



**Integrated Land
Management
Bureau**

Kalum LRMP Resource Monitoring Framework Appendices

Version History

Version	Date	Comments
Version 1.1	August 24, 2007	

**Version 1.0
August 2007**

Table of Contents

General Assumptions.....	4
Vegetation Data.....	4
Tree Farm License (TFL) Data.....	4
Crown Forest Land Base (CFLB).....	4
Ownership.....	5
Harvested Areas.....	5
Roads.....	5
Project Area of Interest.....	5
Indicator 1: Percent of early, mature plus old, and old seral stages within the Crown Forested Land Base, by landscape unit, by BEC variant.....	7
Indicator 2: Percent of old seral stages within the CFLB, in identified undeveloped watersheds by BEC site series.....	15
Indicator 3: Percent of old seral forest within OGMAs by landscape unit by BEC variant.....	23
Indicator 4: Percent of OGMAs with interior forest condition by BEC variant.....	29
Indicator 5: Percent of OGMA with forest harvesting.....	35
Indicator 6: Percent or area (ha) of OGMA harvested for allowable disturbances.....	38
Indicator 7: Percent or area (ha) added to OGMAs to replace disturbed areas.....	41
Indicator 8: Percent of cutblock area or cutblock aggregate retained as Wildlife Tree Patches, by landscape unit by BEC subzone.....	44
Indicator 9: Percent species composition by landscape unit by BEC subzone.	50
Indicator 10: Percent distribution by patch size by natural disturbance type by landscape unit by BEC variant.....	57
Indicator 11: Amount of harvesting within polygon “A” on Map 5 (SRMP), the pass between Kiteen and Cedar drainages.....	65
Indicator 12: Percent of harvested area within polygon “B”.....	73
Indicator 13: Percent of area in old and mature seral stages in the level pass between the Williams and Thomas/Clore watersheds identified on Map 5.....	74
Indicator 14: Amount of harvesting in the Skeena Islands.....	78
Indicator 15: Percent of area harvested within Subzone 1, Lakelse River SRMZ.....	81
Indicator 16: Percent of early seral within Subzone 2, Lakelse River SRMZ.....	82
Indicator 17: Size (ha) of openings within Subzone 2, Lakelse River SRMZ.....	86
Indicator 18: Percent retention within cut blocks within Subzone 2, Lakelse River SRMZ.....	88
Indicator 19: Amount (ha) of timber harvesting within the Upper Kitsumkalum SRMZ.....	92
Indicator 20: Amount (ha) of industrial activity within the Miligit Valley Sensitive Area.....	93
Indicator 21: Number of kilometres of roads constructed in riparian areas, wildlife habitat areas and forest ecosystem networks.....	94
Indicator 22: Equivalent clearcut area (ECA) for community watersheds.....	98
Indicator 24: Amount of mid-seral forest within the forested land base, excluding hardwood, in the McKay-Davis and Copper watersheds.....	99
Indicator 25: Number of stems per hectare within free growing managed forests on rich and wetter sites within watersheds identified on Map 7 (SRMP).....	103
Indicator 26: Number of bridge crossings on the Upper Copper River upstream of confluence with Limonite Creek at any given time.....	104
Indicator 27: Width (metres) of no-harvest reserve along both sides of the Upper Copper River above	

Limonite Creek.....	105
Indicator 28: Size (metres) of harvest openings visible from the Sue Channel/Hawkesbury Island Protected Area.....	106

General Assumptions

All indicators are analysed with the assumption that the Crown Forested Land Base (CFLB) has been applied to the area of interest. For example, where identified undeveloped watersheds are the area of interest, the total area used for the analysis is the total CFLB within the undeveloped watersheds, not the total area of the undeveloped watersheds.

Vegetation Data

The Kalum LRMP area consists of TSA land, as well as TFL1 and TFL41 lands. The vegetation data (VRI) utilized for this project was pulled from the LRDW. In contrast, the SRMP OGMA analysis utilized the Forest Cover data (now retired). The vegetation layers required a conversion from multi-part to single-part. For some reason, multi-part polygons is the default option in many analysis functions. When appending the three datasets together, NULL values had to be removed on the common field attributes. All datasets had to be in single poly form.

XtoolsPro was used to trim down the related tables by deleting multiple fields at once. Watch for field mismatches, as fields visible in Xtools Pro do not always match the fields visible in the table view.

The fields retained within the vegetation layers for the analyses are:

FC_TAG, SPC1, PCT1, SPC2, PCT2, AGE_PRJ, TYPID_PR

Tree Farm License (TFL) Data

TFL reallocation is still underway with no final boundary revisions¹. The reallocation project is not relevant to this project as all the TSA and TFL lands were combined into one seamless vegetation coverage. According to the Kalum MOF office, the TFL data has not changed significantly since the LRMP was completed².

Crown Forest Land Base (CFLB)

The methodology from the most recent Timber Supply for the Kalum TSA was applied to create the Crown Forested Land Base (CFLB). The Landscape Unit Planning Guide, March 1999, was also referenced.

Initial Approach:

ownership = 62C and 69C

remove non-forested land with projected type id = 6 or 8

The ownership (f_own) coverages provided no accurate information. When running a query to pull out ownership codes 62C and 69C, only water and inlets were selected.

1 Telephone conversation (June 26, 2007) with Diane Roberge, ILMB Prince George office.

2 Email correspondence with Hubert Burger, MOF Kalum District Office, May 24th, 2007.

Instead, to calculate ownership, the same methodology used in the Lakes SLIMP project was utilized. See Section entitled ownership.

To remove non-forested (alpine, swamps, grasslands, avalanche chutes, np forest, np brush, NTA and non-commercial) selected by `typid_pr = 6, 8 and 5`. This selection was negated to select only the forested land base. This query was carried out on all TFL areas as well as the TSA area.

The Erase command was used to remove the private land within the Forested Land Base.

Ownership

The ownership layer was created as follows:

- load `WHSE_CADASTRE.ICI_PARCEL_FABRIC_PUBLIC_VW` (this layer has no data within Kalum LRMP area)
- load `WHSE_TANTALIS.TA_SURVEY_PARCELS_VW`
- select above layers by area of interest and save as layer file
- Export layer file
- highlight `WHSE_TANTALIS.TA_SURVEY_PARCEL_SHAPES` export layer, and choose Joins and Relates -- > join.
- Join the table `WHSE_TANTALIS.TA_SURVEY_PARCELS_VW` with the field `PIN_SID`.
- export the entire layer again to capture the joined table
- clip to area of interest

Harvested Areas

A harvested layer was created by clipping out the `RSLT` layer from the `LRDW`. The layer was further trimmed by selecting out the roads, swamps, and non-productive areas. This layer is called `harvested_erase`.

Roads

`FTEN` and `TRIM` roads were both examined within the Kalum LRMP area for completeness. When compared, the difference in length between the two layers was less than 100 m. This small difference is essentially negligible, and therefore only `TRIM` roads were used for the analysis.

Project Area of Interest

Kalum LRMP area of interest

The original LRMP boundary was utilized to clip the project data. To simplify the clipping process (due to lengthy processing times within ArcGIS), the LRMP boundary was simplified to reduce the number of vertices. The steps taken were:

- *feature to line* tool to convert the polygon to a line

- the *simplify line* tool with a tolerance of 300m
- *feature to line* to convert line back to polygon

SRMP area of interest

At times throughout the data preparation phase, the SRMP boundary was required as a clip boundary. To reduce processing time (and prevent the *clip* function from freezing), the original SRMP boundary was simplified using the same process as the Kalum LRMP boundary above.

Indicator 1: Percent of early, mature plus old, and old seral stages within the Crown Forested Land Base, by landscape unit, by BEC variant.

The Kalum SRMP, Page 6 refers to this indicator within the context of the Forest Land Base. The GIS analysis below is run on the CFLB to ensure consistency across indicators. Definitions for seral stage distribution is taken from the Biodiversity Guidebook.

```
# -----
# bio_ind1.py
# Scripting Dates: July 30th
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates early, mature, and old seral stages within CFLB by LU by BEC variant
# Data sources are personal geodatabases
# -----

#====Section 1====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#====

#====Section 2====
# allow user to enter path to personal geodatabase
files_exist = 1
option_var = string.capitalize(raw_input("Enter Y to use default data paths or enter N to use different paths"))
print 'option_var', option_var
if option_var == 'N':
    # allow user to enter names of various geodatabases and feature classes required
    print 'To enter data paths use the format: W:\\srm\smt\\Workarea\\...'
    print 'Ensure you use double forward slashes'

lu_fclass = raw_input("Enter the data path and name of the Landscape Unit feature class -->")
if gp.exists(lu_fclass):
    print 'Landscape Unit Feature Class exists'
else:
```

```

print 'Landscape Unit Feature Class does not exist'
files_exist = 0

bec_fclass = raw_input("Enter the data path and name of the Biogeoclimatic Zones feature class -->")
if gp.exists(fec_flclass):
    print 'BEC feature class exists'
else:
    print 'BEC feature class does not exist'
    files_exist = 0

cflb_fclass = raw_input("Enter the name of the Crown Forested Land Base -->")
if gp.exists(cflb_fclass):
    print 'CFLB feature class exists'
else:
    print 'CFLB feature class does not exist'
    files_exist = 0

interm_pgdb =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.mdb"
workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
else:
    print 'using default files'
    lu_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.mdb\\flu_kmv4"
    bec_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.mdb\\abec_kmv2"
    cflb_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.mdb\\cflb_kmv2"
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    interm_pgdb =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.mdb"

if files_exist:
#====Section 3====
# union the landscape units with the BEC variants
# into the interm resultant feature class lu_bec
    lubec_var = interm_pgdb + "\\lu_bec"
    if gp.exists(lubec_var):
        gp.Delete_management(lubec_var)
        print 'deleting resultant lubec_var'
# perform the union
    try:

```

```

    gp.union_analysis(lu_fclass + ";" + bec_fclass, lubec_var, "ALL" )
    print 'union of lu and bec'
except:
    print gp.GetMessages()
    print 'union did not work'

# intersect the crown forested land base with the above resultant
union_var = interm_pgdb + "\\union_all"
if gp.exists(union_var):
    gp.Delete_management(union_var)
    print 'deleting union_var'

var3 = \" + cflb_fclass + \"

try:
    gp.intersect_analysis(var3 + ";" + lubec_var, union_var, "ALL")
    print 'intersection of cflb with bec and lu worked'
except:
    print gp.GetMessages()

#=====

# read in the lu names and place them in a list - this is the list of keys for the dictionary
# cursor through the lu .dbf and dump names into the data dictionary.
lu_keylist = []
curObj = gp.SearchCursor(lu_fclass)
rowObj = curObj.next()
while rowObj:
    if ((rowObj.getValue('LU_NAME')) not in lu_keylist) and (rowObj.getValue('LU_NAME') <> ""):
        lu_keylist.append(rowObj.getValue('LU_NAME'))
    rowObj = curObj.next()
print lu_keylist

# create an empty 2-D list for the values in the dictionary
complete_list = []

#=====
# cycle through landscape units and calculate seral stage areas for each bec variant
#calculate early and place in list

for luname in lu_keylist:
    #create an empty list for the bec values for each lu

#set up loop for BEC Variant
bec_keylist = []
curObj = gp.SearchCursor(union_var)
rowObj = curObj.next()
while rowObj:

```

```

        if (((rowObj.getValue('BECLABEL')) not in bec_keylist) and ((rowObj.getValue('LU_NAME'))
== lname)):
            bec_keylist.append(rowObj.getValue('BECLABEL'))
            rowObj = curObj.next()
            print 'bec variant list for', lname, ' is', bec_keylist

#now run the query for calculating areas, looping through the bec variants
for variant in bec_keylist:
    result_list = []
    result_list.append(lname)

#calculate area of bec subzone within lu
result_tmp1 = workarea + "interm_results.mdb\\sel1_tmp1"
if gp.exists(result_tmp1):
    gp.Delete_management(result_tmp1)
    print 'deleting selected class'
    phrase01 = "[LU_NAME] = \" + lname + "\""
    phrase02 = " AND [BECLABEL] = \" + variant + "\""
    phrase03 = phrase01 + phrase02
    try:
        gp.select_analysis(union_var, result_tmp1, phrase03)
        print 'running the selection'
    except:
        print gp.GetMessages()
        print 'no select run'

# now run stats for full area of bec subzone
out_tbl4 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl1"
if gp.exists(out_tbl4):
    gp.Delete_management(out_tbl4)
    print 'deleting stats table'
    try:
        print 'running area stats for bec subzone', variant
        gp.Statistics_analysis(result_tmp1, out_tbl4, "Shape_Area SUM", "BECLABEL")
    except:
        print gp.GetMessages()
        print 'stats table didn't work'

# pull out EARLY table items
result_tmp2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\sel_tmp1"
if gp.exists(result_tmp2):
    gp.Delete_management(result_tmp2)
    print 'deleting selected early class'
    phrase1 = "([ZONE_] = 'CWH' or [ZONE_] = 'ESSF' or [ZONE_] = 'ICH' or [ZONE_] =

```

```

\MH\)
phrase2 = " AND [AGE_PRJ] < 40"
phrase3 = " AND ([LU_NAME] = \" + luname + "\) AND ([BECLABEL] = \" + variant + "\)"
phrase4 = phrase1 + phrase2 + phrase3
try:
    gp.select_analysis(union_var, result_tmp2, phrase4)
    print 'running the early selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting early stats table'
try:
    print 'running early stats'
    gp.Statistics_analysis(result_tmp2, out_tbl1, "Shape_Area SUM", "BECLABEL")
except:
    print gp.GetMessages()
    print 'no stats table created'

#now read beczone stats table and save it to results_list
if (gp.Exists(out_tbl4)) and (gp.Getcount_management(out_tbl4) > 0):
    curObj = gp.SearchCursor(out_tbl4)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append('no bec areas')

#now cursor through the resulting table and dump values into a list.
#set up a loop which will interate through the records in the feature data set
if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        result_list.append('early')
#        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:

```

```

        result_list.append("early")
#       result_list.append(variant)
        result_list.append('no value')

#now run the query for mature plus old
    print 'running mature plus old'
    result_tmp3 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\selmo_tmp2"

    if gp.exists(result_tmp3):
        gp.Delete_management(result_tmp3)
        print 'deleting selected class'

    phrase5 = "([ZONE_] = 'CWH' AND [AGE_PRJ] > 80)"
    phrase6 = " OR ([ZONE_] = 'ESSF' AND [AGE_PRJ] > 120)"
    phrase7 = " OR ([ZONE_] = 'ICH' AND [AGE_PRJ] > 100)"
    phrase8 = " OR ([ZONE_] = 'MH' AND [AGE_PRJ] > 120))"
    phrase9 = " AND ([LU_NAME] = '\" + lname + "\') AND ([BECLABEL] = '\" + variant + "\')"
#   phrase9 = " AND ([LU_NAME] = '\" + lname + "\'"
    phrase10 = phrase5 + phrase6 + phrase7 + phrase8 + phrase9
    try:
        gp.select_analysis(union_var, result_tmp3, phrase10)
        print 'running the mature plus old selection'
    except:
        print gp.GetMessages()

# now run stats command and create stats table
    out_tbl2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
    #kill table if it already exists
    if gp.exists(out_tbl2):
        gp.Delete_management(out_tbl2)
        print 'deleting mature plus old stats table'
    try:
        print 'running mat + old stats'
        gp.Statistics_analysis(result_tmp3, out_tbl2, "Shape_Area SUM", "BECLABEL")
    except:
        print gp.GetMessages()
        print 'No stats table created for mat + old'

#now cursor through the resulting .dbf and dump values into a list
    if (gp.Exists(out_tbl2)) and (gp.Getcount_management(out_tbl2) > 0):
        curObj = gp.SearchCursor(out_tbl2)
        rowObj = curObj.next()
        while rowObj:

