If statements/data selection/NA

Introduction

Please install the packages:

- ggplot2
- tufte

What is TRUE/FALSE?

Today we will be working a lot with `TRUE` and `FALSE`.

Let us start off with a recap of last week. `TRUE/FALSE` are `boolean` variables. We can print them:

```
print(TRUE)
```

```
## [1] TRUE
```

```
print(FALSE)
```

```
## [1] FALSE
```

We can also assign them:

```
a <- TRUE
print(a)
```

```
## [1] TRUE
```

```
b <- FALSE
print(b)
```

```
## [1] FALSE
```

We can also **CREATE** them from questions:

Note the use of ==
 Double == is a **QUESTION**
 Single = is an **ASSIGNMENT**

```
3 == 4
```

```
## [1] FALSE
```

We can also save the result of the question ("is 3 equal to 4?") in the variable `a` and then print the result:

```
a <- 3 == 4
a
```

```
## [1] FALSE
```

We can also ask other questions:

! means **NOT**

```
3 != 4  # is 3 not equal to 4?
```

```
## [1] TRUE
```

```
3 < 4  # is 3 less than 4?
```

```
## [1] TRUE
```

```
3 <= 4  # is 3 less than or equal to 4?
```

```
## [1] TRUE
```

```
3 > 4  # is 3 greater than 4?
```

```
## [1] FALSE
```

```
3 >= 4  # is 3 greater than or equal to 4?
```

```
## [1] FALSE
```

```
c(1, 2, 3, 4) == 3  # FOUR questions simultaneously
```

```
## [1] FALSE FALSE  TRUE FALSE
```

```
c(3, 4) < c(4, 1)  # TWO questions simultaneously (not recommended)
```

```
## [1]  TRUE FALSE
```

The above questions (==, !=, <, <=, >, >=) **MUST** have either:

1. Have one single value on the right side of the question (recommended)
2. Have variables that are the same length on the left and the right of the question (not recommended)

The number of questions we ask **ALWAYS** corresponds to the length of the variable on the left side of the question

The below questions can have different length variables on each side of the question:

```
3 %in% c(1, 2)  # is 3 equal to 1 or 2?
```

```
## [1] FALSE
```

```
!3 %in% c(1, 2)  # is 3 NOT equal to 1 or 2?
```

```
## [1] TRUE
```

```
c(1, 3) %in% c(2, 3, 4, 5)  # TWO questions simultaneously
```

```
## [1] FALSE  TRUE
```

We do not use **c** as a variable because it is already a function

We can obviously repeat all of these questions using variables instead of numbers:

```
a <- 3
b <- 4
x <- c(1, 2)
a != b  # is 3 not equal to 4?
```

```
## [1] TRUE
```

```
a < b  # is 3 less than 4?
```

```
## [1] TRUE
```

```
a <= b  # is 3 less than or equal to 4?
```

```
## [1] TRUE
```

```
a > b  # is 3 greater than 4?
```

```
## [1] FALSE
```

```
a >= b  # is 3 greater than or equal to 4?
```

```
## [1] FALSE
```

```
a %in% x  # is 3 equal to 1 or 2?
```

```
## [1] FALSE
```

```
!a %in% x  # is 3 NOT equal to 1 or 2?
```

```
## [1] TRUE
```

Exercises

Check to see if `myAge` is greater than or equal to `ageLimitForBuyingAlcohol`:

```
myAge <- 13
ageLimitForBuyingAlcohol <- 18

# your code goes here
```

Check to see if `myFriendsAges` are greater than or equal to `ageLimitForBuyingAlcohol`:

```
myFriendsAges <- c(15, 16, 16, 14, 20)
ageLimitForBuyingAlcohol <- 18

# your code goes here
```

Check to see if `placeWhereILive` exists in `validCityNames`:

```
placeWhereILive <- "Norway"
validCityNames <- c("Oslo", "Bergen", "Trondheim")

# your code goes here
```

Check to see if `placesWhereMyFriendsLive` are **NOT** in `placesInTheEU`:

```
placesWhereMyFriendsLive <- c("Australia", "UK",
    "London", "Paris")
placesInTheEU <- c("France", "Paris", "Sweden")

# your code goes here
```

AND/OR

If we have multiple questions we can join them together using `&` (AND) and `|` (OR).

```
TRUE & TRUE

## [1] TRUE

TRUE & FALSE

## [1] FALSE

FALSE & FALSE

## [1] FALSE

TRUE | TRUE

## [1] TRUE

TRUE | FALSE

## [1] TRUE

FALSE | FALSE

## [1] FALSE
```

Put into more practical examples:

Whenever using | and & it is always smart to use () to ensure your order of operations is correct

```
(3 < 4) & (2 < 5)

## [1] TRUE

(3 < 4) & (5 < 2)
```

```
## [1] FALSE
```

```
(4 < 3) & (5 < 2)
```

```
## [1] FALSE
```

```
(3 < 4) | (2 < 5)
```

```
## [1] TRUE
```

```
(3 < 4) | (5 < 2)
```

```
## [1] TRUE
```

```
(4 < 3) | (5 < 2)
```

```
## [1] FALSE
```

