Pipes or Brackets?

SatRday Conference

April 6, 2019

UCLA

Presented by Amy Linehan and Jeremy Guinta

What is this debate about?

Dplyr (Pipes, because we love them!)

- Created by Hadley (self appointed R god)
- https://twitter.com/hadleywickham

- dplyr loves pipes.
- dplyr is modular and allows for output to be piped from one function into the next.
- dplyr is a vast improvement over base R functions.
- dplyr is **ok** with large data.

Data.Table (Brackets, because we love them!)

- Created by Matt Dowle (who?)
- https://twitter.com/MattDowle

- data.table hates pipes but loves brackets
- data.table is modular and allows for output to be pushed from one function to the next.
- data.table is a vast improvement over base R functions.
- data.table is a must for large data.

What is this debate about?

- You will spend about 80%[1] of your time with the data, processing, cleaning, and doing basic analysis.
- In other words, 80% of your time getting your data ready for analysis and performing basic analysis. Not doing the fun stuff… (e.g., machine learning, visuals, etc…)
- We are here today to discuss, not which package or ecosystem is best for this process, but to discuss:
  - Costs and Benefits to using dplyr()
  - Costs and Benefits to using data.table()
  - How you can use both to maximize efficiency, processing, and speed of your code.

[1]https://www.forbes.com/sites/teradata/2015/01/16/your-math-is-all-wrong-flipping-the-8020-rule-for-analytics/#1b14f5626435

What is dplyr?

- The basics:

```
install.packages(dplyr)
require(dplyr)
```

What do you get in this package?

- Pipes, Pipes, and more Pipes.
- Simple, easy-to-follow code
- A complete ecosystem of data processing functions.

- Elongated code.
- Can be slow.

Pipes, Pipes & Pipes

filter(dta,value>100) ⇔ dta %>% filter(value>100)

- Can chain multiple actions into a single statement
- No need to wrap functions or save out intermediate steps

$$f(g(x)) = (f \circ g)(x)$$

dplyr Syntax

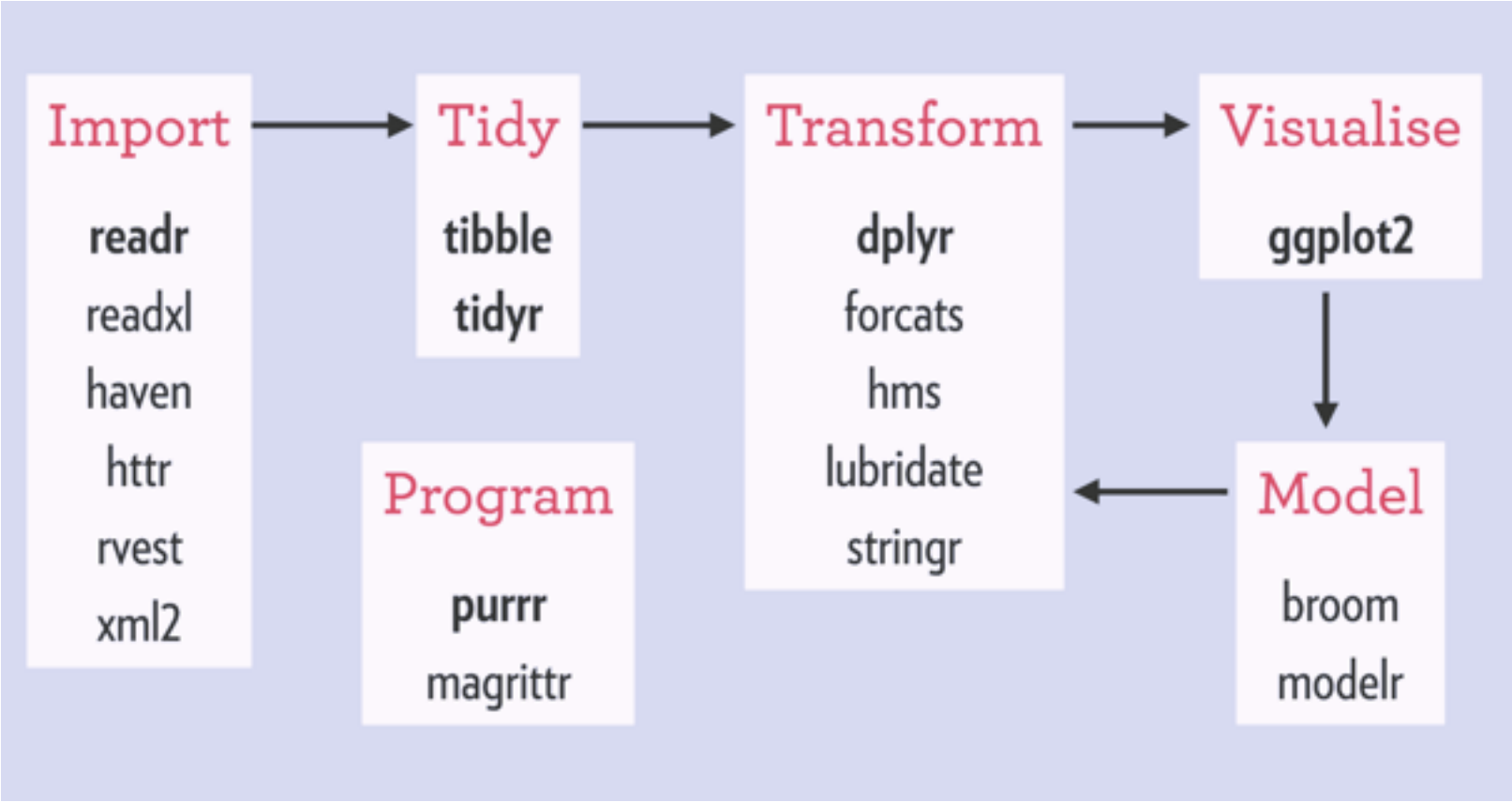
- 5 main verbs that perform most data manipulation
  - Select, Filter, Arrange, Mutate, Group by, Summarize
  - Other verbs… mutate_at, summarize_all, count etc.

