Python vs R for Beginners: A Comprehensive Guide to Choose the Right Tool for Your Data Science Journey

By: Harish Muhammad

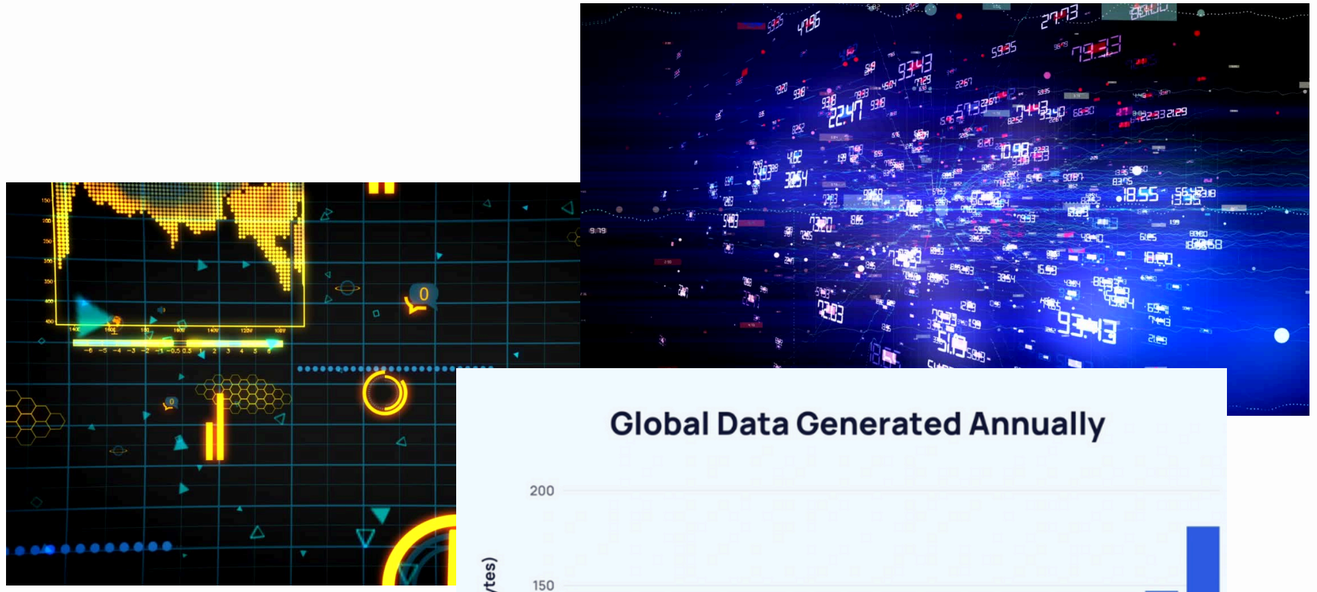
Contents

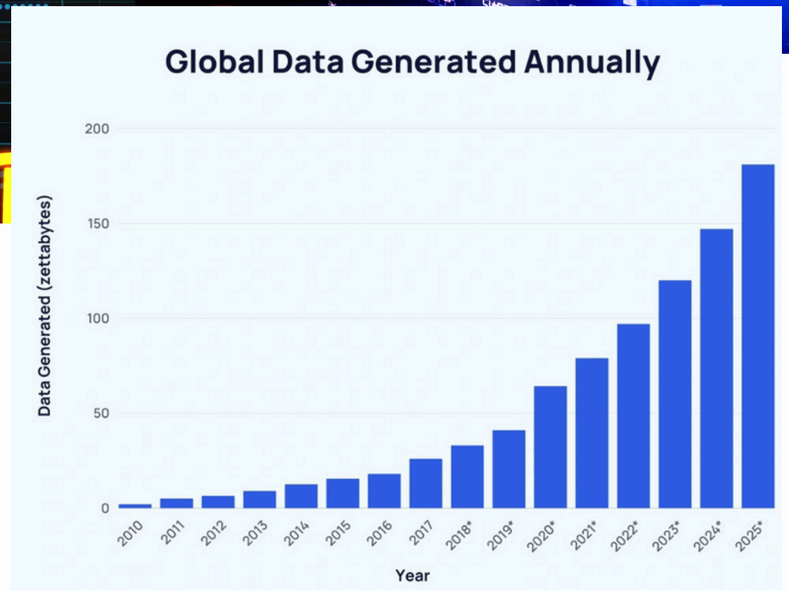
- Intro - data science journey
- Which programming skill (Python vs R)
- History of Python and R
- Popularity of Python and R
- User friendliness
- Community support
- Library availability
- Syntax comparison
- Syntax and performance demonstration
- Syntax comparison
- Computing process
- Data manipulation
- Data wrangling
- Data cleaning
- Data visualization
- Conclusion

