Using the dplyr package

Steve Pittard wsp@emory.edu

November 2, 2015

Motivations

This slide deck owes much to "Becoming a data ninja with dplyr" which can be found at `https://speakerdeck.com/dpastoor/becoming-a-data-ninja-with-dplyr`

Motivations

- A data frame is a set of columns. Every column is same length but of possibly different types.
- It has characteristics of both a matrix, (each row is the same data type),
- Each column can be a different data type
- Bracket notation offers a convenient way to search through the data drame

Motivations

There are some common activites associated with a data frame:

- filter - find observations satisfying some condition(s)
- select - selecting specific columns by name
- mutate - adding new columns or changing existing ones
- arrange - reorder or sort the rows
- summarize - do some aggregation or summary by groups

Motivations

```
df <- data.frame(id = 1:5,
                 gender = c("MALE","MALE","FEMALE","MALE","FEMALE"),
                 age = c(70,76,60,64,68))
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

Filter

```
filter(df,gender == "FEMALE")
  id gender age
1  3 FEMALE  60
2  5 FEMALE  68
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	AGE
3	FEMALE	60
5	FEMALE	68

Filter

```
filter(df, id %in% c(1,3,5))
  id gender age
1  1   MALE  70
2  3 FEMALE  60
3  5 FEMALE  68
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	AGE
1	MALE	70
3	FEMALE	60
5	FEMALE	68

Mutate

Mutate is used to add or remove columns in a data frame

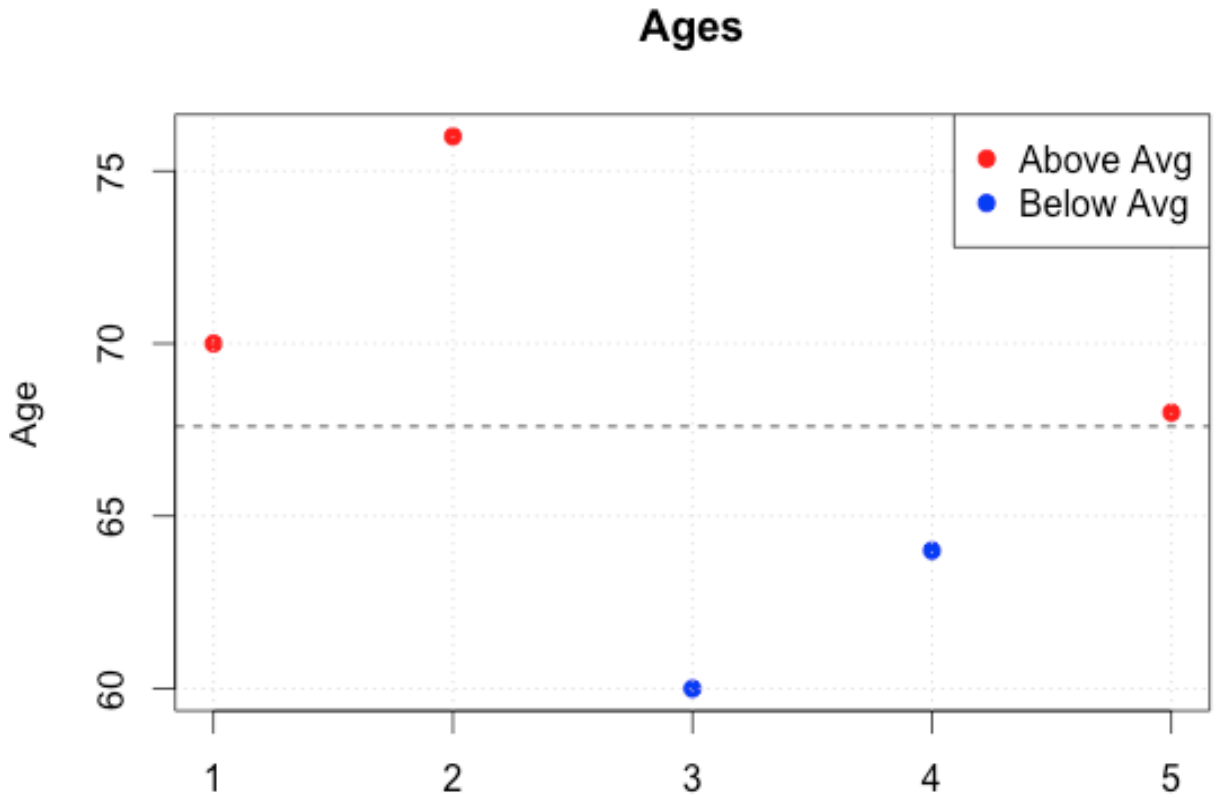
```
mutate(df,meanage = mean(age))
  id gender age meanage
1  1   MALE  70     67.6
2  2   MALE  76     67.6
3  3 FEMALE  60     67.6
4  4   MALE  64     67.6
5  5 FEMALE  68     67.6
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	AGE	MEANWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

