

# IXIS Data Science Challenge

William Terpstra

2022-08-08

```
#set working directory & load data
setwd("C:/Users/Will/Documents/IXIS_Test/Ecommerce_Data_Sci_Challenge")
#loading data
adds_df <- read.csv("DataAnalyst_Ecom_data_addsToCart.csv")
session_df <- read.csv("DataAnalyst_Ecom_data_sessionCounts.csv")
```

```
#first lets verify data types
str(adds_df)
```

```
## 'data.frame': 12 obs. of 3 variables:
## $ dim_year : int 2012 2012 2012 2012 2012 2012 2012 2013 2013 2013 2013 ...
## $ dim_month : int 7 8 9 10 11 12 1 2 3 4 ...
## $ addsToCart: int 191504 217666 123726 139803 186572 168972 147619 135882 109797 183842 ...
```

```
str(session_df)
```

```
## 'data.frame': 7734 obs. of 6 variables:
## $ dim_browser : chr "Safari" "Internet Explorer" "Chrome" "Amazon Silk" ...
## $ dim_deviceCategory: chr "tablet" "desktop" "tablet" "tablet" ...
## $ dim_date : chr "7/1/12" "7/1/12" "7/1/12" "7/1/12" ...
## $ sessions : int 2928 1106 474 235 178 120 10 9 5 4 ...
## $ transactions : int 127 28 3 4 6 7 0 0 0 0 ...
## $ QTY : int 221 0 13 5 11 0 0 0 0 0 ...
```

```
#adds_df seems fine, could convert the dates into a date object
#but doesn't seem necessary
```

```
#session_df on the other hand requires some conversions, again I could convert
#the dates into a date object but I'll actually do the separate() approach
session_df <- session_df %>%
  #separating the date column into month, date, and year
  separate(dim_date, c("month","day","year"), "/", convert = TRUE)
#while we are at it, lets rename QTY to follow the naming conventions of other variables
session_df <- session_df %>% rename(quantity = QTY)
#next dim_browser and dim_deviceCategory should be factors
session_df$dim_browser <- as.factor(session_df$dim_browser)
session_df$dim_deviceCategory <- as.factor(session_df$dim_deviceCategory)
#okay data types are all addressed except dim_browser is a factor with
#57 levels, it makes sense to condense that
```

```
#but first lets perform de-duplication since duplicates could influence
#browser frequency
setdiff(session_df %>% distinct(), session_df)
```

```
## [1] dim_browser      dim_deviceCategory month      day
## [5] year              sessions      transactions  quantity
## <0 rows> (or 0-length row.names)
```

```
#it seems there are no duplicates so we are good to proceed
```

```
#now it is time to finish addressing data type issues
```

```
#looking at counts and deciles of browser frequency to determine a cut off
```

```
session_df %>% count(dim_browser) %>% arrange(desc(n)) %>% mutate(decile = ntile(n, 10))
```

```
##           dim_browser    n decile
## 1             Chrome 679      10
## 2      Internet Explorer 673      10
## 3             Safari 669      10
## 4              Edge 535      10
## 5            Firefox 522      10
## 6      Safari (in-app) 476       9
## 7              Opera 471       9
## 8      Android WebView 458       9
## 9      Samsung Internet 380       9
## 10           Amazon Silk 366       9
## 11              error 364       8
## 12      Android Browser 351       8
## 13           BlackBerry 224       8
## 14           SeaMonkey 204       8
## 15           Opera Mini 161       8
## 16           UC Browser 155       7
## 17           Mozilla 143       7
## 18           Maxthon 127       7
## 19           YaBrowser 121       7
## 20             Puffin 99       7
## 21           (not set) 98       7
## 22 Mozilla Compatible Agent 93       6
## 23      osee2unifiedRelease 92       6
## 24             Coc Coc 65       6
## 25              Iron 55       6
## 26           BrowserNG 22       6
## 27           DESKTOP 20       6
## 28           Truefitbot 17       5
## 29      DDG-Android-3.1.1 16       5
## 30           MRCHROME 10       5
## 31      NokiaC7-00 10       5
## 32      NokiaE52-1 7       5
## 33           YelpWebView 6       5
## 34      IE with Chrome Frame 5       4
## 35              Seznam 5       4
## 36      Apple-iPhone7C2 3       3
## 37      DDG-Android-3.0.14 3       4
## 38             LG-C410 3       4
```

```
## 39          NetFront      3      4
## 40          TimesTablet  3      4
## 41          Amazon.com   2      3
## 42          Nokia Browser 2      3
## 43          SonyEricssonK700c 2      3
## 44          anonymous    1      1
## 45          Chromeless 1.2.0 1      1
## 46          DDG-Android-3.0.11 1      1
## 47          DDG-Android-3.0.17 1      1
## 48          FeeddlerPro  1      1
## 49 HubSpot inbound link reporting check 1      1
## 50          Job Search   1      2
## 51          Mobile       1      2
## 52          NetNewsWire Browser 1      2
## 53          Nintendo Browser 1      2
## 54          Playstation 3 1      2
## 55          Python-urllib 1      2
## 56          turnaround   1      3
## 57          X-WebBrowser  1      3
```

```
#lets just do a cut off at the top 25 browsers, and lump the rest into other
session_df$dim_browser <- fct_lump_n(session_df$dim_browser, 25)
#verifying it worked
#session_df%>% count(dim_browser) %>% arrange(desc(n))

#creating a function to check for NAs
NAcheck <- function(df) {
  names <- c()
  percent_of_missing_values <- c()
  for(i in 1:ncol(df)) { # for-loop over columns in the data frame

    #adding the name of each column to a vector
    names <- append(names, colnames(df[i]))
    #adding the amount of missing values of each column to a vector
    percent_of_missing_values <- append(percent_of_missing_values, sum(is.na(df[,i]))/nrow(df))
  }
  #using the two vectors to output a data frame
  #with the names of columns and their amount of missing values
  data.frame(names, percent_of_missing_values)
}

#checking for missing values
NAcheck(adds_df)
```

```
##          names percent_of_missing_values
## 1  dim_year                0
## 2  dim_month                0
## 3  addsToCart               0
```

```
NAcheck(session_df)
```

```
##          names percent_of_missing_values
## 1  dim_browser                0
```

```
## 2 dim_deviceCategory      0
## 3           month          0
## 4           day           0
## 5           year          0
## 6       sessions          0
## 7   transactions          0
## 8       quantity          0
```

