Sok-1005 assignment 4

44

```
rm(list=ls())
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.2.2

Warning: package 'ggplot2' was built under R version 4.2.2

Warning: package 'tidyr' was built under R version 4.2.2

Warning: package 'readr' was built under R version 4.2.2

Warning: package 'purrr' was built under R version 4.2.2

Warning: package 'dplyr' was built under R version 4.2.2

Warning: package 'stringr' was built under R version 4.2.2

Warning: package 'forcats' was built under R version 4.2.2

Warning: package 'lubridate' was built under R version 4.2.2

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.0     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.1     v tibble    3.1.8
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(lubridate)
library(quantmod)
```

```
Warning: package 'quantmod' was built under R version 4.2.2
```

```
Loading required package: xts
```

```
Warning: package 'xts' was built under R version 4.2.2
```

```
Loading required package: zoo
```

```
Warning: package 'zoo' was built under R version 4.2.2
```

```

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```

################################### WARNING ###################################
# We noticed you have dplyr installed. The dplyr lag() function breaks how    #
# base R's lag() function is supposed to work, which breaks lag(my_xts).      #
#                                                                             #
# Calls to lag(my_xts) that you enter or source() into this session won't     #
# work correctly.                                                             #
#                                                                             #
# All package code is unaffected because it is protected by the R namespace   #
# mechanism.                                                                  #
#                                                                             #
# Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
#                                                                             #
# You can use stats::lag() to make sure you're not using dplyr::lag(), or you #
# can add conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop   #
# dplyr from breaking base R's lag() function.                                #
################################### WARNING ###################################

Attaching package: 'xts'
```

```
The following objects are masked from 'package:dplyr':

    first, last

Loading required package: TTR

Warning: package 'TTR' was built under R version 4.2.2

Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

```r
library(janitor)
```

```

Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
library(plotly)
```

```
Warning: package 'plotly' was built under R version 4.2.3


Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

```r
library(knitr)
library(dplyr)
```

You can add options to executable code like this

```r
df <- read.csv("https://raw.githubusercontent.com/uit-sok-1005-v23/uit-sok-1005-v23.github
  clean_names()
```

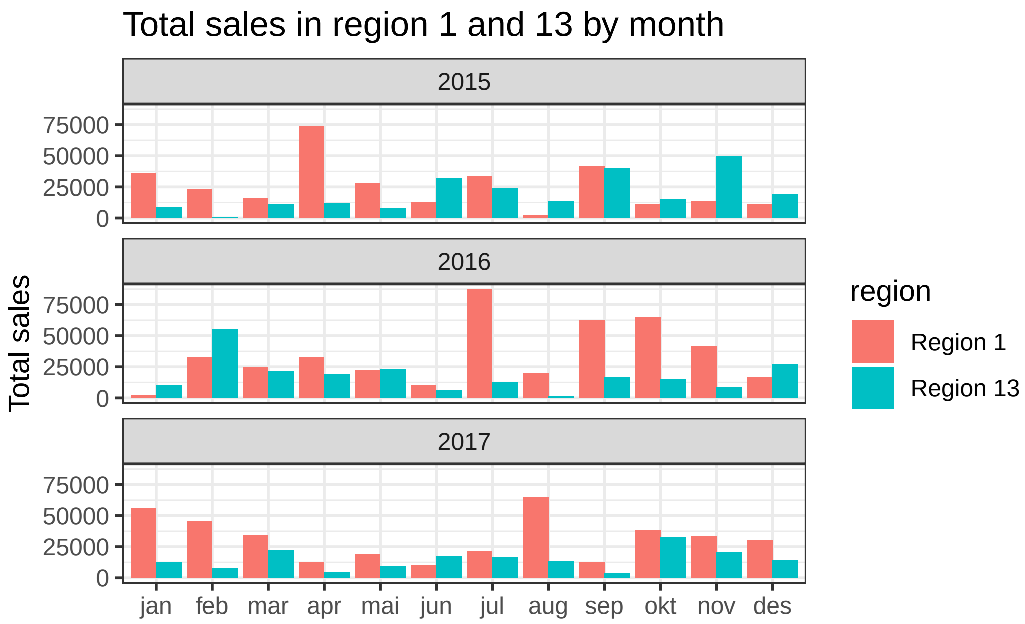
```r
table1 <- df %>%
  mutate(order_date = as.Date(order_date)) %>%
  mutate(year = year(order_date),
         month = month(order_date),
         day = day(order_date)) %>%
  filter(year=="2017",
         month >= 10,
         customer_segment %in% c("Corporate", "Consumer"),
         region %in% c("Region 1", "Region 9")) %>%
  group_by(region, month, customer_segment) %>%
  summarize(sales = sum(sales)) %>%
  rename("Region" = "region", "Month" = "month", "Customer segment" = "customer_segment")
```

```
`summarise()` has grouped output by 'region', 'month'. You can override using
the `.groups` argument.
```

```r
figure1 <- df %>%
  mutate(order_date = as.Date(order_date)) %>%
  mutate(year = year(order_date), month = month(order_date, label = TRUE),  day = day(orde
  filter(year %in% c("2015", "2016", "2017"),region %in% c("Region 1", "Region 13")) %>%
  group_by(region, month, year) %>%
  summarize(sales = sum(sales)) %>%
    mutate(date = make_date(year,month))
```

```
`summarise()` has grouped output by 'region', 'month'. You can override using
the `.groups` argument.
```

```r
fig1 <- figure1 %>%
  ggplot(aes(x=month, y=sales))+
  geom_col(aes(fill=region), position="dodge")+
  labs(x="", y="Total sales", title="Total sales in region 1 and 13 by month", color="Regi
  facet_wrap(~year, nrow=3) +
  theme_bw()

fig1
```



```
table2 <- figure1[c(41,50,52,54,61,67,71), ]

table2
```

```
# A tibble: 7 x 5
# Groups:   region, month [6]
  region    month  year  sales date      
  <chr>     <ord> <dbl>  <dbl> <date>    
1 Region 13 feb    2016 55632. 2016-02-01
2 Region 13 mai    2016 22822. 2016-05-01
3 Region 13 jun    2015 32307. 2015-06-01
4 Region 13 jun    2017 17430. 2017-06-01
5 Region 13 sep    2015 39825. 2015-09-01
6 Region 13 nov    2015 49686. 2015-11-01
7 Region 13 des    2016 26890. 2016-12-01
```

```
table3 <- df %>%
    mutate(year = year(order_date), month = month(order_date, label = TRUE),  day = day(or
  filter(region %in% c("Region 1", "Region 2", "Region 4", "Region 6", "Region 7", "Region
filter(year > 2016)

table3 <- table3[c(5,7:10)]

table3 <- table3 %>%
  group_by(customer_segment, product_category, region, year) %>%
    summarize(avg_profit = mean(profit))
```

```
`summarise()` has grouped output by 'customer_segment', 'product_category',
'region'. You can override using the `.groups` argument.
```

```
#Small Business within Technology in Region 11
#was the customer segment with the highest average
#profit for 2017 with 3585.120000.
```

```
xom <- data.frame(getSymbols("XOM", src = "yahoo", from = "2010-1-04", to = "2022-12-31",
xom <- tibble::rownames_to_column(xom, var = "Date")
```

