Data Wrangling - 2

COMM 205 - Lecture 20 - R5

Hasan Cavusoglu

2020

Agenda

- `filter()` with NA's
- `group_by()` function
  - `group_by()` with `mutate()`
  - `group_by()` with `summarise()`
- counting observations with `n()`

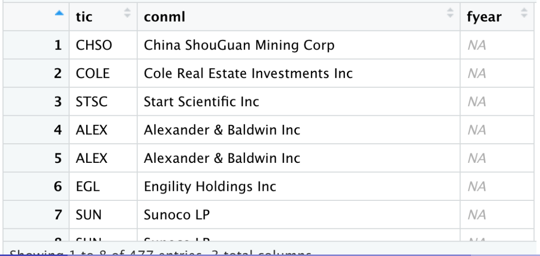
filter() with NA's

- filter() will only keep those observations resulting in TRUE in filtering vector passed onto the filter() function.

Question

Suppose you are asked to list the **ticker** (i.e., tic), **company name** (i.e., conml) of companies for the observation **fyear** is missing.

```
library(tidyverse)
companies <- readRDS("data/North_American_Stock_Market_1994-2013.rds")
missing_fyear <- companies %>%
  filter(is.na(fyear)) %>% select(tic, conml, fyear)
```



	tic	conml	fyear
1	CHSO	China ShouGuan Mining Corp	NA
2	COLE	Cole Real Estate Investments Inc	NA
3	STSC	Start Scientific Inc	NA
4	ALEX	Alexander & Baldwin Inc	NA
5	ALEX	Alexander & Baldwin Inc	NA
6	EGL	Engility Holdings Inc	NA
7	SUN	Sunoco LP	NA
8	SUN	Sunoco LP	NA

Showing 1 to 8 of 477 entries, 3 total columns

`filter()` with NA's (cont'd)

- if the variable of interest in a *logical condition* in `filter()` has NA's in some observations, those observations are **excluded** from the outcome.
  - `filter()` will only keep those observations resulting in `TRUE` in filtering vector passed onto the `filter()` function. But, NAs would result in NAs in that vector.

Question

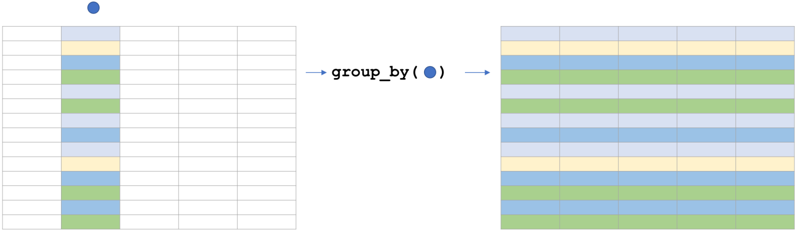
Suppose you are only interested in observations where **assets**, **sales** and **number of employees** are **nonmissing** and **non-negative** values.

```
filtered <- companies %>%
  filter(at>0, sale>0, emp>0)
```

Note that `filter(at>0, sale>0, emp>0)` excludes observations whose `at`, `sale`, or `emp` value is *either* **NA** *or* **negative**. Therefore, you do not need to specify `filter(at>0, sale>0, emp>0,!is.na(at), !is.na(sale), !is.na(emp))`.

`group_by()` Function

- `group_by` creates groups of observations based on one or more variables.
- Many real life situations require data analyses to be performed on basis of groups.
- `group_by()` takes an existing data frame and converts it into a **grouped** dataframe where subsequent operations will be performed **group by group**.



- Observations (i.e., rows) in the same group are ***not* collated** (collected) together. The original locations of the observations do ***not*** change.

