オープンソースのGISソフトウェア

QGISセミナー

～QGISの使い方・中級編～

Ver. 2.4版

OSGeo財団日本支部

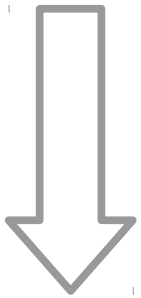
OSGeo.JP

セミナーの目標

- **GISを用いた空間分析機能の基本**を習得する
  - ベクターの分析機能
  - ラスターの分析機能
    - プラグイン利用方法
  - 他のアプリケーションとの連携
  - 応用的な分析
- 操作が不明なときは**遠慮せずにご質問下さい**
  - それが一番スムーズに進みます
- 最後に質問の時間もあります
  - 込み入った質問等は、そちらでお願いします

はじめに

GISによる空間分析



QGISの分析機能の概要

GISの分析を
一言でまとめると？

GIS Analysis

is

Lego®!

http://www.flickr.com/photos/bigdaddynelson/2667256880/in/set-72157606154373202/

http://www.techeblog.com/index.php/tech-gadget/lego-aircraft-carrier

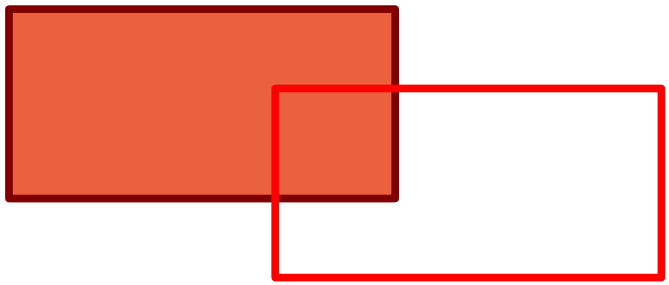
http://lnl.sourceforge.jp/pukiwiki/index.php?LEGO%2FEx-S-GUNDAM

GISの分析は"**組みあわせ**"

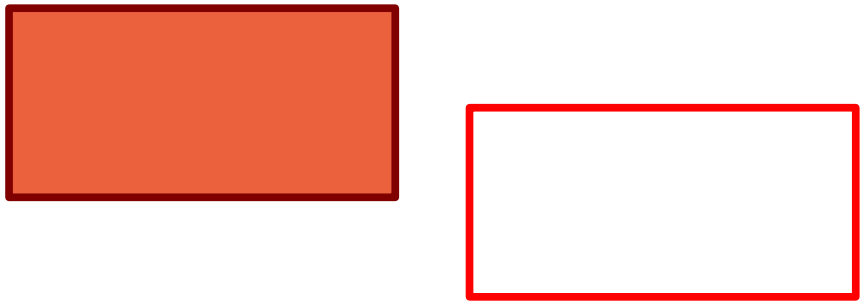
- 個別の機能は，"**汎用的**"だが"**単機能**"
  - 複数の機能を組みあわせることが必要
- 分析の"**流れ**"を作ることが重要
  - 機能を理解することが必要
    - 既存文献等も参考に
- 基本的な分析機能
  - 位置と属性が鍵になる

GISの分析機能の基本

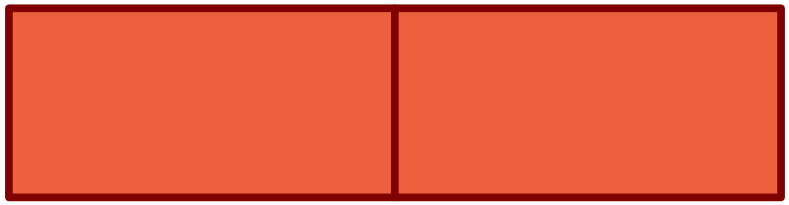
- **”位置”**と**”属性”**に基づく演算
  - 下記以外にも色々な場合がある



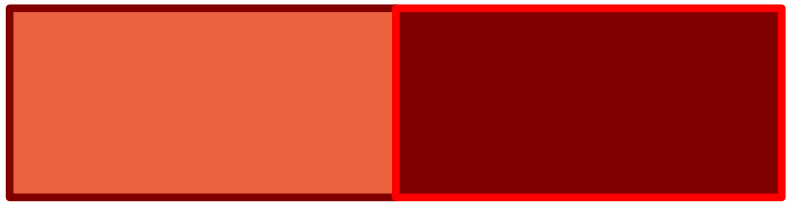
重なってる



重なってない



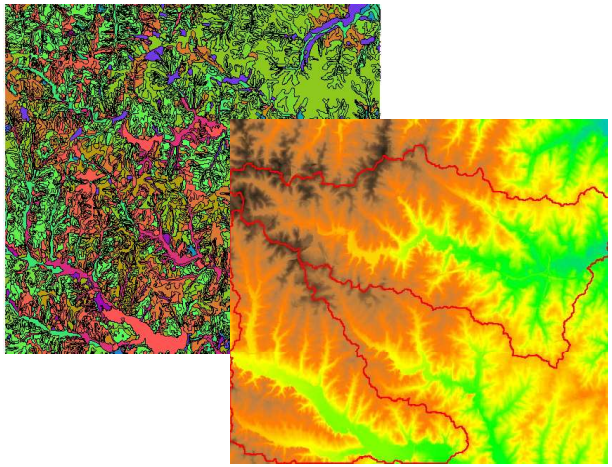
同じ属性

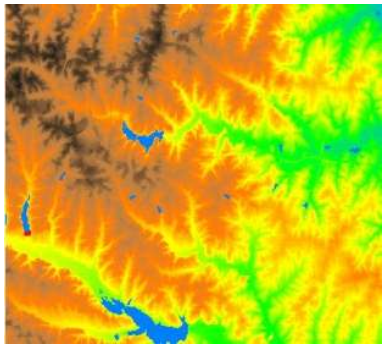


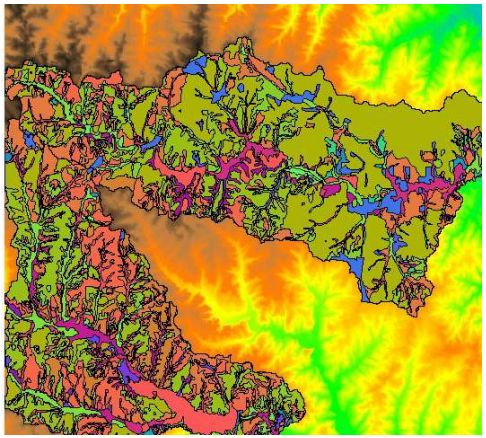
違う属性

位置に基づく分析例

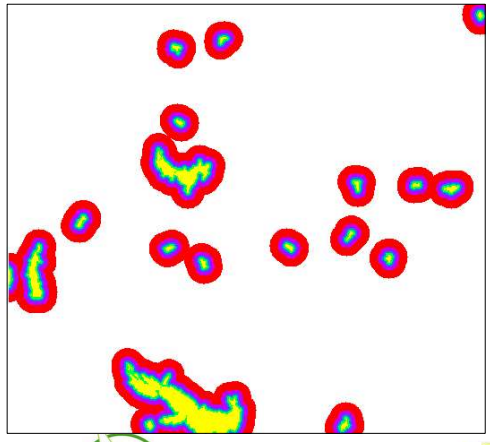
- 重ねあわせ
  - 交差，クリップ
  - 選択
- 距離
  - バッファー







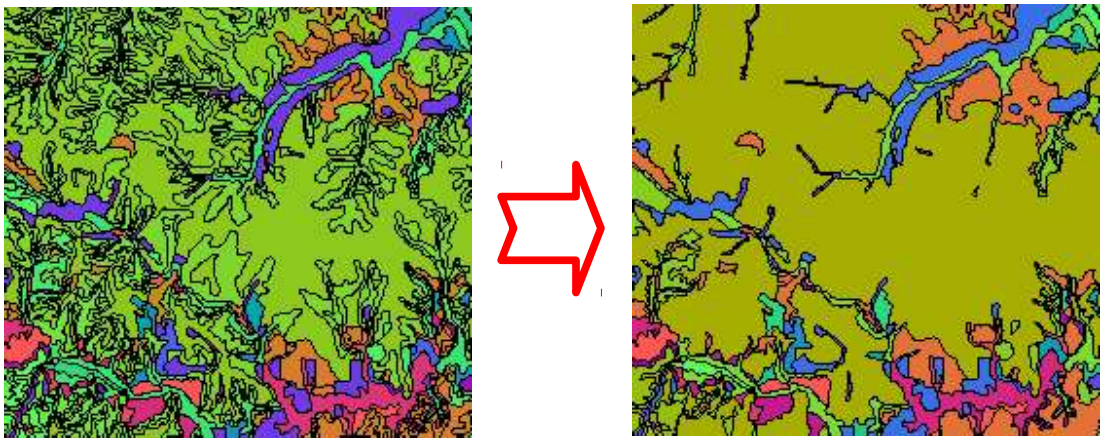
インターセクト



バッファー

属性に基づく演算

- 同じ属性で境界を消去
  - ディゾルブ
- 同じ属性を結合
  - ジョイン



ディゾルブ

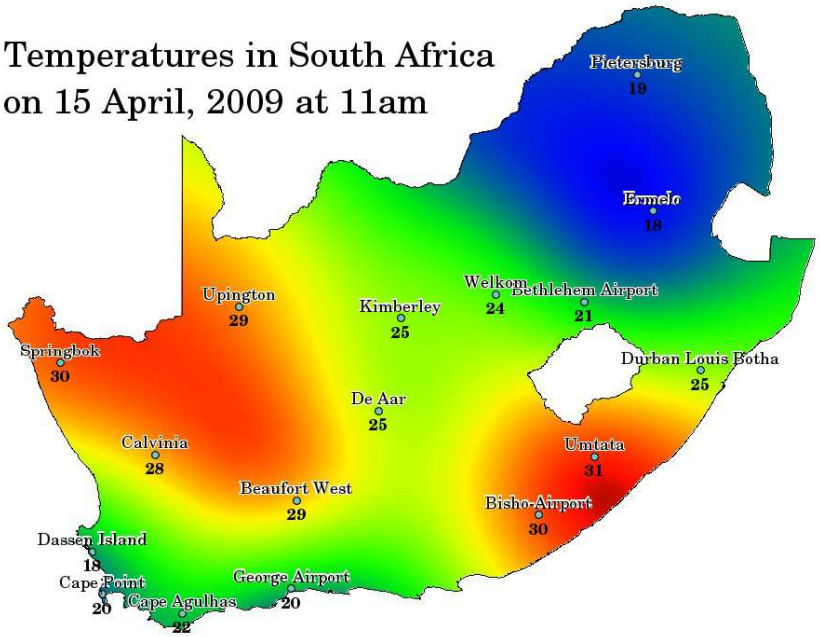
テーブル結合

lat	lon	code
34.99	135.93	1
34.96	135.91	1
34.97	135.97	2
34.98	135.88	1
34.71	135.98	2
34.94	135.91	1
34.99	135.94	1

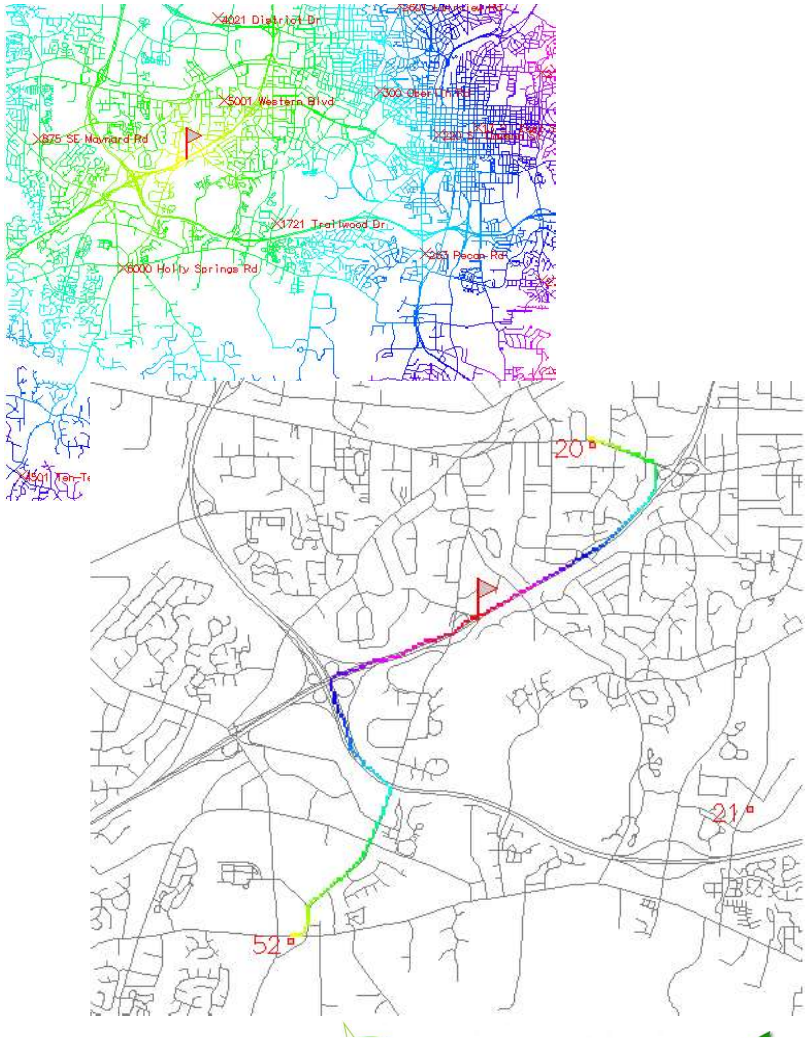
code	name	dist
1	spA	500
2	spB	1000

位置と属性を用いた分析

- 経路検索
  - 隣接するセルでの距離計算
- 空間補間
  - 位置から値を推定



温度の空間補間



最高速度を考慮した最短距離検索

今回使用するQGISのバージョン

- OSGeo4W版のQGIS 2.4.0 Chugiak 32bit版
  - 64bit版は幾つかのプラグインで不安定なため
    - 大容量データを扱う場合は64bit版をおすすめ
      - OSも64bitである必要あり

今回使用する資料について

- 本資料の利用は、Creative Commonsの「表示 - 2.1 日本」でお願いします
  - 出典を明示していただければ、OKです
  - http://creativecommons.org/licenses/by-nc/2.1/jp/

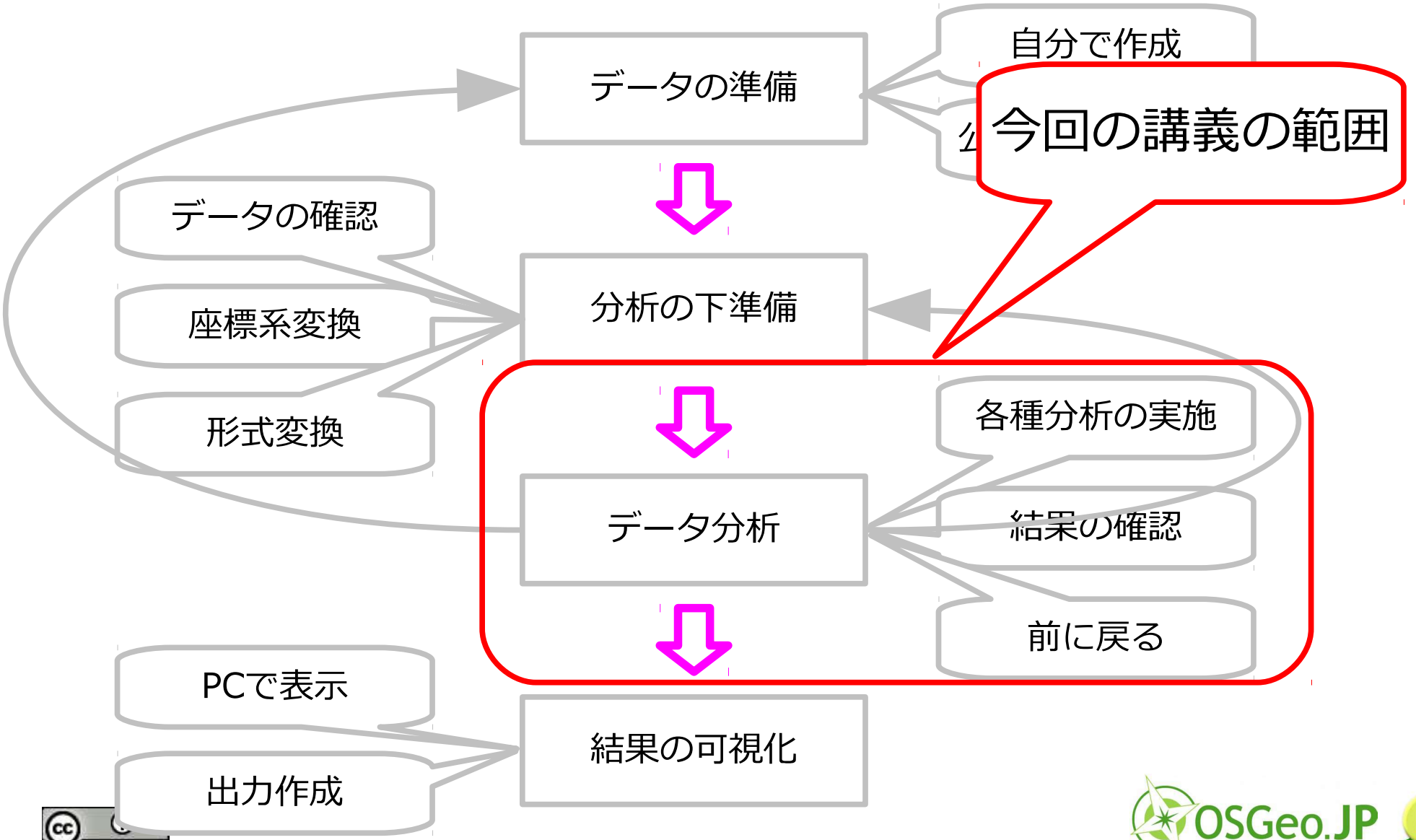
セミナーの流れ

- OSGeo4W版QGISのインストール
- QGISの分析機能の紹介
  - ベクター分析（ftools, MMQGIS）
  - ラスター分析（GDAL tools）
  - 高度な分析（sextante）
- ベクタデータの分析
  - fTools
  - ベクタデータの活用と表示
    - テーブル結合
    - 座標変換
    - データ型の変換

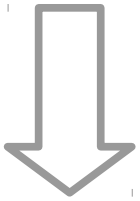
セミナーの流れ

- ラスタデータの利用
  - ラスタデータの表示
    - 単バンド，３バンド
  - ラスタデータの分析
    - 投影変換，斜面傾斜
- Pythonコンソールの利用
- PythonからRを呼び出す
- 実践的な分析

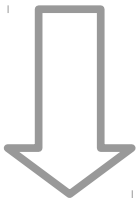
GISのワークフロー



はじめに



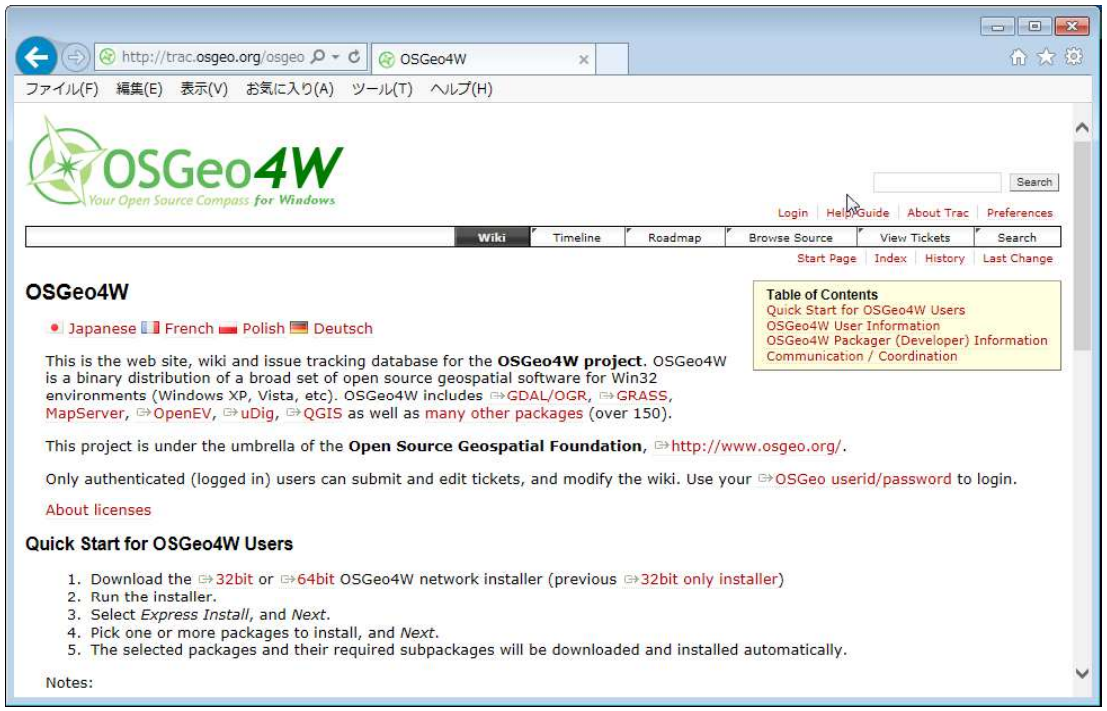
OSGeo4W版QGISのインストール



ベクターデータの分析機能

OSGeo4Wとは

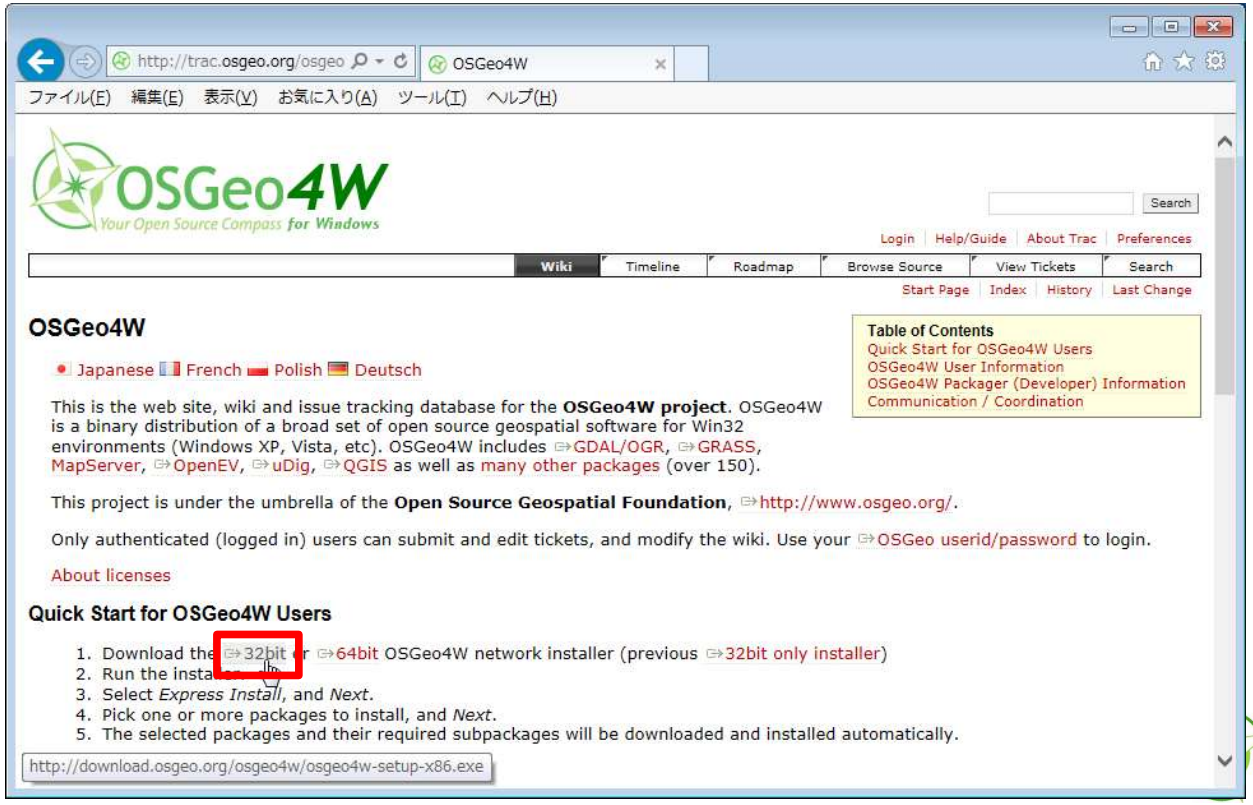
- Windows用のFOSS4Gの統合パッケージ
  - QGISだけでなく、GRASS、MapServer等の各種ソフトをインストールできる
    - プラグインを利用するために必要なライブラリもインストール可能



OSGeo.JP

OSGeo4W版QGISのインストール

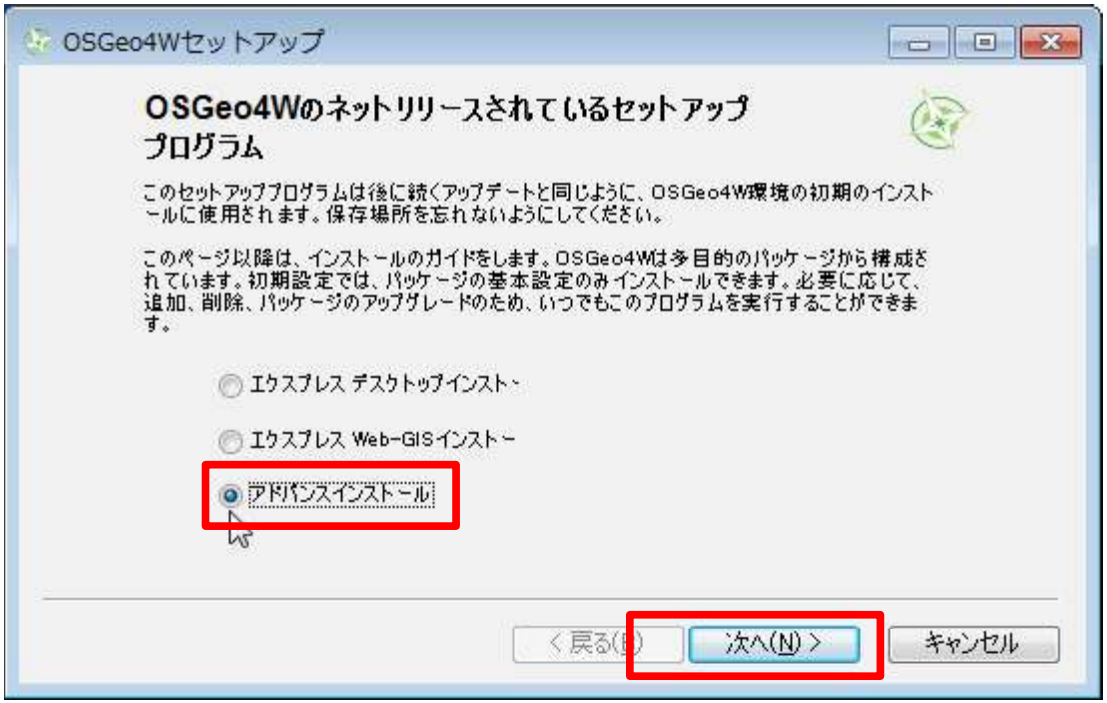
- 以下のページからインストーラーをダウンロード
  - http://trac.osgeo.org/osgeo4w/wiki/WikiStart
- 今回は32bit版を使用（c:¥tmpにコピー済み）
  - 64bit版だとプラグインが不調な場合もあり



OSGeo.JP

OSGeo4W版QGISのインストール

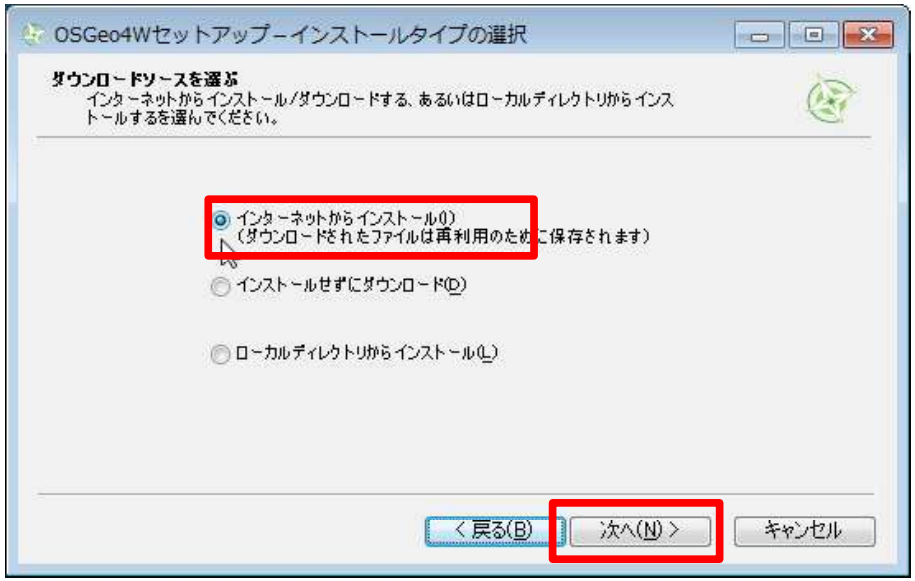
- DLした「osgeo4w-setup-x86.exe」を実行
- 「アドバンスインストール」を選択して次

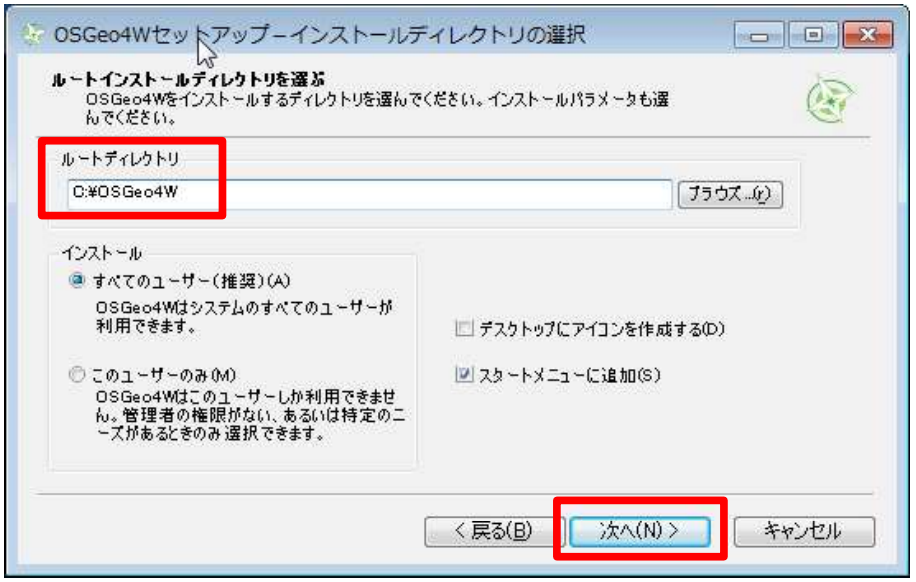


OSGeo.JP

OSGeo4W版QGISのインストール

- 「インターネットからインストール」を選択
- 「ルートディレクトリ」として「C:¥OSGeo4W」を選択（デフォルトのまま）

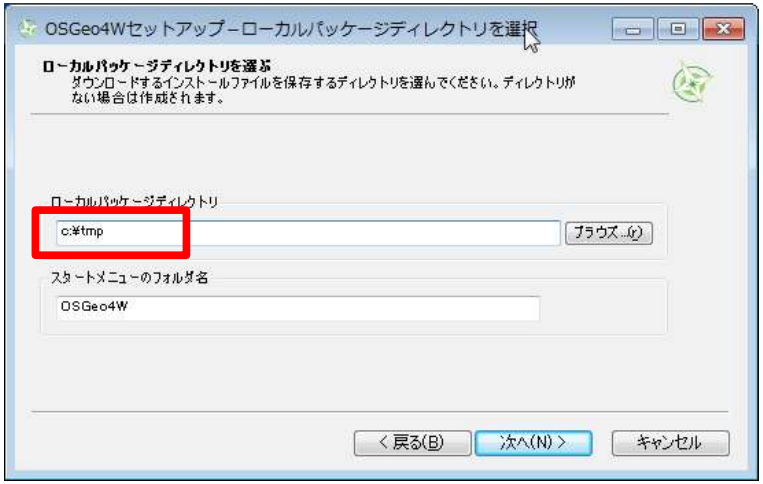


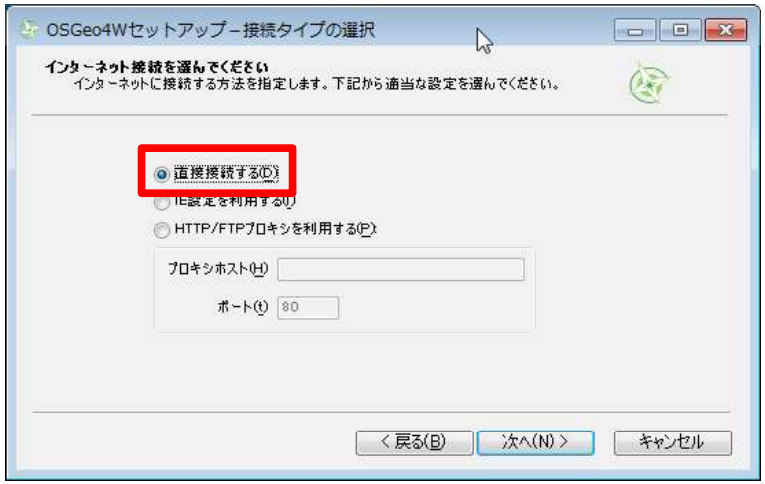


OSGeo.JP

OSGeo4W版QGISのインストール

- 「ローカルパッケージディレクトリ」は「C:¥tmp」に設定
  - インストールするソフトを保存しておくところ
- インターネット接続→それぞの環境に依存
  - 通常は直接接続

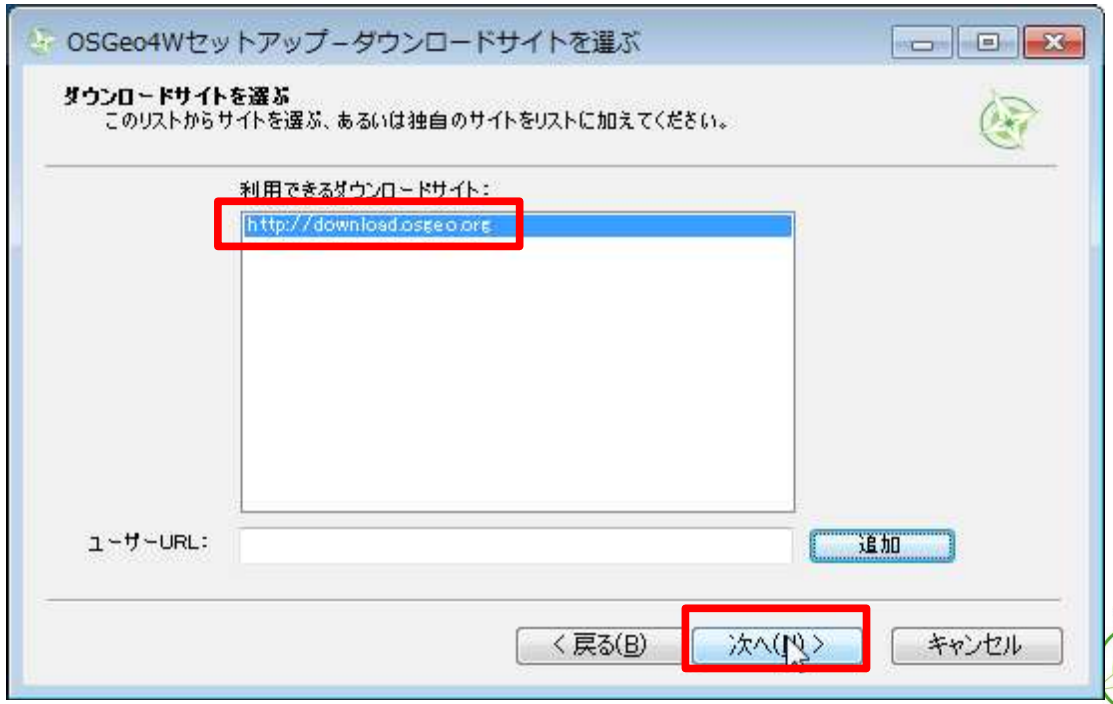




OSGeo.JP

OSGeo4W版QGISのインストール

- ダウンロードサイトの選択
  - 「http://download.osgeo.org」を選択して「次へ」をクリック



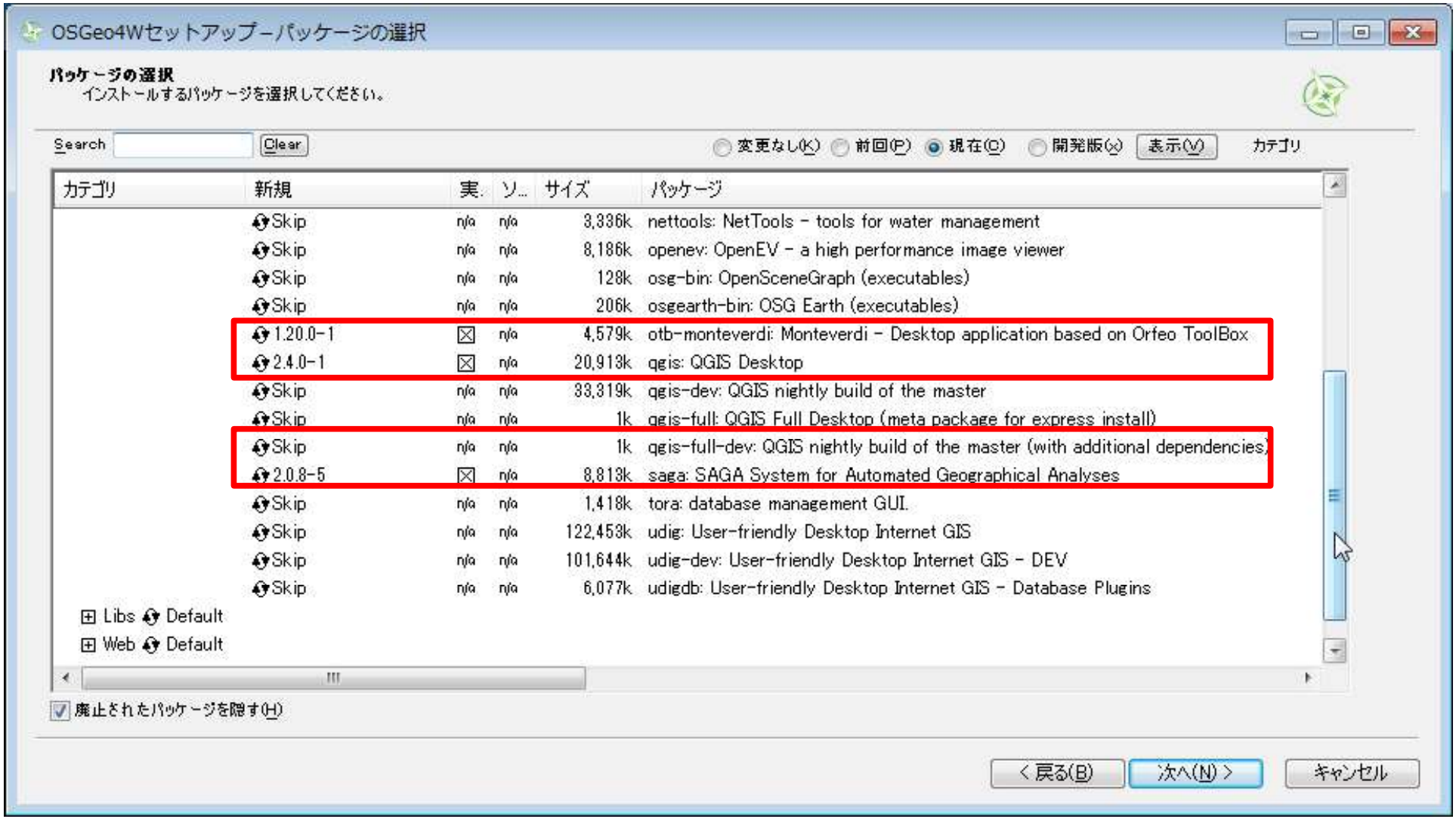
OSGeo.JP

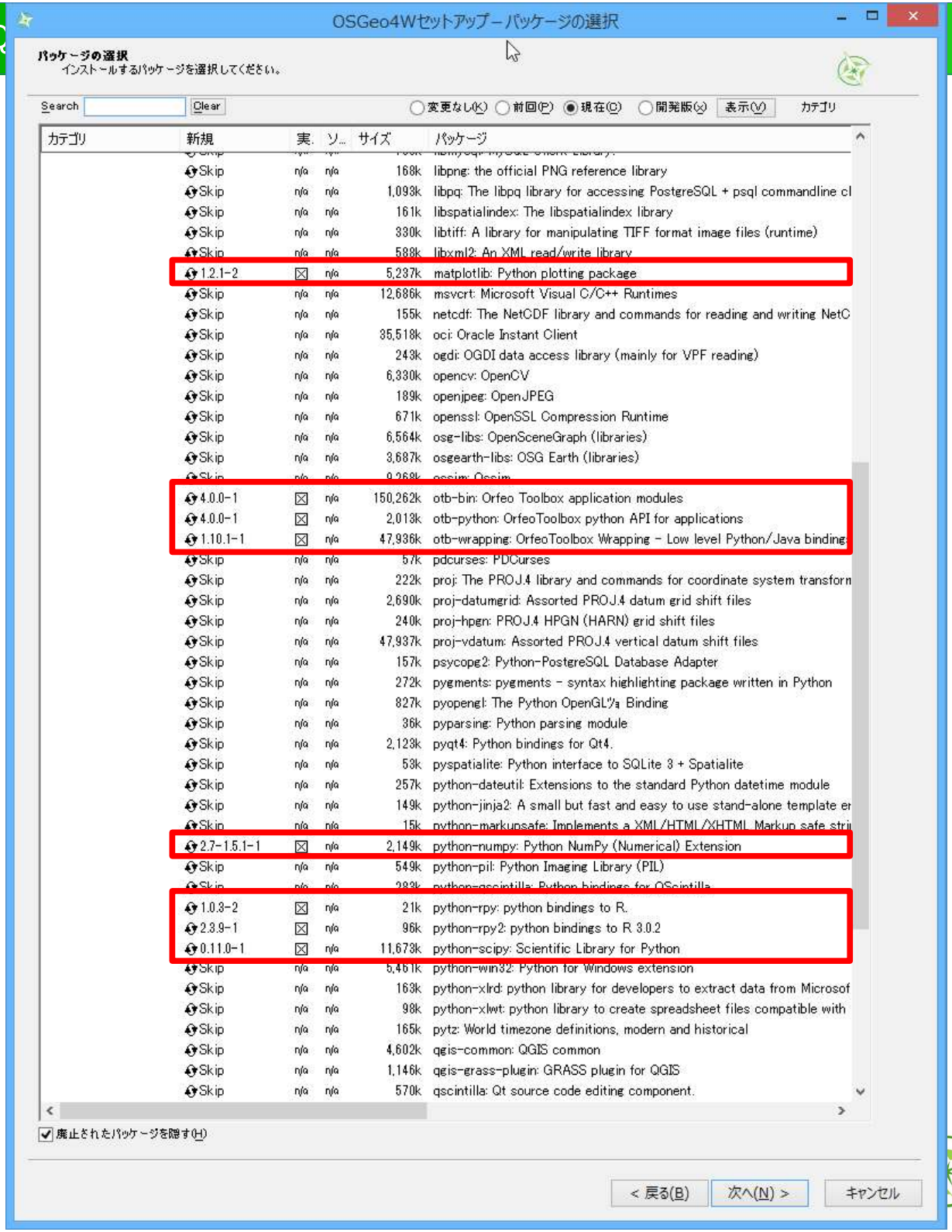
OSGeo4W版QGISのインストール

- パッケージの選択で「Desktop」左の「＋」をクリックして、展開
- 「qgis: QGIS Desktop」の左の「Skip」をクリック
  - 「2.4.0-1」と変わる
    - これで自動的にダウンロードされて、インストールされる

OSGeo4W版QGISのインストール

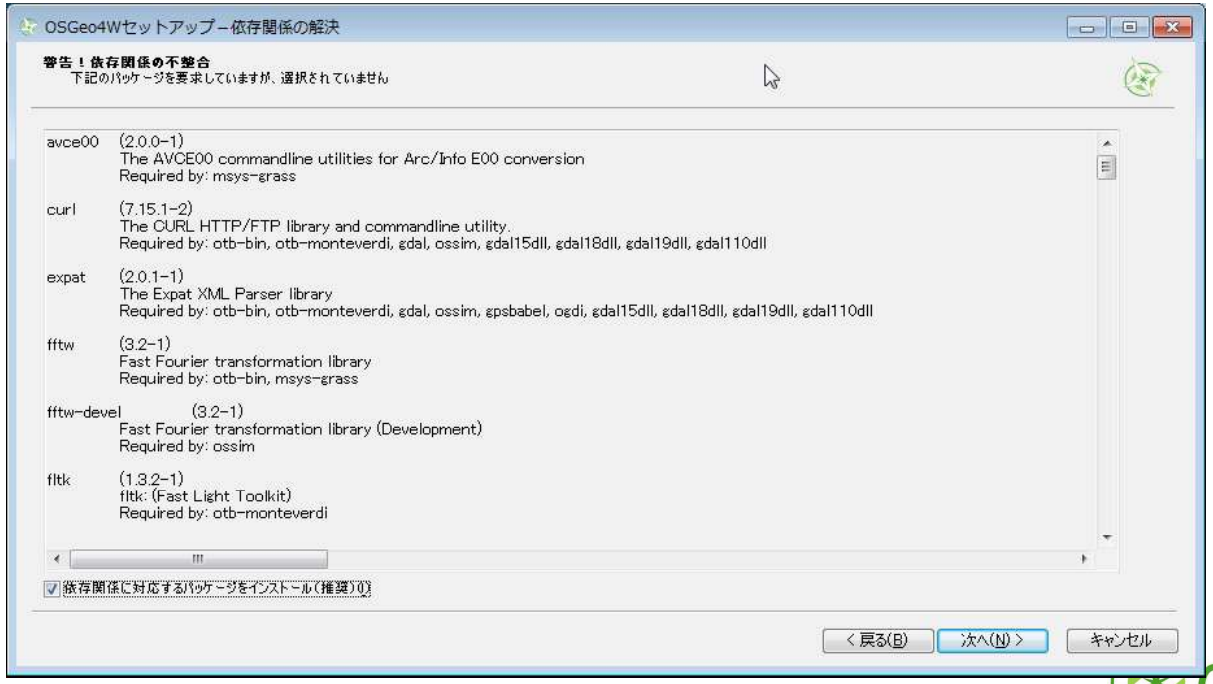
- その他
  - Desktopで「otb-monteverdi」,「saga」
  - Libで「otb-bin」,「otb-python」,「otb-wrapping」,「qgis-grass-plugin」,「python-rpy」、「pthon-rpy2」を選択
    - 実際は、プラグインなどを使用する際に、必要なものをインストールすることが多い


OSGeo4Wセットアップ－パッケージの選択
パッケージの選択
インストールするパッケージを選択してください。
Search
Clear
変更なし(K)
前回(P)
現在(C)
開発版(x)
表示(V)
カテゴリ
カテゴリ 新規 実. ソ... サイズ パッケージ
Skip n/a n/a 3,336k nettools: NetTools - tools for water management
Skip n/a n/a 8,186k openev: OpenEV - a high performance image viewer
Skip n/a n/a 128k osg-bin: OpenSceneGraph (executables)
Skip n/a n/a 206k osgearth-bin: OSG Earth (executables)
1.20.0-1 n/a 4,579k otb-monteverdi: Monteverdi - Desktop application based on Orfeo ToolBox
2.4.0-1 n/a 20,913k qgis: QGIS Desktop
Skip n/a n/a 33,319k qgis-dev: QGIS nightly build of the master
Skip n/a n/a 1k qgis-full: QGIS Full Desktop (meta package for express install)
Skip n/a n/a 1k qgis-full-dev: QGIS nightly build of the master (with additional dependencies)
2.0.8-5 n/a 8,813k saga: SAGA System for Automated Geographical Analyses
Skip n/a n/a 1,418k tora: database management GUI
Skip n/a n/a 122,453k udig: User-friendly Desktop Internet GIS
Skip n/a n/a 101,644k udig-dev: User-friendly Desktop Internet GIS - DEV
Skip n/a n/a 6,077k udigdb: User-friendly Desktop Internet GIS - Database Plugins
Libs Default
Web Default
廃止されたパッケージを隠す(H)
< 戻る(B)
次へ(N) >
キャンセル


OSGeo4Wセットアップ－パッケージの選択
パッケージの選択
インストールするパッケージを選択してください。
Search
Clear
変更なし(K)
前回(P)
現在(C)
開発版(x)
表示(V)
カテゴリ
カテゴリ
新規
実.
ソ...
サイズ
パッケージ
Skip n/a n/a 168k libpng: the official PNG reference library
Skip n/a n/a 1,093k libpq: The libpq library for accessing PostgreSQL + psql commandline cl
Skip n/a n/a 161k libspatialindex: The libspatialindex library
Skip n/a n/a 330k libtiff: A library for manipulating TIFF format image files (runtime)
Skip n/a n/a 588k libxml2: An XML read/write library
1.2.1-2 n/a 5,237k matplotlib: Python plotting package
Skip n/a n/a 12,686k msvcrt: Microsoft Visual C/C++ Runtimes
Skip n/a n/a 155k netcdf: The NetCDF library and commands for reading and writing NetC
Skip n/a n/a 35,518k oci: Oracle Instant Client
Skip n/a n/a 243k ogdi: OGDI data access library (mainly for VPF reading)
Skip n/a n/a 6,330k opencv: OpenCV
Skip n/a n/a 189k openjpeg: OpenJPEG
Skip n/a n/a 671k openssl: OpenSSL Compression Runtime
Skip n/a n/a 6,564k osg-libs: OpenSceneGraph (libraries)
Skip n/a n/a 3,687k osgearth-libs: OSG Earth (libraries)
4.0.0-1 n/a 150,262k otb-bin: Orfeo Toolbox application modules
4.0.0-1 n/a 2,013k otb-python: OrfeoToolbox python API for applications
1.10.1-1 n/a 47,936k otb-wrapping: OrfeoToolbox Wrapping - Low level Python/Java binding
Skip n/a n/a 57k pdcurses: PDCurses
Skip n/a n/a 222k proj: The PROJ.4 library and commands for coordinate system transforn
Skip n/a n/a 2,690k proj-datumgrid: Assorted PROJ.4 datum grid shift files
Skip n/a n/a 240k proj-hpgn: PROJ.4 HPGN (HARN) grid shift files
Skip n/a n/a 47,937k proj-vdatum: Assorted PROJ.4 vertical datum shift files
Skip n/a n/a 157k psycopg2: Python-PostgreSQL Database Adapter
Skip n/a n/a 272k pygments: pygments - syntax highlighting package written in Python
Skip n/a n/a 827k pyopengl: The Python OpenGL ªª Binding
Skip n/a n/a 36k pyparsing: Python parsing module
Skip n/a n/a 2,123k pyqt4: Python bindings for Qt4.
Skip n/a n/a 53k pyspatialite: Python interface to SQLite 3 + Spatialite
Skip n/a n/a 257k python-dateutil: Extensions to the standard Python datetime module
Skip n/a n/a 149k python-jinja2: A small but fast and easy to use stand-alone template er
2.7-1.5.1-1 n/a 2,149k python-numpy: Python NumPy (Numerical) Extension
Skip n/a n/a 549k python-pil: Python Imaging Library (PIL)
1.0.3-2 n/a 21k python-rpy: python bindings to R.
2.3.9-1 n/a 96k python-rpy2: python bindings to R 3.0.2
0.11.0-1 n/a 11,673k python-scipy: Scientific Library for Python
Skip n/a n/a 5,461k python-win32: Python for Windows extension
Skip n/a n/a 163k python-xlrd: python library for developers to extract data from Microsof
Skip n/a n/a 98k python-xlwt: python library to create spreadsheet files compatible with
Skip n/a n/a 165k pytz: World timezone definitions, modern and historical
Skip n/a n/a 4,602k qgis-common: QGIS common
Skip n/a n/a 1,146k qgis-grass-plugin: GRASS plugin for QGIS
Skip n/a n/a 570k qscintilla: Qt source code editing component.
廃止されたパッケージを隠す(H)
< 戻る(B)
次へ(N) >
キャンセル

