# Data Wrangling with R - Part 1

## Agenda

Use `dplyr` to:

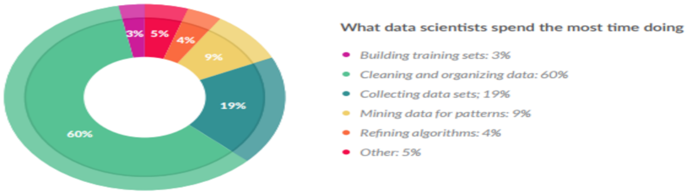
- select variables/columns
- filter data
- arrange data
- generate new variables
- grouped summaries

## Introduction

According to a survey by CrowdFlower, data scientists spend most of their time cleaning and manipulating data rather than mining or modeling them for insights. As such, it becomes important to have tools that make data manipulation faster and easier. In today's post, we introduce you to dplyr, a grammar of data manipulation.

### How a Data Scientist Spends Their Day

Here's where the popular view of data scientists diverges pretty significantly from reality. Generally, we think of data scientists building algorithms, exploring data, and doing predictive analysis. That's actually not what they spend most of their time doing, however.



As you can see from the chart above, 3 out of every 5 data scientists we surveyed actually spend the most time cleaning and organizing data. You may have heard this referred to as "data wrangling" or compared to digital janitor work. Everything from list verification to removing commas to debugging databases–that time adds up and it adds up immensely. Messy data is by far the more time-consuming aspect of the typical data scientist's work flow. And nearly 60% said they simply spent too much time doing it.

## Why dplyr

dplyr helps us solve the most common data manipulation challenges such as filtering rows, selecting columns, sorting data, creating new columns, summarizing data etc. In order to truly appreciate dplyr, we will compare it to the functions in base R.

## Libraries

```r
library(dplyr)
library(readr)
```

## dplyr Verbs

dplyr provides a set of verbs that help us solve the most common data manipulation challenges while working with tabular data (dataframes, tibbles):

- `select`: returns subset of columns
- `filter`: returns a subset of rows
- `arrange`: re-order or arrange rows according to single/multiple variables
- `mutate`: create new columns from existing columns
- `summarise`: reduce data to a single summary

## Case Study

We will explore dummy data that we have created to resemble web logs of a ecommerce company. You can download the data from here or import it directly using `read_csv()` from the readr package. We will use dplyr to answer a few questions we have about the above data:

- what is the average order value by device types?
- what is the average number of pages visited by purchasers and non-purchasers?
- what is the average time on site for purchasers vs non-purchasers?
- what is the average number of pages visited by purchasers and non-purchasers using mobile?

## Data

```r
ecom <- read_csv('data/web.csv')
ecom
```

```
## # A tibble: 1,000 x 11
##       id referrer device bouncers n_visit n_pages durati
##    <int> <chr>    <chr>  <chr>      <int>   <dbl>     <db
##  1     1 google   laptop true          10    1.00    693
##  2     2 yahoo    tablet true           9    1.00    459
##  3     3 direct   laptop true           0    1.00    996
##  4     4 bing     tablet false          3   18.0     468
##  5     5 yahoo    mobile true           9    1.00    955
##  6     6 yahoo    laptop false          5    5.00    135
##  7     7 yahoo    mobile true          10    1.00     75
##  8     8 direct   mobile true          10    1.00    908
##  9     9 bing     mobile false          3   19.0     209
## 10    10 google   mobile true           6    1.00    208
## # ... with 990 more rows, and 3 more variables: purchase
## #   order_items <dbl>, order_value <dbl>
```

## Data Dictionary

Below is the description of the data set:

- id: row id
- referrer: referrer website/search engine
- os: operating system
- browser: browser
- device: device used to visit the website
- n_pages: number of pages visited
- duration: time spent on the website (in seconds)
- repeat: frequency of visits
- country: country of origin
- purchase: whether visitor purchased
- order_value: order value of visitor (in dollars)

