that happens is a function call."

- John Chambers

demo

Extending R



CRAN packages

Manipulation
dplyr, data.table

Plotting
ggplot2, lattice, plotly

Modeling
stats, survey, randomForest, caret

Reporting
rmarkdown, bookdown, pkgdown

Web scraping
httr, rvest

Web app frameworks
shiny, OpenCPU

string interpolation glue 📦

```r
name <- "Fred"
age <- 50
anniversary <- as.Date("1991-10-12")


paste0(
  "My name is ", name, "\n",
  "my age next year is ", age + 1, "\n",
  "my anniversary is ", format(anniversary, "%A, %B %d, %Y"), "."
)
```

```
library(glue)

glue('
  My name is {name}
  my age next year is {age + 1}
  my anniversary is {format(anniversary, "%A, %B %d, %Y")}.')
#> My name is Fred
#> my age next year is 51
#> my anniversary is Saturday, October 12, 1991.
```

```r
mt <- head(mtcars)

glue_data(mt,
 "{model} has {hp} hp",
  model = rownames(model)
)
#> Mazda RX4 has 110 hp
#> Mazda RX4 Wag has 110 hp
#> Datsun 710 has 93 hp
#> Hornet 4 Drive has 110 hp
#> Hornet Sportabout has 175 hp
#> Valiant has 105 hp
```

piping / chaining

magrittr 

Little bunny Foo Foo
Hopping through the forest
Scooping up the field mice
And boppin' 'em on the head!

- Common nursery rhyme

```
foo_foo <- little_bunny()
foo_foo <- hop(foo_foo, through = forest)
foo_foo <- scoop(foo_foo, up = field_mice)
foo_foo <- bop(foo_foo, on = head)
```

```
foo_foo <- little_bunny()
foo_foo <- hop(foo_foo, through = forest)
foo_foo <- scoop(foo_foo, up = field_mice)
foo_foo <- bop(foo_foo, on = head)
```

```
bop(
  scoop(
    hop(foo_foo, through = forest),
    up = field_mice
  ),
  on = head
)
```

```
bop(
  scoop(
    hop(foo_foo, through = forest),
    up = field_mice
  ),
  on = head
)
```

Dagwood Sandwich

```
library(magrittr)

foo_foo <- little_bunny() %>%
  hop(through = forest) %>%
  scoop(up = field_mice) %>%
  bop(on = head)
```

```
my_pipe <- function(.) {
   . <- hop(., through = forest)
   . <- scoop(., up = field_mice)
   bop(., on = head)
}
my_pipe(foo_foo)
```

```r
library(magrittr)
mtcars %>%
  subset(hp > 100) %>%
  lm(mpg ~ hp + wt, data = .) %>%
  summary()
#>
#> Call:
#> lm(formula = mpg ~ hp + wt, data = .)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.2126 -1.1578 -0.1503  0.7979  4.6669
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 33.231786   1.886344  17.617 1.20e-13 ***
#> hp          -0.020698   0.008114  -2.551    0.019 *
#> wt          -3.410342   0.559159  -6.099 5.83e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.052 on 20 degrees of freedom
#> Multiple R-squared:  0.7869, Adjusted R-squared:  0.7656
#> F-statistic: 36.92 on 2 and 20 DF,  p-value: 1.933e-07
```

tidy evaluation

dplyr / rlang 

demo

quoting functions

passed the *expression*, rather than the *value*

```
mt <- subset(mtcars, mpg > 20)

with(mt,
  plot(wt * 1000, mpg)
)
```