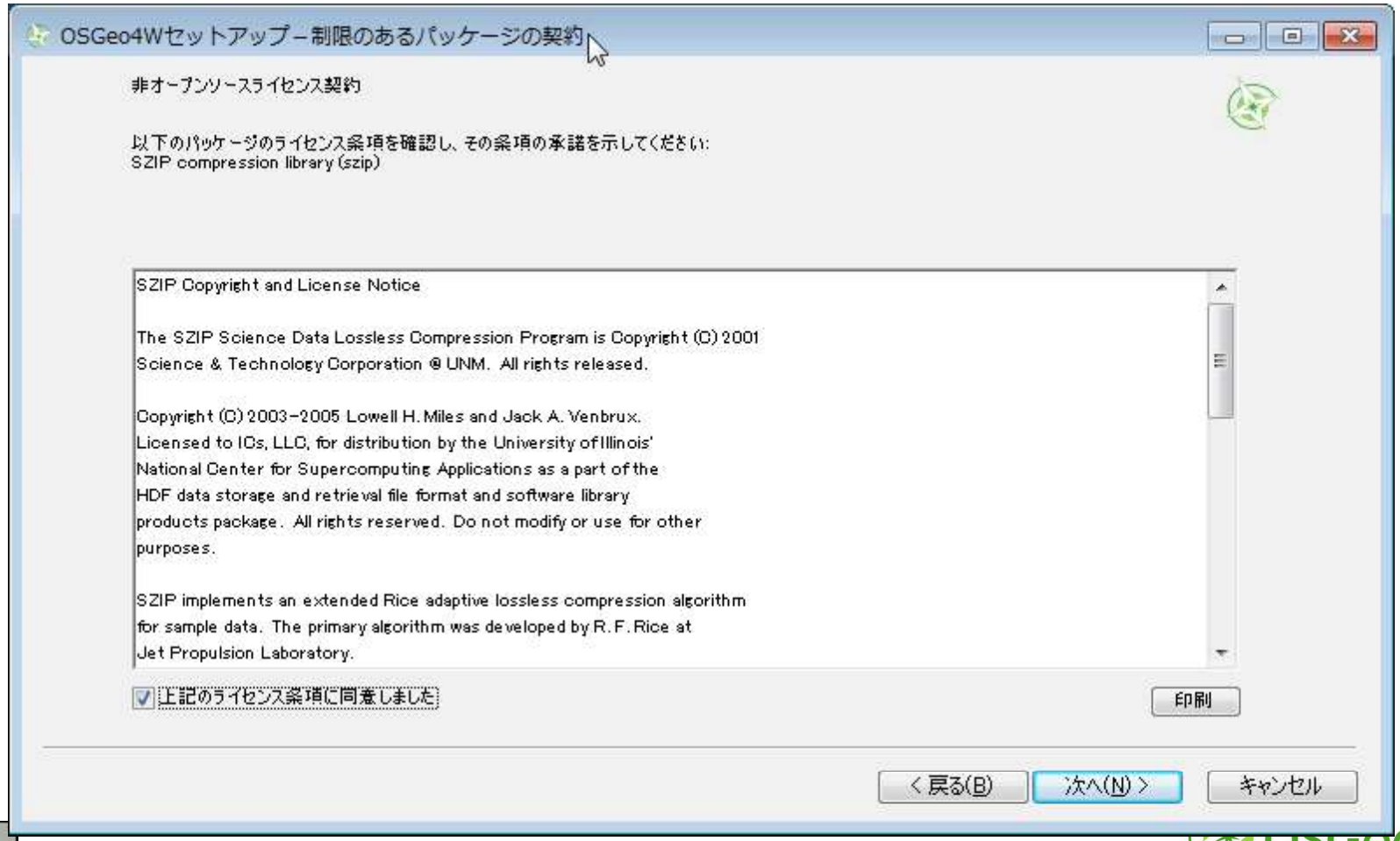
OSGeo4W版QGISのインストール

- 依存関係の確認
  - 選択したソフトを走らせるために必要なものを自動的に選択してくれる



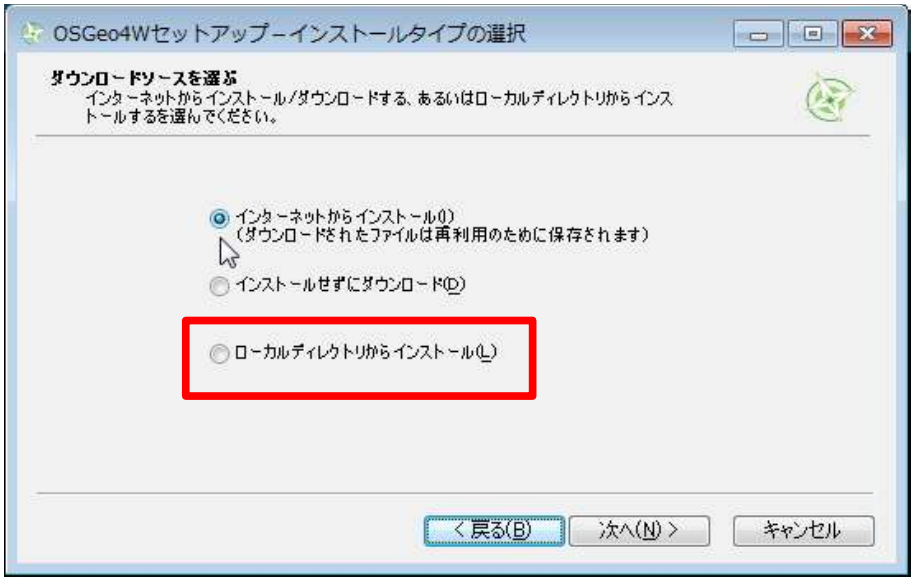
OSGeo4W版QGISのインストール

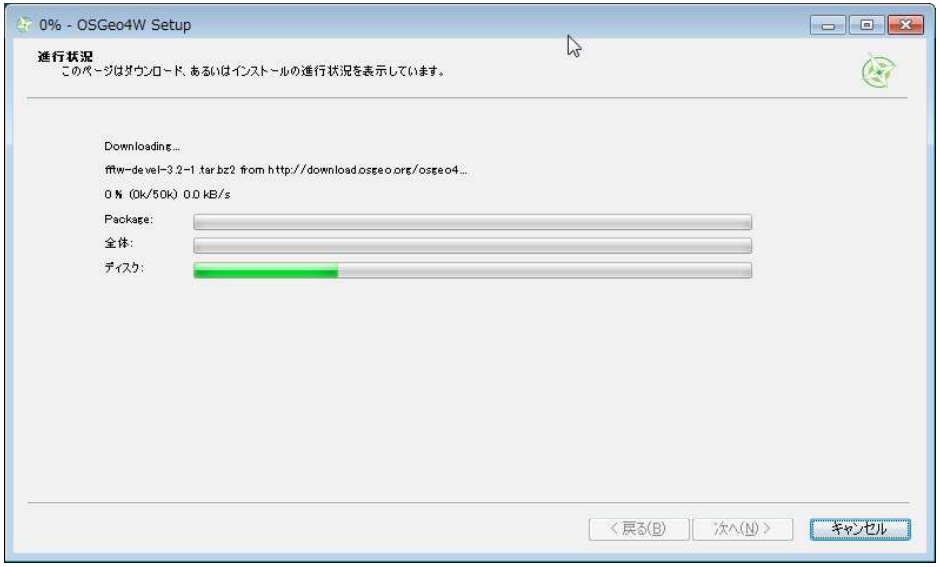
- ライセンスの同意
  - 非オープンソースの個別ライセンスを確認



OSGeo4W版QGISのインストール

- 後はダウンロードして自動的に実行
  - 今回はあらかじめDLしたファイルを使用
    - ダウンロードソースで「ローカルディレクトリからインストール」を選択

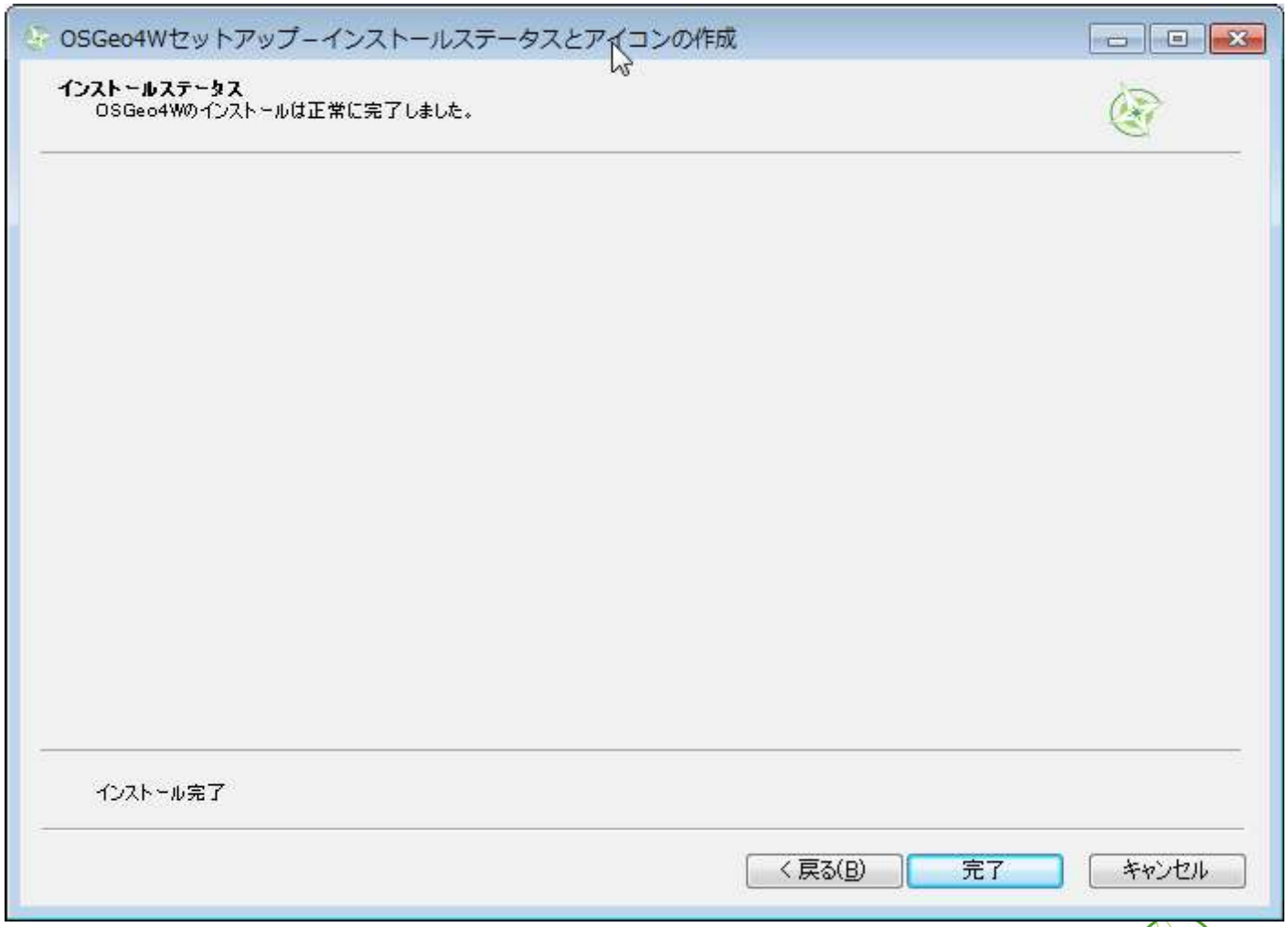




OSGeo.JP

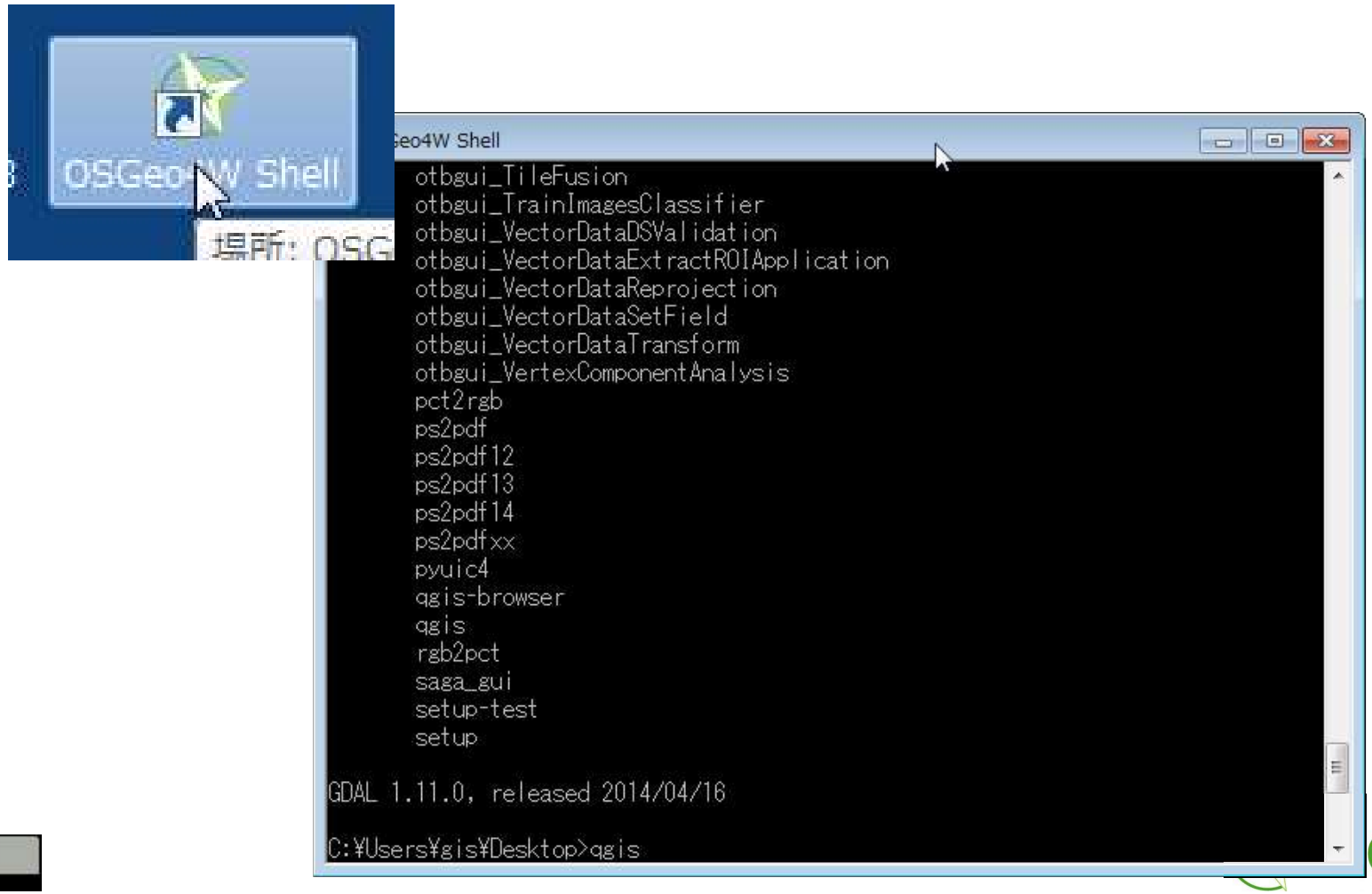
OSGeo4W版QGISのインストール

- 下の画面が出れば完了



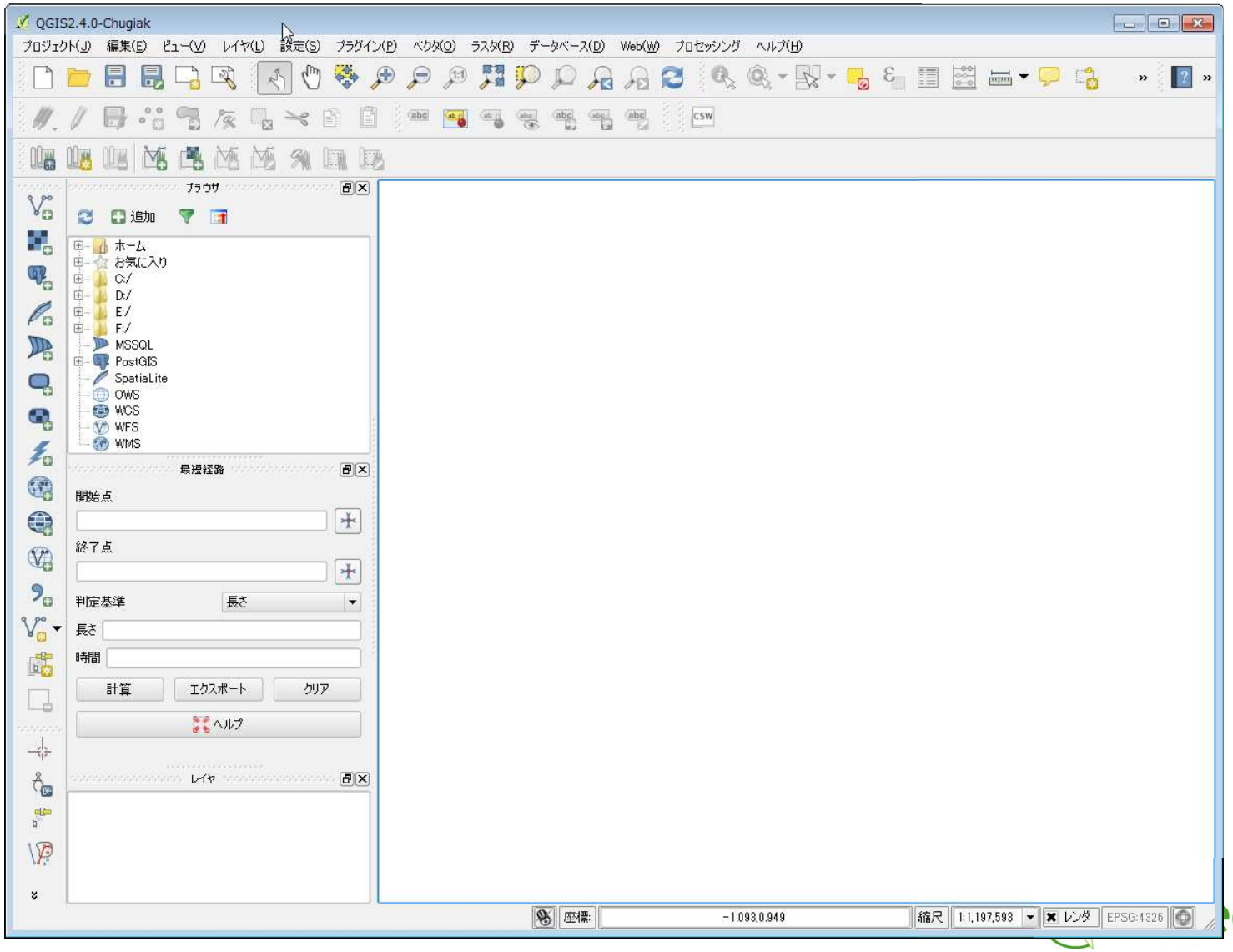
起動方法

- OSGeo4W Shellをダブルクリック
- 「qgis」と入力してエンターすれば起動



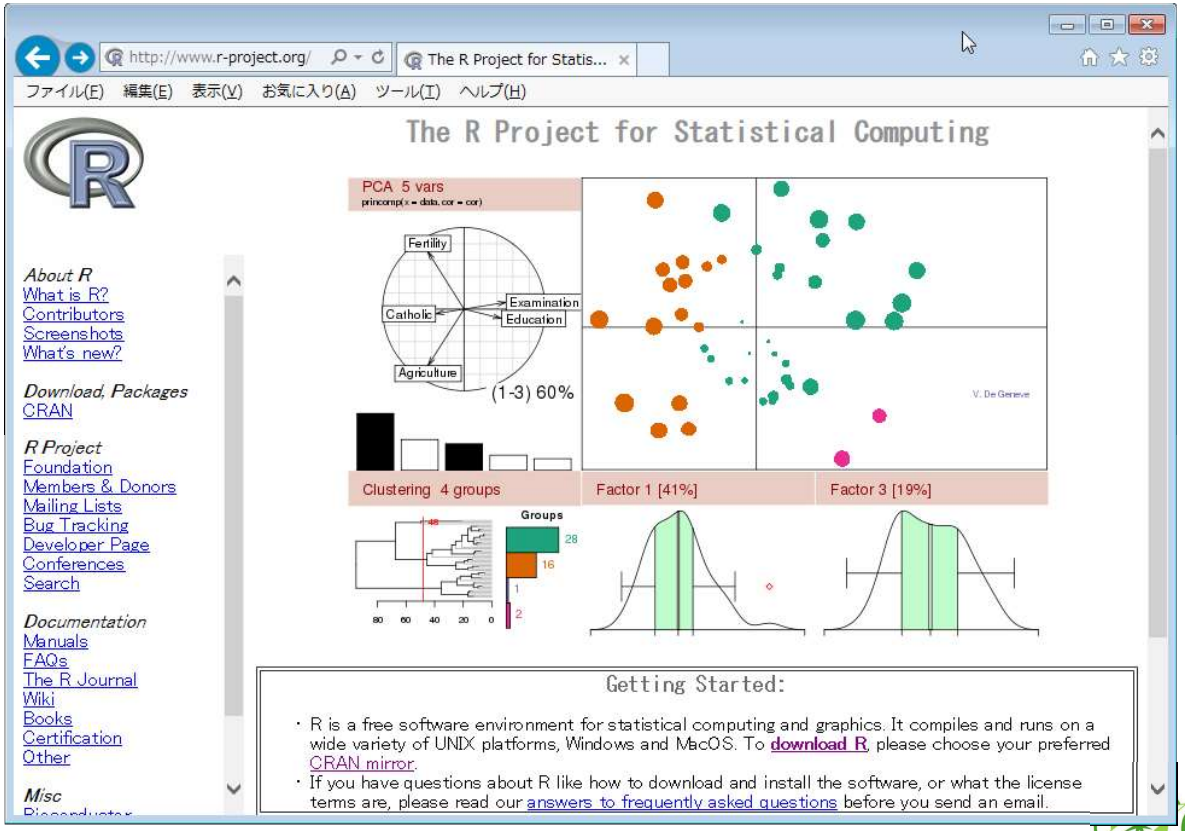
起動方法

- GUIはスタンドアローン版と変わりません



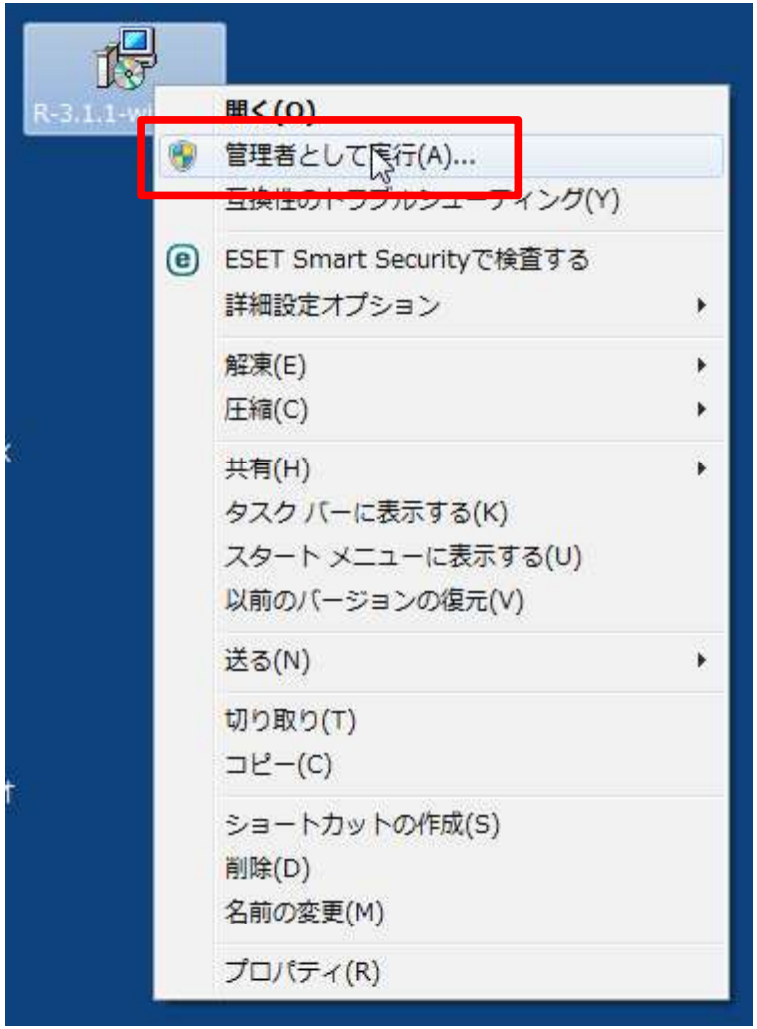
Rのインストール

- QGISと同じくオープンソースの統計ソフト
  - ファイルを配布するのでインストールを実行
    - 同じくC:¥tmpの中

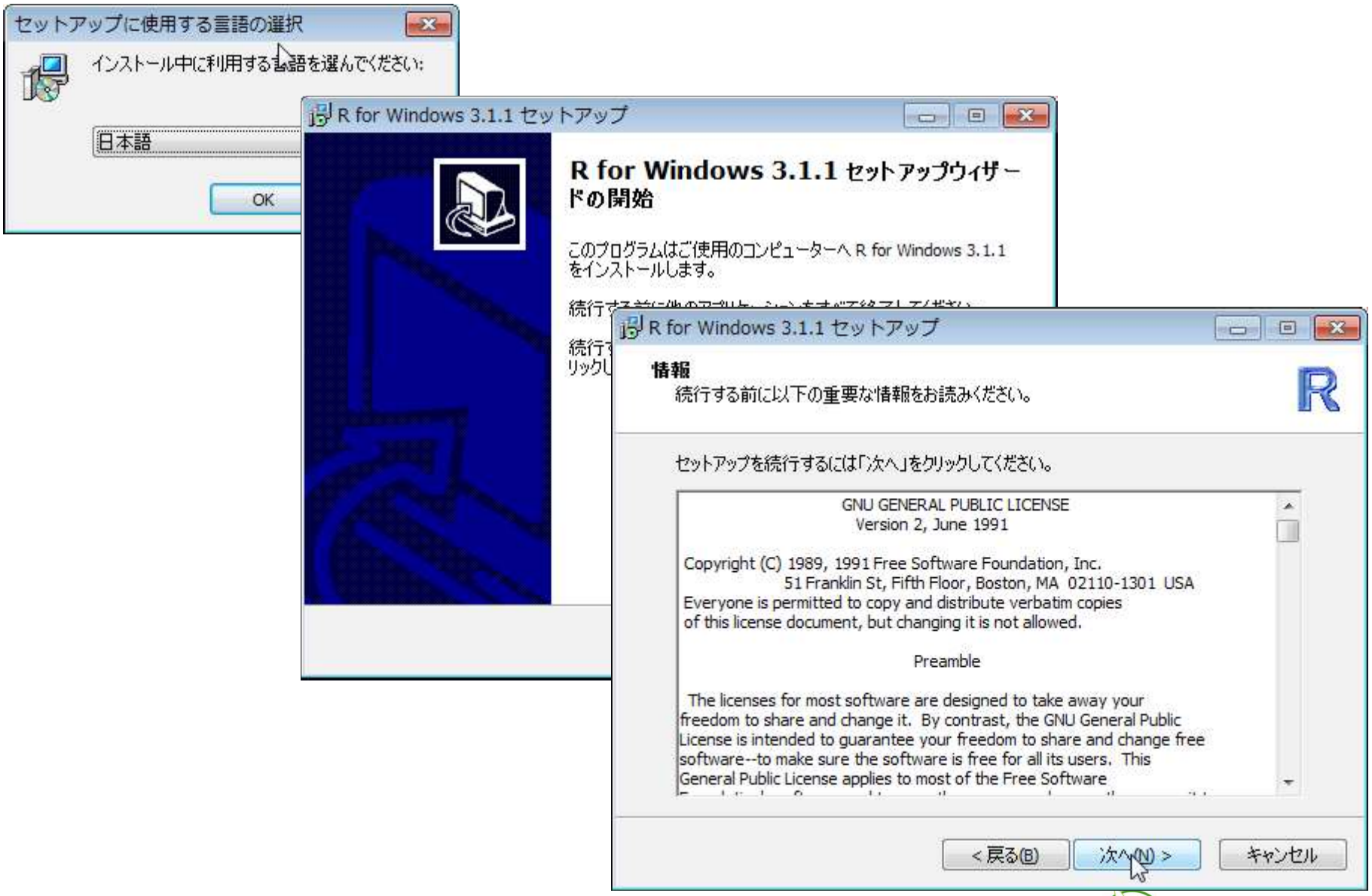


Rのインストール

- 管理者として実行を選択

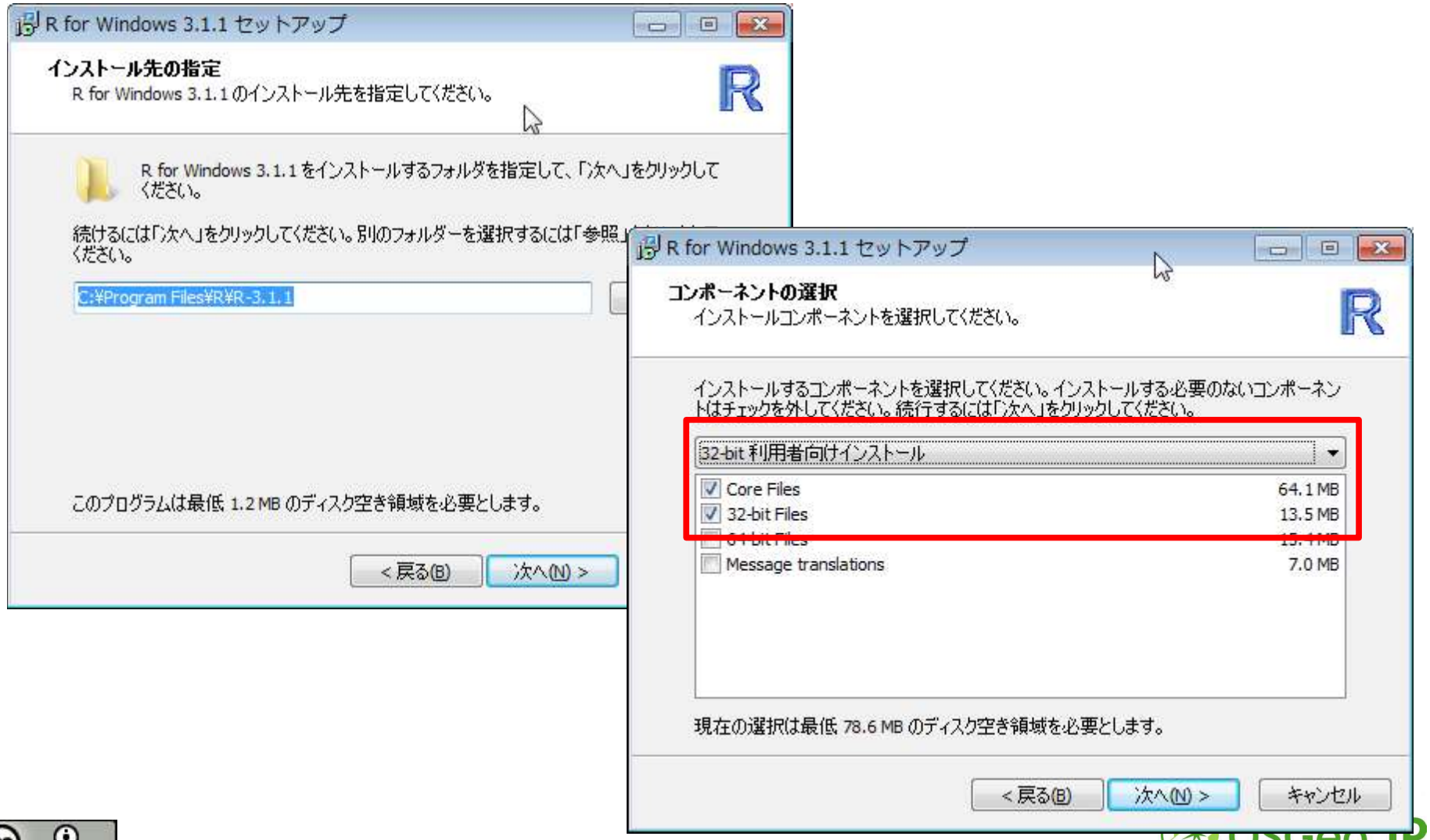


Rのインストール



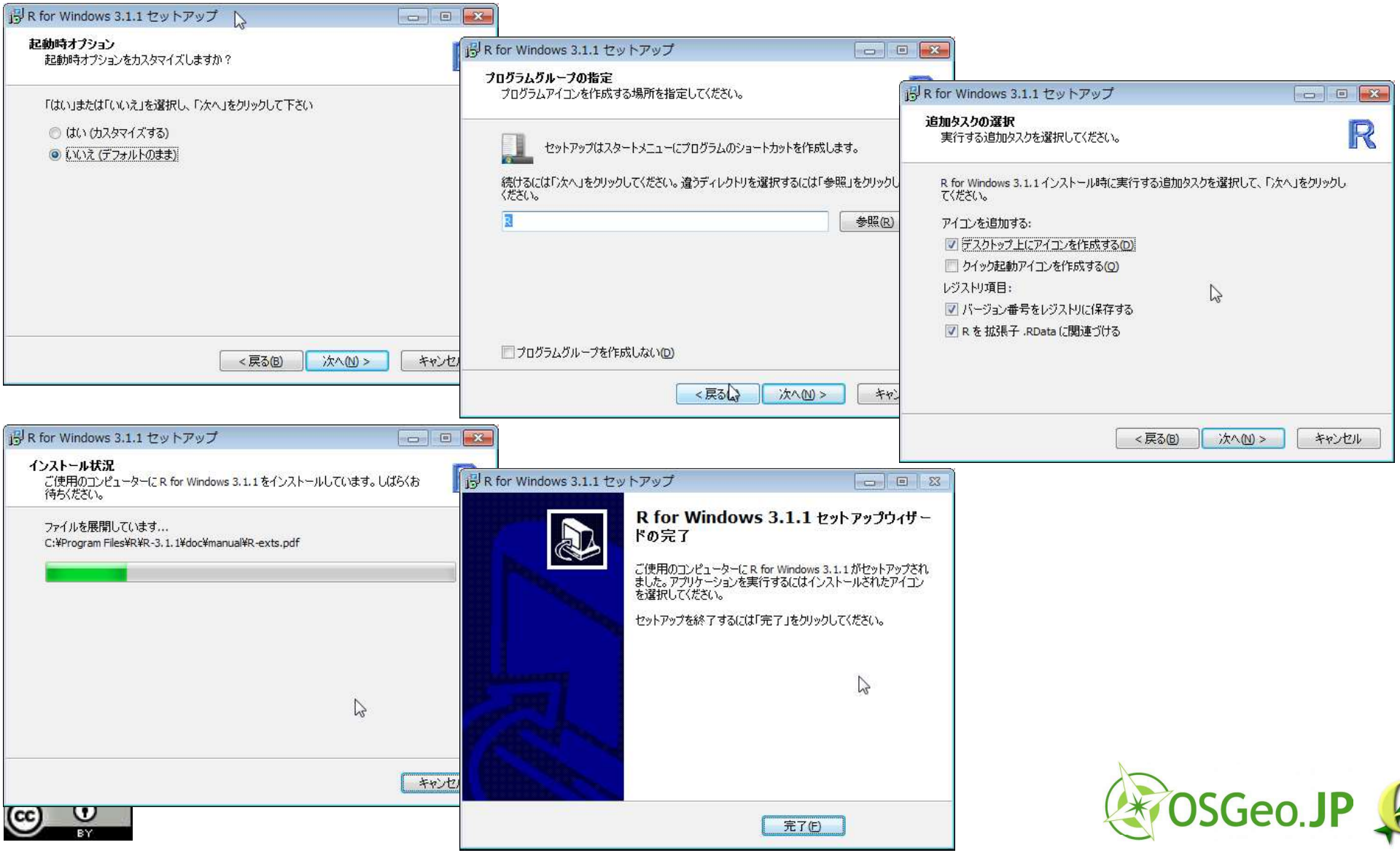
Rのインストール

- 32bit利用者向けインストールを選択してください



Rのインストール

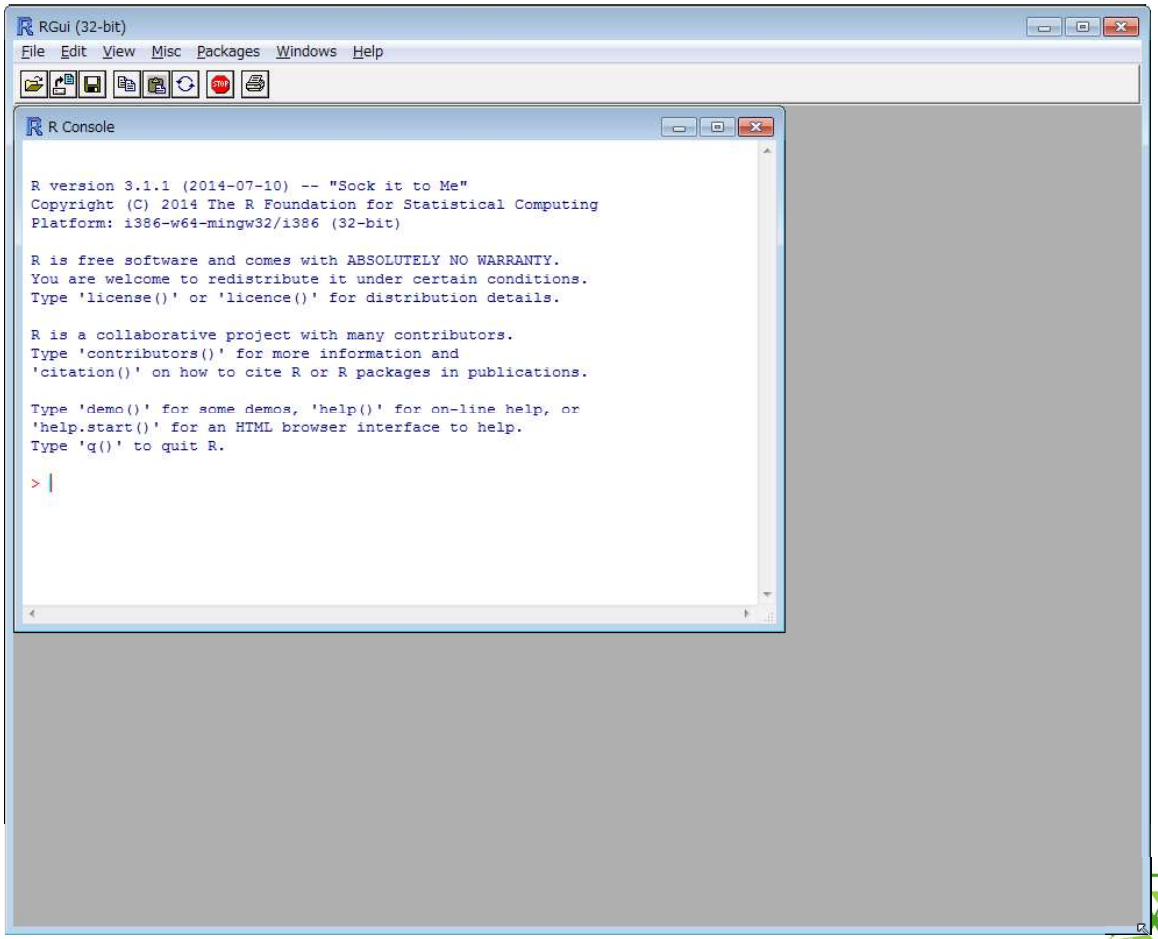
- デフォルトのままインストール



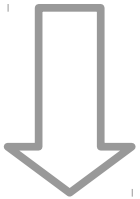
OSGeo.JP

Rのインストール

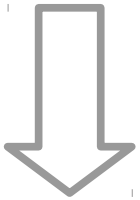
- 完了したら、起動を確認。
  - 後半で使います。



インストール



QGISの分析ツールの概要

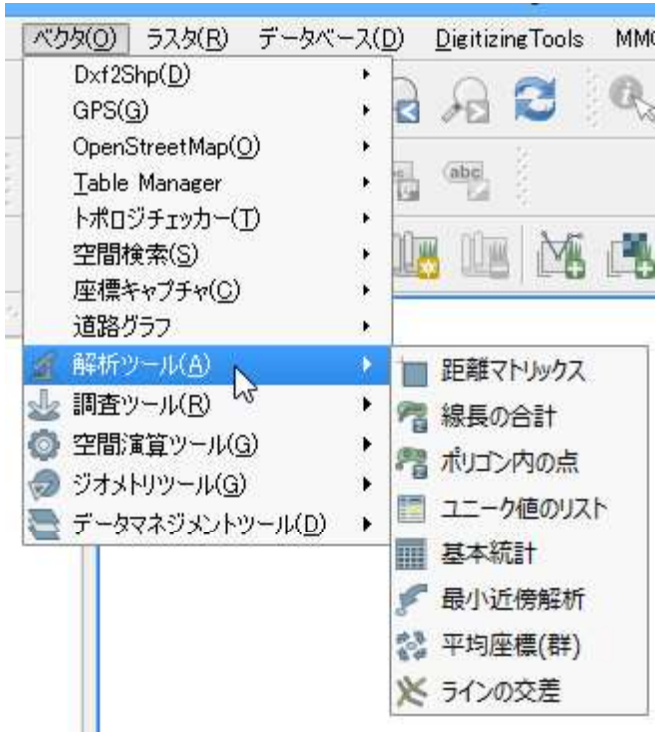


ベクターデータの分析機能

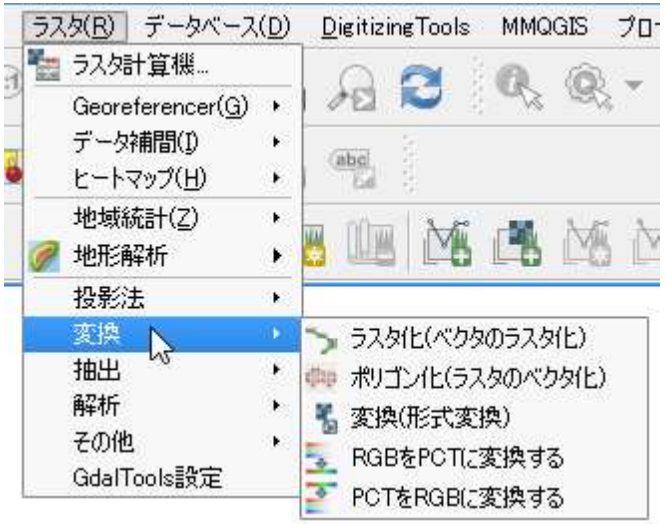
QGISの分析機能

- プラグインとして実装されている
  - 初めから入っているのは以下の２つ
- fTools
  - ベクタに関する基本的分析機能
    - ベクタメニューから選択
- GDAL Tools
  - ラスタに関する基本的分析機能
    - ラスタメニューから選択
- 本セミナーでは主にこの2つを説明

fTools



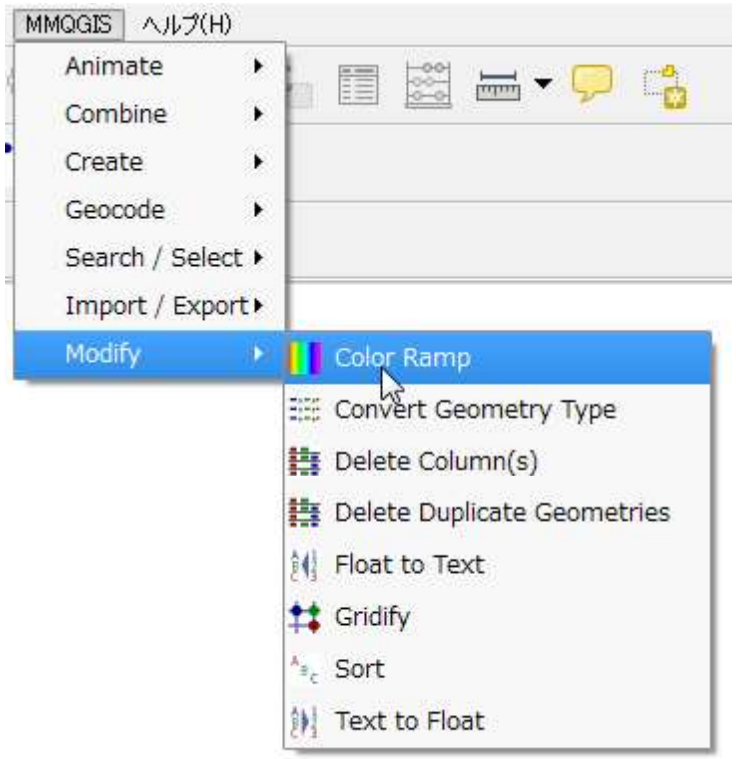
GDAL Tools



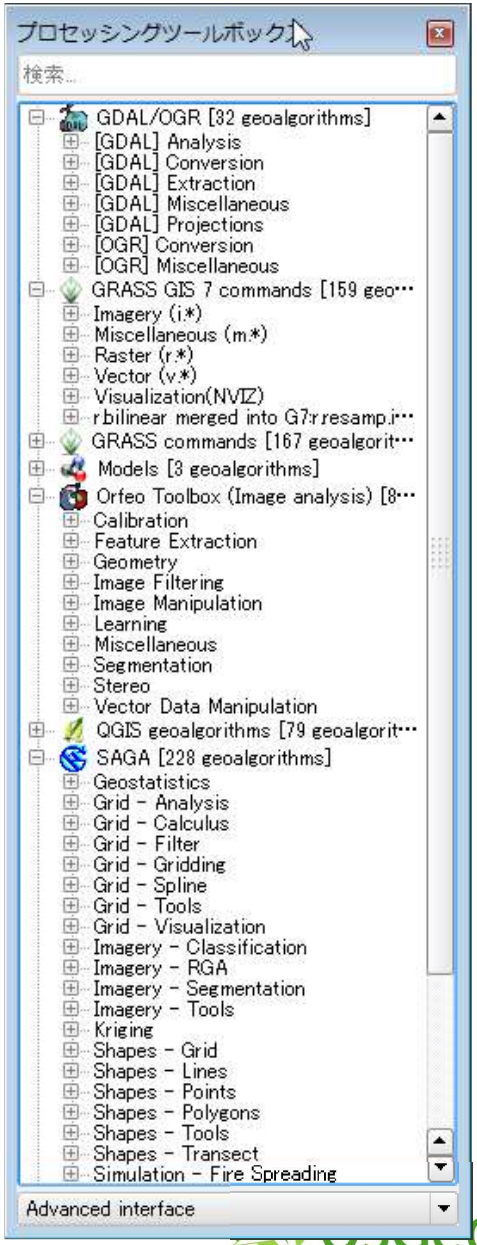
QGISの分析機能

- mmqgis
  - fToolsと同様にベクタの分析を行う
    - 補完的に利用できる
- プロセッシングツールボックス
  - 高度な分析を行うプラグイン
    - GRASS, R, SAGAと連携可能
- 高機能な分析ツール
  - 別途インストールが必要
  - そのほかにも多くの単機能分析ツールがある。

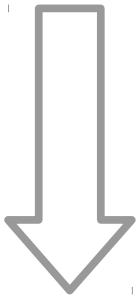
mmqgis



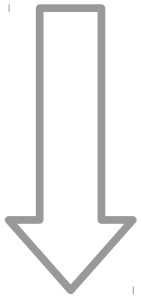
プロセッシング



QGISの分析機能の概要



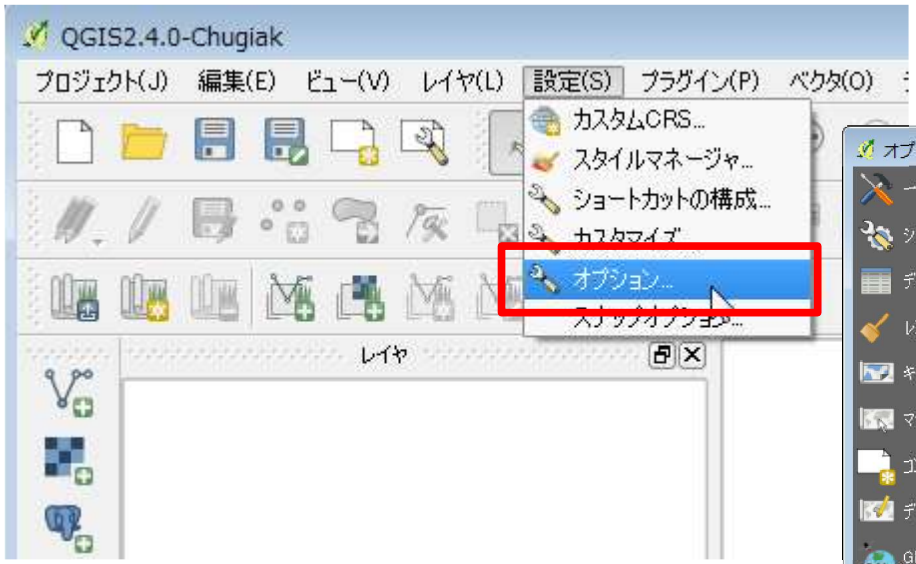
ベクタデータの分析

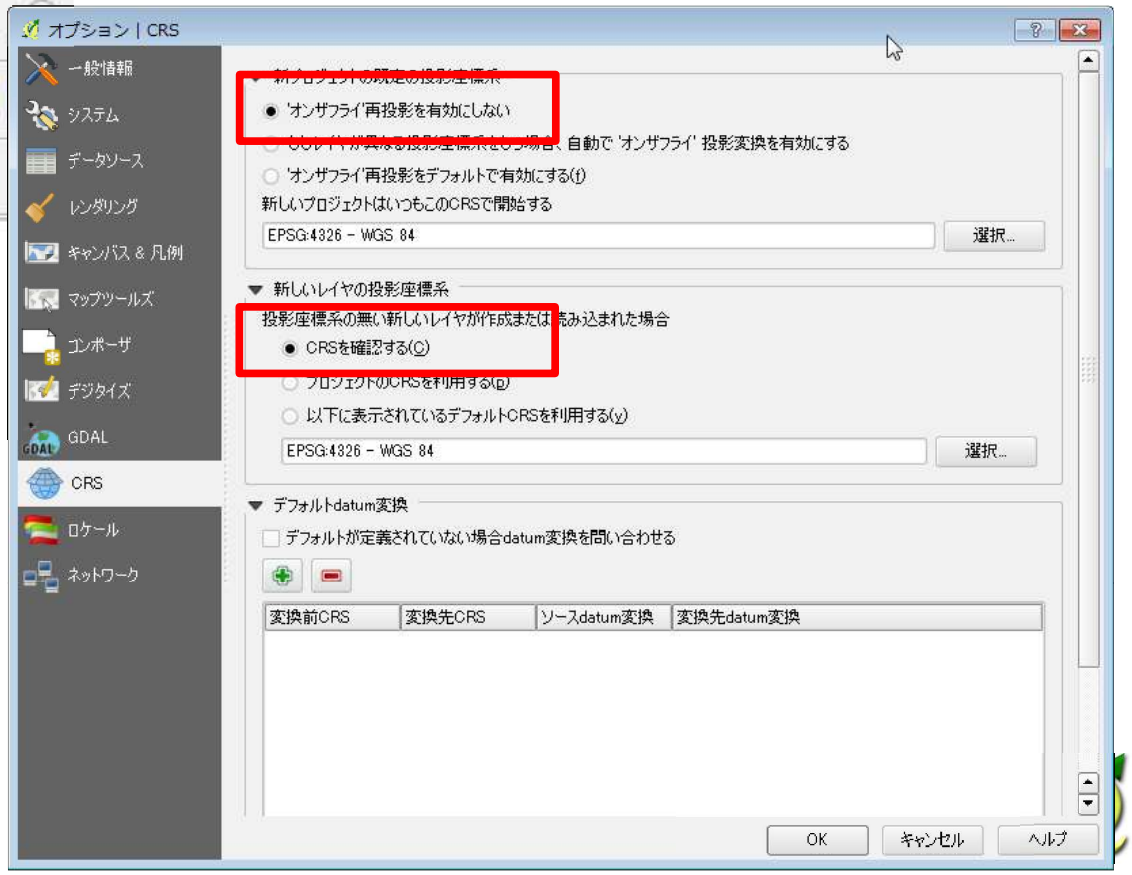


ベクタデータの活用

QGISを起動

- まずQGISを起動
  - CRSセッティングを変更



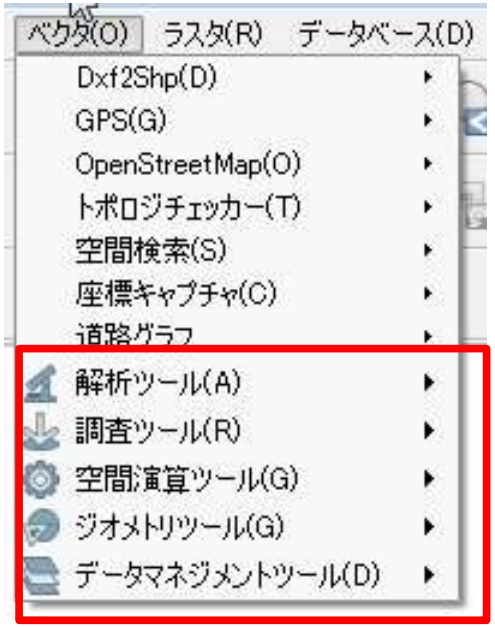


座標系に関する設定

- GISデータは位置情報を持っている
  - 測地系・座標系に関する情報が無い場合もある
    - そうしたデータを開くときのルールを決めておく
    - 設定しないと地図が重ならない場合も
- メニューの「設定」→「オプション」をクリック
  - 「オプション」が表示されるので、「CRS」を選択、「CRSを確認する」をチェックして「OK」
    - CRSはCoordinate reference System, 座標参照系の略です。