Data Selection

Vectors

When we want to select data, we can either provide the location of the data we want (index) or a vector of `TRUE/FALSE` that essentially specifies `include/exclude` for every datapoint.

```
myData <- c("a", "b", "c", "d", "e")
myData
```

```
## [1] "a" "b" "c" "d" "e"
```

```
myData[2]  # using indexes
```

```
## [1] "b"
```

```
myData[c(2, 4)]  # using indexes
```

```
## [1] "b" "d"
```

```
myData[c(FALSE, TRUE, TRUE, FALSE, FALSE)]  # using TRUE/FALSE
```

```
## [1] "b" "c"
```

data.frames

We use a data.frame provided to us by the R package `ggplot2` (you might need to install this package):

```
data(diamonds, package = "ggplot2")
diamonds <- diamonds[1:20, ]
diamonds
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21   Premium     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29   Premium     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22   Premium     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20   Premium     E     SI2  60.2    62
## 16  0.32   Premium     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price    x    y    z
## 1    326 3.95 3.98 2.43
## 2    326 3.89 3.84 2.31
## 3    327 4.05 4.07 2.31
## 4    334 4.20 4.23 2.63
## 5    335 4.34 4.35 2.75
## 6    336 3.94 3.96 2.48
## 7    336 3.95 3.98 2.47
## 8    337 4.07 4.11 2.53
## 9    337 3.87 3.78 2.49
## 10   338 4.00 4.05 2.39
## 11   339 4.25 4.28 2.73
## 12   340 3.93 3.90 2.46
## 13   342 3.88 3.84 2.33
## 14   344 4.35 4.37 2.71
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
## 17   348 4.31 4.34 2.68
## 18   351 4.23 4.29 2.70
## 19   351 4.23 4.26 2.71
## 20   351 4.21 4.27 2.66
```

Remember that each column of a data.frame is a vector, so we can reuse what we learnt with vectors (selecting the first three values in the vector):

```
diamonds$carat[c(1:3)]
```

```
## [1] 0.23 0.21 0.23
```

```
diamonds$cut[c(1:3)]
```

```
## [1] Ideal   Premium Good
## 5 Levels: Fair < Good < ... < Ideal
```

We can also select the first three rows of a data.frame:

We reference cells in data.frames by [ROWS, COLUMNS]. So diamonds[c(1:3),] means 'first three rows, all the columns'

```
diamonds[c(1:3), ]
```

```
##   carat     cut color clarity depth table
## 1  0.23   Ideal     E     SI2  61.5    55
## 2  0.21 Premium     E     SI1  59.8    61
## 3  0.23    Good     E     VS1  56.9    65
##   price    x    y    z
## 1   326 3.95 3.98 2.43
## 2   326 3.89 3.84 2.31
## 3   327 4.05 4.07 2.31
```

The first two rows and the second and third columns (not recommended):

```
diamonds[c(1:2), c(2:3)]
```

```
##       cut color
## 1   Ideal     E
## 2 Premium     E
```

We can also specify the columns by name (recommended):

```
diamonds[c(1:2), c("cut", "color")]
```

```
##       cut color
## 1   Ideal     E
## 2 Premium     E
```

We can also use a vector of `TRUE/FALSE` instead of indexes:

```
diamonds[c(TRUE, TRUE, FALSE, FALSE, FALSE, FALSE,
    FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
    FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
    FALSE, FALSE), c("cut", "color")]
```

```
##       cut color
## 1   Ideal     E
## 2 Premium     E
```

Most importantly we can select rows by asking questions:

```
isCutPremium <- diamonds$cut == "Premium"
isCutPremium
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE FALSE
##  [7] FALSE FALSE FALSE FALSE FALSE FALSE
## [13]  TRUE FALSE  TRUE  TRUE FALSE FALSE
## [19] FALSE FALSE
```

```
diamonds[isCutPremium, ]
```

```
##    carat     cut color clarity depth table
## 2   0.21 Premium     E     SI1  59.8    61
## 4   0.29 Premium     I     VS2  62.4    58
## 13  0.22 Premium     F     SI1  60.4    61
## 15  0.20 Premium     E     SI2  60.2    62
## 16  0.32 Premium     E      I1  60.9    58
##    price    x    y    z
## 2    326 3.89 3.84 2.31
## 4    334 4.20 4.23 2.63
## 13   342 3.88 3.84 2.33
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
```

And we can make this more elegant:

```
diamonds[diamonds$cut == "Premium", ]
```

```
##    carat     cut color clarity depth table
## 2   0.21 Premium     E     SI1  59.8    61
## 4   0.29 Premium     I     VS2  62.4    58
## 13  0.22 Premium     F     SI1  60.4    61
## 15  0.20 Premium     E     SI2  60.2    62
## 16  0.32 Premium     E      I1  60.9    58
##    price    x    y    z
## 2    326 3.89 3.84 2.31
## 4    334 4.20 4.23 2.63
## 13   342 3.88 3.84 2.33
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
```

