Label and attribute-based topographic point thinning

Paulo Raposo, Cynthia A. Brewer, Lawrence V. Stanislawski

ICA Commission on Generalisation and Multiple Representation,
August 23 & 24, 2013, Dresden, Germany

Basic concept

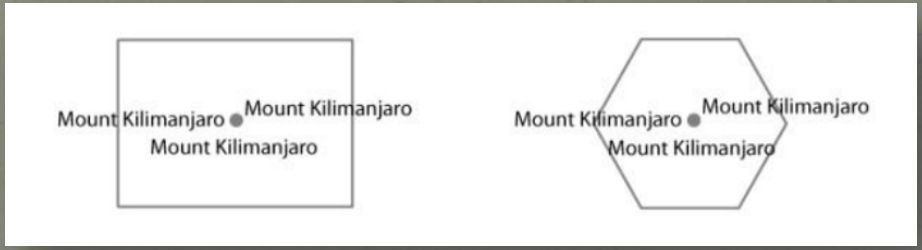
- Many data have only basic attributes
- Various automatable geoprocessing routines can derive new attributes from inter-feature or inter-layer calculations, and use these to drive generalization

Overview – Problem context

- USGS multiscale topographic mapping, anticipating Maplex engine labeling decisions
- As summit points become dense at small scales, we want to simplify the selection & placement problems Maplex will face between summits and other feature types (e.g., towns)
- We want:
  - An objective and automatable importance metric;
  - Objective and automatable generalization.

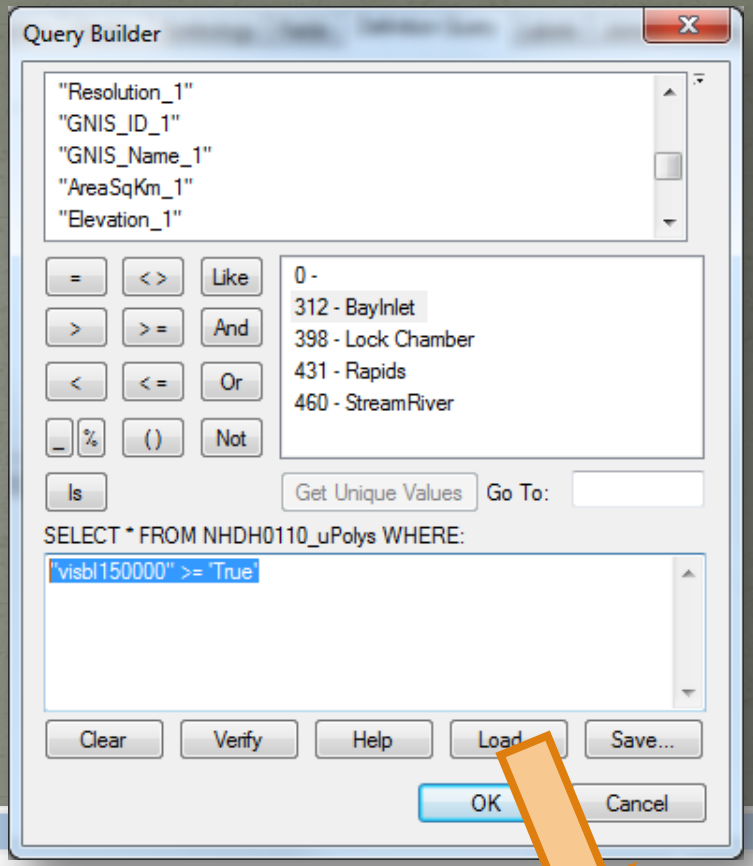
Overview - Rationale

- Will label summits as local space allows:
  - Partition the map space in a rectangular tessellation, with each rectangle approximating a label "neighborhood". One label per neighborhood allowed.
  - Use rectangles 2cm wide at the golden ratio (ϕ), since they approximate a label footprint.



Attribute use

- Use typical SQL queries when mapping to select “visible” summits
- Product is similar to that of “Thin Road Network” tool



