Why R?

A Brief Introduction to My Favourite Language

Harshvardhan

There are only two kinds of languages: the ones people complain about and the ones nobody uses.

— Bjarne Stroustrup

What is R?

Roger Peng called R Programming Language a dialect of S language.[1] Though I would say this dialect deserves a place among the "important" languages as its own, it begs the question: what is S? S is a programming language developed as an internal statistical analysis tool at AT&T Bell Laboratories. It was initiated in 1976 when computer programs were not commonplace for statistical analyses. The Bell Labs scientists would go to talks and show amazing graphics and results — sometimes even live demos.

Soon, statisticians everywhere wanted to use it for their analyses. Bell Labs was conservative in sharing the software with the broader public; it was an internal tool. Even if it was made available, there were several roadblocks. First, you needed a Unix computer to run S. You would need a modem connecting to the Bell Labs and using their computing powers. Second, there was no binary installation file like ".exe" or ".dmg" that you could double-click and go. You would get the source code from Bell Labs, and you had to compile it yourself.

Let's say you were brave enough to attempt compiling it yourself. But in those days, processors were not standardized.[2] Every processor has different requirements. In 1993, Bell Labs gave an exclusive license to StatSci (which later became Insightful Corp.) to develop and sell S. Insightful added some bells and whistles to it and made it S-Plus. S-Plus quickly gained popularity. It was available as a binary installation file for Unix and Microsoft Windows.

[1] Peng, Roger D. *R programming for data science.* Victoria, BC, Canada: Leanpub, 2016.

[2] Even today, when Apple launched M1, it took a lot of time for the developers to create native programs for Apple's M1 chip. M1 and M2 Macs have Rosetta 2, which could convert Intel-based software to a format required by M1.

In the first half of the 1990s, Ross Ihaka and Robert Gentleman, two professors in the Department of Statistics at the University of Auckland, wondered if a Macintosh version was available. The department had recently gotten many of Apple's latest computers and needed software for their statistics class. Since there was no S or S-Plus version for Mac, they decided to write a version of the S language. Their first attempt is noted in the classic paper published in the Journal of Computational and Graphical Statistics.

> Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5(3):299–314, 1996

Martin Mächler convinced them to release R with GNU General Public License. This was critical for two reasons. First, it made R freely available. Both S-Plus and S required a paid license. Second, it allowed virtually anyone to tinker with its source code and suggest improvements. Soon after, a mailing list was created where people shared new enhancements. In 1997, R Core Group was formed, which, even today, controls the source code of R. In 2000, R 1.0.0 was released to the public.

Rest is history.

What can R do?

For this question, I asked Dr Swati Sonal, a postdoctoral research fellow at the Massachusetts General Hospital and Harvard Medical School. She uses R for her quantitative analyses, primarily focussed on clinical and translational research in Colorectal Cancer. Here is what she had to say.

> I think the primary reason I love R is that it is so cool! It makes me feel like I'm analyzing without getting bogged down by the minute programmatic details like one does in Python. I can do the "default" analysis, but it is simple to customize as per my requirements, which is difficult to achieve in alternatives like SPSS and STATA. The output is neat. I can get publication-ready tables and visualizations quickly. There is nothing like the **dfSummary()** function, which can do much so quickly.[3]
>
> Even making custom changes in them is intuitive. Functions are versatile as well. No matter which model I use, I can always rely on **predict()** for making predictions. R Markdown is handy for reproducible research. Like a lab note, I can write my thoughts, do the analysis and produce the results — all using a single document. Doesn't it sound like magic?

[3]**dfSummary()** is a function from the summaryTools package that can summarise data with a single command. Learn more: https://cran.r-project.org/web/packages/summarytools/vignettes/introduction.html

Packages and CRAN

It is clear R can achieve a lot of things for free. Much of R's functionalities are extended with packages, which are pieces of code freely available for everyone to use. Comprehensive R Archive Network (CRAN) hosts the binary file for R software as well as an archive of packages. The R software comes with specific default capabilities, identified as "base R". Beyond that, it can be extended by packages which one can install from CRAN, GitHub, Bioconductor, among others. rdrr.io is an excellent search engine for anything related to R, including R packages. On November 15, 2022, it had information about 23,097 CRAN packages, 2,130 Bioconductor packages, 2,207 R-Forge packages and 85,750 GitHub packages.

