Data Wrangling - 1

COMM 205 - Lecture 19 - R4

Hasan Cavusoglu

2020

Agenda

- Data Wrangling
- select()
- filter()
- mutate()
- summarise()
- pipe operator

Data Wrangling

- Wikipedia defines data wrangling as "*the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics*".
- For data wrangling in R, we will mainly use two packages, namely `dplyr` and `magrittr`, shipped with `tidyverse`.
  - `dplyr` provides a grammar for data manipulation, providing a consistent set of verbs that solve most common data manipulation challenges.
  - `magrittr` package offers a set of operators which make your code more readable.
- Since we will use functions from `tidyverse`, we can go ahead and load it.

```
library(tidyverse)
```

```
> library(tidyverse)
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
✓ ggplot2 3.3.2     ✓ purrr   0.3.4
✓ tibble  3.0.4     ✓ dplyr   1.0.2
✓ tidyr   1.1.2     ✓ stringr 1.4.0
✓ readr   1.4.0     ✓ forcats 0.5.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
>
```

Now, you can make use of all the functions available in packages included in `tidyverse`.

Main dplyr functions

We will cover the following functions in this course.

- `select()` extracts columns based on their names.
- `filter()` extracts rows based on values.
- `mutate()` adds new columns that are functions of existing columns
- `summarise()` reduces multiple observations down to a single value.
- `arrange()` changes the ordering of the rows (will be covered later)
- `group_by()` allows you to perform any of the above operations "by group" (will be covered later).

North American Stock Market 1994-2013 Dataset

- We will be frequently using a data set stored in `North_American_Stock_Market_1994-2013.rds` file. The original data was put together by Dr. Adam Saunders.
- The data set contains information on virtually all publicly traded firms in Canada and the United States from 1994 to 2013 (obtained from Compustat).
- Most of the data is from the United States, but some companies are from Canada.
- If you wish, you can also crosscheck the data in this dataset with the ones you can access from Google Finance (https://www.google.ca/finance). Type a ticker symbol and click on Financials.
- Note that the dollar amounts in this data set are **in millions** of USD (or CAD), except for per share items.

Reading/Loading an RDS file

You can read an `rds` file by using `readRDS()` function by specifying the location of the file. Note that the file location is specified within quotation marks. The output of the function is a data frame.

Syntax

readRDS("*location-of-the-file*")
where **"*location-of-the-file*"** is
- absolute path to the file or
- relative path with respect to the working directory.

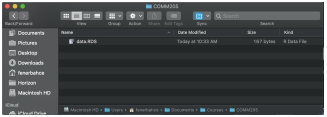
How?

1) Using GUI to load the data
2) Executing readRDS() directly by specifying the path

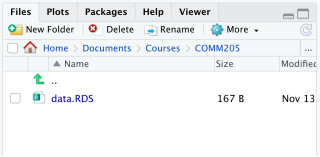
How to in Mac

- Suppose you are given an RDS file called `data.RDS`.
- Alternative #1

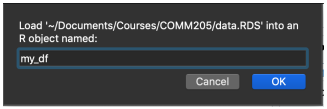
1) Save the file



2) Locate the file



3) Name the Data Frame



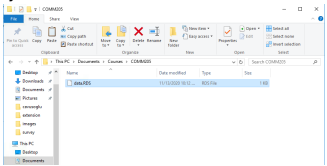
- Alternative #2

```
my_df <- readRDS("~/Documents/Courses/COMM205/data.RDS")
```

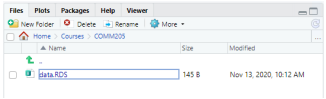
How to in Windows

- Suppose you are given an RDS file called `data.RDS`.
- Alternative #1

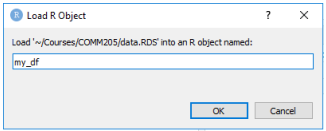
1) Save the file



2) Locate the file



3) Name the Data Frame



- Alternative #2

```
my_df <- readRDS("~/Courses/COMM205/data.RDS")
```

Loading Companies Data

- Do **not** click on the RDS file on your Finder/File Explorer.
- Either use graphical interface on RStudio to locate the file file or directly specify the path.

```
companies <- readRDS("<path_to_your_file>/North_American_Stock_Mark
```

- In my case, I saved the rds file in a folder at: `"/Users/fenerbahce/Personal Content/Horizon/TEACHING/COMM 205/R/"`.

```
companies <- readRDS("/Users/fenerbahce/Personal Content/Horizon/TE
```

North American Stock Market Dataset (1994-2013)

If you view the data frame, you will see that there are 50 variables and 232,362 observersations. Remember two ways to see a data frame stored in a session.

