An R-based Introduction to Analysing Buyer Behaviour Using Consumer Panel Data

Bruce G.S. Hardie

April 2023

As its title indicates, *An Excel-based Introduction to Analysing Buyer Behaviour Using Consumer Panel Data* (http://brucehardie.com/notes/042/) provides an introduction to basic analyses of buyer behaviour we can undertake using consumer panel data. All the analyses are undertaken in Excel. The advantage of using Excel is that it makes the process completely transparent to all. However, it is not the best environment for repeated analyses.

The objective of this note is to document how to perform these same analyses using R. It assumes you have used R, but is written with a relatively new R user in mind.

A few observations before we start:

- This is not a standalone document. It is assumed that you have worked through the Excel note. You should work through the material in this note with the Excel note at your side.
- When working through this document, do not copy and paste the code. Typing it out for yourself is part of the learning process.
- As any user of R quickly learns, there are frequently multiple ways of performing a particular piece of analysis. The approaches taken here should not be viewed as definitive.
- We are dealing with two small datasets. No attention has been paid to performance issues.
- A conscious decision has been made to use base R and not the 'tidyverse'.[1]
- A conscious decision has been made to use only the functionality built into base R. (This includes the functionality of those packages automatically loaded in a standard installation of R.) Our objective is to learn the logic and "mechanics" of the calculations. If you find yourself performing these types of analyses on a regular basis, you will definitely want to write your own functions and/or make use of some other packages. The `data.table` package is an obvious choice.

---

[1] See https://matloff.wordpress.com/2022/08/24/base-r-and-tidyverse-code-side-by-side/ for a good reflection on the case for focusing on base R for those coming to R without a coding background.

- All the plots are created using the base graphics system. They are not intended to be publication ready. We will not use the `ggplot2` package as its syntax can be a bit puzzling if you are a beginner. A good reference on R graphics is

  Murrell, Paul (2019), *R Graphics*, 3rd edn, Boca Raton, FL: CRC Press.

- Given our focus on how to perform the calculations, we will not consider how to create nicely formatted tables, etc. The numbers that we would choose to report in a document will be left in a data frame, matrix, or table.

This document can be found at http://brucehardie.com/notes/043/. The data files can be found at http://brucehardie.com/notes/042/.

Chapter 3 Analyses

We start by loading the two csv files:

```
df_edible_grocery <- read.csv("C:/Users/bhard/Desktop/edible_grocery.csv",
                              fileEncoding = "UTF-8-BOM")
df_sku_weight <- read.csv("C:/Users/bhard/Desktop/sku_weight.csv",
                          fileEncoding = "UTF-8-BOM")
```

> **[Optional] Technical question**
> Why do we need `fileEncoding = "UTF-8-BOM"`?

Next, we merge the two files so that we know the weight of each SKU:

```
df <- merge(df_sku_weight, df_edible_grocery, by = "sku_id")
```

> **[Optional] Technical aside**
> We have performed what is called an inner join. How is this different from a left outer join, right outer join, and full outer join?

The volume purchased and spend variables are created in the following manner. We also create a year variable.

```
df <- within(df, {
             volume <- units * weight / 1000
             spend <- units * price
             year <- floor((week - 1) / 52) + 1
}
)
```

We declare the panel size:

```
num_panellists <- 5021
```

The next step is to compute weekly revenue by brand:

```r
df_tmp <- aggregate(spend ~ week + brand,
                    data = df,
                    FUN = sum)
```

This new data frame has a so-called long format. We would like to reshape it to a so-called wide format, where the rows correspond to weeks and the columns correspond to brands.

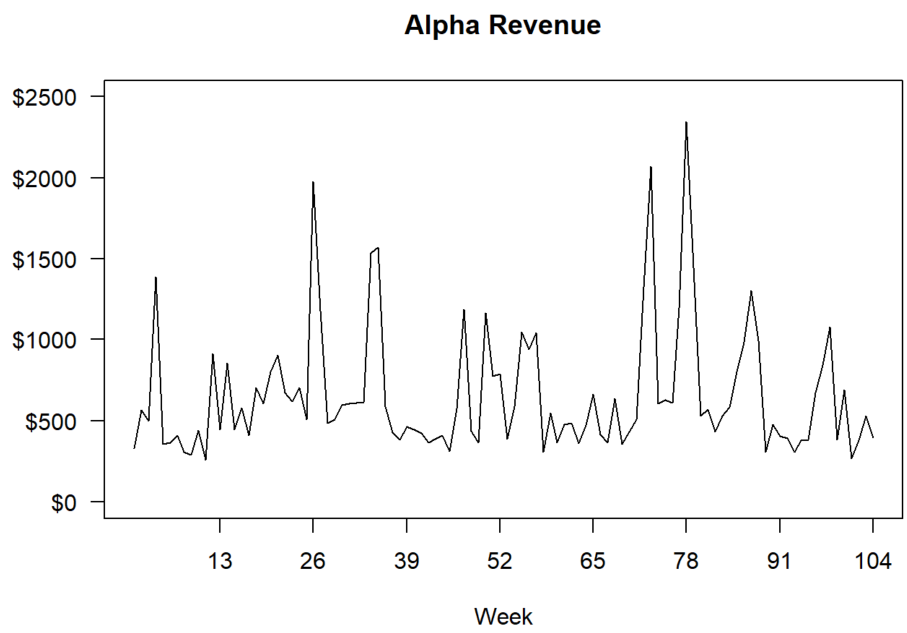
```r
df_weekly_rev <- reshape(data = df_tmp,
                         idvar = "week",
                         v.names = "spend",
                         timevar = "brand",
                         direction = "wide")
```

We add column names and compute category revenue.

```r
colnames(df_weekly_rev)[-1] <- c("alpha", "bravo", "charlie", "delta",
                                 "other")
df_weekly_rev$category <- rowSums(df_weekly_rev[, c(2:6)])
rm(df_tmp)
```

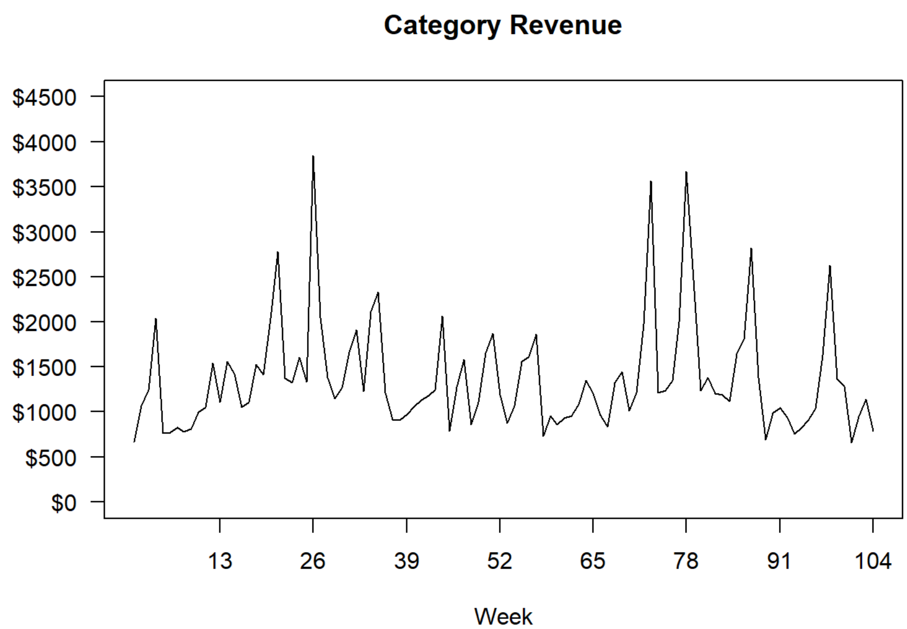
We create a plot of weekly dollar sales for Alpha in the following manner.

```r
with(df_weekly_rev,
     plot(week, alpha,
          type = "l",
          main = "Alpha Revenue",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 2500)
          )
     )
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = seq(0, 2500, by = 500),
     labels = c("$0", "$500", "$1000", "$1500", "$2000", "$2500"),
     las = 1)
```



Minor changes to the code generates a plot of weekly category revenue.

```
with(df_weekly_rev,
     plot(week, category,
          type = "l",
          main = "Category Revenue",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 4500)
          )
     )
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = seq(0, 4500, by = 500),
     labels = c("$0", "$500", "$1000", "$1500", "$2000", "$2500",
                "$3000", "$3500", "$4000", "$4500"),
     las = 1)
```



We compute weekly volume sales by brand in a similar manner to that used above for revenue.

```
df_tmp <- aggregate(volume ~ week + brand,
                    data = df,
                    FUN=sum)
df_weekly_vol <- reshape(data = df_tmp,
                         idvar = "week",
                         v.names = "volume",
                         timevar = "brand",
                         direction = "wide")

colnames(df_weekly_vol)[-1] <- c("alpha", "bravo", "charlie", "delta",
                                  "other")
df_weekly_vol$category <- rowSums(df_weekly_vol[, c(2:6)])
rm(df_tmp)
```

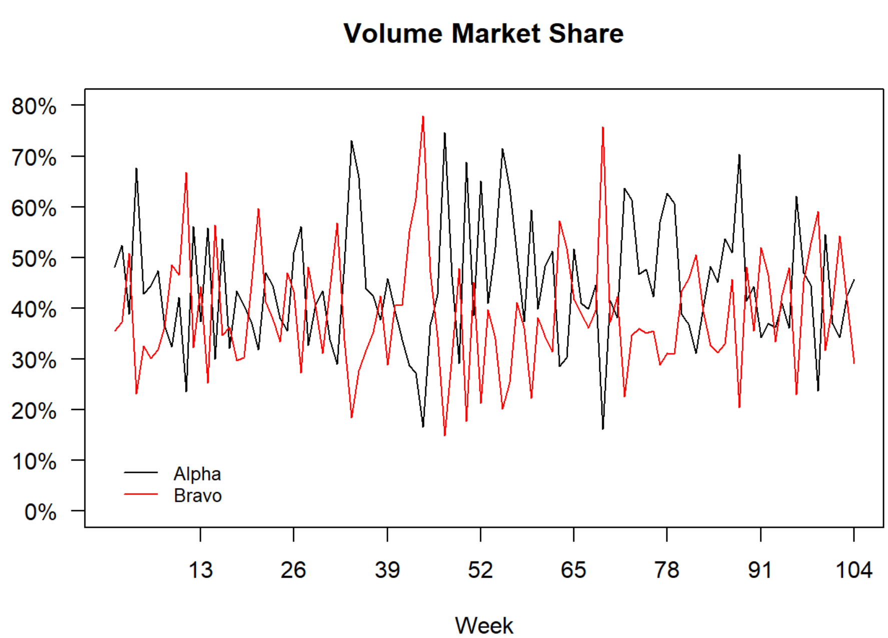
We compute weekly volume shares in the following manner.

```
df_vol_share <- 100 * df_weekly_vol[, c(2:6)] / df_weekly_vol[, 7]
df_vol_share$week <- df_weekly_vol$week
```

The following code generates a plot of weekly volume market share for Alpha and Beta.

```
with(df_vol_share,
     plot(week, alpha,
          type = "l",
          main = "Volume Market Share",
          xaxt = "n",
          yaxt = "n",
```

```
          xlab = "Week",
          ylab = "",
          ylim = c(0, 80)
          )
     )

with(df_vol_share,
     lines(week, bravo,
           col = "red"
          )
     )

axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = seq(0, 80, by = 10),
     labels = c("0%", "10%", "20%", "30%", "40%", "50%", "60%", "70%",
                "80%"),
     las = 1)

legend(x = 0, y = 12,
       legend = c("Alpha", "Bravo"),
       lty = 1:1,
       col = c("black", "red"),
       cex = 0.8,
       box.lty = 0
       )
```



We compute the total revenue by year at both the brand and category level, and compute the percentage change across the two years.

```r
tmp <- with(df,
            tapply(spend, list(year, brand), sum)
            )
annual_tot_rev <- cbind(tmp, rowSums(tmp))
colnames(annual_tot_rev)[6] <- "Category"
annual_tot_rev

     Alpha    Bravo Charlie   Delta   Other Category
1 33570.94 28603.35 5120.87 3271.51 1535.23 72101.90
2 35250.75 26926.87 3922.68 2820.81 1739.82 70660.93

100 * (annual_tot_rev[2, ] / annual_tot_rev[1, ] - 1)

     Alpha      Bravo    Charlie      Delta      Other   Category
  5.003762  -5.861132 -23.398173 -13.776513  13.326342  -1.998519
```

We compute each brand's dollar market share by year, and compute the percentage change across the two years.

```r
dollar_mkt_share <- 100 * annual_tot_rev[, -6] / annual_tot_rev[, 6]
dollar_mkt_share

     Alpha    Bravo  Charlie    Delta    Other
1 46.56041 39.67073 7.102268 4.537342 2.129250
2 49.88719 38.10715 5.551413 3.992036 2.462209

100 * (dollar_mkt_share[2, ] / dollar_mkt_share[1, ] - 1)

     Alpha      Bravo    Charlie      Delta      Other
  7.145077  -3.941382 -21.836051 -12.018180  15.637377
```

Chapter 4 Analyses

Creating the required datasets

We first need to create datasets that summarise each panellist's brand and category purchasing. We will create a separate dataset for transactions, spend, and volume purchasing, focusing on the first year.

