

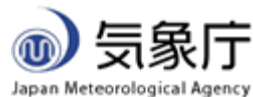
「Gfdnavi」の目指す方向と現状

西澤誠也¹, 堀之内武², 渡辺知恵美³,
伴林晃紀⁴, 諫本有加³, 大塚成徳⁵

¹神戸大, ²北海道大, ³お茶大, ⁴松岸寺, ⁵京都大

研究背景

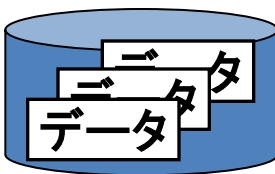
データ生産者・提供者



シミュレーションデータ

客観解析データ

観測データ



関数

解析

関数

手法

可視化

手法

Fortran / C

GrADS / IDL

Ruby

GPhys

知見

図

論文



GPhysライブラリ

• 簡単に解析・可視化を実現

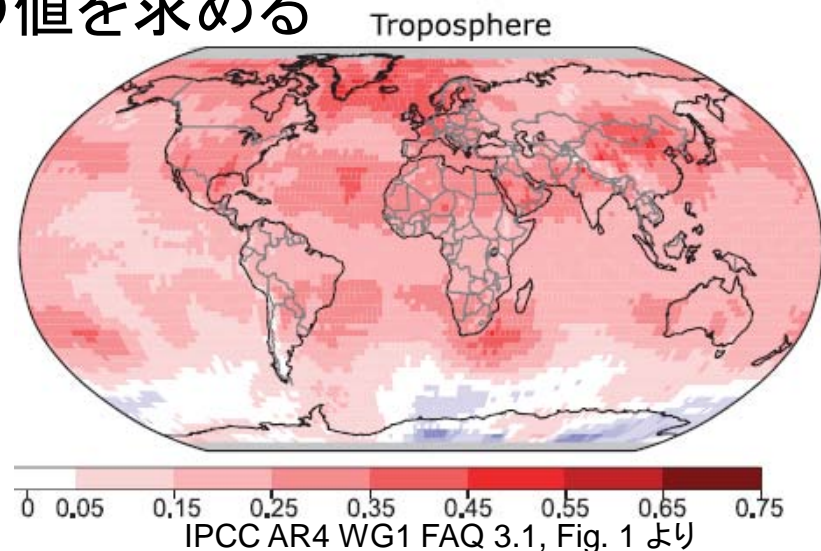


研究活動の1例

温暖化をシミュレーションするためのモデルを開発したが、モデルの妥当性を検証する

- 過去のシミュレーション実験を行い、現実大気の日データと比較する
 - 指標
 - 対流圏で平均した水平2次元の気温
 - 1979年から2005年間の線形トレンド (K/year or K/decade)

1. シミュレーション実験を行い, GrADS形式のデータを得る
 - 4次元データ; 空間3次元(経度,緯度,気圧) + 時間 1次元
2. 現実大気データとして, NetCDF形式の客観解析データ (NCEP/NCAR Reanalysis)をデータ提供サーバーからダウンロードする
 - 4次元データ
 - 空間3次元の大きさ: 144x73x17 ~ 0.7MB
 - 6時間毎データだとおおよそ 25GB, 月平均データだとおおよそ 210MB
3. それぞれのデータに対し, 高度方向の平均をとる
 - 3次元データ; 水平2次元+時間1次元
4. それぞれの格子点毎に,トレンドの値を求める
 - 水平2次元データ
5. 2次元トーン・等値線図を作成する
6. 2データの差のデータを作成する
7. 差のトーン・等値線図を作成する
8. 考察をメモにする



