Outro

You, Me, and NSE

Non-standard evaluation is bananas!

```
nseFun <- function(fruit){
    fruit <- deparse(substitute(fruit))
    if(grepl("bananas", fruit, ignore.case = T)){
        return("Bananas!")
    }
    return("Not bananas...")
}
```

```
nseFun(oranges)
```

```
## [1] "Not bananas..."
```

```
nseFun(bananas)
```

```
## [1] "Bananas!"
```

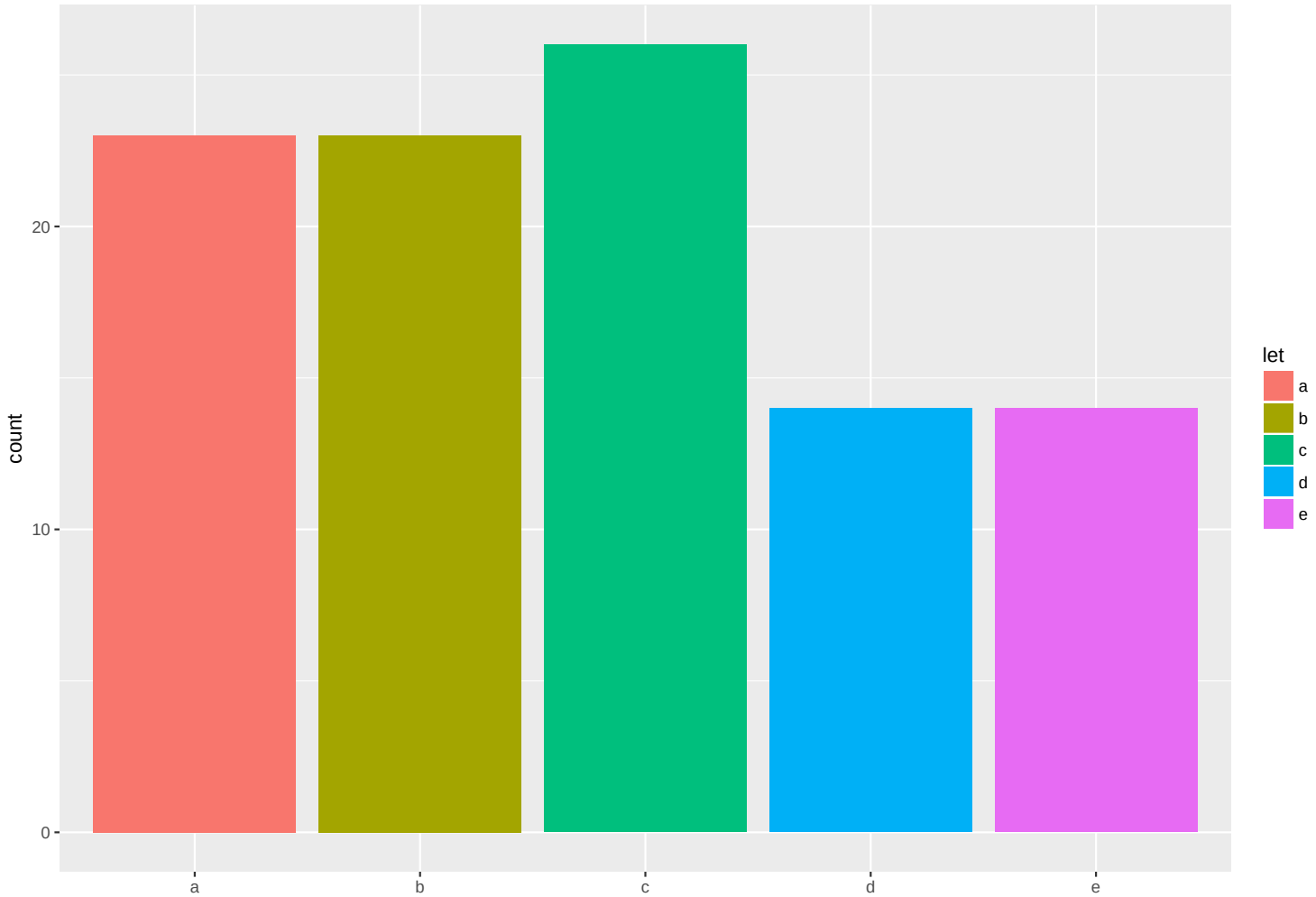
ggplot2 - Graphics at the spead of thought

