

Importer des données avec Mathematica

Importer des données simples de tableur, image, et de cartes

Le géographe peut importer pratiquement tout type de données avec Mathematica. Une liste de tous les formats pouvant être lus est donnée par l' instruction `system` :

```
In[ ]:= $ImportFormats  
[formats d'importation]
```

```
Out[ ]:= {3DS, ACO, Affymetrix, AgilentMicroarray, AIFF, ApacheLog, ArcGRID, AU, AVI,  
Base64, BDF, Binary, Bit, BMP, BSON, Byte, BYU, BZIP2, CDED, CDF, CDX, CDXML,  
Character16, Character32, Character8, CIF, CML, Complex128, Complex256,  
Complex64, CSV, CUBE, CUR, DAE, DBF, DICOM, DICOMDIR, DIF, DIMACS, Directory,  
DOT, DXF, EDF, EML, EPS, ExpressionJSON, ExpressionML, FASTA, FASTQ, FBX, FCHK,  
FCS, FITS, FLAC, GaussianLog, GenBank, GeoJSON, GeoTIFF, GIF, GPX, Graph6,  
Graphlet, GraphML, GRIB, GTOPO30, GXL, GZIP, HarwellBoeing, HDF, HDF5, HEIF,  
HIN, HTML, HTTPRequest, HTTPResponse, ICC, ICNS, ICO, ICS, Ini, Integer128,  
Integer16, Integer24, Integer32, Integer64, Integer8, JavaProperties,  
JavaScriptExpression, JCAMP-DX, JPEG, JPEG2000, JSON, JSONLD, J VX, KML,  
LaTeX, LEDA, List, LWO, M4A, MAT, MathML, Matroska, MBOX, MCTT, MDB, MESH,  
MGF, MIDI, MMCIF, MO, MOL, MOL2, MP3, MP4, MPS, MTP, MTX, MX, MXNet, NASACDF,  
NB, NDK, NetCDF, NEXUS, NOFF, NQuads, NTriples, OBJ, ODS, OFF, Ogg, OggVorbis,  
ONNX, OpenEXR, OWLFunctional, Package, Pajek, PBM, PCAP, PCX, PDB, PDF,  
PGM, PHPIni, PLY, PNG, PNM, PPM, PXR, PythonExpression, QuickTime, Raw,  
RawBitmap, RawJSON, RDFXML, Real128, Real32, Real64, RIB, RLE, RSS, RTF,  
SCT, SDF, SDTS, SDTSDem, SFF, SHP, SMA, SME, SMILES, SND, SP3, SPARQLQuery,  
SPARQLResultsJSON, SPARQLResultsXML, SPARQLUpdate, Sparse6, STL, String,  
SurferGrid, SXC, Table, TAR, TerminatedString, TeX, Text, TGA, TGF, TIFF,  
TIGER, TLE, TriG, TSV, Turtle, UBJSON, UnsignedInteger128, UnsignedInteger16,  
UnsignedInteger24, UnsignedInteger32, UnsignedInteger64, UnsignedInteger8,  
USGSDEM, UUE, VCF, VCS, VideoFormat, VTK, WARC, WAV, Wave64, WDX, WebP,  
WLNet, WMLF, WXF, XBM, XHTML, XHTMLMathML, XLS, XLSX, XML, XPORT, XYZ, ZIP}
```

Cette liste, qui s'allonge lors de chaque mise à niveau, montre l'ampleur et la variété des formats supportés par Mathematica. Il est même possible de lire directement des fichiers compressés. La ligne d'instruction :

```
In[ ]:= data = Import[SystemDialogInput["FileOpen"]]  
[importe [entrée de dialogue de système]
```

fait apparaître une fenêtre où l'utilisateur va choisir le fichier auparavant sauvegardé dans son ordinateur. Ce fichier est alors ouvert et affiché, que ce soit des nombres, du texte, des images ou des cartes.

Souvent, des options ou quelques options donnent des informations complémentaires sur le fichier importé. Dans un fichier Excel classique la première ligne et la première colonne fournissent respectivement le nom des variables et le nom des objets. De plus, cette ouverture simple génère parfois un jeu de parenthèses supplémentaires. Pour lire convenablement les données d'un tableau Excel,

on utilise le petit programme ci - dessous :

```
ClearAll["Global`*"]
[efface tout]
data = Flatten[Import[SystemDialogInput["FileOpen"]], 1]
[aplatis [importe [entrée de dialogue de système]
nomcolonne = data[[1, 2 ;;]]
nomligne = data[[2 ;;, 1]]
nombres = data[[2 ;;, 2 ;;]];
```

La première ligne, nettoie la mémoire. La fonction **Flatten**[..., 1] importe les données et élimine les parenthèses superflues. Les trois fonctions suivantes sauvegardent le nom des variables, le nom des objets, et les données numériques consignées dans le tableau Excel. Remarquez les double crochets. Pour une image, on utilisera souvent le programme ci - dessous :

```
ClearAll["Global`*"]
[efface tout]
image = Import[SystemDialogInput["FileOpen"]]
[importe [entrée de dialogue de système]
ImageDimensions[image]
[dimensions d'image]
nombre = ImageData[image, "Byte"];
[données d'image [octet]
```

