FE8828 Programming Web Applications in Finance

Week 3: 8. dplyr/2: More Verbs and EDA

Dr. Yang Ye yy@runchee.com

Nanyang Business School

Oct 1, 2020

1. Lecture 8: dplyr/2: More Verbs and EDA

Section 1

Lecture 8: dplyr/2: More Verbs and EDA

From tree (solo df) to forest (multi-df)

We have been dealing with one data frame. Let's move onto multiple data frames with `join`.

Joins

If we use arithmetic operators to represent different `join`.

- `full_join` is *
- `anti_join` is -
- `inner_joins` is - to reduce to shared common rows and + (columns)
- `left_join/right_join` is + (columns).

full_join and anti_join

- full_join(a, b): Find all combinations between table a and b. i.e. `a * b`
- anti_join(a, b): Find those in a but not in b.

```r
# From something simple
df <- full_join(tibble(a = 1:2), tibble(a = 7:8), by = "a")
```

a
1
2
7
8

```r
df <- anti_join(tibble(a = 1:2), tibble(a = 2:6), by = "a")
```

a
1

full_join and anti_join More

```
# All possible combination between job and education
x <- full_join(distinct(bank, job) %>% mutate(dummy = 1),
               distinct(bank, education) %>% mutate(dummy = 1),
               by = "dummy") %>%
     select(-dummy)
# actual combination of job and education in bank dataset
y <- distinct(bank, job, education)

nrow(x)
## [1] 48
nrow(y)
## [1] 48

df1 <- anti_join(x, y, by = c("job", "education"))
df2 <- anti_join(y, x, by = c("job", "education"))
cat(paste0("nrow(df1):", nrow(df1)))
## nrow(df1):0
cat(paste0("nrow(df2):", nrow(df2)))
## nrow(df2):0
```

We can conclude that, in the bank dataset, there are all combinations for job and education.

left/right/anti/full_join

Sample data:

- data_day1

Date	Position_id	Buy/Sell	Quantity	Risk Factor	Traded Price
2019-11-07	00010001	B	100	DCE_IO_1901	505.3
2019-11-07	00010002	B	100	DCE_IO_1901	506.8

- data_day2

Date	Position_id	Buy/Sell	Quantity	Risk Factor	Traded Price
2019-11-07	00010001	B	100	DCE_IO_1901	505.3
2019-11-07	00010002	B	100	DCE_IO_1901	506.8
2019-11-08	00010003	S	-100	DCE_IO_1901	507.9

Positions are additive (to close a position, we won't change the original position but to do a new reverse trade). Suppose we have two days of position data.

left/right/anti/full_join

In order to find the new positions. We will use:

```
# order matters, data_day2 needs to be placed first.
# anti_join is like "data_day2 - data_day1"
anti_join(data_day2, data_day1, by = "position_id")
```

In order to find older positions, we will use:

- `inner_join` find the common positions

```
inner_join(data_day2, data_day1, by = "position_id")
```

- Because data_day2 includes all data from data_day1. Following two produce the same result

```
left_join(data_day1, data_day2, by = "position_id")
right_join(data_day2, data_day1, by = "position_id")
```

- Produce all items in data_day2

```
left_join(data_day2, data_day1, by = "position_id")
```

Use case for `left_join` / `right_join`

They can be used to do mapping table (aka. vlookup)

Table `Product`:

```
| type_code | type_name |
| 1         | orange    |
| 2         | banana    |
```

Table `Transaction`:

```
| type_code | quantity | customer_id |
| 1         | 1        | A           |
| 2         | 3        | B           |
| 3         | 4        | C           |
| 2         | 2        | D           |
| 1         | 6        | B           |
```

Table `Customer`:

```
| customer_id | customer_phone |
| A           | +123           |
| B           | +456           |
| C           | +789           |
```

Use `left_join` to create a full report

```
left_join(Transaction, Product, by = "type_code") %>%
left_join(Customer, by = "customer_id")
```

type_code	quantity	customer_id	type_name	customer_phone
1	1	A	orange	+123
2	3	B	banana	+456
3	4	C	NA	+789
2	2	D	banana	NA
1	6	B	orange	+456

`group_by` / `summarize`

`group_by` is the way leading to analyze the data at lower-dimension, reducing it to summary. `group_by` can be used together with `summarize`, `mutate`