```

```

        result_list.append('mature and old')
#         result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
    else:
        result_list.append('mature and old')
#         result_list.append(variant)
        result_list.append("NOLABEL")

#selection and stats for Old Category
    result_tmp4 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\selold_tmp3"
    if gp.exists(result_tmp4):
        gp.Delete_management(result_tmp4)
        print 'deleting selected class'
    phrase11 = "([ZONE_] = 'CWH' OR [ZONE_] = 'ESSF' OR [ZONE_] = 'ICH' OR [ZONE_]
= 'MH')"
    phrase12 = " AND [AGE_PRJ] > 250"
    phrase13 = " AND ([LU_NAME] = '\" + luname + '\" ) AND ([BECLABEL] = '\" + variant + '\" )"
    phrase14 = phrase11 + phrase12 + phrase13
    try:
        gp.select_analysis(union_var, result_tmp4, phrase14)
        print 'running the old selection'
    except:
        print gp.GetMessages()

# now run stats command and create stats table
    out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
    #kill table if it already exists
    if gp.exists(out_tbl1):
        gp.Delete_management(out_tbl1)
        print 'deleting stats table'
    try:
        print 'running old stats'
        gp.Statistics_analysis(result_tmp4, out_tbl1, "Shape_Area SUM", "BECLABEL")
    except:
        #if an error occurred while running a tool print the message
        print gp.GetMessages()
        print 'old stats table didn't work'

    if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
        curObj = gp.SearchCursor(out_tbl1)
        rowObj = curObj.next()
        while rowObj:
            result_list.append('old')

```

```

#         result_list.append(rowObj.getValue('BECLABEL'))
#         result_list.append(rowObj.getValue('SUM_Shape_Area'))
#         rowObj = curObj.next()
else:
    result_list.append('old')
#     result_list.append(variant)
#     result_list.append("NOLABEL")

complete_list.append(result_list)

#=====
# write results to a .CSV file
#the file will be overwritten if it exists
final_results = workarea + "results\\indicator_01.txt"
file_hand = open(final_results,'w')
for x in complete_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

else:
    print 'The user supplied data paths and feature classes do not exist. Try again'

```

Indicator 2: Percent of old seral stages within the CFLB, in identified undeveloped watersheds by BEC site series.

As per conversations with ILMB staff, site series data for the Kalum was determined as not reliable until further field verification has been carried out. BEC variant data was supplemented to augment the analysis instead.

```
# -----  
# bio_ind1.py  
# Scripting Dates: July 30th  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates early, mature, and old seral stages within CFLB by LU by BEC variant  
# Data sources are personal geodatabases  
# -----  
  
#====Section 1=====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")  
  
# Load required toolboxes...  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")  
#=====  
  
#====Section 2=====  
# allow user to enter path to personal geodatabase  
files_exist = 1  
option_var = string.capitalize(raw_input("Enter Y to use default data paths or enter N to use different  
paths"))  
print 'option_var', option_var  
if option_var == 'N':  
    # allow user to enter names of various geodatabases and feature classes required  
    print 'To enter data paths use the format: W:\\srm\smt\\Workarea\\...'  
    print 'Ensure you use double forward slashes'  
  
    lu_fclass = raw_input("Enter the data path and name of the Landscape Unit feature class -->")  
    if gp.exists(lu_fclass):  
        print 'Landscape Unit Feature Class exists'  
    else:  
        print 'Landscape Unit Feature Class does not exist'  
        files_exist = 0
```

```

bec_fclass = raw_input("Enter the data path and name of the Biogeoclimatic Zones feature class -->")
if gp.exists(fec_fclass):
    print 'BEC feature class exists'
else:
    print 'BEC feature class does not exist'
    files_exist = 0

cflb_fclass = raw_input("Enter the name of the Crown Forested Land Base -->")
if gp.exists(cflb_fclass):
    print 'CFLB feature class exists'
else:
    print 'CFLB feature class does not exist'
    files_exist = 0

interm_pgdb =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b"
workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
else:
    print 'using default files'
    lu_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\tlu_kmv4"
    bec_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\abec_kmv2"
    cflb_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\cflb_kmv2"
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    interm_pgdb =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b"

if files_exist:
#=====Section 3 =====
# union the landscape units with the BEC variants
# into the interm resultant feature class lu_bec
lubec_var = interm_pgdb + "\\lu_bec"
if gp.exists(lubec_var):
    gp.Delete_management(lubec_var)
    print 'deleting resultant lubec_var'
# perform the union
try:
    gp.union_analysis(lu_fclass + ";" + bec_fclass, lubec_var, "ALL" )
    print 'union of lu and bec'

```

```

except:
    print gp.GetMessages()
    print 'union did not work'

# intersect the crown forested land base with the above resultant
union_var = interm_pgdb + "\\union_all"
if gp.exists(union_var):
    gp.Delete_management(union_var)
    print 'deleting union_var'

var3 = "\" + cflb_fclass + "\"

try:
    gp.intersect_analysis(var3 + ";" + lubec_var, union_var, "ALL")
    print 'intersection of cflb with bec and lu worked'
except:
    print gp.GetMessages()

#=====

# read in the lu names and place them in a list - this is the list of keys for the dictionary
# cursor through the lu .dbf and dump names into the data dictionary.
lu_keylist = []
curObj = gp.SearchCursor(lu_fclass)
rowObj = curObj.next()
while rowObj:
    if ((rowObj.getValue('LU_NAME')) not in lu_keylist) and (rowObj.getValue('LU_NAME') <> "):
        lu_keylist.append(rowObj.getValue('LU_NAME'))
    rowObj = curObj.next()
print lu_keylist

# create an empty 2-D list for the values in the dictionary
complete_list = []

#=====
# cycle through landscape units and calculate seral stage areas for each bec variant
#calculate early and place in list

for luname in lu_keylist:
    #create an empty list for the bec values for each lu

#set up loop for BEC Variant
bec_keylist = []
curObj = gp.SearchCursor(union_var)
rowObj = curObj.next()
while rowObj:
    if (((rowObj.getValue('BECLABEL')) not in bec_keylist) and ((rowObj.getValue('LU_NAME'))
== luname)):

```

```
    bec_keylist.append(rowObj.getValue('BECLABEL'))
    rowObj = curObj.next()
print 'bec variant list for', luname, ' is', bec_keylist
```

```
#now run the query for calculating areas, looping through the bec variants
```

```
for variant in bec_keylist:
```

```
    result_list = []
```

```
    result_list.append(luname)
```

```
#calculate area of bec subzone within lu
```

```
    result_tmp1 = workarea + "interm_results.mdb\\sel1_tmp1"
```

```
    if gp.exists(result_tmp1):
```

```
        gp.Delete_management(result_tmp1)
```

```
        print 'deleting selected class'
```

```
    phrase01 = "[LU_NAME] = \" + luname + "\"
```

```
    phrase02 = " AND [BECLABEL] = \" + variant + "\"
```

```
    phrase03 = phrase01 + phrase02
```

```
    try:
```

```
        gp.select_analysis(union_var, result_tmp1, phrase03)
```

```
        print 'running the selection'
```

```
    except:
```

```
        print gp.GetMessages()
```

```
        print 'no select run'
```

```
# now run stats for full area of bec subzone
```

```
    out_tbl4 =
```

```
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.mdb\\stats_tbl1"
```

```
    if gp.exists(out_tbl4):
```

```
        gp.Delete_management(out_tbl4)
```

```
        print 'deleting stats table'
```

```
    try:
```

```
        print 'running area stats for bec subzone', variant
```

```
        gp.Statistics_analysis(result_tmp1, out_tbl4, "Shape_Area SUM", "BECLABEL")
```

```
    except:
```

```
        print gp.GetMessages()
```

```
        print 'stats table didn't work'
```

```
# pull out EARLY table items
```

```
    result_tmp2 =
```

```
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.mdb\\sel_tmp1"
```

```
    if gp.exists(result_tmp2):
```

```
        gp.Delete_management(result_tmp2)
```

```
        print 'deleting selected early class'
```

```
    phrase1 = "([ZONE_] = 'CWH' or [ZONE_] = 'ESSF' or [ZONE_] = 'ICH' or [ZONE_] =
```

```
'MH')
```

```
    phrase2 = " AND [AGE_PRJ] < 40"
```

```

phrase3 = " AND ([LU_NAME] = \" + luname + "\") AND ([BECLABEL] = \" + variant + "\")"
phrase4 = phrase1 + phrase2 + phrase3
try:
    gp.select_analysis(union_var, result_tmp2, phrase4)
    print 'running the early selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting early stats table'
try:
    print 'running early stats'
    gp.Statistics_analysis(result_tmp2, out_tbl1, "Shape_Area SUM", "BECLABEL")
except:
    print gp.GetMessages()
    print 'no stats table created'

#now read beczone stats table and save it to results_list
if (gp.Exists(out_tbl4)) and (gp.Getcount_management(out_tbl4) > 0):
    curObj = gp.SearchCursor(out_tbl4)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append('no bec areas')

#now cursor through the resulting table and dump values into a list.
#set up a loop which will interate through the records in the feature data set
if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        result_list.append('early')
#        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append("early")
#        result_list.append(variant)

```

```

    result_list.append('no value')

#now run the query for mature plus old
    print 'running mature plus old'
    result_tmp3 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\selmo_tmp2"

    if gp.exists(result_tmp3):
        gp.Delete_management(result_tmp3)
        print 'deleting selected class'

    phrase5 = "([ZONE_] = 'CWH' AND [AGE_PRJ] > 80)"
    phrase6 = " OR ([ZONE_] = 'ESSF' AND [AGE_PRJ] > 120)"
    phrase7 = " OR ([ZONE_] = 'ICH' AND [AGE_PRJ] > 100)"
    phrase8 = " OR ([ZONE_] = 'MH' AND [AGE_PRJ] > 120))"
    phrase9 = " AND ([LU_NAME] = '\" + luname + "\') AND ([BECLABEL] = '\" + variant + "\")"
# phrase9 = " AND ([LU_NAME] = '\" + luname + "\")"
    phrase10 = phrase5 + phrase6 + phrase7 + phrase8 + phrase9
    try:
        gp.select_analysis(union_var, result_tmp3, phrase10)
        print 'running the mature plus old selection'
    except:
        print gp.GetMessages()

# now run stats command and create stats table
    out_tbl2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
    #kill table if it already exists
    if gp.exists(out_tbl2):
        gp.Delete_management(out_tbl2)
        print 'deleting mature plus old stats table'
    try:
        print 'running mat + old stats'
        gp.Statistics_analysis(result_tmp3, out_tbl2, "Shape_Area SUM", "BECLABEL")
    except:
        print gp.GetMessages()
        print 'No stats table created for mat + old'

#now cursor through the resulting .dbf and dump values into a list
    if (gp.Exists(out_tbl2)) and (gp.Getcount_management(out_tbl2) > 0):
        curObj = gp.SearchCursor(out_tbl2)
        rowObj = curObj.next()
        while rowObj:
            result_list.append('mature and old')
#            result_list.append(rowObj.getValue('BECLABEL'))

```

```

        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
    else:
        result_list.append('mature and old')
#     result_list.append(variant)
        result_list.append("NOLABEL")

#selection and stats for Old Category
    result_tmp4 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\selold_tmp3"
    if gp.exists(result_tmp4):
        gp.Delete_management(result_tmp4)
        print 'deleting selected class'
    phrase11 = "([ZONE_] = 'CWH' OR [ZONE_] = 'ESSF' OR [ZONE_] = 'ICH' OR [ZONE_]
= 'MH')"
    phrase12 = " AND [AGE_PRJ] > 250"
    phrase13 = " AND ([LU_NAME] = '\" + luname + "\') AND ([BECLABEL] = '\" + variant + "\'"
    phrase14 = phrase11 + phrase12 + phrase13
    try:
        gp.select_analysis(union_var, result_tmp4, phrase14)
        print 'running the old selection'
    except:
        print gp.GetMessages()

# now run stats command and create stats table
    out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
    #kill table if it already exists
    if gp.exists(out_tbl1):
        gp.Delete_management(out_tbl1)
        print 'deleting stats table'
    try:
        print 'running old stats'
        gp.Statistics_analysis(result_tmp4, out_tbl1, "Shape_Area SUM", "BECLABEL")
    except:
        #if an error occurred while running a tool print the message
        print gp.GetMessages()
        print 'old stats table didn"t work'

if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        result_list.append('old')
#     result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))

```

```
        rowObj = curObj.next()
    else:
        result_list.append('old')
#       result_list.append(variant)
        result_list.append("NOLABEL")

    complete_list.append(result_list)

#=====
# write results to a .CSV file
#the file will be overwritten if it exists
    final_results = workarea + "results\\indicator_01.txt"
    file_hand = open(final_results,'w')
    for x in complete_list:
        file_hand.writelines(str(x))
        file_hand.write('\n')
    file_hand.close()

else:
    print 'The user supplied data paths and feature classes do not exist. Try again'
```

Indicator 3: Percent of old seral forest within OGMAs by landscape unit by BEC variant.

```
# -----  
# bio_ind3.py  
# Scripting Dates: August 2, 2007  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates percent of old seral forest within OGMAs by LU by BEC variant  
# Data sources are personal geodatabases  
# -----  
  
#====Section 1=====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")  
  
# Load required toolboxes...  
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")  
#=====  
  
#====Section 2=====  
# allow user to enter path to personal geodatabase  
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))  
if option_var == "N":  
    # allow user to enter names of various geodatabases and feature classes required  
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")  
    lu_fclass = raw_input("Enter the name of the Landscape Unit feature class -->")  
    bec_fclass = raw_input("Enter name of the Biogeoclimatic Zones feature class -->")  
    cflb_fclass = raw_input("Enter the name of the TSA portion of the Forested Land Base -->")  
    resultant_pgd = raw_input("Enter the path and name to the Personal Geodatabase that holds the results of the analysis -->")  
    ogma_fclass = raw_input("Enter the name of the OGMA feature class")  
else:  
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"  
# biopgdb_path =  
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.mdb"
```

```

# lu_fclass =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\kmbiodiversity.m
db\tlu_kmv3"
lu_fclass = workarea + "kmbiodiversity.mdb\tlu_kmv4"
bec_fclass = workarea + "kmbiodiversity.mdb\abec_kmv2"
cflb_fclass = workarea + "kmbiodiversity.mdb\cflb_kmv2"
ogma_fclass = workarea + "kmbiodiversity.mdb\ogma_may07"
resultant_pgdb = workarea + "final_results.mdb"

#set up interm resultant file names and paths
interm_pgdb = workarea + "interm_results.mdb"

#====Section 3====
# union the landscape units with the BEC variants
# into the interm resultant feature class lu_bec
# first check if resultant fclass exists, if so, kill it
lubec_var = interm_pgdb + "\\lu_bec3"
if gp.exists(lubec_var):
    gp.Delete_management(lubec_var)
    print 'deleting resultant lubec_var'
# perform the union
try:
    gp.union_analysis(lu_fclass + ";" + bec_fclass, lubec_var, "ALL" )
    print 'union of lu and bec'
except:
    print gp.GetMessages()
    print 'union of bec and lu did not work'

#clip to OGMAs only
lubec_ogma = interm_pgdb + "\\lubec_clip"
if gp.exists(lubec_ogma):
    gp.Delete_management(lubec_ogma)
    print 'deleting lubec_ogma'
try:
    gp.clip_analysis(lubec_var, ogma_fclass, lubec_ogma)
    print 'clip worked'
except:
    print gp.GetMessages()

#clip cflb to OGMAs only
#broke the clips down into two steps because ArcGIS kept bailing when clipping the 4 layers together
cflb_ogma = interm_pgdb + "\\cflb_clip"
if gp.exists(cflb_ogma):
    gp.Delete_management(cflb_ogma)
    print 'deleting cflb_ogma'
try:
    gp.clip_analysis(cflb_fclass, ogma_fclass, cflb_ogma)
    print 'clip worked'

