

Modul 04 - Business Analytics Process and Data Exploration

Roni Yunis

3/28/2022

Pengantar

Dalam pembahasan kali ini, kita akan membahas secara umum proses analisis bisnis dan kaitannya dengan ekplorasi data. Tujuan dari analitika bisnis adalah untuk mendapatkan informasi dari data sehingga dapat membuat keputusan bisnis yang tepat. Dalam proses analisis bisnis ada beberapa tahapan yang harus dilalui yaitu: 1. Memahami masalah bisnis 2. Mengumpulkan data dan mengintegrasikan data 3. Pra Proses data 4. Ekplorasi dan visualisasi data 5. Menentukan teknik pemodelan atau algoritma 6. Evaluasi model 7. Laporkan hasilnya kepada pihak manajemen 8. Kembangkan model

Dari 8 tahapan tersebut, tahapan yang sangat penting dan berpengaruh pada hasil pengembangan model keputusan adalah tahap **Exploratory Data Analysis (EDA)**. EDA adalah proses ekplorasi data yang bertujuan untuk memahami isi dan komponen penyusun data. Biasanya EDA dilakukan dengan beberapa cara; *analisis deskriptif dengan satu variabel*, *analisis relasi dengan dua variabel*, dan *analisis dengan menggunakan lebih dari atau sama dengan tiga variabel*.

Exploratory Data Analysis (EDA)

Dalam EDA, secara sederhana ada 4 aktivitas yang akan dilakukan, yaitu: menyiapkan data, membersihkan data, Ekplorasi data, dan visualisasi data. Sebelum kita memulai 4 tahapan tersebut, ada beberapa library yang kita perlukan, yaitu `dplyr`, `lubridate` dan `ggplot2`

```
# library yang digunakan untuk data wrangling
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
```

```
## Attaching package: 'dplyr'
```

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

```
##
```

```
## filter, lag
```

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

```
##
```

```
## intersect, setdiff, setequal, union
```

```
# library untuk visualisasi data
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
# library untuk berkerja dengan date
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

Data Preparation

Kita akan import dataset, dataset yang kita gunakan adalah **online_retail.csv**. Kita akan simpan data online retail tersebut kedalam sebuah objek *retail*

```
retail <- read.csv("data/online_retail.csv")
```

Kita akan melihat struktur data dari objek retail dengan fungsi `str()` atau menggunakan fungsi `glimpse()`, fungsi ini merupakan fungsi yang ada pada packages `dplyr` baru kita panggil, dan fungsinya adalah untuk melihat struktur data.

```
# melihat strukturdata dengan glimpse()
glimpse(retail)
```

```
## Rows: 234,920
## Columns: 8
## $ InvoiceNo    <int> 536381, 536381, 536381, 536381, 536381, 536381, 536381, 53~
## $ StockCode   <chr> "71270", "22262", "22637", "21166", "37444A", "37444C", "2~
## $ Description <chr> "PHOTO CLIP LINE", "FELT EGG COSY CHICKEN", "PIGGY BANK RE~
## $ Quantity    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ InvoiceDate  <chr> "12/1/10 9:41", "12/1/10 9:41", "12/1/10 9:41", "12/1/10 9~
## $ UnitPrice   <dbl> 1.25, 0.85, 2.55, 1.95, 2.95, 2.95, 2.95, 0.85, 0.85, 1.45~
## $ CustomerID  <int> 15311, 15311, 15311, 15311, 15311, 15311, 15311, 15311, 15~
## $ Country     <chr> "United Kingdom", "United Kingdom", "United Kingdom", "Uni~
```

fungsi dan hasilnya hampir sama dengan fungsi `str()`

```
str(retail)
```

```
## 'data.frame':   234920 obs. of  8 variables:
## $ InvoiceNo    : int  536381 536381 536381 536381 536381 536381 536381 536381 536381 536381 ...
## $ StockCode    : chr   "71270" "22262" "22637" "21166" ...
```