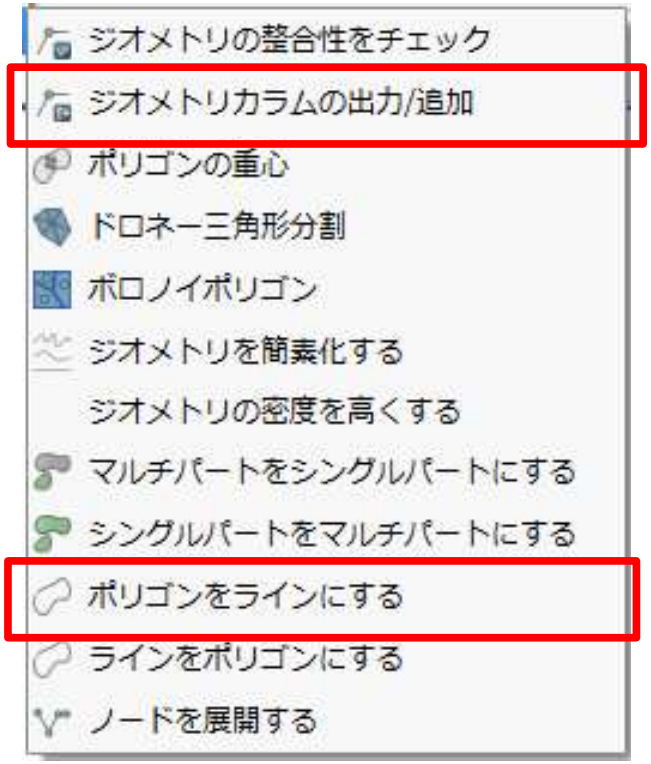
QGISのベクトル分析機能

- メニューのベクタから選択
  - 解析ツール
  - 調査ツール
  - 空間演算ツール
  - ジオメトリツール
  - データマネージメントツール
    - そのほかはデフォルトで入っているプラグイン



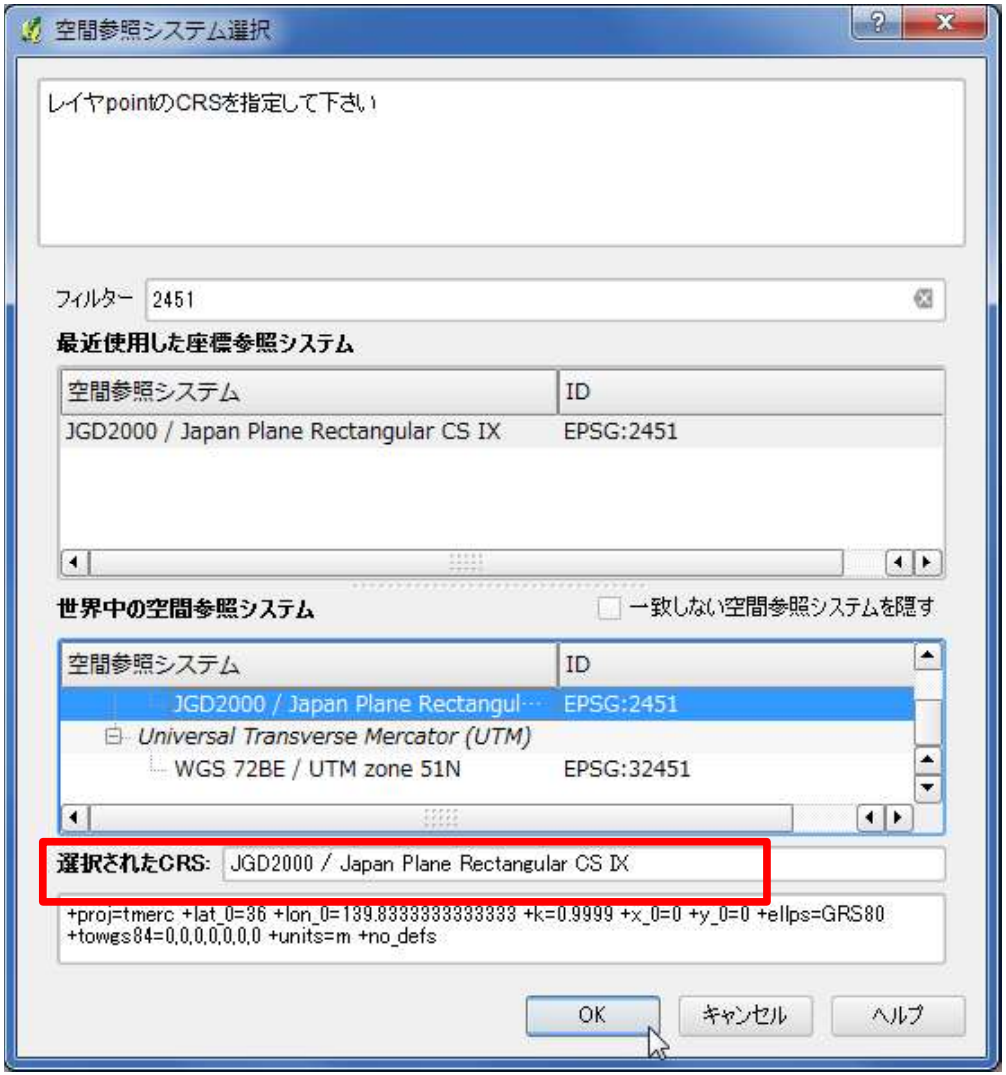
ジオメトリツール

- ベクタデータの幾何学的属性（位置，形，形式など）の操作
  - ジオメトリカラムの出力/追加
    - 座標や面積を追加
  - ポリゴンをラインにする
    - 形式の変換



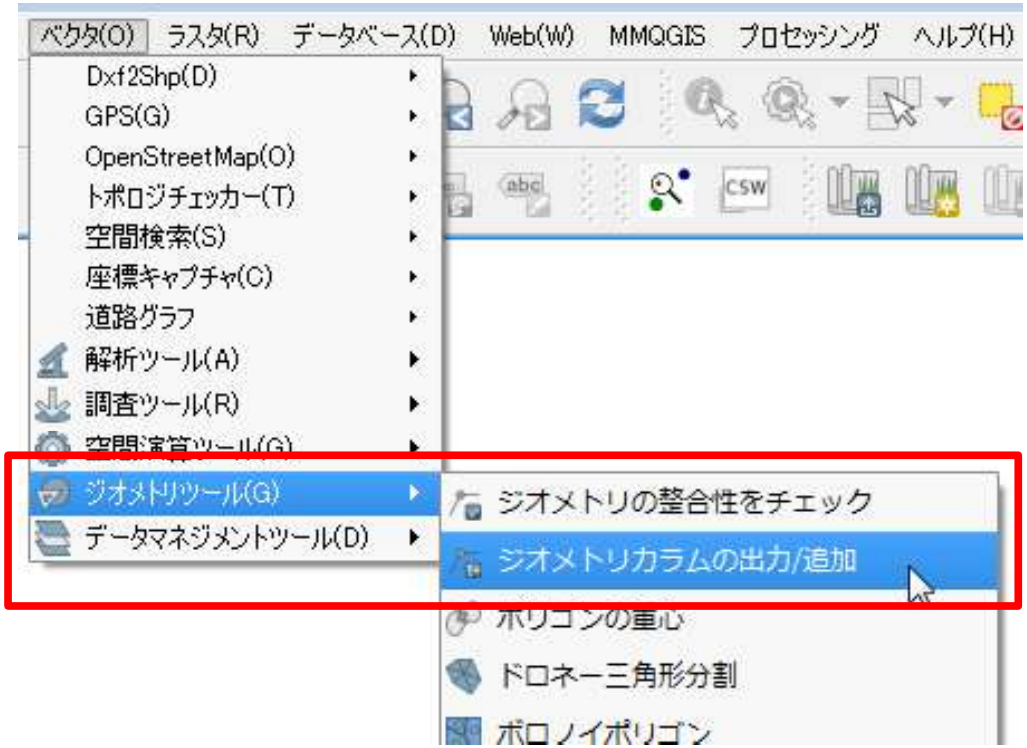
ジオメトリカラムの追加

- “C:¥GIS_DATA¥Advance”のpoint.shpを開く
  - 測地系はEPSG:2451



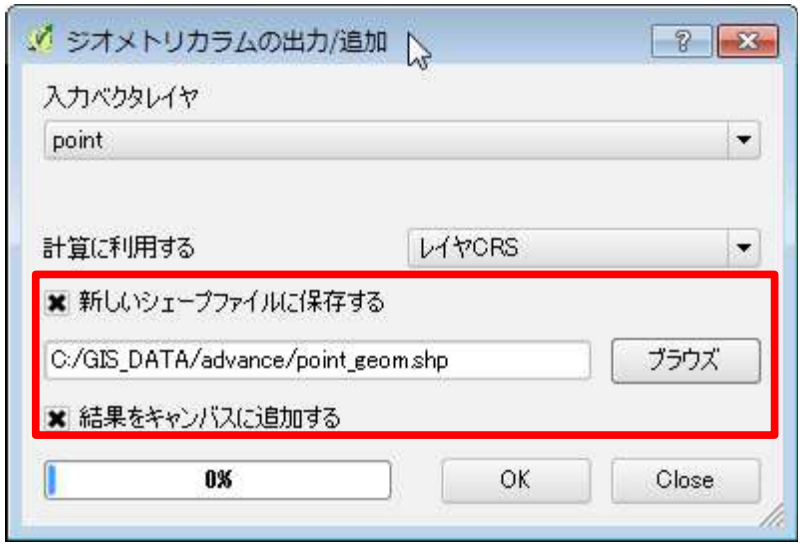
ジオメトリカラムの追加

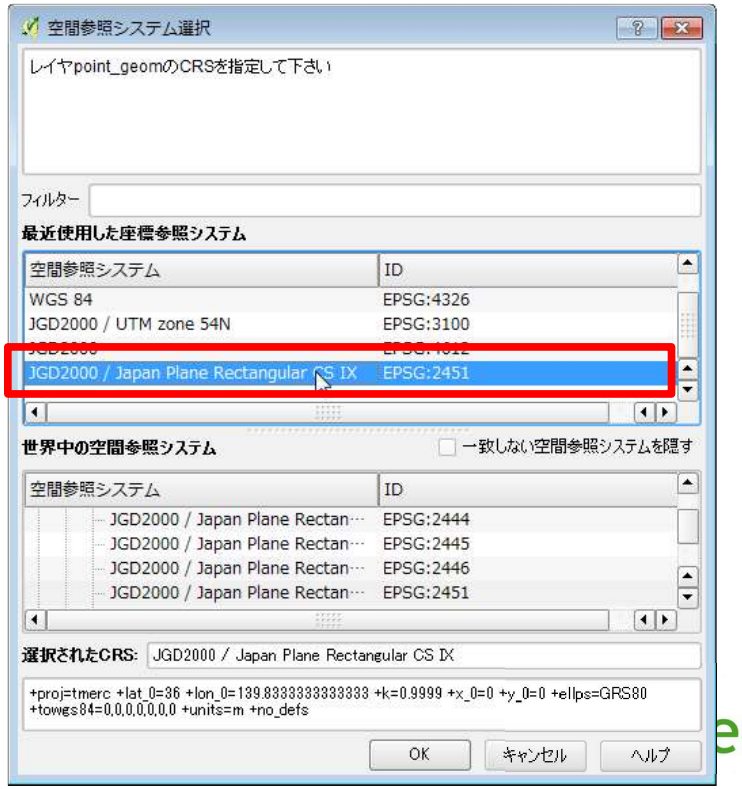
- メニューから「ベクタ」→「ジオメトリツール」
  →「ジオメトリカラムの出力/追加」をクリック



ジオメトリカラムの追加

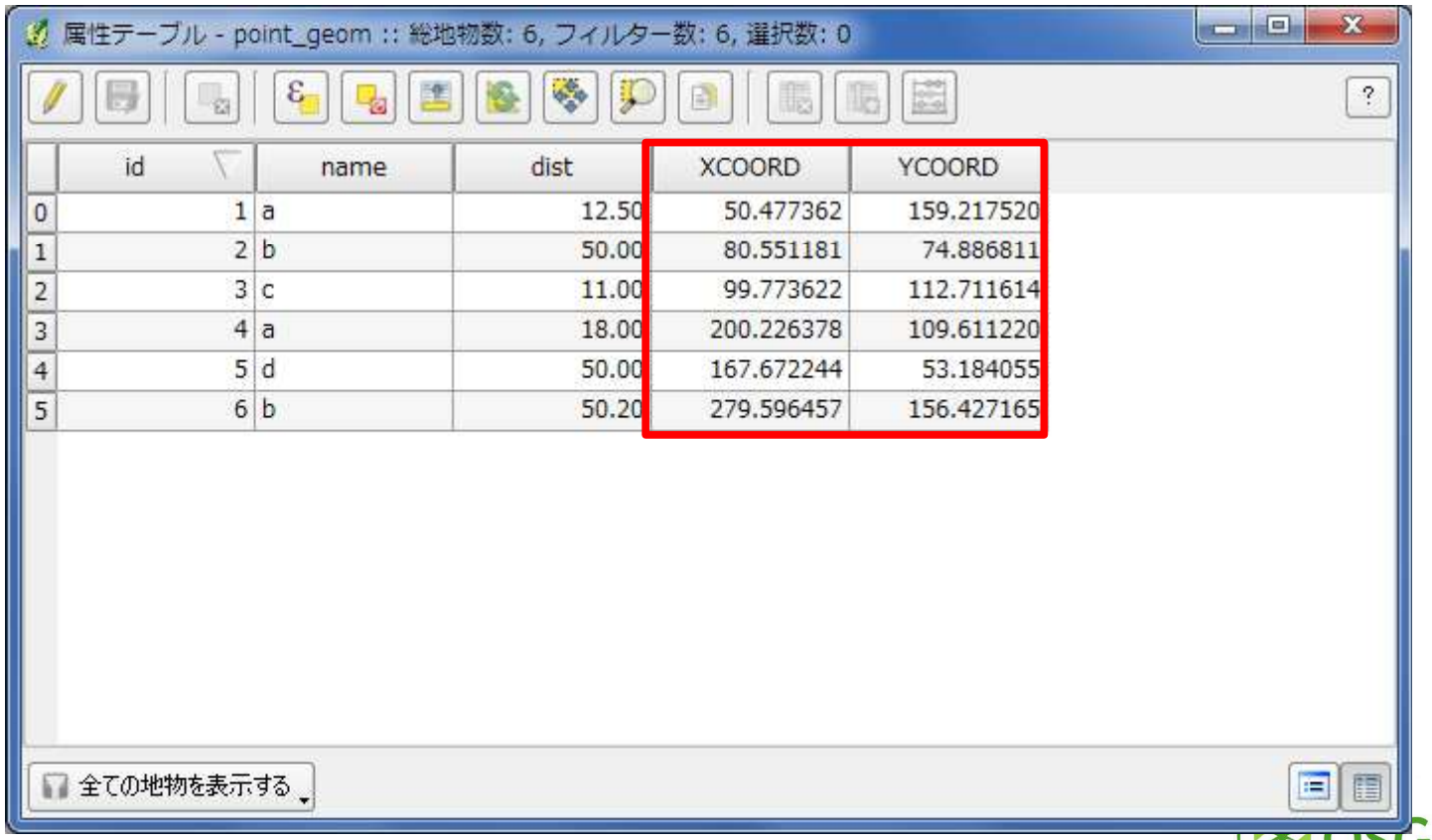
- ウィンドウが表示されたら，"新しいシェープファイルに保存する”と「結果をキャンバスに表示する」にチェック
- 出力先を”point_geom.shp”とし，OK





ジオメトリカラムの追加

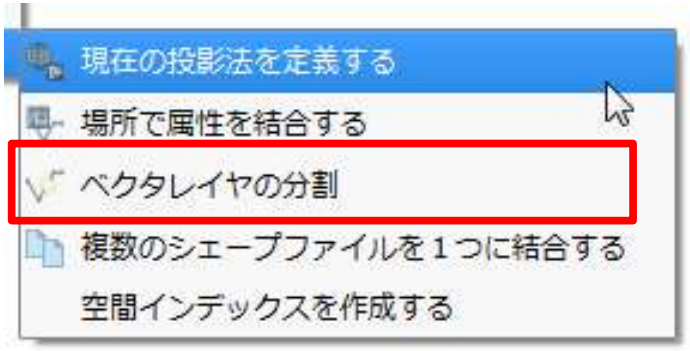
- テーブルを開くとXとYの座標値が入っている
  - XCOORDとYCOORD
  - ラインの場合は長さ，ポリゴンの場合は面積と周長

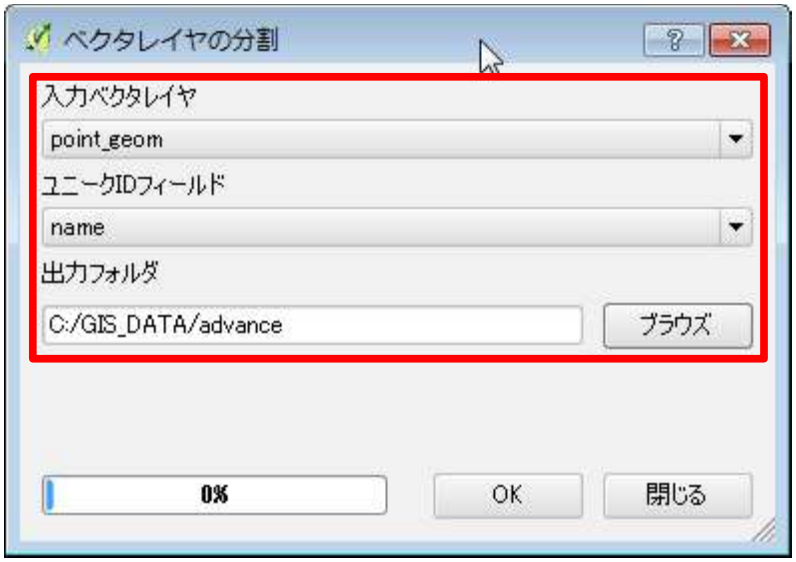


	id	name	dist	XCOORD	YCOORD
0	1	a	12.50	50.477362	159.217520
1	2	b	50.00	80.551181	74.886811
2	3	c	11.00	99.773622	112.711614
3	4	a	18.00	200.226378	109.611220
4	5	d	50.00	167.672244	53.184055
5	6	b	50.20	279.596457	156.427165

データマネージメントツール

- ベクタデータの結合や分割，投影法の定義など
  - ”ベクタレイヤの分割”を選択
  - 入力ベクタレイヤに”point_geom”，ユニークIDに”name”，出力フォルダを　”C:¥GIS_DATA¥advance”
  - a, b, c, dの4つのファイルができる
    - 個別属性でshpを分割



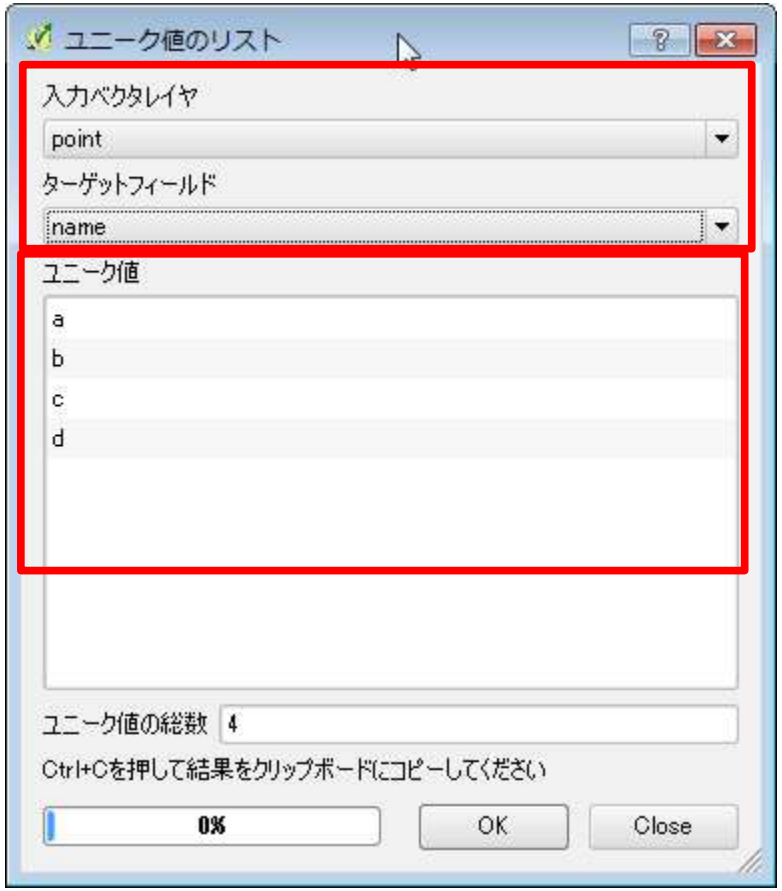


解析ツール

- 距離や属性値の統計に関するツール
  - 距離マトリクス
    - 異なるレイヤの点間の距離
  - ポリゴン内の点
    - ポリゴンの中に含まれる点を数える
  - ラインの交差
    - 線が交差してる場所に点を作成

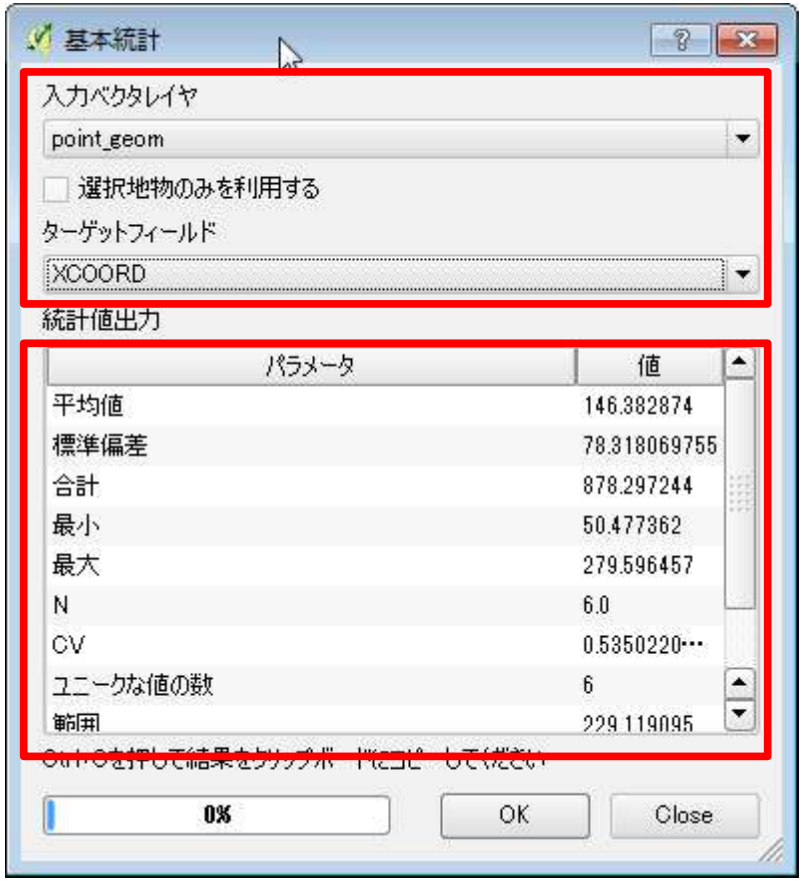
ユニーク値のリスト

- レイヤにpoint_geom, ターゲットフィールドに nameを選択してOK。
  - name列の個別値がリストアップされる
    - 属性が多い場合に有効



基本統計

- レイヤにpoint_geom，ターゲットフィールドにXCOORDを選択して実行
  - 平均，標準偏差等々が出力
    - 市町村，農村集落単位データの集計など

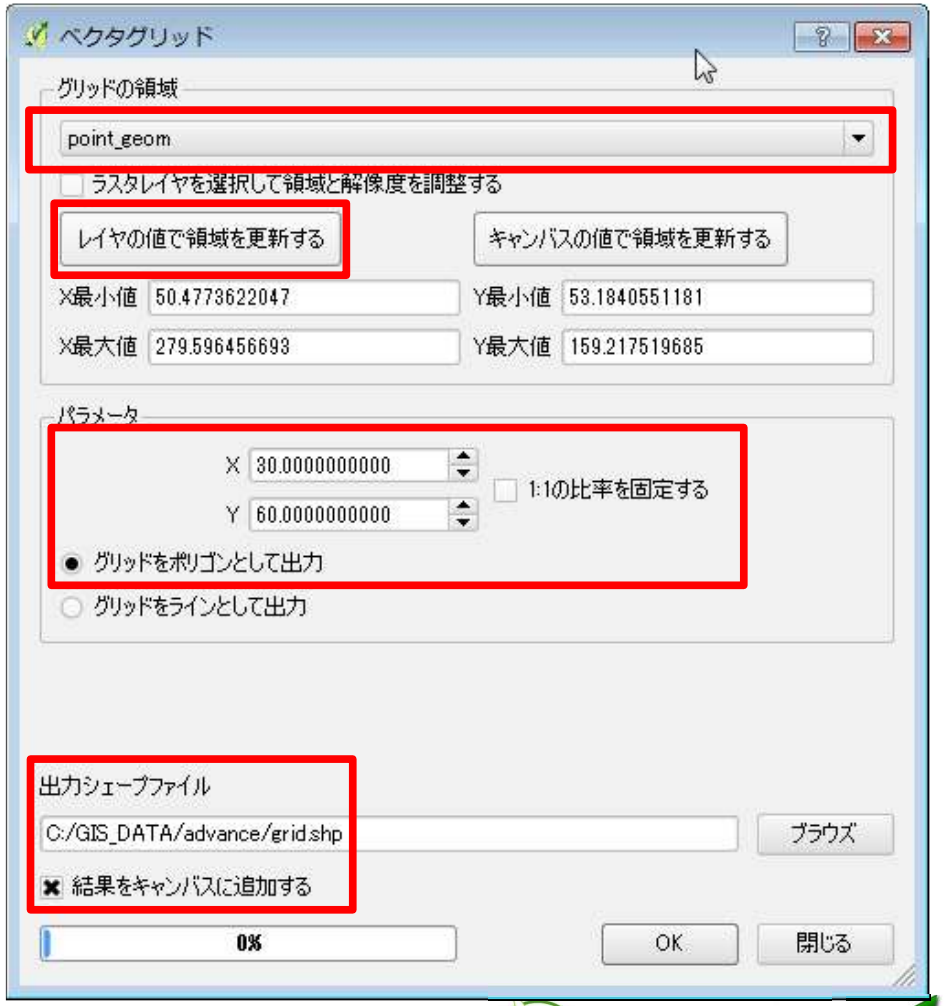


調査ツール

- データの選択や定型的データの作成
  - ランダム選択・サブセットのランダム選択
    - 一定の数や割合で選択。サブセットの場合は特定の属性値。
  - ランダム点群・規則的な点群
    - ランダム，または規則的な点を生成

ベクタグリッド

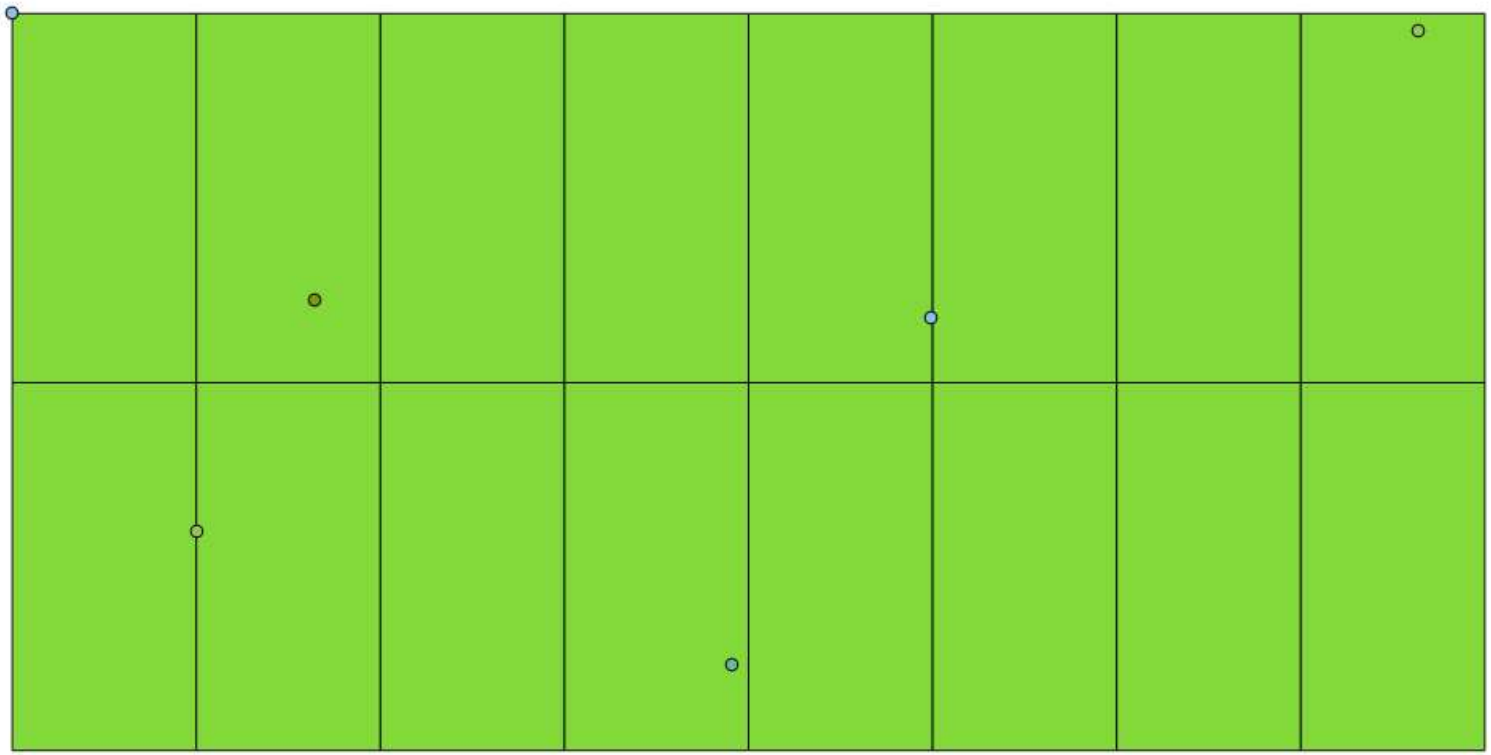
- 格子状のベクタデータを作成
  - 線，面両方ともできる。
  - point_geomを選択
  - レイヤの値で更新をクリック
  - パラメータとしてXに30，Yに60
  - グリッドをポリゴンとして出力にチェック



OSGeo.JP

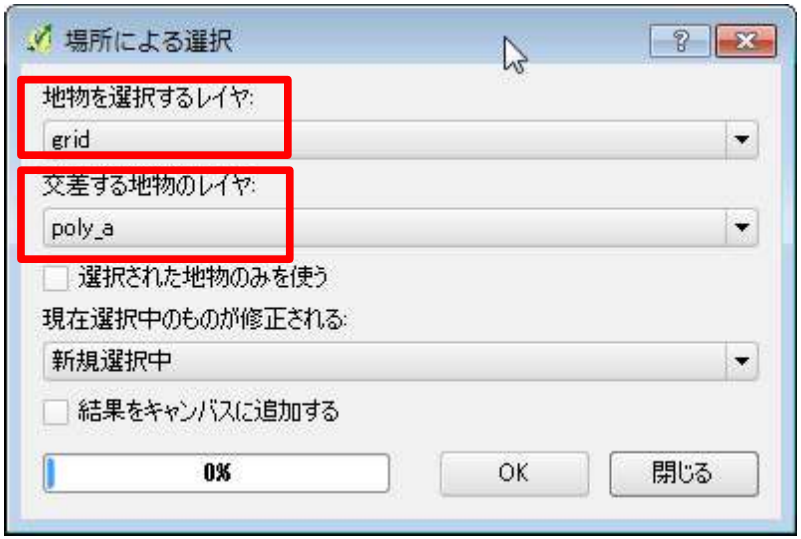
ベクタグリッド

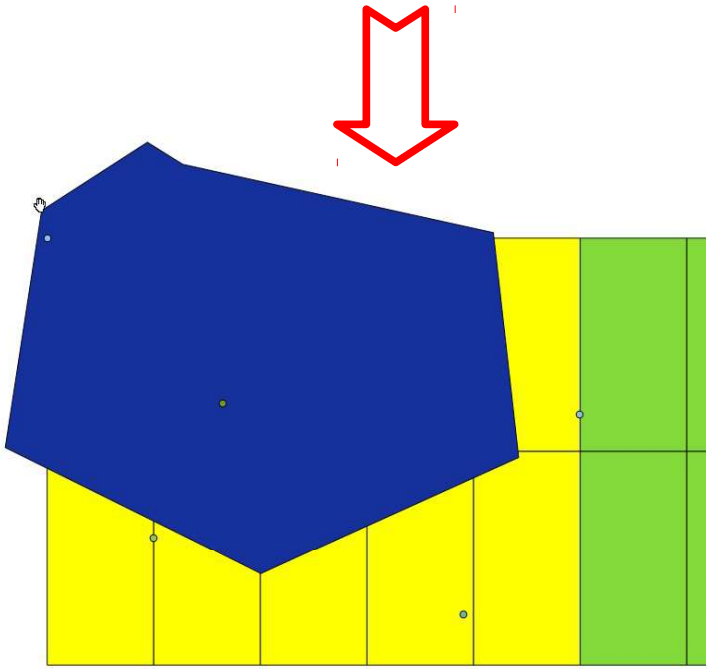
- 点を内包するグリッドが作られる
  - 左上を基準に作成される



場所による選択

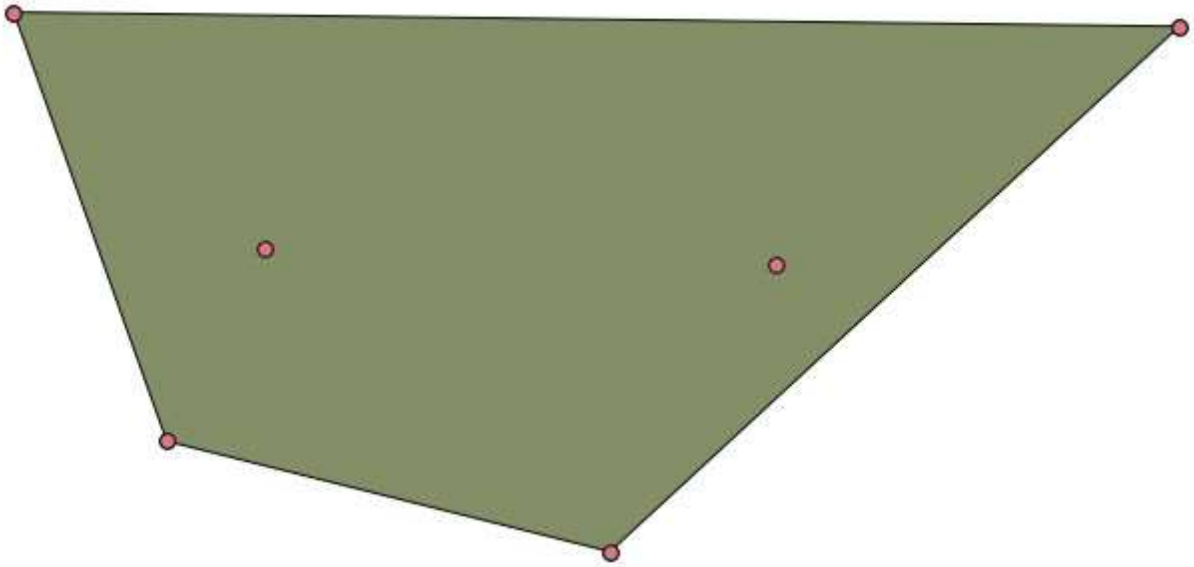
- 異なるレイヤ間で重なったデータを選択
  - poly_aを開く
  - “地物を選択するレイヤ”にgrid
    - こちらが選択される
  - “交差する地物のレイヤ”にpoly_a
    - OKをクリック
  - gridのうちpoly_aと重なっているものが選択される





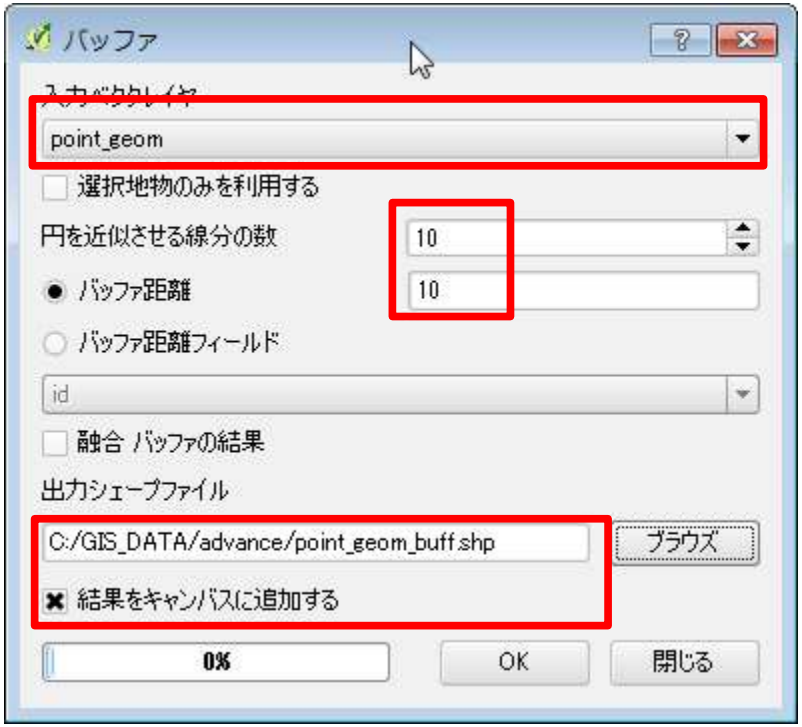
空間演算ツール

- 最も良く使うツール
  - 凸包
    - 対象となる点を内包する最小の多角形を作成



バッファ

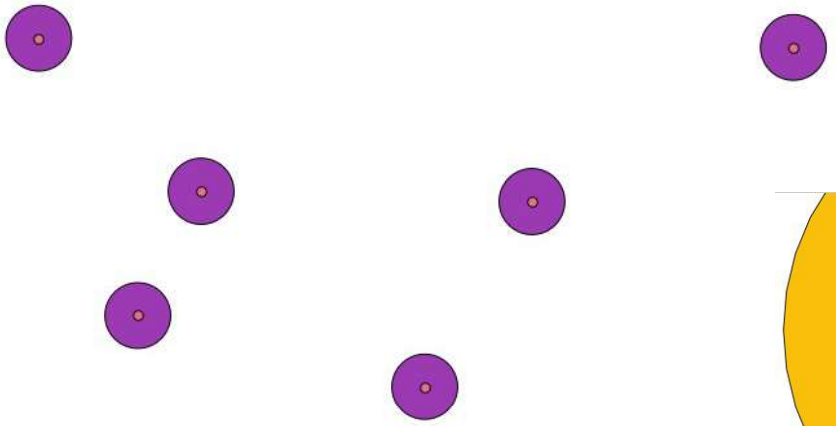
- ベクタレイヤから”バッファ”を発生させる
  - ベクタレイヤとして”point_geom”を選択
  - 円を近似させる線分の数に10
  - バッファ距離に10
    - 表示座標系での距離
  - 出力をpoint_buff_10m.shpを指定



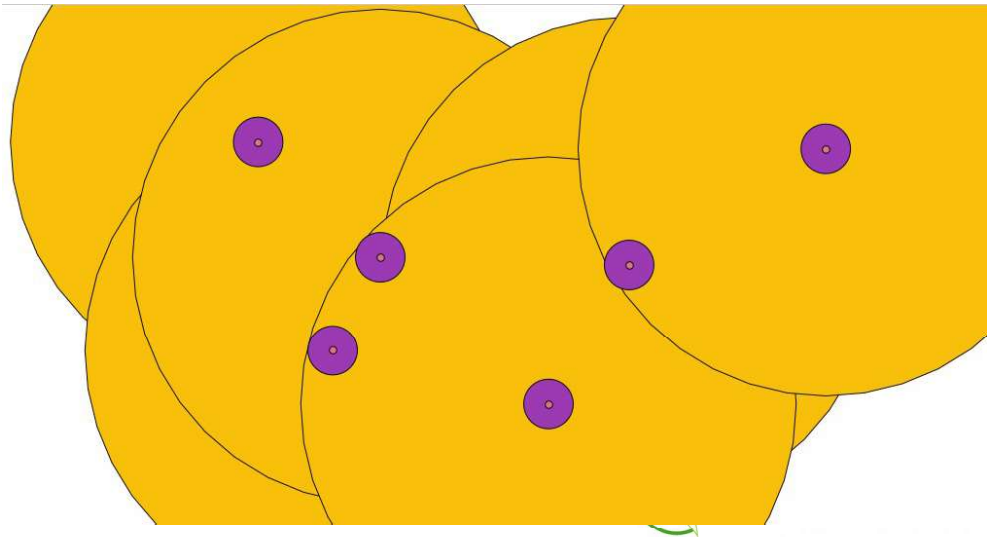
バッファ

- 点から10のバッファーが生成
  - 平面直角だから10m。緯度経度では”度”なので注意
  - 距離を100にすれば100になる

10mのバッファ

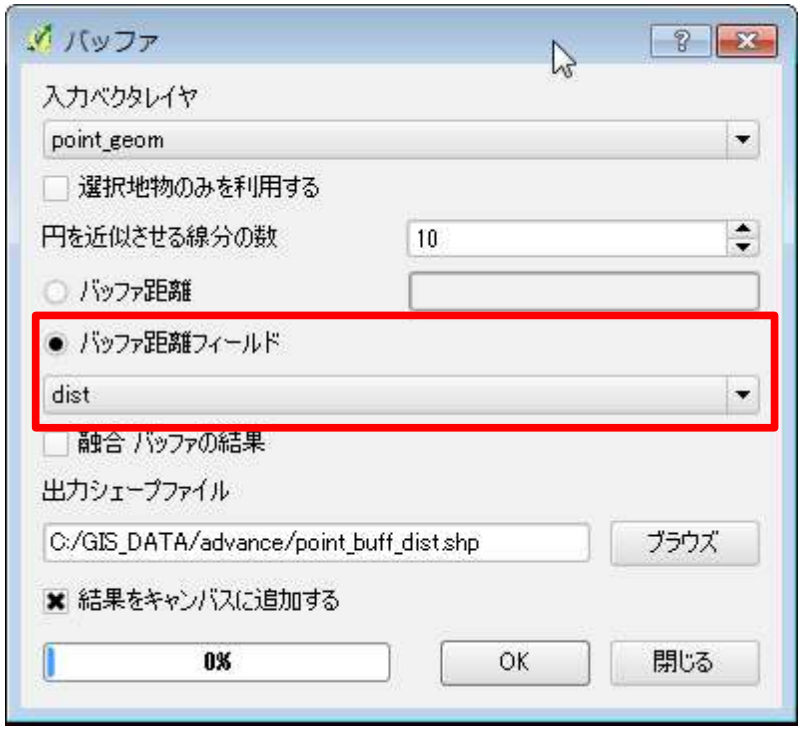


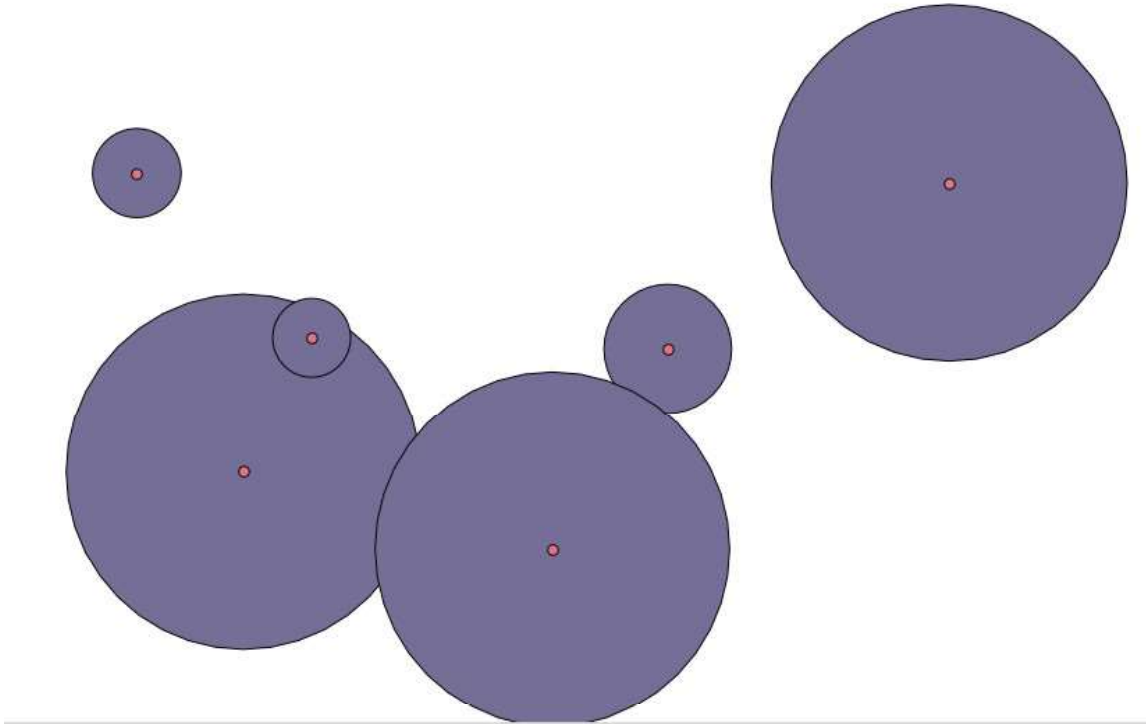
100mのバッファ



バッファ

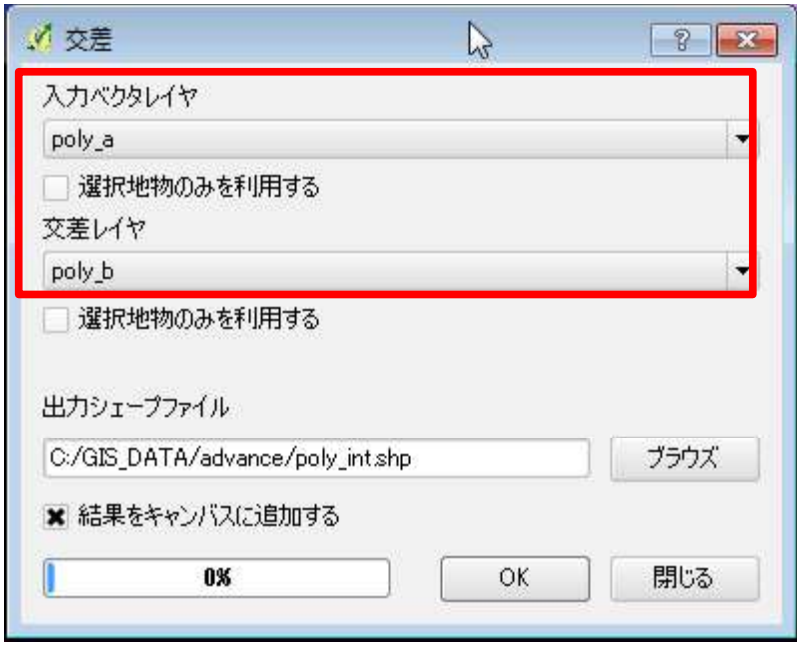
- フィールド値を利用したバッファの生成も可能
  - distを使用
    - データは数値型であること

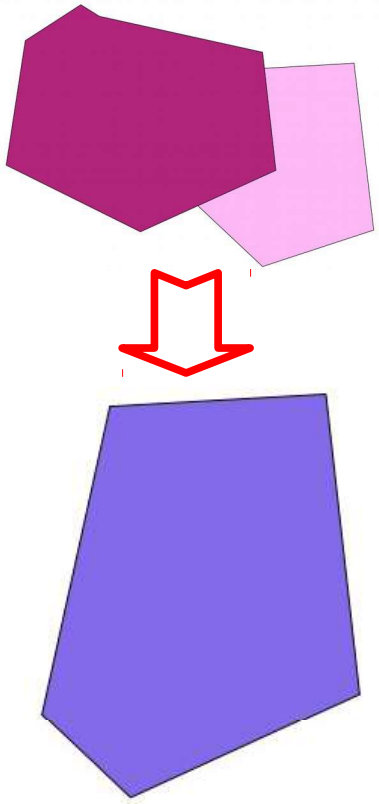




交差（Intersect）

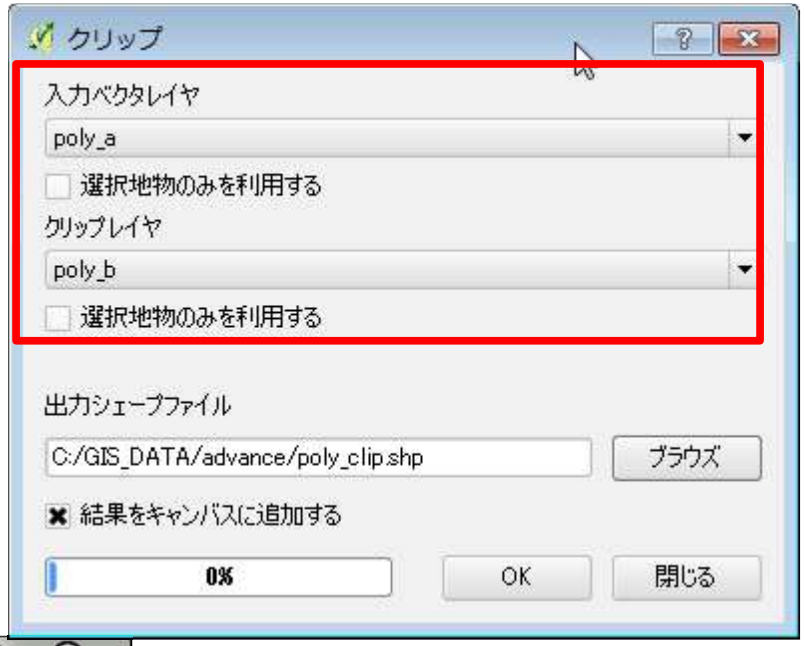
- 2つのポリゴンの重なった部分を抽出
  - poly_a，poly_bを追加
  - 入力ベクタレイヤにpoly_a，交差レイヤにpoly_b，出力をpoly_int.shp
  - 重なった部分が出力

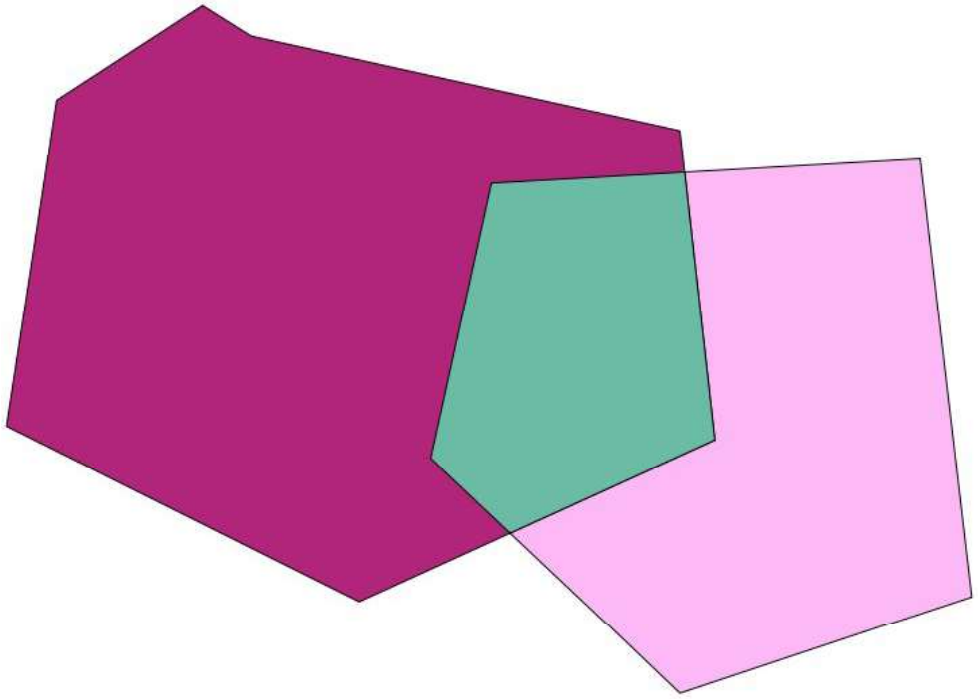




クリップ

- 1つのポリゴンで，もう1つのポリゴンを切り抜く
  - 入力ベクタレイヤにpoly_a，クリップレイヤにpoly_b，出力をpoly_clip.shp
  - 重なった部分が出力