INTRO TO DATA SCIENCE JOURNEY

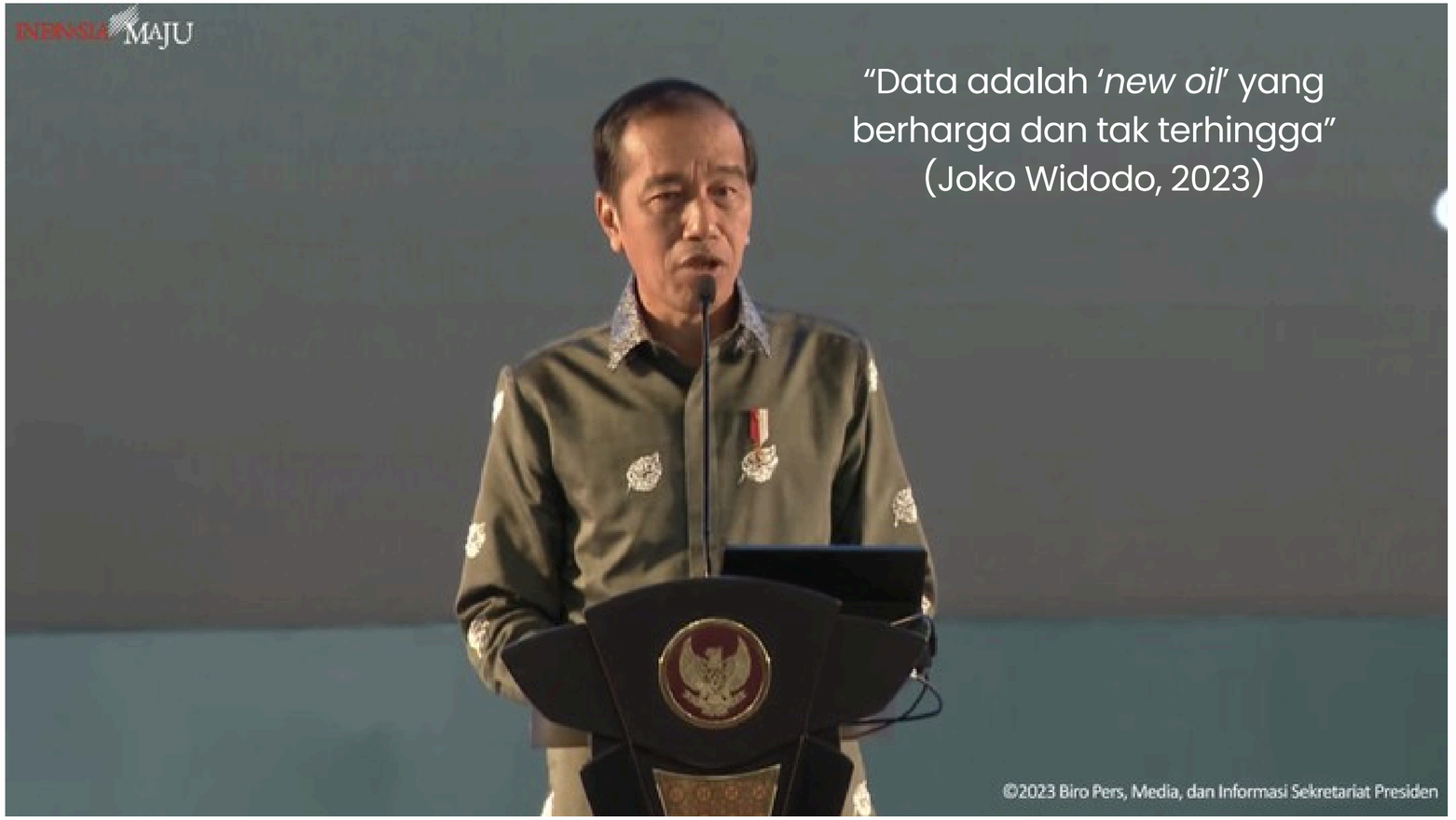
Digital Era

Each day 0,33 ZB or 328, 7 M TB data generated





DATA is a NEW OIL !!!



Winning the competition

“Data-driven organizations are 23 times more likely to acquire customers, six times as likely to retain customers, and 19 times as likely to be profitable as a result.”

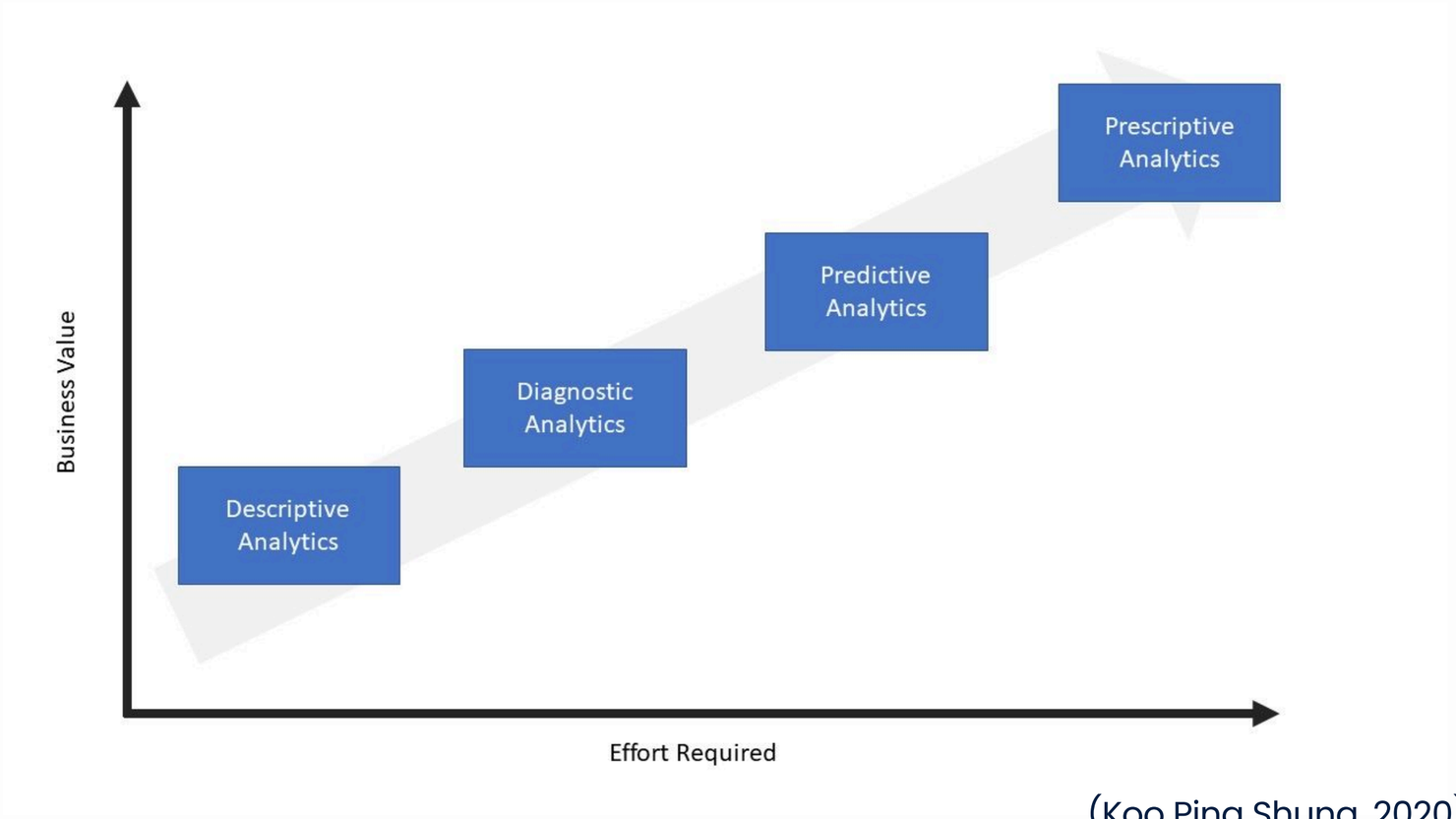
McKinsey & Company

How data analysis support an organization?

(SHRM Survey, 2016)

Organizations need someone with skills to translate their data into insights

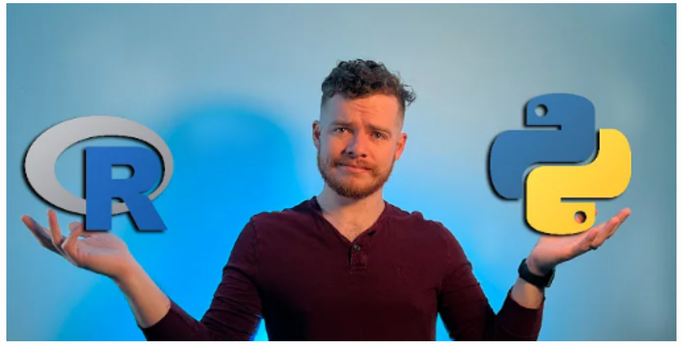
Data science journey challenges



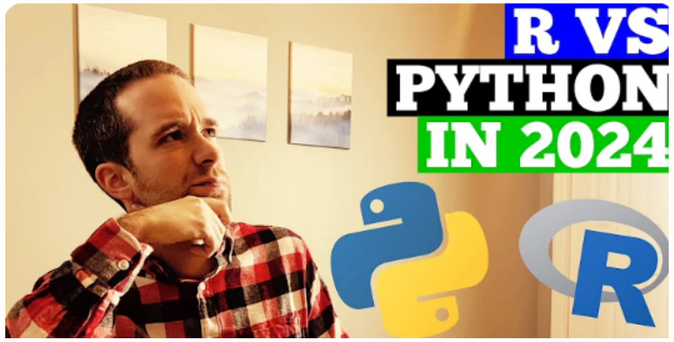
(Koo Ping Shung, 2020)

Which programming skill?