- by clicking on the name of the data frame on the Environment pane, or
- Typing `View(companies)` on the Console.

Filter

	gvkey	datadate	fyear	indfmt	consol	popsrc	datafmt	tic	cusip	conm	curcd	fyr	act	at	capx	ceq
1	001004	1995-05-31	1994	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	321.632	425.814	9.073	197.11
2	001004	1996-05-31	1995	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	338.012	437.846	7.547	204.63
3	001004	1997-05-31	1996	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	414.100	529.584	30.292	269.25
4	001004	1998-05-31	1997	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	468.400	670.559	17.495	300.85
5	001004	1999-05-31	1998	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	508.186	726.630	36.131	326.03
6	001004	2000-05-31	1999	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	511.267	740.998	22.344	339.51
7	001004	2001-05-31	2000	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	485.856	701.854	13.134	340.21
8	001004	2002-05-31	2001	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	436.656	710.199	12.112	310.23
9	001004	2003-05-31	2002	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	396.412	686.621	9.930	294.98
10	001004	2004-05-31	2003	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	432.204	709.292	10.286	301.68
11	001004	2005-05-31	2004	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	474.542	732.230	13.033	314.74
12	001004	2006-05-31	2005	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	624.454	978.819	16.296	422.71
13	001004	2007-05-31	2006	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	645.721	1067.633	29.891	494.24
14	001004	2008-05-31	2007	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	783.431	1362.010	30.334	585.25
15	001004	2009-05-31	2008	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	851.312	1377.511	27.535	656.89
16	001004	2010-05-31	2009	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	863.429	1501.042	28.855	746.90
17	001004	2011-05-31	2010	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	913.985	1703.727	124.879	835.84
18	001004	2012-05-31	2011	INDL	C	D	STD	AIR	000361105	AAR CORP	USD	5	1063.272	2195.653	91.218	864.64

Columns

no	variable	type	label
1	gvkey	character	Global Company Key
2	datadate	Date	Data Date
3	fyear	numeric	Data Year - Fiscal
4	indfmt	character	Industry Format
5	consol	character	Level of Consolidation - Company Annual Descriptor
6	popsrc	character	Population Source
7	datafmt	character	Data Format
8	tic	character	Ticker Symbol
9	cusip	character	CUSIP
10	conm	character	Company Name
11	curcd	character	ISO Currency Code
12	fyr	numeric	Fiscal Year-end Month
13	act	numeric	Current Assets - Total
14	at	numeric	Assets - Total
15	capx	numeric	Capital Expenditures
16	ceq	numeric	Common/Ordinary Equity - Total
17	ch	numeric	Cash
18	che	numeric	Cash and Short-Term Investments
19	cogs	numeric	Cost of Goods Sold
20	csho	numeric	Common Shares Outstanding

Columns (cont'd)

no	variable	type	label
21	dlc	numeric	Debt in Current Liabilities - Total
22	dltt	numeric	Long-Term Debt - Total
23	ebit	numeric	Earnings Before Interest and Taxes
24	ebitda	numeric	Earnings Before Interest
25	emp	numeric	Employees
26	invt	numeric	Inventories - Total
27	lct	numeric	Current Liabilities - Total
28	lt	numeric	Liabilities - Total
29	ni	numeric	Net Income (Loss)
30	niadj	numeric	Net Income Adjusted for Common/Ordinary Stock (Capital) Equivalents
31	oiadp	numeric	Operating Income After Depreciation
32	oibdp	numeric	Operating Income Before Depreciation
33	ppent	numeric	Property, Plant and Equipment - Total (Net)
34	pstk	numeric	Preferred/Preference Stock (Capital) - Total
35	rect	numeric	Receivables - Total
36	sale	numeric	Sales/Turnover (Net)
37	seq	numeric	Stockholders Equity - Parent
38	exchg	numeric	Stock Exchange Code
39	cik	character	CIK Number
40	costat	character	Active/Inactive Status Marker

