5 Tips for R

MHirni

2024-09-18

Introduction

Welcome to **5 Helpful Tips for R Users**! In this tutorial, we'll guide you through essential tips for utilizing `R` whether you are just starting or more familiar with the language. Whether you're exploring the IDE or visualizing data, these tips will make your journey more enjoyable.

To introduce the language, `R` is an object-oriented programming (OOP) language with two main elements—*objects* and *functions*. Objects are data structures like vectors, matrices, and data frames. Functions are commands that perform operations on objects. You can create, manipulate, and analyze objects using functions. Alas, we present the five tips below!

Tip 1: Save Time! Shortcuts Are All Around `R`.

`R` is open source, and users can contribute to its development. As a result, we have incredible IDEs like RStudio that make coding more accessible. There are also keyboard shortcuts, packages containing extra functions, and a vast community to help you along the way (e.g., check out StackExchange).

Some Shortcuts Specifically within RStudio Include:

- **Ctrl/Command + Enter**: Run the current line or selection
- **Ctrl/Command + Shift + M**: Insert a pipe (`%>%`) for tidyverse users
- Buttons to run code, clean environments, save projects, and more!

A markdown file like this one is a great place to see and practice shortcuts. Here, you will find "chunks" of code you can run with the click of a button. Markdown files are a great way to share code, results, and explanations in one place. You can also control output style, add LaTeX equations, and code in different languages all in one document.

Here's a snippet to test:

```
x <- rnorm(100)
mean(x)
```

```
## [1] -0.02805147
```

In this code above, `x` is assigned 100 random numbers from a normal distribution, and `mean(x)` calculates the mean of `x`. Try running this code using the shortcuts mentioned above!

Tip 2: Anything goes! Import any Data Source.

You might be pulling CSV files for your own research, or maybe you're working from a system like SharePoint at MU. Using Waqar Ali's dataset about cat characteristics, here's how you'd typically load data with a .csv file.

```
# Import a CSV file
cats_df <- read.csv("./cats.csv")

# Check out the beginning of a dataset by using the function head()
head(cats_df)
```

```
##              Breed Age..Years. Weight..kg.         Color Gender
## 1     Russian Blue          19           7 Tortoiseshell Female
## 2 Norwegian Forest          19           9 Tortoiseshell Female
## 3        Chartreux           3           3         Brown Female
## 4          Persian          13           6         Sable Female
## 5          Ragdoll          10           8         Tabby   Male
## 6           Ocicat           9           8          Blue Female
```

Note that I used a "." to indicate the current working directory which works for Unix-based systems. You can set a directory using `setwd()` and paste a pathway to `cats.csv`, or in RStudio, go to File -> Import Dataset -> From Text (base) to import data.

There are ways to import Excel, SPSS, SAS and other formatted files if you take a look. If you import data from an source like SharePoint (or have your computer connected to a cloud), the process may involve a secure download link or an API connection. Always ensure you've got the correct permissions and file paths!

Tip 3: What's in the Box? Inspect Your Objects.

In R, knowing your object types makes information more transparent and can help you understand errors. Use `class()`, `str()`, and `summary()` to figure out what you're working with. This will save you from unexpected output when you manipulate or analyze your data.

```
class(cats_df)  # Class of the object
```

```
## [1] "data.frame"
```

```
summary(cats_df) # Summary of the object
```

```
##     Breed            Age..Years.     Weight..kg.      Color
##  Length:1000        Min.   : 1.00   Min.   :2.00   Length:1000
##  Class :character   1st Qu.: 5.00   1st Qu.:4.00   Class :character
##  Mode  :character   Median :10.00   Median :6.00   Mode  :character
##                     Mean   :10.21   Mean   :5.55
##                     3rd Qu.:15.00   3rd Qu.:7.00
##                     Max.   :19.00   Max.   :9.00
##     Gender
##  Length:1000
##  Class :character
```

```
##  Mode  :character
##
##
##
```

```
str(cats_df)    # Structure of the object
```

```
## 'data.frame':    1000 obs. of  5 variables:
##  $ Breed      : chr  "Russian Blue" "Norwegian Forest" "Chartreux" "Persian" ...
##  $ Age..Years.: int  19 19 3 13 10 9 6 12 2 12 ...
##  $ Weight..kg.: int  7 9 3 6 8 8 5 3 7 3 ...
##  $ Color      : chr  "Tortoiseshell" "Tortoiseshell" "Brown" "Sable" ...
##  $ Gender     : chr  "Female" "Female" "Female" "Female" ...
```

We see in the `str` output that some names look a little strange. We can overwrite column object names using the `colnames()` function:

```
# Rename columns
colnames(cats_df) <- c("Breed", "Age", "Weight_kg", "Color", "Gender")
str(cats_df)    # Check the new column names
```