Python VS R


R


python
Vs
R


R VS
PYTHON
IN 2024
R


Python
vs R
IBM Cloud


R vs Python
Which is better?
Alex Freberg
Data Analyst
1mo
Python is better than R.
1,758
245 comments
153,690


R VS Python
Choose your skills wisely

Inventors

Python inventor

Guido van Rossum
Dutch Programmer

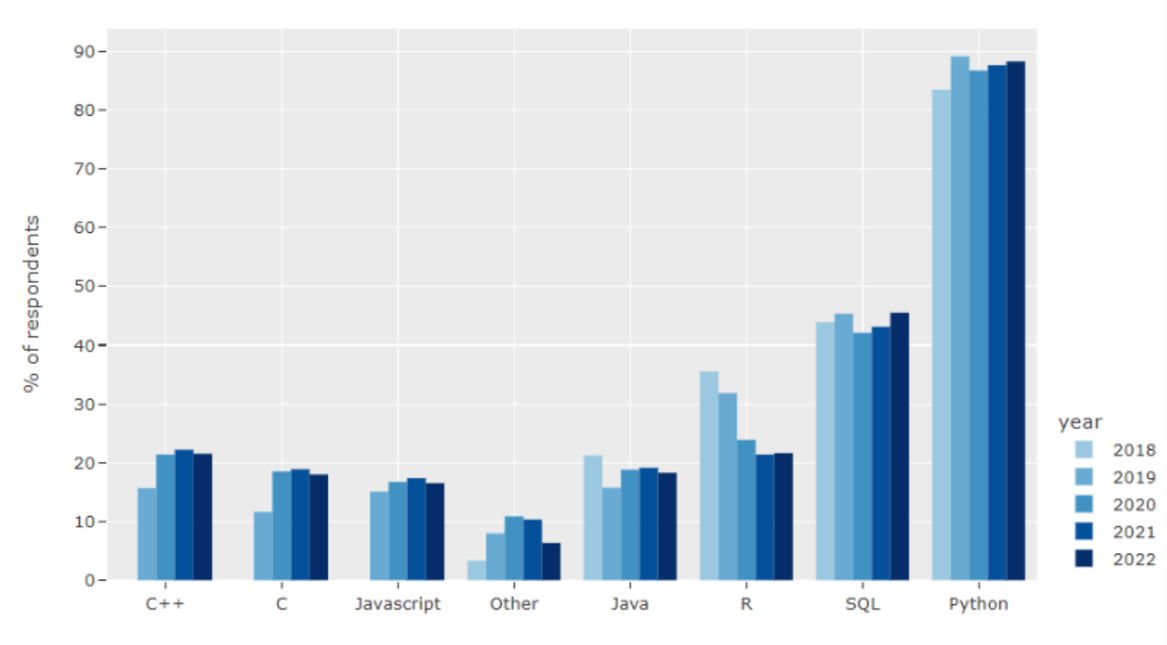
R inventors

Prof. Ross Ihaka
New Zealand Statician

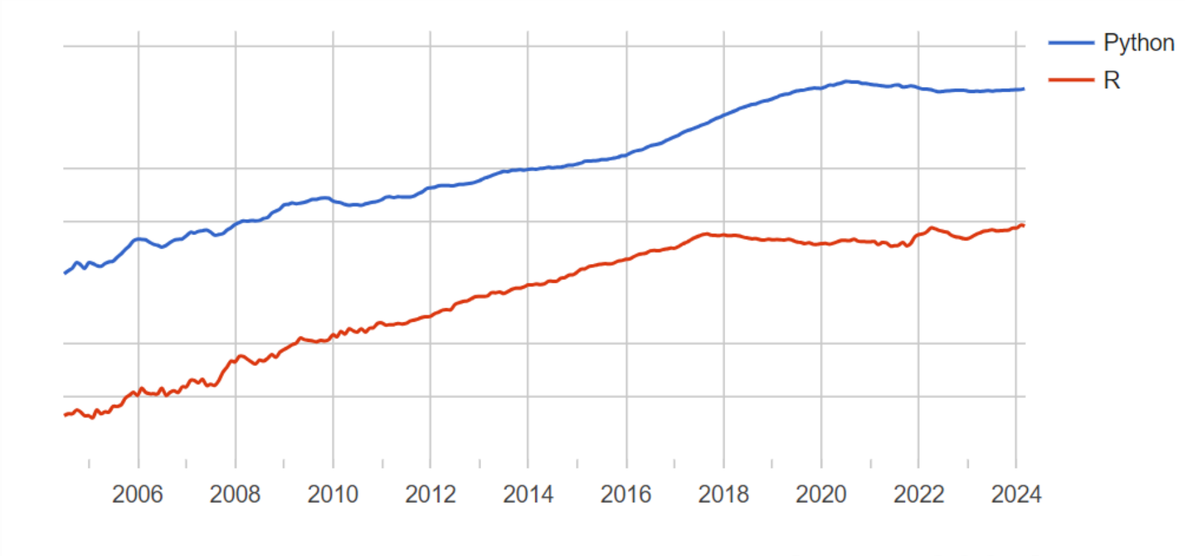
Robert Gentleman
Canadian Statician &
Bioinformatician

Popularity

Popularity survey on data science community



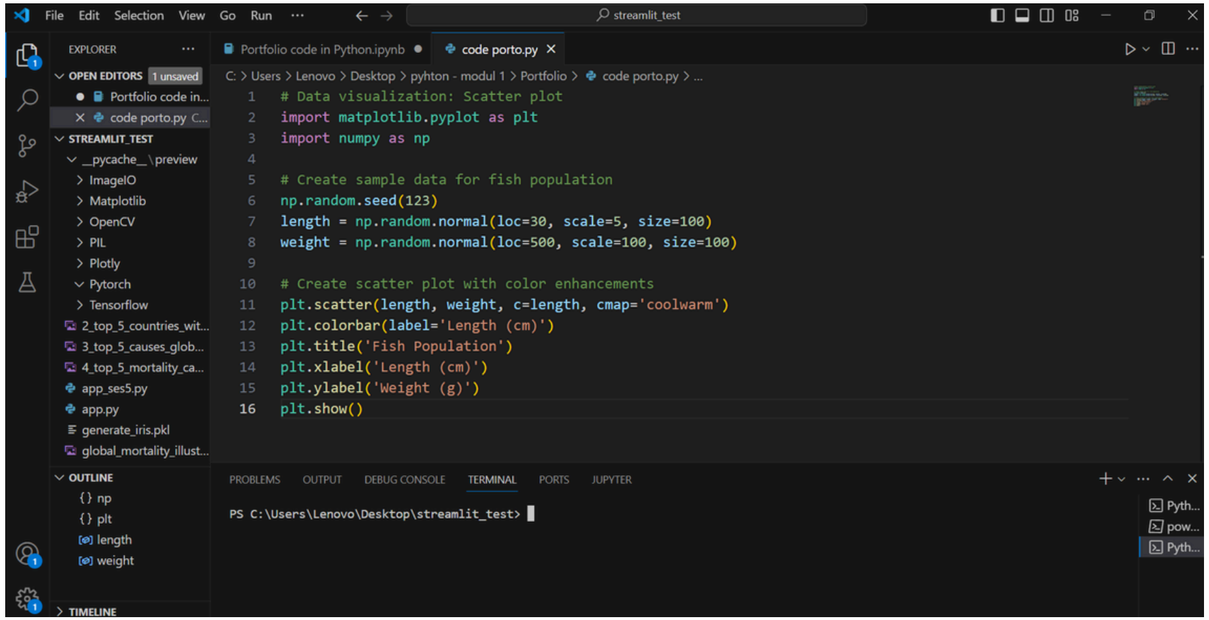
(Kaggle, 2022)