Columns (cont'd)

no	variable	type	label
41	fic	character	Current ISO Country Code - Incorporation
42	naicsh	numeric	North America Industrial Classification System - Historical
43	sich	numeric	Standard Industrial Classification - Historical
44	prcc_c	numeric	Price Close - Annual - Calendar
45	prcc_f	numeric	Price Close - Annual - Fiscal
46	conml	character	Company Legal Name
47	ein	character	Employer Identification Number
48	loc	character	Current ISO Country Code - Headquarters
49	naics	character	North American Industry Classification Code
50	sic	character	Standard Industry Classification Code

select() Function

`select()` extracts columns specified from a data frame. The output is a data frame.

select() syntax:

select(data_object_name, variable_name(s))

Question

Let's use our companies and create a new data frame which has the **company name** (i.e., *conm*), its **headquarter location** (i.e., *loc*), its **ticker** (i.e., *tic*), the **fiscal year** (i.e., *fyear*), and its **stock price at the end of the calendar year** (i.e., *prcc_c*).

```
stock_price_year_end <-
  select(companies,  conm, loc,
         tic, fyear, prcc_c)
```

Note that the new data frame will have the same number of observations (i.e., 232,362) and *only* five columns specified.

New data frame

Here are the first 30 observations from `stock_price_year_end`:

	conm	loc	tic	fyear	prcc_c
1	AAR CORP	USA	AIR	1994	13.374999
2	AAR CORP	USA	AIR	1995	21.999998
3	AAR CORP	USA	AIR	1996	30.249985
4	AAR CORP	USA	AIR	1997	38.749992
5	AAR CORP	USA	AIR	1998	23.875000
6	AAR CORP	USA	AIR	1999	17.937500
7	AAR CORP	USA	AIR	2000	12.625000
8	AAR CORP	USA	AIR	2001	9.010000
9	AAR CORP	USA	AIR	2002	5.150000
10	AAR CORP	USA	AIR	2003	14.950000
11	AAR CORP	USA	AIR	2004	13.620000
12	AAR CORP	USA	AIR	2005	23.950000
13	AAR CORP	USA	AIR	2006	29.190000
14	AAR CORP	USA	AIR	2007	38.030000
15	AAR CORP	USA	AIR	2008	18.410000
16	AAR CORP	USA	AIR	2009	22.980000
17	AAR CORP	USA	AIR	2010	27.470000
18	AAR CORP	USA	AIR	2011	19.170000
19	AAR CORP	USA	AIR	2012	18.680000
20	AAR CORP	USA	AIR	2013	28.010000
21	ABS INDUSTRIES INC	USA	ABSI	1994	11.750000
22	ACF INDUSTRIES HOLDING CORP	USA	4165A	1994	*NA*
23	ACF INDUSTRIES HOLDING CORP	USA	4165A	1995	*NA*
24	ACF INDUSTRIES HOLDING CORP	USA	4165A	1996	*NA*
25	ACF INDUSTRIES HOLDING CORP	USA	4165A	1997	*NA*
26	ACF INDUSTRIES HOLDING CORP	USA	4165A	1998	*NA*
27	ACF INDUSTRIES HOLDING CORP	USA	4165A	1999	*NA*
28	ACF INDUSTRIES HOLDING CORP	USA	4165A	2000	*NA*
29	ACF INDUSTRIES HOLDING CORP	USA	4165A	2001	*NA*
30	ACF INDUSTRIES HOLDING CORP	USA	4165A	2002	*NA*

Obviously, you can use `View(stock_price_year_end)` to browse the the newly created data frame.

`filter()` Function

`filter()` keeps rows/observations where condition(s) specified are satisfied.

filter() syntax:

filter(data_object_name, condition(s))
where conditions are logical operations on the column(s) of the data object.

- Multiple conditions are combined with `&` or `|`.
- Only rows where the condition evaluates to `TRUE` are kept.
- Note that rows where the condition(s) evaluate to `NA` are **dropped**.
- If you separate the conditions with commas, it combines with conditions with `&` (i.e., all the conditions must be satisfied).

filter() example

Question

Again using our North American Stock Market 1994-2013 Dataset, let's create a new data frame which containing observersations for which **headquarter location** (i.e., *loc*) is "CAN".

```
canadian_companies <- filter(companies, loc=="CAN")
```

You can browse the canadian_companies data frame by typing the following at your console.

```
View(canadian_companies)
```

As you can see, filter retains all the columns of the original data only keep the observations satisfied the condition.

select() and filter() combined

If you are interested in specific columns for these companies.

