Fast data munging in R

with plyr, dplyr & data.table

Alex Konduforov
Data Science Group Leader @ AltexSoft

Data Munging

Transformation of raw data to a usable format.

An important step in every Data Science project. Sometimes the most difficult and time-consuming.

What's in data munging

1. Subsetting / filtering data
2. Aggregating data
3. Sorting data
4. Merging data
5. Reshaping data
6. Type conversions, renaming, etc.

3 problems with data munging in R

1. It takes time to write code
2. Not easy sometimes (especially for novice)
3. Execution takes time for big datasets / complex operations

Main options in R

1. Basic R capabilities
2. Plyr
3. Dplyr
4. Data.table

NYC Flights data '13

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
1	2013	1	1	517	2	830	11	UA	N14228	1545	EWR	IAH	227	1400	5	17
2	2013	1	1	533	4	850	20	UA	N24211	1714	LGA	IAH	227	1416	5	33
3	2013	1	1	542	2	923	33	AA	N619AA	1141	JFK	MIA	160	1089	5	42
4	2013	1	1	544	-1	1004	-18	B6	N804JB	725	JFK	BQN	183	1576	5	44
5	2013	1	1	554	-6	812	-25	DL	N668DN	461	LGA	ATL	116	762	5	54
6	2013	1	1	554	-4	740	12	UA	N39463	1696	EWR	ORD	150	719	5	54
7	2013	1	1	555	-5	913	19	B6	N516JB	507	EWR	FLL	158	1065	5	55
8	2013	1	1	557	-3	709	-14	EV	N829AS	5708	LGA	IAD	53	229	5	57
9	2013	1	1	557	-3	838	-8	B6	N593JB	79	JFK	MCO	140	944	5	57
10	2013	1	1	558	-2	753	8	AA	N3ALAA	301	LGA	ORD	138	733	5	58
11	2013	1	1	558	-2	849	-2	B6	N793JB	49	JFK	PBI	149	1028	5	58
12	2013	1	1	558	-2	853	-3	B6	N657JB	71	JFK	TPA	158	1005	5	58
13	2013	1	1	558	-2	924	7	UA	N29129	194	JFK	LAX	345	2475	5	58
14	2013	1	1	558	-2	923	-14	UA	N53441	1124	EWR	SFO	361	2565	5	58
15	2013	1	1	559	-1	941	31	AA	N3DUAA	707	LGA	DFW	257	1389	5	59

Basic R: subsetting

```
flights[1:4, c("carrier", "origin", "dest", "dep_delay", "arr_delay")]
```

```
##   carrier origin dest dep_delay arr_delay
## 1      UA    EWR  IAH         2        11
## 2      UA    LGA  IAH         4        20
## 3      AA    JFK  MIA         2        33
## 4      B6    JFK  BQN        -1       -18
```

```
sub = subset(flights, origin == "EWR" & arr_delay == 0,
             select=c("carrier", "origin", "dest", "dep_delay", "arr_delay"))
head(sub, 4)
```

```
##     carrier origin dest dep_delay arr_delay
## 444      EV    EWR  CMH        -2         0
## 455      B6    EWR  TPA        -7         0
## 667      UA    EWR  SNA         4         0
## 900      EV    EWR  RDU        -3         0
```

Basic R: aggregating / sorting

```
aggr = aggregate(cbind(dep_delay, arr_delay) ~ carrier, data=flights, FUN=mean)
head(aggr[order(-aggr$arr_delay),], 4)
```

```
##    carrier dep_delay arr_delay
## 7       F9  20.20117  21.92070
## 8       FL  18.60598  20.11591
## 6       EV  19.83893  15.79643
## 16      YV  18.89890  15.55699
```

```
aggr = aggregate(arr_delay ~ origin + dest, data=flights, mean)
head(aggr[order(-aggr$arr_delay),], 4)
```

```
##     origin dest arr_delay
## 35     EWR  CAE  44.58511
## 220    EWR  TYS  41.15016
## 217    EWR  TUL  33.65986
## 142    EWR  OKC  30.61905
```

Basic R: merging

Merge() function

Supports all types of joins.

```
aggr = aggregate(arr_delay ~ carrier, data=flights, mean)
merg = merge(aggr, airlines, by.x = "carrier", by.y = "carrier")
head(merg, 8)
```

```
##   carrier  arr_delay                       name
## 1      9E  7.3796692           Endeavor Air Inc.
## 2      AA  0.3642909      American Airlines Inc.
## 3      AS -9.9308886        Alaska Airlines Inc.
## 4      B6  9.4579733             JetBlue Airways
## 5      DL  1.6443409        Delta Air Lines Inc.
## 6      EV 15.7964311    ExpressJet Airlines Inc.
## 7      F9 21.9207048      Frontier Airlines Inc.
## 8      FL 20.1159055 AirTran Airways Corporation
```