## Average Order Value

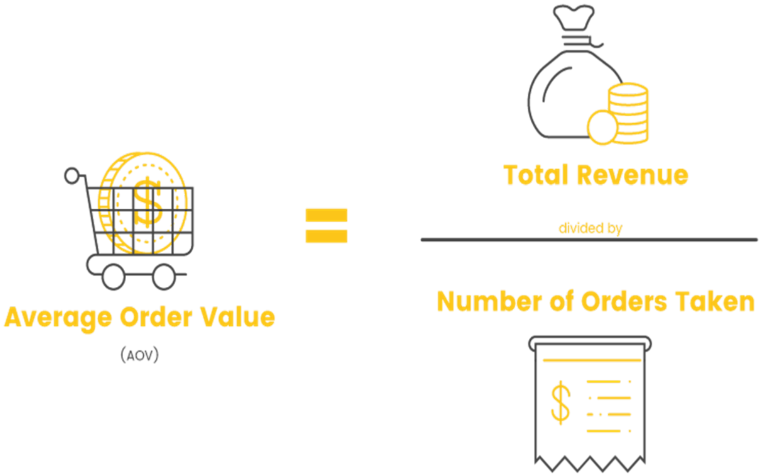
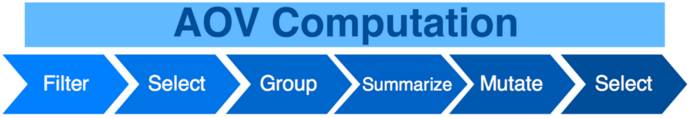


Figure 2: alt text

## AOV Computation



| | | | | | |
|---|---|---|---|---|---|
| Filter purchasers from the data set. | Select data related to AOV (order value, order items) and device. | Group order value and order items by device type. | Compute total order value and order items for each device type. | Compute AOV for each device type. | Select device and AOV columns. |

## AOV by Devices

```r
ecom %>%
  filter(purchase == 'true') %>%
  select(device, order_value, order_items) %>%
  group_by(device) %>%
  summarise_all(funs(sum)) %>%
  mutate(
    aov = order_value / order_items
  ) %>%
  select(device, aov)
```

```
## # A tibble: 3 x 2
##   device   aov
##   <chr>  <dbl>
## 1 laptop   353
## 2 mobile   280
## 3 tablet   261
```

## Step 1: Filter Purchasers

In order to compute the AOV, we must first separate the purchasers from non-purchasers. We will do this by filtering the data related to purchasers using the `filter()` function. It allows us to filter rows that meet a specific criteria/condition. The first argument is the name of the data frame and the rest of the arguments are expressions for filtering the data. Let us look at a few examples:

## Filter

| device | purchase |
|---|---|
| mobile | FALSE |
| tablet | FALSE |
| laptop | TRUE |
| laptop | FALSE |
| mobile | TRUE |
| laptop | TRUE |
| tablet | FALSE |
| mobile | TRUE |
| laptop | TRUE |
| laptop | FALSE |

Filter data for traffic from mobile

```
filter(data, device == "mobile")
```

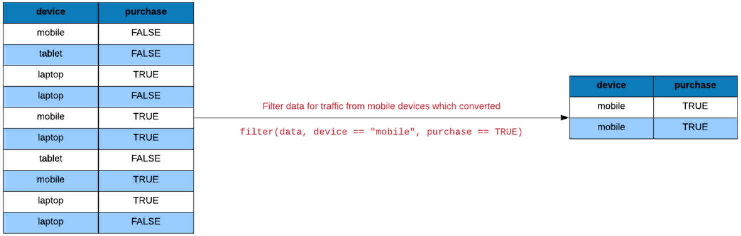
| device | purchase |
|---|---|
| mobile | FALSE |
| mobile | TRUE |
| mobile | TRUE |

