atmos_helpers.m Documentation

atmos_helpers.m is a *Mathematica* package containing several useful functions for interfacing with the atmospherics.lossofgenerality.com backend. It will be continually expanded as the need for new features arise. This guide, and the code of the package itself, is arranged by rough categories of function purposes.

The package can be called using the *Mathematica* Get command:

In[348]:=
```
(*<<< "http://atmospherics.lossofgenerality.com/static/atmos_helpers.m"*)
<<"C:\\Users\\Jacob Simmons\\My Work\\atmospherics\\atmospherics\\atmospherics\\static\\at
```

Meta functions

These functions are used to fetch information about the package itself.

AtmosHelp[]:

Calling this function will return usage information for all functions in the package.

Examples:

Get an interactive documentation block to learn about the available functions.

In[349]:=
```
AtmosHelp[]
```

▼ atmospherics`

Annotations	ArrayPixRegion	CoordValue	GeoFilter	InTimeRange	MergeData	TimeFilter
ArrayCoordRegion	AtmosHelp	CorrelateDatasets	GetDataset	LatToIndex	ParseDate	
ArrayDim	CoordToIndex	EncodeDate	InGeoRange	LonToIndex	QueryDataAPI	

Array index and coordinate functions

These functions are used to convert between geographical coordinates and indices within an array.

LatToIndex[lat, height, (roundDown)];

Calling this function with a latitude (ranging between -90 and 90) and the height of an array, returns the vertical index corresponding to the given line of latitude. The optional argument roundDown gives you the option of whether the function rounds the index up or down during the calculation.

Examples:

Find the row corresponding with the line of latitude 64 N in an 800x400 array

In[350]:=
```
LatToIndex[64, 400]
```

Out[350]= 342

LonToIndex[lon, width, (roundDown)];

Calling this function with a longitude (ranging between -180 and 180) and the width of an array, returns the horizontal index corresponding to the given line of longitude. The optional argument roundDown gives you the option of whether the function rounds the index up or down during the calculation.

Examples:

Find the column corresponding with the line of longitude 89 W in an 800x400 array

In[351]:=
```
LonToIndex[-89, 800]
```

Out[351]= 202

CoordToIndex[{lat, lon}, {height, width}, (roundDown)];

Essentially just a convenient syntax for calling LatToIndex and LonToIndex at the same time. Given a pair of coordinates and the dimensions of a rectangular array, returns the indices corresponding to that location. The optional argument roundDown gives you the option of whether the function rounds the index up or down during the calculation.

Examples:

Find the entry corresponding with the point (57 S, 48 E) in an 800x400 array

In[352]:=
```
CoordToIndex[{-57, 48}, {400, 800}]
```

Out[352]= {73, 506}

CoordValue[{lat, lon}, array];

Calling this function with a pair of coordinates and a rectangular array of geographical data, returns the

value of the array corresponding to the given point.

Examples:

Load a map image and Find the RGB pixel color corresponding with the point (57 S, 48 E)

In[353]:=
```
img =ImageData[  ];

CoordValue[{-57, 48}, img]
```

Out[354]= {0.337255, 0.396078, 0.235294}

ArrayDim[{latmin, latmax}, {lonmin, lonmax}, resolution];

Calling this function with a geographical region denoted by latitude and longitude boundaries, and a resolution value in degrees per pixel, returns the dimensions of the corresponding array. This function takes longitudinal rollover into account so the order of lonmin and lonmax is important.

Note that, because resolution is measured in degrees per pixel, larger numbers correspond with coarser images.

Examples:

Find the size of corresponding with the region between 151 E, 45 W and 45 S, 86 N at .5 degrees per pixel.

In[355]:=
```
ArrayDim[{-45, 86}, {151, -45}, 0.5]
```

Out[355]= {262, 392}

Array and coordinate region functions

These functions are used to fetch and learn about geographical regions. There is some conceptial overlap between this and the previous section so the choice of section for some of these functions may seem somewhat arbitrary.

