Summarizing and Manipulating Data

PSS SUMMER SCHOOL

Anton Badev, Daniel Nikolic, Justin Skillman

June 26, 2017

Outline

- Subsetting
- Summarizing
- `dplyr` verbs
- Merging
- Reshaping
- Regressions

Subsetting review

- Subset by index: `df[row_index, col_index]`
- Subset by name: `df["rowname", "colname"]`
- Subset by logic: `df[boolean_rows, boolean_cols]`
- *Example:*
  - `df[1:10, ]`
  - `df[ , c(T,F,F,T)]`
  - `df[ , c("var1,"var2")]`

Summarizing and grouping

- Calculate summary statistics for data by different subgroups

```
>group_by([dataframe], [column name(s)])
>
>summarise([table],
          [new column]=[function]([column name]),
          ...)
```

- First we use the group_by function to designate the variables by which we want to aggregate (combine unique values)
- Then we specify what functions to run over which columns as well as the column name of the new variable formed from the result

Salaries

- Consider the following dataset *banks*: Loan amounts for a select group of American banks

bankID	type	district	loanID	amount
1347	Commercial	Dallas	J1	22.05
2499	Savings	Chicago	J2	400.00
1612	Commercial	Atlanta	G1	11.10
2270	Commercial	Atlanta	G1	15.00
1288	Credit Union	Dallas	J1	35.00
1863	Credit Union	Chicago	G2	0.50
2952	Commercial	Atlanta	J2	10.00
1130	Savings	Dallas	J1	28.50

Summarizing and grouping with dplyr: example

- Goal: to find the mean and sum of commercial bank loans in each district

```
>df <- banks[banks$type == ‘Commercial’, ]
>tmp <- group_by(df, district)
>summarise(tmp,
          count=n(),
          tloan=sum(amount),
          mloan=mean(amount))
```

- Result:

district	count	tloan	mloan
Atlanta	3	36.10	12.03
Dallas	1	22.05	22.05

Do() function in dplyr

- do() is a generic function that works well with group_by()
- Can be used to call and apply any function to each group of a dataset, not just data reductions like with summarise()

```
>by_bank <- group_by(banks, bankID)
>
>filler <- data.frame(matrix(ncol = ncol(banks), nrow = 2))
>names(filler) <- names(banks)
>
>rep_banks <- do(by_bank, bind_rows(., filler))
>head(rep_banks)
  bankID          type district loanID amount
   <dbl>         <chr>    <chr>  <chr>  <dbl>
1   1130       Savings   Dallas     J1   28.5
2     NA          <NA>     <NA>   <NA>     NA
3     NA          <NA>     <NA>   <NA>     NA
4   1288 Credit Union   Dallas     J1   35.0
5     NA          <NA>     <NA>   <NA>     NA
6     NA          <NA>     <NA>   <NA>     NA
```

More dplyr verbs

Base R	dplyr Equivalent*
`df[`*rows*`, ]`	`filter(...)`
`df[ , `*cols*`]`	`select(...)`
`df$new.var`	`mutate(...)`

- The pipe operator in `dplyr` is %>%
- *Example:*
  - `x %>% f(y)` is equivalent to `f(x,y)`

*Does **not** always behave the same as base R

Merge function

- Related data can often be combined to help you address your questions in meaningful ways
- Combine two data frames by matching a set of common identifiers that link them

  ```
  merge(x, y, ...)
  ```

Merge function: example 1

- **loans**

loanID	length	intrate
J1	1	.0005
J2	5	.075
J3	10	.09
G1	30	.15
G2	40	.17

Merge function: example 1

- What kind of join is this?

```
>merge(banks, loans)
```

```
  loanID bankID         type district amount length intrate
1     G1   1612   Commercial  Atlanta  11.10     30  0.1500
2     G1   2270   Commercial  Atlanta  15.00     30  0.1500
3     G2   1863 Credit Union  Chicago   0.50     40  0.1700
4     J1   1347   Commercial   Dallas  22.05      1  0.0005
5     J1   1288 Credit Union   Dallas  35.00      1  0.0005
6     J1   1130      Savings   Dallas  28.50      1  0.0005
7     J2   2499      Savings  Chicago 400.00      5  0.0750
8     J2   2952   Commercial  Atlanta  10.00      5  0.0750
```