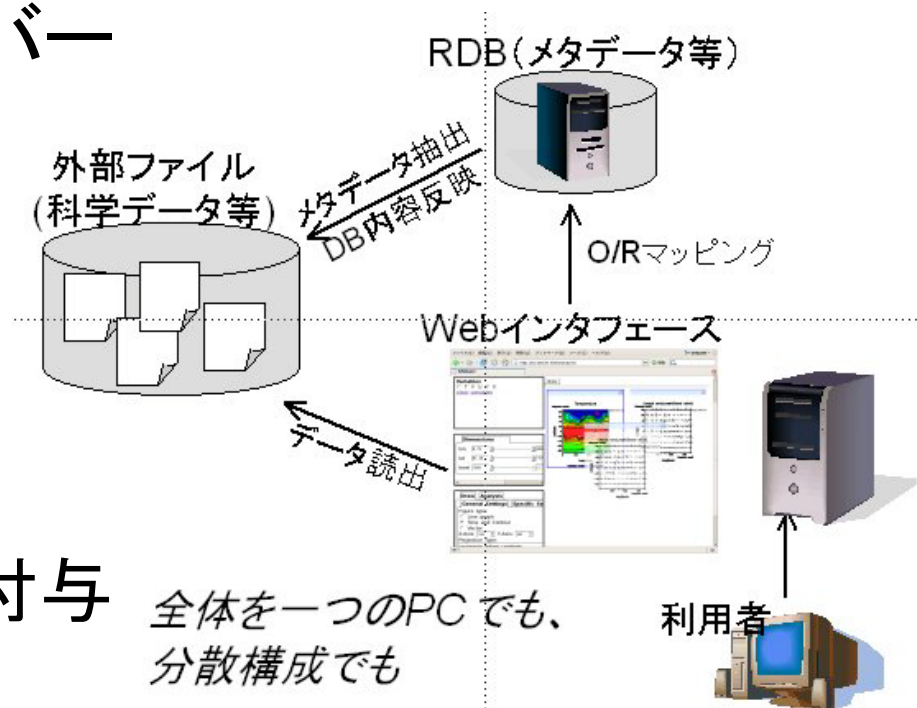
問題点

- データの存在・ありかを知っている必要がある
- データを手元にDownloadする必要がある
- データの配布方法, フォーマットがさまざま
- 解析結果データ・図の保存場所, 解析結果データ・図と元データ・プログラムの関係の管理が大変. 共有する場合はより困難.
- 解析結果データ・図と, それに対する知見 (in 脳内, メモ) とが乖離し, 後に関係づけるのが大変

「Gfdnavi」とは



- Web Application / Web Service
- 対象
 - データ公開サーバー
 - グループ内非公開サーバー
 - 個人デスクトップツール
- 機能
 - データベース (検索)
 - 解析・可視化
 - データへの知見情報の付与



設計思想

- 既存の研究用途に耐えうるもの
 - 基本的には, Gfdnaviだけで, データ生産後の, 一通りの研究生活を送ることができる
 - 我々自身が "使える" もの
- 現状の研究における問題点を解決する
 - データ・知見情報を一元的に管理
- 更に, 新たな研究手法の提供
 - どこからでも, どこのデータでも
 - 共同研究を円滑に
 - 学際研究の促進

データベース

- メタデータのデータベース化
 - 時空間情報 (必須ではない)
 - 1地点, 領域, (swath)
 - キーワード属性
 - 任意のキーワード, 値の組
- 検索のため
- 登録・更新
 - コマンドにより, 指定ディレクトリツリーをスキャン

解析・可視化

- サーバー上で解析・可視化を行う
 - GUI on web-browser
- 描画を再現するスクリプト&必要最低限データのダウンロードが可能
- 独自の解析メソッド, 描画メソッドを登録可能

知見情報

- 数値データや, (描画した)画像データに知見情報を付与できる
 - データから知見情報へ, 知見情報からデータへのリンクが自動で張られ, 相互アクセス可能
 - 知見情報内の図の再現, 描画パラメータ等を変更しての再描画が可能
 - 情報発信
- 他の知見情報に対し, コメントを付与できる
 - グループ内ディスカッション
 - 教官-学生間の指導