Let's start with the transaction summary. We aggregate the purchase records to the transaction level:

```r
df_tmp1 <- aggregate(panel_id ~ trans_id + brand,
                     data = df[df$year == 1, ],
                     FUN = max
                     )
```

Next we determine the numbers of transaction occasions on which each brand was purchased by each panellist.

```
df_tmp2 <- aggregate(trans_id ~ panel_id + brand,
                     data = df_tmp1,
                     FUN = length
                     )
```

We reshape the resulting data frame and replace missing values with 0.

```
df_tmp3 <- reshape(data = df_tmp2,
                   idvar = "panel_id",
                   v.names = "trans_id",
                   timevar = "brand",
                   direction = "wide"
                   )
df_tmp3[is.na(df_tmp3)] <- 0
```

Finally, we determine the number of category transactions made by each panellist and merge this with the brand-level summary to create our final transaction dataset.

```
df_tmp4 <- aggregate(trans_id ~ panel_id,
                      data = df_tmp1,
                      function(x) length(unique(x))
                      )

df_panellist_trans <- merge(df_tmp3, df_tmp4)
colnames(df_panellist_trans)[-1] <- c("alpha", "bravo", "charlie", "delta",
                                      "other", "category")
rm("df_tmp1", "df_tmp2", "df_tmp3", "df_tmp4")
```

Creating the spend summary is much easier, as i) we can simply sum up the spend associated with each row of `df` by panellist id and brand, and ii) category spend is simply the sum of brand spend.

```
df_tmp <- aggregate(spend ~ panel_id + brand,
                     data = df[df$year == 1, ],
                     FUN = sum
)
df_panellist_spend <- reshape(data = df_tmp,
                              idvar = "panel_id",
                              v.names = "spend",
                              timevar = "brand",
                              direction = "wide"
)
df_panellist_spend[is.na(df_panellist_spend)] <- 0
colnames(df_panellist_spend)[-1] <- c("alpha", "bravo", "charlie", "delta",
                                      "other")
df_panellist_spend$category <- rowSums(df_panellist_spend[, c(2:6)])
rm("df_tmp")
```

The year 1 summary of each panellist’s volume purchasing by brand is created in the same manner.

```
df_tmp <- aggregate(volume ~ panel_id + brand,
                    data = df[df$year == 1, ],
                    FUN = sum
)
df_panellist_vol <- reshape(data = df_tmp,
                            idvar = "panel_id",
                            v.names = "volume",
                            timevar = "brand",
                            direction = "wide"
)
df_panellist_vol[is.na(df_panellist_vol)] <- 0
colnames(df_panellist_vol)[-1] <- c("alpha", "bravo", "charlie", "delta",
                                    "other")
df_panellist_vol$category <- rowSums(df_panellist_vol[, c(2:6)])
rm("df_tmp")
```

> **[Optional] Checking our work to date**
>
> As a necessary (but not sufficient) check that the datasets we've created match those we created in Excel, let's see if the total brand and category numbers match those from the Excel datasets.
>
> ```
> colSums(df_panellist_trans[, 2:7])
>
>    alpha    bravo  charlie    delta    other category
>     9060     8255     1882      859      422    20030
>
> colSums(df_panellist_spend[, 2:7])
>
>    alpha    bravo  charlie    delta    other category
> 33570.94 28603.35  5120.87  3271.51  1535.23 72101.90
>
> colSums(df_panellist_vol[, 2:7])
>
>     alpha     bravo   charlie     delta     other  category
>  9166.250  8240.350  2171.125   921.000   286.275 20785.000
> ```
>
> They do.

Examining purchase frequency

The penetration and purchases per buyer (PPB) numbers are computed as follows

```
tot_trans <-colSums(df_panellist_trans[, 2:7])
num_buyers <- colSums(df_panellist_trans[, 2:7] != 0)
penetration <- 100 * num_buyers / num_panellists
ppb <- tot_trans / num_buyers

round(penetration, digits = 1)

   alpha    bravo  charlie    delta    other category
    52.3     51.0     16.2      7.6      3.5     91.1
```

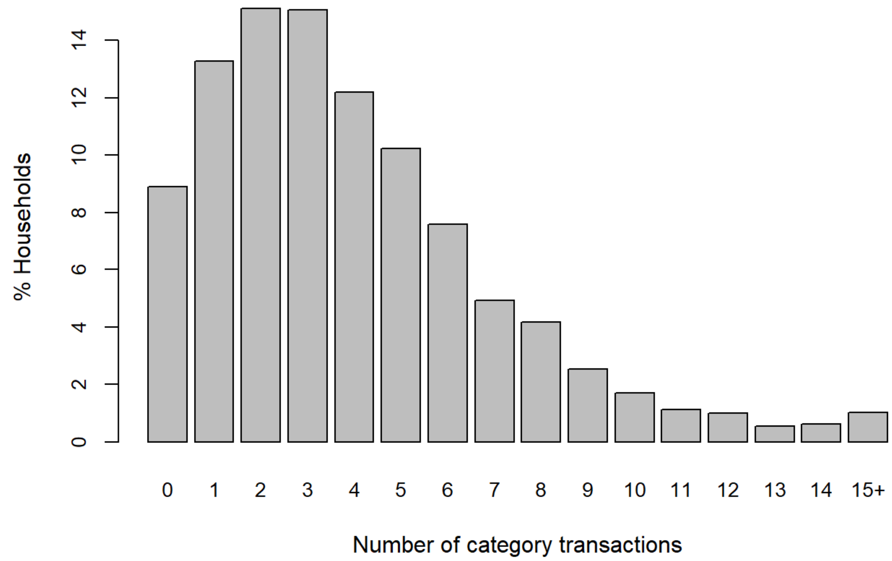
```
round(ppb, digits = 2)

   alpha    bravo  charlie    delta    other category
    3.45     3.22     2.31     2.26     2.40     4.38
```

We wish to create of plot of the distribution of category purchase frequency. First we create the frequency distribution of category purchasing.

```
df_tmp <- aggregate(panel_id ~ category,
                      data = df_panellist_trans,
                      length
                      )
colnames(df_tmp) <- c("num_trans", "freq")
df_freq_cat <- rbind(data.frame(num_trans = 0, freq = num_panellists -
sum(df_tmp$freq)),
                  df_tmp)
rm("df_tmp")
```

We right censor the distribution at 15 and plot the percentage of panellists making 0, 1, 2, ..., 15+ category purchases.

```
tmp <- df_freq_cat[1:15, 2]
tmp[16] <- sum(df_freq_cat[16:nrow(df_freq_cat), 2])

barplot(100 * tmp / num_panellists,
        cex.names = 0.89,
        cex.axis = 0.89,
        names.arg = c(c(0:14), "15+"),
        xlab = "Number of category transactions",
        ylab = "% Households"
        )
```



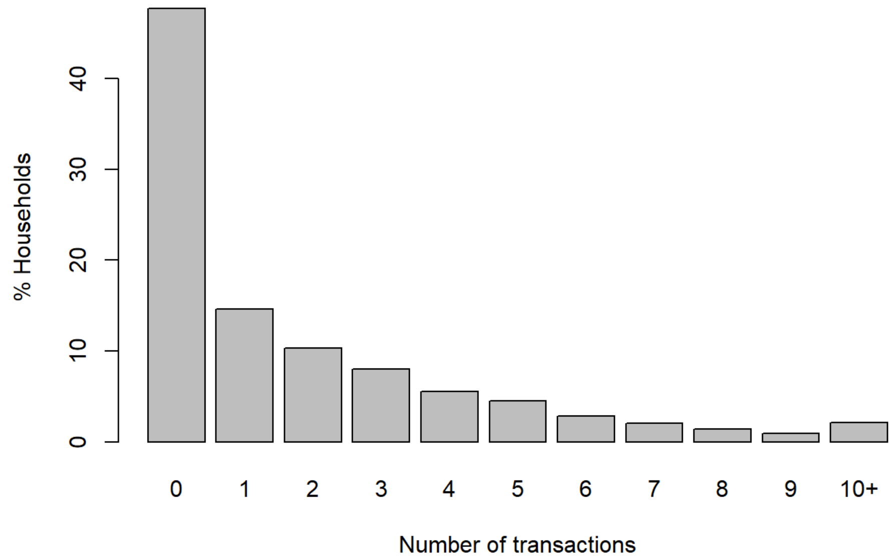
> **Technical aside**
>
> The `cex.names = 0.89` option is required to rescale the x-axis labels so that the 15+ label is plotted in this document. It is not needed when executing the code in RStudio with a large monitor. We add `cex.axis = 0.89` to make the y-axis labels the same size as those of the x-axis.

We wish to create of plot of the distribution of the number of purchase occasions on which Alpha was purchased. First we create the frequency distribution of Alpha purchasing.

```
df_freq_alpha <- aggregate(panel_id ~ alpha,
                           data = df_panellist_trans,
                           length
)
colnames(df_freq_alpha) <- c("num_trans", "freq")
df_freq_alpha[1,2] <- num_panellists -
sum(df_freq_alpha[2:nrow(df_freq_alpha), 2])
```

We right censor the distribution at 10 and plot the percentage of panellists that purchased Alpha on 0, 1, 2, ..., 10+ (category) purchase occasions.

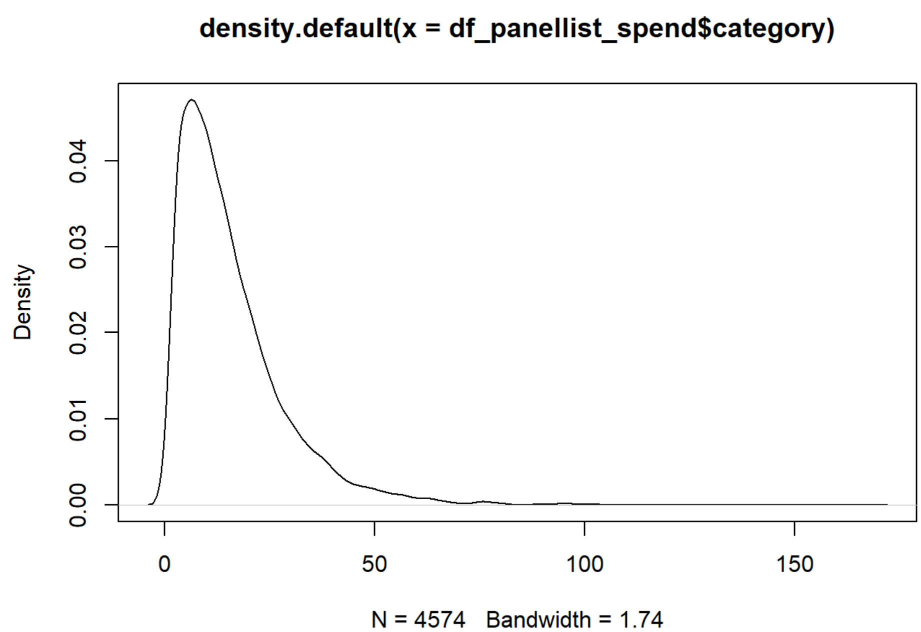
```
tmp <- df_freq_alpha[1:10, 2]
tmp[11] <- sum(df_freq_alpha[11:nrow(df_freq_alpha), 2])

barplot(100 * tmp / num_panellists,
        names.arg = c(c(0:9), "10+"),
        xlab = "Number of transactions",
        ylab = "% Households"
)
```



Examining spend

We wish to visualise the variability in category spend. Given this objective, some readers would automatically think of creating a (kernel) density plot.

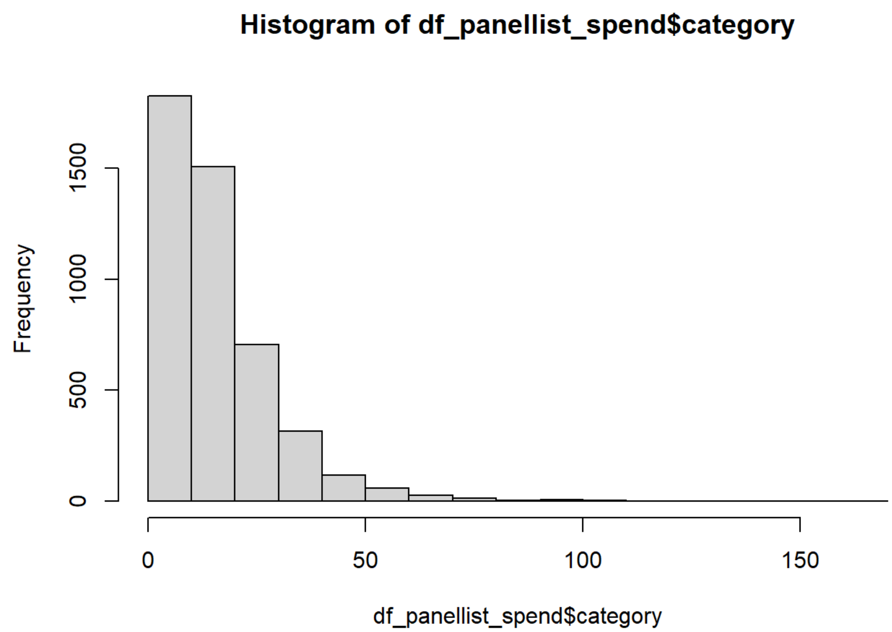
```
plot(density(df_panellist_spend$category))
```



While this provides a good visualisation of the shape of the distribution, it can be difficult for most "consumers" of the plot to extract some additional information that may be of interest. For example, it is not easy to answer the question "What percentage of category buyers spent $30 or less in year 1?".