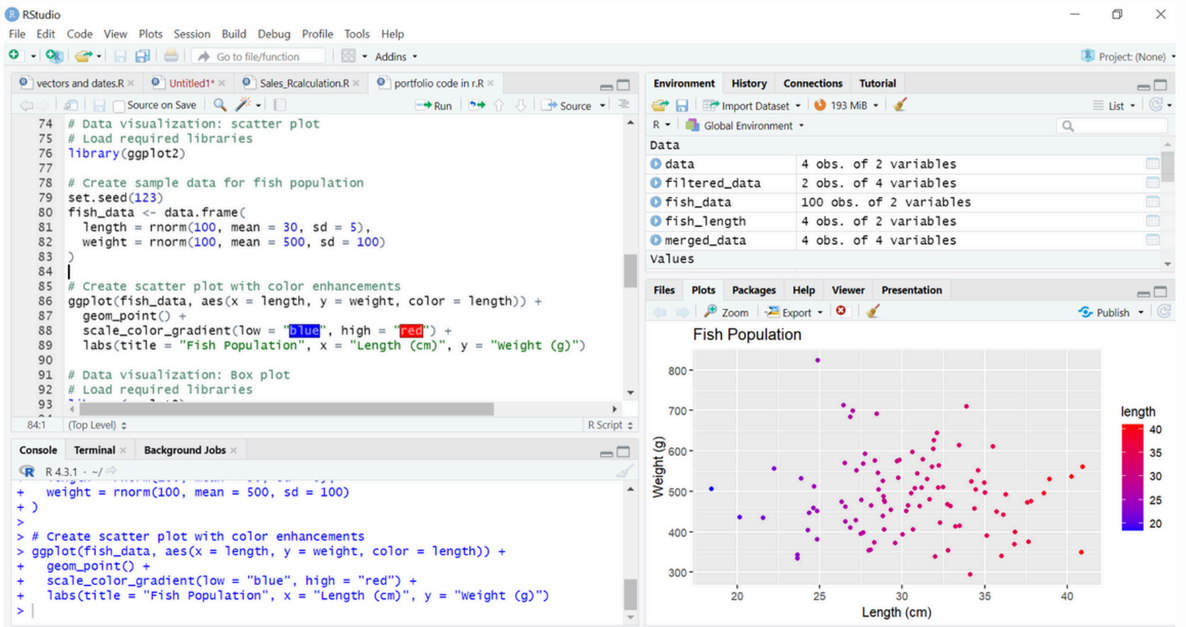
(PYPL 2024)

User-Friendliness

Python (Visual Studio Code)



R (R Studio)



Community support

VS

CRAN R
The Comprehensive R Archive Network

NHS-R COMMUNITY

Library Availability

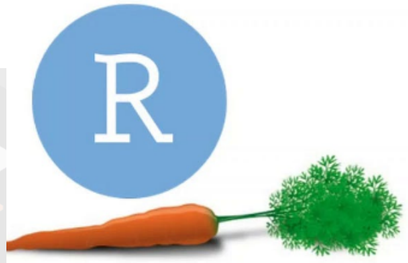
pandas

NumPy

seaborn

matplotlib

VS



SYNTAX & PERFORMANCE DEMONSTRATION

Syntax comparison

SYNTAX	Python	R
Importing Libraries	import numpy as np import pandas as pd	library(ggplot2) library (dplyr)
Import files	pd.read_csv('file.csv')	read_csv("file.csv")
Assignment	=	<- or ->
Data types	Numeric, Set, Dictionary, Boolean, Sequence Type	Numeric, Logical, Integer, Complex, Character , Raw
Data Structures	Dictionary, Set, Tuple, List	Vector, List, Dataframe, Matrice, Factor
Indexing	starts from [0]	starts from [1]
Arithmetic Operators	+, -, *, /, **	+, -, *, /, ^
Relational Operators	>, <, ==, <=, >=, !=	>, <, ==, <=, >=, !=
Logical Operators	AND, OR, NOT	&, |, &&, ||, !

(Almabetter, 2023)

Computing Process

```python
import time
import memory_profiler

# Example loop in Python
start_time = time.time()  # Start measuring time
mem_usage = memory_profiler.memory_usage()

for i in range(10000):
    print(i)

end_time = time.time()  # Stop measuring time
mem_usage.append(memory_profiler.memory_usage()[0])

execution_time = end_time - start_time
print("Execution time for Python loop:", execution_time, "seconds")
print("Memory usage:", max(mem_usage) - min(mem_usage), "MB")
```

```r
# Example loop in R
start_time <- Sys.time()  # Start measuring time
mem_usage <- numeric()

Rprof(filename = "Rprof.out", memory.profiling = TRUE)
for (i in 1:10000) {
  print(i)
  mem_usage <- c(mem_usage, object.size(i)) # Measure memory usage for each
iteration
}
Rprof(NULL)

end_time <- Sys.time()  # Stop measuring time
execution_time <- end_time - start_time
print(paste("Execution time for R loop:", execution_time, "seconds"))

# Calculate memory usage during the loop in megabytes
memory_used <- sum(mem_usage) / (1024 * 1024)
print(paste("Memory usage during R loop:", memory_used, "MB"))
```

Computing Process

Results

Python posses higher speed

```
...
9998
9999
Execution time for Python loop: 0.21841764450073242 seconds
Memory usage: 0.96875 MB
```

R consumed less memory

```
>
> end_time <- Sys.time()  # Stop measuring time
> execution_time <- end_time - start_time
> print(paste("Execution time for R loop:", execution_time, "seconds"))
[1] "Execution time for R loop: 3.09364700317383 seconds"
>
> # Calculate memory usage during the loop in megabytes
> memory_used <- sum(mem_usage) / (1024 * 1024)
> print(paste("Memory usage during R loop:", memory_used, "MB"))
[1] "Memory usage during R loop: 0.5340576171875 MB"
```

Data Manipulation

Python

```python
# Data Manipulation: Data Frame

# Importing required library
import pandas as pd

# Create sample fish population data
fish_data = pd.DataFrame({
    'species': ['Trout', 'Salmon', 'Perch', 'Bass'],
    'population': [1000, 2000, 1500, 3000]
})

# Add a new column for average weight
fish_data['avg_weight']  =  fish_data['species'].map({'Trout':  1.2,
'Salmon': 1.5, 'Perch': 0.8, 'Bass': 2})

# Print the modified data
print(fish_data)
```

