Peeking Under the Hood

Malcolm Hawkes : CRUG Beginner's Night

- Generic functions and methods
- Accessing 'hidden' functions
- Calling C routines from R
- Data.table package (Matthew Dowle)

What function is acting on my data?

Generic Method : 'print'

Specific Methods : 'print.lm', 'print.data.frame', 'print.data.table'

```
> print
function (x, ...)
UseMethod("print")
<bytecode: 0x000000000b2c0f78>
<environment: namespace:base>
```

Method Dispatch: Depends on class e.g

```
> library(data.table)
> dt <- data.table(Id = 1:rnorm(260) Value = values,
                  Code = LETTERS[1:26])
> class(dt)
[1] "data.table" "data.frame"
```

print(dt) calls print.data.table(dt), if not then print.data.frame(dt) if not then print.default(dt)

<u>To see all specific methods for a generic:</u>

```
> methods(print)
  [1] print.acf*                                    print.anova
  [3] print.aov*                                    print.aovlist*
  [5] print.ar*                                     print.Arima*
```

```
[61] print.data.table*                             print.Date
```

```
[185] print.xngettext*                             print.xtabs*
```

```
   Non-visible functions are asterisked
```

Cool. Now I know what function is being called how do I see it ?

```
> print.data.table
Error: object 'print.data.table' not found
```

ARRRRRGHHHH What ? Why not ? Hang on it must be in the data.table package

Let's try "::" (to access variables in a 'namespace').

- Can be used to differentiate between different versions of a function with the same name in different packages
- Can be used to access functions in a package even if you haven't used library (or require) to load the package and put it on the search list

```
> data.table::print.data.table
Error: 'print.data.table' is not an exported object from 'namespace:da
ta.table'
```

ARRRRRGHHHH What ? Come on … Stupid R …… Hang on again. What's that stuff about 'is not exported' (and that little asterisk).

OK I know it's an S3 method. How about

```
> getS3method("print", "data.table")
function (x, topn = getOption("datatable.print.topn"), nrows = getOpti
on("datatable.print.nrows"),
    digits = NULL, ...)
```

YESSSSS !

We can call it this way too (although obviously a bit pointless)

```
> getS3method("print", "data.table")(dt)
            Id      Value Code
    1:       1 -1.0975703    A
    2:       2 -0.5498686    B
```

<u>There's another way to access thing not exported from a package ":::"</u>

```
> data.table:::print.data.table
function (x, topn = getOption("datatable.print.topn"), nrows = getOpti
on("datatable.print.nrows"),
    digits = NULL, ...)
```

And we can call it this way too

```
> data.table:::print.data.table(dt)
            Id      Value Code
    1:       1 -1.0975703    A
    2:       2 -0.5498686    B
```

OK I can see that ":::" is a nice shorthand way of getting a function in a package that is not exported. But what's the point? It gets called when I use the generic function print(dt).

The point: You can access any function hidden in a namespace with ":::", not just specific instances of generic S3 methods.

Why is that cool?

Functions not exported are often not exported for a reason, however, you can you them to access low-level C functions for speed gains.

Multiple-Regression Speed Example

```
> w <- rnorm(5)
> X <- cbind(1, matrix(rnorm(4 * 1000), ncol = 4))
> y <- X %*% w
```

```
> microbenchmark::microbenchmark(lm(y ~ X),
+                                lm.fit(X,y),
+                                .Call(stats:::C_Cdqrls, X, y, 1e-07))
Unit: microseconds
                                 expr    median
                            lm(y ~ X)  2130.334
                         lm.fit(X, y)  225.938
 .Call(stats:::C_Cdqrls, X, y, 1e-07)  138.552
```

A 20-fold increase in speed by cutting out the overhead … makes Cross-Validation a much more feasible option for say subset selection

Data Table Package

Authored by Matthew Dowle

- Fast subset,
- Fast grouping,
- Fast assign
- Fast ordered joins
- Short and flexible syntax, for faster development.

- 10+ times faster than `tapply()`
- 100+ times faster than ==
- 500+ times faster than `dt[i,j] <- value`

Inherits from data.frame so can be used in functions that can only handle data.frames

```
values <- rnorm(260000)
df <- data.frame(Id = 1:length(values), Value = values,
                 Code = LETTERS[1:26])
Dt <- as.data.table(df)
setkey(dt, Code)
```

Fast Subset

```
Unit: milliseconds
                 expr    median
 df[df$Code == "C", ] 49.489006
         dt["C"]       2.494625
```

Fast Grouping

```
Unit: milliseconds
                            expr        median
 tapply(df$Value, df$Code, mean) 171.681632
    dt[, mean(Value), by = Code]   9.445684
```

Fast Update

```
Unit: milliseconds
                              expr       median
 df[dt$Code == "A", "Value"] <- NA 42.266057
    dt["A", `:=`(Value, NA_real_)] 2.552175
```