One possible solution is to plot a histogram.

```
hist(df_panellist_spend$category)
```



The distributions of many customer behaviours have a long right tail. Accommodating the range of values can make it difficult to get a clear sense of what is happening on the left side of the distribution. It can therefore be helpful to bin the data (as with a histogram) but to right censor the data, assigning all of the observations with a value of x or higher to an $x+$ bin. We plot create a bar chart of the associated frequencies.

We compute key summary stats in the following manner.

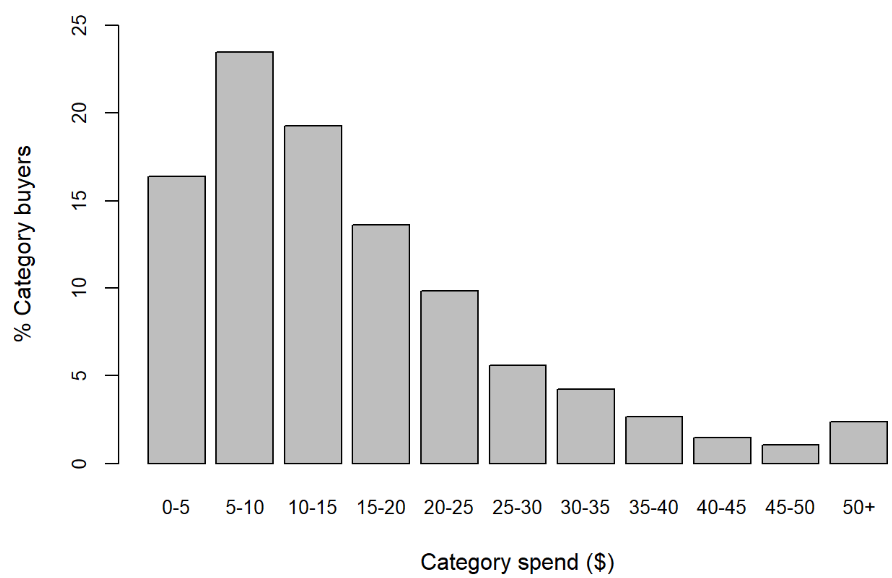
```
summary(df_panellist_spend$category)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.27    6.76   12.57   15.76   20.74  166.70

quantile(df_panellist_spend$category, probs = seq(0, 1, 0.05))

      0%       5%      10%      15%      20%      25%      30%      35%
  1.2700   2.6900   3.3900   4.6700   5.8800   6.7600   7.6340   9.0655
     40%      45%      50%      55%      60%      65%      70%      75%
 10.0600  11.0285  12.5700  13.7300  15.2400  16.8400  18.7900  20.7375
     80%      85%      90%      95%     100%
 23.4300  26.8805  31.8100  39.7185 166.7000
```

We bin the data into bins of width $5 with a $50+ bin,

```
boundaries <- c(seq(0, 50, 5), max(df_panellist_spend$category) + 1)
tmp <- cut(df_panellist_spend$category, breaks = boundaries)
```

and plot the associated relative frequencies.

```
barplot(100 * table(tmp) / sum(table(tmp)),
        cex.names = 0.85,
        cex.axis = 0.85,
        names.arg = c("0-5", "5-10", "10-15", "15-20", "20-25", "25-30",
                      "30-35", "35-40", "40-45", "45-50", "50+"),
        ylim = c(0, 25),
        xlab = "Category spend ($)",
        ylab = "% Category buyers"
        )
```



Technical aside

Let's look at the distribution of spend,

```
table(tmp)

tmp
   (0,5]   (5,10]  (10,15]  (15,20]  (20,25]  (25,30]  (30,35]  (35,40] 
     750     1073      882      623      450      256      193      122 
 (40,45]  (45,50] (50,168] 
      68       48      109 
```

and compare it to the distribution we created in Excel. The numbers match up except for two bins. In R we get 193 and 122, while in Excel we get 192 and 123. What's going on? Notice that the boundary of these two categories is 35. We have one panellist that spent $35 in year 1:

```
df_panellist_spend$panel_id[df_panellist_spend$category == 35]

[1] 3116045
```

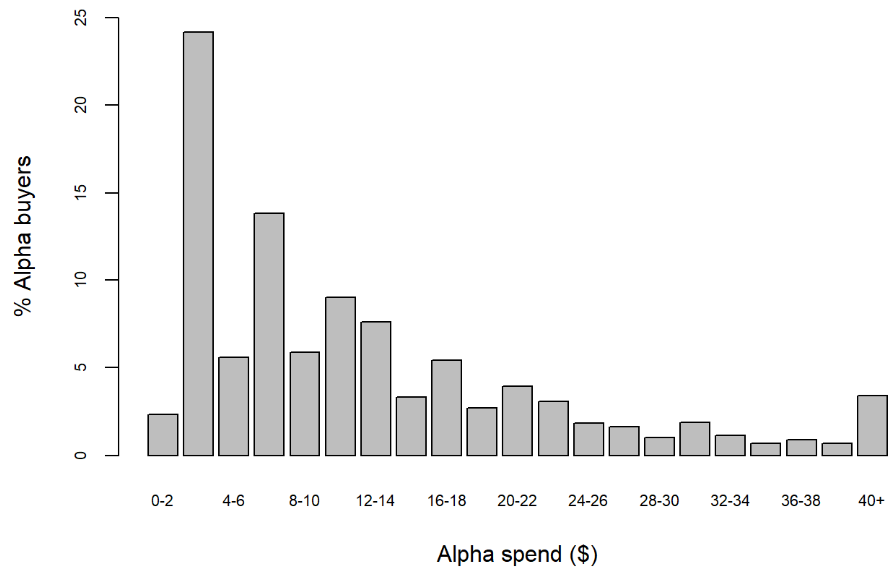
When defining intervals, we will see both parentheses, ( ), and square brackets, [ ], being used. The notation (a, b] is used to indicate an interval from a to b that excludes a but includes b. R includes this person in the $30-35 bin. Excel is putting them in the $35-40 bin.

Minor changes to this code gives us a plot of the distribution of spend on Alpha with $2-wide bins and a $40+ bin.

```
boundaries <- c(seq(0, 40, 2), max(df_panellist_spend$alpha) + 1)
tmp <- cut(df_panellist_spend$alpha[df_panellist_spend$alpha > 0],
           breaks = boundaries)
```

Note the use of `[df_panellist_spend$alpha > 0]` to exclude those panellists that had zero spend on Alpha in year 1.

```
barplot(100 * table(tmp) / sum(table(tmp)),
        cex.names = 0.675,
        cex.axis = 0.675,
        names.arg = c("0-2", "", "4-6", "", "8-10", "", "12-14", "", "16-18",
                      "", "20-22", "", "24-26", "", "28-30", "", "32-34", "",
                      "36-38", "", "40+"),
        ylim = c(0, 25),
        xlab = "Alpha spend ($)",
        ylab = "% Alpha buyers"
        )
```



Basic decile analysis

We first create a dataset that reports, for each category buyer in year 1, the number of category transactions, total category spend, and the unique number of brands purchased.

Before doing so, let's check that the three source datasets have the same ordering by panellist id.

```
identical(df_panellist_spend$panel_id,df_panellist_vol$panel_id)

[1] TRUE

identical(df_panellist_trans$panel_id,df_panellist_vol$panel_id)

[1] FALSE

identical(df_panellist_trans$panel_id,df_panellist_spend$panel_id)

[1] FALSE
```

> **[Optional] Digging deeper into R**
> Why is it that `df_panellist_trans` is sorted by `panel_id` but `df_panellist_spend` and `df_panellist_vol` are not?

Let's sort `df_panellist_spend` and `df_panellist_vol` by `panel_id`.

```
df_panellist_spend <- df_panellist_spend[order(df_panellist_spend$panel_id),
]
df_panellist_vol <- df_panellist_vol[order(df_panellist_vol$panel_id), ]
```

```
identical(df_panellist_spend$panel_id,df_panellist_vol$panel_id)

[1] TRUE

identical(df_panellist_trans$panel_id,df_panellist_vol$panel_id)

[1] TRUE

identical(df_panellist_trans$panel_id,df_panellist_spend$panel_id)

[1] TRUE
```

We can now create the desired summary dataset.

```
num_brands <- rowSums(df_panellist_trans[, c(2:6)] > 0)

df_panellist_cat_sum <- as.data.frame(cbind(df_panellist_trans$panel_id,
                                         df_panellist_trans$category,
                                         df_panellist_spend$category,
                                         num_brands
                                         )
)
colnames(df_panellist_cat_sum) <- c("panel_id", "trans", "spend",
                                    "num_brands")
```

We create a rank number where a rank of 1 is assigned to the biggest spender, and convert the rank into a decile number, where the first decile represents the highest spending 10% of customers.

```
df_panellist_cat_sum$rank <- rank(-df_panellist_cat_sum$spend,
                                  ties.method = "first")
df_panellist_cat_sum$decile <- floor(10 * (df_panellist_cat_sum$rank - 1) /
                                       length(df_panellist_cat_sum$rank)) + 1
df_panellist_cat_sum$count <- 1
```

Next, we create a summary of the key variables by decile.

```
df_decile_tots <- aggregate(cbind(trans, spend, num_brands, count) ~ decile,
                            df_panellist_cat_sum, FUN = "sum")
```

We can now create the entries for our decile table.

```
df_decile_sum <- df_decile_tots["decile"]
df_decile_sum$pct_hh <- 100 * df_decile_tots$count /
  sum(df_decile_tots$count)
df_decile_sum$pct_spend <- 100 * df_decile_tots$spend /
  sum(df_decile_tots$spend)
df_decile_sum$pct_trans <- 100 * df_decile_tots$trans /
  sum(df_decile_tots$trans)
df_decile_sum$spend_hh <- df_decile_tots$spend / df_decile_tots$count
df_decile_sum$cat_trans_hh <- df_decile_tots$trans / df_decile_tots$count
```

```
df_decile_sum$aov <- df_decile_tots$spend / df_decile_tots$trans
df_decile_sum$avg_brands <- df_decile_tots$num_brands /
  df_decile_tots$count

df_decile_sum

   decile    pct_hh pct_spend pct_trans  spend_hh cat_trans_hh      aov
1       1 10.013118 28.361250 24.278582 44.648472    10.617904 4.205017
2       2  9.991255 17.206689 16.310534 27.147374     7.148796 3.797475
3       3 10.013118 13.315363 13.444833 20.962074     5.879913 3.565032
4       4  9.991255 10.716070 11.108337 16.906980     4.868709 3.472580
5       5  9.991255  8.754984  9.470794 13.812932     4.150985 3.327628
6       6 10.013118  7.074127  7.783325 11.136638     3.403930 3.271700
7       7  9.991255  5.689614  6.709935  8.976630     2.940919 3.052321
8       8 10.013118  4.281593  4.877683  6.740415     2.133188 3.159785
9       9  9.991255  2.888079  3.689466  4.556586     1.617068 2.817808
10     10  9.991255  1.712229  2.326510  2.701422     1.019694 2.649249
   avg_brands
1    1.847162
2    1.636761
3    1.582969
4    1.540481
5    1.501094
6    1.410480
7    1.374179
8    1.224891
9    1.212254
10   1.000000
```

This decile analysis uses deciles that represent 10% of the category buyers. An alternative approach is to create deciles that represent 10% of category spend. The only change to what we have done above is how we create the decile variable.

We start by recreating `df_panellist_cat_sum`.

```
df_panellist_cat_sum <- as.data.frame(cbind(df_panellist_trans$panel_id,
                                            df_panellist_trans$category,
                                            df_panellist_spend$category,
                                            num_brands
)
)
colnames(df_panellist_cat_sum) <- c("panel_id", "trans", "spend",
                                    "num_brands")
df_panellist_cat_sum$count <- 1
```

We sort the dataset by category spend, from highest to lowest.

```
df_panellist_cat_sum <- df_panellist_cat_sum[order(
  -df_panellist_cat_sum$spend), ]
```

Next we create a variable that reports the percentage of total spend accounted for by this customer and those customers that spent more than this customer in year 1.

```
df_panellist_cat_sum$cum <- 100 * cumsum(df_panellist_cat_sum$spend) /
  sum((df_panellist_cat_sum$spend))
```

This variable is converted to a decile number.

```
df_panellist_cat_sum$decile <- floor((df_panellist_cat_sum$cum - 1e-6) / 10)
+ 1
```

> **[Optional] Technical aside**
> Why are we subtracting 1e-6? When working with other datasets, you should not blindly subtract this number. How would you determine whether it is OK to use this number or whether you should use a smaller number?

All the other calculations are as for our first decile table.