## Select all visits from mobile

```r
filter(ecom, device == "mobile")
```

```
## # A tibble: 344 x 11
##        id referrer device bouncers n_visit n_pages durat
##     <int> <chr>    <chr>  <chr>      <int>   <dbl>    <db
##  1      5 yahoo    mobile true           9    1.00    955
##  2      7 yahoo    mobile true          10    1.00     75
##  3      8 direct   mobile true          10    1.00    908
##  4      9 bing     mobile false          3   19.0     209
##  5     10 google   mobile true           6    1.00    208
##  6     13 direct   mobile false          9   14.0     406
##  7     15 yahoo    mobile false          7    1.00     19
##  8     22 google   mobile true           5    1.00    147
##  9     23 bing     mobile false          0    7.00    196
## 10     29 google   mobile true          10    1.00    338
## # ... with 334 more rows, and 3 more variables: purchase
## #   order_items <dbl>, order_value <dbl>
```

## Filter



| device | purchase |
|---|---|
| mobile | FALSE |
| tablet | FALSE |
| laptop | TRUE |
| laptop | FALSE |
| mobile | TRUE |
| laptop | TRUE |
| tablet | FALSE |
| mobile | TRUE |
| laptop | TRUE |
| laptop | FALSE |

Filter data for traffic from mobile devices which converted

```
filter(data, device == "mobile", purchase == TRUE)
```

| device | purchase |
|---|---|
| mobile | TRUE |
| mobile | TRUE |

## Visits from mobile that converted

```r
filter(ecom, device == "mobile", purchase == "true")
```

```
## # A tibble: 36 x 11
##       id referrer device bouncers n_visit n_pages durat
##    <int> <chr>    <chr>  <chr>      <int>   <dbl>    <db
##  1    13 direct   mobile false          9    14.0       4
##  2    41 bing     mobile false          4    20.0       4
##  3    98 bing     mobile false          3    18.0       2
##  4   112 social   mobile false         10    11.0       2
##  5   125 yahoo    mobile false          6    14.0       3
##  6   134 google   mobile false          1    18.0       2
##  7   143 social   mobile false          7    16.0       3
##  8   156 direct   mobile false          4    18.0       3
##  9   219 social   mobile false          1    20.0       5
## 10   227 yahoo    mobile false          0    13.0       3
## # ... with 26 more rows, and 3 more variables: purchase
## #   order_items <dbl>, order_value <dbl>
```

## From mobile & visited > 5 pages

```r
filter(ecom, device == "mobile", n_pages > 5)
```

```
## # A tibble: 139 x 11
##        id referrer device bouncers n_visit n_pages durat
##     <int> <chr>    <chr>  <chr>      <int>   <dbl>    <db
##  1      9 bing     mobile false          3   19.0       2
##  2     13 direct   mobile false          9   14.0       4
##  3     23 bing     mobile false          0    7.00      1
##  4     30 yahoo    mobile false          8    9.00      2
##  5     41 bing     mobile false          4   20.0       4
##  6     42 direct   mobile false          1   13.0       2
##  7     89 direct   mobile false          4    8.00      1
##  8     92 google   mobile false          5    8.00      1
##  9     98 bing     mobile false          3   18.0       2
## 10    112 social   mobile false         10   11.0       2
## # ... with 129 more rows, and 3 more variables: purchase
## #   order_items <dbl>, order_value <dbl>
```

## Case Study: Visits that converted

```r
filter(ecom, purchase == "true")
```

```
## # A tibble: 103 x 11
##       id referrer device bouncers n_visit n_pages durat
##    <int> <chr>    <chr>  <chr>      <int>   <dbl>    <db
##  1     4 bing     tablet false          3    18.0       4
##  2    13 direct   mobile false          9    14.0       4
##  3    17 bing     tablet false          5    16.0       3
##  4    19 social   tablet false          7    10.0       2
##  5    27 direct   tablet false          2    19.0       3
##  6    34 social   tablet false          9    20.0       4
##  7    41 bing     mobile false          4    20.0       4
##  8    94 yahoo    tablet false          2    16.0       4
##  9    98 bing     mobile false          3    18.0       2
## 10   101 yahoo    tablet false          2    14.0       3
## # ... with 93 more rows, and 3 more variables: purchase
## #   order_items <dbl>, order_value <dbl>
```

