2012-03-05

GPhys / GGraph

手軽で本格的な多次元データ解析可視化のためのRubyライブラリ

堀之内　武 （北大地球環境）

GPhys概要

- 多次元(0,1次元も含む)空間における物理量 (語源: Gridded Physical quantity)
- 単位と座標(と名前)を持つ数値データを扱う. 任意メタデータ(属性)をもてる.
- 多様なデータ実体(in 各種外部ファイル / on 実行時メモリ)を隠蔽し統一された操作を実現 (キャッチフレーズ「データ形式が違っても, 同じことは *"同じような"* ではなく *"同じ"* プログラムでできる」)
- 地球流体科学／気象学で標準的な様々なデータ形式をサポート(拡張性大. 概念的に同様なデータモデルを使う科学分野は多いが, その分野のデータ形式を解釈するモジュールは作成しないとならない.)
- 地球流体科学／気象学で用いられる計算や可視化をサポートする応用ライブラリ群を持つ
  - GSL(GNU Scientific Library) , LAPACK, FFTW3,…の利用も
- とっても動的な言語であるRubyのクラスライブラリなので利用者が好きに拡張・カスタマイズできる

GGraph概要

- GPhys付属の可視化ライブラリ
  - GPhysの応用ライブラリの一つという位置づけ（可視化にGGraphを用いなくても良い）
- DCL (Dennou Club Library) のRubyインターフェースRubyDCLを利用
  - DCLの諸関数と協調して使うようになっている (※もう少しDCLを直接呼ばないですむようにしてほしいという声もあるので今後検討したい)
- クイックルックから論文清書レベルまで対応.（凝り方の度合いにリーズナブルに応じた手数で – 凝ろうと思うと急激にプログラムが長くなったりしないようにできている）
  - GPhysの機能を使ってすばやい描画 – 対象を限定する（~GPhysのデータモデルにフィットする）ことで手厚いケアを実現
- 気象な分野で比較すると：GrADS並み（or以上）に簡単に描画できる.（対話的描画／プログラムによる描画）GrADSと違ってできることは無限

GPhysについて

サンプルコード

スタートアップファイルによる設定簡略化をしない場合

```
require ”numru/ggraph”   #可視化なしなら”numru/gphys”で良い
include NumRu      # 名前空間確保用のNumRuをつけなくていいよう
gp = GPhys::IO.open 'air.2011.nc', 'air'     # データ読込
gp = gp[false,0..10,true]
        #^ 後ろから2番目の軸(高度)の添え字で絞込み(バーチャル)
gp = gp.cut(”lon”=>0..120)   # 座標値で絞込み(バーチャル)
gpmean = gp.mean(”time”)
        #^ 時間(最後の軸)について平均(ここで初めてデータ読み込み)
        #   結果はメモリ上に. でもクラスはGPhysのまま
GGraph.tone gpmean     # (GGraphによる) 色塗り描画
DCL.grcls
```

上記のrequire と include (いつものおまじない)のあと:

```
gp = GPhys::IO.open 'T.ctl', 'T'     # データ読込(GrADS形式)
file = NetCDF.open(”air.mean.nc”)    # NetCDFファイル
GPhys::IO.write(file, gpmean)        # メタデータ含めコピー
file.close                           # ファイル閉じ
```

サンプルコード

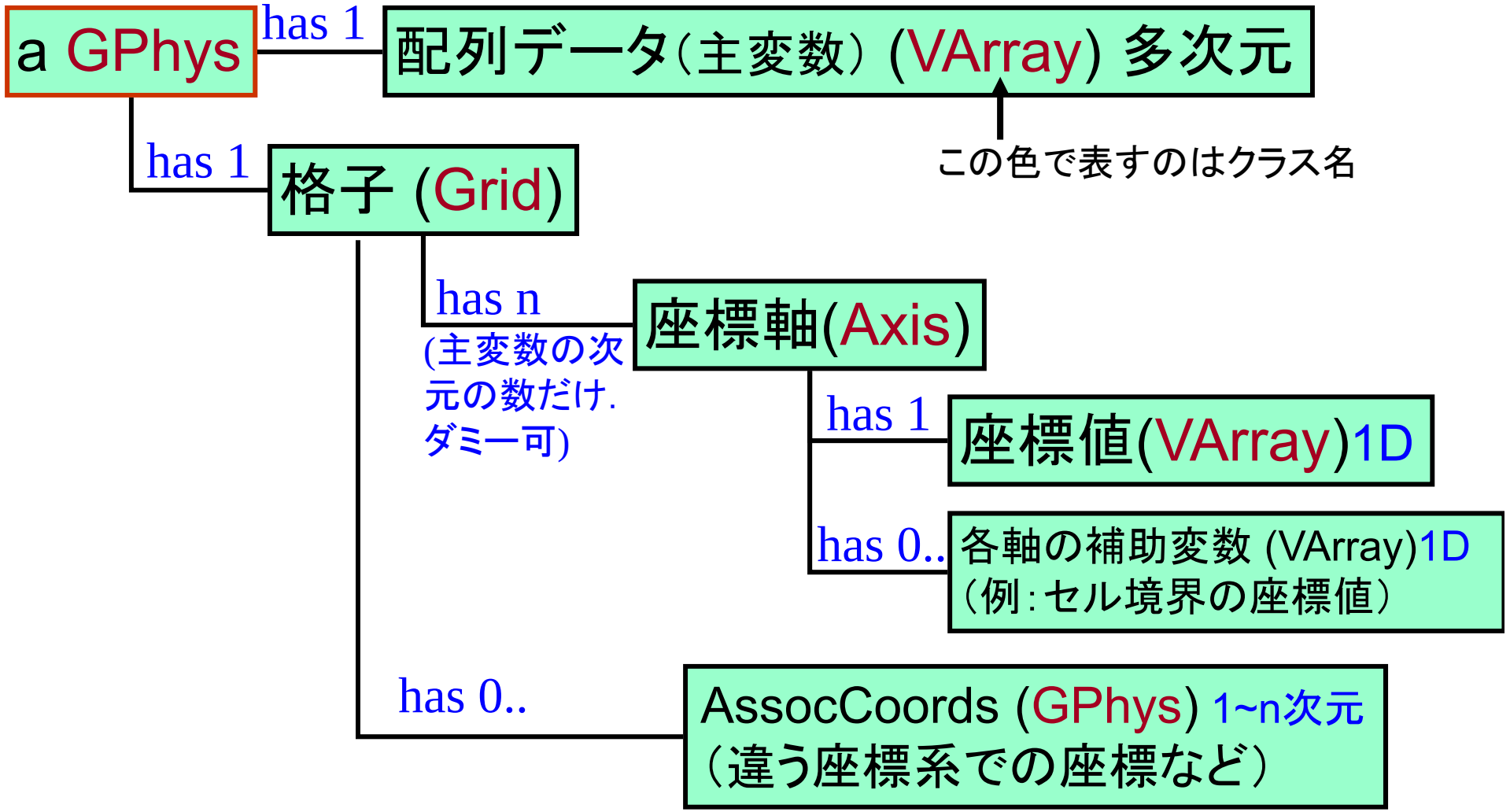
lon, lat,.., time なデータから東西波数-振動数2次元スペクトルを求める

```
fc = gp.detrend(-1).fft(false, 0,-1)
     # 最後の次元（時間）でトレンド除去し，最初(0)と最後(-1)の次元で2D
     # FFT．座標軸はちゃんと波数に置き換わる．
sp = (fc.abs**2).rawspect2powerspect(0,-1).
                 spect_zero_centering(0).spect_one_sided(-1)
       # 帯状波数-振動数なスペクトル導出（単純スペクトル→パワースペ
       # クトル → 東西波数ゼロを中心に→振動数片側に，の連鎖で）
```