- `group_by(df, col1, col2, ...)`
- `summarize(df, new_field = some_func_can_process_bulk_data())`

Functions can process bulk data:

- sum/mean/median/sd: basic statistics
- min(x), quantile(x, 0.25), max(x): min/max/quantile
- n()/n_distinct(): count and count distint
- ntile: a rough divide into a few groups
- first(x), last(x), nth(x, 2)
- ...

group_by / summarize: Examples - 1

```
# Add parameter na.rm, if there is NA among the data.
df <- tibble(a = c(1, 3, 4, NA))
```

a
1
3
4
NA

```
summarise(df, total = sum(a))
```

total
NA

group_by / summarize: Examples - 2

```
summarise(df, total = sum(a, na.rm = TRUE))
```

total
8

```
summarise(df, total = mean(a))
```

total
NA

```
summarise(df, total = mean(a, na.rm = TRUE))
```

total
2.666667

group_by / summarize: Examples - 3

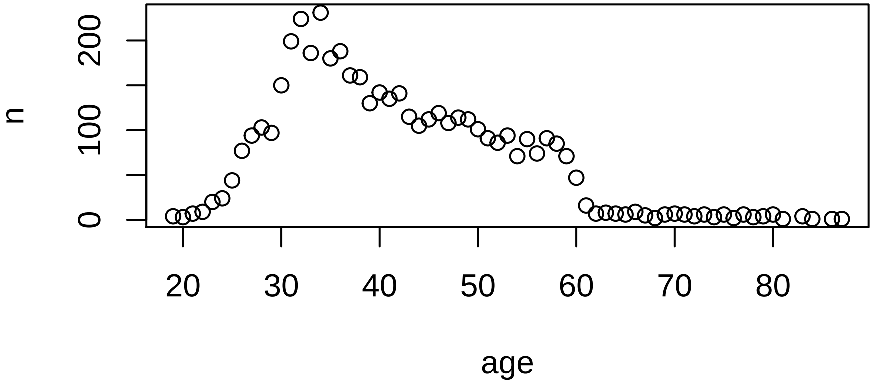
```r
# count number of people in each age group
group_by(bank, age) %>% summarise(n = n())
```

age	n
19	4
20	3
21	7
22	9
23	20
24	24

...

group_by / summarize: Examples - 4

```
group_by(bank, age) %>% summarise(n = n()) %>% plot
```



group_by / summarize: Examples - 5

Use ifelse in summarize/mutate for conditional statement.

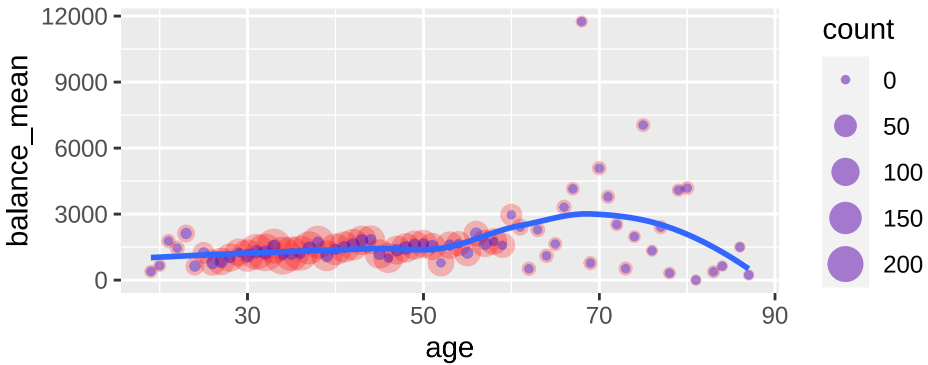
```
bank_age <- group_by(bank, age) %>%
         summarise(balance_mean = mean(balance),
                   count = n(),
                   default_count = sum(ifelse(default == "no", 0, 1)))
```

age	balance_mean	count	default_count
19	393.5000	4	0
20	661.3333	3	0
21	1774.2857	7	0
22	1455.3333	9	0
23	2117.9500	20	1
24	634.6250	24	1
25	1240.0682	44	1
26	788.5584	77	3
27	851.7766	94	4
28	1025.0971	103	1

...