```

```

except:
    print gp.GetMessages()

#combine the lu, bec, and cflbs
union_all = interm_pgdb + "\\union_all3"
if gp.exists(union_all):
    gp.Delete_management(union_all)
    print 'deleting union_all'
try:
    gp.union_analysis(cflb_ogma + ";" + lubec_ogma, union_all, "ALL")
except:
    print gp.GetMessages()

#select the cflb within the ogmas within lu. This just cleans up the data a bit more
sumogma_tmp1 = workarea + "interm_results.mdb\\sel_sumogma"
if gp.exists(sumogma_tmp1):
    gp.Delete_management(sumogma_tmp1)
    print 'deleting selected cflb'
phrase7 = "[SPC1] <> \\"
try:
    gp.select_analysis(union_all, sumogma_tmp1, phrase7)
    print 'running the ogma selection'
except:
    print gp.GetMessages()

# read in the lu names and place them in a list
lu_list = []
curObj = gp.SearchCursor(sumogma_tmp1)
rowObj = curObj.next()
while rowObj:
    if ((rowObj.getValue('LU_NAME')) not in lu_list) and (rowObj.getValue('LU_NAME') <> ""):
        lu_list.append(rowObj.getValue('LU_NAME'))
    rowObj = curObj.next()
print lu_list

# now run stats command and create stats table to collect BEC Variant areas for each LU areas
lubec_list = []
tmp_list = []
for luname in lu_list:
    lubec_list.append(luname)
    # select all the BEC variants for the landscape unit
    result_tmp1 = workarea + "interm_results.mdb\\sel1_tmp3"
    if gp.exists(result_tmp1):
        gp.Delete_management(result_tmp1)
        print 'deleting selection'
    phrase1 = "[LU_NAME] = \" + luname + "\""
    try:
        gp.select_analysis(sumogma_tmp1, result_tmp1, phrase1)

```

```

    print 'running the lu selection'
except:
    print gp.GetMessages()
    print 'lu selection died'

# dump selection to stats file summarized by bec variant
sum_tbl0 = workarea + "interm_results.mdb\\sum_ogma_ind3"
#kill table if it already exists
if gp.exists(sum_tbl0):
    gp.Delete_management(sum_tbl0)
    print 'deleting ogma summary table'
try:
    print 'running ogma summary stats'
    gp.Statistics_analysis(result_tmp1, sum_tbl0, "Geometry_Area SUM", "BECLABEL")
except:
    print gp.GetMessages()
    print 'No ogma summary stats table created'
#now dump to tmp list
tmp_list = []
curObj = gp.SearchCursor(sum_tbl0)
rowObj = curObj.next()
while rowObj:
    tmp_list.append(rowObj.getValue('BECLABEL'))
    tmp_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
    rowObj = curObj.next()
    print tmp_list

lubec_list.append(tmp_list)

lubec_results = workarea + "results\\indicator2_03.txt"
file_hand = open(lubec_results,'w')
for x in lubec_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

print lubec_list

# create an empty list for the bec variant and area values
result_list = []
#create an empty list -- this will be a 2 dimentional list
final_list = []
#=====
# cycle through landscape units and calculate seral stage areas for each bec variant
#calculate early and place in list
for name in lu_list:
    print 'lu is ', name

```

```

result_list = []
result_list.append(name)
bec_keylist = []
curObj = gp.SearchCursor(sumogma_tmp1)
rowObj = curObj.next()
while rowObj:
    if (((rowObj.getValue('BECLABEL')) not in bec_keylist) and ((rowObj.getValue('LU_NAME')) ==
name)):
        bec_keylist.append(rowObj.getValue('BECLABEL'))
        rowObj = curObj.next()
print 'bec variant list for', name, ' is', bec_keylist
#now run the query for mature plus old, looping through the bec variants
for variant in bec_keylist:
    print 'running mature plus old for', variant
    result_tmp3 = workarea + "interm_results.mdb\\sel_tmp3"
    if gp.exists(result_tmp3):
        gp.Delete_management(result_tmp3)
        print 'deleting selected old class'
    phrase3 = "([ZONE_] = 'CWH' OR [ZONE_] = 'ESSF' OR [ZONE_] = 'ICH' OR [ZONE_] =
\MH\)"
    phrase4 = " AND [AGE_PRJ] > 250"
    phrase5 = " AND ([LU_NAME] = '\" + name + "\'"
    phrase6 = " AND ([BECLABEL] = '\" + variant + "\'"
    phrase7 = phrase3 + phrase4 + phrase5 + phrase6
    try:
        # print 'result_tmp3 is', result_tmp3
        gp.select_analysis(sumogma_tmp1, result_tmp3, phrase7)
        print 'running the old selection'
    except:
        print gp.GetMessages()
        print 'no select run'

# now run stats command and create stats table
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting stats table'
try:
# if gp.exists(result_tmp3):
    print 'running old stats for', name, variant
    gp.Statistics_analysis(result_tmp3, out_tbl1, "GEOMETRY_Area SUM", "BECLABEL")
except:
    print gp.GetMessages()
    print 'old stats table didn't work'

```

```
#now read stats table and save it to results_list
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
#         print 'The BEC zone is ', rowObj.getValue('BECLABEL')
        result_list.append(rowObj.getValue('BECLABEL'))
#         print 'the area is', rowObj.getValue('SUM_GEOMETRY_Area')
        result_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
        rowObj = curObj.next()
    print result_list

    final_list.append(result_list)

print final_list

final_results = workarea + "results\\indicator_03.txt"
file_hand = open(final_results,'w')
for x in final_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

Indicator 4: Percent of OGMA's with interior forest condition by BEC variant.

The definition of Interior Forest Condition was taken from the Bulkley State of the Forest Report. Interior Forest Condition is defined as old seral forest greater than 200m from young (early) seral.

```
# -----
# bio_ind3.py
# Scripting Dates: August 2, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates percent of old seral forest within OGMA's by LU by BEC variant
# Data sources are personal geodatabases
# -----

#====Section 1====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

#====Section 2====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")
    lu_fclass = raw_input("Enter the name of the Landscape Unit feature class -->")
    bec_fclass = raw_input("Enter name of the Biogeoclimatic Zones feature class -->")
    cflb_fclass = raw_input("Enter the name of the TSA portion of the Forested Land Base -->")
    resultant_pgd = raw_input("Enter the path and name to the Personal Geodatabase that holds the results of the analysis -->")
    ogma_fclass = raw_input("Enter the name of the OGMA feature class")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
# biopgdb_path =
```

```

"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db"
# lu_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\tlu_kmv3"
lu_fclass = workarea + "kmbiodiversity.mdb\\tlu_kmv4"
bec_fclass = workarea + "kmbiodiversity.mdb\\abec_kmv2"
cflb_fclass = workarea + "kmbiodiversity.mdb\\cflb_kmv2"
ogma_fclass = workarea + "kmbiodiversity.mdb\\ogma_may07"
resultant_pgdb = workarea + "final_results.mdb"

#set up interm resultant file names and paths
interm_pgdb = workarea + "interm_results.mdb"

#====Section 3====
# union the landscape units with the BEC variants
# into the interm resultant feature class lu_bec
# first check if resultant fclass exists, if so, kill it
lubec_var = interm_pgdb + "\\lu_bec3"
if gp.exists(lubec_var):
    gp.Delete_management(lubec_var)
    print 'deleting resultant lubec_var'
# perform the union
try:
    gp.union_analysis(lu_fclass + ";" + bec_fclass, lubec_var, "ALL" )
    print 'union of lu and bec'
except:
    print gp.GetMessages()
    print 'union of bec and lu did not work'

#clip to OGMAs only
lubec_ogma = interm_pgdb + "\\lubec_clip"
if gp.exists(lubec_ogma):
    gp.Delete_management(lubec_ogma)
    print 'deleting lubec_ogma'
try:
    gp.clip_analysis(lubec_var, ogma_fclass, lubec_ogma)
    print 'clip worked'
except:
    print gp.GetMessages()

#clip cflb to OGMAs only
#broke the clips down into two steps because ArcGIS kept bailing when clipping the 4 layers together
cflb_ogma = interm_pgdb + "\\cflb_clip"
if gp.exists(cflb_ogma):
    gp.Delete_management(cflb_ogma)
    print 'deleting cflb_ogma'
try:

```

```

gp.clip_analysis(cflb_fclass, ogma_fclass, cflb_ogma)
print 'clip worked'
except:
    print gp.GetMessages()

#combine the lu, bec, and cflbs
union_all = interm_pgdb + "\\union_all3"
if gp.exists(union_all):
    gp.Delete_management(union_all)
    print 'deleting union_all'
try:
    gp.union_analysis(cflb_ogma + ";" + lubec_ogma, union_all, "ALL")
except:
    print gp.GetMessages()

#select the cflb within the ogmas within lu. This just cleans up the data a bit more
sumogma_tmp1 = workarea + "interm_results.mdb\\sel_sumogma"
if gp.exists(sumogma_tmp1):
    gp.Delete_management(sumogma_tmp1)
    print 'deleting selected cflb'
phrase7 = "[SPC1] <> \\"
try:
    gp.select_analysis(union_all, sumogma_tmp1, phrase7)
    print 'running the ogma selection'
except:
    print gp.GetMessages()

# read in the lu names and place them in a list
lu_list = []
curObj = gp.SearchCursor(sumogma_tmp1)
rowObj = curObj.next()
while rowObj:
    if ((rowObj.getValue('LU_NAME')) not in lu_list) and (rowObj.getValue('LU_NAME') <> ""):
        lu_list.append(rowObj.getValue('LU_NAME'))
    rowObj = curObj.next()
print lu_list

# now run stats command and create stats table to collect BEC Variant areas for each LU areas
lubec_list = []
tmp_list = []
for luname in lu_list:
    lubec_list.append(luname)
    # select all the BEC variants for the landscape unit
    result_tmp1 = workarea + "interm_results.mdb\\sel1_tmp3"
    if gp.exists(result_tmp1):
        gp.Delete_management(result_tmp1)
        print 'deleting selection'
    phrase1 = "[LU_NAME] = \" + luname + "\"

```

```

try:
    gp.select_analysis(sumogma_tmp1, result_tmp1, phrase1)
    print 'running the lu selection'
except:
    print gp.GetMessages()
    print 'lu selection died'

# dump selection to stats file summarized by bec variant
sum_tbl0 = workarea + "interm_results.mdb\\sum_ogma_ind3"
#kill table if it already exists
if gp.exists(sum_tbl0):
    gp.Delete_management(sum_tbl0)
    print 'deleting ogma summary table'
try:
    print 'running ogma summary stats'
    gp.Statistics_analysis(result_tmp1, sum_tbl0, "Geometry_Area SUM", "BECLABEL")
except:
    print gp.GetMessages()
    print 'No ogma summary stats table created'
#now dump to tmp list
tmp_list = []
curObj = gp.SearchCursor(sum_tbl0)
rowObj = curObj.next()
while rowObj:
    tmp_list.append(rowObj.getValue('BECLABEL'))
    tmp_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
    rowObj = curObj.next()
    print tmp_list

lubec_list.append(tmp_list)

lubec_results = workarea + "results\\indicator2_03.txt"
file_hand = open(lubec_results,'w')
for x in lubec_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

print lubec_list

# create an empty list for the bec variant and area values
result_list = []
#create an empty list -- this will be a 2 dimentional list
final_list = []
#=====

```

```

# cycle through landscape units and calculate seral stage areas for each bec variant
#calculate early and place in list
for name in lu_list:
    print 'lu is ', name
    result_list = []
    result_list.append(name)
    bec_keylist = []
    curObj = gp.SearchCursor(sumogma_tmp1)
    rowObj = curObj.next()
    while rowObj:
        if (((rowObj.getValue('BECLABEL')) not in bec_keylist) and ((rowObj.getValue('LU_NAME')) ==
name)):
            bec_keylist.append(rowObj.getValue('BECLABEL'))
            rowObj = curObj.next()
    print 'bec variant list for', name, ' is', bec_keylist
    #now run the query for mature plus old, looping through the bec variants
    for variant in bec_keylist:
        print 'running mature plus old for', variant
        result_tmp3 = workarea + "interm_results.mdb\\sel_tmp3"
        if gp.exists(result_tmp3):
            gp.Delete_management(result_tmp3)
            print 'deleting selected old class'
        phrase3 = "([ZONE_] = 'CWH' OR [ZONE_] = 'ESSF' OR [ZONE_] = 'ICH' OR [ZONE_] =
'MH')"
        phrase4 = " AND [AGE_PRJ] > 250"
        phrase5 = " AND ([LU_NAME] = '\" + name + "\'"
        phrase6 = " AND ([BECLABEL] = '\" + variant + "\'"
        phrase7 = phrase3 + phrase4 + phrase5 + phrase6
        try:
            #    print 'result_tmp3 is', result_tmp3
            gp.select_analysis(sumogma_tmp1, result_tmp3, phrase7)
            print 'running the old selection'
        except:
            print gp.GetMessages()
            print 'no select run'

# now run stats command and create stats table
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting stats table'
try:
#    if gp.exists(result_tmp3):
    print 'running old stats for', name, variant
    gp.Statistics_analysis(result_tmp3, out_tbl1, "GEOMETRY_Area SUM", "BECLABEL")

```

```
except:
    print gp.GetMessages()
    print 'old stats table didn"t work'

#now read stats table and save it to results_list
curObj = gp.SearchCursor(out_tbl1)
rowObj = curObj.next()
while rowObj:
#     print 'The BEC zone is ', rowObj.getValue('BECLABEL')
    result_list.append(rowObj.getValue('BECLABEL'))
#     print 'the area is', rowObj.getValue('SUM_GEOMETRY_Area')
    result_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
    rowObj = curObj.next()
print result_list

final_list.append(result_list)

print final_list

final_results = workarea + "results\\indicator_03.txt"
file_hand = open(final_results,'w')
for x in final_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