EOF (経験的直交関数の計算，主成分分析)

```
eofs, rates = gp.eof("time", "norder"=>3)
  #（多次元の）時系列データより寄与率第3位までのEOFを求める
```

GPhys の構成

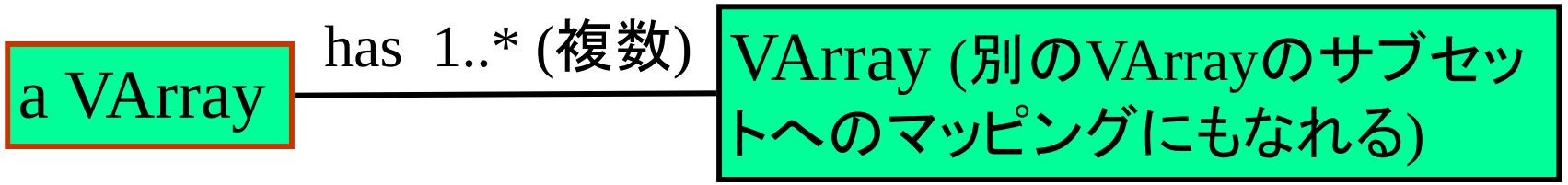

a GPhys
has 1
配列データ(主変数) (VArray) 多次元
この色で表すのはクラス名
has 1
格子 (Grid)
has n
(主変数の次元の数だけ. ダミー可)
座標軸(Axis)
has 1
座標値(VArray)1D
has 0..
各軸の補助変数 (VArray)1D
(例:セル境界の座標値)
has 0..
AssocCoords (GPhys) 1~n次元
(違う座標系での座標など)

VArray クラス

- Virtual Array（バーチャルな配列，配列のようなもの）．GPhysの「カプセル化」最下層．データ形式ごとにアダプターを持つ．
- 配列のように振舞うが，Ruby用多次元配列(NArray)や外部ファイル中の配列様変数などをラップ（参照ポインターを持つ）．他のVArrayのサブセットだったり，複数のVArray合成の場合も．
- NetCDF同様「属性」を持てる．単位，名前あり．