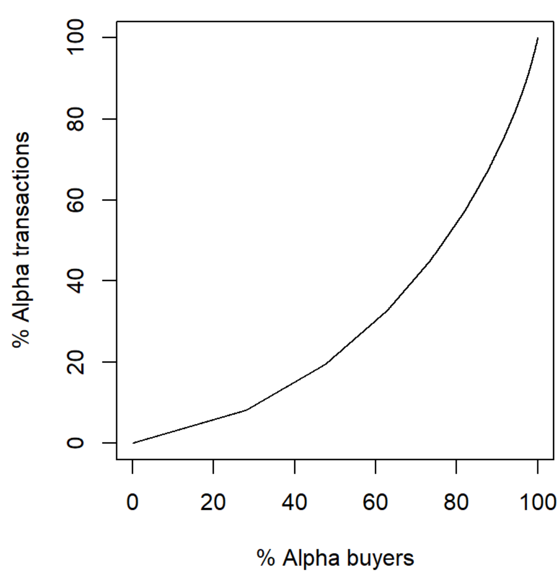
```
df_decile_tots <- aggregate(cbind(trans, spend, num_brands, count) ~ decile,
                            df_panellist_cat_sum, FUN = "sum")

df_decile_sum <- df_decile_tots["decile"]
df_decile_sum$pct_hh <- 100 * df_decile_tots$count /
  sum(df_decile_tots$count)
df_decile_sum$pct_spend <- 100 * df_decile_tots$spend /
  sum(df_decile_tots$spend)
df_decile_sum$pct_trans <- 100 * df_decile_tots$trans /
  sum(df_decile_tots$trans)
df_decile_sum$spend_hh <- df_decile_tots$spend / df_decile_tots$count
df_decile_sum$cat_trans_hh <- df_decile_tots$trans / df_decile_tots$count
df_decile_sum$aov <- df_decile_tots$spend / df_decile_tots$trans
df_decile_sum$avg_brands <- df_decile_tots$num_brands / df_decile_tots$count

df_decile_sum

   decile    pct_hh pct_spend pct_trans  spend_hh cat_trans_hh      aov
1       1  2.470485  9.994591  7.583625 63.772478    13.442478 4.744101
2       2  3.716659  9.981096  8.911633 42.332706    10.500000 4.031686
3       3  4.634893  9.988724  9.241138 33.971981     8.731132 3.890902
4       4  5.596852 10.017683  9.410884 28.214609     7.363281 3.831798
5       5  6.646261  9.989251  9.860210 23.692237     6.496711 3.646805
6       6  7.892436 10.020554 10.214678 20.013878     5.667590 3.531285
7       7  9.466550 10.007323 10.369446 16.663903     4.796767 3.473987
8       8 11.674683  9.991082 10.873689 13.490187     4.078652 3.307511
9       9 15.675557 10.003204 11.228158 10.059275     3.136681 3.206981
10     10 32.225623 10.006491 12.306540  4.894756     1.672320 2.926925

   avg_brands
1    1.920354
2    1.829412
3    1.778302
```

```
4     1.628906
5     1.648026
6     1.567867
7     1.540416
8     1.479401
9     1.389121
10    1.162144
```

Creating Lorenz curves

We create the Lorenz curve for (Alpha) transactions using the logic associated with the spend Lorenz curve in the Excel note.

```
sorted_trans <- sort(df_panellist_trans$alpha[df_panellist_trans$alpha > 0])
pct_trans  <- 100 * cumsum(sorted_trans) / sum(sorted_trans)
pct_buyers <- 100 * seq(1, length(pct_trans)) / length(pct_trans)

par(pty="s")
plot(pct_buyers, pct_trans,
     type = "l",
     xlab = "% Alpha buyers",
     ylab = "% Alpha transactions",
     xlim = c(0, 100),
     ylim = c(0, 100)
     )
```



What is the value of x/20?

```
min(pct_trans[pct_buyers >= 80])
```

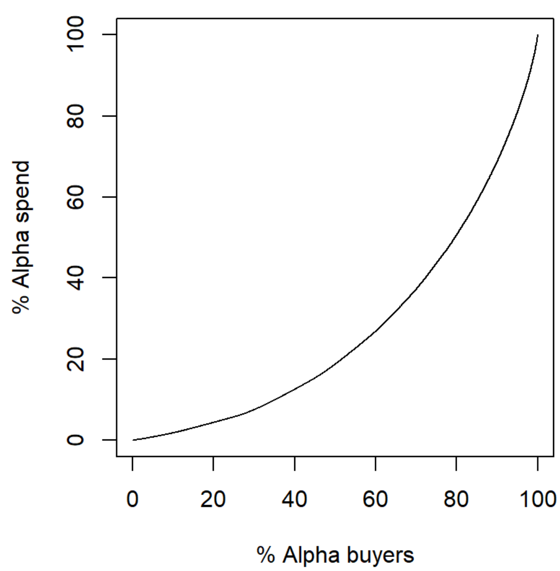
```
[1] 54.52539
```

What is the value of 50/y?

```
100 - min(pct_buyers[pct_trans >= 50])

[1] 23.09451
```

The Lorenz curve for (Alpha) spend is created in the same manner.

```
sorted_spend <- sort(df_panellist_spend$alpha[df_panellist_spend$alpha > 0])
pct_spend  <- 100 * cumsum(sorted_spend) / sum(sorted_spend)
pct_buyers <- 100 * seq(1, length(pct_spend)) / length(pct_spend)

par(pty="s")
plot(pct_buyers, pct_spend,
     type = "l",
     xlab = "% Alpha buyers",
     ylab = "% Alpha spend",
     xlim = c(0, 100),
     ylim = c(0, 100)
     )
```




```
min(pct_spend[pct_buyers >= 80])

[1] 50.883

100 - min(pct_buyers[pct_spend >= 50])

[1] 20.50305
```

Chapter 5 Analyses

Our analysis of multibrand buying behaviour in year 1 makes use of the following three datasets created above: `df_panellist_trans`, `df_panellist_spend`, and `df_panellist_vol`.

Before we undertaken any further analysis, let's check that these three source datasets have the same ordering by panellist id.

```
identical(df_panellist_spend$panel_id,df_panellist_vol$panel_id)

[1] TRUE

identical(df_panellist_trans$panel_id,df_panellist_vol$panel_id)

[1] TRUE

identical(df_panellist_trans$panel_id,df_panellist_spend$panel_id)

[1] TRUE
```

We create the distribution of the number of separate brands purchased by category buyers in year 1.

```
num_brands <- rowSums(df_panellist_trans[, c(2:6)] > 0)
100 * table(num_brands) / length(num_brands)

num_brands
         1          2          3          4
64.8010494 27.9405334  6.4057718  0.8526454
```

We determine the number of different brands purchased in the year as a function of the number of category purchases made during the year.

```
num_brands_by_cat_trans <- table(df_panellist_trans$category, num_brands)
num_brands_by_cat_trans

    num_brands
       1   2   3   4
  1  655  12   0   0
  2  573 184   2   0
  3  516 216  24   1
  4  370 193  47   2
  5  278 189  43   4
  6  196 124  53   8
  7  121  93  28   5
  8   87  94  25   3
  9   51  51  22   3
  10  42  34   9   0
  11  24  24   6   3
```

```
  12  16  21   9   4
  13  11   9   6   1
  14  13   9   8   1
  15   4   8   3   0
  16   3   6   4   3
  17   1   3   3   0
  18   1   3   0   0
  19   0   2   0   0
  20   2   1   1   0
  22   0   1   0   0
  25   0   1   0   0
  27   0   0   0   1
```

We compute the average number of brands purchased for each level of category purchasing.

```
rowSums(num_brands_by_cat_trans %*% diag(c(1:4))) /
  rowSums(num_brands_by_cat_trans)

       1        2        3        4        5        6        7        8
1.017991 1.247694 1.352708 1.478758 1.558366 1.666667 1.663968 1.732057
       9       10       11       12       13       14       15       16
1.818898 1.611765 1.789474 2.020000 1.888889 1.903226 1.933333 2.437500
      17       18       19       20       22       25       27
2.285714 1.750000 2.000000 1.750000 2.000000 2.000000 4.000000
```

Duplication of purchase

We create the duplication of purchase table.

```
ever_buyers <- 1 * as.matrix(df_panellist_trans[, 2:6] > 0)
duplication_counts <- t(ever_buyers) %*% ever_buyers
dop <- 100*diag(1 / diag(duplication_counts)) %*% duplication_counts
diag(dop) <- NA
rownames(dop) <- colnames(dop)
dop

           alpha    bravo  charlie     delta    other
alpha         NA 34.14634 15.35823  9.108232 2.629573
bravo   34.97268       NA 14.91023  5.464481 4.293521
charlie 49.56950 46.98647       NA 14.268143 3.198032
delta   62.89474 36.84211 30.52632        NA 2.631579
other   39.20455 62.50000 14.77273  5.681818       NA
```

Share of category requirements (SCR)

We compute each brand's SCR.

```
brand_purchasing <- colSums(df_panellist_vol[, c(2:6)])
category_purchasing <- colSums(ever_buyers * (df_panellist_vol[,
"category"]))
scr <- 100 * brand_purchasing / category_purchasing
scr

   alpha    bravo  charlie    delta    other
68.84153 67.98282 45.42459 40.44085 29.21770
```

Cross purchasing

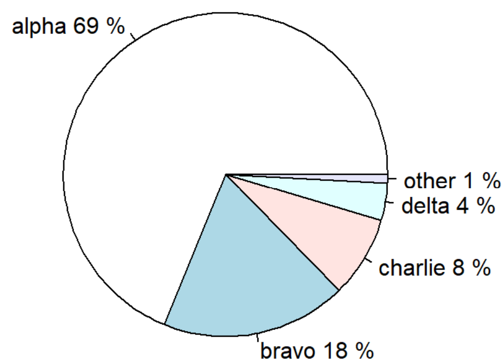
We create the cross purchasing analysis for year 1.

```
tmp <- t(ever_buyers) %*% as.matrix(df_panellist_vol[, c(2:6)])
cross_purchasing <- 100 * tmp /colSums(ever_buyers *
(df_panellist_vol[,"category"]))
rownames(cross_purchasing) <- colnames(cross_purchasing)
cross_purchasing

           alpha    bravo   charlie     delta      other
alpha   68.84153 18.48592  8.083552  3.781449  0.8075479
bravo   20.80483 67.98282  7.485011  2.322373  1.4049735
charlie 25.55640 23.65771 45.424589  4.608102  0.7531972
delta   30.76096 13.64056 14.574734 40.440854  0.5829016
other   22.74954 40.56950  5.447540  2.015717 29.2176975
```

We create the importance of competition plot for Alpha.

```
pie_labels <- paste(colnames(cross_purchasing),
                    round(cross_purchasing[1, ], 0), "%")
pie(cross_purchasing[1, ],
    labels = pie_labels,
    main = "Important of Competition to Buyers of Alpha"
    )
```

Important of Competition to Buyers of Alpha



We create the importance against expectation plot for Alpha. First we compute the (volume) market share across all buyers. (The volume market share for each brand just for buyers of Alpha is given in the first row of `cross_purchasing`.)

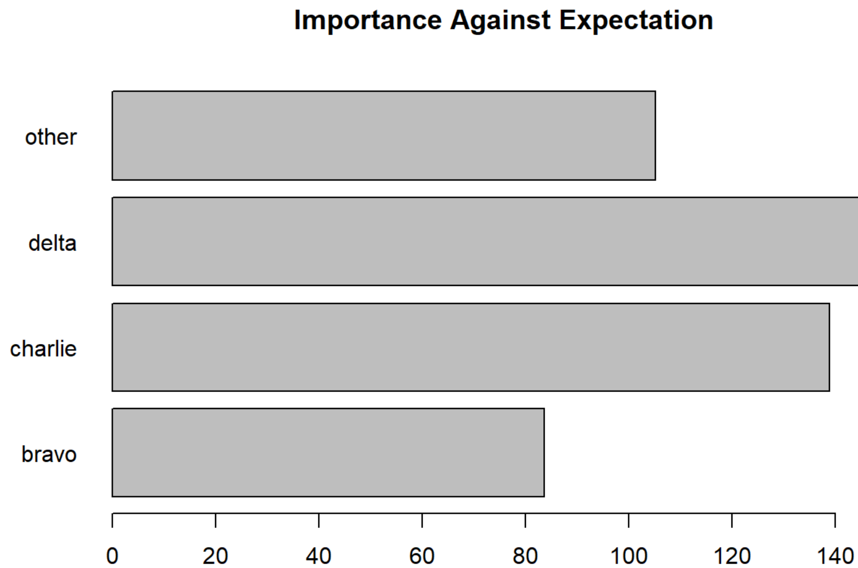
```
mkt_share <- 100 * colSums(df_panellist_vol[, 2:6]) /
  sum(df_panellist_vol[, 7])
```

Removing Alpha, we compute the share of residual (volume) purchasing for buyers of Alpha and for all buyers.

```
sorp_alpha <- cross_purchasing[1, 2:5] / (100 - cross_purchasing[1, 1])
sorp_cat <- mkt_share[2:5] / (100 - mkt_share[1])
```

We compute the index against expectation and plot the index by brand.

```
index_ae <- 100 * sorp_alpha / sorp_cat
index_ae

    bravo   charlie     delta     other
 83.65245 138.83557 153.10240 105.18848

barplot(index_ae,
        horiz = TRUE,
        las = 1,
        main = "Importance Against Expectation"
        )
```



Finally, we perform a cross purchasing analysis for year 1 using spend (instead of volume, as above).

```
tmp <- t(ever_buyers) %*% as.matrix(df_panellist_spend[, c(2:6)])
cross_purchasing_spend <- 100 * tmp /
  colSums(ever_buyers * (df_panellist_spend[, "category"]))
rownames(cross_purchasing_spend) <- colnames(cross_purchasing_spend)
cross_purchasing_spend

           alpha    bravo   charlie     delta      other
alpha   70.72216 18.77681  5.490578  3.806608  1.2038441
bravo   22.12248 68.10065  4.942957  2.400243  2.4336704
charlie 31.16527 26.80614 35.618017  5.513734  0.8968373
delta   32.95122 14.57527 10.231985 41.513042  0.7284904
other   21.20421 34.37785  3.043571  1.903325 39.4710388
```

