Visualiser avec Mathematica?

Avant et après tout traitement le chercheur doit visualiser ses données et ses résultats. Ce qui revient à construire des modèles graphiques. Dans ce notebook nous présentons quelques outils utilisés pour visualiser les textes et les données, en excluant la cartographie, qui sera abordée dans un autre notebook.
Le géographe doit donc visualiser :

-des textes lors d'approches qualitatives,
-des données structurelles,
-des données temporelles,

et parfois élaborer des graphiques dynamiques, ce qui est possible sur un écran d'ordinateur.

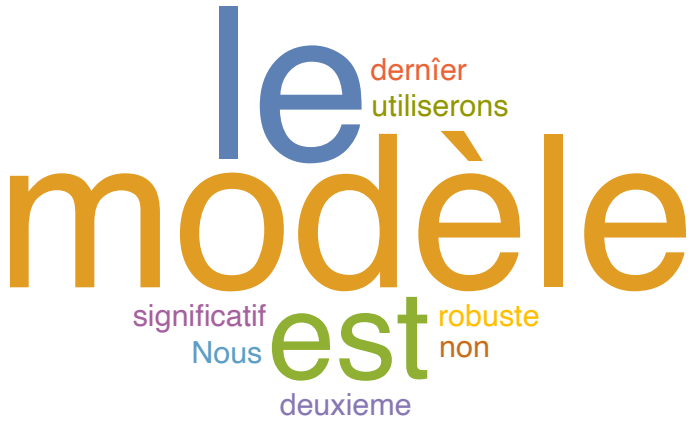
Visualiser des données textuelles

Le nuage de mots

Une première visualisation se fait avec la fonction **WordCloud[]**. Voici un premier exemple très simple avec un texte très bref :

In[21]:=
```
text = "le modèle est robuste. le deuxieme modèle
     est non significatif. Nous utiliserons le dernîer modèle";
WordCloud[
 nuage de mot
 text]
```

Out[22]=



Il n'est pas interdit de traiter ainsi des textes de célèbres géographes. Puis, dans le précédent cahier nous avons vu comment se servir des bases de données offertes par Mathematica. Nous pouvons les utiliser. La première ligne d'instruction collecte le nom et la population des États. La deuxième ligne en donne une visualisation textuelle.

In[1]:=
```
data = EntityValue[CountryData[], {"Name", "Population"}];
```
valeur d'entité — données de pays

```
WordCloud[data]
```
nuage de mot

Out[ ]=

Comme pour tous les graphiques une vingtaine d'options permettent de modifier la forme, les couleurs, la présentation de ce graphique. Voici un exemple emprunté à l'aide que Mathematica fournit pour cette fonction:

In[ ]:=
```
WordCloud[data, PlotTheme -> "Marketing"]
```
nuage de mot — thème de tracé

Out[ ]=

Ce type de graphique peut s'appliquer à tout objet Mathematica, notamment du texte ou des images. Dans l'aide de cette fonction, le lecteur peut trouver le bel exemple d'une application plus

originale relative aux tableaux de Picasso.

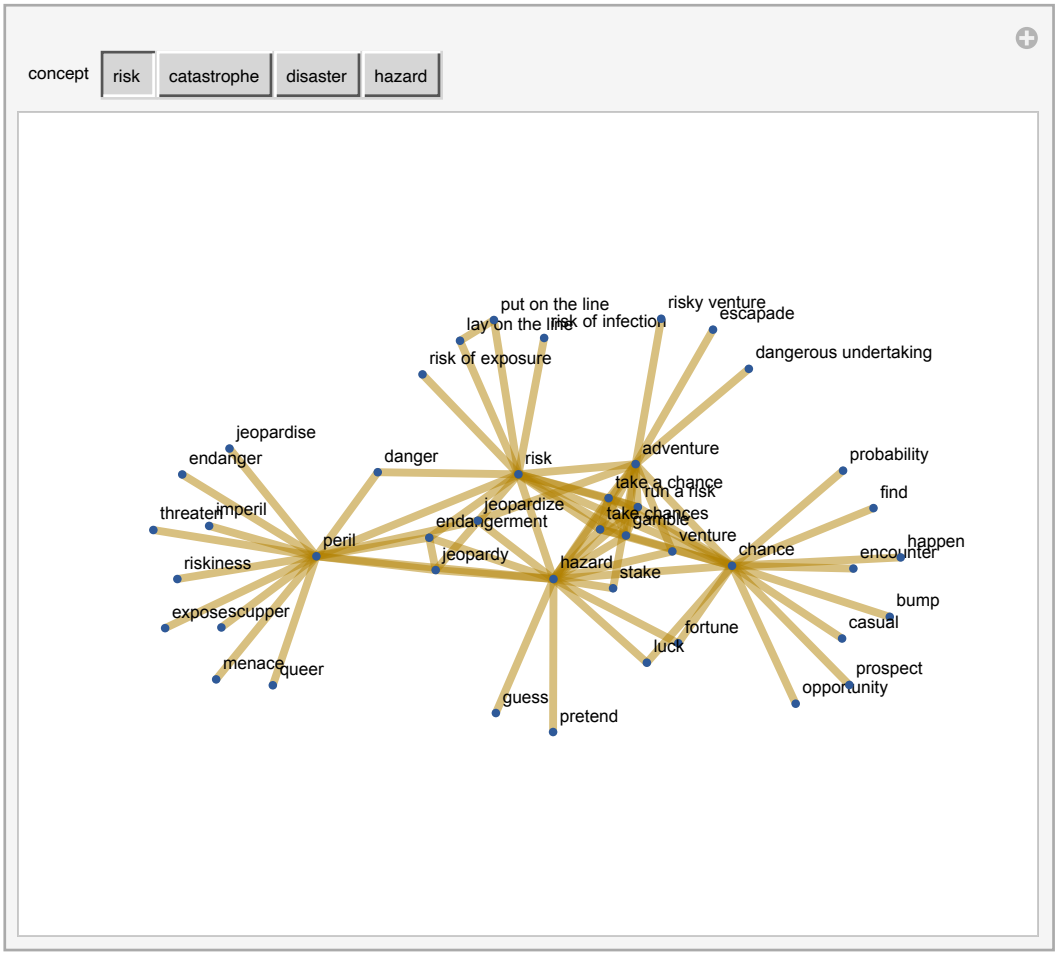
Des graphes pour construire des modèles conceptuels