Mutate

```
tmp <- mutate(df, color = ifelse(age > mean(age),"red","blue"))
plot(tmp$age,col=tmp$color,type="p",pch=19,main="Ages",ylab="Age")
grid()
abline(h=mean(tmp$age),lty=2)
legend("topright",c("Above Avg","Below Avg"),col=c("red","blue"),pch=19)
```



Arrange

Use arrange for sorting the data frame by a column(s)

```
# Sort df by age from highest to lowest

arrange(df, desc(age))
  id gender age
1  2   MALE  76
2  1   MALE  70
3  5 FEMALE  68
4  4   MALE  64
5  3 FEMALE  60

# Sort df by gender (alphabetically) and then by age
# from highest to lowest

arrange(df, gender,desc(age))
  id gender age
1  5 FEMALE  68
2  3 FEMALE  60
3  2   MALE  76
4  1   MALE  70
5  4   MALE  64
```

Select

Select allows us to select groups of columns from a data frame

```
select(df,gender,id,age)  # Reorder the columns
  gender id age
1   MALE  1  70
2   MALE  2  76
3 FEMALE  3  60
4   MALE  4  64
5 FEMALE  5  68

select(df,-age)   # Select all but the age column
  id gender
1  1   MALE
2  2   MALE
3  3 FEMALE
4  4   MALE
5  5 FEMALE

select(df,id:age)  # Can use : to select a range
  id gender age
1  1   MALE  70
2  2   MALE  76
3  3 FEMALE  60
4  4   MALE  64
5  5 FEMALE  68
```

Select

You can select by regular expressions or numeric paterns

```
library(ggplot2)
data(diamonds)
names(diamonds)
 [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
 [8] "x"       "y"       "z"

head(select(diamonds,starts_with("c")))
  carat       cut color clarity
1  0.23     Ideal     E     SI2
2  0.21   Premium     E     SI1
3  0.23      Good     E     VS1
4  0.29   Premium     I     VS2
5  0.31      Good     J     SI2
6  0.24 Very Good     J    VVS2

head(select(diamonds,ends_with("t")))
  carat       cut
1  0.23     Ideal
2  0.21   Premium
3  0.23      Good
4  0.29   Premium
5  0.31      Good
6  0.24 Very Good
```

Select

You can select by regular expressions or numeric paterns

```
testdf <- expand.grid(m_1=seq(60,70,10),age=c(25,32),m_2=seq(50,60,10))

head(testdf, 4)
  m_1 age m_2
1  60  25  50
2  70  25  50
3  60  32  50
4  70  32  50

head( select(testdf,matches("_")) ,2)
  m_1 m_2
1  60  50
2  70  50

head( select(testdf,contains("_"), 2)
  m_1 m_2
1  60  50
2  70  50

head( select(testdf,num_range("m_",1:2)), 2)
  m_1 m_2
1  60  50
2  70  50
```