*#seems there are no missing values*

*#next lets verify the dates span a 12 month period*

*#and there isn't any odd overlap we would have to take into account*

```
session_df %>%
  select(month, year) %>%
  distinct() %>%
  arrange(month)
```

```
##      month year
## 1         1  13
## 2         2  13
## 3         3  13
## 4         4  13
## 5         5  13
## 6         6  13
## 7         7  12
## 8         8  12
## 9         9  12
## 10        10  12
## 11        11  12
## 12        12  12
```

*#seems the date range of the sent data is correct*

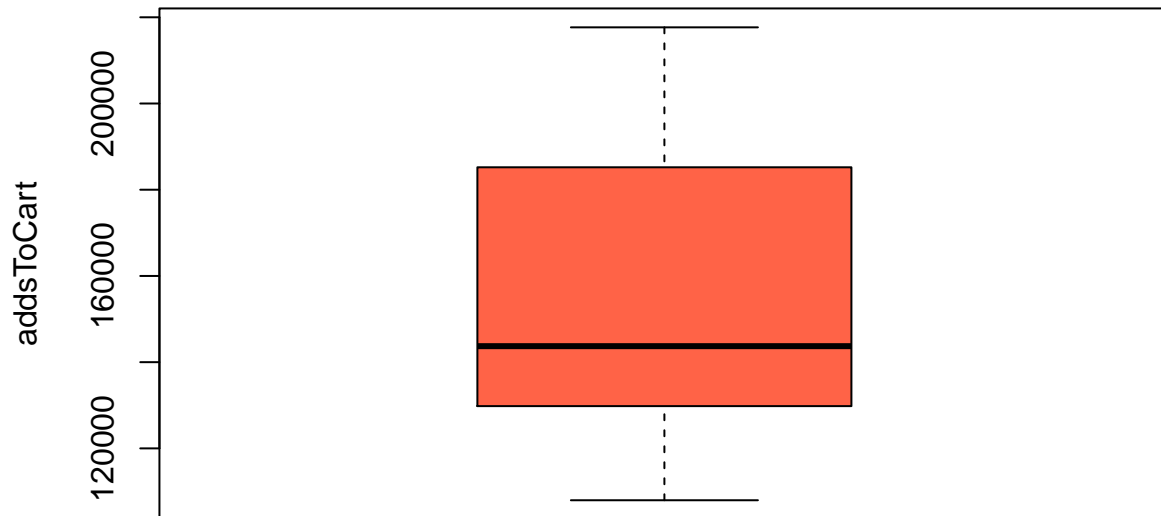
*#next lets check for anomalous values/outliers*

*#adds\_df is so small I manually reviewed it for errors*

*#but to be diligent lets make a quick and dirty box plot*

```
boxplot(adds_df$addsToCart,
        main = "addsToCart Outlier Check",
        ylab = "addsToCart",
        col = "tomato")
```

