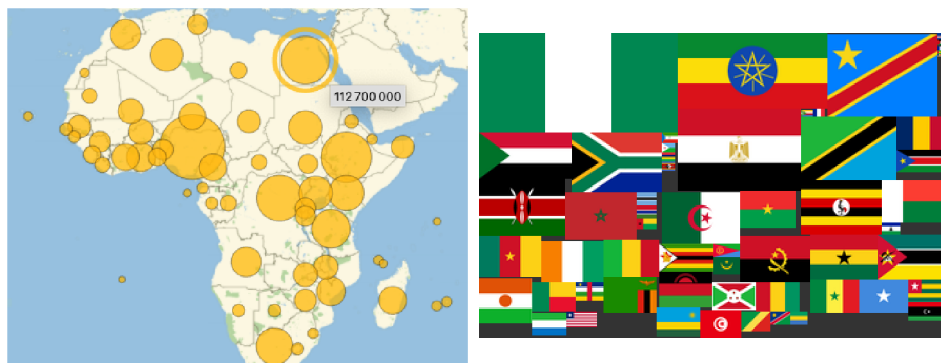


実践編6：アフリカ大陸の国のデータを可視化しよう

Wolfram 知識ベースにはさまざまなタイプの計算可能な知識データが用意されています。知識ベースのデータは「実体（Entity）」と呼ばれ、例えば、国の実体（Entity）には、その国の地理データ、人口やGDPなどの社会・経済データ、天気のデータや有名人のデータまで多くのデータが紐づいています。


この演習では、アフリカ大陸の各国に関する“Entity”（実体），“Name”（名前表記），“Population”（人口），“GDP”，“FlagImage”（国旗）のデータを使って、地図上に色分けして描画したり、ワードクラウドにしたり、国旗のコラージュを作ったりして、データを可視化します。同じデータでも可視化の手法で見えてくるものが違ってきます。さまざまなデータの可視化を体験しましょう。（*1）

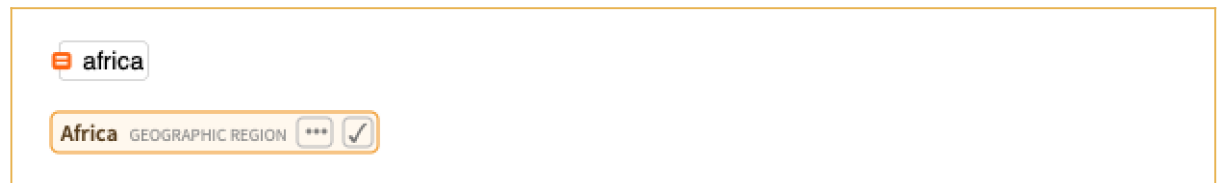


図P6-1 アフリカ大陸の国のデータの可視化の例

（*1）Wolfram知識ベースの出力は、Wolfram Cloudでは英語で表示されることがあります。

知識ベースのデータは「実体 (Entity)」

最初に、`CTRL`キーとセミコロン(;)または`CTRL`キーとイコール(=)を入力して  を表示します。このボックスのなかに「africa」と入力してEnterキーを押すと、Wolfram言語は「africa」というワードの意味を解釈して、Wolfram知識ベースから「アフリカ大陸」を導き出します。



これは「アフリカ大陸」という地域全体を意味する知識データで「実体(Entity)」と呼ばれます。このオレンジの `Africa` という実体(Entity)の中には、「アフリカ大陸」に関するデータ（国や地形など）が紐づいていますので、`Africa` のうしろに["Countries"]を入力して実行すると、アフリカ大陸の国の実体のリストが得られます。