## Step 2: Select relevant columns

After filtering the data, we need to select relevent variables to compute the AOV. Remember, we do not need all the columns in the data to compute a required metric (in our case, AOV). The `select()` function allows us to select a subset of columns. The first argument is the name of the data frame and the subsequent arguments specify the columns by name or position. Let us look at a few examples:

## Select

| id | referrer | device | purchase | duration |
|---|---|---|---|---|
| VF001 | google | mobile | FALSE | 32 |
| VF002 | social | tablet | FALSE | 56 |
| VF003 | direct | laptop | TRUE | 306 |
| VF004 | facebook | laptop | FALSE | 100 |
| VF005 | affiliate | mobile | TRUE | 341 |
| VF006 | google | laptop | TRUE | 432 |

Select device and purchase columns

```
select(data, device, purchase)
```

| device | purchase |
|---|---|
| mobile | FALSE |
| tablet | FALSE |
| laptop | TRUE |
| laptop | FALSE |
| mobile | TRUE |
| laptop | TRUE |

## Select device and purchase columns

```r
select(ecom, device, purchase)
```

```
## # A tibble: 1,000 x 2
##    device purchase
##    <chr>  <chr>
##  1 laptop false
##  2 tablet false
##  3 laptop false
##  4 tablet true
##  5 mobile false
##  6 laptop false
##  7 mobile false
##  8 mobile false
##  9 mobile false
## 10 mobile false
## # ... with 990 more rows
```

## Select

| id | referrer | device | purchase | duration |
|---|---|---|---|---|
| VF001 | google | mobile | FALSE | 32 |
| VF002 | social | tablet | FALSE | 56 |
| VF003 | direct | laptop | TRUE | 306 |
| VF004 | facebook | laptop | FALSE | 100 |
| VF005 | affiliate | mobile | TRUE | 341 |
| VF006 | google | laptop | TRUE | 432 |

Select all columns from referrer till purchase

```
select(data, referrer:purchase)
```

| referrer | device | purchase |
|---|---|---|
| google | mobile | FALSE |
| social | tablet | FALSE |
| direct | laptop | TRUE |
| facebook | laptop | FALSE |
| affiliate | mobile | TRUE |
| google | laptop | TRUE |

## all columns from device to purchase

```r
select(ecom, device:purchase)
```

```
## # A tibble: 1,000 x 7
##    device bouncers n_visit n_pages duration country
##    <chr>  <chr>      <int>   <dbl>    <dbl> <chr>
##  1 laptop true          10    1.00    693   Czech Republ
##  2 tablet true           9    1.00    459   Yemen
##  3 laptop true           0    1.00    996   Brazil
##  4 tablet false          3   18.0     468   China
##  5 mobile true           9    1.00    955   Poland
##  6 laptop false          5    5.00    135   South Africa
##  7 mobile true          10    1.00     75.0 Bangladesh
##  8 mobile true          10    1.00    908   Indonesia
##  9 mobile false          3   19.0     209   Netherlands
## 10 mobile true           6    1.00    208   Czech Republ
## # ... with 990 more rows
```

## Select

| id | referrer | device | purchase | duration |
|---|---|---|---|---|
| VF001 | google | mobile | FALSE | 32 |
| VF002 | social | tablet | FALSE | 56 |
| VF003 | direct | laptop | TRUE | 306 |
| VF004 | facebook | laptop | FALSE | 100 |
| VF005 | affiliate | mobile | TRUE | 341 |
| VF006 | google | laptop | TRUE | 432 |

Select all columns except id and duration

`select(data, -id, -duration)`

| referrer | device | purchase |
|---|---|---|
| google | mobile | FALSE |
| social | tablet | FALSE |
| direct | laptop | TRUE |
| facebook | laptop | FALSE |
| affiliate | mobile | TRUE |
| google | laptop | TRUE |

