Introduction to dplyr and magrittr

Denver R Users Group

www.meetup.com/DenverRUG

Peter DeWitt

peter.dewitt@ucdenver.edu

1 July 2014

Goals:

- Showcase `dplyr`, compare the ease of use compared to base R.
- Introduce the data manipulation grammar and philosophy behind `dplyr`
- Illustrate the usefulness of the forward-piping operator which is part of `dplyr` and extended further in `magrittr`.

dplyr: a grammar of data manipulation

- Authored by Hadley Wickham and Romain Francois
- Current CRAN version 0.2

dplyr: a grammar of data manipulation

- Authored by Hadley Wickham and Romain Francois
- Current CRAN version 0.2
- Paraphrasing from a post on the RStudio blog http://blog.rstudio.org/2014/01/17/introducing-dplyr
  - dplyr is the next iteration of plyr
  - focuses only on data.frames
  - faster, thanks in part to Francois work in Rcpp, some use of multiple processors.
  - improved API.
  - interface with remote database (PostgreSQL, MySQL, SQLite, and Google bigquery) tables using the same verbs for interacting with data.frames. (Extendible to other backends)
  - Common operations:
    - group_by, summarize, mutate, filter, select, and arrange.

Data Import

`dplyr` does not have special tools for reading in data, but, if you need to `rbind` sets together...

```
# FAAs wildlife strikes on aircraft since 1990.  The data
# can be downloaded, in a Microsoft Access DB,  from
# http://www.faa.gov/airports/airport_safety/wildlife/database/
# Tables in the DB were exported to csv files.
# A data dictionary, in an Excel file, was also
# included in the download from faa.gov

# column classes are set (in R code not shown) to ensure
# that each column of the imported data is of the same class
wls.90.99 <-
  read.csv("../data/STRIKE_REPORTS (1990-1999).csv",
           colClasses = clclss)
wls.00.09 <-
  read.csv("../data/STRIKE_REPORTS (2000-2009).csv",
           colClasses = clclss)
wls.10.14 <-
  read.csv("../data/STRIKE_REPORTS (2010-Current).csv",
           colClasses = clclss)
```

Data Import

```
# Base does not require the columns to be of the same class,
# only the same name
# dplyr requires that the columnns are of the same class.
dim(wls.90.99)

## [1] 30150    94

nrow(wls.90.99) + nrow(wls.00.09) + nrow(wls.10.14)

## [1] 142911

bnchmrk <-
  benchmark(base = rbind(wls.90.99, wls.00.09, wls.10.14),
            dplyr = rbind_list(wls.90.99, wls.00.09, wls.10.14),
            replications = 100)
bnchmrk[, c("test", "replications", "elapsed", "relative")]

##    test replications elapsed relative
## 1  base          100   87.99    3.878
## 2 dplyr          100   22.69    1.000
```

Data Import

```
wls_df <- rbind(wls.90.99, wls.00.09, wls.10.14)
class(wls_df)

## [1] "data.frame"

wls <- rbind_list(wls.90.99, wls.00.09, wls.10.14)
class(wls)

## [1] "data.frame"

# A data frame tbl wraps a local data frame. The main
# advantage to using a tbl_df over a regular data frame is
# the printing: tbl objects only print a few rows and all
# the columns that fit on one screen, providing describing
# the rest of it as text. [source: R help doc]
wls_tbl_df <- tbl_df(wls)
class(wls_tbl_df)

## [1] "tbl_df"     "tbl"        "data.frame"
```

Data Printing

```
# print(wls_df)  # takes a long time, not helpful
# head(wls_df)   # two many columns to be useful
print(wls_tbl_df, n = 2)

## Source: local data frame [142,911 x 94]
##
##    INDEX_NR OPID          OPERATOR     ATYPE AMA AMO EMA EMO
## 1    100000  AAL AMERICAN AIRLINES     B-727 148  10  34  10
## 2    100001  UAL   UNITED AIRLINES B-737-300 148  24  10  01
## ..      ...  ...                ...       ... ... ... ... ...
## Variables not shown: AC_CLASS (chr), AC_MASS (int), NUM_ENGS
##   (chr), TYPE_ENG (chr), ENG_1_POS (chr), ENG_2_POS (int),
##   ENG_3_POS (chr), ENG_4_POS (int), REG (chr), FLT (chr),
##   REMAINS_COLLECTED (lgl), REMAINS_SENT (lgl), INCIDENT_DATE
##   (chr), INCIDENT_MONTH (int), INCIDENT_YEAR (int),
##   TIME_OF_DAY (chr), TIME (int), AIRPORT_ID (chr), AIRPORT
##   (chr), STATE (chr), FAAREGION (chr), ENROUTE (chr), RUNWAY
##   (chr), LOCATION (chr), HEIGHT (int), SPEED (int), DISTANCE
##   (dbl), PHASE_OF_FLT (chr), DAMAGE (chr), STR_RAD (lgl),
##   DAM_RAD (lgl), STR_WINDSHLD (lgl), DAM_WINDSHLD (lgl),
##   STR_NOSE (lgl), DAM_NOSE (lgl), STR_ENG1 (lgl), DAM_ENG1
##   (lgl), STR_ENG2 (lgl), DAM_ENG2 (lgl), STR_ENG3 (lgl),
```

magrittr: a forward-pipe operator for R

ceci n'est pas un pipe (this is not a pipe)

- dplyr funcationality is made more powerful via the %>%, or equivalently, \%.%$, operator.

magrittr: a foward-pipe operator for R

Examples

```
data(diamonds, package = "ggplot2")

# find the mean price of the diamonds
# Standard R syntax
mean(diamonds$price)

## [1] 3933

# with the pipe
diamonds %>%
extract("price") %>%
unlist() %>%
mean()

## [1] 3933
```

What's the point?

Reproducibility

The data, code, sides, etc. all at
`github.com/dewittpe/dplyr-demo`

```
print(sessionInfo(), locale = FALSE)

## R version 3.1.0 (2014-04-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets
## [6] methods   base
##
## other attached packages:
## [1] rbenchmark_1.0.0 dplyr_0.2         magrittr_1.0.1
## [4] knitr_1.6        vimcom_0.9-93     setwidth_1.0-3
## [7] colorout_1.0-3
##
## loaded via a namespace (and not attached):
##  [1] assertthat_0.1  codetools_0.2-8 digest_0.6.4
##  [4] evaluate_0.5.5  formatR_0.10    highr_0.3
##  [7] parallel_3.1.0  Rcpp_0.11.1     stringr_0.6.2
## [10] tools_3.1.0
```