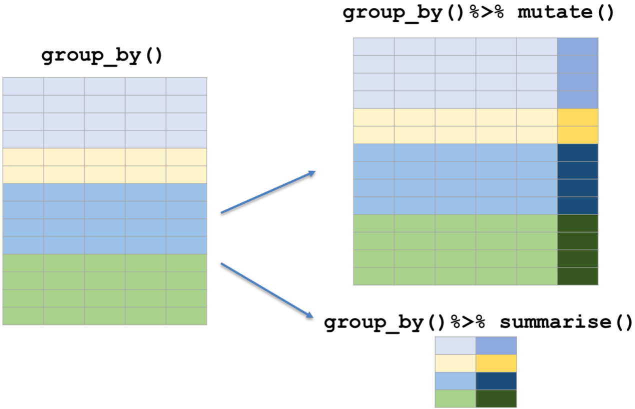
Two ways of using `group_by()`

- with `mutate()`
- with `summarise()`

Rule of Thumb

- If you want to create a new column whose elements are calculated at the group level, you need to use first `group_by()` and then `mutate()`.
- If you want to create a summary table in which each group is represented in a single row, you need to use first `group_by()` and then `summarise()`.

Two ways of using group_by() (cont'd)



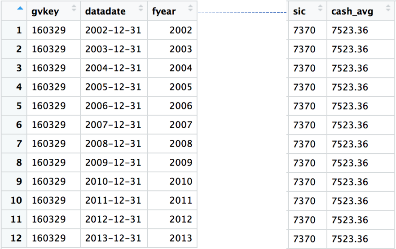
Using group_by() with mutate()

Question

Suppose, for example, you want to calculate the **mean** (average) amount of **cash** (*ch*) held by each firm over the years and record the firm's average cash in a new column for every observations of the firm.

Let's do this for only one firm; let's say, Google (`tic=="GOOG"`).

```r
q1_google <- companies %>% filter(tic == "GOOG") %>%
  mutate(cash_avg = mean(ch))
```



	gvkey	datadate	fyear	sic	cash_avg
1	160329	2002-12-31	2002	7370	7523.36
2	160329	2003-12-31	2003	7370	7523.36
3	160329	2004-12-31	2004	7370	7523.36
4	160329	2005-12-31	2005	7370	7523.36
5	160329	2006-12-31	2006	7370	7523.36
6	160329	2007-12-31	2007	7370	7523.36
7	160329	2008-12-31	2008	7370	7523.36
8	160329	2009-12-31	2009	7370	7523.36
9	160329	2010-12-31	2010	7370	7523.36
10	160329	2011-12-31	2011	7370	7523.36
11	160329	2012-12-31	2012	7370	7523.36
12	160329	2013-12-31	2013	7370	7523.36

Using `group_by()` with `mutate()` (cont'd)

- To answer the question, we need to **repeat** what we have done for Google for **each firm** in our dataset.
- **Each firm** in our dataset will be **a group** for which average cash value will be calculated by using all the observations of the firm.
- Using `group_by()` with `mutate()`, the calculated value will be recorded under a new column, the value of which would be same for all observations of that firm.

Syntax for group_by() with mutate()

Syntax

```
data_object %>%
  group_by(group_variable(s)) %>%
  mutate(new_variable = function(existing_variable(s)))
```

where function is any function resulting in an atomic value. Generally, the function is an aggregate_function, which can be (not exhaustive):

- For the central tendency: `mean()`, `median()`
- For the spread: `sd()`
- For the range: `min()`, `max()`
- For count: `n()`, `n_distinct()`
- For aggregating: `sum()`

Using group_by() with mutate() (cont'd)

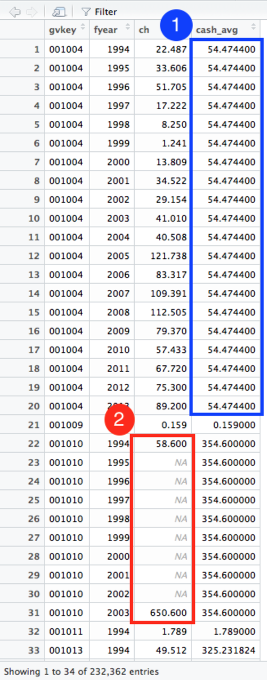
- Since each group represents a company, we need to find a column that **uniquely** identifies companies.
- In this dataset, **Compustat** assigns a number, called the **Global Company Key** (gvkey) to each firm. Each key ***uniquely*** identifies a particular company.
- As a unique identifier, gvkey can never be missing. The reason we use gvkey and not another variable such as name (conm) is that companies can sometimes change their name but still operate as the same firm. For example, Apple Inc. was called Apple Computer until January 9, 2007.

