R Workshop: CLV and Data Manipulation

Contents

This tutorial demonstrates how to calculate customer lifetime value (CLV), and how to use dplyr for data manipulation.

You will need to follow the below packages for the session:

- readxl
- ggplot2
- dplyr

1 Customer Lifetime Value (CLV)

1.1 The Marketing Problem

We need to understand our customers or know if a specific customer is relevant to our organization. Therefore,

- We want to know how much value a customer is generating for the organization.
- We want to know how the generated value is going to evolve.
- We want to segment customers based on value.

So that we can perform specific actions for different customer groups.

1.2 Customer Lifetime Value (CLV)

The technique to solve the business problem is to understand the customer's value. We have many options to understand customer's value. One of the most relevant is Customer Lifetime Value (CLV).

CLV is a prediction of all the value a business will derive from its entire relationship with a customer.

The value generated by all our customers is called *Customer Equity* and is used to evaluate companies.

Main Concepts. There are many factors that affect customer's value and must be considered in the CLV calculation:

- *Cash flow*: the net present value of the income generated by the customer to the organization throughout their relationship with the organization.
- *Lifecycle*: the duration of the relationship with the organization
- *Maintenance costs*: associated costs to ensure that the flow of revenue per customer is achieved.
- *Risk costs*: risk associated with a client.
- *Acquisition costs*: costs and effort required to acquire a new customer.
- *Retention costs*: costs and effort required to retain a new customer.
- *Recommendation value*: impact of the recommendations of a customer in its sphere of influence on company revenue.
- *Segmentation improvements*: value of customer information in improving customer segmentation models

CLV formula

$$CLV = \sum_{n=0}^{T} \frac{(p_t - c_t)r^t}{(1+i)^t} - AC$$

where,

p_t = price paid by the customer in time t,

c_t = direct costs for customer service in time t,

i = discount rate or cost of money for the firm,

r = probability that the client returns to buy or is alive in time t

AC = acquisition cost, and

T = time horizon to estimate CLV.

A simplification

Let's consider that p and c are constant values and r is a decreasing function in time, then the formula becomes:

$$CLV = \sum_{n=0}^{\infty} \frac{(p-c)r^t}{(1+i)^t} = (p-c)(\frac{r}{1+i-r}) = m(\frac{r}{1+i-r})$$

where

$p - c = m$ = margin, r = retention rate, i = discount rate.

Final consideration

- Generally, we will have a dataset for each of the concepts of the formula.
- We'll have to estimate the rest
- Therefore, it is important to remember that: *Our ability to predict the future is limited by the fact that, to some extent, it is contained in the past.*

1.3 Implementation Process and Use Cases

The Implementation process is as follows:

- Discuss whether CLV fits as a metric in our business
- Identification and understanding of sources and metadata
- Extract, transform, clean and load data
- Choose the CLV method
- Analyze results and adjust parameters
- Present and explain the results

CLV is commonly used to:

- Create market strategies based on CLV
- Customer segmentation based on CLV
- Forecasting and customer evolution analysis across segments
- Create different communication, services and loyalty programs based on CLV
- Awake "non-active" customers
- Estimate the value of a company (startup, in the context of acquisition)

1.4 ARPU/ARPA as CLV approximation

Average Revenue per User (ARPU) or Average Revenue per Account (ARPA) can be used to calculate historical CLV. The process is given below:

- calculate the average revenue per customer per month (i.e., total revenue/ number of months since the customer joined)
- add them up
- multiply by 12 or 24 to get a one- or two-year CLV.

Exercise: Suppose Josep and Laura are your only customers, and their purchases look like this:

Customer Name	Purchase Date	Amount
Josep	Jan 1, 2023	$150

Customer Name	Purchase Date	Amount
Josep	May 15, 2023	\$50
Josep	Jun 15, 2023	\$100
Laura	May 1, 2023	\$45
Laura	Jun 15, 2023	\$75
Laura	Jun 30, 2023	\$100

Suppose today is July 1, 2023

The average monthly revenue from Josep is

$$(150 + 50 + 100)/6 = 50$$

and the average monthly revenue from Laura is

$$(45 + 75 + 100)/2 = 110$$

.

