

この教材は『手を動かしながらやさしく学べるはじめてのAIデータサイエンスリテラシー』（技術評論社）の実践編として本書のv～vii ページで紹介している発展的な実践教材です。本書で学んだことを活かして、発展的で面白いプロジェクトに挑戦できます。この教材はPDFで提供しています。必要に応じて印刷し、Wolframノートブックに自分でコードを入力して実行してください。（Wolframノートブックでの配布はしていません。）

なお、ここでは、本書独自のフォーマットを使い、プログラムの入力部分にオレンジの枠、出力部分にブルーの枠をつけてプログラムの部分をわかりやすく表示しています。新規でノートブックを開いた時にはこれらの色枠は付きません。

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、天気など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）

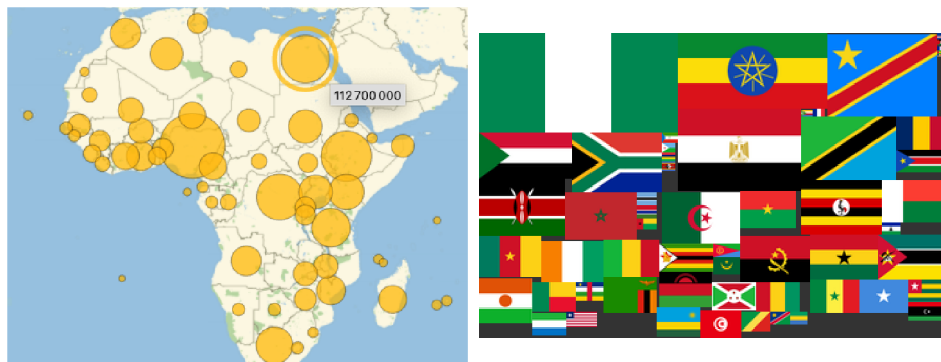

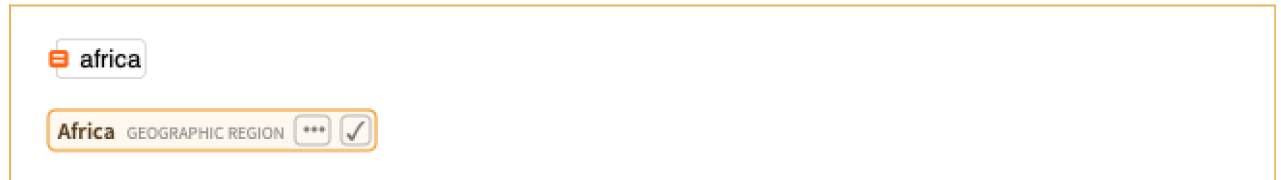


図1 アフリカ大陸の国のデータの可視化の例

（*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[1]:= Africa GEOGRAPHIC REGION ["Countries"]
```



```
Out[1]= {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、天気の詳細データや有名人の詳細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いきます。




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[2]:= africadata = EntityValue[ Africa GEOGRAPHIC REGION ["Countries"],  
                                実体値  
                                {"Entity", "Name", "Population", "GDP", "FlagImage"}];  
                                実体
```

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