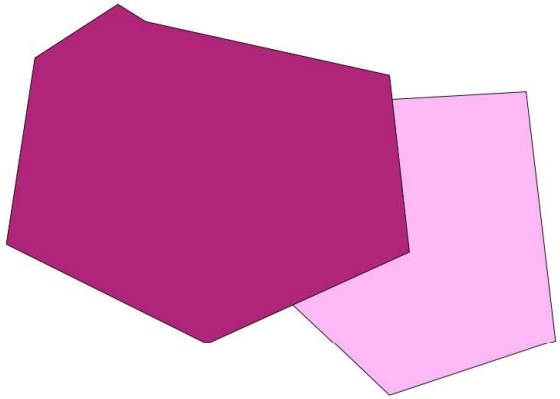
交差とクリップの違い

- 生成されたベクタの属性が違う
  - 交差は両方の属性
  - クリップは片方の属性のみ
    - ”入力ベクタレイヤ”で選択されたレイヤの属性

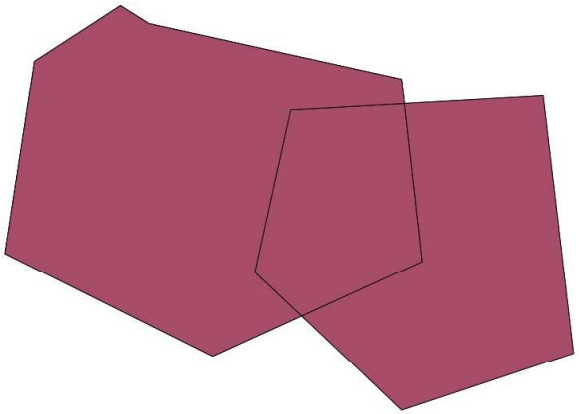
交差

クリップ

その他の演算

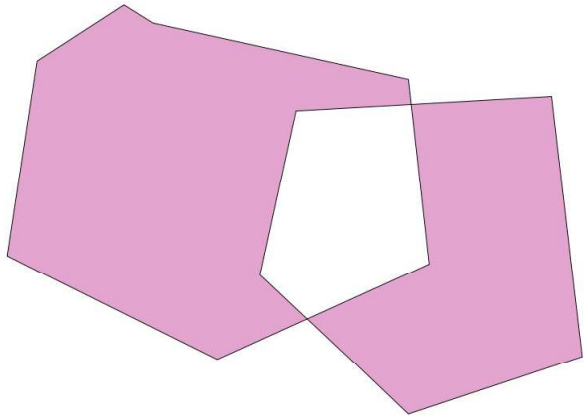


統合



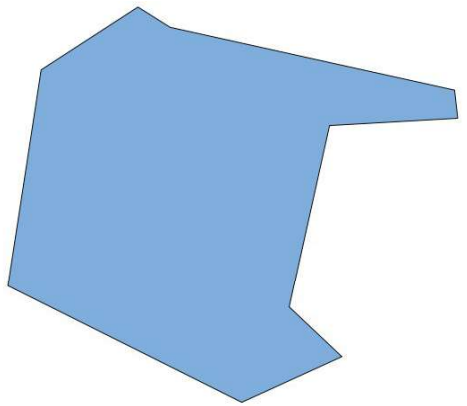
両方残る

対象差分



重なってない部分
が両方残る

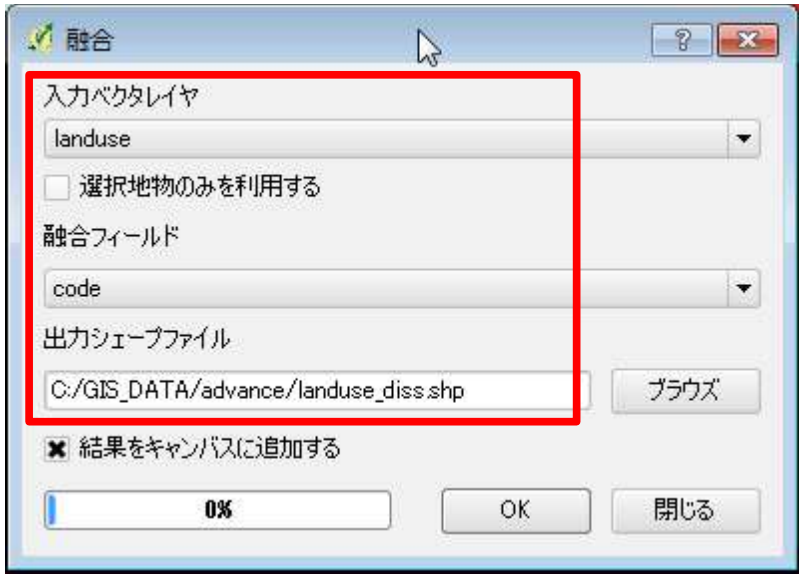
差分



重なってない部分
が片方残る

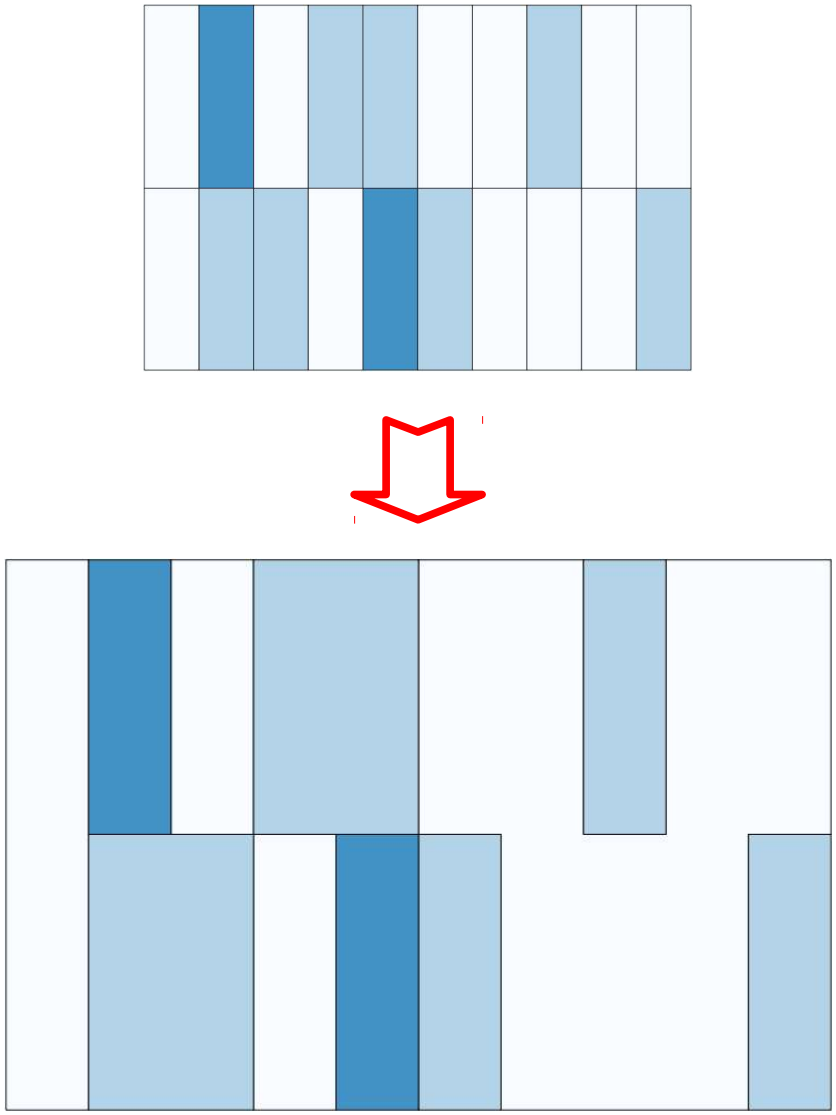
融合（ディゾルブ）

- 同じ属性を持つ場合，境界を消去
  - ポリゴンに利用
  - landuse.shpを追加
  - 入力ベクタレイヤにlanduse，融合フィールドにcode，出力にlanduse_diss.shpを設定



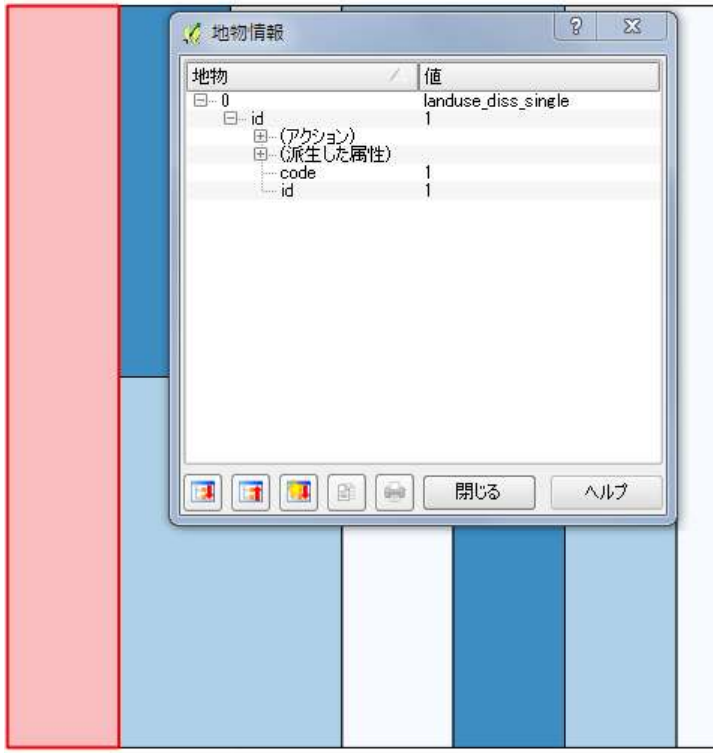
融合（ディゾルブ）

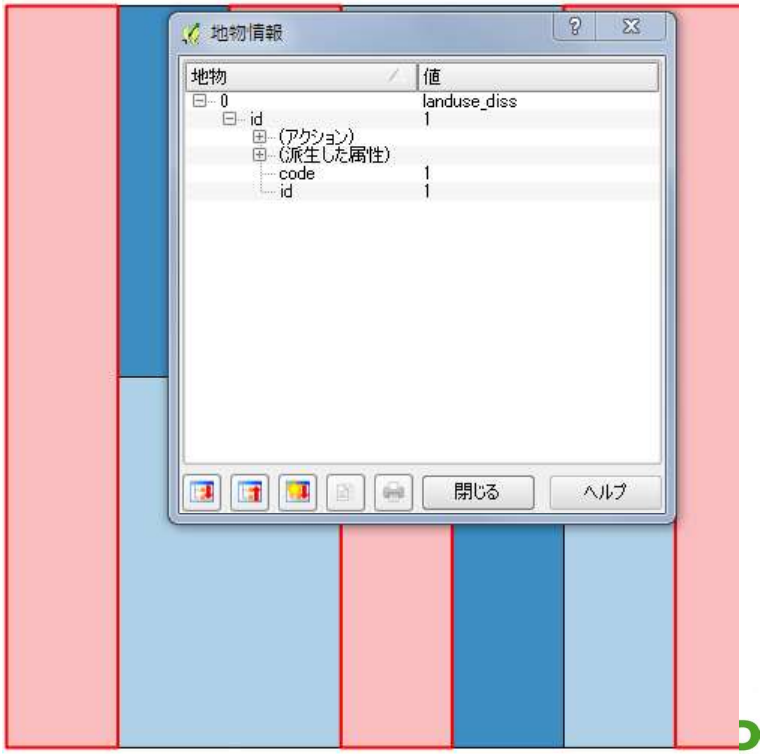
- 境界が消去される
  - 生成されるのはマルチパートポリゴン
  - 必要があればシングルパートポリゴンに変更



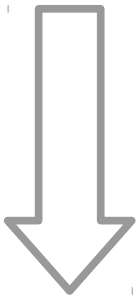
シングルパートとマルチパート

- シングルパート
  - １つの地物が1つの図形から構成される
- マルチパート
  - １つの地物が複数の図形から構成される

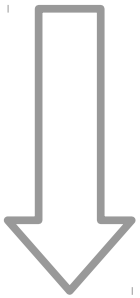




ベクタデータの分析機能



ベクタデータの活用



ラスタデータの高度な表示

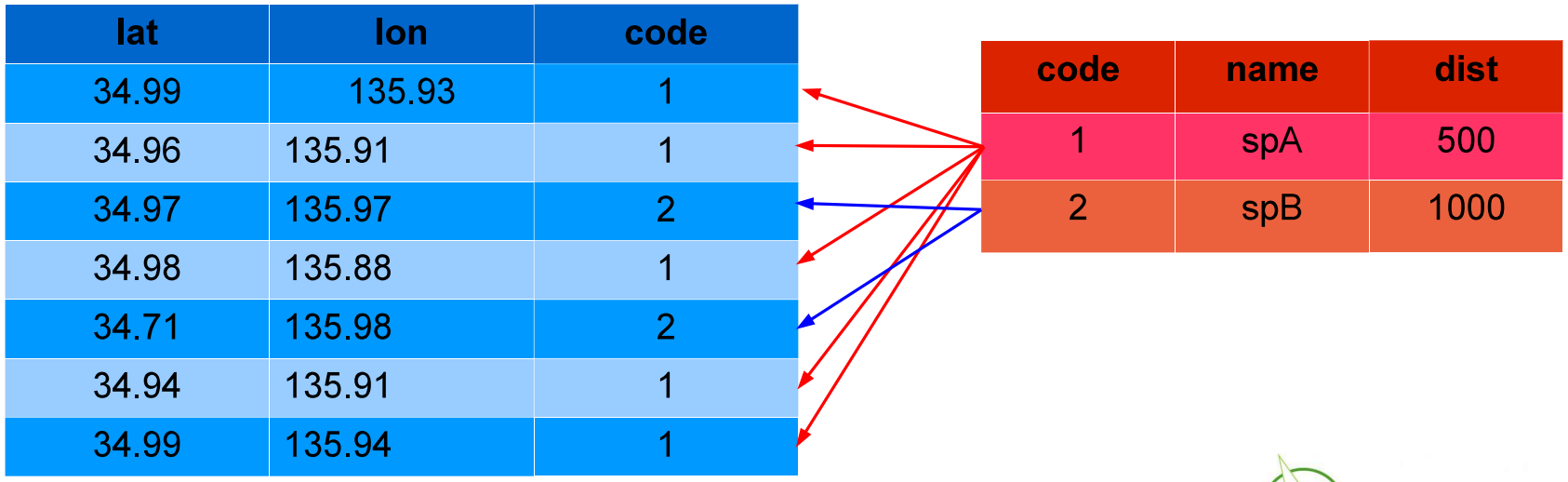
ベクタデータの活用

- ベクタデータは属性テーブルに基づく処理が可能
  - 属性結合
  - データ型変換
  - 数値に基づく表示
- 測地系の変換
  - データの測地系を変換する

テーブル結合

- ベクタレイヤのテーブルに，属性に基づき他のテーブルを加える
  - 例えば，市町村ポリゴンに人口の統計データを加えるなど

テーブル結合のイメージ

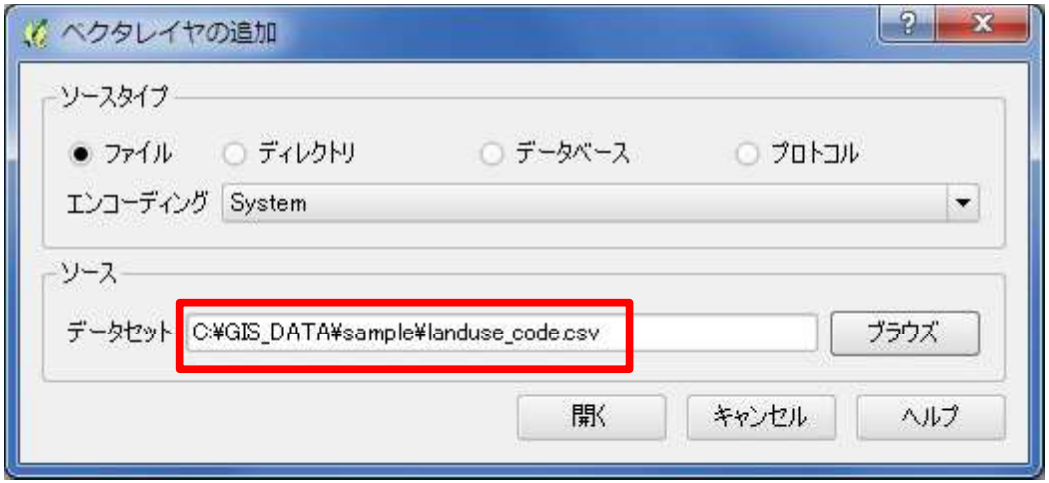


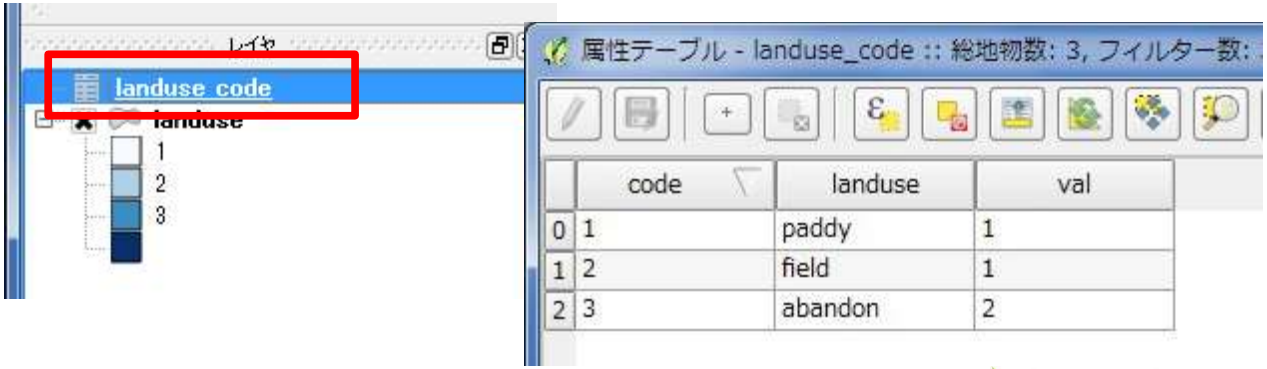
lat	lon	code
34.99	135.93	1
34.96	135.91	1
34.97	135.97	2
34.98	135.88	1
34.71	135.98	2
34.94	135.91	1
34.99	135.94	1

code	name	dist
1	spA	500
2	spB	1000

テーブル結合

- ベクタレイヤの追加で”landuse_code.csv”を開く
- レイヤに”landuse_code”が追加される



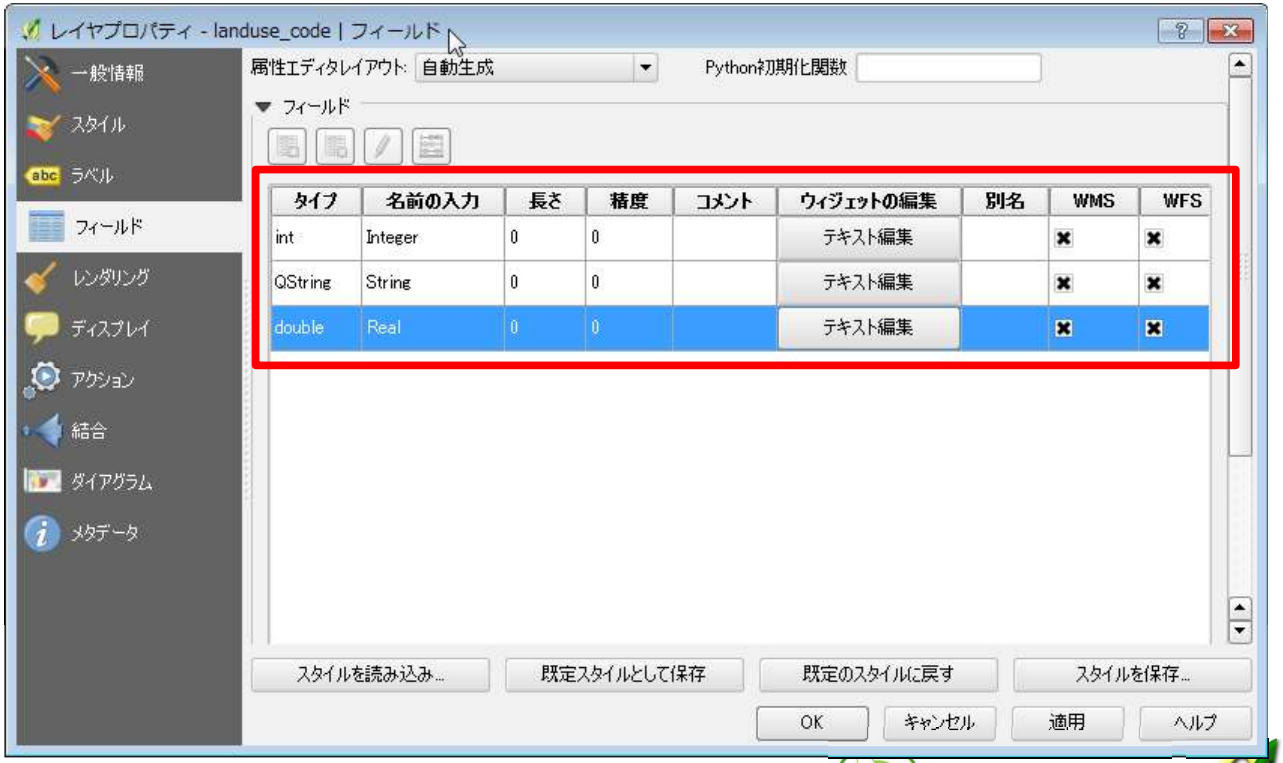


csvtファイル

- csvファイルのデータは文字列として認識される
- ファイル名が同じで拡張子を”csvt”としたファイルを作ると，データ型を定義できる
  - 文字形 → String
  - 整数型 → Integer
  - 実数型 → Real

csvtファイル

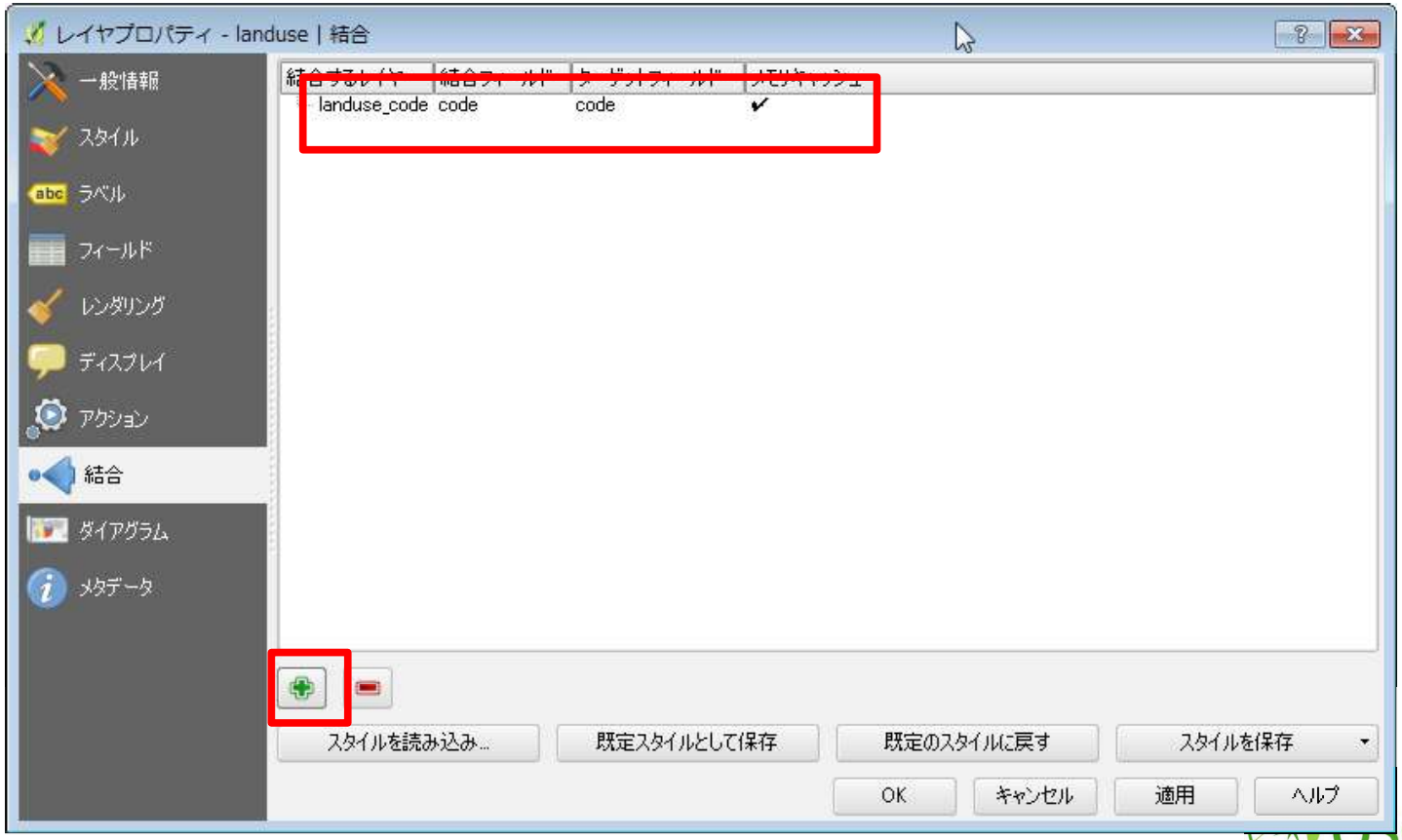
- landuse_code.csvtを開いてみる
  - カンマ区切りのテキスト
- QGISのフィールドでも定義されている



データ結合

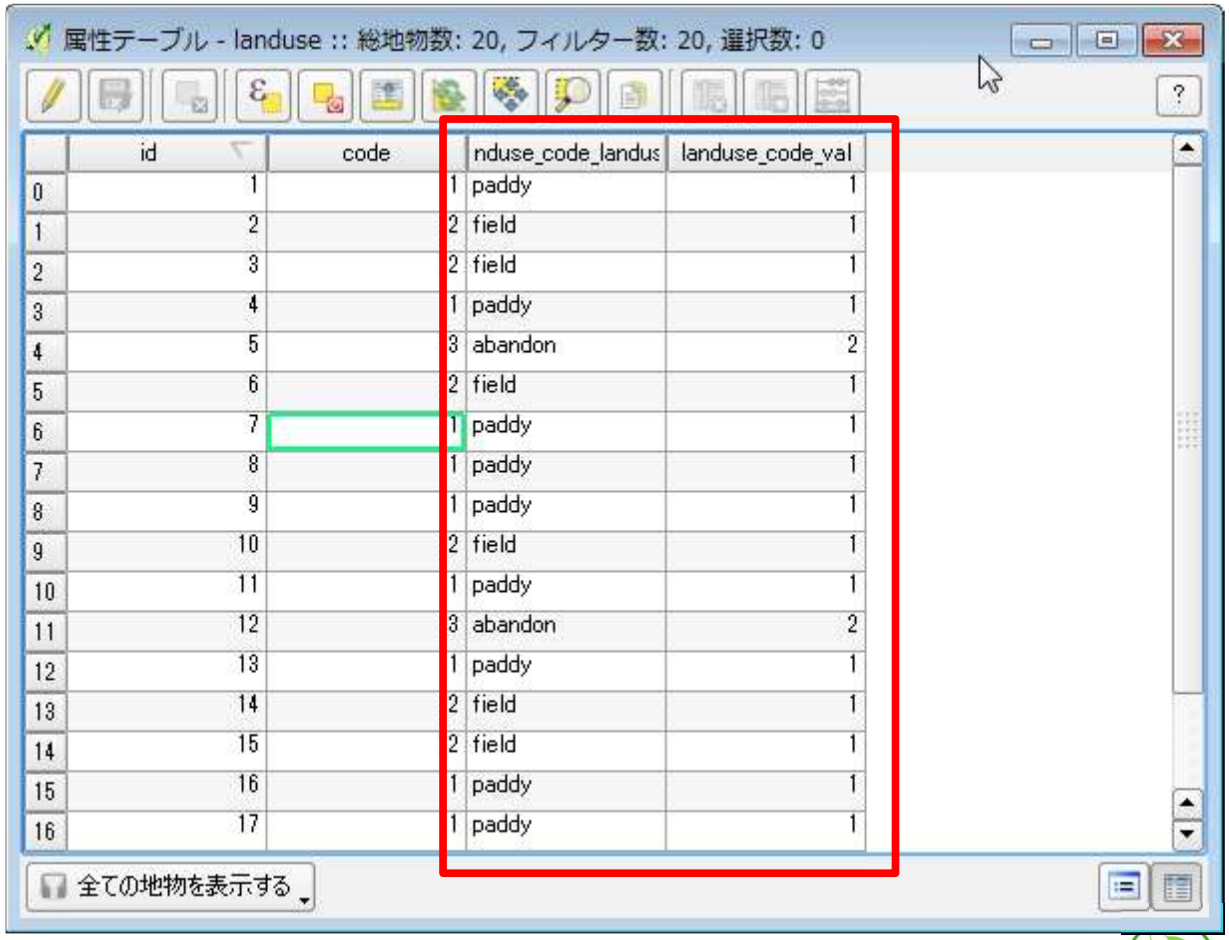
- “landuse”のプロパティを開き結合を選択
- 「＋」ボタンをクリック
  - “結合するレイヤ”に”landuse_code”
  - “結合フィールド”に”code”
    - csvファイルの属性
  - “ターゲットフィールド”に"code”
    - shpファイルの属性


ベクタ結合の追加
結合するレイヤ
landuse_code
結合フィールド
code
ターゲットフィールド
code
結合レイヤをバーチャルメモリにキャッシュする
結合フィールドに属性インデックスを作成する
OK
キャンセル


レイヤプロパティ - landuse | 結合
一般情報
スタイル
ラベル
フィールド
レンダリング
ディスプレイ
アクション
結合
ダイアグラム
メタデータ
landuse_code
code
code
スタイルを読み込み...
既定スタイルとして保存
既定のスタイルに戻す
スタイルを保存
OK
キャンセル
適用
ヘルプ

テーブル結合

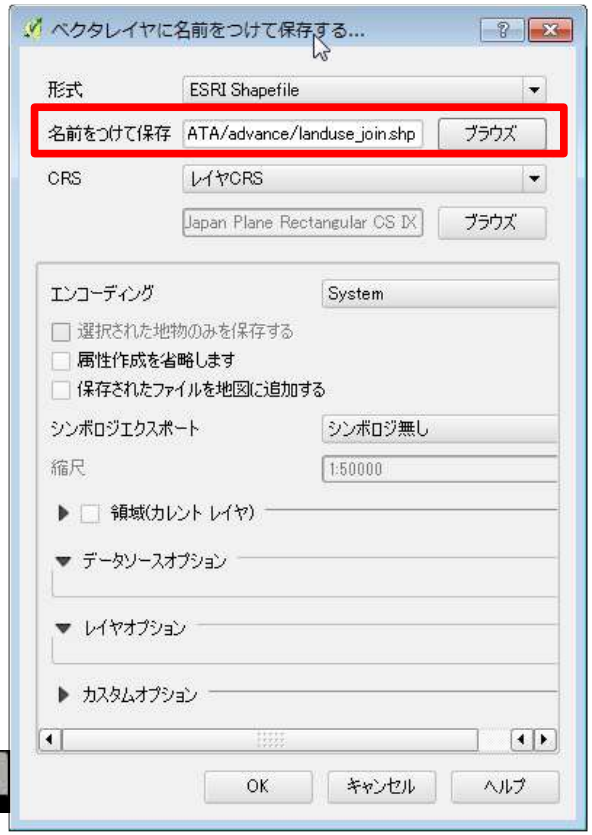
- 属性テーブルを確認すると，landuseとval列が追加されている

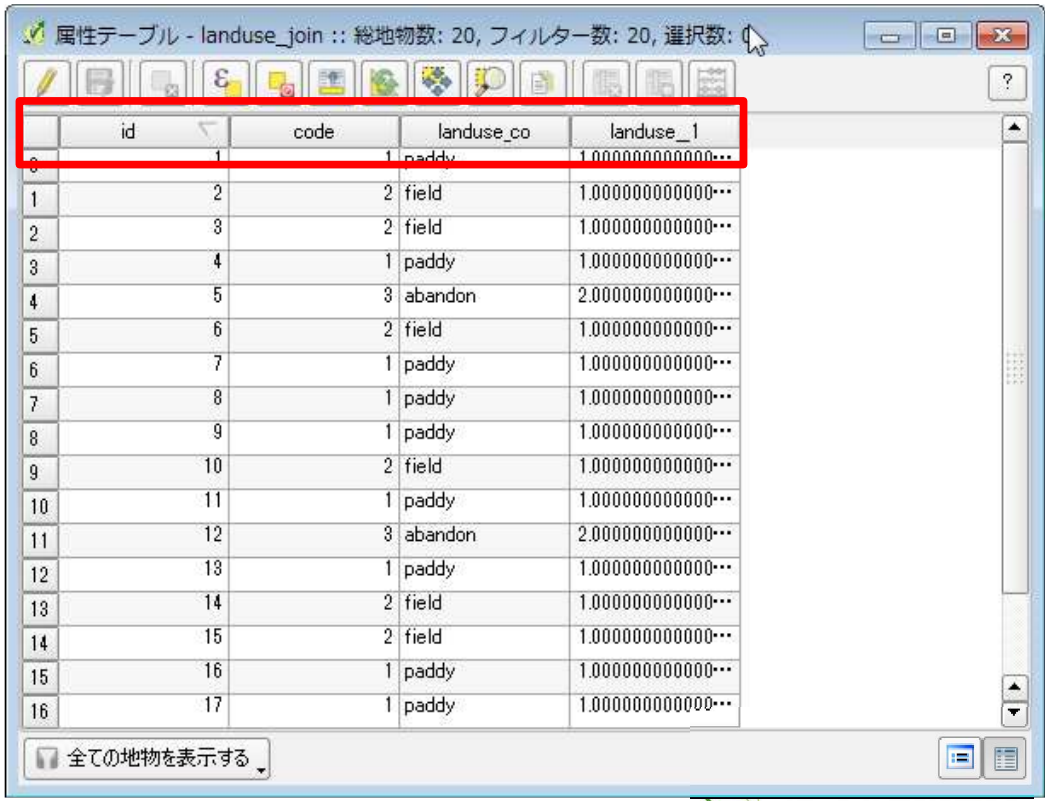


	id	code	nduse_code_landus	landuse_code_val
0	1		1 paddy	1
1	2		2 field	1
2	3		2 field	1
3	4		1 paddy	1
4	5		3 abandon	2
5	6		2 field	1
6	7		1 paddy	1
7	8		1 paddy	1
8	9		1 paddy	1
9	10		2 field	1
10	11		1 paddy	1
11	12		3 abandon	2
12	13		1 paddy	1
13	14		2 field	1
14	15		2 field	1
15	16		1 paddy	1
16	17		1 paddy	1

テーブル結合

- 一時的な結合なので，保持する場合は保存する
  - レイヤの上で右クリック
  - 名前をつけて保存で新しく保存
    - データを開いて確認

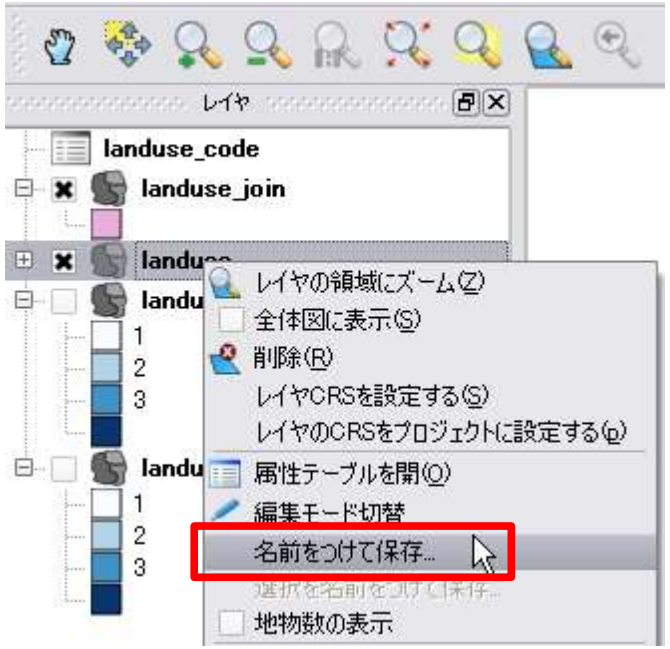


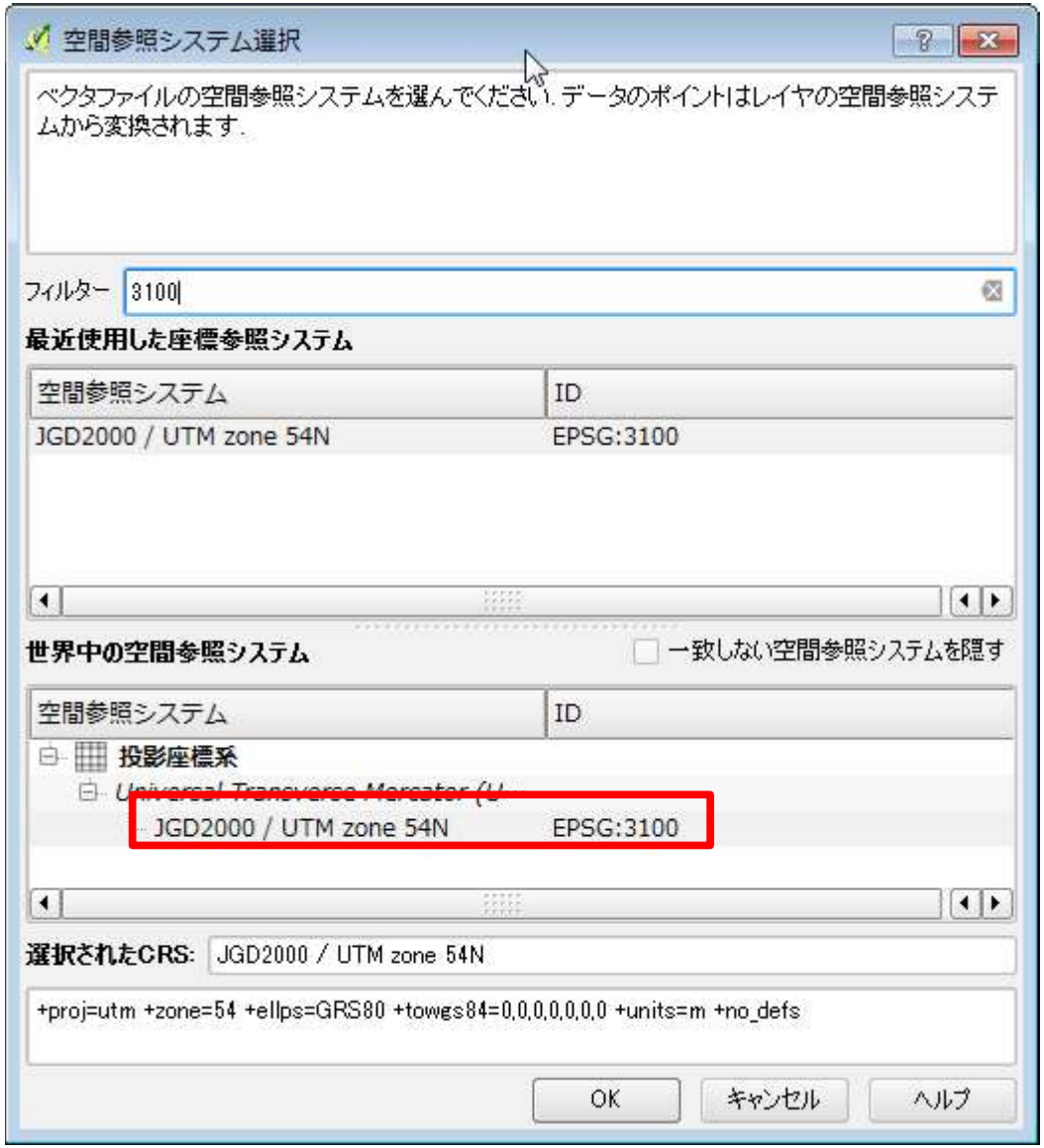


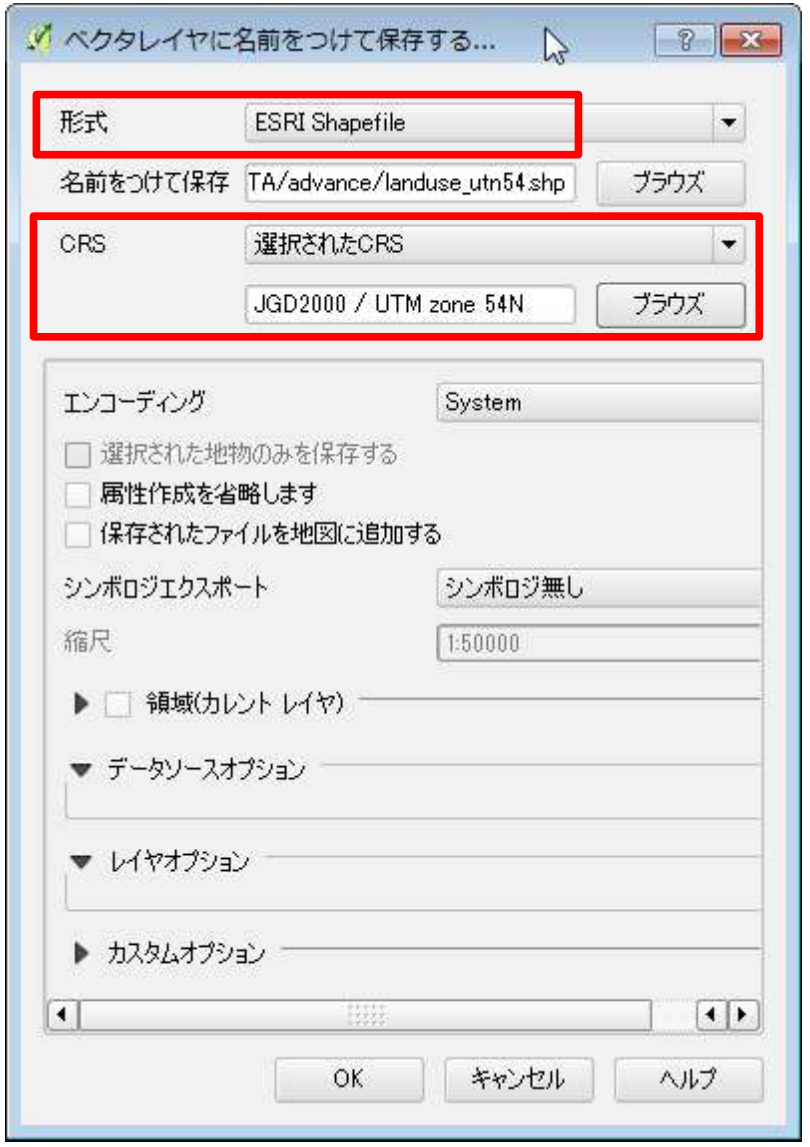
	id	code	landuse_co	landuse_1
0	1	1	paddy	1.000000000000…
1	2	2	field	1.000000000000…
2	3	2	field	1.000000000000…
3	4	1	paddy	1.000000000000…
4	5	3	abandon	2.000000000000…
5	6	2	field	1.000000000000…
6	7	1	paddy	1.000000000000…
7	8	1	paddy	1.000000000000…
8	9	1	paddy	1.000000000000…
9	10	2	field	1.000000000000…
10	11	1	paddy	1.000000000000…
11	12	3	abandon	2.000000000000…
12	13	1	paddy	1.000000000000…
13	14	2	field	1.000000000000…
14	15	2	field	1.000000000000…
15	16	1	paddy	1.000000000000…
16	17	1	paddy	1.000000000000…

ベクタデータの座標変換

- ”名前をつけて保存”の時に座標変換も可能
  - “landuse”で名前をつけて保存を選択
  - CRSで”選択されたCRS”を選び，ブラウズから”JGD2000 / UTM zone 54N”を選択
    - EPSG:3100

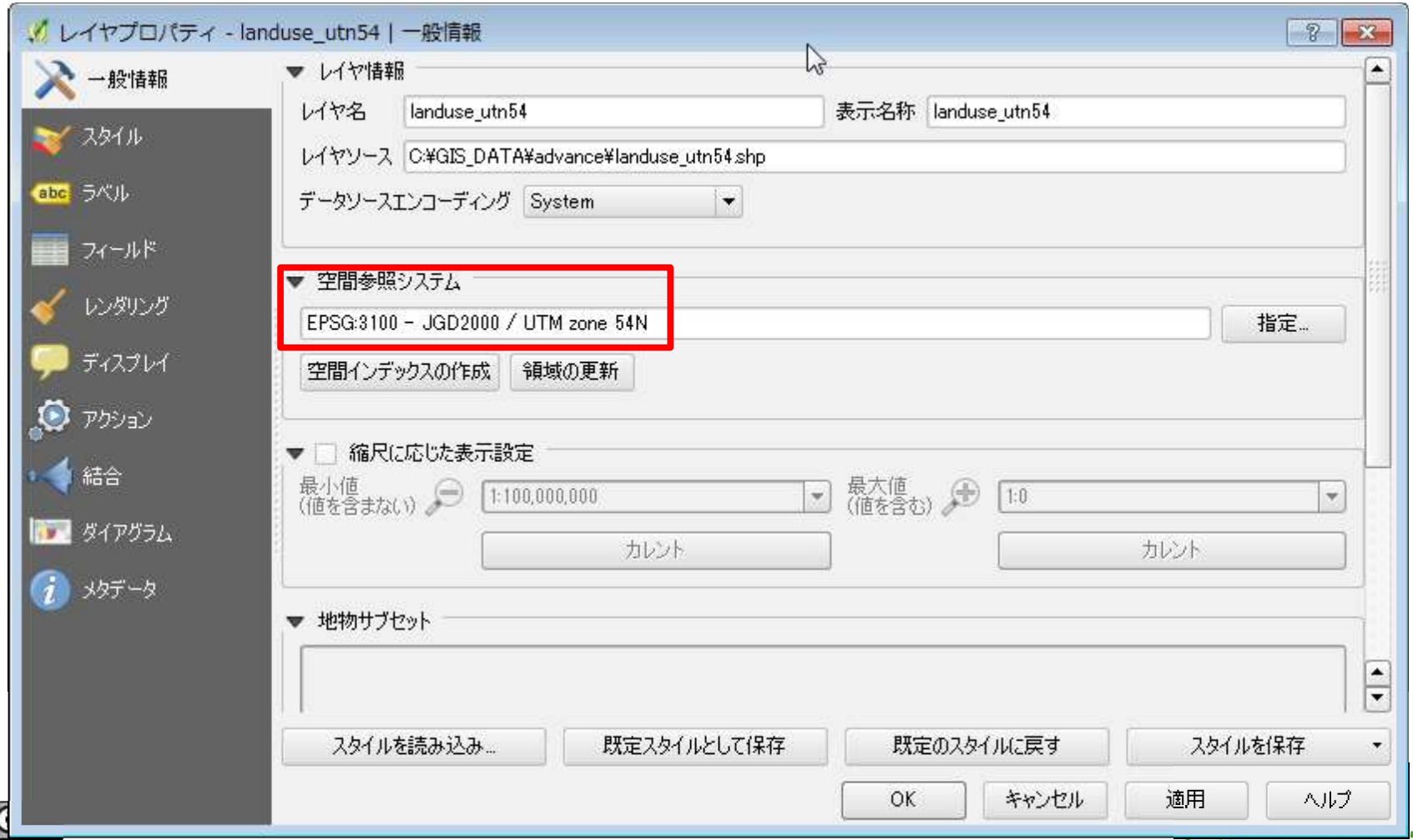



空間参照システム選択
ベクタファイルの空間参照システムを選んでください. データのポイントはレイヤの空間参照システムから変換されます.
フィルター
3100
最近使用した座標参照システム
空間参照システム
ID
JGD2000 / UTM zone 54N
EPSG:3100
世界中の空間参照システム
一致しない空間参照システムを隠す
空間参照システム
ID
投影座標系
JGD2000 / UTM zone 54N
EPSG:3100
選択されたCRS:
JGD2000 / UTM zone 54N
+proj=utm +zone=54 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
OK
キャンセル
ヘルプ


ベクタレイヤに名前をつけて保存する...
形式
ESRI Shapefile
名前をつけて保存
TA/advance/landuse_utn54.shp
ブラウズ
CRS
選択されたCRS
JGD2000 / UTM zone 54N
ブラウズ
エンコーディング
System
選択された地物のみを保存する
属性作成を省略します
保存されたファイルを地図に追加する
シンボロジエクスポート
シンボロジ無し
縮尺
1:50000
領域(カレント レイヤ)
データソースオプション
レイヤオプション
カスタムオプション
OK
キャンセル
ヘルプ

ベクタデータの座標変換

- プロパティの一般情報で，測地系が変更されていることを確認



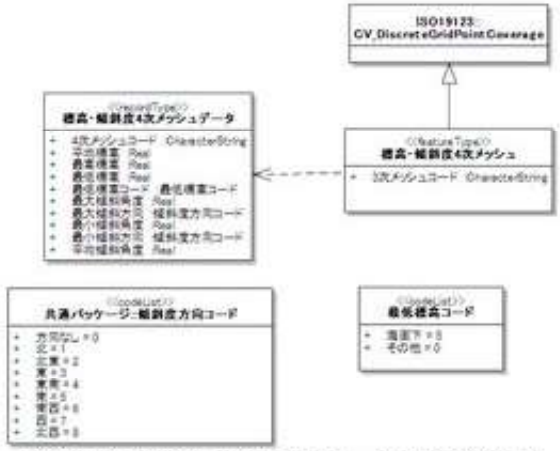
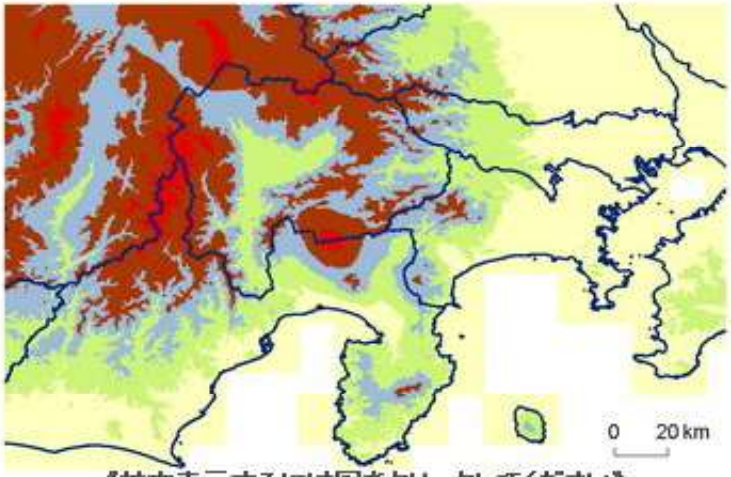
データ型の変更

- shpファイルは必要なデータ型でない場合がある
  - 数字なのに文字列型になっている
- テーブル処理でデータ型を変更することができる
  - 国土数値情報の4次メッシュ標高値を例に説明
- ELEV500_mesh.shpを開く

