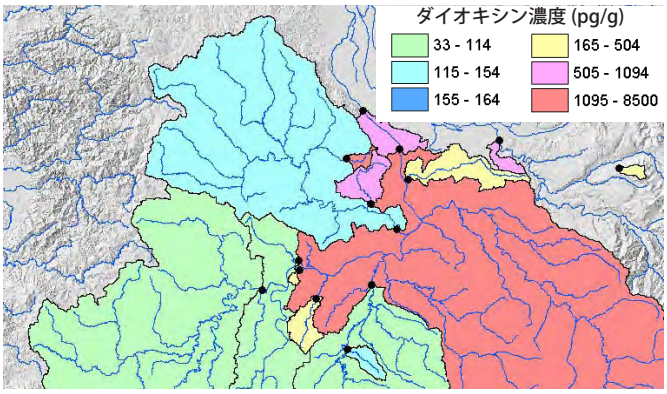
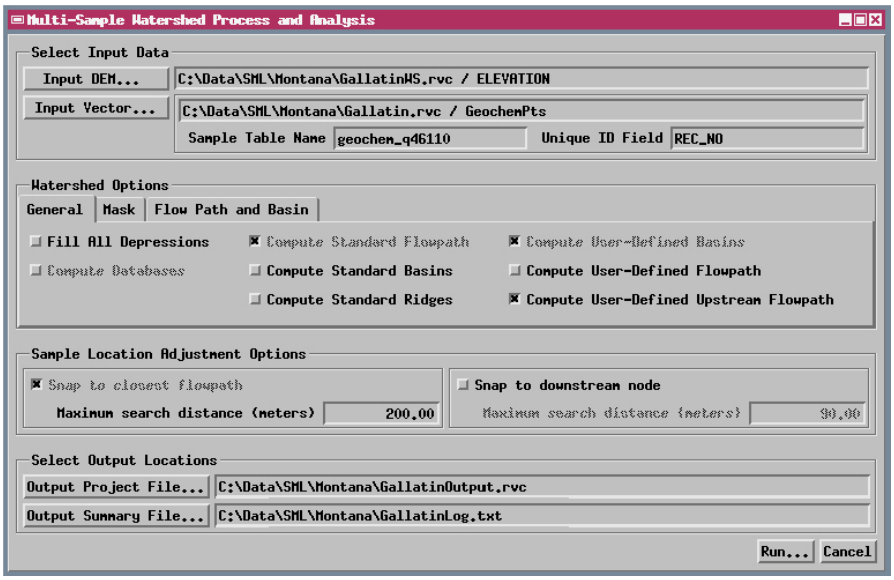
サンプルスクリプト

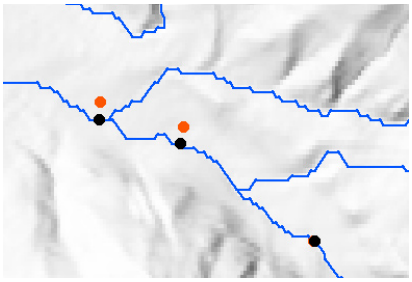
試料採取点の集水域マッピング

マイクロイメージ社が開発した SampleCatchments スクリプトでは、SML(TNT 地理空間記述言語) を使って複雑な地理空間データ処理を行い、ユーザのニーズに合った解析機能を提供します。このスクリプトではベクタオブジェクトの多数のポイントデータで表される試料採取地点での上流側集水域を明らかにするため、TNTmips の流水解析に関連する機能やデータ構造を使用します。TNTmips の流水解析の処理では、上流の集水域を計算するためシードポイントを対話的に配置することができますが、このような対話的な方法は試料採取地点が少ない場合にしか役に立ちません。SampleCatchment スクリプトでは数百数千の試料採取地点を使った場合の処理を自動化し、得られた集水域に対して属性を作成します。スクリプトによって試料採取地点毎に得られる集水域は、ポイントの位置で採取された水や河川堆積物の構成要素の原因となる地域の広がりを表しています。

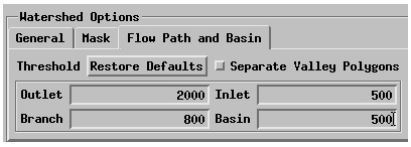


ウィラミット川下流流域 (米国オレゴン州) におけるダイオキシン汚染物質の検出レベルをモニタリングする河川堆積物試料の採取位置 (黒い点)。これらの地点に対して SampleCatchments スクリプトによって確定された上流側集水域ポリゴンをダイオキシン濃度 (サンプル試料 1 グラム当たりのダイオキシン濃度；ピコグラムで表示) により主題図表示しています。





入力の試料採取点 (オレンジ色の点) は流水解析処理で計算される流路上にないかもしれません。そのため、スクリプトでは上流の集水域を求める前に、試料採取地点を指定されたサーチ距離内で一番近い流路ライン上で最も近接した地点にくっつけています (黒い点)。



もともとの DEM か、流水解析処理で作成した窪み無しの DEM のどちらでも使用できます。後者の場合は、コンピュータの CPU に負荷が集中するステップを省略するために [全ての窪地を埋める (Fill All Depressions)] トグルボタンをオフにしてください。[流路と盆地 (Flow Path and Basin)] パネルを使ってスクリプトで作られる流路の密度と複雑度をコントロールするパラメタを設定します。多数の試料採取地点を最も効率的に処理するには、初めに流水解析処理で DEM を処理します。そこで最小限必要な密度と複雑さを持った流路ネットワークを作成する流路パラメタを決めるようにテストをしてください。

このスクリプトによって作られるベクタオブジェクトと関連するデータベースは、水質汚染研究や鉱物探査における異常なサンプル構成に関係のありそうな地域を特定するのに利用できます。