Adding these two numbers gives you an average monthly revenue per customer of $160/2 = 80$. To find a 12-month or 24-month CLV, multiply that number by 12 or 24.

The benefit of an ARPU approach is that it is simple to calculate, but it does not take into account changes in your customers' behaviour

1.5 How to implement CLV in R

Load the packages:

```r
library(ggplot2) #install if needed
library(readxl) #install if needed
```

Load data into a data frame from the Excel data sheet 2:

```r
CLV.df <- read_excel("Data_CLV.xlsx", sheet = "Ex2")
CLV.df
```

1.5.1 Exploratory Data Analysis

The Starting point is always exploratory data analysis.

Summary. What can we observe from the summary?

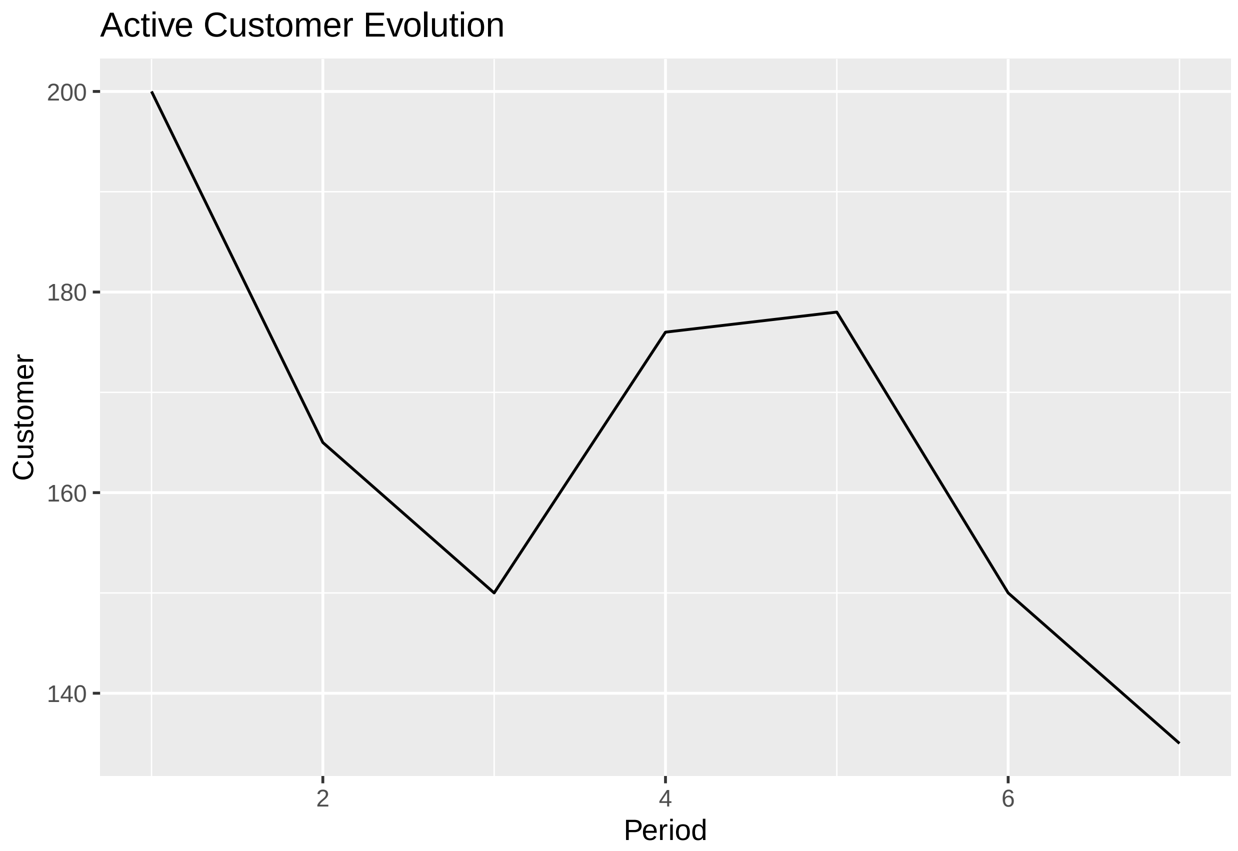
```r
summary(CLV.df)
```

```
##        t           active            p             c            i       
##  Min.   :1.0   Min.   :135.0   Min.   :375   Min.   :20   Min.   :0.01  
##  1st Qu.:2.5   1st Qu.:150.0   1st Qu.:375   1st Qu.:20   1st Qu.:0.01  
##  Median :4.0   Median :165.0   Median :375   Median :20   Median :0.01  
##  Mean   :4.0   Mean   :164.9   Mean   :375   Mean   :20   Mean   :0.01  
##  3rd Qu.:5.5   3rd Qu.:177.0   3rd Qu.:375   3rd Qu.:20   3rd Qu.:0.01  
##  Max.   :7.0   Max.   :200.0   Max.   :375   Max.   :20   Max.   :0.01  
##        r         
##  Min.   :0.6750  
##  1st Qu.:0.7500  
##  Median :0.8250  
##  Mean   :0.8114
##  3rd Qu.:0.8850
##  Max.   :0.9100
```