```
## $ Description: chr "PHOTO CLIP LINE" "FELT EGG COSY CHICKEN" "PIGGY BANK RETROSPOT " "COOK WITH WI
## $ Quantity : int 1 1 1 1 1 1 1 1 1 1 ...
## $ InvoiceDate: chr "12/1/10 9:41" "12/1/10 9:41" "12/1/10 9:41" "12/1/10 9:41" ...
## $ UnitPrice : num 1.25 0.85 2.55 1.95 2.95 2.95 2.95 0.85 0.85 1.45 ...
## $ CustomerID : int 15311 15311 15311 15311 15311 15311 15311 15311 15311 15311 ...
## $ Country : chr "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
```

Setelah kita melihat struktur data dari dataset, maka kita akan melihat ringkasan data, untuk melihat apakah ada data yang “Missing Value” atau **NA’s**. Kita akan menggunakan fungsi `summary()`

```
summary(retail)
```

```
## InvoiceNo StockCode Description Quantity
## Min. :536365 Length:234920 Length:234920 Min. : 1.00
## 1st Qu.:541413 Class :character Class :character 1st Qu.: 1.00
## Median :546635 Mode :character Mode :character Median : 3.00
## Mean :546827 Mean : 10.51
## 3rd Qu.:552298 3rd Qu.: 10.00
## Max. :558077 Max. :74215.00
##
## InvoiceDate UnitPrice CustomerID Country
## Length:234920 Min. : 0.000 Min. :12346 Length:234920
## Class :character 1st Qu.: 1.250 1st Qu.:13859 Class :character
## Mode :character Median : 2.100 Median :15147 Mode :character
## Mean : 4.121 Mean :15283
## 3rd Qu.: 4.150 3rd Qu.:16818
## Max. :13541.330 Max. :18287
## NA's :66245
```

Kalau kita lihat dari hasil diatas ternyata ada data NA’s 66245. Sehingga kita harus membersihkan data ini. Untuk menyelesaikan ini, kita akan bahas pada bagian **Clean the Data**

Untuk melihat data yang *missing value* atau NA’s juga bisa menggunakan fungsi ini:

```
colSums(is.na(retail))
```

```
## InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice
## 0 0 0 0 0 0
## CustomerID Country
## 66245 0
```

Clean the Data

Membersihkan data NA’s

Untuk membersihkan data NA’s kita bisa menggunakan fungsi `na.omit()`

```
retail <- na.omit(retail)
summary(retail)
```

```
## InvoiceNo      StockCode      Description      Quantity
## Min. :536365   Length:168675   Length:168675   Min. : 1.0
## 1st Qu.:542102   Class :character   Class :character   1st Qu.: 2.0
## Median :547243   Mode :character    Mode :character    Median : 6.0
## Mean :547309                      Mean : 13.3
## 3rd Qu.:552796                      3rd Qu.: 12.0
## Max. :558077                      Max. :74215.0
## InvoiceDate      UnitPrice      CustomerID      Country
## Length:168675   Min. : 0.000   Min. :12346   Length:168675
## Class :character   1st Qu.: 1.250   1st Qu.:13859   Class :character
## Mode :character    Median : 1.950   Median :15147   Mode :character
##                      Mean : 3.286   Mean :15283
##                      3rd Qu.: 3.750   3rd Qu.:16818
##                      Max. :8142.750   Max. :18287
```

Nah kalau kita lihat data yang NA's sudah dihilangkan, sehingga data sudah bersih dan siap untuk dianalisis.