group_by

group_by let's you organize a data frame by some factor or grouping variable

```
df
  id gender age
1  1   MALE  70
2  2   MALE  76
3  3 FEMALE  60
4  4   MALE  64
5  5 FEMALE  68

group_by(df,gender)   # Hmm. Did this really do anything ?

Source: local data frame [5 x 3]
Groups: gender

  id gender age
1  1   MALE  70
2  2   MALE  76
3  3 FEMALE  60
4  4   MALE  64
5  5 FEMALE  68
```

group_by

group_by let's you organize a data frame by some factor or grouping variable

```
df
  id gender age
1  1   MALE  70
2  2   MALE  76
3  3 FEMALE  60
4  4   MALE  64
5  5 FEMALE  68

( gdf <- group_by(df,gender)    # Hmm. Did this really do anything ?

Source: local data frame [5 x 3]
Groups: gender

  id gender age
1  1   MALE  70
2  2   MALE  76
3  3 FEMALE  60
4  4   MALE  64
5  5 FEMALE  68
```

Summarize

```
summarize(group_by(df,gender),total=n())
Source: local data frame [2 x 2]

  gender total
1 FEMALE     2
2   MALE     3
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

GENDER	TOTAL
FEMALE	2
MALE	3

Summarize

```
summarize(group_by(df,gender),av_age=mean(age))
Source: local data frame [2 x 2]

  gender av_age
1 FEMALE     64
2   MALE     70
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

GENDER	AV_AGE
FEMALE	64
MALE	70

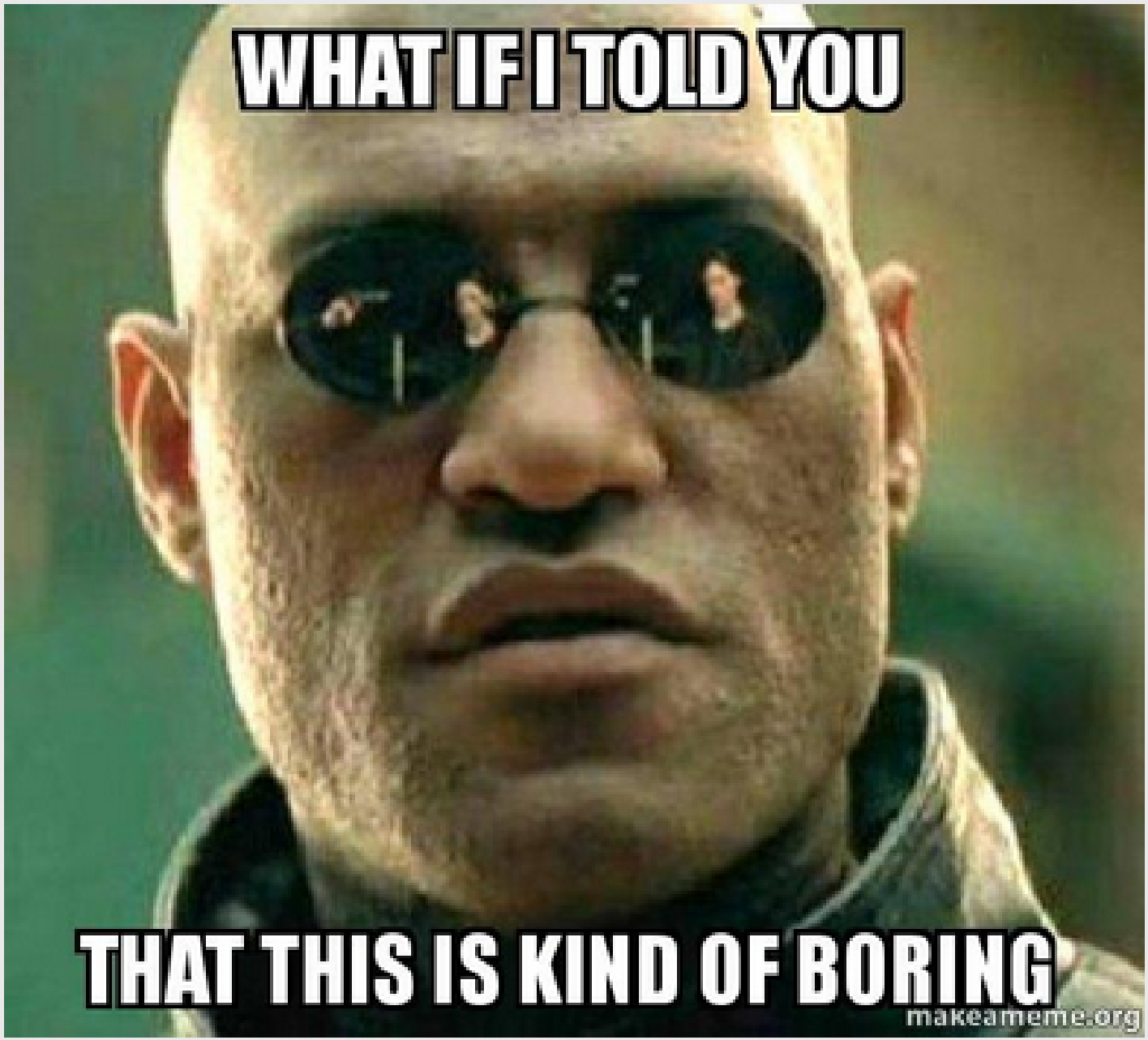
Summarize

```
summarize(group_by(df,gender),av_age=mean(age),total=n())
Source: local data frame [2 x 3]

  gender av_age total
1 FEMALE      64     2
2   MALE      70     3
```