Lors d'une première phase de recherche, il est souvent utile de préciser le vocabulaire disponible et les liens qui unissent les concepts qui vont être mobilisés. Le géographe construit des modèles conceptuels qualifiés parfois d'ontologies. Pour visualiser et approfondir ces modèles il utilise la théorie des graphes. Le petit programme ci-dessous construit un tel modèle pour comparer les univers du risque, de la catastrophe et du hasard. Le modèle est construit en collectant les synonymes de chaque mot dans le dictionnaire Mathematica. Chaque ensemble de mots constitue un réseau (edges) qui est représenté par un graphe (style). La fonction **Manipulate[]**, que nous rencontrons souvent , permet de passer d'un terme à l'autre sur la fenêtre affichée. Ce graphe, sauvegardé dans le fichier g peut être analysé avec tous les outils de la théorie des graphes. En jouant avec le programme, en cliquant sur les différents concepts, le lecteur comprend immédiatement que l'univers conceptuel du risque est nettement plus riche que celui de la catastrophe.

```
Manipulate[
 Module[{edges, style},
  edges = Flatten[Rest[NestList[
      Union[Flatten[Thread[# → WordData[#, "Synonyms", "List"]] & /@ Last /@ #]] &,
      {"" → w}, 2]]];
  style = {VertexLabels → "Name", ImagePadding → 45,
    ImageSize → {500, 400}, GraphStyle → "ThickEdge",
    EdgeStyle → Directive[Opacity[0.5], Thickness[0.01], Hue[0.12, 1, .7]]};
  g = SimpleGraph[UndirectedGraph[Graph[edges], style]]],

 {{w, "risk", "concept"}, {"risk", "catastrophe", "hazard"}}]
```

Out[ ]=



Visualiser des données structurelles

Le géographe étudie souvent des phénomènes en un temps bien défini, par exemple les températures dans les stations françaises le 10 décembre 2020, la population des communes d'une région au recensement de 1881, et mille autres données. Pour représenter ces informations, Mathematica dispose d'un grand nombre d'outils graphiques plus ou moins classiques. Chaque outil comprend un très grand nombre d'options. En voici une liste non exhaustive : **ListPlot, ListLinePlot, ListLogPlot, ListLogLogPlot, ListPolarPlot**. Pour des représentations en deux dimensions, le géographe mobilisera les fonctions **ListDensityPlot, ListCountourPlot, ReliefPlot, ArrayPlot, MatrixPlot**, et en trois dimensions, citons **ListPointPlot3D, ListSurfacePlot3D**. À cette liste il convient d'ajouter les graphiques classiques à barres (**BarChart**), secteur (**PieChart, SectorChart**), les diagrammes statistiques (**Histogram, SmoothHistogram, BoxWiskerChart, QuantilePlot, ProbabilityPlot**).

Pour chacun de ces graphiques, de multiples options sont disponibles. Les plus utiles concernent la légende, les grilles, les axes, les couleurs et les marqueurs. Ci-dessous, trois exemples de graphiques simples. Nous créons deux séries de données avec la fonction **RandomInteger[]**. Puis, nous réalisons un graphique sans option pour la série data1, un graphique avec quatre options pour construire un graphique plus lisible de la même série 1, un graphique incluant les deux séries et enfin une représentation sous forme de camembert.