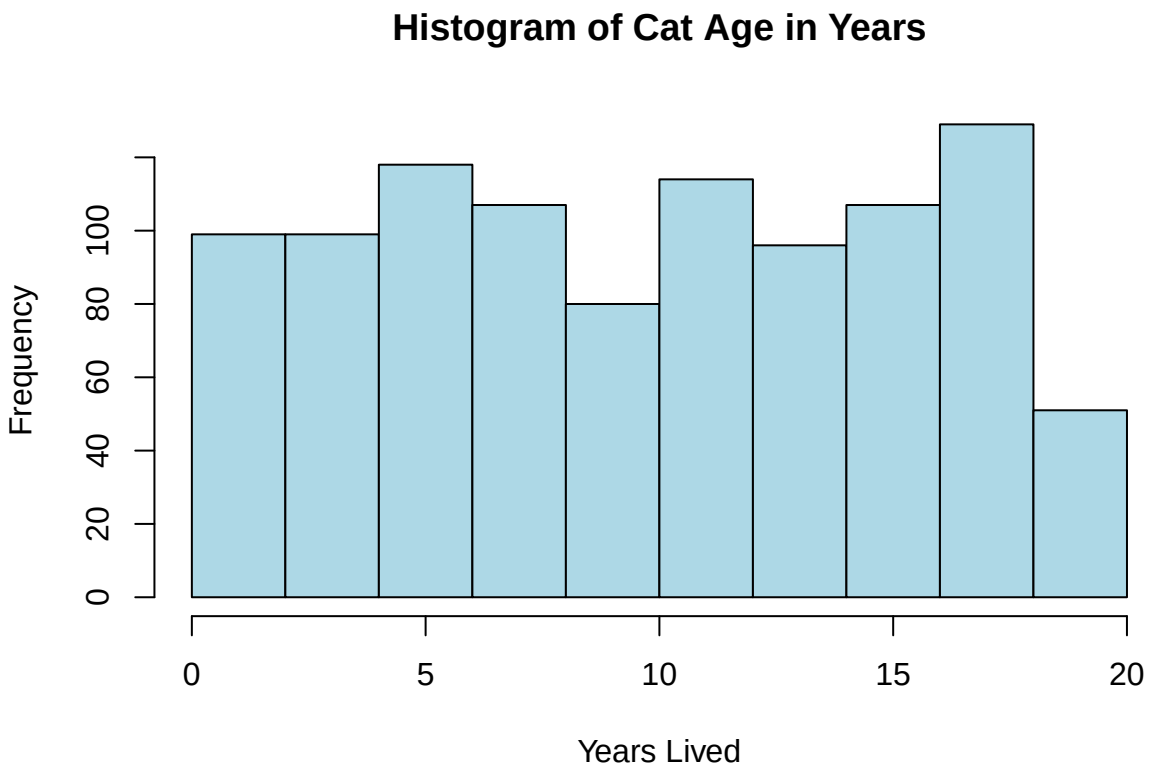
```
## 'data.frame':    1000 obs. of  5 variables:
##  $ Breed    : chr  "Russian Blue" "Norwegian Forest" "Chartreux" "Persian" ...
##  $ Age      : int  19 19 3 13 10 9 6 12 2 12 ...
##  $ Weight_kg: int  7 9 3 6 8 8 5 3 7 3 ...
##  $ Color    : chr  "Tortoiseshell" "Tortoiseshell" "Brown" "Sable" ...
##  $ Gender   : chr  "Female" "Female" "Female" "Female" ...
```

If you know how to inspect objects, you can better identify issues, use appropriate code for different object types, and understand