En l'absence d'un point - virgule à la fin de la deuxième ligne, l'image est directement affichée sur l'écran. Puis l'instruction suivante, **ImageDimensions**[], donne les dimensions verticale et horizontale en nombre de pixels. Enfin, dans la dernière ligne la fonction **ImageData**[] traduit l'image en un tableau de nombres donnant la valeur de chaque pixel, de 0 à 255. Si l' image importée est en couleur, ce sont les trois valeurs RGB qui sont produites. Et, il est possible de remplacer l'option **Byte** pour obtenir une autre représentation numérique de l' image, par exemple en noir et blanc avec l'option **Bit**. Il est souvent préférable de bien indiquer le type de fichier à importer. L'instruction :

```
In[*]:= ClearAll["Global`*"]
[efface tout]
data = Import[SystemDialogInput["FileOpen"], "XLS"]
[importe [entrée de dialogue de système]
```

permet d'importer un fichier Excel ancien tandis que l'instruction :

```
In[*]:= ClearAll["Global`*"]
[efface tout]
data = Import[SystemDialogInput["FileOpen"], "CSV"]
[importe [entrée de dialogue de système]
```

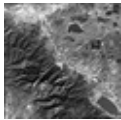
permet d'importer un fichier Excel ou autre de format CSV.

Suivant le type de fichier à importer, diverses options sont disponibles. Elles sont indiquées en ajoutant l'option "**Elements**" à l'instruction.

La ligne d'instruction suivante importe les informations relatives à une image sauvegardée en format GeoTIFF, dans une base de donnée fournie avec le logiciel Mathematica :

```
In[ ]:= Import["ExampleData/cea.tif", {"GeoTIFF", "Summary"}]  
[importe [données d'exemples
```

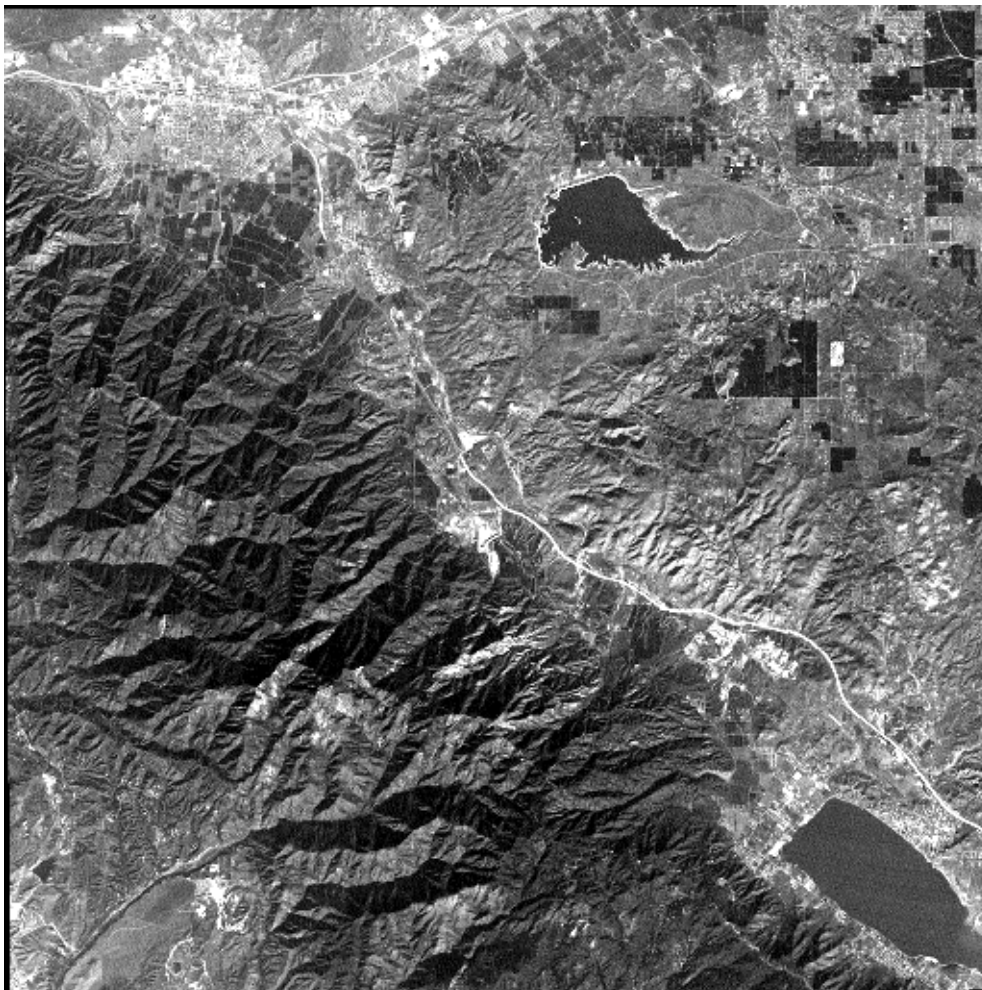
```
Out[ ]:=
```

Thumbnail	
Format	GeoTIFF
ImageSize	{514, 515}
ColorSpace	GrayLevel
Channels	1
BitDepth	8
FileName	cea.tif
FileSize	264.6 kB

Et, l'image satellite s'obtient avec la seule instruction :

```
In[ ]:= Import["ExampleData/cea.tif", "GeoTIFF"]  
[importe [données d'exemples
```

```
Out[ ]:=
```