R

```r
# Data Manipulation - Data frame
# Load required libraries
library(dplyr)

# Create sample fish population data
fish_data <- data.frame(
  species = c("Trout", "Salmon", "Perch", "Bass"),
  population = c(1000, 2000, 1500, 3000)
)

# Add a new column for average weight
fish_data <- mutate(fish_data, avg_weight =
                                          ifelse(species == "Trout",
1.2,
                                         ifelse(species == "Salmon",
1.5,
                                          ifelse(species == "Perch",
0.8,
                                           ifelse(species == "Bass",
2, NA
)))))
```

Data Manipulation

Output in Python

```
species  population  avg_weight
0   Trout        1000         1.2
1  Salmon        2000         1.5
2   Perch        1500         0.8
3    Bass        3000         2.0
```

Output in R

```
species population avg_weight
1   Trout        1000         1.2
2  Salmon        2000         1.5
3   Perch        1500         0.8
4    Bass        3000         2.0
```

Data Wrangling

Python

```python
# Data wrangling
# Assuming we have additional data for fish length
fish_length = pd.DataFrame({
    'species': ['Trout', 'Salmon', 'Perch', 'Bass'],
    'avg_length': [30, 35, 25, 40]
})

# Merge fish_data with fish_length
merged_data = pd.merge(fish_data, fish_length, on='species')

# Print the merged data
print(merged_data)
```

	species	population	avg_weight	avg_length
0	Trout	1000	1.2	30
1	Salmon	2000	1.5	35
2	Perch	1500	0.8	25
3	Bass	3000	2.0	40

R

```r
# Data wrangling
# Load required libraries
library(dplyr)

# Assuming we have additional data for fish length
fish_length <- data.frame(
  species = c("Trout", "Salmon", "Perch", "Bass"),
  avg_length = c(30, 35, 25, 40)
)

# Merge fish_data with fish_length
merged_data <- inner_join(fish_data, fish_length, by = "species")

# Print the merged data
print(merged_data)
```

	species	population	avg_weight	avg_length
1	Trout	1000	1.2	30
2	Salmon	2000	1.5	35
3	Perch	1500	0.8	25
4	Bass	3000	2.0	40

Data cleaning

Python

```python
# Data Cleaning - Missing values
# Check for missing values in fish_data
missing_values = fish_data.isna().sum()

# Print the number of missing values
print("Number of missing values:", missing_values)
```

```
Number of missing values: species       0
population    0
avg_weight    0
dtype: int64
```

R

```r
# Data cleaning
# Check for missing values in fish_data
missing_values <- sum(is.na(fish_data))

# Print the number of missing values
print(paste("Number of missing values:", missing_values))
```

```
[1] "Number of missing values: 0"
```

Data cleaning

Python

```python
# Check for duplicated rows in fish_data
duplicated_rows = fish_data.duplicated().sum()

# Print the number of duplicated rows
print("Number of duplicated rows:", duplicated_rows)
```

```
Number of duplicated rows: 0
```

R

```r
# Duplicated data
# Check for duplicated rows in fish_data
duplicated_rows <- sum(duplicated(fish_data))

# Print the number of duplicated rows
print(paste("Number of duplicated rows:", duplicated_rows))
```

```
[1] "Number of duplicated rows: 0"
```

DATA VISUALIZATION COMPARISON

Data visualization

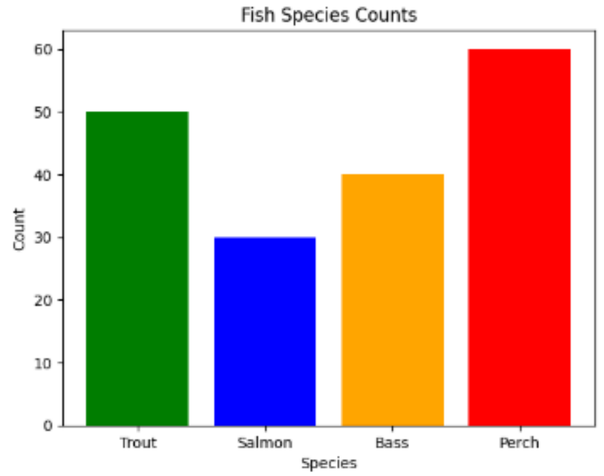
Python

Data visualization code for creating a bar chart in Python

```python
# Data visualization: Bar chart
# Create sample data for fish species counts
fish_species = ['Trout', 'Salmon', 'Bass', 'Perch']
fish_counts = [50, 30, 40, 60]

# Create bar chart with color enhancements
plt.bar(fish_species, fish_counts, color=['green', 'blue', 'orange',
'red'])
plt.title('Fish Species Counts')
plt.xlabel('Species')
plt.ylabel('Count')
plt.show()
```

Output of bar chart in Python



R

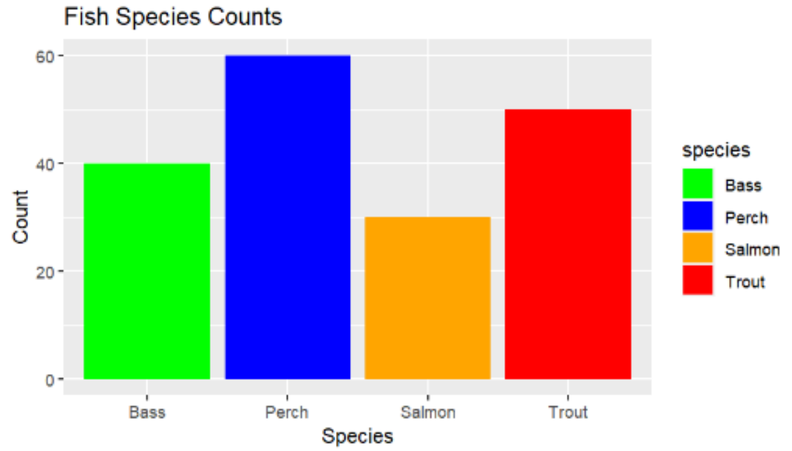
Data visualization code for creating a bar chart in R

```r
# Data visualization: Bar chart
# Load required libraries
library(ggplot2)

# Create sample data for fish species counts
fish_species <- c("Trout", "Salmon", "Bass", "Perch")
fish_counts <- c(50, 30, 40, 60)
fish_data <- data.frame(species = fish_species, count = fish_counts)

# Create bar chart with color enhancements
ggplot(fish_data, aes(x = species, y = count, fill = species)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("green", "blue", "orange", "red")) +
  labs(title = "Fish Species Counts", x = "Species", y = "Count")
```

Output of bar chart in R



Data visualization

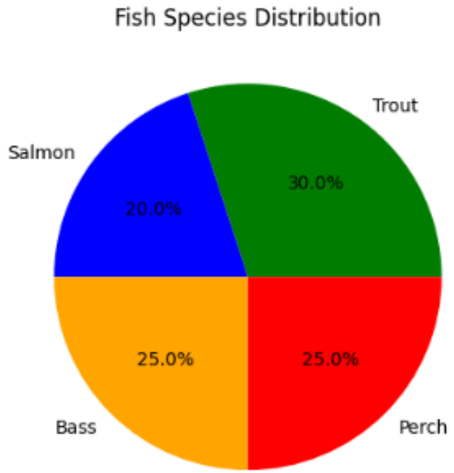
Python

Data visualization code for creating a pie chart in Python

```
# Data visualization
# Create sample data for fish species distribution
fish_species = ['Trout', 'Salmon', 'Bass', 'Perch']
fish_distribution = [0.3, 0.2, 0.25, 0.25]

# Create pie chart with color enhancements
plt.pie(fish_distribution, labels=fish_species, colors=['green',
'blue', 'orange', 'red'], autopct='%1.1f%%')
plt.title('Fish Species Distribution')
plt.show()
```