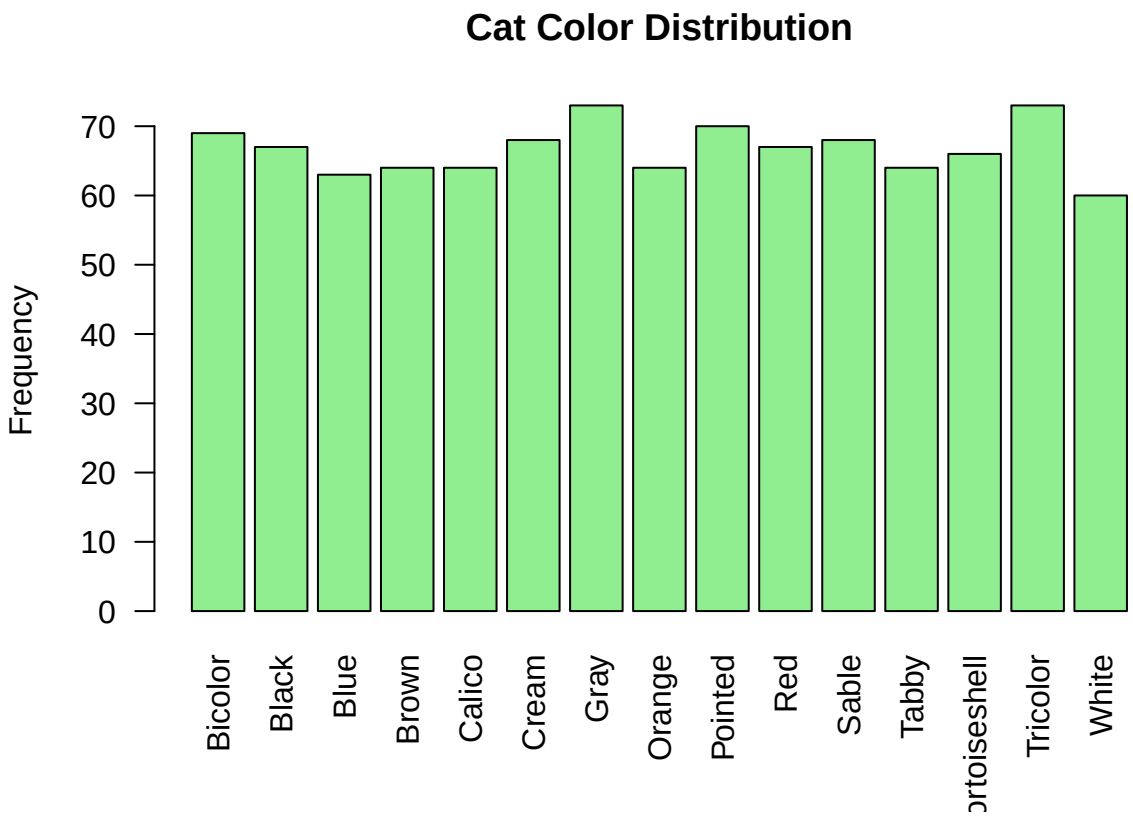
Tip 4: Visualize! When in Doubt, Plot it Out.

A great way to get a feel for your data is through visualization. Simple plots in base R can help you quickly understand the relationships between variables. Here's how you can make a basic scatter plot and add a linear model to it:

```
# Continuous plot
hist(cats_df$Age, main = "Histogram of Cat Age in Years",
     xlab = "Years Lived", col = "lightblue")
```




```
# Categorical plot
barplot(table(cats_df$Color), main = "Cat Color Distribution",
        xlab = "", ylab = "Frequency", col = "lightgreen", las = 2)
```



R automatically guesses what plots may fit your data best based on the structure of the objects (i.e., you can type the generic `plot()` function, and R will guess which visuals may fit your function best). Visualizing your data helps you see trends, outliers, or relationships between variables before diving into deeper analysis.

Tip 5: Explore! Find Descriptive Statistics for Your Data.

Before running complex models, it's essential to explore the descriptive statistics of your data. R makes it easy to get a quick summary:

```r
#install.packages("psych")
library(psych)

#For Continuous Descriptives
describe(cats_df)
```

```
##           vars    n  mean   sd median trimmed   mad min max range  skew
## Breed*       1 1000 15.31 8.67     15   15.28 11.86   1  30    29  0.00
## Age          2 1000 10.21 5.54     10   10.26  7.41   1  19    18 -0.03
## Weight_kg    3 1000  5.55 2.23      6    5.56  2.97   2   9     7 -0.02
## Color*       4 1000  8.00 4.30      8    8.01  5.93   1  15    14 -0.01
## Gender*      5 1000  1.50 0.50      2    1.51  0.00   1   2     1 -0.02
##           kurtosis   se
## Breed*       -1.22 0.27
## Age          -1.26 0.18
```

```
## Weight_kg    -1.19 0.07
## Color*       -1.20 0.14
## Gender*      -2.00 0.02
```

```r
#For Categorical Descriptives
table(cats_df$Color)
```

```
##
##       Bicolor         Black          Blue         Brown        Calico
##            69            67            63            64            64
##         Cream          Gray        Orange       Pointed           Red
##            68            73            64            70            67
##         Sable         Tabby Tortoiseshell      Tricolor         White
##            68            64            66            73            60
```

Using the `psych` package, you can see descriptive statistics with the `describe()` function (best suited for continuous variables) and frequency tables with the `table()` function (best suited for categorical variables).

Before you move onto more advanced analyses, make sure you understand the structure of your data through visuals and descriptives.

Conclusion

Thank you for reaching the end! Learning R is a hands-on experience, so the more you play around with these concepts and tools, the more proficient you'll become. Happy coding!