We can also ask multiple questions:

```
diamonds[diamonds$cut == "Premium" & diamonds$color ==
    "E", ]
```

```
##    carat     cut color clarity depth table
## 2   0.21 Premium     E     SI1  59.8    61
## 15  0.20 Premium     E     SI2  60.2    62
## 16  0.32 Premium     E      I1  60.9    58
##    price    x    y    z
## 2    326 3.89 3.84 2.31
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
```

```
diamonds[diamonds$cut == "Premium" | diamonds$color ==
    "E", ]
```

```
##    carat     cut color clarity depth table
## 1   0.23   Ideal     E     SI2  61.5    55
## 2   0.21 Premium     E     SI1  59.8    61
## 3   0.23    Good     E     VS1  56.9    65
## 4   0.29 Premium     I     VS2  62.4    58
## 9   0.22    Fair     E     VS2  65.1    61
## 13  0.22 Premium     F     SI1  60.4    61
## 15  0.20 Premium     E     SI2  60.2    62
## 16  0.32 Premium     E      I1  60.9    58
##    price    x    y    z
## 1    326 3.95 3.98 2.43
## 2    326 3.89 3.84 2.31
## 3    327 4.05 4.07 2.31
## 4    334 4.20 4.23 2.63
## 9    337 3.87 3.78 2.49
## 13   342 3.88 3.84 2.33
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
```

```
diamonds[diamonds$cut %in% c("Premium", "Good") |
    diamonds$color == "E", ]
```

```
##    carat     cut color clarity depth table
## 1   0.23   Ideal     E     SI2  61.5    55
## 2   0.21 Premium     E     SI1  59.8    61
## 3   0.23    Good     E     VS1  56.9    65
## 4   0.29 Premium     I     VS2  62.4    58
## 5   0.31    Good     J     SI2  63.3    58
## 9   0.22    Fair     E     VS2  65.1    61
## 11  0.30    Good     J     SI1  64.0    55
## 13  0.22 Premium     F     SI1  60.4    61
## 15  0.20 Premium     E     SI2  60.2    62
## 16  0.32 Premium     E      I1  60.9    58
## 18  0.30    Good     J     SI1  63.4    54
## 19  0.30    Good     J     SI1  63.8    56
##    price    x    y    z
## 1    326 3.95 3.98 2.43
## 2    326 3.89 3.84 2.31
## 3    327 4.05 4.07 2.31
## 4    334 4.20 4.23 2.63
## 5    335 4.34 4.35 2.75
## 9    337 3.87 3.78 2.49
## 11   339 4.25 4.28 2.73
## 13   342 3.88 3.84 2.33
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
## 18   351 4.23 4.29 2.70
## 19   351 4.23 4.26 2.71
```

```
diamonds[diamonds$cut %in% c("Premium", "Good") &
    diamonds$price < 350, ]
```

```
##    carat     cut color clarity depth table
## 2   0.21 Premium     E     SI1  59.8    61
## 3   0.23    Good     E     VS1  56.9    65
## 4   0.29 Premium     I     VS2  62.4    58
## 5   0.31    Good     J     SI2  63.3    58
## 11  0.30    Good     J     SI1  64.0    55
## 13  0.22 Premium     F     SI1  60.4    61
## 15  0.20 Premium     E     SI2  60.2    62
## 16  0.32 Premium     E      I1  60.9    58
##    price    x    y    z
## 2    326 3.89 3.84 2.31
## 3    327 4.05 4.07 2.31
## 4    334 4.20 4.23 2.63
## 5    335 4.34 4.35 2.75
## 11   339 4.25 4.28 2.73
## 13   342 3.88 3.84 2.33
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
```

Once we have selected the rows we are interested in, we can then choose columns/variables:

```
diamonds[diamonds$cut == "Premium" & diamonds$color ==
    "E", ]$price
```

```
## [1] 326 345 345
```

```
diamonds[diamonds$cut == "Premium" | diamonds$color ==
    "E", ]$price
```

```
## [1] 326 326 327 334 337 342 345 345
```

```
diamonds[diamonds$cut %in% c("Premium", "Good") |
    diamonds$color == "E", ]$price
```

```
##  [1] 326 326 327 334 335 337 339 342 345 345
## [11] 351 351
```

```
diamonds[diamonds$cut %in% c("Premium", "Good") &
    diamonds$price < 350, ]$price
```

```
## [1] 326 327 334 335 339 342 345 345
```

You can start to work with the data to get summary statistics:

```
mean(diamonds[diamonds$cut == "Premium" & diamonds$color ==
    "E", ]$price)
```

```
## [1] 338.6667
```

```
sd(diamonds[diamonds$cut == "Premium" & diamonds$color ==
    "E", ]$price)
```

```
## [1] 10.96966
```

```
quantile(diamonds[diamonds$cut == "Premium" &
    diamonds$color == "E", ]$price)
```

```
##    0%   25%   50%   75%  100% 
## 326.0 335.5 345.0 345.0 345.0
```