The summary shows that the price and cost are constant. The range of active customers is [135,200]. The range of retention ratio is [0.83,1.11].

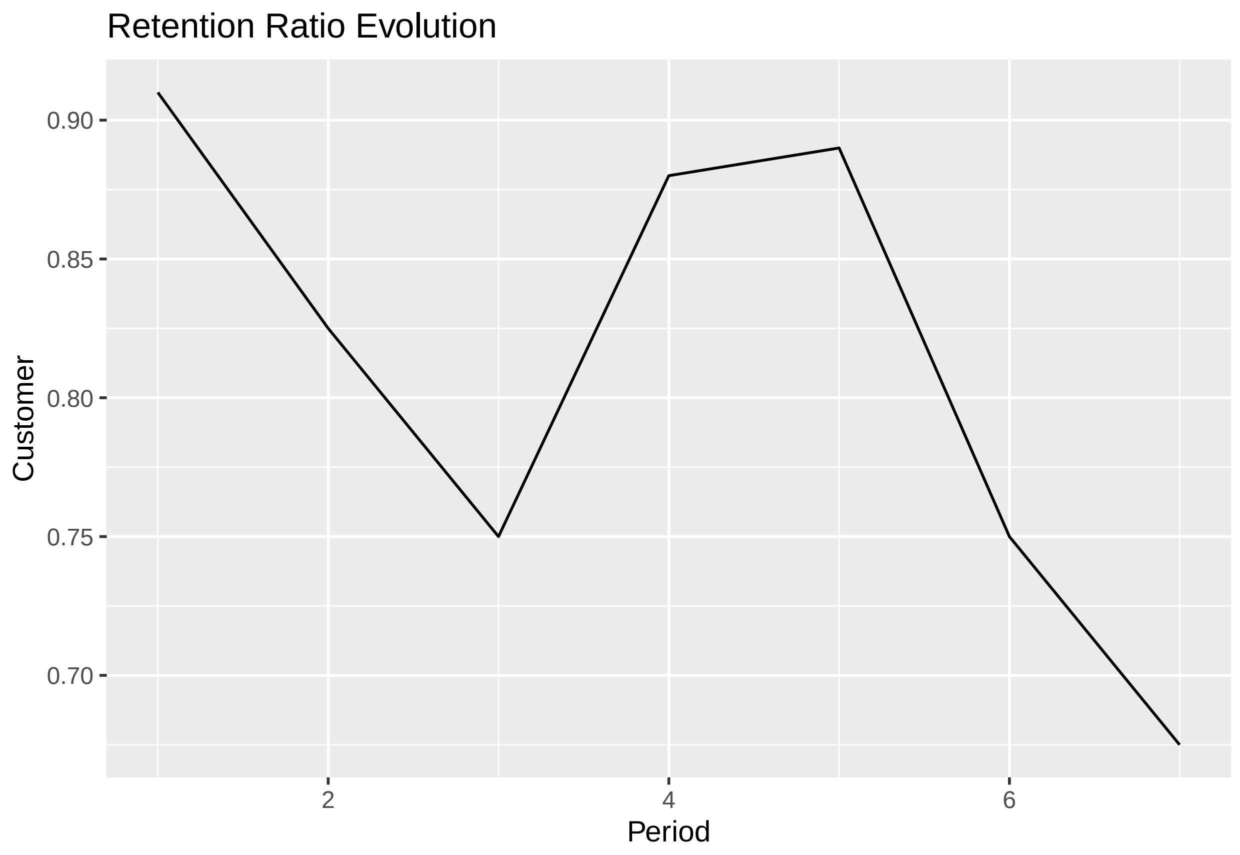
Customer evolution. We can create a line chart to understand how the number of customers is evolving. *ggplot()* is used to construct the initial plot object and is almost always followed by + to add components to the plot.

```r
ggplot(CLV.df, aes(x = t, y = active)) +
  geom_line() + ylab("Customer") +
  xlab("Period") + ggtitle("Active Customer Evolution")
```