Chapter 6 Analyses – Established Products

Understanding temporal variations in sales

The first step is to create a dataset that summarises, for each week, the number of panellists that made at least one purchase of Alpha, the total number of category purchase occasions on which Alpha was purchased, and Alpha's (dollar and volume) sales. (Yes, the revenue and volume numbers were created as part of the Chapter 3 analyses, but let's create them independently here.)

```r
df_alpha_weekly <- aggregate(cbind(trans_id, panel_id) ~ week,
                             data = df[df$brand == "Alpha", ],
                             function(x) length(unique(x))
)
colnames(df_alpha_weekly)[-1] <- c("num_trans", "num_buyers")

df_tmp <- aggregate(cbind(spend, volume) ~ week,
                    data = df[df$brand == "Alpha", ],
                    FUN = sum)
df_alpha_weekly$rev <- df_tmp$spend
df_alpha_weekly$vol <- df_tmp$volume
rm("df_tmp")
```

Next we compute the numbers associated with the revenue decomposition.

```r
df_alpha_weekly <- within(df_alpha_weekly,
                          {
                            penet <- num_buyers / num_panellists
                            ppb <- num_trans / num_buyers
                            aoval <- rev / num_trans
                            aovol <- vol / num_trans
                            avg_price_kg <- rev / vol
                          }
)
```

We compute the correlations between weekly revenue and the components of its (multiplicative) decomposition across the two years,

```r
cor(df_alpha_weekly[, c("rev", "penet", "aoval", "aovol", "avg_price_kg")])

                    rev      penet       aoval      aovol avg_price_kg
rev           1.0000000  0.9818491  0.54234453  0.7584870  -0.58957906
penet         0.9818491  1.0000000  0.38687004  0.7034499  -0.65403436
aoval         0.5423445  0.3868700  1.00000000  0.6914778  -0.07304781
aovol         0.7584870  0.7034499  0.69147783  1.0000000  -0.75415741
avg_price_kg -0.5895791 -0.6540344 -0.07304781 -0.7541574   1.00000000
```

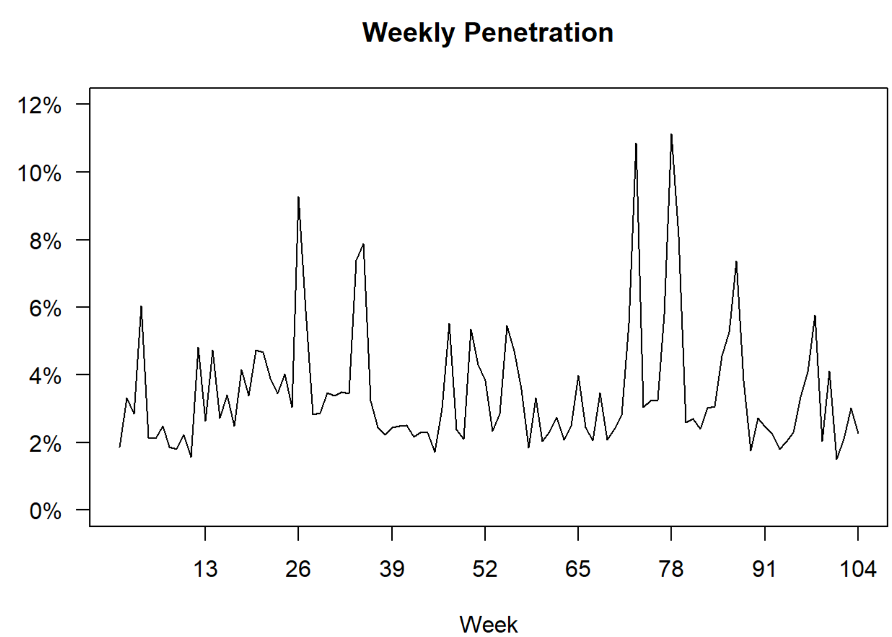
and separately for each of two years.

```r
cor(df_alpha_weekly[df_alpha_weekly$week <= 52,
                    c("rev", "penet", "aoval", "aovol", "avg_price_kg")]
    )

                    rev      penet      aoval      aovol avg_price_kg
rev           1.0000000  0.9901953  0.7876445  0.8397546   -0.5335911
penet         0.9901953  1.0000000  0.7072936  0.7778116   -0.5531073
aoval         0.7876445  0.7072936  1.0000000  0.9564885   -0.3491929
aovol         0.8397546  0.7778116  0.9564885  1.0000000   -0.6064231
avg_price_kg -0.5335911 -0.5531073 -0.3491929 -0.6064231    1.0000000
```

```
cor(df_alpha_weekly[df_alpha_weekly$week >= 53,
                    c("rev", "penet", "aoval", "aovol", "avg_price_kg")]
    )

                   rev      penet       aoval      aovol avg_price_kg
rev          1.0000000  0.9779584  0.41359141  0.7699498  -0.68502064
penet        0.9779584  1.0000000  0.22708650  0.7104118  -0.74828547
aoval        0.4135914  0.2270865  1.00000000  0.6020706  -0.03484664
aovol        0.7699498  0.7104118  0.60207059  1.0000000  -0.80414759
avg_price_kg -0.6850206 -0.7482855 -0.03484664 -0.8041476   1.00000000
```

We plot the weekly penetration numbers.

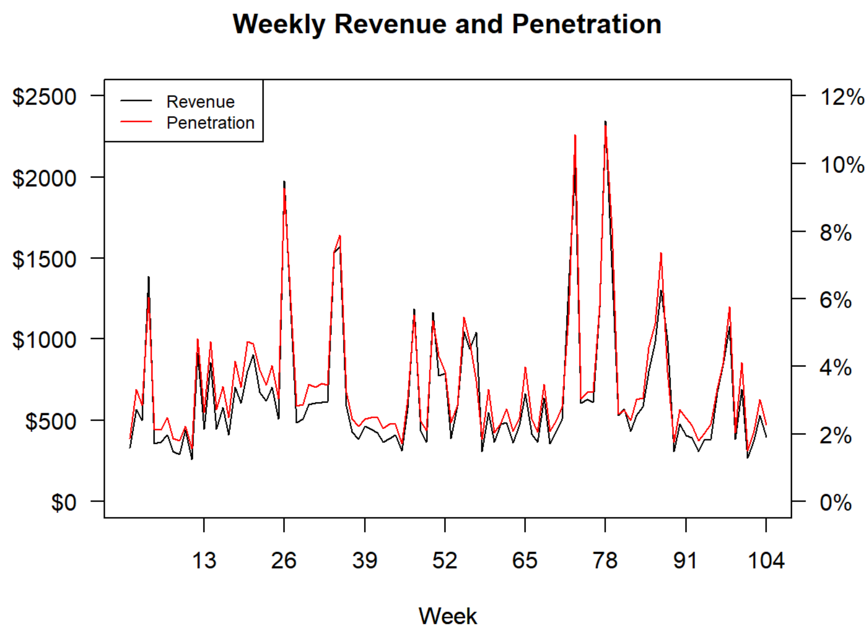
```
with(df_alpha_weekly,
     plot(week, 100 * penet,
          type = "l",
          main = "Weekly Penetration",
          xaxt="n",
          yaxt="n",
          xlab = "Week",
          ylab = "",
          ylim = c(0,12)
     )
)
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = seq(0, 12, by = 2),
     labels = c("0%", "2%", "4%", "6%", "8%", "10%", "12%"),
     las = 1)
```



In order to get a sense of how changes in revenue reflect changes in penetration, we want to plot both time series of the same set of axes.

```
# In order to create sufficient space for a second y-axis labels on the RHS
of
# the plot, we add extra space to right margin of plot within frame.
par(mar=c(5, 4, 4, 6) + 0.1)

# We plot the revenue data and draw the associated axes.
with(df_alpha_weekly,
     plot(week, rev,
          type = "l",
          main = "Weekly Revenue and Penetration",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 2500)
     )
)
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = seq(0, 2500, by = 500),
     labels = c("$0", "$500", "$1000", "$1500", "$2000", "$2500"),
     las = 1)

# We overlay a second plot, and add the second y-axis and legend.
par(new=TRUE)

with(df_alpha_weekly,
     plot(week, 100 * penet,
          type = "l",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 12),
          col="red"
     )
)
axis(4, at = seq(0, 12, by = 2),
     labels = c("0%", "2%", "4%", "6%", "8%", "10%", "12%"),
     las = 1,
     col.axis = "black")

legend("topleft",
       legend = c("Revenue", "Penetration"),
       lty = 1:1,
       cex = 0.75,
```

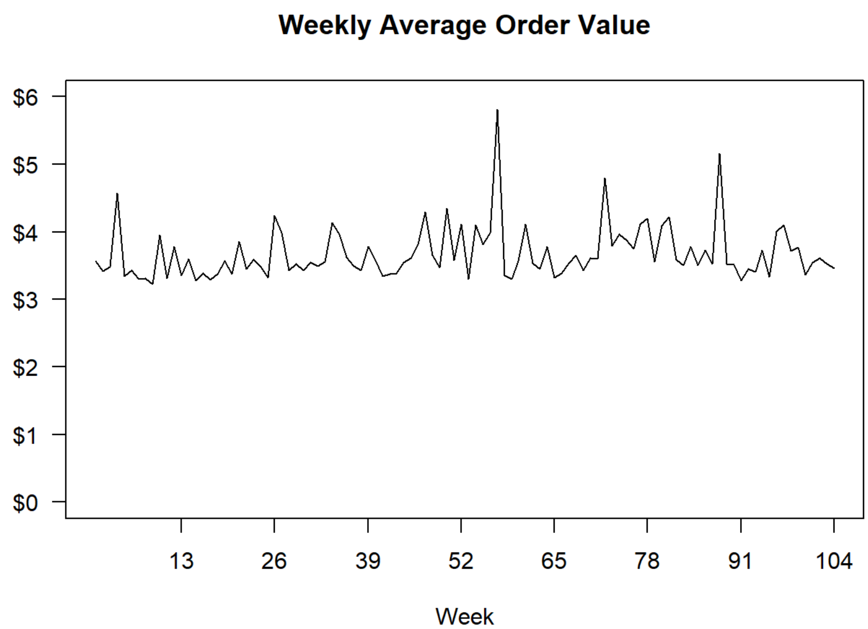
```
        col = c("black", "red")
)
```



We create the plots of the other components of the revenue decomposition.

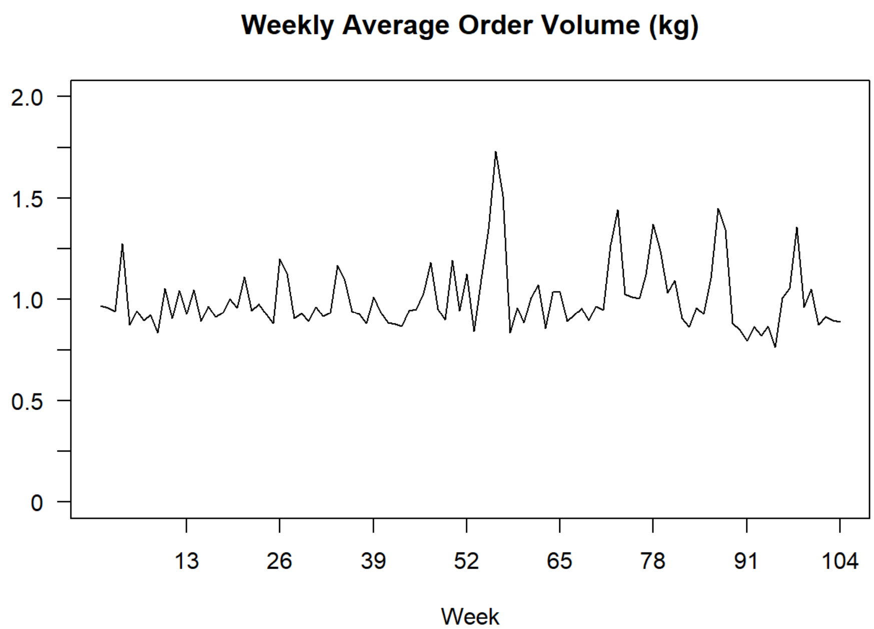
Average order value:

```
with(df_alpha_weekly,
     plot(week, aoval,
          type = "l",
          main = "Weekly Average Order Value",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 6)
     )
)
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = 0:6,
     labels = c("$0", "$1", "$2", "$3", "$4", "$5", "$6"),
     las = 1)
```