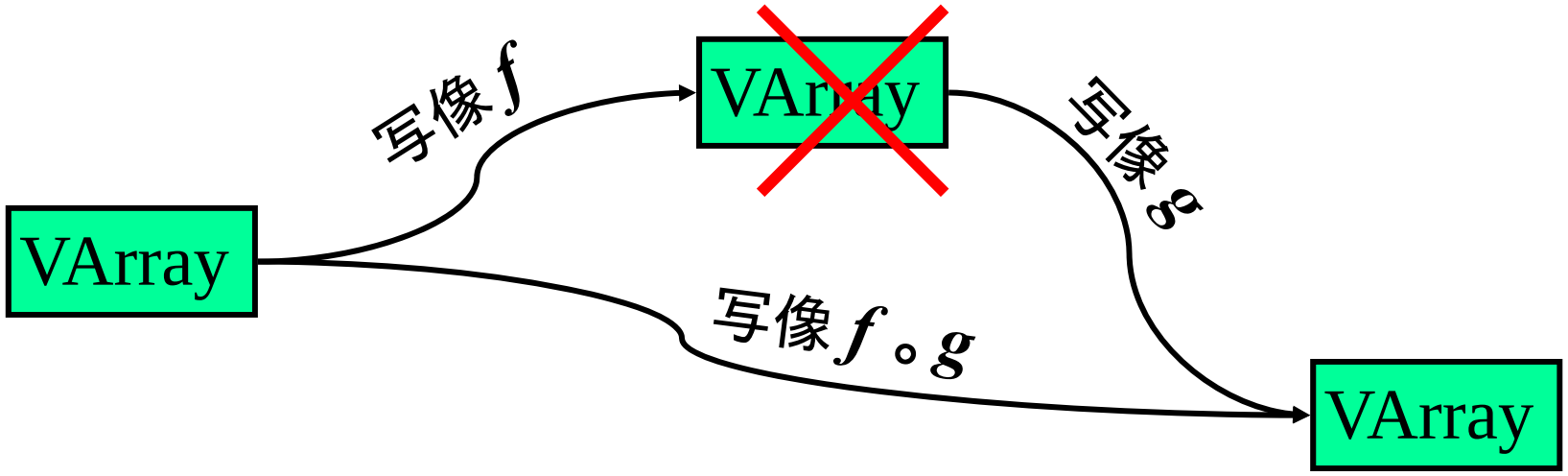
パターン1

パターン2



VArrayにおけるサブセットマッピング

- 写像(Mapping)の合成により、常に1段に保つ（連鎖が際限なく長くならないように）。
- サブセット切りだし（va[0..10,3]などとする）は、サブセットマッピングで行なう。よって、
  - 効率的
  - ファイルなどに対する部分的な書き込みに適する
- （同じことだが）値を取り出す命令が呼ばれるまでサブセットの本当の切り出しは遅延される。（NArrayも同様になる？）



GPhys::IO モジュール

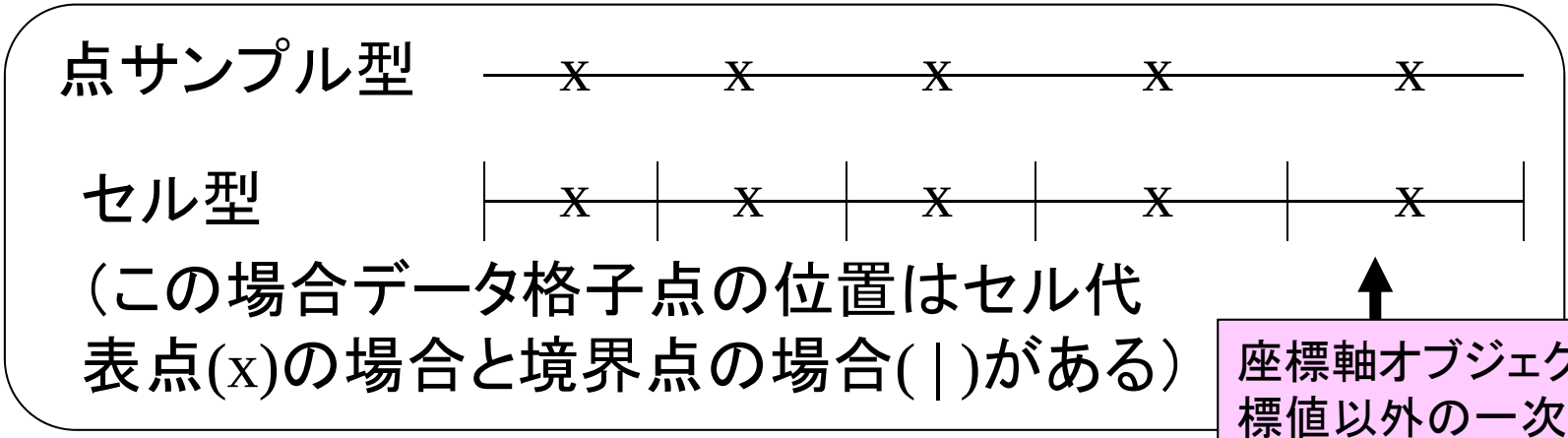
- 入出力ハンドラ.
- 各データ形式ごとに下請けをもつ.
- ファイルオープン時：各形式中の配列（やメタファイル中のデータ）を，どのようにGPhysの構造に当てはめるか等を担当

GPhysが標準で自動対応するファイル形式

- NetCDF: 名前ベースで辿る，構造化されたバイナリファイル形式．大気海洋業界（アカデミック界）標準．Self-consistentな「自己記述的」ファイル形式．
- GRIB (GRIB1. GRIB2も対応しているが大幅改訂見込み): 世界気象機関(WMO)が定める気象現業機関用データ形式，水平2次元スライス（の集合），極度に符号化されたヘッダ（解釈はアプリケーション）．-- GPhysではwgribにおける名付けを援用してIDベースでなく名前ベースでのアクセスを実現
  - NuSDAS: 日本の気象庁内で主に使われているデータ形式．GRIBをより高度にしたようなもの（というと語弊あるか…）
- GrADS: 可視化ツールGrADS用データ形式．バイナリファイルにメタデータテキスト(data descriptor file; 通称コントロールファイル)を添える．座標値はこれに記入（GrADSなので「気象」的な限定強）．
- Gtool3: 故沼口敦氏作成のAGCM5やその流れを汲むGCM (CCSR/NIES…等）で使われるデータ形式．固定長ヘッダと3次元バイナリデータの繰り返し．座標軸定義は別ファイル．
- HDF-EOS5: 地球観測衛星データに近年よく用いられるデータ形式．HDF5ベース．（GPhysが対応しているのは衛星Swathデータ用の形式のみ）