Table

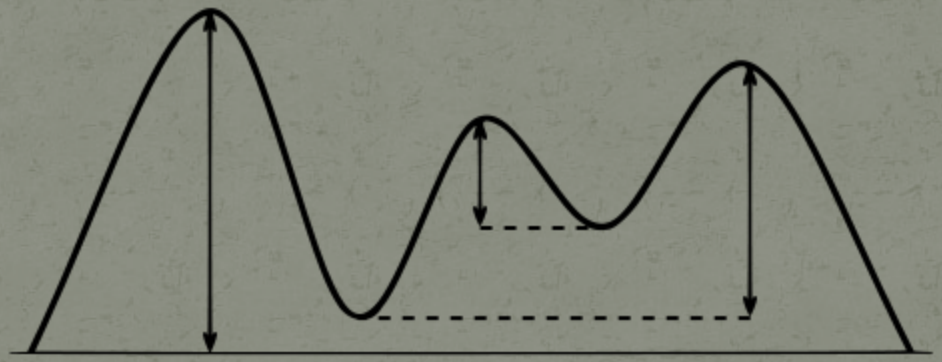
WVsummits_scalethinned

FEATURE_NAME	ELEV_IN_M	Wv_elev_wgs841	Prominence	visbl24000	visbl40000	visbl60000	visbl100000	visbl150000	visbl250000
Big Mountain	1177	1177.442993	82.442993	True	True	True	True	True	<Null>
Bother Knob	1323	1322.696655	2.696655	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>
Brierpatch Mountain	1348	1348.251465	253.251465	True	True	True	True	True	True
Brushy Mountain	1004	1004.035522	164.035522	True	True	True	True	True	True
Brushy Mountain	1258	1258.400146	283.400146	True	True	True	True	True	True
Bulls Head	838	837.54071	177.54071	True	True	True	True	True	True
Castle Mountain	1032	1032.903076	132.903076	True	True	True	True	True	True
Castle Rock	709	706.523193	106.523193	True	True	True	True	True	True
Cave Knob	711	710.976868	65.976868	True	True	True	True	True	<Null>
Cave Mountain	854	853.514648	418.514648	True	True	True	True	True	True

Attribute Enrichment

Summit prominence

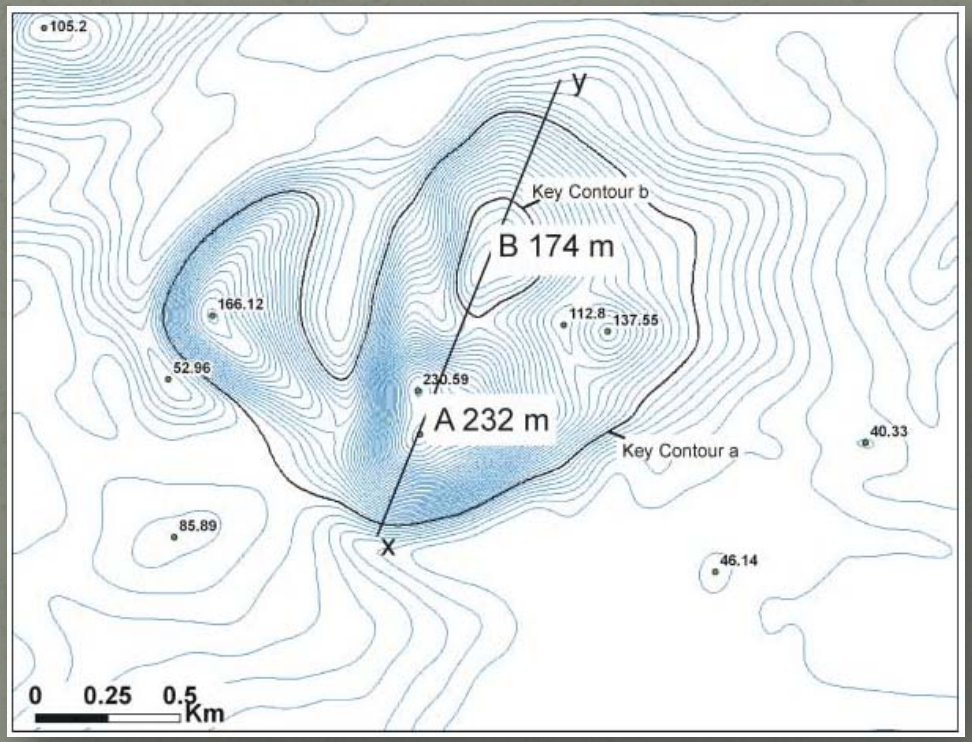
- Use prominence (i.e., local elevation) as criteria for summit selection through scale



