
Polars Ruby

:fire: Blazingly fast DataFrames for Ruby, powered by Polars



Installation

Add this line to your application's Gemfile:

```
1 gem "polars-df"
```

Getting Started

This library follows the Polars Python API.

```
1 Polars.read_csv("iris.csv")
2 .lazy
3 .filter(Polars.col("sepal_length") > 5)
4 .group_by("species")
5 .agg(Polars.all.sum)
6 .collect
```

You can follow Polars tutorials and convert the code to Ruby in many cases. Feel free to open an issue if you run into problems.

Reference

- Series
- DataFrame
- LazyFrame

Examples

Creating DataFrames

From a CSV

```
1 Polars.read_csv("file.csv")
2
3 # or lazily with
4 Polars.scan_csv("file.csv")
```

From Parquet

```
1 Polars.read_parquet("file.parquet")
2
3 # or lazily with
4 Polars.scan_parquet("file.parquet")
```

From Active Record

```
1 Polars.read_database(User.all)
2 # or
3 Polars.read_database("SELECT * FROM users")
```

From JSON

```
1 Polars.read_json("file.json")
2 # or
3 Polars.read_ndjson("file.ndjson")
4
5 # or lazily with
6 Polars.scan_ndjson("file.ndjson")
```

From Feather / Arrow IPC

```
1 Polars.read_ipc("file.arrow")
2
3 # or lazily with
4 Polars.scan_ipc("file.arrow")
```

From Avro

```
1 Polars.read_avro("file.avro")
```

From a hash

```
1 Polars::DataFrame.new({
2   a: [1, 2, 3],
3   b: ["one", "two", "three"]
4 })
```

From an array of hashes

```
1 Polars::DataFrame.new([
2   {a: 1, b: "one"},
3   {a: 2, b: "two"},
4   {a: 3, b: "three"}
5 ])
```

From an array of series

```
1 Polars::DataFrame.new([
2   Polars::Series.new("a", [1, 2, 3]),
3   Polars::Series.new("b", ["one", "two", "three"])
4 ])
```

Attributes

Get number of rows

```
1 df.height
```

Get column names

```
1 df.columns
```

Check if a column exists

```
1 df.include?(name)
```

Selecting Data

Select a column

```
1 df["a"]
```

Select multiple columns

```
1 df[["a", "b"]]
```

Select first rows

```
1 df.head
```

Select last rows

```
1 df.tail
```

Filtering

Filter on a condition

```
1 df[Polars.col("a") == 2]
2 df[Polars.col("a") != 2]
3 df[Polars.col("a") > 2]
```

```
4 df[Polars.col("a") >= 2]
5 df[Polars.col("a") < 2]
6 df[Polars.col("a") <= 2]
```

And, or, and exclusive or

```
1 df[(Polars.col("a") > 1) & (Polars.col("b") == "two")] # and
2 df[(Polars.col("a") > 1) | (Polars.col("b") == "two")] # or
3 df[(Polars.col("a") > 1) ^ (Polars.col("b") == "two")] # xor
```

Operations

Basic operations

```
1 df["a"] + 5
2 df["a"] - 5
3 df["a"] * 5
4 df["a"] / 5
5 df["a"] % 5
6 df["a"] ** 2
7 df["a"].sqrt
8 df["a"].abs
```

Rounding

```
1 df["a"].round(2)
2 df["a"].ceil
3 df["a"].floor
```

Logarithm

```
1 df["a"].log # natural log
2 df["a"].log(10)
```

Exponentiation

```
1 df["a"].exp
```

Trigonometric functions

```
1 df["a"].sin
2 df["a"].cos
3 df["a"].tan
4 df["a"].asin
5 df["a"].acos
6 df["a"].atan
```

Hyperbolic functions

```
1 df["a"].sinh
2 df["a"].cosh
3 df["a"].tanh
4 df["a"].asinh
5 df["a"].acosh
6 df["a"].atanh
```

Summary statistics

```
1 df["a"].sum
2 df["a"].mean
3 df["a"].median
4 df["a"].quantile(0.90)
5 df["a"].min
6 df["a"].max
7 df["a"].std
8 df["a"].var
```

Grouping

Group

```
1 df.groupby("a").count
```

Works with all summary statistics

```
1 df.groupby("a").max
```

Multiple groups

```
1 df.groupby(["a", "b"]).count
```

Combining Data Frames

Add rows

```
1 df.vstack(other_df)
```

Add columns

```
1 df.hstack(other_df)
```

Inner join

```
1 df.join(other_df, on: "a")
```

Left join

```
1 df.join(other_df, on: "a", how: "left")
```

Encoding

One-hot encoding

```
1 df.to_dummies
```

Conversion

Array of hashes

```
1 df.rows(named: true)
```

Hash of series

```
1 df.to_h
```

CSV

```
1 df.to_csv
2 # or
3 df.write_csv("file.csv")
```

Parquet

```
1 df.write_parquet("file.parquet")
```

Numo array

```
1 df.to_numo
```

Types

You can specify column types when creating a data frame

```
1 Polars::DataFrame.new(data, schema: {"a" => Polars::Int32, "b" =>
    Polars::Float32})
```

Supported types are:

- boolean - `Boolean`

-
- float - `Float64`, `Float32`
 - integer - `Int64`, `Int32`, `Int16`, `Int8`
 - unsigned integer - `UInt64`, `UInt32`, `UInt16`, `UInt8`
 - string - `String`, `Binary`, `Categorical`
 - temporal - `Date`, `Datetime`, `Time`, `Duration`
 - nested - `List`, `Struct`, `Array`
 - other - `Object`, `Null`

Get column types

```
1 df.schema
```

For a specific column

```
1 df["a"].dtype
```

Cast a column

```
1 df["a"].cast(Polars::Int32)
```

Visualization

Add Vega to your application's Gemfile:

```
1 gem "vega"
```

And use:

```
1 df.plot("a", "b")
```

Specify the chart type (`line`, `pie`, `column`, `bar`, `area`, or `scatter`)

```
1 df.plot("a", "b", type: "pie")
```

Group data

```
1 df.group_by("c").plot("a", "b")
```

Stacked columns or bars

```
1 df.group_by("c").plot("a", "b", stacked: true)
```

History

View the changelog

Contributing

Everyone is encouraged to help improve this project. Here are a few ways you can help:

- Report bugs
- Fix bugs and submit pull requests
- Write, clarify, or fix documentation
- Suggest or add new features

To get started with development:

```
1 git clone https://github.com/ankane/polars-ruby.git
2 cd polars-ruby
3 bundle install
4 bundle exec rake compile
5 bundle exec rake test
6 bundle exec rake test:docs
```