

M46-retail2

LearningSpoonsR

2019-02-11

Background

I. Background

- ▶ Continued from M43-retail1
 1. 배송 기간 분석 (Ship Date를 기반으로 배송이 가장 오래걸리는 상품은 무엇인가?)
 2. 마진이 가장 많이 남는 상품은 무엇인가?
 3. M46에서는 소비자 별 구매 기록을 분석해보자
- ▶ 맥주와 기저귀
 - ▶ Data-driven marketing의 고전적 예제
 - ▶ 맥주와 기저귀가 동시에 많이 팔린다.
 - ▶ Why? 아이를 위해 기저귀를, 나를 위해 맥주를 사는 아이 아빠들...
 - ▶ So what? 두 상품의 진열을 가깝게? 아니면 멀게해서 동선을 확보? 묶음 상품으로?
- ▶ Online Commerce?
 - ▶ “이 상품을 구매한 사람이 자주 본 물건”
 - ▶ “You may also like...”

Goal

II. Goal

목표 결과물 (Target output)

- ▶ 이 노트는 제작 과정을 최대한 재현한 노트입니다. 분석가의 생각의 흐름을 느껴보세요.
- ▶ From M24,

0. About

1. Correlation: 두 변수가 얼마나 상관관계가 있는가?

2. Deviation

3. Ranking

4. Distribution

5. Composition

6. Change

7. Groups

M24-ggplot2 Gallery

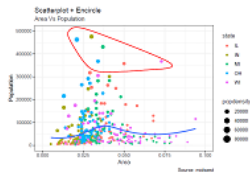
Learning Spoons 2019-01-20

Contents

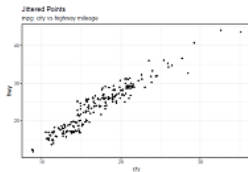
0. About	2
종속적인 차등한 여왕 과일바구니	2
1. Correlation	3
Scatterplot	3
Scatterplot With Encoding	5
Jitter Plot	7
Counts Chart	9
Bubble plot	10
Marginal Histogram / Boxplot	11
2. Deviation	13
Diverging Bars	13
Diverging Lollipop Chart	15
Diverging Dot Plot	16
Area Chart	17
3. Ranking	19
Ordered Bar Chart	19
Lollipop Chart	21
Dot Plot	22
Shape Chart	23
Dumbbell Plot	25
4. Distribution	27
Histogram	27
Histogram on a continuous variable	27
Histogram on a categorical variable	29
Density plot	30
Box Plot	31
Dot + Box Plot	33
Tufte Boxplot	34
Violin Plot	35
Population Pyramid	36
5. Composition	37
Waffle Chart	37
Pie Chart	39
Bar Chart	43
6. Change	45
Time Series Plot From a Time Series Object ts()	45
Time Series Plot From a Data Frame	46
Default X Axis Labels	47
Time Series Plot For a Monthly Time Series	47
Time Series Plot For a Yearly Time Series	49
Time Series Plot From Long Data Format: Multiple Time Series in Same DataFrame Columns	51
Time Series Plot From Wide Data Format: Data in Multiple Columns of DataFrame	55
Stacked Area Chart	57
Calendar Heatmap	59
Seasonal Plot	61
7. Groups	61
Hierarchical Dendrogram	61
Clusters	63

목표 결과물 (Target output)

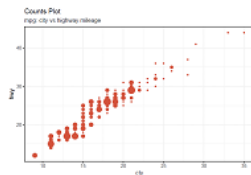
► From M24. 1. Correlation,



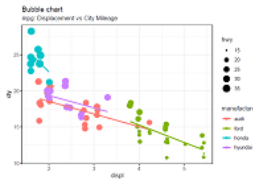
Scatterplot



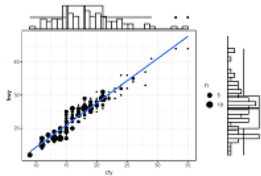
Jittered Plot



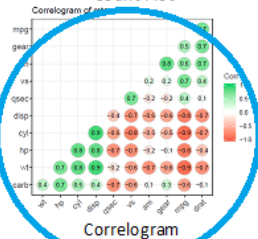
Count Plot



Bubble Chart



+ Histogram



Correlogram

필요한 데이터셋의 구조

We need...