国土数値情報 標高・傾斜度4次メッシュデータ です。

■最新のデータは製品仕様書第1.0版に基づいています。 （データ作成年度:平成23年度）

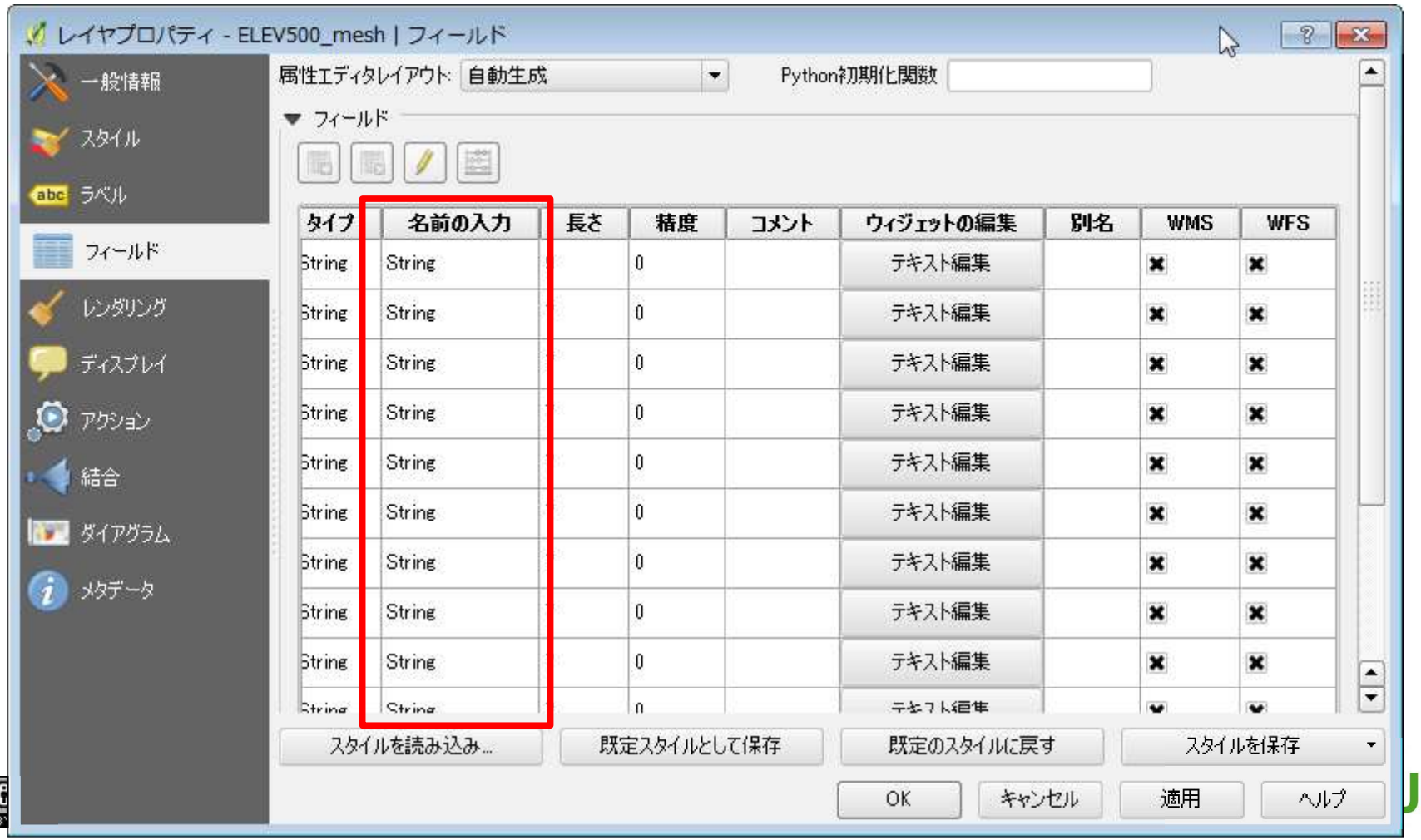
標高傾斜度4次メッシュ 第1.0版		識別子	G04-c
内容	標高(平均、最高、最低)、最大傾斜角度・方角、最小傾斜角度・方角について4次メッシュ(500mメッシュ)毎に整備したものである。		
関連する法律	－		
データ作成年度	平成23年度		
原典資料	基盤地図情報数値標高モデル 10m メッシュ(国土地理院)		
作成方法	4次メッシュ内に含まれる標高傾斜度5次メッシュの標高値から平均標高・最大標高・最小標高を算出した。傾斜角度、傾斜方向についても、標高傾斜度5次メッシュの標高を使い傾斜角度(最大・最小・平均)及び傾斜方向(最大・最小)を算出した。		
座標系	JGD2000, TP / (B, L), H		
データ形状	メッシュ		

データ構造	イメージ
	
《拡大表示するには図をクリックしてください》	《拡大表示するには図をクリックしてください》

地物情報	地物名	説明	
	標高傾斜度4次メッシュ	標高(平均，最高，最低)，最大傾斜角度・方角，最小傾斜角度・方角について4次メッシュ(500mメッシュ)毎に整備したデータ。	
	属性名 (かっこ内はshp属性名)	説明	属性の型
	4次メッシュコード (G04c_001)	標準地域メッシュの4次メッシュ	文字列型
	平均標高 (G04c_002)	10mメッシュの標高値から算出する平均標高(m)	実数型 ※整備データが存在しない場合、“unkown”とする。
			実数型

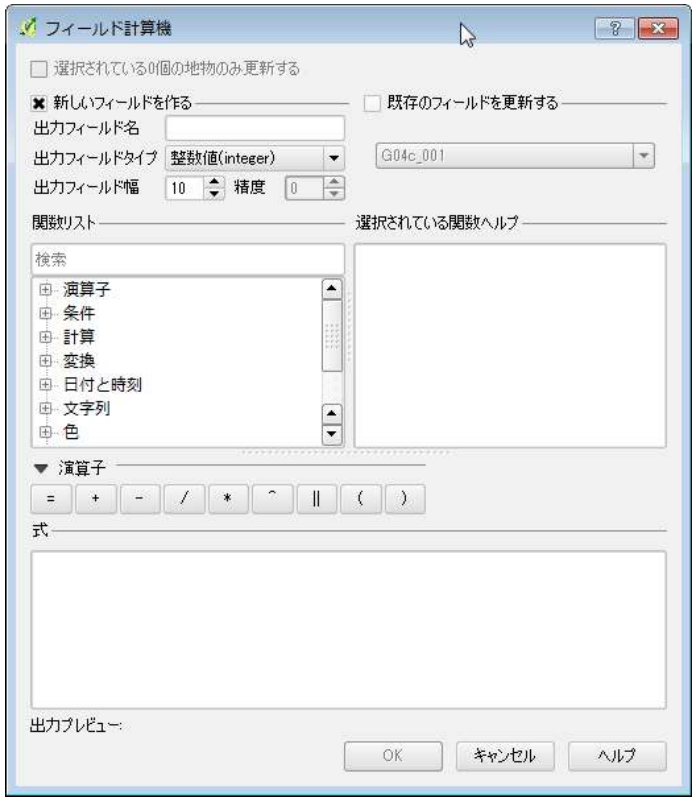
データ型の変更

- プロパティでデータ型を確認
  - 文字列になっている



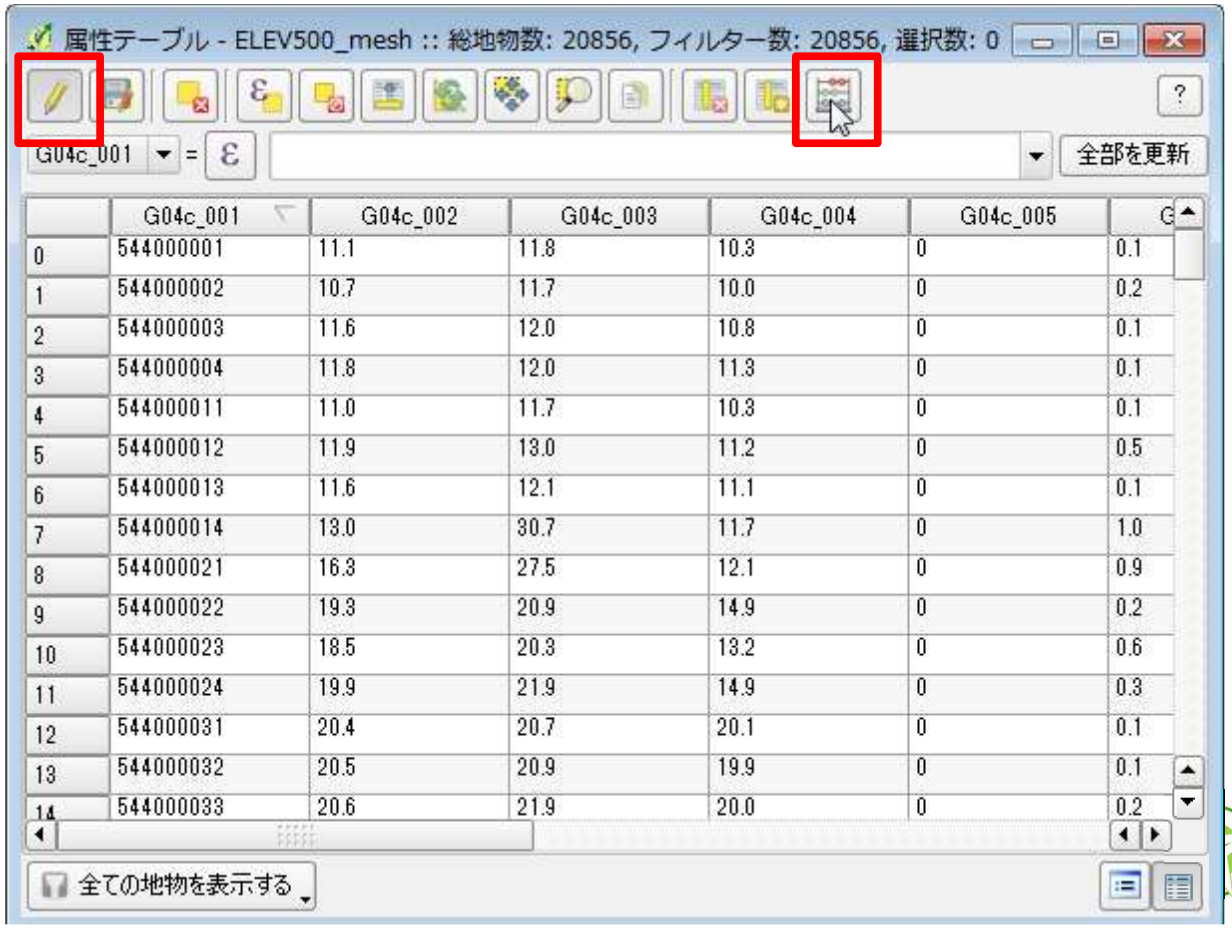
データ型の変更

- 既存列のデータ型を変えることはできない
  - データベースなので
- 新しく列を作り変換した値を入れる
  - フィールド計算機を使う



データ型変換

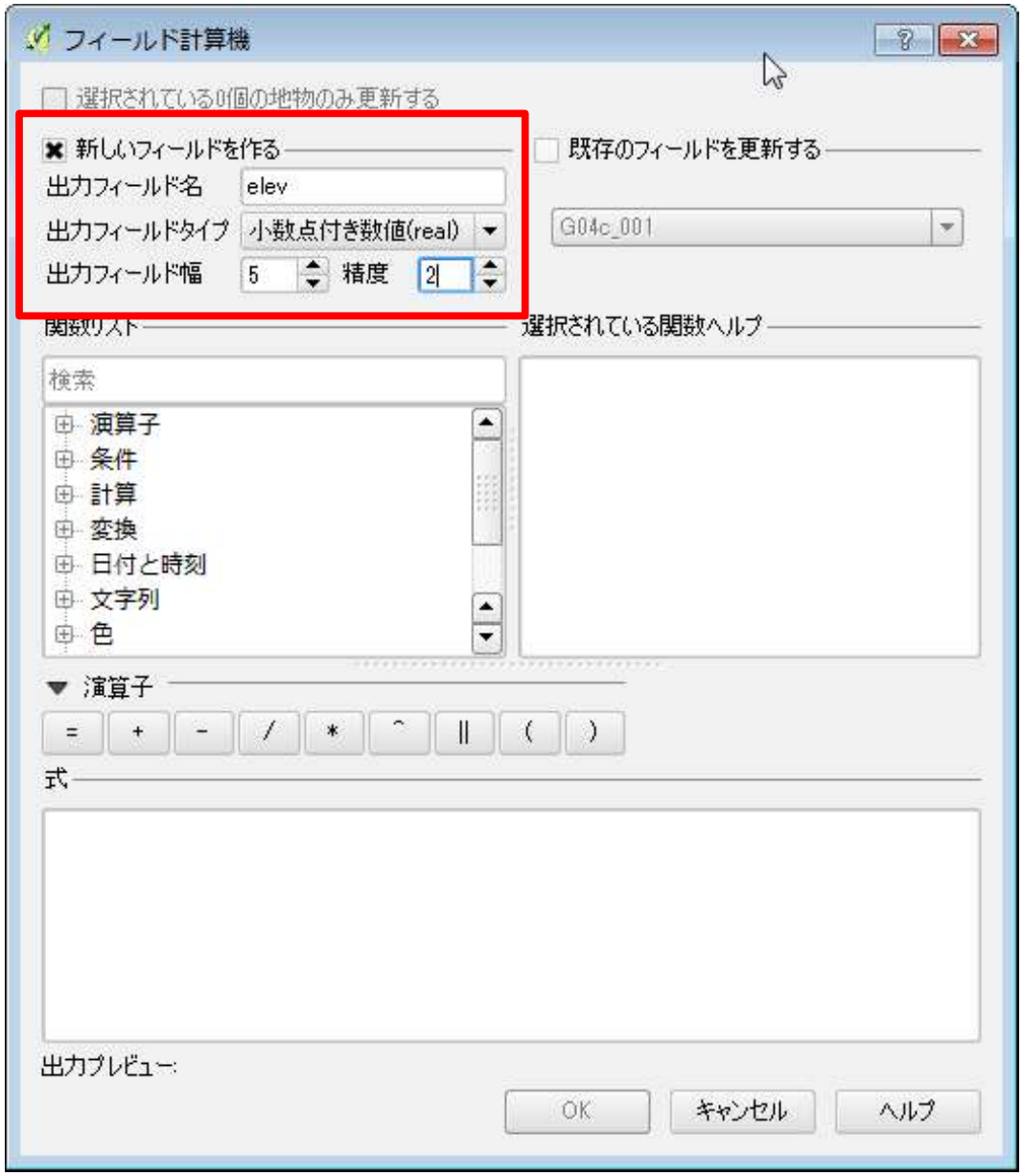
- 属性テーブルを開く
  - “編集モード”の切り替えをクリック
  - “フィールド計算機”をクリック



	G04c_001	G04c_002	G04c_003	G04c_004	G04c_005	G
0	544000001	11.1	11.8	10.3	0	0.1
1	544000002	10.7	11.7	10.0	0	0.2
2	544000003	11.6	12.0	10.8	0	0.1
3	544000004	11.8	12.0	11.3	0	0.1
4	544000011	11.0	11.7	10.3	0	0.1
5	544000012	11.9	13.0	11.2	0	0.5
6	544000013	11.6	12.1	11.1	0	0.1
7	544000014	13.0	30.7	11.7	0	1.0
8	544000021	16.3	27.5	12.1	0	0.9
9	544000022	19.3	20.9	14.9	0	0.2
10	544000023	18.5	20.3	13.2	0	0.6
11	544000024	19.9	21.9	14.9	0	0.3
12	544000031	20.4	20.7	20.1	0	0.1
13	544000032	20.5	20.9	19.9	0	0.1
14	544000033	20.6	21.9	20.0	0	0.2

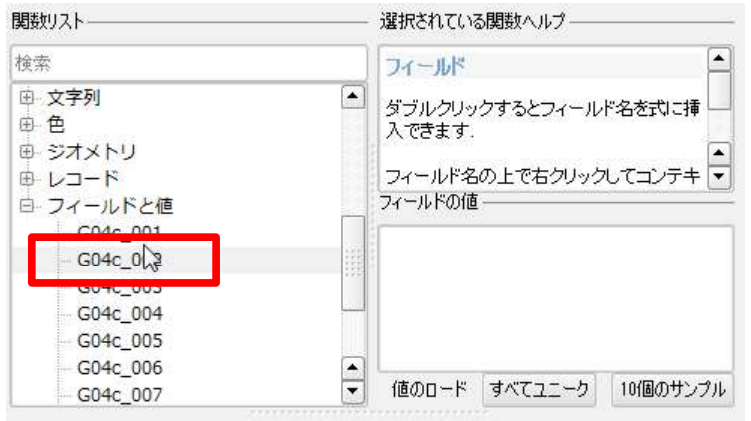
データ型変換

- フィールド計算機で
  - “新しいフィールド”を作るにチェック
  - 出力フィールド名に”elev”
  - 出力フィールドタイプで”小数点付き数値（real）”
  - 出力フィールド幅を5，精度を2とする
    - 全体のデータが5桁，小数点以下が2桁ということ



データ型変換

- 関数リストで以下を実施
  - ”変換”から”toreal”をダブルクリック
  - “フィールドと値”から”G04c_002”をダブルクリック
- 下の「式」のかっこを閉じる
  - 「）」を入力



データ型変換

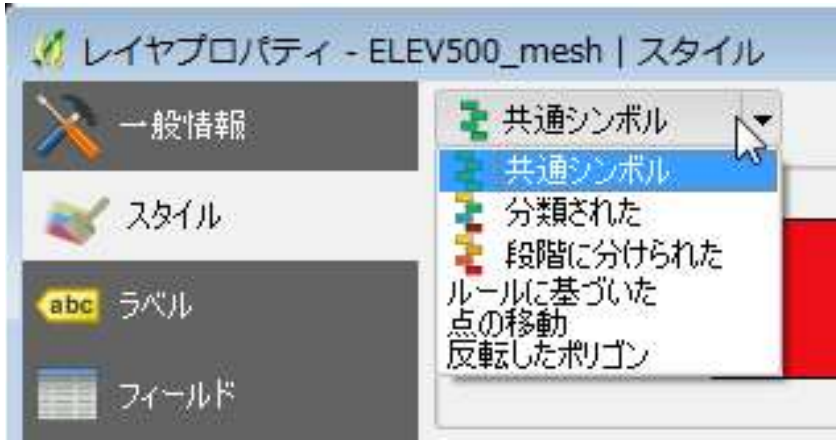
- OKをクリックして，「elev」列を確認
  - 数字が右寄せであれば整数もしくは実数型
- 「編集を保存する」をクリックし，「編集モードの切り替え」をクリックする

データ型変換

- レイヤプロパティでelev列が追加されているのを確認

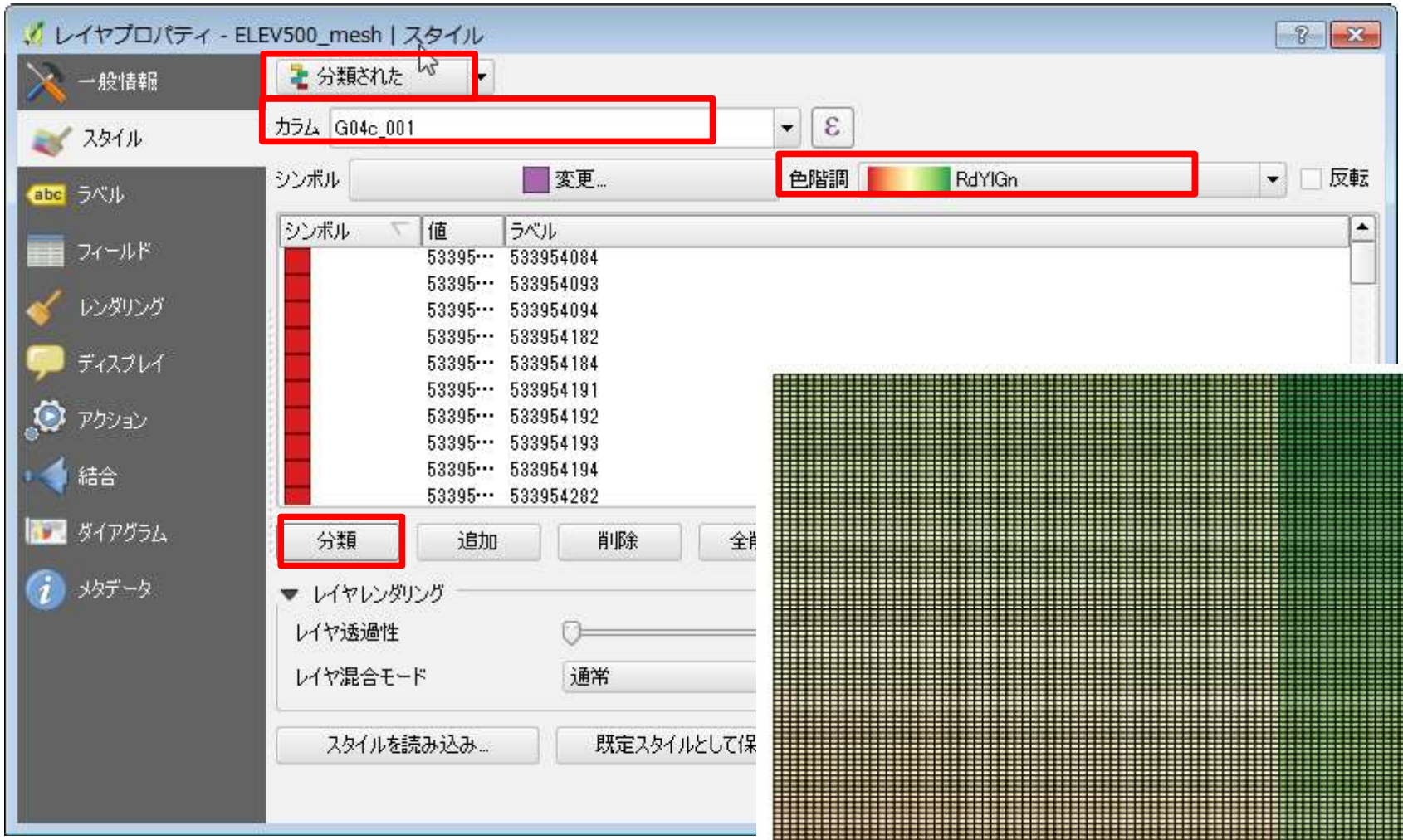
ポリゴンの表示

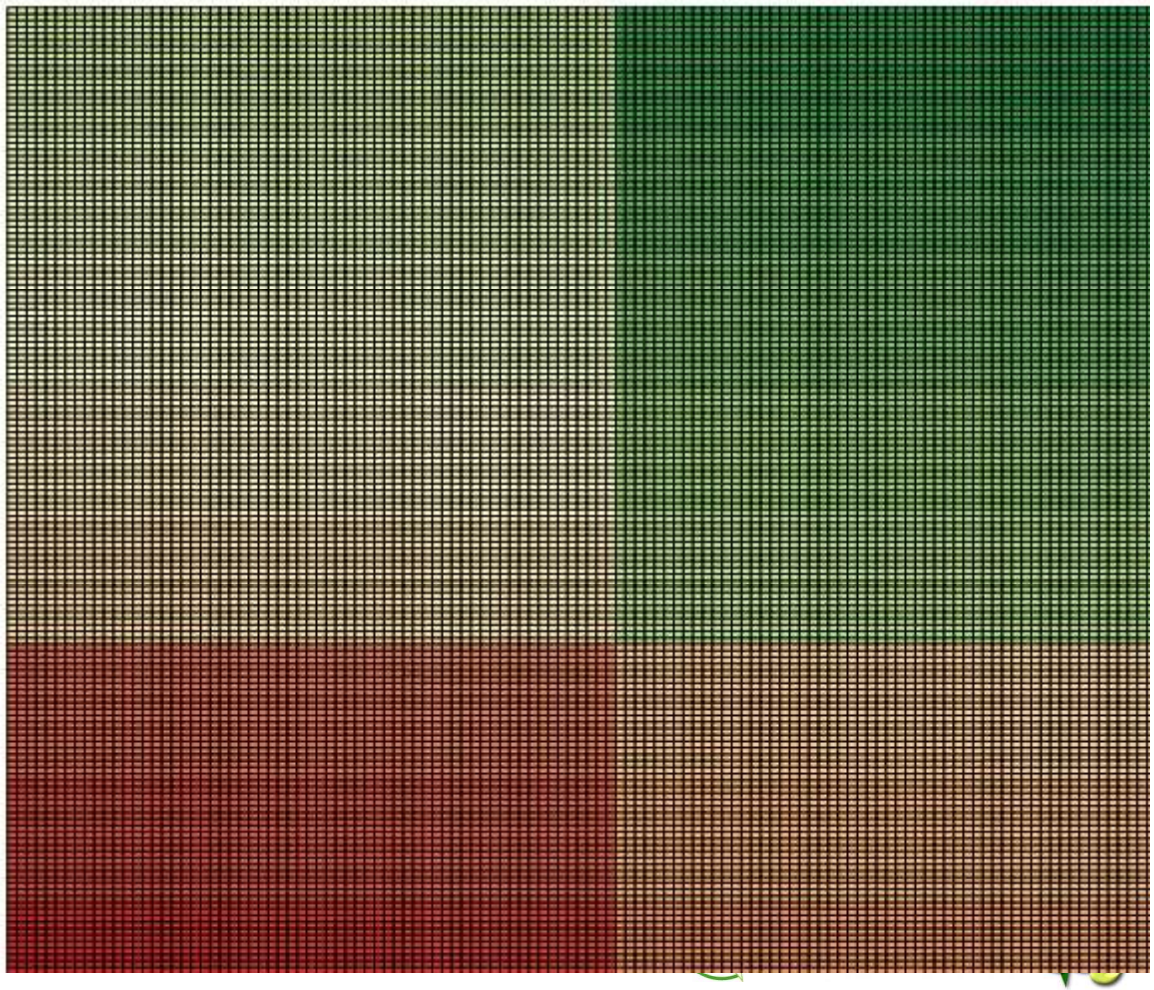
- ポリゴンの表示は主に3つの方法
  - 共通シンボル
    - 全て同一色
  - 分類された
    - 個別値に基づき表示
      - 文字列，数値
  - 段階に分けられた
    - 一定の範囲毎に色を変える
      - 数値型



個別値の表示

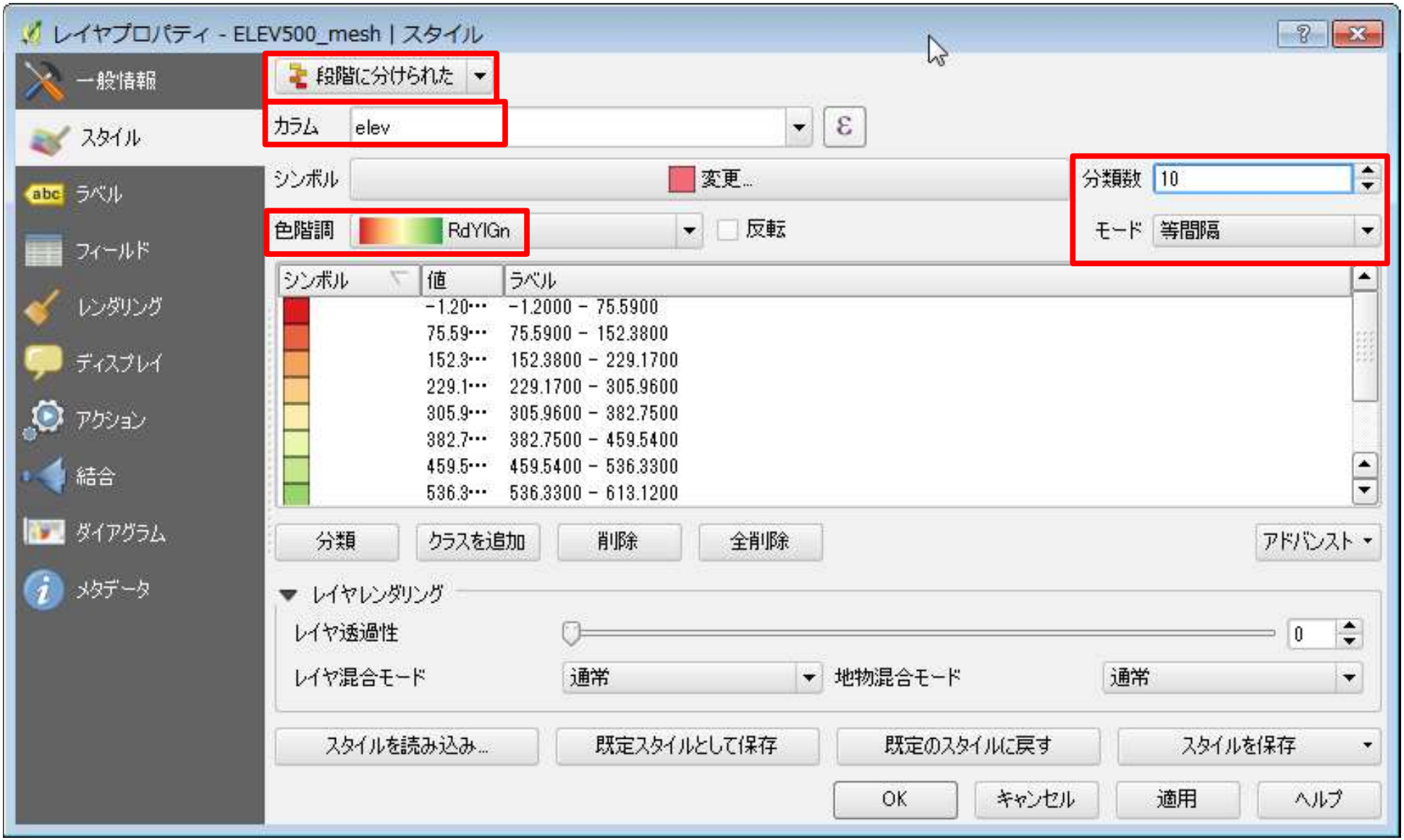
- “分類された”を選択
- カラムで”G04c_001”を選択
  - 標準地域メッシュの4次メッシュ
- 色階調で任意のものを選択
  - RdYlGnを選択
- 分類をクリック
  - 個別のコードの自動的に色が割り振られる
    - 「クラスが多いです！」と警告が出ますが「OK」をクリック
  - OKを押して確認


レイヤプロパティ - ELEV500_mesh | スタイル
一般情報
スタイル
ラベル
フィールド
レンダリング
ディスプレイ
アクション
結合
ダイアグラム
メタデータ
分類された
カラム G04c_001
ε
シンボル
変更...
色階調 RdYlGn
反転
シンボル 値 ラベル
53395… 533954084
53395… 533954093
53395… 533954094
53395… 533954182
53395… 533954184
53395… 533954191
53395… 533954192
53395… 533954193
53395… 533954194
53395… 533954282
分類
追加
削除
レイヤレンダリング
レイヤ透過性
レイヤ混合モード
通常
スタイルを読み込み...
既定スタイルとして保



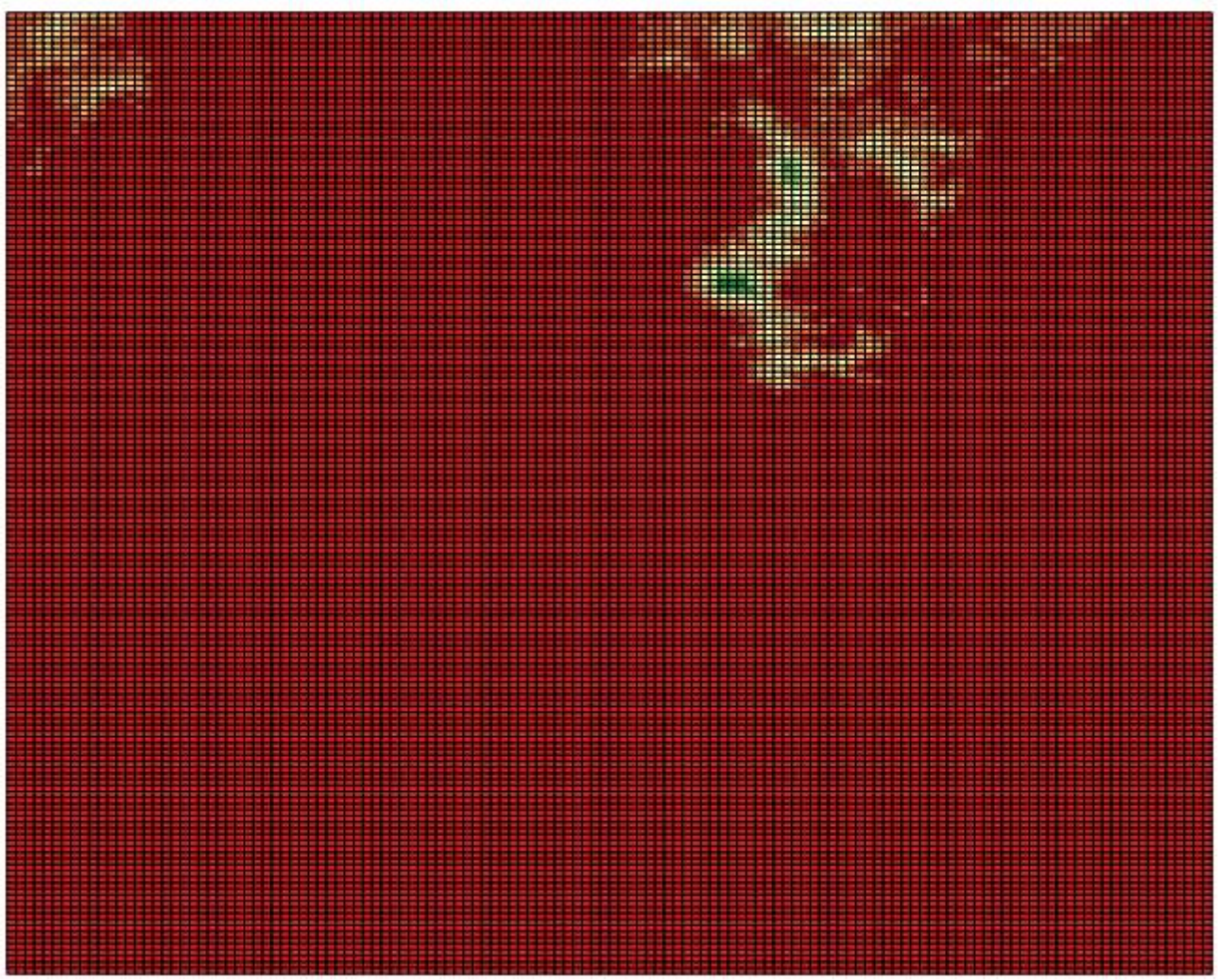
連続値(数値)の表示

- “段階に分けられた”を選択
  - カラムはelev
    - 数値型はelevしかない
  - 色階調はRdYlGn
  - 分類数は10
    - 分類数を変えると自動的に区分が変更
  - モードは「等間隔」
    - 最小値から最大値を10等分
  - OKをクリックし確認


レイヤプロパティ - ELEV500_mesh | スタイル
一般情報
スタイル
ラベル
フィールド
レンダリング
ディスプレイ
アクション
結合
ダイアグラム
メタデータ
段階に分けられた
カラム
elev
シンボル
変更...
分類数
10
色階調
RdYlGn
反転
モード
等間隔
シンボル
値
ラベル
-1.20… -1.2000 - 75.5900
75.59… 75.5900 - 152.3800
152.3… 152.3800 - 229.1700
229.1… 229.1700 - 305.9600
305.9… 305.9600 - 382.7500
382.7… 382.7500 - 459.5400
459.5… 459.5400 - 536.3300
536.3… 536.3300 - 613.1200
分類
クラスを追加
削除
全削除
アドバンスト
レイヤレンダリング
レイヤ透過性
0
レイヤ混合モード
通常
地物混合モード
通常
スタイルを読み込み...
既定スタイルとして保存
既定のスタイルに戻す
スタイルを保存
OK
キャンセル
適用
ヘルプ

連続値の表示

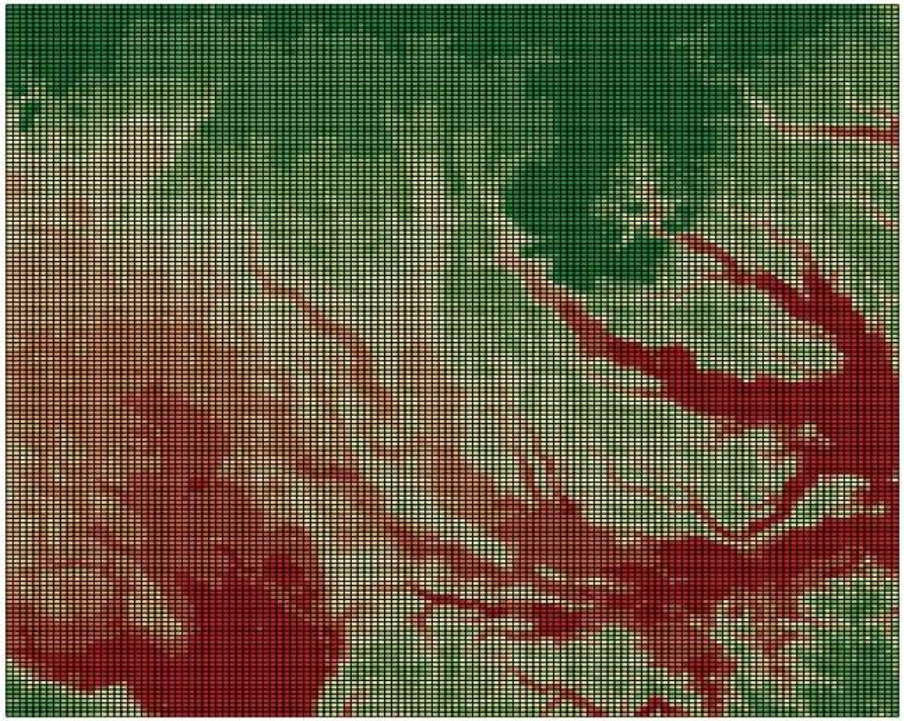
- 等間隔なので，低いところは一様になってしまう



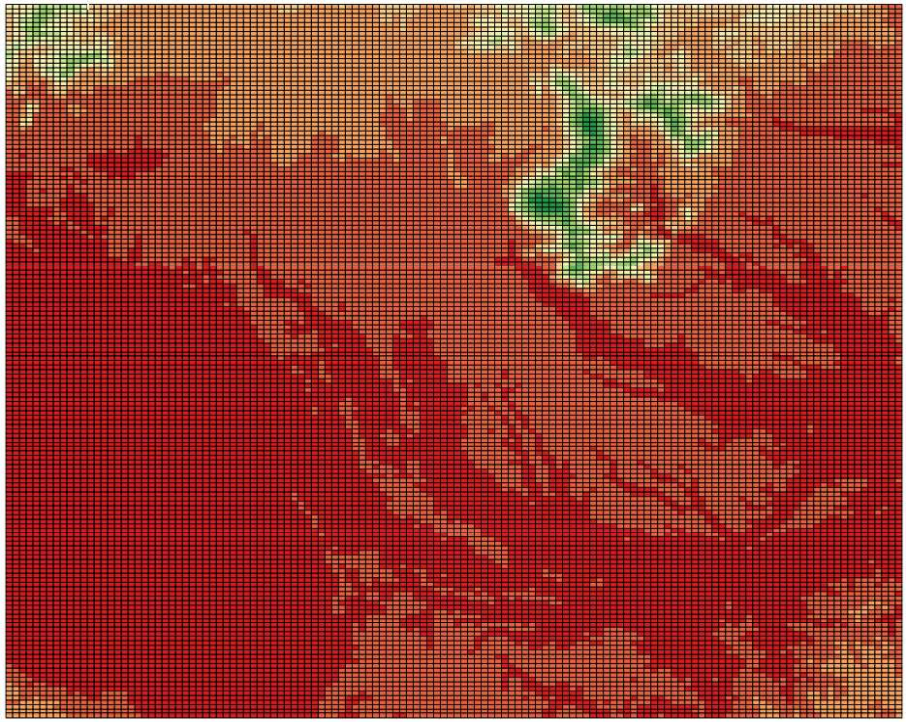
連続値の表示

- モードを変更する
  - いくつか試し，最も適したものを使用

分位（等量）



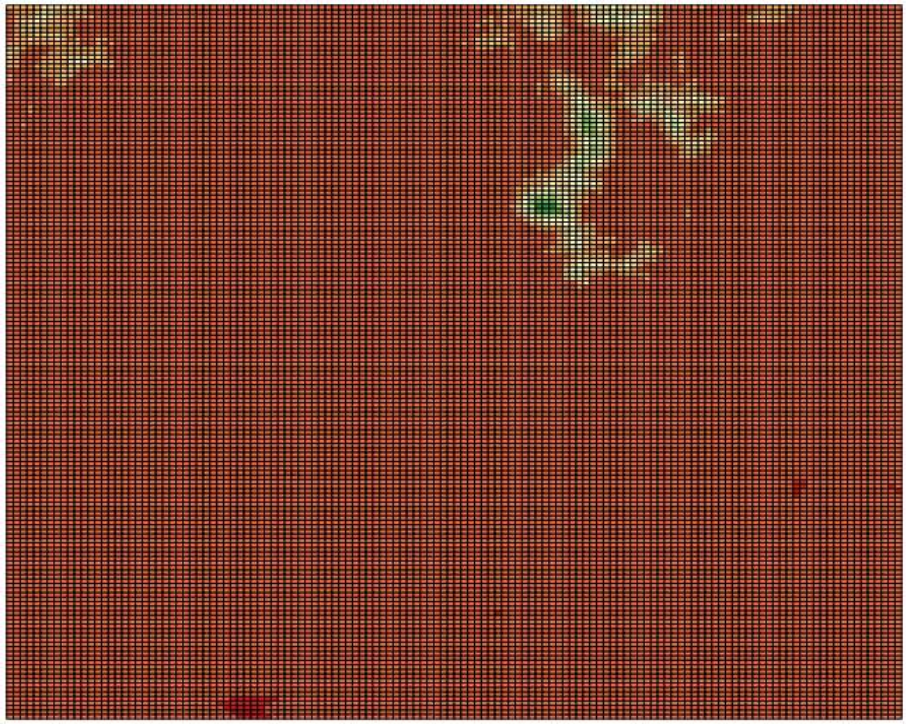
自然なブレイク(Jenks)



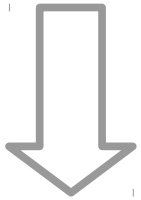
連続値の表示

標準偏差

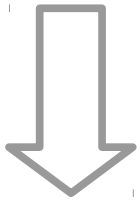
プリティブレイク



ベクタデータの活用



ラスタデータの高度な表示



ラスタデータの分析

ラスタデータの表示

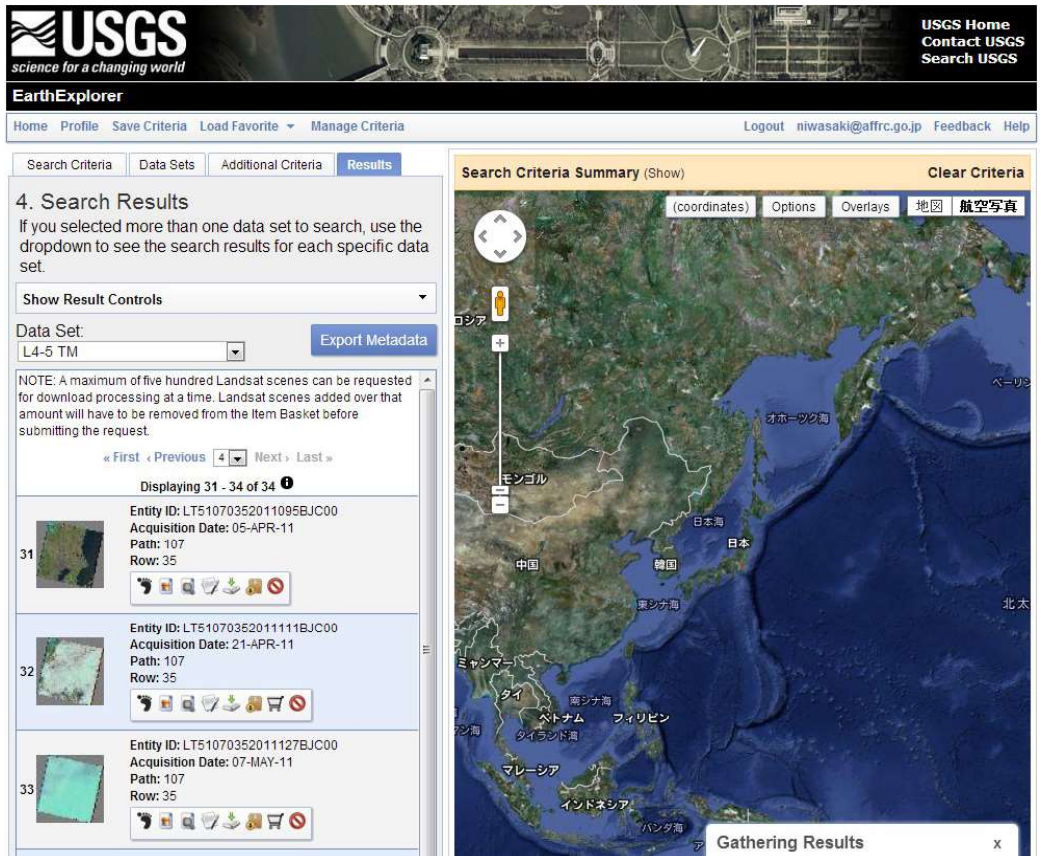
- ラスタデータの表示は2種類ある
  - 単バンドラスタ
    - DEMや主題図，国土地理院1/25,000など
      - 1つのレイヤに色を割り付ける
  - マルチバンドラスタ
    - 衛星画像や空中写真，スキャンした地図など
      - 複数のレイヤを組みあわせて表示する

単バンドラスタ

マルチバンドラスタ

使用するデータ

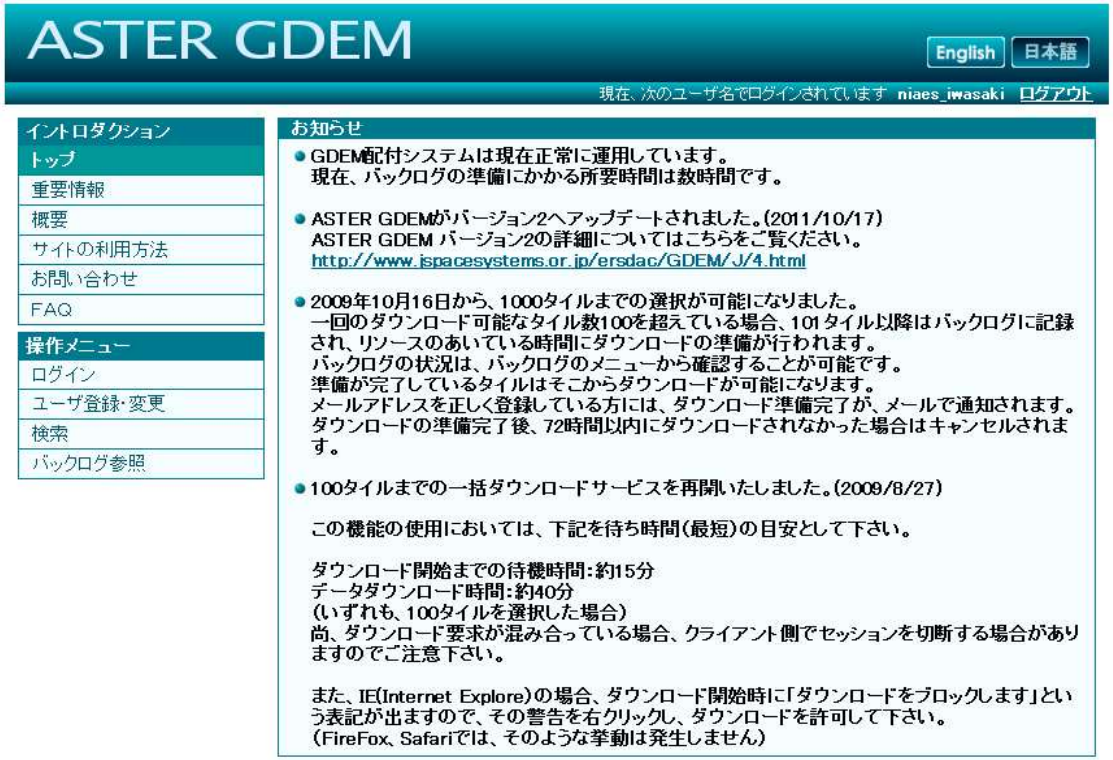
- LANDSAT TM
  - 2011年4月5日撮影
  - 7band，30m解像度
    - EarthExplorerより入手
    - http://earthexplorer.usgs.gov/
  - 近年はLandsat8も
    - Landsat-8 日本受信・即時公開サイト
      - http://landsat8.geogrid.org/l8/index.php/ja/



http://earthexplorer.usgs.gov/

使用するデータ

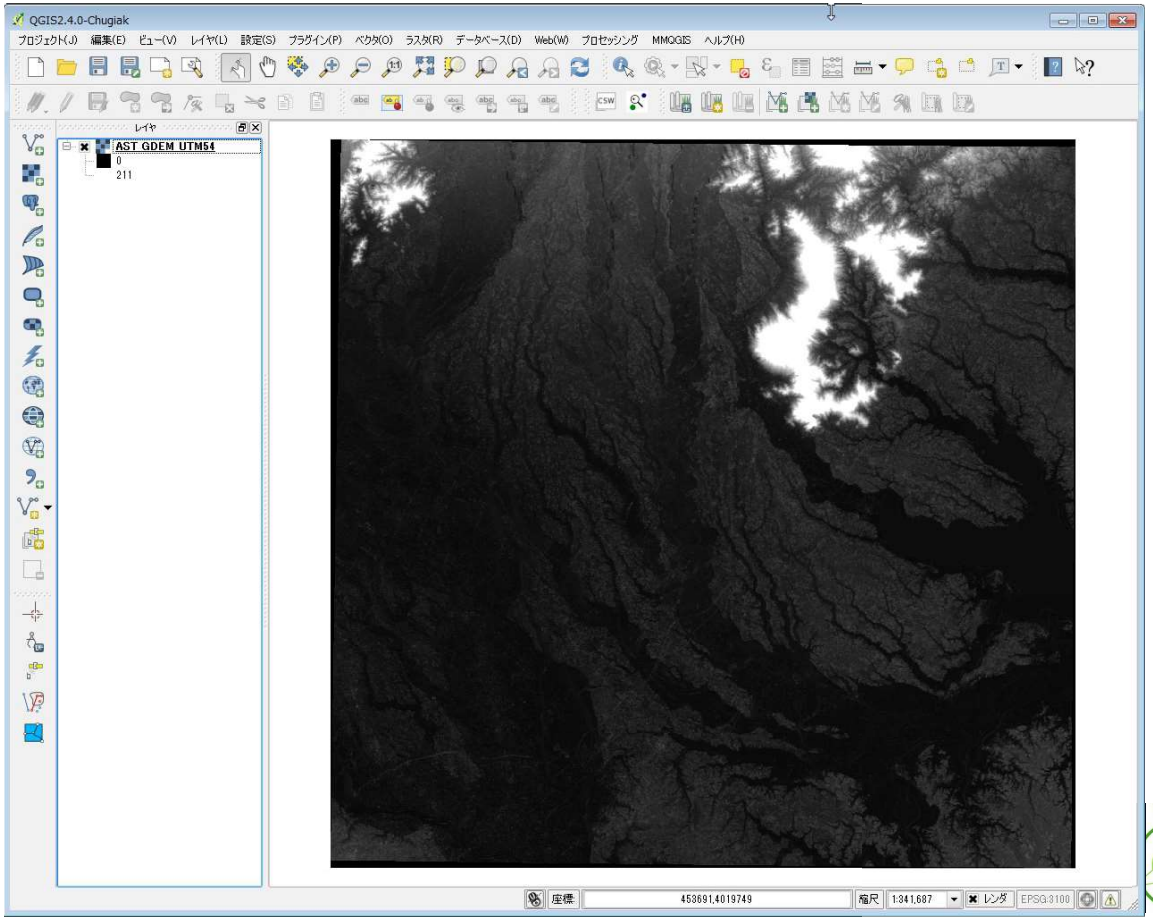
- ASTER GDEM
  - 全球30m解像度のDEMデータ
  - 以下より入手
    - http://gdem.ersdac.jspacesystems.or.jp/