group_by / summarize: Examples

```
# If combined with ggplot, to be learned in next session
bank_age %>%
  ggplot(aes(x = age, y = balance_mean)) +
  geom_point(aes(size = count), alpha = 1/4, color = "red") +
  geom_point(aes(size = default_count), alpha = 1/3, color = "blue") +
  geom_smooth(se = FALSE)
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Group filter

```r
# Find the maximum and minimum balance on each age.
df <- bank %>%
  group_by(age) %>%
  filter(min_rank(balance) == 1 | min_rank(desc(balance)) == 1) %>%
  arrange(age, balance)
```

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previ
19	student	single	unknown	no	0	no	no	cellular	11	feb	123	3	-1	
19	student	single	unknown	no	1169	no	no	cellular	6	feb	463	18	-1	
20	student	single	secondary	no	291	no	no	telephone	11	may	172	5	371	
20	student	single	secondary	no	1191	no	no	cellular	12	feb	274	1	-1	
21	student	single	secondary	no	6	no	no	unknown	9	may	622	1	-1	
21	student	single	secondary	no	6844	no	no	cellular	14	aug	126	3	127	
22	student	single	unknown	no	47	no	no	cellular	3	jul	69	3	-1	
22	admin.	single	secondary	no	4111	no	yes	cellular	19	aug	65	1	-1	
23	technician	single	secondary	no	-306	yes	no	unknown	4	jun	217	2	-1	
23	student	single	secondary	no	9216	no	no	cellular	5	jun	471	2	-1	

…

Count for condition

- `sum(TRUE) == 1`, `sum(FALSE) == 0`

```
# Generate a report for balance and job
d1 <- group_by(bank, job) %>%
  summarise(`balance > 500` = sum(balance > 500))
d2 <- group_by(bank, job) %>%
  summarise(`balance <= 500` = sum(balance <= 500))
# df collects all jobs, in case some jobs are missing from either d1 or d2
# This is a typical example for collecting data.
df <- distinct(bank, job) %>% arrange(job)
df <- left_join(df, d1, by = "job")
df <- left_join(df, d2, by = "job")
df <- mutate(df, total = `balance > 500` + `balance <= 500`)
```

Count for condition - Result

job	balance > 500	balance <= 500	total
admin.	226	252	478
blue-collar	423	523	946
entrepreneur	74	94	168
housemaid	42	70	112
management	521	448	969
retired	127	103	230
self-employed	89	94	183
services	154	263	417
student	41	43	84
technician	353	415	768
unemployed	63	65	128
unknown	21	17	38

group_by and mutate - 1

```r
# mutate with group_by
df <- group_by(tibble(a = 1:10), quantile = ntile(a, 2)) %>%
  mutate(b = a / sum(a))
```

a	quantile	b
1	1	0.0666667
2	1	0.1333333
3	1	0.2000000
4	1	0.2666667
5	1	0.3333333
6	2	0.1500000
7	2	0.1750000
8	2	0.2000000
9	2	0.2250000
10	2	0.2500000

group_by and mutate - 2

```
# filter with group_by
df <- group_by(bank, age) %>% filter(balance == max(balance))
```

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	pre
22	admin.	single	secondary	no	4111	no	yes	cellular	19	aug	65	1	-1	
78	housemaid	married	secondary	no	499	no	no	telephone	16	mar	80	4	-1	
23	student	single	secondary	no	9216	no	no	cellular	5	jun	471	2	-1	
46	management	married	secondary	no	12186	no	no	unknown	20	jun	29	3	-1	
64	retired	married	unknown	no	2923	no	no	cellular	12	mar	120	1	-1	
77	retired	married	tertiary	no	7802	no	no	telephone	4	may	421	1	92	
39	management	single	tertiary	no	12437	no	no	telephone	18	nov	40	1	-1	
28	student	single	secondary	no	11555	no	no	cellular	8	apr	125	2	-1	
81	retired	married	secondary	no	1	no	no	cellular	19	aug	65	5	-1	
33	housemaid	single	tertiary	no	23663	yes	no	cellular	16	apr	199	2	146	
40	self-employed	married	tertiary	no	13669	no	no	cellular	15	oct	138	1	136	
31	housemaid	single	primary	no	26965	no	no	cellular	21	apr	654	2	-1	
30	management	single	tertiary	no	19358	no	no	cellular	19	nov	258	2	-1	
67	blue-collar	married	secondary	no	16353	no	no	cellular	27	oct	223	2	-1	
49	retired	single	primary	no	25824	no	no	unknown	17	jun	94	1	-1	

…

summarize/summarise Example

