言語地図作成プログラム**MAPAS.xlsm**

　2012年7月にウィーンの「国際方言学言語地理学会」でスペイン・カタルーニャ語の動詞形態論の地域差バリエーションについて研究を発表したのは「複雑な方言差を多変量解析する方法を開発してほしい」というバルセロナ大学のカタルーニャ語学教授マリア・ピラル・ペレアさんの依頼によるものでした。共同研究発表では地点と語形から作成したクロス表を多変量の統計係数によって並べ替え全体の分布パタンの傾向をつかむ、という趣旨のことを述べました。発表後、会場の参加者から「多変量解析によって得られた地点の係数を地図上に数量記号化して示してはどうか」という示唆をいただきました。

　私も数量記号地図化の必要性をいつも感じていたのですが、いろいろな既存の統計パッケージを使うのではなく、いつか独自の地図作成システムを作ってみたいと思いながら後回しにしてきました。独自のプログラムにこだわる理由は、プログラムの仕様を自分が必要としていることに合致させることができること、出力を自分で説明ができること、プログラムを改訂しながら発展させることができること、などです。学会から帰国して2週間ほど試行錯誤しながらMAPAS（スペイン語で「地図」複数形）と名づけるこのシステムを一応完成させたので、その悪戦苦闘した記録をここに残しておきたいと思います。

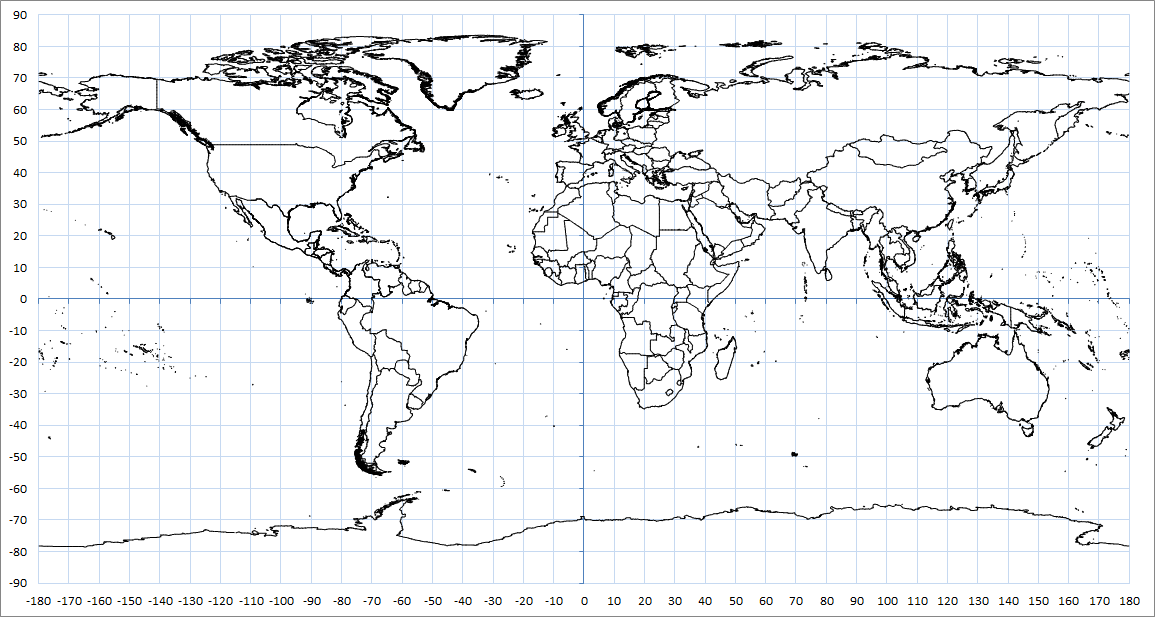
　プログラムは完成すればそれで全作業が終了するわけではなく、かならずドキュメント（記録文書）を残しておかなければなりません。そうでないと修理・改善・変更などの保守作業が製作者でさえできなくなるからです。ましてや他人が見てもほとんど解読不能のコードになるでしょう。一方、しっかりとした文書があれば、本人の保守作業は容易になるし、他人はそれを読み込むことでコードの使い方の良い面や悪い面が参考になります。その意味でも以下の文章はプログラミングが一応完成した直後の１週間に急いで書き上げたものです。スピードにこだわったのは、私の記憶が薄らいでいくのが怖かったことと、後回しにしておけば面倒になってモチベーションが落ちることがわかっているからです。そして、説明文を書くプロセスの中で、新しいアイデアがわいて、プログラムを改訂することもできました。文章は後で推敲すればよいでしょう。形式にもこだわりません。とにかく今記憶していることをどんどん書き進めていきたいと思います。

　そして記録を残しておくことには別の理由があります。大学では言語情報科学専攻で言語処理プログラミングの授業を担当していますが、言語科学を専攻する文系の学生たちは必ずしもプログラミングの数理の導出やアルゴリズムを理解することを目指しているわけではなく、その思考法に慣れていないため、教室やコンピュータ室で説明をしながら途方に暮れることがあります。はじめからできあがった数理やプログラムを説明すると、その意図や制作過程がわからないと思います。そこで、このMAPASのシステムがどのように制作されたのかを、製作者（私）の記憶が鮮明なうちに、その過程をたどったドキュメントを残しておけば、きっと言語処理プログラミングの履修者たちの参考になるのではないだろうかと考えました。

　この文書を読んでいただいて、わかりにくい部分や訂正すべき部分が見つかったときご連絡をいただければ幸いです。新しい提案もお待ちいたします。よろしくお願いいたします。

　＊MAPASは言語地図を作成するためのシステムですが、言語地図だけでなく、地名、文字列、記号、数量を地図上にプロットする目的であるならば他にもいろいろな用途に使えます。また白地図を削除すれば一般の散布図も出力することができます。

**MAPASの第1シート**



　白地図の上に地名、語形、記号、数量化記号を簡単にプロットするためのシステムです。「表紙」に続いて「白地図」「地名・語形地図」「記号地図」「数量地図」のシートがあります。

　白地図の経度と緯度のデータは自由な使用許可のある以下のサイトからダウンロードし、必要な精度に調整しました。

Longitudes and latitudes revised from World Vector Shoreline data set available through the U.S. National Geophysical Data Center, NESDIS, NOAA, NGDC GEODAS Coastline Extractor, 2012.

http://www.ngdc.noaa.gov/mgg/shorelines/shorelines.html

　詳細は次のホームページをご覧ください。

http://lecture.ecc.u-tokyo.ac.jp/~cueda/gengo/index.html

　このプログラムの使用と改変は自由です。お気づきの点がありましたらお知らせください。使用して得られた成果をお送りいただければ幸いです。

上田博人© 2012 Hiroto Ueda

uedahiroto@jcom.home.ne.jp

# 座標データの収集

　前期課程の「言語データ分析」の授業では簡単な統計学と情報処理の基礎を説明しています。統計学で扱う「相関係数」だけを提示されてもどのように評価してよいかわからないことが多いので、履修者には相関係数だけでなく散布図も用意するように勧めています。散布図を見ると実際にはあまり相関が見えないのに相関係数が高い数値を示していたり、逆に、よく観察すれば相関があるのに相関係数が低くなっていることがわかるケースがあるからです。

　散布図はExcelのグラフを使えば簡単に作ることができます。手作業で散布図を作るのは大変ですが、範囲を選択して一定の操作をすると瞬時に散布図のグラフを描きます。この機能を使って白地図を描き、その白地図に図形や文字列を書き込むプログラムを作成してみたいと思います。とくに新しいソフトウェアを購入しなくても、ふだん使っているWordやExcelを活用すれば、ある程度実用性のあるプログラムが作れます。

　自分でプログラムを作成するといろいろな点で便利です。まず、経費がかかりません。そして、プログラムの仕様が自分の思い通りになり、自由に改訂ができます。ほかのユーザーからの意見を聞くのも楽しみです。そこで、この自作システムをMAPASと名付けてウェブで発表することにしました。「地図」（複数形）という意味のスペイン語です。

　MAPASプログラミング計画は最初に経度と緯度による海岸線と国境線の座標データを入手し、これを使って白地図を描き、その白地図に、地名、語形、分類記号、数量記号をプロットする、という手順にしました。この全手順を小さな規模のデータでシンプルなプログラムの全体を作ってみるとMAPASは比較的小規模なデータでは実現可能でした。このMAPASが巨大なデータでもちゃんと動作するのか確かめてみたいと思います[[1]](#footnote-1)。

　第一日目の仕事は座標データの収集・加工です。インターネットで検索するとアメリカ合衆国の公的サイト（後述するNOAA）に使用自由のテキストデータがあることがわかりました。早速イベリア半島（スペインとポルトガル）の範囲をダウンロードしデータを検証すると、それが非常に精密なものであり、十分に私たちの目的にかなうものでした。

　当初はイベリア半島だけの言語地図を考えていたのですが、このサイトのデータの提供方法が緯度と経度による範囲指定であることを見て、MAPASでもこの方式を使って、イベリア半島だけでなく全世界の一定の地域の経度・緯度の開始点と終了点を指定すれば「白地図」が描けるようにしようと計画しました。NOAAでもウェブ上に該当部分の白地図を示しています。

　全世界の海岸・国境線の座標データは膨大な量になりました。この規模のデータはExcelでは扱うことができないので、どうにか処理可能な量になるように、実際上必要な精度にまで落とし最適なデータを作成しました。はじめに、きれいな地図が出力されるという条件を保ちながら、どこまで座標データの小数点以下の桁数を下げることができるかを確かめる実験をしました。小数点以下の桁数を下げると同一データが連続するようになります。これらの重複するデータは無駄ですから1つだけにまとめました。

　次に、データ量を間引いて減らし、全体の行数を、どこまで落とせるのかを試す実験をしました。このとき、はじめは機械的に入力データを一定の数だけ飛ばす、という方法をやってみたのですが、この方法よりも数字が類似するデータが並ぶ場合は1つにまとめる、という方法を考えました。この間引き方のほうが丁寧なので結果も満足できるものでした。

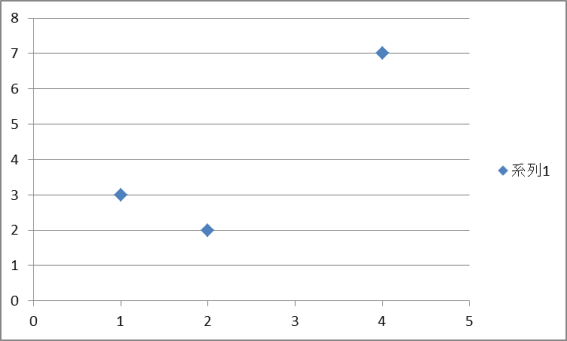
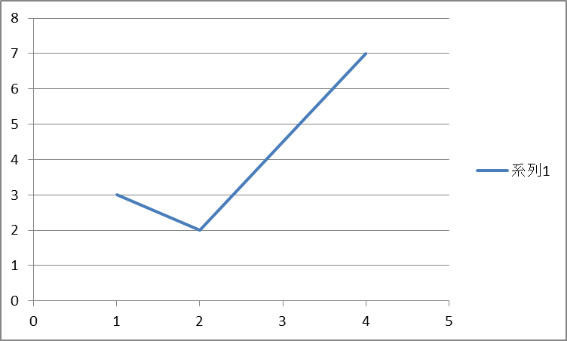
　このような実験をするときの手順について、順次検査法と二倍二分検索法を比較しながら考察しました。学習の一般的な類型についても考えてみました。以下では座標データの入手法と加工法を説明します[[2]](#footnote-2)。

## エクセルの散布図

　Excelシートに2列の数値データがあれば、それを座標とした散布図を簡単に描くことができます。たとえば、

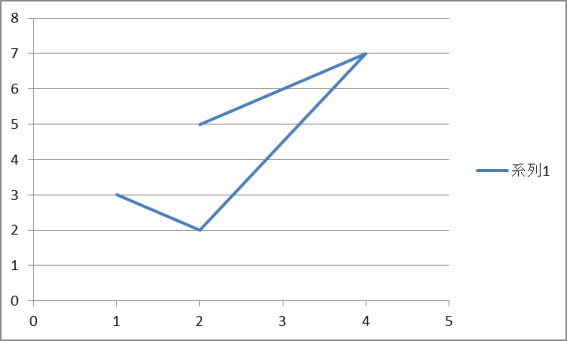
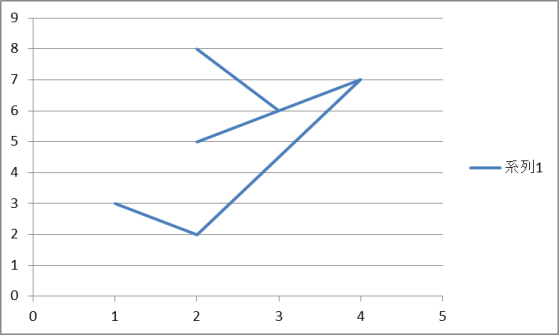
|  |  |
| --- | --- |
| 1 | 3 |
| 2 | 2 |
| 4 | 7 |

というデータ行列を選択し、「挿入」→「グラフ」→「散布図」という操作をすれば下左図の散布図が描かれます。これを「散布図（直線）」にすると下右図のようになります。

　実は後者の「散布図（直線）」は散布図としてだけではなく描画のために使うことが可能です。たとえば先のデータの後に3, 5を加えてグラフ化してみましょう（下左図）。さらに空白を挟んだデータでは別の線を書き加えることもできます（下右図）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 |  | 1 | 3 |
| 2 | 2 |  | 2 | 2 |
| 4 | 7 |  | 4 | 7 |
| 2 | 5 |  | 2 | 5 |
|  |  |  |  |  |
|  |  |  | 2 | 8 |
|  |  |  | 3 | 6 |

　海岸線と国境線を示す座標の連続データが入手できれば、それを使って白地図を描くことが可能なはずです。現在のExcelは104万行まで処理が可能です。実験をしてみると確かに100万行のデータであってもちゃんと散布図を描くことがわかりました（後述）。

## データを入手する

　インターネットで検索するとアメリカ合衆国政府のNOAA (National Oceanic and Atmospheric Administration)の次のサイトが見つかりました。

http://www.noaa.gov/

の中の次のサイトです。これは使用自由のpublic domainです。

http://www.ngdc.noaa.gov/mgg/coast/

　はじめに全世界の座標データを入手しましょう。そのためには経度(longitude)の範囲を-180から180までとし、緯度の範囲を90から-90とします。これで西経180度から東経180度まで、そして北緯90度から南緯90までの経度と緯度の座標が得られます[[3]](#footnote-3)。次は座標の冒頭の部分です。

>

-80.974613 51.999999

-80.973439 51.999706

-80.972852 51.998826

(…)

　小数点以下6桁の経度と緯度の行列が全体で1000万以上の行数になっています。線の区切りには「>」の記号が使われています。それに続いて2つの数字がタブコードに分かれて行を作っています。このようにタブコードで分かれているデータはそのままExcelにコピーできるのですが、このように大規模なデータをそのままExcelにインポートすることはできません。Excelで扱える行数は104万という限度があるからです。また、MAPASのシステムで小数点以下6桁という高い精度が必要であるか、試してみなければならないでしょう。そこでデータ縮約の方針として次の2つを考えました。

(1) 精度を落としていくために、エディターを使って小数点以下を6, 5, 4 …桁というデータに変形し地図出力の精度を確かめる。

(2) データを間引くためにVBAのプログラムを作って連続データ間の差が小さいデータを読み飛ばす。

　大きなデータを扱うとき、はじめは小さなサンプルデータで試行錯誤を繰り返しながら一通りの手順の実験をしておき、一応方法が見つかったら、今度は本番のデータを扱うとよいでしょう。はじめから、フルに道具や手順を用意してから実験をすると、実験なのに時間や手間が多くかかってしまい無駄になることが多いのです。

　たとえば、イベリア半島のおおよその範囲は次の通りです。

|  |  |  |
| --- | --- | --- |
| イベリア半島 | 経度 | 緯度 |
| 始点 | -10 | 35 |
| 終点 | 5 | 45 |

　この範囲で出力したデータの冒頭部分は次のようになっています。

|  |  |
| --- | --- |
| > |  |
| -6.25109 | 35.00011 |
| > |  |
| -2.13474 | 35.00011 |
| -2.13474 | 35.00011 |
| > |  |
| -5.99998 | 35.58623 |
| -6.00056 | 35.58594 |
| -6.00056 | 35.58506 |
| -6.00291 | 35.58242 |
| -6.00291 | 35.58154 |
| -6.00379 | 35.58095 |
| -6.00379 | 35.57655 |
| -6.00203 | 35.57509 |
| -6.00203 | 35.57421 |
| -6.00174 | 35.57362 |
| -5.99998 | 35.57362 |
| > |  |