	배기량	기어비	노력	가속
배기량	1000	580	31	3
기어비		1500	10	2
노력			5000	3
가속				570

Figure 1: 필요한 데이터 구조

for...

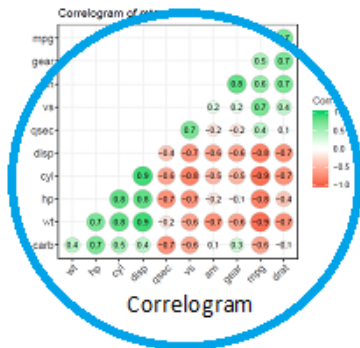


Figure 2: 최종 output 시안

Data Review

III. Data Review

```
library(tidyverse) # dplyr + ggplot2 + tidyr + ...

## -- Attaching packages ----- tidyverse 1.2.1 --
## v ggplot2 3.1.0      v purrr   0.3.0
## v tibble  2.0.1      v dplyr  0.7.8
## v tidyr   0.8.2      v stringr 1.3.1
## v readr   1.3.1      v forcats 0.3.0
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
library(readxl)
dataset <- read_excel("retail.xlsx")
```

- ▶ **tidyverse** 많이도 포함하고 있네요.
- ▶ **rmarkdown** 문서에 r chunk의 아웃풋 출력을 안하려면 **message=FALSE**하면 됩니다.

이렇게 생긴 데이터이고...

`str(dataset)`

```
## Classes 'tbl_df', 'tbl' and 'data.frame':   9994 obs. of  20 variables:
## $ Country      : chr  "United States" "United States" "United States" "United States"
## $ Region       : chr  "East" "East" "South" "South" ...
## $ State        : chr  "Ohio" "Ohio" "Virginia" "Virginia" ...
## $ City         : chr  "Akron" "Akron" "Alexandria" "Alexandria" ...
## $ Postal Code  : num  44312 44312 22304 22304 22304 ...
## $ Category     : chr  "Furniture" "Furniture" "Furniture" "Furniture" ...
## $ Sub-Category : chr  "Tables" "Furnishings" "Furnishings" "Furnishings" ...
## $ Segment      : chr  "Corporate" "Consumer" "Corporate" "Home Office" ...
## $ Product Name : chr  "Chromcraft Rectangular Conference Tables" "Deflect-o Glass Clear
## $ Manufacturer : chr  "Chromcraft" "Deflect-o" "DAX" "Eldon" ...
## $ Customer Name: chr  "Ed Braxton" "Ted Trevino" "Andrew Gjertsen" "Shirley Daniels" .
## $ Order Date   : POSIXct, format: "2014-10-21" "2011-05-18" ...
## $ Order ID     : chr  "CA-2014-147277" "CA-2011-164224" "CA-2012-104241" "US-2011-1555
## $ Ship Date    : POSIXct, format: "2014-10-25" "2011-05-20" ...
## $ Ship Mode    : chr  "Standard Class" "Second Class" "Standard Class" "Standard Class"
## $ Discount     : num  0.4 0.2 0 0 0 0.3 0.2 0.32 0.3 0.6 ...
## $ Profit       : num  -76 4 69 4 31 -31 3 -36 -350 -11 ...
## $ Profit Ratio : num  -0.27 0.03 0.36 0.36 0.49 -0.13 0.29 -0.18 -0.14 -0.48 ...
## $ Quantity     : num  2 3 14 3 3 2 2 2 5 3 ...
## $ Sales        : num  284 149 192 12 63 ...
```