Nous pouvons aussi importer un fichier Arcgrid et voir ce qu'il contient avec l'instruction :

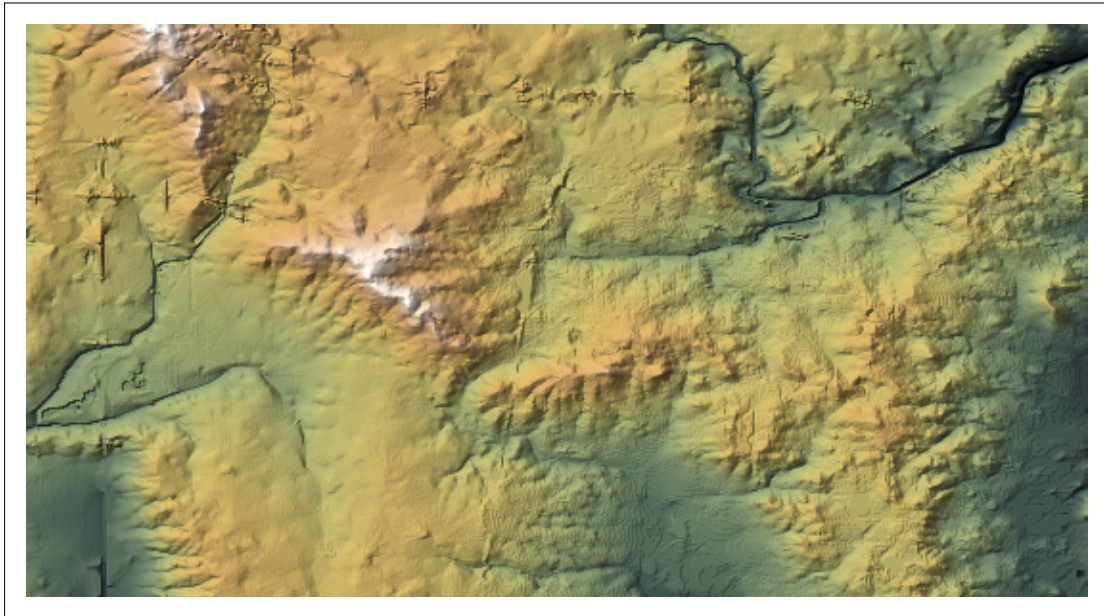
```
In[ ]:= Import[ "http://exampledata.wolfram.com/ArcGRID.zip", {"ArcGRID", "Elements"}]
[importe
```

```
Out[ ]:= {Centering, CentralScaleFactor, CoordinateSystem,
  CoordinateSystemInformation, Data, DataFormat, Datum, ElevationRange,
  Graphics, GridOrigin, Image, InverseFlattening, LinearUnits, Projection,
  ProjectionName, RasterSize, ReferenceModel, ReliefImage, SemimajorAxis,
  SemiminorAxis, SpatialRange, SpatialResolution, StandardParallels}
```

ou afficher directement son contenu visuel :

```
In[ ]:= Import[ "http://exampledata.wolfram.com/ArcGRID.zip",
[importe
  "ArcGRID", ImageSize → Large]
[taille d'image [taille grande
```

```
Out[ ]:=
```



Importer depuis les bases de données thématiques Mathematica

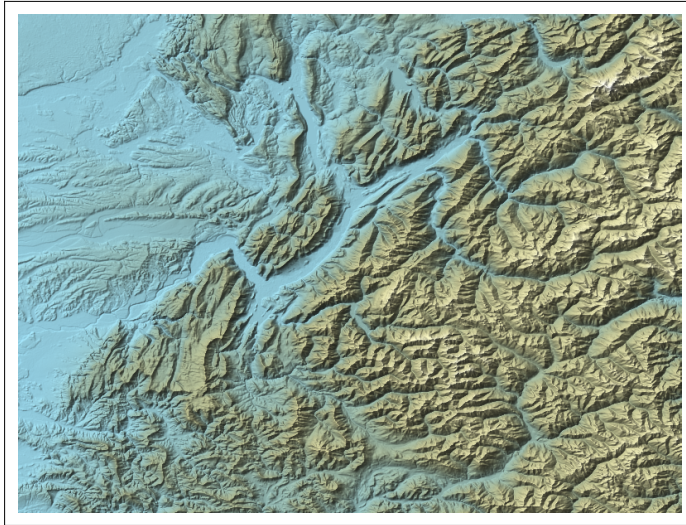
En outre, Mathematica met à la disposition du géographe des données, qui sont réparties dans plusieurs bases de données. Les plus importantes concernent les MNT (modèles numériques de terrain), la climatologie, les finances, les villes, les États, les cartes et les données satellitaires. Voici quelques exemples :

Un modèle numérique de terrain des Alpes du Nord avec GeoElevationData

D'abord, la première ligne permet de récupérer dans le fichier data les altitudes dans le rectangle correspondant aux géo-positions. Puis, la deuxième ligne élabore à partir de la matrice des données une carte du relief.


```
In[ ]:= data = GeoElevationData[{GeoPosition[{44.5, 5}], GeoPosition[{46, 7}]}];
           [données d'élévation géog. · [position géographique] [position géographique]
ReliefPlot[data, DataReversed → True, ColorFunction → "LightTerrain"]
           [tracé en relief] [données inversées] [vrai] [fonction de couleur]
```