```r
# summarise with group_by
df <- group_by(tibble(a = 1:10), quantile = ntile(a, 2)) %>%
  summarise(b = sum(a))
```

quantile	b
1	15
2	40

```r
# summarise without a group_by. It will treat entire df as one piece.
df <- summarise(bank,
                with_housing = sum(housing == "yes") / n(),
                age_min = min(age),
                duration_mean = mean(duration))
```

with_housing	age_min	duration_mean
0.5660252	19	263.9613

group_by/ungroup

ungroup() removes group definition, restores the “ungrouped” data frame back to entire data.

```
# wrong
df_wrong <- group_by(bank, age) %>%
  filter(balance == max(balance)) %>%
  summarize(balance = mean(balance)) %>%
  head(n = 3)
```

age	balance
19	1169
20	1191
21	6844

```
# correct
df_correct <- group_by(bank, age) %>%
  filter(balance == max(balance)) %>%
  ungroup %>%
  summarize(balance = mean(balance))
```

balance
13541.21

group_by/ungroup

```r
# If we miss ungroup, we can't remove age. R will prompt."
df1 <- group_by(bank, age) %>%
  filter(balance == max(balance)) %>%
  select(-age) %>% head(n = 3)
## Adding missing grouping variables: `age`
```

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	prev
22	admin.	single	secondary	no	4111	no	yes	cellular	19	aug	65	1	-1	[illegible]
78	housemaid	married	secondary	no	499	no	no	telephone	16	mar	80	4	-1	[illegible]
23	student	single	secondary	no	9216	no	no	cellular	5	jun	471	2	-1	[illegible]

```r
# With ungroup, we can remove age.
df2 <- group_by(bank, age) %>%
  filter(balance == max(balance)) %>%
  ungroup %>%
  select(-age) %>% head(n = 3)
```

job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previo
admin.	single	secondary	no	4111	no	yes	cellular	19	aug	65	1	-1	0
housemaid	married	secondary	no	499	no	no	telephone	16	mar	80	4	-1	0
student	single	secondary	no	9216	no	no	cellular	5	jun	471	2	-1	0

rowwise

`rowwise()` is a special `group_by` which makes every one row a group.

```
df <- tibble(throw_dices = 1:10)
df <- rowwise(df) %>% mutate( mean = mean(sample(1:6, throw_dices, replace = TRUE)))
```

With increasing number of sample size, mean is closer to 3.5.

throw_dices	mean
1	4.000000
2	5.500000
3	1.000000
4	4.250000
5	2.800000
6	4.000000
7	2.428571
8	3.875000
9	3.222222
10	3.600000

Take-home: `group_by` and `summarise/summarize`

- `group_by` is a like folding a paper without tearing it later.
- `summarise` tears the paper to do individual pieces.
- Therefore, `group_by` can be used with other verbs, `mutate`, `filter`, which will work within the group.
- `summarise` can be used without group_by, then it will apply to entire data as one whole group.
- `ungroup` is to unfold it
- `rowwise` is to create one-row group for all.

bind_rows

- bind_rows is the + operator for data frames.

```
# add empty data frame is the same.
df1 <- bind_rows(tibble(a = 3:4), tibble())
```

a
3
4

```
df2 <- bind_rows(tibble(), tibble(a = 3:4))
```

a
3
4

bind_rows: Use case - 1

I usually use bind_rows to collect results. For example,

```
new_positions <- tibble()
closed_positions <- tibble()

for (i in length(dates)-1) {
  old_date <- dates[i]
  new_date <- dates[i+1]

  new_data <- filter(position, date == new_date)
  old_data <- filter(position, date == old_date)

  new_positions <- bind_rows(new_positions,
                             anti_join(new_data, old_data, by = "position_id"))
}

# new_positions contains all new positions on their day 1
```

bind_rows: Use case - 2

If row order matters, bind_row can be used to re-order/splice and recombine.