Indicator 5: Percent of OGMA with forest harvesting.

```
# -----
# bio_ind5.py
# Scripting Dates: August 15, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates percent of OGMA with forest harvesting
# Data sources are personal geodatabases
# -----

#====Section 1 =====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

# =====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")
    ogma_fclass = raw_input("Enter the name of the OGMA feature class")
    harvest_fclass = raw_input("Enter the name of the harvested feature class")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    ogma_fclass = workarea + "kmbiodiversity.mdb\\ogma_may07"
    harvest_fclass = workarea + "kmbiodiversity.mdb\\harvested_Erase"
    resultant_dir = workarea + "results\\"

#set up interm resultant file names and paths
interm_pgdb = workarea + "interm_results.mdb"
```

```

#-----
#calculate the area of the ogmas
# run stats command and create summary stats table
sum_tbl1 = workarea + "results\\sum_ogma_ind5.dbf"
#kill table if it already exists
if gp.exists(sum_tbl1):
    gp.Delete_management(sum_tbl1)
    print 'deleting ogma summary table'
try:
    print 'running total ogma summary stats'
    gp.Statistics_analysis(ogma_fclass, sum_tbl1, "Shape_Area SUM", "OGMA")
except:
    print gp.GetMessages()
    print 'No ogma summary stats table created'
#-----
#write the results to the results_list
results_list = []
results_list.append('ogma total')
curObj = gp.SearchCursor(sum_tbl1)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_'))
    rowObj = curObj.next()
print 'results list is ', results_list
#-----
# intersect ogmas with harvested areas
harv_ogmas = workarea + "interm_results.mdb\\harv_ogma"
#kill fclass if it already exists
if gp.exists(harv_ogmas):
    gp.Delete_management(harv_ogmas)
    print 'deleting ogma and harvest intersect'
try:
    gp.intersect_analysis(ogma_fclass + ";" + harvest_fclass, harv_ogmas, "ALL")
    print 'intersect worked'
except:
    print gp.GetMessages()

#-----
# run stats command to calculate areas from intersect
#-----
#calculate the area of the ogmas harvested
# run stats command and create summary stats table
sum_tbl1 = workarea + "results\\sum_harvogma_ind5.dbf"
#kill table if it already exists
if gp.exists(sum_tbl1):
    gp.Delete_management(sum_tbl1)
    print 'deleting ogma summary table'
try:

```

```

    print 'running harvested ogma summary stats'
    gp.Statistics_analysis(harv_ogmas, sum_tbl1, "Shape_Area SUM", "OGMA")
except:
    print gp.GetMessages()
    print 'No ogma summary stats table created'
#-----
#write the results to the results_list
results_list.append('harvested total')
curObj = gp.SearchCursor(sum_tbl1)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_'))
    rowObj = curObj.next()
print 'results list is ', results_list
#-----
#write results_list to a txt file
#the file will be overwritten if it exists
final_results = workarea + "results\\indicator_05.txt"
file_hand = open(final_results,'w')
for x in results_list:
    tmp_list = str(x)
    file_hand.writelines(tmp_list)
    file_hand.write('\n')
file_hand.close()

```

Indicator 6: Percent or area (ha) of OGMA harvested for allowable disturbances.

```
# -----  
# bio_ind6.py  
# Scripting Dates: August 15, 2007  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates areas of individual ogmas harvested  
# Data sources are personal geodatabases  
# -----  
  
#====Section 1 =====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")  
  
# Load required toolboxes...  
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")  
#=====  
  
# =====Section 2 =====  
# allow user to enter path to personal geodatabase  
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or  
N"))  
if option_var == "N":  
    # allow user to enter names of various geodatabases and feature classes required  
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the  
biodiversity data -->")  
    resultant_pgd = raw_input("Enter the path and name to the Personal Geodatabase that holds the  
results of the analysis -->")  
    ogma_fclass = raw_input("Enter the name of the OGMA feature class")  
    harvest_fclass = raw_input("Enter the name of the harvested feature class")  
else:  
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"  
    ogma_fclass = workarea + "kmbiodiversity.mdb\\ogma_may07"  
    harvest_fclass = workarea + "kmbiodiversity.mdb\\harvested_Erase"  
    resultant_pgd = workarea + "final_results.mdb"  
  
#set up interm resultant file names and paths  
interm_pgd = workarea + "interm_results.mdb"
```

```

#-----
#write the ogma areas to the results_list, which is a list of tuples
results_list = []
curObj = gp.SearchCursor(ogma_fclass)
rowObj = curObj.next()
while rowObj:
    value1=(rowObj.getValue('OGMA_20061'))
    value2=(rowObj.getValue('Shape_Area'))
    tuplelist = (value1,value2)
    results_list.append(tuplelist)
    rowObj = curObj.next()
print 'results list is ', results_list
#-----
#eliminate the areas of the ogmas that overlap with the harvested areas
harv_elim = workarea + "interm_results.mdb\\harv_elim_tmp1"
if gp.exists(harv_elim):
    gp.delete_management(harv_elim)
    print 'deleting harv_elim'
try:
    gp.erase_analysis(ogma_fclass, harvest_fclass, harv_elim, "")
    print 'running erase'
except:
    print gp.GetMessages()

#-----
#write the ogmas nos. and areas to the results_list
results_list2 = []
curObj = gp.SearchCursor(harv_elim)
rowObj = curObj.next()
while rowObj:
    value1=(rowObj.getValue('OGMA_20061'))
    value2=(rowObj.getValue('Shape_Area'))
    tuplelist = (value1,value2)
    results_list2.append(tuplelist)
    rowObj = curObj.next()
print 'results2 list is ', results_list2

#-----
#the file will be overwritten if it exists
final_results = workarea + "results\\ind_06_list1.txt"
file_hand = open(final_results,'w')
for x in results_list:
    tmp_list = str(x)
    file_hand.writelines(tmp_list)
    file_hand.write('\n')

```

```
file_hand.close()
print 'results written to txt file'

#the file will be overwritten if it exists
final_results = workarea + "results\\ind_06_list2.txt"
file_hand = open(final_results,'w')
for x in results_list2:
    tmp_list = str(x)
    file_hand.writelines(tmp_list)
    file_hand.write('\n')
file_hand.close()
print 'results written to second txt file'
```

Indicator 7: Percent or area (ha) added to OGMAs to replace disturbed areas.

```
# -----
# bio_ind7.py
# Scripting Dates: August 15, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates areas added to ogmas to replace disturbed areas
# Data sources are personal geodatabases
# -----

#====Section 1 =====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

# =====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")
    ogma_fclassv2 = raw_input("Enter the name of the original ogma feature class")
    ogma_fclassv1 = raw_input("Enter the name of the updated ogma feature class")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    ogma_fclassv1 = workarea + "kmbiodiversity.mdb\\ogma_may07"
    ogma_fclassv2 = workarea + "kmbiodiversity.mdb\\ogma_feb07"
    resultant_dir = workarea + "results\\"

#-----
#make copy of ogma feature classes and add unique id fields for each fclass
```

```

ogma_fcvar1 = workarea + "interm_results.mdb\\ogma_tmp1"
ogma_fcvar2 = workarea + "interm_results.mdb\\ogma_tmp2"
if gp.exists(ogma_fcvar1):
    gp.Delete_management(ogma_fcvar1)
    print 'deleting copied table'
try:
    gp.copyfeatures(ogma_fclassv1, ogma_fcvar1)
    print 'copied first ogma fclass'
except:
    print gp.GetMessages()

if gp.exists(ogma_fcvar2):
    gp.Delete_management(ogma_fcvar2)
    print 'deleting copied table'

gp.copyfeatures(ogma_fclassv2, ogma_fcvar2)
print 'copies first ogma fclass'

#create new fields in new feature classes to preserve the versionid
# both fclasses currently do not have the field names to identify version
#field names will be ogma_ver=may2007 and ogma_ver=ogma2007
gp.addfield(ogma_fcvar1, "OGMA_VER", "TEXT", "#", "#", "10")
print 'added first field'
try:
    gp.calculatefield_management(ogma_fcvar1, "OGMA_VER", ""UPDATED_VER")
    print 'first calculate field worked'
except:
    print gp.GetMessages(0)

gp.addfield(ogma_fcvar2, "OGMA_VER", "TEXT", "#", "#", "10")
print 'added second first field'
try:
    gp.calculatefield_management(ogma_fcvar2, "OGMA_VER", ""ORIGINAL_VER")
    print 'second calculate field worked'
except:
    print gp.GetMessages(0)

#run symmetrical difference
symdiff_var = workarea + "interm_results.mdb\\symdif_tmp1"
if gp.exists(symdiff_var):
    gp.delete_management(symdiff_var)
try:
    gp.SymDiff(ogma_fcvar1, ogma_fcvar2, symdiff_var)
    print 'sym difference worked'
except:
    print gp.GetMessages()
#-----
#calculate OGMA_VER = OGMA_VER_1 where OGMA_VER <> UPDATED_VER

```

```

curObj = gp.UpdateCursor(symdiff_var)
rowObj = curObj.next()
while rowObj:
    if(rowObj.getValue('OGMA_VER'))<> 'UPDATED_VER':
        rowObj.OGMA_VER = rowObj.OGMA_VER_1
        print 'updating atributes'
        curObj.updaterow(rowObj)
        rowObj = curObj.next()

#-----
#calculate the area of the ogmas
#run stats on sym diff tool, using shape_area and ogma_ver as summary fields
# run stats command and create summary stats table
sum_tbl1 = workarea + "results\\sum_ogma_ind6.dbf"
#kill table if it already exists
if gp.exists(sum_tbl1):
    gp.Delete_management(sum_tbl1)
    print 'deleting ogma summary table'
try:
    print 'running total ogma summary stats'
    gp.Statistics_analysis(symdiff_var, sum_tbl1, "Shape_Area SUM", "OGMA_VER")
except:
    print gp.GetMessages()
    print 'No ogma summary stats table created'
#
#-----
#write the results to the results_list
results_list = []
curObj = gp.SearchCursor(sum_tbl1)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('OGMA_VER'))
    results_list.append(rowObj.getValue('SUM_Shape_'))
    rowObj = curObj.next()
print 'results list is ', results_list

#dump to text file
#-----
#write results_list to a txt file
#the file will be overwritten if it exists
final_results = workarea + "results\\indicator_07.txt"
file_hand = open(final_results,'w')
for x in results_list:
    tmp_list = str(x)
    file_hand.writelines(tmp_list)
    file_hand.write('\n')
file_hand.close()

```

Indicator 8: Percent of cutblock area or cutblock aggregate retained as Wildlife Tree Patches, by landscape unit by BEC subzone.

The percent total for this indicator is taken from the BEC breakdown within the Landscape Unit.

```
#-----
# bio_ind8.py
# Scripting Dates: August 15, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates % of cutblock retained as WTP by LU by BEC subzone
# Data sources are personal geodatabases
# -----

#====Section 1====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

#====Section 2====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")
    harvest_fclass = raw_input("Enter the name of the harvested areas feature class")
    wtp_fclass = raw_input("Enter the name of the reserves feature class")
    lu_fclass = raw_input("Enter the name of the Landscape Unit feature class -->")
    bec_fclass = raw_input("Enter name of the Biogeoclimatic Zones feature class -->")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    harvest_fclass = workarea + "kmbiodiversity.mdb\\harvested_Erase"
```

```

wtp_fclass = workarea + "kmbiodiversity.mdb\\reserve_wtp"
lu_fclass = workarea + "kmbiodiversity.mdb\\tlu_kmv4"
bec_fclass = workarea + "kmbiodiversity.mdb\\abec_kmv2"
resultant_pgd = workarea + "final_results.mdb"
#-----
#calculate the bec subzone field: zone_ + subzone
# make copy of bec, add field, and calculate
bec_var = workarea + "interm_results.mdb\\bec_tmp1"
if gp.exists(bec_var):
    gp.Delete_management(bec_var)
    print 'deleting copied table'
try:
    gp.copyfeatures(bec_fclass, bec_var)
    print 'copied bec fclass'
except:
    print gp.GetMessages()

# add field to copied fclass
gp.addfield(bec_var, "BEC_SUBZONE", "TEXT", "#", "#", "20")
print 'added first field'
try:
    gp.calculatefield_management(bec_var, "BEC_SUBZONE", "[ZONE_] + [SUBZONE]")
    print 'calculate field worked'
except:
    print gp.GetMessages()
    print 'calc field did not work'

#combine LUs and BEC subzones
lubec_var = workarea + "interm_results.mdb\\lu_bectmp1"
if gp.exists(lubec_var):
    gp.delete_management(lubec_var)
# perform the union
try:
    gp.union_analysis(lu_fclass + ";" + bec_var, lubec_var, "ALL" )
    print 'union of lu and bec'
except:
    print gp.GetMessages()
    print 'union did not work'

#get rid of sliver polygons
lubec_var2 = workarea + "interm_results.mdb\\lubecwtp_tmp2"
if gp.exists(lubec_var2):
    gp.delete_management(lubec_var2)
phrase10 = "[LU_NAME] <> " or [BEC_SUBZONE] = ""
try:
    gp.select_analysis(lubec_var, lubec_var2, phrase10)
    print 'rid of slivers'
except:

```

```

print gp.GetMessages()
print 'no sliver select run'

#now add the wtps into the mix
harv_wtp = workarea + "interm_results.mdb\\harv_wtp"
if gp.exists(harv_wtp):
    gp.delete_management(harv_wtp)
# perform the union
try:
    gp.union_analysis(harvest_fclass + ";" + wtp_fclass, harv_wtp, "ALL" )
    print 'union of wtp with harvested areas'
except:
    print gp.GetMessages()
    print 'union did not work'

# cut the lu x bec union to include only cut block areas
union_clip= workarea + "interm_results.mdb\\union_tmp1"
if gp.exists(union_clip):
    gp.delete_management(union_clip)
try:
    gp.intersect_analysis(lubec_var2 + ";" + harv_wtp, union_clip, "ALL")
    print 'intersect with all layers'
except:
    print gp.GetMessages()
    print 'intersect all didn't work'

#create a list of lus from the unioned resultant
lu_list = []
curObj = gp.SearchCursor(union_clip)
rowObj = curObj.next()
while rowObj:
    if ((rowObj.getValue('LU_NAME')) not in lu_list) and (rowObj.getValue('LU_NAME') <> ""):
        lu_list.append(rowObj.getValue('LU_NAME'))
        rowObj = curObj.next()
#print lu_list

final_list = []
for name in lu_list:
    print 'lu is ', name
    result_list = []
    result_list.append(name)
    result_list.append('wtp data')
    result_list2 = []
    result_list2.append(name)
    result_list2.append('cutblock data')
    bec_keylist = []
    curObj = gp.SearchCursor(union_clip)
    rowObj = curObj.next()

```

```

while rowObj:
    if (((rowObj.getValue('BEC_SUBZONE')) not in bec_keylist) and
((rowObj.getValue('LU_NAME')) == name)):
        bec_keylist.append(rowObj.getValue('BEC_SUBZONE'))
        rowObj = curObj.next()
print 'bec subzone list for', name, ' is', bec_keylist

#now run the query for calculating areas, looping through the bec variants
for subzone in bec_keylist:
    print 'calculating wtp areas in subzone', subzone
    result_tmp8 = workarea + "interm_results.mdb\\sel3_tmp8"
    if gp.exists(result_tmp8):
        gp.Delete_management(result_tmp8)
        print 'deleting selected class'
    phrase3 = "([WTP_STATUS] = \'WTP\')"
    phrase4 = " AND ([LU_NAME] = \'\" + name + "\')"
    phrase5 = " AND ([BEC_SUBZONE] = \'\" + subzone + "\')"
    phrase6 = phrase3 + phrase4 + phrase5
    print 'phrase 6 is', phrase6
    try:
        gp.select_analysis(union_clip, result_tmp8, phrase6)
        print 'running the wtp selection'
    except:
        print gp.GetMessages()
        print 'no select run'

# also calculating how much cutblock is in the bec subzone
result_tmp7 = workarea + "interm_results.mdb\\sel2_tmp8"
if gp.exists(result_tmp7):
    gp.Delete_management(result_tmp7)
    print 'deleting selected cblock class'
    phrase7 = "([LU_NAME] = \'\" + name + "\')"
    phrase8 = " AND ([BEC_SUBZONE] = \'\" + subzone + "\')"
    phrase9 = phrase7 + phrase8
    print 'phrase9 is', phrase9
    try:
        gp.select_analysis(union_clip, result_tmp7, phrase9)
        print 'running the cblock selection'
        count = gp.GetCount_management(result_tmp7)
        print 'select cblock count is', count
    except:
        print gp.GetMessages()
        print 'cblock selection did not work'

# now run stats command for wtps and create stats table
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl1"

