

dplyr

Jeffrey Leek

May 18, 2016

The data frame is a key data structure in statistics and in R.

- ▶ There is one observation per row
- ▶ Each column represents a variable or measure or characteristic
- ▶ Primary implementation that you will use is the default R implementation
- ▶ Other implementations, particularly relational databases systems

dplyr

- ▶ Developed by Hadley Wickham of RStudio
- ▶ An optimized and distilled version of `plyr` package (also by Hadley)
- ▶ Does not provide any “new” functionality per se, but **greatly** simplifies existing functionality in R
- ▶ Provides a “grammar” (in particular, verbs) for data manipulation
- ▶ Is **very** fast, as many key operations are coded in C++

dplyr Verbs

- ▶ `select`: return a subset of the columns of a data frame
- ▶ `filter`: extract a subset of rows from a data frame based on logical conditions
- ▶ `arrange`: reorder rows of a data frame
- ▶ `rename`: rename variables in a data frame
- ▶ `mutate`: add new variables/columns or transform existing variables
- ▶ `summarise` / `summarize`: generate summary statistics of different variables in the data frame, possibly within strata

There is also a handy `print` method that prevents you from printing a lot of data to the console.

dplyr Properties

- ▶ The first argument is a data frame.
- ▶ The subsequent arguments describe what to do with it, and you can refer to columns in the data frame directly without using the \$ operator (just use the names).
- ▶ The result is a new data frame
- ▶ Data frames must be properly formatted and annotated for this to all be useful

Load the dplyr package

This step is important!

```
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##      filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##      intersect, setdiff, setequal, union
```

select

```
chicago <- readRDS("chicago.rds")  
dim(chicago)
```

```
## [1] 6940      8
```

```
head(select(chicago, 1:5))
```

##	city	tmpd	dptp	date	pm25tmean2
## 1	chic	31.5	31.500	1987-01-01	NA
## 2	chic	33.0	29.875	1987-01-02	NA
## 3	chic	33.0	27.375	1987-01-03	NA
## 4	chic	29.0	28.625	1987-01-04	NA
## 5	chic	32.0	28.875	1987-01-05	NA
## 6	chic	40.0	35.125	1987-01-06	NA

```
select
```

```
names(chicago)[1:3]
```

```
## [1] "city" "tmpd" "dptp"
```

```
head(select(chicago, city:dptp))
```

```
##   city tmpd  dptp
## 1 chic 31.5 31.500
## 2 chic 33.0 29.875
## 3 chic 33.0 27.375
## 4 chic 29.0 28.625
## 5 chic 32.0 28.875
## 6 chic 40.0 35.125
```


select

In dplyr you can do

```
head(select(chicago, -(city:dptp)))
```

Equivalent base R

```
i <- match("city", names(chicago))  
j <- match("dptp", names(chicago))  
head(chicago[, -(i:j)])
```

filter

```
chic.f <- filter(chicago, pm25tmean2 > 30)
head(select(chic.f, 1:3, pm25tmean2), 10)
```

##	city	tmpd	dptp	pm25tmean2
## 1	chic	23	21.9	38.10
## 2	chic	28	25.8	33.95
## 3	chic	55	51.3	39.40
## 4	chic	59	53.7	35.40
## 5	chic	57	52.0	33.30
## 6	chic	57	56.0	32.10
## 7	chic	75	65.8	56.50
## 8	chic	61	59.0	33.80
## 9	chic	73	60.3	30.30
## 10	chic	78	67.1	41.40

filter

```
chic.f <- filter(chicago, pm25tmean2 > 30 & tmpd > 80)  
head(select(chic.f, 1:3, pm25tmean2, tmpd), 10)
```

##	city	tmpd	dptp	pm25tmean2
## 1	chic	81	71.2	39.6000
## 2	chic	81	70.4	31.5000
## 3	chic	82	72.2	32.3000
## 4	chic	84	72.9	43.7000
## 5	chic	85	72.6	38.8375
## 6	chic	84	72.6	38.2000
## 7	chic	82	67.4	33.0000
## 8	chic	82	63.5	42.5000
## 9	chic	81	70.4	33.1000
## 10	chic	82	66.2	38.8500

arrange

Reordering rows of a data frame (while preserving corresponding order of other columns) is normally a pain to do in R.

```
chicago <- arrange(chicago, date)
head(select(chicago, date, pm25tmean2), 3)
```

```
##           date pm25tmean2
## 1 1987-01-01      NA
## 2 1987-01-02      NA
## 3 1987-01-03      NA
```

```
tail(select(chicago, date, pm25tmean2), 3)
```

```
##           date pm25tmean2
## 6938 2005-12-29    7.45000
## 6939 2005-12-30   15.05714
## 6940 2005-12-31   15.00000
```