In[•]:=
```
ClearAll["Global`*"]
```
efface tout

```
data1 = RandomInteger[10, 30];
```
entier aléatoire

```
data2 = RandomInteger[5, 30];
```
entier aléatoire

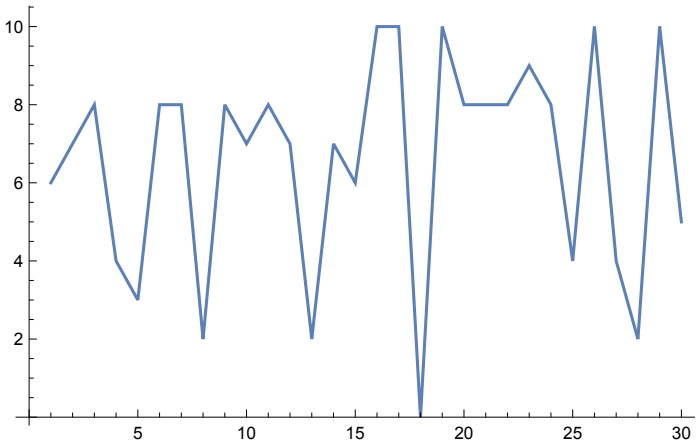
```
ListLinePlot[data1]
```
tracé de liste de ligne

```
ListLinePlot[data1, PlotLegends → "Pays1",
  AxesOrigin → {1, 0}, AxesLabel → {x, y}, Filling → Axis]
```
tracé de liste de ligne — légendes de tracé — origine des axes — titre d'axe — remplissage — axe

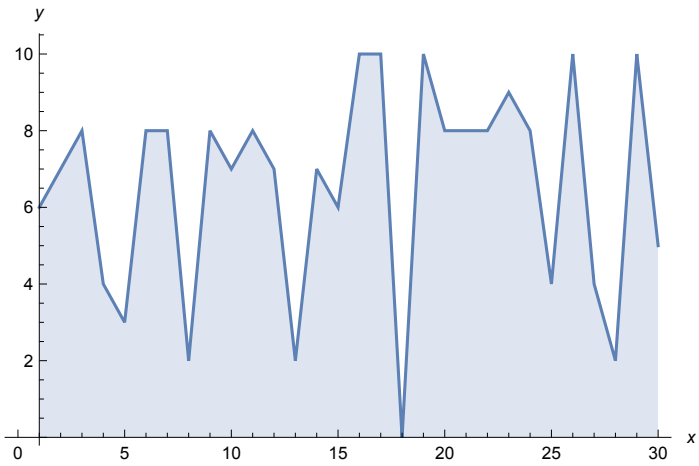
```
ListLinePlot[{data1, data2}, PlotLegends → {"Pays1", "Pays2"},
  AxesOrigin → {1, 0}, AxesLabel → {x, y}, Filling → Axis]
```
tracé de liste de ligne — légendes de tracé — origine des axes — titre d'axe — remplissage — axe

```
PieChart[data2]
```
diagramme circulaire

Out[•]=

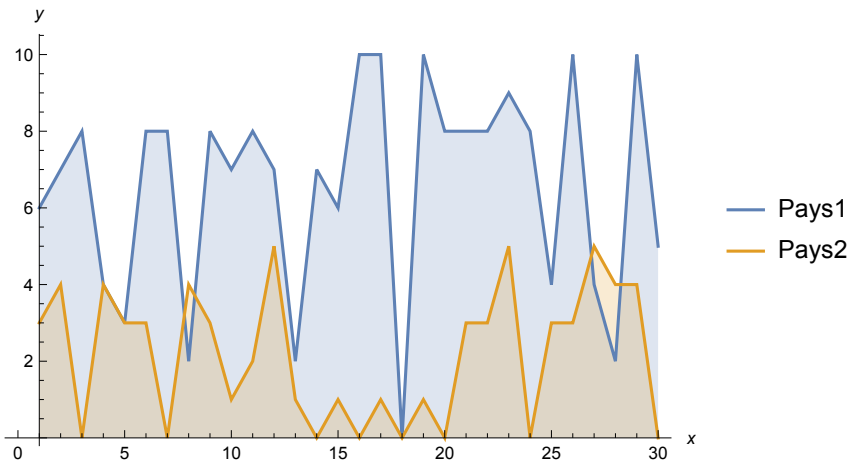


Out[•]=

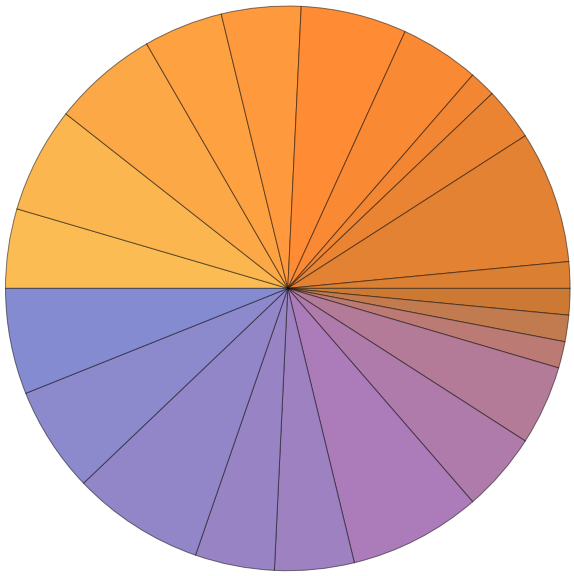


Pays1

Out[•]=



Out[•]=

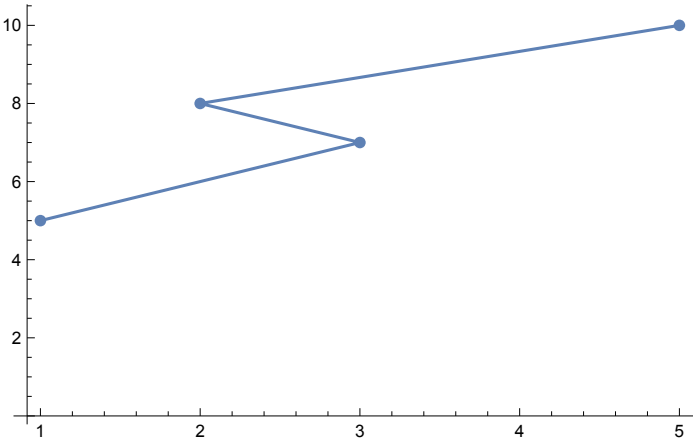


Comme pour **ListPlot**[], un très grand nombre d'options permettent d'habiller et de rendre très lisible les graphiques **PieChart**[] ou **SectorChard**[].