Merge function: example 2

```
>merge(banks, loans, all.y = TRUE)
```

```
  loanID bankID         type district amount length intrate
1     G1   1612   Commercial  Atlanta  11.10     30  0.1500
2     G1   2270   Commercial  Atlanta  15.00     30  0.1500
3     G2   1863 Credit Union  Chicago   0.50     40  0.1700
4     J1   1347   Commercial   Dallas  22.05      1  0.0005
5     J1   1288 Credit Union   Dallas  35.00      1  0.0005
6     J1   1130      Savings   Dallas  28.50      1  0.0005
7     J2   2499      Savings  Chicago 400.00      5  0.0750
8     J2   2952   Commercial  Atlanta  10.00      5  0.0750
9     J3     NA         <NA>     <NA>     NA     10  0.0900
```

In-class exercise

- Using **banks** and **loans** complete the following tasks:
    - Calculate aggregate revenue on loans for banks within each of the districts
    - Calculate proportions of the district revenue held by each bank
    - Return the largest contributor to revenue in each district

Reshaping data

Data may be represented in wide form (left) or long form (right).

stocks_w

Stock	2007	2008	2009
AAPL	400	450	500
ADBE	30	10	40
AMZN	200	150	200

stocks_l

Stock	Year	Price
AAPL	2007	400
AAPL	2008	450
AAPL	2009	500
AMZN	2007	200
AMZN	2008	150
AMZN	2009	200
ADBE	2007	30
ADBE	2008	10
ADBE	2009	40

Reshaping data: example 1

```
>library(reshape2)
> dcast(stocks_l, Stock ~ Year)

  stock 2007 2008 2009
1  ADBE   30   10   40
2  AMZN  200  150  200
3  APPL  400  450  500
```

Reshaping data: example 2

```
>library(reshape2)
> melt(stock_w, id.vars=c(‘stock’),
       variable.name=’year’, value.name=‘Price’)
```

```
  stock year Price
1  ADBE 2007    30
2  AMZN 2007   200
3  APPL 2007   400
4  ADBE 2008    10
5  AMZN 2008   150
6  APPL 2008   450
7  ADBE 2009    40
8  AMZN 2009   200
9  APPL 2009   500
```

Regression analysis

- Data manipulation techniques are necessary to get to the end goal: *data analysis*
- Regression analysis:
  - Purpose: find (potentially) meaningful relationships in your data
  - Estimation of the relationship between 2 or more variables
  - Potential pitfalls: omitted variables, reverse causality, mismeasurement

Linear models

- A linear model is a useful first pass at understanding the relationships between variables in your data
- They are straightforward to run using the `lm()` function from the base package
  - $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \epsilon$

```
>formula <- y ~ x1 + x2 + x3
>est <- lm(data,formula)
>summary(est)
```

Linear models: example

```
data(ToothGrowth)

> est <- lm(data=ToothGrowth, len ˜ supp + dose)
> summary(est)
```

```
Call:
lm(formula = len ˜ supp + dose, data = ToothGrowth)

Residuals:
   Min     1Q Median     3Q    Max 
-6.600 -3.700  0.373  2.116  8.800 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   9.2725     1.2824   7.231 1.31e-09 ***
suppVC       -3.7000     1.0936  -3.383   0.0013 ** 
dose          9.7636     0.8768  11.135 6.31e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.236 on 57 degrees of freedom
Multiple R-squared:  0.7038,Adjusted R-squared:  0.6934 
F-statistic: 67.72 on 2 and 57 DF,  p-value: 8.716e-16
```