arrange

Columns can be arranged in descending order too.

```
chicago <- arrange(chicago, desc(date))  
head(select(chicago, date, pm25tmean2), 3)
```

```
##           date pm25tmean2  
## 1 2005-12-31    15.00000  
## 2 2005-12-30    15.05714  
## 3 2005-12-29     7.45000
```

```
tail(select(chicago, date, pm25tmean2), 3)
```

```
##           date pm25tmean2  
## 6938 1987-01-03         NA  
## 6939 1987-01-02         NA  
## 6940 1987-01-01         NA
```

rename

Renaming a variable in a data frame in R is surprising hard to do!

```
head(chicago[, 1:5], 3)
```

```
##    city tmpd dptp      date pm25tmean2
## 1 chic    35 30.1 2005-12-31    15.00000
## 2 chic    36 31.0 2005-12-30    15.05714
## 3 chic    35 29.4 2005-12-29     7.45000
```

```
chicago <- rename(chicago, dewpoint = dptp,
                    pm25 = pm25tmean2)
head(chicago[, 1:5], 3)
```

```
##    city tmpd dewpoint      date    pm25
## 1 chic    35     30.1 2005-12-31 15.00000
## 2 chic    36     31.0 2005-12-30 15.05714
## 3 chic    35     29.4 2005-12-29  7.45000
```

mutate

```
chicago <- mutate(chicago,  
                    pm25detrend=pm25-mean(pm25, na.rm=TRUE))  
head(select(chicago, pm25, pm25detrend))
```

##	pm25	pm25detrend
## 1	15.00000	-1.230958
## 2	15.05714	-1.173815
## 3	7.45000	-8.780958
## 4	17.75000	1.519042
## 5	23.56000	7.329042
## 6	8.40000	-7.830958

group_by

Generating summary statistics by stratum

```
chicago <- mutate(chicago,
                    tempcat = factor(1 * (tmpd > 80),
                                     labels = c("cold", "hot"))
hotcold <- group_by(chicago, tempcat)
summarize(hotcold, pm25 = mean(pm25, na.rm = TRUE),
           o3 = max(o3tmean2),
           no2 = median(no2tmean2))
```

```
## Source: local data frame [3 x 4]
```

```
##
```

##	tempcat	pm25	o3	no2
##	(fctr)	(dbl)	(dbl)	(dbl)
## 1	cold	15.97807	66.587500	24.54924
## 2	hot	26.48118	62.969656	24.93870
## 3	NA	47.73750	9.416667	37.44444

group_by

Generating summary statistics by stratum

```
chicago <- mutate(chicago,  
                    year = as.POSIXlt(date)$year + 1900)  
years <- group_by(chicago, year)  
summarize(years, pm25 = mean(pm25, na.rm = TRUE),  
           o3 = max(o3tmean2, na.rm = TRUE),  
           no2 = median(no2tmean2, na.rm = TRUE))
```

```
## Source: local data frame [19 x 4]
```

```
##
```

```
##      year      pm25      o3      no2
```

```
##      (dbl)    (dbl)    (dbl)    (dbl)
```

```
## 1  1987      NaN 62.96966 23.49369
```

```
## 2  1988      NaN 61.67708 24.52296
```

```
## 3  1989      NaN 59.72727 26.14062
```

```
## 4  1990      NaN 52.22917 22.59583
```

```
## 5  1991      NaN 63.10417 21.38194
```

```
## 6  1992      NaN 50.00000 24.70000
```

%>%

```
chicago %>% mutate(month = as.POSIXlt(date)$mon + 1)
%>% group_by(month)
%>% summarize(pm25 = mean(pm25, na.rm = TRUE),
               o3 = max(o3tmean2, na.rm = TRUE),
               no2 = median(no2tmean2, na.rm = TRUE))
```

```
## Source: local data frame [12 x 4]
```

```
##
```

##	month	pm25	o3	no2
##	(dbl)	(dbl)	(dbl)	(dbl)
## 1	1	17.76996	28.22222	25.35417
## 2	2	20.37513	37.37500	26.78034
## 3	3	17.40818	39.05000	26.76984
## 4	4	13.85879	47.94907	25.03125
## 5	5	14.07420	52.75000	24.22222
## 6	6	15.86461	66.58750	25.01140
## 7	7	16.57087	59.54167	22.38442
## 8	8	16.93380	53.96701	22.98333

Once you learn the dplyr “grammar” there are a few additional benefits

- ▶ dplyr can work with other data frame “backends”
- ▶ `data.table` for large fast tables
- ▶ SQL interface for relational databases via the DBI package