座標軸(Axis)についてより詳しく

- ３種類想定：
  - 点サンプル型
  - セル型
  - 単なる順番並び型（これは座標とはいえないが）



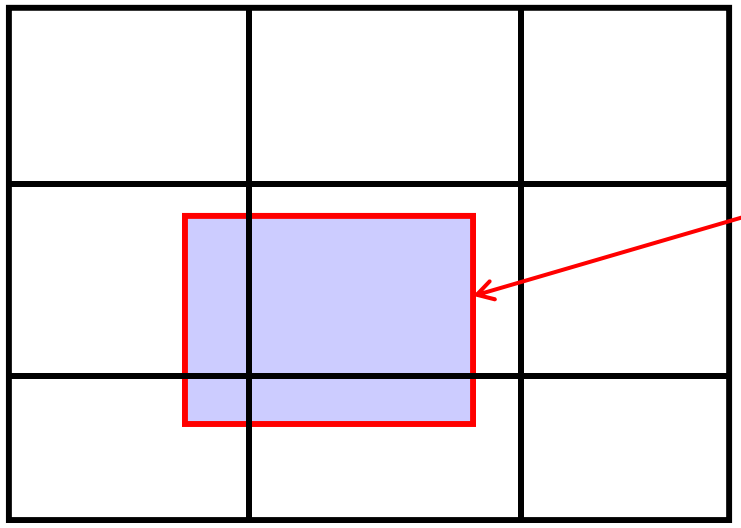
- 例えば、軸に沿っての積分をどういう方法で実行するかは、その軸が「知っている」。GPhysは積分を要求されるとGridに投げ、GridはさらにAxisに投げる。デフォルト積分法は、点サンプル型及びセルの境界点の場合は台形公式、セル代表点なら各々のセル長を掛けて和を取る（ユーザー設定可）。

AssocCoords – 補助的な(別の)座標

- 複数の座標系での表現を担う
- フォーマルには座標軸で主変数に関連付けられるGPhysオブジェクト.（格子(Grid)が直接持つ座標軸群（以下主座標軸と表現）のサブセットを座標軸に持つGPhysオブジェクト）
- 例:
  - 主座標軸:x, y (Cartesian) – AssocCoordsの座標:原点からの距離 r(x,y), 方位角φ(x,y) (極座標)
  - 主座標軸 lon, lat, p, t – AssocCoordsの座標:温位 θ(lon,lat,p,t)
  - 主座標軸（ゾンデの）観測時刻 t – AssocCoordsの座標:観測高度 z(t)
- GPhysのメソッド cut（座標値による切り出し）は, AssocCoordsでの切り出しにも対応

タイリング

- タイル状に分割されたデータを、ひとつのGPhysとして扱える。期間毎にわかれた時系列のファイルや、領域分割シミュレーションの出力ファイルをそのまま一体として扱える（VArrayCompositeを利用）。
- サブセットを取る場合、下図のような状況に正しく対処する。
- ファイルの自動処理は現在NetCDFのみ（早く全てに広げたい）。ファイルとして配列か正規表現を与える（後者の例：/data_x(¥d)y(¥d).nc/ for data_x0y0.nc, data_x0y1.nc, data_x1y0.nc, data_x1y1.nc）。



タイル分割されたデータへのサブセット切りだし要求への対応に関する概念図

大きなデータへの対応

- 遅延処理が基本(cutや[]による切り出しはバーチャル．データ読み込みは必要になるまで遅延．)
- メモリーに一度に読み込むには大きすぎるデータの処理用に、処理を自動分割するイテレーター有
  - 結果も巨大なケースが多いので、結果をファイルに書き出す GPhys::IO.each_along_dims_write など
    - 利用例：
      - イテレーターなしの場合：

```
in = GPhys::IO.open(infile, varname)
ofile = NetCDF.create(ofilename)
out = in.mean(0)      #  ここでデータが一遍にメモリ上に読み込まれる
GPhys::IO.write( ofile, out )
ofile.close
```

      - 同じことをイテレーターを使って、最終次元に関しループで：

```
in = GPhys::IO.open(infile, varname)
ofile = NetCDF.create(ofilename)
out = GPhys::IO.each_along_dims_write(in, ofile, -1){ |in_sub|
        [ in_sub.mean(0) ]  #  結果は毎回ファイル(ofile)に書き出し
      }
ofile.close
```

物理量の単位の扱い

- NmRu::Units 利用
- 積,商,etc.: 正しく更新
- 和,差 : 第一項に合わせる。
  - 例えば m と km の足し算なら、第2項をまず1000倍してmに換算。換算不可なら警告(エラーにはしない)
- 単位付定数クラス UNumeric の導入
  - GPhys, VArray, UNumeric は相互に演算可能(この順に強い)
  - 例えば風速のGPhysオブジェクト u (m/s) にコリオリパラメターを掛けるなら

```
f = UNumeric[1e-4,"s-1"]
coriolis_frc = f * u        # これで単位は m.s-2 に
```

標準添付応用ライブラリ

- 主成分分析(EOF) (in GAnalysis – require “numru/ganalysis”で)
- 多次元フーリエ解析
- 統計解析 (一部は in GAnalysis)
- 数値微分，補間，座標変換など
- EPフラックス計算
- 球座標ライブラリPlanets (in GAnalysis)
- 気象解析（温位，PV,…） (in GAnalysis)

- なお，「GFD電脳Ruby小物置き場」等から利用者による応用ライブラリがいろいろ提供されている

GGraphについて

- GPhys付属ライブラリ．GPhysオブジェクトを可視化する
- 可視化関数（メソッド）を収めるモジュール
  （図をオブジェクト化はしない．cf. GGraphベースでオブジェクト化するのがGfdnavi用VizShot）
- 多彩なオプション
- RubyDCLと協調で，お手軽クイックルックから凝った絵まで