ArrayPixRegion[array, {vmin, vmax}, {hmin, hmax}]:

Calling this function with a rectangular array and a pixel region, denoted by vertical and horizontal boundaries, returns the desired subarray. This function takes into account horizontal rollover so the order of hmin and hmax is important.

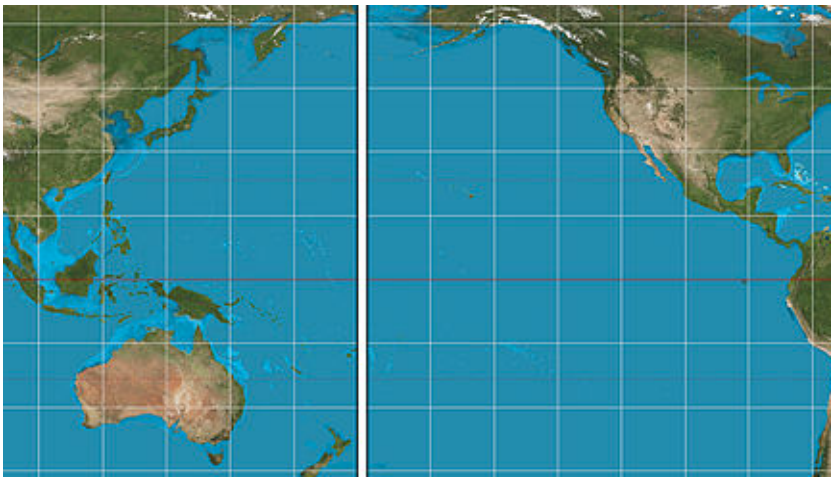
Examples:

Get an arbitrary section of a map image.

In[356]:=

```
img =ImageData[  ];
Image[ArrayPixRegion[img, {100, 344}, {614, 244}]]
```

Out[357]=



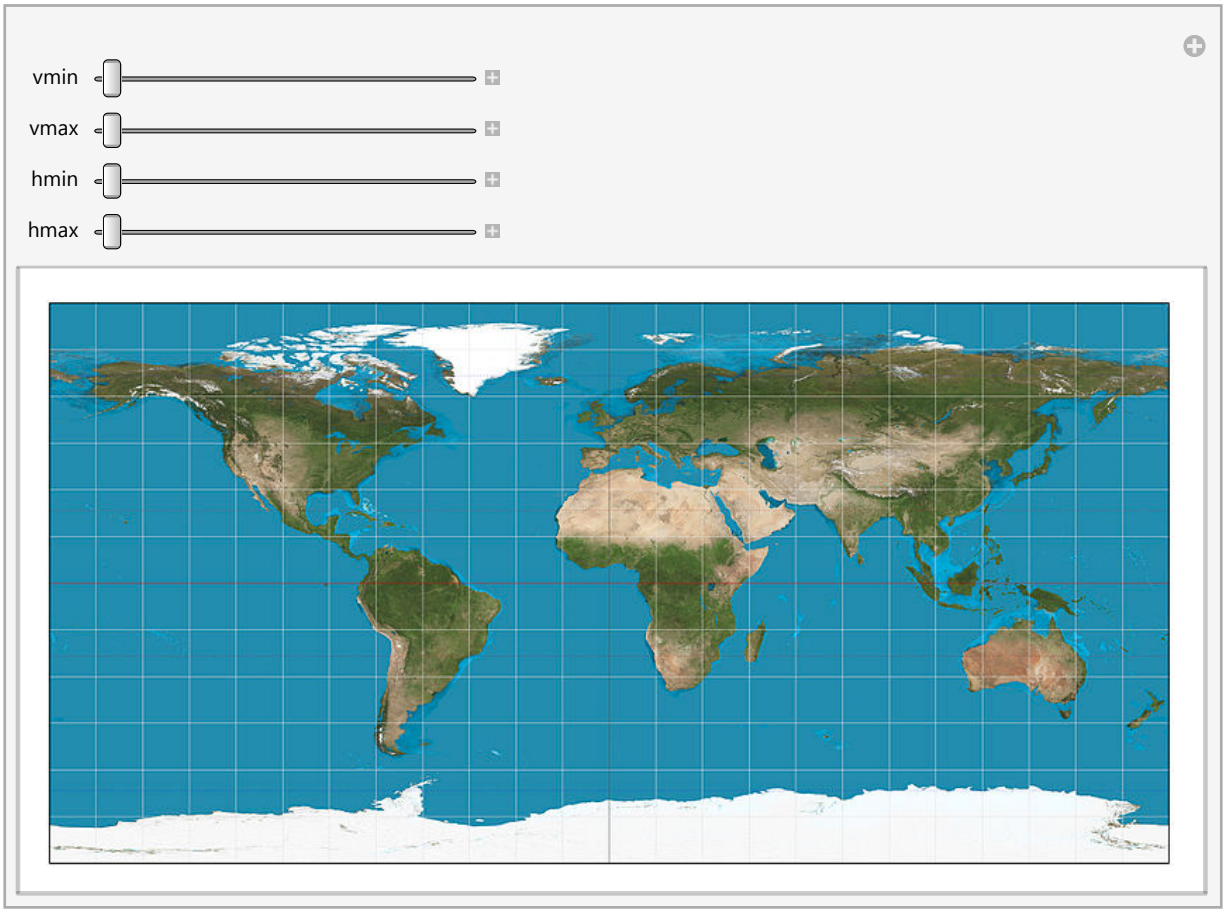
Interactively select a map area using Manipulate.