Toutes ces fonctions, notamment **ListPlot**[] et **ListLinePlot**[], sont aussi déployées pour les données qui se présentent sous forme d'une paire (x,y)

In[•]:=
```
ListLinePlot[{{1, 5}, {3, 7}, {2, 8}, {5, 10}}, PlotMarkers → Automatic]
```
tracé de liste de ligne — marqueurs de tracé — automatique

Out[•]=



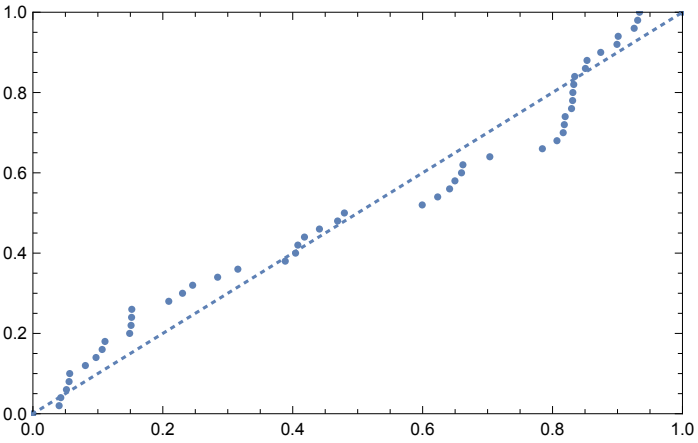
Des graphiques qui répondent à des objectifs scientifiques précis

Certains graphiques répondent à des questions scientifiques plus spécifiques. Soit l'exemple de la fonction **ProbabilityPlot**[]. Ayant une série, data3, nous pouvons la comparer à la distribution normale. Cette dernière est représentée par la diagonale du graphique. Cette opération s'écrit simplement :

In[•]:=
```
data3 = RandomReal[1, 50];
```
nombre réel aléatoire

```
ProbabilityPlot[data3]
```
tracé de probabilité

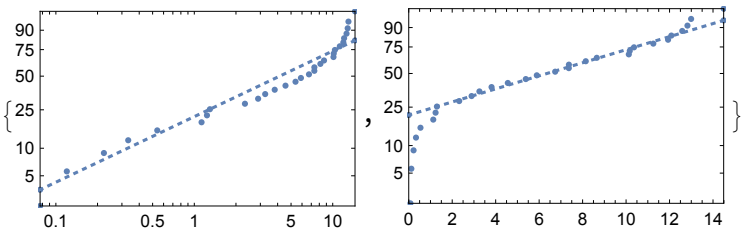
Out[•]=



et on en déduit visuellement que les données suivent ou ne suivent pas une loi normale. Bien d'autres solutions plus élaborées sont possibles pour réaliser ce test. Plus intéressante, la fonction **ProbabilityScalePlot**[] permet de visualiser la comparaison entre les données et différentes distribu tions probabilistes (normale, Weibull, exponentielle, log-normale, Fréchet ou Gumbel entre toutes les variables d'un tableau de données. La dernière ligne donne la liste des deux graphiques comparant les données avec les distributions de Weibull et de Gumbel.

```
In[•]:= data4 = RandomReal[15, 30];
        nombre réel aléatoire
        {ProbabilityScalePlot[data4, "Weibull"], ProbabilityScalePlot[data4, "Gumbel"]}
        tracé d'échelle de probabilité                tracé d'échelle de probabilité
```

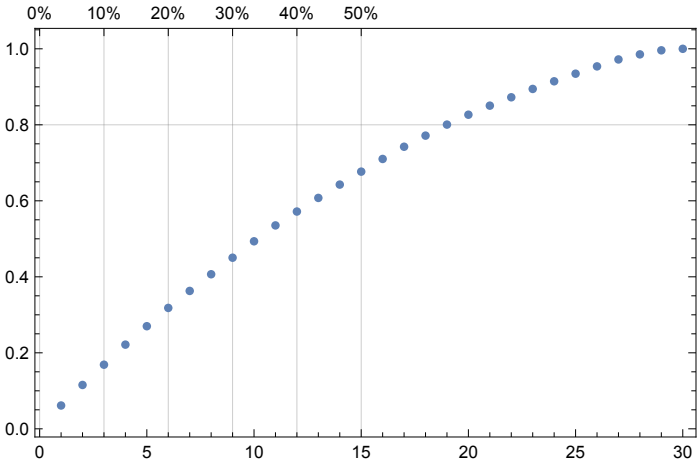
Out[•]=



Depuis la version 12, des fonctions créées par la communauté Mathematica sont mises à la disposition du chercheur. Elles sont appelées avec la fonction générale **ResourceFunction[]**. Directement utilisables, elles répondent à des besoins très ciblés. Tous les vendredis de nouvelles fonctions sont proposées à l'utilisateur. Citons deux exemples le graphique de Pareto qui permet de bien voir si la distribution des données est paretienne.

```
In[•]:= data4 = RandomReal[15, 30];
        nombre réel aléatoire
        ResourceFunction["ParetoPrinciplePlot"][data4]
        fonction ressource
```