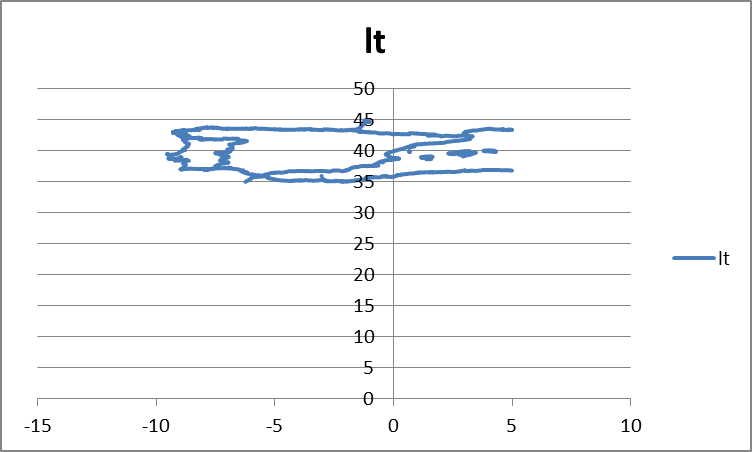
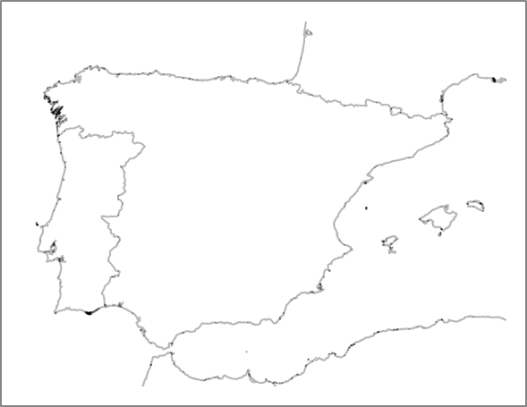
## マクロの記録

　このデータのかたまり、つまり1つの線を示すデータの切れ目に「>」の記号が使われています。先に見たようにExcelで散布図を描くにはこれを空白にしなければいけないので、検索・置換の機能を使って空白に置換します。セルB1に仮の文字を記入しておくと、これが散布図のタイトルと凡例になります。

|  |  |
| --- | --- |
|  | lt |
| -6.25 | 35 |
|  |  |
| -2.13 | 35 |
|  |  |
| -5.99 | 35.58 |
| -6 | 35.58 |

(…)

　これをExcelシートのA:B列にコピーしてから散布図（平滑線）を描くと下左図のような「散布図」ができます[[4]](#footnote-4)。かろうじてイベリア半島であることがわかりますが、かなり上下につぶれているようです。これは縦軸の目盛の最小値が0であることが原因です。凡例を削除し、縦軸と横軸の目盛の最小値と最大値を緯度の最小値(35)と最大値(45)に修正し、他にも修正点を加えると下右図のようになります[[5]](#footnote-5)。これで私たちがよく見るイベリア半島の姿に近づきました。

　上左図から上右図に修正するためにさまざまな操作を行いました。この操作はこれから何度も繰り返すのでマクロに記録しておくと便利です。散布図修正のための一連の操作は次のとおりです。散布図にリンクするデータ数が多いので操作の反応は少し時間がかかります。ゆっくりと進めてください。

(1) 縦の「軸の書式」から最小値、最大値を設定する[[6]](#footnote-6)。

(2) 横の「軸の書式」から最小値、最大値を設定する。

(3) タイトル、凡例、縦軸、横軸、目盛線を選択し削除する。

(4) 線の色を黒に、透明度を50%に、太さを1ptにする[[7]](#footnote-7)。

(5) 全体の横幅を狭くする[[8]](#footnote-8)。

　一度実験をしてうまくできたら、[Ctrl]+Zを数回押してで最初の状態まで戻し、今度は「開発」タブのコードの「マクロの記録」をクリックしてから一連の操作を繰り返し、最後に「記録終了」のボタンをクリックします。Visual Basic Editorのコード画面で見ると次のようなコードが自動的に書き加えられていることがわかります。

Sub Macro5()

'

' Macro5 Macro

'

'

ActiveSheet.Shapes.AddChart.Select

ActiveChart.ChartType = xlXYScatterSmoothNoMarkers

ActiveChart.SetSourceData Source:=Range("Sheet2!$A:$B")

ActiveSheet.Shapes("グラフ 6").IncrementLeft -131.25

ActiveSheet.Shapes("グラフ 6").IncrementTop -201.75

ActiveChart.Axes(xlValue).Select

ActiveChart.Axes(xlValue).MinimumScale = 0

ActiveChart.Axes(xlValue).MinimumScale = 35

ActiveChart.Axes(xlValue).MaximumScale = 50

ActiveChart.Axes(xlValue).MaximumScale = 45

ActiveChart.Axes(xlCategory).Select

ActiveChart.Axes(xlCategory).MinimumScale = -15

ActiveChart.Axes(xlCategory).MinimumScale = -10

ActiveChart.Axes(xlCategory).MaximumScale = 10

ActiveChart.Axes(xlCategory).MaximumScale = 5

ActiveChart.ChartTitle.Select

Selection.Delete

ActiveSheet.ChartObjects("グラフ 6").Activate

ActiveChart.Legend.Select

Selection.Delete

ActiveSheet.ChartObjects("グラフ 6").Activate

ActiveChart.Axes(xlValue).Select

Selection.Delete

ActiveSheet.ChartObjects("グラフ 6").Activate

ActiveChart.Axes(xlCategory).Select

Selection.Delete

ActiveSheet.ChartObjects("グラフ 6").Activate

ActiveChart.Axes(xlValue).MajorGridlines.Select

Selection.Delete

ActiveSheet.ChartObjects("グラフ 6").Activate

ActiveChart.SeriesCollection(1).Select

With Selection.Format.Line

.Visible = msoTrue

.ForeColor.ObjectThemeColor = msoThemeColorText1

.ForeColor.TintAndShade = 0

.ForeColor.Brightness = 0

End With

With Selection.Format.Line

.Visible = msoTrue

.ForeColor.ObjectThemeColor = msoThemeColorText1

.ForeColor.TintAndShade = 0

.ForeColor.Brightness = 0

.Transparency = 0.5

End With

With Selection.Format.Line

.Visible = msoTrue

.Weight = 1

End With

ActiveChart.ChartArea.Select

ActiveSheet.Shapes("グラフ 6").ScaleWidth 0.7166666667, msoFalse, \_

msoScaleFromTopLeft

End Sub

　マクロを記録すると多くの余計なコードが入ります。たとえば、最初の5行のコメント行は不要ですし、また次の動きもとくに必要ではありません。

ActiveSheet.Shapes("グラフ 6").IncrementLeft -131.25

ActiveSheet.Shapes("グラフ 6").IncrementTop -201.75

ActiveChart.Axes(xlValue).Select

　これらが縦軸を選択したときのコードであることがわかりますが、これは次の最小値と最大値の入力のために必要だった操作にすぎません。最小値と最大値の入力ができれば十分です。

ActiveChart.Axes(xlValue).MinimumScale = 0

ActiveChart.Axes(xlValue).MinimumScale = 35

ActiveChart.Axes(xlValue).MaximumScale = 50

ActiveChart.Axes(xlValue).MaximumScale = 45

　ここに最小値と最大値の初期値と変更値が記入されていますが、実は変更値だけが必要なのです。

ActiveChart.ChartTitle.Select

Selection.Delete

　これはタイトルを選択してから選択部分を削除する、というコードですが、これは次の1行にすることができます。

ActiveChart.ChartTitle.Delete

　ほかにも不要な指定を外して簡略化したのが次のコードです。

Sub 白地図描画実験()

ActiveSheet.Shapes.AddChart.Select 'チャートを追加

With ActiveChart '現チャートについて

.ChartType = xlXYScatterSmoothNoMarkers '散布図（平滑線）

.SetSourceData Source:=Range("A:B") 'XY値のソース

.ChartTitle.Delete 'タイトルを削除

.Legend.Delete '凡例を削除

.Axes(xlValue).Delete '縦軸を削除

.Axes(xlValue).MajorGridlines.Delete 'グリッド線を削除

.Axes(xlValue).MinimumScale = 35 '緯度最小値

.Axes(xlValue).MaximumScale = 45 '緯度最大値

.Axes(xlCategory).Delete '横軸を削除

.Axes(xlCategory).MinimumScale = -10 '経度最小値

.Axes(xlCategory).MaximumScale = 5 '経度最大値

.SeriesCollection(1).Format.Line.Weight = 1 '線の太さ

.SeriesCollection(1).Format.Line.ForeColor.ObjectThemeColor = \_

msoThemeColorText1 '線の色

.SeriesCollection(1).Format.Line.Transparency = 0.5 '線の透明度

End With

End Sub

　以下ではこのコードを使って繰り返し実験します。

　Excelシートにグラフを出力するときはデータを選択し、「挿入」からグラフの種類を選択すると標準的な初期設定によるグラフが出力されます。次に、ユーザーがそれを目的にあった形に編集してグラフを完成させる、という手順になりますが、これはその場限りのアドホックな手続きです。その場限りなので、異なるデータで同じグラフを出力するときでも、やはり同じ手順を繰り返さなければなりません。一方、マクロプログラムを使用すると、AB列のデータが変更されても、すべて自動化してグラフを出力します。このように手続きが一般化されるのです。一度だけの操作ならばアドホックな手続きを行い、何度でも繰り返す作業ならばプログラムを組む、というのが得策です。

## データを削る

　この操作をするのにExcelは少し動作が遅くなっているような感じがします。データが非常に多いことが原因だと思われます。調べてみると67920行あります。これだけの線を描くのに、これほど大規模なデータの精度と量が必要でしょうか？

　先の方針に従ってデータの削減と絞り込みをしてみましょう。小数点以下の桁数を下げるには、エディター（たとえばフリーウェアの「サクラエディター」など）の「正規表現」を使って置換することになります。正規表現は個別の文字列を置換するのではなく、一定のパタンにマッチしたすべての文字列を置換するために使われる表現形式です。次は小数点以下2桁に変換するための正規表現です。

置換前：(\.\d{2})\d\*

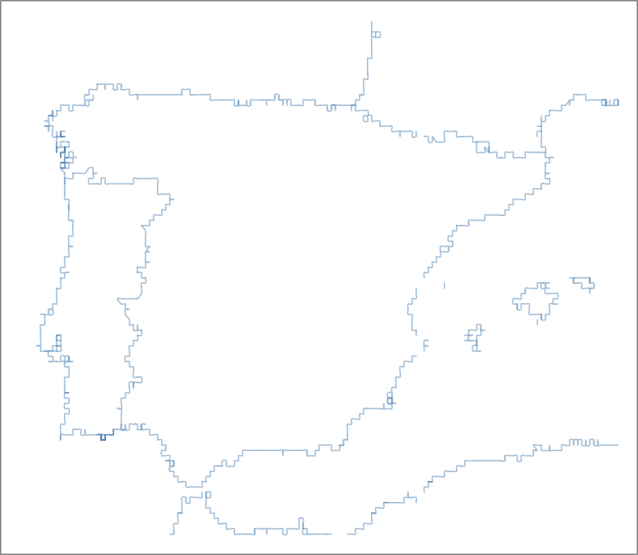
置換後：$1

　置換前の正規表現の最初の文字は開き括弧「(」です。これについてはすぐ後で説明します。開き括弧の後にある「\.」はピリオド「.」そのものを示します。ここで「\」をつける理由は、「\」が正規表現でピリオド「.」が任意の文字を示す、という規定があるためです。「\」は、それを避けてピリオドそのものを示すときに使われる記号です。次に続く「\d」は数字1文字（0 ~ 9）を示します。直後の{2}は直前の正規表現（ここでは\d）に一致した数を示します。よって「\d{2}」は2桁の数字を示すことになります。その後にある「\d\*」は数字がゼロ個以上、つまり何もないか、または数字の連続を示します。

　丸括弧「(…)」は、この中に記述された正規表現に一致した文字列を、置換後の$で再生されることを意味します。置換後の$の後にある数字は、置換前の正規表現の「(…)」が複数あるときの区別に用いられ、ここでは「(…)」が1個しかないので「$1」を使います。

　たとえば、エディターの中にある「-80.974613」を正規表現の「(\.\d{2})\d\*」で検索すると「.974613」がヒットし、そのなかの「.97」が「(\.\d{2})」に対応し、「4613」が「\d\*」に対応します。置換後は「$1」によって「.97」だけが再生されます。小数点の前の「-80」は置換前の正規表現にヒットしないので、置換の対象にならないため、そのまま再生され、結局-80.97という数字に置換されます。

　この置換前の正規表現「(\.\d{2})\d\*」の中の「{2}」の数字を変えていけば、小数点以下の数字の精度を調節できる。先に用意したマクロプログラムを使って実験してみると、Excelで地図を描くために必要な精度は小数点以下2桁で十分なことがわかりました。小数点以下1桁にすると小さな範囲の地図を描くときに次のように線が荒くなります。

　日本地図（上左図）のように範囲が比較的大きい場合は目立ちませんが、イベリア半島地図（上右図）などのように範囲が小さくなると線の粗さが目立つようです。

　さて、正規表現を使ってNOAAの座標データを小数点以下2桁に揃えると次のように多くの同じデータが連続するようになります。

-6.25 35.00

-2.13 35.00

-2.13 35.00

-5.99 35.58

-6.00 35.58

-6.00 35.58

-6.00 35.58

　これでは、データの記憶域もデータ処理も無駄になるので、同じ内容のデータ行は1つに絞りこまなければなりません。たとえばサクラエディターには「連続した重複行の削除(uniq)」という機能があります[[9]](#footnote-9)。全範囲を選択し、これをクリックすると次のようなデータが得られる。

-6.25 35.00

-2.13 35.00

-5.99 35.58

-6.00 35.58

　このように重複行を削除すると全体で15852行にまで減らすことができました。もちろん重複行を削除しても地図の精度に変化はありません。これは先のマクロで実験して確かめることができます。

## データを絞る

　次に、先の方針(2) に従って、座標データの連続をとびとびに抽出して、結果が満足できるところまでデータ数を絞り込みたいと思います。ここで10ずつのステップで抽出してみましょう。次はそのためのVBAプログラムです。

Sub 少数データ間引きセル()

Dim x, y, i&, j&, Df!

Df = 0.04

For i = 1 To 100000 '全座標数

x = Cells(i, 1): y = Cells(i, 2)

If x = Empty Then

j = j + 1 '出力行

Cells(j, 1) = x: Cells(j, 2) = y '出力

ElseIf Abs(x - Cells(j, 1)) > Df Or Abs(y - Cells(j, 2)) > Df Then

j = j + 1 '出力行

Cells(j, 1) = x: Cells(j, 2) = y '出力

End If

Next

Range("J1") = j '出力数

End Sub

　最初に使用する2つの変数iとjを**Dim i&, j&**で宣言します。これらには長整数（絶対値が34000を超える非常に大きな整数）のデータ型を示す接尾辞&をつけます。一般にはこのために次のような型宣言子**As Long**を使いますが、接尾辞による表記によって簡略化できます。

Dim i As Long, j As Long

**For *a* = *b* to *c*… Next**は***a***が***b***から***c***に至るまでの間**Next**までの命令を実行せよ、という意味です。ここでは***i***が1から15852（全座標数）になるまで繰り返されます。ふつうは***i***は1つずつ増えていくような方法をとるのですが、ここではステップするために**Step 10**としてあります。その後に'「全座標数、ステップ」という説明があります。これはプログラムの実行に関係しないメモです。このようなアポストロフィ( ' )の後のメモを「コメント文」といいますが、コメント文はなるべく書くようにしましょう。別の人コードを読むとき、コメント文がないと製作者の意図がよくわからないことがあるし、製作者自身も時がたてば忘れてしまうことがあるからです。

　この**For … Next**の繰り返し構文の中にある**x = Cells(i, 1): y = Cells(i, 2)**は、セルの内容を**x, y**の変数に代入します。変数を使わなくても可能ですが、変数を使った方が読みやすくなります。この文によって、読み手は**x, y**の入力セルの位置を一度確認すれば出力のセルだけに注意を向ければよいからです。

　次のIf A Then X ElseIf B Then Y End Ifは条件Aが満たされればXの処理を行い、そうでなくて条件Bが満たされればYの処理を行う、という意味です。

**j = j + 1**は現在の**j**の値に1を加えてあらたに**j**に代入せよ、という命令を示します。プログラムコードのイコール( = )は右辺の値を左辺に代入する、という働きをします。**For … Next**の繰り返しによって、このコードを通過するごとに**j**の値は1つずつ増えていくことになる。次のコロン( : )によって区切られる2つの命令文はよく似ています。左の**Cells(j, 1) = x**は、xの内容を**Cells(j, 1)**に代入せよ、という命令です。

**For … Next**の繰り返しを終えると、**j**は最終的な出力行の位置を示しているので、最後に**Range("J1") = j**によって出力数( **j** )をセルJ1に代入しています。

## レンジを使う

　先のプログラムでは座標データをCellsに出力し、出力数をRangeに出力しました。どちらもExcelシートに代入されるのですが、代入のされ方が違います。Cellsでは個別にセルの位置（行と列）を指定します。一方、Rangeでは特定の範囲を指定して、その範囲から一括して入力したり、一括して出力したりすることができます。実は先のプログラムではたった1つのデータだったのでRangeを使う理由はあまりありません。たとえばRange("J1")はCells(1, 10)であっても同じことになります。あえて言えば、Range("J1")と書いた方が、すぐにその位置を確かめることができるので便利だと思います。一方Cells(1, 10)だと、1行10列の位置を数えて探していかなければならないのです[[10]](#footnote-10)。

　次は先の「ステップ少数データ」のプログラムをRangeを使って書き換えたものです。

Sub 少数データ間引きレンジ()

Dim Iv, Ov(100000, 1 To 2), x, y, i&, j&, Df!