```r
# Get head and tail
# Note: use { } to use the .
df <- arrange(bank, age) %>%
      { bind_rows(head(., n = 5), tail(., n = 5)) }
```

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previ
19	student	single	primary	no	103	no	no	cellular	10	jul	104	2	-1	0
19	student	single	unknown	no	0	no	no	cellular	11	feb	123	3	-1	0
19	student	single	secondary	no	302	no	no	cellular	16	jul	205	1	-1	0
19	student	single	unknown	no	1169	no	no	cellular	6	feb	463	18	-1	0
20	student	single	secondary	no	502	no	no	cellular	30	apr	261	1	-1	0
83	retired	divorced	primary	no	0	no	no	telephone	31	may	664	1	77	3
83	retired	divorced	primary	no	1097	no	no	telephone	5	mar	181	1	-1	0
84	retired	divorced	primary	no	639	no	no	telephone	18	may	353	3	-1	0
86	retired	married	secondary	no	1503	no	no	telephone	18	mar	165	3	101	1
87	retired	married	primary	no	230	no	no	cellular	30	oct	144	1	-1	0

bind_rows: Use case - 3

```r
# summary
df1 <- summarise_if(bank, is.numeric, mean)
```

age	balance	day	duration	campaign	pdays	previous
41.1701	1422.658	15.91528	263.9613	2.79363	39.76664	0.5425791

```r
# add summary to the records
df2<- tail(bind_rows(bank, summarise_if(bank, is.numeric, mean)), n = 1)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	prev
4522	41.1701	NA	NA	NA	NA	1422.658	NA	NA	NA	15.91528	NA	263.9613	2.79363	39.76664	0.54

bind_rows: Use case - 4

```r
# bind_rows can match column names and type.
# let's adjust the column order.
# As due-deligence, better to check the result.
# I remember earlier version of dplyr doesn't do match.
df <- tail(bind_rows(bank, summarise_if(bank, is.numeric, mean) %>%
  select(balance, day, everything())), n = 1)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	prev
4522	41.1701	NA	NA	NA	NA	1422.658	NA	NA	NA	15.91528	NA	263.9613	2.79363	39.76664	0.54

bind_cols

- bind_cols is to extend the data frame in width.

Use cases

- It's a lazyman's `left_join` or `select`
- It copies the columns
- I usually find it useful to generate data frame for reports.

```
dt1 <- bind_cols(select(bank, job), select(bank, education))
dt1[1:3,]
```

job	education
unemployed	primary
services	secondary
management	tertiary

bind_cols

If there are same-name columns in the data frames, they will be renamed by `...1`, `...2`, `...n`.

```
dt2 <- bind_cols(dt1, dt1)
## New names:
## * job -> job...1
## * education -> education...2
## * job -> job...3
## * education -> education...4
dt2[1:3,]
```

job...1	education...2	job...3	education...4
unemployed	primary	unemployed	primary
services	secondary	services	secondary
management	tertiary	management	tertiary

bind_cols: Use cases

```r
d1 <- filter(bank, month == "sep") %>%
  summarize(duration = mean(duration)) %>%
  rename(`Duration Sep` = duration)
d2 <- filter(bank, month == "oct") %>%
  summarize(duration = mean(duration)) %>%
  rename(`Duration Oct` = duration)
d3 <- filter(bank, month == "nov") %>%
  summarize(duration = mean(duration)) %>%
  rename(`Duration Nov` = duration)

df <- bind_cols(d1, d2, d3)
```

Duration Sep	Duration Oct	Duration Nov
215.7308	272.8	272.0668

Exercise

1. How to know the row number of the wrong date

```
df <- tibble(dt = c("2019-10-01", "2019-31-12",
                     "2019-03-17", "2019-02-29",
                     "2019-09-30"))
```

dt
2019-10-01
2019-31-12
2019-03-17
2019-02-29
2019-09-30

Output:

```
## Wrong dates on rows: 2, 4
```

Exercise

2. How to get sub-total and total on mean of age and balance, group by job and education?

job	education	mean(Age)	median(Balance)
services	primary	...	...
services			
services	+	...	...
...			
+	+	...	...

Exercise

3. To evaluate a portfolio of options for its total value.