Output of pie chart in Python



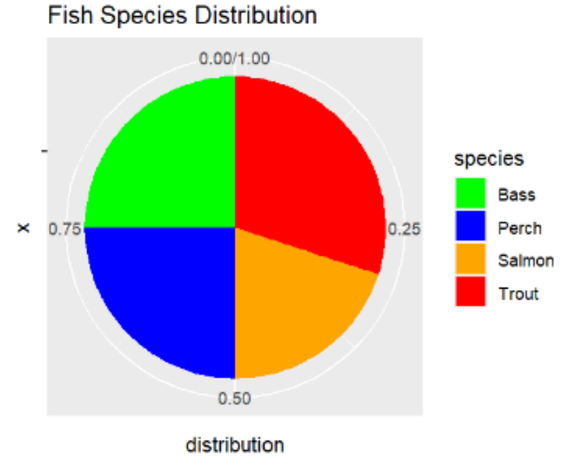
R

Data visualization code for creating a pie chart in R

```
# Data visualization: Pie chart
# Create sample data for fish species distribution
fish_species <- c("Trout", "Salmon", "Bass", "Perch")
fish_distribution <- c(0.3, 0.2, 0.25, 0.25)
fish_data <- data.frame(species = fish_species, distribution =
fish_distribution)

# Create pie chart with color enhancements
ggplot(fish_data, aes(x = "", y = distribution, fill = species)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  scale_fill_manual(values = c("green", "blue", "orange", "red")) +
  labs(title = "Fish Species Distribution")
```

Output of pie chart in R



Data visualization

Python

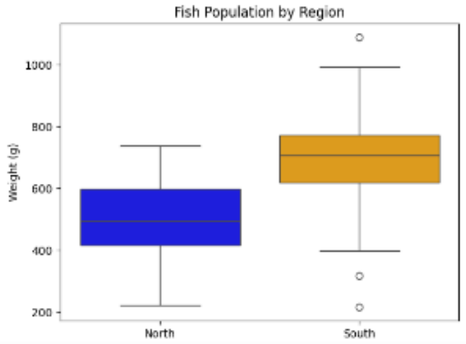
Data visualization code for creating a box plot in Python

```python
# Data visualization: Box plot
import seaborn as sns
import pandas as pd
import numpy as np

# Create sample data for fish population in different regions
np.random.seed(123)
region = np.repeat(['North', 'South'], 100)
weight = np.concatenate([np.random.normal(500, 100, 100),
np.random.normal(700, 150, 100)])
fish_data = pd.DataFrame({'region': region, 'weight': weight})

# Create box plot with color enhancements
sns.boxplot(x='region', y='weight', data=fish_data, palette=['blue',
'orange'])
plt.title('Fish Population by Region')
plt.xlabel('Region')
plt.ylabel('Weight (g)')
plt.show()
```

Output of box plot in Python



R

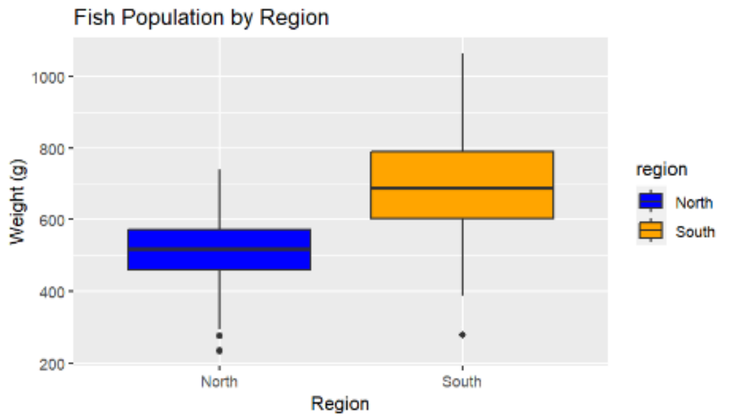
Data visualization code for creating a box plot in R

```r
# Data visualization: Box plot
# Load required libraries
library(ggplot2)

# Create sample data for fish population in different regions
fish_data <- data.frame(
  region = rep(c("North", "South"), each = 100),
  weight = c(rnorm(100, mean = 500, sd = 100), rnorm(100, mean = 700, sd =
150))
)

# Create box plot with color enhancements
ggplot(fish_data, aes(x = region, y = weight, fill = region)) +
  geom_boxplot() +
  scale_fill_manual(values = c("blue", "orange")) +
  labs(title = "Fish Population by Region", x = "Region", y = "Weight (g)")
```

Output of box plot in R



Data visualization

Python

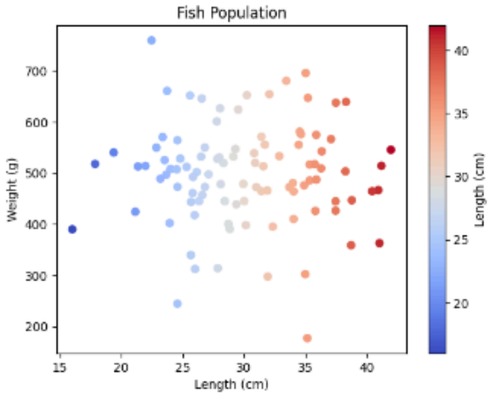
Data visualization code for creating a scatter plot in Python

```python
# Data visualization: Scatter plot
import matplotlib.pyplot as plt
import numpy as np

# Create sample data for fish population
np.random.seed(123)
length = np.random.normal(loc=30, scale=5, size=100)
weight = np.random.normal(loc=500, scale=100, size=100)

# Create scatter plot with color enhancements
plt.scatter(length, weight, c=length, cmap='coolwarm')
plt.colorbar(label='Length (cm)')
plt.title('Fish Population')
plt.xlabel('Length (cm)')
plt.ylabel('Weight (g)')
plt.show()
```

Output of scatter plot in Python



R

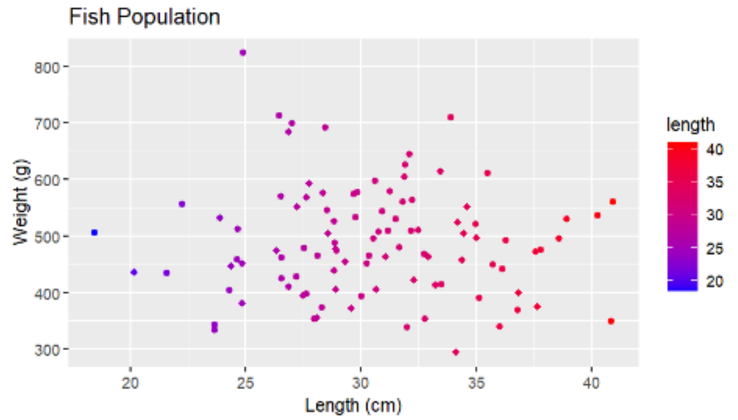
Data visualization code for creating a scatter plot in R

```r
# Data visualization: scatter plot
# Load required libraries
library(ggplot2)

# Create sample data for fish population
set.seed(123)
fish_data <- data.frame(
  length = rnorm(100, mean = 30, sd = 5),
  weight = rnorm(100, mean = 500, sd = 100)
)

# Create scatter plot with color enhancements
ggplot(fish_data, aes(x = length, y = weight, color = length)) +
  geom_point() +
  scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Fish Population", x = "Length (cm)", y = "Weight (g)")
```