- Relatively easy to follow for all skill levels

```
dta %>%
       filter(value>100) %>%# filter to over 100
       group_by(cat1,cat2) %>%
       summarize( cnt=n()
                 ,avg=mean(value)
                   )
```

Tidyverse Ecosystem

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures"



https://rviews.rstudio.com/2017/06/08/what-is-the-tidyverse/

What is data.table?

- The basics:

  install.packages(data.table)

  require(data.table)
- What do you get in this package?
  - Really, really, really, compact code!
  - Pure unadulterated speed.
  - More memory efficient.
  - A completely unintelligible syntax.

What is data.table?

- The basics (cont.)
  - What can I do with data.table()
    - Data Management
    - Data Processing
    - data.table() also interfaces nicely with dplyr and the tidyverse, so you can use (nearly) all of those functions and packages as well!
  - Syntax
    - dta[i,j,by]
    - i – filter (subsetting the data)
    - j – select (selecting variables and building new variables)
    - by – grouping (aggregate)
  - Piping: dta[i, j, by][i, j, by]
  - You can use existing r functions (i.e., merge) and get the benefits of data.table() without having to rewrite your code.

What is data.table?

- Really, really, really, compact code!
  - data.table() might be hard to interpret, but I hate typing

Example 1 – Summary Table

```
tbl1<-dta %>%
filter(cat_var==“blue”) %>%
group_by(grp_var) %>%
summarize(cnt=n()) %>%
mutate(pct=cnt/sum(cnt))
```

v.s.

```
tbl1<-dta[cat_var==“blue”, .N, grp_var][, pct:=N/sum(N)]
```

Example 2 – If Else Statement

```
dta<-dta %>%
mutate(new_var=
        ifelse(cat_var==“blue”, 1,
ifelse(cat_var==“red”, 2, 3)))
```

v.s.

```
dta[, new_var:=3]
dta[cat_var==“blue”, new_var:=1]
dta[car_var==“red”, new_var:=2]
```

What is data.table?