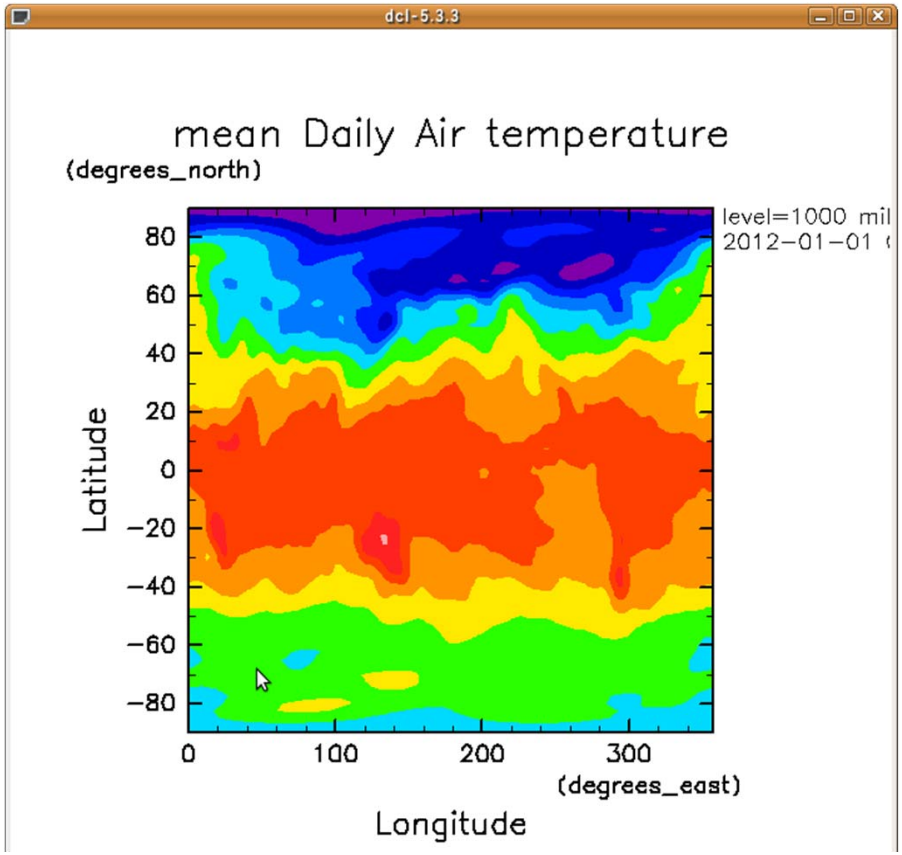
例 「GPhys, GGraphチュートリアル」(http://ruby.gfd-dennou.org/products/gphys/tutorial2/) より.

（irb用スタートアップファイルを利用して）

GPhys::IO “air.2012-01,nc”, “air” と同じ

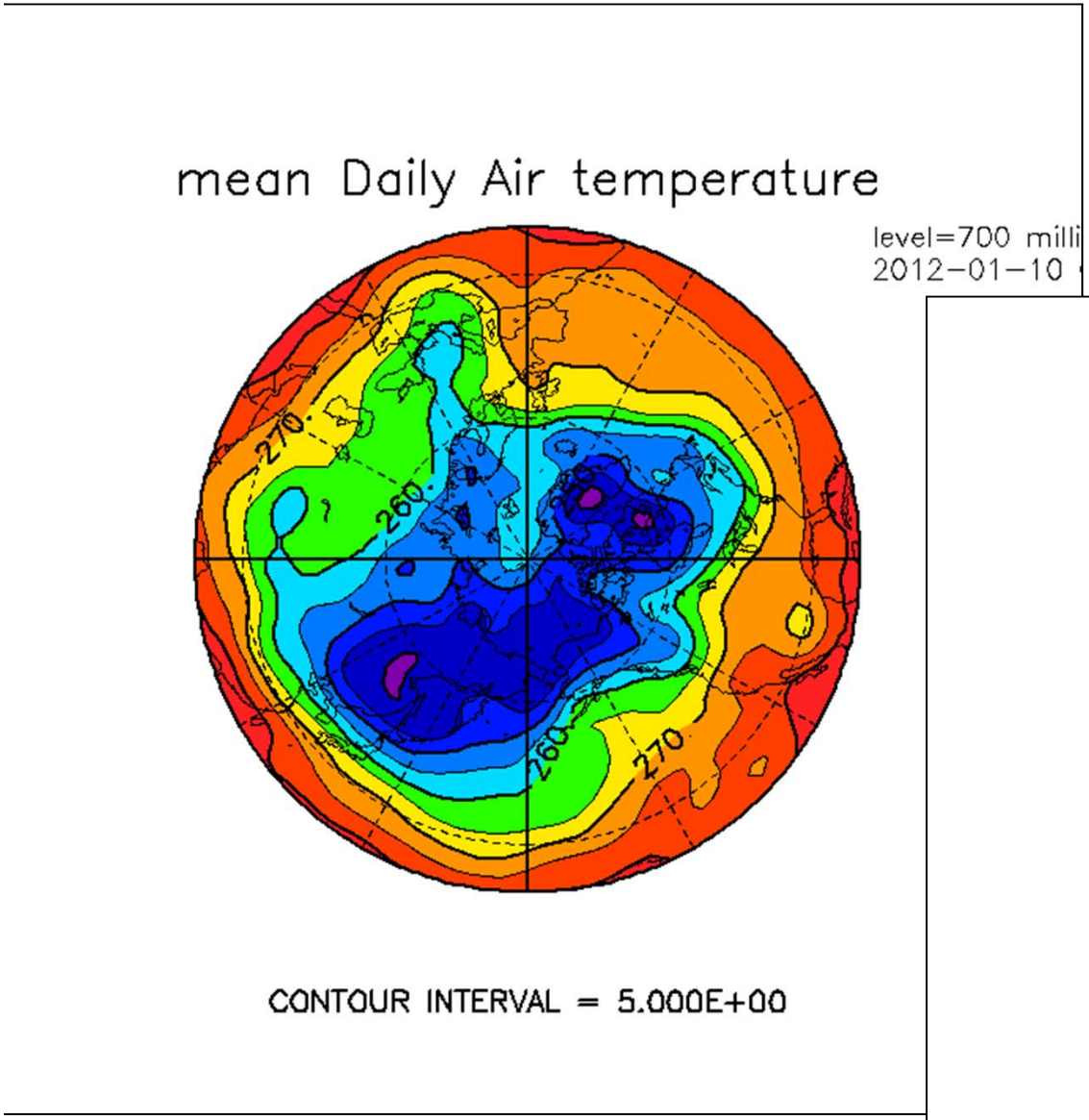
```
$ irb_ggraph
>>  gp = gpopen 'air.2012-01.nc/air'     # データ読込
>>  tone gp                               # 色塗り描画
```