Retention ratio evolution. We can create a line chart to understand how the retention ratio is evolving.

```r
ggplot(CLV.df, aes(x = t, y = r)) + geom_line() + ylab("Customer") +
  xlab("Period") +
  ggtitle("Retention Ratio Evolution")
```



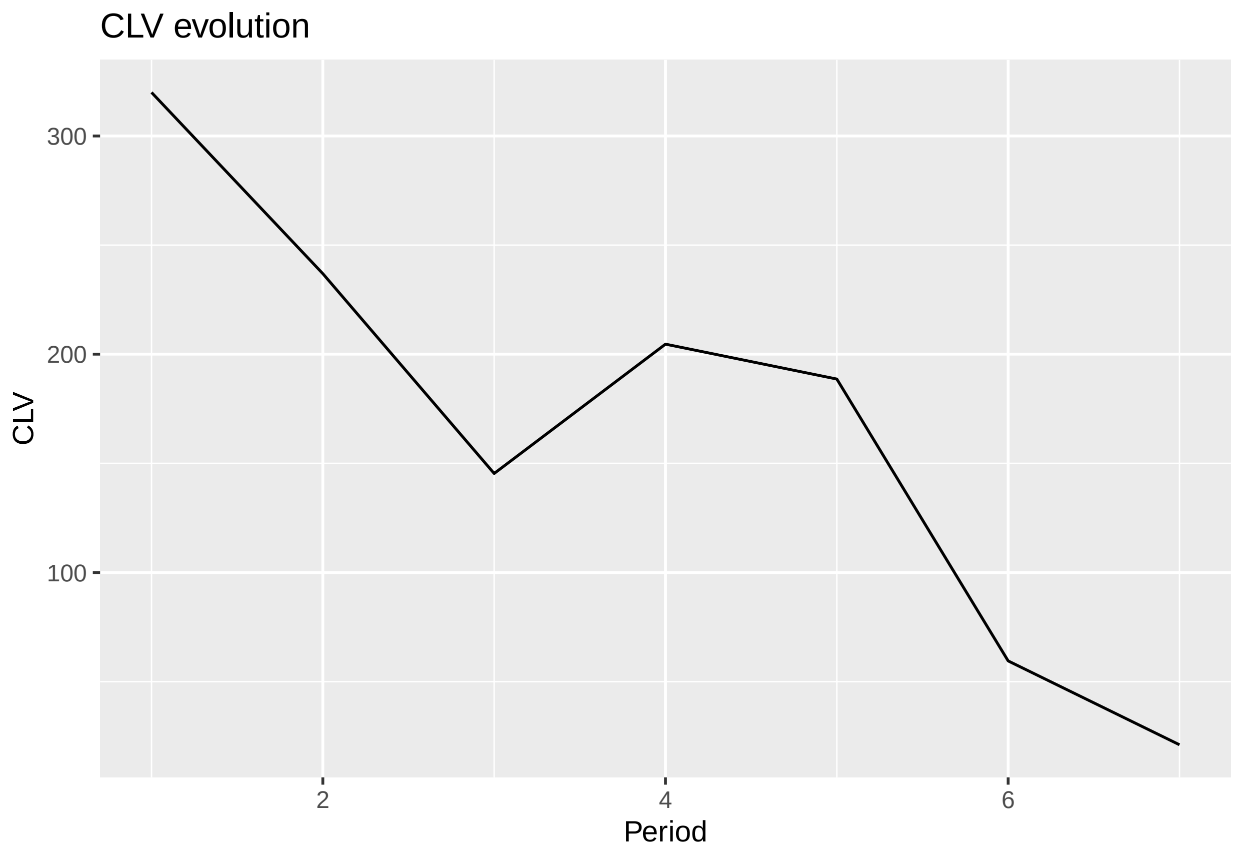
1.5.2 Calculate CLV

Now, we can calculate the (historic) CLV. First, we create the new column with CLV per period:

```
CLV.df$CLV <- (CLV.df$p-CLV.df$c)*CLV.df$r CLV.df$t/(1+CLV.df$i) CLV.df$t
CLV.df
```

CLV evolution. Now we create a chart to see how the CLV is evolving:

```
ggplot(CLV.df, aes(x = t, y = CLV)) +
  geom_line() + ggtitle("CLV evolution") +
  ylab("CLV") + xlab("Period")
```



Question: What do we observe? As we have some constant variables, the CLV only depends on the amount of customers. The chart is similar to the previous ones.

Final step. Finally, we can calculate the CLV value:

```
CLV <- apply(CLV.df, 2, sum)
CLV[7]
```

```
##      CLV
## 1175.932
```

Questions: What does this value mean?

1.6 Exercise:

What happens if the retention ratio has a constant value of 0.80?