Remember that behind each of these row selections is a vector containing TRUE/FALSE that includes/excludes certain rows:

```
diamonds$cut == "Premium" & diamonds$color ==
    "E"
```

```
##  [1] FALSE  TRUE FALSE FALSE FALSE FALSE
##  [7] FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE  TRUE  TRUE FALSE FALSE
## [19] FALSE FALSE
```

```
diamonds$cut == "Premium" | diamonds$color ==
    "E"
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
##  [7] FALSE FALSE  TRUE FALSE FALSE FALSE
## [13]  TRUE FALSE  TRUE  TRUE FALSE FALSE
## [19] FALSE FALSE
```

```
diamonds$cut %in% c("Premium", "Good") | diamonds$color ==
    "E"
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [7] FALSE FALSE  TRUE FALSE  TRUE FALSE
## [13]  TRUE FALSE  TRUE  TRUE FALSE  TRUE
## [19]  TRUE FALSE
```

```
diamonds$cut %in% c("Premium", "Good") & diamonds$price <
    350
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE
##  [7] FALSE FALSE FALSE FALSE  TRUE FALSE
## [13]  TRUE FALSE  TRUE  TRUE FALSE FALSE
## [19] FALSE FALSE
```

Data manipulation/cleaning

Once you have selected your rows, you can also manipulate your data:

```
diamonds  # before
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21   Premium     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29   Premium     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22   Premium     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20   Premium     E     SI2  60.2    62
## 16  0.32   Premium     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price    x    y    z
## 1    326 3.95 3.98 2.43
## 2    326 3.89 3.84 2.31
## 3    327 4.05 4.07 2.31
## 4    334 4.20 4.23 2.63
## 5    335 4.34 4.35 2.75
## 6    336 3.94 3.96 2.48
## 7    336 3.95 3.98 2.47
## 8    337 4.07 4.11 2.53
## 9    337 3.87 3.78 2.49
## 10   338 4.00 4.05 2.39
## 11   339 4.25 4.28 2.73
## 12   340 3.93 3.90 2.46
## 13   342 3.88 3.84 2.33
## 14   344 4.35 4.37 2.71
## 15   345 3.79 3.75 2.27
## 16   345 4.38 4.42 2.68
## 17   348 4.31 4.34 2.68
## 18   351 4.23 4.29 2.70
## 19   351 4.23 4.26 2.71
## 20   351 4.21 4.27 2.66
```

```
diamonds[diamonds$cut == "Premium" & diamonds$color ==
    "E", ]$y <- 1000  # manipulate
diamonds  # after
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21   Premium     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29   Premium     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22   Premium     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20   Premium     E     SI2  60.2    62
## 16  0.32   Premium     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price    x       y    z
## 1    326 3.95    3.98 2.43
## 2    326 3.89 1000.00 2.31
## 3    327 4.05    4.07 2.31
## 4    334 4.20    4.23 2.63
## 5    335 4.34    4.35 2.75
## 6    336 3.94    3.96 2.48
## 7    336 3.95    3.98 2.47
## 8    337 4.07    4.11 2.53
## 9    337 3.87    3.78 2.49
## 10   338 4.00    4.05 2.39
## 11   339 4.25    4.28 2.73
## 12   340 3.93    3.90 2.46
## 13   342 3.88    3.84 2.33
## 14   344 4.35    4.37 2.71
## 15   345 3.79 1000.00 2.27
## 16   345 4.38 1000.00 2.68
## 17   348 4.31    4.34 2.68
## 18   351 4.23    4.29 2.70
## 19   351 4.23    4.26 2.71
## 20   351 4.21    4.27 2.66
```

This is how we clean our data. Here we change `cut=="Premium"` to `Ideal`:

```
diamonds  # before
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21   Premium     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29   Premium     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22   Premium     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20   Premium     E     SI2  60.2    62
## 16  0.32   Premium     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price    x       y    z
## 1    326 3.95    3.98 2.43
## 2    326 3.89 1000.00 2.31
## 3    327 4.05    4.07 2.31
## 4    334 4.20    4.23 2.63
## 5    335 4.34    4.35 2.75
## 6    336 3.94    3.96 2.48
## 7    336 3.95    3.98 2.47
## 8    337 4.07    4.11 2.53
## 9    337 3.87    3.78 2.49
## 10   338 4.00    4.05 2.39
## 11   339 4.25    4.28 2.73
## 12   340 3.93    3.90 2.46
## 13   342 3.88    3.84 2.33
## 14   344 4.35    4.37 2.71
## 15   345 3.79 1000.00 2.27
## 16   345 4.38 1000.00 2.68
## 17   348 4.31    4.34 2.68
## 18   351 4.23    4.29 2.70
## 19   351 4.23    4.26 2.71
## 20   351 4.21    4.27 2.66
```

```
diamonds[diamonds$cut == "Premium", ]$cut <- "Ideal"  # manipulate
diamonds  # after
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price    x       y    z
## 1    326 3.95    3.98 2.43
## 2    326 3.89 1000.00 2.31
## 3    327 4.05    4.07 2.31
## 4    334 4.20    4.23 2.63
## 5    335 4.34    4.35 2.75
## 6    336 3.94    3.96 2.48
## 7    336 3.95    3.98 2.47
## 8    337 4.07    4.11 2.53
## 9    337 3.87    3.78 2.49
## 10   338 4.00    4.05 2.39
## 11   339 4.25    4.28 2.73
## 12   340 3.93    3.90 2.46
## 13   342 3.88    3.84 2.33
## 14   344 4.35    4.37 2.71
## 15   345 3.79 1000.00 2.27
## 16   345 4.38 1000.00 2.68
## 17   348 4.31    4.34 2.68
## 18   351 4.23    4.29 2.70
## 19   351 4.23    4.26 2.71
## 20   351 4.21    4.27 2.66
```