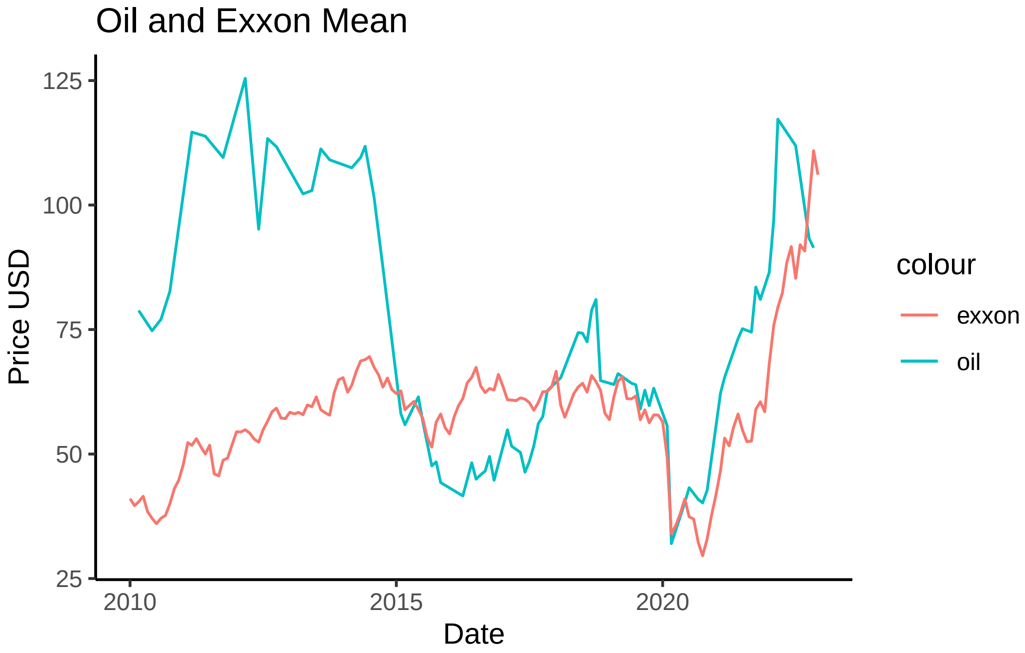
```
crudeoil <- data.frame(getSymbols("DCOILBRENTEU", src = "FRED", from = "2010-1-04", to = "

crudeoil <- tibble::rownames_to_column(crudeoil, var = "Date")
```

```
xom_ <- xom %>%
  mutate(Date = as.Date(Date)) %>%
  mutate(year_month = format(Date, "%Y-%m")) %>%
  group_by(year_month) %>%
  mutate(exxon = weighted.mean(XOM.Adjusted))

crudeoil_ <- crudeoil %>%
  mutate(Date = as.Date(Date)) %>%
  mutate(year_month = format(Date, "%Y-%m")) %>%
  group_by(year_month) %>%
  mutate(oil = mean(DCOILBRENTEU))

crudeoil_ <- na.omit(crudeoil_)
crudeoil_ <- crudeoil_ %>%
  filter(Date >= "2010-01-04" & Date < "2022-12-01")
```

```
xom_ <- xom_ %>%
  select(year_month,exxon) %>%
  mutate(year_month = ym(year_month)) %>%
  distinct()
```

```
crudeoil_ <- crudeoil_ %>%
  select(year_month,oil) %>%
  mutate(year_month = ym(year_month)) %>%
  distinct()
```

```
ggplot() +
  geom_line(data = crudeoil_ , aes(x=year_month,y=oil, col = "oil")) +
  geom_line(data = xom_,aes(x=year_month,y=exxon,col="exxon"))+   xlab("Date") + ylab("Pri
theme_classic()
```