Question

Create a new data frame which has the **company name** (i.e., *conm*), its **headquarter location** (i.e., *loc*), its **ticker** (i.e., *tic*), the **fiscal year** (i.e., *fyear*), and its **stock price at the end of the calendar year** (i.e., *prcc_c*) for **companies whose headquarter is in Canada**.

This can be done with nesting:

```
can_companies_stock_v1 <-
  select(filter(companies,loc=="CAN"),
         conm, loc, tic, fyear, prcc_c)
```

This is equivalent to creating an intermediate dataframe and passing that to the next command as follows:

```
canadian_companies <- filter(companies, loc=="CAN")
canadian_stock_stock_v1 <-
  select(canadian_companies,
         conm, loc, tic, fyear, prcc_c)
```

NA

Remember that *"rows where the condition evaluates to NA are dropped".*

If you are only interested in 'conm', 'fyear', and 'ch' for the company with 'gvkey' of '001010', you will see that 'ch' values for some years are 'NA'.

```
select(filter(companies, gvkey=="001010"),
       gvkey, conm, fyear, ch)
```

gvkey	conm	fyear	ch
001010	ACF INDUSTRIES HOLDING CORP	1994	58.6
001010	ACF INDUSTRIES HOLDING CORP	1995	NA
001010	ACF INDUSTRIES HOLDING CORP	1996	NA
001010	ACF INDUSTRIES HOLDING CORP	1997	NA
001010	ACF INDUSTRIES HOLDING CORP	1998	NA
001010	ACF INDUSTRIES HOLDING CORP	1999	NA
001010	ACF INDUSTRIES HOLDING CORP	2000	NA
001010	ACF INDUSTRIES HOLDING CORP	2001	NA
001010	ACF INDUSTRIES HOLDING CORP	2002	NA
001010	ACF INDUSTRIES HOLDING CORP	2003	650.6

Data Wrangling with NA

Now, if you want to `filter` of the observations of that company of which `ch` is positive, you can add another condition to the `filter` as follows.

```
select(filter(companies, gvkey=="001010", ch>0),
       gvkey, conm, fyear, ch)
```

gvkey	conm	fyear	ch
001010	ACF INDUSTRIES HOLDING CORP	1994	58.6
001010	ACF INDUSTRIES HOLDING CORP	2003	650.6

If you compare the results of the last two codes, you will see that the rows (observations) where the condition evaluates to `NA` are dropped in the latter code.

Because seperating the conditions with commas is equivalent to combining the conditions with logical `AND` operators, the code above is identical to

```
select(filter(companies, gvkey=="001010" & ch>0),
       gvkey, conm, fyear, ch)
```

Advantage(s) of `magrittr`

Most data manipulations would require multiple operations. Combining those functions either through nesting or through intermediate data frames would be cumbersome. `magrittr` comes to the rescue.

The **magrittr** package is created to make the data manipulation code more readable:

- structuring sequences of data operations **left-to-right** (not inside-out),
- avoiding nested function calls,
- minimizing the need for intermediate objects, and
- making it easy to add steps anywhere in the sequence of operations.

Sequencing the operations with pipe operator

Forward Pipe Operator

%>% pipes its **left-hand side** value forward into the expression that appears on its **right-hand side**.

Basic Piping

- `x %>% f` is equivalent to f(x).
- `x %>% f(y)` is equivalent to `f(x, y)`
- `x %>% f %>% g %>% h` is equivalent to `h(g(f(x)))`

```
filter(companies, loc=="CAN") # is equivalent to
companies %>% filter(loc =="CAN")
```

Piping Approach to a previous question

Question

Create a new data frame which has the **company name** (i.e., *conm*), its **headquarter location** (i.e., *loc*), its **ticker** (i.e., *tic*), the **fiscal year** (i.e., *fyear*), and its **stock price at the end of the calendar year** (i.e., *prcc_c*) for **companies whose headquarter is in Canada**.

```
canadian_stock_stock_v2 <- companies %>%
  filter(loc=="CAN") %>%
  select(conm, loc, tic, fyear, prcc_c)
```

As you can see, when pipe operator is used, the dplyr functions take the data from the previous step as their input. Pictorially, here is what we have done.

