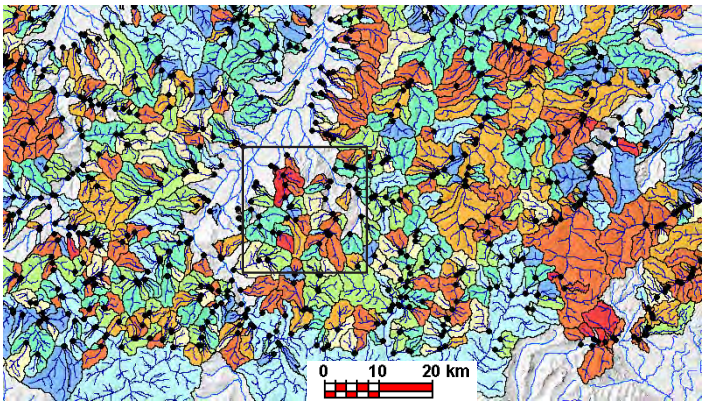
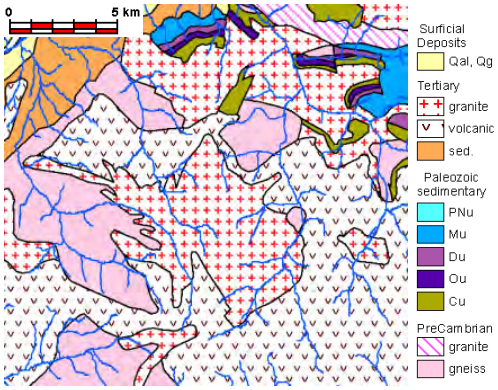
サンプルスクリプト

鉱石堆積物のための集水域解析

鉱物資源探査では河川堆積物の地球化学的な地域調査が主要な手法となっています。河川の本支流流域において適切な試料採取地の選定は、鉱床の存在を示す異常な元素濃度の検出を広域で可能にします。マイクロイメージ社が作成した SampleCatchment スクリプトは数百何千もの試料採取地点を用いて地球化学的なデータセットの地理空間的な前処理を行い、その後の地理空間的・統計的解析の入力データとして使用できます。このスクリプトは試料採取地点での上流側流域の集水域を明らかにするために数値地形モデル (DEM) を使い、試料採取地点の属性を対応する集水域ポリゴンに転写します (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マッピング *(Sample Script:Mapping Catchment Areas for Sample Points)*」を参照)。1 つの流域内の複数サンプルはサンプル毎にサブ集水域を生成し、上流と下流の集水域を決定します。サンプルの組成は上流にあるサブ集水域全ての影響を受けます。そのため、各集水域は寄与する全てのサブ集水域全域で特徴付けられます。

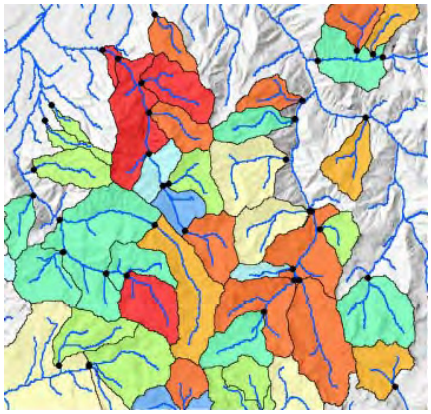


黒い点の地点で採取した地球化学的河川堆積物試料中の銅濃度 (ppm) によって表示をした主題図集水域。SampleCatchments スクリプトを使って処理した 1200 個を超える試料採取地点のある 26,000 平方キロメートルの地域の一部。四角で囲まれた部分が下の図です。

