Outro

You, Me, and NSE

Non-standard evaluation is bananas!

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

```r
nseFun(oranges)
```

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

```r
nseFun(bananas)
```

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

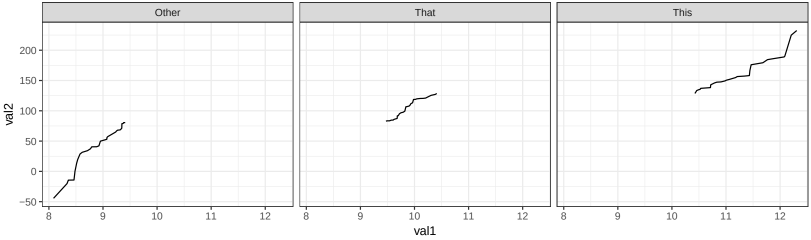
ggplot2 - Graphics using NSE

Really fast plotting with ggplot2.

```r
library(ggplot2)
dat <- data.frame(let = sample(letters[c(1:5,1:3,1:2)],
                               100, replace = T),
                  frm = sample(factor(c("This", "That", "Other")),
                               100, replace = T))
dat <- dat[order(dat$frm),]
dat$val1 <- sort(rnorm(100, mean = 10))
dat$val2 <- sort(rnorm(100, mean = 100, sd = 50))
```

ggplot2 - Graphics using NSE

```r
ggplot(dat) +
    geom_line(aes(val1, val2)) +
    facet_wrap(~frm) +
    theme_bw()
```



tidyverse

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

Includes ggplot2. Very easy and intuative for beginners.

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

```r
suppressMessages(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))
```

tidyverse

See how nicely it prints?

```
tib
```

```
## # A tibble: 10,000 x 3
##         x     y let
##     <dbl> <dbl> <chr>
##  1  81.6  8.37 c
##  2  92.9 90.7  a
##  3  15.6 75.5  e
##  4  75.2 14.5  a
##  5  97.6 52.4  b
##  6  97.5 17.6  a
##  7  35.7 53.6  c
##  8  40.0  1.37 b
##  9  95.1 45.8  a
## 10  11.6 72.1  b
## # ... with 9,990 more rows
```

tidyverse

Pipe-lining with magrittr: %>%

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

tib
```

```
## # A tibble: 7,557 x 4
##         x     y let    prod
##     <dbl> <dbl> <chr> <dbl>
##  1  81.6  8.37 c       683
##  2  92.9 90.7  a      8427
##  3  75.2 14.5  a      1088
##  4  97.6 52.4  b      5118
##  5  97.5 17.6  a      1714
##  6  35.7 53.6  c      1916
##  7  40.0  1.37 b        54.7
##  8  95.1 45.8  a      4359
##  9  11.6 72.1  b       834
```

data.table

Super fast tabular data manipulation with a very succinct syntax.

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

```r
suppressMessages(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 2511
## 2:   b 2496
## 3:   c 2569
## 4:   d 1169
## 5:   e 1255
```

data.table

Assignment by reference.

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

```
##                x         y let       prod
##     1: 32.422187 36.123468   b 1171.20182
##     2:  6.532529  8.252705   b   53.91103
##     3: 94.853497 56.187762   c 5329.60571
##     4: 94.727460 95.803954   d 9075.26521
##     5: 39.066377 42.305121   b 1652.70780
##    ---
##  9996: 81.158352 83.204138   a 6752.71072
##  9997:  2.875698 42.421539   d  121.99156
##  9998: 35.316338 69.473578   a 2453.55237
##  9999: 15.201988 44.432907   b  675.46852
## 10000: 56.116149 64.141454   c 3599.37141
```