Readability

You can imagine that sequencing several function calls with the pipe-operator would be even more beneficial.

Note that, as we have done in our solution, data wranglers often start a new line by hitting enter after each pipe-operator to improve *readability* of the code. The code creating `canadian_companies_stocks` data frame could have been written in a single line.

```
canadian_stock_stock_v2 <- companies %>%
  filter(loc=="CAN") %>%
  select(conm, loc, tic, fyear, prcc_c)
```

```
canadian_stock_stock_v2 <- companies %>% filter(loc=="CAN") %>% select(conm, lo
```

mutate() function

`mutate()` adds new column(s) into a data frame and preserves the existing ones.

mutate() syntax:

mutate(data_object_name, *new_variable_name = ...*) where ..... can be function(s) and arithmetic/logical operations.

With `mutate()`, you can create multiple new columns by separating them with comma.

Question

Let's create a data frame where contains **company name** (i.e., *conm*), its **ticker** (i.e., *tic*), its **cash** (i.e., *ch*), not in millions as recorded in the original data frame but as a full amount (e.g., if **1** is recorded, it should be **1000000** in the new data set.) for those companies whose **headquarter** was in **Canada** (i.e., *CAN*) and whose **cash** was greater than 10 million in **financial year** 2010 (i.e., *fyear == 2010*).

```
canadian_comp_full_cash <- companies %>%
  mutate(full_ch= 1000000*ch) %>%
  filter(loc=="CAN" & fyear==2010 & ch>10) %>%
  select(conm, tic, full_ch)
```

Resultant data frame

	conm	tic	full_ch
1	MORGUARD CORP	MRCBF	27535000
2	AGNICO EAGLE MINES LTD	AEM	95560000
3	ALGOMA CENTRAL CORP	AGMJF	45537000
4	ARBOR MEMORIAL SERVICES INC	AROBF	12827000
5	ATCO LTD -CL I	ACLLF	647700000
6	BARRICK GOLD CORP	ABX	3968000000
7	BCE INC	BCE	774000000
8	RESOLUTE FOREST PRODUCTS INC	RFP	319000000
9	BROOKFIELD ASSET MANAGEMENT	BAM	1713000000
10	CATALYST PAPER CORP	CYSTF	95400000
11	TELUS CORP	TU	17000000
12	CAE INC	CAE	276400000
13	CANADIAN NATIONAL RAILWAY CO	CNI	490000000
14	NEXEN INC	NXY	1005000000
15	CANADIAN PACIFIC RAILWAY LTD	CP	360600000
16	CANADIAN TIRE CORP -CL A	CDNAF	554300000
17	CANADA BREAD CO LTD	CBDLF	84401000
18	RUSSEL METALS INC	RUSMF	323700000
19	FINNING INTERNATIONAL INC	FINGF	349857000
20	MAGELLAN AEROSPACE CORP	MALIF	24952000

summarise() Function

summarise() is used to create an aggregate statistic over the observations.

summarise() syntax:

summarise(data_object, new_var_name = aggregate_funct(existing_var_name))

Aggregate functions that can be used (*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()

When summarise() is used with group_by, the aggregate statistic is calculated over each group (we will be covered this next lecture).

Question

Question

Let's now find out the *maximum* and *minimum* **cash** recorded for **Canada-headquartered companies** in **2010**.

```
max_min_cash_Canadian_2010 <- companies %>%
  filter(loc=="CAN" & fyear==2010) %>%
  summarise(max_cash = max(ch, na.rm = TRUE),
            min_cash = min(ch, na.rm = TRUE))
```

Since `ch` has NA's, we needed to specify `max()` and `min()` to ignore `NA`s with `na.rm = TRUE`.

The resultant data frame has **two (new) columns** and **one row**.

	max_cash	min_cash
1	17368	0

The End

Thanks for watching

See you in next time!

© 2020 Hasan Cavusoglu - UBC

This content is protected and may not be shared, uploaded, or distributed.