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

GENDER	AV_AGE	TOTAL
FEMALE	64	2
MALE	70	3


WHAT IF I TOLD YOU
THAT THIS IS KIND OF BORING
makeameme.org

Split -> Apply -> Combine

Split -> Apply -> Combine group_by

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	AGE
1	MALE	70
2	MALE	76
4	MALE	64

ID	GENDER	AGE
3	FEMALE	60
5	FEMALE	68

AVG
70

AVG
64

ID	GENDER	AVG
1	MALE	70
2	FEMALE	64

Split -> Apply -> Combine

But do you really need dplyr to do this ? No but it makes it a lot easier

```
df
  id gender age
1  1   MALE  70
2  2   MALE  76
3  3 FEMALE  60
4  4   MALE  64
5  5 FEMALE  68

tapply(df$age,df$gender,mean)    # tapply function
FEMALE   MALE
    64     70

aggregate(age~gender,data=df,mean) # aggregate works also
 gender age
1 FEMALE  64
2   MALE  70

lapply(split(df,df$gender),function(x) mean(x$age)) # complicated
$FEMALE
[1] 64

$MALE
[1] 70
```

Split -> Apply -> Combine: Chaining

What about this ? We can chain together the output of one command to the input of another !

```
df %>% group_by(gender) %>% summarize(avg=mean(age))
Source: local data frame [2 x 2]

  gender avg
1 FEMALE  64
2   MALE  70

df %>% group_by(gender) %>% summarize(avg=mean(age),total=n())
Source: local data frame [2 x 3]

  gender avg total
1 FEMALE  64     2
2   MALE  70     3

df %>% filter(gender == "MALE") %>% summarize(med_age=median(age))
  med_age
1      70
```

Split -> Apply -> Combine: Chaining

What about this ? We can chain together the output of one command to the input of another !

```
df %>% filter(gender == "MALE") %>% summarize(med_age=median(age))
  med_age
1      70
```

df filter summarize

ID	GENDER	AGE
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	AGE
1	MALE	70
2	MALE	76
4	MALE	64

med_age
70

Split -> Apply -> Combine: Chaining

Using the built in mtcars dataframe filter out records where the wt is greater than 3.3 tons.

Then create a column called ab_be (Y or N) that indicates whether that observation's mpg is greater (or not) than the average mpg for the filtered set.