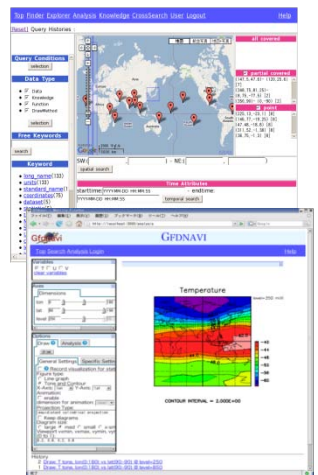
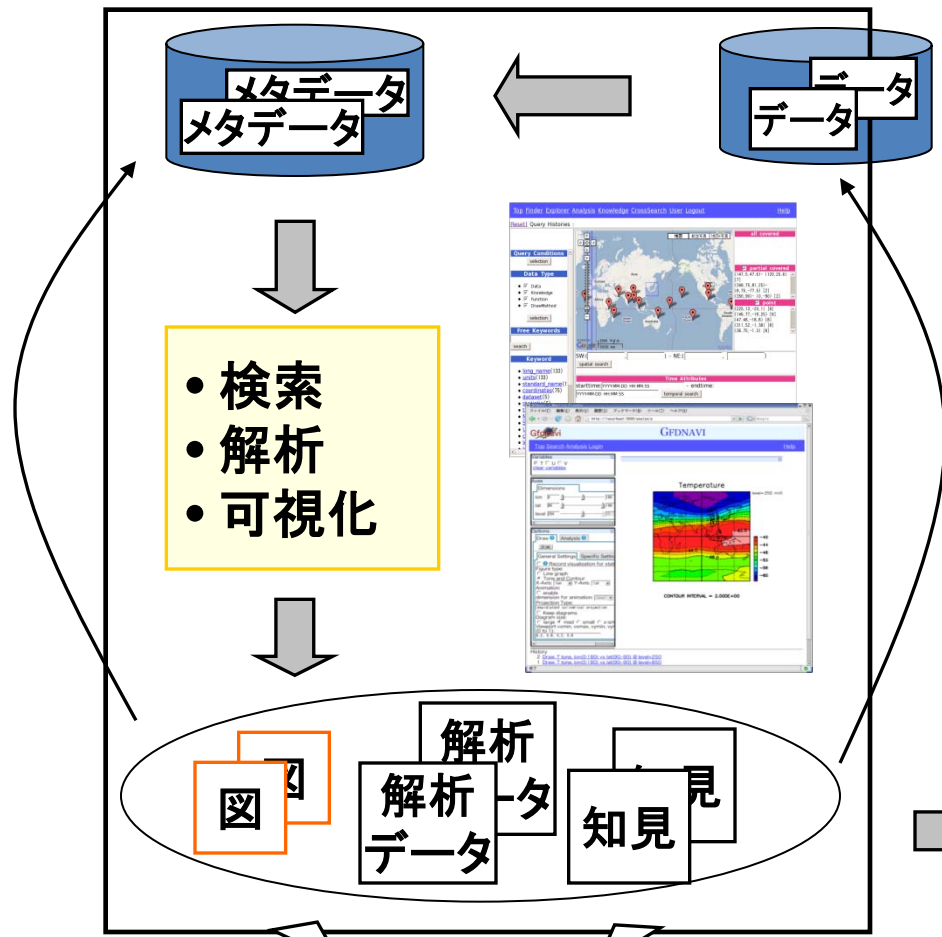
データ生産者・提供者



シミュレーションデータ

客観解析データ

観測データ



- シミュレーション結果を Gfdnavi に登録する
- 客観解析データをダウンロードし, Gfdnavi に登録する
- 高度平均をとる
- (トレンドを求める関数を登録する)
- トレンドを求める
- 2次元トーン・等値線図を作成する
- 2データのトレンドの差を求める
- 差のトーン・等値線図を作成する
- 考察の結果を知見情報として登録する

Gfdnavi
上で行う

- どこにいても, ブラウザさえあれば同じ環境で解析・可視化できるようになった
- 元データ, 解析データ, 図, 知見を容易に管理できるようになった

これで我々は幸せになったか？

Not yet

便利にはなったが, 問題がある

- Gfdnavi は孤立している
 - Gfdnavi できないこと, やりにくいことがあると, 結局データをダウンロードして, ローカルで処理しなければならない
 - 繰り返し処理
 - 複雑な処理
 - 他のサーバーにあるデータとの比較
 - ローカルでの作業を Gfdnavi に登録する場合, ファイルのアップロードが必要
- バグがなくなならない
 - 昔は動いていたが, 動かなくなっていることがある

Web Service の導入

- SOAP で実装
 - 解析・可視化部分の全てを 実行可能
 - 繰り返し処理等は容易になった
- Web Application との共存
 - 大部分はコードを共有