Out[]:=



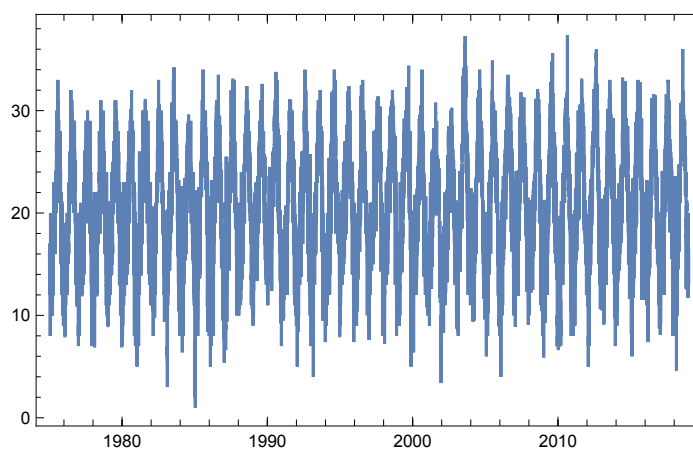
Il suffit de modifier les coordonnées géographiques pour obtenir un autre MNT. Et pour la visualisation, le géographe peut utiliser diverses représentations, et pour chacune d'elle un grand nombre d'options sont disponibles.

Accéder aux données climatiques avec WeatherData

La base de données **WeatherData** contient les données de stations référencées sur le globe. La ligne d'instruction ci-dessous importe les températures maximales de la station de Barcelone (LEBL), du 1er janvier 1975 au 31 décembre 2018. Le point virgule à la fin de la ligne d'instruction évite un affichage des données sur l'écran. Puis, ces données sont représentées sur un graphique simple.

```
In[ ]:= data = WeatherData["LEBL",
           [données du temps]
           "MaxTemperature", {{1975, 01, 1}, {2018, 12, 31}}, "Day"];
DateListPlot[
           [tracé de liste de dates]
           data]
```

Out[]:=



CountryData : Une base de données par État très documentée

Dans la base de données **CountryData**, chaque État du globe est décrit par 223 indicateurs de toute nature[forme, démographie, PNB, etc]. Pour avoir la liste par ordre alphabétique de toutes les variables intégrées et mises sans cesse à jour entrez la ligne :

```
In[*]:= CountryData["Properties"]
```

```
[données de pays | propriétés
```

```
Out[*]:= {AdultPopulation, AgriculturalProducts, AgriculturalValueAdded, Airports,
  AlternateNames, AlternateStandardNames, AMRadioStations, AnnualBirths,
  AnnualDeaths, AnnualHIV/AIDSDeaths, ArableLandArea, ArableLandFraction, Area,
  BirthRateFraction, BorderingCountries, BordersLengths, BoundaryLength,
  CallingCode, CapitalCity, CapitalLocation, CapitalLocationLink,
  CellularPhones, CenterCoordinates, CenterLocationLink, ChildPopulation,
  Classes, ClimateTypes, CoastlineLength, ConstructionValueAdded,
  Continent, Coordinates, Countries, CountryCode, CropsLandArea,
  CropsLandFraction, CurrencyCode, CurrencyName, CurrencyShortName,
  CurrencyUnit, CurrentAccountBalance, DeathRateFraction, Dependencies,
  DependencyParent, EconomicAid, ElderlyPopulation, ElectricalGridFrequency,
  ElectricalGridPlugImages, ElectricalGridPlugs, ElectricalGridSocketImages,
  ElectricalGridSockets, ElectricalGridVoltages, ElectricityConsumption,
  ElectricityExports, ElectricityImports, ElectricityProduction,
  EnvironmentalAgreements, EnvironmentalIssues, EthnicGroups,
  EthnicGroupsFractions, ExchangeRate, ExpenditureFractions, ExportCommodities,
  ExportPartners, ExportPartnersFractions, ExportValue, ExternalDebt,
  FemaleAdultPopulation, FemaleChildPopulation, FemaleElderlyPopulation,
  FemaleInfantMortalityFraction, FemaleLifeExpectancy, FemaleLiteracyFraction,
  FemaleMedianAge, FemalePopulation, FiscalYearDate, FixedInvestment, Flag,
  FlagDescription, FMRadioStations, ForeignExchangeReserves, ForeignOwnedShips,
  ForeignRegisteredShips, FullCoordinates, FullName, FullNativeName,
  FullPolygon, GDP, GDPAtParity, GDPPerCapita, GDPRealGrowth, GDPSectorFractions,
  GiniIndex, GovernmentConsumption, GovernmentDebt, GovernmentExpenditures,
  GovernmentReceipts, GovernmentSurplus, GrossInvestment, Groups,
  HighestElevation, HighestPoint, HIV/AIDSDeathRateFraction, HIV/AIDSFraction,
  HIV/AIDSPopulation, HouseholdConsumption, ImportCommodities, ImportPartners,
  ImportPartnersFractions, ImportValue, IndependenceDate, IndependenceYear,
  IndustrialProductionGrowth, IndustrialValueAdded, InfantMortalityFraction,
  InfectiousDiseases, InflationRate, InternationalOrganizations,
  InternationalOrganizationsObserver, InternetCode, InternetHosts, InternetUsers,
  InventoryChange, IrrigatedLandArea, IrrigatedLandFraction, ISOName,
  LaborForce, LandArea, Languages, LanguagesDialects, LanguagesFractions,
  LargestCities, LifeExpectancy, LiteracyFraction, LowestElevation, LowestPoint,
  MajorIndustries, MajorPorts, MaleAdultPopulation, MaleChildPopulation,
  MaleElderlyPopulation, MaleInfantMortalityFraction, MaleLifeExpectancy,
  MaleLiteracyFraction, MaleMedianAge, MalePopulation, ManufacturingValueAdded,
  MaritimeClaims, MedianAge, Memberships, MerchantShips, MerchantShipsDeadWeight,
  MerchantShipsGross, MerchantShipTypes, MigrationRateFraction,
  MilitaryAgeFemales, MilitaryAgeMales, MilitaryAgePopulation, MilitaryAgeRate,
```

```

MilitaryExpenditureFraction, MilitaryExpenditures, MilitaryFitFemales,
MilitaryFitMales, MilitaryFitPopulation, MiscellaneousValueAdded, Name,
NationalIncome, NationalityName, NativeName, NaturalGasConsumption,
NaturalGasExports, NaturalGasImports, NaturalGasProduction, NaturalGasReserves,
NaturalHazards, NaturalResources, OilConsumption, OilExports, OilImports,
OilProduction, OilReserves, PavedAirportLengths, PavedAirports, PavedRoadLength,
PhoneLines, Pipelines, Polygon, Population, PopulationGrowth, PovertyFraction,
PriceIndex, RadioStations, RailwayGaugeLengths, RailwayGaugeRules,
RailwayLength, RegionNames, Regions, Religions, ReligionsFractions,
RoadLength, SchematicCoordinates, SchematicPolygon, SectorLaborFractions,
Shape, ShortWaveRadioStations, SignedEnvironmentalAgreements, StandardName,
SuffrageType, TelevisionStations, TerrainTypes, TimeZones, TotalConsumption,
TotalFertilityRate, TradeValueAdded, TransportationValueAdded,
UNCode, UnemploymentFraction, UNNumber, UnpavedAirportLengths,
UnpavedAirports, UnpavedRoadLength, ValueAdded, WaterArea, WaterwayLength}