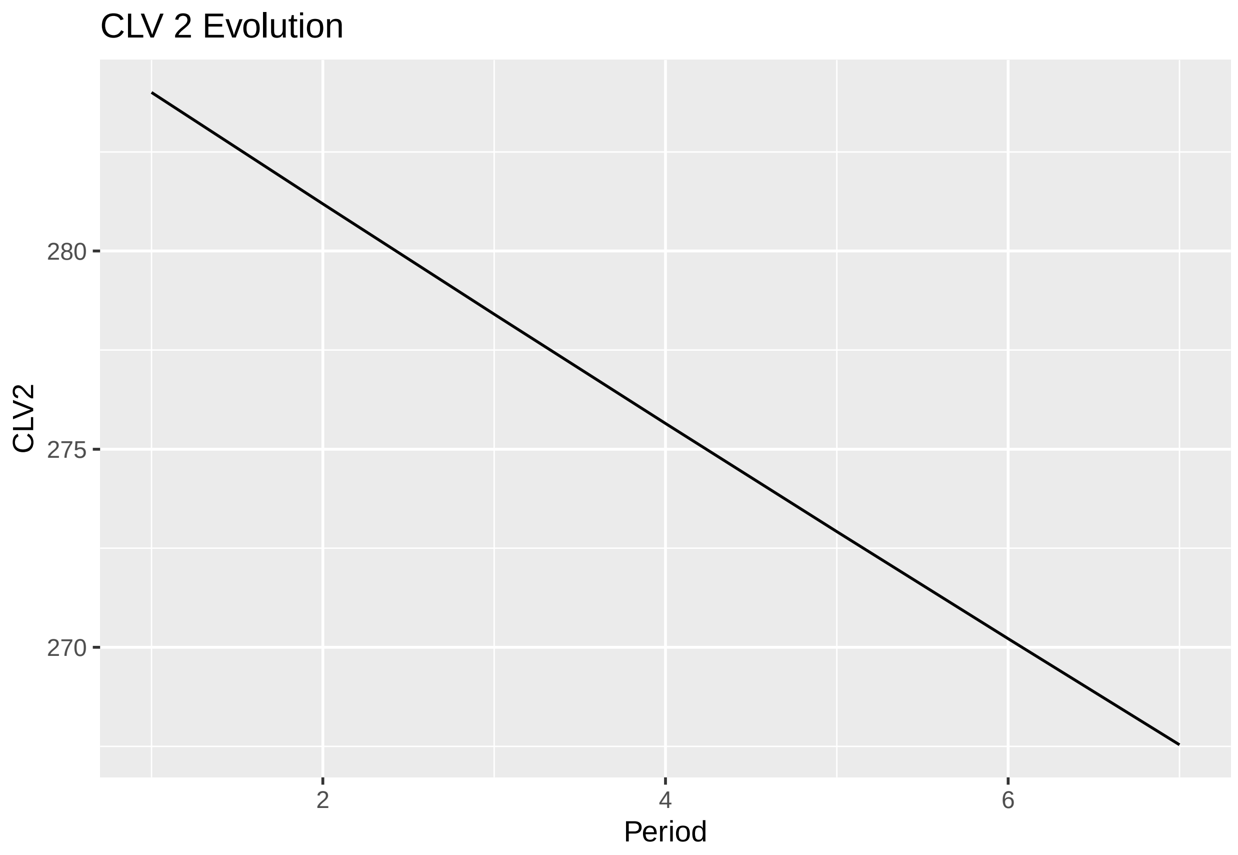
Solution

Let's add a new column:

```
CLV.df$CLV2 <- (CLV.df$p-CLV.df$c)*0.8/(1+CLV.df$i)^(CLV.df$t-1)
CLV.df
```

We can create a new chart as well.

```
ggplot(CLV.df, aes(x = t, y = CLV2)) + geom_line() +
ylab("CLV2") + xlab("Period") + labs(title = "CLV 2 Evolution")
```



And finally, we can compare the previous CLV with the new one.

```
CLV <- apply(CLV.df, 2, sum)
CLV[7]
```

```
##      CLV
## 1175.932
```

```
CLV[8]
```

```
##     CLV2
## 1929.915
```

2 Data Manipulation

You will need to install the package *dplyr*. The easiest way to get *dplyr* is to install *tidyverse*: *install.packages("tidyverse")*.

tidyverse is a collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structure. The following packages are included in the core tidyverse:

- *ggplot2* is a system for declaratively creating graphics. * *dplyr* provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges.
- *tidyr* provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable.
- *readr* provides a fast and friendly way to read rectangular data (like csv, tsc, and fwf)
- *stringr* provides a cohesive set of functions designed to make working with strings as easy as possible.
- *forcats* provides a collection of useful tools that solve common problems with factors (categorical variables)
- *purr*
- *tibble*

Alternatively, install just *dplyr*: *install.packages("dplyr", dependencies=TRUE). Dependencies = TRUE* will install the package with dependencies required for this package.