이런 변수들이 있구나...

```
colnames(dataset)
```

```
## [1] "Country"      "Region"      "State"      "City"
## [5] "Postal Code"  "Category"    "Sub-Category" "Segment"
## [9] "Product Name" "Manufacturer" "Customer Name" "Order Date"
## [13] "Order ID"     "Ship Date"   "Ship Mode"   "Discount"
## [17] "Profit"       "Profit Ratio" "Quantity"    "Sales"
```

관련 데이터는...

```
dataset %>% select(`Product Name`, `Customer Name`) %>% head()
```

```
## # A tibble: 6 x 2
##   `Product Name`      `Customer Name`
##   <chr>              <chr>
## 1 Chromcraft Rectangular Conference Tables Ed Braxton
## 2 Deflect-o Glass Clear Studded Chair Mats Ted Trevino
## 3 DAX Wood Document Frame Andrew Gjertsen
## 4 Eldon Image Series Black Desk Accessories Shirley Daniels
## 5 GE General Use Halogen Bulbs, 100 Watts, 1 Bulb per Pack Shirley Daniels
## 6 Chromcraft Round Conference Tables Anna Gayman
```

각각 컬럼의 **unique**한 value의 갯수는...

```
dataset %>% select(`Product Name`, `Customer Name`) %>%  
  sapply(function(x) length(unique(x)))
```

```
## Product Name Customer Name  
##           1850           793
```

암튼 **dataset**을 이용해서 Figure 1 모양의 구조를 만드는 preprocessing을 하면 되겠구나!

Preprocessing

Preprocessing 1: Assign IDs

Issue → Solution → Discussion → Implementation

Issue

1. **Customer Name**의 유일성 (uniqueness)
 - ▶ Akron에 사는 Barrack Obama → 기저귀 구매
 - ▶ L.A.에 사는 Barrack Obama → 맥주 구매
 - ▶ 그런데 둘은 다른 사람!
2. **Product Name**의 유일성과 value의 길이
 - ▶ 마찬가지로 유일성이 담보되지 않음
 - ▶ 값이 너무 김
 - ▶ ex) Chromcraft Rectangular Conference Tables

Discussion

1. ID variable(Key variable)은 데이터를 체계적으로 관리하는 도구
2. 없다면 유일성을 담보할 수 있는 방법으로 만들어 내야 함
3. 체계적으로 준비된 데이터 셋은 join 및 조회가 용이함

Solution

1. Customer Name의 유일성 (uniqueness)

- ▶ 각각 다른 CustomerID가 있어야 함
- ▶ Postal Code(44312)와 Customer Name (Ed Braxton)으로 CustomerID를 만들자.
- ▶ cus01321와 같은 형식으로 만들자.

2. Product Name의 유일성과 value의 길이

- ▶ 각각 다른 ProductID가 있어야 함.
- ▶ Manufacturer와 Product Name으로 ProductID를 만들자.
- ▶ prod08231와 같은 형식으로 만들자.

Implementation

Step 1. `unique`를 사용해서 유일한 조합을 생성하여 `customerTable`에 저장

```
customerTable <- dataset %>% select(`Postal Code`, `Customer Name`) %>% unique()  
customerTable %>% head()
```

```
## # A tibble: 6 x 2  
##   `Postal Code` `Customer Name`  
##         <dbl> <chr>  
## 1      44312 Ed Braxton  
## 2      44312 Ted Trevino  
## 3      22304 Andrew Gjertsen  
## 4      22304 Shirley Daniels  
## 5      75002 Anna Gayman  
## 6      18103 Caroline Jumper
```

Step 1-1. 구매기록에 몇명의 고객이 있는지 확인

```
dim(customerTable)  
## [1] 4910    2
```

Step 1-2. CustomerID 변수를 만들기

1. 일련 번호 생성
2. google "how to add leading zeros R"

```
library(stringr)
x <- 1:nrow(customerTable) %>% str_pad(5, pad = "0") # make 5-digit string by padding "0"
head(x, 3)

## [1] "00001" "00002" "00003"
```