```

Les deux lignes suivantes donnent le nom des pays européens et l'espérance de vie de leur population. La fonction **DeleteCases**[... {_, **_Missing**}] permet d'éliminer les pays européens dont les données manquent. Puis, sont sélectionnées les seules valeurs numériques de l'espérance de vie, qui sont incluses dans la colonne 2 de la liste data, la colonne 1 contenant le nom des États. Ces valeurs numériques sont collectées dans le fichier nombres.

```

In[*]:= data = DeleteCases[Table[{i, CountryData[i, "LifeExpectancy"]},
  |supprime cas |table |données de pays
  {i, CountryData["Europe"]}], {_, _Missing}]
  |données de pays
nombres = QuantityMagnitude[data[[All, 2]]]
  |ampleur de quantité |tout
Out[*]= {{ Albania , 78.6 yr }, { Andorra , 82.9 yr }, { Austria , 81.7 yr },
  { Belarus , 73.2 yr }, { Belgium , 81.2 yr }, { Bosnia and Herzegovina , 77.1 yr },
  { Bulgaria , 74.8 yr }, { Croatia , 76.3 yr }, { Cyprus , 79. yr },
  { Czech Republic , 78.9 yr }, { Denmark , 81. yr }, { Estonia , 77. yr },
  { Faroe Islands , 80.6 yr }, { Finland , 81.1 yr }, { France , 82. yr },
  { Germany , 80.9 yr }, { Gibraltar , 79.7 yr }, { Greece , 80.8 yr },
  { Guernsey , 82.7 yr }, { Hungary , 76.3 yr }, { Iceland , 83.1 yr },
  { Ireland , 81. yr }, { Isle of Man , 81.4 yr }, { Italy , 82.4 yr }, { Jersey , 82. yr },
  { Kosovo , 71.646341 yr }, { Latvia , 74.9 yr }, { Liechtenstein , 82. yr },
  { Lithuania , 75.2 yr }, { Luxembourg , 82.4 yr }, { North Macedonia , 75.9 yr },
  { Malta , 82.7 yr }, { Moldova , 71.3 yr }, { Monaco , 89.4 yr },
  { Montenegro , 77.116 yr }, { Netherlands , 81.5 yr }, { Norway , 82. yr },
  { Poland , 77.9 yr }, { Portugal , 80.9 yr }, { Romania , 75.6 yr },
  { San Marino , 83.4 yr }, { Serbia , 75.9 yr }, { Slovakia , 77.4 yr },
  { Slovenia , 81.2 yr }, { Spain , 81.8 yr }, { Sweden , 82.2 yr },
  { Switzerland , 82.7 yr }, { Ukraine , 72.4 yr }, { United Kingdom , 80.9 yr }}
Out[*]= {78.6, 82.9, 81.7, 73.2, 81.2, 77.1, 74.8, 76.3, 79., 78.9, 81., 77.,
  80.6, 81.1, 82., 80.9, 79.7, 80.8, 82.7, 76.3, 83.1, 81., 81.4, 82.4, 82.,
  71.646341, 74.9, 82., 75.2, 82.4, 75.9, 82.7, 71.3, 89.4, 77.116, 81.5,
  82., 77.9, 80.9, 75.6, 83.4, 75.9, 77.4, 81.2, 81.8, 82.2, 82.7, 72.4, 80.9}

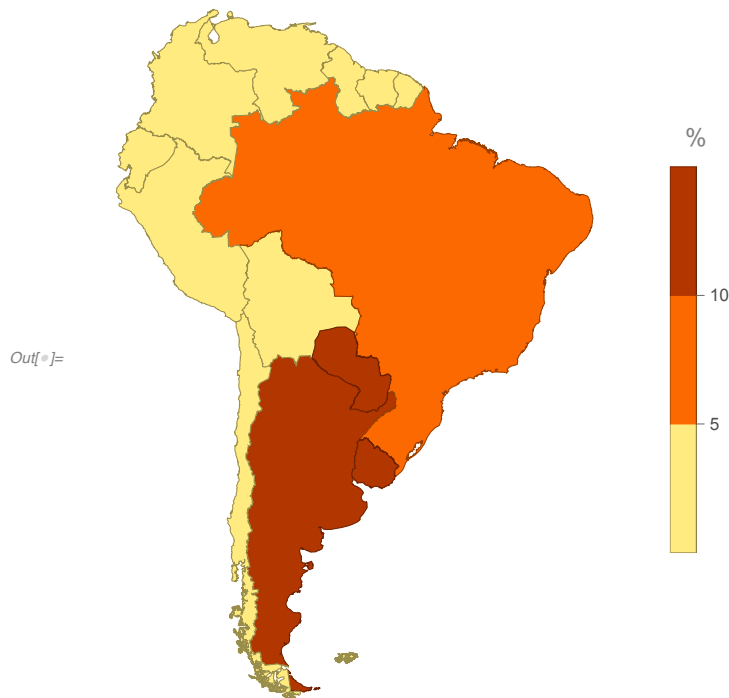
```