## addsToCart Outlier Check

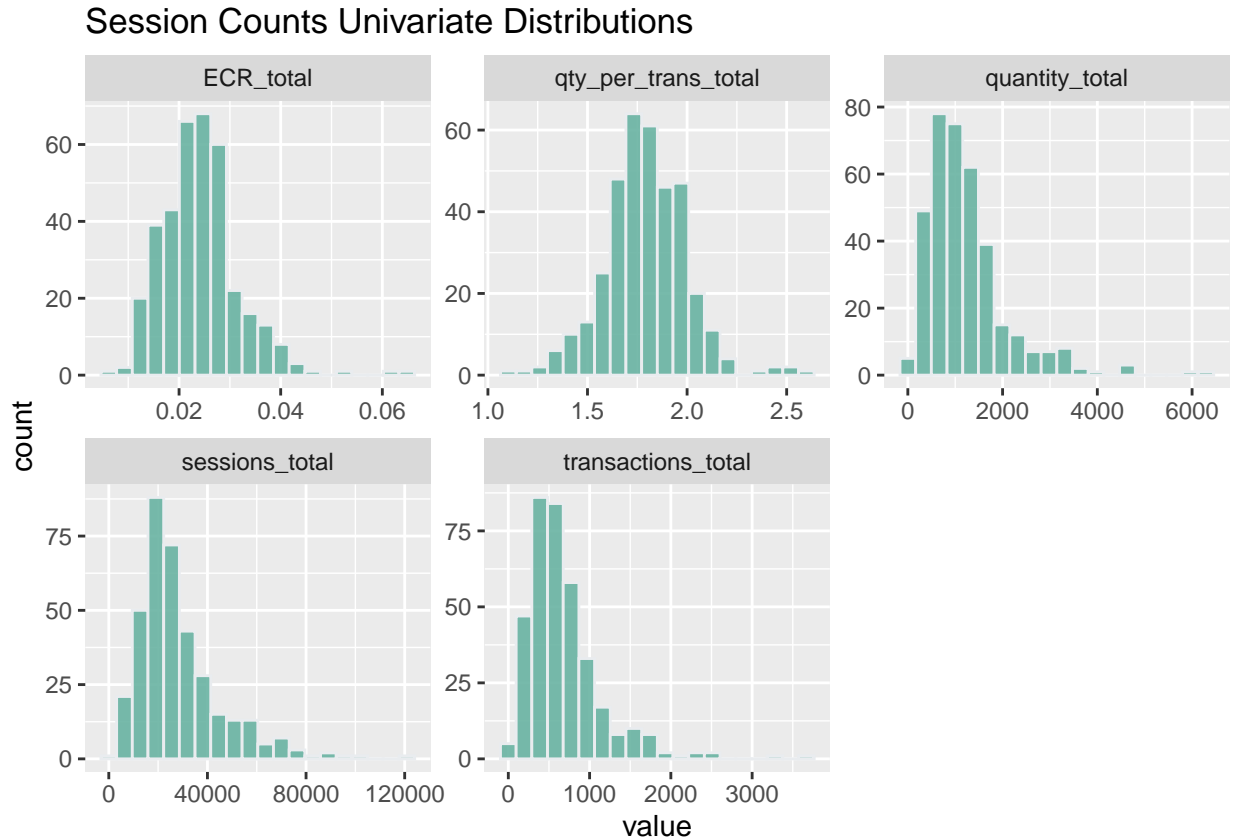


```
#the data seems plausible with no egregious outliers

#now sessions_df is much larger and must be assessed through code and visualizations
#lets start by looking at daily data
daily_s_df <- session_df %>%
  #grouping by day and month so we can summarize each statistic by day
  group_by(month, day) %>%
  #removing year
  dplyr::select(-c(year)) %>%
  #summarizing the daily average and standard deviation for each statistic
  summarise_if(is.numeric, list(total = sum)) %>%
  #calculating ECR to look at that as well
  mutate(ECR_total = transactions_total/sessions_total) %>%
  mutate(qty_per_trans_total = quantity_total/transactions_total)
#un-grouping data
daily_s_df <- daily_s_df %>% ungroup

#looking at univariate distributions to assess for anomalies
ggplot(gather(daily_s_df %>%
  #since we are looking at data summarized by day and month
  #it doesn't make sense to look at these variables
  dplyr::select(-c(day, month)) %>%
  dplyr::select(where(is.numeric))),
  aes(value)) +
  geom_histogram(fill="#69b3a2", color="#e9ecef", alpha=0.9, bins = 20) +
  facet_wrap(~key, scales = 'free') +
```

```
labs(title = "Session Counts Univariate Distributions")
```



*#there is a slight chance for an outlier in sessions avg per day  
 #and a significant chance for an outlier in transactions avg per day  
 #maybe something went viral, or there was a sale, let's investigate*

*#seems Jan 12th and June 8th are the odd ones out*  
 daily\_s\_df %>% arrange(desc(transactions\_total))

```
## # A tibble: 365 x 7
##   month  day sessions_total transactions_total quantity_total ECR_total
##   <int> <int>         <int>             <int>           <int>      <dbl>
## 1     6     8          96162              3721           6376    0.0387
## 2     5    25         123562              3222           5910    0.0261
## 3     5    19         103443              2599           4581    0.0251
## 4     1    12          55110              2577           4480    0.0468
## 5     5    29          91723              2288           4524    0.0249
## 6     6    20          74372              2236           3542    0.0301
## 7     4    12          53373              2117           4071    0.0397
## 8     6     1          83964              1957           3554    0.0233
## 9     5    16          70390              1877           3413    0.0267
## 10    6    19          75439              1816           3342    0.0241
## # ... with 355 more rows, and 1 more variable: qty_per_trans_total <dbl>
```

```

#it is quite odd that they are one off days
#sessions are at the high end of their distribution these days
#but more significantly it corresponds with a max for Quantity
#which may seem obvious but this led me to discover there can be transactions
#with a corresponding quantity of zero, indicating maybe transactions are recorded
#before a purchase is finalized, this may be something that should be addressed
#in terms of improving data collection or leveraged with additional data assets
#to discover why prospective customers start but do not complete transactions

#looking at mean, median, quantiles, max, min etc. of the variables
summary(daily_s_df)

