

Learn to code – free 3,000-hour curriculum

APRIL 25, 2022 / #GOLANG

Data Analysis in Go – How to Use the Gota Package



Ukeje Chukwuemeriwo Goodness

Data analysis is the process of filtering, manipulating, and processing raw data and datasets to get insights from them.

Python and R are usually the go-to languages for data analysis. But these days Go is becoming more and more popular for this purpose.

In this tutorial, we will be going over Gota, a data analysis package in Go, and its core functions and uses.

Prerequisites

- Knowledge of functional programming in Golang.
- A Golang IDE with Go installed (I use Goland and Go 1.17.6, but you can use any other)

What is Gota?

Gota is a dataframe and data wrangling module for the Go programming language.

Learn to code – free 3,000-hour curriculum

like Pandas and Numpy.

The Gota module makes data wrangling (transforming and manipulating) operations in Go very easy. It works with Go inbuilt data types and various file formats, like JSON, CSV, and HTML.

Here's what we'll cover:

- Gota Series.
- Gota Dataframes.
- Reading Files as Dataframes.
- Operations On Gota Dataframes.
- Exporting and Saving Files.

How to Get Started with Gota

Installing Gota is easy. Paste the command below into your terminal:

```
go get -u https://github.com/go-gota/gota
```

This should output a successful install message. If it doesn't, update your Golang to a newer version and retry installing.

Basic Gota Concepts

Learn to code – free 3,000-hour curriculum

Column	Column	Column
Row	Value 2	Value 3
Row	Value 5	Value 6
Row	Value 8	Value 9
Row	Value 11	Value 12

→ Series

Let's learn a bit about some Gota basics before diving in:

A **Dataset** is a collection of data, tabular or otherwise.

Dataframes are data structures that organise data into two-dimensional (rows and columns) tables, usually for the purpose of analysis.

A **series** is a collection of one-dimensional data belonging to a dataframe.

Note that `DataFrame` is the variable name of the dataframe object used as an example throughout this article.

What are Gota Series?

```
import "github.com/go-gota/gota/series"
```

Gota series are created using the `series.New` method on compound data types, like slices and maps.

Learn to code – free 3,000-hour curriculum
BE CONTAINED IN THE SERIES), AND A COLUMN NAME.

```
series.New([]string{"z", "y", "d", "e"}, series.String, "col")
```

Series can also be made from maps by initializing the keys to be of type `series` and using the `Type` method to insert series types.

```
a := map[string]series.Type{
    "A": series.String,
    "D": series.Bool,
}
```

These slices can be passed into dataframes for further manipulation and operations.

What are Gota Dataframes?

Dataframe functions are contained in Gota in the `dataframe` sub-module.

```
import "github.com/go-gota/gota/dataframe"
```

Dataframes are data structures of other data structures. Essentially, they format the data into two-dimensional tables so you can manipulate those data. So to use dataframes, we read other data structures and data types.

Learn to code – free 3,000-hour curriculum

How to Convert a Series to a Dataframe Object

You can convert a series or set of series to a dataframe object using the `dataframe.New` method. It takes in the series as arguments.

```
dataFrame := dataframe.New(  
    series.New([]string{"a", "b", "c", "d", "e"}, series.Stri  
    series.New([]int{5, 4, 2, 3, 1}, series.Int, "numbers"),  
    series.New([]string{"a1", "b2", "c3", "d4", "e5"}, series  
    series.New([]bool{true, false, true, true, false}, series  
)  
  
fmt.Println(dataFrame)
```

Output:

```
alphas  numbers  alnums  state  
0: a      5        a1      true  
1: b      4        b2      false  
2: c      2        c3      true  
3: d      3        d4      true  
4: e      1        e5      false  
    <string> <int>    <string> <bool>
```

Dataframe of Structs Types

You can use structs to create dataframes.

Learn to code – free 3,000-hour curriculum

```
    Height int
    Vaccinated bool
}

dogs := []Dog{
    {"buster", "black", 56, false},
    {"jake", "white", 61, false},
    {"bingo", "brown", 50, true},
    {"gray", "cream", 68, false},
}

dogsDf := dataframe.LoadStructs(dogs)
fmt.Println(dogsDf)
```

You do this by creating a slice of instances of the struct type and creating dataframes using the `dataframe.LoadStructs` method which takes in the slice.