In[358]:=

```
img =ImageData[ ];

Manipulate[Image[ArrayPixRegion[img, {vmin, vmax}, {hmin, hmax}]],
    {vmin, 1, Dimensions[img][[1]], 1},
    {vmax, 1, Dimensions[img][[1]], 1},
    {hmin, 1, Dimensions[img][[2]], 1},
    {hmax, 1, Dimensions[img][[2]], 1}]
```

Out[359]=



ArrayCoordRegion[array, {latmin, latmax}, {lonmin, lonmax}]:

Exactly like ArrayPixRegion but accepts latitudes and longitudes rather than indices.

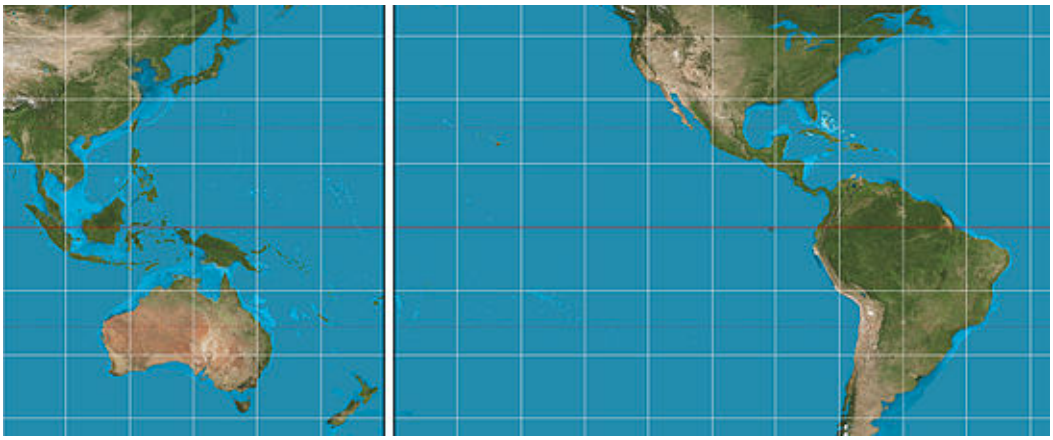
Examples:

Get an arbitrary region of a map image.