Df = 0.04

Iv = Range("A:B") '座標データ

For i = 1 To 100000 '全座標数

x = Iv(i, 1): y = Iv(i, 2)

If x = Empty Then

j = j + 1 '出力行

Ov(j, 1) = x: Ov(j, 2) = y '出力

ElseIf Abs(x - Ov(j, 1)) > Df Or Abs(y - Ov(j, 2)) > Df Then

j = j + 1 '出力行

Ov(j, 1) = x: Ov(j, 2) = y '出力

End If

Next

Range("H:I") = Ov '配列をレンジに出力

Range("J1") = j '出力数

End Sub

**Dim Iv, Ov(1000000, 1 To 2)**で2つのデータの変数を宣言します。どちらもバリアント型といってデータ型が自由なタイプの変数です。これを「配列」として使います。配列は文字や数字などの個々のデータの行列を扱います。Ivには次の行で入力レンジの内容を一括して代入されるので、このような場合は配列の大きさを決めずに、代入された時点でその大きさが決定されます。一方、Ovは添え字を使って、配列の大きさを決めます。これは以下の処理で個別に配列の一部に代入されていくためです。Ov(15852, 1) のような大きさにすると、縦に15852行用意され、横に0と1の2行用意されます。このように配列は0から始まりますが、セルと同じように1から始めると、処理がわかりやすくなります。そこで、Ov(15852, 1 To 2)と記入することで、横列の添え字は1, 2になります。このような配列を用意すると、後で見るように、シートへの書き出しの2列に一括して代入することができます。

**Iv = Range("A:B")**はシートのA列とB列の範囲(Range)にあるデータ全部を**Iv**という配列に代入します。これでたとえばセルA1にあるデータはIv(1, 1)という添え字つきの変数に代入され、セルB1にあるデータはIv(1, 2)に代入されます。以下同様にすべてのデータが**Iv**の配列（行列）に一気に代入されるのです。これを**Range**を使わないでCellsを使って書くと次のように**For … Next**を二重にして命令を記述しなければなりません。

For i = 1 to 15852

For j = 1 to 2

Iv(i, j) = Cells(i, j)

Next

Next

　この方法は、記述が煩雑になるだけでなく、非常に多くの回数でCellsの内容を呼び出しているので時間がかかりPCに負担があり、回数が多いとしばしばフリーズしたような状態になります[[11]](#footnote-11)。一方、Rangeを使った**Iv = Range("A:B")**という命令ならば一瞬にしてA列とB列にある巨大なデータセットが配列に代入されます。

**If … End If**の中にある**Ov(j, 1) = x: Ov(j, 2) = y**は、配列**Iv**の要素を配列**Ov**に代入しています。これは先のプログラムでは配列を使わずにセルCellsを使っていました。どちらでも同じように見えますが、実は配列は変数なので直接セルというオブジェクトに働きかけていません。セルに働きかけると一定の時間がかかります変数間のやり取りは非常に高速です。

**For … Next**のループを終えたら、**Range("H:I") = Ov**によって配列変数Ovの内容をシートのH列とI列に一括して代入します。これもCellsを使うと次のようなコードになります。

For i = 1 to j

For j = 1 to 2

Cells(i, j) = Ov(i, j)

Next

Next

　ここでも毎回セルにアクセスするので時間と負担がかかります。以下のプログラミングではセルへのアクセス数が多いときはレンジを使用します。一方、少ないアクセスであればわざわざレンジを使うメリットはありません。

## 全データ処理

　これで比較的小さなサンプルデータについて座標データ作成の手順を一応終わらせたことになります。これは実験です。今度はいよいよ本番の世界全図のデータの座標を作成することにしよう。

　はじめにNOAAのデータを小数点以下2桁にして重複行をまとめても、実験のような数ではなく今度は2776935行にもなりました。このままではExcelにインポートすることができません。Excelは104万行が限界だからです。そこで100万行ごとにコピーしてExcelシートの列にペーストすることにましたた。エディターではそれぞれの行が2列になっているので、Excelシートではそれが3倍されることになり、次のように6列になりました。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| -80.97 | 51.99 | -106.43 | 23.19 | -76.55 | 39.25 |
| -80.97 | 51.98 | -106.42 | 23.19 | -76.56 | 39.25 |
| -80.96 | 51.98 | -106.42 | 23.18 | -76.57 | 39.26 |
| -80.96 | 51.97 | -106.43 | 23.18 | -76.57 | 39.27 |
| -80.95 | 51.97 | -106.43 | 23.17 | -76.58 | 39.27 |

(…)

　次が先の「ステップ少数データレンジ」に一部追加して作った「ステップ多数データレンジ」のプログラムです。

Sub 多数データ間引きレンジ()

Dim Iv, Ov(2776935, 1 To 2), x, y, i&, j&, Df!

Df = 0.04

Iv = Range("A:F") '座標データ

For i = 1 To 2776935 '全座標数

If i > 2000000 Then '第3列

x = Iv(i - 2000000, 5): y = Iv(i - 2000000, 6)

ElseIf i > 1000000 Then '第2列

x = Iv(i - 1000000, 3): y = Iv(i - 1000000, 4)

Else '第1列

x = Iv(i, 1): y = Iv(i, 2)

End If

If x = Empty Then

j = j + 1 '出力行

Ov(j, 1) = x: Ov(j, 2) = y '出力

ElseIf Abs(x - Ov(j, 1)) > Df Or Abs(y - Ov(j, 2)) > Df Then

j = j + 1 '出力行

Ov(j, 1) = x: Ov(j, 2) = y '出力

End If

Next

Range("H:I") = Ov '配列をレンジに出力

Range("J1") = j '出力数

End Sub

　入力用のデータは大きくレンジ列A:Fを代入します。これは2列ごとに3組あるデータなので、全体で1000000行6列になっています。**For … Next**のループの中iの値によって、3つの条件に分岐します。**If *A* Then *X*, ElseIf *B* Then *Y* Else *Z* End If**という構造では、***A***であれば***X***の処理をし、***A***でなくて***B***であれば***Y***の処理をし***A***でも***B***でもなければ***Z***の処理をします。

　ここでA, B, Elseの条件は、それぞれ200万を超えるとき、100万を超えるとき、そして100万以下、ということになります。iが200万を超えるときは第3列（E:F列)から入力データを取得します。iが100万を超えるときは第2列（C:D列)から入力データを取得します。それら以外の場合は第1列A:B列)から入力データを取得します。

　ここで、次のように、条件の数を昇順にすることはできません。

If i > 0 Then '第1列

経 = Iv(i, 1): 緯 = Iv(i, 2)

ElseIf i > 1000000 Then '第2列

経 = Iv(i - 1000000, 3): 緯 = Iv(i - 1000000, 4)

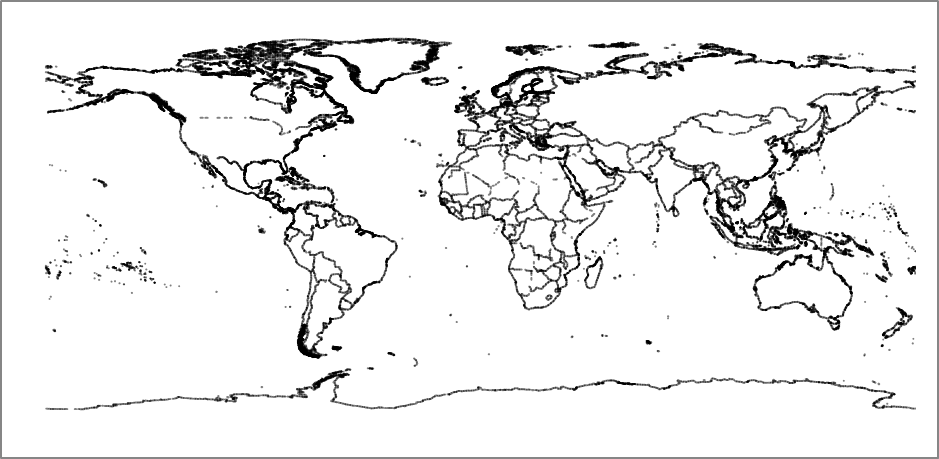
Else '第3列

経 = Iv(i - 2000000, 5): 緯 = Iv(i - 2000000, 6)

End If

　このようにすると、最初の条件**If i > 0 Then**にすべてのケースが一致してしまいます。

　次が世界全図の座標データを先のマクロの記録を使って出力した結果です。

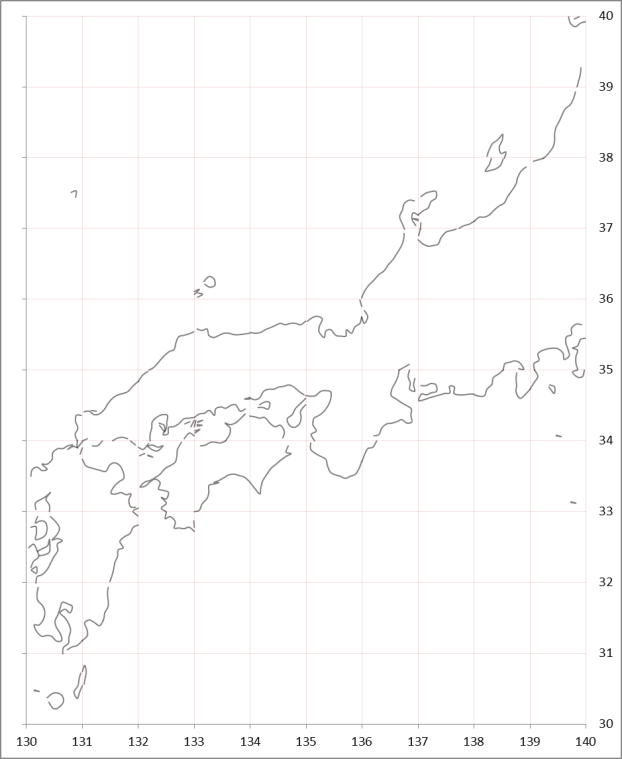
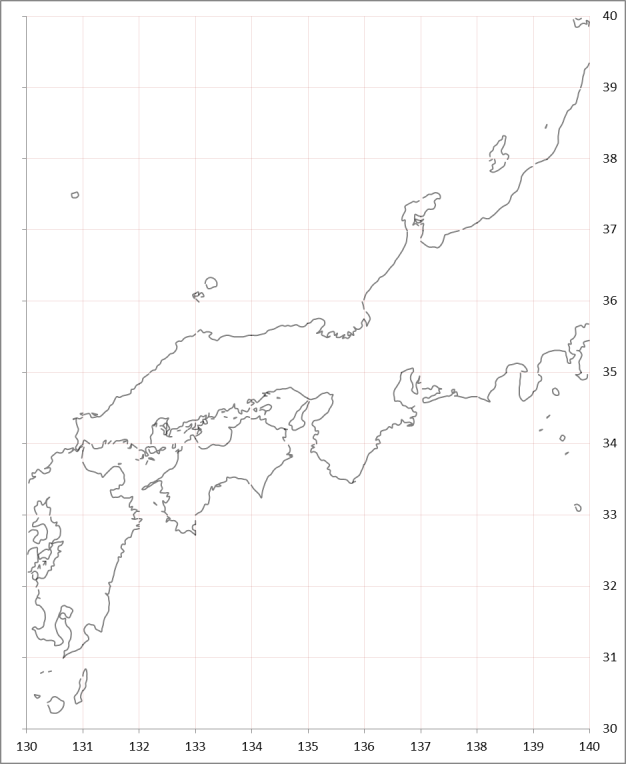


## 順次検索と二倍二分検索

　座標データを「削る」ときに小数点以下の適切な桁数を探さなければならなりませんでした。ここで適切な桁数を「OK値」と呼び、適切でない桁数を「NG値」と呼びましょう。OK値を探すには小数点以下1桁、2桁、…というように順次実験を繰り返してNG値からOK値に変わる位置を見つけなければなりません。ここで扱った座標データの小数点以下の桁数は6桁なので、OK値は1, 2, …., 6のいずれかです。それならば、1から順に実験を重ねていけば6回以内の実験でOK値が見つかるはずです。逆に6, 5, …のように降順でもかまいません。そのときは6がOK値なのでNG値になるまで順次NG値を検索することになります。このような検索法は「順次検索」と呼ばれています。順次検索では次のようにひたすら一方向で検索値を設定していきます。

　1 (OK), 2 (OK), 3 (OK), 4 (OK), 5 (NG)

　一方、座標データを「絞る」ときには適切な数値差（プログラムの変数Df）を探さなければなりません。このとき0.01から順次OK値（よい結果を示すステップ数）を1.00まで検索するのは得策ではありません。そこで0.01で実験しそれがOK値ならば次はそれを2倍して0.02で実験しOKかNGかを判断する。0.02もOKならば、次は順次検索のように0.03ではなく、0.02の2倍の0.04で実験する。これもOK値ならば2倍の0.08を試します。この実験で0.08は下左図のようにNG値でした。経度と緯度のそれぞえ10度の幅にある西日本の範囲です。あまりきれいな図でないのでNGとします。そうすると求める最大のOK値は0.04以上0.08未満ということになります。そこで0.05, 0.06, 0.07という風に順次検索することも考えられますが、そうすると3回の実験が必要になります。そこで0.04と0.08の中間値0.06を試し、それがNGならば次は降下して5を、それがOKならば次は上昇して0.07を実験します。これで0.05（または0.07）を検査すればよいので、どちらにしても全部で2回の実験をすればよいでしょう。この座標データでは0.05がNG値だったので、結局求めるOK値は0.04になりました（下右図）。

Df = 0.08 / 529,670 Df = 0.04 / 655,848

　このような検索法を「二倍二分検索」と呼ぶことにしましょう[[12]](#footnote-12)。二倍二分検索では次のように検索値が上昇したり降下したりします。

0.01 (OK), 0.02 (OK), 0.04 (OK), 0.08 (NG), 0.06 (NG), 0.05 (NG)

　順次検索は1つずつ着実にOK値（またはNG値）を探すので最大検索回数が少ない場合に有効です。一方、二倍二分検索は多くのデータ数の中から未知の最適OK値を求める場合に有利です。ここで扱った座標データの場合、たまたま最大OK値は0.04だったので、これを順次検索しても、0.01 (OK), 0.02 (OK), 0.03 (OK), 0.04 (OK), 0.05 (NG) という5回の実験で得られる数ですから大した手間にはならないでしょう。一方、二倍二分検索をすると、0.01 (OK), 0.02 (OK), 0.04 (OK), 0.08 (NG), 0.06 (NG), 0.05 (NG)で、6回の実験が必要になります。これではシンプルで早い順次検索を選択したくなります。しかし、最大OK値=0.04は実験をしたからわかった数字です。検索するときは、それがわからないのですから、いきなり先の見えない順次検索をするわけにはいきません。いつかは見つかるはずなのですが、それがいつになるのか、まったく目安がたたないのです。

　たとえば最大OK 値が0.23だったとしましょう。これを二倍二分検索で見つけるには次のような手順になります。

　0.02 (OK), 0.04 (OK), 0.08 (OK), 0.16 (OK), 0.32 (NG), 0.24 (NG), 0.22 (OK), 0.23 (OK)

　このようにはじめはNG値(0.32)になるまで二倍ずつ検査値を上昇させます。それがNG になったら今度は下降しますが、このとき0.16と0.32の間を二分する値（中間値）を求めるに0.16とその前の数（またはその半分の数）8を足して24を実験します。次に24がNGであることがわかりました。そこでもう一度検査値を下降します。16と、その前の前の数4を足して20を実験します。20はOKですから16の前の前の前の数を足して22を実験し、これもOKであるので、さらに前の数を足して23に至る。このように実際の計算ではすでに計算した倍数の連続値を使うので、あまり面倒ではありません。

　それでは、まったく範囲のわからない未知の最適OK値を探すにはどうすべきでしょうか？このときは100, 101, 102, 104, 108, …のように、たとえば10を底にして、その指数を使った二倍二分検索をして、おおよその範囲を決めてから、その範囲内で整数の二倍二分検索をするとよいでしょう。

## 学習の類型

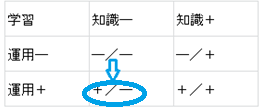
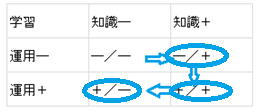
　サクラエディターで小数点以下2桁にするための正規表現は次でした。

置換前：(\.\d{2})\d\*

置換後：$1

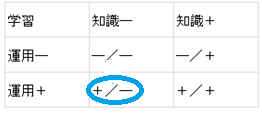
　正規表現は他にもいろいろな規則がありますが、その数はそれほど多くありません。すぐ覚えられる量です。しかし、その使い方に慣れるのに少し練習が必要です。参考書やウェブで、検索・置換の例がたくさん載っていますから、それを見る・読むだけではなく、実際に使って確認し納得するとよいでしょう。そうでないと、知識だけが身について、その運用ができないことになるからです。

　外国語などの学習は一般に「(1) －知識・－運用」→「(2) ＋知識・－運用」→「(3) ＋知識・＋運用」→「(4) －知識・＋運用」という過程で進みます。はじめは誰でも知識もなければ運用もできません（「(1) －知識・－運用」）。外国語を学ぶことによって知識が得られますが、はじめは運用がなかなかうまくいきません（「(2) ＋知識・－運用」）。次に練習によって運用ができるようになります（「(3) ＋知識・＋運用」）。最後は知識を意識しないで、また、動詞の活用などを忘れても、自由に運用ができるようになります（「(4) －知識・＋運用」）。上のようなプロセスで学んだブラインドタッチのタイピングができる人が、たとえば「W」の文字がどこにあるのか質問されてもすぐに答えられないことがあります。これは最終の「－知識・＋運用」の段階を示しています。つまり、個々のキーがどの位置にあるかを知識としてもっていなくても自然にキーインができる状態です。学んで覚えるスポーツも同じことでしょう。