```

```

##      month      day  sessions_total  transactions_total
##  Min.   : 1.000   Min.   : 1.00   Min.   : 2701   Min.   : 54
##  1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 17635  1st Qu.: 396
##  Median : 7.000   Median :16.00   Median : 24716  Median : 558
##  Mean   : 6.526   Mean   :15.72   Mean   : 28545  Mean   : 684
##  3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.: 34759  3rd Qu.: 846
##  Max.   :12.000   Max.   :31.00   Max.   :123562  Max.   :3721
##  quantity_total  ECR_total      qty_per_trans_total
##  Min.   : 85     Min.   :0.006711   Min.   :1.112
##  1st Qu.: 703     1st Qu.:0.019003   1st Qu.:1.680
##  Median :1038     Median :0.023920   Median :1.783
##  Mean   :1235     Mean   :0.024176   Mean   :1.793
##  3rd Qu.:1541     3rd Qu.:0.027974   3rd Qu.:1.923
##  Max.   :6376     Max.   :0.065355   Max.   :2.609

```

```

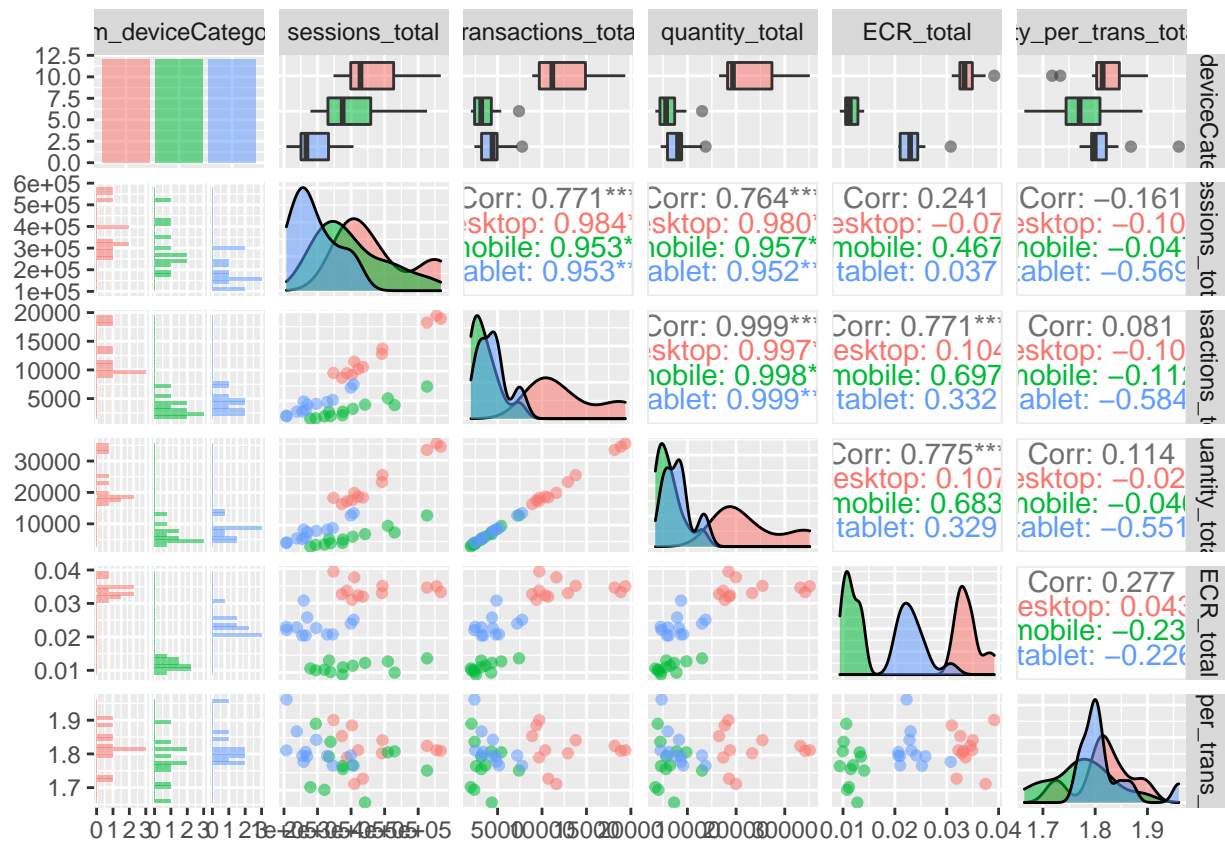
#visualizing all variable pairs
session_df %>%
  #grouping by device and month so we can summarize each statistic
  group_by(dim_deviceCategory, month) %>%
  #removing year and day
  dplyr::select(-c(year, day)) %>%
  #summarizing the daily average and standard deviation for each statistic
  summarise_if(is.numeric, list(total = sum)) %>%
  #calculating ECR to look at that as well
  mutate(ECR_total = transactions_total/sessions_total) %>%
  #also assessing quantity sold per transaction
  mutate(qty_per_trans_total = quantity_total/transactions_total) %>%
  ungroup() %>%
  #removing month from the pairwise assessment
  dplyr::select(-c(month)) %>%
  #finally conducting a pairwise visualization
  GGally::ggpairs(aes(colour = dim_deviceCategory, alpha = 0.4))