GGraph::tone gp と同じ

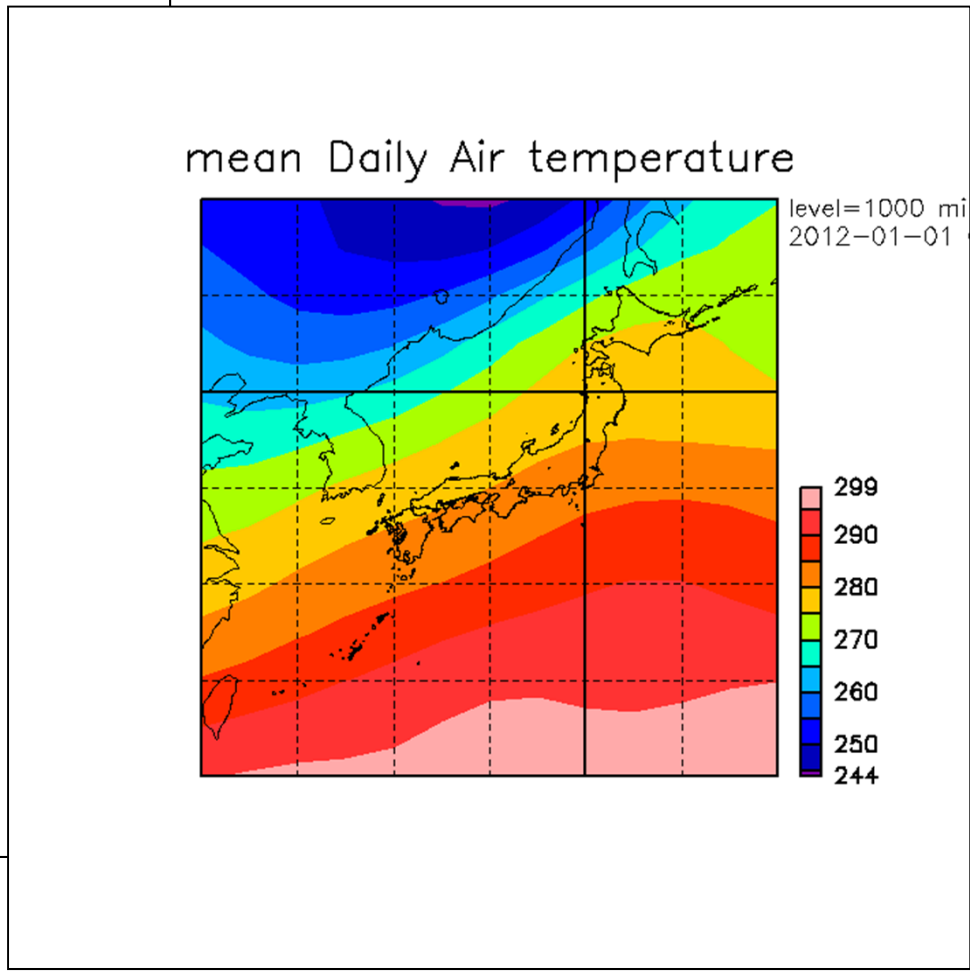


← 座標軸等データから読み込んで表示. 切り出し情報も欄外に

ギャラリー（GGraphで書いた図）

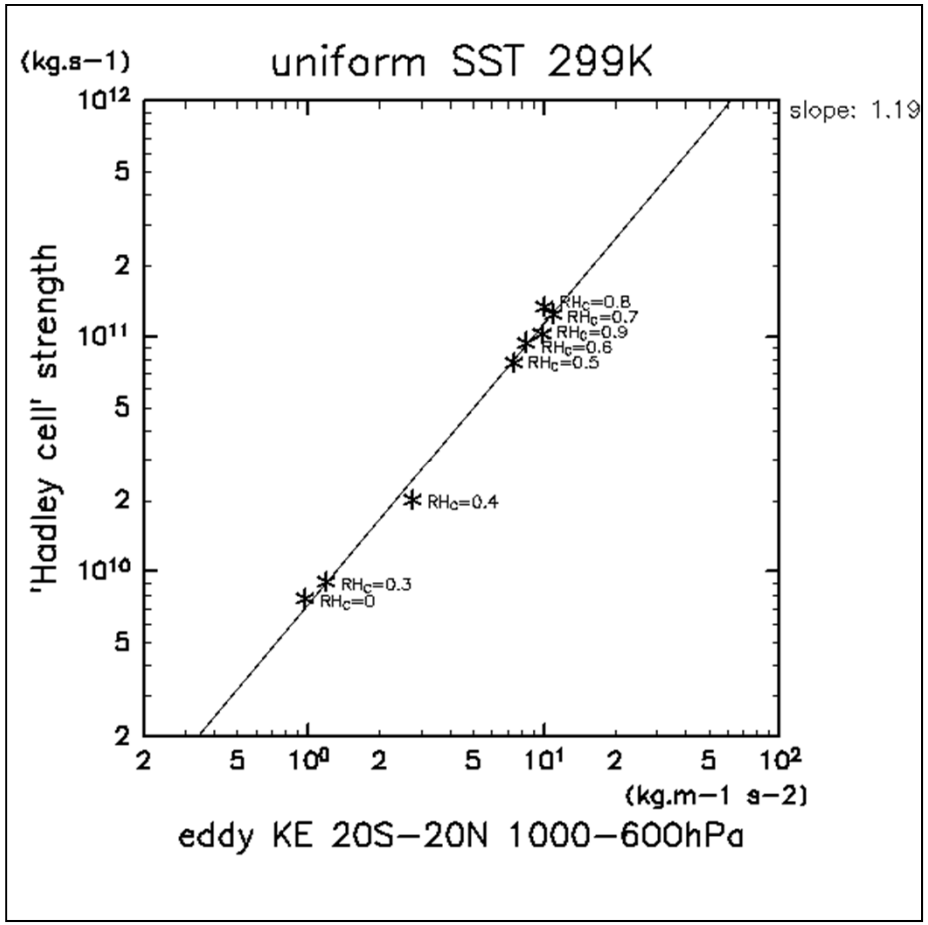


「GPhys, GGraphチュートリアル」より.
難易度: 易 (両図とも1行で書ける)