　一方、運用能力の獲得が「(1) －知識・－運用」→「(2) －知識・＋運用」という過程で一気に進むことがあります。これは「習うより慣れろ」という方法です。外国語を学校で習わなくても、外国で生活している中で実際に使っているうちに自然に身につけてしまうことがありますが、それがこのケースです。

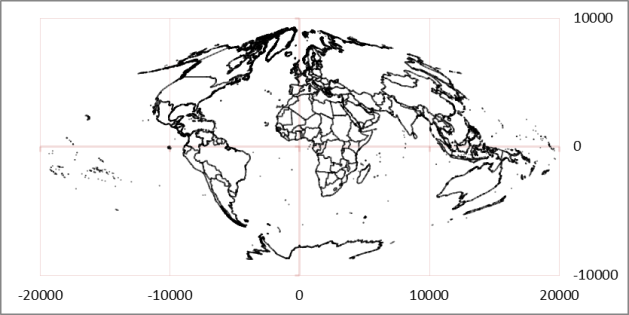
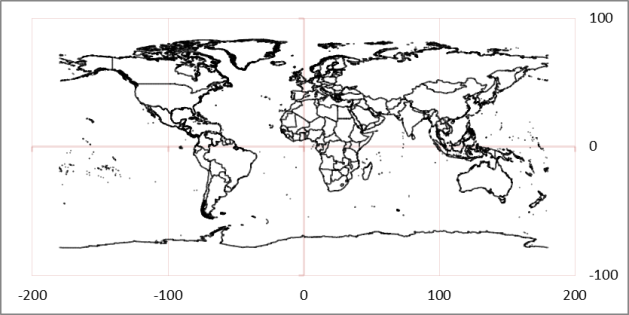
　さらに、機械を実際に使っていなくても、はじめからその運用ができてしまう、ということがあります。たとえば小学生が教わっていない機器類を直感や類推で操作できてしまう、というケースです。これは「－知識・＋運用」という状態が最初から存在していることになります。



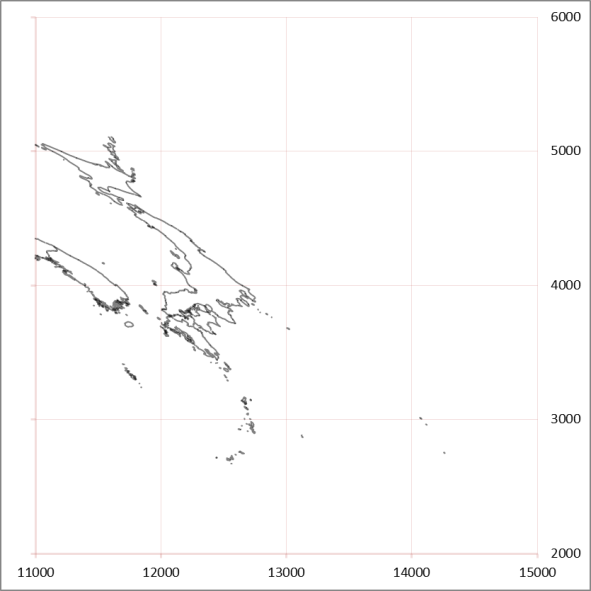
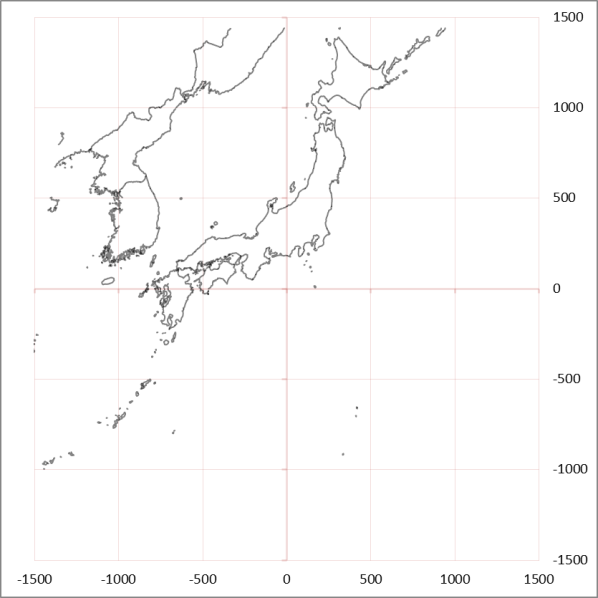
　正規表現には一種の外国語のような文法規則がありますが、この学習はどうしても「＋知識」という要素がある前者の過程を踏まなければなりません。外国語と違って、その言語が使われる環境で生活し、自然に身に着ける、というわけにはいかないからです。そして、正規表現を見れば直感的にその意味が理解できる、ということはほとんど不可能に近いものです。

# 白地図

　球体の地球上にある地点を平面の地図に書き込むには白地図を作成しなければなりません。この作業が一筋縄にいかないのは、球面上の座標（経度と緯度）を平面上の座標（横座標xと縦座標y）に変換する方法がさまざまに考えられるからです。私たちの目的は言語地図を作成することなので、方位や距離について高い厳密性を求めません。正式のメルカトル図法（経度と緯度で作る升目が正方形ではなく、高緯度地域では縦長の長方形になる）のように高度な技法を取り入れてプログラムを複雑にしたり、コード数を増やしたりすることは、プログラムの数理を理論的に探究するためには意味がありますが、私たちの実際の目的には不必要なことです。言語地図を描くためには、よく見慣れた海岸線と国境線が描ければ十分です。そこで、厳密なメルカトル図法を簡略化した「正方図法」（下左図）と、経度と緯度から距離を算出してキロメートルを単位とした座標を作る「距離図法」（下右図）のプログラムを考えました。



　当初は正方図法だけを考えていたのですが、この図法では私が専攻するスペイン語の言語地図を描くのは不都合であることに気づきました。正方図法だと緯度が高くなるにつれて、縦と横の距離関係の崩れ方が大きくなるのです。そこで、狭い地域の地図には、経度・緯度よりも東西方向と南北方向の平行線間の距離を示す座標を使うことを考えました（距離図法）。はじめは、経度と緯度をそれぞれゼロ度からの距離をキロメートルに換算し、散布図にプロットすると、そのゼロ地点[[13]](#footnote-13)を中心にして形がいびつになることがわかりました（下左図）。逆に、中央付近では形が正確なので指定する座標の範囲の経度・緯度の始点と終点の中間値からの差を使って距離を計算することにしました。そうすれば出力される地図は座標の中央に位置するので形が正確になります（下右図）。

　試行錯誤の結果、イベリア半島（スペインとポルトガル）のような比較的小さな地域の地図化のためには、距離図法のほうが有効だとわかりました。逆にスペインとアフリカ・ギニア共和国、ラテンアメリカの広域スペイン語圏の地図化のためには正方図法が適しています。

　経度・緯度による座標から距離（キロメートル）の座標に変換するとき、緯度1度にあたる距離は地球上のどこでも同じなのに、経度1度にあたる距離は赤道と極地に近いところでは大きく違うことに注意しなければなりません。極地に近くなるとどんどん距離が短くなり、極点（北極と南極）では距離がなくなってしまいます。そこで経度1度あたりの距離を計算するために緯度に依存する関数を作ることにしました。このときに高校で習った三角関数のコサイン(cos)の使い方を思い出さなければなりませんでした。

　地図のプロットのためには座標の計算が必要ですが、それを実際の尺度でプロットするのではなく、ユーザーが決めた横幅に合わせて、縦幅を計算し、その両サイズに合わせた記入のための計算をしなければなりません。このときに中学校で習った比例式を活用します。

　ところで、緯度1度にあたる距離は地球の円周約4万キロを360度で割った商である約111.11キロになります。地球の円周は正確に4万キロであることは計測の結果4万キロであったのではなく、逆に、「メートル」を子午線の90度の距離を使って、それを1000万メートルと定義したからです。これをフランス革命後のフランス国民議会が承認しました[[14]](#footnote-14)。

## 使い方

　　はじめに経度と緯度の座標データを使って白地図を作成します。白地図はさまざまな地図情報を記入するためのベースになるものです。全座標データから指定する範囲のデータを抽出してExcelの散布図を使って白地図をプロットします。海岸線と国境線の色とは赤、緑、青のRGB値の混色と透明度で設定します。図法の種類には、全体を完全な正方形の升目にする「正方図法」と緯度と経度を距離（キロメートル）に直してプロットする「距離図法」の中から選択します。

　「白地図」シート内の指定位置(E2:F3、シート内では青く塗りました)に対象とする地域の経度と緯度の始点と終点を記入し[[15]](#footnote-15)、「白地図チャート」をクリックすると次のように白地図が出力されます。「全チャート削除」ボタンを押すとすべてのチャートが消去されますが経度と緯度などのデータは残ります[[16]](#footnote-16)。

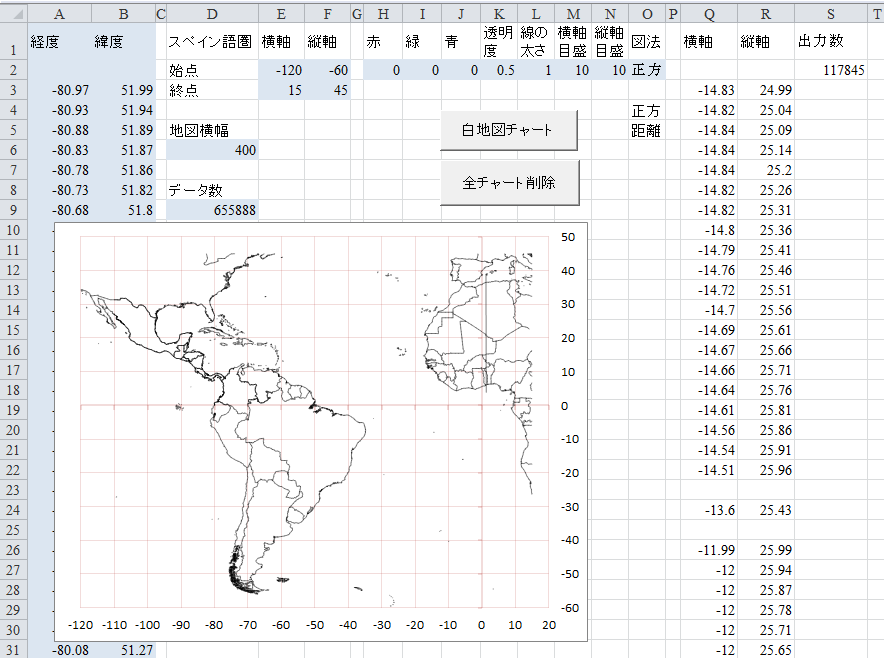
　シート内の経度と緯度の数値行列(A:B)から指定の範囲(E2:F3)にしたがってQ:R列に抽出した経度緯度データを出力し、同時に出力数セルS2に表示します。海岸・国境線の色と透明度と太さはH2:K2で指定します。色は赤、緑、青の値(0 ~ 255)を使って単色または混色で指定してください。混色は光の3原色の原則に従い、たとえばすべての色をゼロ(0)にすると黒になり、すべての色を最大値(255)にすると白になります。T列以降はメモとして使います。透明度を0とすると完全な色になり、1にすると完全に透明になるので線が見えなくなります。太さの単位は「ポイント」です（1ポイント＝約0.35mm）。

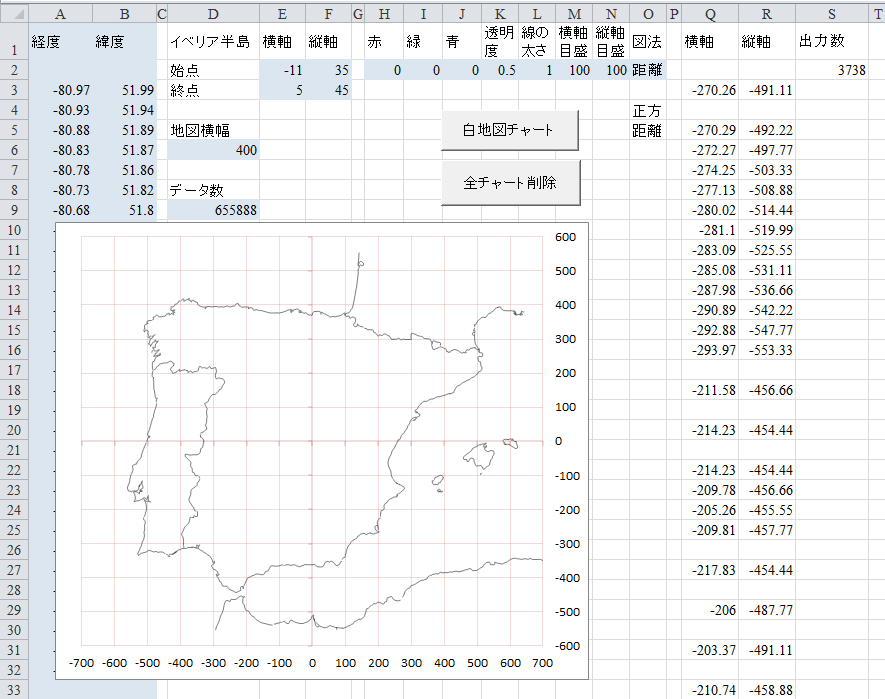
　下図は「正方図法」です。セルO2で「正方」を選択します。その下の図は緯度と経度をキロメートルで換算しプロットした「距離図法」です。正方図法では緯度・経度の度数で、距離地図ではキロメートルを単位にして「横軸目盛」と「縦軸目盛」を指定してください。

　「横軸目盛」と「縦軸目盛」はM2N2で指定します。間隔をゼロ(0)にすると目盛が表示されません。正方図法では経度と緯度の度数を単位として使い、距離地図ではキロメートルを使います。単位は1, 10, 100, 1000などの切りのよい数を使うとよいでしょう。

　海岸・国境線を非表示にし（QR列を削除します）、経線と緯線を表示し、「図法」を「正方」にすると、以下の「地名・語形地図」「記号地図」「数量地図」を「名称散布図」「記号散布図」「数量散布図」として使用することも可能です。そのときは地図座標ではなく、「白地図」の範囲(E2F3)もそれに応じて変更し、「地名・語形地図」「記号地図」「数量地図」のAB列に観察データの数値を入れてください。

　チャートの線図はこのシートの数値QR列とリンクしているので、チャートが表示されている間はExcel全体の操作が少し重くなります。そこで、チャートが不要であれば「全チャート削除」をクリックしてチャートを削除するか、またはチャートを保存するときは「ピクチャー」または「ビットマップ」でコピーし（「ホーム」→「コピー」→「図としてコピー」）、適当な場所にペーストして、元のチャートは削除してください。または単純にカットしてWord内にPNG形式などでペーストしてもよいでしょう。

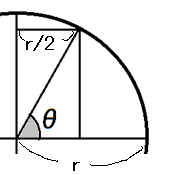




## 正方図法と距離図法

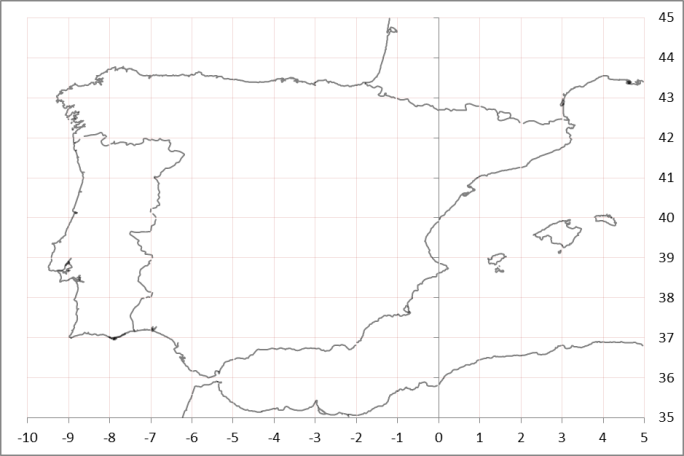
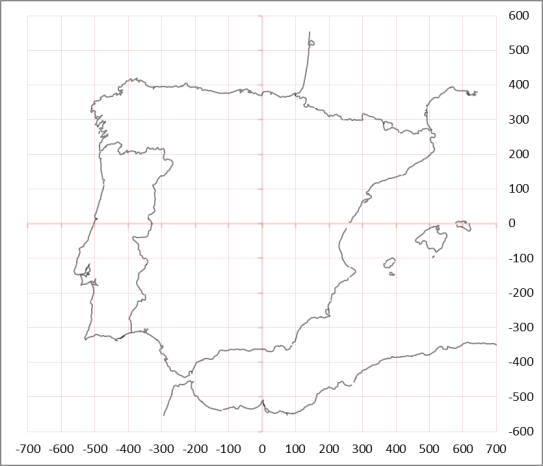
　このMAPASの正方図法はメルカトル図法に類似します。メルカトル図法の特徴は経線と緯線が地図上のどこでも水平線と垂直線になることです。一般にメルカトル図法の地図では緯度の絶対値が高くなると、つまり90度に近くなる部分では、それに応じてこの升目が縦長になっています[[17]](#footnote-17)。この理由は地球（球体）を平面に投射するとき0度から離れるに応じて同じ緯度の間隔（たとえば10度）の地図上の距離が実際の距離に比べて相当長くなるので、それに見合うように経度の間隔も伸ばしているからです。その結果地図上の面積は非常に大きくなります。