3. CustomerID 변수 완성

```
customerTable$CustomerID <- paste0("cus", x)
customerTable %>% head(3)

## # A tibble: 3 x 3
##   `Postal Code` `Customer Name` CustomerID
##         <dbl> <chr>          <chr>
## 1      44312 Ed Braxton      cus00001
## 2      44312 Ted Trevino     cus00002
## 3      22304 Andrew Gjertsen cus00003
```

Step 2. 같은 방법으로 productTable 만들기

```
productTable <- dataset %>% select(Manufacturer, `Product Name`) %>% unique()
productTable <- productTable %>%
  mutate(ProductID = paste0("prod",
                             1:nrow(productTable) %>% str_pad(5, pad = "0")))
productTable %>% head(3)
```

```
## # A tibble: 3 x 3
##   Manufacturer `Product Name`      ProductID
##   <chr>        <chr>            <chr>
## 1 Chromcraft  Chromcraft Rectangular Conference Tables prod00001
## 2 Deflect-o   Deflect-o Glass Clear Studded Chair Mats prod00002
## 3 DAX         DAX Wood Document Frame          prod00003
```

Step 3. 원래 데이터셋에 CustomerID와 ProductID를 부여

Step 3-1. CustomerID 부여

```
head(dataset[,c(5, 11:12)], 2)
```

```
## # A tibble: 2 x 3
##   `Postal Code` `Customer Name` `Order Date`
##         <dbl> <chr>          <dtm>
## 1      44312 Ed Braxton      2014-10-21 00:00:00
## 2      44312 Ted Trevino     2011-05-18 00:00:00
```

```
head(customerTable, 2)
```

```
## # A tibble: 2 x 3
##   `Postal Code` `Customer Name` CustomerID
##         <dbl> <chr>          <chr>
## 1      44312 Ed Braxton      cus00001
## 2      44312 Ted Trevino     cus00002
```

```
dataset <- left_join(x = dataset, y = customerTable,
                     by = c("Postal Code", "Customer Name"))
dataset[,c(5, 11:12, 21)] %>% head(2)
```

```
## # A tibble: 2 x 4
##   `Postal Code` `Customer Name` `Order Date`      CustomerID
##         <dbl> <chr>          <dtm>          <chr>
## 1      44312 Ed Braxton      2014-10-21 00:00:00 cus00001
## 2      44312 Ted Trevino     2011-05-18 00:00:00 cus00002
```

성공!

Step 3-2. 같은 방법으로 ProductID 부여

```
dataset <- left_join(x = dataset, y = productTable,
                     by = c("Manufacturer", "Product Name"))
```

Step 4. 전처리 완료! 필요한 변수만 모아서 dataset2라고 이름 붙임!

```
dataset2 <- dataset %>%
  select(CustomerID, ProductID, `Postal Code`,
         `Customer Name`, Manufacturer, `Product Name`) %>%
  arrange(CustomerID, ProductID)
dataset2[108:110,]

## # A tibble: 3 x 6
##   CustomerID ProductID `Postal Code` `Customer Name` Manufacturer
##   <chr>      <chr>      <dbl> <chr>          <chr>
## 1 cus00038   prod00047      60505 Barry Franzosi~ Global
## 2 cus00038   prod00543      60505 Barry Franzosi~ Acco
## 3 cus00038   prod00550      60505 Barry Franzosi~ TOPS
## # ... with 1 more variable: `Product Name` <chr>
```


Preprocessing 2

Strategy

Customer ID	Product ID	Customer ID	Purchase Records
Cus0001	Prod0001	Cus0001	Prod0001, Prod0005, ...
Cus0001	Prod0005	Cus0002	" 0001, " 0008, " 0003
" 0002	" 0007		
" 0002	" 0008		
" 0002	" 0003		