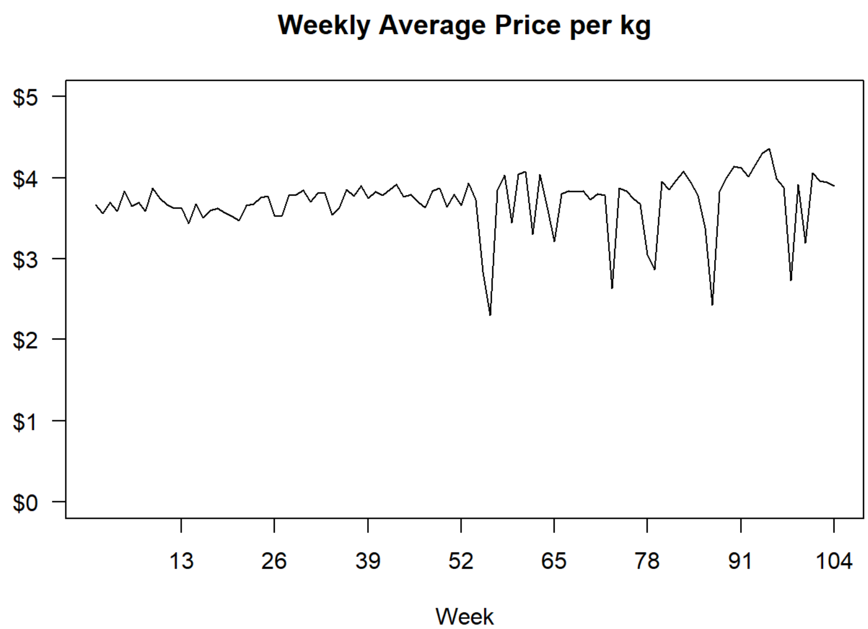
Average order volume:

```
with(df_alpha_weekly,
     plot(week, aovol,
          type = "l",
          main = "Weekly Average Order Volume (kg)",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 2)
     )
)
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = seq(0, 2, by = 0.25),
     labels = c("0", "", "0.5", "", "1.0", "", "1.5", "", "2.0"),
     las = 1)
```



Average price per kg:

```
with(df_alpha_weekly,
     plot(week, avg_price_kg,
          type = "l",
          main = "Weekly Average Price per kg",
          xaxt = "n",
          yaxt = "n",
          xlab = "Week",
          ylab = "",
          ylim = c(0, 5)
     )
)
axis(1, at = seq(13, 104, by = 13),
     las = 1)
axis(2, at = 0:5,
     labels = c("$0", "$1", "$2", "$3", "$4", "$5"),
     las = 1)
```



In order to perform a similar decomposition of annual revenue, we first need to create a dataset that summarises, for each year, the number of panellists that made at least one purchase of Alpha, the total number of category purchase occasions on which Alpha was purchased, and Alpha's (dollar and volume) sales. We use the same logic as above, aggregating by `year` as opposed to `week`.

```
df_alpha_annual <- aggregate(cbind(trans_id, panel_id) ~ year,
                             data = df[df$brand == "Alpha", ],
                             function(x) length(unique(x))
)
colnames(df_alpha_annual)[-1] <- c("num_trans", "num_buyers")

df_tmp <- aggregate(cbind(spend, volume) ~ year,
                    data = df[df$brand == "Alpha", ],
                    FUN = sum)
df_alpha_annual$rev <- df_tmp$spend
df_alpha_annual$vol <- df_tmp$volume
rm("df_tmp")

t(df_alpha_annual)

               [,1]     [,2]
year           1.00     2.00
num_trans   9060.00  9240.00
num_buyers  2624.00  2759.00
rev        33570.94 35250.75
vol         9166.25 10346.40
```

Next we compute the numbers associated with the revenue decomposition.

```
df_alpha_annual <- within(df_alpha_annual,
                          {
                            penet <- num_buyers / num_panellists
                            ppb <- num_trans / num_buyers
                            aoval <- rev / num_trans
                            aovol <- vol / num_trans
                            avg_price_kg <- rev / vol
                          }
)

t(df_alpha_annual[, c("penet", "ppb", "aoval", "aovol", "avg_price_kg")])

                  [,1]      [,2]
penet        0.5226051 0.5494921
ppb          3.4527439 3.3490395
aoval        3.7054018 3.8150162
aovol        1.0117274 1.1197403
avg_price_kg 3.6624508 3.4070546
```

We compute the percentage changes in each quantity.

```
100 * (df_alpha_annual[2, c(2:10)] / df_alpha_annual[1, c(2:10)] - 1)

  num_trans num_buyers      rev      vol avg_price_kg    aovol    aoval
2  1.986755   5.144817 5.003762 12.87495    -6.973369 10.67609 2.958234
        ppb    penet
2 -3.003536 5.144817
```

Temporal variation in customer-level purchasing

We first need to create a dataset that documents the number of times Alpha was purchased in years 1 and 2 by each panellist.

```
df_tmp <- aggregate(trans_id ~ panel_id + year,
                          data = df[df$brand == "Alpha",],
                          function(x) length(unique(x))
)

df_ann_trans_sum_alpha <- reshape(data = df_tmp,
                                  idvar = "panel_id",
                                  v.names = "trans_id",
                                  timevar = "year",
                                  direction = "wide"
                                  )
colnames(df_ann_trans_sum_alpha)[-1] <- c("year_1", "year_2")
df_ann_trans_sum_alpha[is.na(df_ann_trans_sum_alpha)] <- 0
rm("df_tmp")
```

We create the basic joint distribution,

```r
joint_dist_trans <- table(df_ann_trans_sum_alpha$year_1,
df_ann_trans_sum_alpha$year_2)
```

and add in the number of panellists that made no purchase of Alpha in either year.

```r
joint_dist_trans[1,1] <- num_panellists - sum(joint_dist_trans)
```

We right censor the distribution at 10+.

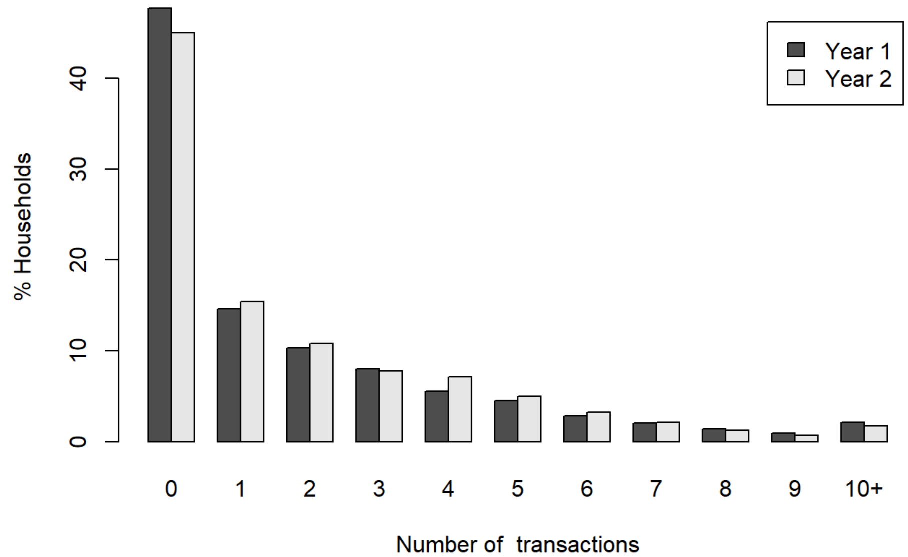
```r
tmp <-  rowSums(joint_dist_trans[, -c(1:10)])
joint_dist_trans <- cbind(joint_dist_trans[, c(1:10)], tmp)
tmp <-  colSums(joint_dist_trans[-c(1:10), ])
joint_dist_trans <- rbind(joint_dist_trans[c(1:10), ], tmp)

rownames(joint_dist_trans)[11] <- "10+"
colnames(joint_dist_trans)[11] <- "10+"
```

This gives us the following summary of the joint frequency distribution.

```r
joint_dist_trans

       0   1   2  3  4  5  6  7  8 9 10+
0   1879 342 105 39 18  9  4  1  0 0   0
1    259 201 128 79 40 14  6  3  2 0   1
2     83 120 108 80 75 27 11  9  3 1   0
3     25  60  78 83 65 54 21  9  3 1   1
4      8  28  62 45 54 34 26  8  3 6   3
5      5  13  28 31 49 46 23 20  5 3   4
6      1   6  15 17 24 31 20 13 11 3   3
7      0   2   7  5 15 15 16 14  8 4  15
8      1   1   3  5  9  9 16  6 10 8   3
9      1   0   3  4  4  4  7  8  4 5   7
10+    0   1   3  0  3  8 12 14 12 5  49
```

We compute the marginal distribution for each year and create the associated clustered bar chart.

```r
dist_y1 <- 100 * rowSums(joint_dist_trans) / num_panellists
dist_y2 <- 100 * colSums(joint_dist_trans) / num_panellists

barplot(rbind(dist_y1, dist_y2),
        beside = T,
        xlab = "Number of  transactions",
        ylab = "% Households",
        legend.text = c("Year 1", "Year 2")
        )
```



We compute the conditional distribution of transaction counts (i.e., the empirical probability of making x_2 transactions in year 2 given the panellist made x_1 transactions in year 1).

```
cond_dist_trans <- 100 * joint_dist_trans / rowSums(joint_dist_trans)
cond_dist_trans

              0          1         2         3          4          5
0   78.3896537 14.2678348  4.380476  1.627034  0.7509387  0.3754693
1   35.3342428 27.4215553 17.462483 10.777626  5.4570259  1.9099591
2   16.0541586 23.2108317 20.889749 15.473888 14.5067698  5.2224371
3    6.2500000 15.0000000 19.500000 20.750000 16.2500000 13.5000000
4    2.8880866 10.1083032 22.382671 16.245487 19.4945848 12.2743682
5    2.2026432  5.7268722 12.334802 13.656388 21.5859031 20.2643172
6    0.6944444  4.1666667 10.416667 11.805556 16.6666667 21.5277778
7    0.0000000  1.9801980  6.930693  4.950495 14.8514851 14.8514851
8    1.4084507  1.4084507  4.225352  7.042254 12.6760563 12.6760563
9    2.1276596  0.0000000  6.382979  8.510638  8.5106383  8.5106383
10+  0.0000000  0.9345794  2.803738  0.000000  2.8037383  7.4766355
              6           7          8          9        10+
0    0.1668753  0.04171882  0.0000000  0.0000000  0.0000000
1    0.8185539  0.40927694  0.2728513  0.0000000  0.1364256
2    2.1276596  1.74081238  0.5802708  0.1934236  0.0000000
3    5.2500000  2.25000000  0.7500000  0.2500000  0.2500000
4    9.3862816  2.88808664  1.0830325  2.1660650  1.0830325
5   10.1321586  8.81057269  2.2026432  1.3215859  1.7621145
6   13.8888889  9.02777778  7.6388889  2.0833333  2.0833333
7   15.8415842 13.86138614  7.9207921  3.9603960 14.8514851
8   22.5352113  8.45070423 14.0845070 11.2676056  4.2253521
```

```
9   14.8936170 17.02127660  8.5106383 10.6382979 14.8936170
10+ 11.2149533 13.08411215 11.2149533  4.6728972 45.7943925
```

[Optional] Creating the joint distribution of category spend in years 1 and 2

Let's explore how to create the joint distribution of category spend in years 1 and 2, something we didn't do in Excel.

The logic follows that of the binning of spend used to create the distribution of category spend in year 1 and the creation of `df_ann_trans_sum_alpha` above.

```
df_tmp <- aggregate(spend ~ panel_id + year,
                    data = df,
                    FUN = sum
)

boundaries <- c(-Inf, seq(0, 50, 5), max(df_tmp$spend) + 1)
df_tmp$bin <- cut(df_tmp$spend, breaks = boundaries)
df_tmp <- df_tmp[,-c(3)]

df_ann_spend_sum_cat <- reshape(data=df_tmp,
                                idvar="panel_id",
                                v.names = "bin",
                                timevar = "year",
                                direction="wide"
                                )
colnames(df_ann_spend_sum_cat)[-1] <- c("year_1", "year_2")
df_ann_spend_sum_cat[is.na(df_ann_spend_sum_cat)] <- "(-Inf,0]"
rm("df_tmp")
```

The one important change concerns the definition of `boundaries`. When we created the spend distribution, we used `c(seq(0, 50, 5), max(df_tmp$spend) + 1)`, which means the first bin excludes 0. This was fine when we were just looking at the distribution of spend among category buyers in that year. But we want to consider panellists that purchased in one year but not the other. Using `c(-Inf, seq(0, 50, 5), max(df_tmp$spend) + 1)` creates a bin that accommodates those with zero spend in one of the two years.