　地球上の緯度によって同じ経線間の実際の距離が異なるのですが、赤道における計算間の距離と比べてどのような比率になるのでしょうか。それは次の図を見れば簡単にわかります。地球の半径と外周の関係を考えてみましょう。



　たとえば緯度（θ）が60度であれば、その緯線の長さは緯度が0度の地点と比べて半分（cosθ= 0.5）になるはずです。上の図では地球の中心からの角度で緯度を表し、横線は地軸から地表への距離を示していますが、横の直線は半径を示しますから、これが半分になれば、外周も経度1度の距離（キロメートル）も半分になります（円周＝2π\*半径）。

　下左図は未調整のイベリア半島（スペインとポルトガル）の白地図（正方図法）です。緯度と経度の幅は同じにしてあるのですが、いつも見慣れた地図と比べてかなり横長に見えます。

　イベリア半島がおよそ北緯40度を中心にして位置しているためにかなり赤道から離れているので、経度1度の距離は短くなるはずです。それを正方図法では同じ長さで示しているので横に広がるのです。上右図の距離図法と比べてください。こちらの方が形が正確です。

　地球儀を見ると緯線間の実際の距離はどこでも同じなのですが、経線間の距離はその緯度によって異なっていることがわかります。たとえば緯度0度に近いところでは地球の外周4万キロを360度で割ると経度1度は111.11kmになるはずです(40000 / 360 = 111.11)。しかし、先に見たように緯度が60度の場所では経度1度がその半分の距離になります（111.11 \* cos 60度 = 55.56）。たとえばアフリカ・ケニアの首都ナイロビに近い赤道での経度差1度の距離（地球の横方向の距離）はおよそ111.11キロメートルになりますが、ロシアのサンクトペテルブルグに近い緯度60度の場所では経度差1度の距離はおよそ55.56キロです。

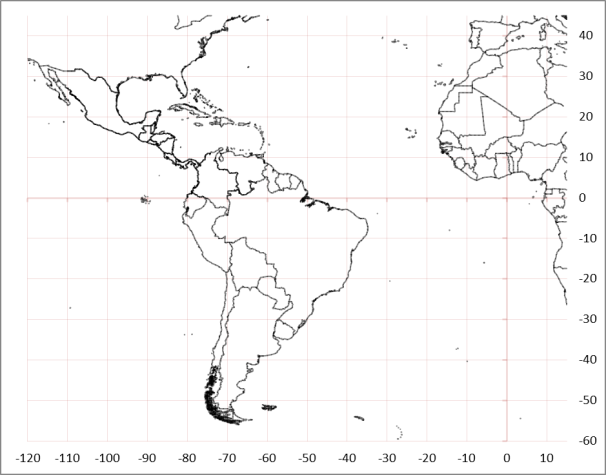
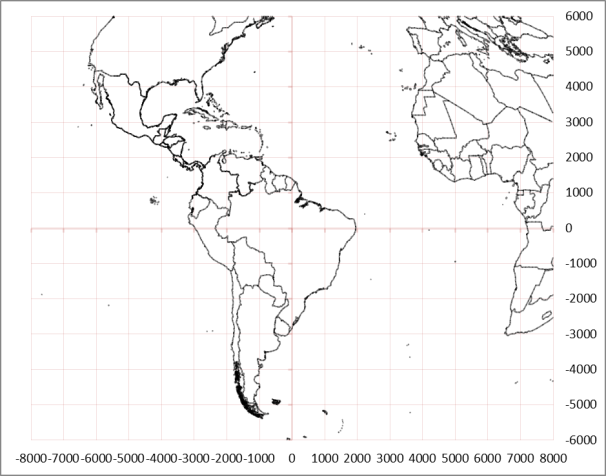
　MAPASの距離図法では座標データの経度と緯度を範囲の中心点からの距離（キロメートル）に計算しなおして地図をプロットします。よって、縦の距離も横の距離も正確になります。具体的には次の式を用います。

横座標＝中心からの経度差 \* 111.11 \* cos (緯度)

縦座標＝中心からの緯度差 \* 111.11

　縦座標はどこでも同じ距離になるので、すべて一律に緯度1度＝111.11キロメートルで計算します。横座標の距離を計算するためは経度に111.11を掛けて、さらに緯度のコサインを掛けます。

　下左図と下右図はそれぞれMAPASの正方図法と距離図法で描いたスペイン語圏の地図です。

　正方図法では方位が常に一定ですが、緯線が相対的に正しい距離を示していません。一方、距離図法ではすべての升目の辺はすべて同じ距離になるように調整していますが、方位の正しさが犠牲になります。また、地図の中心から離れた地域の形がゆがんで見えます。このように両者は一長一短なので優劣をつけるのではなく、相補的に評価すべきでしょう。先に見たイベリア半島のように比較的小さな地域では距離図法を使い、スペイン語圏のように大きな地域では正方図法を使う、という方法が考えられると思います。

## プログラム

Option Explicit 'MAPAS.xlsm 言語地図作成システム

Dim 横1!, 横2!, 縦1!, 縦2!, 横軸, 縦軸, 図横!, 図縦!, 例横!, 例縦!

Dim 形$, 文字列$, 形横!, 形縦!, 赤%, 緑%, 青%, 透!

Dim i&, j&, x!, y!, Iv, Ov

**Option Explicit**は変数を明示的に宣言する、という意味です。この文を外しておけば変数を宣言しなくてもプログラムを書くことができますが、そうすると変数名に間違いがあるかどうか、わからなくなります。

プログラムで使用する「変数」を**Dim**で宣言します。変数にはその値として具体的な内容が代入されます。その代入された値についてさまざまな処理が行えるのです。変数を使わないとプログラムが一般化できません。

　ここで使用する変数には次の型があります。

|  |  |  |
| --- | --- | --- |
| **変数型** | **範囲** | **印** |
| 整数 | -32,768～32,767 | % |
| 長整数 | -2,147,483,648～2,147.483,647 | & |
| （単精度）小数 | 小数点がある数 | ! |
| 文字列 |  | $ |
| バリアント型 | なし |  |

　上の右列にある印を変数の後につけて変数型を宣言します。それぞれの変数については該当部分で説明します。

　変数名に漢字を使うと、その値の意味がわかりやすくなり、コメント文が必要でなくなります。道具のようにして使う変数名はアルファベットにしました。これはとくに値の意味をあまり考えなくてもよい変数です。

Sub ■白地図チャート(): On Error Resume Next 'エラー処理

ActiveSheet.ChartObjects.Delete '全チャート削除

Call ▲図追加: Call ●データセット: Call ▲地図記入

End Sub

　チャートがないときにそれを削除しようとするとエラーになるので、**On Error Resume Next**をつけて、エラーがあってもプログラムは実行を続けるようにします。ただし、プログラミングの作業中は**On**の前にアポストロフィー( ' )をつけてコメント文にしておくとよいでしょう。プログラミングの最中は多くのエラーが発生し、その場所の特定が必要だからです。

　白地図チャートのプログラムの構成です。Callによってそれぞれの作業（サブルーチン）を呼び出します。

　プログラムはなるべくわかりやすいコードにしておくべきです。たとえば、この「■白地図チャート」のプログラム全体を1つにまとめることもできますが、そうすると長くなって全体の見通しが悪くなります。そこで、働きの種類によってプログラムを分割し、それぞれのプログラムを個別に理解していく、という方法をとります。ちょうど連続する文章を段落に分けると読みやすくなるような感じです。1つの段落（サブルーチン）には1つの内容（働き・機能）だけにする、という原則を守ることをお勧めします。

　メインルーチンには■という記号をつけました。サブルーチンには2種類あると考えます。1つのメインルーチンから呼び出されるサブルーチンを「従属ルーチン」と呼びことにして、これに●という記号をつけました。また、別のサブルーチンでは複数のメインルーチンから呼び出されるサブルーチンを「道具ルーチン」とよび、これに▲という記号をつけました。従属ルーチンはメインルーチンの段落に分けたものですが、道具ルーチンは各所で随時に呼び出します。このプログラムの全体は■［▲●▲］のような構造になっています。

Sub ▲図追加()

Dim 桁数%

'(1) 範囲を取得

横1 = Range("白地図!E2"): 横2 = Range("白地図!E3") '横始点：終点

横中 = (横1 + 横2) / 2 '横中間点

縦1 = Range("白地図!F2"): 縦2 = Range("白地図!F3") '縦始点：終点

縦中 = (縦1 + 縦2) / 2 '縦中間点

'(2) 距離地図ならば緯度を経度をキロメートルに換算

横1 = X座標調整(横1, 縦中): 縦1 = Y座標調整(縦1):

横2 = X座標調整(横2, 縦中): 縦2 = Y座標調整(縦2)

'(3) 軸目盛の最小値と最大値を切りのよい数に変換

With Application

桁数 = -Int(.Log10(Range("白地図!M2")))

横1 = IIf(横1 > 0, .RoundDown(横1, 桁数), .RoundUp(横1, 桁数))

横2 = .RoundUp(横2, 桁数)

桁数 = -Int(.Log10(Range("白地図!N2")))

縦1 = IIf(縦1 > 0, .RoundDown(縦1, 桁数), .RoundUp(縦1, 桁数))

縦2 = .RoundUp(縦2, 桁数)

End With

'(4) 地図チャートの横幅と縦幅を決めて図を追加

図横 = Range("白地図!E4") '地図の横幅

図縦 = 図横 \* (縦2 - 縦1) / (横2 - 横1) '地図の縦幅

ActiveSheet.ChartObjects.Add(20, 150, 図横, 図縦).Select

'図追加（x, y, 幅、高）

End Sub

(1) 横軸（経度）と縦軸（緯度）の範囲の最小値と最大値をシートから取得します。範囲の中間点を示す「横中」「縦中」は、(2)範囲を中間地点からの距離に変換したり、キロメートル換算のコサイン処理に使ったりします。

(2) X座標調整とY座標調整は距離図法のための変換をする関数です。

(3) 地図の目盛が大きい場合、切りの良い数でまとめる処理をします。数字の桁数を割り出すのには10を底とする対数計算が必要ですが、このためにExcel関数**Log10**を使用します。**Int**はVBAの関数で、数値の整数部を返します。切り上げ(**RoundUp**)と切り下げ(RoundDown)の関数はVBAにないので、Excel (Application)の関数を呼び出します。小数点以下の桁数で切り上げたり切り下げたりしますが、小数点以上の桁のときは桁数を負にすることで可能です。RoundUpは負の数を対象とするときは、絶対値を大きくするように切り上げるので（たとえばRoundUp(-125.67, -1）は-130を返します）、正と負で場合分けをしなければなりません。**IIf (*A, B, C*)**は条件***A***が満たされれば***B***の演算の結果を渡し、そうでなければ***C***の演算の結果を渡す関数です。

(4) ユーザーがセルE4に指定した地図の横幅（既知）から、地図の縦幅（未知）を計算します。このために地図の横の範囲と縦の範囲を使って次の比例式から、「図縦」を導きます。これで横幅に見合った縦が決まります。

図横：図縦＝(横2 - 横1) : (縦2 - 縦1)

　 図縦 = 図横 \* (縦2 - 縦1) / (横2 - 横1)

　最後に、方位地図の場合、チャートの高さをコサインによって調整します（→後述「簡易メルカトル図法」）。

このとき、該当する角度（緯度）として、緯度の中間値（(緯2 + 緯1) / 2：最大値＋最小値／2）を使うことにしました。最後にチャートを追加します。

Function X座標調整(W, Z)

If Range("白地図!O2") = "距離" Then W = (W - 横中) \* 111.11 \* Cos(Z \* 3.14 / 180)

X座標調整 = W

End Function

Function Y座標調整(W)

If Range("白地図!O2") = "距離" Then W = (W - 縦中) \* 111.11

Y座標調整 = W

End Function

　関数は次のような数学の関数で理解するとわかりやすいでしょう。たとえばf (x, y) = 2x + yという関数は、x=1, y=1ならば3を返します。x=1, y = 2ならば4を返します。そこでこのf(x) に対応するのがX座標調整(W, Z)です。関数は渡された値、ここでは「▲図追加」の「横1」, 「縦中」の内容を**W, Z**という変数で受け取り、その変数を使って「X座標調整」という変数の内容を規定して、関数を呼び出した変数に代入します。

　関数「X座標調整」の計算は、WとZでそれぞれ経度と緯度の数値を受けて、経度(W)とその中間値との差に111.11を掛けて、さらに緯度(Z)のコサインを掛けます。先に見たように、経度（緯度0の場合）と緯度の1度は、おおよそ111.11キロメートルになります。さらに経度は、その位置（緯度）によって長さが変わるのでコサインを掛けます。　ExcelVBAのコサイン関数(Cos)の引数はラジアン(rad: 360度=2πrad)を使います。

360度: 2πラジアン = 角度 : x ラジアン

360 \* x = 角度 \* 2π

x = 角度 \* 2π / 360

x = 角度 \* π / 180

　そこで角度（緯度）にπ(=3.14)を掛けて180で割ります。

Sub ●データセット()

Dim 経度, 緯度, 経度始!, 経度終!, 緯度始!, 緯度終!

ReDim Ov(655849, 1 To 2) '出力行列

経度始 = Range("白地図!E2"): 経度終 = Range("白地図!E3") '横始点：終点

緯度始 = Range("白地図!F2"): 緯度終 = Range("白地図!F3") '縦始点：終点

Iv = Range("A:B"): j = 2 '地図データ：初期値

For i = 2 To 655849

経度 = Iv(i, 1): 緯度 = Iv(i, 2) '横座標：縦座標

If ((経度 = Empty And 緯度 = Empty) \_

And (Ov(j - 1, 1) <> Empty Or Ov(j - 1, 2) <> Empty)) \_

Or ((経度 <> Empty And 経度 > 経度始 And 経度 < 経度終) \_

And (緯度 <> Empty And 緯度 > 緯度始 And 緯度 < 緯度終)) Then

'該当行が空白で先行する出力行が空白でないか

'または経度と緯度が空白でなく範囲内ならば

If 経度 <> Empty Then 経度 = X座標調整(経度, 緯度)

If 緯度 <> Empty Then 緯度 = Y座標調整(緯度)

Ov(j, 1) = 経度: Ov(j, 2) = 緯度

j = j + 1

End If

Next

Range("Q:R") = Ov '配列をレンジに出力

Range("Q1") = "横軸": Range("R1") = "縦軸"

Range("S1") = "出力数": Range("S2") = j - 1 '出力数

End Sub

　シート内の地図データを指定された条件によって抽出します。はじめに入力データのレンジを配列Ivに代入し、この中から抽出したデータを配列Ovに出力し、最後にこのOv配列をレンジ(Q:R)に出力します。出力数J1に表示します。

　条件は「該当行が空白で先行する出力行が空白の連続でないか」、または「経度と緯度が範囲内であるか」のどちらかです。空白(Empty)はExcelのグラフでは線を結合させない働きがあるので必要ですが、複数連続する必要はありません。

　条件に該当するデータをOvに代入しQ:R列に出力します。タイトル「横軸」「縦軸」をQ1, R1に代入し、出力数のタイトルと値をS1, S2に代入します。

Sub ▲地図記入(): 'On Error Resume Next 'エラー処理

With ActiveChart '現図について

'(0) 全体

.ChartType = xlXYScatterSmoothNoMarkers '散布図（平滑線）

.SetSourceData Source:=Range("白地図!Q:R") 'XY値のソース

.ChartTitle.Delete 'タイトルを削除

.Legend.Delete '凡例を削除

'(1) 横軸

With .Axes(xlCategory)

.MinimumScale = 横1 '横軸最小値

.MaximumScale = 横2 '横軸最大値

.TickLabelPosition = xlLow '横線の位置＝下端

If Range("白地図!M2") > 0 Then '横線の設定

.HasMajorGridlines = True '横線有

.MajorUnit = Range("白地図!M2") '横線の間隔

With .Format.Line

.Transparency = 0.1 '線の透明度

.ForeColor.ObjectThemeColor = msoThemeColorAccent2 '赤

.Weight = 1.5

.Transparency = 0.8 '線の透明度

End With

With .MajorGridlines.Format.Line

.ForeColor.ObjectThemeColor = msoThemeColorAccent2 '赤

.Transparency = 0.8 '線の透明度

End With

Else

.HasMajorGridlines = False '横線なし

.Delete '縦軸を削除

End If

End With

'(2) 縦軸

With .Axes(xlValue)

.MinimumScale = 縦1 '縦軸最小値

.MaximumScale = 縦2 '縦軸最大値

.TickLabelPosition = xlHigh '縦線の位置＝右端

If Range("白地図!N2") > 0 Then '縦線の設定

.HasMajorGridlines = True '縦線有

.MajorUnit = Range("白地図!N2") '縦線の間隔

With .Format.Line

.Transparency = 0.1 '線の透明度

.ForeColor.ObjectThemeColor = msoThemeColorAccent2 '赤

.Weight = 1.5

.Transparency = 0.8 '線の透明度

End With

With .MajorGridlines.Format.Line

.ForeColor.ObjectThemeColor = msoThemeColorAccent2 '赤

.Transparency = 0.8 '線の透明度

End With

Else

.HasMajorGridlines = False '縦線なし

.Delete '縦軸を削除

End If

End With

'(3) 国境・海岸線

With .SeriesCollection(1).Format.Line

.ForeColor.RGB \_

= RGB(Range("白地図!H2"), Range("白地図!I2"), Range("白地図!J2")) '線色

.Weight = Range("白地図!L2") '線の太さ

.Transparency = Range("白地図!K2") '線の透明度

End With

End With

End Sub

　範囲内の経度と緯度のデータ行列を使ってExcel散布図を作成します。

**With … End With**はこの範囲内にあるピリオドではじまるオブジェクトに共通する上位のオブジェクトを指します。たとえば、最初の**.ChartType**は**ActiveChart.ChartType**と同じです。Withを使わないと、すべての命令に同じオブジェクト名を書かなければなりません。一方、Withを使ったほうがコードは見やすくなり、上位のオブジェクトへの参照を1回で済ませることができるので、プログラムの実行速度も速くなります。**With … End With**は入れ子にして階層化することができます。