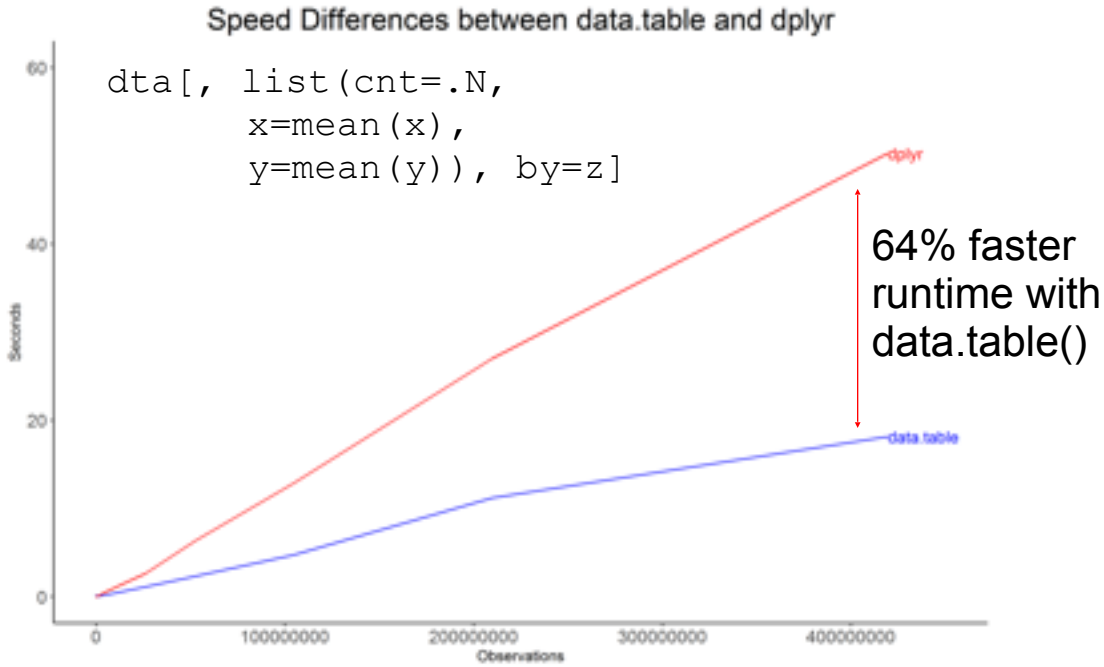
- Pure unadulterated speed
  - Operations that take minutes in base R or dplyr can take seconds in data.table()
  - Hadley has indicated that dplyr is “fast-enough” It is not![1]
  - Largest speed gains:
    - Reading, writing data (use fread(), instead of read_csv())
    - Variable creation and aggregation
  - data.table() is absolutely essential for massive datasets

1. https://stackoverflow.com/questions/27511604/dplyr-on-data-table-am-i-really-using-data-table/27520688#comment43492306_27520688

What is data.table?

- Pure unadulterated speed!

- Very clear difference in speed as the number of observations increase.
- This adds up!



ankura
COLLABORATION DRIVES RESULTS

What is data.table?

- More memory efficient.
  - Data table is optimized to handle memory more efficiently.
  - It allows for set operations to occur directly on the variable in memory.
  - For example, when creating a new variable
    - In data.table() you would use dta[, newvar:=oldvar+5]
    - The := operator performs the operation directly on the data in memory.
    - It is a set operation and is much faster than having to copy the data via the <- operator.

Side-by-Side Commands

data.table

```
1.dta[, .N, by=cat1]
```

Dplyr

```
1.dta %>% group_by(cat1) %>%
          summarize(n())
```

Side-by-Side Commands

data.table

```
2. dta[   value>100,
          list(cnt=.N, avg=mean(value)),
          by=list(cat1, cat2)]
```

Dplyr

```
2. dta %>% filter(value>100) %>%
    group_by(cat1,cat2) %>%
    summarize(cnt=n(), avg=mean(value))
```

Side-by-Side Commands

data.table

```
3. dta[   value>100,
          list(cnt=.N, avg=mean(value)),
          by=list(cat1, cat2)][cnt>100]
```

dplyr

```
3. dta %>% filter(value>100) %>%
    group_by(cat1,cat2) %>%
    summarize(cnt=n(),
              avg=mean(value)) %>%
    filter(cnt>100)
```