```
glimpse(retail)
```

```
## Rows: 168,675
## Columns: 8
## $ InvoiceNo    <int> 536381, 536381, 536381, 536381, 536381, 536381, 536381, 53~
## $ StockCode   <chr> "71270", "22262", "22637", "21166", "37444A", "37444C", "2~
## $ Description <chr> "PHOTO CLIP LINE", "FELT EGG COSY CHICKEN", "PIGGY BANK RE~
## $ Quantity    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ InvoiceDate  <chr> "12/1/10 9:41", "12/1/10 9:41", "12/1/10 9:41", "12/1/10 9~
## $ UnitPrice   <dbl> 1.25, 0.85, 2.55, 1.95, 2.95, 2.95, 2.95, 0.85, 0.85, 1.45~
## $ CustomerID  <int> 15311, 15311, 15311, 15311, 15311, 15311, 15311, 15311, 15~
## $ Country     <chr> "United Kingdom", "United Kingdom", "United Kingdom", "Uni~
```

Merubah type data

Kalau kita perhatikan type data dari InvoiceDate bertipe character, kalau kita ingin analisis, maka kita harus rubah type datanya menjadi date atau datetime. Untuk merubah type data kita bisa menggunakan fungsi `mdy()`, fungsi ini ada dalam library `lubridate` yang sebelumnya sudah kita install. Dalam kasus ini, kita juga menggunakan fungsi `mutate()` fungsi ini ada dalam library `dplyr`, fungsi ini gunanya untuk membuat variabel baru yang diturunkan dari variabel yang sudah ada. Dalam kasus ini kita merubah variabel InvoiceDate dari type character ke variabel InvoiceDate dengan type data DateTime.

Objek retail yang sudah dibersihkan kita simpan dalam objek *retailClean*.

```
# Merubah type data InvoiceDate
```

```
retailClean <- retail %>%
  mutate(InvoiceDate = mdy_hm(InvoiceDate)) %>%
  arrange(InvoiceDate)
glimpse(retailClean)
```

```
## Rows: 168,675
## Columns: 8
## $ InvoiceNo    <int> 536365, 536365, 536365, 536365, 536365, 536365, 536365, 53~
## $ StockCode   <chr> "22752", "85123A", "71053", "84029G", "84029E", "21730", "~
```

```
## $ Description <chr> "SET 7 BABUSHKA NESTING BOXES", "WHITE HANGING HEART T-LIG~
## $ Quantity      <int> 2, 6, 6, 6, 6, 6, 8, 6, 6, 2, 3, 3, 3, 3, 3, 3, 4, 4, 6, 6~
## $ InvoiceDate    <dtm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12-01 08:2~
## $ UnitPrice     <dbl> 7.65, 2.55, 3.39, 3.39, 3.39, 4.25, 2.75, 1.85, 1.85, 9.95~
## $ CustomerID    <int> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17~
## $ Country       <chr> "United Kingdom", "United Kingdom", "United Kingdom", "Uni~
```

Bisa kita lihat bahwa sekarang variabel *InvoiceDate* type datanya sudah berubah menjadi type *datetime* (*dtm*)

Keterangan: Operator Pipeline atau %>% (dibaca piping) digunakan untuk merangkai beberapa fungsi dalam urutan operasi. Sehingga kita dapat menuliskan lebih dari satu fungsi sekaligus tanpa harus menyimpannya terlebih dahulu. Operator pipeline bisa dibuat dengan cepat menggunakan kombinasi “**ctrl + shift + m**”

Ekplorasi Data

Untuk Ekplorasi data atau proses data, bisa disesuaikan dengan kebutuhan. Untuk mendukung hal tersebut kita bisa menggunakan library `dplyr`. Berikut ini beberapa fungsi lain yang ada dalam `dplyr` yang bisa kita gunakan.

Filter

Fungsi `filter()` digunakan untuk menyeleksi dan menampilkan data sesuai dengan kebutuhan. Misalnya kita ingin memfilter `StockCode = 85123A`. maka penulisan fungsi filter bisa dilakukan seperti ini. Hasil filter kita simpan dalam objek *stockcode85123A*

```
stockcode85123A <- filter(retailClean, StockCode == "85123A")
head(stockcode85123A)
```

##	InvoiceNo	StockCode	Description	Quantity
## 1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
## 2	536373	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
## 3	536375	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
## 4	536390	85123A	WHITE HANGING HEART T-LIGHT HOLDER	64
## 5	536394	85123A	WHITE HANGING HEART T-LIGHT HOLDER	32
## 6	536396	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
##	InvoiceDate	UnitPrice	CustomerID	Country
## 1	2010-12-01 08:26:00	2.55	17850	United Kingdom
## 2	2010-12-01 09:02:00	2.55	17850	United Kingdom
## 3	2010-12-01 09:32:00	2.55	17850	United Kingdom
## 4	2010-12-01 10:19:00	2.55	17511	United Kingdom
## 5	2010-12-01 10:39:00	2.55	13408	United Kingdom
## 6	2010-12-01 10:51:00	2.55	17850	United Kingdom

Maka bisa dilihat bahwa, semua data akan ditampilkan hanya `StockCode = 85123A`

Misalnya kita akan memfilter jumlah transaksi hanya dari asal negara United Kingdom pada `StockCode 85123A`, maka fungsi filter bisa tulis seperti ini.