　マクロの記録によって得られたコードを解読して、読みやすくなるように編集し、それぞれの変数の内容をシートのセルから代入できるように書き換えました。経線と緯線の変数はレンジE2F2で指定しますが、これがゼロ(0)であれば経線と緯線を表示せず、ゼロ以上ならばそれを間隔として使います。

Sub ■全チャート削除()

On Error Resume Next 'エラー処理

ActiveSheet.ChartObjects.Delete

End Sub

　「前チャート削除」ボタンとリンクしたプログラムです。シート上のすべてのチャートを削除します。

## フォームコントロール

　シートに「白地図チャート」と「全チャート削除」ボタンを配置します。シートに配置するボタンは「開発」→「挿入」→「フォームコントロール」で最初にある「ボタン」を選択し、シート上に左上点と右下点をドラッグするとマクロを登録する画面がでます。マクロを登録した後、ボタンを選択して右クリックし「テキスト」でボタンのキャプションを変えることができます。

## 具体から抽象へ

　白地図の幅と高さを計算するとき、次のように経度と緯度の幅を考慮します。

'(4) 地図チャートの横幅と縦幅を決めて図を追加

図横 = Range("白地図!E4") '地図の横幅

図縦 = 図横 \* (縦2 - 縦1) / (横2 - 横1) '地図の縦幅

　このコードを設定するには、経度と緯度の始点と終点を使った数式を一般式の形で示さなければなりませんが、いきなり一般式を導出するのは困難です。たとえば、

三角形の面積 = 底辺 \* 高さ / 2

は一般式ですが、このような一般式に到達するためは一度具体的な例で考えてみるとよいでしょう。たとえば、底辺が5cm、高さが4cmの三角形を考えます。その面積は両者を掛け合わせて2で割って計算します（ 5 \* 4 / 2 = 10）。その理由については、上下の向きが違う三角形を2つ並べて平行四辺形を作る、という証明を思い出してください。

　白地図の幅と高さの関係は、先に見たように、求める地図の経度差と緯度差によって変化します。たとえば幅を450としたとき、高さはどのようになるでしょうか？イベリア半島の地図では次のような範囲で白地図を出力します。

|  |  |  |
| --- | --- | --- |
| イベリア半島 | 横軸 | 縦軸 |
| 始点 | -10 | 35 |
| 終点 | 5 | 45 |

　よって経度差は15度(5 - -10)、緯度差は10度(45 – 35)です。そこで、白地図は正方形ではなく、少し横長にしなければなりません。横幅と縦幅の比が15 : 10になるのです。そこでユーザーが指定した地図の横幅が450ならば、地図の縦幅をxとすると、次のような比例式ができます。

幅：高 = 450 : x = (5 - -10) : (45 – 35)

450 : x = 15 : 10

　この式から

x \* 15 = 450 \* 10

x = 450 \* 10 / 15 = 300

　となります。このように一度具体的な例で計算できたら今度は変数名を使った一般式にしましょう。

図横 : 図縦 = (横2 - 横1) : (縦2 - 縦1)

図縦 \* (横2 - 横1) = (縦2 - 縦1)

図縦 = 図横 \* (縦2 - 縦1) / (横2 - 横1)

　これで数式のコードが求められました。いかがでしょうか。初見で上の式を理解するのは困難ですが、一度具体的な例で確認してみれば納得できるでしょう。ふつう、数理の理解では一般式から具体的な数字をあてはめて計算する、という手順を踏みますが、そうすると、出発点の一般式が理解できていないのに、その操作だけを技術として身に着けることになってしまいます。式の理解には、式の要素をそれぞれ具体例で確かめてみる、という方法をお勧めします。

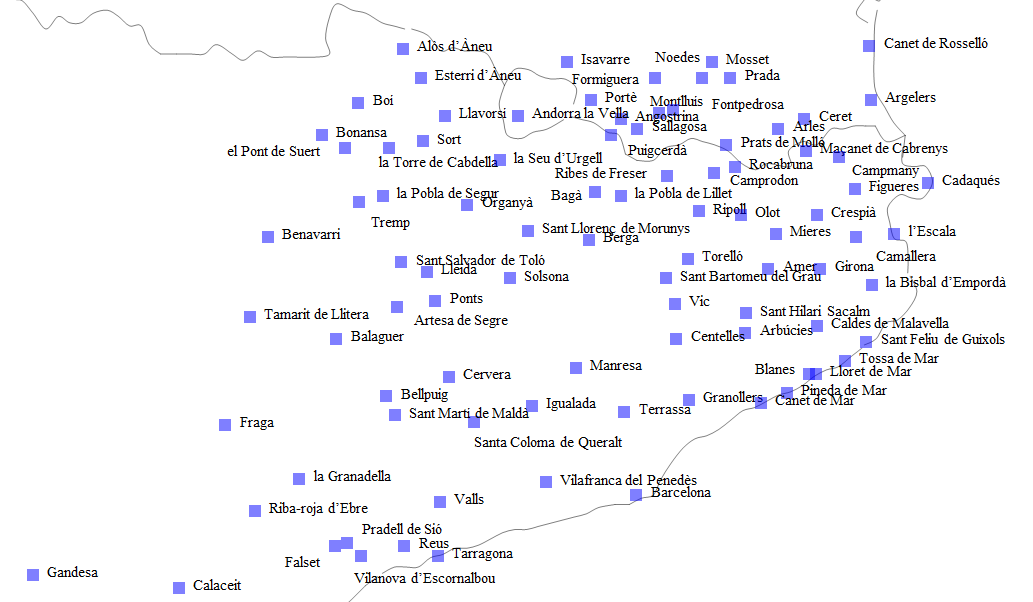
# 記入地図

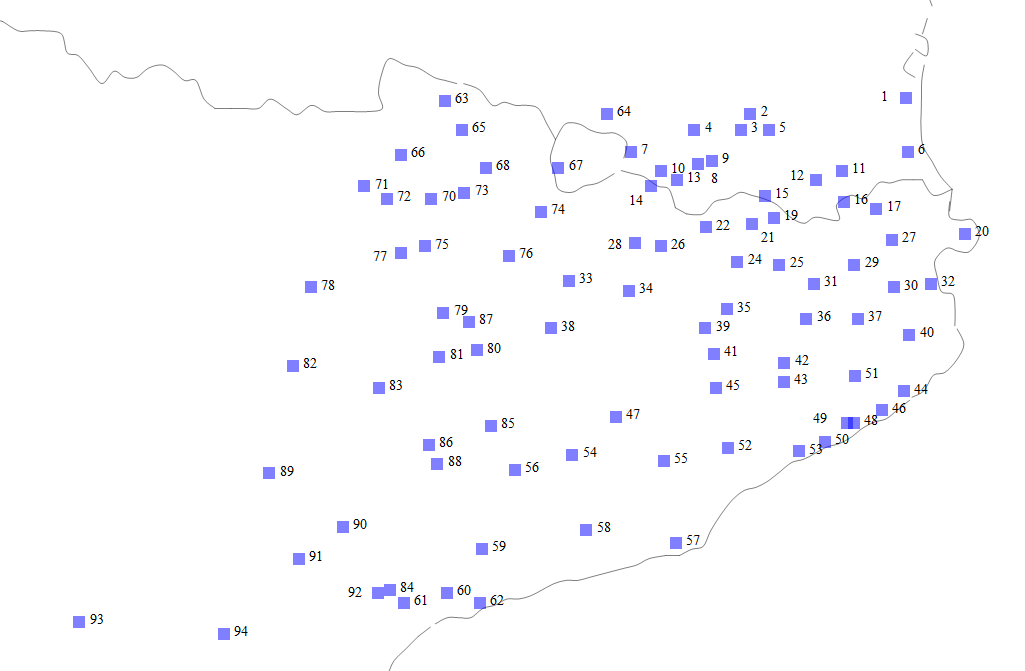
　方言学や地理地理学・歴史言語学で使う言語地図には地点にさまざまな言語特徴が記入されています。言語学者が各地を訪問して調査した結果やアンケート調査によって一斉に調べ上げた言語形式が各地に分布する様子を観察すると言語特徴の拡がりがわかるだけでなく、その分布形式から言語変化の様子を探ったり、地点や言語特徴を変数とした多変量解析が可能になります。そのような言語地図は「語形地図」「記号地図」「数量地図」の3種に分類できます。

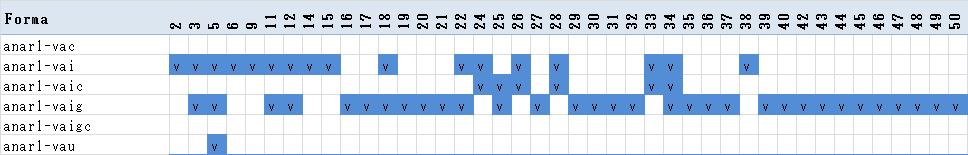
　最近ではPCの処理能力が高くなったので、Excelなどに用意した地点と語形のリストがあれば、それを白地図の上に簡単にプロットすることができるようになりました。それまでは言語地図の作製はすべて手作業でした。一枚の言語地図を手作業で書き込むには相当の時間と労力が必要でした。多くの言語地図を集めた言語地図帳の出版には数年や十数年の時間が必要でした。

　スペインの伝統的な言語地理学では語形の実相を重視します。限られた紙面の上に精密な音声表記が専門家の手でぎっしりと書きこまれています。『イベリア半島言語地図帳』や『アンダルシア言語民俗地図帳』などは語形地図の代表です。一方、最近の出版物である『メキシコ言語地図帳』では具体的な語形ではなく、それに対応した記号が各種の形や色を使って地図上にプロットされています。どちらの方法にも一長一短があり一概に優劣を決められません。語形地図に記された音声表記は個々の現象を探るための貴重な資料です。それに対して各種の記号が一面にプロットされた記号地図を見ると分布状態全体を概観することができます。

　次の図は、スペインの北東部カタルーニャ地方の一部です。20世紀のはじめにアルコベールという方言学者が各地を調査しカタルーニャ語の動詞の形態を記録しました。その調査地点をMAPASを使ってプロットしたものです。地名や語形を地図上にプロットしようとすると、スペースが重なって困難なことがあります2番目の地図は地名ではなく地名の番号を記入しました。MAPASでは地名に対応する語形の列に連続番号を記入すればよいのです。地名地図は最初の1枚だけ出力すれば十分です。その後の分析では統一してこの地名番号を使ってデータ行列を作成します（下3番目の図の第1行）。







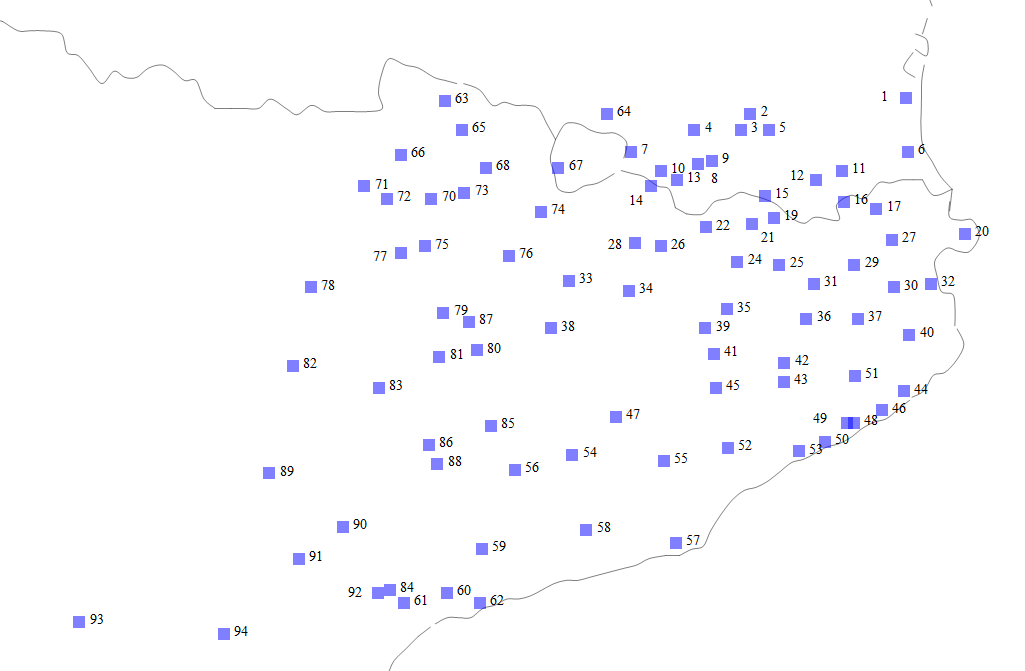
　デジタル化された言語地図では同じ資料から同時に語形地図と音声地図を出力させることができます。Excelの列に並んだ語形は対応する記号とリンクさせているからです。手作業の言語地図ではやり直しが困難ですが、システムをデジタル化すれば、リンクした記号のパラメータ（形、縦と横のサイズ、色の混合、透明度）を自由に変化させ仕上げを確認することができます。

　デジタル化にはもう一つのメリットがあります。地点と語形の二次元の配列は各種の多変量解析にかけるデータ行列にすることができるので、語形の選択反応による地点の分析や、逆に地点の選択反応による語形の分析、そして地点と語形を同時に分析することも可能です。地点の分析結果は各地点の統計量を各種のグラフに出力することができますが、統計量を記号のパラメータを数量に応じて変化させて表現することができます。数量地図は、このようにして一定の規則で変化した記号を地図上にプロットしたものです。

　先に見た白地図はExcelのチャートという形式に出力されていますが、チャートにはさらに多くの文字列や記号を含めることができます。このチャートを扱うプログラミングはそれほど難しくはありません。文字列や記号の種類を決めて、それらを配置する座標を計算すればよいからです。はじめは具体的な例を示す「定数」で実験し、実験の結果がよければ定数を「変数」に変えてプログラムを一般化させます。

　小学校の算数はたとえば「1 + 2」などのような定数を扱う世界です。中学校の数学から「a + b」などのような文字式の計算を学習しました。このaやbなどの文字をプログラミングでは「変数」と呼びます。実際のプログラミングで、いきなり変数を使うよりも小学校レベルの定数で理解してから中学校レベルの変数に進む方がよいと思います。変数は抽象的なのでイメージすることがむずかしいときがあるからです。もっとも比較的簡単な計算では最初から変数を使ってもよいでしょう。

　最後にプログラムのコードをユーザーが使いやすいようにボタンなどにリンクさせてシステム全体のデザインを考えます。MAPASのシステムはとてもシンプルなのですが、ユーザーはそれでも「なんだかむずかしそう、私には使えない」という印象をもつ可能性があります。そのために避けられてしまうと残念です。いろいろな工夫をしてわかりやすく使いやすいシステムにしたいと思います。

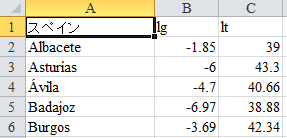


## (0) 地点座標

　はじめに次のような地点、経度、緯度のデータを準備します。

　経度と緯度を調べるには地図帳を見るほかに、次の方法があります。Google Mapで地点を選択し、「この場所について」をクリックすると、検索窓に緯度、経度の順に表示されます。これをエディターやエクセルで編集します。緯度・経度の順を経度・緯度の順に変え、精度を小数点以下2桁にしてください。精度は高いままでもかまいませんが、順番は必ず変えてください。

　このデータが記入地図全体で共通に使われます。

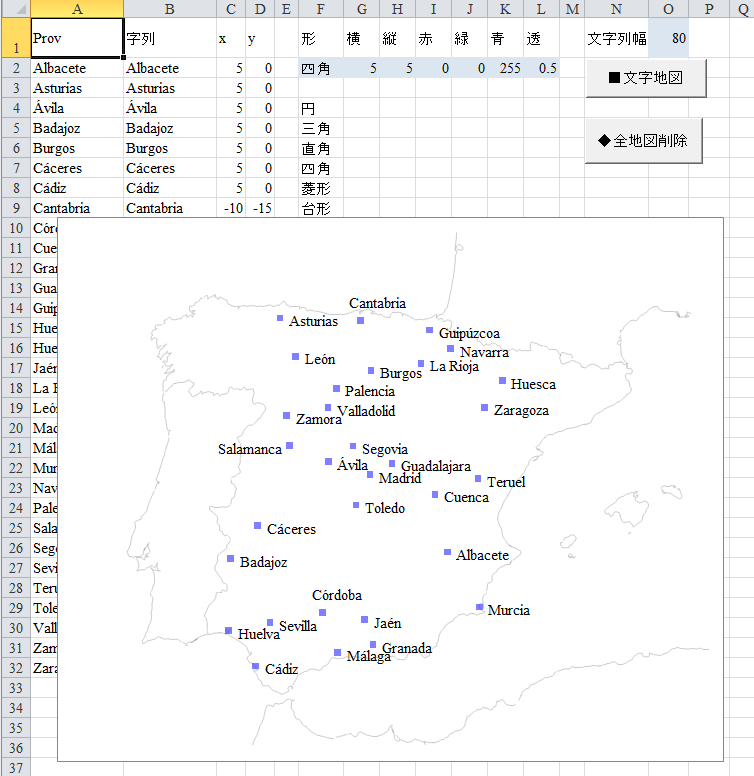


(…)