Given this binned summary, we create the joint distribution.

```
joint_dist_spend <- table(df_ann_spend_sum_cat$year_1,
df_ann_spend_sum_cat$year_2)
joint_dist_spend


           (-Inf,0] (0,5] (5,10] (10,15] (15,20] (20,25] (25,30] (30,35]
  (-Inf,0]        0   125     88      38       9       2       1       0
  (0,5]         129   260    211     103      30       5       9       1
  (5,10]         87   217    379     196     113      52      17       7
  (10,15]        28   107    248     223     140      78      24      15
```

```
  (15,20]        12     35     102      164      138      86      43      19
  (20,25]         4     19      52       82       90      81      63      25
  (25,30]         0      4      21       29       54      47      45      22
  (30,35]         0      3      10       18       28      38      17      32
  (35,40]         0      0       3        7       22      13      14      17
  (40,45]         0      0       0        1        7       8      11      10
  (45,50]         0      1       0        2        2       7       4       6
  (50,168]        1      1       0        3        4       5       8      13

           (35,40] (40,45] (45,50] (50,168]
  (-Inf,0]       1       0       0        0
  (0,5]          1       0       0        1
  (5,10]         2       0       0        3
  (10,15]        6       4       7        2
  (15,20]       13       5       2        4
  (20,25]       17       9       5        3
  (25,30]       21       6       2        5
  (30,35]       23      11       9        4
  (35,40]       19       9       8       10
  (40,45]        6      14       6        5
  (45,50]        3       6       5       12
  (50,168]       9       7      11       47
```

This is the joint distribution of spend for those panellists that made at least one category purchase across the two years. If we want to include those panellists that didn't make a category purchase, we can modify the top-left entry in this table.

```
joint_dist_spend[1,1] <- num_panellists - sum(joint_dist_spend)
joint_dist_spend
```

```

           (-Inf,0] (0,5] (5,10] (10,15] (15,20] (20,25] (25,30] (30,35]
  (-Inf,0]      183   125     88      38       9       2       1       0
  (0,5]         129   260    211     103      30       5       9       1
  (5,10]         87   217    379     196     113      52      17       7
  (10,15]        28   107    248     223     140      78      24      15
  (15,20]        12    35    102     164     138      86      43      19
  (20,25]         4    19     52      82      90      81      63      25
  (25,30]         0     4     21      29      54      47      45      22
  (30,35]         0     3     10      18      28      38      17      32
  (35,40]         0     0      3       7      22      13      14      17
  (40,45]         0     0      0       1       7       8      11      10
  (45,50]         0     1      0       2       2       7       4       6
  (50,168]        1     1      0       3       4       5       8      13
```

```
            (35,40] (40,45] (45,50] (50,168]
  (-Inf,0]        1       0       0        0
  (0,5]           1       0       0        1
  (5,10]          2       0       0        3
  (10,15]         6       4       7        2
  (15,20]        13       5       2        4
  (20,25]        17       9       5        3
  (25,30]        21       6       2        5
  (30,35]        23      11       9        4
  (35,40]        19       9       8       10
  (40,45]         6      14       6        5
  (45,50]         3       6       5       12
  (50,168]        9       7      11       47
```

Repeat rates

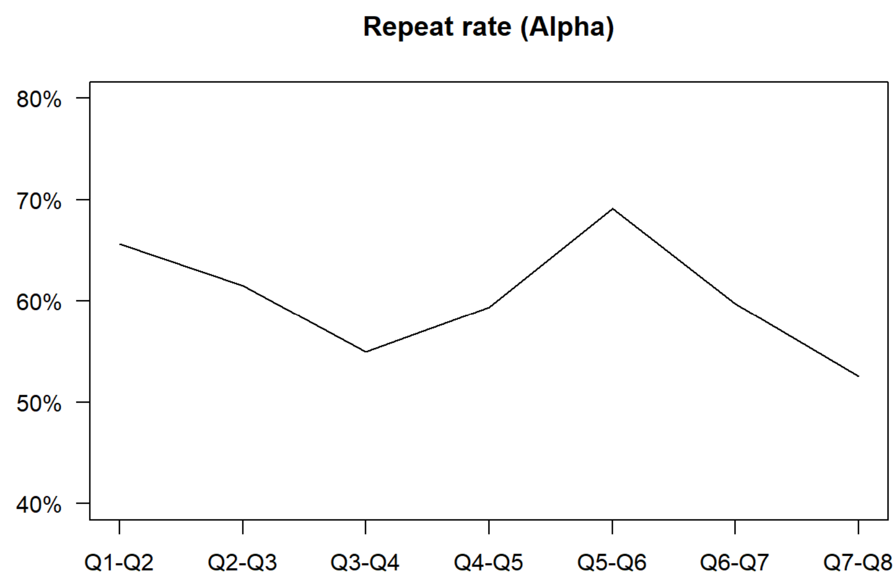
We wish to compute the quarterly repeat rate (or repeat-buying rate) numbers for Alpha. The first thing we do is create a quarterly incidence matrix that indicates whether or not each panellist purchased Alpha each quarter.

```
df_tmp <- df[df$brand == "Alpha",]
df_tmp$quarter = floor((df_tmp$week - 1) / 13) + 1
alpha_qtrly_incid <- 1 * (table(df_tmp$panel_id, df_tmp$quarter) > 0)
rm("df_tmp")
```

(Panellists that never purchased Alpha in the two-year period are automatically excluded.) The repeat buying rate is the proportion of buyers in one quarter that purchased again in the next quarter.

```
rbr <- numeric(7)
for (q in 1:7){
  rbr[q] <- sum(alpha_qtrly_incid[, q] * alpha_qtrly_incid[, q + 1]) /
    sum(alpha_qtrly_incid[, q])
}
rbr

[1] 0.6566820 0.6149218 0.5493134 0.5934718 0.6914008 0.5974877 0.5254975

plot(100 * rbr,
    type = "l",
    main = "Repeat rate (Alpha)",
    xaxt = "n",
    yaxt = "n",
    xlab = "",
    ylab = "",
    ylim = c(40, 80))

axis(1, at = 1:7,
     labels = c("Q1-Q2", "Q2-Q3", "Q3-Q4", "Q4-Q5", "Q5-Q6", "Q6-Q7",
```

```
                "Q7-Q8"),
     las = 1)
axis(2, at = seq(40, 80, by = 10),
     labels = c("40%", "50%", "60%", "70%", "80%"),
     las = 1)
```



Chapter 6 Analyses – New products

Setting up the data

We will be working with a new dataset. Let's clear the workspace and load the associated csv file.

```
rm(list = ls())
df_kiwibubbles_trans <-
read.csv("C:/Users/bhard/Desktop/kiwibubbles_trans.csv",
                                 fileEncoding = "UTF-8-BOM")
```

We will only work with market 2.

```
df <- df_kiwibubbles_trans[df_kiwibubbles_trans$Market == 2, ]
df <- df[-df$Market]
num_panellists <- 1499
```

We create a day of year variable, where the 1 corresponds to the day the new product was launched.

```
df <- within(df,
             {
               doy <- (Week - 1) * 7 + Day
```

```
             }
             )
```

We shouldn’t assume that the dataset is sorted by time of transaction for each panellist.

```
df <- df[order(df$ID, df$doy), ]
```

> **[Optional] Double checking the data**
>
> The smallest unit of time for our analyses is day. Do we have any panellists with more than one transaction on any day?
>
> ```
> df_tmp <- aggregate(Units ~ ID + doy,
>                      data = df,
>                      FUN = sum
> )
> nrow(df) == nrow(df_tmp)
>
> [1] TRUE
> ```
>
> So, we’re OK. (Do you understand what we just did with this bit of code?)

The next step is to create a depth of repeat variable, where 0 = trial purchase, 1 = first repeat purchase, and so on.

```
df$dor <- numeric(nrow(df))
for(i in 2:nrow(df)){
  if (df$ID[i] == df$ID[i - 1]){
    df$dor[i] <- df$dor[i - 1] + 1
  }
}
```

Did at least one panellist make a purchase of this new product each week?

```
length(unique(df$Week))

[1] 49
```

No. So which weeks are missing?

```
setdiff(c(1:52), unique(sort(df$Week)))

[1] 25 39 41
```

Let’s add empty rows in `df` which correspond to the missing weeks.

```
missing_wks <- setdiff(c(1:52), unique(sort(df$Week)))
for(i in 1:length(missing_wks)){
  df[nrow(df) + 1, 2] = missing_wks[i]
}
```

Basic plots

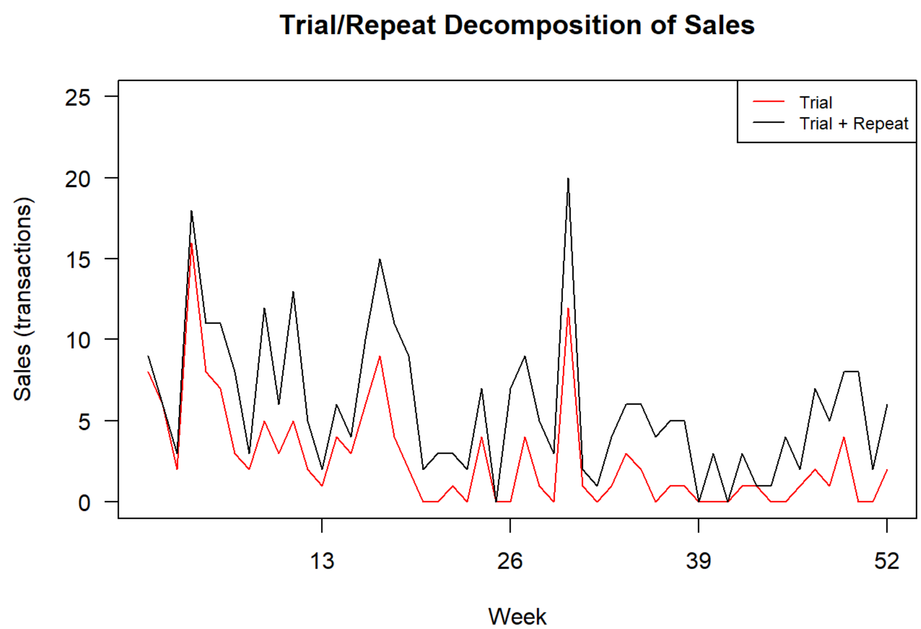
The basic plots are created off a summary of the dataset that gives us the number of trial, first repeat, and additional repeat transactions for each week.

```
trans_wk_dor <- table(df$Week, df$dor)
trial <- trans_wk_dor[, 1]
rpt <- rowSums(trans_wk_dor[, -c(1)])
fr <- trans_wk_dor[, 2]
ar <- rpt - fr
```

It is very difficult to create the stacked area plots we created in Excel using base R. We create equivalent plots in the following manner.

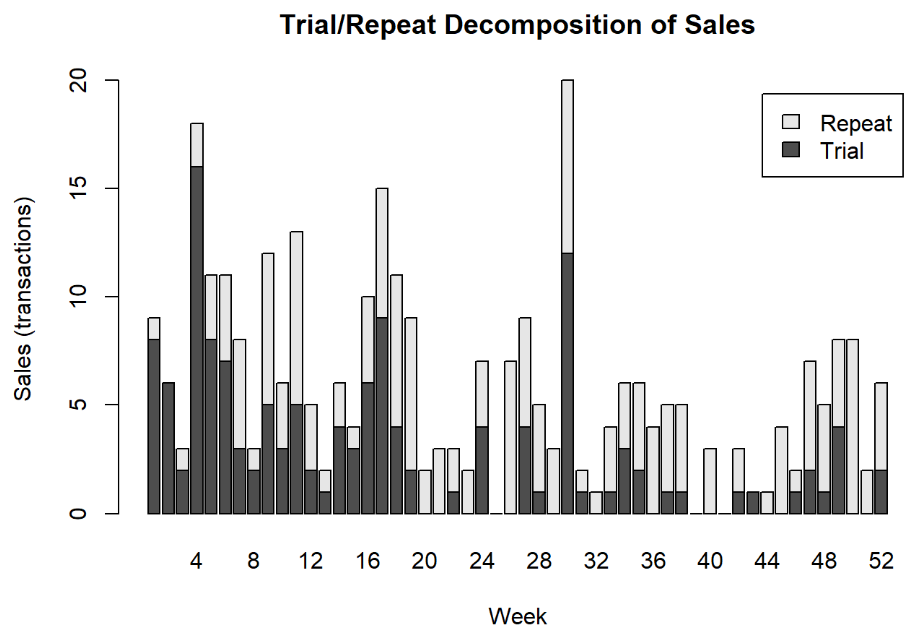
First, let's plot a trial/repeat decomposition of total weekly sales (where sales in the number of transactions).

```
plot(1:52, trial,
     type = "l",
     col = "red",
     xlab = "Week",
     ylab = "Sales (transactions)",
     xaxt = "n",
     yaxt = "n",
     ylim = c(0, 25),
     main = "Trial/Repeat Decomposition of Sales"
)
axis(1, at = seq(13, 52, by = 13),
     las = 1)
axis(2, at = seq(0, 25, by = 5),
     las = 1)

lab <- rep(NA, 52)
lab[seq(4, 52, by = 4)] <- seq(4, 52, by = 4)

lines(trial + rpt,
      type = "l")

legend("topright",
       legend = c("Trial", "Trial + Repeat"),
       lty = 1:1,
       col = c("red", "black"),
       cex = 0.75
)
```



An alternative way of plotting the data would be to use a stacked bar chart.

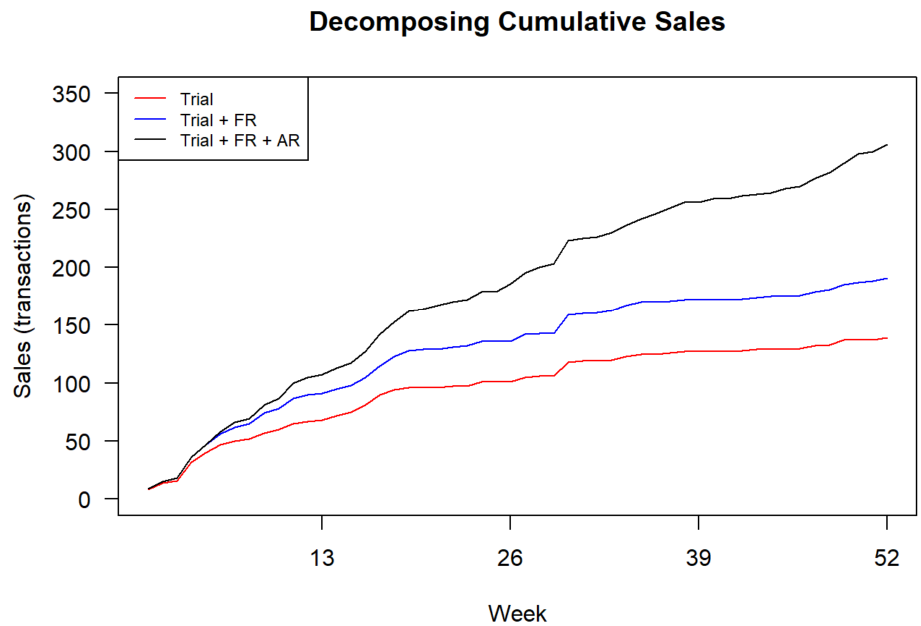
```
barplot(t(cbind(trial, rpt)),
        xlab = "Week",
        ylab = "Sales (transactions)",
        main = "Trial/Repeat Decomposition of Sales",
        legend.text = c("Trial", "Repeat"),
        names.arg = lab
)
```



Next we create a plot that decomposes cumulative sales into its trial, first repeat, and additional repeat components.

