1 Data Visualization

1.1 Julia 言語におけるデータの可視化

Julia 言語では、「Makie」,「Plots」などデータ可視化用のパッケージが提供されている。つぎ、「Makie」の特徴について説明しよう。Makie は、Julia で作られた新しいパッケージである。またバイオリンプロットを始めとする高度な可視化ツールを含む。そのため、グラフの描写という目的において Makie は強力なツールとなり得る。また、レイアウトの作成が簡単かつ細部まで技工を凝らすことが出来るというメリットがあり、科学論文に掲載する画像を作成するには最適だろう。次に「Plots」の特徴を説明しよう。「Plots」の特徴は、「PyPlot」や「GR」などの既成品のデータ可視化パッケージをバックエンドとして使用しグラフを描写する。つまり、「Plots」を使うことによって共通のコマンドで異なるパッケージを使用したデータの可視化を行うことが出来る。

そこで本章では、「Makie」,「Plots」の使用方法について説明する。もし、一つのパッケージ使用方法のみを習得したい場合は、「Makie」の学習を強くおすすめする。

1.2 Makie の使用方法について

まず、Makie の使用方法の概要について説明しよう。Makie は、

1. 軸ラベルなどの軸に関する描写を行う Axis layout
2. 図の配置や図の大きさを構成する Figure layout
3. 構成した Figure layout 上に図を描写する関数

の 3 要素を操作することによってグラフを作成することが出来る。このセクションでは、図の書き方を説明した後、Axis layout と Figure layout の操作を説明する。

1.2.1　Makie のインストール

まず、Makie のインストールについて説明しよう。Makie は、デフォルトでは Vector ファイルのサポートがされていない。そこで、Vector ファイルの作成を可能とする「CairoMakie」を同時にインストールする必要がある。その他にも、HTML に画像の埋め込みをするための「WGLMakie」があるので必要に応じてインストールを行う。それらのインストールは、下記のコマンド (Code. 1.2.1) を実行することによって完了することが出来る。

Makie のインストール

```
1  using Pkg;
2  Pkg.add("Makie")
3  # vector ファイルを作成するために必要
4  Pkg.add("CairoMakie")
5  # に埋め込む場合HTML
6  # Pkg.add("WGLMakie")
```

1.2.2　Makie を使用したグラフの描写

この章では、

- 一次元グラフ
  1. ヒストグラム (histogram)
  2. カーネル密度推定 (kernel density plot)

3. バイオリンプロット (violin plot)
4. 棒グラフ (bar plot)

- 2 次元グラフ
  1. 線グラフ (line plot)
  2. 散布図 (scatter plot)
  3. エラーバー (error bar plot)
  4. ベクトル図 (arrow plot)
  5. ヒートマップ (heatmap)
- 3 次元グラフ
  1. ヒストグラム (histogram)
  2. メッシュプロット (mesh plot)

の描写方法について説明する。また、使用可能なオプションに関しても適宜紹介する。

ここで、Makie を使用した基本的なグラフの描写方法について説明しよう。Makie を使用してグラフを書く際には、

Step1.　「Figure」関数を呼び出す
Step2.　「Axis」関数を呼び出し、どの窓に描写するかを定義する
Step3.　どの Axis に描写するかを明示的に示した上でグラフを書く

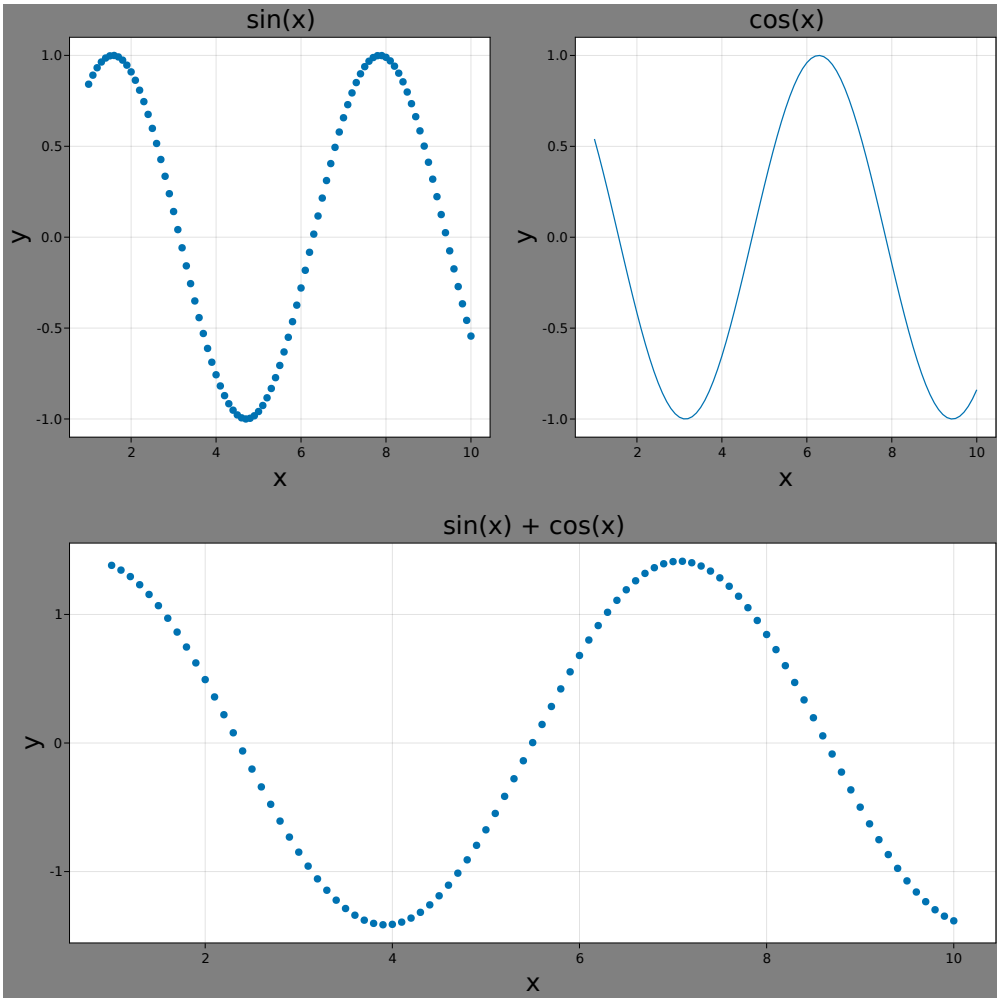
の手続きを踏む。具体的な例として、2×2 のレイアウトのグラフの描写手続きについて説明しよう。