## all columns excluding id and country

```
select(ecom, -id, -country)
```

```
## # A tibble: 1,000 x 9
##    referrer device bouncers n_visit n_pages duration pur
##    <chr>    <chr>  <chr>      <int>   <dbl>    <dbl> <ch
##  1 google   laptop true          10    1.00    693   fal
##  2 yahoo    tablet true           9    1.00    459   fal
##  3 direct   laptop true           0    1.00    996   fal
##  4 bing     tablet false          3   18.0     468   tru
##  5 yahoo    mobile true           9    1.00    955   fal
##  6 yahoo    laptop false          5    5.00    135   fal
##  7 yahoo    mobile true          10    1.00     75.0 fal
##  8 direct   mobile true          10    1.00    908   fal
##  9 bing     mobile false          3   19.0     209   fal
## 10 google   mobile true           6    1.00    208   fal
## # ... with 990 more rows, and 1 more variable: order_val
```

## Case Study: Select

For our case study, we need to select the columns order value and order items to calculate the AOV. We also need to select the device column as we are computing the AOV for different devices.

```r
select(ecom, device, order_value, order_items)
```

```
## # A tibble: 1,000 x 3
##    device order_value order_items
##    <chr>        <dbl>       <dbl>
##  1 laptop           0        0
##  2 tablet           0        0
##  3 laptop           0        0
##  4 tablet         434        6.00
##  5 mobile           0        0
##  6 laptop           0        0
##  7 mobile           0        0
##  8 mobile           0        0
##  9 mobile           0        0
## 10 mobile           0        0
## # ... with 990 more rows
```

## Case Study: Select

But we want the above data only for purchasers. We will combine `filter()` and `select()` functions to extract data related to purchasers.

```r
ecom1 <- filter(ecom, purchase == "true")
ecom2 <- select(ecom1, device, order_value, order_items)
ecom2
```

```
## # A tibble: 103 x 3
##    device order_value order_items
##    <chr>        <dbl>       <dbl>
##  1 tablet         434        6.00
##  2 mobile         651        3.00
##  3 tablet        1049        6.00
##  4 tablet        1304        9.00
##  5 tablet         622        5.00
##  6 tablet        1613        7.00
##  7 mobile         184        3.00
##  8 tablet         286        9.00
##  9 mobile         764        6.00
## 10 tablet        1667        6.00
## # ... with 93 more rows
```

## Step 3: Group data by devices

Since we want to compute the AOV for each device, we need to compute the total order value and total order items for each device. To achieve this, we will group the selected variables by device type. Using the `group_by()` function, we will group our case study data by device types. The first argument is the name of the data frame and the second argument is the name of the column based on which the data will be split. Let us look at a few examples:

## Group data by referrer type

```r
group_by(ecom, referrer)
```

```
## # A tibble: 1,000 x 11
## # Groups:   referrer [5]
##       id referrer device bouncers n_visit n_pages durati
##    <int> <chr>    <chr>  <chr>      <int>   <dbl>    <db
##  1     1 google   laptop true          10    1.00    693
##  2     2 yahoo    tablet true           9    1.00    459
##  3     3 direct   laptop true           0    1.00    996
##  4     4 bing     tablet false          3   18.0     468
##  5     5 yahoo    mobile true           9    1.00    955
##  6     6 yahoo    laptop false          5    5.00    135
##  7     7 yahoo    mobile true          10    1.00     75
##  8     8 direct   mobile true          10    1.00    908
##  9     9 bing     mobile false          3   19.0     209
## 10    10 google   mobile true           6    1.00    208
## # ... with 990 more rows, and 3 more variables: purchase
## #   order_items <dbl>, order_value <dbl>
```

## Case Study: Group

In the second line in the previous output, you can observe `Groups: referrer [5]`. The data is split into 5 groups as the referrer variable has 5 distinct values. For our case study, we need to group the data by device type.