Really fast plotting with ggplot2.

```
library(ggplot2)
dat <- data.frame(let = sample(letters[c(1:5,1:3,1:2)],
                               100,
                               replace = T))
```

ggplot2 - Graphics at the spead of thought

```
ggplot(dat) +
    geom_bar(aes(x=let, fill = let), stat="count")
```



tidverse

A whole suite of data procesing packages that allow for very readable, easy to write code.

Very easy for beginners to learn.

Write data processing using verb like syntax, chaining together a pipeline of data processing routines.

```
library(tidyverse)
tib <- tibble(x = runif(10000, 1, 100),
              y = runif(10000, 1, 100),
              let = sample(letters[c(1:5,1:3)],
                           10000, replace = T))
```

tidverse

See how nicely it prints?

```
tib
```

```
## # A tibble: 10,000 x 3
##         x     y let
##     <dbl> <dbl> <chr>
##  1  65.8 27.4  a
##  2  36.0 66.4  a
##  3  27.8 22.0  a
##  4  99.3  1.52 c
##  5  63.7 11.2  d
##  6  22.1 29.5  c
##  7  13.8 45.0  d
##  8  48.3 69.6  c
##  9  92.5  3.07 c
## 10  60.3 95.6  b
## # ... with 9,990 more rows
```

tidverse

Pipe-lining with magrittr: %>%

```
tib %>%
    filter(let %in% c("a", "b", "c")) %>%
    mutate(prod = x*y) -> tib

tib
```

```
## # A tibble: 7,549 x 4
##         x     y let    prod
##     <dbl> <dbl> <chr> <dbl>
##  1  65.8 27.4  a      1801
##  2  36.0 66.4  a      2388
##  3  27.8 22.0  a       611
##  4  99.3  1.52 c       151
##  5  22.1 29.5  c       651
##  6  48.3 69.6  c      3364
##  7  92.5  3.07 c       284
##  8  60.3 95.6  b      5764
##  9  73.4 91.3  b      6703
```

data.table

Super fast tabular data manipulation with a very succinct syntax.

Building a `data.table` object is just like `data.frame`.

```
library(data.table)
dt <- data.table(x = runif(10000, 1, 100),
                 y = runif(10000, 1, 100),
                 let = sample(letters[c(1:5,1:3)],
                              10000, replace = T))
```

data.table

Interfacing with the object leverages NSE for quick, succinct coding.

Group by using the by argument, counting the grouped elements.

```
dt[order(let), .N, by = let]
```

```
##    let    N
## 1:   a 2527
## 2:   b 2499
## 3:   c 2558
## 4:   d 1174
## 5:   e 1242
```

data.table

Assignment by reference.

```
dt[, prod:=x*y]
dt
```

```
##                 x        y let       prod
##     1: 74.492806 66.60992   a 4961.95987
##     2: 83.617783 27.58495   a 2306.59213
##     3:  3.245873 13.28463   d   43.12023
##     4: 31.960953 32.99187   e 1054.45149
##     5: 24.086816 81.34454   c 1959.33107
##    ---
##  9996: 80.536242 41.09855   c 3309.92293
##  9997: 72.212796 68.96125   b 4979.88494
##  9998: 32.479728 11.31963   a  367.65865
##  9999: 12.578302 12.61115   a  158.62681
## 10000: 45.768143 20.13591   d  921.58309
```