In[360]:=

```
img =ImageData[ <image> ];
Image[ArrayCoordRegion[img, {-48, 52}, {90, -25}]]
```

Out[361]=



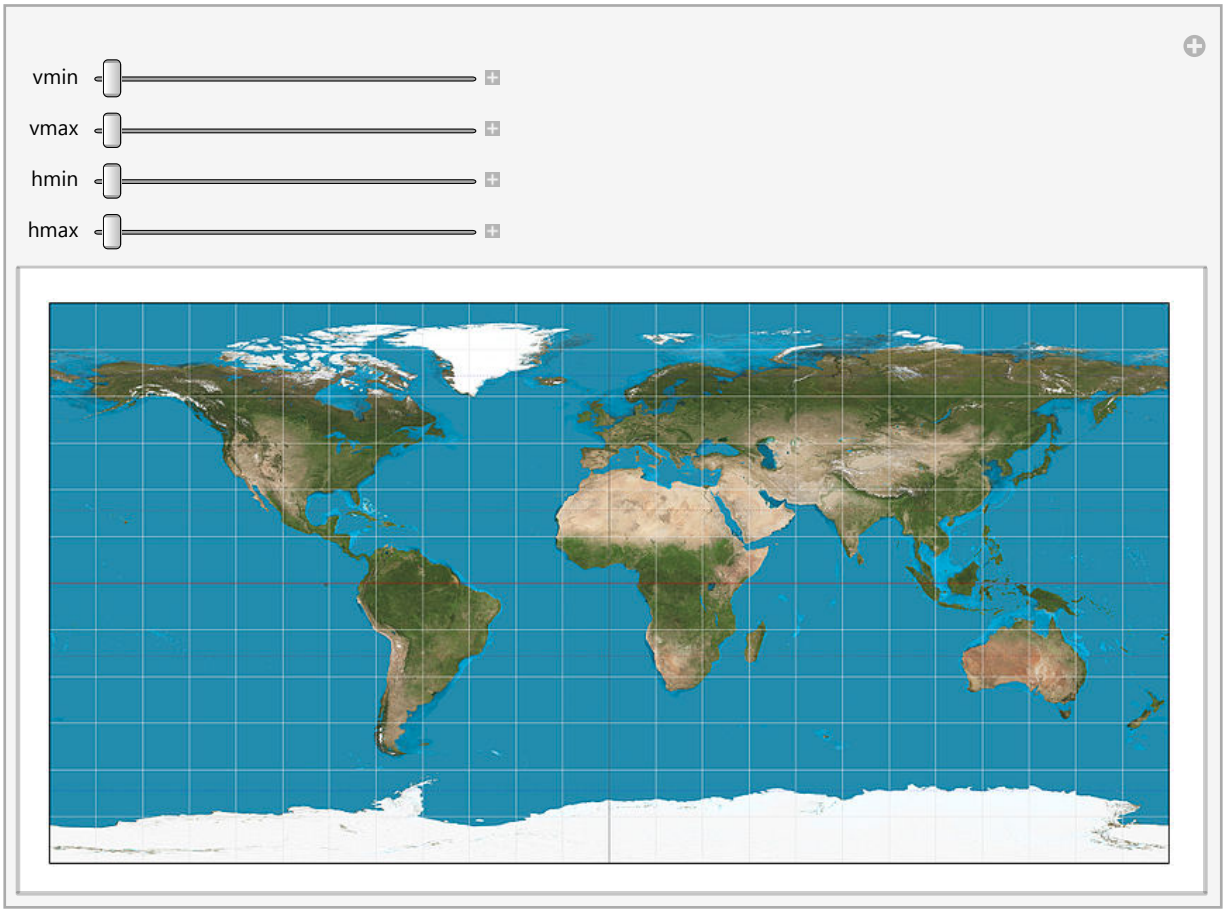
Interactively select a map region using Manipulate.

In[362]:=

```
img =ImageData[  ];

Manipulate[Image[ArrayCoordRegion[img, {vmin, vmax}, {hmin, hmax}]],
    {vmin, -90, 90},
    {vmax, -90, 90},
    {hmin, -180, 180},
    {hmax, -180, 180}]
```

Out[363]=



InGeoRange[{latmin, latmax}, {lonmin, lonmax}, coords];

Given a latitude and longitude range and a list of coordinates, returns whether any of the points lies in the region while accounting for longitudinal rollover. This is very useful for determining whether the contents of a file are of geographical interest.