```

```

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



*#transactions and quantity sold are extremely correlated*

```
device_ag_df <- session_df %>%
  #using the group_by() and summarize() functions to calculate the totals of
  #numeric variables by device category and month
  #I also include year for clarity about the past 12 month period being analyzed
  group_by(dim_deviceCategory, month, year) %>%
  summarise_if(is.numeric, sum) %>%
  #adding a column for the effective conversion rate
  mutate(ECR = transactions/sessions) %>%
  #mutating year just for clarification
  mutate(year = year + 2000) %>%
  #capitalizing device names for plotting
  mutate(dim_deviceCategory = str_to_title(dim_deviceCategory)) %>%
  dplyr::select(-c(day))
#ungrouping
device_ag_df <- device_ag_df %>% ungroup()

#arranging the data for intuitive display
device_ag_df <- device_ag_df %>% arrange(dim_deviceCategory, year, month)
#seems this part of the deliverable displays correctly
device_ag_df
```

```
## # A tibble: 36 x 7
##   dim_deviceCategory month   year sessions transactions quantity    ECR
```



```
##      <chr>          <int> <dbl>      <int>      <int>      <int> <dbl>
## 1 Desktop          7  2012    335429      10701    18547 0.0319
## 2 Desktop          8  2012    392079      12912    23316 0.0329
## 3 Desktop          9  2012    272771       8898    16507 0.0326
## 4 Desktop         10  2012    302682       9373    17675 0.0310
## 5 Desktop         11  2012    320717      10350    18778 0.0323
## 6 Desktop         12  2012    309718      11613    19947 0.0375
## 7 Desktop          1  2013    393723      13793    25424 0.0350
## 8 Desktop          2  2013    247632       9699    18437 0.0392
## 9 Desktop          3  2013    287837       9679    17362 0.0336
## 10 Desktop         4  2013    567510      18868    34200 0.0332
## # ... with 26 more rows
```

```
#creating a visualization of avg monthly session share by device
p1 <- device_ag_df %>% group_by(dim_deviceCategory) %>%
  dplyr::select(-c(month, year)) %>%
  #calculating the monthly average
  summarize_if(is.numeric, list(monthly_avg = mean)) %>%
  #calculating the percent of monthly average sessions each device accounts for
  mutate(prc_ses_m_avg = sessions_monthly_avg/sum(sessions_monthly_avg)) %>%
  #making a bar plot ordered by device share of avg monthly sessions
  #which is the same as the share of total sessions, which is a
  #much more intuitive framing for stakeholders
  ggplot(aes(reorder(dim_deviceCategory, prc_ses_m_avg),
    prc_ses_m_avg, fill=dim_deviceCategory)) +
  geom_col(show.legend = FALSE) +
  scale_y_continuous(labels = scales::percent, limits = c(0,1)) +
  #assigning colors that correspond to each device, will be used consistently
  scale_fill_manual(values = c("Desktop" = "#F8766D",
    "Mobile"="#7CAE00",
    "Tablet"="#00BFC4",
    "Total"="#C77CFF")) +

  coord_flip() +
  #adding a label to the bar plot
  geom_text(size = 5,
    aes(label = scales::percent(round(prc_ses_m_avg, 3)),
      y = prc_ses_m_avg),
    hjust = -.1) +
  labs(title = "Share of Sessions by Device",
    x = "Device",
    y = "Sessions")

#this simply adds 12 new rows corresponding to the monthly
#totals across all devices for each statistic
device_ag_df <- rbind(device_ag_df, session_df %>%
  #using the group_by() and summarize() functions to calculate the totals of
  #numeric variables by device category and month
  #I also include year for clarity about the past 12 month period being analyzed
  group_by(year, month) %>%
  dplyr::select(-c(day)) %>%
  summarise_if(is.numeric, sum) %>%
  #adding a column for the effective conversion rate
  mutate(ECR = transactions/sessions) %>%
  #mutating year just for clarification
```

```

mutate(year = year + 2000) %>%
mutate(dim_deviceCategory = "Total"))