```
Out[3]= {{アルジェリア, アルジェリア, 46 164 219 people ,  
            
          $2.47626 \times 10^{11} \text{ per year } ,  
          }, {アンゴラ, アンゴラ,  
            
          36 749 906 people , $8.48247 \times 10^{10} \text{ per year } ,  
          }, {ベナン, ベナン, 14 111 034 people , $1.9676 \times 10^{10} \text{ 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[4]:= africadata[[All, {1, 3}]] // Short
```

すべて

省略

Out[4]//Short=

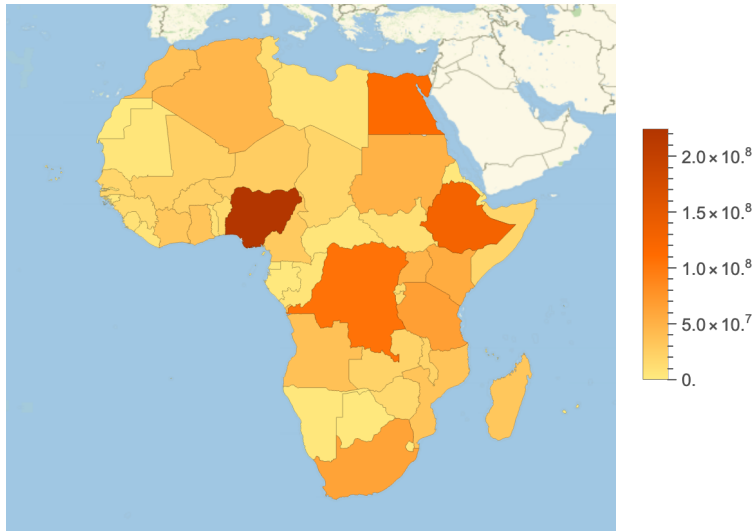
```
{ { アルジェリア , 46 164 219 people } , { アンゴラ , 36 749 906 people } ,
  { ベナン , 14 111 034 people } , { ボツワナ , 2 480 244 people } ,
  { ブルキナファソ , 23 025 776 people } , { ブルンジ , 13 689 450 people } ,
  { カメルーン , 28 372 687 people } , { カーボベルデ , 600 000 people } ,
  { 中央アフリカ共和国 , 5 152 421 people } , { チャド , 19 319 064 people } ,
  { コモロ , 850 387 people } , { コンゴ民主共和国 , 105 789 731 people } ,
  { ジブチ , 1 152 944 people } , { エジプト , 114 535 772 people } ,
  { 赤道ギニア , 1 700 000 people } , { エリトリア , 3 470 390 people } ,
  { エチオピア , 128 691 692 people } , { ガボン , 2 484 789 people } ,
  { ガンビア , 2 800 000 people } , { ガーナ , 34 100 000 people } ,
  { ギニア , 14 200 000 people } , { ギニアビサウ , 2 153 339 people } ,
  <<15>> , { ナイジェリア , 223 800 000 people } ,
  { コンゴ共和国 , 6 182 885 people } , { レユニオン , 874 883 people } ,
  { ルワンダ , 14 100 000 people } , { セントヘレナ , 5289 people } ,
  { サントメ・プリンシペ , 230 871 people } , { セネガル , 18 077 573 people } ,
  { セーシェル , 127 951 people } , { シエラレオネ , 8 460 512 people } ,
  { ソマリア , 18 358 615 people } , { 南アフリカ共和国 , 63 212 384 people } ,
  { 南スーダン , 11 483 374 people } , { スーダン , 50 042 791 people } ,
  { エスワティニ , 1 230 506 people } , { タンザニア , 66 617 606 people } ,
  { トーゴ , 9 304 337 people } , { チュニジア , 12 200 431 people } ,
  { ウガンダ , 48 656 601 people } , { 西サハラ , 600 000 people } ,
  { ザンビア , 20 723 965 people } , { ジンバブエ , 16 340 822 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。

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

測地地域の値プロット [すべて]

Out[5]=



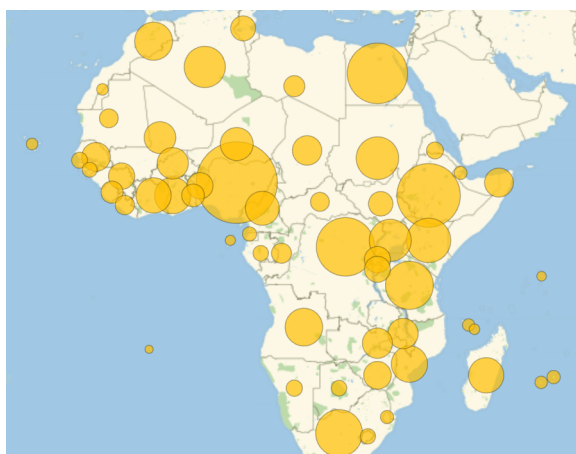
このように、**Entity** には最初に**africadata**として取り出したデータ以外に地理情報なども紐づいているので、地理データを可視化する**GeoRegionValuePlot**関数の引数に国名の**Entity**を指定するだけで、その国の地図上の位置や形のデータを使って可視化できます。