Examples:

Find out if a file has points within a particular region

In[364]:=
```
coords = CorrelateDatasets["http://atmospherics.lossofgenerality.com/data/GPROF/GMI/2A
    {"/S1/Latitude","/S1/Longitude"}];
InGeoRange[{-48, 52}, {90, -25}, coords]
```

Out[365]= True

GeoFilter[{latmin, latmax}, {lonmin, lonmax}, datapoints, latindex, lonindex];

Given a latitude and longitude range and a list of data points which include latitudes and longitudes at the specified indices, returns a *flattened* list of any points which lie in the region while accounting for longitudinal rollover. This is useful for stripping contents of a file which are not of geographical interest. Note that, at this time, using this function strips some structure from the input data

Examples:

Find the first ten points in a file which lie within a particular region

In[366]:=
```
file = "http://atmospherics.lossofgenerality.com/data/GPROF/GMI/2A.GPM.GMI.GPROF2014v1
datapoints = CorrelateDatasets[file,
    {"/S1/Latitude","/S1/Longitude", "/S1/mixedWaterPath", "/S1/surfacePrecipitation"}
TableForm@GeoFilter[{-48, 52}, {90, -25}, coords, 1, 2][[;;10]]
```

Out[368]//TableForm=

-42.1273	89.9998
-42.1776	89.9781
-42.228	89.9572
-42.2786	89.937
-42.3294	89.9177
-42.3804	89.8992
-42.4315	89.8814
-42.4828	89.8645
-42.5342	89.8483
-42.5857	89.833

Datetime functions

These functions are used to make routine datetime conversions and other operations using datetime formats native to *Mathematica* and the atmospherics APIs.

ParseDate[dateString]:

Given a datetime string such as those given in extra_args, returns the corresponding Mathematica date list.

Examples:

Convert a datestring provided in the API format to a *Mathematica* date list.

In[369]:=
```
ParseDate["20130520124500"]
```

Out[369]= {2013, 5, 20, 12, 45, 0.}

EncodeDate[dateList]:

Given a Mathemematica date list, returns the corresponding datetime string as accepted by the data API.

Examples:

Convert some datelists created in the *Mathematica* to a usable API datestrings.

In[370]:=
```
EncodeDate[DateList[]] (* Now *)
EncodeDate[DateList[]-{0, 0, 0, 1, 0, 0}] (* Exactly one hour ago *)
```

Out[370]= 20141016172348

Out[371]= 20141016162348

InTimeRange[{startDateList, endDateList}, dateListsList]:

Given a start date list end date list pair and a list of date lists, retutns whether any of the test dates lie in the given timeframe. This is useful for determining whether the contents of a file are in a desired range.

Examples:

Determine whether a file contains data from the last three days

In[372]:=
```
file = "http://atmospherics.lossofgenerality.com/data/GPROF/GMI/2A.GPM.GMI.GPROF2014v1
end = DateList[];
start = DateList[]-{0, 0, 3, 0, 0, 0};
times = CorrelateDatasets[file,
    {"/S1/ScanTime/Year", "/S1/ScanTime/Month", "/S1/ScanTime/DayOfMonth", "/S1/ScanTi
];
InTimeRange[{start, end}, times]
```

Out[376]= False

TimeFilter[startDateList, endDateList, dateListsList, {dateListIndices}]:

Given a start date list end date list pair and a list of data points which include a date list constructed from the values at the specified indices, returns only the data points in the specified timeframe

Examples:

Find the time of all data points from the first 30 seconds of a data file.

In[377]:=
```
file = "http://atmospherics.lossofgenerality.com/data/GPROF/GMI/2A.GPM.GMI.GPROF2014v1
datapoints = CorrelateDatasets[file,
    {"/S1/ScanTime/Year", "/S1/ScanTime/Month", "/S1/ScanTime/DayOfMonth", "/S1/ScanTi
];
start = datapoints[[1, Range[6]]];
end = start + {0, 0, 0, 0, 0, 30};
TableForm @ (DateString[#] &/@ TimeFilter[{start, end}, times, Range[6]])
```

