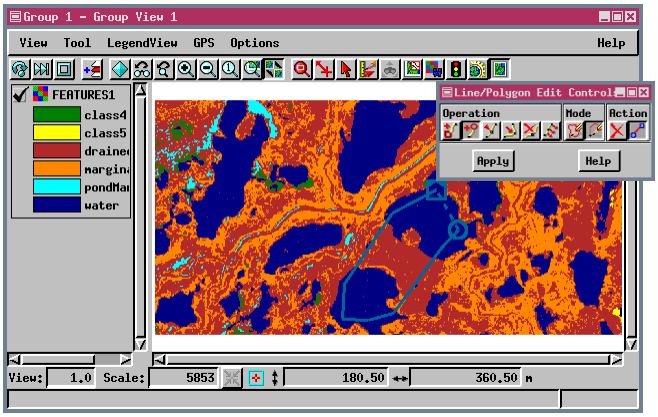
Sample SML Tool Script

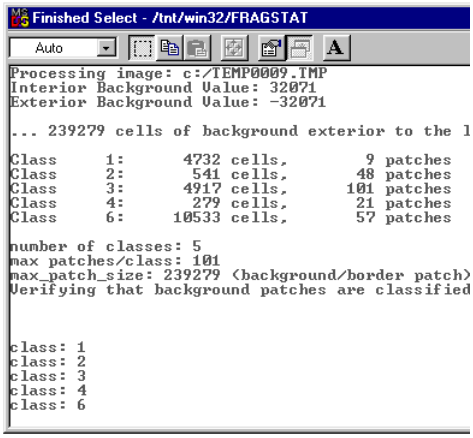
Running FRAGSTATS with TNTmips



The patch, which is the basic unit of landscape ecology, is a piece of the landscape that is considered homogeneous at the scale of a particular study. A landscape, or area of interest, is a mosaic of patches of different types. Patch type is equivalent to a class in TNTmips terminology (patches of a single type in the landscape belong to the same class).

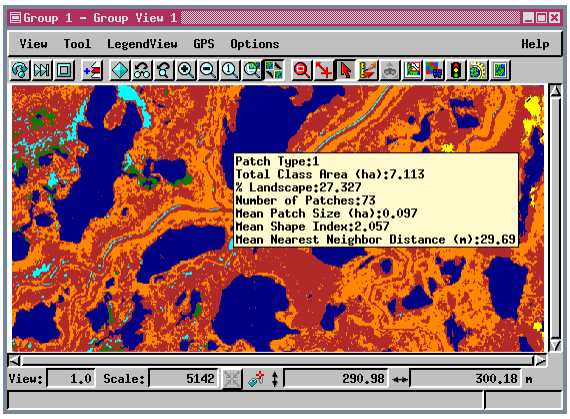
In order to study landscape function and change, you need to be able to quantify landscape structure. The FRAGSTATS program was developed for this purpose by Kevin McGarigal and Barbara Marks. FRAGSTATS calculates a number of statistics for each patch, for all the patches of a single class, and for the landscape as a whole. FRAGSTATS is concerned with both landscape composition and landscape configuration. Landscape composition addresses the variety and abundance of patches within the landscape, while landscape configuration is concerned with physical distribution and spatial character of patches.

FRAGSTATS runs under DOS and outputs four files, each with the name you provide and a different extension (*.cla, *.ful, *.lnd, *.pat). Because it runs under DOS, output file names are restricted to eight characters. Three of these files are designed for direct database import from text while the fourth file (*.ful) combines information on individual patches, patch classes, and the landscape as a whole into a single, more humanly readable report.



Two separate scripts for running FRAGSTATS are available with this release of the TNT products. One is a tool script that lets you draw a region to define the area of the underlying raster to use for calculation of statistics. The other script is run through the SML process and requires the landscape raster and a mask raster to define your area of interest within the landscape. The FRAGSTATS tool script demonstrates that a tool script can run an external program using objects from TNTmips Project Files and that FRAGSTATS can be used with an interactively designed mask (a region).

Once you have drawn and applied your region (or selected both the landscape and mask raster), you are asked to supply an edge distance in meters. The edge distance is the setback distance (in meters) within each patch for the purpose of calculating core area metrics. Next, a DOS shell opens and reports progress while FRAGSTATS is running. When FRAGSTATS is done, the DOS shell will say *finished* in the title bar. You need to close the DOS shell in order to continue working in TNTmips' Spatial Data Display process. The amount of time it takes FRAGSTATS to run is determined by the number of cells selected for processing and the number of patches within the selected area.



Summary statistics for each patch type were imported to a database related to the landscape raster by cell value. A multiline DataTip that incorporates specific statistics of interest (from the 40 in the *.cla file) was then constructed using a string expression field.

The information shown in the multiline DataTip (left) was selected from the table that resulted from import of the .cla file produced by FRAGSTATS. The imported table can be related to the internal table using cell value (Internal.Value), which corresponds to the patch type in the FRAGSTATS output.

Macro and Tool Scripts can be created using SML in any TNTmips process that uses a View window (Options / Customize from the View window menu bar). These scripts are then available from an icon, which you select or design, on the toolbar. Sample scripts have been prepared to illustrate how you might use these features, which are available only in TNTmips 6.4 or later, to assist with specific tasks you perform on a regular basis. If possible, the full script is printed below for your quick perusal. When a script is too long to fit on one page, key sections are reproduced below. The sample Tool Script illustrated can be downloaded from the SML script exchange at www.microimages.com/sml/ftpsmllink/TNT_Products_V6.5_CD.

