ITPASS
Information Training Program
with a Spirit of Self-help

データと可視化

高橋芳幸
神戸大学大学院理学研究科惑星学専攻
2020 年 9 月 29 日

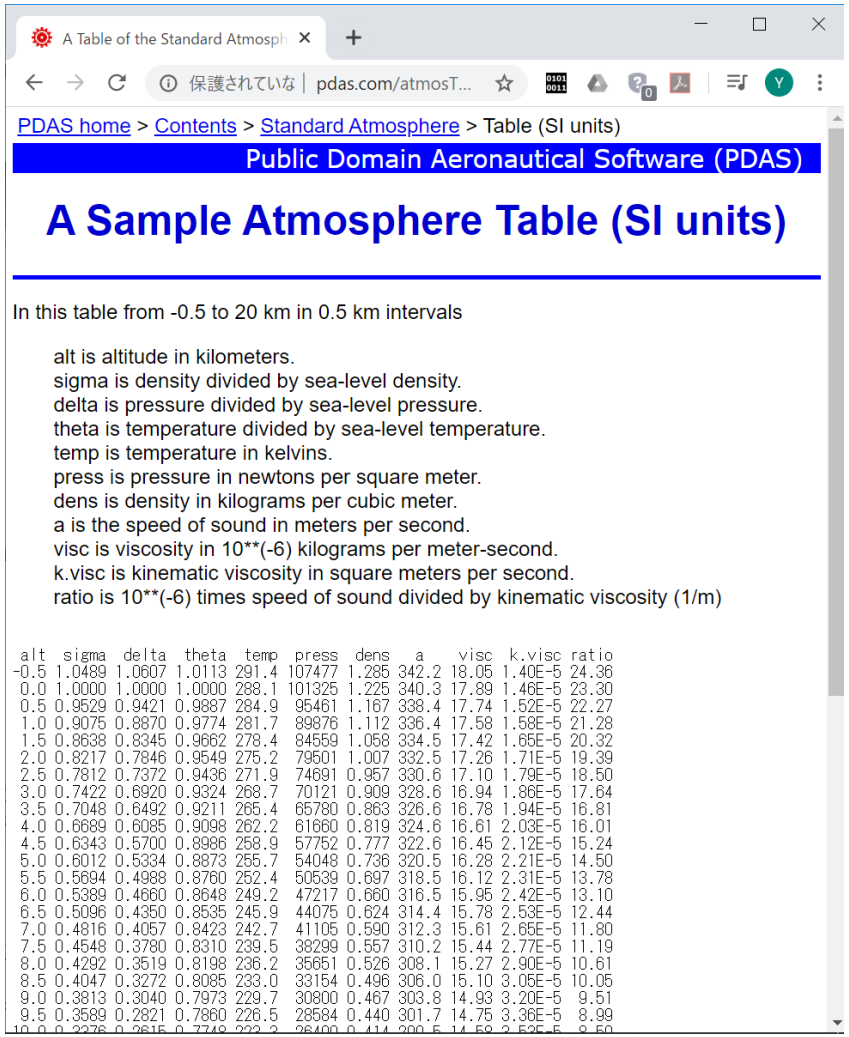
はじめに

- 惑星学/地球惑星科学は, 様々なデータが生成・取得され, 蓄積・保存され, それらの解析を通して様々な成果が生産されている.
- しかし, どんなに貴重なデータを生成・取得しても, 自分を含めて利用者が使いやすい形で蓄積・保存しなければ実際には利用されることはない.
  - 自分で取得・生成したデータでも, 時間がたつとその詳細を忘れてしまうことはよくある.
- ここでは惑星学/地球惑星科学で利用されるデータの形式について概観し, その中から特に NetCDF を取り上げて特徴を解説する. また, NetCDF を可視化するソフトウェアの一つとして GPhys について説明する.

よくありそうな ASCII で書かれたデータ

- 例: 米国標準大気 (US Standard Atmosphere)（右）
  - 米国標準大気拡張委員会によって定められた, 中緯度の「標準的な」大気構造.
- データの特徴
  - 数値が並んでいる.
    - この例では空白で値を区切っている
    - コンマで区切ることも多い (CSV ファイル)
      - CSV = Comma Separated Value
  - それぞれの数値の意味, 単位が書かれている.