Out[•]=

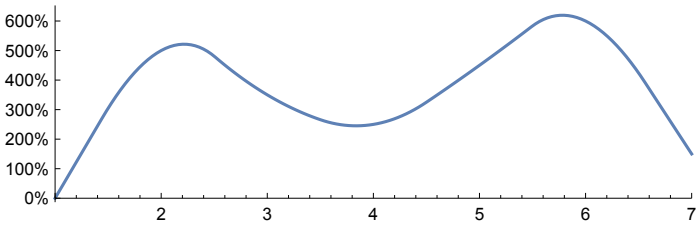


Puis le graphique qui donne en pourcentage l'évolution de la croissance ou de décroissance tout le long d'une série de nombres :

```
In[•]:= datax = {2, 12, 9, 7, 11, 14, 5}
        ResourceFunction["ListGrowthPlot"][datax]
        fonction ressource
```

Out[•]= {2, 12, 9, 7, 11, 14, 5}

Out[•]=



La plupart des graphiques généraux de type ListLinePlot ou PieChart, existent aussi en 3 dimensions.

Illustrer les tableaux de données

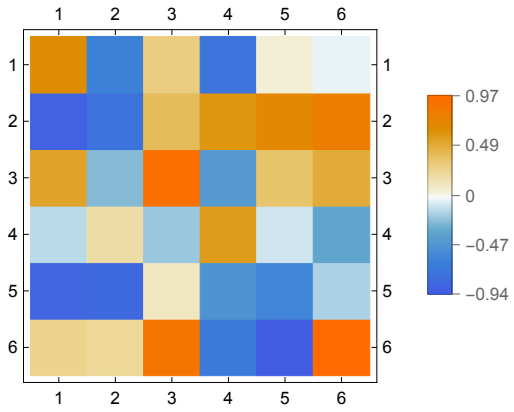
Pour les tableaux de données, les fonctions **ArrayPlot[]** et **MatrixPlot[]** sont à privilégier. Soit un tableau de coefficients de corrélation calculés entre 6 variables. Nous créons ce tableau de matière artificielle et aléatoire, avec des nombres compris entre -1 et +1. Puis nous en donnons une illustration graphique avec la fonction **MatrixPlot[]** et deux options relatives à la légende et à la taille du graphique.

In[•]:=
```
ClearAll["Global`*"]
```
efface tout

```
correl = RandomReal[{-1, 1}, {6, 6}];
```
nombre réel aléatoire

```
MatrixPlot[correl, PlotLegends → Automatic, ImageSize → 200]
```
tracé de matrice | légendes de tracé | automatique | taille d'image

Out[•]=



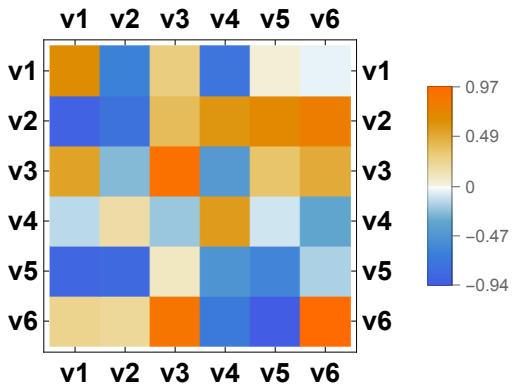
Il est aussi utile de figurer le nom des variables. Ce qui s'obtient de la façon suivante :

In[•]:=
```
tikc = {v1, v2, v3, v4, v5, v6};
htick = {#, Style[tikc[[#]], 14, Bold]} & /@ Range[6];
```
style | gras | plage

```
MatrixPlot[correl, FrameTicks → {htick, htick, htick, htick},
```
tracé de matrice | graduations du cadre

```
  PlotLegends → Automatic, ImageSize → 200]
```
légendes de tracé | automatique | taille d'image