```
UK <- stockcode85123A %>%
  filter(Country == "United Kingdom")
head(UK)
```

##	InvoiceNo	StockCode	Description	Quantity
## 1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
## 2	536373	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
## 3	536375	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
## 4	536390	85123A	WHITE HANGING HEART T-LIGHT HOLDER	64
## 5	536394	85123A	WHITE HANGING HEART T-LIGHT HOLDER	32
## 6	536396	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6

##	InvoiceDate	UnitPrice	CustomerID	Country
## 1	2010-12-01 08:26:00	2.55	17850	United Kingdom
## 2	2010-12-01 09:02:00	2.55	17850	United Kingdom
## 3	2010-12-01 09:32:00	2.55	17850	United Kingdom
## 4	2010-12-01 10:19:00	2.55	17511	United Kingdom
## 5	2010-12-01 10:39:00	2.55	13408	United Kingdom
## 6	2010-12-01 10:51:00	2.55	17850	United Kingdom

Count dan Group By

Fungsi `count()` digunakan untuk mengetahui jumlah data berdasarkan kategori/variabel yang sudah ditentukan sebelumnya. Misalnya dalam kasus ini kita akan hitung jumlah transaksi berdasarkan variabel *Country* yang ada pada objek *stockcode85123A*

```
count(stockcode85123A, Country)
```

##	Country	n
## 1	Australia	1
## 2	Channel Islands	1
## 3	Cyprus	3
## 4	EIRE	15
## 5	Finland	1
## 6	France	1
## 7	Malta	1
## 8	Netherlands	4
## 9	Portugal	2
## 10	Singapore	1
## 11	Spain	3
## 12	Switzerland	1
## 13	United Kingdom	1097

Group By dan Arrange

Fungsi `group_by()` digunakan untuk mengelompokkan data berdasarkan satu atau lebih variabel. Fungsi `arrange()` digunakan untuk mengurutkan data berdasarkan variabel. Pengurutan bisa dilakukan dari kecil ke besar atau sebaliknya. Misalnya dalam kasus ini kita ingin mengelompokkan data berdasarkan variabel *Country* dan sekaligus menghitung jumlah transaksinya dan diurutkan dari besar ke kecil.

```
stockcode85123A %>%
  group_by(Country) %>%
  count() %>%
  arrange(-n)
```

```
## # A tibble: 13 x 2
```

```
## # Groups:   Country [13]
##   Country      n
##   <chr>      <int>
## 1 United Kingdom 1097
## 2 EIRE           15
## 3 Netherlands     4
## 4 Cyprus          3
## 5 Spain           3
## 6 Portugal        2
## 7 Australia        1
## 8 Channel Islands  1
## 9 Finland         1
## 10 France          1
## 11 Malta           1
## 12 Singapore       1
## 13 Switzerland     1
```

Bisa kita lihat negara yang paling banyak melakukan transaksi adalah United Kingdom yaitu sebanyak 1097 kali

Sekarang kita akan hitung berapa jumlah transaksi berdasarkan StockCode, maka bisa kita tuliskan seperti ini.

```
retailClean %>%
  group_by(StockCode) %>%
  count() %>%
  arrange(-n)
```

```
## # A tibble: 3,264 x 2
## # Groups:   StockCode [3,264]
##   StockCode      n
##   <chr>      <int>
## 1 85123A      1131
## 2 22423       961
## 3 47566       787
## 4 85099B      704
## 5 84879       664
## 6 20725       633
## 7 21212       584
## 8 22720       562
## 9 22960       515
## 10 22457      512
## # ... with 3,254 more rows
```

Transaksi yang paling banyak adalah untuk StockCode 85123A sebanyak 1131.

Contoh lain, bagaimana kita menghitung jumlah transaksi berdasarkan InvoiceNo.