Output:

```
Name      Colour  Height Vaccinated
0: buster  black   56     false
1: jake    white   61     false
2: bingo   brown   50     true
3: gray    cream   68     false
  <string> <string> <int>  <bool>
```

How to Query the Dataframe in Gota

When we have a dataframe object, we can query it for information about the composition of the dataframe using various methods.

- `dataFrame.Dims()` → Outputs the dimensions of the

Learn to code – free 3,000-hour curriculum

- `dataframe.Types()` → Outputs the datatypes that constitute the dataframe.
- `dataframe.Names()` → Outputs the column names of the dataframe.
- `dataframe.Nrow()` → Outputs the number of rows.
- `dataframe.Ncol()` → Outputs the number of columns.

How to Query Columns

There are many methods that come with a Gota dataframe column that help with querying column values.

- `.IsNaN()` → Checks if it's a null column.
- `.Mean()` → Returns the mean (average) value of the column.
- `.Copy()` → Creates a new copy of the column.
- `.HasNaN()` → Checks if there's a null value in the column.
- `.Records()` → returns the values in the column.

```
aCol := dataframe.Col("column_name") //selects a column
fmt.Println(aCol.HasNaN)
```

How to Read Files into Dataframe Objects

JSON and CSV strings can be passed to `dataframe.ReadJSON` and `dataframe.ReadCSV` respectively.

How to Read JSON Strinas

Learn to code – free 3,000-hour curriculum

Read JSON using `strings.NewReader` which returns a buffered JSON string.

```
jsonString := `[
  {
    "Name": "John",
    "Age": 44,
    "Colour": "Red",
    "Height(ft)": 6.7
  },
  {
    "Name": "Mary",
    "Age": 40,
    "Colour": "Blue",
    "Height(ft)": 5.7
  }
]`

dataRead := dataframe.ReadJSON(strings.NewReader(jsonString))
fmt.Println(dataRead)
}
```

How to Read CSV Strings

Here we have the same string in CSV format:

```
import (
    "fmt"
    "github.com/go-gota/gota/dataframe"
    "strings"
)

csvString := `
Name, Age, Colour, Height(ft)
John,44,Red,6.7
Mary,40,Blue,5.7`

dataRead := dataframe.ReadCSV(strings.NewReader(csvString))
fmt.Println(dataRead)
```


Learn to code – free 3,000-hour curriculum

Output.

	Name	Age	Colour	Height(ft)
0:	John	44	Red	6.700000
1:	Mary	40	Blue	5.700000

How to Read CSV Files

Here's the CSV:

```
Name, Age, Colour, Height(ft)
John, 44, Red, 6.7
Mary, 40, Blue, 5.7
Esther, 35, Black, 4.9
Jason, 36, Green, 5.2
```

You can read CSV files by reading a file using `os.Open` which takes in the file name. `defer file.Close()` is a context manager that helps us close the file once the program is run to prevent data loss.

```
file, err := os.Open("example.csv")
defer file.Close()
if err != nil {
    log.Fatal(err)
}
dataFrame := dataframe.ReadCSV(file)

fmt.Println(dataFrame)
```

How to Read JSON Files

Learn to code – free 3,000-hour curriculum

Here's the JSON:

```
[
  {
    "Name": "John",
    "Age": 44,
    "Colour": "Red",
    "Height(ft)": 6.7
  },
  {
    "Name": "Mary",
    "Age": 40,
    "Colour": "Blue",
    "Height(ft)": 5.7
  },
  {
    "Name": "Esther",
    "Age": 35,
    "Colour": "Black",
    "Height(ft)": 4.9
  },
  {
    "Name": "Mary",
    "Age": 40,
    "Colour": "Green",
    "Height(ft)": 5.2
  }
]
```

And here's how you read the file:

```
file, err := os.Open("example.json")
defer file.Close()
if err != nil {
    log.Fatal(err)
}
dataFrame := dataframe.ReadJSON(file)
```

```
fmt.Println(dataFrame)
```

Learn to code – free 3,000-hour curriculum

Output:

```
Age  Colour  Height(ft) Name
0: 44    Red    6.700000  John
1: 40    Blue   5.700000  Mary
2: 35    Black  4.900000  Esther
3: 40    Green  5.200000  Mary
<int> <string> <float> <string>
```

Gota Dataframe Operations

How to Select Rows in Gota

You can select rows using the `Subset` method of the dataframe object. `dataFrame.Subset` takes in a slice of two integers that depict the number of rows that can be selected.