Basic R: Split-apply-combine

split() function + *apply family:
apply (matrix), lapply (list), sapply (simplify), tapply, mapply, etc.

```
sapply(
  split(flights, flights$carrier),
  function(x) mean(x$arr_delay, na.rm=T)
)
```

```
##         9E         AA         AS         B6         DL         EV
##  7.3796692  0.3642909 -9.9308886  9.4579733  1.6443409 15.7964311
##         F9         FL         HA         MQ         OO         UA
## 21.9207048 20.1159055 -6.9152047 10.7747334 11.9310345  3.5580111
##         US         VX         WN         YV
##  2.1295951  1.7644644  9.6491199 15.5569853
```

Plyr package

Powerful split-apply-combine framework

Why better than basic functions:

- totally consistent names, arguments and outputs
- input from and output to data.frames, matrices and lists
- progress bars to keep track of long running operations
- built-in error recovery, and informative error messages
- at least the same or better performance (parallelisation via foreach)

Plyr API

	array	data frame	list	nothing
array	aaply	adply	alply	a_ply
data frame	daply	ddply	dlply	d_ply
list	laply	ldply	llply	l_ply
n replicates	raply	rdply	rlply	r_ply
function arguments	maply	mdply	mlply	m_ply

Parameters:

- data - data to process
- variables - variables to split data by
- fun - function to apply to each piece
- parallel - run in parallel

Plyr: aggregating

Ddply() function example:

```
aggr = ddply(
  .data = flights,
  .variables = "carrier",
  .fun = function(p){
    summarize(p, dep_delay = mean(p$dep_delay, na.rm=T), arr_delay = mean(p$arr_delay, na.rm=T))
  }
)

aggr = ddply(flights, .(carrier), summarize,
             dep_delay = mean(dep_delay), arr_delay = mean(arr_delay))
```

Plyr: merging

Join() function

A bit faster than merge(), but less featureful. Supports inner, left, right, full joins.

```
aggr = aggregate(arr_delay ~ carrier, data=flights, mean)
merg = join(aggr, airlines, by="carrier", type="inner")
head(merg, 8)
```

Dplyr package

Next iteration of plyr from the same author (Hadley Wickham).

Differences:

- only dataframes are supported
- much faster than plyr because of C++ implementation
- new simplified API and syntax
- rich functionality
- works with databases

Dplyr API

tbl_df class

SQL-like functions:

- filter(), slice() - where
- group_by(), summarize() - group by
- arrange() - order by
- select(), rename() - select
- mutate() - new column
- head() - top
- distinct() - distinct

%>% - chain/piping operator

Dplyr: row subsetting

```
flights.df = tbl_df(flights)
sub = filter(flights.df, origin=="JFK" & arr_delay==0)
head(sub, 5)
```

```
## Source: local data frame [5 x 16]
##
##   year month day dep_time dep_delay arr_time arr_delay carrier tailnum
## 1 2013     1   1      627        -3     1018         0      US  N535UW
## 2 2013     1   1      807        -3     1043         0      DL  N308DE
## 3 2013     1   1     1240         5     1415         0      MQ  N828MQ
## 4 2013     1   1     1333        -2     1608         0      B6  N306JB
## 5 2013     1   1     1714        -6     1915         0      AA  N3CVAA
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Dplyr: column subsetting

```
sub = flights.df %>%
  filter(origin=="JFK" & arr_delay==0) %>%
  select(carrier, origin, dest, dep_delay, arr_delay)
head(sub, 5)
```

```
## Source: local data frame [5 x 5]
##
##   carrier origin dest dep_delay arr_delay
## 1      US    JFK  PHX        -3         0
## 2      DL    JFK  ATL        -3         0
## 3      MQ    JFK  RDU         5         0
## 4      B6    JFK  JAX        -2         0
## 5      AA    JFK  ORD        -6         0
```

Dplyr: aggregating / sorting

```
flights.df %>%
  filter(origin=="JFK") %>%
  group_by(carrier) %>%
  summarize(dep_delay = mean(dep_delay, na.rm=T), arr_delay = mean(arr_delay, na.rm=T)) %>%
  mutate(delay = dep_delay - arr_delay) %>%
  arrange(desc(delay)) %>%
  filter(delay > 6)
```

```
## Source: local data frame [5 x 4]
##
##   carrier dep_delay arr_delay     delay
## 1      HA  4.900585 -6.915205 11.815789
## 2      DL  8.333188 -2.379250 10.712438
## 3      VX 13.279441  2.827722 10.451719
## 4      9E 19.001517  8.843327 10.158190
## 5      AA 10.302155  2.081250  8.220905
```

Dplyr: merging

Mutating joins:

- inner_join()
- left_join(), right_join()
- full_join()

Filtering joins:

- semi_join() - all rows in A that have a match in B
- anti_join() - all rows in A that don't have a match in B

Set operations:

- intersect() - rows that appear in both A and B
- union() - rows that appear in either or both A and B
- setdiff() - rows that appear in A but not B

Data.table package

Extension of data.frame created to reduce both programming and compute time.

Major differences:

- completely different and slightly complicated (-) syntax
- very rich functionality
- allows creating keys / indexes for faster computation
- compliant with all R functions working with data.frame

Data.table API

data.table class

DT[i, j, by] command:

- i - where
- j - select
- by - group by

Take DT, subset rows using "i", then calculate "j" grouped by "by"

A lot of additional functions are available.

Data.table: subsetting