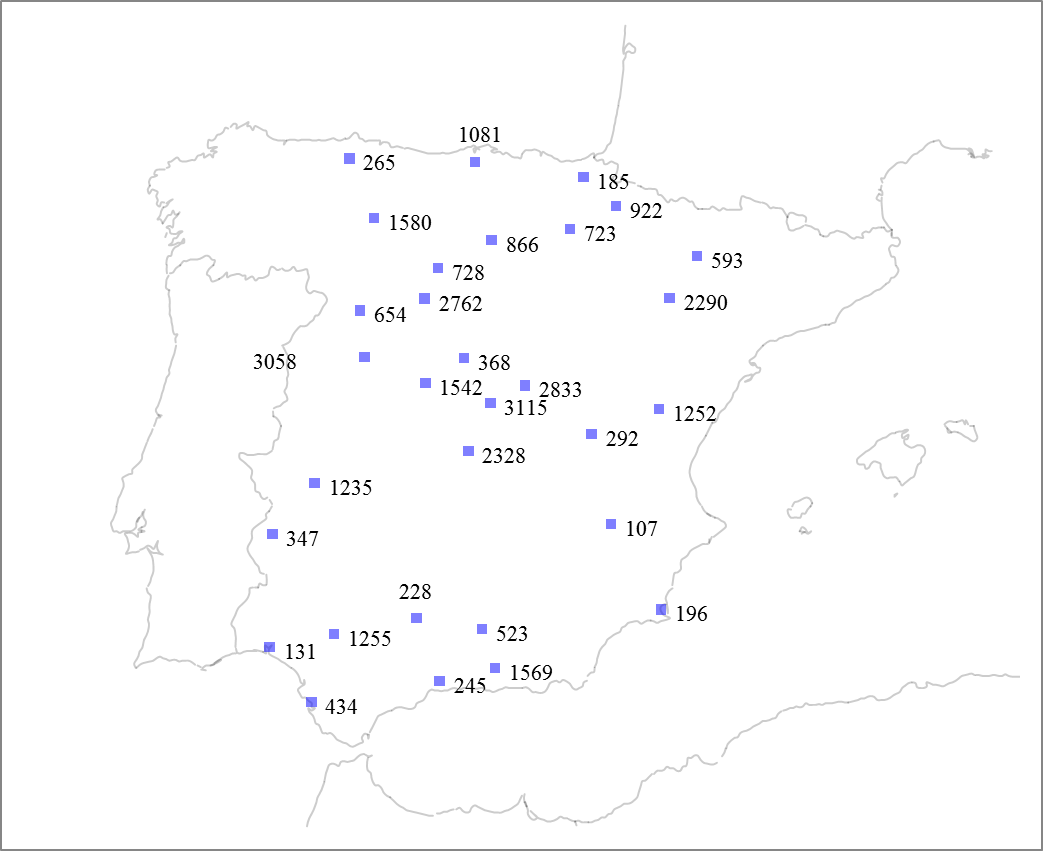
A列に地点番号などの数値を使うときは、そのデータ型に注意してください。記入地図で同じ番号を使ってもデータ型が文字になっていると、座標を認識しません。単なる1, 2, 3, …などの番号ではなくL001, L002などの記号を使うか、または、数値を文字型に変換するか、または以下の文字地図、記号地図、数量地図のA列の数字のデータ型を「地点座標」のデータ型に合わせるようにしてください。文字型から数値型に変換するときは、「形式を選択して貼り付け」で「加算貼り付け」をしてください。

## (1) 文字地図

　「文字地図」は白地図に地名や語形を記入します[[18]](#footnote-18)。



　A-D列に地名、該当する文字列、x位置、y位置を記入し、「■文字地図」ボタンを押すと先に用意した白地図上に地点と文字（地名）が表示されます。この例ではB列に文字として地名をコピーしてありますが、ここに文字や数字を入れれば、それが地図に記入されます（次図）。



　F2のそれぞれのセルを選択すると、プルダウンのリストが現れますから、その中から選択してください。

　「図」のそれぞれの形の一覧は次の「記号地図」の凡例で見ることができます。「横」と「縦」の長さの単位は「ポイント」と呼ばれるもので、1ポイント＝0.3528mmです。文字列幅(O1)はとくに設定の変更を必要としません。

　地名や語形が重なってしまった場合は、そのテキストボックスを選択してドラッグしたり、テキストボックスの大きさを変更したりして調節することができます。適当な線をつけることも可能です。

　または、シートのx, y位置で調整すれば、この値は保存されてていますから、毎回個別に調整する必要がなくなります。先の図のCantabriaを参照してください。

## プログラム

Sub ■文字地図()

Call ▲図追加: Call ●形と文字プロット: Call ▲地図記入

End Sub

　「■文字地図」ボタンとリンクしています。

Sub ●形と文字プロット(): 'On Error Resume Next 'エラー処理

形 = Range("F2"): 形横 = Range("G2"): 形縦 = Range("H2")

赤 = Range("I2"): 緑 = Range("J2"): 青 = Range("K2")

透 = Range("L2"): 例横 = Range("O1")

For i = 2 To Range("A1").End(xlDown).Row

x = Application.VLookup(Cells(i, 1), Range("地点座標!A:C"), 2, False)

y = Application.VLookup(Cells(i, 1), Range("地点座標!A:C"), 3, False)

Call ▲プロット位置: Call ▲形記入

x = x - 8 + Cells(i, 3) + 形横 '-8はテキストボックス

y = y - 8 + Cells(i, 4) + 形縦 / 2

文字列 = Cells(i, 2)

Call ▲文字列記入

Next

End Sub

　それぞれの変数に値を代入し、チャートに図と文字列を記入するための位置(x, y)を計算し、「▲形記入」で記号の形を記入した後、「▲文字列記入」でテキストボックスに記入します。経度と緯度はExcel関数VLookupを使っています。

Sub ▲プロット位置()

x = X座標調整(x, y): y = Y座標調整(y)

x = 図横 \* (x - 横1) / (横2 - 横1) '地点のX位置

If Range("白地図!M2") > 0 Then '横軸があれば横座標を微調整

x = x \* ActiveChart.PlotArea.InsideWidth / ActiveChart.PlotArea.Width

End If

y = 図縦 \* (縦2 - y) / (縦2 - 縦1) ' - 形縦 / 2 '地点のY位置

If Range("白地図!N2") > 0 Then '縦軸があれば縦座標を微調整

y = y \* ActiveChart.PlotArea.InsideHeight / ActiveChart.PlotArea.Height

End If

End Sub

　形をチャートにプロットするためにその座標を求めます。はじめにX座標調整とY座標調整によって距離図法の場合には単位を変えます。xの位置はチャートの幅全体の中で、その開始点 / 範囲を掛けた位置になります。さらに形の横幅の半分を引いて少し左に寄せます。横軸があると、チャートの描画部分が少し小さくなるので微調整します。

　Y座標も同様ですが、縦軸は上から座標が始まるので、 (縦2 - y)という式を使います。

Sub ▲形記入(): 'On Error Resume Next 'エラー処理

With ActiveChart.Shapes

Select Case 形

Case "円":

.AddShape(msoShapeOval, x, y, 形横, 形縦).Select

Case "三角":

.AddShape(msoShapeIsoscelesTriangle, x, y, 形横, 形縦).Select

Case "直角":

.AddShape(msoShapeRightTriangle, x, y, 形横, 形縦).Select

Case "四角":

.AddShape(msoShapeRectangle, x, y, 形横, 形縦).Select

Case "菱形":

.AddShape(msoShapeDiamond, x, y, 形横, 形縦).Select

Case "十字":

.AddShape(msoShapeCross, x, y, 形横, 形縦).Select

Case "台形":

.AddShape(msoShapeTrapezoid, x, y, 形横, 形縦).Select

Case "平行":

.AddShape(msoShapeParallelogram, x, y, 形横, 形縦).Select

Case "五角":

.AddShape(msoShapeRegularPentagon, x, y, 形横, 形縦).Select

Case "六角":

.AddShape(msoShapeHexagon, x, y, 形横, 形縦).Select

Case "上矢":

.AddShape(msoShapeUpArrow, x, y, 形横, 形縦).Select

Case "下矢":

.AddShape(msoShapeDownArrow, x, y, 形横, 形縦).Select

Case "右矢":

.AddShape(msoShapeRightArrow, x, y, 形横, 形縦).Select

Case "左矢":

.AddShape(msoShapeLeftArrow, x, y, 形横, 形縦).Select

Case "星4":

.AddShape(msoShape4pointStar, x, y, 形横, 形縦).Select

Case "星5":

.AddShape(msoShape5pointStar, x, y, 形横, 形縦).Select

Case "星6":

.AddShape(msoShape6pointStar, x, y, 形横, 形縦).Select

Case "星7":

.AddShape(msoShape7pointStar, x, y, 形横, 形縦).Select

End Select

End With

With Selection.ShapeRange '形の属性

.Fill.ForeColor.RGB = RGB(赤, 緑, 青)

.Fill.Transparency = 透

.Line.Visible = msoFalse '枠線なし

End With

End Sub

End Sub

　それぞれの選択された形を位置(x, y)と横幅（「形横」）、縦幅（「形縦」）にしたがって記入します。図の色を「赤」、「緑」、「青」によって指定し、透明度を「透」によって指定します。枠線をつけません。

Sub ▲文字列記入(): 'On Error Resume Next 'エラー処理

ActiveChart.Shapes.AddTextbox(msoTextOrientationHorizontal, \_

x, y, 例横 - 10, 15).Select 'テキストボックス

With Selection

.Characters.Text = 文字列

.Characters.Font.Size = 11

.Characters.Font.Name = "Times New Roman"

.AutoSize = True 'オートサイズ

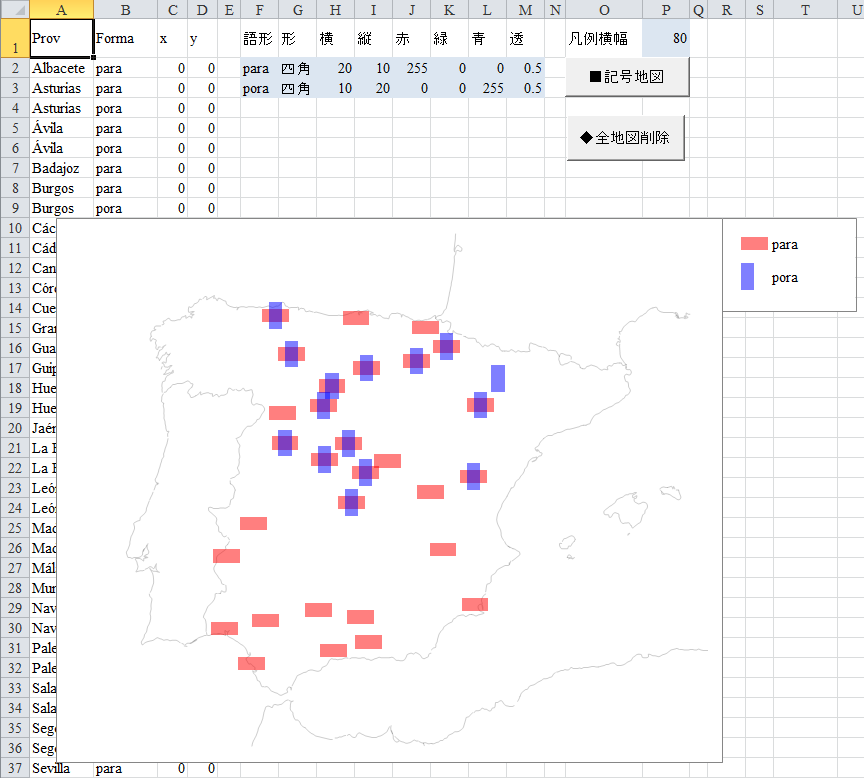
End With

End Sub

　位置（x, y）と横幅、縦幅にしたがって、テキストボックスに文字列を記入します。縦幅は一定(15)にしましたが、横幅は文字列の長さに応じて自動で変化させます(**AutoSize = True**)。

## (2) 記号地図

　白地図に各語形に対応する記号を記入します。同一地点に複数の語形があるときは重ね書きをします。同時に記号リストの索引を出力します。



　ユーザーは上の青く塗ったF:M列に各語形に対応する図、縦と横の長さ、指定色を記入します。この表にE:K列がVLOOKUP関数によってリンクします。Y列は「形」のリストのデータの入力規則に使うリストの内容です。

## プログラム

Sub ■記号地図チャート()

Call ▲図追加: Call ●記号プロット: Call ▲地図記入: Call ●記号凡例

End Sub

Sub ●記号プロット(): 'On Error Resume Next 'エラー処理

For i = 2 To Range("A1").End(xlDown).Row '行数

形 = Application.VLookup(Cells(i, 2), Range("F:M"), 2, False)

形横 = Application.VLookup(Cells(i, 2), Range("F:M"), 3, False)

形縦 = Application.VLookup(Cells(i, 2), Range("F:M"), 4, False)

赤 = Application.VLookup(Cells(i, 2), Range("F:M"), 5, False)

緑 = Application.VLookup(Cells(i, 2), Range("F:M"), 6, False)

青 = Application.VLookup(Cells(i, 2), Range("F:M"), 7, False)

透 = Application.VLookup(Cells(i, 2), Range("F:M"), 8, False)

x = Application.VLookup(Cells(i, 1), Range("地点座標!A:C"), 2, False)

y = Application.VLookup(Cells(i, 1), Range("地点座標!A:C"), 3, False)

Call ▲プロット位置

x = x + Cells(i, 3) - 形横 / 2

y = y + Cells(i, 4) - 形縦 / 2

Call ▲形記入

Next

End Sub

　シートからそれぞれの変数の値を取得し、それにしたがってサブルーチン「▲プロット位置」と「 Call ▲形記入」を呼び出します。

Sub ●記号凡例()

'On Error Resume Next 'エラー処理

例縦 = 0: 例横 = Range("W1"): 図横 = 0 '凡例の縦幅：横幅

For i = 2 To Range("M1").End(xlDown).Row '行数

例縦 = 例縦 + Cells(i, 16) + 10 '縦の累計

If Cells(i, 15) > 図横 Then 図横 = Cells(i, 15) '幅の最大値

Next

図横 = 図横 + 例横

ActiveSheet.ChartObjects.Add(520, 150, 図横, 例縦 + 20).Select

'図を追加（x, y, 幅、高）

y = 0: 図縦 = 0: 例縦 = 0: 形縦 = 0

For i = 2 To Range("M1").End(xlDown).Row '行数

x = 10: 例縦 = 例縦 + 形縦 + 10: y = 例縦

形 = Cells(i, 14): 形横 = Cells(i, 15): 形縦 = Cells(i, 16)

赤 = Cells(i, 17): 緑 = Cells(i, 18): 青 = Cells(i, 19)

透 = Cells(i, 20)

Call ▲形記入

x = 10 + 図横: y = 例縦 + 形縦 / 2 - 10: 文字列 = Cells(i, 13)

Call ▲文字列記入

Next

End Sub

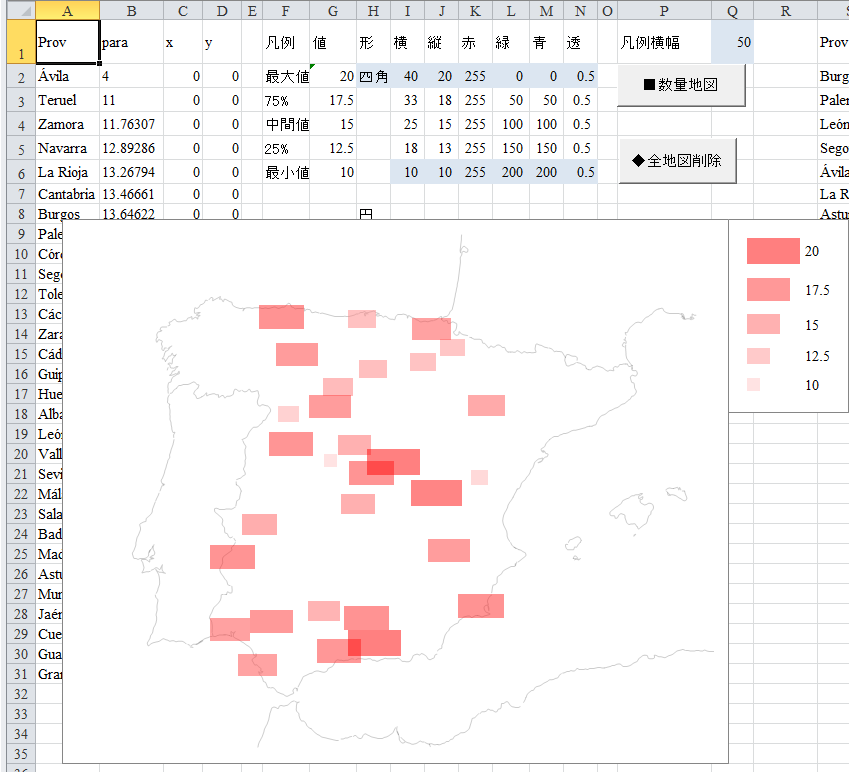
　凡例の内容を出力します。「例縦」は凡例の高さを示し、個々の形の縦幅の累計から求めます。「図横」は文字列の開始位置を示し、図の幅の最大値から求めます。幅は凡例の幅を示します。「図横」と「例縦」のデータを使って、凡例のチャートを出力し、次のFor … Nextでそれぞれの図と文字列を出力します。文字列のy位置は図の縦の長さを考慮します（**形縦 / 2**）。