- Inputs: DEM, GNIS summit points

Summit prominence - procedure

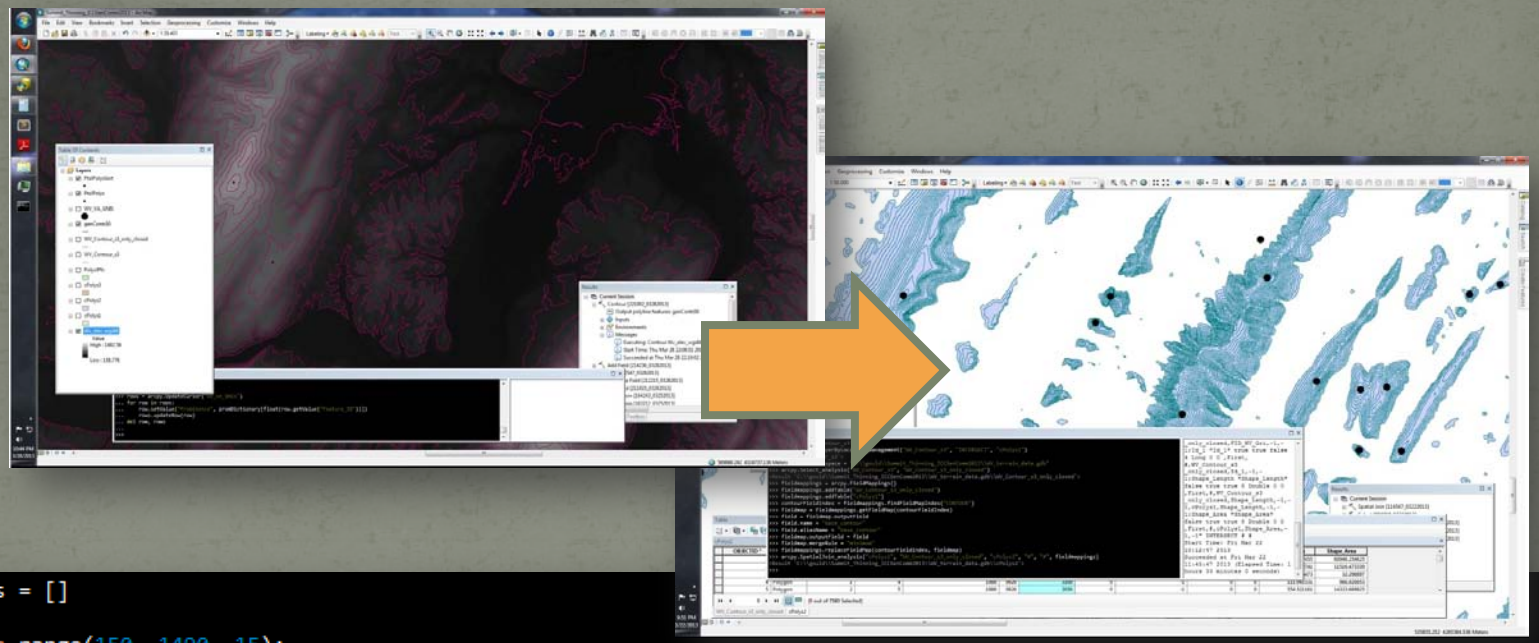
- Determine prominence using Chaudhry & Mackaness† method



† Chaudhry, O. Z., & Mackaness, W. A. (2008). Creating Mountains out of Mole Hills: Automatic Identification of Hills and Ranges Using Morphometric Analysis. *Transactions in GIS, 12(5), 567-589.*