```
flights.dt = data.table(flights)
sub = flights.dt[origin=="JFK" & arr_delay==0, .(carrier, origin, dest, dep_delay, arr_delay)]
head(sub, 5)
```

```
##    carrier origin dest dep_delay arr_delay
## 1:      US    JFK  PHX        -3         0
## 2:      DL    JFK  ATL        -3         0
## 3:      MQ    JFK  RDU         5         0
## 4:      B6    JFK  JAX        -2         0
## 5:      AA    JFK  ORD        -6         0
```

```
flights.dt[carrier %in% c("US","DL")]
```

Data.table: aggregating

```
sub = flights.dt[origin=="JFK",
          .(dep_delay = mean(dep_delay, na.rm=T), arr_delay = mean(arr_delay, na.rm=T)),
          by=.(carrier)]
head(sub, 3)
```

```
##    carrier dep_delay arr_delay
## 1:      AA  10.30216  2.081250
## 2:      B6  12.75745  8.893702
## 3:      UA   7.90000  2.510496
```

```
head(flights.dt[origin=="JFK", .N, by=.(carrier)], 3)
```

```
##    carrier     N
## 1:      AA 13783
## 2:      B6 42076
## 3:      UA  4534
```

Data.table: adding/updating columns using :=

```
flights.dt[, delay := dep_delay - arr_delay,]
```

```
flights.dt[, delay := NULL,]
```

Data.table: indexing and keys

```
setkey(flights.dt, tailnum)
```

```
head(flights.dt["N14228", .(carrier, tailnum, flight)], 3)
```

```
##    carrier tailnum flight
## 1:      UA  N14228   1545
## 2:      UA  N14228   1579
## 3:      UA  N14228   1142
```

Data.table: chaining commands

```
flights.dt[, .(arr_delay = mean(arr_delay, na.rm=T)), by=carrier][arr_delay>10][order(-arr_delay)]
```

```
##    carrier arr_delay
## 1:      F9  21.92070
## 2:      FL  20.11591
## 3:      EV  15.79643
## 4:      YV  15.55699
## 5:      OO  11.93103
## 6:      MQ  10.77473
```

```
head(flights.dt["N14228", .(carrier, tailnum, flight)], 3)
```

```
##    carrier tailnum flight
## 1:      UA  N14228   1545
## 2:      UA  N14228   1579
## 3:      UA  N14228   1142
```

Speed: base R

```
system.time(aggr <- aggregate(cbind(dep_delay, arr_delay) ~ tailnum, data=flights, mean))
```

```
##    user  system elapsed
##    0.86    0.04    0.94
```

```
dim(aggr)
```

```
## [1] 4037    3
```

```
system.time(aggr <- aggregate(cbind(dep_delay, arr_delay) ~ carrier, data=flights, mean))
```

```
##    user  system elapsed
##    0.41    0.01    0.42
```

Speed: plyr

```
system.time(aggr <- ddply(flights, .(tailnum), summarize,
            dep_delay = mean(dep_delay, na.rm=T), arr_delay = mean(arr_delay, na.rm=T)))
```

```
##    user  system elapsed
##    2.93    0.02    2.99
```

```
dim(aggr)
```

```
## [1] 4044    3
```

```
system.time(aggr <- ddply(flights, .(carrier), summarize,
            dep_delay = mean(dep_delay, na.rm=T), arr_delay = mean(arr_delay, na.rm=T)))
```

```
##    user  system elapsed
##    0.11    0.01    0.12
```

Speed: dplyr

```
system.time(aggr <- flights.df %>%
  group_by(tailnum) %>%
  summarize(dep_delay = mean(dep_delay, na.rm=T), arr_delay = mean(arr_delay, na.rm=T)))
```

```
##    user  system elapsed
##    0.31    0.00    0.47
```

```
dim(aggr)
```

```
## [1] 4044    3
```

Speed: data.table

```
flights.dt = data.table(flights)
system.time(aggr <- flights.dt[,
    .(dep_delay = mean(dep_delay, na.rm=T), arr_delay = mean(arr_delay, na.rm=T)),
    by=.(tailnum)])
```

```
##    user  system elapsed
##    0.02    0.00    0.02
```

```
dim(aggr)
```

```
## [1] 4044    3
```

Thank you for listening

AI Ukraine 2015, Kharkiv, Sep 2015
http://aiukraine.com

alex_konduforov
@konduforov
http://merle-amber.blogspot.com
http://aikharkov.wordpress.com