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。**GeoBubbleChart**関数では、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[6]:= GeoBubbleChart[africadata[All, {1, 3}]]
```

地理バブルチャート [すべて]

Out[6]=



バブルにマウスを近づけると、図2のように、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。

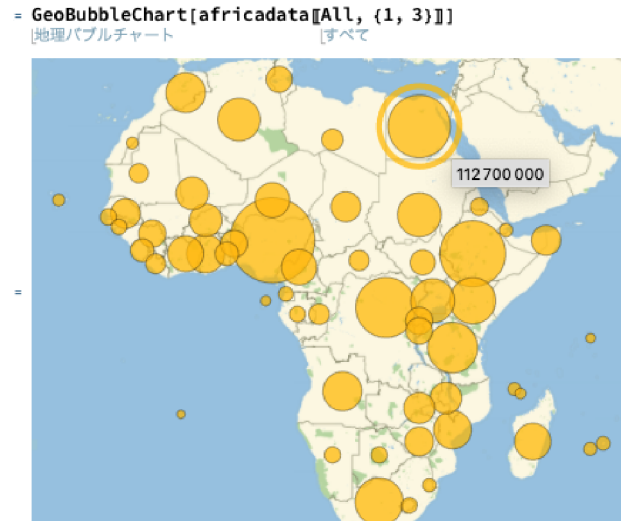


図2 バブルにマウスを近づけると、データ（人口）がポップアップ表示される

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[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**）を取り出しなさい」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[7]:= africadata[[1, {2, 3}]]
```

```
Out[7]= {アルジェリア, 46 164 219 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[8]:= africadata[All, {2, 3}] // Short
```

すべて

省略

Out[8]//Short=

```
{ {アルジェリア, 46 164 219 people}, {アンゴラ, 36 749 906 people},
  {ベナン, 14 111 034 people}, {ボツワナ, 2 480 244 people},
  {ブルキナファソ, 23 025 776 people}, {ブルンジ, 13 689 450 people},
  {カメルーン, 28 372 687 people}, {カーボベルデ, 600 000 people}, <<42>>,
  {エスワティニ, 1 230 506 people}, {タンザニア, 66 617 606 people},
  {トーゴ, 9 304 337 people}, {チュニジア, 12 200 431 people},
  {ウガンダ, 48 656 601 people}, {西サハラ, 600 000 people},
  {ザンビア, 20 723 965 people}, {ジンバブエ, 16 340 822 people} }
```

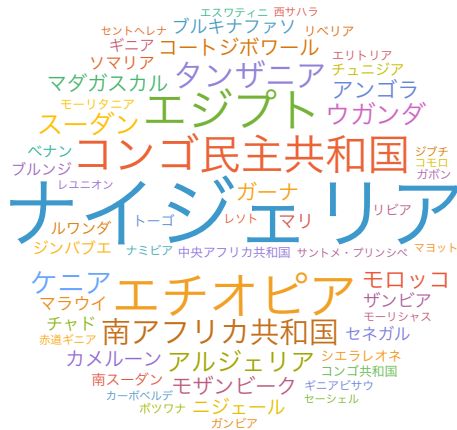
以下のように、**WordCloud**関数に与える引数を`africadata[[All,{2,3}]]`とすると、人口が多いほど国名が大きく描画されるワードクラウドが作成できます。

```
In[9]:= WordCloud[africadata[All, {2, 3}]]
```

ワードクラウドの生成

すべて

Out[9]=



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一目瞭然でわかりますね。

練習問題(1)：
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[10]:= africadata[[1, {5, 3}]]
```

```
Out[10]=
```



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？先ほどのワード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[11]:= ImageCollage[africadata[[All, {5, 3}]]]
          |画像のコラージュ          |すべて
```

```
Out[11]=
```