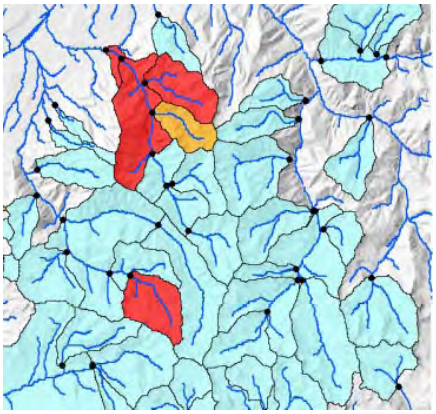


調査地域の地質図の一部

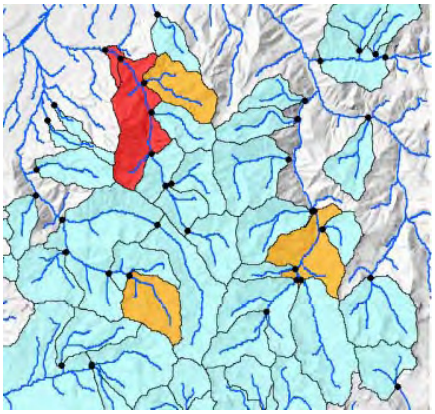
このページの図には、TNTmips のサンプル地点毎の集水域の地理空間解析が地球化学的データの理解や異常濃度の同定においてどのように使われるか示しています。銅などの各要素のバックグラウンド値は、集水域に寄与する様々な岩石の相対的比率が違うので、集水域毎に変化することが予想されます。銅のバックグラウンド値を予測するために、ポリゴンの集計 (Polygon Properties) 処理を使って集水域に地質図を重ね、各サブ集水域について面積と構成する岩体の割合を決定しました。それから、このページの裏側に抜粋を掲載している GeolUnitArea スクリプトを使って各集水域に対して寄与する集水域全てを特定し、これらの集水域の岩石構成の面積合計を出しました。その出力である銅の濃度と構成割合の表を使い、統計処理プログラム中で多重線形回帰を実行し、各集水地に対して銅濃度の予測値と残差を計算しました。TNTmips へ再読み込みした後、計算フィールドを使って残差に対して寄与する全集水域で重み付けを行い、調査地域内で異常に銅濃度が高い 2 つの集水域を示す主題図を作成しました。



測定した銅濃度による集水域の主題図
(凡例は右上の地図と同じ)