Comme la base de donnée contient les cartes des États sous plusieurs formes, raster ou vectorielle, il est facile de réaliser des cartes très simplement pour illustrer les données de la base ou même d'autres données importées par le géographe. La ligne d'instruction suivante donne une représentation cartographique de la fraction des terres arables en Amérique du Sud :


```

In[*]:= GeoRegionValuePlot[
  [représentation géographique de regions associées à des valeurs
    EntityClass["Country", "SouthAmerica"] -> "ArableLandFraction",
    [classe d'entités
    GeoBackground -> None]
    [fond géographique      [aucun

```



CityData : Une base de donnée pauvre, mais utile

La base de données **CityData** donne quelques informations (nom, altitude, coordonnées, population) de toutes les communes de tous les États. Il est possible cependant de s'en servir en liaison avec d'autres fonctions. Le programme ci - dessous utilise la base de données **CityData**. La première ligne donne la liste de toutes les communes de l' ancienne région Midi - Pyrénées. Mais, comme nous utilisons la fonction **Short[]**, seules les noms de quelques unes sont affichées sur l'écran. Puis la seconde ligne d'instruction affiche le graphique qui représente la loi rang - taille appliquée à l' ensemble des populations communales de cette région :

```

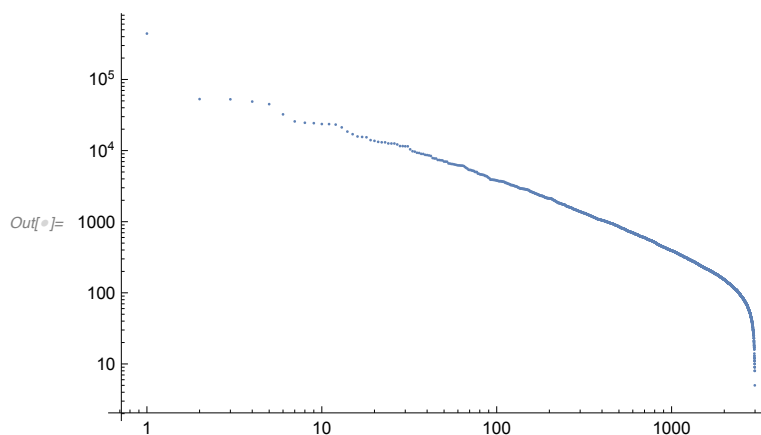
CityData[{"All", "MidiPyrenees", "France"}] // Short
[données des· · [tout] [court]

ListLogLogPlot[Reverse[
[tracé log log de liste] [renverses]

Sort[CityData[#, "Population"] & /@ CityData[{"All", "MidiPyrenees", "France"}]]]
[trie] [données des villes] [données des· · [tout]

Out[ ]//Short= { Toulouse , Castres , Montauban , Albi , Tarbes , Colomiers , Tournefeuille ,
Auch , Rodez , Millau , Muret , Cahors , Blagnac , Plaisance-du-Touch ,
Pamiers , Lourdes , Balma , Cugnaux , Villefranche-de-Rouergue , Moissac ,
Castelsarrasin , L'Union , Graulhet , Gaillac , Saint-Orens-de-Gameville ,
Ramonville-Saint-Agne , Saint-Gaudens , Onet-le-Château , Castanet-Tolosan ,
Figeac , Mazamet , Carmaux , Portet-sur-Garonne , Foix , Lavaur ,
Villeneuve-Tolosane , Saint-Jean , Bagnères-de-Bigorre , Castelginest , Revel ,
Saint-Affrique , Pibrac , Aureilhan , Fonsorbes , Condom , Lavelanet , Auterive ,
Aussillon , <<2895>> , Sainte-Foi , Gonez , Bouilh-Devant , Bramevaque ,
Burret , Balacet , Uz , Loubaut , Fréchendets , Saint-Jean-du-Castillonnais ,
Artigues , Benque-Dessous-et-Dessus , Illier-et-Laramade , Le Puch , Sode ,
Binos , Orus , Arrodets , Caychax , Suzan , Francazal , Larnat ,
Barthe , Cazaril-Laspènes , Camous , Tignac , Jurvielle , Ardengost ,
Ens , Thuy , Caubous , Grailhen , Ris , Cazaux-Debat , Bestiac ,
Saccourvielle , Mazouau , Fréchet-Aure , Bourg-d'Oueil , Montaillou ,
Gestiès , Baren , Appy , Cirès , Samuran , Senconac , Casterets }