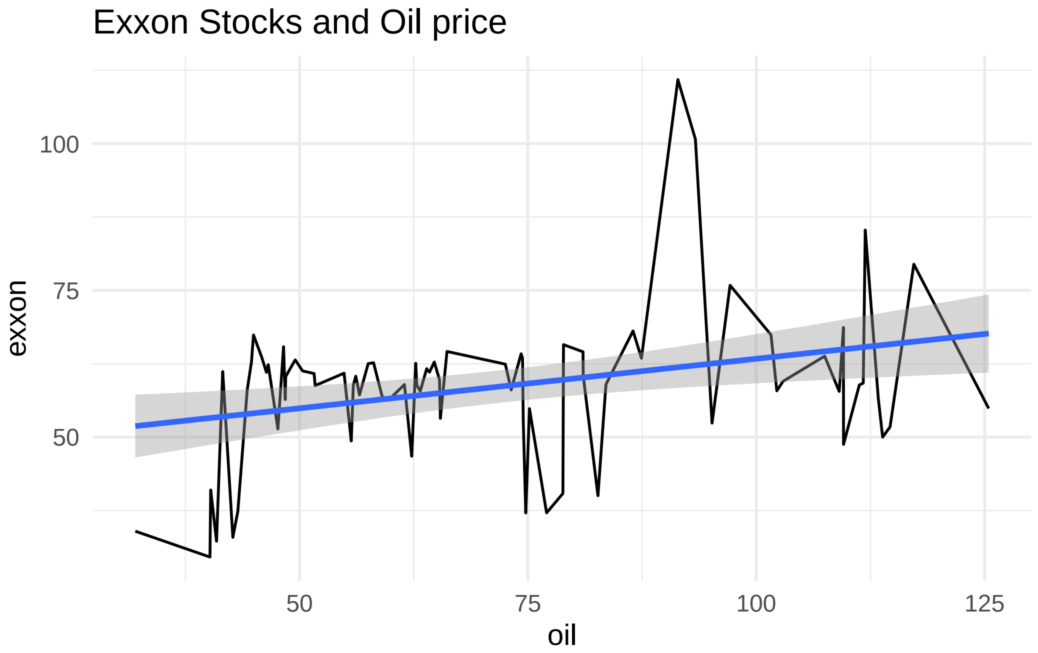
```r
#the red line shows us the mean by weight of exxon stocks.
#The blue line shows us the mean of oil prices.
#We obsere a growth of exxon stock prices that seem
#to follow the rising price of oil.
```

```r
oil_exxon <- merge(xom_,crudeoil_)

lm(exxon ~ oil, data = oil_exxon)
```

```

Call:
lm(formula = exxon ~ oil, data = oil_exxon)

Coefficients:
(Intercept)          oil
    46.4749       0.1688
```

```r
oil_exxon %>%
  ggplot(aes(x=oil,y=exxon)) +
  geom_line() +
  theme_minimal() +
  xlab("oil") +
  ylab("exxon") +
  ggtitle("Exxon Stocks and Oil price") +
  geom_smooth(method = lm)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



```
# (Intercept)          oil
   # 46.4749       0.1688

# the intercept tells us the average price of the stocks, and the other coefficient tells

#us how much more value is added if the oil prices were to be raised by one.

#We can tell there is a clear corrolation between oil prices

#and the prices of exxon stocks.
```