In[]:= `Africa GEOGRAPHIC REGION ["Countries"]`

Out[]:=

```
{
  アルジェリア, アンゴラ, ベナン, ボツワナ, ブルキナファソ, ブルンジ, カメルーン,
  カーボベルデ, 中央アフリカ共和国, チャド, コモロ, コンゴ民主共和国, ジブチ,
  エジプト, 赤道ギニア, エリトリア, エチオピア, ガボン, ガンビア, ガーナ,
  ギニア, ギニアビサウ, コートジボワール, ケニア, レソト, リベリア, リビア,
  マダガスカル, マラウイ, マリ, モーリタニア, モーリシャス, マヨット,
  モロッコ, モザンビーク, ナミビア, ニジェール, ナイジェリア, コンゴ共和国,
  レユニオン, ルワンダ, セントヘレナ, サントメ・プリンシペ, セネガル, セーシェル,
  シエラレオネ, ソマリア, 南アフリカ共和国, 南スーダン, スーダン, エスワティニ,
  タンザニア, トーゴ, チュニジア, ウガンダ, 西サハラ, ザンビア, ジンバブエ}

```

各国の実体には、その国の地理データ、人口やGDPなどの社会・経済データ、天気の日データや有名人のデータまで多くのデータが紐づいています。ここでは、アフリカ大陸の各国に関する{“Entity”（実体）, “Name”（名前表記）, “Population”（人口）, “GDP”, “FlagImage”（国旗）}のデータを取り出して、いろいろな形でデータを可視化してみましょう。

データを変数に代入する




EntityValue関数を使って、アフリカの国々のデータを変数africadataに代入します。（ここでは紙面の関係上、プログラムの最後にセミコロン(;)をつけて、出力を表示しないようにしています）

```
In[ ]:= africadata = EntityValue[ Africa GEOGRAPHIC REGION ["Countries"],
      EntityValue["Entity", "Name", "Population", "GDP", "FlagImage"]];
```

最初の3つのデータを見てみましょう。africadata[[1;;3]]は、「africadataのリストの1番目から3番目までの値を取り出さない」というプログラムです。

```
In[ ]:= africadata[[1 ;; 3]]
```

Out[]:=

```
{ { { アルジェリア , アルジェリア ,
      45 600 000 people , $2.47626 × 1011 per year ,  } ,
  { アンゴラ , アンゴラ , 36 700 000 people , $8.48247 × 1010 per year ,  } ,
  { ベナン , ベナン , 13 700 000 people , $1.9676 × 1010 per year ,  } }
```

最初の3つの国であるアルジェリア、アンゴラ、ベナンの{“Entity”（実体）, “Name”（名前表記）, “Population”（人口）, “GDP”, “FlagImage”（国旗）}の各値が出力されていますね。

それでは、このデータリストafricadataを使って、いろいろな可視化をしていきましょう。

地図上に可視化する

EntityとPopulationのデータを使って、地図上に人口のデータを色分けして描画します。africadataには、1列目にEntity、3列目にPopulationが入っています。

africadata[[All,{1,3}]]は「africadataのリストから、全て(All)の行の、1列目 (Entity)と3列目 (Population) を取り出さない」というプログラムです。ここでは、プログラムの最後に //Short をつけて、表示するデータ数を少なくしています。

```
In[ ]:= africadata[[All, {1, 3}]] // Short
          [すべて]          [省略]
```

Out[]:= Short=

```
{ { アルジェリア , 45 600 000 people }, { アンゴラ , 36 700 000 people },
  { ベナン , 13 700 000 people }, { ボツワナ , 2 700 000 people },
  { ブルキナファソ , 23 300 000 people }, { ブルンジ , 13 200 000 people },
  { カメルーン , 28 600 000 people }, { カーボベルデ , 600 000 people },
  { 中央アフリカ共和国 , 5 700 000 people }, { チャド , 18 300 000 people },
  { コモロ , 900 000 people }, { コンゴ民主共和国 , 102 300 000 people },
  { ジブチ , 1 100 000 people }, { エジプト , 112 700 000 people },
  { 赤道ギニア , 1 700 000 people }, { エリトリア , 3 700 000 people },
  { エチオピア , 126 500 000 people }, <<24>>, { セントヘレナ , 5346 people },
  { サントメ・プリンシペ , 200 000 people }, { セネガル , 17 800 000 people },
  { セーシェル , 100 000 people }, { シエラレオネ , 8 800 000 people },
  { ソマリア , 18 100 000 people }, { 南アフリカ共和国 , 60 400 000 people },
  { 南スーダン , 11 100 000 people }, { スーダン , 48 100 000 people },
  { エスワティニ , 1 205 887 people }, { タンザニア , 67 400 000 people },
  { トーゴ , 9 100 000 people }, { チュニジア , 12 500 000 people },
  { ウガンダ , 48 600 000 people }, { 西サハラ , 600 000 people },
  { ザンビア , 20 600 000 people }, { ジンバブエ , 16 700 000 people } }
```