Thus, to creare a new column with the average cash value of the firm:

```
companies_with_cash_avg <- companies %>%
  group_by(gvkey) %>%
  mutate(cash_avg = mean(ch, na.rm = TRUE))
```

To verify that we have executed the command properly (compare with the figure next page), you can run the following command :

```
View(companies_with_cash_avg %>%
       select(gvkey, fyear, ch, cash_avg))
```



	gvkey	fyear	ch	cash_avg
1	001004	1994	22.487	54.474400
2	001004	1995	33.606	54.474400
3	001004	1996	51.705	54.474400
4	001004	1997	17.222	54.474400
5	001004	1998	8.250	54.474400
6	001004	1999	1.241	54.474400
7	001004	2000	13.809	54.474400
8	001004	2001	34.522	54.474400
9	001004	2002	29.154	54.474400
10	001004	2003	41.010	54.474400
11	001004	2004	40.508	54.474400
12	001004	2005	121.738	54.474400
13	001004	2006	83.317	54.474400
14	001004	2007	109.391	54.474400
15	001004	2008	112.505	54.474400
16	001004	2009	79.370	54.474400
17	001004	2010	57.433	54.474400
18	001004	2011	67.720	54.474400
19	001004	2012	75.300	54.474400
20	001004	2013	89.200	54.474400
21	001009		0.159	0.159000
22	001010	1994	58.600	354.600000
23	001010	1995	NA	354.600000
24	001010	1996	NA	354.600000
25	001010	1997	NA	354.600000
26	001010	1998	NA	354.600000
27	001010	1999	NA	354.600000
28	001010	2000	NA	354.600000
29	001010	2001	NA	354.600000
30	001010	2002	NA	354.600000
31	001010	2003	650.600	354.600000
32	001011	1994	1.789	1.789000
33	001013	1994	49.512	325.231824

Showing 1 to 34 of 232,362 entries

- First thing to reiterate is that the same value appears in under **cash_avg** column for a given company
- Notice that the company with **gvkey** value of **"001010"** has missing ch values. The **cash_avg** is calculated based on the other **two** years when data on **ch** were available

Using group_by() with summarise()

- If the objective is to perform an aggregate operation over each group and to display **one** aggregate **result *per* group**, you should use group_by() with summarise().

The syntax

```
data_object %>%
  group_by(group_variable(s)) %>%
  summarise(new_variable = function(existing_variable(s)))
```

where, again, function is any function resulting in an atomic value. Generally, the function is an aggregate_function, which can be (not exhaustive):

- For the central tendency: mean(), median()
- For the spread: sd()
- For the range: min(), max()
- For count: n(), n_distinct()
- For aggregating:sum()

Using group_by() with summarise() (cont'd)

Question

You are asked to create a new *summary* data frame where each company's **average cash** (i.e., *cash_avg*) is recorded along with its *gvkey*. That, there should be one observation per firm.

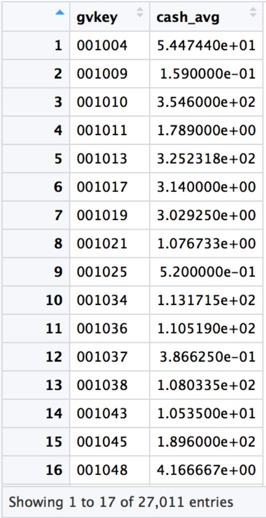
The following code will create the data frame asked.

```
gvkey_cash_avg <- companies %>%
  group_by(gvkey) %>%
  summarise(cash_avg = mean(ch, na.rm = TRUE))
```

To verify that we have executed the command properly (compare with the figure next page), you can run the following command :

```
View(gvkey_cash_avg)
```

Quesion (cont'd)



	gvkey	cash_avg
1	001004	5.447440e+01
2	001009	1.590000e-01
3	001010	3.546000e+02
4	001011	1.789000e+00
5	001013	3.252318e+02
6	001017	3.140000e+00
7	001019	3.029250e+00
8	001021	1.076733e+00
9	001025	5.200000e-01
10	001034	1.131715e+02
11	001036	1.105190e+02
12	001037	3.866250e-01
13	001038	1.080335e+02
14	001043	1.053500e+01
15	001045	1.896000e+02
16	001048	4.166667e+00

Showing 1 to 17 of 27,011 entries

- Note that **mean()** will produce a **numeric value** as long as company has at least **one nonmissing** cash value.
- **Outside the Scope of this course** When an aggregate function is used with na.rm = TRUE on a vector whose elements are all NA's, the result will *not* be a number. The result will be one of the followings: NA, NaN, Inf, and -Inf.