しかし,

- SOAPは使いにくい (要 SOAPライブラリ)
 - 人に渡しにくい (要 実行環境)
- 実装が大変 (特に Web Application との協調)
 - 検索・知見部分はない
 - 登録機能はない (一方通行)
- Web Application と, 機能的には重複している
が分ける必要があるコードがある
 - 開発・維持を困難に
- Web Application がベースであり, ステートフルなため, テストが困難
 - エンバグの温床

そこで,

- Web Service で全ての機能を実装する
- Web Service をベースにして, Web Application は Web Service を利用
 - コードの重複をなくす
- ステートレスにする
 - テストが容易

SOAP から REST に変更

- よりシンプル (開発が楽)

REST

- 統一インターフェース
 - HTTPプロトコル/メソッドを利用
 - ステートレス
- リソース指向
 - すべては“リソース”
 - リソースは少なくともURIを1つ持っている
 - リソース間の関係は, リンクで表現
 - リソースの表現形は, ハイパーメディア

- REST にするにあたり, 考えなければならなかったこと
 - データベースの行データはリソースとの対応付けの(準)標準があるが, 解析・可視化等, 動的なものをどのようにリソースで表現するか
 - 「解析する」 から 「解析されたデータ」 へ
 - SOAPでは自由にメソッドを定義できるが, RESTではHTTPメソッドのみ
 - 「解析する」 から 「解析データの取得」 へ
 - 動的リソースのURIはどうか
 - とりあえず試行錯誤でsyntaxを決めた

- URI syntax

`http[s]://{host}:{port}/{document_root}/data/{resource_path}.{suffix}[?{options}]`

- *suffix*: 表現形の指定 (html, xml, yml, nc, gphys, png)

- *options*: 表現形に対するオプション

- *resource_path*

- データ・ディレクトリリソース (データベースの行データ)

- e.g. /T.jan.nc/T

- 複数データリソース

- e.g. /[UV.jan.nc/U,/UV.jan.nc/V]

- 動的生成リソース

- `/{original_resource_path}/{resource_type}({params})[¥[{{index}}¥]]`

- e.g. /T.jan.nc/T/analysis(mean;longitude)

- /T.jan.nc/T/analysis(mean;longitude)/plot(tone)

- /find(name=Temperature)[0]

- RESTful Gfdnavi Web Service を利用した解析用スクリプト
 - 文字列処理・通信が中心
 - URI・リンクの処理
 - HTTP通信
 - データ解析・可視化と直感的に一致しにくい

ラッパーライブラリを用意 (GfdnaviData)

