R + essentials = Rsentials

Maks Chudzicki

02.12.2016

Who am I?

Education

- Chemistry
- Physics
- Programming by self-study

Professional

- Current: Data Scientist position at a steel trading company
- Project Manager/ Business analyst at a startup

Programming experience

- R
- bash
- Python
- C / C++ / Fortran

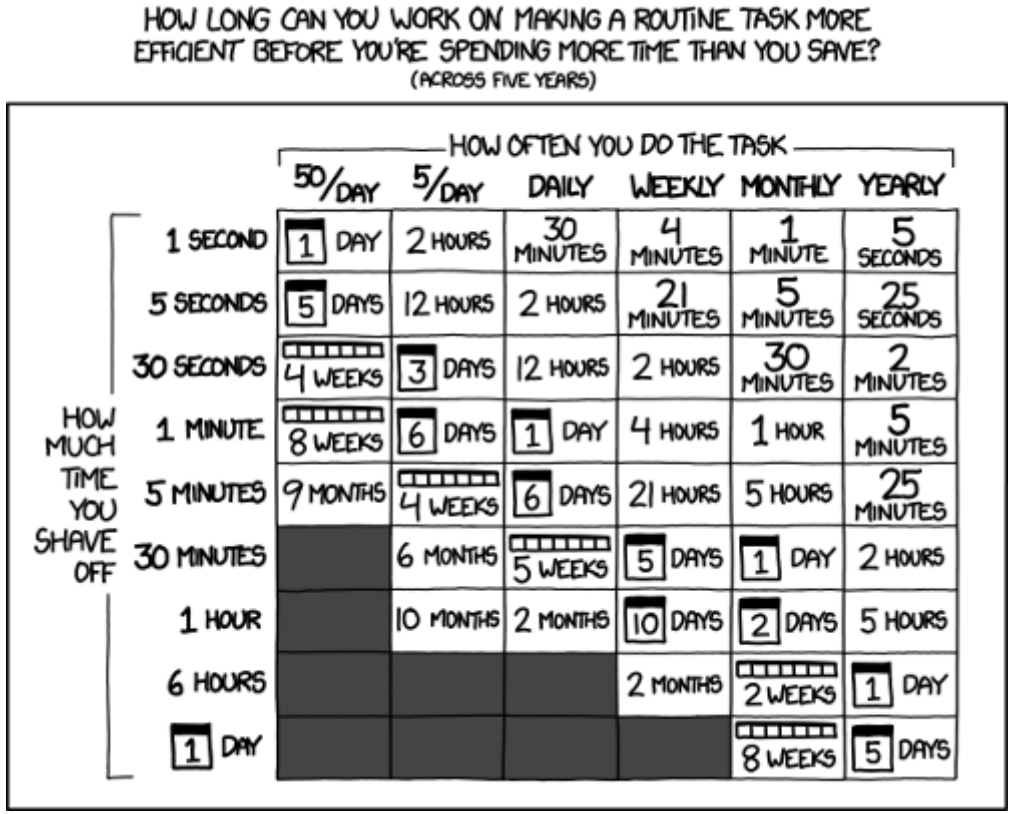
Affinities

- Data Science
- Getting things done efficiently

Why this talk?

Because less data-prepping = more time for modelling

Also, someone already made a package for {your task here}



R tool and package reference

Rstudio

- Why not do everything in Rstudio?

great packages

```
# tidyverse
library(dplyr)
library(tidyr)
library(stringr)
library(lubridate)
library(ggplot2)
library(...)
```

```
# graphics/visualisation
library(shiny)
library(DiagrammeR)
```

other great packages

```
# Piping
library(magrittr)

# Config files
library(yaml)

# Unit testing
library(testthat)

# pretty colors!
library(RColorBrewer)

# code profiling
library(Profvis)
```

What we will cover today

```
# Pipe!
library(magrittr)

# Dataframe manipulation verbs
library(dplyr)

# Pretty interactive webpages
library(shiny)

# code profiling
library(profvis)

# flowcharting
library(DiagrammeR)
```

datasets

mtcars and iris

```
head(iris, 4)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
```

```
head(mtcars, 4)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
```

magrittr

"Standard" R code

```
# Nested approach
  arrange(
    summarise(
      filter(data, foo == "bar"),
      Total = sum(foo)
      ),
    desc(Total)
  )

  # sequential approach
  a <- filter(data, foo == "bar")
  b <- summarise(a, Total = sum(foo))
  c <- arrange(b, desc(Total))
```

The Pipe

%>%
magrittr

Ceci n'est pas un pipe.