http://gdem.ersdac.jspacesystems.or.jp/

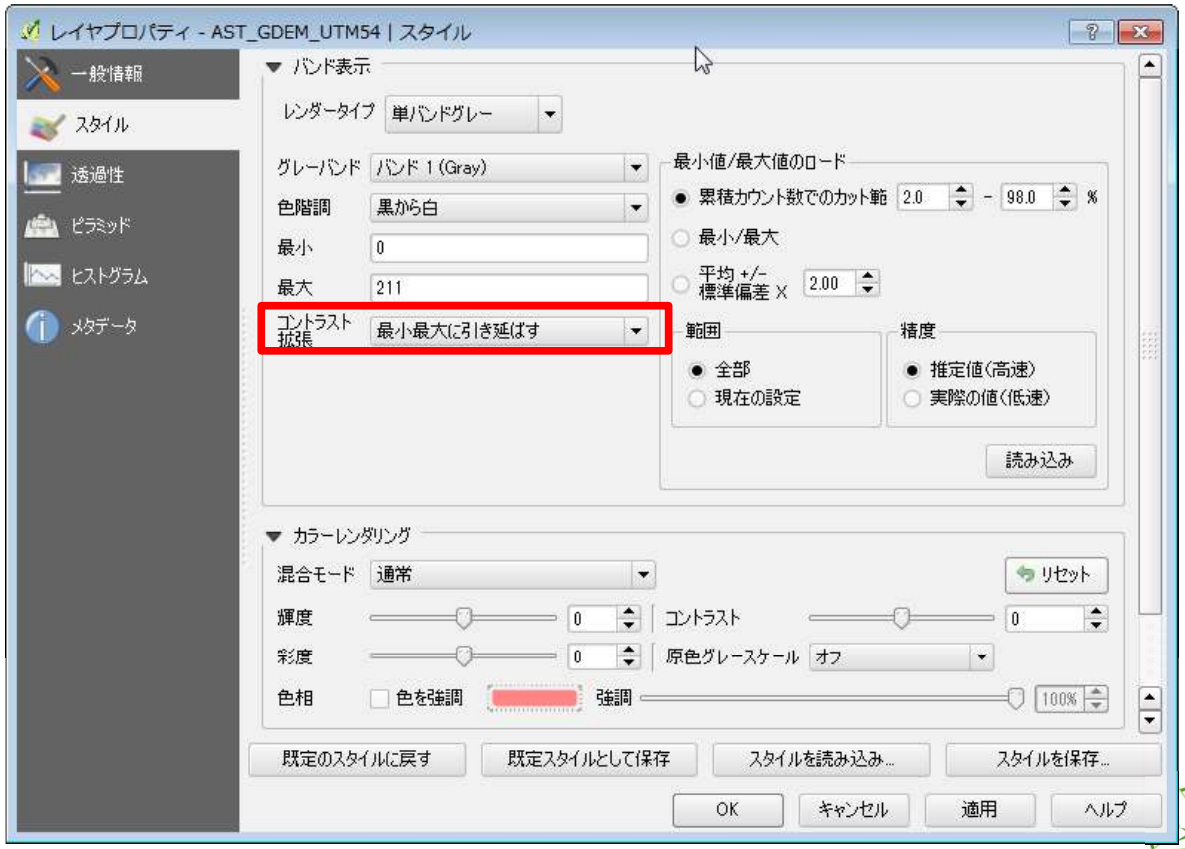
単バンドラスタの表示

- ASTER GDEMを開く
  - advanceの中の”AST_GDEM_UTM54.tif”
    - 以下のように表示される



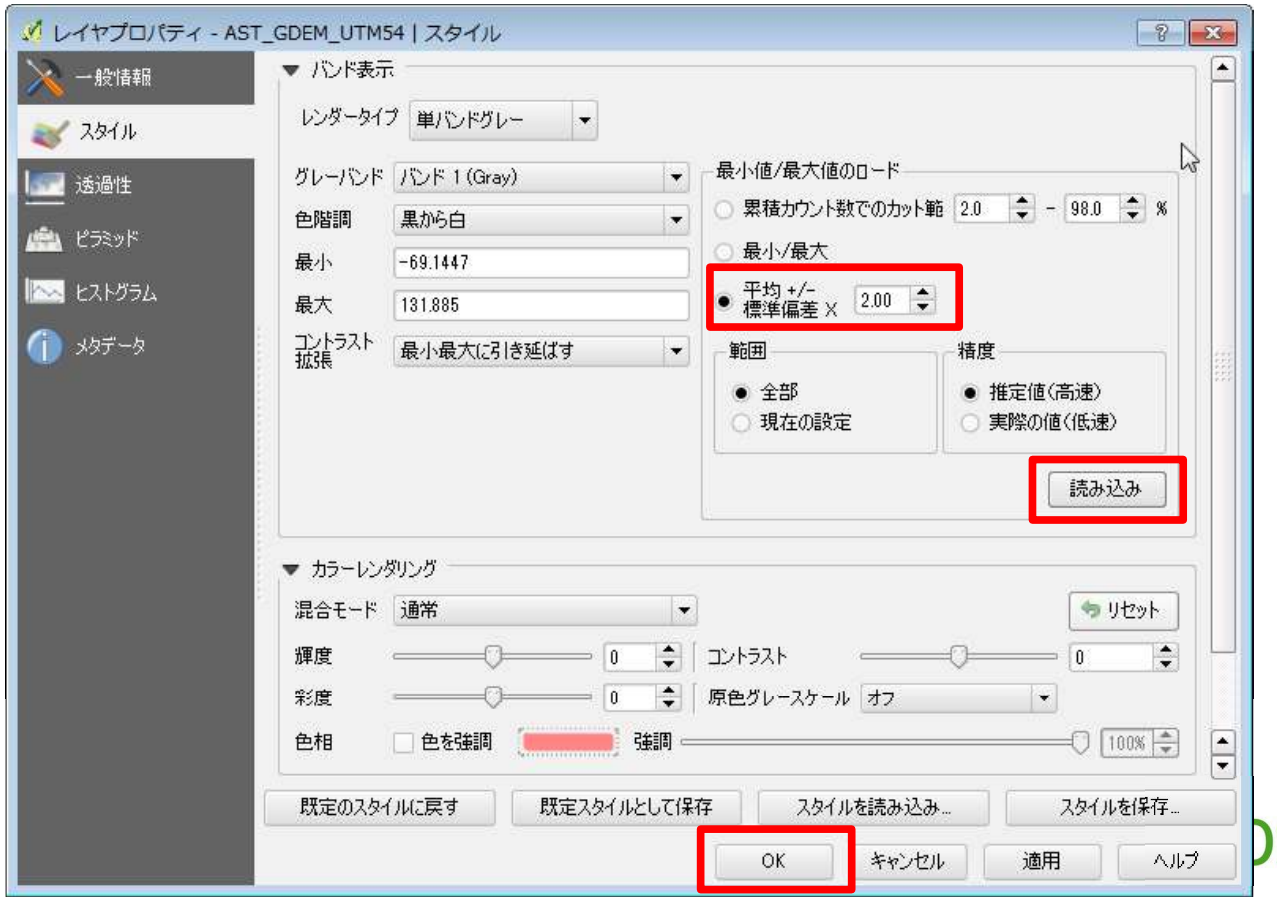
単バンドラスタの表示

- プロパティを開く
  - コントラスト拡張で”最小最大に引き延ばす”が選択されている



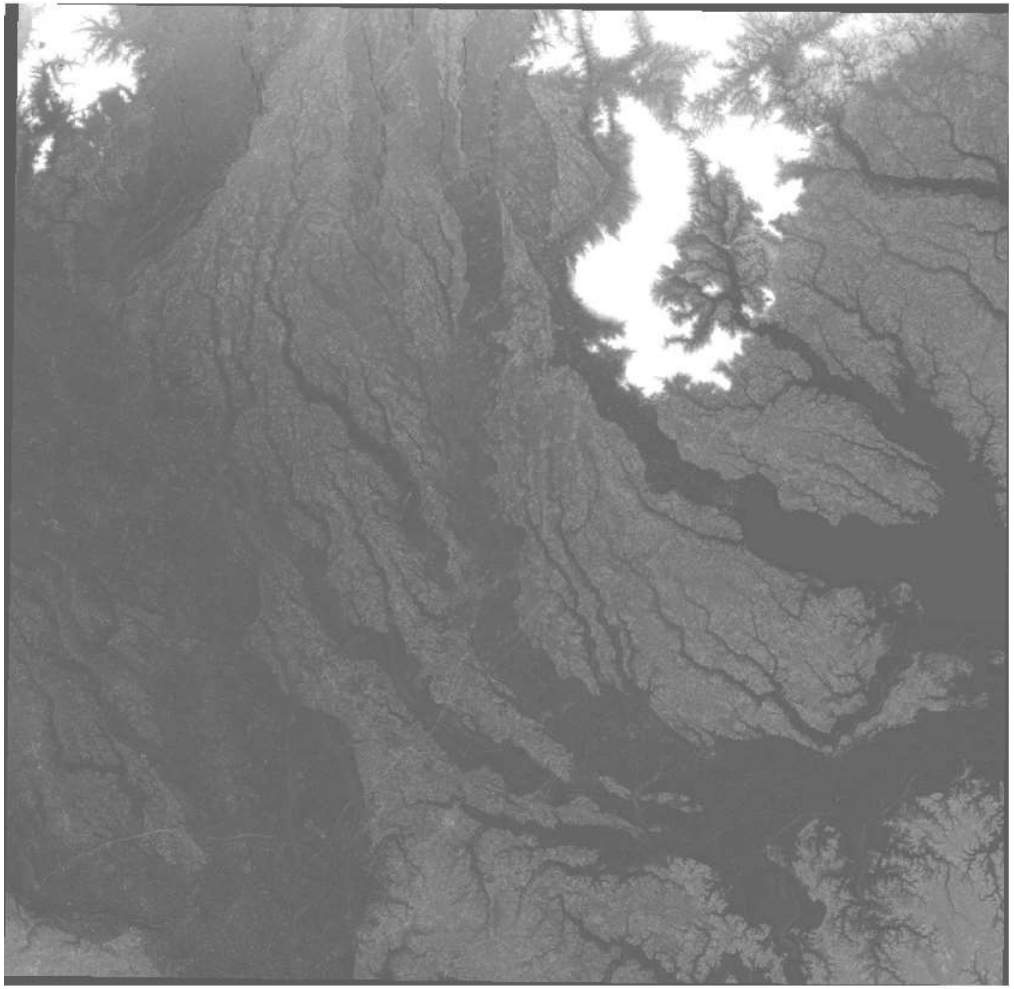
単バンドラスタの表示

- “平均+/-標準偏差X”をチェック
  - 値は”2.0”
  - 読み込みをクリックしてOK



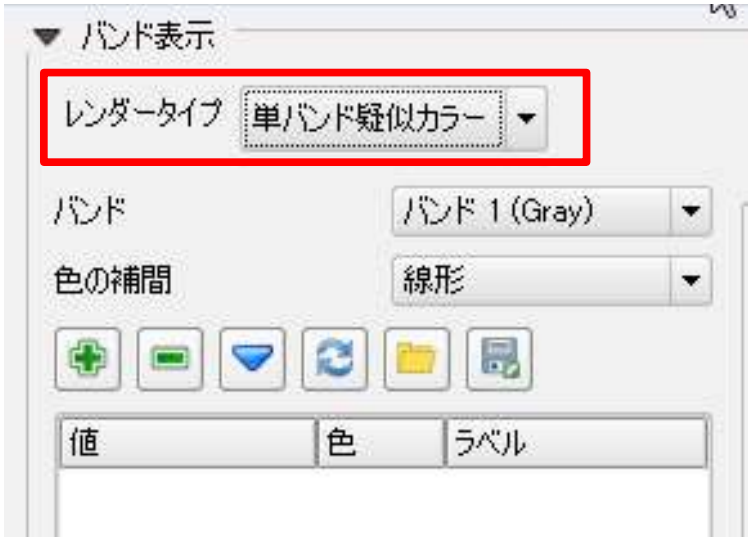
単バンドラスタの表示

- 高標高と低標高が潰れるが，中間が表示される
  - 約95%のデータを強調して表示



単バンドラスタの表示

- 単バンド疑似カラー
  - 値と色を明示的に指定
- バンド表示のレンダータイプで”単バンド疑似カラー”を選択

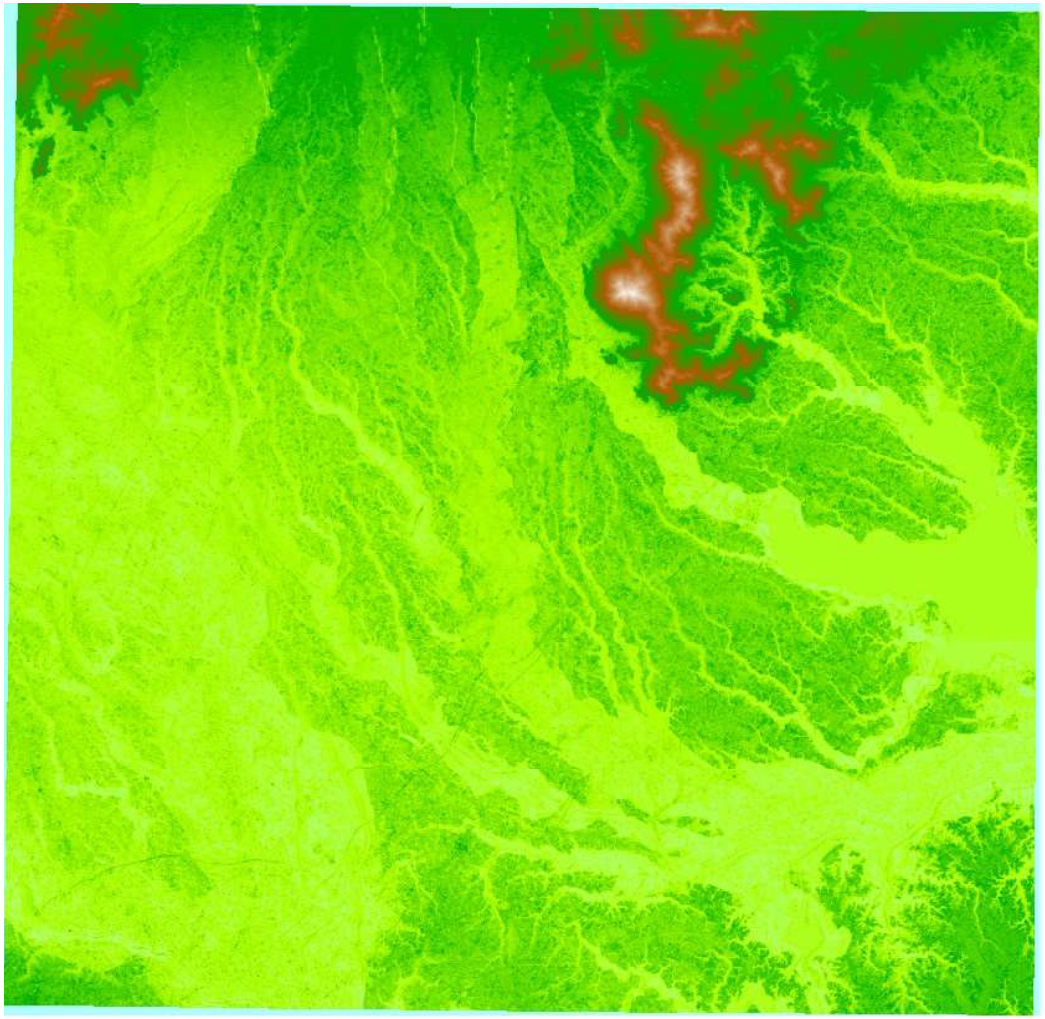


単バンドラスタの表示

- カラーマップを調整する
  - 色の補完を”線形”にする
  - エントリーの追加で行を追加，ダブルクリックして色と値を変更する(900は白)

単バンドラスタの表示

- 低標高，高標高ともに強調できる
  - エキスポートで保存も可能

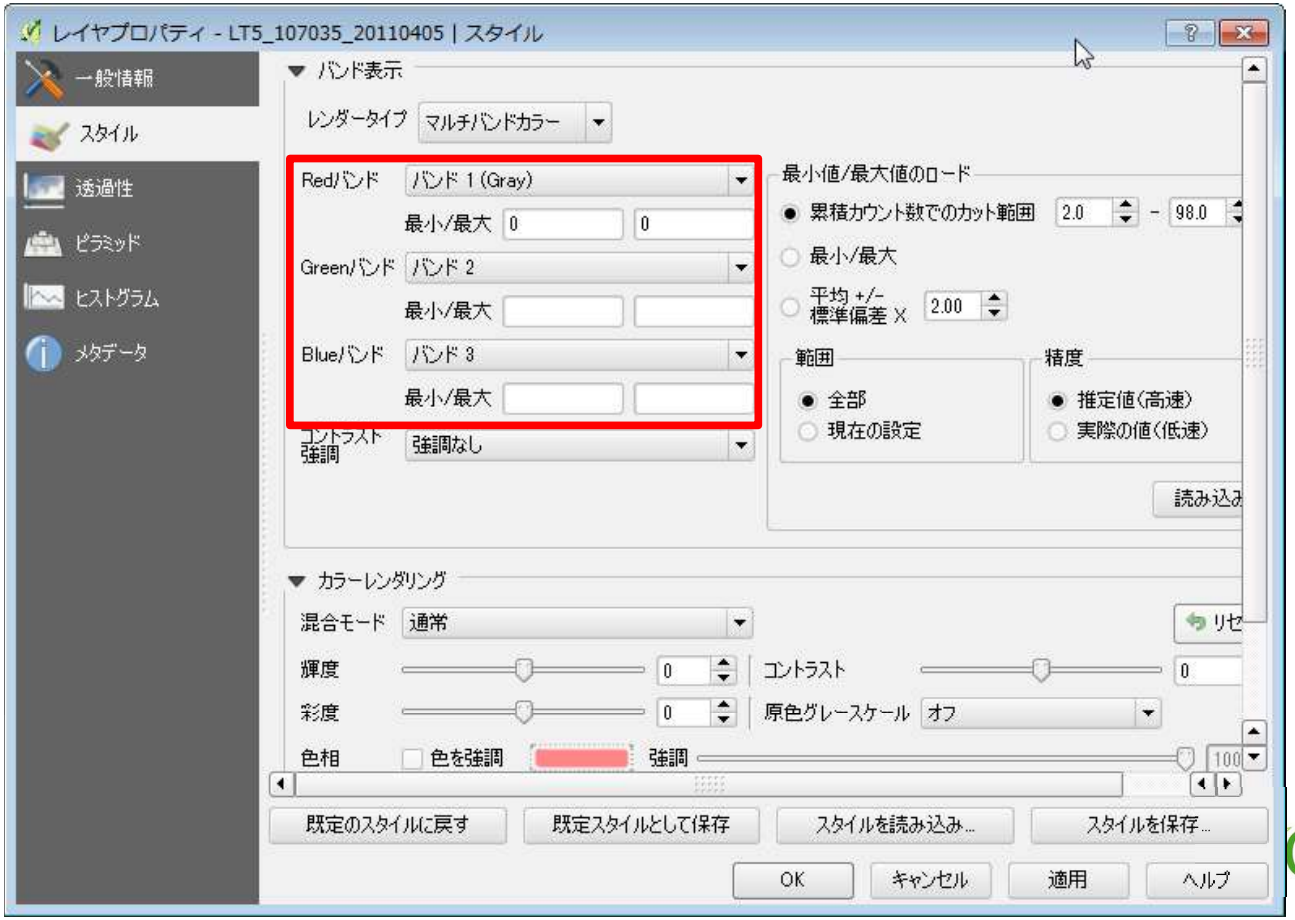


マルチバンドカラーの表示

- LT5_107035_20110405を開く
  - 色調が不自然

マルチバンドカラーの表示

- “RGBモードでのバンド選択とスケーリング”が選択可能
  - Redが1，Greenが2，Blueが3



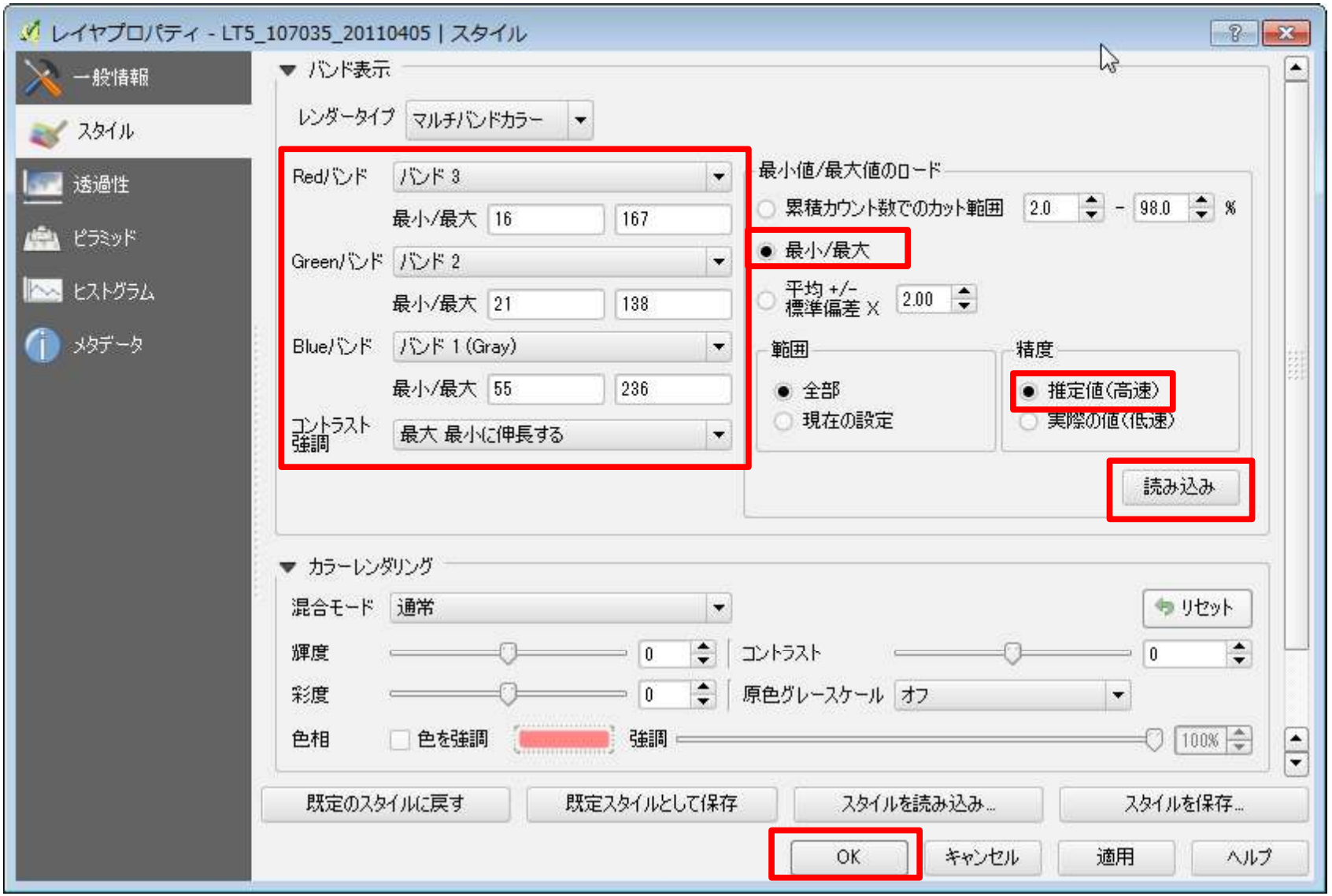
OSGeo.JP

マルチバンドデータ

- 1つのデータが複数バンドデータから構成される
  - 通常の画像の場合，バンド1が赤，バンド２が緑，バンド3が青
- 衛星データではセンサー毎に異なる
  - Landsat TMの場合，
    - band1：青，band2：緑，band3：赤
    - band4：近赤外，band5：中間赤外
    - band6：熱赤外，band7：中間赤外
  - 可視とは違うも強調が可能

マルチバンドカラーの表示

- 以下のように調整
  - バンド選択でRedを3，Greenを2，Blueを1に変更
    - 赤，緑，青を正しく定義
  - “最小値/最大値のロード”で”最大/最小”と”推定値(高速)”をチェックして”読み込み”をクリック
    - データの幅を定義する
  - “コントラスト強調”で”最大　最小に伸長する”を選択
    - 表示をデータのある範囲に限定


レイヤプロパティ - LT5_107035_20110405 | スタイル
一般情報
スタイル
透過性
ピラミッド
ヒストグラム
メタデータ
バンド表示
レンダータイプ
マルチバンドカラー
Redバンド
バンド 3
最小/最大 16 167
Greenバンド
バンド 2
最小/最大 21 138
Blueバンド
バンド 1 (Gray)
最小/最大 55 236
コントラスト強調
最大 最小に伸長する
最小値/最大値のロード
累積カウント数でのカット範囲 2.0 - 98.0 %
最小/最大
平均 +/- 標準偏差 X 2.00
範囲
全部
現在の設定
精度
推定値(高速)
実際の値(低速)
読み込み
カラーレンダリング
混合モード 通常
リセット
輝度 0
コントラスト 0
彩度 0
原色グレースケール オフ
色相
色を強調
強調 100%
既定のスタイルに戻す
既定スタイルとして保存
スタイルを読み込み...
スタイルを保存...
OK
キャンセル
適用
ヘルプ

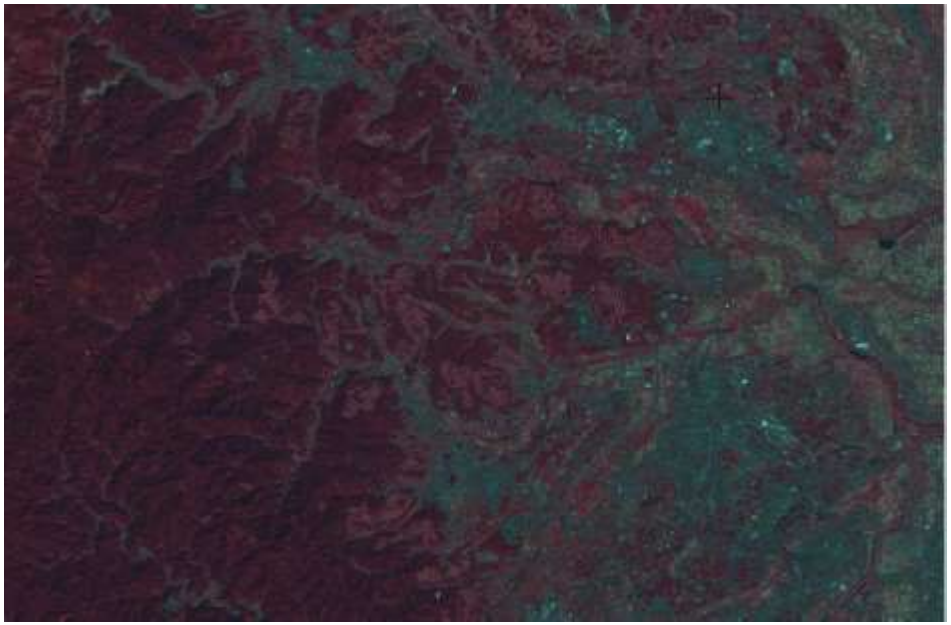
マルチバンドカラーの表示

- 自然な色調に変更

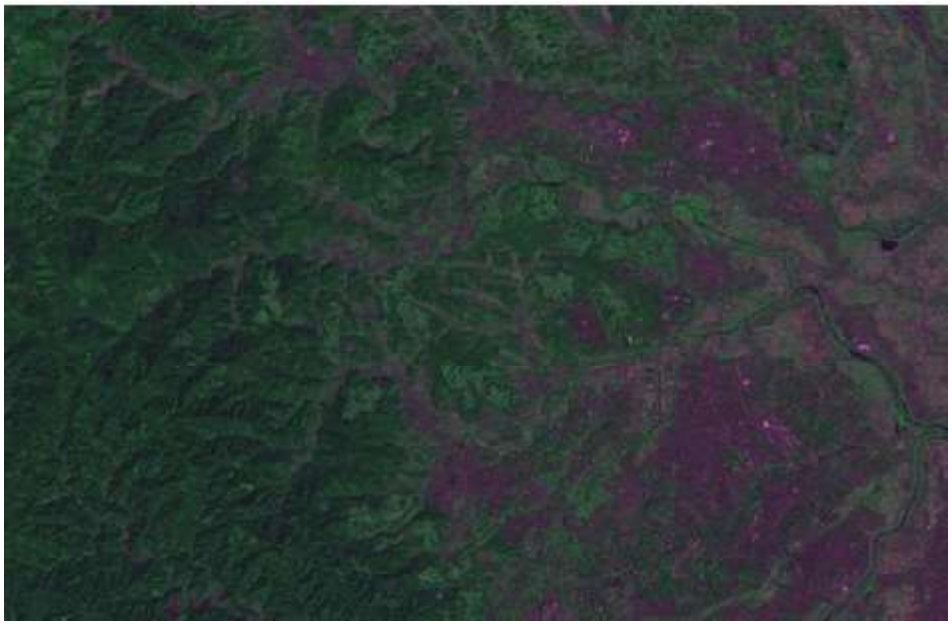
マルチバンドカラーの表示

- 代表的なバンド選択例

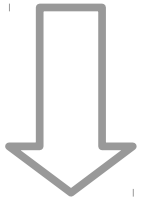
フォルスカラー（R=4, G=3, B=2）



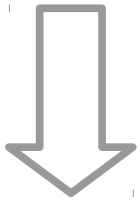
ナチュラルカラー（R=3, G=4, B=2）



ラスターデータの高度な表示



ラスターデータの分析



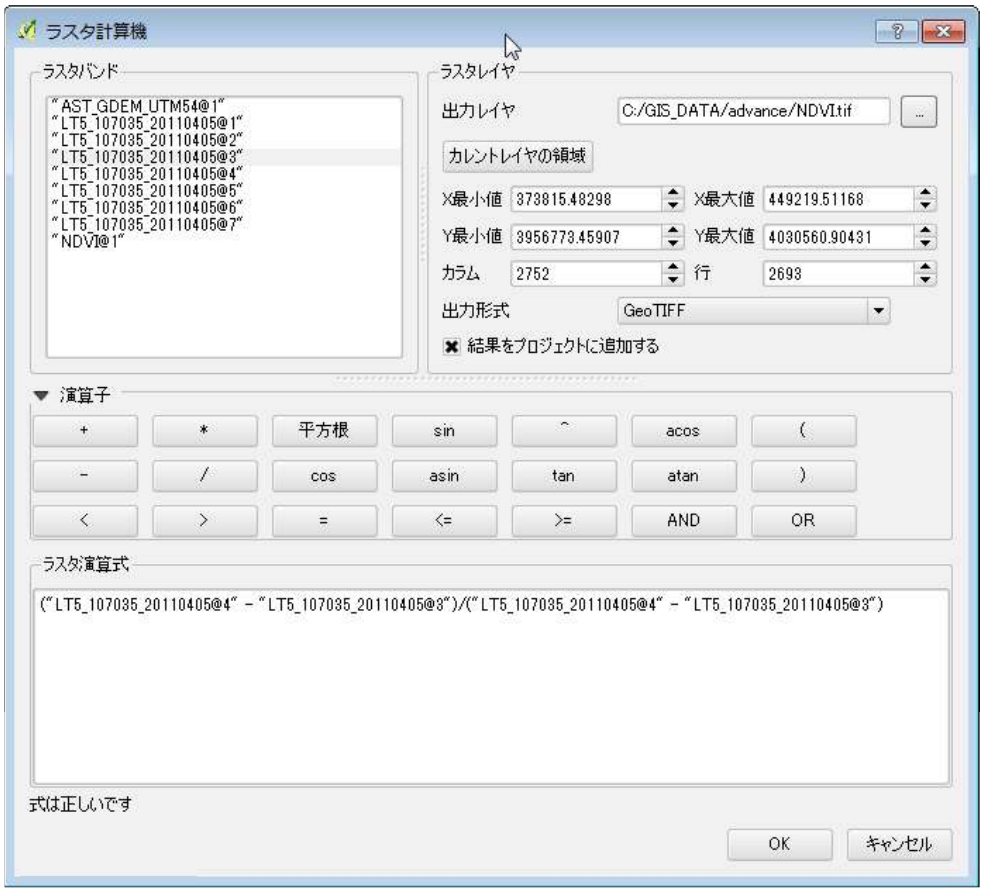
Pythonコンソールの利用

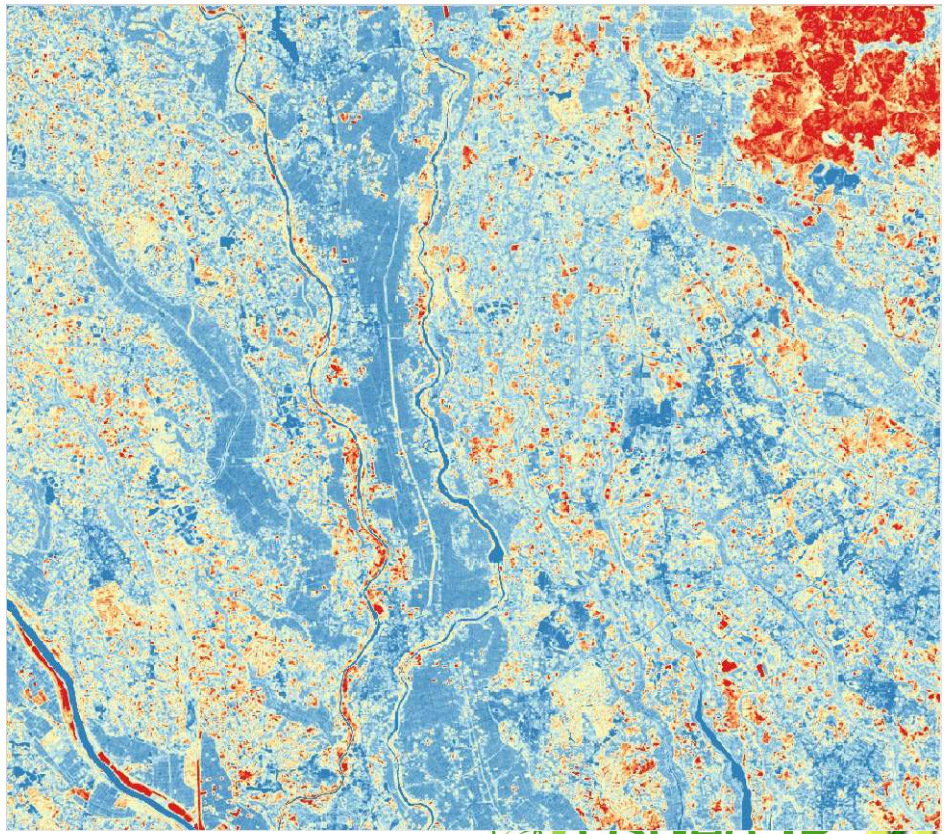
QGISのラスタ分析

- メニューのラスタから選択
  - ラスタ計算機
  - 投影法
  - 変換
  - 抽出
  - 解析
  - その他
  - GdalTools設定

ラスタ計算機

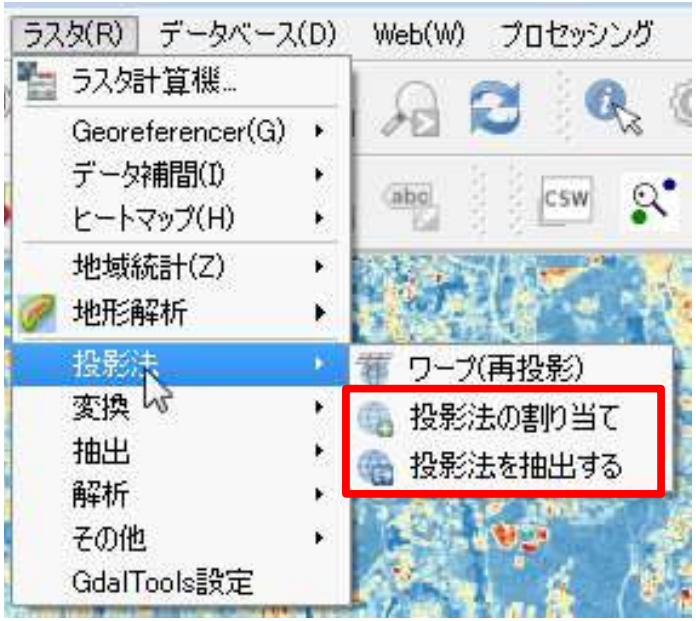
- ラスタレイヤ間，ラスタバンド間での計算が可能
  - NDVI（正規化植生指数）の算出など
    - NDVI=(band4-band3)/(band4+band3)





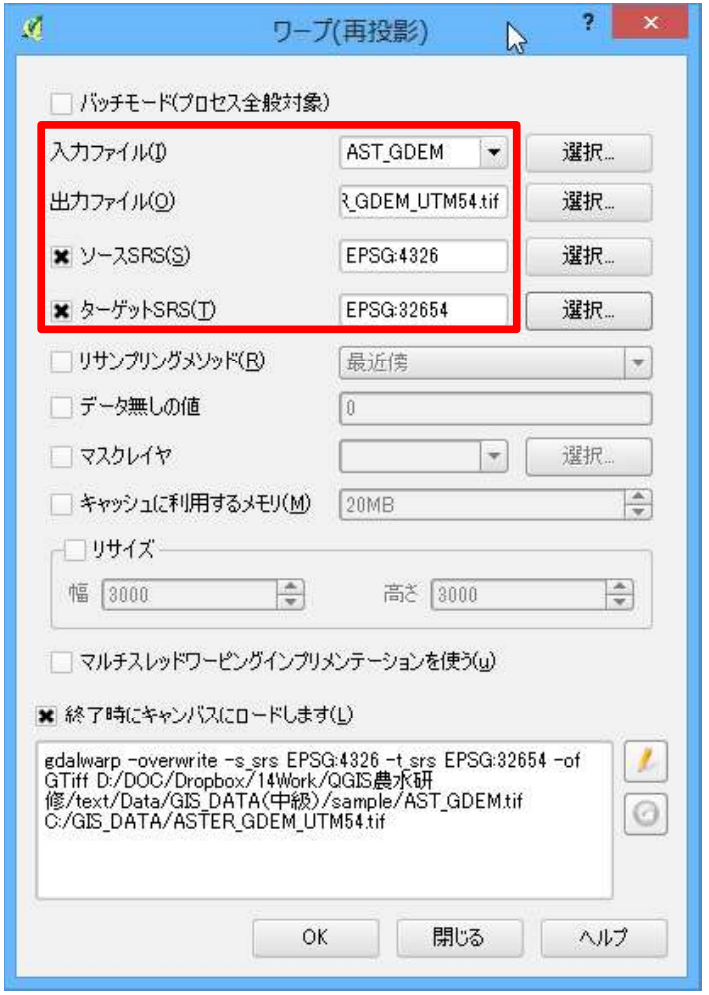
投影法メニュー

- 投影法の定義や変換
  - 投影法の割り当て
    - 投影法のないデータに投影法を定義する
  - 投影法を抽出する
    - データからwldファイルやprjファイルを作る



ワープ（再投影）ツール

- 投影法の変換を行う
  - 緯度経度 ←→ UTM
  - ワープ(再投影)をクリック
  - 入力ファイルで”AST_GDEM”を選択
  - 出力ファイルを”AST_GDEM_UTM54_2.tif”と設定
  - ターゲットSRSを”EPSG:32654”を選択
    - 緯度経度なのでUTM54に変換する

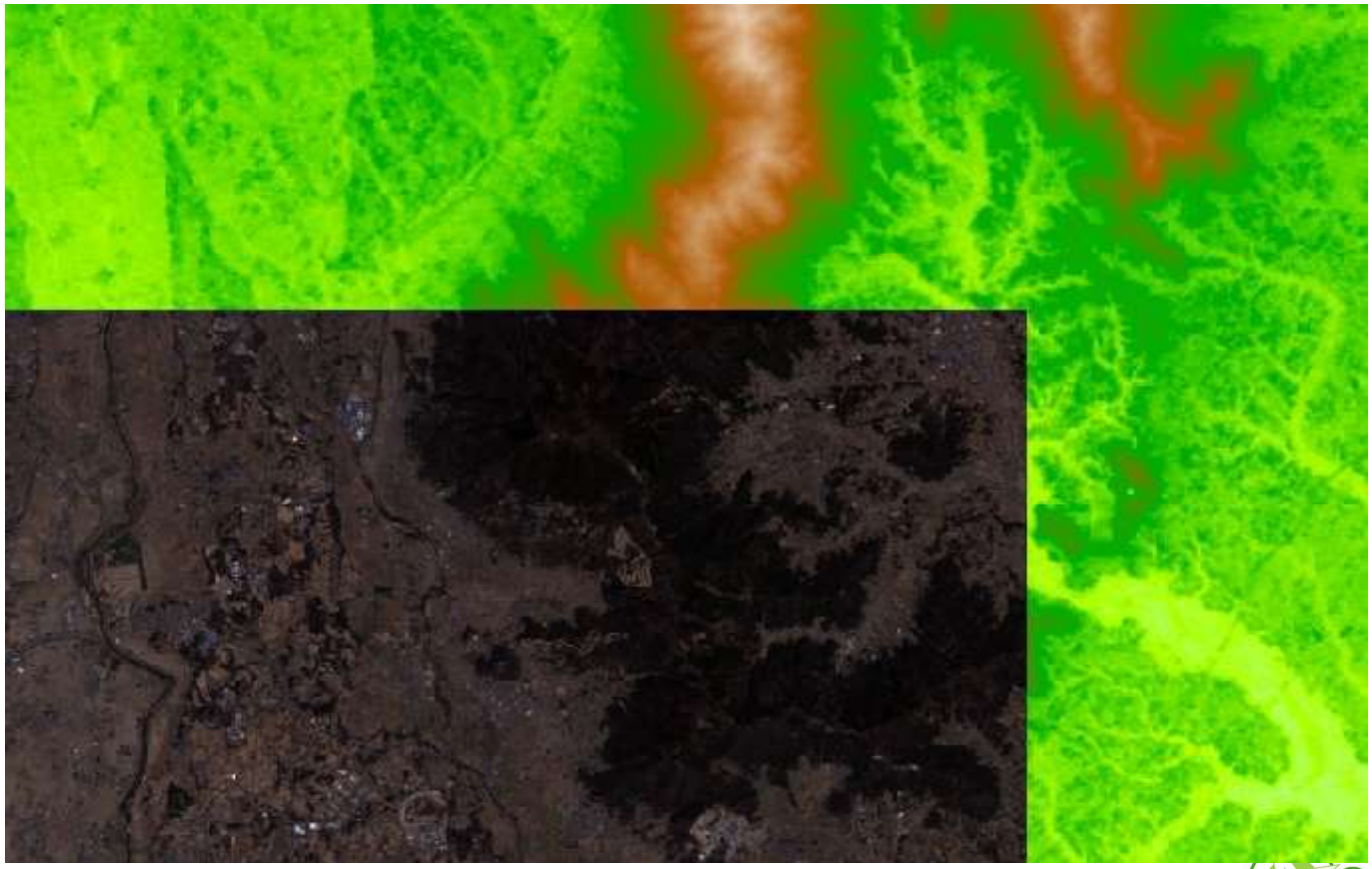


ワープ（再投影）ツール

- 下の枠で”-of GMT”となっている場合
  - 右の鉛筆マークをクリック
  - “-of GTiff”と書き換える。
    - 出力ファイル形式GeoTiffに指定する。

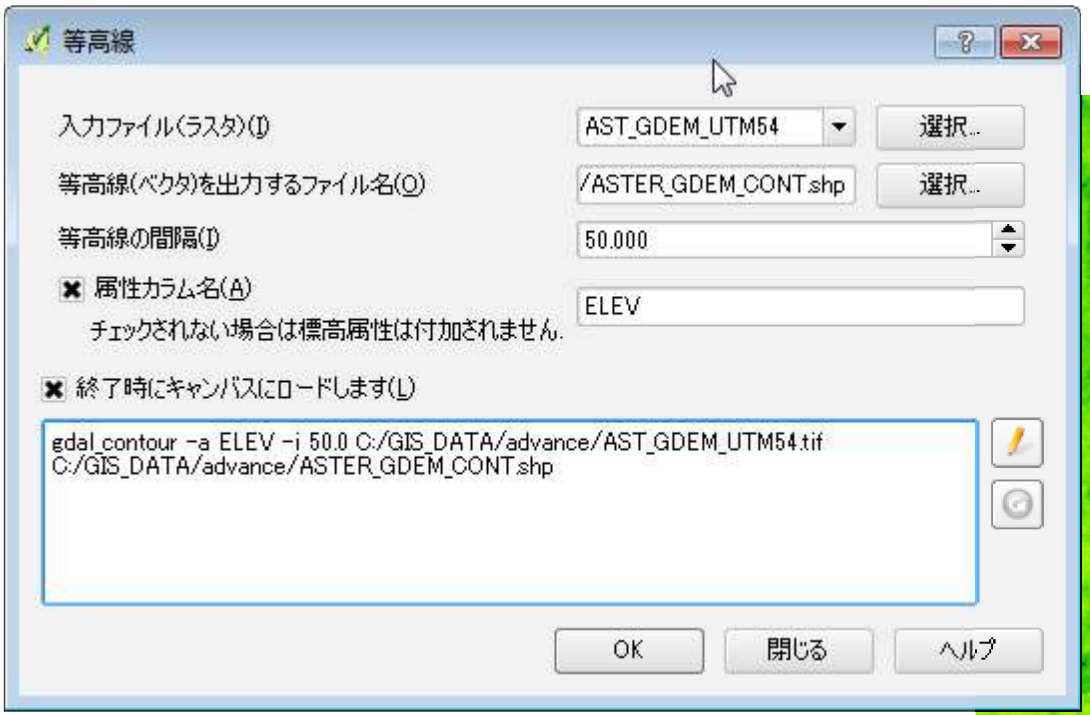
ワープ（再投影）ツール

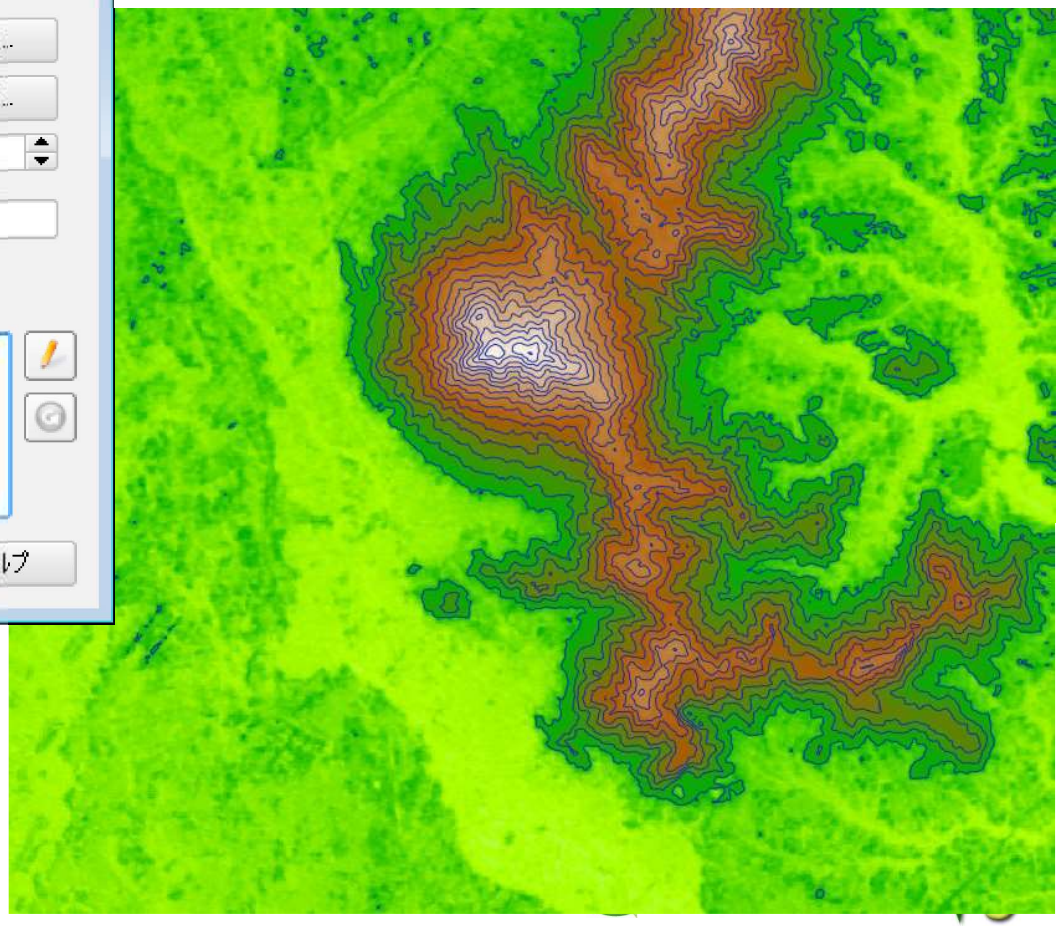
- LANDSATのデータと重なるのを確認
  - 同じ投影法に変換された



抽出メニュー

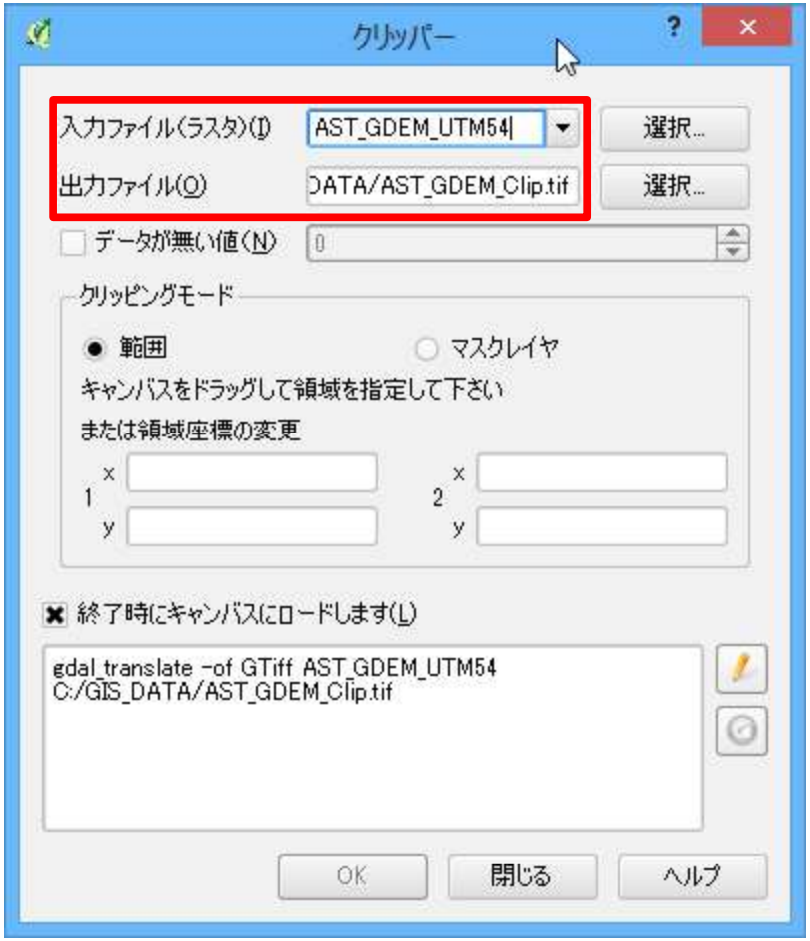
- ラスタ→抽出→等高線
  - DEMから等高線を作成





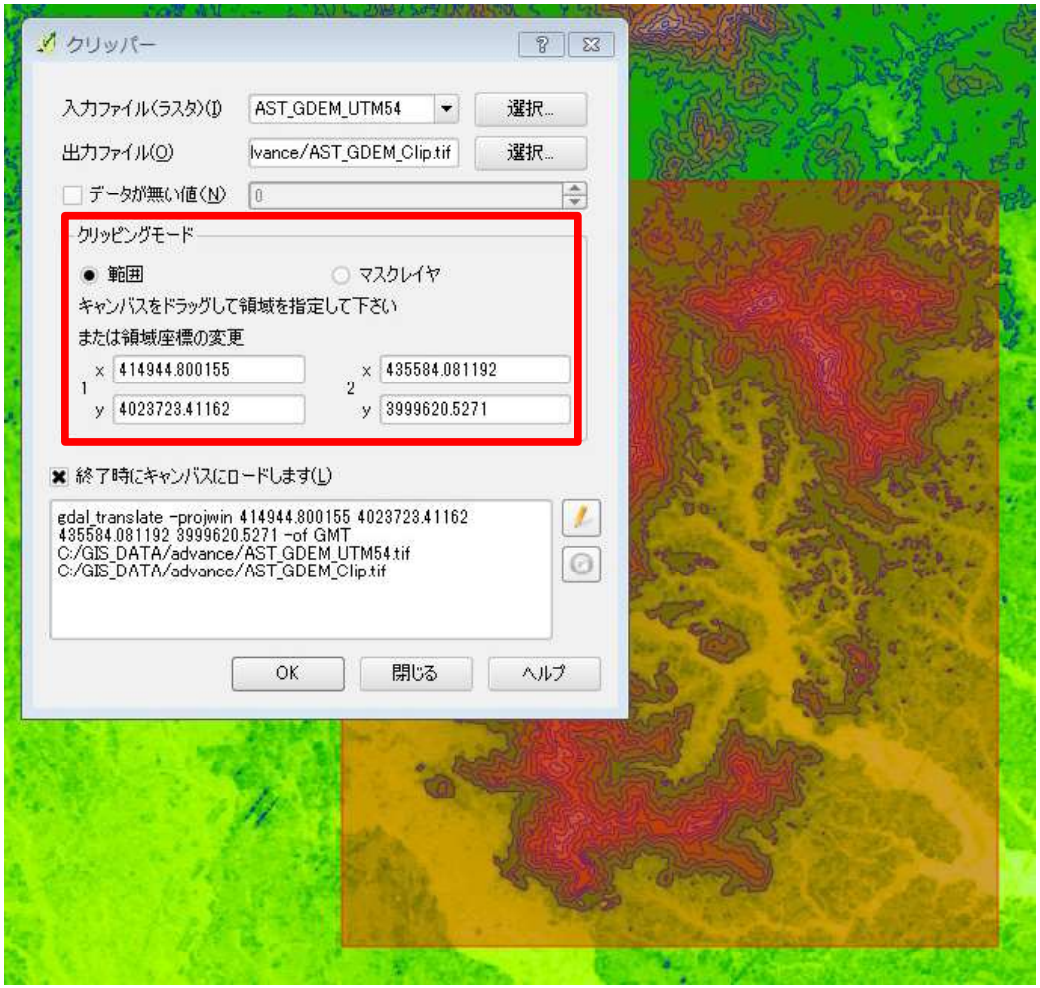
クリッパー

- ラスタデータの切抜きを行う
  - ラスタ→抽出→クリッパー
  - 入力ファイルに”AST_GDEM_UTM54”を選択
  - 出力ファイルを”AST_GDEM_Clip.tif”と設定
    - ワープツールと同様に出力形式を"-of GTiff”に変更



