

R 기본문법과 전처리

📖 목차 📖

[여러개의 패키지를 한 번에 다 불러오는 법](#)

Tidyverse

[dplyr 패키지의 함수](#)

[filter\(\)로 행 선택하기](#)

[arrange\(\)로 행 정렬하기](#)

[select\(\)를 이용하여 변수 이름으로 열 선택하기](#)

[mutate\(\)로 새로운 변수 만들기](#)

[summarize\(\)로 변수 요약하기\(기술통계분석\)](#)

[변수 한 번에 처리하기 mutate_at\(\), mutate_if\(\)](#)

[개별 데이터 값 변환해주기](#)

결측값 확인 및 처리

[각 칼럼 안에 있는 고유한 개수/고유값 확인](#)

[특정 조건을 만족하는 행 삭제](#)

[levels 옵션을 활용해 level의 순서 정해주기](#)

[stringr 패키지를 이용하여 문자열 추출하기](#)

[문자열 분리](#)

[gsub 함수를 이용하여 R에서 숫자 혹은 문자만 분리하는 방법](#)

[특정 열 이름 바꾸기](#)

[join 계열 함수로 데이터 병합하기](#)

lubridate

[setdiff로 차집합 구하기](#)

[gather/spread를 이용한 데이터 형태 변환](#)

여러개의 패키지를 한 번에 다 불러오는 법

`pacman::p_load(패키지)`

```
pacman::p_load(tidyverse,
               caret,
               data.table,
               magrittr,
               corrplot,
               cluster,
               Rtsne)
```

Tidyverse

: 다양한 패키지 적재

```
install.packages("tidyverse")

library(tidyverse)

#> — Attaching packages — tidyverse 1.3.2 —
#> ✓ ggplot2 3.3.6 ✓ purrr 0.3.4
#> ✓ tibble 3.1.8 ✓ dplyr 1.0.9
#> ✓ tidyr 1.2.0 ✓ stringr 1.4.0
#> ✓ readr 2.1.2 ✓ forcats 0.5.1
#> — Conflicts — tidyverse_conflicts() —
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag() masks stats::lag()
```

dplyr 패키지의 함수

<code>filter()</code>	데이터에서 관측(대상)을 측정값을 기준으로 선택
<code>arrange()</code>	관측(대상)을 기준으로 데이터를 정렬
<code>select()</code>	변수이름으로 일부 변수만 데이터에서 선택
<code>mutate()</code>	기존 변수를 사용하여 새로운 변수를 데이터에 추가
<code>summarize()</code>	: 여러 측정치를 하나의 통계량으로 요약

이러한 작업은 `group_by()` 함수와 같이 사용되어 전체 데이터에서 작업이 수행되는 것이 아니라 관측(대상)의 그룹별로 수행되도록 조정할 수 있음

filter()로 행 선택하기

1. 선택조건이 하나인 경우 : `filter(데이터프레임, 조건)`

```
filter(mpg, manufacturer=="hyundai")
filter(mpg, cty > 28)
```

2. 여러 조건을 만족하는 행 추출하기 : `filter(데이터프레임, 조건1, 조건2, ..., 조건n)`

```
filter(mpg, manufacturer=="hyundai", cty >= 28)
```

```
filter(mpg, model=="sonata" | cty >= 28)
```

Cf) 다만 복합연산자 (and, or)을 쓸 때는 우선순위를 고려하여 괄호를 이용

arrange()로 행 정렬하기

```
arrange(데이터프레임, 첫번째 정렬 기준 변수, 두번째 정렬 기준 변수, ....)
```

- `desc()` 를 이용하여 내림차순으로 정렬하기

```
arrange(a, desc(cyl))
```

- 기본 R 함수를 이용한 정렬

```
sort(decreasing = FALSE)
data[order(data)]
```

select()를 이용하여 변수 이름으로 열 선택하기

- 변수 이름을 통해 열을 선택하거나 제외 가능 (제외 시에는 "-" 을 앞에다 쓰기)
- 변수의 이름을 매칭하여 선택하기

<code>starts_with("abs")</code>	abc 로 이름이 시작하는 모든 변수 예) <code>select(a, starts_with("c"))</code>
<code>ends_with("abs")</code>	abc 로 이름이 끝나는 모든 변수
<code>contains("abs")</code>	abc 를 이름에 포함하고 있는 모든 변수 예)
<code>matches("(.)\\1")</code>	정규 표현식을 만족하는 이름을 가진 모든 변수
<code>num_range("x", 1:3)</code>	"x1," "x2," "x3"이라는 이름의 변수