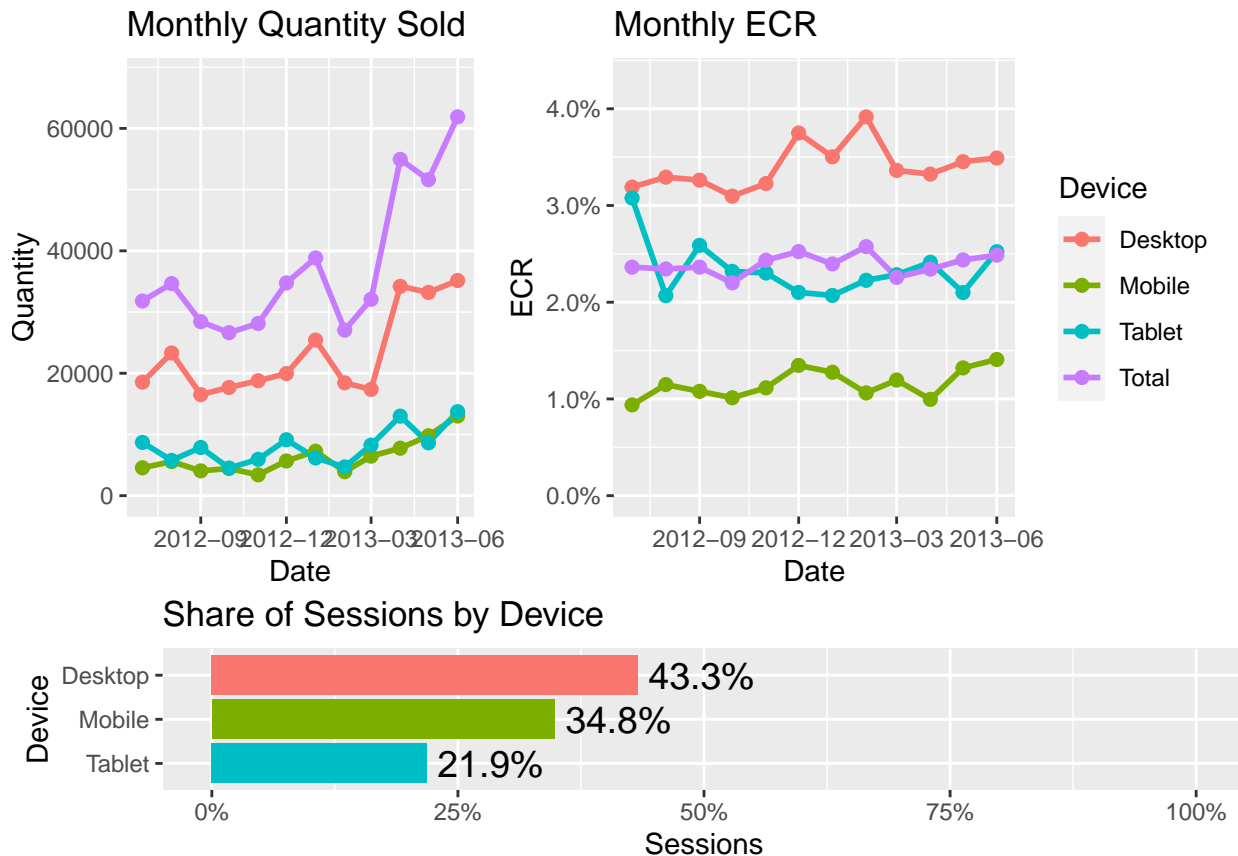
#adding a column for a formal date object for ggplot visualizations
device_ag_df$Date<-as.Date(with(device_ag_df,paste(year,month,1,sep="-")), "%Y-%m-%d")

#plotting monthly quantity by device
p2 <- ggplot(device_ag_df, aes(x=Date, y = quantity, color = dim_deviceCategory)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  scale_fill_manual(values = c("Desktop" = "#F8766D",
                                "Mobile"="#7CAE00",
                                "Tablet"="#00BFC4",
                                "Total"="#C77CFF")) +
  scale_x_date(date_labels= "%Y-%m", date_breaks = "3 months") +
  scale_y_continuous(limits = c(0, max(device_ag_df$quantity*1.1))) +
  labs(x = "Date",
       y = "Quantity",
       title = "Monthly Quantity Sold") +
  theme(plot.margin = margin(0,.5,0,0, "cm"),
        legend.position="none")

#plotting monthly ECR by device
p3 <- device_ag_df %>% rename(Device = dim_deviceCategory) %>%
  ggplot( aes(x=Date, y = ECR, color = Device)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  scale_fill_manual(values = c("Desktop" = "#F8766D",
                                "Mobile"="#7CAE00",
                                "Tablet"="#00BFC4",
                                "Total"="#C77CFF")) +
  scale_x_date(date_labels= "%Y-%m", date_breaks = "3 months") +
  scale_y_continuous(labels = scales::percent, limits = c(0, max(device_ag_df$ECR*1.1))) +
  labs(x = "Date",
       y = "ECR",
       title = "Monthly ECR") +
  theme(plot.margin = margin(0,.5,0,0, "cm"))

lay <- rbind(c(1,1,2,2,2),
             c(1,1,2,2,2),
             c(3,3,3,3,3))
grid.arrange(p2, p3, p1, layout_matrix = lay)

```



```
#removing date column to reduce redundancy in the final deliverable,
#could remove month and year columns instead
device_ag_df <- device_ag_df %>% dplyr::select(-c(Date)) %>%
  #removing the rows for totals across all devices
  filter(dim_deviceCategory != "Total")

#creating the second deliverable

month_ag_df <- session_df %>%
  #using the group_by() and summarize() functions to calculate the totals of
  #numeric variables by device category and month
  #I also include year for clarity about the past 12 month period being analyzed
  group_by(year, month) %>%
  dplyr::select(-c(day)) %>%
  summarise_if(is.numeric, sum) %>%
  #adding a column for the effective conversion rate
  mutate(ECR = transactions/sessions) %>%
  #mutating year just for clarification
  mutate(year = year + 2000)
#ungrouping
month_ag_df <- month_ag_df %>% ungroup()