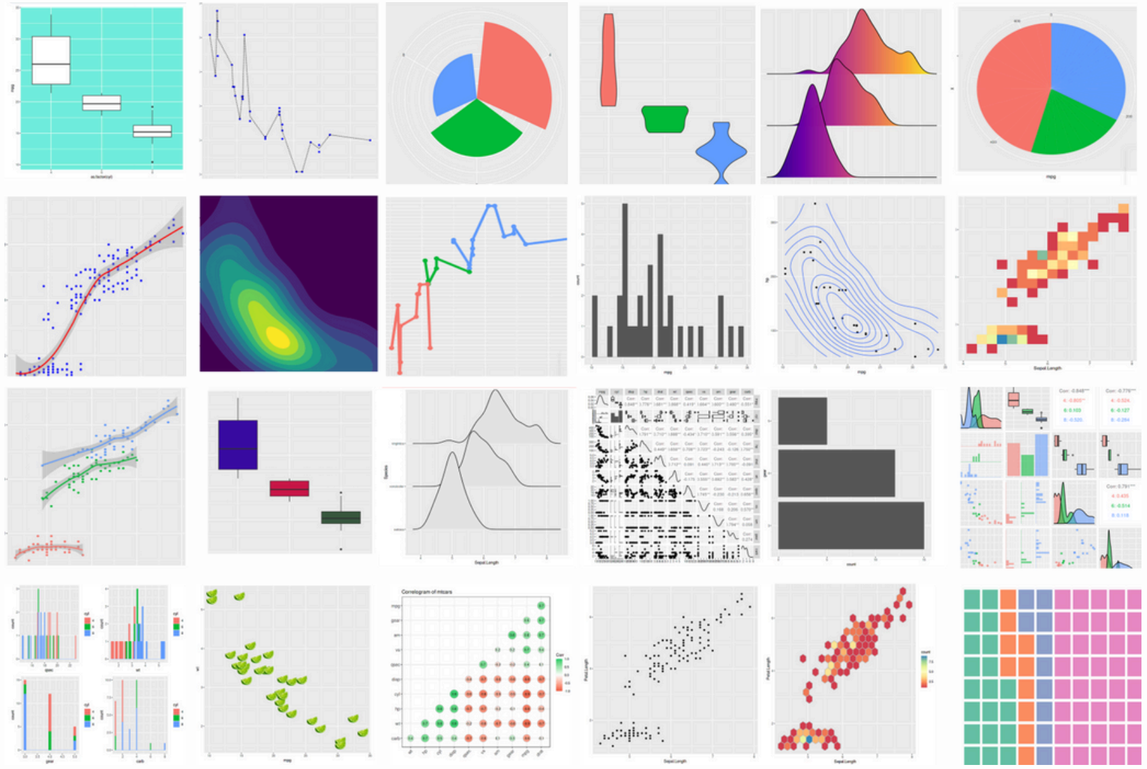
Output of scatter plot in R



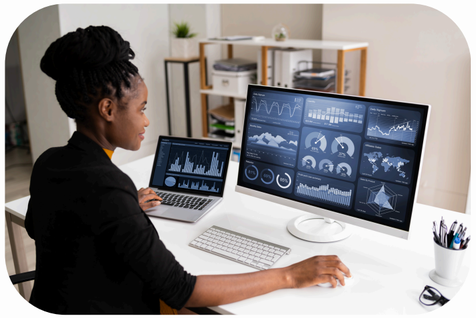
Data Visualization

Python

R



Opportunity to job sectors



Data analysts

Python's versatility and extensive libraries make it well-suited for data analysis tasks, including data cleaning, exploration, and visualization.

Data Engineers

Python's robust ecosystem and efficient data processing capabilities make it a preferred choice for data engineering tasks, such as data extraction, transformation, and loading (ETL)

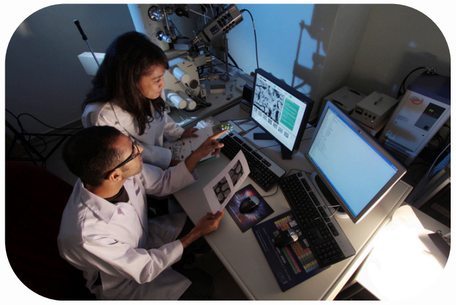
Data Scientists

Python's rich set of libraries for machine learning, combined with its flexibility and integration, make it indispensable for DS. However, DS can use R for statistics

Opportunity to job sectors

ML Engineers

Python's dominance in machine learning libraries like TensorFlow, PyTorch, and Scikit-learn makes it the preferred choice for building and deploying machine learning models.



Academic Researchers

Both Python and R are widely used in academia for data analysis, statistical modeling, and research purposes. The choice between the two often depends on the researcher's domain expertise and project requirements.

Statisticians

R remains popular among statisticians for its specialized statistical functions and packages, Python's growing ecosystem for statistical analysis offers a viable alternative.

Conclusion

The choice between Python and R ultimately depends on your specific needs and career goals:

Need only master one programming language

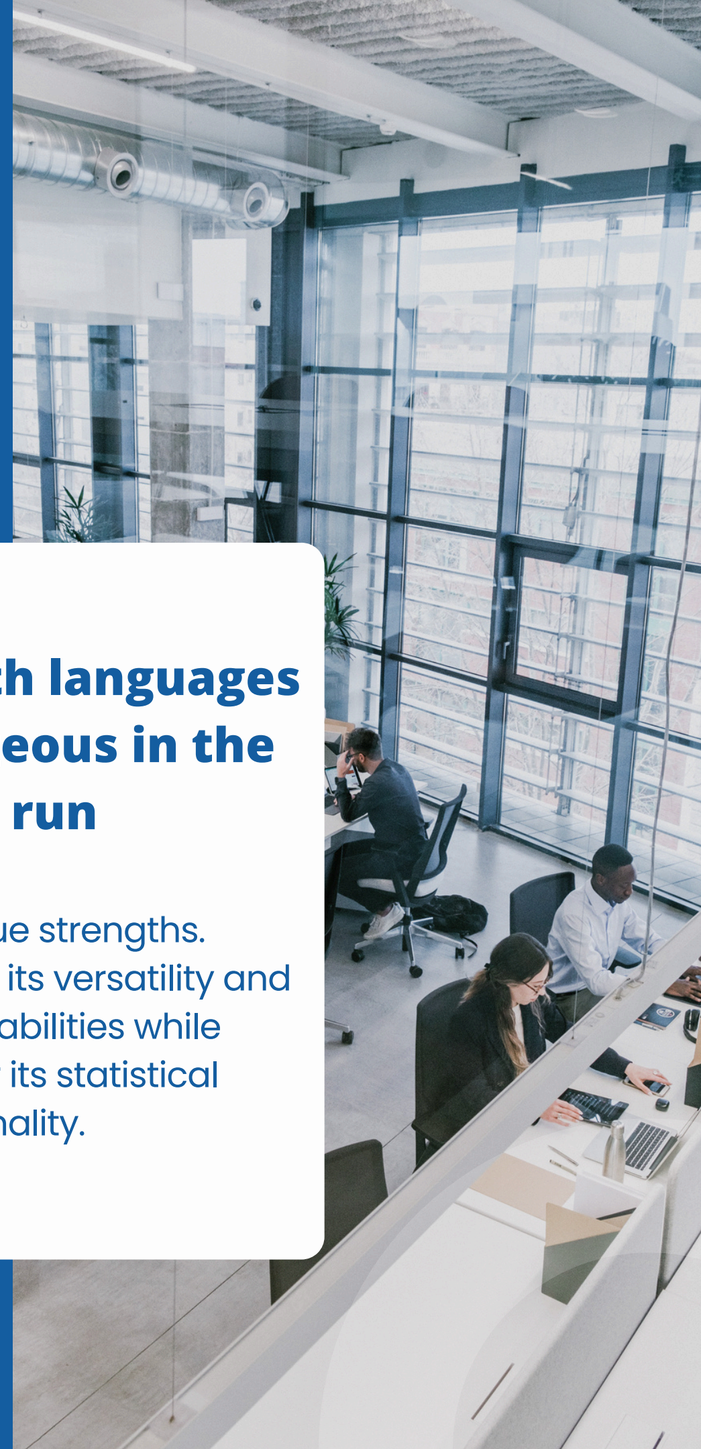
If you prioritize ease of learning, community support, and broad applicability across various data science and non-data science domains, Python could be the ideal choice.