Out[•]=



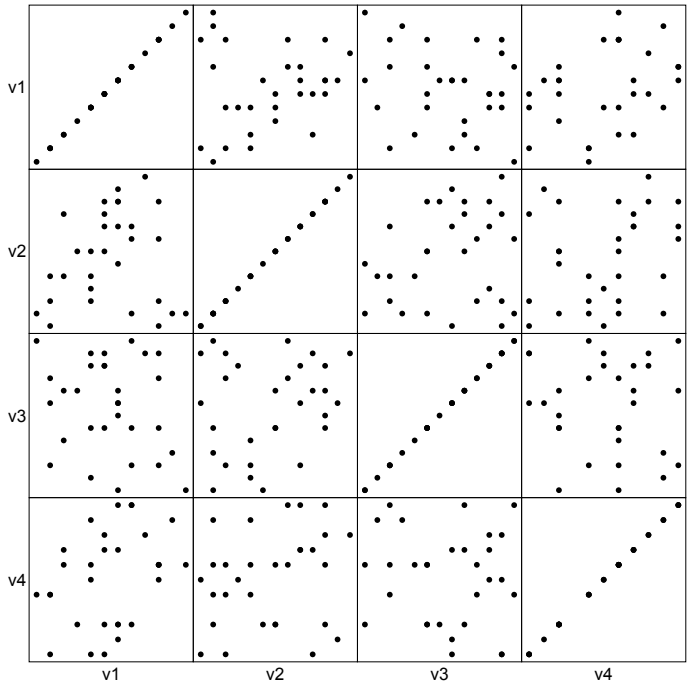
Non moins intéressante, la fonction **PairwiseScatterPlot[]** permet de visualiser toutes les relations entre les variables d'un tableau. Mais, cette fonction impose de faire appel à un package inclus dans Mathematica avec la fonction **Needs[]**. Soit un tableau de 30 lignes et 4 variables. On obtient le programme suivant :

In[•]:=
```
data5 = RandomInteger[{0, 12}, {30, 4}];
```
entier aléatoire

```
Needs["StatisticalPlots`"]
```
nécessite

```
PairwiseScatterPlot[data5, DataLabels → {"v1", "v2", "v3", "v4"}]
```

Out[•]=

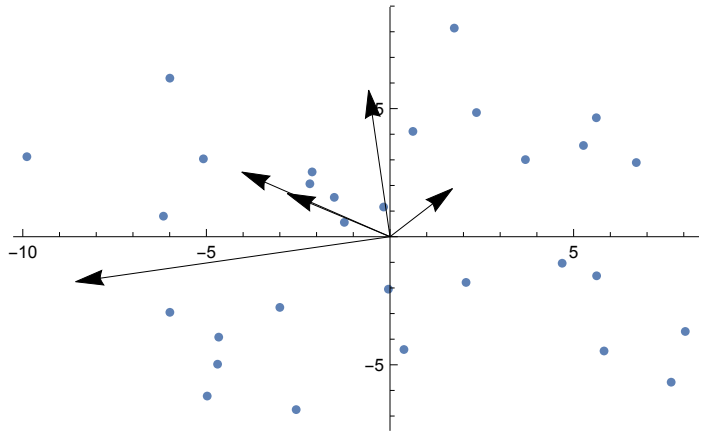


Bien d'autres fonctions permettent de représenter les données initiales ou traitées. Voici une autre exemple de Resource function qui donne immédiatement le graphique des deux premiers axes d'une analyse en composantes principales sur un tableau de données. On repère les 30 espaces et les 5 variables. Ce graphique donne une idée préalable à une analyse de ce type.

In[•]:=
```
dataD = RandomInteger[{0, 12}, {30, 5}];
```
entier aléatoire

```
ResourceFunction["BiPlot"][dataD]
```
fonction ressource

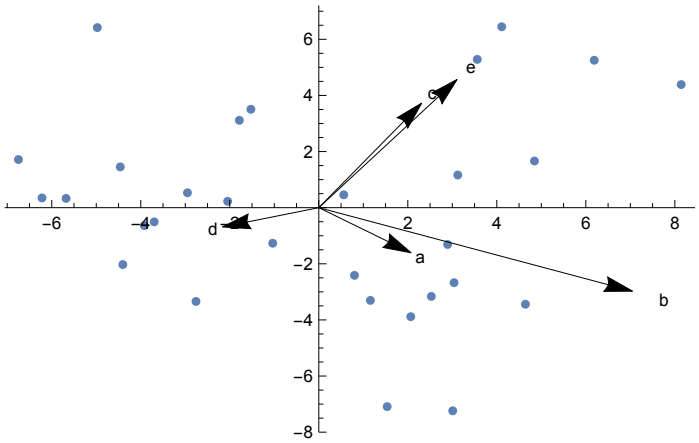
Out[•]=



Mais il est possible de choisir d'autres axes et de donner le nom des variables, voir de dessiner un graphique avec 3 axes.

```
In[•]:= ResourceFunction["BiPlot"][dataD,
         fonction ressource
  {2, 3}, "ColumnNames" → {"a", "b", "c", "d", "e"}]
```

Out[•]=



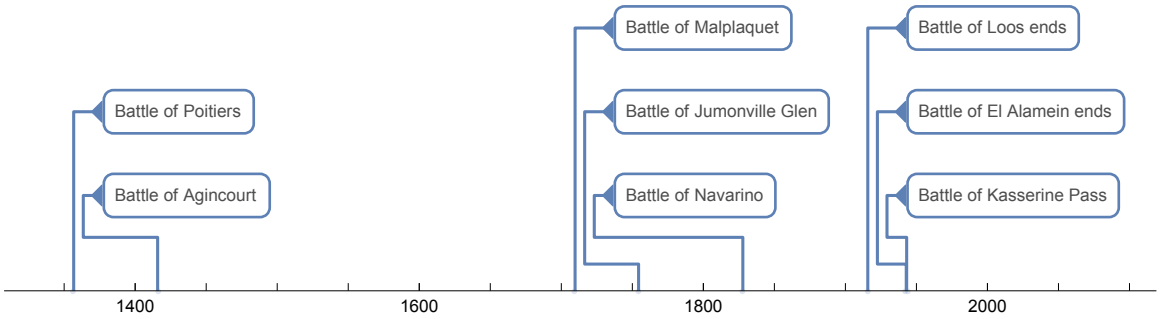
Nous sommes très loin des programmes qui demandent des lignes et des lignes de codes.