Then present the average mpg for each group

```
mtcars %>% filter(wt > 3.3)  %>%
         mutate(ab_be=ifelse(mpg > mean(mpg),"Y","N")  ) %>%
         group_by(ab_be) %>% summarize(mean_mpg=mean(mpg))
```

```
Source: local data frame [2 x 2]

  ab_be mean_mpg
1     N 13.77778
2     Y 18.10000
```

Split -> Apply -> Combine: Chaining

Using the built in mtcars dataframe filter out records where the wt is greater than 3.3 tons.

```
mtcars %>% filter(wt > 3.3)
```

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
2  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
3  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
4  19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
5  17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
6  16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
7  17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
8  15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
9  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
10 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
11 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
12 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
13 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
14 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
15 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
16 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

Split -> Apply -> Combine: Chaining

Create a column called ab_be (Y or N) that indicates whether that observation's mpg is greater (or not) than the average mpg for the filtered set.

```
mtcars %>% filter(wt > 3.3) %>%
             mutate(ab_be=ifelse(mpg > mean(mpg),"Y","N")  )
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb ab_be
1  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2     Y
2  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1     Y
3  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4     N
4  19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4     Y
5  17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4     Y
6  16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3     Y
7  17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3     Y
8  15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3     N
9  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4     N
10 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4     N
11 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4     N
12 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2     N
13 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2     N
14 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4     N
15 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2     Y
16 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8     N
```

Split -> Apply -> Combine: Chaining

Then present the average mpg for each group as defined by ab_be

```
mtcars %>% filter(wt > 3.3)  %>%
         mutate(ab_be=ifelse(mpg > mean(mpg),"Y","N")  ) %>%
         group_by(ab_be) %>% summarize(mean_mpg=mean(mpg))
```

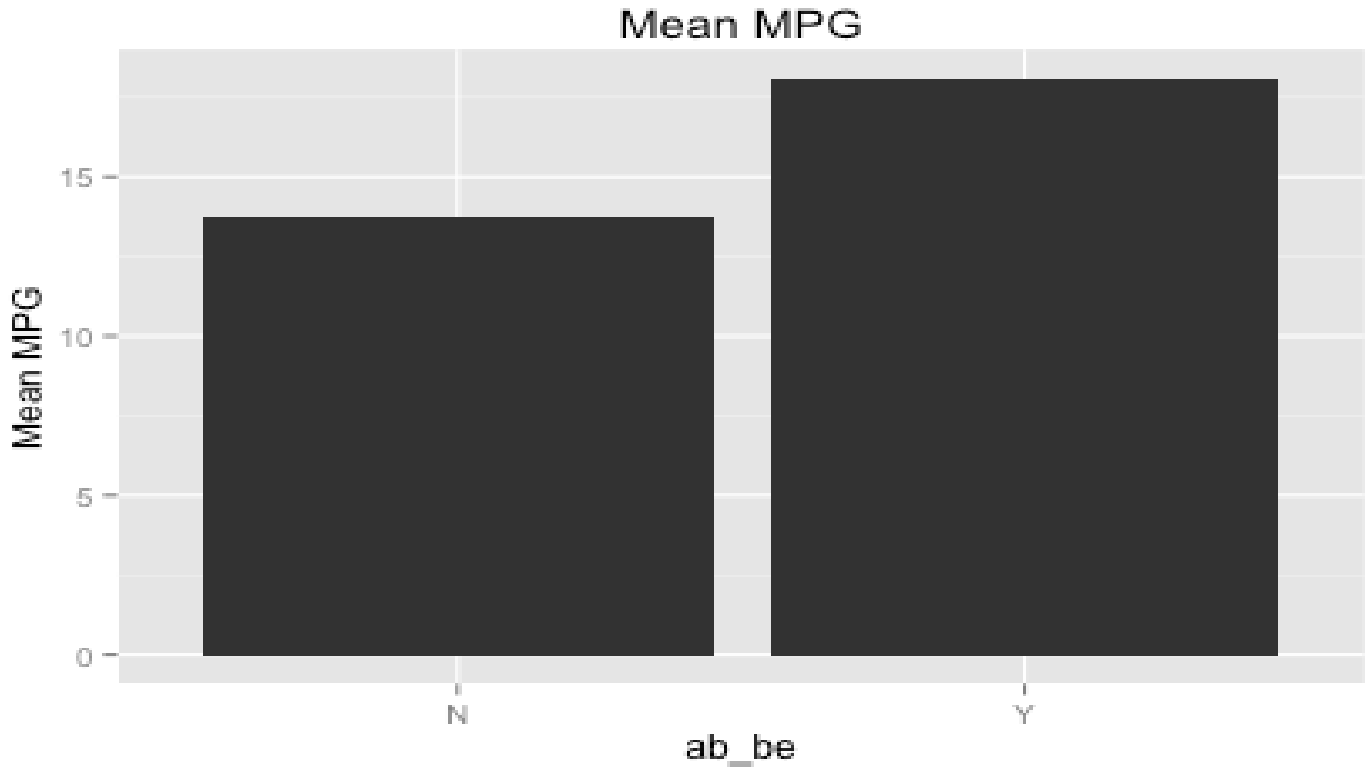
```
Source: local data frame [2 x 2]

  ab_be mean_mpg
1     N 13.77778
2     Y 18.10000
```