Side-by-Side Commands

data.table

```
4. setkey(dta1, cat1)
   setkey(dta2, cat1)
   dta1[dta2, nomatch=0]
```

dplyr

```
4. inner_join(dta1, dta2,
by=c("cat1"="cat1")
```

Side-by-Side Commands

data.table

```
5.  setname(dta2, "cat4", "cat2")
    setkey(dta1,cat1,cat2)
    setkey(dta2,cat1,cat2)
    dta1[dta2, ]
```

dplyr

```
5. left_join(dta1, dta2,
by=c("cat1"="cat1", "cat2"="cat4"))
```

Side-by-Side Commands

data.table

```
6. dta[, newvar:=ifelse(value>100, “Large”,
          “Small)]
```

dplyr

```
6. dta<-dta %>%
       mutate(newvar=ifelse(value>100,
       “Large”, “Small”))
```

Side-by-Side Commands

data.table

```
7. dta_w<-reshape(dta, idvar=c("ID"),
timevar=c("ord"), direction="wide")
```

dplyr

```
7. dta_w<-dta %>% spread(ord, value)
```

Example

Timekeeping

usrcode	src	evt_code	starttime_tz	finishtime_tz
ABbbb	3	1	10/17/2012 9:20	10/17/2012 9:20
ABbbb	3	2	10/17/2012 9:20	10/17/2012 10:16
ABbbb	3	2	10/17/2012 9:20	10/17/2012 9:20
ABbbb	3	2	10/17/2012 9:20	10/17/2012 9:20
ABbbb	3	2	10/17/2012 9:20	10/17/2012 9:20
ABbbb	3	2	10/17/2012 9:20	10/17/2012 9:20
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

User List

employeenumber	user_src1	user_src2	user_src3	user_src4	user_src5	user_src6
54	NA	NA	ABbbb	ABbbb	AaBbb	Aaaa Bbbb
69	NA	NA	NA	NA	NA	NA
98	NA	NA	NA	NA	NA	NA
114	NA	NA	CDddd	CDddd	CcDdd	Cccc Dddd
116	NA	NA	EFfff	EFfff	EeFff	Eeee Fffff
147	NA	NA	MNn	MNn	MmNn	Mmmm Nn

Example

dplyr

```
st <- Sys.time()
tmkp <- read_csv(file="tmkp.csv")
users <- read_csv(file="users.csv")
tmkp <- tmkp %>%
  mutate(date_st = str_extract(pattern="[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}",as.character(starttime_tz))) %>%
  mutate(date = as.Date(date_st,format= "%Y-%m%-%d"))
tmkp <- tmkp %>%
  left_join(as.data.frame(users),by=c("usrcode"="user_alt"))
end <- Sys.time()
dplyr <- end-st
```

```
> dplyr
Time difference of 6.576506 hours
```

data.table

```
st <- Sys.time()
tmkp <- fread(file="tmkp.csv")
users <- fread(file="users.csv")
tmkp[,date_st:=str_extract(pattern="[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}",as.character(starttime_tz))]
tmkp[,date := as.Date(date_st,format= "%Y-%m-%d")]
setkey(tmkp,usrcode)
setnames(users,old="user_alt",new="usrcode")
setkey(users,usrcode)
tmkp <- tmkp[users,on="usrcode"]
end <- Sys.time()
dt <- end-st
```

```
> dt
Time difference of 1.057563 hours
```

data.table is 501% faster (5 times faster) than dplyr

Example

Combined

```
st <- Sys.time()
tmkp <- fread(file="tmkp.csv")
users <- fread(file="users.csv")
tmkp <- as.data.frame(tmkp) %>%
  mutate(date_st = str_extract(pattern="[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}",as.character(starttime_tz))) %>%
  as.data.table()
tmkp[,date := as.Date(date_st,format= "%Y-%m-%d")]
setkey(tmkp,usrcode)
setnames(users,old="user_alt",new="usrcode")
setkey(users,usrcode)
tmkp <- tmkp[users,on="usrcode"]
end <- Sys.time()
combine <- end-st
```

Combined method is 1% faster than data.table and 628% faster than dplyr.

```
> combine
Time difference of 1.004628 hours
```

Questions?

Jeremy Guinta

jeremy.guinta@ankura.com

Amy Linehan

amy.linehan@ankura.com