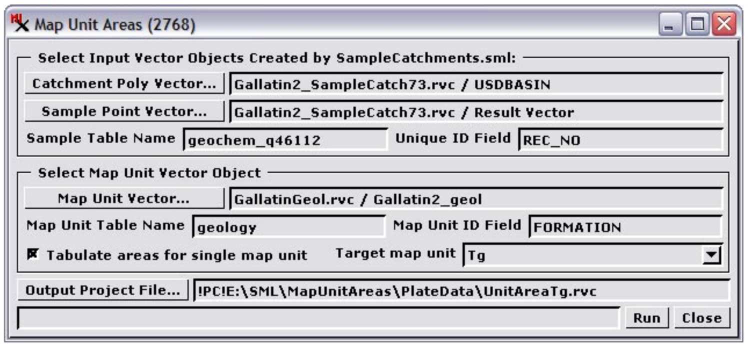
Sample Script

Compute Map Unit Areas for Catchments


Map Unit Areas (2768)

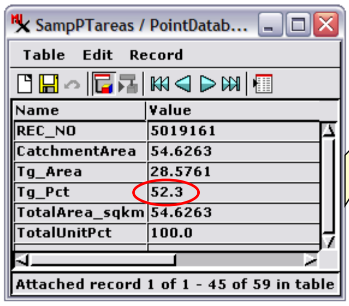
Select Input Vector Objects Created by SampleCatchments.sml:

Catchment Poly Vector...	Gallatin2_SampleCatch73.rvc / USDBASIN
Sample Point Vector...	Gallatin2_SampleCatch73.rvc / Result Vector
Sample Table Name	geochem_q46112
Unique ID Field	REC_NO

Select Map Unit Vector Object

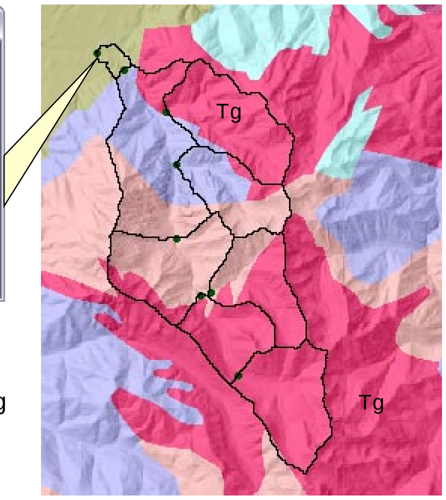
Map Unit Vector...	GallatinGeol.rvc / Gallatin2_geol
Map Unit Table Name	geology
Map Unit ID Field	FORMATION
[✓ Tabulate areas for single map unit]	
Target map unit	Tg
Output Project File...	!PC!E:\SML\MapUnitAreas\PlateData\UnitAreaTg.rvc

Sampling and geochemical analysis of stream sediment is an important tool used in exploration for mineral resources. The sediment in a single sample is sourced from the upstream catchment area, so any prospective geochemical anomalies must relate to that area. If multiple samples lie within the same drainage, all of the smaller upstream catchments contribute to the sediment sampled at the more downstream locations. MicroImages provides a custom geospatial processing script (SampleCatchments) that uses watershed functions to delineate the upstream catchment area for each point in a large sample set (see the Technical Guide entitled *Sample Script: Mapping Catchment Areas for Sample Points*). The script generates a local upstream catchment polygon for each sample and records the number and identities of any additional upstream contributing catchments.


SampPTareas / PointDatab...

Name	Value
REC_NO	5019161
CatchmentArea	54.6263
Tg_Area	28.5761
Tg_Pct	52.3
TotalArea_sqkm	54.6263
TotalUnitPct	100.0

Attached record 1 of 1 - 45 of 59 in table



In this illustration the MapUnitAreas script was used to tabulate areas for a single geologic map unit, Tg (Tertiary granite, shown in red in the map to the right). The black polygons outline catchments for individual stream sediment samples (dark green dots). All of the catchments shown lie within a single stream drainage. The sample point in the upper left is farthest downstream, so its total catchment area is the sum of all of the catchment polygons shown. The Tg unit area and area percentage for this sample (shown in the Single Record View above left) are therefore computed using all of the contributing catchment polygons. Tertiary granite underlies 52.3% of the total catchment area of this sample.

Identifying geochemical anomalies in these sediment samples requires knowledge of the rock units and soils within the contributing areas. In particular, the areal extents of different rock and soil units can affect the background values of the chemical elements measured and influence the identification of anomalies. To aid in this analysis, MicroImages has created another custom geospatial processing script that tabulates the areas of map units that occur in each catchment polygon and any upstream catchments.

The inputs to this MapUnitAreas script (excerpted on the reverse) are topological vector objects: the sample point and catchment polygon objects created by the Sample Catchments script and an overlapping vector geologic, soil, or other map. You can choose to compute area information for all map units or restrict the computation to a single target map unit. The script performs a logical intersection (vector AND) of the catchment polygons and map unit polygons, which subdivides each catchment polygon into its constituent map unit polygons. Map unit areas (in square kilometers) and area percentages are then computed recursively (summed over all contributing catchments) for each sample point and stored in a database table. This map unit area table is written to a copy of the input sample point vector object and its records are attached to the corresponding points. The table is also written to a copy of the input catchment polygon vector object and its records attached to the corresponding polygons.

Tertiary

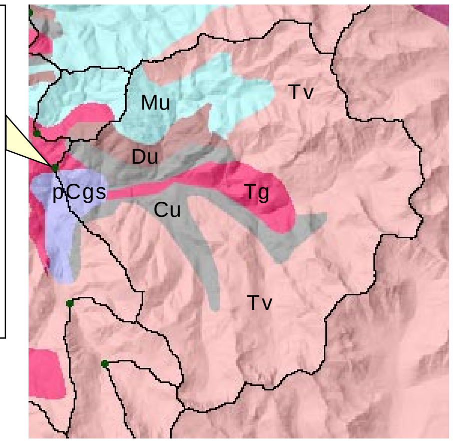
Tv (volcanic):	67.9
Tg (granite):	6.1

Paleozoic
(sedimentary)

Mu:	9.0
Du:	3.8
Cu:	11.2

Precambrian

pCgs(gneiss):	1.9
	99.9%



In this illustration the MapUnitAreas script was used to tabulate areas for all geologic map units. The resulting percentage values are shown for a sample and catchment polygon that have no upstream contributing catchments, so the tabulation shown covers only this polygon. Zero values for map units not included within this catchment are omitted in the list above.

Many sample scripts have been prepared to illustrate how you might use the features of the TNT products' scripting language for scripts and queries. These scripts can be downloaded from www.microimages.com/downloads/scripts.htm.

Script Excerpts for GeolMapUnits.sml

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

```
func class STRINGLIST getMapUnitList (class DBTABLEINFO dbInfo, class
      STRING fieldName$)
   {
   for i = 1 to dbInfo.NumRecords
      {     read map unit identifier field in record
      if (mapUnitIdFieldTypeStr == "string")
         {
         unit$ = 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, "NumUpSamples",
      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] ) );
                                         add ID as string to
         tempList.AddToEnd(tempIDstr);   temporary stringlist
         }
         loop through temporary list of UpSample fields to get
         their adjacent upstream 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);
         }
      }
   }
```