Visualiser des données temporelles

Souvent le géographe travaille sur des séries chronologiques. Plusieurs fonctions servent à illustrer ces séries temporelles. La première, **TimeLinePlot**[] est surtout utile pour les historiens. Elle affiche sous différents formats des événements historiques :

```
In[•]:= TimelinePlot[
         trace ligne temporelle
  {Battle of Kasserine Pass HISTORICAL EVENT, Battle of Agincourt HISTORICAL EVENT,
   Battle of El Alamein ends HISTORICAL EVENT, Battle of Jumonville Glen HISTORICAL EVENT,
   Battle of Loos ends HISTORICAL EVENT, Battle of Malplaquet HISTORICAL EVENT,
   Battle of Navarino HISTORICAL EVENT, Battle of Poitiers HISTORICAL EVENT}]
```

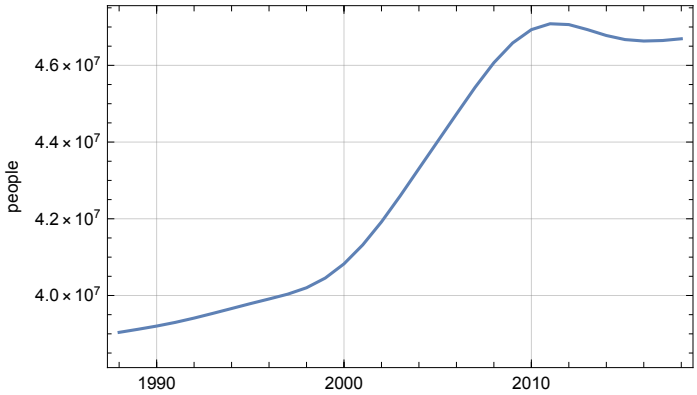
Out[•]=



La fonction **DateListPlot**[] est la plus classique, et comme toutes les fonctions graphiques elle possède un grand nombre d'options. Après avoir collecté les données de population pour l'Espagne, de 1988 à 2018, dans la base de données **CountryData** (ligne 1), la fonction **DateListPlot**[] trace le graphique :

In[79]:=
```
pop = CountryData["Spain", {{"Population"}, {1988, 2018}}];
      données de pays
DateListPlot[pop, GridLines → Automatic, FrameLabel → Automatic]
tracé de liste de dates   lignes de grille   automatique   titre de cadre   automatique
```

Out[80]=



La fonction **DateListLogPlot**[] visualise les données sous forme de logarithmes, ce qui est pratique lors de croissances très inégales. En outre, Mathematica met à la disposition du géographe des graphiques particuliers pour illustrer certaines techniques spécifiques employées dans des disciplines voisines, mais qui peuvent être transposées. Ainsi le graphique de Kagi, élaboré en économie financière permet de visualiser les époques de croissance et de décroissance des séries boursières, peut être utile dans certaines recherches géographiques.

Et, les Resource functions donnent aussi des graphiques spécifiques d'une série temporelle. Nous verrons plus en détail les graphiques spécifiques, par exemple ceux qui figurent les résultats d'une décomposition en ondelettes en abordant les analyses multi-échelles dans le temps et l'espace.

Visualisation dynamique des données

Très souvent le géographe souhaite inter-agir avec ses données, leur traitement, les résultats chiffrés ou graphiques. Mathematica intègre différentes fonction pour réaliser ce type d'action : **Dynamic**[], **Animate**[] et surtout **Manipulate**[]. Cette dernière est la plus pratique et elle s'applique à tous les objets Mathematica.

Illustrer des données suivant plusieurs hypothèses

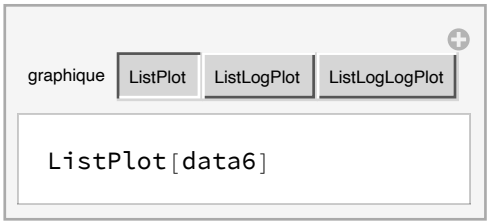
Le géographe qui traite visuellement une série peut se servir de la fonction **Manipulate**[] pour créer une page dynamique. En cliquant sur chaque nom il obtient le graphique correspondant. La ligne 1 du programme ci-dessous crée une liste de 30 nombres aléatoires, de 0 à 12. La ligne 2 crée une fenêtre avec une représentation graphique des données. Mais en cliquant sur la case ListLogLogPlot, vous obtenez les graphiques avec les deux axes en logarithmes.

In[81]:=
```
data6 = RandomInteger[12, 30];
        entier aléatoire

Manipulate[graphique[data6],
manipule

  {graphique, {ListPlot, ListLogPlot, ListLogLogPlot}}]
               tracé de liste  tracé log de liste  tracé log log de liste
```

Out[82]=



ListPlot: data6 is not a list of numbers or pairs of numbers.