Let us load the package and the dataset. The mtcars dataset comes with the dplyr package. We can use the packages to explore data.

```r
library(dplyr)
```

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

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```r
head(mtcars)
```

```r
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

2.1 Manipulate cases

filter() picks cases based on their values.

```r
filter(mtcars, gear == 4)    # filtering can be used with all standard logical operators
```

```r
filter(mtcars, gear == 3 | gear == 4)    # i.e. >, <, =>, <=, !=, ==
```

```r
filter(mtcars,  mpg > 21)
```

arrange() orders the rows of a data frame by the values of selected columns.

```r
arrange(mtcars, gear)
```

```r
arrange(mtcars, gear, mpg)
```

```r
arrange(mtcars, desc(mpg))      # desc sorts in descending order
```

2.2 Manipulate variables

select() picks variables based on their names.

```r
select(mtcars, gear, mpg, hp)
```

```r
select(mtcars, -drat) # - sign before name means remove variable/column
```

slice() selects certain rows by row number.

```r
slice(mtcars, 1:3)
```

sample_n() and *sample_frac()* select random rows:

```r
sample_n(mtcars, 5) # numbers of rows to select
```

```r
sample_frac(mtcars, .1) #fraction of rows to select
```

mutate() adds new variables that are functions of existing variables.

```r
mutate(mtcars, wt_mpg = wt * mpg)
```

```r
mutate(mtcars, mpg_mean = mean(mpg), mpg_diff_mean = mpg - mpg_mean)
```

```r
mutate(mtcars, mpg_mean_diff = mpg - mean(mpg))
```

2.3 Summarise cases

summarise() create a new data frame. It will have a single row summarizing all observations, and contain one column for each of the summary statistics that you have specified. If there are grouping variables, it will have one row for each combination of grouping variables.

```r
summarise(mtcars, sd(disp))
```

```r
summarise(mtcars, median(mpg), mean(mpg), max(mpg), min(mpg))
```

2.4 Group cases

Use *group_by()* to create a "grouped" copy of a table. *dplyr* function will manipulate each "group" separately and then combine the results.

```r
group_by(mtcars, cyl)
```

2.5 Pipes %>%

What if you wanted to filter and select the same data? Pipe operator *%>%* lets you take the output of one function and send it directly to the input of the next function. You can type the pipe %>% with *Ctrl+Shift+M* if you have Windows or *Cmd+Shift+M* if you have Mac.

When using pipes remember the following: 1. first argument is always a data frame 2. subsequent arguments say what to do with the data frame 3. the sequence of arguments will return a new data frame (data frame used in the first argument is not changed unless you do so by assignation such as using "<-")

```r
mtcars %>%              # take dataframe, then
    group_by(gear) %>%          # group it by gear, then
    summarise(mean(mpg))        # summarise the mean mpg for each level of gear
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
mtcars %>%
  filter(mpg > 21) %>%
  select(gear, mpg, hp) %>%
  arrange(gear)
```

If you want to create a new object with this smaller version of the data, we could do so by assigning it a new name:

```r
mtcars_mpg <- mtcars %>%
  filter(mpg > 21) %>%
  select(gear, mpg, hp) %>%
  arrange(gear)
mtcars_mpg
```

2.6 Excercise

Now you can test your data manipulation skills. First import the red wine dataset using:

```r
redwine <- read.csv("Data_redwine.csv", header=TRUE)
head(redwine)
```

The data is wine testing data, containing 12 variables:

Variables based on physicochemical tests: 1 - fixed acidity 2 - volatile acidity 3 - citric acid 4 - residual sugar 5 - chlorides 6 - free sulfur dioxide 7 - total sulfur dioxide 8 - density 9 - pH 10 - sulphates 11 - alcohol

Variable based on sensory data: 12 - quality (score between 0 and 10)

Now try to create code for the following:

1. Create a data frame containing only wines with a fixed acidity of 8 or greater.
2. Create a data frame containing only wines with a quality of 7.
3. Create a data frame with only 2 columns (pH and quality)
4. Create a data frame with only 2 columns (pH and quality) where the wines are ordered in ascending order of pH
5. Create a data frame with only 2 columns (pH and quality) where the wines are ordered in descending order of quality
6. Create a data frame with only wines without citric acid listed in order of descending quality.
7. Create a version of the red wine dataset with an additional column containing alcohol times density
8. Create a data frame summarising every variable in the red wine dataset.