Out[381]//TableForm=

```
Thu 18 Sep 2014 13:31:47
Thu 18 Sep 2014 13:31:49
Thu 18 Sep 2014 13:31:51
Thu 18 Sep 2014 13:31:53
Thu 18 Sep 2014 13:31:55
Thu 18 Sep 2014 13:31:57
Thu 18 Sep 2014 13:31:59
Thu 18 Sep 2014 13:32:00
Thu 18 Sep 2014 13:32:02
Thu 18 Sep 2014 13:32:04
Thu 18 Sep 2014 13:32:06
Thu 18 Sep 2014 13:32:08
Thu 18 Sep 2014 13:32:10
Thu 18 Sep 2014 13:32:12
Thu 18 Sep 2014 13:32:14
Thu 18 Sep 2014 13:32:15
Thu 18 Sep 2014 13:32:17
```

Data Retrieval functions

These functions are used to fetch data from the atmospherics server.

GetDataset[filepath, dataset]:

Given a filepath and the name of a dataset or set of dataset names, returns the contents of that dataset.

Examples:

Fetch a dataset from the server.

In[382]:=
```
Image[GetDataset["http://atmospherics.lossofgenerality.com/data/NOAA_Data/air.sig995.1
```

Out[382]=

To get a list of the available datasets, use Import.

In[383]:=
```
Import["http://atmospherics.lossofgenerality.com/data/NOAA_Data/air.sig995.1968.nc"]
```

Out[383]= {lat, lon, time, air}

Annotations[filepath]:

Given a filepath, returns the annotations, if any, associated with the file. This can contain important information such as units, special values for missing data, etc.

Examples:

Fetch Annotations from the server.

In[384]:=
```
TableForm @ Annotations["http://atmospherics.lossofgenerality.com/data/NOAA_Data/air.s
```

Out[384]//TableForm=

units → degrees_north	actual_range → 90.
units → degrees_east	long_name → Longitude
units → hours since 1-1-1 00:00:0.0	long_name → Time
long_name → mean Daily Air temperature at sigma level 995	unpacked_valid_range → 1

QueryDataAPI[{ {key1, val1}, {key2, val2}... }]:

Given a list of key value pairs, returns the returns the links returned by the corresponding data API query.
The keys supported by the api are ‘start’, ‘end’, ‘ids’, ‘tag’, ‘org’, and ‘set’.

Examples:

Get urls of data tagged “gprof” from the last five hours

In[385]:=
```
start = EncodeDate[DateList[]-{0, 0, 0, 5, 0, 0}];
TableForm @ QueryDataAPI[{{"set", "GPROF-AMSR2"}, {"start", start}}]
```

Out[386]//TableForm=

```
http://atmospherics.lossofgenerality.com/data/GPROF/AMSR2/2A.GCOMW1.AMSR2.GPROF2014
http://atmospherics.lossofgenerality.com/data/GPROF/AMSR2/2A.GCOMW1.AMSR2.GPROF2014
http://atmospherics.lossofgenerality.com/data/GPROF/AMSR2/2A.GCOMW1.AMSR2.GPROF2014
http://atmospherics.lossofgenerality.com/data/GPROF/AMSR2/2A.GCOMW1.AMSR2.GPROF2014
```

Data transformation functions

These functions are used to fetch data from the server in a more useful format. This function preserves all additional structure in the source data, only performing the correlation at the lowest hierarchical level.

CorrelateDatasets[filepath, datasetList]:

Given a filepath and a list of dataset names, returns a list of datapoints represented by a list of values from each dataset.

Examples:

Get compoud data points from a file

In[387]:=
```
file = "http://atmospherics.lossofgenerality.com/data/GPROF/GMI/2A.GPM.GMI.GPROF2014v1
TableForm @ CorrelateDatasets[file, {"/S1/Latitude","/S1/Longitude", "/S1/surfacePreci
```

Out[388]//TableForm=

-39.9508	99.3535	-9999.
-39.9186	99.2996	-9999.
-39.887	99.2453	-9999.
-39.8558	99.1905	-9999.
-39.8251	99.1352	-9999.
-39.7948	99.0796	-9999.
-39.7651	99.0235	-9999.
-39.7359	98.967	-9999.
-39.7071	98.9101	-9999.
-39.6789	98.8527	-9999.