Script for Running FRAGSTATS on RVC Data (fragtool.sml)

```
class XmForm form, buttonRow;
class PushButtonItem closeButton;
class MdispRegionTool tool;
class Raster targetRaster;
class LAYER rasterLayer;
string rasterName$, frag$;
class XmDrawingArea da;
class GraphicsContext gc;

func checkLayer() {
    local boolean valid = false;
    # Get name of active layer if it is usable.  If not output an error message.
    if (Group.ActiveLayer.Type == "Raster") {
        rasterLayer = Group.ActiveLayer;
        DispGetRasterFromLayer(targetRaster, rasterLayer);
        if (targetRaster.$Info.Type == "32-bit float" or targetRaster.$Info.Type == "64-bit float") {
            rasterName$ = "Type not supported!";
            }
        else {
            rasterName$ = rasterLayer.Name;
            valid = true;
            }
        }
    else
        rasterName$ = "Not a raster!";
    return valid;
    }

proc cbRedraw() {
    if (gc == 0) return;
    ActivateGC(gc);

    SetColorName("gray75");
    FillRect(0, 0, da.width, da.height);
    SetColorName("black");
    DrawInterfaceText(rasterName$, 0, 12);
    }

proc cbLayer() {
    checkLayer();
    cbRedraw();
    }

proc cbToolApply(class RegionTool tool) {
    if (checkLayer()) {
        local region MyRgn;
        local class StatusHandle status;
        local class StatusContext context;
        local numeric lins, cols, csize, edist, value;
        string type$, tempFile$, fragout$;

        status = StatusDialogCreate(form);
        context = StatusContextCreate(status);
        StatusSetMessage(context, "Running fragstats...");

        MyRgn = tool.Region;
        MyRgn = RegionTrans(MyRgn, ViewGetTransViewToScreen(View, 1));
        MyRgn = RegionTrans(MyRgn, ViewGetTransLayerToView(View, rasterLayer, 1));

        lins = NumLins(targetRaster);
        cols = NumCols(targetRaster);
        tempFile$ = CreateTempFileName();
        fragout$ = GetToken(GetOutputFileName(_context.ScriptDir, "Where would you like the results?",
            ""), ".", 0);
        csize = (LinScale(targetRaster) + ColScale(targetRaster)) / 2;
        edist = PopupNum("Enter the edge distance in meters:");

        outFile = fopen(tempFile$);

        value = 32071;
        for row = 0 to lins - 1 step 1 {
            for column = 0 to cols - 1 step 1 {
             if (PointInRegion(row, column, MyRgn)) then {
                        fprintf(outFile, "%d ", targetRaster[row, column]);
                        }
             else fprintf(outFile, "%d ", -value);
             }
            if (row != lins - 1)
             fprintf(outFile, "\n");
            }
        fclose(outFile);

        run(sprintf("%s %s %s %d %d 2 %d %d %d $ $ $ $ $ y y y y y", frag$, tempFile$, fragout$, csize,
            edist, lins, cols, value), 1);

        DeleteFile(Rout.$Info.Filename);
        DeleteFile(tempFile$);

        StatusContextDestroy(context);
        StatusDialogDestroy(status);
        }
    else PopupMessage(rasterName$);
    }

proc cbClose() {
    tool.Managed = 0;
    DialogClose(form);
    if (setDefaultWhenClose) {
        setDefaultWhenClose = false;
        View.SetDefaultTool();
        }
    }

func OnInitialize () {
    WidgetAddCallback(Group.LayerSelectedCallback, cbLayer);

    form = CreateFormDialog("Fragstat");
    form.marginHeight = 2;
    form.marginWidth = 2;
    WidgetAddCallback(form.Shell.PopdownCallback, cbClose);

    da = CreateDrawingArea(form, 15, 400);
    da.topWidget = form;
    da.leftWidget = form;
    da.rightWidget = form;
    WidgetAddCallback(da.ExposeCallback, cbRedraw);

    line = CreateHorizontalSeparator(form);
    line.topWidget = da;
    line.leftWidget = form;
    line.rightWidget = form;
    line.topOffset = 2;

    closeButton = CreatePushButtonItem("Close", cbClose);

    buttonRow = CreateButtonRow(form, closeButton);
    buttonRow.topWidget = line;
    buttonRow.leftWidget = form;
    buttonRow.rightWidget = form;
    buttonRow.bottomWidget = form;

    tool = ViewCreatePolygonTool(View);
    ToolAddCallback(tool.ActivateCallback, cbToolApply);

    frag$ = GetInputFileName("c:/tnt/win32/fragstats.exe", "Please locate the fragstat executable.", "exe");
    }  # end of OnInitialize

func OnDestroy () {
    tool.Managed = 0;
    DestroyGC(gc);
    DestroyWidget(form);
    }  # end of OnDestroy

func OnActivate () {
    checkLayer();
    tool.Managed = 1;
    tool.HasPosition = 0;
    DialogOpen(form);
    if (gc == 0)
        gc = CreateGCForDrawingArea(da);
    cbRedraw();
    setDefaultWhenClose = true;
    }  # end of OnActivate

func OnDeactivate () {
    setDefaultWhenClose = false;
    cbClose();
    }  # end of OnDeactivate
```

variable declarations

checks to see if active layer is a raster of valid type for FRAGSTATS

draws text in Fragstat window

check new active layer if changed

sets local variables

procedure when region applied

creates status bar

gets region

writes to text file

applies the region for use in FRAGSTATS

closes output file

runs FRAGSTATS

deletes temporary files

closes Status window

actions when close Fragstat window

called first time tool is activated, creates windows and asks user to locate FRAGSTATS executable

called when tool is destroyed

called when tool is activated

called when tool is deactivated