#now adding addsToCart and ATCR
month_ag_df <- month_ag_df %>%
  left_join(adds_df %>% dplyr::select(-c(dim_year)), by = c("month" = "dim_month")) %>%
  #calculating ATCR
```

```

#small chance this is incorrect since I presume addToCart is on a per session basis
mutate(ATCR = addToCart/sessions)
#verifying the data frame is processed correctly
month_ag_df

```

```

## # A tibble: 12 x 8
##   year month sessions transactions quantity   ECR addToCart   ATCR
##   <dbl> <int>   <int>         <int>    <int> <dbl>   <int>   <dbl>
## 1  2012     7   768589         18161    31804 0.0236   191504 0.249
## 2  2012     8   822493         19279    34648 0.0234   217666 0.265
## 3  2012     9   662653         15658    28426 0.0236   123726 0.187
## 4  2012    10   648639         14275    26626 0.0220   139803 0.216
## 5  2012    11   637780         15527    28132 0.0243   186572 0.293
## 6  2012    12   789634         19929    34752 0.0252   168972 0.214
## 7  2013     1   899992         21560    38846 0.0240   147619 0.164
## 8  2013     2   550227         14166    27048 0.0257   135882 0.247
## 9  2013     3   788820         17804    32082 0.0226   109797 0.139
## 10 2013     4  1296613         30369    54946 0.0234   183842 0.142
## 11 2013     5  1164639         28389    51629 0.0244   136720 0.117
## 12 2013     6  1388834         34538    61891 0.0249   107970 0.0777

```

```

#adding a column for a formal date object for ggplot visualizations
month_ag_df$Date<-as.Date(with(month_ag_df,paste(year,month,1,sep="-")), "%Y-%m-%d")

```

```

#plotting monthly sessions in the past year

```

```

p1 <- ggplot(month_ag_df, aes(x=Date, y = sessions)) +
  geom_line(colour = "black", size = 1) +
  geom_point(colour = "black", size = 2) +
  scale_x_date(date_labels= "%Y-%m", date_breaks = "3 months") +
  scale_y_continuous(limits = c(0, max(month_ag_df$sessions*1.1))) +
  labs(x = "Date",
       y = "Total Sessions",
       title = "") +
  theme(plot.margin = margin(0,.5,0,0, "cm"))

```

```

#plotting monthly quantity sold in the past year

```

```

p2 <- ggplot(month_ag_df, aes(x=Date, y = quantity)) +
  geom_line(colour = "black", size = 1) +
  geom_point(colour = "black", size = 2) +
  scale_x_date(date_labels= "%Y-%m", date_breaks = "3 months") +
  scale_y_continuous(limits = c(0, max(month_ag_df$quantity*1.1))) +
  labs(x = "Date",
       y = "Total Quantity Sold",
       title = "") +
  theme(plot.margin = margin(0,.5,0,0, "cm"))

```

```

#plotting monthly ECR in the past year

```

```

p3 <- ggplot(month_ag_df, aes(x=Date, y = ECR)) +
  geom_line(colour = "black", size = 1) +
  geom_point(colour = "black", size = 2) +
  scale_x_date(date_labels= "%Y-%m", date_breaks = "3 months") +
  scale_y_continuous(labels = scales::percent,
                    limits = c(0, max(month_ag_df$ECR*1.1))) +
  labs(x = "Date",
       y = "ECR",