- 변수 이름 바꾸기 : **(새로운 변수 이름)=(기존 변수 이름)**

```
select(a, model, city=cty, highway=hwy)
```

cf) 전체 데이터를 유지한 상태에서 변수 이름만 변경하려면, `rename()` 함수를 이용

```
rename(a, city=cty, highway=hwy)

train %<>%
  rename('Time Convenient'='Departure/Arrival time convenient',
        'Departure Delay'='Departure Delay in Minutes',
        'Arrival Delay'='Arrival Delay in Minutes')

#(새로운 변수 이름)=(기존 변수 이름)
```

- **변수 순서 바꾸기** : 함수는 나열된 변수의 순서에 따라 새롭게 만들어진 데이터 프레임의 변수의 순서를 조정

```
select(a, cty, hwy)
select(a, cty, hwy, everything()) #everything은 이미 선택된 변수를 제외한 나머지 변수를 의미
```

mutate()로 새로운 변수 만들기

기존 변수를 이용하여 새로운 변수를 만들어 데이터 프레임의 가장 마지막 열로 추가

```
mutate(데이터프레임, 새로운_변수=기존_변수_연산식, ....)
```

새롭게 만들어진 변수만 데이터에 남기려면 mutate() 대신 transmute()를 사용 (문법은 동일)

summarize()로 변수 요약하기(기술통계분석)

```
summarize(데이터프레임, 요약변수이름=요약함수(변수), ....)
```

summarise()	개별 함수에만 적용되는 함수
summarise_at()	지정된 변수명이나 위치에 있는 변수에만 적용되는 함수
summarise_if()	특정 조건을 만족하는 변수에만 적용되는 함수
summarise_all()	모든 변수들에 적용되는 함수

다양한 요약함수

n()	변수의 크기를 구한다. (개수 카운트)
sum()	수치 변수의 합을 구한다.
mean()	수치 변수의 평균을 구한다.
median()	수치 변수의 중위수를 구한다.
sd()	수치 변수의 표준편차를 구한다.
var()	수치 변수의 분산을 구한다
min()	수치 변수의 최소값을 구한다.
max()	수치 변수의 최대값을 구한다
quantile(변수, probs)	수치 변수의 probs`분위수를 구한다.

변수 한 번에 처리하기 mutate_at(),mutate_if()

1. **mutate_if()** : 지정해준 모든 변수에 대해 계산 식을 적용

```
Mutate_If_Data = STOCK %>% mutate_if(is.integer,as.numeric)
```

2. **mutate_at()** : 지정한 변수들에 대해 계산식을 적용

```
data<-data %>% mutate_at(vars(Category,Downloads,`Rated for`),as.factor)
```

개별 데이터 값 변환해주기

1. **mutate** → **_recode**

```
train %<>%
  mutate(`Customer Type` =
    recode(`Customer Type`, "Loyal Customer"="Loyal", "disloyal Customer"="Disloyal"),
    `Type of Travel` =
    recode(`Type of Travel`, "Personal Travel"="Personal", "Business travel"="Business"))

# recode(데이터 column, '원래값'='바꾸고 싶은 값')
```

2. `mutate` → `ifelse`

```
train %<>%
  mutate(`Customer Type` = if_else(`Customer Type` == 'Loyal Customer', 'Loyal', 'Disloyal'),
    `Type of Travel` = if_else(`Type of Travel` == 'Personal Travel', 'Personal', 'Business'))
```

결측값 확인 및 처리

- 결측값이 포함되어 있는지 확인하는 방법: `is.na()`
- 결측값이 총 몇 개인지 계산하는 방법: `sum(is.na())`
- `colSums()` : 데이터 프레임 내 다수 변수들의 결측치 구할 수 있음
- 결측값을 통계분석 시 제외(미포함) : `na.rm = TRUE`
- 결측값이 들어있는 행 전체를 데이터 셋에서 제거 : `na.omit()`
- 특정 행과 열에 결측값이 들어있는 행을 데이터 셋에서 제거 : `complete.cases()`

```
# Cars93 데이터 프레임의 "Rear.seat.room" 칼럼 내 결측값이 있는 행 전체 삭제
> Cars93_2 <- Cars93[ complete.cases(Cars93[ , c("Rear.seat.room")]), ]
> sum(is.na(Cars93_2))

> # Cars93 데이터 프레임의 23~24번째 칼럼 내 결측값이 있는 행 전체 삭제
> Cars93_3 <- Cars93[ complete.cases(Cars93[ , c(23:24)]), ]
> sum(is.na(Cars93_3))
```