RStudio

RStudio is the most popular Integrated Development Environment (IDE) for using R. The IDE provides a functional and beautiful interface to R with a set of tools to help you be more productive with R. It has a built-in console, Terminal, GitHub client, plot viewer, debugging tool, and a host of other valuable appliances. I'd recommend you install R and RStudio to get started with your R journey. Posit provides a simple guide to do it all.

Many custom functions can be added using add-ins. Dean Attali provides a list of valuable add-ins in RStudio at https://github.com/daattali/addinslist. Some of my favourite add-ins are esquisse, which offers a graphical interface for creating visualizations with ggplot2; datapasta, with which you can copy any data and paste as a data frame; and styler, which can help you stylize your code using the Tidyverse Style Guide.

Tidyverse

Tidyverse is a collection of R packages designed for data science. They are maintained by Posit PBC (formerly known as RStudio PBC). They include `ggplot2` for data visualization, `dplyr` for data wrangling and manipulation, `tidyr` and `tibble` for getting tidy data[4], `readr` for reading commonly used data formats like `csv`, `tsv`, etc., `purrr` which enhances functional programming capabilities in R, `stringr` for handling strings and `forecast` for handling categorical data, known as "factor" in R.

The best way to start learning about using R for Data Science would be the free book: R for Data Science (2e) by Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund.

[4]Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59* (10), 1–23. https://doi.org/10.18637/jss.v059.i10

R Markdown

R Markdown is a document writing tool designed for data science and promoting reproducibility. The document can be edited by any plain text editor, though RStudio would be the best choice. You can write R, Python, and 60 other languages in a single document.[5] It weaves together the code and output in a beautifully formatted document. The resulting document could be an HTML webpage, a PDF, a Microsoft Word file, a LaTeX file, a book, a handout, or one of the many possible formats. The document can be interactive as well. You can see the Gallery of outputs with example codes at Gallery, R Markdown by RStudio.

Garrett Grolemund's talk on YouTube titled "R Markdown: The bigger picture" gives a great introduction on why using analysis with R Markdown helps the present you, supports the future you and helps the science as well. Rob Hyndman's talk on YouTube titled "How R Markdown changed my life" is an excellent introduction to what is possible with R Markdown.

Slowly, people are transitioning to Quarto. Quarto is a specialized publishing document software that can work with many languages: R, Python, Julia and more. Even though you write in one format, the output could be any of the following formats: articles, reports, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more. You can learn more about it here.

Data Wrangling and Plotting

This article would be incomplete if I didn't introduce some basic functionalities in R. The first step to good data analysis is understanding and visualizing the data. In the following sections, I will showcase what is possible with R in just a few lines of code. Of course, if you're willing to invest time, you will have excellent results!

Data Wrangling

Hadley Wickham (2014) formalized the idea of organizing datasets: variables are columns, observations are rows, and values are cells.[6] Tidyverse is built on this principle. With this understanding, let's learn how to wrangle data into practical formats.

For the examples below, I would use "Cause wise Age and Gender - wise Distribution of Suicides during 2020" data from Government of India.[7] A snapshot is also available on my

[5]You can find the complete list in the R Markdown Book or by typing `names(knitr::knit_engines$get())` in your console.

[6]Wickham, H. . (2014). Tidy Data. *Journal of Statistical Software*, 59 (10), 1–23. https://doi.org/10.18637/jss.v059.i10

[7]I've slightly modified the data to simplify it.

GitHub. All my wrangling begins by loading **tidyverse** into my R session. Following this, I will load the dataset.