このデータリストをGeoRegionValuePlot関数の引数に指定して実行すると、人口の数によって色を塗り分けたアフリカ大陸の地図が描画されます。色が濃いほど、人口が多いことがわかります。

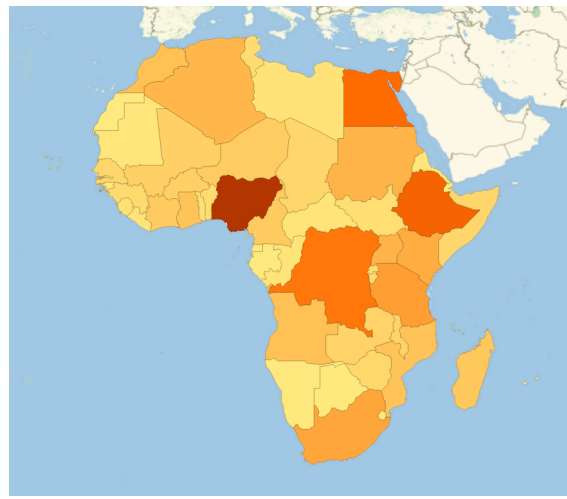
このように、Entity には最初にafricadataとして取り出したデータ以外に地理情報なども紐づいているので、地理データを可視化するGeoRegionValuePlot関数の引数に国名のEntityを指定するだけで、その国の地図上の位置や形のデータを使って可視化できます。

```
In[ ]:= GeoRegionValuePlot[africadata[All, {1, 3}]]
```

測地域域の値プロット

[すべて]

Out[]:=



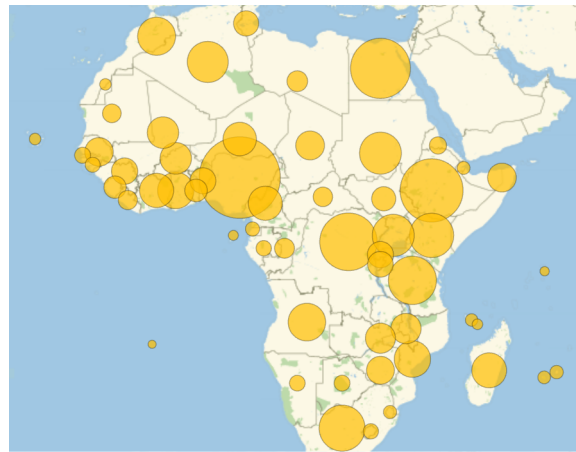
今度は、GeoBubbleChart関数を使って、同じデータリストを描画してみましょう。データの値が大きいほど、円（バブル）が大きく描画されます。この結果から、「円（バブル）が大きい国ほど人口が多い」ということがわかります。

```
In[ ]:= GeoBubbleChart[africadata[All, {1, 3}]]
```

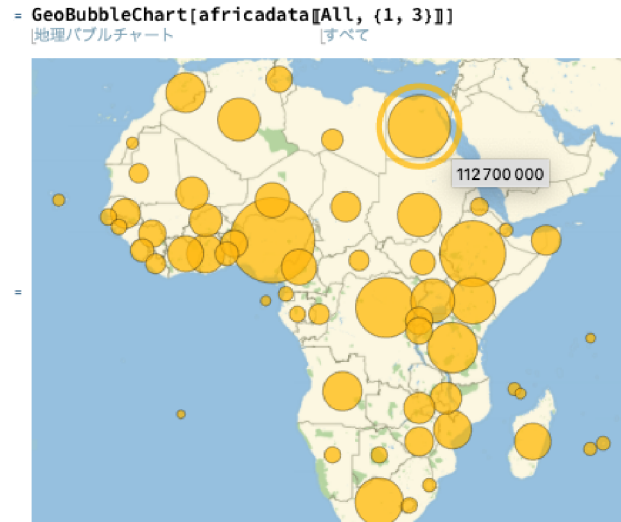
地理バブルチャート

[すべて]