각 칼럼 안에 있는 고유한 개수/고유값 확인

- `n_distinct()` (summarise의 내장함수) : 고유값 개수 확인 가능
- R에서 중복된 행을 삭제하려면 `unique()` 를 사용

```
> x
  name
1   a
2   c
3   a
4   b
5   b
6   c
7   d

> y <- unique(x)
> y
  name
1   a
2   c
4   b
7   d
```

- unique 값만 출력하고 싶을때

```
data %>% select(Category, Downloads, `Rated for`) %>% apply(2, unique)
```

- `uplicated()` : 사용 시 : 중복된 행을 T/F 값으로 반환해줌 (인덱스를 알고 싶다면 which와 함께 쓰기)

[R.아르] 중복된 행 삭제하기 unique() / duplicated()

R에서 중복된 행을 삭제하려면 unique()를 사용하면 된다. 다음과 같은 x에서 unique()를 실행시키면 중복된 행들이 사라지는 것을 알 수 있다. 중복된 행이 몇 개나 되는지 숫자가 알고 싶을 때는? 지난 번에 본 nrow를 쓰면 되겠다. 여러 개의 열이 있을 경우에는 행의 모든 값이 완전히 동일할 때만 중복으로 간주한다.


<https://lightblog.tistory.com/18>

3

a

각 칼럼안에 있는 고유한 이름(혹은 값)은 몇 개나 있나 (n_distinct)

R로 작업을 하다보면, 각 칼럼에 고유한 값이 몇 번 나오는지 궁금할 때가 있습니다. 예를 들어 제가 들고 있는 데이터프레임 이름이 value인데, 그 데이터의 구조는 다음과 같은데요. GeoName이라는 칼럼에는 미국의 주 이름이, Description에는 각 산업 이름이 들어가 있습니다. 연도별 데이터를 모으다보니 주 이름이나 산업 이름이 겹쳐서 여러 번 나오게 됩니다.

 <https://approximation.tistory.com/56>

	<INDS>	<INDS>	<INDS>	<INDS>
1	1000	Alabama All Industries	X2012	3606738
2	1000	Alabama All Industries	X2013	3633426
3	1000	Alabama All Industries	X2014	3652139
4	1000	Alabama All Industries	X2015	3740572
5	1000	Alabama All Industries	X2016	3809422
6	1000	Alabama All Industries	X2017	3936511
7	1000	Alabama All Industries	X2018	4068476
8	1000	Alabama All Industries	X2019	4167984
9	1000	Alabama " Private Industries"	X2012	3325090
10	1000	Alabama " Private Industries"	X2013	3387036

특정 조건을 만족하는 행 삭제

```
df <- df[!(df$column == "condition" ), ]
```

levels 옵션을 활용해 level의 순서 정해주기

```
f <- factor(c(1,2,3), levels=c(3,2,1))
# Levels: 3 2 1

levels(f)
[1] "3" "2" "1"
```

order option : level 순서 정하기 (추후 plot을 그릴 때 order 순서대로 그려진다.)

```
f <- factor(c(1,2,3), levels=c(3,2,1), order=T)

# Rated_for -> "3+", "7+", "12+", "16+", "18+"
data$`Rated for` %<% factor(levels=c("3+", "7+", "12+", "16+", "18+"), order= T)

# Downloads

data$Downloads %<% factor(levels=c("10T+", "50T+", "1L+", "5L+", "10L+", "50L+", "1Cr+", "5Cr+", "10Cr+", "50Cr+", "100Cr+", "500Cr+"), order=T)

str(data$`Rated for`)
## Ord.factor w/ 5 levels "3+<"7+<"12+<...: 1 1 3 1 1 3 1 3 3 3 ...
```

stringr 패키지를 이용하여 문자열 추출하기

- R stringr 패키지 패턴 매칭
- stringr 패키지/정규표현식

```
install.packages("stringr")
library(stringr)
```

```
fruit <- c("apple", "banana", "pear", "pinapple")
```