Once the data frame is loaded, you can see it by typing its name in your console.

```
df
```

```
# A tibble: 20 x 26
   `Si. No` Cause             `Below 18 year~` `Below 18 year~` `Below 18 year~`
      <dbl> <chr>                        <dbl>            <dbl>            <dbl>
 1        1 Bankruptcy or In~               13                9                0
 2        2 Marriage Related~               60               98                0
 3        3 Failure in Exami~              560              569                0
 4        4 Impotency/Infert~                3                4                0
 5        5 Family Problems               2019             1987                0
 6        6 Illness (Total)                563              764                0
 7        7 Death of Dear Pe~               30               31                0
 8        8 Drug Abuse/Alcoh~               67                2                0
 9        9 Fall in Social R~               14               18                0
10       10 Ideological Caus~               12               14                0
11       11 Love Affairs                   573              764                0
12       12 Poverty                         30               40                0
13       13 Unemployment                    21               13                0
14       14 Property Dispute                 9                3                0
15       15 Suspected/ Illic~               32               37                0
16       16 Illegitimate Pre~                0               12                0
17       17 Physical Abuse (~                0               18                0
18       18 Professional/Car~               35               22                0
19       19 Causes Not Known               638              783                0
20       20 Other Causes                   713              816                0
# ... with 21 more variables: `Below 18 years - Total` <dbl>,
#   `18 and Above-Below 30 years - Male` <dbl>,
#   `18 and Above-Below 30 years - Female` <dbl>,
#   `18 and Above-Below 30 years - Transgender` <dbl>,
#   `18 and Above-Below 30 years - Total` <dbl>,
#   `30 and Above-Below 45 years - Male` <dbl>,
#   `30 and Above-Below 45 years - Female` <dbl>, ...
```

Piping Functions

Typically, the names are not "standardized" — there are spaces and ".". I like them in snake case, where all of them are small letters with _ for spaces. **janitor** has a function

clean_names() which can do this for me. I can **pipe** this function with the data load.

Piping takes in what is piped, and applies that function to it. It is a more elegant way to write functions. Many call it "functional programming".

```
df = read_csv("deaths.csv") %>%
   janitor::clean_names()
```

```
Rows: 20 Columns: 26
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (1): Cause
dbl (25): Si. No, Below 18 years - Male, Below 18 years - Female, Below 18 y...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df
```

```
# A tibble: 20 x 26
   si_no cause                 below_18_years_~ below_18_years_~ below_18_years_~
   <dbl> <chr>                            <dbl>            <dbl>            <dbl>
 1     1 Bankruptcy or Indeb~                13                9                0
 2     2 Marriage Related Is~                60               98                0
 3     3 Failure in Examinat~               560              569                0
 4     4 Impotency/Infertili~                 3                4                0
 5     5 Family Problems                   2019             1987                0
 6     6 Illness (Total)                    563              764                0
 7     7 Death of Dear Person                30               31                0
 8     8 Drug Abuse/Alcoholi~                67                2                0
 9     9 Fall in Social Repu~                14               18                0
10    10 Ideological Causes/~                12               14                0
11    11 Love Affairs                       573              764                0
12    12 Poverty                             30               40                0
13    13 Unemployment                        21               13                0
14    14 Property Dispute                     9                3                0
15    15 Suspected/ Illicit ~                32               37                0
16    16 Illegitimate Pregna~                 0               12                0
17    17 Physical Abuse (Rap~                 0               18                0
18    18 Professional/Career~                35               22                0
19    19 Causes Not Known                   638              783                0
```

```
20    20 Other Causes                      713              816                0
# ... with 21 more variables: below_18_years_total <dbl>,
#   x18_and_above_below_30_years_male <dbl>,
#   x18_and_above_below_30_years_female <dbl>,
#   x18_and_above_below_30_years_transgender <dbl>,
#   x18_and_above_below_30_years_total <dbl>,
#   x30_and_above_below_45_years_male <dbl>,
#   x30_and_above_below_45_years_female <dbl>, ...
```

Selecting Columns

You can select columns using the **select** function. You can pass in the column name directly (like "total_male") or use support functions like **contains**, **starts_with**, and other **tidyselect** functions.

```
df %>%
   select(cause, total_total)
```

```
df %>%
   select(cause, contains("below_18_years"))
```