We can also add different columns together:

```
diamonds  # before
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price    x       y    z
## 1    326 3.95    3.98 2.43
## 2    326 3.89 1000.00 2.31
## 3    327 4.05    4.07 2.31
## 4    334 4.20    4.23 2.63
## 5    335 4.34    4.35 2.75
## 6    336 3.94    3.96 2.48
## 7    336 3.95    3.98 2.47
## 8    337 4.07    4.11 2.53
## 9    337 3.87    3.78 2.49
## 10   338 4.00    4.05 2.39
## 11   339 4.25    4.28 2.73
## 12   340 3.93    3.90 2.46
## 13   342 3.88    3.84 2.33
## 14   344 4.35    4.37 2.71
## 15   345 3.79 1000.00 2.27
## 16   345 4.38 1000.00 2.68
## 17   348 4.31    4.34 2.68
## 18   351 4.23    4.29 2.70
## 19   351 4.23    4.26 2.71
## 20   351 4.21    4.27 2.66
```

```
diamonds[diamonds$cut == "Ideal", ]$x <- diamonds[diamonds$cut ==
    "Ideal", ]$y + diamonds[diamonds$cut == "Ideal",
    ]$z
diamonds  # after
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326    6.41    3.98 2.43
## 2    326 1002.31 1000.00 2.31
## 3    327    4.05    4.07 2.31
## 4    334    6.86    4.23 2.63
## 5    335    4.34    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9    337    3.87    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11   339    4.25    4.28 2.73
## 12   340    6.36    3.90 2.46
## 13   342    6.17    3.84 2.33
## 14   344    7.08    4.37 2.71
## 15   345 1002.27 1000.00 2.27
## 16   345 1002.68 1000.00 2.68
## 17   348    7.02    4.34 2.68
## 18   351    4.23    4.29 2.70
## 19   351    4.23    4.26 2.71
## 20   351    4.21    4.27 2.66
```

When we need to make multiple references to particular row selections, it is often cleaner to create a variable that contains the row selections:

```
# get row selection
rows <- diamonds$cut %in% c("Ideal", "Good") |
    diamonds$color == "E"
rows
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [7] FALSE FALSE  TRUE FALSE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [19]  TRUE FALSE
```

```
diamonds  # before
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326    6.41    3.98 2.43
## 2    326 1002.31 1000.00 2.31
## 3    327    4.05    4.07 2.31
## 4    334    6.86    4.23 2.63
## 5    335    4.34    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9    337    3.87    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11   339    4.25    4.28 2.73
## 12   340    6.36    3.90 2.46
## 13   342    6.17    3.84 2.33
## 14   344    7.08    4.37 2.71
## 15   345 1002.27 1000.00 2.27
## 16   345 1002.68 1000.00 2.68
## 17   348    7.02    4.34 2.68
## 18   351    4.23    4.29 2.70
## 19   351    4.23    4.26 2.71
## 20   351    4.21    4.27 2.66
```

```
diamonds[rows, ]$x <- diamonds[rows, ]$y + diamonds[rows,
    ]$z * 100
diamonds  # after
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 3    327  235.07    4.07 2.31
## 4    334  267.23    4.23 2.63
## 5    335  279.35    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9    337  252.78    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11   339  277.28    4.28 2.73
## 12   340  249.90    3.90 2.46
## 13   342  236.84    3.84 2.33
## 14   344  275.37    4.37 2.71
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
## 18   351  274.29    4.29 2.70
## 19   351  275.26    4.26 2.71
## 20   351    4.21    4.27 2.66
```

Creating new data.frames

Once you have selected your rows, you can also save it to a new data.frame:

```
myNewDataFrame <- diamonds[diamonds$cut == "Ideal" &
    diamonds$color == "E", ]
myNewDataFrame
```

```
##    carat   cut color clarity depth table
## 1   0.23 Ideal     E     SI2  61.5    55
## 2   0.21 Ideal     E     SI1  59.8    61
## 15  0.20 Ideal     E     SI2  60.2    62
## 16  0.32 Ideal     E      I1  60.9    58
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
```