1. `str_detect(string, pattern)` : 패턴 매칭 T/F

```
> str_detect(fruit, "a")
[1] TRUE TRUE TRUE TRUE
# a가 있으면 T/없으면 F

> str_detect(fruit, "^a") # ^:맨앞
[1] TRUE FALSE FALSE FALSE
# a로 시작하면 T

> str_detect(fruit, "a$") # $:맨뒤
[1] FALSE TRUE FALSE FALSE
```

```
# a로 끝나면 T

> str_detect(fruit, "b")
[1] FALSE TRUE FALSE FALSE

> str_detect(fruit, "[ek]")
[1] TRUE FALSE TRUE TRUE
#[ek] : e 나 k 있으면 T
```

2. `str_count(string, pattern = "")` : 매치하는 곳의 수

```
> str_count(fruit, "a")
[1] 1 3 1 1
> str_count(fruit, "p")
[1] 2 0 1 3
> str_count(fruit, "e")
[1] 1 0 1 2
> str_count(fruit, c("a", "b", "p", "p"))
[1] 1 1 1 3
```

3. `str_extract(string, pattern)` : 매치된 부분 문자열 추출, 매치되지 않으면 NA

```
shopping_list <- c("apples x4", "bag of flour", "bag of sugar", "milk x2")
```

```
> str_extract(shopping_list, "\\d") # d:숫자
[1] "4" NA NA "2"

> str_extract(shopping_list, "[a-z]+") # 영단어
[1] "apples" "bag" "bag" "milk"

# 대문자도 포함하고 싶으면 "[A-Z]+"
# 한글의 경우는 "[가-힣]"
```

문자열 분리

- `str_sub` : 범위에 해당하는 부분의 문자열 추출해줌
- `str_extract` : `\\d`는 숫자만 찾는 패턴 / `\\w`는 영어, 한글, 숫자를 모두 찾는 패턴
- `str_split_fixed` : 문자열 분리 이후 행렬로 변환 (`str_extract_all (simplify=TRUE)`와 유사)
 - `str_split(string, pattern, n = Inf, simplify = FALSE)`

```
> fruits <- c(
+   "apples and oranges and pears and bananas",
+   "pineapples and mangos and guavas"
+ )
>
> str_split(fruits, " and ") # list로 반환
[[1]]
[1] "apples" "oranges" "pears" "bananas"

[[2]]
[1] "pineapples" "mangos" "guavas"

> str_split(fruits, " and ", simplify = TRUE) # matrix로 반환
      [,1]      [,2]      [,3]      [,4]
[1,] "apples"  "oranges" "pears"  "bananas"
[2,] "pineapples" "mangos" "guavas" ""

>
> # n : 분리 갯수 제한
> str_split(fruits, " and ", n = 3)
[[1]]
[1] "apples" "oranges" "pears and bananas"

[[2]]
[1] "pineapples" "mangos" "guavas"

> str_split(fruits, " and ", n = 2)
[[1]]
[1] "apples" "oranges and pears and bananas"

[[2]]
```

```
[1] "pineapples"      "mangos and guavas"

>
> # n이 크면 분리할 수 있는 만큼 분리
> str_split(fruits, " and ", n = 5)
[[1]]
[1] "apples"  "oranges" "pears"   "bananas"

[[2]]
[1] "pineapples" "mangos"    "guavas"

>
> # 행렬로 변환
> str_split_fixed(fruits, " and ", 3)
      [,1]      [,2]      [,3]
[1,] "apples"   "oranges" "pears and bananas"
[2,] "pineapples" "mangos"  "guavas"
> str_split_fixed(fruits, " and ", 4)
      [,1]      [,2]      [,3]      [,4]
[1,] "apples"   "oranges" "pears"   "bananas"
[2,] "pineapples" "mangos"  "guavas"  ""
```