スクリプトの入力データには標高ラスタオブジェクト (DEM) と試料採取地点を含んだベクタオブジェクトを使用します。これらのオブジェクトを選択し、スクリプトによって開かれるカスタムダイアログウィンドウを使って他のパラメータを設定します。スクリプトは最も近い流路に再配置した試料採取地点を含むベクタオブジェクトと、集水域ポリゴンを含むベクタオブジェクトを出力します (このページの裏側にスクリプトの抜粋を掲載しています)。試料採取地点にアタッチされたデータベース属性は対応する集水域ポリゴンに自動的にアタッチされます。このスクリプトの別の使用例がテクニカルガイド「サンプルスクリプト：鉱石堆積物のための集水域解析 *(Sample Script: Catchment Analysis for Locating Ore Deposits)*」 に掲載されています。

TNT 製品のスクリプト言語の機能を解説する多くのサンプルスクリプトが用意されています。これらのスクリプトは www.microimages.com/downloads/scripts.htm よりダウンロードできます。

SampleCatchments.sml スクリプトの抜粋

```
string watershedparms$ = "";
if ( mainDLG.GetCtrlByID("id_FILL_ALL_DEP").GetValueNum()) {
    watershedparms$ += "FillAllDepressions,";
    }
if ( mainDLG.GetCtrlByID("id_COMP_USR_FLOWPATH").GetValueNum()) {
    watershedparms$ += "FlowPath,";
    }
watershedparms$ += "Basin";
WatershedComputeElements(watershed, x, y, numSamplePoints, watershedparms$);

if ( mainDLG.GetCtrlByID("id_COMP_USR_FLOWPATH").GetValueNum()) {
    WatershedGetObject(watershed, "VectorUserFlowPath", ObjectFilename, ObjectName);
    ObjNumber = ObjectNumber(ObjectFilename, ObjectName, "VECTOR");
    CopyObject(ObjectFilename, ObjNumber, OutputProjectFile);
    }

WatershedGetObject(watershed, "VectorUserBasin", ObjectFilename, ObjectName);
ObjNumber = ObjectNumber(ObjectFilename, ObjectName, "VECTOR");
CopyObject(ObjectFilename, ObjNumber, OutputProjectFile);


OpenVector(UBasinVECT, OutputProjectFile, "USDBASIN");
local class GEOREF UBSNgeoref = GetLastUsedGeorefObject(UBasinVECT);

local numeric numBasinsComputed = 0;
local numeric numUserBasins = NumVectorPolys(UBasinVECT);
local class DATABASE UBasinDB = OpenVectorPolyDatabase(UBasinVECT);

if (TableCopy(SampleDB, SampleTBL, UBasinDB) < 0) {
    message = "Error occured while trying to copy the sample table.\n" + "Exiting the script now... \n";
    print(message);
    fwritestring(summaryFile, message);

    CloseRaster(SampleDEM);     CloseVector(SampleVECT);
    CloseVector(WarpVECT);      CloseVector(ResultVECT);      CloseVector(UBasinVECT);
    Exit();
    }

local class DBTABLEINFO UBasinSampleTBL = DatabaseGetTableInfo(UBasinDB, sampleTblName);
UBasinSampleTBL.OneRecordPerElement = 1;

local class DBTABLEINFO UBasinPolyToPolyTBL = TableCreate(UBasinDB, PolyToPolyTblName, PolyToPolyTblDesc);

TableAddFieldString(UBasinPolyToPolyTBL, sampleIDFldName, 12);

TableAddFieldInteger(UBasinPolyToPolyTBL, numPolysPerSampleFldName, 15);

TableAddFieldFloat(UBasinPolyToPolyTBL, areaSamplePolysFldName, 15, 4);

for i = 1 to numUserBasins {
    records[1] = TableWriteRecord(UBasinPolyToPolyTBL, 0, "", 1, -1);
    TableWriteAttachment(UBasinPolyToPolyTBL, i, records, 1, "polygon");
    }

for i = 1 to numSamplePoints {
    Temp1ObjX = ResultVECT.Point[i].Internal.x;
    Temp1ObjY = ResultVECT.Point[i].Internal.y;
    ObjectToMap(ResultVECT, Temp1ObjX, Temp1ObjY, RSLTgeoref, Temp1MapX, Temp1MapY);

    polyNum = FindClosestPoly(UBasinVECT, Temp1MapX, Temp1MapY, UBSNgeoref);

    if (polyNum > 0) {
        TableReadAttachment(ResultSampleTBL, i, records, "point");
        sampleNum$ = TableReadFieldStr(ResultSampleTBL, sampleIDFldName, records[1]);

        TableReadAttachment(UBasinPolyToPolyTBL, polyNum, records, "polygon");
        TableWriteField(UBasinPolyToPolyTBL, records[1], sampleIDFldName, sampleNum$);
        TableWriteField(UBasinPolyToPolyTBL, records[1], numPolysPerSampleFldName, 1);
        }
    }
```

ユーザ定義の集水域を新規に計算するために新規の試料採取地点を使用します

新たに修正したポイントを使って集水域要素を再生成します

watershed 構造体から計算した盆地用のベクタオブジェクトハンドルを取得

"VectorUserBasin" をオープンし、そのジオリファレンスを取得

ユーザ定義の盆地を盆地内の適切な試料採取地点に関連付けます

オリジナルのベクタデータから試料のデータベーステーブルをコピーします

盆地ポリゴンに関連付けされた試料 ID

試料に関連付けされたベクタポリゴンの数

試料に関連付けされた全ベクタポリゴン総数

各盆地のテーブル内にレコードを作成します

オブジェクトの座標

ポリゴンが見つかった場合、試料名をポリゴンテーブルに保存します