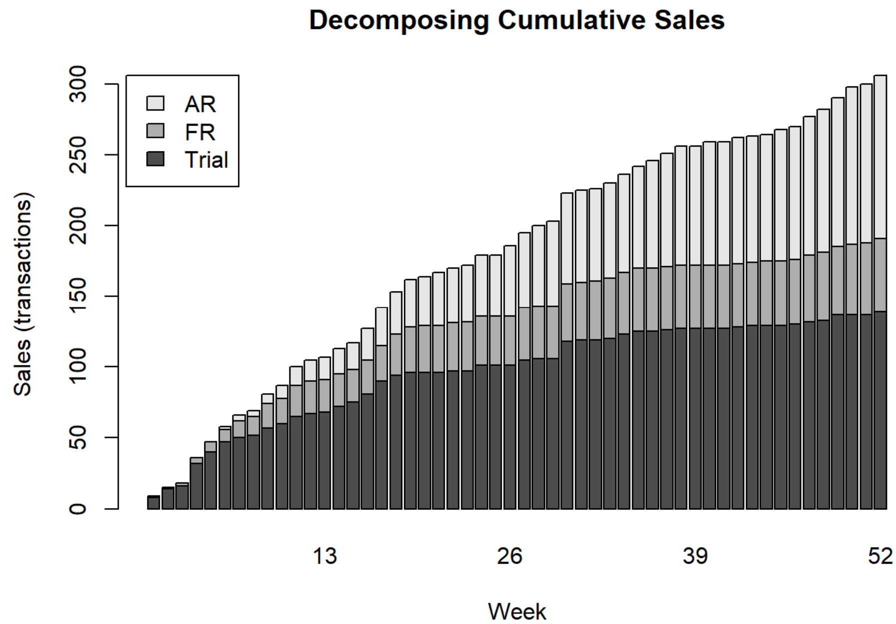
```
plot(1:52, cumsum(trial),
     type = "l",
     col = "red",
     xlab = "Week",
     ylab = "Sales (transactions)",
     xaxt = "n",
     yaxt = "n",
     ylim = c(0, 350),
     main = "Decomposing Cumulative Sales"
)
axis(1, at = seq(13, 52, by = 13),
     las = 1)
axis(2, at = seq(0, 350, by = 50),
     las = 1)

lab <- rep(NA, 52)
lab[seq(13, 52, by=13)] <- c(13, 26, 39, 52)

lines(cumsum(trial + fr),
      type = "l",
      col = "blue")
lines(1:52, cumsum(trial + fr + ar),
      type = "l",
      col = "black")

legend("topleft",
```

```
       legend = c("Trial", "Trial + FR", "Trial + FR + AR"),
       lty = c(1, 1, 1),
       col = c("red", "blue", "black"),
       cex = 0.75
)
```



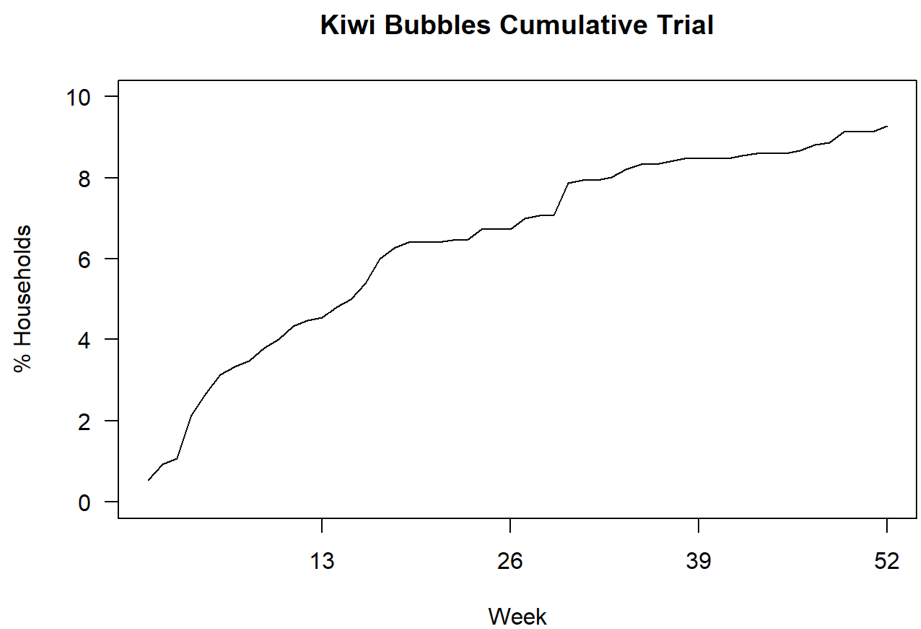
The stacked bar chart version:

```
barplot(t(cbind(cumsum(trial), cumsum(fr), cumsum(ar))),
        xlab = "Week",
        ylab = "Sales (transactions)",
        main = "Decomposing Cumulative Sales",
        legend.text = c("Trial", "FR", "AR"),
        args.legend = list(x ='topleft', inset=c(0.01, 0)),
        names.arg = lab
)
```



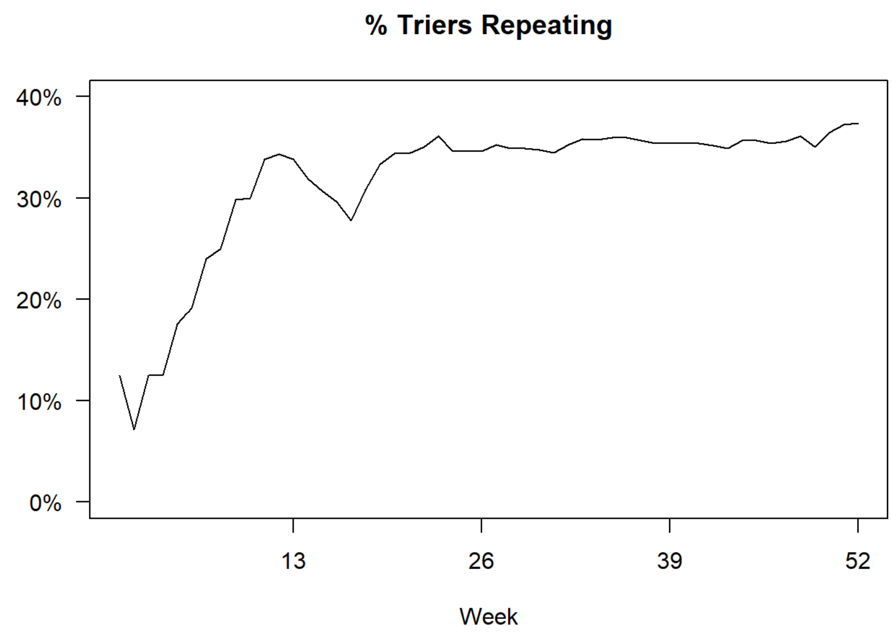
Next we plot cumulative trial as a percent of panel size (sometimes called cumulative penetration).

```
plot(1:52, 100 * cumsum(trial) / num_panellists,
     type = "l",
     xlab = "Week",
     ylab = "% Households",
     xaxt = "n",
     yaxt = "n",
     ylim = c(0, 10),
     main = "Kiwi Bubbles Cumulative Trial"
)
axis(1, at = seq(13, 52, by = 13),
     las = 1)
axis(2, at = seq(0, 10, by = 2),
     las = 1)
```



Finally, we create a plot of the percentage of triers making a repeat purchase.

```
pct_triers_rpting <- 100 * cumsum(fr) / cumsum(trial)
plot(1:52, pct_triers_rpting,
     type = "l",
     xlab = "Week",
     ylab = "",
     xaxt = "n",
     yaxt = "n",
     ylim = c(0, 40),
     main = "% Triers Repeating"
     )
axis(1, at = seq(13, 52, by = 13),
     las = 1)
axis(2, at = seq(0, 40, by = 10),
     labels = c("0%", "10%", "20%", "30%", "40%"),
     las = 1)
```



> **[Optional] Exercise**
> The unit of sales used in these plots is transactions. How would you create plots where the unit of sales is units purchased?

Exploring time to first repeat

The first step is to create a table that reports how many panellists made a (first) repeat purchase so many weeks after their trial purchase, broken down by week of trial.

We start by removing the three rows we added to account for the weeks in which no transactions occurred.

```
df <- df[-c((nrow(df) - 2):nrow(df)), ]
```

Next we create a "week of trial purchase" variable and a variable the counts the number of weeks between a panellist's trial and first repeat purchase.

```
df$trial_wk <- rep(-99, nrow(df))

for(i in 1:(nrow(df))){
  if (df$dor[i] == 0){
    df$trial_wk[i] <- df$Week[i]}
  }

for(i in 1:(nrow(df) - 1)){
  if (df$dor[i + 1] == 1){df$fr_delta[i] <- df$Week[i + 1] - df$Week[i]}
  else {df$fr_delta[i] <- -99
  }
}
```

We cannot assume that all the trial and “time from trial to FR” weeks are observed in the dataset, so we fill in the missing values.

```
missing_trial_wks <- setdiff(c(-99, 1:52), unique(sort(df$trial_wk)))
for(i in 1:length(missing_trial_wks)){
  df[nrow(df) + 1, 7] = missing_trial_wks[i]
}

missing_fr_delta <- setdiff(c(-99, 0:51), unique(sort(df$fr_delta)))
for(i in 1:length(missing_fr_delta)){
  df[nrow(df) + 1, 8] = missing_fr_delta[i]
}
```

We create the desired table.

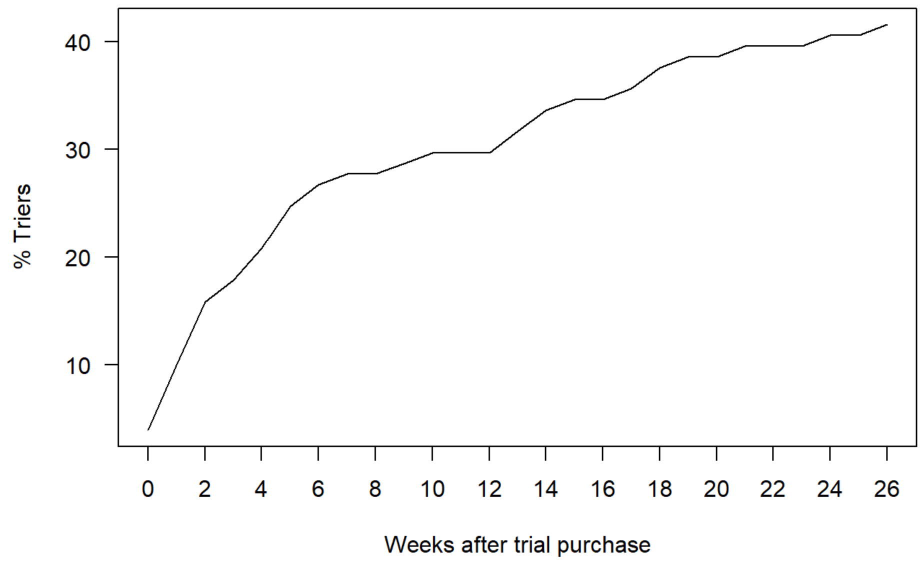
```
time_to_fr_by_trial <- table(df$trial_wk, df$fr_delta)
```

As a final step before creating the desired plot, we create a cumulative version of this table, focusing on those customers that had 26 weeks after their trial purchase to make a first repeat purchase.

```
cum_fr_by_trial <- matrix(0, nrow = 26, ncol = 27)
for (i in 1:26){
  cum_fr_by_trial[i, ] <- cumsum(time_to_fr_by_trial[i + 1, c(2:28)])
}
```

Now we can create the plot that shows the percentage of triers that have made a first repeat purchase within 26 weeks of their trial purchase.

```
time_to_fr <- colSums(cum_fr_by_trial) / sum(trial[c(1:26)])

plot(0:26, 100 * time_to_fr,
     type = "l",
     xlab = "Weeks after trial purchase",
     ylab = "% Triers",
     xaxt = "n",
     yaxt = "n"
)
axis(1, at = seq(0, 26, by = 2),
     las = 1)
axis(2, at = seq(10, 40, by = 10),
     las = 1)
```



[Optional] Exercise

Replicate the undocumented time from first repeat to second repeat analysis reported in `solution__chapter_6b.xlsx`.

[Optional] Exercise

Reflecting on the time to first repeat analysis we have just undertaken, someone who made their trial purchase on day 1 of week 2 and their first repeat purchase on day 7 of week 3 has the same `fr_delta` as someone who made their trial purchase on day 7 of week 2 and their first repeat purchase on day 1 of week 3. An alternative (and arguably more correct) approach would be to create `fr_delta` off doy. Recreate the time to first repeat figure using this alternative measure of time between trial and first repeat purchases.