Filter Values

You can filter the values using **filter** function. Let's look at causes that result in more than 5000 suicides. I'm also sorting them with **arrange** and passing the variable into **desc**.

```
df %>%
   filter(total_total > 5000) %>%
   select(cause, total_female, total_male, total_transgender, total_total) %>%
   arrange(desc(total_total))
```

```
# A tibble: 8 x 5
  cause                         total_female total_male total_transgend~ total_total
  <chr>                                <dbl>      <dbl>            <dbl>       <dbl>
1 Family Problems                      16140      35333                4       51477
2 Illness (Total)                       8750      18866                7       27623
3 Causes Not Known                      4660      11273                0       15933
4 Other Causes                          4249      10795                1       15045
5 Drug Abuse/Alcoholic Add~              193       8974                2        9169
6 Marriage Related Issues ~             4152       3484                0        7636
```

```
7 Love Affairs                          2655       4101                1         6757
8 Bankruptcy or Indebtedne~              468       4744                1         5213
```

Apparently, "Family Problems" are the biggest reason to commit suicide.

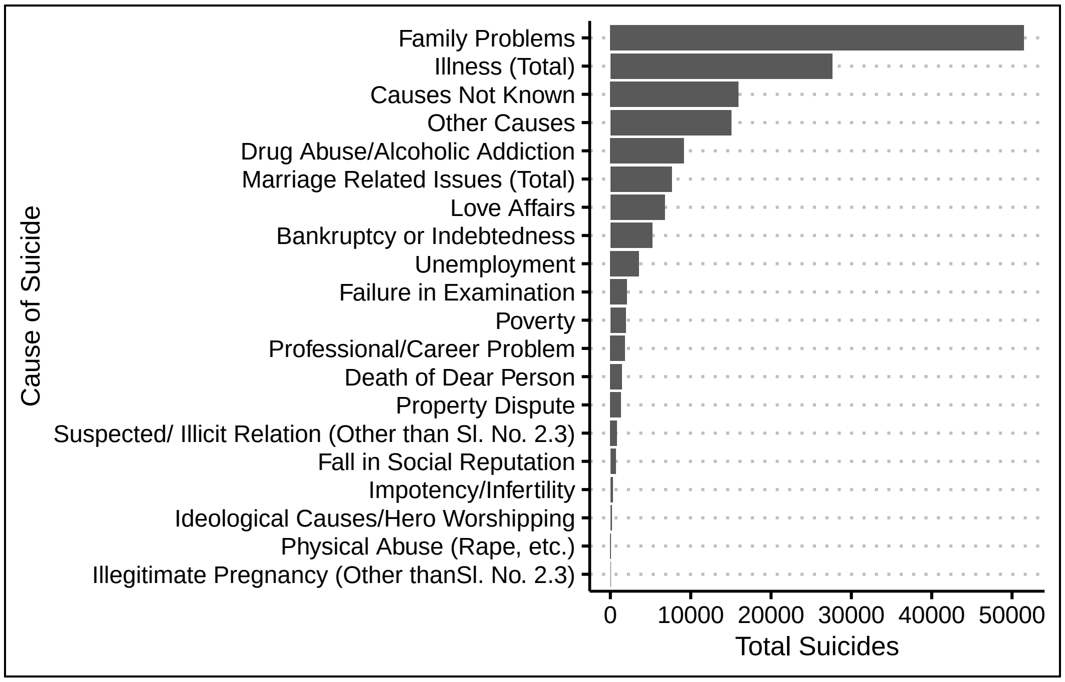
Plotting

You can also visualize these findings using the **ggplot2** package in R. It is also part of **tidyverse**. The general syntax of the **ggplot2** function call is something like this.

```
ggplot(data = df, aes(x = x_var, y = y_var) +
geom_object()
```

Let's see it in action. You can flip the coordinates with **coord_flip()** and use custom themes. I'm using **ggtheme**'s **theme_clean()** theme. You can install the package with `install.package("ggthemes")` and set the theme using `theme_set(ggthemes::theme_clean())` . Changing the labels is easy too.

```
df %>%
   ggplot(aes(x = reorder(cause, total_total), y = total_total)) +
   geom_col() +
   labs(x = "Cause of Suicide", y = "Total Suicides") +
   coord_flip() +
   ggthemes::theme_clean()
```



You might want to visualize this separating by gender. For that, you will need to "tidy" the data. One function that comes in handy for this is **pivot_longer()**. Then, I can use the gender (male, female, transgender) to decide the colour fill. Like always, learn more about the function by typing **?pivot_longer()** in your console.