The Pipe %>%

- Pipe outout of one function to the first position of the next function call

```
# Pretty code
mtcars %>%
 filter(hp < 100)  # same as filter(mtcars, hp < 100)
 summarise(Total = sum(foo)) %>%
 arrange(desc(Total))


mtcars %>%
 filter(hp < 100, .data = .)

# Rstudio shortcut: ctrl+shift+m
%>%
```

dplyr

dplyr functions

a function for each basic verb of data manipulation

```
select() # NSE select columns
select_() # SE select columns

filter() # filter rows

group_by() # group by levels within column

summarise() # collapse and calculate metric

mutate() # add columns

arrange() # sort

join() # merge dataframes
```

select

select columns

```
# base-r
mtcars[, c("hp", "cyl", "mpg")]

# dplyr
mtcars %>% select(hp, cyl, mpg) # NSE
mtcars %>% select_(.dots = c("hp", "cyl", "mpg")) # SE
mtcars %>% select(-hp)
```

helper functions

```
# select using Regular expressions!
iris %>% select(matches("Petal|Species"))
iris %>% select(starts_with("Petal"))
iris %>% select(ends_with("Length"))
```

filter

filter rows

```
# base-r
iris[iris$Sepal.Length > 5, ]

# dplyr
iris %>% filter(Sepal.Length > 5)
```

helper functions

```
# remove duplicate rows
iris %>% distinct()

# sample 10 rows with replacement
iris %>% sample_n(10, replace = TRUE)

# select row number 10:15
iris %>% slice(10:15)
```

group_by and summarise

group by unique values and calculate group-metrics

```
# get mean and sd of Sepal length of each species
iris %>%
  group_by(Species) %>%
  summarise(mean = mean(Sepal.Length),
            sd = sd(Sepal.Length))

#      Species  mean        sd
#      <fctr> <dbl>     <dbl>
#     setosa 5.006 0.3524897
# versicolor 5.936 0.5161711
#  virginica 6.588 0.6358796
```

mutate

adding columns

```
# Add Petal and Sepal sum-columns
iris %>%
  mutate(PetalSum = Petal.Width + Petal.Length,
         SepalSum = Sepal.Width + Sepal.Length) %>%
  select(Species, PetalSum, SepalSum)

#   Species PetalSum SepalSum
#   setosa      1.6      8.6
#   setosa      1.6      7.9
#   setosa      1.5      7.9
```

Putting it all together

```
# apply group
mtcars %>%
  mutate(eff.cat = ifelse(mpg > 25, "efficient", "inefficient"))  %>%
  group_by(eff.cat, cyl) %>%
  summarize(avg.hp = mean(hp)) %>%
  arrange(eff.cat, cyl)

#     eff.cat   cyl   avg.hp
#       <chr> <dbl>    <dbl>
#   efficient     4  75.5000
# inefficient     4  91.2000
# inefficient     6 122.2857
# inefficient     8 209.2143
```

data.table

dplyr vs. data.table

Which one is better?

- different philosophies
  - dplyr uses verbs that can be chained
  - data.table has a more convoluted syntax

```
# "Take DT, subset rows using i, then calculate j grouped by by"
DT[i, j, by]
```

- dplyr is less memory-efficient
- data.table works like a database, i.e. it has indexes and outperforms dplyr

Shiny

Web Applications with Shiny

- Shiny is the easiest way to create webapps - **guaranteed**
- https://shiny.rstudio.com/gallery

```
library(shiny)
# Shiny app with 11 lines of code
n <- 200

# Define the UI
ui <- bootstrapPage(
  numericInput('n', 'Number of obs', n),
  plotOutput('plot')
)

# Define the server code
server <- function(input, output) {
  output$plot <- renderPlot({
    hist(runif(input$n))
  })
}

shinyApp(ui = ui, server = server)
```

Profvis

Fast R code

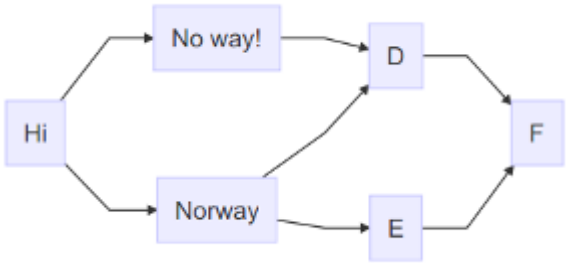
- Making R fast requires trial and error and and code profiliing
- but: Profiling R code is easy

```
library(profvis)
# Generate data
times <- 4e5
cols <- 150
data <- as.data.frame(x = matrix(rnorm(times * cols, mean = 5), ncol = cols))

profvis({
  data1 <- data   # Store in another variable for this run

  # Four different ways of getting column means
  means <- apply(data1[, names(data1)], 2, mean)
  means <- colMeans(data1[, names(data1)])
  means <- lapply(data1[, names(data1)], mean)
})
```

DiagrammeR

Easy flowcharting with mermaid

```
library(DiagrammeR)
mermaid("
graph LR
A[Hi!]
B[No way!]
C[Norway]
A-->B
A-->C
C-->E
B-->D
C-->D
E-->F")
```



Further reading

Helpful resources

- ultimate data science and machine learning compendium
  - https://github.com/acastrounis/data-science-machine-learning-ai-big-data-resources
- Hadley Wickham
  - books
    - Advanced R https://adv-r.had.co.nz/
    - R for Data Sciencer https://r4ds.had.co.nz
  - other works
    - https://hadley.github.io/
    - Various publications http://vita.had.co.nz/
- R project skeleton
  - https://github.com/sloria/r-lcfd