Example

```
using CairoMakie, Makie
# Figure 関数を呼び出す。
# resolution = ( 縦の大きさ, 横の大きさ ) で画像の大きさを指定できる
# backgroundcolor = 色で背面の色を指定できる
fig = Figure(resolution = (1200,1200),
             backgroundcolor="gray"
             )
# 使用するグラフの形状のレイアウトを決める
# Axis を使用して Figure を分割する
ax1 = Axis(fig[1,1]) bt6
```

```
11  ax2 = Axis(fig[1,2])
12  ax3 = Axis(fig[2,1:2])
13  x = collect(1:0.1:10)
14  scatter!(ax1, x, sin.(x)
15  title )
16  lines!(ax2, x, cos.(x) )
17  scatter!(ax3, x, sin.(x) + cos.(x) )
18  fig
```

こんな感じで、書く。



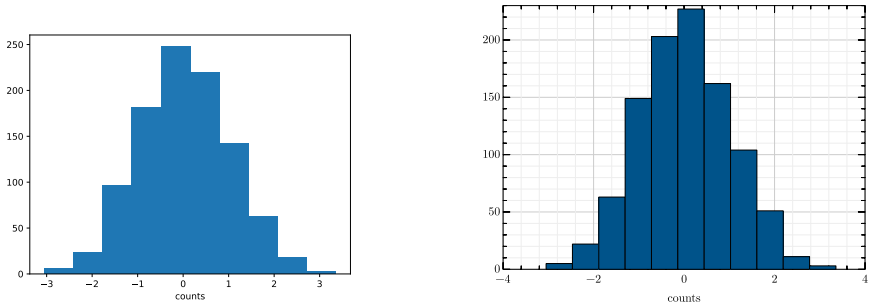
1.2.3　1 次元グラフ

データ分析を行う際には各変数の定性的な性質を分析することが重要である。例えば、分布関数の形状や条件付き分布などが統計量の定性的な性質を示す統計量である。これらの統計的な定性的な性質を調べることは、理論モデルの発展や適切な統計分析を行う上で重要である。

まず、分布関数の定性的な性質を描写するヒストグラムの描写方法について説明しよう。ここでは、サンプルデータとして iris データセットを使用する。

2 次元グラフ

線グラフは、独立変数である x と従属変数である y に明確な関係がある時に有効な可視化手法である。例えば、観測されるデータに $y = x^2$ という関係式が期待される場合を考えよう。



▲ 図 1.1: 左:PyPlots を使用した histogram の描写。右:GR を使用した histogram の描写

1.3 Plots

ここで、Plots について紹介しよう。まず、Plots の特徴である「共通のコマンド使用しグラフを描写」を説明するために「PyPlot」と「GR」を使用したヒストグラムのコード (Code 1.1) を比較しよう。

Code 1.1: PyPlot と GR を使用したヒストグラムの描写

```
1  using GR, PyPlot
2  plt = PyPlot
3  gr = GR
4  # PyPlot
5  x = randn(1000)
6  plt.hist(x)
7  plt.xlabel("counts")
8  plt.savefig("hist_pyplot.pdf")
9  plt.show()
10 # GR
11 gr.histogram(x)
12 gr.xlabel("counts")
13 gr.savefig("hist_gr.pdf")
```

このようにしてヒストグラムを生成できるのだが、「PyPlot」では「hist」、「GR」では「histogram」のように使用する関数名が異なる。この差は異なるパッケージを使用する時

に、エラー発生確率の上昇 (GR を使用している時に PyPlot のコマンドを誤って使用する) のような形で現れると期待される。そのため、関数名が揃っていることは重要である。

1.4 Makie

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 1.1: 表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。