クライアント側



- サーバへの通信は GfdnaviDataライブラリが担当
- GPhysライブラリと似た関数で処理

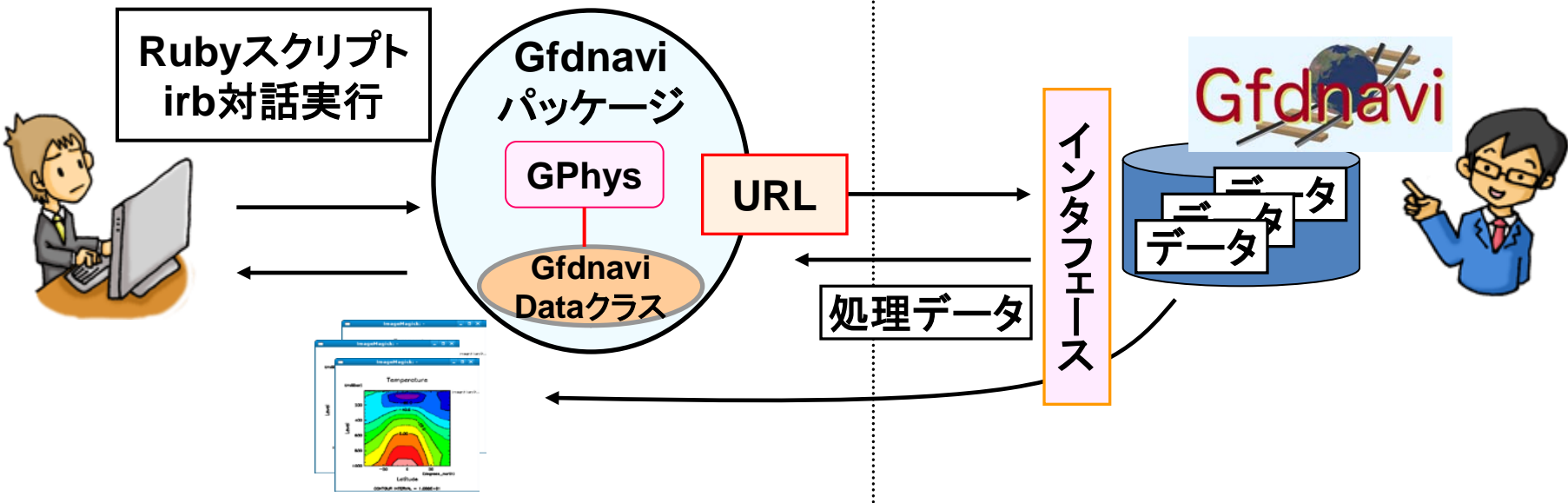
サーバ側



- プログラムの通信に対応
⇒ 結果の受け渡し

REST

- URLでリソースにアクセス

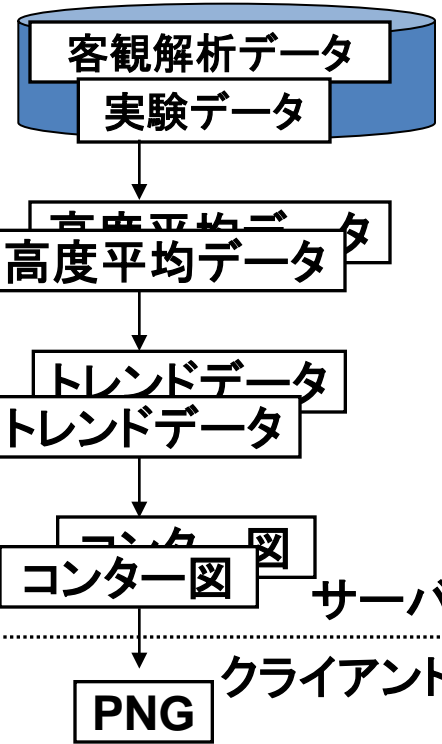


- シミュレーション結果を Gfdnavi に登録する
- 客観解析データをダウンロードし, Gfdnavi に登録する
- 高度平均をとる (サーバーサイド)
- トレンドを求める
- 2次元トーン・等値線図を作成する
- 2データのトレンドの差を求める
- 差のトーン・等値線図を作成する
- 考察の結果を知見情報として登録する

Gfdnavi
Data
を利用

スクリプト例

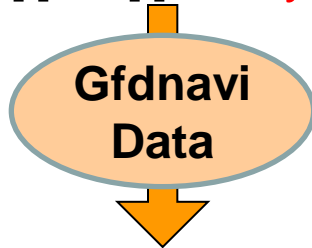
```
1: require "numru/gfdnavi_data "  
2: include NumRu  
3: t = Array.new; t_zm = Array.new; t_trend = Array.new  
  
4: t[0] = GfdnaviData.parse("http://host:port/data/simulation/T.ctrl/T")  
5: t[1] = GfdnaviData.parse("http://host:port/data/reanalysis/T.nc/T")  
  
6: 2.times do |i|  
7:   t_zm[i] = t[i].analysis("mean", "pressure")  
8: end  
  
9: 2.times do |i|  
10:  t_trend[i] = t_zm[i].analysis("regress", "time")  
11: end  
  
12: 2.times do |i|  
13:  display_png( t_trend[i].plot("tone").to_png )  
14: end  
  
15: t_diff = t_trend[0] - t_trend[1]  
16: display_png( t_diff.plot("tone").to_gng )
```