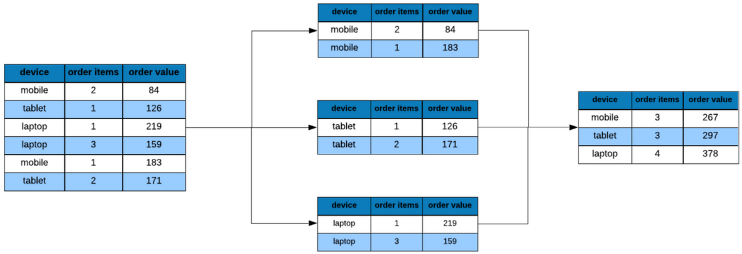
```r
ecom3 <- group_by(ecom2, device)
ecom3
```

```
## # A tibble: 103 x 3
## # Groups:   device [3]
##    device order_value order_items
##    <chr>        <dbl>       <dbl>
##  1 tablet         434        6.00
##  2 mobile         651        3.00
##  3 tablet        1049        6.00
##  4 tablet        1304        9.00
##  5 tablet         622        5.00
##  6 tablet        1613        7.00
##  7 mobile         184        3.00
##  8 tablet         286        9.00
##  9 mobile         764        6.00
## 10 tablet        1667        6.00
## # ... with 93 more rows
```

## Step 4: Total order value and order items

The next step is to compute the total order value and total order items for each device. We will use them to then compute the average order value. Now we need to reduce the order value and order items data to a single summary. We can achieve this using the `summarise()` function. The first argument is the name of a data frame and the subsequent arguments are functions that can generate a summary. For example, we can use `min`, `max`, `sum`, `mean` etc.

## Summarize



| device | order items | order value |
|---|---|---|
| mobile | 2 | 84 |
| tablet | 1 | 126 |
| laptop | 1 | 219 |
| laptop | 3 | 159 |
| mobile | 1 | 183 |
| tablet | 2 | 171 |

| device | order items | order value |
|---|---|---|
| mobile | 2 | 84 |
| mobile | 1 | 183 |

| device | order items | order value |
|---|---|---|
| tablet | 1 | 126 |
| tablet | 2 | 171 |

| device | order items | order value |
|---|---|---|
| laptop | 1 | 219 |
| laptop | 3 | 159 |

| device | order items | order value |
|---|---|---|
| mobile | 3 | 267 |
| tablet | 3 | 297 |
| laptop | 4 | 378 |

## Summarise

For our case study, we need the totals of order value and order items. What function can we use to obtain them? The `sum()` function will generate the sum of the values and hence we will use it inside the `summarise()` function. Remember, we need to provide a name to the summary being generated.

```
ecom4 <- summarise(ecom3, total_value = sum(order_value),
          total_items = sum(order_items))
ecom4
```

```
## # A tibble: 3 x 3
##   device total_value total_items
##   <chr>        <dbl>       <dbl>
## 1 laptop       56531         160
## 2 mobile       51504         184
## 3 tablet       51321         197
```

## Summarise

There you go, we have the total order value and total order items for each device type. Another way to achieve the above result is to use the `summarise_all()` function. How does that work? It generates the specified summary for all the columns in the data set except for the column based on which the data has been grouped. So we need to ensure that the data frame does not have any irrelevant columns.

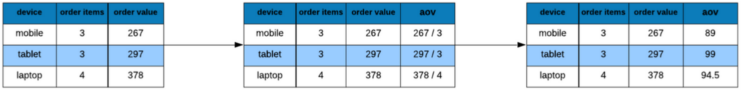
## Case Study: Summarise

In our case study, we have split the data based on the device type and we have 2 other columns which are order value and order items. If we use `summarise_all()` function, it will generate the summary for the two columns based on the function specified. To specify the functions, we need to use another argument `funs` and it can take any number of valid functions.

```r
ecom4 <- summarise_all(ecom3, funs(sum))
ecom4
```

```
## # A tibble: 3 x 3
##   device order_value order_items
##   <chr>        <dbl>       <dbl>
## 1 laptop       56531         160
## 2 mobile       51504         184
## 3 tablet       51321         197
```