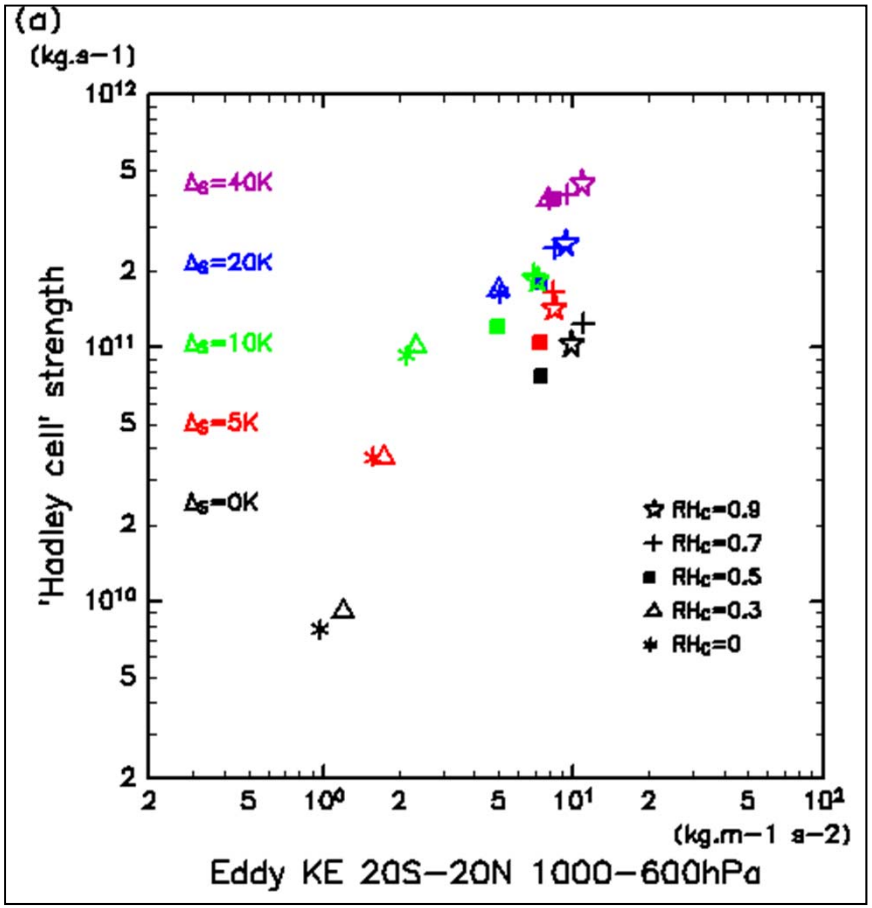


ギャラリー（GGraphで書いた図）

マーカー描画と回帰直線
難易度：（各マーク横の文字列以外は）易 （回帰直線はオプションでかける．欄外のスロープ表示も）

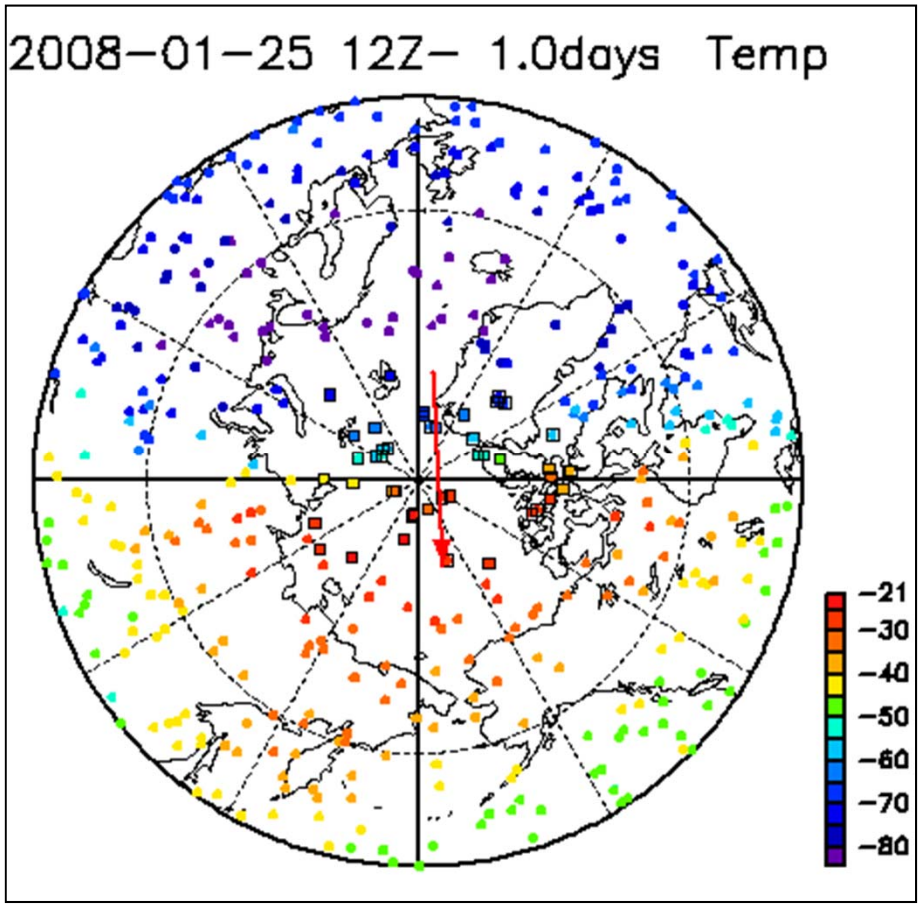


マーカー描画
難易度：凡例表示がやや難（ループで少しずつずらし表示．色等の連動や特殊文字や添え字の多用も初心者には難しいかも），**凡例表示以外は易**

