

Importação de Dados em Julia

Com Comparação a R e Python

Uma análise comparativa de desempenho e facilidade de uso para importação de dados em três linguagens poderosas de ciência de dados.



Por Que Importar Dados Corretamente?



Fundação da Análise

A importação de dados é o primeiro passo crítico em qualquer projeto de data science.



Formatos Diversos

CSV, TXT, XLSX e arquivos grandes são os formatos mais comuns na prática profissional.



Performance Importa

Comparar Julia, R e Python revela diferenças significativas em velocidade e eficiência.



Criando Bases de Exemplo em Julia

Bases Geradas

- CSV pequeno: 10 registros com dados pessoais
- TXT com separador tab: 5 produtos
- CSV grande: 100.000 linhas de dados
- XLSX: planilha Excel com múltiplas colunas

Todas as bases foram criadas com dados aleatórios para testes realistas de importação.

```
using CSV, DataFrames, XLSX  
df_peq = DataFrame(  
    id = 1:10,  
    nome = ["Ana","Bruno",...],  
    idade = rand(18:60, 10),  
    salario = rand(2500:9000, 10)  
)  
CSV.write("dados_pequenos.csv", df_peq)
```



Importação em Julia: Resultados Reais

01

CSV Pequeno

10 registros importados com CSV.read() - Ana, Bruno, Carla e outros colaboradores com idade e salário.

02

TXT com Tab

5 produtos (A-E) com preços entre 26-89 e estoque variando de 30 a 185 unidades.

03

CSV Grande

100.000 linhas importadas com sucesso - categorias A-D, valores 1000-5000, quantidade 1-100.

04

XLSX

Planilha Excel lida com XLSX.readtable() e convertida para DataFrame.



Benchmark Julia: 30.5ms

100K

Linhas Processadas

Arquivo CSV grande com quatro
colunas de dados variados.

30.5

Milissegundos

Tempo médio de importação
usando CSV.read() com
BenchmarkTools.

Julia demonstra excelente desempenho com código limpo e estável para
grandes volumes de dados.



Importação em R: Duas Abordagens

`data.table::fread()`

```
library(data.table)  
df_big <- fread("dados_grandes.csv")
```

Tempo: 6ms

A função mais rápida de todas as testadas, otimizada para grandes volumes.

`readr::read_csv()`

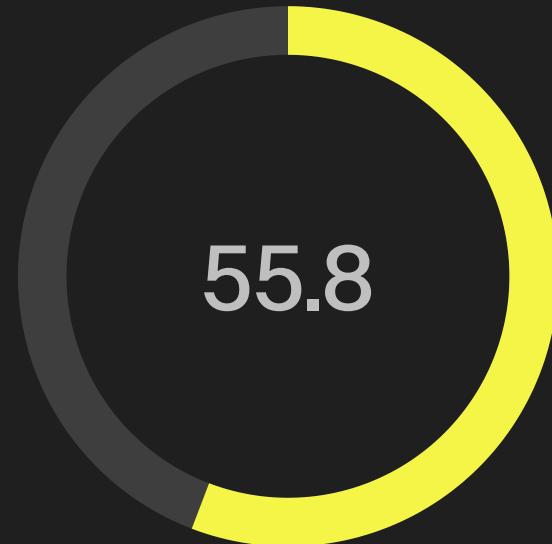
```
library(readr)  
df_csv <- read_csv("dados_pequenos.csv")
```

Tempo: 21ms

Boa usabilidade com sintaxe moderna do tidyverse.

Python: Simplicidade com Pandas

```
import pandas as pd  
df = pd.read_csv("dados_grandes.csv")
```



Tempo Médio (ms)