## Mutate



| device | order items | order value |
|---|---|---|
| mobile | 3 | 267 |
| tablet | 3 | 297 |
| laptop | 4 | 378 |

| device | order items | order value | aov |
|---|---|---|---|
| mobile | 3 | 267 | 267 / 3 |
| tablet | 3 | 297 | 297 / 3 |
| laptop | 4 | 378 | 378 / 4 |

| device | order items | order value | aov |
|---|---|---|---|
| mobile | 3 | 267 | 89 |
| tablet | 3 | 297 | 99 |
| laptop | 4 | 378 | 94.5 |

## Step 5: Compute AOV

Now that we have the total order value and total order items for each device category, we can compute the AOV. We will create a new column to store the result. To create a new column, we will use the `mutate()` function. The first argument is the name of the data frame and the subsequent arguments are expressions for creating new columns based out of existing columns.

```r
ecom5 <- mutate(ecom4, aov = order_value / order_items)
ecom5
```

```
## # A tibble: 3 x 4
##   device order_value order_items   aov
##   <chr>        <dbl>       <dbl> <dbl>
## 1 laptop       56531         160   353
## 2 mobile       51504         184   280
## 3 tablet       51321         197   261
```

## Step 6: Select relevant columns

The last step is to select the relevant columns. We require the device type and the corresponding aov and hence we can get rid of other columns. Use the `select()` function to extract the relevant columns.

```r
ecom6 <- select(ecom5, device, aov)
ecom6
```

```
## # A tibble: 3 x 2
##   device   aov
##   <chr>  <dbl>
## 1 laptop   353
## 2 mobile   280
## 3 tablet   261
```

## Average Order Value

## AOV by Devices

Let us combine all the code from the above steps:

```
ecom1 <- filter(ecom, purchase == "true")
ecom2 <- select(ecom1, device, order_value, order_items)
ecom3 <- group_by(ecom2, device)
ecom4 <- summarise_all(ecom3, funs(sum))
ecom5 <- mutate(ecom4, aov = order_value / order_items)
ecom6 <- select(ecom5, device, aov)
ecom6
```

```
## # A tibble: 3 x 2
##   device    aov
##   <chr>   <dbl>
## 1 laptop    353
## 2 mobile    280
## 3 tablet    261
```

## AOV by Devices

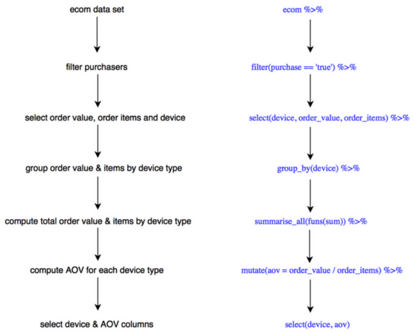
If you observe, at each step we create a new variable(data frame) and then use it as an input in the next step i.e. the output from one function becomes the input for another function. Can we achieve the final outcome i.e. `ecom6` without creating the intermediate data frames (ecom1 - ecom5)? Yes, we can. We will use the %>% operator to chain the above steps so that we can avoid creating the intermediate data frames. Let us see how to do that.

## AOV by Devices

```r
ecom %>%
  filter(purchase == 'true') %>%
  select(device, order_value, order_items) %>%
  group_by(device) %>%
  summarise_all(funs(sum)) %>%
  mutate(
    aov = order_value / order_items
  ) %>%
  select(device, aov)
```

```
## # A tibble: 3 x 2
##   device   aov
##   <chr>  <dbl>
## 1 laptop   353
## 2 mobile   280
## 3 tablet   261
```

## AOV by Devices


ecom data set
filter purchasers
select order value, order items and device
group order value & items by device type
compute total order value & items by device type
compute AOV for each device type
select device & AOV columns

```
ecom %>%
filter(purchase == 'true') %>%
select(device, order_value, order_items) %>%
group_by(device) %>%
summarise_all(funs(sum)) %>%
mutate(aov = order_value / order_items) %>%
select(device, aov)
```

## Practice Questions

- what is the average number of pages visited by purchasers and non-purchasers?
- what is the average time on site for purchasers vs non-purchasers?
- what is the average number of pages visited by purchasers and non-purchasers using mobile?

## Thank You

For more information please visit our website
www.rsquaredacademy.com