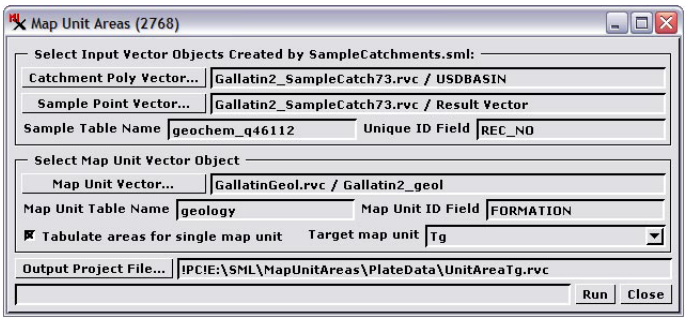
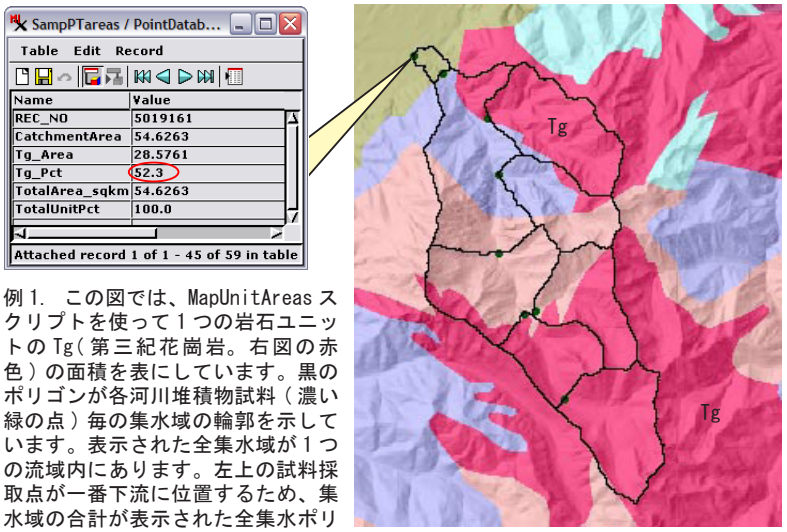
集水域下の岩石ユニットの面積計算



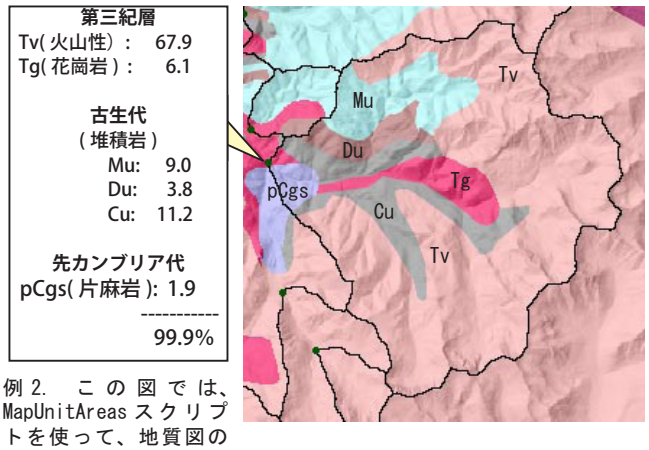
河川堆積物の試料採取や地球化学的な分析は、鉱物資源の調査において使われる重要な手法です。1 個の試料中の堆積物は上流の集水域に由来するため、地球化学的な異常がある場合にはその上流域に原因があると考えられます。同じ流域に複数の試料がある場合は、上流の小さい集水域の全てが下流で採取した堆積物に影響を及ぼします。マイクロイメージ社は Watershed 機能を使って、大量の試料の各採取点に対応する上流側の集水域を求めるカスタム地理空間処理スクリプト (SampleCatchment) を提供しています (テクニカルガイド「サンプルスクリプト：試料採取点の集水域のマッピング (Sample Script: Mapping Catchment Areas for Sample Points)」を参照)。そのスクリプトは、採取点毎に上流にあるローカルの集水域ポリゴンを生成し、それに影響を及ぼすさらに上流の集水域の数や ID 番号を記録します。



例 1.　この図では、MapUnitAreas スクリプトを使って 1 つの岩石ユニットの Tg (第三紀花崗岩。右図の赤色) の面積を表にしています。黒のポリゴンが各河川堆積物試料 (濃い緑の点) 毎の集水域の輪郭を示しています。表示された全集水域が 1 つの流域内にあります。左上の試料採取点が一番下流に位置するため、集水域の合計が表示された全集水ポリゴンの合計と一致します。従って、この試料に対する Tg ユニットの面積や面積割合 (左上の単一レコード表示に示されています) は、関連する全ての集水域ポリゴンを用いて計算されています。第三紀花崗岩は、この試料が関係する全集水域の 52.3%を占めています。

堆積物試料中に地球化学的な異常を見つけるには、影響を及ぼす領域内の岩石ユニットや土壌の知識が必要です。特に、異なる岩石や土壌ユニットの面積的な規模が、計測される化学元素のバックグラウンド値に影響を及ぼし、異常の特定に影響します。マイクロイメージ社はこの解析を補助するため、集水ポリゴンやさらに上流の集水域にある岩石や土壌ユニットの面積を一覧表示する新しいカスタム地理空間処理スクリプトを開発しました