Split -> Apply -> Combine: Chaining

This could then be chained to the ggplot command

```
mtcars %>% filter(wt > 3.3)  %>%
          mutate(ab_be=ifelse(mpg > mean(mpg),"Y","N")  ) %>%
          group_by(ab_be) %>% summarize(mean_mpg=mean(mpg)) %>%
          ggplot(aes(x=ab_be,y=mean_mpg)) + geom_bar(stat="identity") +
          ggtitle("Mean MPG") + labs(x = "ab_be", y = "Mean MPG")
```



Split -> Apply -> Combine: Chaining

How do the dplyr commands work on a really large data file ? Here we read in a 31 million row file using **data.table** which is a package designed to handle large files

```
library(data.table)

system.time(dt <- fread("combined_wiki.txt"))

Read 31164567 rows and 4 (of 4) columns from 1.307 GB file in 00:02:58
   user  system elapsed
177.457   6.492 281.963

dt[,.N]             # 31,164,567 rows !
[1] 31164567

nrow(dt)
[1] 31164567
```

Split -> Apply -> Combine: Chaining

The first column (V1) is called a "project code" for the media wiki page of interest. The V4 column represents the number of bytes downloaded for that page.

Using dplyr commands, summarize the mean number of bytes (in megabytes) per unique project page and sort the resulting table in descencing order by the average in megabytes.

```
nrow(dt)
[1] 31164567
```

```
head(dt,5)
    V1                                 V2 V3   V4
1 aa.b                        Main_Page  1 5565
2 aa.b              MediaWiki:Image_sample  1 5179
3 aa.b       MediaWiki:Upload_source_file  1 5195
4 aa.b           Wikibooks:Privacy_policy  1 4925
5 aa.d MediaWiki:Group-abusefilter-member  1 4912
```

Split -> Apply -> Combine: Chaining

Using dplyr commands, summarize the mean number of bytes (in megabytes) per unique project page and sort the resulting table in descencing order by the average in megabytes.

```
dt %>% mutate(MB=V4/1000000) %>%
       group_by(V1)%>%
       summarize(avg=round(mean(MB),2))
       %>% arrange(desc(avg))
Source: local data table [1,266 x 2]   # Note we have 1,266 rows

      V1      avg
1  en.mw 77518.22
2  ja.mw  9126.98
3  fr.mw  2020.45
4  ru.mw  1311.16
5  de.mw  1214.59
6  es.mw  1187.93
7  it.mw   472.27
8  zh.mw   374.91
9  ko.mw   234.63
10 pt.mw   207.78
```