```
retailClean %>%
  group_by(InvoiceNo) %>%
  count() %>%
  arrange(-n)
```

```
## # A tibble: 8,590 x 2
## # Groups:   InvoiceNo [8,590]
##   InvoiceNo     n
##   <int> <int>
## 1    547063   294
## 2    554098   264
## 3    543040   259
## 4    556484   205
## 5    552039   176
## 6    540372   171
## 7    537224   169
## 8    545901   164
## 9    537781   161
## 10   540247   157
## # ... with 8,580 more rows
```

Bisa dilihat bahwa InvoiceNo 547063 berisi sebanyak 294 transaksi.

Sampling

Fungsi `sample_n()` digunakan untuk mengambil secara acak data, artinya kita bisa mengambil sampel dari data secara acak. Misalnya kita ingin mengambil sebanyak 5 sampel data dari variabel Quantity pada objek UK.

```
sample_n(UK, size = 10)
```

```
##   InvoiceNo StockCode Description Quantity
## 1    540689   85123A WHITE HANGING HEART T-LIGHT HOLDER    1010
## 2    556926   85123A WHITE HANGING HEART T-LIGHT HOLDER         1
## 3    540242   85123A WHITE HANGING HEART T-LIGHT HOLDER        32
## 4    554079   85123A WHITE HANGING HEART T-LIGHT HOLDER        12
## 5    543251   85123A WHITE HANGING HEART T-LIGHT HOLDER         6
## 6    543668   85123A WHITE HANGING HEART T-LIGHT HOLDER         3
## 7    541564   85123A WHITE HANGING HEART T-LIGHT HOLDER         1
## 8    546913   85123A WHITE HANGING HEART T-LIGHT HOLDER         3
## 9    548711   85123A WHITE HANGING HEART T-LIGHT HOLDER         3
## 10   553781   85123A WHITE HANGING HEART T-LIGHT HOLDER        10
##   InvoiceDate UnitPrice CustomerID Country
## 1 2011-01-11 08:43:00     3.24    17450 United Kingdom
## 2 2011-06-15 14:26:00     2.95    16324 United Kingdom
## 3 2011-01-05 15:30:00     2.55    15046 United Kingdom
## 4 2011-05-22 11:41:00     2.95    14546 United Kingdom
## 5 2011-02-04 14:51:00     2.95    14194 United Kingdom
## 6 2011-02-11 11:17:00     2.95    18118 United Kingdom
## 7 2011-01-19 11:38:00     2.95    16904 United Kingdom
## 8 2011-03-17 20:18:00     2.95    15861 United Kingdom
## 9 2011-04-03 14:24:00     2.95    18116 United Kingdom
## 10 2011-05-19 10:54:00     2.95    16008 United Kingdom
```

Select

Fungsi `select()` digunakan untuk mengambil satu atau beberapa variabel tertentu yang ada dalam dataset. Sebagai contoh disini kita akan mengambil variabel InvoiceNo, dan Quantity dan tampilkan hanya 6 data

teratas.

```
head(select(UK, c(1,4)))
```

```
##   InvoiceNo Quantity
## 1    536365        6
## 2    536373        6
## 3    536375        6
## 4    536390       64
## 5    536394       32
## 6    536396        6
```

Summarise

Fungsi `summarise()` digunakan untuk meringkas beberapa nilai data menjadi sebuah nilai. Dalam prakteknya fungsi ini akan sangat berguna kalau digabungkan dengan fungsi-fungsi yang lain. Sebagai contoh dalam kasus ini kita akan menampilkan jumlah transaksi dari negara UK berdasarkan jumlah Quantity harian. Nilainya kita akan simpan pada objek *UK_daily_retail*

```
UK_daily_retail <- UK %>%
  group_by(InvoiceDate) %>%
  summarise(
    jmlTrans = sum (Quantity)
  )
head(arrange(UK_daily_retail, (-jmlTrans)))
```