```
df <- tibble(type = sample(c("c", "p"), 100, replace = TRUE),
                 strike = round(runif(100) * 100, 0),
                 underlying = round(runif(100) * 100, 0),
                 Time = 1,
                 r = 0.01,
                 b = 0,
                 sigma = 0.3)
df %>% rowwise() %>% mutate(
  price = GBSOption(TypeFlag = "p", S = 3500, X = 3765,
          Time = 1/12, r = 0, b = 0, sigma = 0.3)@price
) %>% ungroup()
```

Not all R functions are able to take in vector and output vector. `GBSVolatility` can only take in single number, not vector.

Use `rowwise() %>% mutate(... = GBSVolatility) %>% ungroup()`. `rowwise()` is a special kind of `group_by()` so it can pair up with `ungroup()`.

Assignment

1. Exploratory Data Work on the bank dataset. Find 7 insights from data. Use R Markdown.

```
---
title: "FE8828 Assignment for Exploratory Data Analysis"
author: "Yang Ye <sub> <Email:yy@runchee.com> </sub>"
date: "Oct 2021"
output: html_document
---

```{r setup, include=FALSE}
library(tidyverse)
library(lubridate)
library(bizdays)
# Use echo = TRUE for assignment is an exception, so code is visible.
knitr::opts_chunk$set(echo = TRUE, fig.align="center", collapse = TRUE, cache = TRUE)
bank <- read.csv("https://goo.gl/PBQnBt", sep = ";")
```

Finding #1
This data contains `r nrow(data)` rows.

Finding #2
```{r}
# Find the big age group
bank %>%
  group_by(age_group = (age %/% 10) * 10) %>%
  summarise(count = n()) %>%
  arrange(age_group) -> res

res

plot(res$age_group, res$count)
```

Discover insights of data frame: bank
- Employment
- Social attributes.
- Count for sub-total / total, plot graph
```

Assignment

2. Book option trades

2.1 Copy the options data from https://www.nasdaq.com/symbol/goog/option-chain?dateindex=1

- Select "December 2020"/Composite/Call&Puts/Near the Money/All(Types).
- Copy the data to Excel, if it spans multiple pages, include all pages.
- Load the data in R Studio as data frame. Clean it to have following columns. Note the original data make calls and puts share the same strike column.

Exp. Date | Strike | Open Int. | OptionType | Bid | Ask | Underlying | Today

```
- Open Int. is the short-form for Open Interest.
- OptionType is "c" for "Calls", "p" for "Puts"
- Underlying/Today can be found on the top of the page.
```

2.2 Calcualte the total valuation of 1) call alone, 2) put alone, 3) call and put. `Total Valuation = Open Interest * (Bid + Ask) / 2`.

2.3 Find those in the money (for calls, strike < underlying. for puts, strike > underlying.) and calculate their total `Open Interest`.

Assignment

2.4. Plot the volatility curve, strike v.s. vol. For strike < current price, use puts' price; for strike > current price, use calls' price.

```r
# GBSVolatility(price, TypeFlag, Underlying, Strike, Time, r, b, tol, maxiter)
# Use Price to back-out implied volatility. Assume r = 0.03
# Example:

GBSVolatility(867.30, "c", 1135.67, 240,
as.numeric((as.Date("2020-12-18") - as.Date("2020-09-29")))/365,
r = 0.03, b = 0)
## [1] 1.770673e-16
```

- Not all R functions are able to take in vector and output vector. `GBSVolatility` can only take in single number, not vector.
- Use `rowwise() %>% mutate(vol = GBSVolatility(...)) %>% ungroup()` as a starting point.
- `rowwise()` is a special kind of `group_by()` so it can pair up with `ungroup()`.

tidyr: `pivot_longer/pivot_wider`

Wide format <=> Long format

- Wide format is more familiar to us. Column name is the data attribute
  - Wide data provides high-density view of data, more human-friendly.
- Long format is what we reformat the data that common attributes are gathered together as a single variable.
  - Long data is processing-friendly. This is call Tidy data principles https://en.wikipedia.org/wiki/Tidy_data

Wide v.s. Long

Wide format

```
wfmt <- tibble(date = seq(from = as.Date("2019-01-01"),by = "day",length.out = 5),
               Copper_qty = round(runif(5) * 1000, 0),
               Gold_qty = round(runif(5) * 1000, 0),
               Silver_qty = round(runif(5) * 1000, 0))
```

date	Copper_qty	Gold_qty	Silver_qty
2019-01-01	891	975	462
2019-01-02	611	131	637
2019-01-03	479	948	386
2019-01-04	48	247	211
2019-01-05	922	43	533

Wide v.s. Long

Long format