Et cette fonction **Manipulate[]** possède de nombreuses options. Et rien ne s'oppose à ce que la fonction s'applique à deux variables.

Calculs dynamiques de paramètres de centralité

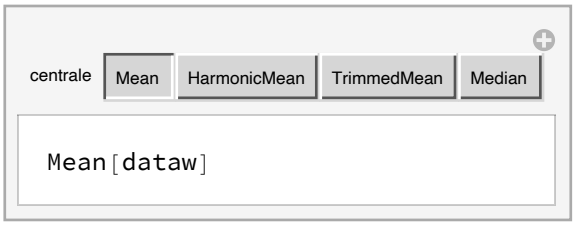
Mais il est aussi très facile d'appliquer cette logique à un calcul, par exemple pour calculer différents paramètres de centralité d'une série. Et le lecteur peut compléter ce programme avec des indicateurs de dispersion.

In[83]:=
```
dataw = RandomInteger[12, 40];
        entier aléatoire

Manipulate[centrale[dataw] // N,
manipule                                valeur numérique

  {centrale, {Mean, HarmonicMean, TrimmedMean, Median}}]
               valeu··· moyenne harmoni··· moyenne tronquée médiane
```

Out[84]=

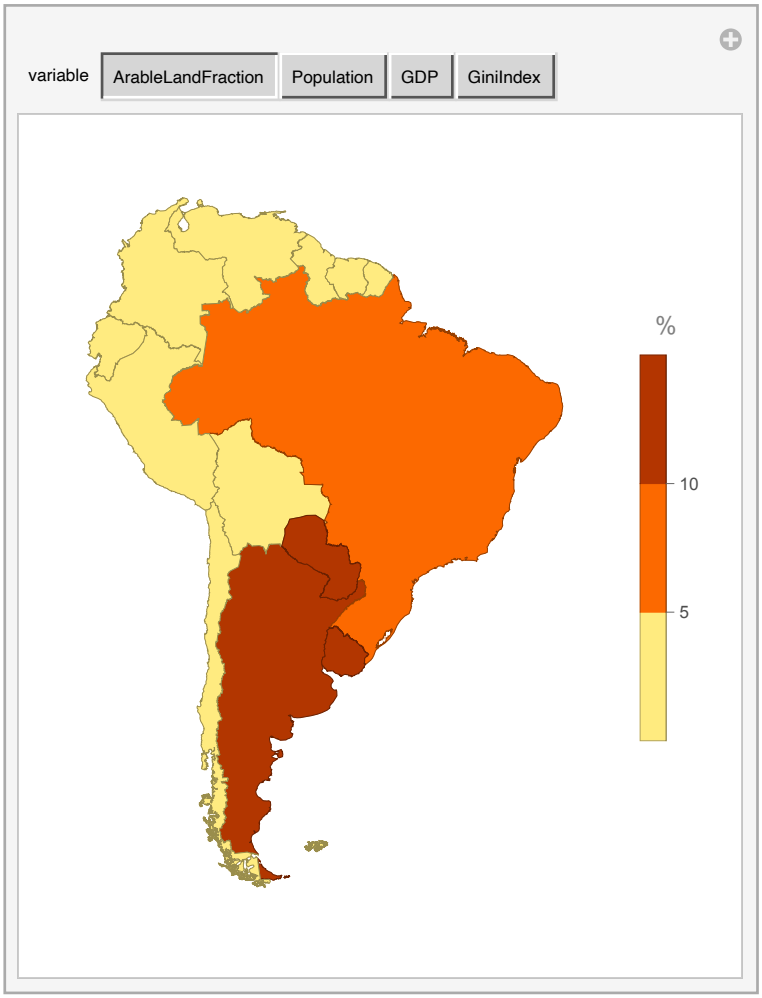


Construire un atlas dynamique de l'Amérique du Sud

La fonction Manipulate sert aussi à une présentation de cartes successives. La ligne d'instruction suivante permet de construire un embryon d'atlas avec seulement quatre variables issus de la base de données **CountryData**. Attention cependant, nous mélangeons des données brutes et des données en pourcentage. Il faudrait donc adapter la cartographie aux données traitées. Quand la donnée est manquante, l'unité spatiale correspondante reste grise.

In[85]:=
```
Manipulate[
 GeoRegionValuePlot[EntityClass["Country", "SouthAmerica"] → variable,
  GeoBackground -> None],
 {variable, {"ArableLandFraction", "Population", "GDP", "GiniIndex"}}]
```
manipule
représentation géographiq··· classe d'entités
fond géographique aucun

Out[85]=



Conclusion

Il est même possible d'enrichir ces graphiques directement, comme sur une feuille de dessin. En faisant un clique-droit avec la souris pointée sur le graphique, on obtient un menu et pour disposer des outils de dessin il suffit de choisir les **Drawing Tools** (outils de dessin). Le lecteur remarquera que ce menu permet aussi de sauvegarder le graphique dans les formats classiques (Pdf, Tiff,...).

Ouvrages recommandés :

André Dauphiné, 2017, *Modèles géographiques avec le langage Mathematica*, ISTE Editions
Heikki Ruskeepaa, 2009, *Mathematica Navigator*, 3rd Edition, Academic Press