```

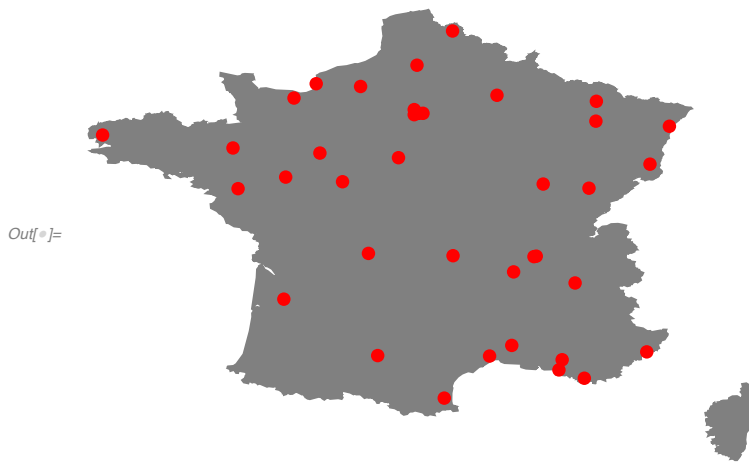


Le petit programme ci-dessous, inspiré de l'aide présentant **CityData**, localise les plus grandes villes françaises. La plupart des termes sont compréhensibles. La fonction Tooltip donne le nom de la ville quand on place le curseur sur le point rouge qui lui correspond, avec un problème pour la métropole parisienne. Il serait possible d'obtenir une autre projection cartographique.

```

In[ ]:= Graphics[{Gray, CountryData["France", "Polygon"], PointSize[Large], Red,
  Tooltip[Point[Reverse[CityData[#, "Coordinates"]]], CityData[#, "Name"]] & /@
  CityData[{Large, "France"}]}]

```



Des billions d'informations accessibles avec WolframAlpha et Entity

Quand le chercheur dispose d'une liaison Internet, une base de connaissance, qui contient des billions de données ordonnées en milliers de domaines, est accessible. Et toutes ces données peuvent être collectées, triées et sauvegardées pour tout travail de recherche. Voici trois exemples :

Accéder aux données avec l'instruction WolframAlpha

In[]:= WolframAlpha["France energy production"]
 [WolframAlpha

Assuming total primary energy | Use [total electricity](#) instead

Input interpretation: +

France total primary energy production

Result: Show non-metric +

5.355×10^{18} J/yr (joules per year) (2012 estimate)

History: +

(from 1980 to 2012)
(in quintillions joules per year)

Unit conversions: +

5.076×10^{15} BTU_{IT}/yr (IT British thermal units per year)

1.391×10^{13} BTU_{IT}/day (IT British thermal units per day)

5.794×10^{11} BTU_{IT}/h (IT British thermal units per hour)

Interpretations: More +

power

Basic unit dimensions: +

[mass] [length]² [time]⁻³

Energy production in France: +

crude oil	36.91 quadrillion J/yr
natural gas	14.17 quadrillion J/yr
coal	0 J/yr
nuclear energy	4.422×10^{18} J/yr
renewable energy	831.1 quadrillion J/yr
total primary energy	5.355×10^{18} J/yr

+ Units

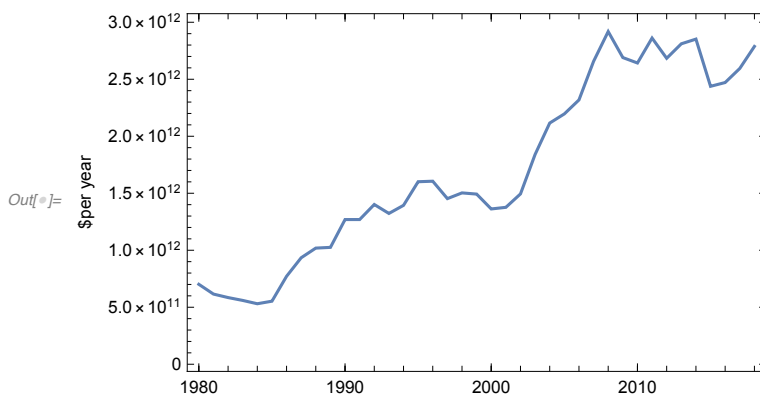
WolframAlpha +

Toutes ces données contenues dans les tableaux ou graphiques peuvent être récupérées.