Exercises

Task 1: Select all rows with:

- colour equals E or I AND
- price less than 400

Solution 1:

```
diamonds[diamonds$color %in% c("E", "I") & diamonds$price <
    400, ]
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 7   0.24 Very Good     I    VVS1  62.3    57
## 9   0.22      Fair     E     VS2  65.1    61
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 3    327  235.07    4.07 2.31
## 4    334  267.23    4.23 2.63
## 7    336    3.95    3.98 2.47
## 9    337  252.78    3.78 2.49
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
```

Task 2: Select all rows with:

- `depth` less than 63
- `price` more than 300

Solution 2:

```
# your code goes here
```

Task 3:
Set `z` to 400 for all rows with:

- `cut` not `Ideal`
- `price` more than 300

Solution 3:

```
# your code goes here
```

Control flow (if statements)

Today's module has so far only focused on using `TRUE`/`FALSE` to select data.

We can also use `TRUE`/`FALSE` to change the flow of the program.

```
if (TRUE) {
    print("a")
} else {
    print("b")
}
```

```
## [1] "a"
```

Above we have used `TRUE`/`FALSE` to select which lines of code would run. This is called an `if` statement.

These will be used more frequently in further modules, but at this stage you could use them in conjuction with `flags` at the start of your script.

```
analyseCheapDiamonds <- TRUE  # a flag

if (analyseCheapDiamonds) {
    analysisData <- diamonds[diamonds$price <
        350, ]
} else {
    analysisData <- diamonds
}

analysisData
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 3    327  235.07    4.07 2.31
## 4    334  267.23    4.23 2.63
## 5    335  279.35    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9    337  252.78    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11   339  277.28    4.28 2.73
## 12   340  249.90    3.90 2.46
## 13   342  236.84    3.84 2.33
## 14   344  275.37    4.37 2.71
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
```

In the above script the user can change the top line `analyseCheapDiamonds <- TRUE` to be `TRUE`/`FALSE` if they want `analysisData` to be only cheap diamonds or all of the diamonds. The rest of the script would continue using `analysisData`, which would mean that you could write 1 analysis script that could be used for 2 (or more) different datasets.

We can also write more complicated `if` statements. In the following code, the flag `analysis` can take the values: `main`, `sensitivity1`, `sensitivity2` and changes the dataset accordingly:

```
analysis <- "main"

if (analysis == "main") {
    analysisData <- diamonds
} else if (analysis == "sensitivity1") {
    analysisData <- diamonds[diamonds$cut != "Good",
        ]
} else if (analysis == "sensitivity2") {
    analysisData <- diamonds[diamonds$color ==
        "E", ]
} else {
    stop("not a valid analysis!")
}

analysisData
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 3    327  235.07    4.07 2.31
## 4    334  267.23    4.23 2.63
## 5    335  279.35    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9    337  252.78    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11   339  277.28    4.28 2.73
## 12   340  249.90    3.90 2.46
## 13   342  236.84    3.84 2.33
## 14   344  275.37    4.37 2.71
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
## 18   351  274.29    4.29 2.70
## 19   351  275.26    4.26 2.71
## 20   351    4.21    4.27 2.66
```