Step 2

	Prod0001	Prod0002	Prod0003	...
Prod0001	27	3	20	
Prod0002				
⋮				
⋮				
⋮				

Step 2

Implementation – Step 1

Before

```
dataset2 %>%  
  select(CustomerID, ProductID) %>%  
  head()
```

```
## # A tibble: 6 x 2  
##   CustomerID ProductID  
##   <chr>      <chr>  
## 1 cus00001   prod00001  
## 2 cus00001   prod00384  
## 3 cus00002   prod00002  
## 4 cus00002   prod00396  
## 5 cus00003   prod00003  
## 6 cus00004   prod00004
```

After

```
purchase <- dataset2 %>%  
  group_by(CustomerID) %>%  
  summarise(  
    Purchases =  
      paste(ProductID, collapse = " ")  
  )  
head(purchase)
```

```
## # A tibble: 6 x 2  
##   CustomerID Purchases  
##   <chr>      <chr>  
## 1 cus00001   prod00001 prod00384  
## 2 cus00002   prod00002 prod00396  
## 3 cus00003   prod00003  
## 4 cus00004   prod00004 prod00005 prod00411  
## 5 cus00005   prod00006 prod00415  
## 6 cus00006   prod00007 prod00417
```

Implementation – Step 2

Prepare an empty matrix,
purchaseCount

```
uProductID <- unique(dataset2$ProductID)
purchaseCount <-
  array(0, c(length(uProductID), length(uProductID)),
        colnames(purchaseCount) <- uProductID
        rownames(purchaseCount) <- uProductID
        purchaseCount[1:4, 1:3])
```

```
##           prod00001 prod00384 prod00002
## prod00001           0           0           0
## prod00384           0           0           0
## prod00002           0           0           0
## prod00396           0           0           0
```

And with **purchase\$Purchase**

```
purchase$Purchases %>% head(3) %>% t() %>% t()
##           [,1]
## [1,] "prod00001 prod00384"
## [2,] "prod00002 prod00396"
## [3,] "prod00003"
```

Fill purchaseCount

```
library(stringr)
a <- Sys.time()
for (i in 1:nrow(purchaseCount)) {
  for (j in 1:nrow(purchaseCount)) {
    purchaseCount[i,j] <-
      sum(
        str_detect(purchase$Purchases, uProductID[i]) &
        str_detect(purchase$Purchases, uProductID[j]))
  }
}
b <- Sys.time()
save.image(file = "M46_middle.Rdata")
library(beepr)
beep()
```

```
purchaseCount[1:4,1:3]
```

```
##           prod00001 prod00384 prod00002
## prod00001         9         1         0
## prod00384         1         5         0
## prod00002         0         0         7
## prod00396         0         0         1
```

```
a
## [1] "2019-01-21 15:16:40 KST"
b
## [1] "2019-01-21 18:16:16 KST"
b-a
## Time difference of 2.993472 hours
```

▶ Code Review

- ▶ 시간이 오래걸린 작업이므로 `save.image(file = "M46_middle.Rdata")`로 현재 메모리 상태 저장
- ▶ `load("M46_middle.Rdata")`를 이용하면 저장된 메모리 상태 불러올 수 있음
- ▶ 소요시간 확인을 위해서 `a`와 `b`를 기록
- ▶ `library(beepR)`, `beep()`으로 소리를 내어서 작업 완료를 알려줌 (기다리면서는 무엇을 하나요? 콧득)

▶ Discussion

- ▶ 속도를 빠르게 하려면 어떻게 해야할까요?
- ▶ Hint?