銅濃度 (ppm) の Log_{10} の残差
(測定値－予測値)
0.00 - 0.19
0.20 - 0.89
0.90 - 1.20



集水域の面積で重み付けをした
銅濃度の Log_{10} の残差
0.00 - 1.99
2.00 - 19.99
20.00 - 61.84

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。

GeolUnitArea.sml スクリプトの抜粋

```
proc getUpstreamIDs ( string sampleID$ )
   {
   local numeric j;
   local string fieldbase$ = "UpSample";

   idList.AddToEnd(sampleID$);

   local numeric recNum =
         TableKeyFieldLookup(ptTable, "REC_NO", sampleID$);

   local numeric numAdjUp =
         TableReadFieldNum(ptTable, "NumUpSamples", recNum);

   if ( numAdjUp > 0 )
      {
      local class STRINGLIST tempList;
      for j = 1 to numAdjUp
         {
         local string field$ = fieldbase$ + NumToStr(j);
         tempList.AddToEnd( TableReadFieldStr(ptTable, field$, recNum) );
         }

      for j = 1 to numAdjUp
         {
         getUpstreamIDs( tempList[j-1] );
         }
      }
   }

for i = 1 to unitAreaTbl.NumRecords
   {
   local numeric j, k, m, n;
   local string sampID$ = BasinVectOut.poly.UnitArea[@i].REC_NO$;

   local numeric recNum;
   local numeric sumBasinArea;

   local numeric cuArea, duArea, muArea;
   local numeric cuPct, duPct, muPct;

   local array numeric polyList[1];
   local numeric numAttachedPolys;

   local array numeric pctRecordList[1];
   local numeric numPctRecords;

   getUpstreamIDs( sampID$ );

流出寄与盆地のリストで地質ユニット面積を合計するループ処理
   for j = 1 to idList.GetNumItems()
      {
      recNum = TableKeyFieldLookup(basinInfoTbl, "REC_NO", idList[j-1]);

   numAttachedPolys =
   TableGetRecordElementList(basinInfoTbl, recNum, polyList);

for k = 1 to numAttachedPolys
   {
   sumBasinArea += BasinVectOut.poly[ polyList[k] ].POLYSTATS.Area;

   numPctRecords =
         TableReadAttachment(percentTbl, polyList[k], pctRecordList);

   for m = 1 to numPctRecords
      {
      n = pctRecordList[m];

      if ( BasinVectOut.poly.PERCENTAGE[@n].FORMATION$ == "Cu")
         then cuArea += BasinVectOut.poly.PERCENTAGE[@n].Area;
      else
      if ( BasinVectOut.poly.PERCENTAGE[@n].FORMATION$ == "Du")
         then duArea += BasinVectOut.poly.PERCENTAGE[@n].Area;
      else
      if ( BasinVectOut.poly.PERCENTAGE[@n].FORMATION$ == "Mu")
         then muArea += BasinVectOut.poly.PERCENTAGE[@n].Area;

      [repeat for all rock units]
      }
   }
}

cuPct = 100 * cuArea / sumBasinArea;
duPct = 100 * duArea / sumBasinArea;
muPct = 100 * muArea / sumBasinArea;
[repeat for all rock units]

totUnitPct = round( (cuPct + duPct + muPct + ouPct + qalPct + tgPct +
      tsPct + tvPct + pCgrPct + pCgsPct ) );

sumBasinArea = sumBasinArea / 1000000;
cuArea = cuArea / 1000000;
duArea = duArea / 1000000;
muArea = muArea / 1000000;
[repeat for all rock units]

TableWriteField(unitAreaTbl, i, "CheckArea", sumBasinArea);

TableWriteField(unitAreaTbl, i, "Cu_Area", cuArea);
TableWriteField(unitAreaTbl, i, "Cu_Pct", cuPct);

TableWriteField(unitAreaTbl, i, "Du_Area", duArea);
TableWriteField(unitAreaTbl, i, "Du_Pct", duPct);

TableWriteField(unitAreaTbl, i, "Mu_Area", muArea);
TableWriteField(unitAreaTbl, i, "Mu_Pct", muPct);

[repeat for all rock units]

TableWriteField(unitAreaTbl, i, "Total_Unit_Pct", totUnitPct);

sumBasinArea = 0;
cuArea = 0;    duArea = 0;    muArea = 0;    ouArea = 0;
qalArea = 0;   tgArea = 0;    tsArea = 0;    tvArea = 0;
pCgrArea = 0;  pCgsArea = 0;  totUnitPct = 0;

idList.Clear();
}
```

任意のサンプルから上流のサンプル ID を取得する再帰関数を定義します

現在のサンプル ID を ID リストに加えます

現在のサンプル ID に対する PointToPoint テーブル内のレコード番号を取得します

現在の盆地に隣接する上流のサンプル数を取得します

近接した上流ポリゴンが存在する場合は、近接した上流サンプルに対するサンプル ID のリストを PolyToPoly テーブルの UpSample[num] フィールドより取得します

適切な UpSample 盆地 ID のあるフィールド名用の文字列を生成し、stringlist に追加します

UpSample フィールドのリストをループ処理して、隣接する上流の盆地 ID を取得し、さらにその上流の盆地をチェックする処理を呼び出します

UnitArea テーブル中のレコードでループ処理を行い、寄与する領域の各地質面積を合計し、各地質に対する割合を計算し、値をテーブルに書き込みます

PolyToPoly テーブルのレコード No

盆地のポリゴン面積を合計します

合計面積と各岩石ユニットの割合を示す変数

保持しているポリゴンの要素数の配列 PolyToPoly レコードがアタッチされます

アタッチされたポリゴンの数

各ポリゴンにアタッチされた PERCENTAGE テーブルレコードのレコード数を入れる配列

上流の全ての試料 / 盆地 ID のリストを取得する再帰関数を呼び出します

PolyToPoly テーブル内の盆地レコードのレコード番号を取得します

そのレコードにアタッチされているポリゴン要素のリストを取得します

ポリゴン数でループ処理

ポリゴンにアタッチされた PERCENTAGE テーブルレコードのレコード数のリストを取得します

アタッチされているレコード分だけループ処理して合計を計算します

盆地面積の合計とユニット面積の合計を使って割合を計算します

チェックのために各盆地に対するユニットの割合を合計します

合計した面積を平方メートルから平方キロメートルに変換します

計算結果を UnitArea テーブルの適切なフィールドに書き込みます

次の処理のために合計用変数を全てクリアします

流出寄与盆地 ID 用の文字列リストをクリアします