One Pager: Tips for Finding Data, Bias, and Datacamp R

Tips for Finding Data

The three tips for finding data are availability, data cleanliness, and good documentation. Availability ensures that the research topic is neither too broad nor too narrow for research documentation to be collected and read. Data cleanliness is achieved when we look at the data sources and see if the data can be cleaned up in any possible way. Good documentation is a factor that we need to look out for when finding data and this can be identified easily by finding out what each variable in the dataset means and how the data was collected, provided that the data and research was quantitative. However, if the research was qualitative, good documentation can be determined by understanding the story behind the documents, thinking about the potential pieces that might be overlooked in the current research present, and filling in those gaps in the current research paper we were writing.

Bias

Bias is a form of prejudice against a thing, person, or group in an unfair way. There are many different types of bias, and the ones most commonly encountered in research papers are publication bias, reporting bias, and confirmation bias. Publication bias is when significant or positive results are more likely to be published. This also means that studies with no statistical significance or data are less likely to publish these studies. Reporting bias is when research focuses on some desirable results while omitting undesirable or non-statistically significant results, and this can also occur when the subjects that are part of the clinical trials of the experiment fail to disclose real information, as seen with drug testing. In simpler terms, reporting bias refers to omitting or hiding unexpected or non-significant results. Confirmation bias is when a researcher looks for or favors a certain result that conforms to their opinion. When the results are contrary to the prior beliefs of the researcher, the researcher questions the experiment. The results are amplified or ignored to fit an expectation or belief. Lastly, recall bias occurs when events or exposures are misremembered by subjects of clinical studies. Recall bias is also when a participant of a study misremembers an event, primarily due to the participant remembering a few things and obscuring the rest.

Datacamp R:

Addition: `+`
Subtraction: `-`
Multiplication: `*`
Division: `/`
Exponentiation: `^`
Modulo: `%%`
Variable assignment → `my_var <- 4`
Print value → my_var
Add variables → my_var + my_var2

One Pager: Tips for Finding Data, Bias, and Datacamp R

Decimal values like `4.5` are called **numerics**.
Whole numbers like `4` are called **integers**. Integers are also numerics.
Boolean values (`TRUE` or `FALSE`) are called **logical**.
Text (or string) values are called **characters**.

Check class → class(my_var)

```
numeric_vector <- c(1, 2, 3)
character_vector <- c("a", "b", "c")
```

```
numeric_vector <- c(1, 10, 49)
character_vector <- c("a", "b", "c")

# Complete the code for boolean_vector
boolean_vector <- c(TRUE, FALSE, TRUE)
boolean_vector
```

Win and lose money → positive and negative numbers

```
some_vector <- c("John Doe", "poker player")
names(some_vector) <- c("Name", "Profession")
c(1, 2, 3) + c(4, 5, 6)
c(1 + 4, 2 + 5, 3 + 6)
c(5, 7, 9)
a <- c(1, 2, 3)
b <- c(4, 5, 6)
c <- a + b
total_poker <- sum(poker_vector)
total_poker > total_roulette
```

```
# Define a new variable based on a selection
poker_wednesday <- poker_vector[3]
poker_vector[c(1, 5)]
```

```
roulette_selection_vector <- roulette_vector[2:5]
poker_vector["Monday"]
poker_vector[c("Monday","Tuesday")]
c(4, 5, 6) > 5
[1] FALSE FALSE TRUE
```

`<` for less than
`>` for greater than
`<=` for less than or equal to
`>=` for greater than or equal to
`==` for equal to each other
`!=` not equal to each other

```
poker_vector[selection_vector]
rownames(my_matrix) <- row_names_vector
colnames(my_matrix) <- col_names_vector
matrix(1:9, byrow = TRUE, nrow = 3)
```

```
rowSums(my_matrix)
big_matrix <- cbind(matrix1, matrix2, vector1 ...)
```

```
# Combine both Star Wars trilogies in one matrix
all_wars_matrix <- rbind(star_wars_matrix, star_wars_matrix2)
all_wars_matrix
```

```
# Total revenue for US and non-US
total_revenue_vector <- colSums(all_wars_matrix)
```

- `my_matrix[1,2]` selects the element at the first row and second column.
- `my_matrix[1:3,2:4]` results in a matrix with the data on the rows 1, 2, 3 and columns 2, 3, 4.

If you want to select all elements of a row or a column, no number is needed before or after the comma, respectively:

- `my_matrix[,1]` selects all elements of the first column.
- `my_matrix[1,]` selects all elements of the first row.

Similar to what you have learned with vectors, the standard operators like `+`, `-`, `/`, `*`, etc. work in an element-wise way on matrices in R.

For example, `2 * my_matrix` multiplies each element of `my_matrix` by two.

As a newly-hired data analyst for Lucasfilm, it is your job to find out how many visitors went to each movie for each geographical area. You already have the total revenue figures in `all_wars_matrix`. Assume that the price of a ticket was 5 dollars. Simply dividing the box office numbers by this ticket price gives you the number of visitors.

```
sex_vector <- c("Male","Female","Female","Male","Male")
factor_sex_vector <- factor(sex_vector)
levels(factor_vector) <- c("name1", "name2",...)
survey_vector <- c("M", "F", "F", "M", "M")
summary(my_var)
```

Factor_survey_vector[1] > Factor_survey_vector[2]

```
factor(some_vector,
       ordered = TRUE,
       levels = c("lev1", "lev2" ...))
```

Head() → first observations
Tail() → last observations
Str() → rapid overview
Data frame → data.frame(variable1, variable2, …)

- `my_df[1,2]` selects the value at the first row and second column in `my_df`.
- `my_df[1:3,2:4]` selects rows 1, 2, 3 and columns 2, 3, 4 in `my_df`.

Sometimes you want to select all elements of a row or column. For example, `my_df[1, ]` selects all elements of the first row. Let us now apply this technique on `planets_df`!

Suppose you want to select the first three elements of the `type` column. One way to do this is

```
planets_df[1:3,2]
```

A possible disadvantage of this approach is that you have to know (or look up) the column number of `type`, which gets hard if you have a lot of variables. It is often easier to just make use of the variable name:

```
planets_df[1:3,"type"]
```

You will often want to select an entire column, namely one specific variable from a data frame. If you want to select all elements of the variable `diameter`, for example, both of these will do the trick:

```
planets_df[,3]
planets_df[,"diameter"]
```

However, there is a short-cut. If your columns have names, you can use the `$` sign:

```
planets_df$diameter
```

```
subset(my_df, subset = some_condition)
```

```
a <- c(100, 10, 1000)
order(a)
[1] 2 1 3
```

```
my_list <- list(comp1, comp2 ...)
my_list <- list(name1 = your_comp1,
                name2 = your_comp2)
my_list <- list(your_comp1, your_comp2)
names(my_list) <- c("name1", "name2")
```

```
shining_list <- list(moviename = mov, actors = act, reviews = rev)
shining_list[[1]]
shining_list[["reviews"]]
shining_list$reviews
```