Your job requires heavily on statistics & domain specific

If your job heavily focused on statistical analysis, data visualization, and utilizing advanced statistical packages, R might be good for you

Learning both languages is advantageous in the long run

Both offer unique strengths. Leverage Python for its versatility and production capabilities while employing R for its statistical functionality.



References

- **Stackoverflow. (2021). Developer Survey 2021. Link**
- **Mooney, Paul. (2023). Kaggle Survey 2022 all results. Link**
- **TIOBE. (2024). TIOBE Index for February 2024. Link**
- **Luna, Javier. (2023). Top Programming language for Data Scientists in 2023. Link**
- **Statista. (2023). Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025. Link**
- **Koopingshung. (2020). Four levels of analytics/Data Science. Link**
- **Freund, Nicholas. (2022). Moving from Being A data-driven organization to a data-led organization. Link**
- **CNBC Indonesia. (2023). Jokowi: Data adalah 'new oil' yang berharga tak terhingga. Link**

THANK YOU!

My Contact

https://github.com/harishmuh

www.linkedin.com/in/harish-muhammad-7b600b102/

harishmuh@gmail.com

APPENDIX
STATISTICAL ANALYSIS

Statistical Analysis

ANOVA with R

```
# ================== Anova
# Load necessary libraries
library(tidyr)
library(dplyr)
library(stats)

# Dummy data for fish weight after different treatments of artificial diets
fish_data <- data.frame(
  Treatment = factor(rep(c("Control", "Diet1", "Diet2", "Diet3"), each =
20)),
  Weight = c(100, 98, 105, 103, 101, 97, 102, 99, 96, 104,
             110, 112, 108, 115, 113, 111, 114, 109, 107, 106,
             120, 122, 118, 121, 119, 117, 123, 125, 124, 126,
             130, 132, 128, 129, 131, 135, 127, 134, 133, 136)
)

# Perform ANOVA
anova_result <- aov(Weight ~ Treatment, data = fish_data)

# Perform Tukey's test for multiple comparisons
tukey_result <- TukeyHSD(anova_result)

# Summary of ANOVA
summary(anova_result)

# Summary of Tukey's test
summary(tukey_result)

# Corrected summary of Tukey's test
print(tukey_result)

# R experiment
# Generate simulated data in R
set.seed(123)  # for reproducibility

# Create a data frame with treatment groups and corresponding weights
fish_data <- data.frame(
  Treatment = factor(rep(c("Control", "Diet1", "Diet2", "Diet3"), each =
20)),
  Weight = c(rnorm(20, mean = 100, sd = 10),
             rnorm(20, mean = 110, sd = 10),
             rnorm(20, mean = 120, sd = 10),
             rnorm(20, mean = 125, sd = 10))
)

# Perform ANOVA
anova_result <- aov(Weight ~ Treatment, data = fish_data)

# Print ANOVA summary
summary(anova_result)
```

Statistical Analysis

ANOVA output with R

```
Df Sum Sq Mean Sq F value Pr(>F)
Treatment    3   8820    2940      84 <2e-16 ***
Residuals   76   2660      35
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> # Summary of Tukey's test
> summary(tukey_result)
          Length Class  Mode
Treatment 24     -none- numeric
>
> # Corrected summary of Tukey's test
> print(tukey_result)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Weight ~ Treatment, data = fish_data)

$Treatment
                       diff        lwr        upr p adj
Diet1-Control  2.100000e+01  16.085713  25.914287     0
Diet2-Control  1.421085e-14  -4.914287   4.914287     1
Diet3-Control  2.100000e+01  16.085713  25.914287     0
Diet2-Diet1   -2.100000e+01 -25.914287 -16.085713     0
Diet3-Diet1    2.842171e-14  -4.914287   4.914287     1
Diet3-Diet2    2.100000e+01  16.085713  25.914287     0
```

Statistical Analysis

ANOVA with Python

```python
# Load necessary libraries
import pandas as pd
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import MultiComparison

# Dummy data for fish weight after different treatments of artificial
diets
control_weights = [100, 98, 105, 103, 101, 97, 102, 99, 96, 104]
diet1_weights = [110, 112, 108, 115, 113, 111, 114, 109, 107, 106]
diet2_weights = [120, 122, 118, 121, 119, 117, 123, 125, 124, 126]
diet3_weights = [130, 132, 128, 129, 131, 135, 127, 134, 133, 136]

fish_data = pd.DataFrame({
    'Treatment': ['Control']*len(control_weights) +
['Diet1']*len(diet1_weights) + ['Diet2']*len(diet2_weights) +
['Diet3']*len(diet3_weights),
    'Weight': control_weights + diet1_weights + diet2_weights +
diet3_weights
})

# Perform ANOVA
anova_result = f_oneway(*[group['Weight'] for name, group in
fish_data.groupby('Treatment')])

# Perform Tukey's test for multiple comparisons
mc = MultiComparison(fish_data['Weight'], fish_data['Treatment'])
tukey_result = mc.tukeyhsd()

# Print ANOVA result
print("ANOVA p-value:", anova_result.pvalue)

# Print Tukey's test results
print(tukey_result)
```

Output in Python

```
ANOVA p-value: 2.2369179259417636e-22
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1 group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Control  Diet1     10.0   0.0  6.3534 13.6466   True
Control  Diet2     21.0   0.0 17.3534 24.6466   True
Control  Diet3     31.0   0.0 27.3534 34.6466   True
  Diet1  Diet2     11.0   0.0  7.3534 14.6466   True
  Diet1  Diet3     21.0   0.0 17.3534 24.6466   True
  Diet2  Diet3     10.0   0.0  6.3534 13.6466   True
-----------------------------------------------------
```