Counting with `n()`

Question

Suppose you want to know how many unique firms appears in **all years** in the filtered data frame (i.e., observation satisfying at$>$0, sale$>$0, and emp$>$0). That is, you are interested in the number of firms which have **20** observations in the filtered data frame.

Solution strategy

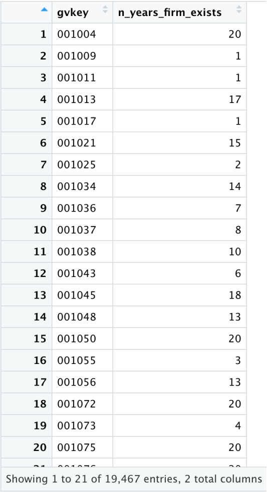
1) Find number of observations per firm (let's call it n_year_firm_exists).
2) Find number of firms with each different n_years_firm_exists values (i.e., 1 to 20)

Step 1

- Basically, you want to group observations by firm (i.e. gvkey) and
- then, in each group you want to count the number of observations:

```
filtered_summarized <- filtered %>%
  group_by(gvkey) %>%
  summarise(n_years_firm_exists = n())
```

Question (cont'd)



	gvkey	n_years_firm_exists
1	001004	20
2	001009	1
3	001011	1
4	001013	17
5	001017	1
6	001021	15
7	001025	2
8	001034	14
9	001036	7
10	001037	8
11	001038	10
12	001043	6
13	001045	18
14	001048	13
15	001050	20
16	001055	3
17	001056	13
18	001072	20
19	001073	4
20	001075	20

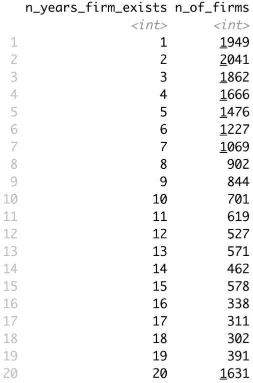
Showing 1 to 21 of 19,467 entries, 2 total columns

- As you see, for each firm (identified by a unique gvkey), we have how many years of observations we have in our dataset (i.e., n_years_firm_exists).

Step 2

```r
filtered_summarized %>%
  group_by(n_years_firm_exists) %>%
  summarise(n_of_firms=n())
```

- Since we are interested in firms which have 20 observations in our dataset (i.e., one for each year), we need to count observations for which n_years_firm_exists is 20.
- To be able to do so, we need to group_by with respect to n_years_firm_exists and then use summarise() to obtain the count (with n()).



```
   n_years_firm_exists n_of_firms
                 <int>      <int>
 1                   1       1949
 2                   2       2041
 3                   3       1862
 4                   4       1666
 5                   5       1476
 6                   6       1227
 7                   7       1069
 8                   8        902
 9                   9        844
10                  10        701
11                  11        619
12                  12        527
13                  13        571
14                  14        462
15                  15        578
16                  16        338
17                  17        311
18                  18        302
19                  19        391
20                  20       1631
```

There are **1631 firms** which appeared **20 times** in our original data frame.

Putting everything together

We can combine two steps and filter the previous data frame for n_years_firm_exists==20.

```
 companies %>%
  filter(at>0, sale>0, emp>0) %>%
  group_by(gvkey) %>%
  summarise(n_years_firm_exists = n()) %>%
  group_by(n_years_firm_exists) %>%
  summarise(n_of_firms=n()) %>%
  filter(n_years_firm_exists==20)
```

```
## # A tibble: 1 x 2
##   n_years_firm_exists n_of_firms
##                 <int>      <int>
## 1                  20       1631
```

The End

Thanks for watching

See you in next time!