- 右のデータは, 数値の意味や単位の説明が書かれている意味でとても良心的.
- しばしば情報がないこともある.
  - 「わからなかったら聞いてね」



US Standard Atmosphere 1976 (http://www.pdas.com/atmosTable2SI.html)

データのやりとりに必要なこと

- データを人に渡すためには, そのデータ(数値)が何であるかの情報が必要.
  - 次元 (軸, 座標)
  - 単位
  - 観測・計算情報
    - 観測者, 観測機器, 誤差
- 上記のような情報を「メタデータ」と呼ぶ.
  - メタデータ = metadata
  - データを説明するデータ
  - メタ : 高次の, 超越した, と言った意味の接頭語

アスキー (ASCII) 形式と バイナリー (Binary) 形式

- アスキー形式は見れば中身が分かる
- アスキー形式のデータはサイズが大きい
  - 4 byte で表現する範囲 (符号なし整数)
    - アスキー　　: 0-9999
    - バイナリー　: 0-4294967295
- バイナリー形式は一つではない
  - リトルエンディアン (little endian)　　: PC などで採用
  - ビッグエンディアン (big endian): 大型計算機で採用

エンディアン

- エンディアン
  - 複数のバイトからなるデータの各バイトを並べる順番
  - バイトオーダ (byte order) とも呼ぶ
- 例
  - 16 進数で 12 34 AB CD と表現される 4 バイトデータを考える
    - ちなみに上の 4 バイトを左から順に解釈すると:
      - 10 進数では 305441741
      - 2 進数では 0001 0010 0011 0100 1010 1011 1100 1101
  - このデータをバイトごとに格納する順番がエンディアンによって異なる
    - リトルエンディアン　　: CD AB 34 12
    - ビッグエンディアン　　: 12 34 AB CD

望ましいデータ形式

- メタデータとデータを同時に格納できる (「自己記述型」)
- データ量が小さい
- エンディアンに悩まされない
  - 別の計算機に持って行っても困らない (「ネットワーク透過」)

惑星学/地球惑星科学で用いられるデータ形式

- NetCDF (Network Common Data Format) [拡張子: .nc]
  - 大気海洋業界で使われている
  - UCAR (University Corporation for Atmospheric Research) の Unidata が開発.
- HDF (Hierarchical Data Format) [拡張子: .hdf, .hdf4, .hdf5, …]
  - 衛星観測結果や計算科学業界で使われている
  - 元は NCSA (National Center for Supercomputer Applications) が開発.
- GrADS (Grid Analysis and Display System) [拡張子: .ctl / .grd]
  - 大気海洋業界で使われている
  - George Mason University の Center for Ocean-Land-Atmosphere Studies (COLA) が開発.
  - データ形式の名前でもありデータを扱うソフトウェアの名前でもある
- GRIB (GRIdded Binary) [拡張子: .grib]
  - 気象業界で使われている
  - World Meteorological Organization (WMO) が開発.

など

NetCDF 形式で保存されるデータの例

- NCEP/NCAR 再解析データ
  - NCEP および NCAR が作成している再解析 (Reanalysis) データ
    - NCEP = National Centers for Environmental Prediction
    - NCAR = National Center for Atmospheric Research
  - 再解析とは, 数値モデルの物理的整合性と観測データの現実の値である利点を組み合わせて, より良い推定結果として場を構築 (データ同化) すること.
- 金星探査機「あかつき」観測結果 (の一部)
  - Level 3 データ (何段階か処理した後のデータ) など
- 惑星大気大循環モデル DCPAM 計算結果
  - DCPAM: 地球流体電脳倶楽部有志が開発している惑星大気大循環モデル