Results

Result 1

Step. 1. 각 ProductID별로 pair 구매 건수를 집계하고 정렬

- ▶ `purchaseCount`의 각 column의 합을 구하고...
- ▶ 해당 제품 구매 건수를 빼주면 pair 구매 건수가 집계됨
- ▶ 그것을 내림차순으로 정렬하면 `pairCount`완성

```
pairCount <- colSums(purchaseCount) - diag(purchaseCount)
pairCount <- sort(pairCount, decreasing = TRUE)
head(pairCount)
```

```
## prod00579 prod00445 prod00423 prod00116 prod00156 prod00018
##          104          90          80          48          46          43
```


Step. 2. pair 구매가 많았던 8개 ProductID만 골라서 8 by 8 표로 표현

- ▶ `mat_size`를 8로 지정
- ▶ `pairCount[1:mat_size]`의 `names`들이 `productID` 형식으로 표현되어 있으므로...
- ▶ `names()`를 이용해서 `purchaseCount`를 subsetting함!!

```
mat_size <- 8
pair_mat <- purchaseCount[names(pairCount[1:mat_size]),
                           names(pairCount[1:mat_size])]
pair_mat %>% head()
```

```
##           prod00579 prod00445 prod00423 prod00116 prod00156 prod00018
## prod00579          46         0         1         2         0         0
## prod00445          0         48         1         1         0         0
## prod00423          1         1        46         0         0         0
## prod00116          2         1         0        18         0         0
## prod00156          0         0         0         0        14         1
## prod00018          0         0         0         0         1        14
##           prod00469 prod00476
## prod00579          0         0
## prod00445          0         0
## prod00423          0         1
## prod00116          0         0
## prod00156          1         0
## prod00018          0         0
```

Figure 1 완성!!!

- ▶ `pair_mat`의 index들을 `productID`형식에서 `productName`형식으로 바꾸려면?
- ▶ 먼저 `productTable`을 이용해서 `pair_mat`의 ID를 매치시킴

```
pair_mat_name <-  
  left_join(data.frame(ProductID = colnames(pair_mat)), productTable)  
  
## Joining, by = "ProductID"  
## Warning: Column `ProductID` joining factor and character vector, coercing  
## into character vector  
pair_mat_name %>% head()  
  
##   ProductID      Manufacturer  
## 1 prod00579          Other  
## 2 prod00445  Staple envelope  
## 3 prod00423 Easy-staple paper  
## 4 prod00116             KI  
## 5 prod00156          Eldon  
## 6 prod00018          Global  
##  
##               Product Name  
## 1                   Staples  
## 2          Staple envelope  
## 3          Easy-staple paper  
## 4  KI Adjustable-Height Table  
## 5  Eldon Wave Desk Accessories  
## 6 Global Wood Trimmed Manager's Task Chair, Khaki
```

▶ pair_mat의 rownames를 교체

```
rownames(pair_mat) <- pair_mat_name[, "Product Name"] %>% substr(1,20)
pair_mat
```

##	prod00579	prod00445	prod00423	prod00116	prod00156
## Staples	46	0	1	2	0
## Staple envelope	0	48	1	1	0
## Easy-staple paper	1	1	46	0	0
## KI Adjustable-Height	2	1	0	18	0
## Eldon Wave Desk Acce	0	0	0	0	14
## Global Wood Trimmed	0	0	0	0	1
## Hot File 7-Pocket, F	0	0	0	0	1
## Storex Dura Pro Bind	0	0	1	0	0
##	prod00018	prod00469	prod00476		
## Staples	0	0	0		
## Staple envelope	0	0	0		
## Easy-staple paper	0	0	1		
## KI Adjustable-Height	0	0	0		
## Eldon Wave Desk Acce	1	1	0		
## Global Wood Trimmed	14	0	0		
## Hot File 7-Pocket, F	0	13	0		
## Storex Dura Pro Bind	0	0	17		

진짜로 Figure 1 완성!!!