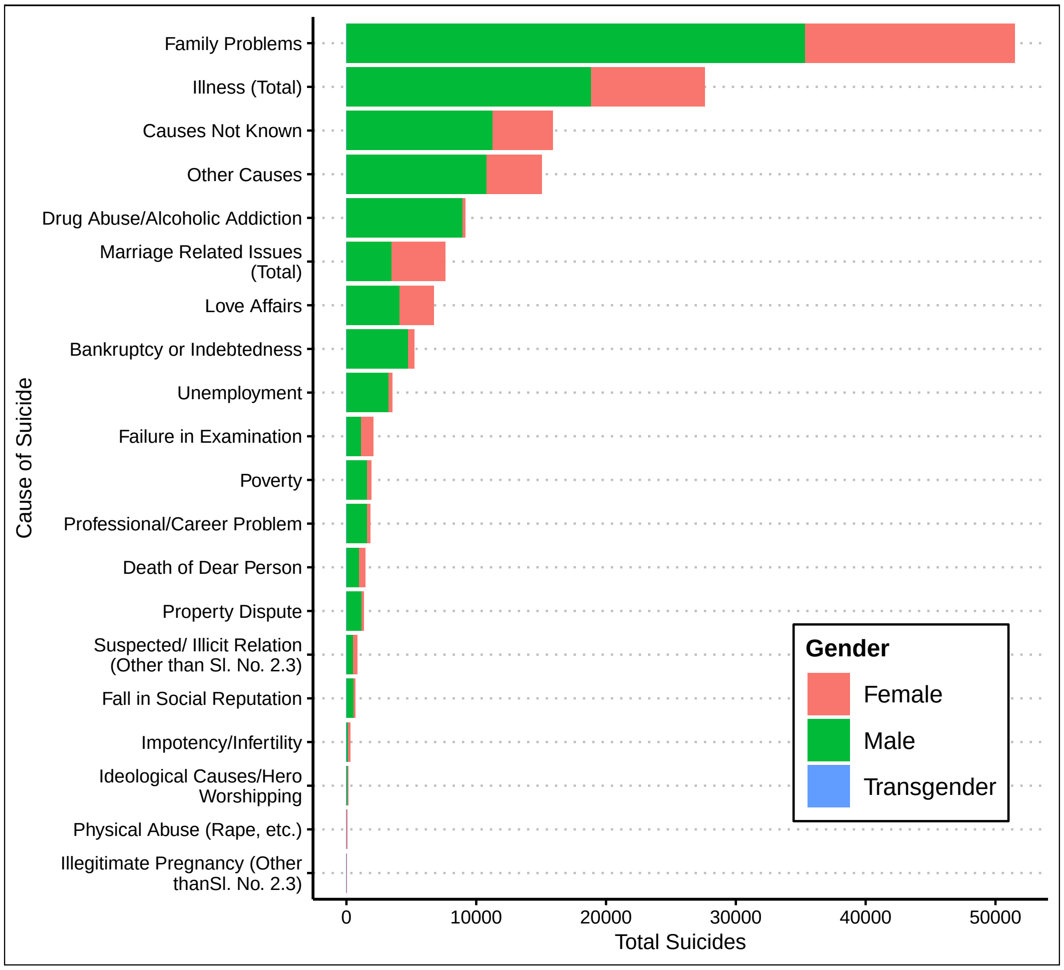
```
df %>%
  select(cause, total_male, total_female, total_transgender) %>%
  pivot_longer(!cause, names_to = "gender", values_to = "suicides")
```

```
# A tibble: 60 x 3
   cause                           gender            suicides
   <chr>                           <chr>                <dbl>
 1 Bankruptcy or Indebtedness      total_male            4744
 2 Bankruptcy or Indebtedness      total_female           468
 3 Bankruptcy or Indebtedness      total_transgender        1
 4 Marriage Related Issues (Total) total_male            3484
 5 Marriage Related Issues (Total) total_female          4152
 6 Marriage Related Issues (Total) total_transgender        0
 7 Failure in Examination          total_male            1147
 8 Failure in Examination          total_female           933
 9 Failure in Examination          total_transgender        0
10 Impotency/Infertility           total_male             125
```

```
# ... with 50 more rows
```