```r
library(tidyr)
df <- pivot_longer(wfmt, col = ends_with("qty"),
                   names_to = "key", values_to = "value")
```

date	key	value
2019-01-01	Copper_qty	891
2019-01-02	Copper_qty	611
2019-01-03	Copper_qty	479
2019-01-04	Copper_qty	48
2019-01-05	Copper_qty	922
2019-01-01	Gold_qty	975
2019-01-02	Gold_qty	131
2019-01-03	Gold_qty	948
2019-01-04	Gold_qty	247
2019-01-05	Gold_qty	43
2019-01-01	Silver_qty	462
2019-01-02	Silver_qty	637
2019-01-03	Silver_qty	386
2019-01-04	Silver_qty	211
2019-01-05	Silver_qty	533

pivot_long example with *Bank* dataset

```
wfmt <- group_by(bank, job) %>%
  summarize(yy = sum(ifelse(default == "yes", 1, 0)),
            nn = sum(ifelse(default == "no", 1, 0))) %>%
  head(., 4) # slide space is limitd. Just take first 4 rows.
df <- pivot_longer(wfmt, cols = c("yy","nn"),
                   names_to = "default", values_to = "value") %>%
      arrange(job, default)
```

job	yy	nn
admin.	6	472
blue-collar	14	932
entrepreneur	7	161
housemaid	2	110

job	default	value
admin.	nn	472
admin.	yy	6
blue-collar	nn	932
blue-collar	yy	14
entrepreneur	nn	161
entrepreneur	yy	7
housemaid	nn	110
housemaid	yy	2

pivor_wider example with *Bank* dataset

```
lfmt <- group_by(bank, job, default) %>% summarize(nn = n()) %>% head(., 4)
df <- pivot_wider(lfmt, names_from=default, values_from=nn)
# How to take care of converting NA to zero?
```

job	default	nn
admin.	no	472
admin.	yes	6
blue-collar	no	932
blue-collar	yes	14

job	no	yes
admin.	472	6
blue-collar	932	14

Combine different columns' Quantity - 1

date	Copper_qty	Gold_qty	Silver_qty
2019-01-01	237	581	921
2019-01-02	557	382	907
2019-01-03	883	126	35
2019-01-04	173	525	596
2019-01-05	896	813	503

```
df <- wfmt %>%
      pivot_longer(!date, names_to="key", values_to="value") %>%
      group_by(date) %>%
      summarize(value1 = sum(value)) %>%
      rename(value = value1) %>%
      mutate(key = "Total") %>%
      pivot_wider(names_from=key, values_from=value) %>%
      inner_join(wfmt, ., by = "date")
```

date	Copper_qty	Gold_qty	Silver_qty	Total
2019-01-01	237	581	921	1739
2019-01-02	557	382	907	1846
2019-01-03	883	126	35	1044
2019-01-04	173	525	596	1294
2019-01-05	896	813	503	2212

Combine different columns' Quantity - 2

```r
# although this works...
# It takes "Hard coding" of column names "Copper_qty Gold_qty Silver_qty".
df <- wfmt %>% mutate(total = Copper_qty + Gold_qty + Silver_qty)
```

date	Copper_qty	Gold_qty	Silver_qty	total
2019-01-01	237	581	921	1739
2019-01-02	557	382	907	1846
2019-01-03	883	126	35	1044
2019-01-04	173	525	596	1294
2019-01-05	896	813	503	2212

Take-Home: CRUD with dplyr

Create:

- add new rows. bind_rows()

Read:

- You have known enough: filter/select/joins/... to get what you need.

Delete:

- Use filter to exclude the row(s). Save the result.

```
new_df <- filter(old_df, a > 1)
```

Take-Home: CRUD with dplyr

Update: - Use either data frame way or mutate.

```
# get all row numbers for students
# . refers to the output of the pipe %>%. .$nnn => df$nnn
row_nums <- mutate(bank, nnn = 1:n()) %>%
            filter(job == "student" & age < 22) %>%
            select(nnn) %>%
            .$nnn

bank1 <- bank
bank1[row_nums, "taxable"] <- "no"
bank1[setdiff(1:nrow(bank), row_nums), "taxable"] <- "yes"

# use dplyr
bank1 <- mutate(bank, taxable = ifelse(job == "student" & age < 22, "no", "yes"))
distinct(bank1, taxable)
```