Step. 3. pair 구매가 2회 이상이었던 10개 case를 random하게 출력

```
topPair_N <- 10
# Replace diagonals to 0
purchaseCount_offdiag <- purchaseCount
diag(purchaseCount_offdiag) <- 0
# Find >=2 counts and build data.frame
top_indexes <-
  which(purchaseCount_offdiag >= 2, arr.ind = TRUE) %>% data.frame()
top_indexes %>% head()

##           row col
## prod00543 106  27
## prod00394 212  33
## prod01458  41  35
## prod00270 694  39
## prod00435  35  41
## prod01466  68  64
colnames(purchaseCount_offdiag) %>% head()

## [1] "prod00001" "prod00384" "prod00002" "prod00396" "prod00003" "prod00004"
```

```
Top_N <- data.frame(  
  ID_x = apply(top_indexes, 1,  
               function(x) colnames(purchaseCount_offdiag)[x[1]]),  
  ID_y = apply(top_indexes, 1,  
               function(x) colnames(purchaseCount_offdiag)[x[2]]),  
  count = apply(top_indexes, 1,  
                function(x) purchaseCount_offdiag[x[1], x[2]]),  
  stringsAsFactors = FALSE,  
  row.names = NULL  
)  
# replace ID with `ProductID` using `left_join`  
Top_N <- Top_N %>%  
  left_join(productTable, by = c("ID_x"="ProductID")) %>%  
  select(`Product Name`, ID_y, count) %>%  
  left_join(productTable, by = c("ID_y"="ProductID")) %>%  
  select(`Product Name.x`, `Product Name.y`, count)
```

```
Top_N[,1] <- substr(Top_N[,1], 1, 30)
Top_N[,2] <- substr(Top_N[,2], 1, 30)
Top_N %>% head(2)
```

```
##               Product Name.x               Product Name.y count
## 1      Acco Hot Clips Clips to Go      Flat Face Poster Frame      2
## 2 Wilson Jones Active Use Binder GBC Recycled VeloBinder Covers      2
```

Top_N에서 random하게 topPair_N개 행을 뽑으면 끝!

```
Top_N[sample(1:nrow(Top_N), topPair_N),]
```

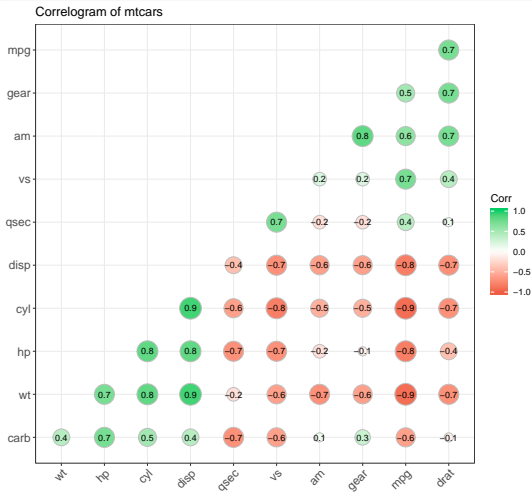
```
##               Product Name.x               Product Name.y count
## 62 Avery Fluorescent Highlighter Acco Side-Punched Conventional      2
## 42 Peel & Stick Add-On Corner Poc      Easy-staple paper      2
## 31              Newell 34              Staple envelope      2
## 32              Xerox 1971              Staple envelope      2
## 53 Carina Double Wide Media Stora 1.7 Cubic Foot Compact "Cube"      2
## 9  DMI Eclipse Executive Suite Bo              Avery 485      2
## 55 1.7 Cubic Foot Compact "Cube" Carina Double Wide Media Stora      2
## 48 Belkin Premiere Surge Master I SAFCO Optional Arm Kit for Wor      2
## 47              Xerox 192 X-Rack File for Hanging Folder      2
## 35              OIC Binder Clips      Eldon Wave Desk Accessories      2
```

Result 2

Step. 1. 예제 코드를 이용해 Correlogram 그리기 연습

```
library(ggplot2)
library(ggcorrplot)
data(mtcars)
corr <- round(cor(mtcars), 1)
fig_sample <-
  ggcorrplot(
    corr, hc.order = TRUE, type = "lower", lab = TRUE,
    lab_size = 3, method="circle",
    colors = c("tomato2", "white", "springgreen3"),
    title="Correlogram of mtcars", ggtheme=theme_bw)
```