- Simplify=T : 행렬로 반환

```
> str_extract_all(shopping_list, "\\b[a-z]+\\b", simplify = TRUE)
      [,1]      [,2]      [,3]
[1,] "apples"   ""         ""
[2,] "bag"      "of"      "flour"
[3,] "bag"      "of"      "sugar"
[4,] "milk"     ""         ""
> str_extract_all(shopping_list, "\\d", simplify = TRUE)
      [,1]
[1,] "4"
[2,] ""
[3,] ""
[4,] "2"
```

gsub 함수를 이용하여 R에서 숫자 혹은 문자만 분리하는 방법

1. 숫자만 분리

```
x="ab123"
gsub('\\\\D','', x)

[1] "123"
```

2. 문자만 분리

```
x="ab123"
gsub('\\\\d','', x)

[1] "ab"
```

특정 열 이름 바꾸기






1. temp 데이터의 1열 이름을 date로 바꾸기

```
colnames(temp)[1]<-"date"
```

2. temp데이터의 2행 이름을 seoul로 바꾸기

```
rownames(temp)[2]<-"seoul"
```

join 계열 함수로 데이터 병합하기

join 종류	설명	dplyr 패키지 함수
내부조인(INNER JOIN)		inner_join() 함수
왼쪽 부분 외부 조인 (LEFT OUTER JOIN)		left_join() 함수
오른쪽 부분 외부 조인 (RIGHT OUTER JOIN)		right_join() 함수
외부 조인 (FULL OUTER JOIN)		full_join() 함수
왼쪽/오른쪽 부분 안티 조인 (LEFT/RIGHT ANTI JOIN)		anti_join() 함수

```
inner_join(Departments, Employees, by = "Department")
left_join(Departments, Employees, by = "Department")
right_join(Departments, Employees, by = "Department")

#왼쪽 안티 조인 실시하기
> anti_join(Departments, Employees, by = "Department")

#오른쪽 안티조인 실시하기
> anti_join(Employees, Departments, by = "Department")
```

- `join_all` : `join_all(dfs, by = NULL, type = "left", match = "all")`
 - type : type of join: left (default), right, inner or full

```
data<-plyr::join_all(list(train, oil, holidays), by="date", type="left", match = "first")
```

lubridate

1. 문자열을 날짜로 변경하기

```
x = c('2015-07-01', '2015-08-01', '2015-09-01')
y = c('07/01/2015', '08/01/2015', '09/01/2015')

YYYY-MM-DD 형태가 아닌 다른 형태로 입력된 경우, format을 직접 입력하여 Date 형태로 변경가능
lubridate 패키지를 이용할 경우 YYYY-MM-DD 형태는 ymd(),
MM-DD-YYYY 형태는 mdy() 함수를 사용해 손쉽게 Date 형태로 변경할 수 있습니다.

ymd(x)

## [1] "2015-07-01" "2015-08-01" "2015-09-01"

mdy(y)

## [1] "2015-07-01" "2015-08-01" "2015-09-01"
```


순서	함수
year, month, day	ymd()
year, day, month	ydm()
month, day, year	mdy()
day, month, year	dmy()
hour, minute	hm()
hour, minute, second	hms()
year, month, day, hour, minute, second	ymd_hms()

2. 날짜 관련 정보 추출

정보	함수
Year	year()
Month	month()
Week	week()
Day of year	yday()
Day of month	mday()
Day of week	wday()
Hour	hour()
Minute	minute()
Second	second()
Time zone	tz()

```
wday(labels=TRUE) # 월/화/수/목/금 이란식으로 변환됨
```

3. 날짜 순서 생성하기 : `seq()` 함수를 이용할 경우 날짜 벡터를 생성

```
1년 단위로 날짜 벡터 생성

seq(ymd('2010-01-01'), ymd('2015-01-01'), by='years')

## [1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01" "2014-01-01"
## [6] "2015-01-01"

2일 간격으로 날짜 벡터 변환

seq(ymd('2010-09-01'), ymd('2010-09-30'), by='2 days')

## [1] "2010-09-01" "2010-09-03" "2010-09-05" "2010-09-07" "2010-09-09"
## [6] "2010-09-11" "2010-09-13" "2010-09-15" "2010-09-17" "2010-09-19"
## [11] "2010-09-21" "2010-09-23" "2010-09-25" "2010-09-27" "2010-09-29"
```