Now, let's plot this. You can learn more about this syntax at the R Graph Gallery. They have a fantastic collection of visualizations possible with R and Python.

```r
df %>%
   select(cause, total_male, total_female, total_transgender) %>%
   pivot_longer(!cause, names_to = "gender", values_to = "suicides") %>%
   ggplot(aes(x = reorder(cause, suicides), y = suicides, fill = gender)) +
   geom_col() +
   labs(x = "Cause of Suicide", y = "Total Suicides", fill = "Gender") +
   scale_fill_discrete(labels = c("Female", "Male", "Transgender")) +
   scale_x_discrete(labels = \(x) str_wrap(x, width = 30)) +
   coord_flip() +
   ggthemes::theme_clean(base_size = 9) +
   theme(legend.position = c(0.8, 0.2))
```



The suicide rate is a lot higher for males than for females. The trend is discernible in almost all categories. We can do all of this with just a few lines of code. Intriguing, right?

Additional Resources

If you are looking for more resources around R, you should look at this section.

Big Book of R

Big Book of R is a collection of mostly free and paid books about data analysis and R. You can find books on virtually any topic.

- aRtistry: Making Art with R

- Data Science
- Data Visualization
- Machine Learning
- Finance
- Geospatial Data
- Sports Analytics
- Time-series Analysis
- and more.

R for Data Science, 2nd Edition

It is the most popular book of R in the present world, with the web version freely available. It talks about data wrangling, data visualization, tidy principles, among other things. Hadley Wickham, Chief Data Scientist at RStudio, is the leading author of the book. If you want to dive into R deeply, this is where you should start. It is available at https://r4ds.hadley.nz.

Next — Today I Learnt About R

Many of you have probably read and worked with R in the past. This weekly newsletter, written by yours truly, presents byte-sized information about R and data science. The format is pretty simple. **There are five stories about the data world, four R packages, three statistics and data science jargon, then two tweets about R and one meme.** It is available for free. I promise, no spam.

https://www.getrevue.co/profile/harshbutjust

I might be biased, but this is probably the best way to remain updated on new developments and come across interesting data science stuff. Here are some past editions that I like.

- Is R-squared Useless? | Next — Issue #20
- Street maps, free APIs and Topic Modelling | Next - Issue #33
- Creating music with R | Next - Issue #39
- Moore's Law for Everything | Next - Issue #41
- Books on Data Science | Next - Issue #42

Concluding Remarks

In this longish post, I rambled about my favourite programming language. I talked about its history — how AT&T Labs developed a language that became super popular because it was free to modify. Then, I introduced the present-day mechanics of R World — RStudio, Tidyverse and R Markdown (Quarto). Then, I gave a short example of data wrangling and visualization using data on suicides in India. Family reasons are the cause of most suicides; males commit more suicides than women.

Finally, I added additional resources you should look for to learn more about R. Big Book of R is a good collection of books on virtually every topic in data science. Hadley Wickham's R for Data Science is an excellent starting point. My newsletter can bring fresh content to your inbox for free every Wednesday.

Happy R!

About the Author

Harshvardhan is a second-year PhD student of Business Analytics and Statistics at the Haslam College of Business, University of Tennessee. His research interests are in the application of machine learning and applied statistics in the supply chain domain. He tinkers with R, reads some blogs, and makes coffee in his free time. He invites you to his digital garden at https://www.harsh17.in.