…

NetCDF データの内容の確認

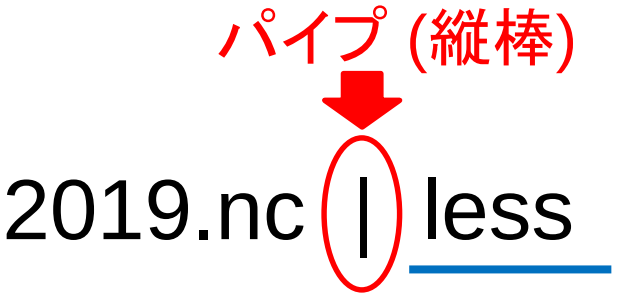
- NetCDF データはバイナリー形式のため, cat/more/less/vi/emacs などでは内容を確認できない.
- NetCDF データの内容を確認するための方法が準備されている.
  - Unix/Linux コマンド
    - ncdump
  - Fortran/C/C++/Java API (Application Programming Interface)
    - 例 (Fortran90)
      - NF90_OPEN(…)
      - NF90_INQ_DIMID(…)
      - NF90_INQ_VARID(…)
      - NF90_GET_ATT(…)
      - NF90_GET_VAR(…)
      - NF90_CLOSE(…)

NetCDF 形式のデータの内容を見てみる

- コマンドラインで ncdump コマンドを用いることで NetCDF ファイルの内容を確認.
- ncdump の使い方

  $ ncdump <ファイル名>

- 例



パイプで more/less/lv などを使うことを推奨.
たいていの場合は出力結果がとても長いため.

上のファイルは ftp://ftp.cdc.noaa.gov/Datasets/ncep.reanalysis.dailyavgs/pressure/air.2019.nc

出力結果

```
netcdf air.2019 {
dimensions:
        level = 17 ;
        …
variables:
        float level(level) ;                          次元変数の型・サイズ
                level:units = “millibar" ;            次元変数の属性
                …
        …
        float air(time, level, lat, lon) ;            変数の型・サイズ
                air:long_name = "mean Daily Air temperature" ;
                …                                     変数の属性
        …
// global attributes:
                :Conventions = "COARDS" ;
                …
data:

level = 1000, 925, 850, 700, 600, 500, 400, 300, 250, 200, 150, …
…                                                     次元変数の値
air =
 245.8, 245.8, 245.8, 245.8, 245.8, 245.8, 245.8, 245.8, 245.8,
 …                                                    変数の値
}
```

次元,
次元サイズ

変数の情報

大域属性

メタデータ

変数の値

NetCDF データを解釈して可視化できるソフトウェア

- NetCDF は自己記述型データ形式であるため, メタデータを解釈すれば「簡単に」図が描ける.
- NetCDF を解釈する可視化ソフトウェア
  - GPhys ((a class of multi-dimensional) Gridded Physical quantities)
    - 地球流体電脳倶楽部の有志によって開発.
  - GMT (The Generic Mapping Tools)
    - 固体系の研究者に良く使われている印象.
  - GrADS (Grid Analysis and Display System)
    - 大気海洋業界で使われている.
  - IDL (Interactive Data Language)
    - 天文, 超高層, 大気海洋, … など様々な業界で使われている
    - 有料
  - Panoply
    - 大気海洋を含む地球科学業界で使われている.
    - GUI で動作する
  - その他たくさんのソフトウェアが http://www.unidata.ucar.edu/software/netcdf/software.html で紹介されている.

GPhys

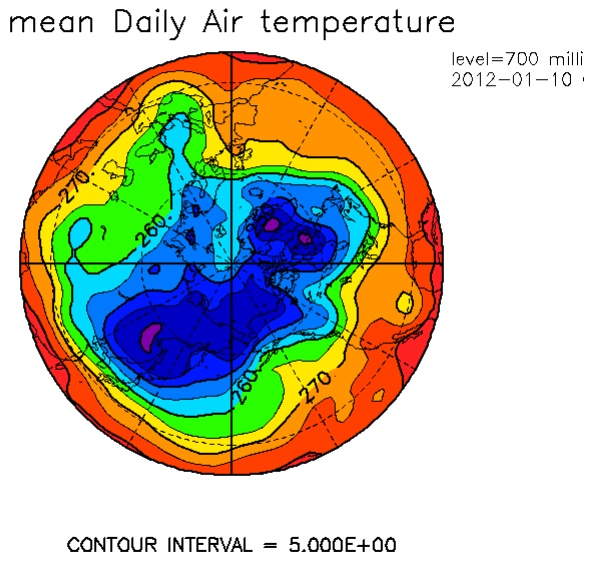
- 地球流体電脳倶楽部有志が開発した可視化ソフトウェア
  - http://www.gfd-dennou.org/library/ruby/products/gphys/
- スクリプト言語 Ruby を用いて内部で DCL を使い, NetCDF データを解釈して「便利に」可視化するソフトウェア
  - Ruby
    - まつもとゆきひろ氏が開発したオブジェクト指向プログラミング言語
  - DCL (Dennou Club Library)
    - 1990 年代から地球流体電脳倶楽部の有志によって開発された Fortran で作図するライブラリ