```

```

#kill table if it already exists
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting stats table'
try:
#    if gp.exists(result_tmp3):
        print 'running wtp stats for', name, subzone
        gp.Statistics_analysis(result_tmp8, out_tbl1, "GEOMETRY_Area SUM", "BEC_SUBZONE")
except:
    print gp.GetMessages()
    print 'stats table didn't work'

# now run stats command for cutblocks and create stats table
out_tbl2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl2"
#kill table if it already exists
if gp.exists(out_tbl2):
    gp.Delete_management(out_tbl2)
    print 'deleting stats table'
try:
#    if gp.exists(result_tmp3):
        print 'running cblock stats for', name, subzone
        gp.Statistics_analysis(result_tmp7, out_tbl2, "GEOMETRY_Area SUM", "BEC_SUBZONE")
except:
    print gp.GetMessages()
    print 'stats table didn't work'

#now read wtp stats table and save it to results_list
if (gp.exists(out_tbl1)) and (gp.GetCount_management(out_tbl1)>0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BEC_SUBZONE'))
        result_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
        rowObj = curObj.next()

else:
    result_list.append(subzone)
    result_list.append('NOAREA')
print 'result_list is', result_list

#now read cutblock stats table and save it to results_list
curObj = gp.SearchCursor(out_tbl2)
rowObj = curObj.next()
while rowObj:
    result_list2.append(rowObj.getValue('BEC_SUBZONE'))
    result_list2.append(rowObj.getValue('SUM_GEOMETRY_Area'))

```

```
    rowObj = curObj.next()
    print 'result_list2 is', result_list2
```

```
    final_list.append(result_list)
    final_list.append(result_list2)
```

```
print final_list
```

```
final_results = workarea + "results\\indicator_08.txt"
file_hand = open(final_results,'w')
for x in final_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

Indicator 9: Percent species composition by landscape unit by BEC subzone.

The wording of this indicator was changed to Percent leading species composition by Landscape Unit by BEC subzone.

Each polygon has leading and secondary leading species, as well as a field representing the percent of the species within the polygon. The final area results multiply the area of the polygon for the leading species by the percentage of that species within the polygon. This calculation is carried out within the script.

The results are grouped as follows:

Fir = True Fir (B), Amabilis Fir (BA), Alpine Fir (BL), Douglas Fir (FD)

Hemlock = Hemlock (H), Western Hemlock (WH), Mountain Hemlock (MH)

Spruce = Spruce (S), Sitka Spruce (SS), White Spruce (SW), Spruce crosses (SX)

Cedar = Yellow Cedar (YC)

Pine = Pinus contorta (PL)

E = Birch

AT = Aspen

AC = Cottonwood

CW = Western Red Cedar

DR = Red Alder

LW = Western Larch

```
#-----
# bio_ind9.py
# Scripting Dates: August 21, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates % leading and secondary species by LU by BEC subzone
# Data sources are personal geodatabases
# -----

#====Section 1====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
#gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====
```

```

# =====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or
N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the
biodiversity data -->")
    cflb_fclass = raw_input("Enter the name of the CFLB feature class")
    lu_fclass = raw_input("Enter the name of the Landscape Unit feature class -->")
    bec_fclass = raw_input("Enter name of the Biogeoclimatic Zones feature class -->")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the
analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    cflb_fclass = workarea + "kmbiodiversity.mdb\\cflb_kmv2"
    lu_fclass = workarea + "kmbiodiversity.mdb\\tlu_kmv4"
    bec_fclass = workarea + "kmbiodiversity.mdb\\abec_kmv2"
    resultant_pgdb = workarea + "final_results.mdb"
#-----
#calculate the bec subzone field: zone_ + subzone
# make copy of bec, add field, and calculate
bec_var = workarea + "interm_results.mdb\\bec_tmp1"
if gp.exists(bec_var):
    gp.Delete_management(bec_var)
    print 'deleting copied table'
try:
    gp.copyfeatures(bec_fclass, bec_var)
    print 'copied bec fclass'
except:
    print gp.GetMessages()

# add field to copied fclass
gp.addfield(bec_var, "BEC_SUBZONE", "TEXT", "#", "#", "20")
print 'added first field'
try:
    gp.calculatefield_management(bec_var, "BEC_SUBZONE", "[ZONE_] + [SUBZONE]")
    print 'calculate field worked'
except:
    print gp.GetMessages()
    print 'calc field did not work'

#combine LUs and BEC subzones
lubec_var = workarea + "interm_results.mdb\\lu_bectmp1"
if gp.exists(lubec_var):
    gp.delete_management(lubec_var)
# perform the union

```

```

try:
    gp.union_analysis(lu_fclass + ";" + bec_var, lubec_var, "ALL" )
    print 'union of lu and bec'
except:
    print gp.GetMessages()
    print 'union did not work'

#get rid of sliver polygons
lubec_var2 = workarea + "interm_results.mdb\\lubec_tmp2"
if gp.exists(lubec_var2):
    gp.delete_management(lubec_var2)
phrase10 = "[LU_NAME] <> " or [BEC_SUBZONE] = ""
try:
    gp.select_analysis(lubec_var, lubec_var2, phrase10)
    print 'rid of slivers'
except:
    print gp.GetMessages()
    print 'no sliver select run'

# work in the cflb
union_all = workarea + "interm_results.mdb\\union_alltmp1"
if gp.exists(union_all):
    gp.delete_management(union_all)
try:
    gp.union_analysis(cflb_fclass + ";" + lubec_var2, union_all, "ALL")
except:
    print gp.GetMessages(2)
    print 'all union did not work'

#get rid of garbage in species codes - some sliver fields have empty or null values
union_allv2 = workarea + "interm_results.mdb\\union_alltmp2"
if gp.exists(union_allv2):
    gp.delete_management(union_allv2)

phrase11 = "[SPC1] <> '' and [SPC1] <> " and [SPC1] <> NULL"
#OR ([PCT1] <> NULL)"
#print 'phrase11', phrase11
try:
    gp.Select_analysis(union_all, union_allv2, "([SPC1] <> NULL AND [SPC1] <> '' AND [SPC1] <>
)") AND ([PCT1] <> NULL)"
# gp.select_analysis(union_all, union_allv2, phrase11)
    print 'rid of empty species codes'
except:
    print gp.GetMessages()
    print 'empty species codes still intact'

#add fields to calculate leading species area
gp.addfield(union_allv2, "SP1_AREA", "DOUBLE")

```

```

print 'added lspecies area field'
try:
    gp.calculatefield_management(union_allv2, "SP1_AREA", "[Geometry_Area] * ([PCT1] / 100)")
    print 'calculate sp1_area field worked'
except:
    print gp.GetMessages()
    print 'calc field did not work'

gp.addfield(union_allv2, "SP2_AREA", "DOUBLE")
print 'added second lspecies area field'

# add fields to calculate secondary leading species
# do this using a cursor because sometimes leading species field is not blank but secondary species is
curObj = gp.updateCursor(union_allv2)
rowObj = curObj.next()
while rowObj:
    if (rowObj.getValue('SPC2') <> ' '):
        rowObj.SP2_AREA = rowObj.Geometry_Area * (rowObj.PCT2 * 0.01)
        curObj.updateRow(rowObj)
        print 'sp2 area is ', rowObj.SP2_AREA
        rowObj = curObj.next()
    else:
        rowObj.SP2_AREA = 0
        curObj.updateRow(rowObj)
        rowObj = curObj.next()

print 'updated spc2 areas'

#set up loops for lu and bec
#create a list of lus from the unioned resultant
lu_list = []
curObj = gp.SearchCursor(union_allv2)
rowObj = curObj.next()
while rowObj:
    if ((rowObj.getValue('LU_NAME')) not in lu_list) and (rowObj.getValue('LU_NAME') <> ""):
        lu_list.append(rowObj.getValue('LU_NAME'))
    rowObj = curObj.next()
print lu_list

final_list = []
final2_list = []
#total_list = []
#total2_list = []
for name in lu_list:
    print 'lu is ', name
    bec_keylist = []
    curObj = gp.SearchCursor(union_allv2)
    rowObj = curObj.next()

```

```

while rowObj:
    if (((rowObj.getValue('BEC_SUBZONE')) not in bec_keylist) and
((rowObj.getValue('LU_NAME')) == name)):
        bec_keylist.append(rowObj.getValue('BEC_SUBZONE'))
        rowObj = curObj.next()
print 'bec subzone list for', name, ' is', bec_keylist

#now run the query for calculating areas, looping through the bec variants
for subzone in bec_keylist:
    result_list = []
    result2_list = []
    result_list.append(name)
    result2_list.append(name)
    print 'calculating lspecies areas in subzone', subzone
    result_tmp1 = workarea + "interm_results.mdb\\sel1_tmp9"
    if gp.exists(result_tmp1):
        gp.Delete_management(result_tmp1)
        print 'deleting selected class'
    phrase1 = "([LU_NAME] = '\" + name + '\")"
    phrase2 = " AND ([BEC_SUBZONE] = '\" + subzone + '\")"
    phrase3 = phrase1 + phrase2
    try:
        gp.select_analysis(union_allv2, result_tmp1, phrase3)
        print 'running the selection'
    except:
        print gp.GetMessages()
        print 'no select run'

# now run stats for full area of bec subzone, ls1 and lp2
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats9_tbl1"
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting stats table'
try:
    print 'running area stats for bec subzone', subzone
    gp.Statistics_analysis(result_tmp1, out_tbl1, "GEOMETRY_Area SUM", "BEC_SUBZONE")
except:
    print gp.GetMessages()
    print 'stats table didn't work'

#run stats again for ls1
out_tbl2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats9_tbl2"
if gp.exists(out_tbl2):
    gp.Delete_management(out_tbl2)

```

```

    print 'deleting stats table'
try:
    print 'running areae stats for ls1'
    gp.Statistics_analysis(result_tmp1, out_tbl2, "SP1_Area SUM", "SPC1")
except:
    print gp.GetMessages()
    print 'stats table for ls1 didn"t work'

#run stats again for ls2
out_tbl3 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats9_tbl3"
if gp.exists(out_tbl3):
    gp.Delete_management(out_tbl3)
    print 'deleting stats table'
try:
    print 'running area stats for ls2'
    gp.Statistics_analysis(result_tmp1, out_tbl3, "SP2_Area SUM", "SPC2")
except:
    print gp.GetMessages()
    print 'stats table for ls2 didn"t work'

#now read beczone stats table and save it to results_list
curObj = gp.SearchCursor(out_tbl1)
rowObj = curObj.next()
while rowObj:
    result_list.append(rowObj.getValue('BEC_SUBZONE'))
    result_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
    rowObj = curObj.next()

#now read lu1 stats table and save it to results_list
result_list.append('sp1')
curObj = gp.SearchCursor(out_tbl2)
rowObj = curObj.next()
while rowObj:
    result_list.append(rowObj.getValue('SPC1'))
    result_list.append(rowObj.getValue('SUM_SP1_Area'))
    rowObj = curObj.next()

#save LP2 to a separate file to reduce length of .txt file
#now read beczone stats table and save it to results_list
curObj = gp.SearchCursor(out_tbl1)
rowObj = curObj.next()
while rowObj:
    result2_list.append(rowObj.getValue('BEC_SUBZONE'))
    result2_list.append(rowObj.getValue('SUM_GEOMETRY_Area'))
    rowObj = curObj.next()

```

```
#now read lu2 stats table and save it to results_list
result2_list.append('sp2')
curObj = gp.SearchCursor(out_tbl3)
rowObj = curObj.next()
while rowObj:
    result2_list.append(rowObj.getValue('SPC2'))
    result2_list.append(rowObj.getValue('SUM_SP2_Area'))
    rowObj = curObj.next()
print 'result2_list is', result2_list
```

```
final_list.append(result_list)
final2_list.append(result2_list)
```

```
print 'final 1 list is', final_list
print 'final 2 list is ', final2_list
```

```
final_results = workarea + "results\\indicator_09a.txt"
file_hand = open(final_results,'w')
for x in final_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

```
final2_results = workarea + "results\\indicator_09b.txt"
file_hand = open(final2_results,'w')
for x in final2_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

Indicator 10: Percent distribution by patch size by natural disturbance type by landscape unit by BEC variant.

The results for this indicator for the SRMP area are presented within the Kalum SRMP document. In this Kalum SLIMP project, the analysis run for this indicator covers the entire LRMP area. Although the SRMP indicator used Forest Cover and the LRMP indicator utilized VRI, the results should be similar for landscapes that fall both within the SRMP and LRMP areas.

The definition for Natural Disturbance Types is taken from the Biodiversity Guidebook.

Age Class breakdown used:

<u>Age</u>	<u>Class</u>
0-15	1
16-39	2
40 – 80	3
81 – 100	4
101 – 120	5
121 – 140	6
141 – 180	7
181 – 250	8
250+	9

Patch size classes (ha):

< 40
40 to 80
81 to 250
250 +

```
# -----  
# bio_ind10.py  
# Scripting Dates: July 30th  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates patch size by NDT by LU  
# Data sources are personal geodatabases  
# -----
```

```
#=====Section 1 =====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")
```

```

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

# =====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the data directory -->")
    lu_fclass = raw_input("Enter the name of the Landscape Unit feature class -->")
    ndt_fclass = raw_input("Enter the name of the feature class containing NDTs")
    patch_fclass = raw_input("Enter the name of the Generalized Crown Forested Land Base -->")
else:
    lu_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\tlu_kmv4"
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    ndt_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\abec_ndt"
    patch_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\patch_diss"

#set up interm resultant file names and paths
interm_pgdb =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b"

#=====Section 3 =====

#union the lu with the NDT data
# into the interm resultant feature class lu_bec
lubec_var = interm_pgdb + "\\lu_bec"
if gp.exists(lubec_var):
    gp.Delete_management(lubec_var)
    print 'deleting resultant lubec_var'
#perform the union
try:
    gp.union_analysis(lu_fclass + ";" + ndt_fclass, lubec_var, "ALL" )
    print 'union of lu and ndt and bec'
except:
    print gp.GetMessages()
    print 'union did not work'

```

```

#intersect the patches with lu and ndt
cflb_var = interm_pgdb + "\\cflb_tmp"
if gp.exists(cflb_var):
    gp.Delete_management(cflb_var)
    print 'deleting cflb_var'
try:
    gp.intersect_analysis(patch_fclass + ";" + lubec_var, cflb_var, "ALL")
    print 'intersection of cflb with bec and lu worked'
except:
    print gp.GetMessages()
#=====

complete_list = []
ndt_list = ['NDT1', 'NDT2', 'NDT5']
#=====
# cycle through NDTs and calculate patch distribution for each LU

for ndt in ndt_list:
    #set up loop for
    lu_list = []
    curObj = gp.SearchCursor(cflb_var)
    rowObj = curObj.next()
    while rowObj:
        if ((rowObj.getValue('LU_NAME') not in lu_list) and (rowObj.getValue('NDT_CODE1') == ndt)):
            lu_list.append(rowObj.getValue('LU_NAME'))
            rowObj = curObj.next()
    print 'lu list for', ndt, ' is', lu_list