4. 분석에 사용할 기간 세팅하기

```
lakers %>% filter(time_index <= ymd_hms("2008-10-28 12:00:00"))
```

setdiff로 차집합 구하기

```
x = c(1, 2, 3, 4, 5)
y = c(1, 2, 5, 7, 8)

> setdiff(x, y)
[1] 3 4

> setdiff(y, x)
[1] 7 8
```

gather/spread를 이용한 데이터 형태 변환

▼ 예시로 쓰일 데이터 예제

```
## # A tibble: 100 x 6
##   user_id user_age user_gender song_id streaming_count song_class_flag
##   <int>   <dbl> <chr>      <chr>          <int> <chr>
## 1 10000    49 여성      i             19 비인기곡
## 2 10001    49 여성      m             28 비인기곡
## 3 10002    26 여성      f             21 인기곡
## 4 10003    48 남성      e             14 인기곡
## 5 10004    49 여성      o             17 비인기곡
## 6 10005    37 여성      h             22 비인기곡
## 7 10006    48 여성      n             15 비인기곡
## 8 10007    43 남성      g             22 비인기곡
## 9 10008    22 남성      m             28 비인기곡
## 10 10009    34 여성      g             18 비인기곡
## # ... with 90 more rows
```

데이터의 형태 : long format(테이블 자체가 김) / wide format (옆으로 넓은 형태)

1. **spread** : long format ~> wide format

```
spread(
  data = 데이터,
  key = "넓은 형태로 나열하게 될 변수",
  value = "key 변수에 대한 값"
)
```

▼ 예시

```
## # A tibble: 30 x 3
##   user_gender song_id total_streaming_count
##   <chr>      <chr>          <int>
## 1 남성      a             43
## 2 남성      b            127
## 3 남성      c             33
## 4 남성      d             65
## 5 남성      e             44
## 6 남성      f             98
## 7 남성      g            146
## 8 남성      h             50
## 9 남성      i             48
## 10 남성     j             45
## # ... with 20 more rows

# song_id를 기준으로 wide formatting
temp2_spread <- temp2 %>%
  spread(key = "song_id", value = "total_streaming_count")

temp2_spread

## # A tibble: 2 x 16
##   user_gender a b c d e f g h i j k
##   <chr> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 남성    43 127 33 65 44 98 146 50 48 45 24
```

```
## 2 여성      104    21    37    35    74    63   100    69    19    82    42
## # ... with 4 more variables: l <int>, m <int>, n <int>, o <int>
```

key나 value 파라미터로 받지 않은 변수들은 기존 형태를 유지하되,

key를 파라미터로 받은 변수는 옆으로 나열하게 되고, value를 파라미터로 받은 변수는 key 변수 값으로 매칭

2. `gather`: wide format ~> long format

```
gather(
  key = "긴 형태로 나열하게 될 변수명",
  value = "key 변수에 대한 값 변수명",
  -var1, -var2, ... # 고려하지 않는 변수
)
```

▼ 예시 1

```
temp2_spread

## # A tibble: 2 x 16
##   user_gender      a      b      c      d      e      f      g      h      i      j      k
##   <chr>      <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 남성          43   127   33   65   44   98   146   50   48   45   24
## 2 여성         104    21    37    35    74    63   100    69    19    82    42
## # ... with 4 more variables: l <int>, m <int>, n <int>, o <int>

temp2_gather <- temp2_spread %>%
  gather(key = "song_id", value = "streaming_value", -user_gender)

temp2_gather

## # A tibble: 30 x 3
##   user_gender song_id streaming_value
##   <chr>      <chr>      <int>
## 1 남성      a              43
## 2 여성      a             104
## 3 남성      b              127
## 4 여성      b              21
## 5 남성      c              33
## 6 여성      c              37
## 7 남성      d              65
## 8 여성      d              35
## 9 남성      e              44
## 10 여성     e              74
## # ... with 20 more rows
```

▼ 예시 2

```
train %>%
  gather('X1':'X6', key='variable', value='value') %>%
  mutate_at('Y', as.factor) %>%
  ggplot()+
  geom_density(mapping=aes(x=value, color=Y, group=Y))+
  facet_wrap(vars(variable), ncol=3, scales="free")+
  theme_classic()+
  theme(legend.position = "bottom")+
  labs(x=NULL, y=NULL)
```

sales <dbl>	oil_price <dbl>	onpromotion <dbl>
690270.05	106.06	2171
719637.88	105.18	4270
560254.08	104.76	2224
574178.12	104.76	4017
783740.67	104.76	569
881285.27	104.76	540
613103.50	104.19	460
534441.34	104.06	2029
554522.61	102.93	4247
478136.71	103.61	2323

```
> head(data %>% gather)
```

```
   key    value
1 sales 690270.0
2 sales 719637.9
3 sales 560254.1
4 sales 574178.1
5 sales 783740.7
6 sales 881285.3
```