Gota provides multiple functionalities for dataframe manipulation. Using the example dataframe above, let's go over some of these operations:

```
row := dataFrame.Subset([]int{0, 2})
```

This selects the first two rows of the dataframe.

Output:

```
Age  Colour  Height(ft) Name
0: 44    Red    6.700000  John
```

Learn to code – free 3,000-hour curriculum

How to Select Columns in Gota

The `select` method helps us select columns of a dataframe. `df.Select` takes in a slice of two integers that depict how many columns can be selected.

```
column := dataframe.Select([]int{0, 2})
```

We can also select columns by Index (column names) by passing a slice of strings.

```
column := dataframe.Select([]string{"Name", "Colour"})
```

Output:

```
Name      Colour
0: John    Red
1: Mary    Blue
2: Esther  Black
3: Mary    Green
  <string> <string>
```

How to Update the Dataframe in Gota

We use the `.set` method of the dataframe object to update entries. `dataframe.set` takes in a slice of integers specifying the limit of rows to be updated, and a `dataframe.LoadRecords` function which

Learn to code – free 3,000-hour curriculum

```
dataFrame2 := dataframe.Set(
    []int{0, 3},
    dataframe.LoadRecords(
        [][]string{
            []string{"Jenny", "23", "purple", "2.2"},
            []string{"Jesse", "34", "indigo", "3.5"},
            []string{"Peter", "33", "violet", "3.3"},
        },
    ),
)
```

Output:

Name	Age	Colour	Height(ft)
0: Jesse	34	indigo	3.500000
1: Mary	40	Blue	5.700000
2: Esther	35	Black	4.900000
3: Peter	33	violet	3.300000

<string> <int> <string> <float>

How to Filter Values in Gota

To filter values, we use `.Filter` on the dataframe object. This takes in `dataframe.F` which we pass a struct literal to.

The struct literal takes in a column name `Colname`, a comparator `Comparator`, and a value `Comparando` which is the value for which we want to filter the dataframe.

Comparators:

- `series.Eq` → Equal to =.
- `series.Neq` → Not Equal to ≠

Learn to code – free 3,000-hour curriculum

- `series.GreaterEq` → Greater than or Equal to \geq .
- `series.Less` → Less than $<$.
- `series.LessEq` → Less than or Equal to \leq .
- `series.In` → Is contained In.

In this example, we use the dataframe object from the series to the dataframe section above.

```
fil := dataframe.Filter(
    dataframe.F{Colname: "alphas", Comparator: series.Eq, Con
)
```

Output:

```
alphas  numbers  alnums  state
0: b      2      b2      false
<string> <int>  <string> <bool>
```

How to Sort a Dataframe in Gota

You can sort a dataframe using the `.Arrange` method of the dataframe object. It takes in `dataframe.Sort` or `dataframe.RevSort` which sorts in ascending or descending order respectively. It also takes in the name of the column to be sorted as a string.

Sort by Ascending Order:

```
sorted := dataframe.Arrange(
```

Learn to code – free 3,000-hour curriculum

Output:

	alphas	numbers	alnums	state
0:	e	1	e5	false
1:	c	2	c3	true
2:	d	3	d4	true
3:	b	4	b2	false
4:	a	5	a1	true

<string> <int> <string> <bool>

Sort by Descending Order:

```
sorted := dataframe.Arrange(  
    dataframe.RevSort("numbers"),  
)
```

How to Use Groupby in Gota

You can use groupby to categorize data based on specific columns.

To groupby columns using Gota, we use the `Groupby` method and pass in the column names.

```
categorise := dataframe.GroupBy("Name", "Age")  
fmt.Println(categorise)
```

How to Join Dataframes in Gota

Joins are a combination of dataframes. Joining dataframes with

Learn to code – free 3,000-hour curriculum

Types Of Joins:

- Inner Join → `dataFrame.InnerJoin` returns a dataframe of matching values in both tables.
- Left Join → `dataFrame.LeftJoin` matches similarities in the right dataframe to the left dataframe.
- Right Join → `dataFrame.RightJoin` matches similarities in the left dataframe to the right dataframe.
- Outer Join → `dataFrame .OuterJoin` returns all values of the dataframe.

The syntax for joining dataframes objects is:

```
joinVariableName := dataframeObject.joinType(OtherDataframe, Join
```

The Join key is the column of the dataframe object where the join is to be executed.