#now run the query for calculating areas, looping through the lu's
for lu in lu_list:
    result_list = []
    result_list.append(ndt)
    result_list.append(lu)
    #caculate area of cflb within lu within ndt
    result_tmp1 = workarea + "interm_results.mdb\\sel1_tmp1"
    if gp.exists(result_tmp1):
        gp.Delete_management(result_tmp1)
        print 'deleting selected class'
    phrase01 = "([LU_NAME] = \" + lu + "\")"
    phrase02 = " AND ([NDT_CODE1] = \" + ndt + "\")"
#    phrase03 = " AND [SPC1] <> ""
    phrase04 = phrase01 + phrase02
    try:
        gp.select_analysis(cflb_var, result_tmp1, phrase04)
        print 'running the cflb selection'
    except:
        print gp.GetMessages()

```

```

    print 'no cflb select run'

# now run stats to calculate area of lu
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl1"
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting cflb stats table'
try:
    gp.Statistics_analysis(result_tmp1, out_tbl1, "Shape_Area SUM", "BECLABEL")
    print 'running area stats for cflb in', lu, ndt
except:
    print gp.GetMessages()
    print 'stats table didn't work'

# pull out patches < 40 ha
result_tmp2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\sel2_tmp"
if gp.exists(result_tmp2):
    gp.Delete_management(result_tmp2)
    print 'deleting 40m patch size'
phrase05 = "[Shape_Area] < 400000"
phrase06 = " AND ([LU_NAME] = '\" + lu + '\" ) AND ([NDT_CODE1] = '\" + ndt + '\" )"
phrase07 = phrase05 + phrase06
try:
    gp.select_analysis(cflb_var, result_tmp2, phrase07)
    print 'running the < 40 ha selection'
except:
    print gp.GetMessages()
    print 'no 40 selection'

# now run stats command and create stats table
out_tbl2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl2):
    gp.Delete_management(out_tbl2)
    print 'deleting < 40 ha table'
try:
    gp.Statistics_analysis(result_tmp2, out_tbl2, "Shape_Area SUM", "BECLABEL")
    print 'running 40 ha stats'
except:
    print gp.GetMessages()
    print 'no 40 ha stats table created'

```

```

#now read lu stats table and save it to results_list
if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    result_list.append('lu areas')
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append('no lu areas')

#dump < 40 ha values to list.
if (gp.Exists(out_tbl2)) and (gp.Getcount_management(out_tbl2) > 0):
    result_list.append('less 40 ha')
    curObj = gp.SearchCursor(out_tbl2)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append("less 40 ha")
#    result_list.append(variant)
    result_list.append('no value')

#now run the query for 40 to 80
print 'running 40 to 80'
result_tmp3 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\sel3tmp"
if gp.exists(result_tmp3):
    gp.Delete_management(result_tmp3)
    print 'deleting 40 to 80 class'
phrase08 = "([Shape_Area] >= 400000 AND [Shape_Area] <= 800000)"
phrase09 = " AND ([LU_NAME] = '\" + lu + "\') AND ([NDT_CODE1] = '\" + ndt + "\')"
phrase10 = phrase08 + phrase09
try:
    gp.select_analysis(cflb_var, result_tmp3, phrase10)
    print 'running the 40 to 80 selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl3 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\seral_stats"
#kill table if it already exists

```

```

if gp.exists(out_tbl3):
    gp.Delete_management(out_tbl3)
    print 'deleting 40 to 80 stats table'
try:
    print 'running 40 to 80 stats'
    gp.Statistics_analysis(result_tmp3, out_tbl3, "Shape_Area SUM", "BECLABEL")
except:
    print gp.GetMessages()
    print 'No stats table created for 40 to 80'

#now cursor through the resulting .dbf and dump values into a list
if (gp.Exists(out_tbl3)) and (gp.Getcount_management(out_tbl3) > 0):
    result_list.append('40 to 80')
    curObj = gp.SearchCursor(out_tbl3)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append('40 to 80')
# result_list.append(variant)
result_list.append("NOLABEL")

#selection and stats for 80 to 250
result_tmp4 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\sel4_tmp"
if gp.exists(result_tmp4):
    gp.Delete_management(result_tmp4)
    print 'deleting selected class'
phrase11 = "[Shape_Area] > 800000 AND [Shape_Area] <= 2500000)"
phrase12 = " AND ([LU_NAME] = \" + lu + "\" AND ([NDT_CODE1] = \" + ndt + "\")"
phrase13 = phrase11 + phrase12
try:
    gp.select_analysis(cflb_var, result_tmp4, phrase13)
    print 'running the 80 to 250 selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl4 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl4):
    gp.Delete_management(out_tbl4)
    print 'deleting 80 to 250 stats table'

```

```

try:
    print 'running 80 to 250 stats'
    gp.Statistics_analysis(result_tmp4, out_tbl4, "Shape_Area SUM", "BECLABEL")
except:
    #if an error occurred while running a tool print the message
    print gp.GetMessages()
    print ' stats table didn't work'

if (gp.Exists(out_tbl4)) and (gp.Getcount_management(out_tbl4) > 0):
    result_list.append('80 to 250')
    curObj = gp.SearchCursor(out_tbl4)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
    else:
        result_list.append('80 to 250')
#
    result_list.append(variant)
    result_list.append("NOLABEL")

# final patch is 250 plus
result_tmp5 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\sel5_tmp"
if gp.exists(result_tmp5):
    gp.Delete_management(result_tmp5)
    print 'deleting selected 250 class'
phrase14 = "[Shape_Area] > 2500000"
phrase15 = " AND ([LU_NAME] = '\" + lu + '\" ) AND ([NDT_CODE1] = '\" + ndt + '\" )"
phrase16 = phrase14 + phrase15
try:
    gp.select_analysis(cflb_var, result_tmp5, phrase16)
    print 'running the 250 selection'
except:
    print gp.GetMessages()
    print 'selection for 250 not working'

# now run stats command and create stats table
out_tbl5 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\seral_stats"
#kill table if it already exists
if gp.exists(out_tbl5):
    gp.Delete_management(out_tbl5)
    print 'deleting 250 stats table'
try:
    print 'running 250 stats'

```

```

gp.Statistics_analysis(result_tmp5, out_tbl5, "Shape_Area SUM", "BECLABEL")
except:
    #if an error occurred while running a tool print the message
    print gp.GetMessages()
    print ' stats table didn't work'

if (gp.Exists(out_tbl5)) and (gp.Getcount_management(out_tbl5) > 0):
    result_list.append('250')
    curObj = gp.SearchCursor(out_tbl5)
    rowObj = curObj.next()
    while rowObj:
        result_list.append(rowObj.getValue('BECLABEL'))
        result_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    result_list.append('250')
#    result_list.append(variant)
    result_list.append("NOLABEL")

    print result_list

complete_list.append(result_list)
print complete_list

#=====
# write results to a .CSV file
#the file will be overwritten if it exists
final_results = workarea + "results\\indicator_10.txt"
file_hand = open(final_results,'w')
for x in complete_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

```

Indicator 11: Amount of harvesting within polygon “A” on Map 5 (SRMP), the pass between Kiteen and Cedar drainages.

Indicators 11, 12, 15 and 19 all calculate the amount harvested, but vary by area of interest. One script was written to calculate the results for all four indicators.

```
#-----
# bio_ind11_12_15_19.py
# Scripting Dates: August 21, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates % harvested within polygon A, polygon B, Subzone1 Lakelse SRMZ and Upper
# Kitsumkalum
# Data sources are personal geodatabases
# -----

#====Section 1 =====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

# =====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Enter Y to use default data paths or enter N to use different
paths"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    workarea = raw_input("Enter the path and name of the Personal Geodatabase that holds the
biodiversity data -->")
    harv_fclass = raw_input("Enter the name of the harvested areas feature class")
    polya_fclass = raw_input("Enter the name of the Polygon A (map 5) feature class -->")
    polyb_fclass = raw_input("Enter name of the Polygon B feature class -->")
    lakelse_fclass = raw_input("Enter name of the Lakelse SRMZ feature class -->")
    kits_fclass = raw_input("Enter name of the Kitsumkalum SRMZ feature class -->")
    cflb_fclass = raw_input("Enter the name of the CFLB feature class")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the
analysis -->")
```

```

if option_var == "Y":
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    harv_fclass = workarea + "kmbiodiversity.mdb\\harvested_Erase"
    poly_a_fclass = workarea + "kmadmin.mdb\\poly_a"
    poly_b_fclass = workarea + "kmadmin.mdb\\poly_b"
    lakelse_fclass = workarea + "kmadmin.mdb\\lakelse_diss"
    kits_fclass = workarea + "kmadmin.mdb\\kitsum_v2"
    cflb_fclass = workarea + "kmbiodiversity.mdb\\cflb_kmv2"
    resultant_pgd = workarea + "final_results.mdb"
else:
    print 'Run program again - input invalid'

#-----

#set up list for results
results_list = []

#calc harvested within poly a
polya_harv = workarea + "interm_results.mdb\\polya_tmp1"
if gp.exists(polya_harv):
    gp.delete_management(polya_harv)
try:
    gp.intersect_analysis(polya_fclass + ";" + harv_fclass, polya_harv, "ALL")
    print 'intersect polya with harvested'
except:
    print gp.GetMessages()

# now run stats command for harvested areas
out_tbl11 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl11"
#kill table if it already exists
if gp.exists(out_tbl11):
    gp.Delete_management(out_tbl11)
    print 'deleting stats polya table'
try:
    print 'running stats for poly a'
    gp.Statistics_analysis(polya_harv, out_tbl11, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn't work for poly a'

results_list.append('harvested area poly a')
if (gp.exists(out_tbl11)) and (gp.GetCount_management(out_tbl11)>0):
    curObj = gp.SearchCursor(out_tbl11)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Shape_Area'))

```

```

        rowObj = curObj.next()
else:
    results_list.append('novalue')

#select out cflb within polygon a
#clip CFLB to polygon a only
polya_cflb = workarea + "interm_results.mdb\\polya_clip"
if gp.exists(polya_cflb):
    gp.Delete_management(polya_cflb)
    print 'deleting clipped cflb'
try:
    gp.clip_analysis(cflb_fclass, polya_fclass, polya_cflb)
    print 'clip worked'
except:
    print gp.GetMessages()

#summarize total cflb within poly a
out_tbl1b =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl11b"
#kill table if it already exists
if gp.exists(out_tbl1b):
    gp.Delete_management(out_tbl1b)
    print 'deleting stats polya cflb table'
try:
    print 'running stats for poly a cflb'
    gp.Statistics_analysis(polya_cflb, out_tbl1b, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for poly a'

#add results to the list
results_list.append('poly_a total area')
curObj = gp.SearchCursor(out_tbl1b)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()
#-----
# now do same for polygon b
#calc harvested within poly b
polyb_harv = workarea + "interm_results.mdb\\polyb_tmp1"
if gp.exists(polyb_harv):
    gp.delete_management(polyb_harv)
try:
    gp.intersect_analysis(polyb_fclass + ";" + harv_fclass, polyb_harv, "ALL")
    print 'intersect polyb with harvested'
except:

```

```

print gp.GetMessages()

out_tbl12 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl12"
#kill table if it already exists
if gp.exists(out_tbl12):
    gp.Delete_management(out_tbl12)
    print 'deleting stats polyb table'
try:
    print 'running stats for poly b'
    gp.Statistics_analysis(polyb_harv, out_tbl12, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn't work for poly b'

results_list.append('harvested area poly b')
if (gp.exists(out_tbl12)) and (gp.GetCount_management(out_tbl12)>0):
    curObj = gp.SearchCursor(out_tbl12)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    results_list.append('novalue')

#select out cflb within polygon b
#clip CFLB to polygon b only
polyb_cflb = workarea + "interm_results.mdb\\polyb_clip"
if gp.exists(polyb_cflb):
    gp.Delete_management(polyb_cflb)
    print 'deleting clipped cflb'
try:
    gp.clip_analysis(cflb_fclass, polyb_fclass, polyb_cflb)
    print 'clip worked'
except:
    print gp.GetMessages()

#summarize total cflb within poly b
out_tbl12b =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl12b"
#kill table if it already exists
if gp.exists(out_tbl12b):
    gp.Delete_management(out_tbl12b)
    print 'deleting stats polyb cflb table'
try:
    print 'running stats for poly b cflb'

```

```

gp.Statistics_analysis(polyb_cflb, out_tbl12b, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for poly b'

#add results to the list
results_list.append('poly_b total area')
curObj = gp.SearchCursor(out_tbl12b)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()

#-----
#now do same for subzone 1 Lakelse
#need to select the polygon first
srmz1_sel = workarea + "interm_results.mdb\\srmz1_tmp1"
if gp.exists(srmz1_sel):
    gp.delete_management(srmz1_sel)
gp.Select_analysis(lakelse_fclass, srmz1_sel, "[PRIMARY_ZONE] = 'SRM_LAKELSE1'")
print 'select worked'

srmz_harv = workarea + "interm_results.mdb\\srmz1_tmp2"
if gp.exists(srmz_harv):
    gp.delete_management(srmz_harv)
try:
    gp.intersect_analysis(srmz1_sel + ";" + harv_fclass, srmz_harv, "ALL")
    print 'intersect srmz1 with harvested'
except:
    print gp.GetMessages()

out_tbl15 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl15"
#kill table if it already exists
if gp.exists(out_tbl15):
    gp.Delete_management(out_tbl15)
    print 'deleting stats 15 table'
try:
    print 'running stats for srmz1'
    gp.Statistics_analysis(srmz_harv, out_tbl15, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for srmz1'

results_list.append('harvested area srmz1')
if (gp.exists(out_tbl15)) and (gp.GetCount_management(out_tbl15)>0):
    curObj = gp.SearchCursor(out_tbl15)

```

```

rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()
else:
    results_list.append('novalue')

#select out cflb within polygon srm1
#clip CFLB to srmz only
srmz1_cflb = workarea + "interm_results.mdb\\srmz1_clip"
if gp.exists(srmz1_cflb):
    gp.Delete_management(srmz1_cflb)
    print 'deleting clipped cflb'
try:
    gp.clip_analysis(cflb_fclass, srmz1_sel, srmz1_cflb)
    print 'clip worked'
except:
    print gp.GetMessages()

#summarize total cflb within srmz1
out_tbl15b =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl15b"
#kill table if it already exists
if gp.exists(out_tbl15b):
    gp.Delete_management(out_tbl15b)
    print 'deleting stats srmz1 cflb table'
try:
    print 'running stats for srmz1 cflb'
    gp.Statistics_analysis(srmz1_cflb, out_tbl15b, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn't work for srmz1'

#add results to the list
results_list.append('srmz1')
results_list.append('total area')
curObj = gp.SearchCursor(out_tbl15b)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()

#-----
#do same for upper kitsumkalum
#calc harvested within poly a
kits_harv = workarea + "interm_results.mdb\\kits_tmp1"
if gp.exists(kits_harv):

```

```

    gp.delete_management(kits_harv)
try:
    gp.intersect_analysis(kits_fclass + ";" + harv_fclass, kits_harv, "ALL")
    print 'intersect kits with harvested'
except:
    print gp.GetMessages()

# now run stats command for harvested areas
out_tbl19 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\stats_tbl19"
#kill table if it already exists
if gp.exists(out_tbl19):
    gp.Delete_management(out_tbl19)
    print 'deleting stats kits table'
try:
    print 'running stats for kits'
    gp.Statistics_analysis(kits_harv, out_tbl19, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for kits'

results_list.append('harvested area kits')
if (gp.exists(out_tbl19)) and (gp.GetCount_management(out_tbl19)>0):
    curObj = gp.SearchCursor(out_tbl19)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    results_list.append('novalue')

#select out cflb within kits
#clip CFLB to kits only
kits_cflb = workarea + "interm_results.mdb\kits_clip"
if gp.exists(kits_cflb):
    gp.Delete_management(kits_cflb)
    print 'deleting clipped cflb'
try:
    gp.clip_analysis(cflb_fclass, kits_fclass, kits_cflb)
    print 'clip worked'
except:
    print gp.GetMessages()

#summarize total cflb within poly a
out_tbl19b =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\stats_tbl19b"

```

```

#kill table if it already exists
if gp.exists(out_tbl19b):
    gp.Delete_management(out_tbl19b)
    print 'deleting stats kits cflb table'
try:
    print 'running stats for kits cflb'
    gp.Statistics_analysis(kits_cflb, out_tbl19b, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for kits'

#add results to the list
results_list.append('kitstotal area')
curObj = gp.SearchCursor(out_tbl19b)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()

print results_list

final_results = workarea + "results\\indicator_011.txt"
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

```

Indicator 12: Percent of harvested area within polygon “B”

See notes for Indicator 11.

Indicator 13: Percent of area in old and mature seral stages in the level pass between the Williams and Thomas/Clore watersheds identified on Map 5.