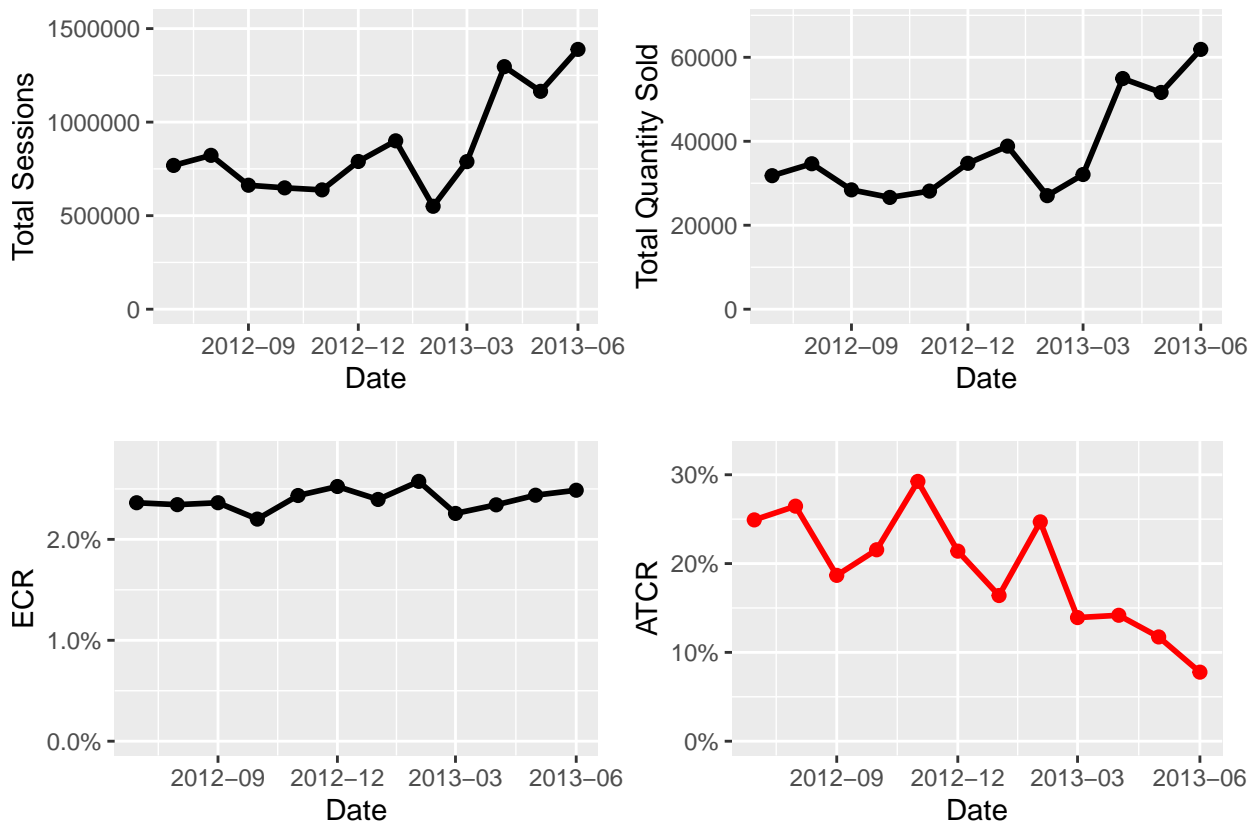
```

```

    title = "") +
  theme(plot.margin = margin(0,.5,0,0, "cm"))
#plotting monthly ATCR in the past year
p4 <- ggplot(month_ag_df, aes(x=Date, y = ATCR)) +
  geom_line(colour = "red", size = 1) +
  geom_point(colour = "red", size = 2) +
  scale_x_date(date_labels= "%Y-%m", date_breaks = "3 months") +
  scale_y_continuous(labels = scales::percent,
                     limits = c(0, max(month_ag_df$ATCR*1.1))) +
  labs(x = "Date",
       y = "ATCR",
       title = "") +
  theme(plot.margin = margin(0,.5,0,0, "cm"))

#plotting all the visuals in a grid
multiplot <- grid.arrange(p1, p2, p3, p4, nrow = 2)

```



```

#finalizing the second deliverable by switching the rows and columns of the data frame
#and adding two additional columns for absolute and relative changes between the past
#2 months
two_month_ag_df <- month_ag_df %>%
  #filtering the two most recent months
  filter(month > 4 & month < 7) %>%
  #removing unnecessary columns
  dplyr::select(-c(year, Date)) %>%

```

```

#reshaping the data to be in a longer format (besides month)
pivot_longer(cols = -month) %>%
#then widening the data and using month as the new columns
pivot_wider(names_from = month) %>%
#renaming columns with clearer names
rename("statistic" = 1, "5/2013" = 2, "6/2013" = 3) %>%
#mutating new columns for absolute and relative changes
mutate("absolute change" = cur_data()[['6/2013']] - cur_data()[['5/2013']]) %>%
mutate("relative change" = `absolute change`/`5/2013`) %>%
as.data.frame()
#verifying the second deliverable is correctly formatted
two_month_ag_df

```

	statistic	5/2013	6/2013	absolute change	relative change
## 1	sessions	1.164639e+06	1.388834e+06	2.241950e+05	0.19250171
## 2	transactions	2.838900e+04	3.453800e+04	6.149000e+03	0.21659798
## 3	quantity	5.162900e+04	6.189100e+04	1.026200e+04	0.19876426
## 4	ECR	2.437579e-02	2.486834e-02	4.925491e-04	0.02020648
## 5	addsToCart	1.367200e+05	1.079700e+05	-2.875000e+04	-0.21028379
## 6	ATCR	1.173926e-01	7.774147e-02	-3.965113e-02	-0.33776514

```

#creating an Excel workbook object
wb = createWorkbook()
#adding two worksheets
sheet1 = addWorksheet(wb, "Month*Device Aggregation")
sheet2 = addWorksheet(wb, "2_Month_Comparison")
#writing the deliverable data to the workbook
writeData(wb, sheet1, device_ag_df)
writeData(wb, sheet2, two_month_ag_df)
#styling relative change as a percentage
addStyle(wb, sheet2, style = createStyle(numFmt = "0%"), cols=5, rows=2:(nrow(month_ag_df)+1), gridExpans
#set column widths so cells fit the data
setColWidths(wb, sheet1, cols = 1:ncol(device_ag_df), widths = "auto")
setColWidths(wb, sheet2, cols = 1:ncol(two_month_ag_df), widths = "auto")

#writing the deliverable
saveWorkbook(wb, "Reference_Tables.xlsx", overwrite = TRUE)

```