If you were running regression analyses, then you could take it even further and have different outcomes, exposures, and confounders for each analysis:

```
analysis <- "main"

if (analysis == "main") {
    analysisData <- diamonds
    outcome <- "price"
    exposure <- "carat"
    confounders <- c("color", "clarity", "depth")
} else if (analysis == "sensitivity1") {
    analysisData <- diamonds[diamonds$cut != "Good",
        ]
    outcome <- "price"
    exposure <- "carat"
    confounders <- c("cut", "clarity")
} else if (analysis == "sensitivity2") {
    analysisData <- diamonds[diamonds$color ==
        "E", ]
    outcome <- "price"
    exposure <- "cut"
    confounders <- c("depth")
} else {
    stop("not a valid analysis!")
}

# create new data.frame
analysisData <- analysisData[, c(outcome, exposure,
    confounders)]
analysisData
```

```
##    price carat color clarity depth
## 1    326  0.23     E     SI2  61.5
## 2    326  0.21     E     SI1  59.8
## 3    327  0.23     E     VS1  56.9
## 4    334  0.29     I     VS2  62.4
## 5    335  0.31     J     SI2  63.3
## 6    336  0.24     J    VVS2  62.8
## 7    336  0.24     I    VVS1  62.3
## 8    337  0.26     H     SI1  61.9
## 9    337  0.22     E     VS2  65.1
## 10   338  0.23     H     VS1  59.4
## 11   339  0.30     J     SI1  64.0
## 12   340  0.23     J     VS1  62.8
## 13   342  0.22     F     SI1  60.4
## 14   344  0.31     J     SI2  62.2
## 15   345  0.20     E     SI2  60.2
## 16   345  0.32     E      I1  60.9
## 17   348  0.30     I     SI2  62.0
## 18   351  0.30     J     SI1  63.4
## 19   351  0.30     J     SI1  63.8
## 20   351  0.30     J     SI1  62.7
```

```
# rename variables in the data.frame
names(analysisData)[c(1:2)] <- c("outcome", "exposure")
names(analysisData)[c(3:ncol(analysisData))] <- paste0("confounder",
    1:length(confounders))
analysisData
```

```
##    outcome exposure confounder1 confounder2
## 1      326     0.23           E         SI2
## 2      326     0.21           E         SI1
## 3      327     0.23            E          VS1
## 4      334     0.29            I          VS2
## 5      335     0.31            J          SI2
## 6      336     0.24            J         VVS2
## 7      336     0.24            I         VVS1
## 8      337     0.26            H          SI1
## 9      337     0.22            E          VS2
## 10     338     0.23            H          VS1
## 11     339     0.30            J          SI1
## 12     340     0.23            J          VS1
## 13     342     0.22            F          SI1
## 14     344     0.31            J          SI2
## 15     345     0.20            E          SI2
## 16     345     0.32            E           I1
## 17     348     0.30            I          SI2
## 18     351     0.30            J          SI1
## 19     351     0.30            J          SI1
## 20     351     0.30            J          SI1
##    confounder3
## 1         61.5
## 2         59.8
## 3         56.9
## 4         62.4
## 5         63.3
## 6         62.8
## 7         62.3
## 8         61.9
## 9         65.1
## 10        59.4
## 11        64.0
## 12        62.8
## 13        60.4
## 14        62.2
## 15        60.2
## 16        60.9
## 17        62.0
## 18        63.4
## 19        63.8
## 20        62.7
```

```
# ’analyse’ some of the data.frame
mean(analysisData$outcome)
```

```
## [1] 339.4
```

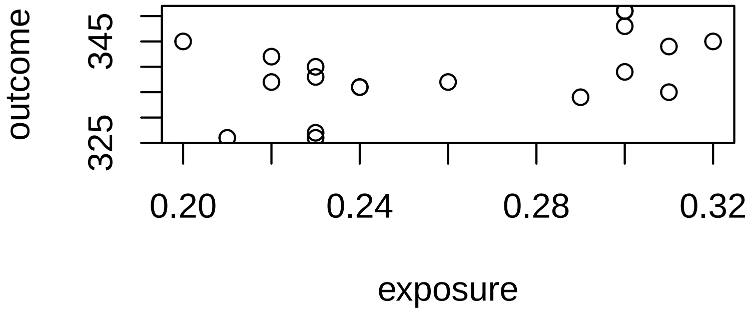
```
mean(analysisData$exposure)
```

```
## [1] 0.262
```

```
cor(analysisData$outcome, analysisData$exposure)
```

```
## [1] 0.5250715
```

```
plot(outcome ~ exposure, data = analysisData)
```



NA/Missing/Invalid Data

There are two kinds of missing/invalid data:

- What **YOU** know is missing/invalid
- What **R** knows is missing/invalid

```
myWeightOverTime <- c(80, 86, 76, 80, -9, -9,
    -9, 0, 0, 0, 89)
myWeightOverTime
```

```
##  [1] 80 86 76 80 -9 -9 -9  0  0  0 89
```

```
mean(myWeightOverTime)
```

```
## [1] 34.90909
```

Here, it is obvious to **me** that the values between 70 and 90 are valid, while `-9` and `0` are obviously missing data.

For **R** all of these values are valid and real. The **ONLY** value that **R** considers to be missing/invalid is `NA`. So we need to translate my knowledge of the data, and my understanding of what values are missing/invalid into `NA`s that R can understand:

```
myWeightOverTime[(myWeightOverTime < 70) | (myWeightOverTime >
    90)] <- NA
myWeightOverTime
```

```
##  [1] 80 86 76 80 NA NA NA NA NA NA 89
```

We can now analyse the data:

```
mean(myWeightOverTime)
```

```
## [1] NA
```

This doesn't work, because there are NAs in the data. We need to explicitly tell R to ignore the NAs:

```
mean(myWeightOverTime, na.rm = TRUE)
```

```
## [1] 82.2
```