解析

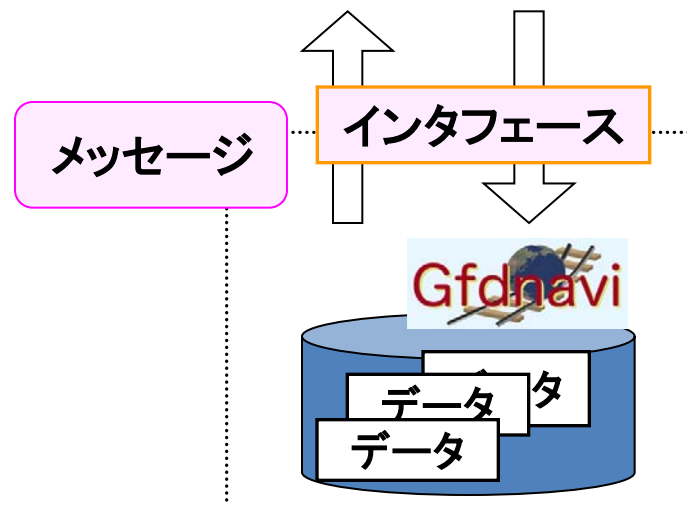
GfdnaviDataオブジェクトはHTTPリソースに対応
 $t[0] \Leftrightarrow \text{http://host:port/data/simulation/T.ctrl/T}$

7: $t_zm[i] = t[i].\text{analysis}(\text{"mean"}, \text{"pressure"})$



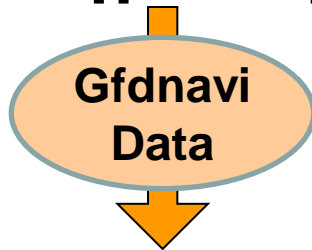
#analysis, #plot, #find は、それぞれ、
解析, 描画, 検索結果のリソースに
対応するオブジェクトを返す

[http://host:port/data/simulation/T.ctrl/T/analysis\(mean;pressure\)](http://host:port/data/simulation/T.ctrl/T/analysis(mean;pressure))

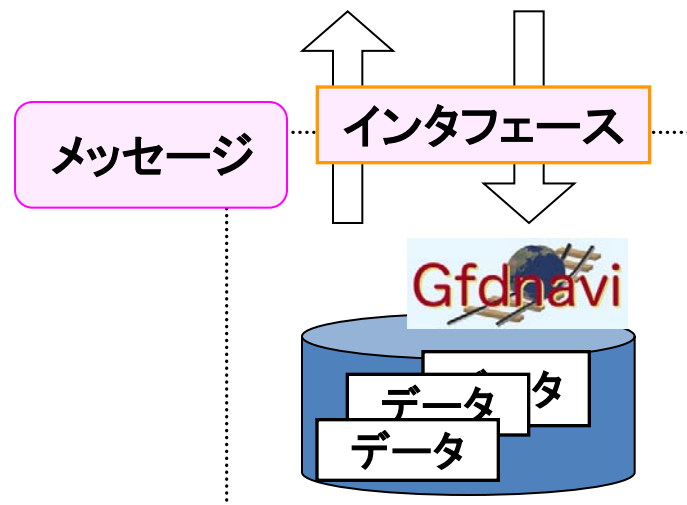


解析

```
10: t_trend[i] = t_zm[i].analysis("regress", "time")
```

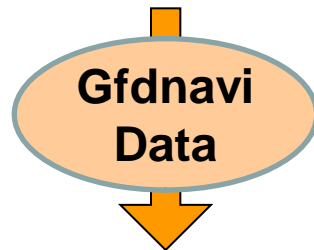


[**http://host:port/data/simulation/T.ctrl/T/analysis\(mean;pressure\)/analysis\(regress;time\)**](http://host:port/data/simulation/T.ctrl/T/analysis(mean;pressure)/analysis(regress;time))