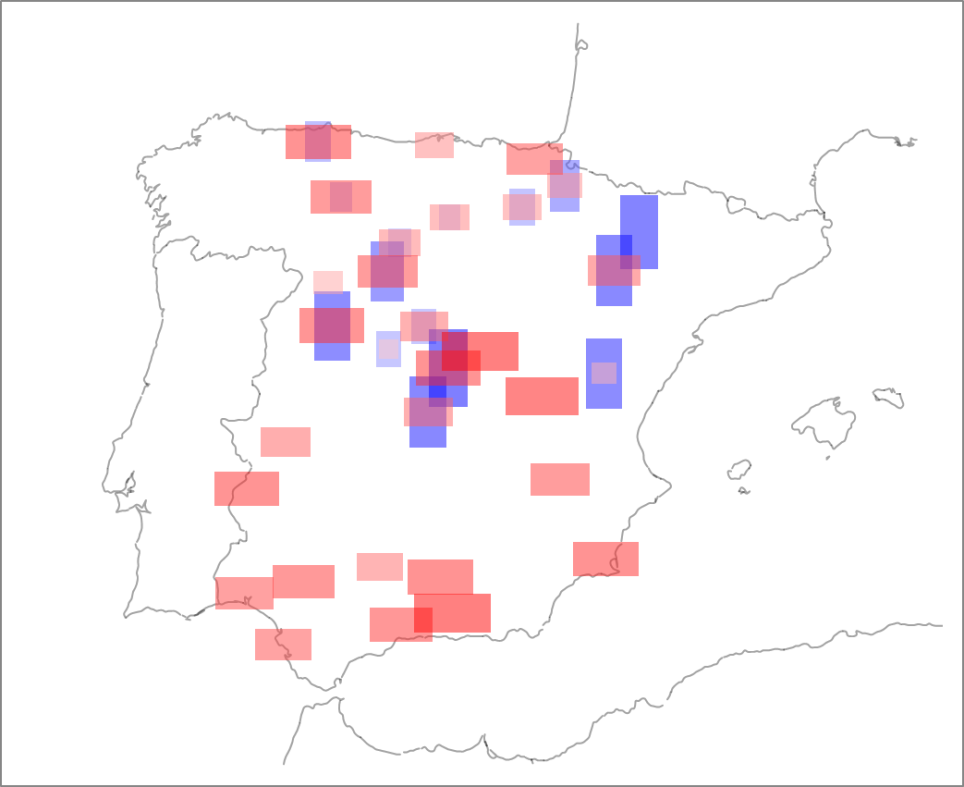
## (3) 数量地図

　各地点に属する何らかの数値を形の大きさや色で連続的に段階を表示します。凡例では最大値と最小値の間を5段階に分けていますが、地図上の記号は範疇化せずに、連続的に大きさや色を変化させます。

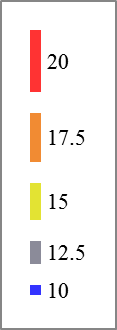
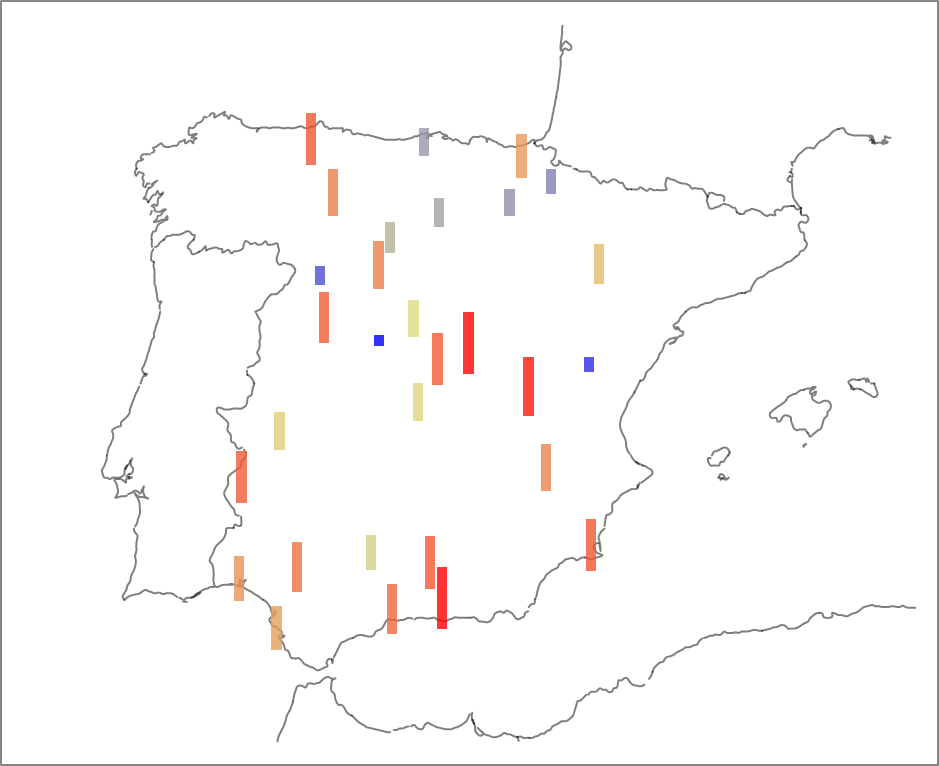
　次のように最大値と最小値の横幅、縦幅、色のパラメータ、透明度を記入すると、各地の数量データはこの尺度に従って、大きさや色を決定し、出力されます。

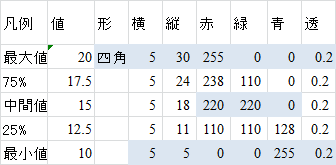
　その次の図は、2つの地図を重ね合わせたものです。地図はチャート内のプロットエリアに、線や図形やテキストボックスなどを書き込めます。そこに他の地図のプロットエリアも重ねることができるからです。





中間値を調節することによって、次のように3色のグラデーションを作ることができます。





## プログラム

Sub ■数量地図チャート()

Call ▲図追加: Call ●数量プロット: Call ▲地図記入: Call ●数量凡例

Range("A1").Select 'セル選択

End Sub

Sub ●数量プロット(): On Error Resume Next 'エラー処理

Dim 数量!, 最大!, 中間!, 最小!, 横大!, 横小!, 縦大!, 縦小!

Dim 赤大!, 赤中!, 赤小!, 緑大!, 緑中!, 緑小!, 青大!, 青中!, 青小!, 透大!, 透小!

最大 = Range("G2"): 中間 = Range("G4"): 最小 = Range("G6")

形 = Range("H2"):

横大 = Range("I2"): 横小 = Range("I6")

縦大 = Range("J2"): 縦小 = Range("J6")

赤大 = Range("K2"): 赤中 = Range("K4"): 赤小 = Range("K6")

緑大 = Range("L2"): 緑中 = Range("L4"): 緑小 = Range("L6")

青大 = Range("M2"): 青中 = Range("M5"): 青小 = Range("M6")

透大 = Range("N2"): 透小 = Range("N6")

For i = 2 To Range("A1").End(xlDown).Row '行数

数量 = Cells(i, 2)

If 数量 > 最大 Then 数量 = 最大

If 数量 < 最小 Then 数量 = 最小

形横 = (数量 - 最小) \* (横大 - 横小) / (最大 - 最小) + 横小

形縦 = (数量 - 最小) \* (縦大 - 縦小) / (最大 - 最小) + 縦小

If 数量 > 中間 Then '数量が中間値より大であれば

赤 = (数量 - 中間) \* (赤大 - 赤中) / (最大 - 中間) + 赤中

緑 = (数量 - 中間) \* (緑大 - 緑中) / (最大 - 中間) + 緑中

青 = (数量 - 中間) \* (青大 - 青中) / (最大 - 中間) + 青中

Else '数量が中間値より小であれば

赤 = (数量 - 最小) \* (赤中 - 赤小) / (中間 - 最小) + 赤小

緑 = (数量 - 最小) \* (緑中 - 緑小) / (中間 - 最小) + 緑小

青 = (数量 - 最小) \* (青中 - 青小) / (中間 - 最小) + 青小

End If

透 = (数量 - 最小) \* (透大 - 透小) / (最大 - 最小) + 透小

x = Application.VLookup(Cells(i, 1), Range("地点座標!A:C"), 2, False)

y = Application.VLookup(Cells(i, 1), Range("地点座標!A:C"), 3, False)

Call ▲プロット位置

x = x + Cells(i, 3) - 形横 / 2

y = y + Cells(i, 4) - 形縦 / 2

Call ▲形記入

Next

End Sub

　最大値と最小値に対応する形、横幅、縦幅、色、透明度を該当する数量に合わせて段階的に表示します。たとえば、横の値は次の式によって求めます。

横 = (数量 - 最小) \* (横大 - 横小) / (最大 - 最小) + 横小

これは次の比例式から導出されます。

(横 – 横小)：(横大－横小) ＝ (数量―最小)：(最大―最小)

Sub ●数量凡例(): 'On Error Resume Next 'エラー処理

例縦 = 0: 例横 = Range("Q1"): 図横 = 0 '凡例の縦幅：横幅

For i = 2 To Range("M1").End(xlDown).Row '行数

例縦 = 例縦 + Cells(i, 10) + 10 '縦の累計

If Cells(i, 9) > 図横 Then 図横 = Cells(i, 9) '幅の最大値

Next

図横 = 図横 + 例横

ActiveSheet.ChartObjects.Add(520, 150, 図横, 例縦 + 20).Select

'図を追加（x, y, 幅、高）

y = 0: 図縦 = 0: 例縦 = 0: 形縦 = 0

For i = 2 To Range("F1").End(xlDown).Row '行数

x = 10: 例縦 = 例縦 + 形縦 + 10: y = 例縦

形 = Range("H2"): 形横 = Cells(i, 9): 形縦 = Cells(i, 10)

赤 = Cells(i, 11): 緑 = Cells(i, 12): 青 = Cells(i, 13)

透 = Cells(i, 14)

Call ▲形記入

x = 10 + 図横: y = 例縦 + 形縦 / 2 - 10: 文字列 = Cells(i, 7)

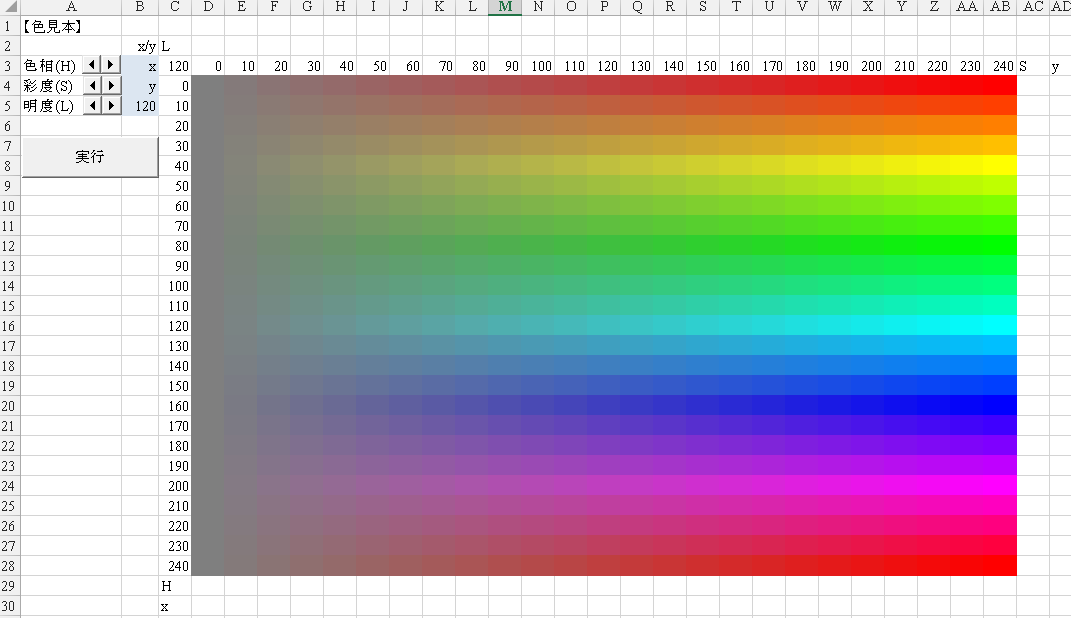
Call ▲文字列記入

Next

End Sub

　先の「●記号凡例」と同様の処理です。

# 色見本



色彩を構成する3つの変数、色相(H), 彩度(S), 明度(L)の変化と、出力の関係を見るために、「色見本」シートを作りました。[B3], [B4], [B5]に、x, y, 0~240を入れて実験してください。[A3], [A4], [A5]のスピンボタンを変えると数値が変化します。プログラムの説明はHPの「色彩」COLORを見てください。

H.Ueda 2014/01/11

1. 大きな研究計画では、一度小さな規模のデータで全体のプロセスを流してみることを勧めます。次にさらに大きな規模のデータを使ったときに、全体の流れがわかって作業を進めると先の経験が役立ち、安心して実行できて自信もつきます。逆に、いきなり大きな規模で計画を実行しようとすると壁にぶつかったとき、解決するための予備知識がないので非常に困難になります。 [↑](#footnote-ref-1)
2. MAPASでは世界全体の国境線と海岸線の座標を用意し、そこから必要な範囲を設定して、その座標を抽出します。とくに、このような一般的なシステムにする必要がなく、特定の小さな地域だけを対象にするのであれば、ここで紹介するNOAAのデータをそのままエクセルのAB列にコピーして使うことも可能です。そのほうが簡単にすぐ白地図を描くことができます。 [↑](#footnote-ref-2)
3. westernmost longitude=-180, easternmost longitude = 180, upper latitude = 90, lower latitude = -90を入力し、SUBMIT – Extract the Coast Line Fileのボタンを押すと、該当する地図の画面が表示されます。その画面内のHere is the coastline data you extracted:の左にあるリンクを押すとデータがテキストファイルで表示されます。これを全選択し、エディターなどにコピーします。 [↑](#footnote-ref-3)
4. 先の実験では散布図（直線）を使って座標と点と線の関係を説明しましたが、地図を描画するときは直線でなく平滑線で結ぶ方が線がスムーズになります。 [↑](#footnote-ref-4)
5. 縦軸の目盛を選択→右クリック→「軸の書式設定」→「軸のオプション」の最小値と最大値のそれぞれ「固定」を選択し数値を入力します。 [↑](#footnote-ref-5)
6. (1) 縦の軸の目盛を選択して右クリック→「軸の書式設定」→「軸のオプション」の「最小値」を「固定」で35とし、「最大値」も「固定」45とします。(2)も同様にして最小値を-10、最大値を5とします。 [↑](#footnote-ref-6)
7. 目盛線の位置選択して右クリック→「データ系列の書式設定」→「線の色」を「線（単色）」とし、透過性を50%にする。「線のスタイル」→「幅」を1ptとします。うまくいかないときは目盛線の他の場所を選択しれからやり直します。 [↑](#footnote-ref-7)
8. チャートの右下にマウスをあてハンドルで操作します。 [↑](#footnote-ref-8)
9. サクラエディターで「編集」→「整形」→「連続した重複行の削除(uniq)」。 [↑](#footnote-ref-9)
10. Excelシートの列をABCのアルファベットではなく、数字にすることも可能ですがあまり一般的な使用法ではありません。 [↑](#footnote-ref-10)
11. このとき、PCはフルに仕事をしているので、ユーザーは画面をクリックしたりして他の命令をPCに与えないようにしましょう。時間を十分にかけてやれば、いずれ終了する作業ですｓ。 [↑](#footnote-ref-11)
12. 一般の「二分検索」binary searchは検索範囲がわかっている場合に使われます。ここで扱っている座標データの適切なステップ数の範囲は未定なので、2倍ずつしながらNG値を探していかなければなりません。 [↑](#footnote-ref-12)
13. グリニッジを通る子午線と赤道が交差する点で、赤道ギニア共和国の近くになります。 [↑](#footnote-ref-13)
14. 現在ではメートルは真空中に光が進む距離を使って定義されます。 [↑](#footnote-ref-14)
15. 経度と緯度は「表紙」の世界地図画像やGoogle の地図検索を使って調べてください。Google の地図検索では、地図にカーソルを置き、右クリックし、「この場所について」を選択すると、検索欄に緯度、経度の順で数値が表示されます。 [↑](#footnote-ref-15)
16. シートの数値情報は「地名・語形地図」「記号地図」「数量地図」で使用します。 [↑](#footnote-ref-16)
17. したがってメルカトル図法では北緯や南緯が非常に高いところでは経線があまりに長くなるので表示できません。経度と緯度を単純な座標とする正方図法や緯度に応じて横座標をを短くする距離図法ならば表示が可能です。 [↑](#footnote-ref-17)
18. 白地図を描くために「白地図」シートのデータを利用しますが、そこに地図が描かれている必要はありません。一度白地図の出力を確認したら、地図は削除してください。地図はExcelシートのデータとリンクしているので、Excelの動作を遅くします。 [↑](#footnote-ref-18)