Accéder aux données avec les instructions *Entity*, *EntityProperty* et *EntityValues*

Il est aussi possible d'obtenir les données de la base de connaissance en mobilisant les instructions de type **Entity**, comme le montre le petit programme ci-dessous qui récupère les données relatives au PNB en France, de 1980 à 2018, puis les affiche dans un graphique avec la fonction **DateListPlot[]** et diverses options.

```
In[ ]:= data =
  EntityValue[Entity["Country", "France"], EntityProperty["Country", "GDP",
    { "CurrencyUnit" -> "CurrentUSDollar",
      "Date" -> Interval[{DateObject[{1980}], DateObject[{2018}]}]}]];
  DateListPlot[data, FrameLabel -> Automatic, Joined -> True]
```



Des fonctions spécifiques pour obtenir des données externes

Le géographe peut obtenir les cartes et les images satellites qu'il souhaite sans grande difficulté.

GeoGraphics [] pour obtenir des cartes

Voici deux cartes de style différent de la France et de l'Italie :

```
In[ ]:= GeoGraphics[{Polygon[Entity["Country", "France"]}]}
  GeoGraphics[{Polygon[Entity["Country", "Italy"]}]}],
  GeoBackground -> GeoStyling["ReliefMap"]]
```

Out[]=



Out[]=



Bien d'autres styles sont disponibles. Mais il est possible d'obtenir tout type de carte, par exemple celle d'une ville avec ses rues et monuments, ou de créer une carte à partir de données prises dans la base de connaissance. Le programme ci-dessous collecte les séisme de magnitude supérieure à 5 observés en France de janvier 1980 à décembre 2018, puis les localise sur la carte de France.

```

In[ ]:= data = EarthquakeData[Entity["Country", "France"],
    |données sismiques |entité
    5, {{1980, 1, 1}, {2018, 12, 31}}, "Position"]["Values"]
    |position |valeurs

GeoGraphics[{Polygon[Entity["Country", "France"]],
    |carte géographique |polygone |entité
    Red, PointSize[0.02], Point[data]}]
    |rouge |taille des points |point

Out[ ]:= {GeoPosition[{48.102001, 6.494}], GeoPosition[{44.870998, 6.704}],
    GeoPosition[{46.012001, 6.3530002}], GeoPosition[{42.827999, 2.533}],
    GeoPosition[{42.900002, 0.60000002}], GeoPosition[{46.689999, -0.99000001}],
    GeoPosition[{42.929001, -0.17}], GeoPosition[{48.34, 6.5700002}]}

```

Out[]:=



Il est facile avec les options de travailler sur des cartes obtenues de divers serveurs cartographiques et pas seulement du serveur Mathematica.

Obtenir des photos satellitaires avec Geolmage

Il existe de nombreuses approches pour obtenir des photographies satellitaires. La plus simple utilise l'instruction **Geolmage[]**. En voici deux exemples simples :


```
In[*]:= GeoImage[GeoRange -> {{43, 48}, {7, 17}}]
[image géo... |gamme géographique]

GeoImage[Entity["Building", "EiffelTower::5h9w8"], GeoZoomLevel -> 11]
[image géo... |entité] [niveau de zoom géographique]
```

Out[*]=



Out[*]=



Obtenir les textes de wikipedia et les données de WikipediaData

L'accès à ces données est directe avec la fonction `WikipediaData[]`. Voici comment Wikipédia définit une ville en 5 phrases :


```
In[*]:= TextSentences[WikipediaData["city"]][[ ; 5]]
|phrases de texte |données Wikipedia
```

```
Out[*]= {A city is a large human settlement.,
  It can be defined as a permanent and densely
    settled place with administratively defined boundaries
    whose members work primarily on non-agricultural tasks.,
  Cities generally have extensive systems for housing, transportation,
    sanitation, utilities, land use, and communication.,
  Their density facilitates interaction between people, government
    organisations and businesses, sometimes benefiting
    different parties in the process, such as improving
    efficiency of goods and service distribution.,
  This concentration also can have significant negative consequences,
    such as forming urban heat islands, concentrating
    pollution, and stressing water supplies and other resources.}
```

Et encore plus de données externes

En fait, il est possible d'importer d'innombrables données, d'utiliser l'internet des objets, de relier Mathematica à une caméra et bien évidemment à d'autres langages. Par exemple, il est facile de collecter toute donnée connaissant son adresse mail. Voici un dernier exemple ludique. Tous les deux ans se tenait le colloque de géographie théorique et quantitative, Géopoint. Le petit programme ci-dessous donne une image de participants au colloque de 2008 :

```

In[ ]:= dupont =
  Import["http://www.groupe-dupont.org/ColloqueGeopoint/Geopoint08/Photos.htm",
  |_importe
    "Images"];
  image = dupont[[21]]

```

Out[]:=



La première ligne importe les diverses images et photos incluses dans le site web du Géopoint 2008, puis l' image 21 est sélectionnée. Le temps des espions est venu.

Bien d'autres possibilités existent pour récupérer des données utiles à une recherche géographique. Elles ne sont guère plus compliquées.

Conclusion

Le géographe peut accéder à de très nombreuses données de toute nature. Et Wolfram propose même, outre son immense base de connaissance, un dépôt, où il peut puiser ou rendre ses données accessibles à tous les utilisateurs du logiciel.