可視化

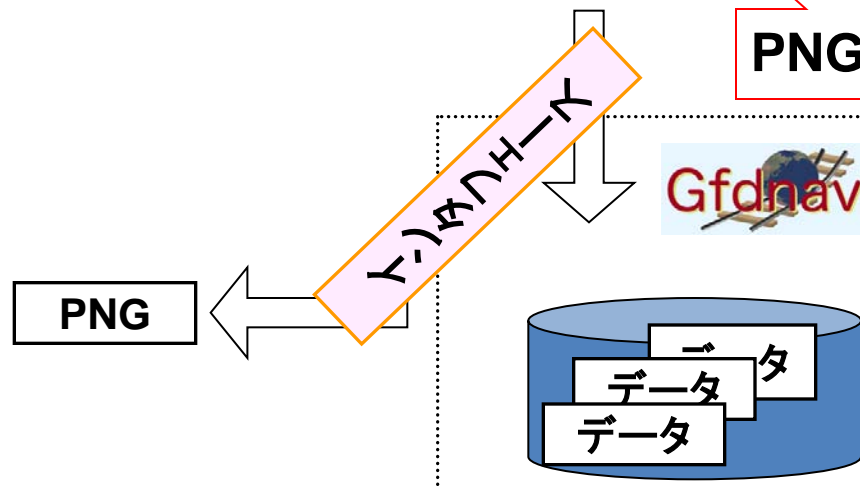
13: `display_png(t_trend[i].plot("tone").to_png)` **実際のデータが必要**



実は今までは解析は実行されていなかった
ここで初めて、全ての解析・描画が実行
#to_XXX メソッドはリソースの表現を取得

[http://host.port/data/simulation/T.ctrl/T/analysis\(mean;pressure\)/analysis\(regress;time\)/plot\(tone\).png](http://host.port/data/simulation/T.ctrl/T/analysis(mean;pressure)/analysis(regress;time)/plot(tone).png)

PNG形式で取得



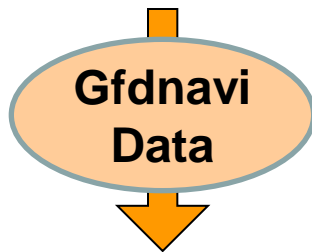
解析

```
15: t_diff = t_trend[0] - t_trend[1]
```

実は

```
t_diff = GfdnaviArray[ t_trend[0], t_trend[1] ].analysis("subtraction")
```

を使いやすいようにしたもの (他の四則演算も同様)



GfdnaviArrayオブジェクトは
GfdnaviData オブジェクトの順序付き配列

```
http://host:port/data/[/simulation/T.ctrl/T/analysis(mean;p  
ressure)/analysis(regress;time),/reanalysis/T.nc/T/analysis  
s(mean;pressure)/analysis(regress;time)]/analysis(subtr  
action)
```

- さらに

- GfdnaviData#to_gphys

- データのバイナリが GPhysオブジェクトとして得られる
→ ローカルで自由に解析

- GfdnaviData#save_as(*path*)

- サーバーのデータベースに保存する
 - データ, 図, 知見情報

サーバーとローカルの行き来が可能

サーバー間連携

- Gfdnaviサーバー間の連携
 - どのサーバーにどのデータがあるのか意識する必要がなくなる
 - クロスサイト検索
 - 知らなかったデータがヒットする可能性
 - クロスサイト解析・可視化
 - データ比較
- Gfdnavi以外の公開データ, Webサービスの利用
 - OPeNDAP
 - GDS (GrADS Data Server)
 - Live Access Server

- RESTful Web Service, サーバー間連携は実装中
- RESTful Web Service に関しては, 試行錯誤の部分が多い
 - 標準的なものができれば (or すでにあれば), それに習いたい
- 完成すれば, データ生産以降のほとんどの研究は, これで行う事ができると期待される
 - 新たな問題が出てくる可能性も否定はできない

まとめ

- 数値・画像データや知見の仮想的な一元管理
 - データの所在を意識せずに利用可能
 - 異なるサーバーにあるデータの比較が容易に
 - 未知のデータとの遭遇
 - データ・解析手法・知見のリンク付け
 - 図を再現可能
 - 知見から解析・可視化を再現可能
 - データから知見を検索可能
 - 共同研究者との議論が楽に (知見・コメント, 再現性)
 - データと知見を一緒に公開することにより, 専門外の人の利用も促進 (GUI も効果的)

- Web Application / Web Service の融合
 - 目的に合わせてGUI とスクリプトの使い分け
 - クイックルック
 - 繰り返し処理
 - ローカルプログラムとのシームレスな連携
 - Gfdnavi では難しいような凝った解析・可視化が可能
 - 結果を Gfdnavi に保存
 - RESTful にすることで, 開発が楽に
 - シンプルなコード
 - ステートレスにすることで, テストが楽に
 - エンバグを防ぐ

Thank you