Out[]:=



バブルにマウスを近づけると、そのバブルのデータ（ここでは人口）がポップアップ表示されます。（図P6-2参照）



図P6-2 バブルにマウスを近づけると、データ（人口）がポップアップ表示される

ワードクラウドで可視化する

今度は、人口が多いほど国名が大きく表示されるワードクラウドで可視化してみましょう。第6章ではWordCloud関数を使って、文章中のワードの数が多いほど大きく表示されるワードクラウドを作成しましたが、ここでは、WordCloud関数に重み付けをして、人口が多いほど国名が大きく表示されるワードクラウドを作成します。

以下のように、africadata[[1, {2,3}]]は「africadataのリストから、1行目の、2列目（Name）と3列目（Population）を取り出さない」というプログラムになり、africadataの1行目に入っているアルジェリアの国名と人口のデータのリストが得られます。

```
In[ ]:= africadata[[1, {2, 3}]]
```

```
Out[ ]:=
```

```
{アルジェリア, 45 600 000 people }
```

このプログラムをアレンジして、africadataに入っているデータの全ての国に対して同じように国名と人口のリストを得るには、africadata[[All,{2,3}]]とします。ここでのAllが「全ての行」という意味になります。ここでは、プログラムの最後に //Short をつけて、表示するデータ数を少なくして見てみましょう。

```
In[ ]:= africadata[All, {2, 3}] // Short
```

[すべて] [省略]

```
Out[ ]:= Short=
```

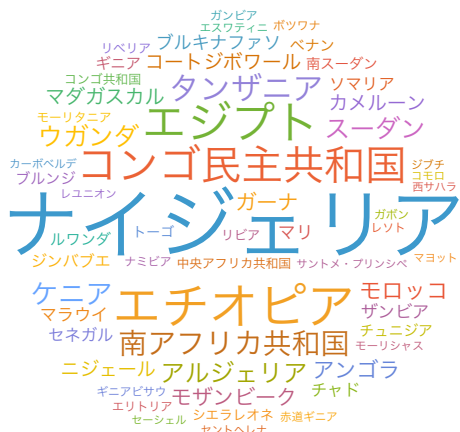
```
{ {アルジェリア, 45 600 000 people },
  {アンゴラ, 36 700 000 people }, {ベナン, 13 700 000 people },
  {ボツワナ, 2 700 000 people }, {ブルキナファソ, 23 300 000 people },
  {ブルンジ, 13 200 000 people }, <<47>>, {チュニジア, 12 500 000 people },
  {ウガンダ, 48 600 000 people }, {西サハラ, 600 000 people },
  {ザンビア, 20 600 000 people }, {ジンバブエ, 16 700 000 people } }
```

WordCloud関数に与える引数をafricadata[[All,{2,3}]]とすると、人口が多いほど国名が大きく描画されるワードクラウドが作成できます。

```
In[ ]:= WordCloud[africadata[All, {2, 3}]]
```

[ワードクラウドの生成] [すべて]

```
Out[ ]:=
```



このワードクラウドでは、ナイジェリアの人口が多いことがわかりました。

練習問題：
第6章の演習6-4を参考にして、ワードクラウドの色や形を変えて描画してみましょう。

国旗画像のコラージュで可視化する

今度はちょっと面白い可視化です。国旗の画像を使って、人口に比例した大きさの国旗を組み合わせ、ひとつの絵にしてみます。複数の画像やテキストを合成して、ひとつの絵に仕上げる手法を「コラージュ」と言います。

以下の[africadata\[\[1,{5,3}\]\]](#)では、africadataの1番目のデータの5列目（FlagImage）と3列目（Population）のリストが出力されます。1番目のアルジェリアの国旗と人口が出力されていますね。

```
In[ ]:= africadata[[1, {5, 3}]]
```

```
Out[ ]:=
```

```
{ , 45 600 000 people }
```