```
#-----  
# bio_ind13.py  
# Scripting Dates: August 23, 2007  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates % old and mature within level pass polygon  
# Data sources are personal geodatabases  
#-----  
  
#====Section 1=====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")  
  
# Load required toolboxes...  
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")  
#=====  
  
#====Section 2=====  
# allow user to enter path to personal geodatabase  
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))  
if option_var == "N":  
    # allow user to enter names of various geodatabases and feature classes required  
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")  
    lpass_fclass = raw_input("Enter the name of the level pass feature class")  
    cflb_fclass = raw_input("Enter the name of the CFLB feature class -->")  
    bec_fclass = raw_input("Enter the name of the BEC feature class -->")  
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")  
else:  
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"  
    cflb_fclass = workarea + "kmbiodiversity.mdb\\cflb_kmv2"  
    bec_fclass = workarea + "kmbiodiversity.mdb\\abec_kmv2"  
    lpass_fclass = workarea + "kmadmin.mdb\\level_pass"  
    resultant_pgd = workarea + "final_results.mdb"
```

```

#-----

#set up list for results
results_list = []

#intersect the cflb with level pass
cflb_lp= workarea + "interm_results.mdb\\lpass_tmp1"
if gp.exists(cflb_lp):
    gp.delete_management(cflb_lp)
try:
    gp.intersect_analysis(cflb_fclass + ";" + lpass_fclass, cflb_lp, "ALL")
    print 'intersect with all layers'
except:
    print gp.GetMessages()
    print 'intersect all didn't work'

#throw in the BEC
int_all = workarea + "interm_results.mdb\\union_tmp1"
if gp.exists(int_all):
    gp.delete_management(int_all)
try:
    gp.intersect_analysis(bec_fclass + ";" + cflb_lp, int_all, "ALL")
    print 'intersect with bec layers'
except:
    print gp.GetMessages()
    print 'intersect all didn't work'

#now run the query for mature plus old
print 'running mature plus old'
result_tmp1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\sel_tmp13"
if gp.exists(result_tmp1):
    gp.Delete_management(result_tmp1)
    print 'deleting selected class'
phrase1 = "([ZONE_] = 'CWH' AND [AGE_PRJ] > 80)"
phrase2 = " OR ([ZONE_] = 'ESSF' AND [AGE_PRJ] > 120)"
phrase3 = " OR ([ZONE_] = 'ICH' AND [AGE_PRJ] > 100)"
phrase4 = " OR ([ZONE_] = 'MH' AND [AGE_PRJ] > 120))"
phrase6 = phrase1 + phrase2 + phrase3 + phrase4
try:
    gp.select_analysis(int_all, result_tmp1, phrase6)
    print 'running the mature plus old selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl1 =

```

```

"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tmp13"
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting mature plus old stats table'
try:
    print 'running mat + old stats'
    gp.Statistics_analysis(result_tmp1, out_tbl1, "Geometry_Area SUM")
except:
    print gp.GetMessages()
    print 'No stats table created for mat + old'

#now cursor through the resulting .dbf and dump values into a list
results_list.append('old and mature')
if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Area'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

#summarize CFLB within area of lpass
out_tbl13b =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tmp13b"
if gp.exists(out_tbl13b):
    gp.Delete_management(out_tbl13b)
    print 'deleting summary stats table'
try:
    gp.Statistics_analysis(cflb_lp, out_tbl13b, "Shape_Area SUM")
except:
    print gp.GetMessages()

results_list.append('lpassarea cflb')
curObj = gp.SearchCursor(out_tbl13b)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()

print results_list

final_results = workarea + "results\\indicator_013.txt"
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))

```

```
file_hand.write('\n')  
file_hand.close()
```

Indicator 14: Amount of harvesting in the Skeena Islands.

All analysis considers only the CFLB within the Skeena Islands.

```
#-----
# bio_ind14.py
# Scripting Dates: August 27, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates % harvested within Skeena Islands
# Data sources are personal geodatabases
# -----

#====Section 1 =====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

#====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")
    harv_fclass = raw_input("Enter the name of the harvested areas feature class")
    skisland_fclass = raw_input("Enter name of the Skeena Islands feature class -->")
    cflb_fclass = raw_input("Enter the name of the CFLB feature class")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    harv_fclass = workarea + "kmbiodiversity.mdb\\harvested_Erase"
    skisland_fclass = workarea + "kmadmin.mdb\\sk_islands_v2"
    cflb_fclass = workarea + "kmbiodiversity.mdb\\cflb_kmv2"
```

```

    resultant_pgd = workarea + "final_results.mdb"
#-----

#set up list for results
results_list = []

#calc harvested within SI
skis_harv = workarea + "interm_results.mdb\\skis_tmp1"
if gp.exists(skis_harv):
    gp.delete_management(skis_harv)
try:
    gp.intersect_analysis(skisland_fclass + ";" + harv_fclass, skis_harv, "ALL")
    print 'intersect skis with harvested'
except:
    print gp.GetMessages()

# now run stats command for harvested and create stats table
out_tbl1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl14"
#kill table if it already exists
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting stats skis table'
try:
    print 'running stats for skis'
    gp.Statistics_analysis(skis_harv, out_tbl1, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for skis'

results_list.append('harvested area')
if (gp.exists(out_tbl1)) and (gp.GetCount_management(out_tbl1)>0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    results_list.append('novalue')

# calculate total CFLB within SKIS
skis_cflb = workarea + "interm_results.mdb\\skis_clip"
if gp.exists(skis_cflb):
    gp.Delete_management(skis_cflb)
    print 'deleting skis cflb'
try:
    gp.clip_analysis(cflb_fclass, skisland_fclass, skis_cflb)

```

```

    print 'clip skis to cflb worked'
except:
    print gp.GetMessages()

# now run stats command
out_tbl2 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\stats_tbl2"
#kill table if it already exists
if gp.exists(out_tbl2):
    gp.Delete_management(out_tbl2)
    print 'deleting stats summary skis table'
try:
    print 'running stats for skis'
    gp.Statistics_analysis(skis_cflb, out_tbl2, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'stats table didn"t work for skis'

results_list.append('total cflb area')
if (gp.exists(out_tbl2)) and (gp.GetCount_management(out_tbl2)>0):
    curObj = gp.SearchCursor(out_tbl2)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Shape_Area'))
        rowObj = curObj.next()
else:
    results_list.append('novalue')

print results_list

final_results = workarea + "results\\indicator_014.txt"
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

```

Indicator 15: Percent of area harvested within Subzone 1, Lakelse River SRMZ

See Indicator 11.

Indicator 16: Percent of early seral within Subzone 2, Lakelse River SRMZ

```
#-----  
# bio_ind16.py  
# Scripting Dates: August 23, 2007  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates % early in subzone2 lakelse river  
# Data sources are personal geodatabases  
# -----  
  
#====Section 1 =====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")  
  
# Load required toolboxes...  
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")  
#-----  
  
# =====Section 2 =====  
# allow user to enter path to personal geodatabase  
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))  
if option_var == "N":  
    # allow user to enter names of various geodatabases and feature classes required  
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")  
    srmz2_fclass = raw_input("Enter the name of the Lakelse River SRMZ2 feature class")  
    cflb_fclass = raw_input("Enter the name of the CFLB feature class -->")  
    bec_fclass = raw_input("Enter the name of the BEC feature class -->")  
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")  
else:  
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"  
    cflb_fclass = workarea + "kmbiodiversity.mdb\\cflb_kmv2"  
    bec_fclass = workarea + "kmbiodiversity.mdb\\abec_kmv2"  
    srmz2_fclass = workarea + "kadmin.mdb\\lakelse_diss"  
    resultant_pgd = workarea + "final_results.mdb"  
#-----
```

```

#need to select the polygon first
srmz2_sel = workarea + "interm_results.mdb\\srmz2_tmp1"
if gp.exists(srmz2_sel):
    gp.delete_management(srmz2_sel)
try:
    gp.Select_analysis(srmz2_fclass, srmz2_sel, "[PRIMARY_ZONE] = 'SRM_LAKELSE2'")
except:
    print gp.GetMessages()

#set up list for results
results_list = []

#intersect the cflb with srmz2
cflb_var = workarea + "interm_results.mdb\\srmz2_tmp2"
if gp.exists(cflb_var):
    gp.delete_management(cflb_var)
try:
    gp.intersect_analysis(cflb_fclass + ";" + srmz2_sel, cflb_var, "ALL")
    print 'intersect with all layers'
except:
    print gp.GetMessages()
    print 'intersect all didn't work'

#throw in the BEC
int_all = workarea + "interm_results.mdb\\union_tmp1"
if gp.exists(int_all):
    gp.delete_management(int_all)
try:
    gp.intersect_analysis(bec_fclass + ";" + cflb_var, int_all, "ALL")
    print 'intersect bec with all layers'
except:
    print gp.GetMessages()
    print 'intersect all didn't work'

#now run the query for early
print 'running early'
result_tmp1 =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\interm_results.md
b\\sel_tmp16"
if gp.exists(result_tmp1):
    gp.Delete_management(result_tmp1)
    print 'deleting selected class'
phrase1 = "([ZONE_] = 'CWH' or [ZONE_] = 'ESSF' or [ZONE_] = 'ICH' or [ZONE_] = 'MH')"
phrase2 = " AND [AGE_PRJ] < 40"
phrase3 = phrase1 + phrase2
try:
    gp.select_analysis(int_all, result_tmp1, phrase3)

```

```

    print 'running the early selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl1 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.md
b\stats_tmp1"
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting early table'
try:
    print 'early stats'
    gp.Statistics_analysis(result_tmp1, out_tbl1, "Geometry_Area SUM")
except:
    print gp.GetMessages()
    print 'No stats table created for seral'

#now cursor through the selection result and dump values into a list
results_list.append('early')
if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Area'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

#pull out cflb within srmz2
# now run stats command and create stats table
out_tbl2 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.md
b\stats_tmp2"
if gp.exists(out_tbl2):
    gp.Delete_management(out_tbl2)
    print 'deleting summary table'
try:
    print 'sum stats'
    gp.Statistics_analysis(cflb_var, out_tbl2, "Shape_Area SUM")
except:
    print gp.GetMessages()
    print 'No stats table created for seral'

#pull out area of srmz2
results_list.append('srmz2 area')
curObj = gp.SearchCursor(out_tbl2)
rowObj = curObj.next()

```

```
while rowObj:  
    results_list.append(rowObj.getValue('SUM_Shape_Area'))  
    rowObj = curObj.next()
```

```
print results_list
```

```
final_results = workarea + "results\\indicator_016.txt"  
file_hand = open(final_results,'w')  
for x in results_list:  
    file_hand.writelines(str(x))  
    file_hand.write('\n')  
file_hand.close()
```

Indicator 17: Size (ha) of openings within Subzone 2, Lakelse River SRMZ

```
#-----
# bio_ind17.py
# Scripting Dates: August 23, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates size of openings (not cutblocks) within srmz2 Lakelse
# Data sources are personal geodatabases
# -----

#====Section 1 =====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

# =====Section 2 =====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")
    srmz2_fclass = raw_input("Enter the name of the Lakelse River SRMZ2 feature class")
    open_fclass = raw_input("Enter the name of the Openings feature class -->")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    open_fclass = workarea + "kadmin.mdb\\openings_km"
    srmz2_fclass = workarea + "kadmin.mdb\\lakelse_diss"
    resultant_pgdb = workarea + "final_results.mdb"
#-----

#need to select the polygon first
```

```

srmz2_sel = workarea + "interm_results.mdb\\srmz2_tmp1"
if gp.exists(srmz2_sel):
    gp.delete_management(srmz2_sel)
try:
    gp.Select_analysis(srmz2_fclass, srmz2_sel, "[PRIMARY_ZONE] = 'SRM_LAKELSE2'")
except:
    print gp.GetMessages()

#cut the openings to the srmz
open_clip = workarea + "interm_results.mdb\\open_tmp1"
if gp.exists(open_clip):
    gp.Delete_management(open_clip)
    print 'deleting clipped cflb'
try:
    gp.clip_analysis(open_fclass, srmz2_fclass, open_clip)
    print 'clip worked'
except:
    print gp.GetMessages()

#set up list for results
results_list = []

#cycle through result and note opening areas
results_list.append('areas of openings:')
if (gp.Exists(open_clip)) and (gp.Getcount_management(open_clip) > 0):
    curObj = gp.SearchCursor(open_clip)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('Geometry_Area'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

print results_list

final_results = workarea + "results\\indicator_17.txt"
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

```

Indicator 18: Percent retention within cut blocks within Subzone 2, Lakelse River SRMZ

```
#-----  
# bio_ind18.py  
# Scripting Dates: August 23, 2007  
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC  
# Project: Kalum LRMP Indicators  
# Calculates percent retention within openings in srmz2 Lakelse  
# Data sources are personal geodatabases  
# -----  
  
#====Section 1====  
# Import system modules  
import sys, string, os, win32com.client  
  
# Create the Geoprocessor object  
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")  
  
# Set the necessary product code  
gp.SetProduct("ArcInfo")  
  
# Load required toolboxes...  
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")  
#====  
  
#====Section 2====  
# allow user to enter path to personal geodatabase  
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y or N"))  
if option_var == "N":  
    # allow user to enter names of various geodatabases and feature classes required  
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the biodiversity data -->")  
    srmz2_fclass = raw_input("Enter the name of the Lakelse River SRMZ2 feature class")  
    res_fclass = raw_input("Enter the name of the Reserves feature class -->")  
    open_fclass = raw_input("Enter the name of the Openings feature class -->")  
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the analysis -->")  
else:  
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"  
    res_fclass = workarea + "kmbiodiversity.mdb\\reserves"  
    open_fclass = workarea + "kmadmin.mdb\\openings_km"  
    srmz2_fclass = workarea + "kmadmin.mdb\\lakelse_diss"  
    resultant_pgd = workarea + "final_results.mdb"
```