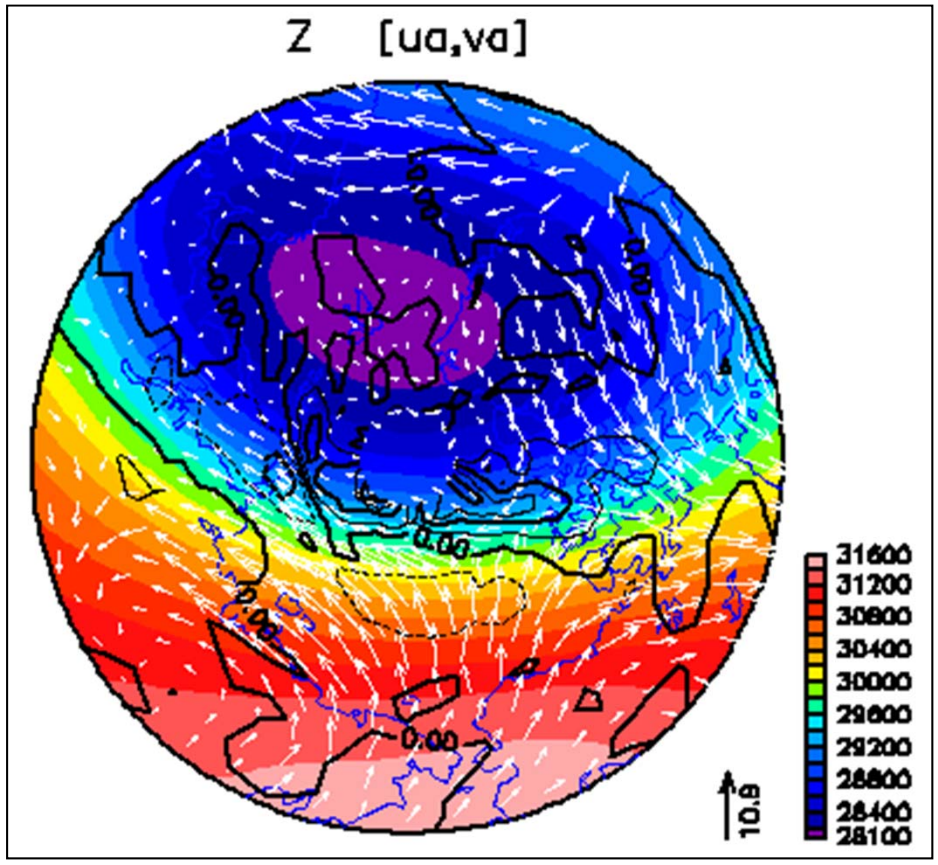


ギャラリー（GGraphで書いた図）

色つき散布図
難易度：それ自体は易
(特殊な矢印等除けば)



色塗り，コンター，ベクトル
難易度：中 (ベクトル矢印の色を白くするところは DCLコールが要る)



ギャラリー（GGraphで書いた図）

折れ線とコンター
難易度：難（余裕を保ちつつ図を詰めるため，軸のラベルやカラーバーを共通化しているところが．普通ここまでやらない．なお，カラーバーは実は独立したフレームに表示してある – DCL.sldiv(“y”,5,2) なのです）