では、これを全ての行（国）について求めるにはどうしたらよいでしょうか？先ほどのワーククラウドのデータを作る時と同じように、[africadata\[\[All,{5,3}\]\]](#)とすると、全ての行（このデータでは国）について国旗と人口のデータのリストが得られますね。これをImageCollage関数に入れてコラージュを作りましょう。人口が多いほど国旗が大きく表示されています。

```
In[ ]:= ImageCollage[africadata[[All, {5, 3}]]]
          |画像のコラージュ          |すべて
```

```
Out[ ]:=
```



練習問題：ここまでの例を参考に、GDPのデータを可視化してみましょう。GDPのデータは、africadataの何列目に入っているかを考えて、可視化するためのリストを作成しましょう。

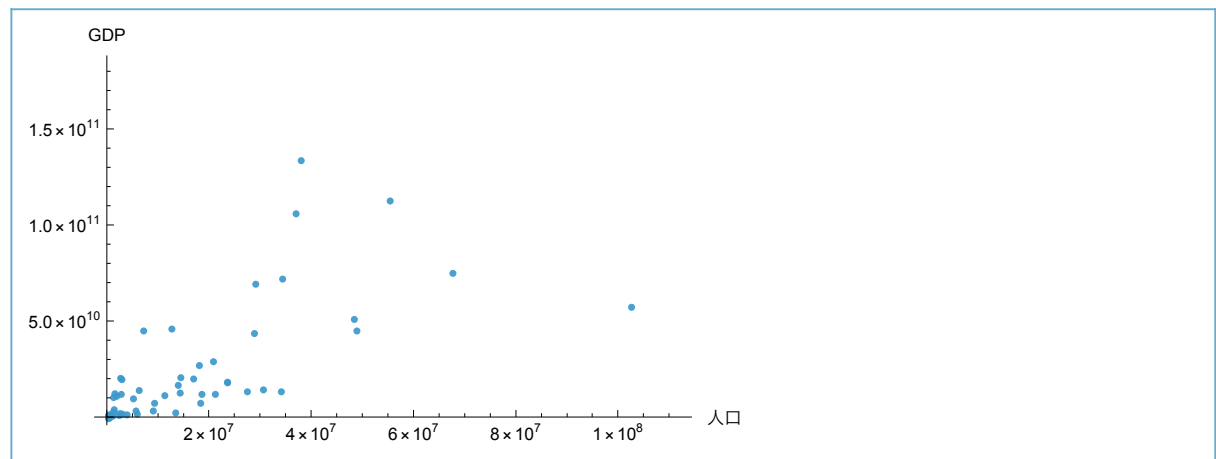
人口とGDPの関係を散布図で可視化する

次は、人口とGDPの関係を散布図を描いて見てみましょう。africadata[[All,{3,4}]]で、全て(All)の国の3列目(Population)と4列目(GDP)のリストが得られます。これをListPlot関数の引数に入れて、散布図を描きます。グラフの点のところにマウスを近づけると、その点の値(座標)が表示されます。

```
In[ ]:= ListPlot[africadata[[All, {3, 4}]], AxesLabel -> {"人口", "GDP"}]
```

リストプロット すべて 軸のラベル

Out[]:=



応用：ラベル付きの散布図を描いてみよう

ただ、このグラフでは、どの国のデータがどの点なのか、わかりません。散布図に国名を表示できたらわかりやすくなりますね。ここからはちょっとプログラムが複雑になりますので、「こんな可視化もできるんだな」と見るだけでも構いません。もちろん、チャレンジしたい人はぜひやってみてください。

点にラベルを表記する方法はいくつかありますが、ここではCallout関数を使います。Callout[{x,y}, label]という形で、座標{x,y}の点にlabelをつけます。ここでは、全ての国についてCallout[{人口, GDP}, 国名]を作成したいので、Map関数を使ってプログラムを作ります。Map関数は、第1引数の#の部分に、第2引数で指定したリストの各要素が入ります。(#と&がセットになっていて、これを「純関数」といいます) 少し難しいですが、Map関数は写像といって、リストの要素の数や形式に関わらずリストの各要素に同じ操作を指定できる便利な関数です。詳しくはWolframのドキュメント(*2)で調べてみてください。