```
## # A tibble: 6 x 2
##   InvoiceDate      jmlTrans
##   <dtm>          <int>
## 1 2011-01-11 12:55:00    1930
## 2 2011-04-18 13:20:00    1930
## 3 2011-01-11 08:43:00   1010
## 4 2011-05-26 19:49:00    608
## 5 2010-12-16 11:07:00    500
## 6 2011-06-02 14:46:00    320
```

```
tail(arrange(UK_daily_retail, (-jmlTrans)))
```

```
## # A tibble: 6 x 2
##   InvoiceDate      jmlTrans
##   <dtm>          <int>
## 1 2011-05-29 14:03:00      1
## 2 2011-06-01 12:05:00      1
## 3 2011-06-02 11:33:00      1
## 4 2011-06-05 15:46:00      1
## 5 2011-06-15 14:26:00      1
## 6 2011-06-23 19:20:00      1
```

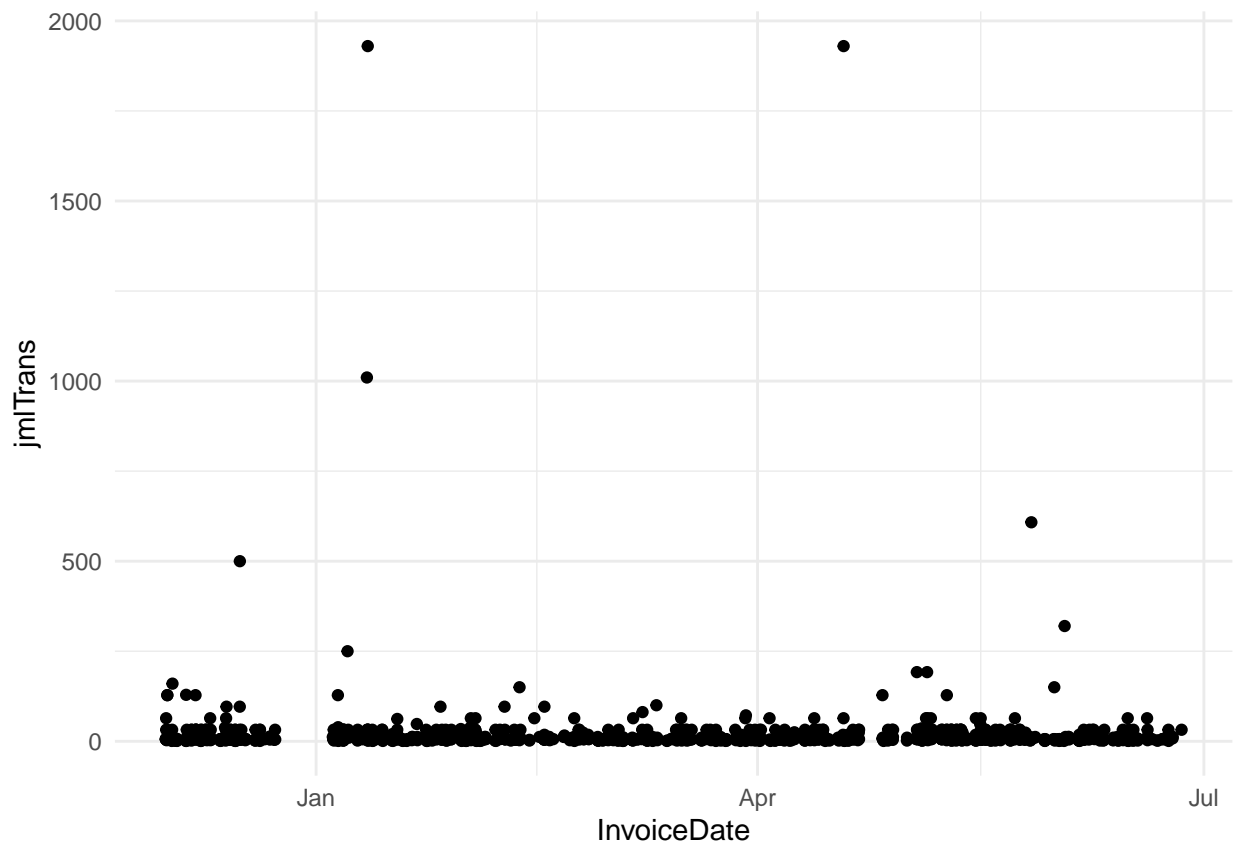
Bisa dilihat bahwa transaksi paling banyak ada pada tanggal 11-01-2011 dan 18-04-2011 sebanyak 1930 transaksi.

Visualization Analysis

Visualiasi analisis ini adalah bagaimana kita memvisualisasikan hasil Explanatory Data Analysis yang sudah kita lakukan sebelumnya. Dalam kasus ini kita akan memvisualisasikan dengan menggunakan library `ggplot2`

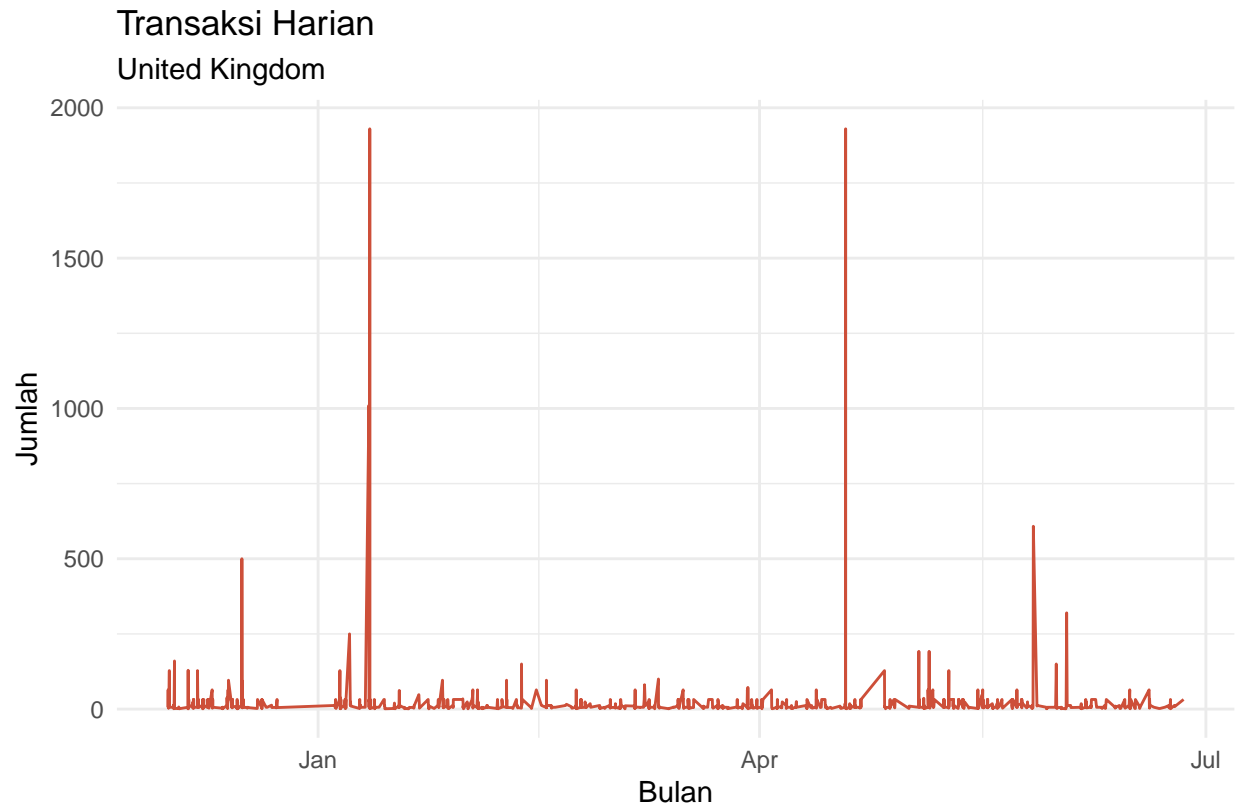
Sebagai contoh kita akan memvisualisasikan hasil dari transaksi harian yang ada pada negara UK yang sebelumnya sudah kita simpan pada objek `UK_daily_retail`