練習問題(2)：

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

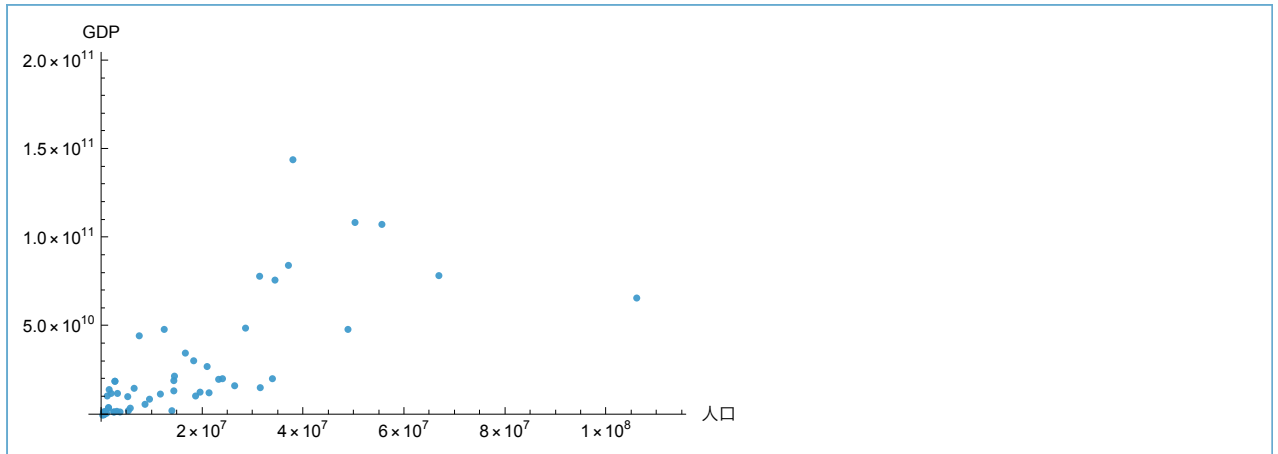
次は、人口とGDPの関係を散布図を描いて見てみましょう。

`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[12]:= 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[12]=



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。詳しくは本書のコラム (28ページ) で紹介しているWolframのドキュメント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。

```
In[13]:= Map[Callout[#[[3, 4]], #[[2]] &, africadata] // Short
適用 コールアウト 省略
```

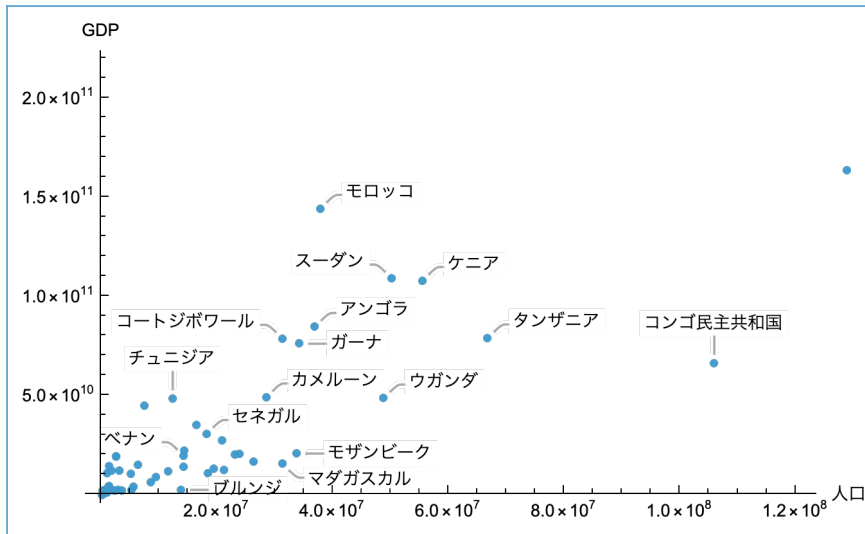
Out[13]//Short=

```
{Callout[{46 164 219 people, $2.47626 × 1011 per year}, アルジェリア],
 Callout[{36 749 906 people, $8.48247 × 1010 per year}, アンゴラ],
 Callout[{14 111 034 people, $1.9676 × 1010 per year}, ベナン],
 Callout[{2 480 244 people, $1.93961 × 1010 per year}, ボツワナ],
 <<51>>, Callout[{600 000 people, Missing[NotAvailable]}, 西サハラ],
 Callout[{20 723 965 people, $2.7578 × 1010 per year}, ザンビア],
 Callout[{16 340 822 people, $3.52314 × 1010 per year}, ジンバブエ]}
```

出力結果は`Callout[{人口, GDP}, 国名]`の形になっています。これを使って、`ListPlot`関数で描いてみましょう。どの国がどこの点なのか、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[14]:= 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[14]=



練習問題(3)：

ここまでの例を参考にして、ほかのいくつかの国についてWolfram知識ベースからデータを取得し、可視化してみましょう。また、Wolfram知識ベースには、国の情報以外にも経済や文化などさまざまなデータがあります。どんなデータがあるか、どんな分析ができるかをWolframオンラインドキュメントで調べてみましょう。