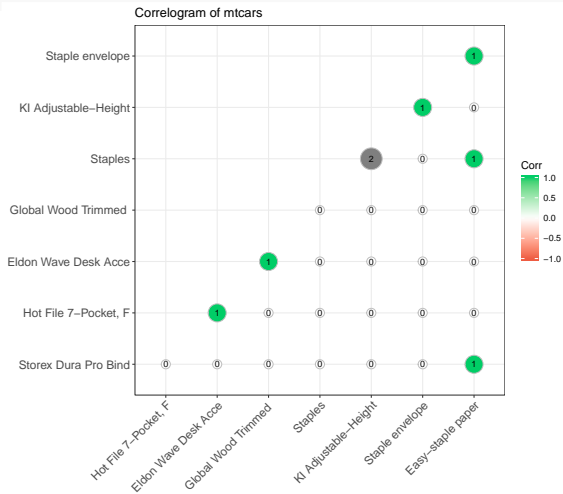
fig_sample



Step. 2. pair_mat를 이용해 correlogram 그리기

```
colnames(pair_mat) <- rownames(pair_mat)
fig_corr <-
  ggcorrplot(
    pair_mat, hc.order = TRUE, type = "lower", lab = TRUE,
    lab_size = 3, method="circle",
    colors = c("tomato2", "white", "springgreen3"),
    title="Correlogram of mtcars", ggtheme=theme_bw)
```

fig_corr



Summary

Summary

```
save.image("M46_final.Rdata")
```

▶ Result1

▶ Step. 2.

- ▶ `pair_mat`

- ▶ pair 구매가 많았던 8개 `ProductID`만 골라서 8 by 8 표로 표현

▶ Step. 3.

- ▶ `Top_N`

- ▶ pair 구매가 2회 이상이었던 10개 case를 random하게 출력

▶ Result2

▶ Step. 2.

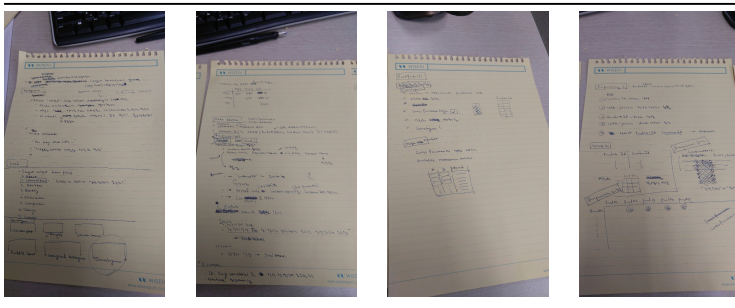
- ▶ `fig_corr`

- ▶ `pair_mat`를 이용해 correlogram 그리기

Shiny Presentation

1. `load("M46_final.Rdata")`를 이용해서 데이터를 불러옵니다.
2. 위의 8과 10을 사용자의 인풋으로 받아서 다음 작업을 수행합니다.
3. Correlogram
 - ▶ `input$mat_size`를 `seq(from = 8, to = 14, by = 2)`에서 선택
 - ▶ `pair_mat` 만들기 (from `purchaseCount` and `productTable`)
 - ▶ `fig_corr` 만들기 (from `pair_mat`)
4. Top_N table
 - ▶ `input$topPair_N`를 `seq(from = 10, to = 30, by = 5)`에서 선택
 - ▶ Top_N 만들기 (from `purchaseCount` and `productTable`)
5. flexdashboard + shiny로 다음을 포함한 대시보드를 만들어 보세요.
 - ▶ 사용자 인풋창
 - ▶ 3번 결과물
 - ▶ 4번 결과물
6. 결과물
 - ▶ <https://learningspoonsr.shinyapps.io/M46-retail2-correl/>

Sketch Notes



- ▶ Verumtamen oportet me bodie et cras et sequenti die ambulare.
- ▶ 오늘도 내일도 그 다음날도 계속해서 내 길을 가야 한다.

blank