Map関数を使って、africadataの全行(リスト)に対して、Callout[{x,y}, label]という形を作っていきます。Map[Callout[#[[{3, 4}]], #[[2]]] &, africadata]は、africadataの各国の3, 4列目の値と、2列目の値をリストにし、Callout関数の引数となります。ここでは//Shortをつけて一部のみ表示してみます。出力結果はCallout[{人口, GDP}, 国名]となっていますね。

```
In[ ]:= Map[Callout[#[{3, 4}], #[2]] &, africadata] // Short
```

適用 コールアウト 省略

```
Out[ ]:= Short=
```

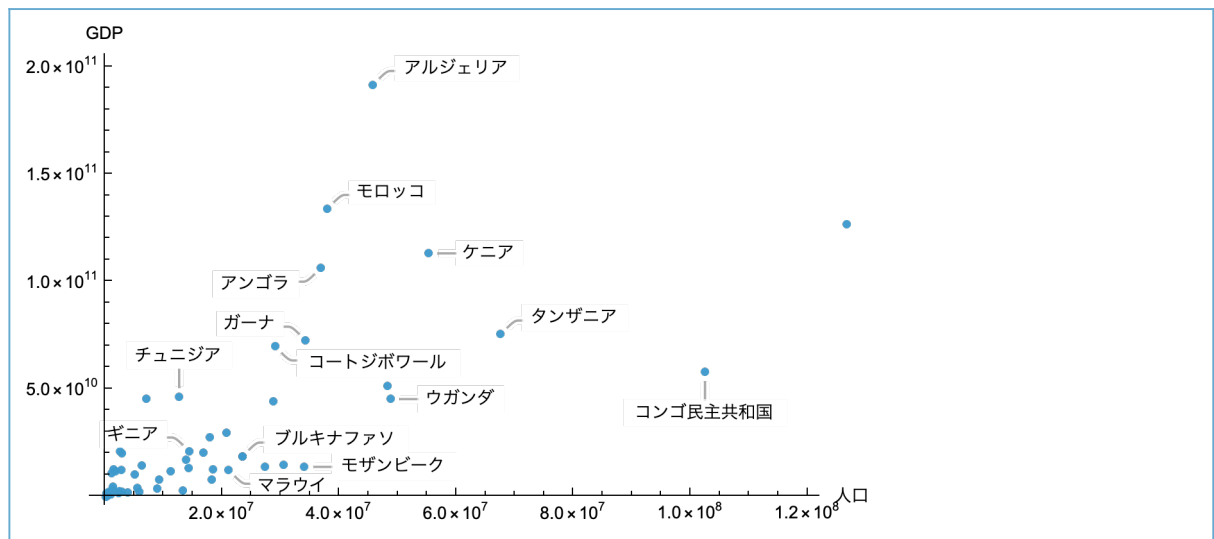
```
{Callout[{45 600 000 people, $1.91913 × 1011 per year}, アルジェリア],
 Callout[{36 700 000 people, $1.06714 × 1011 per year}, アンゴラ],
 Callout[{13 700 000 people, $1.74017 × 1010 per year}, ベナン],
 <<52>>, Callout[{600 000 people, Missing[NotAvailable]}, 西サハラ],
 Callout[{20 600 000 people, $2.97845 × 1010 per year}, ザンビア],
 Callout[{16 700 000 people, $2.06781 × 1010 per year}, ジンバブエ]}
```

これを使って、ListPlotで描いてみましょう。どの国がどの点なのか、わかりやすくなりましたね。人口とGDPの関係から、アフリカの国の特徴を考えてみましょう。

```
In[ ]:= ListPlot[Map[Callout[#[{3, 4}], #[2]] &, africadata],
 AxesLabel → {"人口", "GDP"}]
```

リスト… 適用 コールアウト 軸のラベル

```
Out[ ]:=
```



(*2) Wolframドキュメントについては、本書のコラムで紹介しています。