オブジェクト指向プログラミング言語とオブジェクト

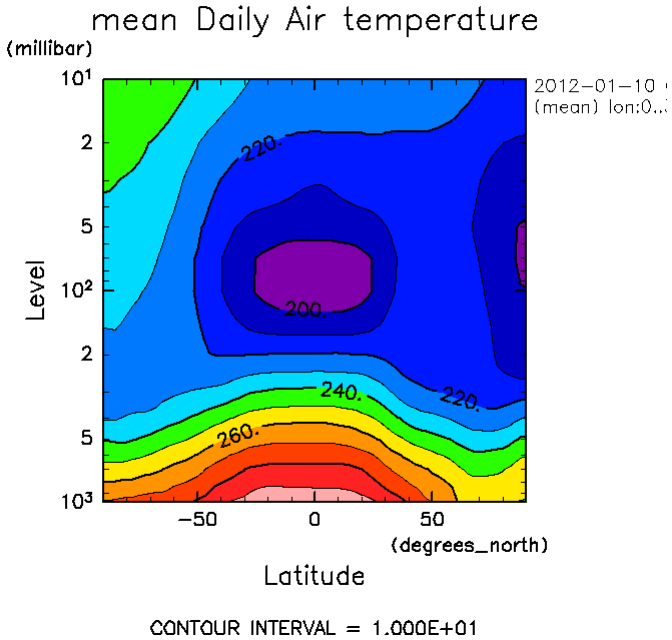
- オブジェクト指向プログラミング言語
  - 「オブジェクト」と称するモノの扱いを中心に考えたプログラミングスタイルを可能にする言語
  - 大規模ソフトウェアの開発に有利と考えられている
  - 普及しているオブジェクト指向言語
    - C++, Java, Javascript, PHP, Ruby, …
- オブジェクト
  - 値だけでなく, 値の属性や値を処理する方法 (メソッド) の情報も保持するモノ
  - オブジェクトは, 自身に対して可能な処理をそれ自身が知っている
    - Ruby では .methods で問い合わせれば分かる

GPhys を用いた可視化例

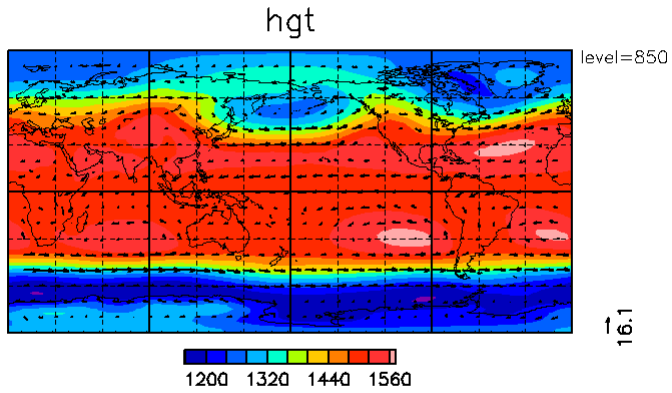
700 hPa 面の温度分布



東西平均温度分布



850 hPa 高度分布



GPhys を使った ruby スクリプトの概要 最小版

```
require "numru/ggraph"
include NumRu

gp = GPhys::IO.open("air.2019.nc", "air“)


DCL.gropn(1)
GGraph.set_fig('itr'=> 1)

GGraph.tone(gp, true)
GGraph.color_bar

DCL.grcls
```

「決まり文句」
ライブラリの読み込み

NetCDF ファイルから
変数 “air” を読む

画面を開く（open）
描画画面を準備

トーンで描画

カラーバーを描画

画面を閉じる（close）

GPhys を使った ruby スクリプトの概要 最小版

```
require "numru/ggraph"
include NumRu

gp = GPhys::IO.open("air.2019.nc", "air")

DCL.gropn(1)
GGraph.set_fig('itr'=> 1)

GGraph.tone(gp, true)
GGraph.color_bar

DCL.grcls
```