この MapUnitAreas スクリプト (2 ページ目に抜粋を掲載) の入力は、以下の位相を持ったベクタオブジェクトです：試料の採取点、Sample Catchment スクリプトで作成した集水域ポリゴンオブジェクト、および位置的に重なるベクタの地質図や土壌図、その他です。例えば、1 つの岩石ユニットだけに限定して計算するか (例 1)、全ての岩石ユニットに対して面積を計算するか (例 2)、選べます。スクリプトは集水域ポリゴンと岩石ユニットポリゴンの論理積 (ベクタの AND) を行い、集水域ポリゴンを岩石ユニットのポリゴンで分割します。各試料採取点に対して岩石ユニットの面積 (平方キロメートル) と面積割合が再帰的に計算され、すなわち、関連する全集水域にわたって合計され、データベーステーブルに保存されます。この岩石ユニットの面積テーブルは、入力の試料採取点のベクタオブジェクトのコピーに書き込まれ、レコードが対応するポイントにアタッチされます。このテーブルは、入力の集水域ポリゴンのベクタオブジェクトのコピーにも書き込まれ、レコードが対応するポリゴンにアタッチされます。



例 2.　この図では、MapUnitAreas スクリプトを使って、地質図の全岩石ユニットに対する面積を一覧表示しています。結果の割合の数値はそれより上流に関連する集水域を持たない試料や集水域ポリゴンに対するものであり、示された表はこのポリゴンだけを対象としています。この集水域に含まれていない岩石ユニットに対する値 (ゼロ) は、上のリストでは省略されています。

www.microimages.com/downloads/scripts.htm にはダウンロード可能な多くのサンプルスクリプトがあり、スクリプトやクエリーで TNT 製品のスクリプト言語をどのように利用したらよいか解説しています。

MapUnitAreas.sml のスクリプト抜粋

function to return stringlist with single entry for each map unit identifier

```
func class STRINGLIST getMapUnitList (class DBTABLEINFO dbInfo,
      class STRING fieldName$)
   {
   for i = 1 to dbInfo.Numrecords
      {
```

read map unit identifier field inrecord

```
    if (mapUnitIdFieldTypeStr == "string")
      {
      unit$ = TableReadFieldStr(dbInfo, fieldName$, i);
      }
    else
      unit$ = NumToStr(TableReadFieldNum(dbInfo, fieldName$, i) );

    if (unit$ <> "")   if string is not empty
      {
```

if stringlist is empty, add the unit identifier to the list

```
      if (list.GetNumItems() == 0)then
        list.AddToEnd(unit$);
```

otherwise loop through the stringlist to see if unit identifier is already in list

```
      else
        {
        unitInList = 0;
        for j = 1 to list.GetNumItems()
          {
          if (list[j - 1] == unit$) then unitInList = 1;
          }

        if (unitInList == 0) then
          list.AddToEnd(unit$);
        }
      }
    }
    list.Sort();
```

sort list alphabetically

```
 return list;
  }
```

procedure to return stringlist with IDs for current catchment and its upstream catchments

```
proc getUpstreamIDs (class STRING IDstr, class STRING sampFld$,
    class DBTABLEINFO sampTbl, class DBTABLEINFO ptpTbl )
    {
```

add current sample ID to global ID List

```
  catchmentIDlist.AddToEnd(IDstr);
```

get record number in the point sample table for the current sample ID; this table is directly attached to the point elements

```
  local numeric recNum = TableKeyFieldLookup(sampTbl, sampFld$,
      IDstr);
```

get the element number of the sample point this record is attached to (function returns an array)

```
  local array numeric ptNums[1];
  TableGetRecordElementList(sampTbl, recNum, ptNums, "point");
```

get the record in the PointToPoint table that is attached to this point

```
  local array numeric recNums[1];
  TableReadAttachment(ptpTbl, ptNums[1], recNums, "point");
```

get number of upstream samples immediately adjacent to the current basin from PointToPoint table

```
  local numeric numAdjUp = TableReadFieldNum(ptpTbl,"NumUp
      Samples", recNums[1]);

  if (numAdjUp > 0)
```

if there are adjacent upstream polygons

```
    {
```

get list of sample IDs for adjacent upstream samples from UpSample[num] fields in the PolyToPoly table

```
    for j = 1 to numAdjUp
      {
```

create string for field name holding the appropriate UpSample basin ID and add to local stringlist

```
      local string field$ = fieldbase$ + NumToStr(j);
      local class STRING tempIDstr;
```

read ID from the current upstream field

```
      if (sampIdFieldTypeStr == "string") then
        tempIDstr = TableReadFieldStr(ptpTbl, field$, recNums[1] );
      else
        tempIDstr = NumToStr( TableReadFieldNum(ptpTbl, field$,
          recNums[1] ) );

      tempList.AddToEnd(tempIDstr);
      }
```

add ID as string to temporary stringlist

```
    loop through temporary list of UpSample fields to get
    their adjacent upsteram basin IDs and call this function
    recursively to check for basins further upstream

    for j = 1 to numAdjUp
      {
      getUpstreamIDs( tempList[j-1], sampFld$, sampTbl, ptpTbl);
      }
    }
  }
```