Example of a Left Join

```
df := dataframe.New(  
    series.New([]string{"a", "b", "c", "d", "e"}, series.Stri  
    series.New([]int{5, 4, 2, 3, 1}, series.Int, "numbers"),  
    series.New([]string{"a1", "b2", "c3", "d4", "e5"}, series  
    series.New([]bool{true, false, true, true, false}, series  
)  
df2 := dataframe.New(  
    series.New([]string{"f", "g", "h", "i", "j"}, series.Stri  
    series.New([]int{1, 2, 3, 4, 5}, series.Int, "numbers"),  
    series.New([]string{"f6", "g7", "h8", "i9", "j10"}, serie  
    series.New([]bool{false, true, false, false, true}, serie  
)
```


Learn to code – free 3,000-hour curriculum

```
fmt.Println(join)
```

Output:

```
[12x7] DataFrame
```

	state	alphas_0	numbers_0	alnums_0	alphas_1	numbers_1	alnums_1
0:	false	b	4	b2	f	1	f6
1:	false	e	1	e5	f	1	f6
2:	true	a	5	a1	g	2	g7
3:	true	c	2	c3	g	2	g7
4:	true	d	3	d4	g	2	g7
5:	false	b	4	b2	h	3	h8
6:	false	e	1	e5	h	3	h8
7:	false	b	4	b2	i	4	i9
8:	false	e	1	e5	i	4	i9
9:	true	a	5	a1	j	5	j10
...
	<bool>	<string>	<int>	<string>	<string>	<int>	<string>

How to Apply Functions to a Dataframe in Gota

To apply functions to columns and rows of a dataframe, we use `Caply` and `Rapply`, respectively. These take in the function to be applied on the column or row.

```
dataFrame.Caply(function)
dataFrame.Rapply(function)
```

Learn to code – free 3,000-hour curriculum

Datatable in Gota

Using `Describe()` on a dataframe object returns descriptive statistics on the values of the dataframe.

```
description := dataframe.Describe()
```

Output:

```
column  alphas  numbers  alnums  state
0: mean    -      3.000000 -      0.600000
1: median  -      3.000000 -      NaN
2: stddev  -      1.581139 -      0.547723
3: min     a      1.000000 a1     0.000000
4: 25%    -      2.000000 -      0.000000
5: 50%    -      3.000000 -      1.000000
6: 75%    -      4.000000 -      1.000000
7: max     e      5.000000 e5     1.000000
  <string> <string> <float>  <string> <float>
```

How to Export Dataframes (Writing Files in Go)

We export the manipulated data using the `WriteCSV` method of the dataframe object. The `dataFrame.WriteCSV` takes in a file name which it creates or inserts into.

```
file, err := os.Create("output.csv")
if err != nil {
    log.Fatal(err)
}
```

Learn to code – free 3,000-hour curriculum

To export JSON, we use `dataFrame.WriteJSON` in the same way.

Conclusion

In this tutorial, you have learned how to perform data analysis in Go. You've also learned about the Gota package's various functions.

It's still a good idea to primarily use Python and R for data analysis as they are considered the industry standard. But Gota is useful for applications requiring speed and homogeneity.

Check out the [Gota documentation](#) to learn more or contribute to the project.

Have fun coding and exploring!



Ukeje Chukwuemeriwo Goodness

I am a Software developer and technical writer, writing about Backend development and distributed ledger technologies

If you read this far, tweet to the author to show them you care.

[Tweet a thanks](#)

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.

[Get started](#)

Learn to code – free 3,000-hour curriculum

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world. Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.

Trending Guides

Zoom Screen Sharing	C++ Vector
Decimal Place Value	What is CPU
How to Get Into BIOS	IPV4 vs IPV6
String to Int in C++	What is IPTV
What is mspeng.exe	HTML Font Size
Facetime Not Working	Change Mouse DPI
Desktop Icons Missing	How to Make a GIF
How to Copy and Paste	Git Rename Branch
Delete a Page in Word	Make a Video Game
vcruntime140.dll Error	CSS Media Queries
How to Open .dat Files	Password Protect Zip File
Record Calls on iPhone	Restore Deleted Word File
Ascending vs Descending	Software Engineering Guide
HTML Email Link Tutorial	How to Find Your IP Address
Python List Comprehension	How to Find iPhone Download

Our Nonprofit

[About](#) [Alumni Network](#) [Open Source](#) [Shop](#) [Support](#) [Sponsors](#) [Academic Honesty](#)

[Code of Conduct](#) [Privacy Policy](#) [Terms of Service](#) [Copyright Policy](#)