メタデータ・数値を含む GPhys オブジェクト

「決まり文句」
ライブラリの読み込み

NetCDF ファイルから
変数 “air” を読む

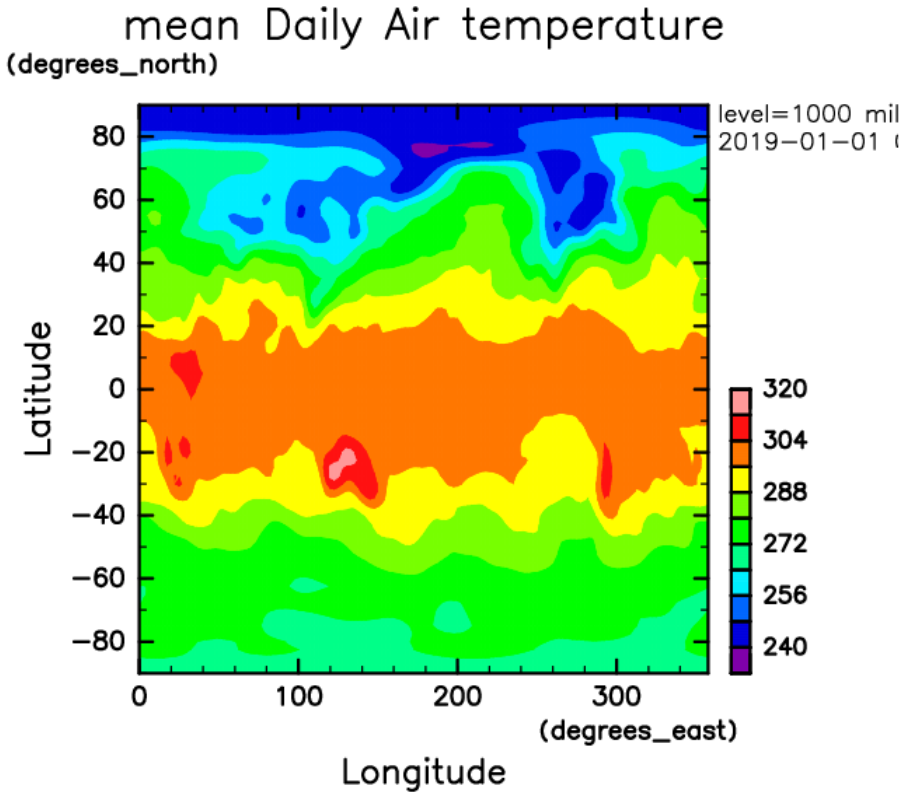
画面を開く（open）
描画画面を準備

トーンで描画

カラーバーを描画

画面を閉じる（close）

可視化結果: 温度
1000 hPa, 2019年1月1日の経度-緯度分布



次元の指定 1-1

$ ncdump air.2019.nc の結果.

```
netcdf air.2019 {
dimensions:
        level = 17 ;
        …
variables:
        float level(level) ;
                level:units = “milliba
                …
        …
        float air(time, level, lat, lon) ;
                air:long_name = "mean Daily Air temperature" ;
                …
        …
// global attributes:
                :Conventions = "COARDS" ;
                …
data:

level = 1000, 925, 850, 700, 600, 500, 400, 300, 250, 200, 150, …
```

変数 air は 4 次元
(lon, lat, level, time)
(経度, 緯度, 鉛直層, 時間)
(Fortran とは) 順番が
逆であることに注意

- 3 次元分布は描画できない!
- 少なくとも一つの次元の値を指定する必要.
  - 指定しない場合には, 最初から二つの次元 (経度(lon), 緯度(lat)) の分布が描画される.
    - 3 次元目以降 (鉛直層(level), 時間(time)) は最初の値が採用される.