Split -> Apply -> Combine: Chaining

Using dplyr commands, summarize the mean number of bytes (in megabytes) per unique project page and sort the resulting table in descencing order by the average in megabytes.

```
system.time( dt %>% mutate(MB=V4/1000000) %>%
       group_by(V1)%>%
       summarize(avg=round(mean(MB),2))
       %>% arrange(desc(avg)) )
```

```
user  system elapsed
  2.776   3.100   7.655
```

dply vs Native R Commands

How long with this take using the standard native R commands ? First we create a function so we can easily time things.

```
myaggre <- function(dt) {
  dt$V4 <- round(dt$V4/1000000,2)
  hold <- aggregate(V4~V1,dt,mean)
  hold <- hold[order(-hold$V4),]
  return(hold)
}

system.time( myaggre(dt))
   user  system elapsed
351.826  11.120 378.115
```

dply additional commands

Other activities are possible

```
mtcars %>% sample_n(2) # Sample 2 records from a data frame

                  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4 Wag    21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Merc 280         19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4

# Sample 2 records from each cylinder group

mtcars %>% group_by(cyl) %>% do(sample_n(.,2))
Source: local data frame [6 x 11]
Groups: cyl

   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
2 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
3 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
4 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
5 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
6 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
```

dply additional commands

Other activities are possible

```
by_cyl <- group_by(mtcars, cyl)
models <- by_cyl %>% do(mod = lm(mpg ~ disp, data = .))

Source: local data frame [3 x 2]
Groups: <by row>

  cyl     mod
1   4 <S3:lm>
2   6 <S3:lm>
3   8 <S3:lm>

summarise(models, rsq = summary(mod)$r.squared)
Source: local data frame [3 x 1]

         rsq
1 0.64840514
2 0.01062604
3 0.27015777

# Here is a one liner that does the above

mtcars %>% group_by(cyl) %>% do(mod = lm(mpg ~ disp, data = .))
       %>% summarize(rsq = summary(mod)$r.squared)
```

Joining data frames

```
idatime <- data.frame(id=rep(1:3,each=2),time=rep(0:1,each=3))
```

```
  id time
1  1    0
2  1    0
3  2    0
4  2    1
5  3    1
6  3    1
```

```
idawt <- data.frame(id=c(1,2,4),wt=c(110,130,115))
```

```
  id  wt
1  1 110
2  2 130
3  4 115
```

Inner joins - inner_join(x,y)

will return all rows from x where there are matching values in y, and all columns from x and y

idatime

id	time
1	0
1	0
2	0
2	1
3	1
3	1

idawt

id	wt
1	110
2	130
4	115

inner_join(idatime, idawt)

id	time	wt
1	0	110
1	0	110
2	0	130
2	1	130

inner_join(idawt, idatime)

id	wt	time
1	110	0
1	110	0
2	130	0
2	130	1

Joining data frames - left_join(x,y)

return all rows from x, and all columns from x and y

idatime

id	time
1	0
1	0
2	0
2	1
3	1
3	1

idawt

id	wt
1	110
2	130
4	115

left_join(idatime, idawt)

id	time	wt
1	0	110
1	0	110
2	0	130
2	1	130
3	1	NA
3	1	NA

left_join(idawt, idatime)

id	wt	time
1	110	0
1	110	0
2	130	0
2	130	1
4	115	NA

Joining data frames - anti_join(x,y)

returns all rows from x where there are not any matching values in y, keeping just the columns from x

idatime

id	time
1	0
1	0
2	0
2	1
3	1
3	1

idawt

id	wt
1	110
2	130
4	115

anti_join(idatime, idawt)

id	time
3	1
3	1

anti_join(idawt, idatime)

id	wt
4	115

Joining data frames

idatime

id	time
1	0
1	0
2	0
2	1
3	1
3	1

idawt

id	wt
1	110
2	130
4	115

semi_join(idatime, idawt)

id	time
1	0
1	0
2	0
2	1

semi_join(idawt, idatime)

id	wt
1	110
2	130

Joining data frames

return all rows from x where there are matching values in y, keeping just columns from x

idatime

id	time
1	0
1	0
2	0
2	1
3	1
3	1

idawt

id	wt
1	110
2	130
4	115

semi_join(idatime, idawt)

id	time
1	0
1	0
2	0
2	1

semi_join(idawt, idatime)

id	wt
1	110
2	130