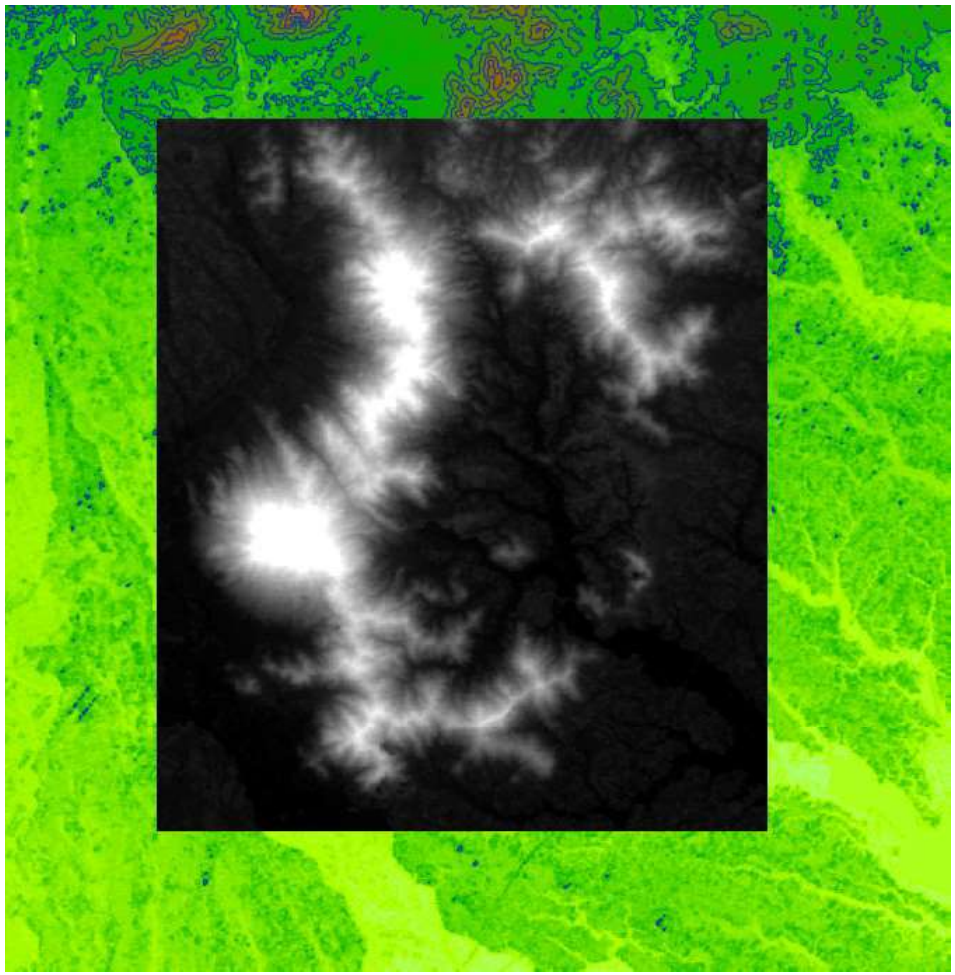
クリッパー

- 範囲の選択
  - クリッピングモードが”範囲”になっているのを確認
  - 切り抜きたい範囲をドラッグ
    - 座標値が更新される
    - OKをクリック
  - 保存先に日本語のフォルダは使用できない



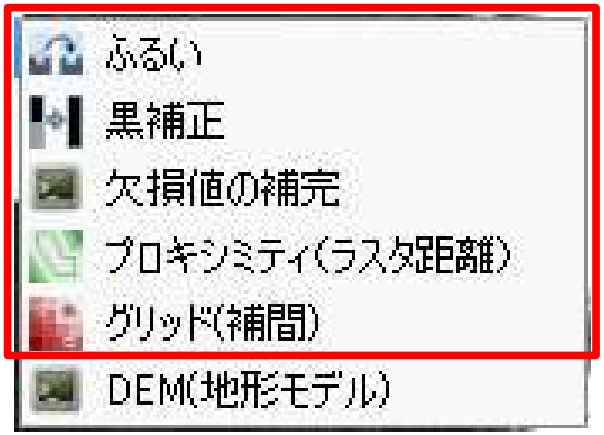
クリッパー

- 選択範囲が切り抜かれる
  - マスクレイヤで他のレイヤの指定も可能



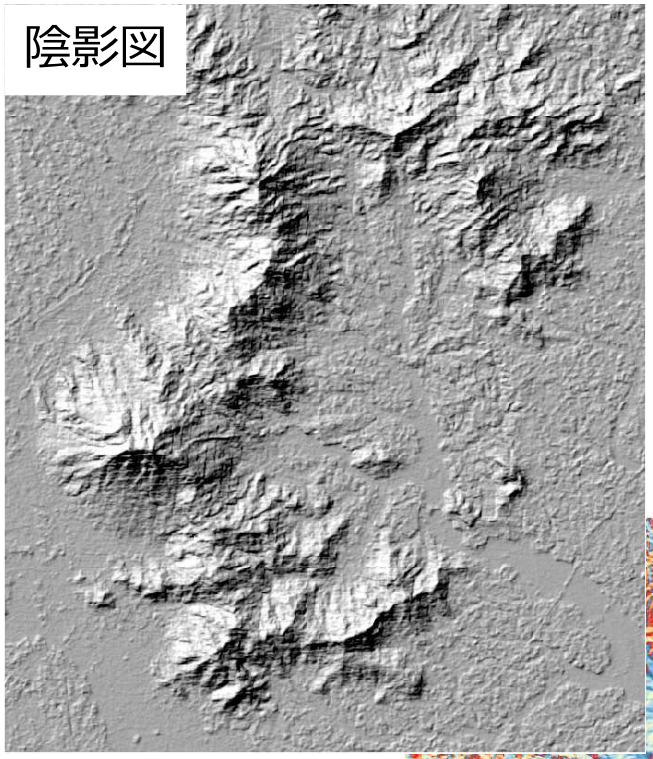
解析手法メニュー

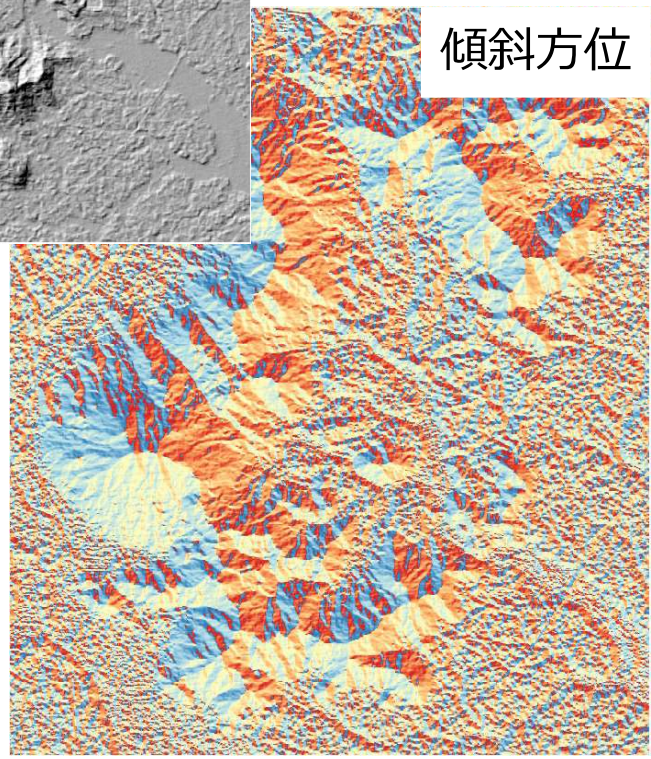
- ふるい
  - 小さなセルを除去
- 黒補正
  - null値を修正する
- 欠損値の補間
  - nodataに値を挿入
- プロキシミティ(ラスタ距離)
  - 距離値を持ったバッファ
- グリッド(補間)
  - ポイントからラスタ作成



DEM（地形モデル）

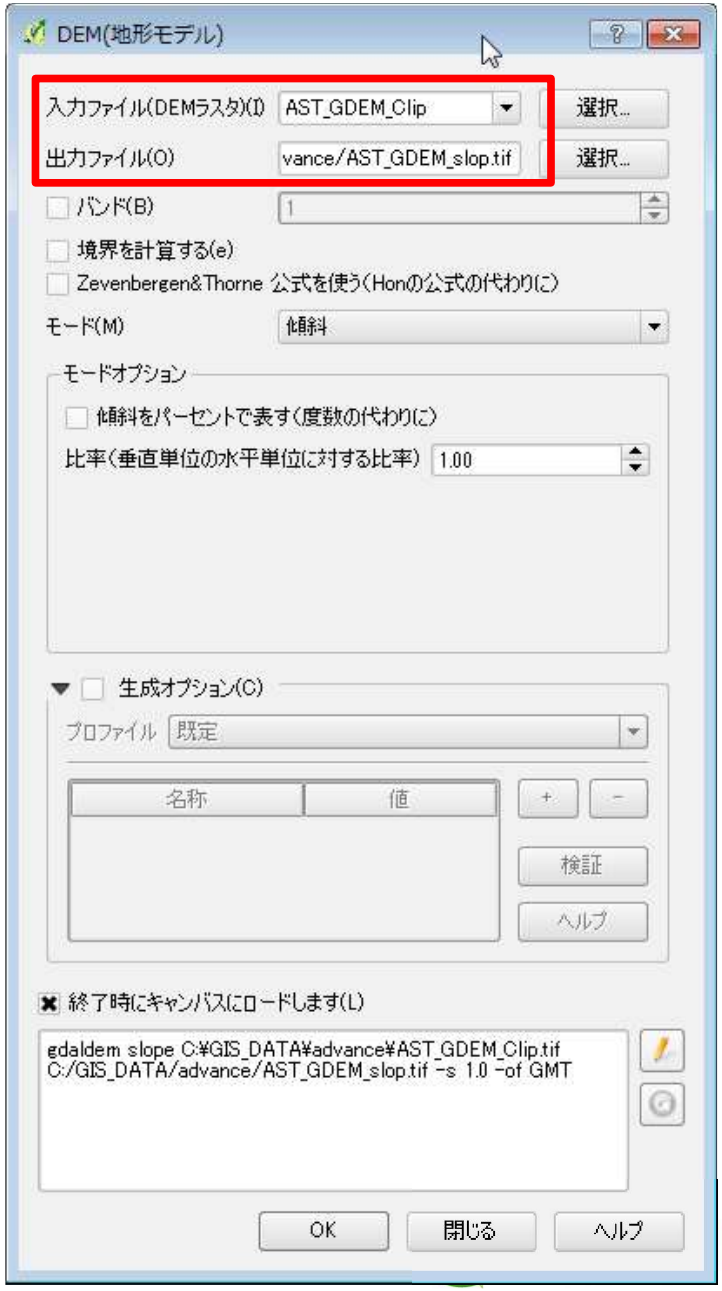
- DEMを使った地形解析
  - モードで選択する
  - 陰影図
    - 影のデータを作成
  - 傾斜方位
    - 斜面の方位を計算





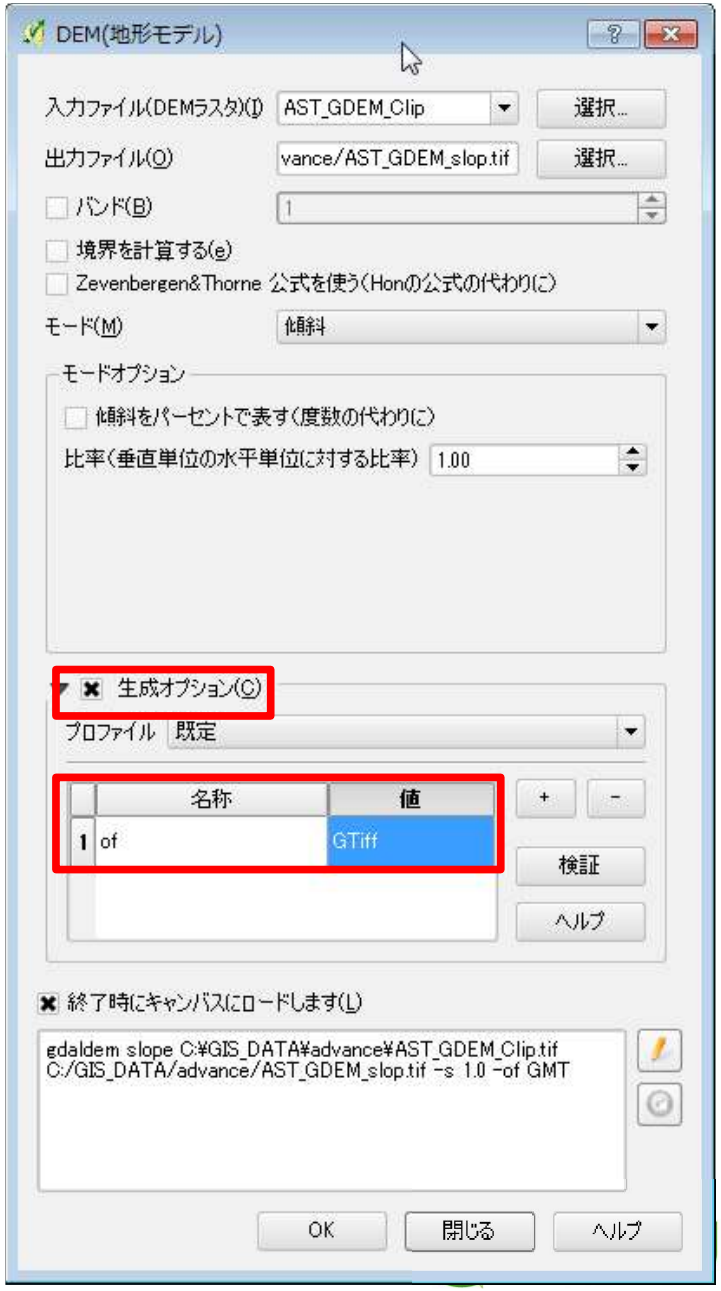
傾斜ツール

- 斜面傾斜を計算する
  - 入力ファイル（DEMラスタ）でAST_GDEM_Clipを選択
  - 出力ファイルに”AST_GDEM_slope.tif”を指定
  - モードで”傾斜”を選択



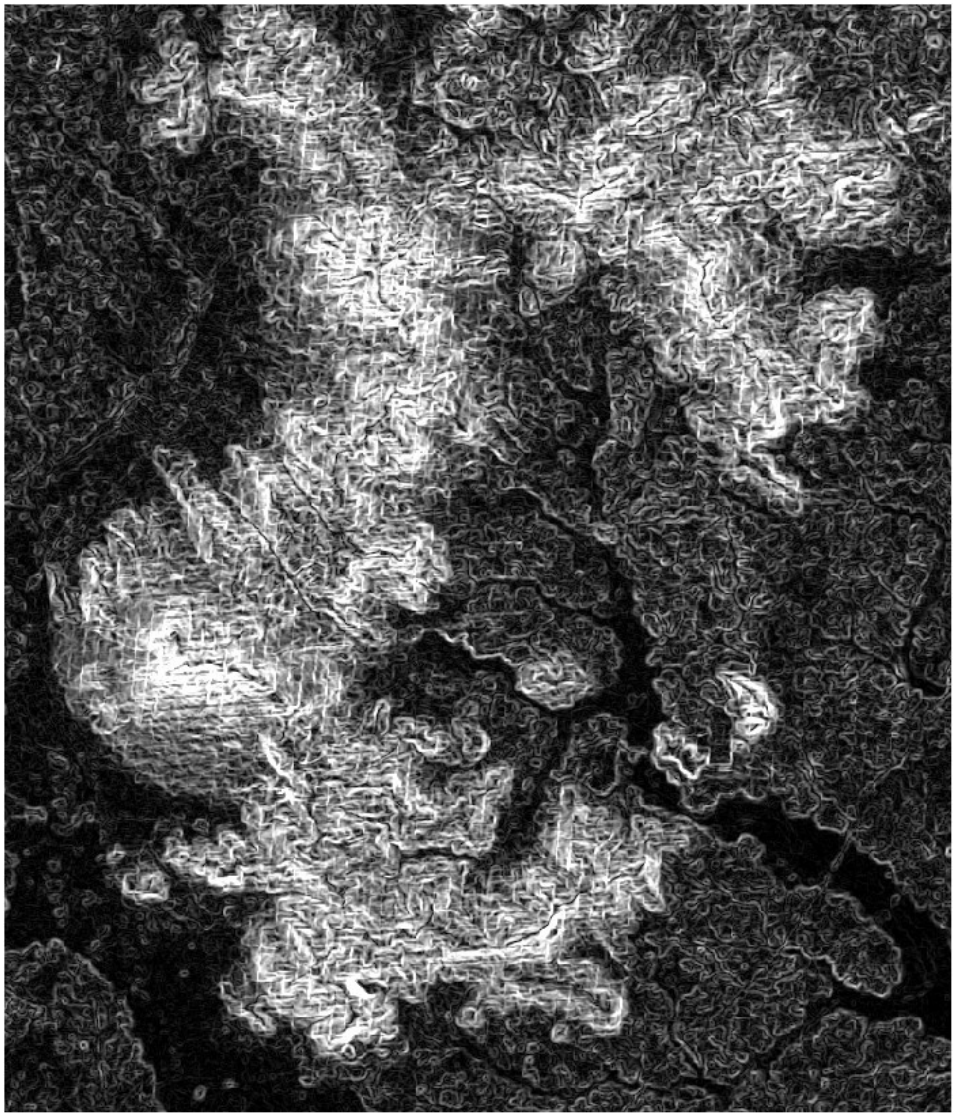
注意点！

- 出力形式をGeoTiffに代える必要あり
  - デフォルトの出力形式がGMT形式
    - DEM(地形モデル)全て
  - 生成オプションにチェック
  - ”名称”に”of”
  - ”値”に”GTiff”
    - 出力形式をGeoTiffにするという指定



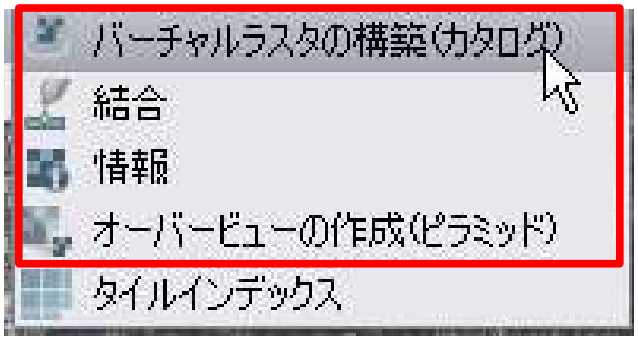
傾斜ツール

- 傾斜計算結果

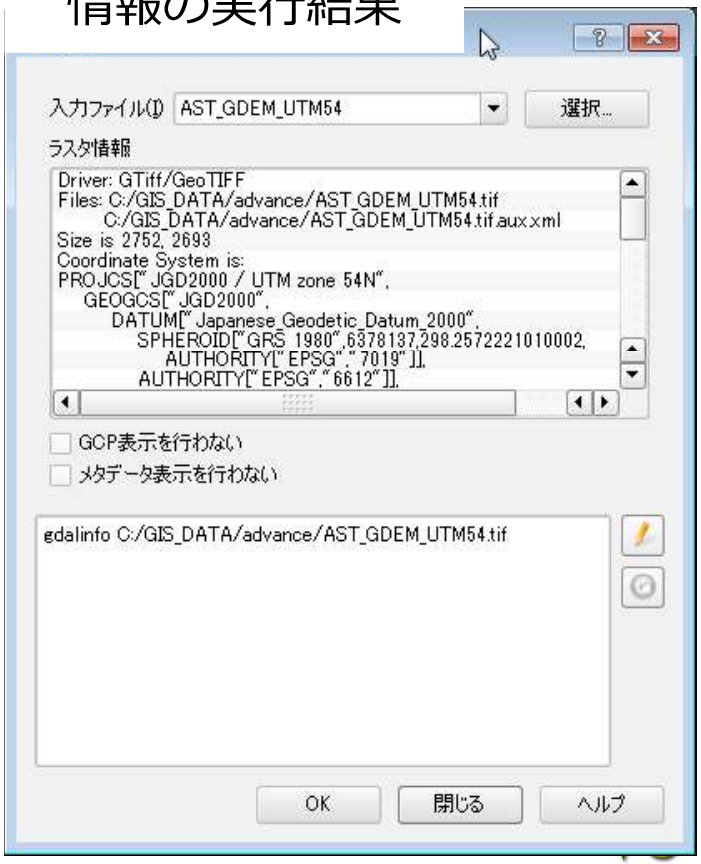


その他メニュー

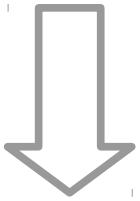
- バーチャルラスターの構築(カタログ)
  - 単バンドラスタからマルチバンドラスタを仮想的に作成など
- 結合
  - 複数のラスタを1つに結合
- 情報
  - 測地系，最大値，最小値等の確認
- オーバービューの作成（ピラミッド）
  - ファイルを高速で表示するためのデータを作成



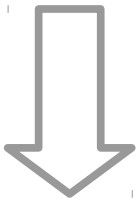
情報の実行結果



ラスターデータの分析



Pythonコンソールの利用

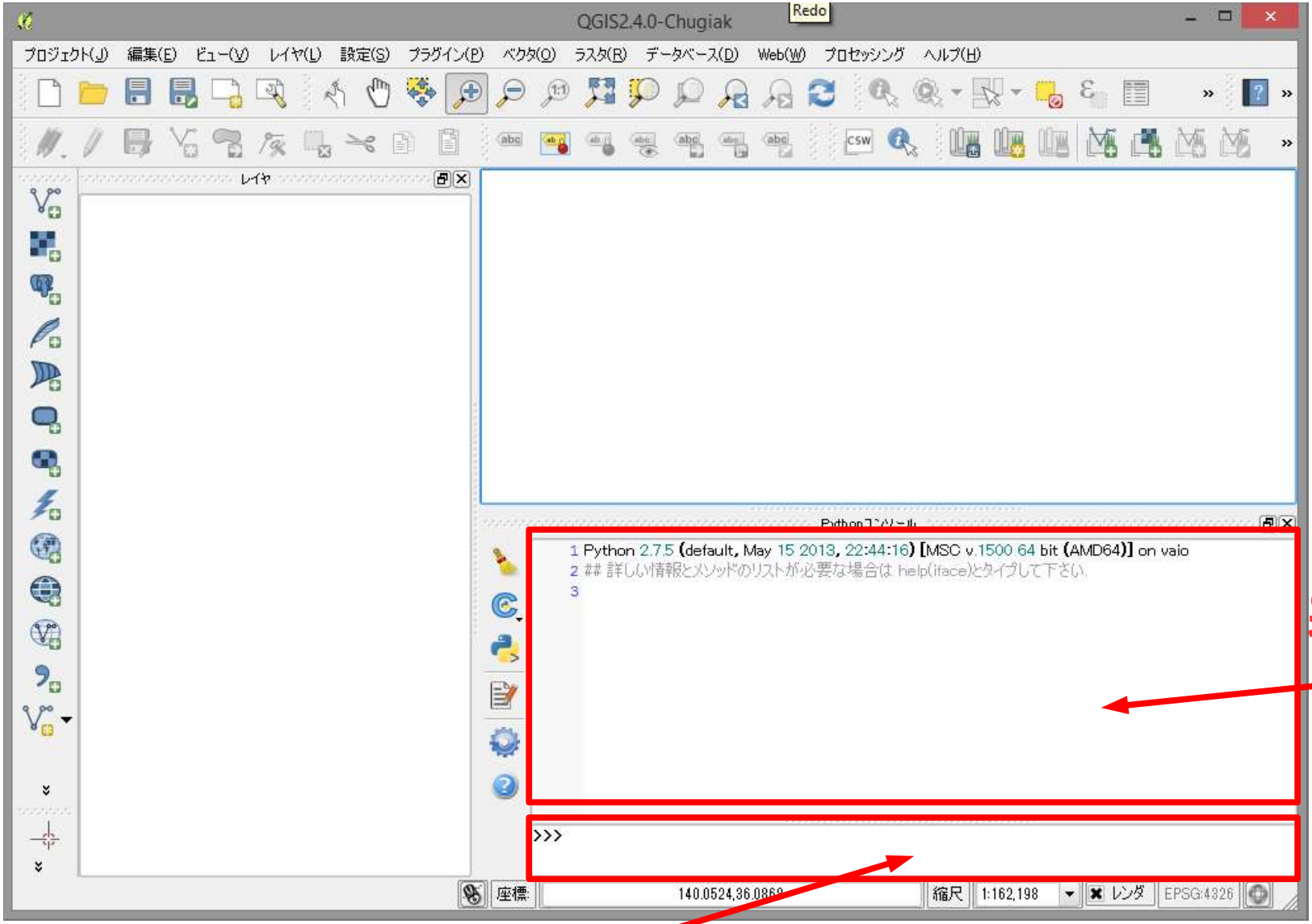


PythonからRの機能の呼び出し

Pythonコンソールとは

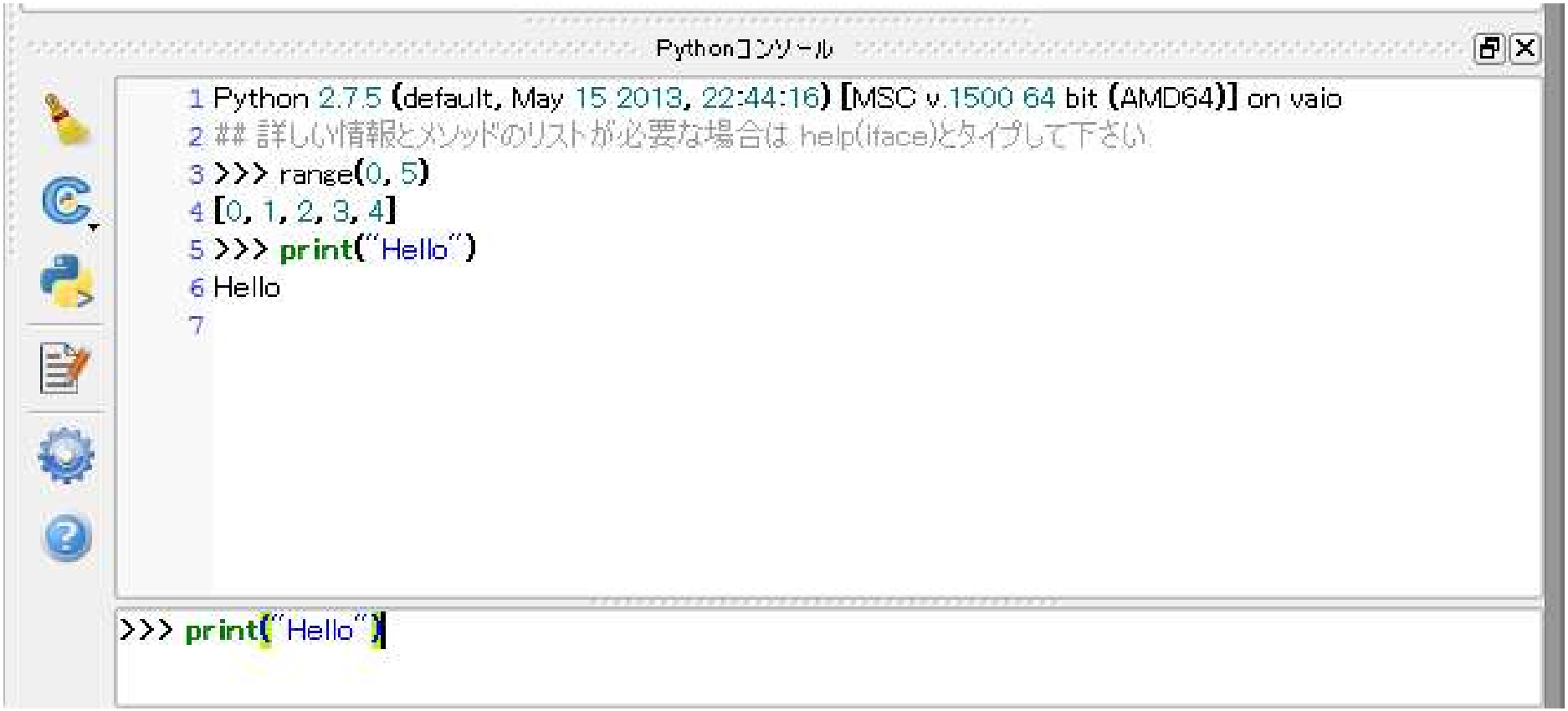
- QGIS本体の機能やプラグインの機能をPythonを用いて呼び出し、バッチ処理（自動処理）が可能
- 「プラグイン」→「Pythonコンソール」

Pythonコンソールとは



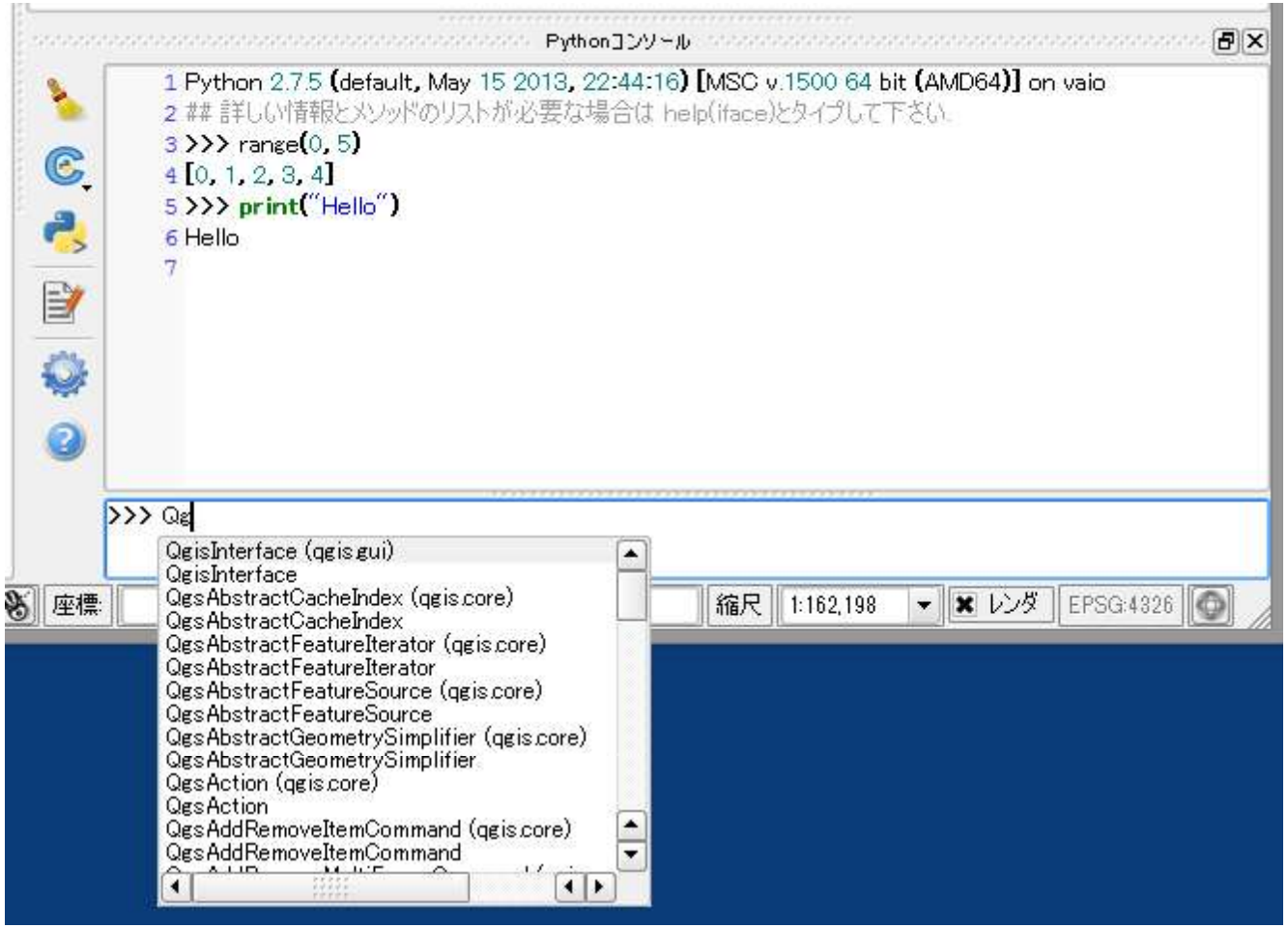
Pythonコンソールの基礎

- 入力エリアにPythonコマンドを打ち込むと、表示エリアに結果が出力される。



Pythonコンソールの基礎

- コマンド名を打ち込む途中でコマンド候補を表示
-



Pythonの基礎（関数）

- Pythonの機能呼び出しは、基本的に関数名（引数1, 引数2,...）

  例．print(“Hello”)や、range(0,5)など

- QGIS独自のPython関数も同様。

  例．QgsVectorLayer(“C:/test.shp”, “test”, “ogr”)

Pythonの基礎（モジュール）

- Pythonでは基本的な関数はそのまま使えるが、様々な拡張モジュールを読み込むことでさらに多くの関数が使えるようになる。
- 例．os、sys

- QGISの基本的なPython関数は、qgis.coreモジュールに多く含まれる。
  ※自動的にインポートされる。

Pythonの基礎（モジュール）

- モジュールのインポート

```
import qgis.core
```

GISデータ読み込み

- ベクターデータ読み込み

  QgsVectorLayer関数を使用してvlayerという名前の変数を作成する場合

```
input_dir=”C:/GIS_DATA/Advance2”
vlayer=QgsVectorLayer(input_dir + “/tsukuba_osm_lines.shp”, “tsukuba_osm_lines”, “ogr”)
```

GISデータ読み込み

- 正しく読み込めたか確認

```
vlayer.isValid()
```

- 正しく読み込めてる場合は、Trueと表示される

GISデータ読み込み

- ラスターデータ読み込み

  QgsRasterLayer関数を使用してrlayerという名前の変数を作成する場合

```
rlayer=QgsRasterLayer(input_dir + “/srtm_dem.tif”)
```

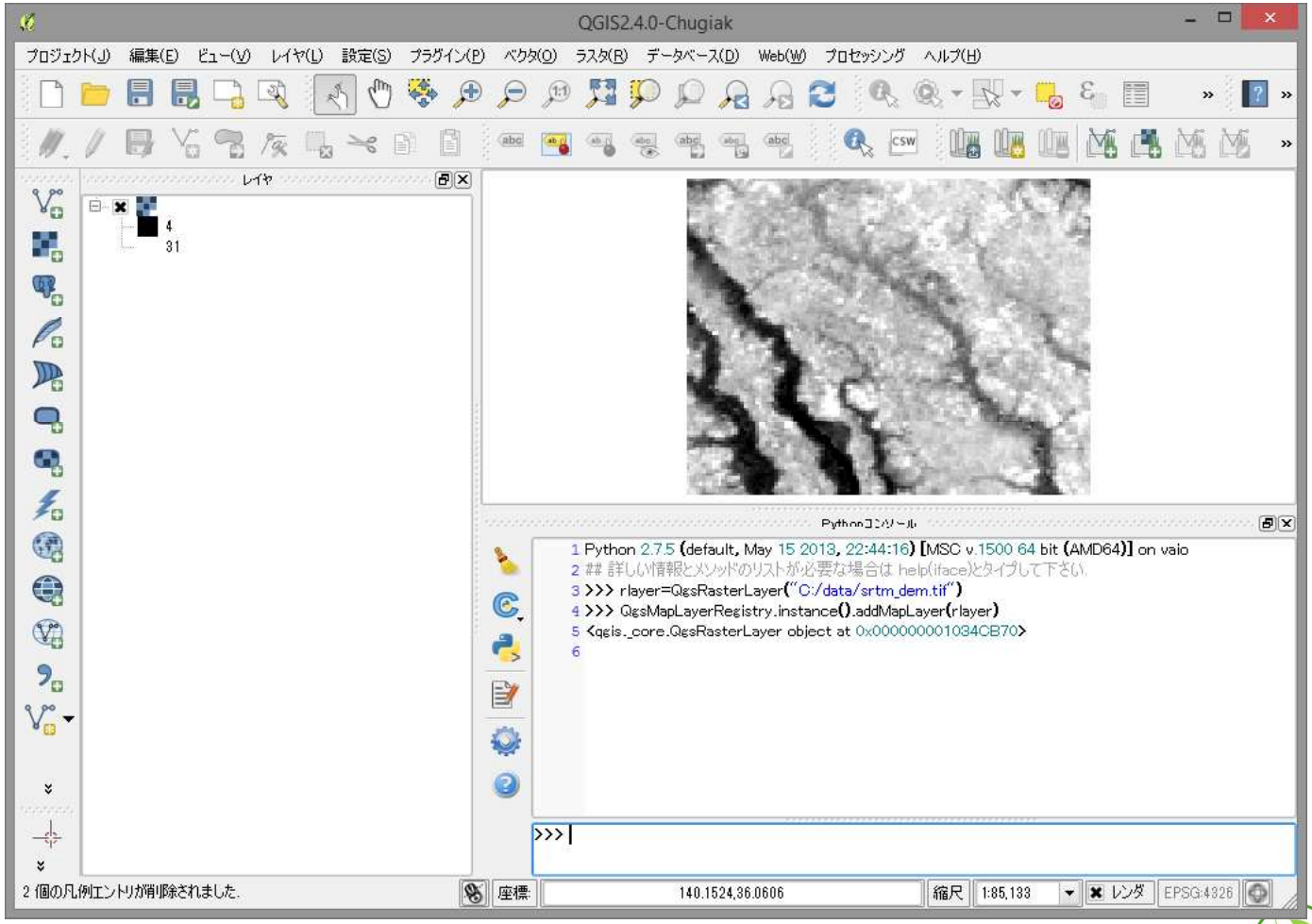
- ベクターデータ同様isValidで確認

```
rlayer.isValid()
```

GISデータの表示

- 先ほどのラスターデータ（rlayer）の表示

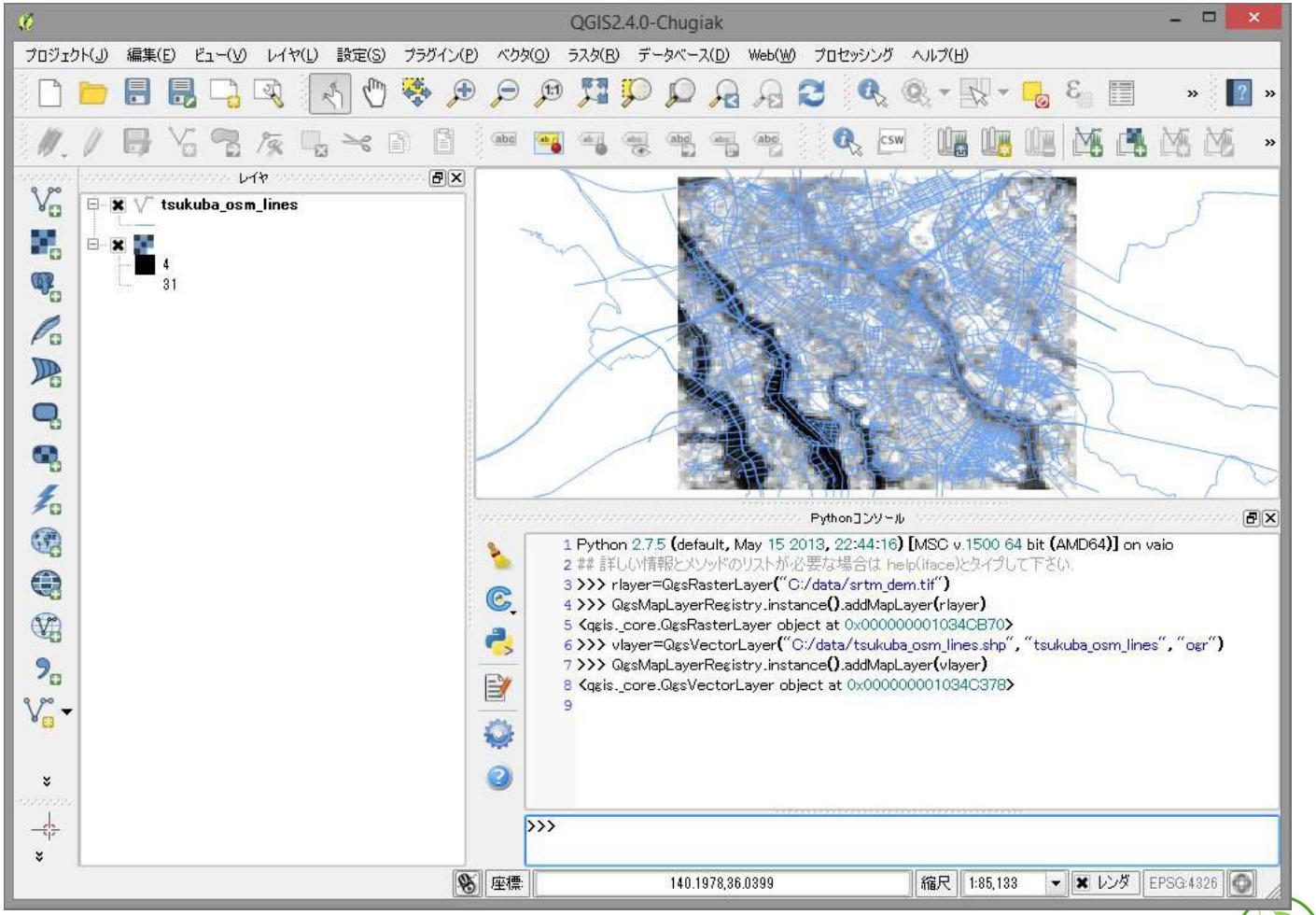
```
QgsMapLayerRegistry.instance().addMapLayer(rlayer)
```



GISデータの表示

- 先ほどのベクターデータ（vlayer）の表示

```
QgsMapLayerRegistry.instance().addMapLayer(vlayer)
```



Pythonによるバッチ処理

- データの読み込み・表示程度では、Pythonをわざわざ使うメリットは少ない。
- 最大のメリットは大量のデータを一括処理するバッチ処理など

  例．フォルダ内のデータ全てに同じ処理

`a.shp, b.shp, c.shp, d.shp`

↓ （座標系変換）

`a_utm.shp, b_utm.shp, c_utm.shp, d_utm.shp`

Pythonによるバッチ処理

- gcs2utm.txtの処理内容（1）
- 出力座標系の指定（UTM zone 54）
- フォルダ内のShapefile (*.shp)を全て読み込む

```
input_dir="C:/GIS_DATA/Advance2"

import os
import glob

exp_crs = QgsCoordinateReferenceSystem(32654,
QgsCoordinateReferenceSystem.EpsgCrsId)

files = glob.glob(os.path.join(input_dir, r'*.shp'))
```

Pythonによるバッチ処理

- gcs2utm.txtの処理内容（1）
- 正しく読み込めてるか確認するため、filesとタイプ

```
files
```

```
1 Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)] on vaio
2 ## 詳しい情報とメソッドのリストが必要な場合は help(iface)とタイプして下さい.
3 >>> #データフォルダの指定
4 >>> input_dir="C:/GIS_DATA/Advance2"
5 >>> import os
6 >>> import glob
7 >>> exp_crs = QgsCoordinateReferenceSystem(32654, QgsCoordinateReferenceSystem.EpsgCrsId)
8 >>> files = glob.glob(os.path.join(input_dir, r'*.shp'))
9 >>> files
10 ['C:/GIS_DATA/Advance2¥¥tsukuba_osm_lines.shp', 'C:/GIS_DATA/Advance2¥¥tsukuba_osm_points.shp', 'C:/GIS
   _DATA/Advance2¥¥tsukuba_osm_polygons.shp']
11
>>>
```

Pythonによるバッチ処理

- gcs2utm.txtの処理内容（2）
- forループで **files** 配列のデータを
  １つずつ **file** 変数に読み込み

```
for file in files:
    print("Input: "+ file)
```

↑この2行だけ実行した場合は（※2行目以降はタブが必要）

```
IS_DATA/Advance2¥¥tsukuba_osm_polygons.shp']
>>> for file in files:
...     print("Input: "+ file)
Input: C:/GIS_DATA/Advance2¥tsukuba_osm_lines.shp
Input: C:/GIS_DATA/Advance2¥tsukuba_osm_points.shp
Input: C:/GIS_DATA/Advance2¥tsukuba_osm_polygons.shp

>>>
```

Pythonによるバッチ処理

- gcs2utm.txtの処理内容（3）
- forループ内で文字列処理
  （出力ファイルとレイヤー名）
- 出力ファイル名の表示

```
out_file=file.replace(".shp", "_utm.shp")
layer_name=os.path.basename(file).replace(".shp", "")

print("Output: "+ out_file)
```

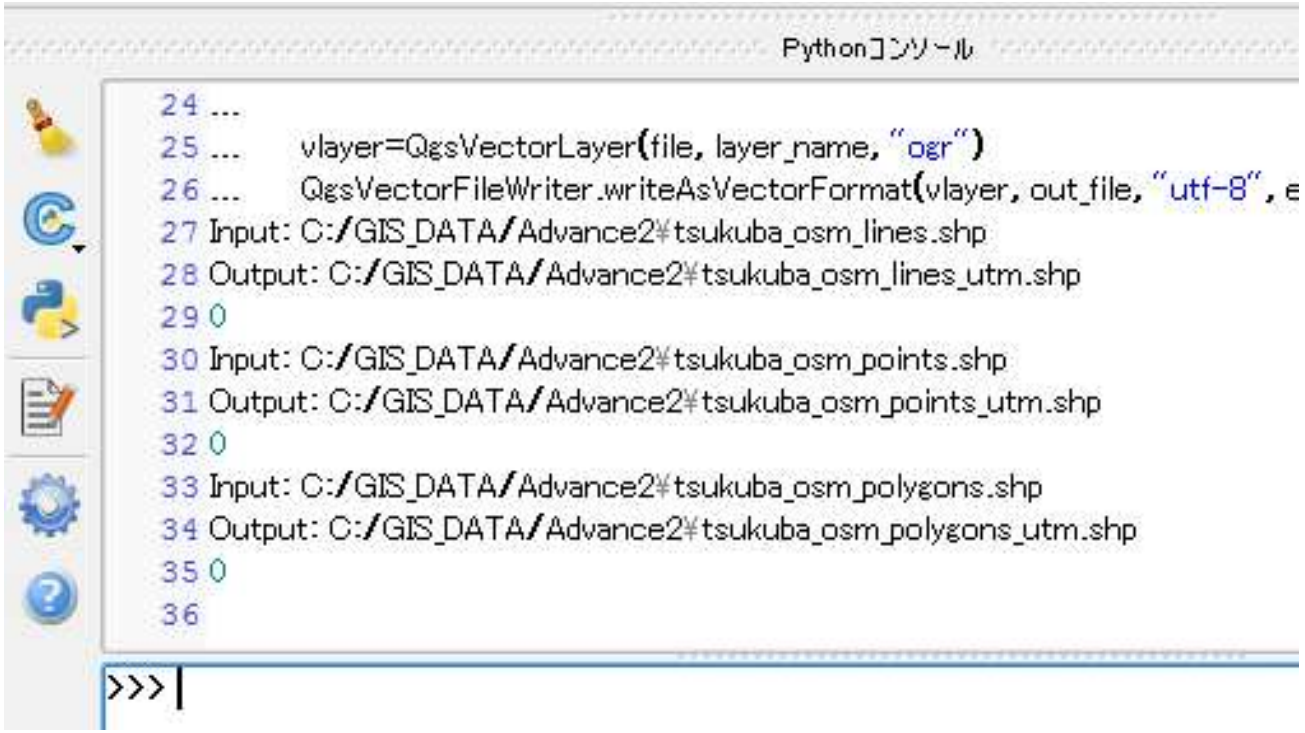
Pythonによるバッチ処理

- gcs2utm.txtの処理内容（4）
- forループ内で１つずつ読み込み、
  座標変換して保存

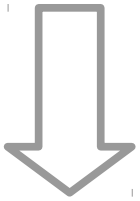
```
vlayer=QgsVectorLayer(file, layer_name, "ogr")
QgsVectorFileWriter.writeAsVectorFormat(vlayer, out_file,
"utf-8", exp_crs, "ESRI Shapefile")
```

Pythonによるバッチ処理

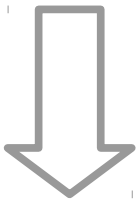
- gcs2utm.txtの実行後
  以下のように画面表示され、UTM座標系の
  Shapefileが出力される
  （※出力先の書き込み権限に注意）



Pythonコンソールの利用



PythonからRの機能の呼び出し



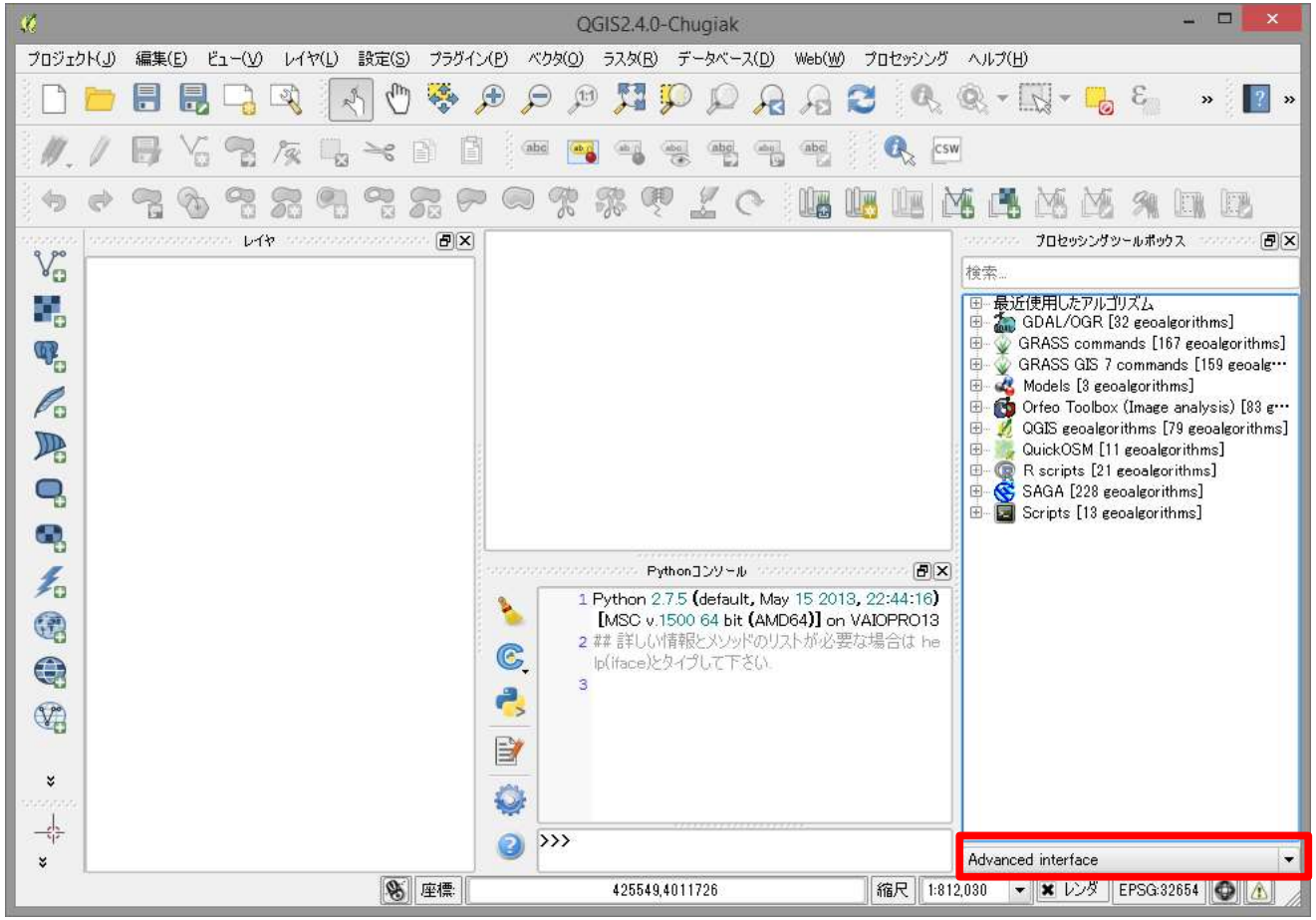
実践的な分析

プロセッシングによる外部プログラムとの連携

- QGIS 2より導入された「プロセッシング」を利用すると外部プログラム（R、GRASSなど）を簡単に呼び出す事が可能
- 「プロセッシング」→「ツールボックス」