次元の指定 1-2

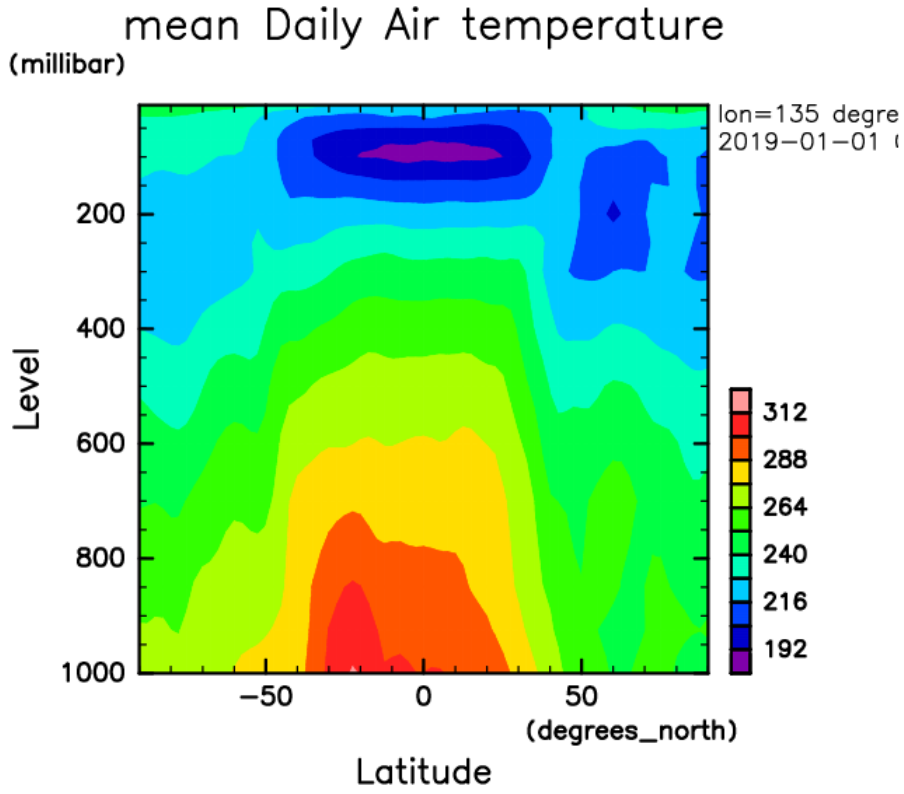
```
require "numru/ggraph"
include NumRu

gp = GPhys::IO.open("air.2019.nc", "air“)
gp = gp.cut(‘lon’=>135)

DCL.gropn(1)
GGraph.set_fig('itr'=> 1)

GGraph.tone(gp, true)
GGraph.color_bar

DCL.grcls
```

経度 (lon) の指定
lon 軸を切り出す (cut)

結果として,
緯度 (lat), 鉛直層 (level)
の 2 次元分布が描画される.
時間 (time) 次元は最初の
値が採用される.

次元の指定 1-3
可視化結果

135˚E, 2019 年 1 月 1 日の温度の緯度-圧力分布



次元の指定 2-1

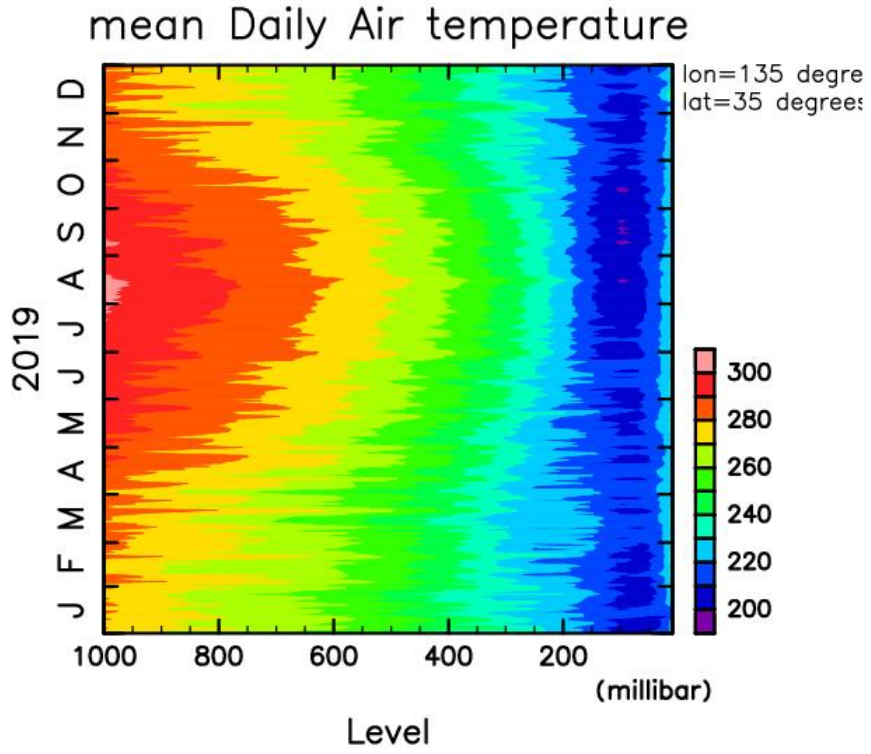
```
require "numru/ggraph"
include NumRu

gp = GPhys::IO.open("air.2019.nc", "air“)
gp = gp.cut(‘lon’=>135, ‘lat’=>35)

DCL.gropn(1)
GGraph.set_fig('itr'=> 1)

GGraph.tone(gp, true)
GGraph.color_bar

DCL.grcls
```