Of course, all of this applies to data.frames:

```
diamonds  # before
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 3    327  235.07    4.07 2.31
## 4    334  267.23    4.23 2.63
## 5    335  279.35    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9    337  252.78    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11   339  277.28    4.28 2.73
## 12   340  249.90    3.90 2.46
## 13   342  236.84    3.84 2.33
## 14   344  275.37    4.37 2.71
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
## 18   351  274.29    4.29 2.70
## 19   351  275.26    4.26 2.71
## 20   351    4.21    4.27 2.66
```

```
diamonds[diamonds$cut %in% c("Good", "Fair"),
    ]$price <- NA  # manipulation
diamonds  # after
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 3   0.23      Good     E     VS1  56.9    65
## 4   0.29     Ideal     I     VS2  62.4    58
## 5   0.31      Good     J     SI2  63.3    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 9   0.22      Fair     E     VS2  65.1    61
## 10  0.23 Very Good     H     VS1  59.4    61
## 11  0.30      Good     J     SI1  64.0    55
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 18  0.30      Good     J     SI1  63.4    54
## 19  0.30      Good     J     SI1  63.8    56
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 3     NA  235.07    4.07 2.31
## 4    334  267.23    4.23 2.63
## 5     NA  279.35    4.35 2.75
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 9     NA  252.78    3.78 2.49
## 10   338    4.00    4.05 2.39
## 11    NA  277.28    4.28 2.73
## 12   340  249.90    3.90 2.46
## 13   342  236.84    3.84 2.33
## 14   344  275.37    4.37 2.71
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
## 18    NA  274.29    4.29 2.70
## 19    NA  275.26    4.26 2.71
## 20   351    4.21    4.27 2.66
```

```
mean(diamonds$price)
```

```
## [1] NA
```

```
mean(diamonds$price, na.rm = TRUE)
```

```
## [1] 339.1429
```

We can now introduce the function (question) `is.na`:

```
is.na(4)
```

```
## [1] FALSE
```

```
is.na("hello")
```

```
## [1] FALSE
```

```
is.na(NA)
```

```
## [1] TRUE
```

And if we apply it to a vector that contains `NA`s:

```
diamonds$price
```

```
##  [1] 326 326  NA 334  NA 336 336 337  NA 338
## [11]  NA 340 342 344 345 345 348  NA  NA 351
```

```
is.na(diamonds$price)
```

```
##  [1] FALSE FALSE  TRUE FALSE  TRUE FALSE
##  [7] FALSE FALSE  TRUE FALSE  TRUE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE  TRUE
## [19]  TRUE FALSE
```

So we can then use `is.na` in row selections, the same as all of the other questions (==, !=, <, <=, >, >=, %in%):

```
diamonds[is.na(diamonds$price), ]
```

```
##    carat  cut color clarity depth table
## 3   0.23 Good     E     VS1  56.9    65
## 5   0.31 Good     J     SI2  63.3    58
## 9   0.22 Fair     E     VS2  65.1    61
## 11  0.30 Good     J     SI1  64.0    55
## 18  0.30 Good     J     SI1  63.4    54
## 19  0.30 Good     J     SI1  63.8    56
##    price      x    y    z
## 3     NA 235.07 4.07 2.31
## 5     NA 279.35 4.35 2.75
## 9     NA 252.78 3.78 2.49
## 11    NA 277.28 4.28 2.73
## 18    NA 274.29 4.29 2.70
## 19    NA 275.26 4.26 2.71
```

```
diamonds[!is.na(diamonds$price), ]
```

```
##    carat       cut color clarity depth table
## 1   0.23     Ideal     E     SI2  61.5    55
## 2   0.21     Ideal     E     SI1  59.8    61
## 4   0.29     Ideal     I     VS2  62.4    58
## 6   0.24 Very Good     J    VVS2  62.8    57
## 7   0.24 Very Good     I    VVS1  62.3    57
## 8   0.26 Very Good     H     SI1  61.9    55
## 10  0.23 Very Good     H     VS1  59.4    61
## 12  0.23     Ideal     J     VS1  62.8    56
## 13  0.22     Ideal     F     SI1  60.4    61
## 14  0.31     Ideal     J     SI2  62.2    54
## 15  0.20     Ideal     E     SI2  60.2    62
## 16  0.32     Ideal     E      I1  60.9    58
## 17  0.30     Ideal     I     SI2  62.0    54
## 20  0.30 Very Good     J     SI1  62.7    59
##    price       x       y    z
## 1    326  246.98    3.98 2.43
## 2    326 1231.00 1000.00 2.31
## 4    334  267.23    4.23 2.63
## 6    336    3.94    3.96 2.48
## 7    336    3.95    3.98 2.47
## 8    337    4.07    4.11 2.53
## 10   338    4.00    4.05 2.39
## 12   340  249.90    3.90 2.46
## 13   342  236.84    3.84 2.33
## 14   344  275.37    4.37 2.71
## 15   345 1227.00 1000.00 2.27
## 16   345 1268.00 1000.00 2.68
## 17   348  272.34    4.34 2.68
## 20   351    4.21    4.27 2.66
```

Sum and Mean on TRUE/FALSE

You can use `sum` to count how many observations are true:

```
sum(c(TRUE, TRUE, FALSE, FALSE, FALSE, NA), na.rm = T)
```

```
## [1] 2
```

```
sum(diamonds$price < 350, na.rm = T)
```

```
## [1] 13
```

You can use `mean` to see the proportion of values that are true:

```
mean(c(TRUE, TRUE, FALSE, FALSE, FALSE, NA), na.rm = T)
```

```
## [1] 0.4
```

```
mean(diamonds$price < 350, na.rm = T)
```

```
## [1] 0.9285714
```