```
UK_daily_retail %>%  
  ggplot(aes(x=InvoiceDate, y=jmlTrans)) +  
  geom_point() +  
  theme_minimal()
```



Contoh visualisasi lain yang dilengkapi dengan title dan subtitle

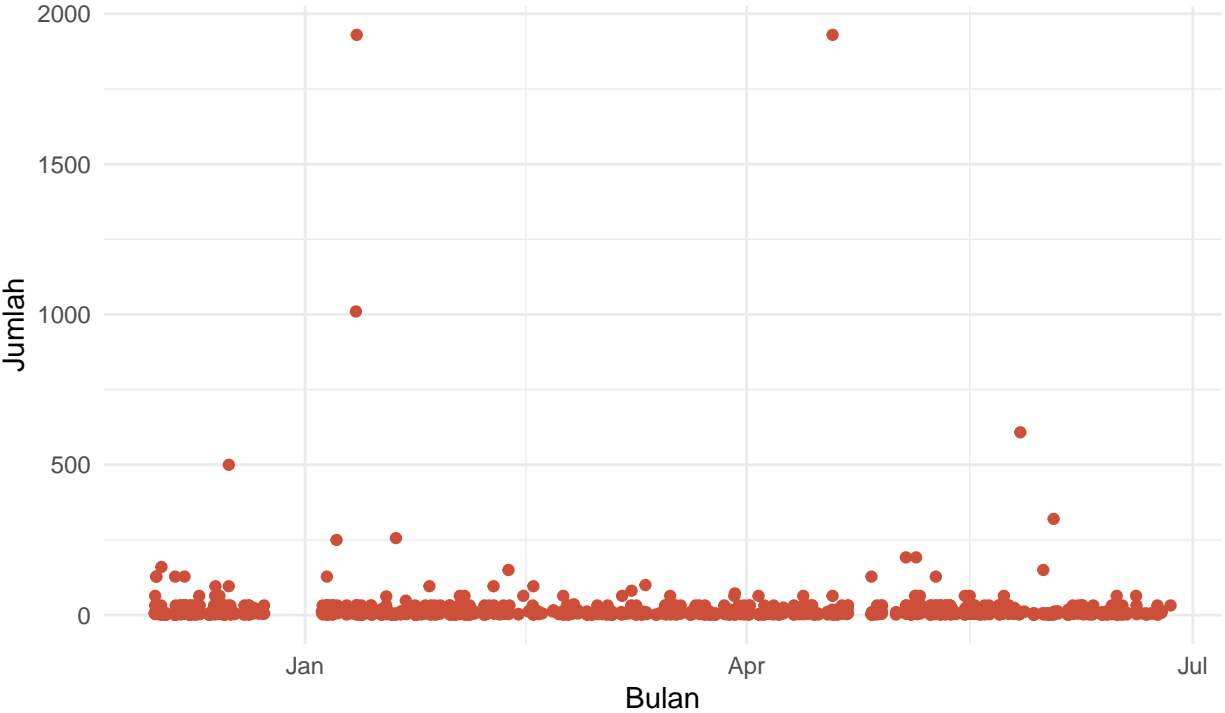
```
UK_daily_retail %>%  
  ggplot(aes(x=InvoiceDate, y=jmlTrans)) +  
  geom_line(color = "tomato3") +  
  labs(  
    title = "Transaksi Harian",  
    subtitle = "United Kingdom",  
    caption = "by: Roni Yunis",  
    x = "Bulan",  
    y = "Jumlah"  
  ) +  
  theme_minimal()
```



Misalkan kita diminta untuk memvisualisasi transaksi berdasarkan InvoiceDate dan Quantity berdasarkan semua transaksi dengan StockCode = 85123A

```
ggplot(stockcode85123A)+
  aes(x=InvoiceDate, y=Quantity) +
  geom_point (colour = "tomato3") +
  labs(
    title = "Transaksi Harian",
    subtitle = "Stock Code 85123A",
    caption = "by: Roni Yunis",
    x = "Bulan",
    y = "Jumlah"
  ) +
  theme_minimal()
```

Transaksi Harian
Stock Code 85123A



by: Roni Yunis