経度 (lon), 緯度 (lat) の指定
lon, lat 軸を切り出す
gp.cut(‘lon’)=>135).cut(‘lat’=>35)
と重ねることもできる

結果として,
鉛直層 (level), 時間 (time)
の 2 次元分布が描画される.

次元の指定 2-2
可視化結果

135˚E, 35˚N の温度の圧力-時間分布



次元の指定 3-1

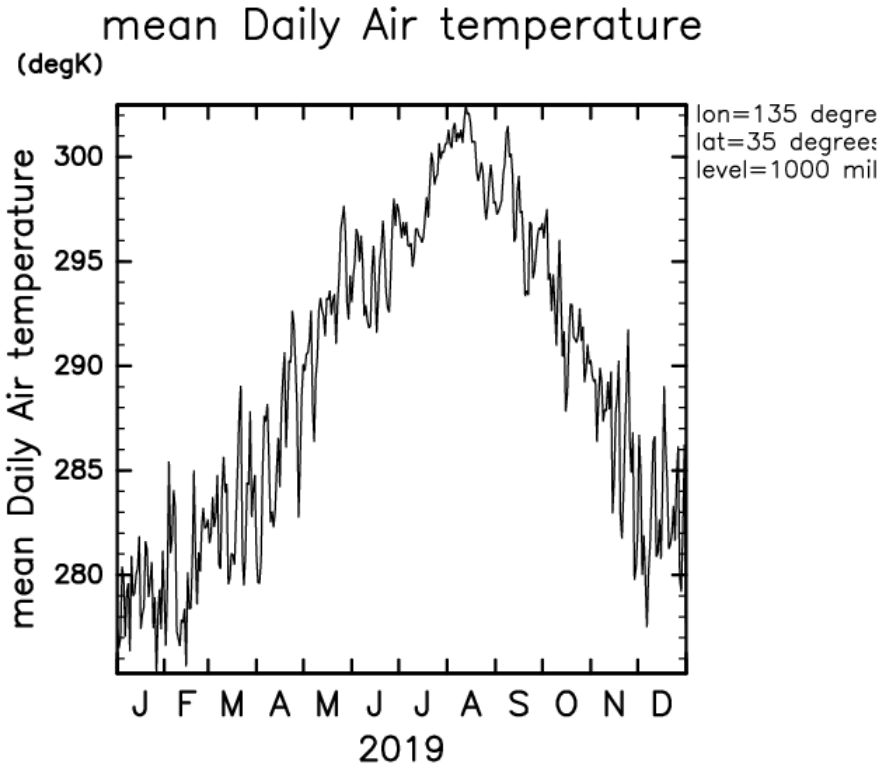
```
require "numru/ggraph"
include NumRu

gp = GPhys::IO.open("air.2019.nc", "air“)
gp = gp.cut(‘lon’=>135, ‘lat’=>35, ‘level’=>1000)

DCL.gropn(1)
GGraph.set_fig('itr'=> 1)

GGraph.line(gp, true)

DCL.grcls
```

経度 (lon), 緯度 (lat), 鉛直層 (level) の指定
lon, lat, level 軸を切り出す (cut)

1 次元分布となるので, トーンではなく,
線グラフとする.