quoting functions

passed the *expression*, rather than the *value*

```
enquo()
```

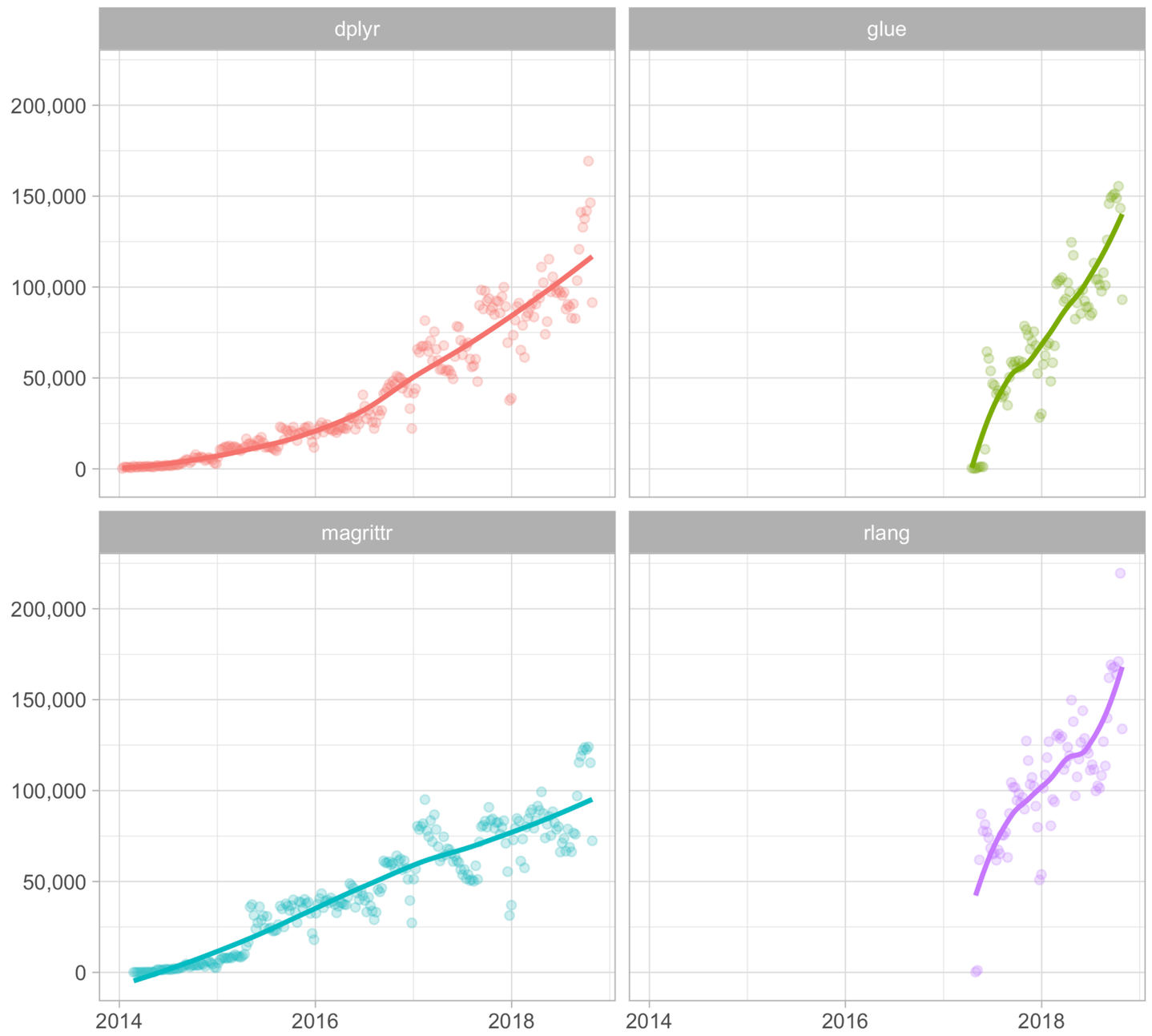
```
group_var <- enquo(group_var)
```

unquoting functions

!! - 'bang bang'

```
data %>%
    group_by(!!group_var) %>%
    summarise(mean = mean(!!summary_var))
```

Do people actually use this?



Alternative Parsers

altparsers 

altparsers 

- https://github.com/jimhester/altparsers
- Multiple parsers available
- User / package author extendable
- Can mix parsers in same package
- REPL(s) available for interactive use
- Experiment in package space

S-Expression parser

Simple proof of concept

Illustrates completely de-novo parser

```
(+ 1 (* 2 3))
```

py parser

Python style whitespace

```
factorial = function(x)
  if (x <= 1)
    return(1)
  x * factorial(x - 1)
```

tidy parser

Raw strings

```
grepl(r"\w+")
```

glue strings

```
x <- 1; g"x = {x}"
```

native pipes

```
mtcars |> filter(hp > 150) |> select(hp, mpg)
```

list generation

```
[1, 2, 3, [4, 5, 6]]
```

Interactive use

Start REPL with a new parser

```
repl(parse_text)
repl(sexp_parse_text)
repl(py_parse_text)
repl(tidy_parse_text)
```

Quit the REPL

```
q()
```

Package use

- Scripts - inst/*
- export / document as normal

```
.onLoad <- function(...) {
  altparsers::src(system.file(package = "ex", "sexp"),
                  package = "ex",
                  altparsers::sexp_parse_file)

  altparsers::src(system.file(package = "ex", "tidy"),
                  package = "ex",
                  altparsers::tidy_parse_file)
}
```

demo

Future directions

- Survey for pain points in current R syntax
  - Design new parser
- Use alternative parsers in user facing package
- Aviral Goel - Type Annotations for R

Extending R syntax

```
install.packages(c("glue", "magrittr", "rlang", "dplyr"))

devtools::install_github("jimhester/altparsers")
```


jimhester_
jimhester
jim.hester@rstudio.com