Média de 5 execuções com `pandas.read_csv()`

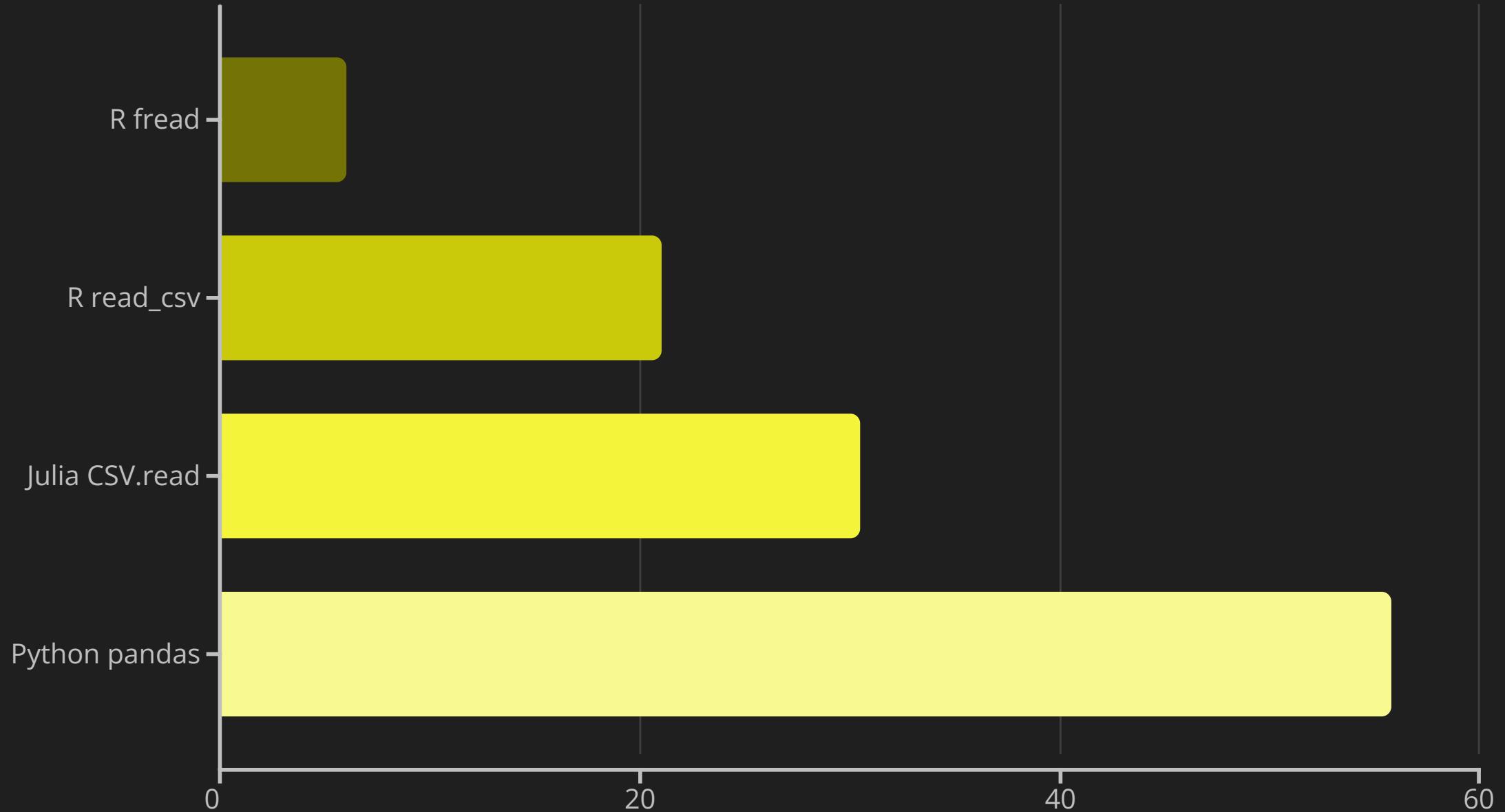


Tempo Mínimo (ms)

Melhor resultado obtido entre as execuções

Python oferece a sintaxe mais simples e intuitiva, mas com desempenho inferior para grandes volumes.

Comparação Final de Performance



R (fread) lidera em velocidade bruta, seguido por Julia com desempenho competitivo e código elegante.

Conclusões Principais

R: Campeão de Velocidade

`fread()` oferece o melhor desempenho bruto para CSV, ideal para grandes volumes de dados.

Julia: Equilíbrio Perfeito

Desempenho competitivo com código limpo e estável, excelente para projetos científicos.

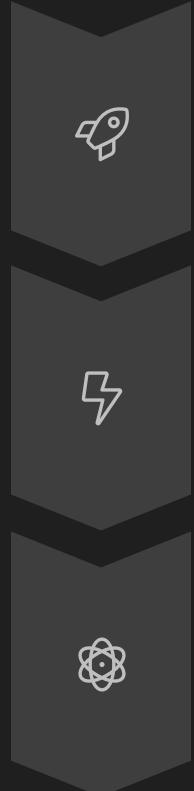
Python: Facilidade de Uso

Sintaxe mais simples e intuitiva, mas desempenho inferior em grandes volumes.

Compatibilidade Universal

Todas as três linguagens importam TXT e XLSX sem dificuldades significativas.

Qual Linguagem Escolher?



Para Análise Rápida

Python - Sintaxe simples, ecossistema rico, ideal para prototipagem.

Para Performance Máxima

R (fread) - Velocidade incomparável em importação de CSV grandes.

Para Projetos Científicos

Julia - Alta performance geral com código elegante e moderno.