次元の指定 3-2
可視化結果

135˚E, 35˚N, 1000 hPa の温度の時間分布



平均 1

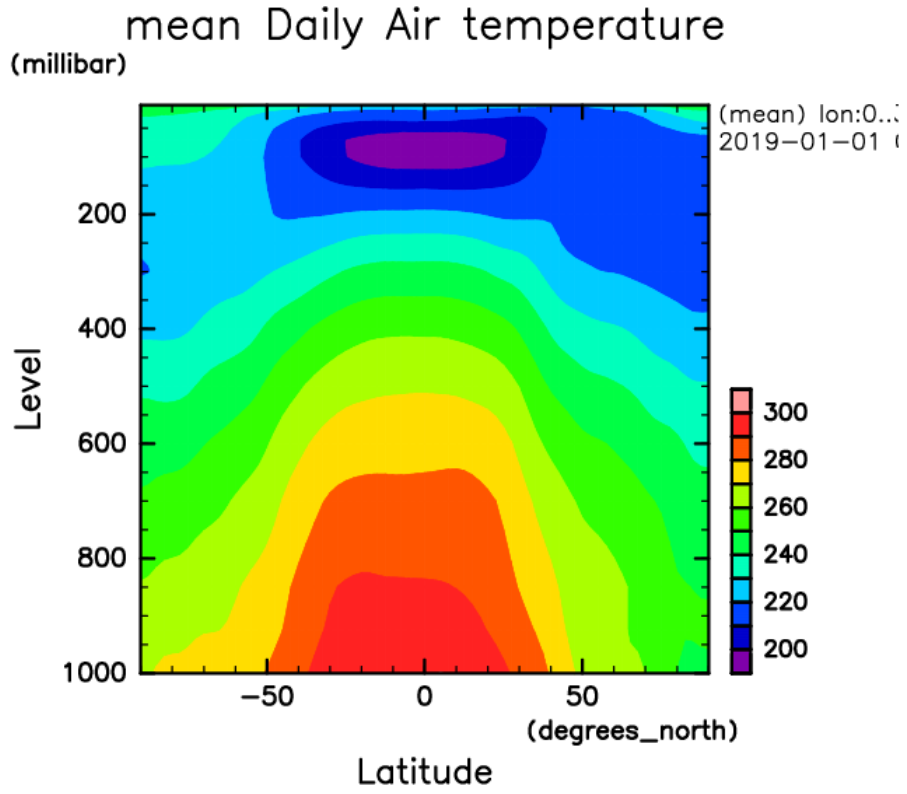
```
require "numru/ggraph"
include NumRu

gp = GPhys::IO.open("air.2019.nc", "air“)
gp = gp.mean(‘lon’)

DCL.gropn(1)
GGraph.set_fig('itr'=> 1)

GGraph.tone(gp, true)
GGraph.color_bar

DCL.grcls
```

経度 (lon) 軸に対して平均

結果として,
緯度 (lat), 鉛直層 (level) の 2 次元分布が描画される.
時間 (time) 次元は最初の値が採用される.

平均 2
可視化結果

135˚E, 2019 年 1 月 1 日の東西平均温度の緯度-圧力分布



まとめ

- データを使いやすくするためには, データを説明する情報 = メタデータが不可欠.
- メタデータを伴うデータ形式 = 自己記述型データ形式として, 惑星学/地球惑星科学業界では, NetCDF, HDF などが用いられている.
- 自己記述型データを解釈する可視化ソフトウェアとしては, GPhys, GMT などがある.
- 実習では, NetCDF と GPhys を使って大気のデータに触れ, 大気場の分布を可視化してみましょう.

参考文献

- 2019 年 北海道大学 情報実習 INEX「地球惑星情報学 大気大循環モデル」
  - http://www.ep.sci.hokudai.ac.jp/~inex/y2019/0712/lecture/pub/190712-inex-dcapm_v02.pdf
- GPhys - a multi-purpose class to handle Gridded Physical quantities
  - http://www.gfd-dennou.org/library/ruby/products/gphys/