```

#-----

#need to select the polygon first
srmz2_sel = workarea + "interm_results.mdb\\srmz2_tmp1"
if gp.exists(srmz2_sel):
    gp.delete_management(srmz2_sel)
try:
    gp.Select_analysis(srmz2_fclass, srmz2_sel, "[PRIMARY_ZONE] = 'SRM_LAKELSE2'")
except:
    print gp.GetMessages()

#cut the openings to the srmz
open_clip = workarea + "interm_results.mdb\\open_tmp1"
if gp.exists(open_clip):
    gp.Delete_management(open_clip)
    print 'deleting clipped cflb'
try:
    gp.clip_analysis(open_fclass, srmz2_fclass, open_clip)
    print 'clip worked'
except:
    print gp.GetMessages()

#clip the reserves to the openings
#cut the openings to the srmz
res_clip = workarea + "interm_results.mdb\\res_tmp1"
if gp.exists(res_clip):
    gp.Delete_management(res_clip)
    print 'deleting clipped res'
try:
    gp.clip_analysis(res_fclass, srmz2_fclass, res_clip)
    print 'clip worked'
except:
    print gp.GetMessages()

#intersect the clipped openings with reserves
open_res = workarea + "interm_results.mdb\\openres_tmp1"
if gp.exists(open_res):
    gp.Delete_management(open_res)
    print 'deleting res and open intersect'
try:
    gp.intersect_analysis(open_clip + ";" + res_clip, open_res, "ALL")
    print 'intersect worked'
except:
    print gp.GetMessages()

sum_tbl1 = workarea + "interm_results.mdb\\tbl_tmp1"
#kill table if it already exists
if gp.exists(sum_tbl1):

```

```

gp.Delete_management(sum_tbl1)
print 'deleting summary table'
try:
    print 'running summary stats'
    gp.Statistics_analysis(open_res, sum_tbl1, "Geometry_Area SUM")
except:
    print gp.GetMessages()
    print 'No summary stats table created'

sum_tbl2 = workarea + "interm_results.mdb\\tbl_tmp2"
#kill table if it already exists
if gp.exists(sum_tbl2):
    gp.Delete_management(sum_tbl2)
    print 'deleting summary table'
try:
    print 'running summary stats'
    gp.Statistics_analysis(open_clip, sum_tbl2, "Geometry_Area SUM")
except:
    print gp.GetMessages()
    print 'No summary stats table created'

#set up list for results
results_list = []

#cycle through result and pull out retention areas
results_list.append('areas of retention within openings')
if (gp.Exists(sum_tbl1)) and (gp.Getcount_management(sum_tbl1) > 0):
    curObj = gp.SearchCursor(sum_tbl1)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Area'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

results_list.append('areas of openings')
if (gp.Exists(sum_tbl2)) and (gp.Getcount_management(sum_tbl2) > 0):
    curObj = gp.SearchCursor(sum_tbl2)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Area'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

print results_list

final_results = workarea + "results\\indicator_18.txt"

```

```
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

Indicator 19: Amount (ha) of timber harvesting within the Upper Kitsumkalum SRMZ.

See Indicator 11.

Indicator 20: Amount (ha) of industrial activity within the Miligit Valley Sensitive Area.

This indicator was not analysed as the definition of industrial activity provided was too broad for GIS analysis.

Indicator 21: Number of kilometres of roads constructed in riparian areas, wildlife habitat areas and forest ecosystem networks.

No Forest Ecosystem Network data exists.

```
#-----
# bio_ind21.py
# Scripting Dates: August 24, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates length of roads in reserve areas (riparian, wha's, and fens)
#currently no wha data or fen data, only using reserves to represent riparian areas
# Data sources are personal geodatabases
# -----

#====Section 1====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

#====Section 2====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y
for default or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the
biodiversity data -->")
    roads_fclass = raw_input("Enter the name of the roads feature class")
    res_fclass = raw_input("Enter the name of the Reserves feature class -->")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the
analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    res_fclass = workarea + "kmbiodiversity.mdb\\reserves"
    roads_fclass = workarea + "kmadmin.mdb\\roads_km"
```

```

    resultant_pgd = workarea + "final_results.mdb"
#-----

#buffer reserves by 20m
res_20m = workarea + "interm_results.mdb\\res20_tmp1"
if gp.exists(res_20m):
    gp.Delete_management(res_20m)
    print 'deleting 20m res buffer'
try:
    gp.buffer_analysis(res_fclass, res_20m, "20", "#", "#", "#", "#")
    print '20m buffer complete'
except:
    print gp.GetMessages()
    print '20m buffer did not work'

#buffer reserves by 100m
res_100m = workarea + "interm_results.mdb\\res100_tmp1"
if gp.exists(res_100m):
    gp.Delete_management(res_100m)
    print 'deleting 100m res buffer'
try:
    gp.buffer_analysis(res_fclass, res_100m, "100", "#", "#", "#", "#")
    print '100m buffer complete'
except:
    print gp.GetMessages()
    print 'buffer did not work'

#cut roads to reserves
roads_clip = workarea + "interm_results.mdb\\roads_tmp1"
if gp.exists(roads_clip):
    gp.Delete_management(roads_clip)
    print 'deleting clipped roads'
try:
    gp.clip_analysis(roads_fclass, res_fclass, roads_clip)
    print 'clip worked'
except:
    print gp.GetMessages()

#cut roads to 20m buffer
roads20_clip = workarea + "interm_results.mdb\\roads20_tmp1"
if gp.exists(roads20_clip):
    gp.Delete_management(roads20_clip)
    print 'deleting clipped 20m roads'
try:
    gp.clip_analysis(roads_fclass, res_20m, roads20_clip)
    print 'clip worked'
except:
    print gp.GetMessages()

```

```

roads100_clip = workarea + "interm_results.mdb\\roads100_tmp1"
if gp.exists(roads100_clip):
    gp.Delete_management(roads100_clip)
    print 'deleting clipped 100m roads'
try:
    gp.clip_analysis(roads_fclass, res_100m, roads100_clip)
    print 'clip worked'
except:
    print gp.GetMessages()

#sum meters of roads within reserves
sum_tbl1 = workarea + "interm_results.mdb\\tbl_tmp1"
if gp.exists(sum_tbl1):
    gp.Delete_management(sum_tbl1)
    print 'deleting summary table'
try:
    print 'running summary stats'
    gp.Statistics_analysis(roads_clip, sum_tbl1, "Geometry_Length SUM")
except:
    print gp.GetMessages()
    print 'No summary stats table created'

#sum meters of roads including 20m of reserves
sum_tbl2 = workarea + "interm_results.mdb\\tbl_tmp2"
if gp.exists(sum_tbl2):
    gp.Delete_management(sum_tbl2)
    print 'deleting 20m summary table'
try:
    print 'running 20m summary stats'
    gp.Statistics_analysis(roads20_clip, sum_tbl2, "Geometry_Length SUM")
except:
    print gp.GetMessages()
    print 'No 20m stats table created'

#sum meters of roads including 100m within reserves
sum_tbl3 = workarea + "interm_results.mdb\\tbl_tmp3"
if gp.exists(sum_tbl3):
    gp.Delete_management(sum_tbl3)
    print 'deleting 100m summary table'
try:
    print 'running 100m summary stats'
    gp.Statistics_analysis(roads100_clip, sum_tbl3, "Geometry_Length SUM")
except:
    print gp.GetMessages()
    print 'No 20m stats table created'

results_list = []

```

```

#read results from stats files and store in a list
results_list.append('length within reserves')
if (gp.Exists(sum_tbl1)) and (gp.Getcount_management(sum_tbl1) > 0):
    curObj = gp.SearchCursor(sum_tbl1)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Length'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

results_list.append('length within 20m reserves')
if (gp.Exists(sum_tbl2)) and (gp.Getcount_management(sum_tbl2) > 0):
    curObj = gp.SearchCursor(sum_tbl2)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Length'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

results_list.append('length within 100m reserves')
if (gp.Exists(sum_tbl3)) and (gp.Getcount_management(sum_tbl3) > 0):
    curObj = gp.SearchCursor(sum_tbl3)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('SUM_Geometry_Length'))
        rowObj = curObj.next()
else:
    results_list.append('NoValue')

print results_list

final_results = workarea + "results\\indicator_21.txt"
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()

```

Indicator 22: Equivalent clearcut area (ECA) for community watersheds.

ECA methodology is located at <http://www.for.gov.bc.ca/tasb/legsregs/fpc/fcpguide/iwap>

The ECA methodology applies only to watersheds with harvesting over 250 ha. Within the Kalum LRMP, there exists only one community watershed that meets this criteria. Within this watershed, only 0.2 % has been harvested.

In addition, personal conversation with the BCE Regional Hydrologist as well as recommendations from the Watershed Assessment Procedure Guidebook (page 25) indicate that ECA is not a meaningful management indicator. Due to above reasons as well as the small amount of possible harvesting in the watershed, ECA was not calculated.

Existing Community Watersheds:

Deep Creek

Drake Creek

Gitzyon Creek

Wathl Creek

Eneeksagilaguaw Creek

Ksa Muntl Am Hawak Creek

Indicator 24: Amount of mid-seral forest within the forested land base, excluding hardwood, in the McKay-Davis and Copper watersheds.

```
#-----
# bio_ind24.py
# Scripting Dates: August 24, 2007
# Written by Johanna Pfalz, Eclipse GIS, Smithers, BC
# Project: Kalum LRMP Indicators
# Calculates area of mid seral in McKay Davis and Copper Watersheds
## Data sources are personal geodatabases
# -----

#====Section 1====
# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Set the necessary product code
gp.SetProduct("ArcInfo")

# Load required toolboxes...
gp.AddToolbox("E:/sw_nt/arcgis/arcexe9x/Toolboxes/Coverage Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("E:/sw_nt/arcgis/ArcToolbox/Toolboxes/Analysis Tools.tbx")
#=====

#====Section 2====
# allow user to enter path to personal geodatabase
option_var = string.capitalize(raw_input("Do you want to use default data paths or enter your own? Y
for default or N"))
if option_var == "N":
    # allow user to enter names of various geodatabases and feature classes required
    biopgdb_path = raw_input("Enter the path and name of the Personal Geodatabase that holds the
biodiversity data -->")
    bec_fclass = raw_input("Enter name of the Biogeoclimatic Zones feature class -->")
    cflb_fclass = raw_input("Enter the name of the Crown Forested Land Base -->")
    wshed_fclass = raw_input("Enter the name of the Watershed feature class")
    resultant_dir = raw_input("Enter the path and name to the directory that holds the results of the
analysis -->")
else:
    workarea = "W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\"
    bec_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\abec_km"
    cflb_fclass =
```

```

"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kmbiodiversity.m
db\\cflb_km"
md_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kadmin.mdb\\m
d_wshed"
copper_fclass =
"W:\\srm\\smt\\Workarea\\Regional_Planning_Projects\\slimp_P060542\\kalum\\data\\kadmin.mdb\\co
pper_wshed"
#-----

#set up list for results
results_list = []

#need to combine the McKay Davis and Copper Watershed
wshed_sel = workarea + "interm_results.mdb\\wshed_tmp1"
if gp.exists(wshed_sel):
    gp.delete_management(wshed_sel)
try:
#   gp.union_analysis(lu_fclass + ";" + bec_fclass, lubec_var, "ALL" )
    gp.merge(md_fclass + ";" + copper_fclass, wshed_sel)
except:
    print gp.GetMessages()

#intersect the cflb with the watersheds
cflb_var = workarea + "interm_results.mdb\\wshed_tmp2"
if gp.exists(cflb_var):
    gp.delete_management(cflb_var)
try:
    gp.intersect_analysis(cflb_fclass + ";" + wshed_sel, cflb_var, "ALL")
    print 'intersect with all layers'
except:
    print gp.GetMessages()
    print 'intersect all didn"t work'

#throw in the BEC
int_all = workarea + "interm_results.mdb\\union_tmp1"
if gp.exists(int_all):
    gp.delete_management(int_all)
try:
    gp.intersect_analysis(bec_fclass + ";" + cflb_var, int_all, "ALL")
    print 'intersect bec with all layers'
except:
    print gp.GetMessages()
    print 'intersect all didn"t work'

#now run the query for early
print 'running early'
result_tmp1 =

```

```

"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\sel_tmp24"
if gp.exists(result_tmp1):
    gp.Delete_management(result_tmp1)
    print 'deleting selected class'
phrase1 = "([ZONE_] = 'CWH' or [ZONE_] = 'ESSF' or [ZONE_] = 'ICH' or [ZONE_] = 'MH')"
phrase2 = " AND ([AGE_PRJ] > 24 AND [AGE_PRJ] < 101)"
phrase3 = " AND ([SPC1] <> 'AC' AND [SPC1] <> 'DR' AND [SPC1] <> )"
phrase4 = phrase1 + phrase2 + phrase3
try:
    gp.select_analysis(int_all, result_tmp1, phrase4)
    print 'running the early selection'
except:
    print gp.GetMessages()

# now run stats command and create stats table
out_tbl1 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\stats_tmp1"
if gp.exists(out_tbl1):
    gp.Delete_management(out_tbl1)
    print 'deleting early table'
try:
    print 'early stats'
    gp.Statistics_analysis(result_tmp1, out_tbl1, "Geometry_Area SUM", "IWG_NAME")
except:
    print gp.GetMessages()
    print 'No stats table created for seral'

#now cursor through the selection result and dump values into a list
results_list.append('mid seral')
if (gp.Exists(out_tbl1)) and (gp.Getcount_management(out_tbl1) > 0):
    curObj = gp.SearchCursor(out_tbl1)
    rowObj = curObj.next()
    while rowObj:
        results_list.append(rowObj.getValue('IWG_NAME'))
        results_list.append(rowObj.getValue('SUM_Geometry_Area'))
        rowObj = curObj.next()
else:
    results_list.append('No Value')

#pull out cflb within wsheds
out_tbl2 =
"W:\srm\smt\Workarea\Regional_Planning_Projects\slimp_P060542\kalum\data\interm_results.mdb\stats_tmp2"
if gp.exists(out_tbl2):
    gp.Delete_management(out_tbl2)
    print 'deleting summary table'

```

```
try:
    print 'sum stats'
    gp.Statistics_analysis(cflb_var, out_tbl2, "Shape_Area SUM", "IWG_NAME")
except:
    print gp.GetMessages()
    print 'No stats table created for seral'

#pull out area of srmz2
results_list.append('wshed area')
curObj = gp.SearchCursor(out_tbl2)
rowObj = curObj.next()
while rowObj:
    results_list.append(rowObj.getValue('IWG_NAME'))
    results_list.append(rowObj.getValue('SUM_Shape_Area'))
    rowObj = curObj.next()

print results_list

final_results = workarea + "results\\indicator_24.txt"
file_hand = open(final_results,'w')
for x in results_list:
    file_hand.writelines(str(x))
    file_hand.write('\n')
file_hand.close()
```

Indicator 25: Number of stems per hectare within free growing managed forests on rich and wetter sites within watersheds identified on Map 7 (SRMP).

This indicator was no analyzed due to insufficient data.

Indicator 26: Number of bridge crossings on the Upper Copper River upstream of confluence with Limonite Creek at any given time.

This indicator was analysed visually by plotting the bridge crossings on the Upper Copper River.

Indicator 27: Width (metres) of no-harvest reserve along both sides of the Upper Copper River above Limonite Creek.

This indicator was analysed visually by displaying the 100m buffer along the Upper Copper River.

Indicator 28: Size (metres) of harvest openings visible from the Sue Channel/Hawkesbury Island Protected Area.

No analysed with GIS. Recommendation is for ILMB staff to discuss site visit with BCE Parks Staff.