プロセッシング

- 「Simplified interface」を「Advanced interface」に変更すると、様々なプログラムが表示される

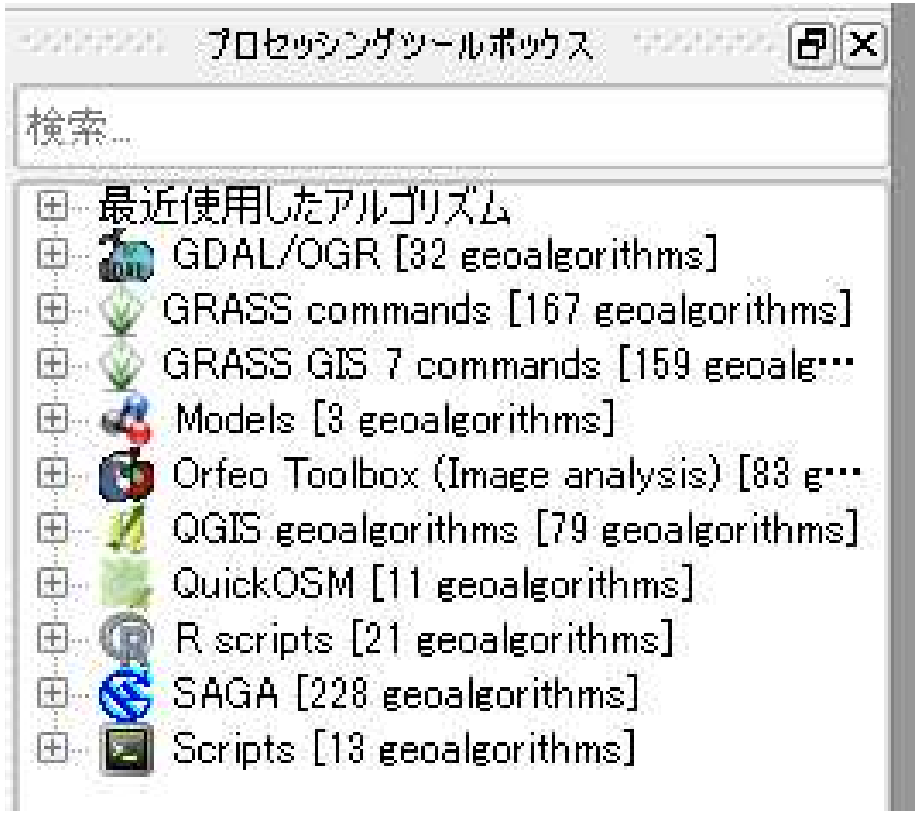


外部プログラム群

ここを変更

プロセッシング

- これらの外部プログラムはGUIだけでなく、Pythonからでも利用可能なため、バッチ処理が可能。
- 自作プログラムの追加もできる

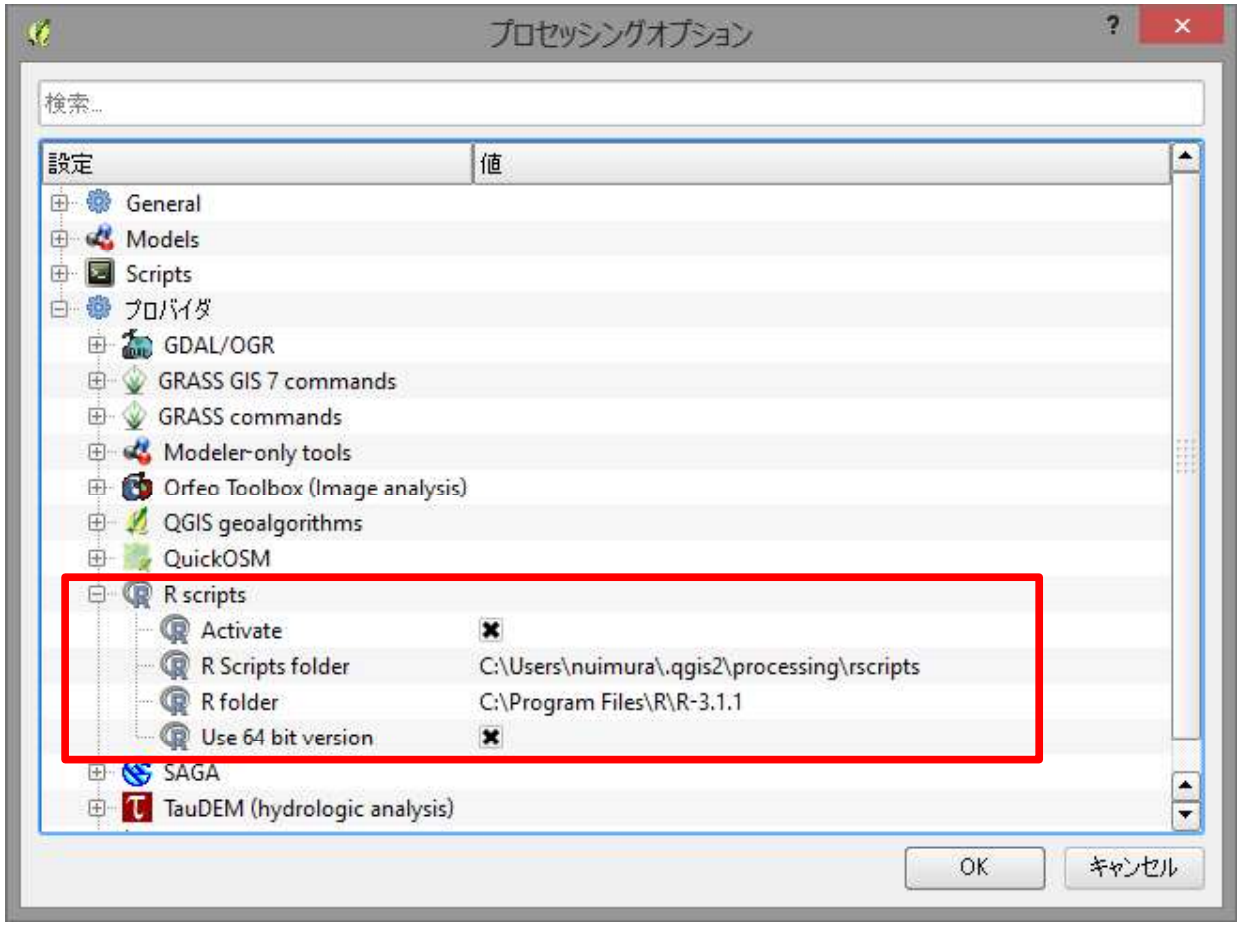


Rの初期設定

- 「プロセッシング」→「オプションと構成」

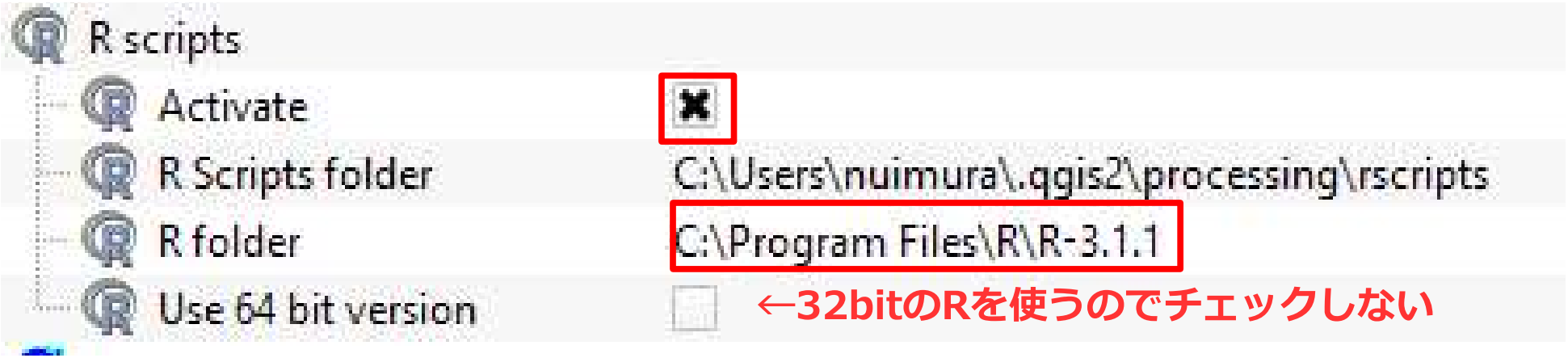
Rの初期設定

- 「プロバイダ」→「R scripts」を開く



Rの初期設定

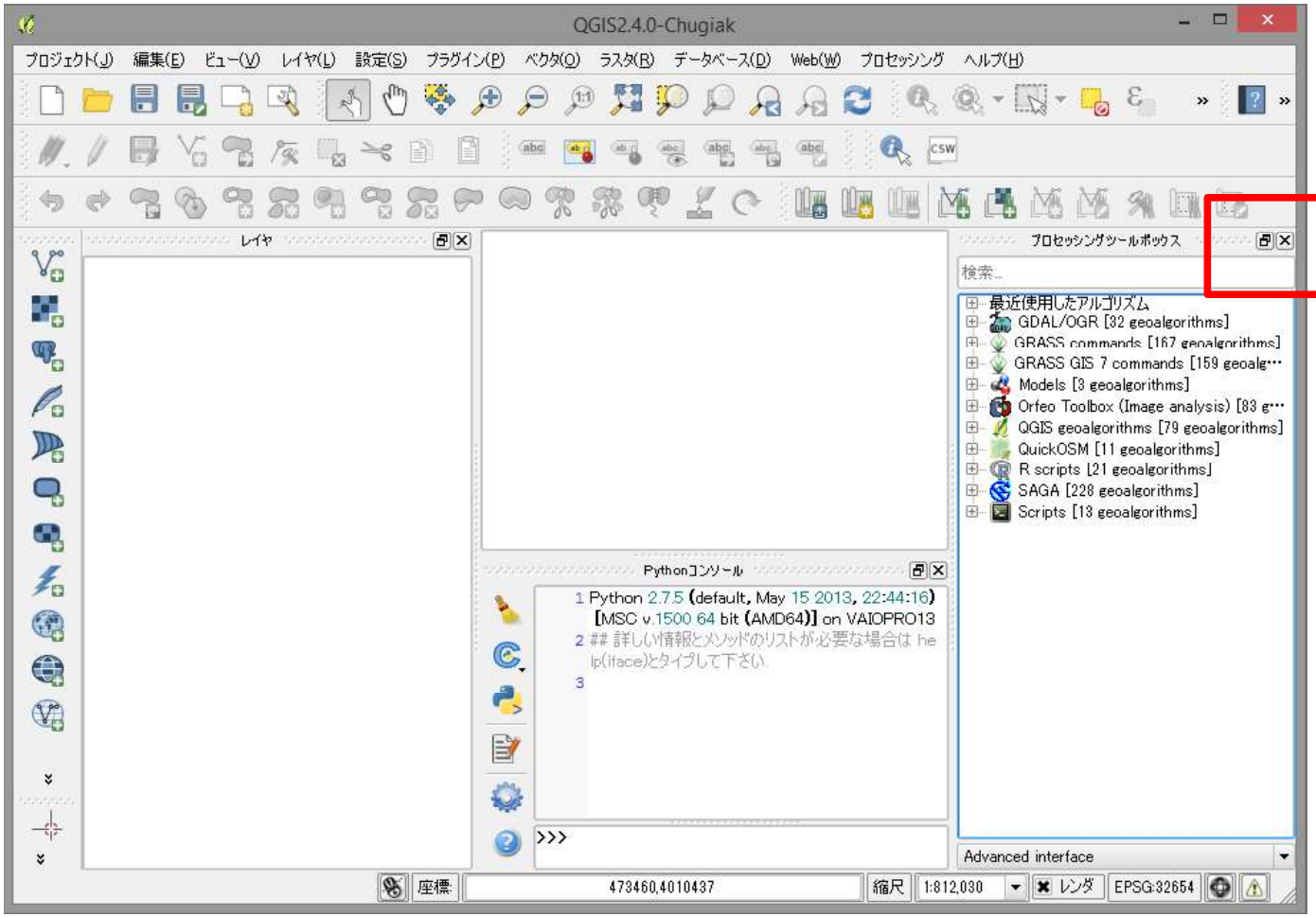
- 「Activate」にチェック
- 「R folder」にR本体のパスを設定
  （下の図はR-3.1.1の場合）



これで有効化は完了

Rの初期設定

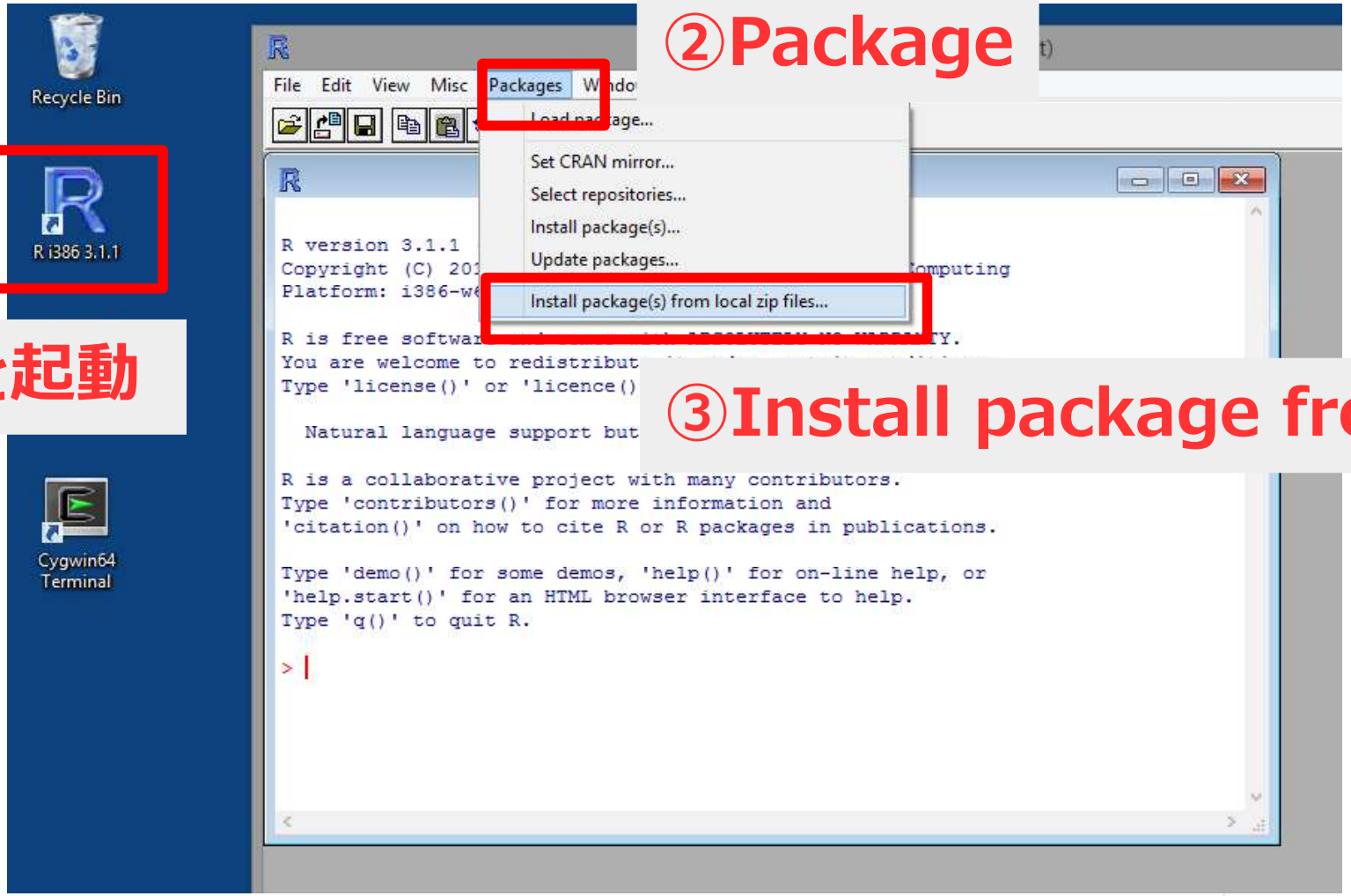
- 今回はGUIではなくPythonから呼び出すので
  プロセッシングウィンドウは非表示にします



×をクリック

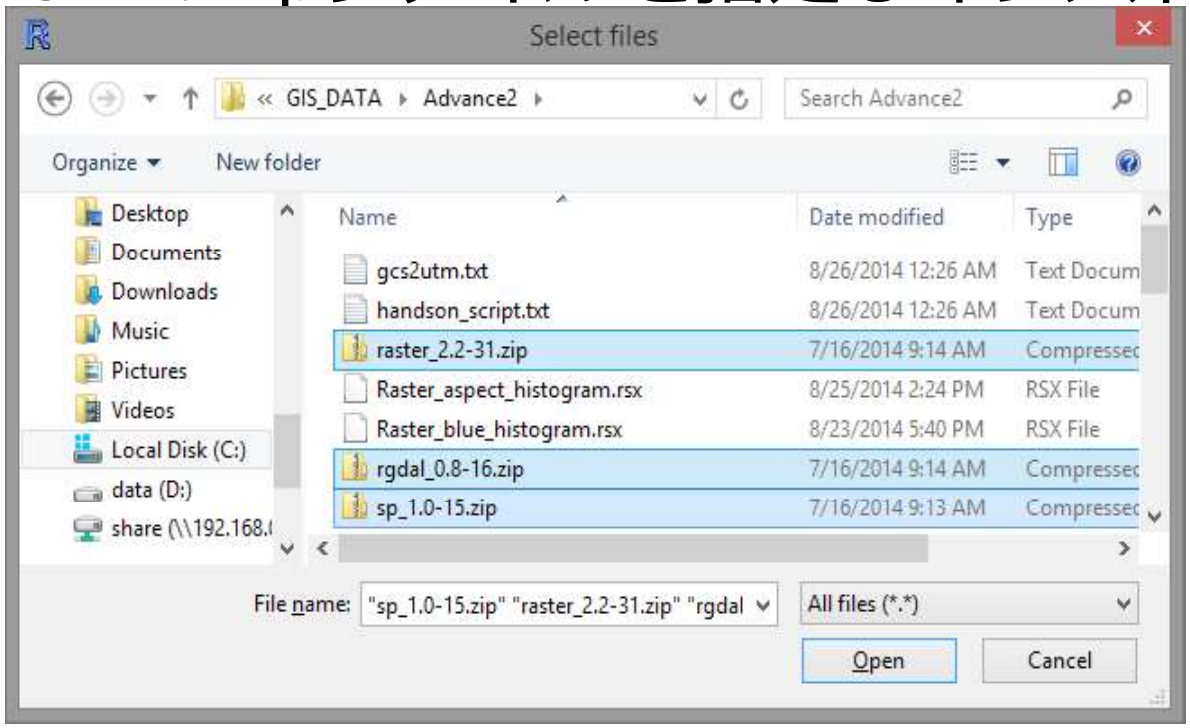
Rの初期設定

- Rを起動して関連ライブラリをインストール



Rの初期設定

- Advance2フォルダにある、raster～、rgdal～、sp～という3つのzipファイルを指定しインストール



- ライブラリを個人フォルダにインストールするか聞いてくるので許可をし、完了後はRを閉じる。

プロセッシングモジュール

- まずはPythonからプロセッシングの機能を呼び出せるようにするため、processing モジュールをインポート

```
import processing
```

プロセッシングモジュール

- 以下のコマンドでプロセッシングで使えるアルゴリズムのリスト表示ができる

```
processing.alglist()
```

いくつか抜粋すると

Clip---------->qgis:clip　QGIS本体のクリップ機能

Histogram------>r:histogram　Rのヒストグラム

v.voronoi------>grass:v.voronoi　GRASSのボロノイ作成機能

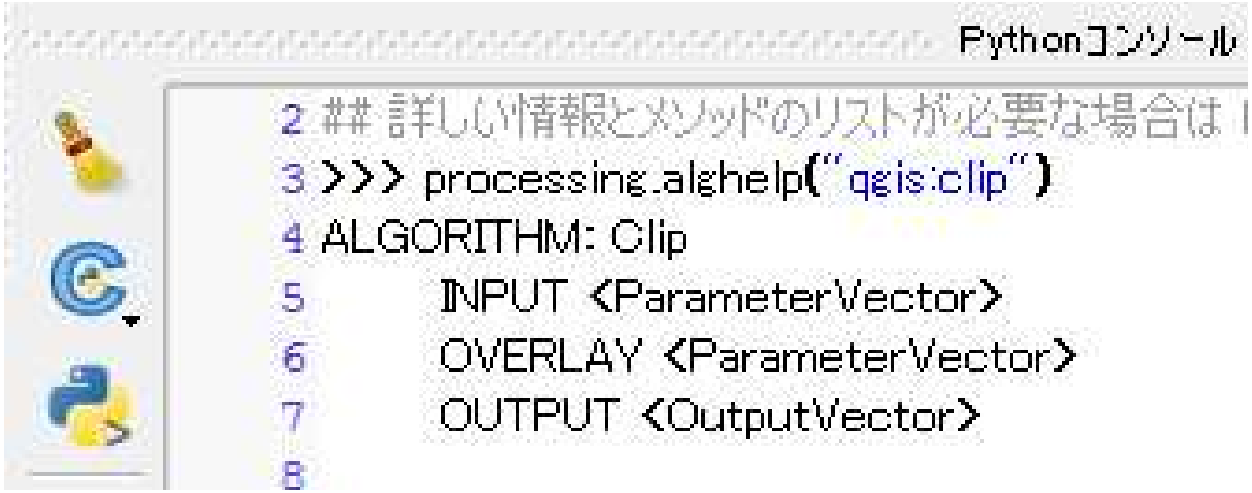
プロセッシングモジュール

- 以下のコマンドでアルゴリズムのヘルプ表示

processing.alghelp()

例えば

processing.alghelp(“qgis:clip”)



アルゴリズムの呼び出し

- 以下のコマンドでアルゴリズムを実行

```
processing.runalg()
```

以下のように使用

```
processing.runalg("アルゴリズム名", 引数1, 引数2)
```

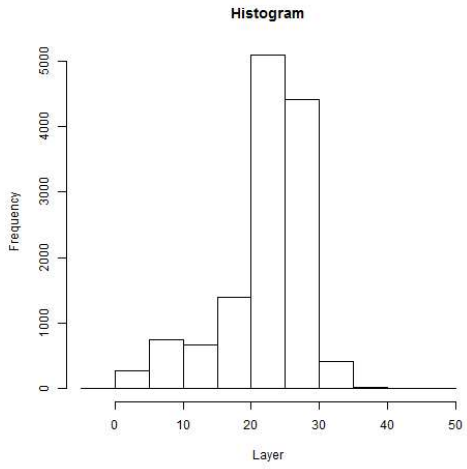
Rの機能の呼び出し

- さきほど読み込んだラスターデータ（DEM）の標高のヒストグラムの作成の場合

  r:rasterhistogramを使用

```
processing.runalg("r:rasterhistogram", rlayer, input_dir + "/graph")
```

input_dirフォルダ内に、graph.htmlとgraph.html.pngが出力



自作のRプログラムの呼び出し

- 自作プログラムを使いたい場合は

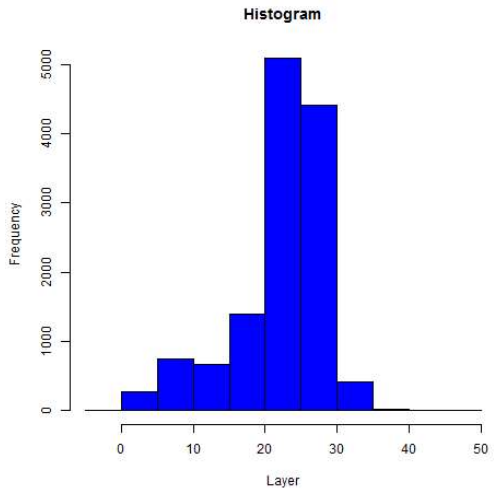
  C:¥Users¥[ユーザー名]¥.qgis2¥processing¥rscripts

  に、拡張子がrsxのRスクリプトファイルを作成し置く必要がある

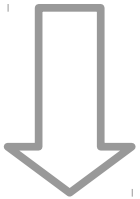
- **Raster_blue_histogram.rsx**をコピーし、プロセッシングオプションを開いてOKをクリックすることで自作プログラムを読み込み、

  以下のコマンドを実行

```
processing.runalg("r:rasterbluehistogram", rlayer, input_dir
+ "/graph2")
```



PythonからRの機能の呼び出し



実践的な分析

実践的な分析

- PythonからQGIS・Rの機能を使い解析・可視化
  - ラスターデータ
    - DEMから様々な地形指標の解析（傾斜、方位など）
    - 計算結果をRで可視化
  - ベクターデータ
    - OpenStreetMapデータでジオメトリ計算
    - 計算結果をRで可視化

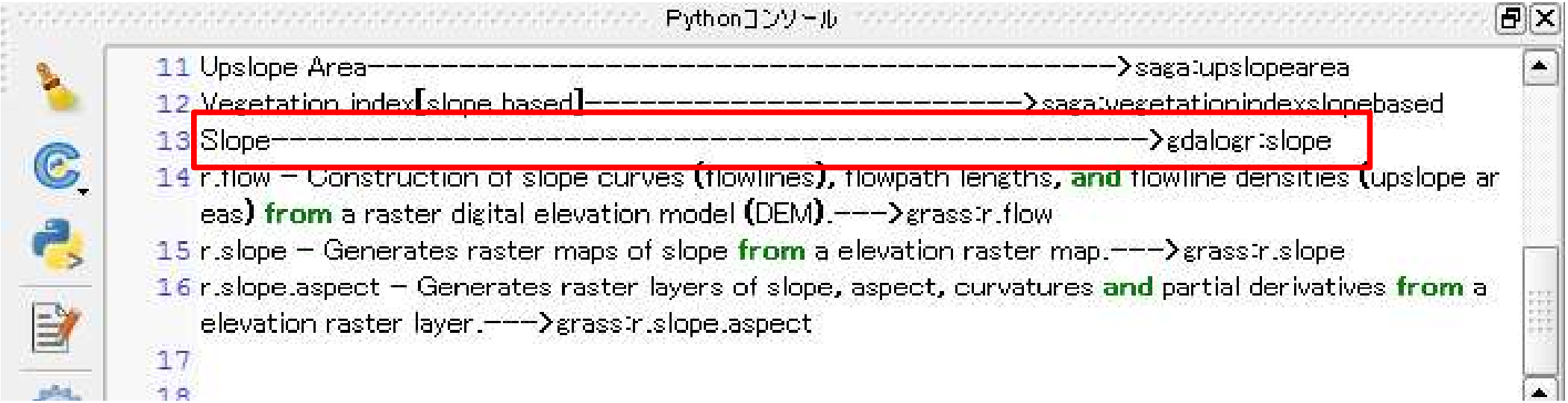
DEMから様々な地形指標の解析

- まずはprocessingモジュールで”slope”というキーワードでアルゴリズムを検索

```
processing.alglist("slope")
```

DEMから様々な地形指標の解析

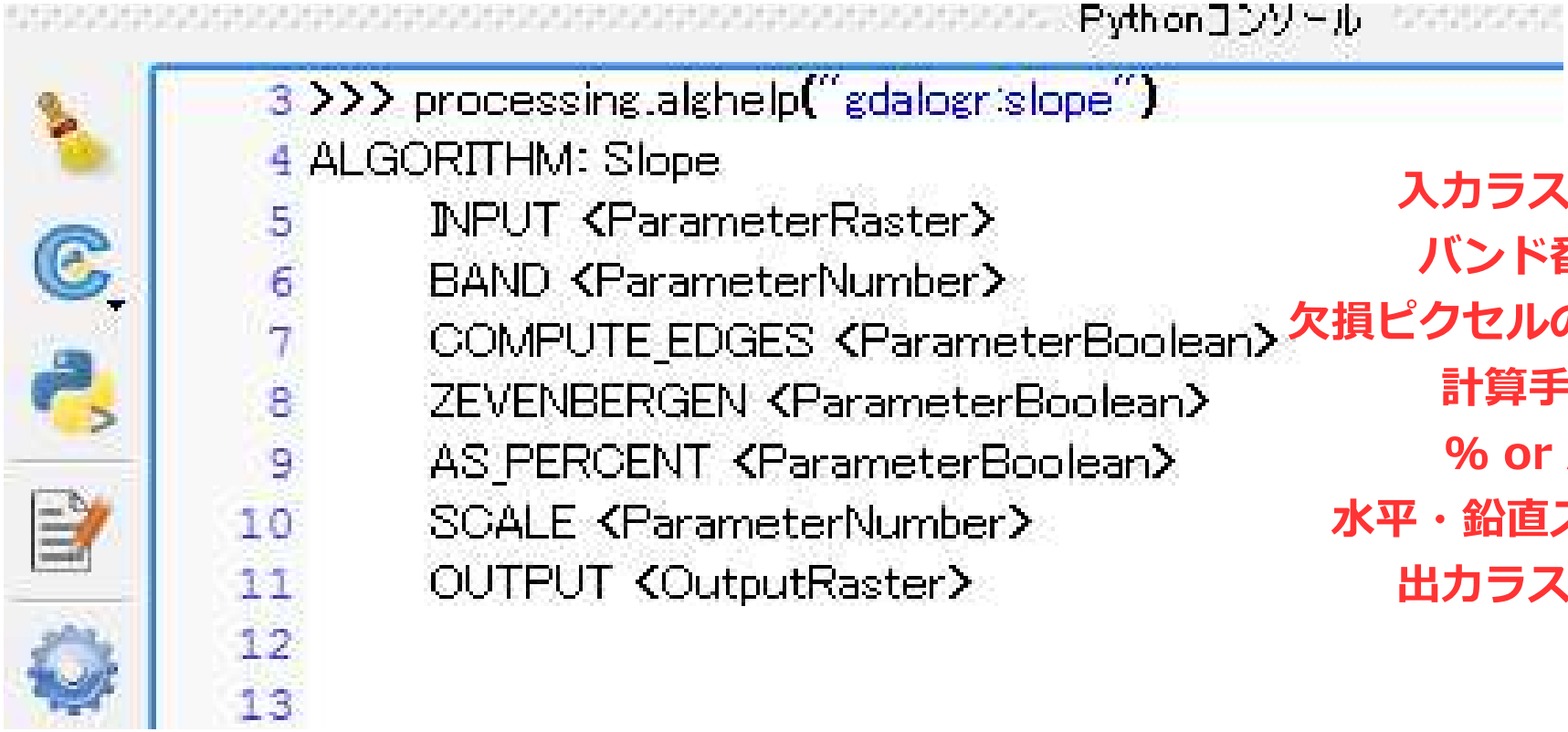
- 多くのアルゴリズムがリストアップされますが、今回は、gdalogr:slope を使用



DEMから様々な地形指標の解析

- gdalogr:slopeのヘルプを表示

```
processing.alghelp("gdalogr:slope")
```



※地理座標系は、水平方向の単位（経緯度）と鉛直方向の単位（m）が異なるので、1度におけるメートルを指定

DEMから様々な地形指標の解析

- 以下のように実行

```
processing.runalg("gdalogr:slope", rlayer, 1, 0, 1, 0,
111000, input_dir + "/slope.tif")
```

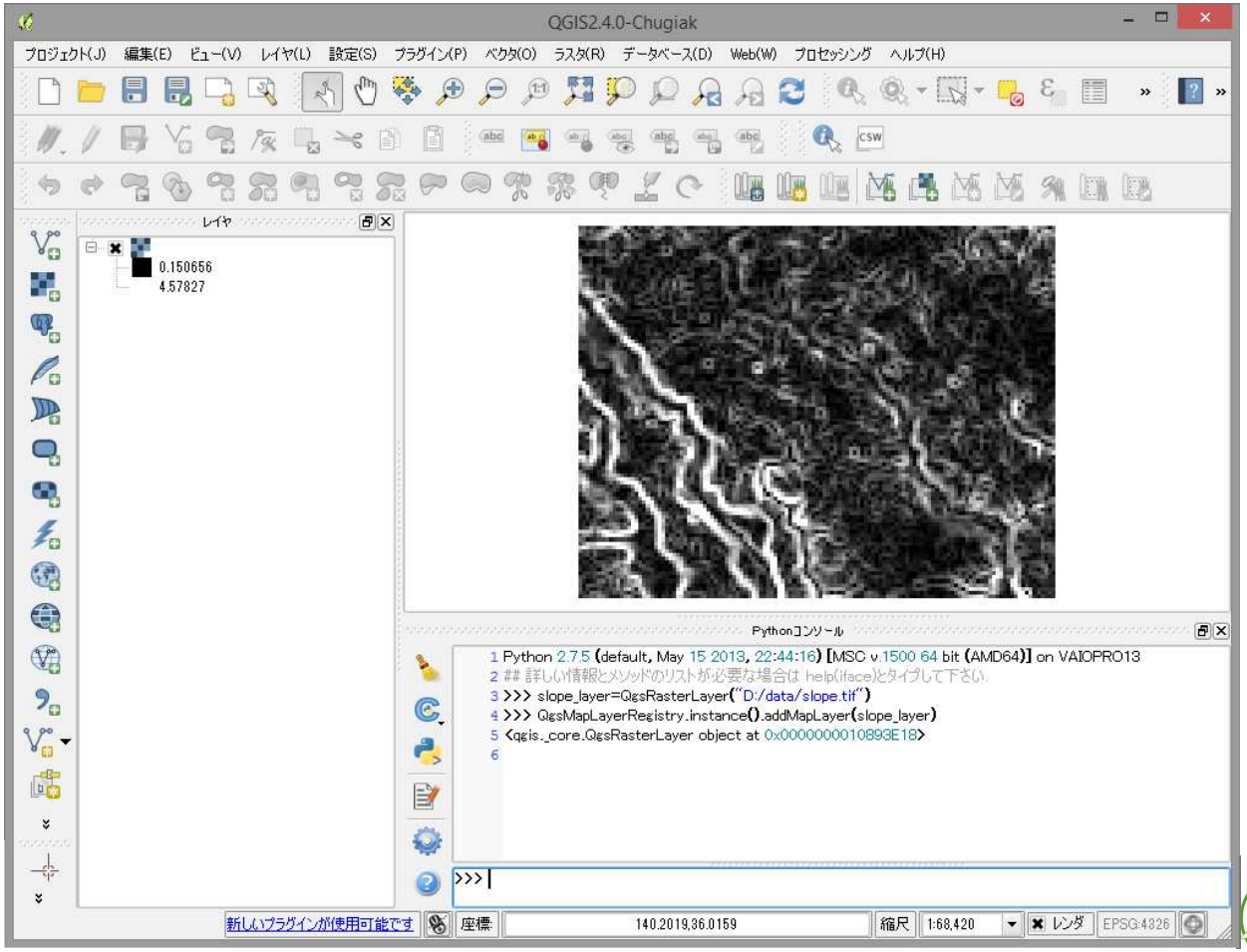
入力ラスター： rlayer
バンド番号： 1
欠損ピクセルの縁の計算： 0（しないという意味）
計算手法： 1（ZevenbergenThorne法）
% or 度： 0（度）
水平・鉛直スケール： 111000（地理座標系なので）
出力ラスター：
"C:/GIS_DATA/Advance2/slope.tif"

DEMから様々な地形指標の解析

- 計算した傾斜ラスターの読み込み

```
slope_layer=QgsRasterLayer(input_dir + "/slope.tif")
```

- `QgsMapLayerRegistry.instance().addMapLayer(slope_layer)`



OSGeo.JP

DEMから様々な地形指標の解析

- 続いて同様にprocessingモジュールで”aspect”というキーワードでアルゴリズムを検索

```
processing.alglist("aspect")
```

DEMから様々な地形指標の解析

- 今回は、gdalogr:aspect を使用

```
elevation raster layer.--->grass7:r.slope.aspect
7 Slope, aspect, curvature---------->saga:slopeaspectcurvatur
8 Aspect---------->gdalogr:aspect
9 r.aspect - Generates raster maps of aspect from a elevation raster map.--->grass:r.aspec
10 r.plane - Creates raster plane layer given dip (inclination), aspect (azimuth) and one point.-
s:r.plane
11 r.slope.aspect - Generates raster layers of slope, aspect, curvatures and partial derivatives
elevation raster layer.--->grass:r.slope.aspect
12
13
```

DEMから様々な地形指標の解析

- gdalogr:aspectのヘルプを表示

```
processing.alghelp("gdalogr:aspect")
```

```
>>> processing.alghelp("gdalogr:aspect")
ALGORITHM: Aspect
        INPUT <ParameterRaster>
        BAND <ParameterNumber>
        COMPUTE_EDGES <ParameterBoolean>
        ZEVENBERGEN <ParameterBoolean>
        TRIG_ANGLE <ParameterBoolean>
        ZERO_FLAT <ParameterBoolean>
        OUTPUT <OutputRaster>

```

入力ラスター
バンド番号
欠損ピクセルの縁の計算
計算手法
方位の基準（東 or 北）
平坦地で0 or -9999
出力ラスター

DEMから様々な地形指標の解析

- 以下のように実行

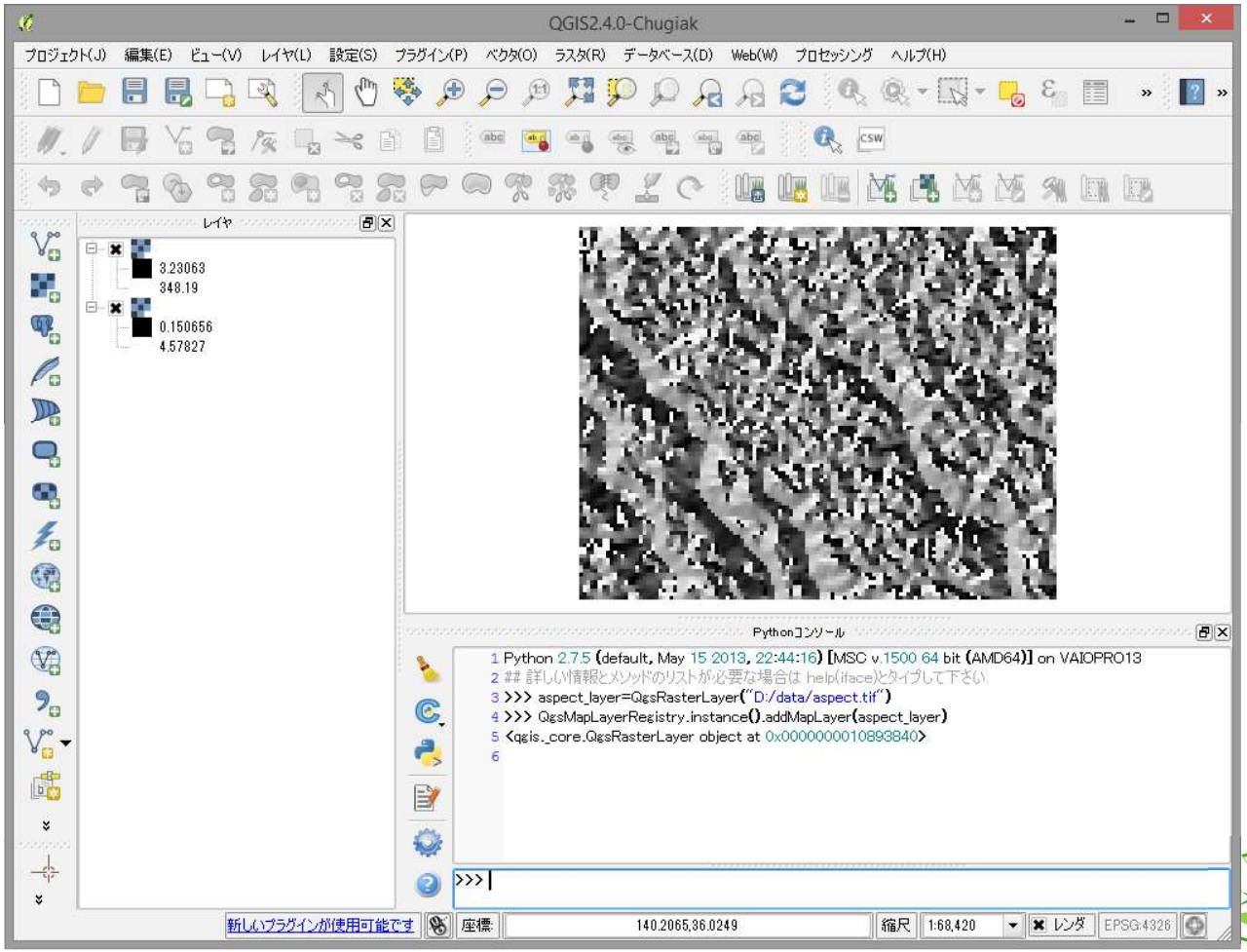
```
processing.runalg("gdalogr:aspect", rlayer, 1, 0, 1, 0,
1, input_dir + "/aspect.tif")
```

入力ラスター： rlayer
バンド番号： 1
欠損ピクセルの縁の計算： 0（しないという意味）
計算手法： 1（ZevenbergenThorne法）
方位の基準： 0（北が0）
平坦地の値： 1（平坦地は0になる）
出力ラスター：
"C:/GIS_DATA/Advance2/aspect.tif"

DEMから様々な地形指標の解析

- 計算した方位ラスターの読み込み

```
aspect_layer=QgsRasterLayer(input_dir + "/aspect.tif")
QgsMapLayerRegistry.instance().addMapLayer(aspect_layer)
```



OSGeo.JP

•計算結果をRで可視化

C:¥Users¥[ユーザー名]¥.qgis2¥processing¥rscripts
に、**Raster_aspect_histogram.rsx**をコピー

- **Raster_aspect_histogram.rsx**をコピーし、プロセッシングオプションを開いてOKをクリックすることで自作プログラムを読み込む

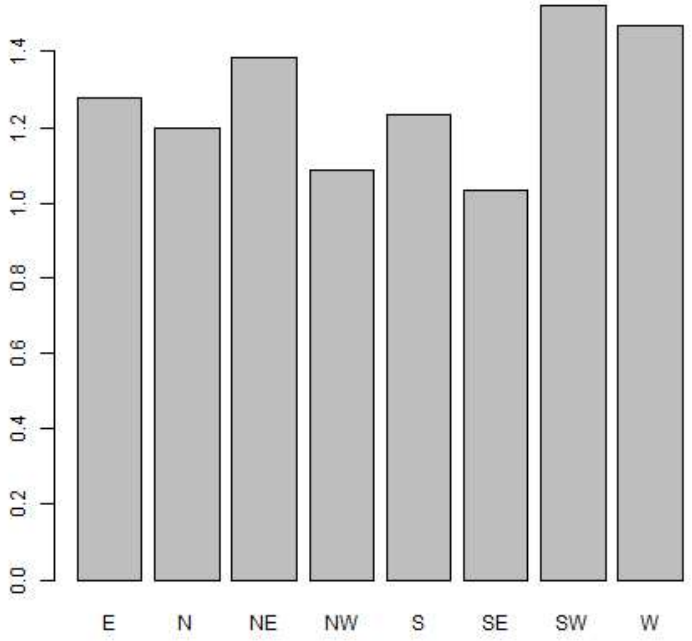
- **Raster_aspect_histogram.rsx**では、
- 斜面方位ラスターともう一つ別のラスターを読み込み、
  斜面方位別の平均値を計算
  （例．斜面方位別の平均傾斜）

•計算結果をRで可視化

以下のように実行

```
processing.runalg("r:rasteraspecthistogram", aspect_layer,
slope_layer, input_dir + "/graph3")
```

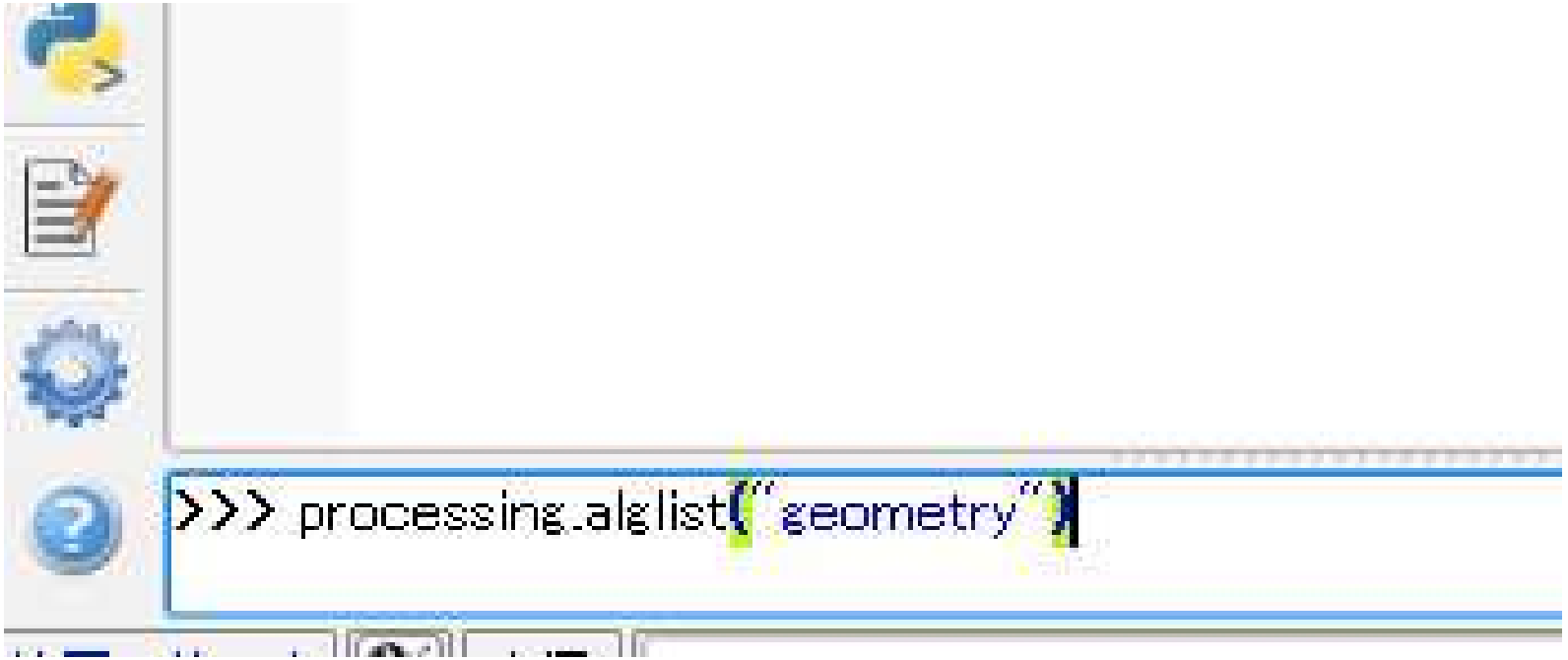
下図のように8方位別の平均傾斜が棒グラフで出力



OpenStreetMapデータでジオメトリ計算

- processingモジュールで”geometry”というキーワードでアルゴリズムを検索

```
processing.alglist("geometry")
```



OpenStreetMapデータでジオメトリ計算

- 今回は、qgis:exportaddgeometrycolumns を使用

```
>>> processing.alglist("geometry")
Convert geometry type------------------------------------>qgis:convertgeometrytype
Export/Add geometry columns------------------------------>qgis:exportaddgeometrycolumns
v.report - Reports geometry statistics for vectors.---->grass7:v.report
v.report - Reports geometry statistics for vectors.---->grass:v.report
```

OpenStreetMapデータでジオメトリ計算

- qgis:exportaddgeometrycolumnsの
  ヘルプを表示
- **processing.alghelp("qgis:exportaddgeometrycolumns")**

```
>>> processing.alghelp("qgis:exportaddgeometrycolumns")
ALGORITHM: Export/Add geometry columns
        INPUT <ParameterVector>
        CALC_METHOD <ParameterSelection>
        OUTPUT <OutputVector>


CALC_METHOD(Calculate using)
        0 - Layer CRS
        1 - Project CRS
        2 - Ellipsoidal
```

入力ベクター
計算に使う座標系
出力ベクター

OpenStreetMapデータでジオメトリ計算

- 以下のように実行

```
#まずは計算するポリゴンデータを読み込み
area_layer=QgsVectorLayer(input_dir +
"/tsukuba_osm_polygons.shp", "tsukuba_osm_polygons",
"ogr")

#ジオメトリ計算を実行
processing.runalg("qgis:exportaddgeometrycolumns",
area_layer, 0,  input_dir + "/area_geometry.shp")
```

- 属性テーブルに面積と周囲長の値の加わったarea_geometry.shpというShapefileが出力される

計算結果をRで可視化

- プロセッシングに最初から入っている
  ベクターデータ用ヒストグラムのアルゴリズム
  （r:histogram）を使用
- **processing.alghelp("r:histogram")**

```
41
42 >>> processing.alghelp("r:histogram")
43 ALGORITHM: Histogram
44         Layer <ParameterVector>
45         Field <ParameterTableField from Layer>
46         RPLOTS <OutputHTML>
47
48
```

計算結果をRで可視化

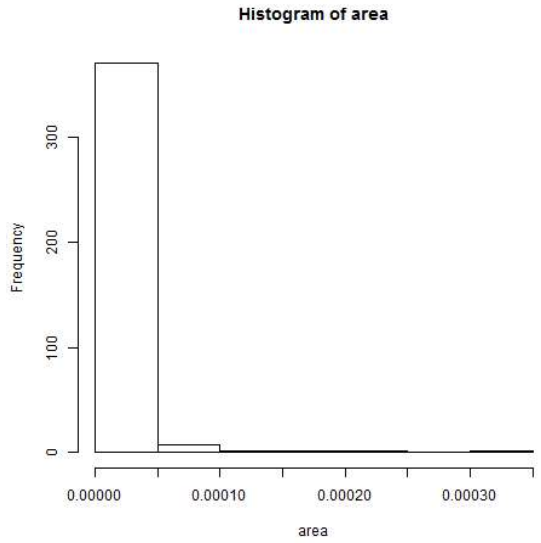
- 以下のように実行

#ジオメトリ計算したポリゴンデータを読み込み

```
geo_area_layer=QgsVectorLayer(input_dir +
"/area_geometry.shp", "area_geometry", "ogr")
```

#ヒストグラムの作成

```
processing.runalg("r:histogram", geo_area_layer,
“area”,  input_dir + "/graph4")
```



おつかれさまでした

- 本日の講習はここまでです
- ご質問等、以下のOSGeo.JPのメーリングリストでも受け付けております。
  - OSGeoJapan-discuss
    - http://lists.osgeo.org/mailman/listinfo/osgeojapan-discuss

  OSGeo.JPの会員も募集しています
    - http://www.osgeo.jp/about/support/

- FOSS4G Tokyo/Osaka/Hokkaidoイベントの開催
  - 10/25~27(Osaka)、10/31～11/2(Tokyo)

OSGeo.JP

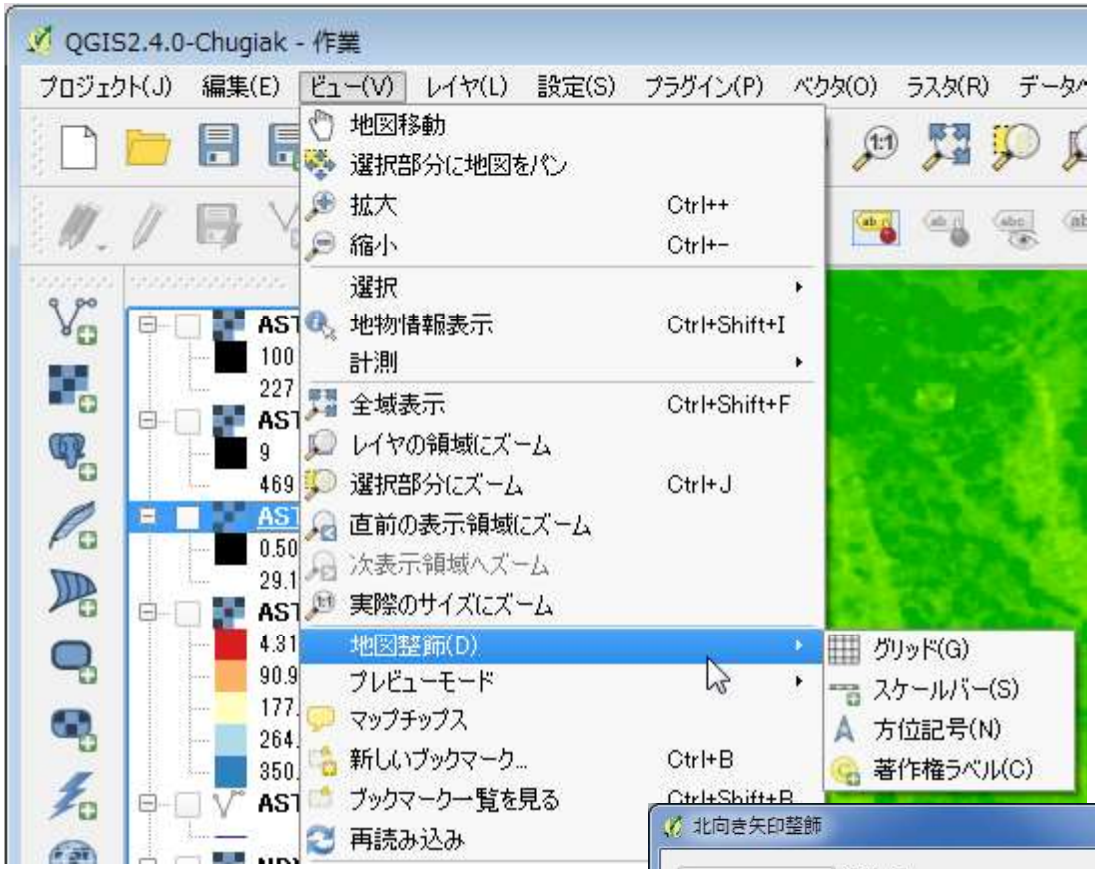
参考資料

地図を画像として保存

方位記号等の表示

- ”ビュー”→”地図装飾”から
  - グリッド
  - スケールバー
  - 方位記号
  - 著作権ラベル

がえらべる



結果を画像として保存

- ”プロジェクト”→”イメージで保存”をクリック
  - 保存先を指定
  - プレゼン等で使うには、これで十分な場合もある