Summit prominence - procedure

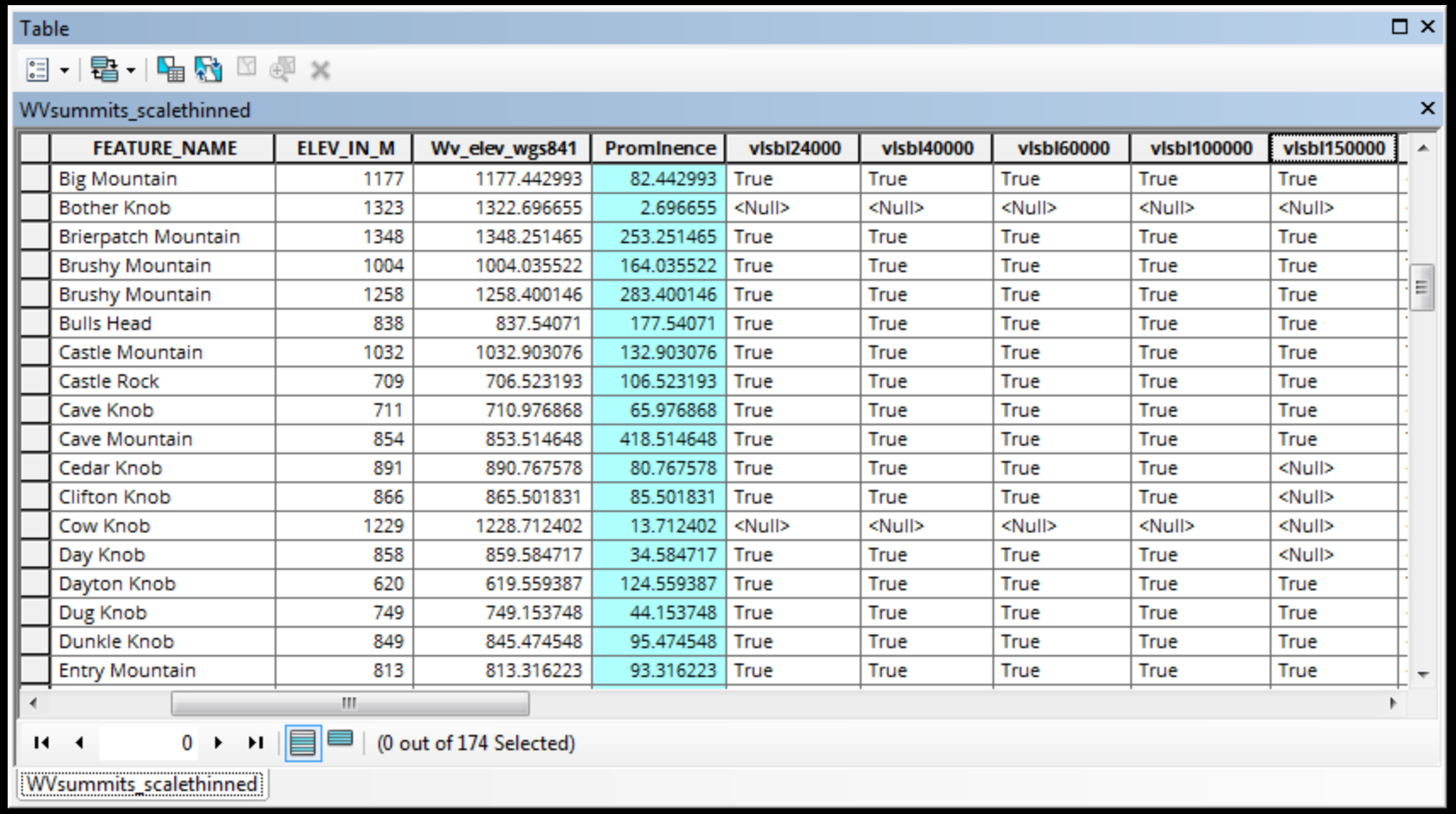
- Use DEM to generate contours, polygonize these.
- For each GNIS summit, use spatial joins to determine all contour polygons a GNIS point intersects.



```
listOfPolys = []

for elev in range(150, 1490, 15):
    print "Elevation: " + str(elev)
    aPlane = Reclassify("Wv_elev_wgs841", "Value", RemapRange([[0, elev-1, "NODATA"], [elev, 1490, elev]]))
    print "Polygonizing..."
    arcpy.RasterToPolygon_conversion(aPlane, "C:\gould\Summit_Thinning_ICCGenComm2013\WV_terrain_data.gdb\p" + str(elev), "NO_SIMPLIFY", "VALUE")
```

```
def containsHigherSummit(ptElev, contID):
    aBoolean = False
    for eachTuple in tupleList:
        if eachTuple[2] == contID and eachTuple[1] > ptElev:
            aBoolean = True
            break
    return aBoolean

# Interate over summitList
for summit in summitList:
    considerationList = []
    # populate considerationList with those tuples with the summit for this for loop:
    for entry in tupleList:
        if entry[0] == summit:
            considerationList.append(entry)
            # now we've got only the ones for a given summit in considerationList

    # if only one item, we've got our prominence:
    if len(considerationList) == 1:
        prom = considerationList[0][1] - considerationList[0][3]

    # if more than one item:
    if len(considerationList) > 1:
        prom = considerationList[0][1] - considerationList[0][3] # take first (highest) contour for prom to start.
        for item in considerationList:
            if containsHigherSummit(item[1], item[2]) == True:   # break if there's a higher summit encircled by this contour.
                break
            else:                                                # otherwise, update the prominence going down the mountain.
                prom = considerationList[considerationList.index(item)][1] - considerationList[considerationList.index(item)][3]

    # populate promDict with prominence for each summit.
    global promDictionary
    promDictionary[summit] = prom


# add field, and use update cursor to update prominence into each GNIS summit
arcpy.AddField_management("PtsJPolys3Sort", "Prominence", "DOUBLE")

rows = arcpy.UpdateCursor("PtsJPolys3Sort")
for row in rows:
    row.setValue("Prominence", promDictionary[float(row.getValue("unqPtId"))])
    rows.updateRow(row)
del row, rows
```

Summit prominence



Table

WVsummits_scalethinned

FEATURE_NAME	ELEV_IN_M	Wv_elev_wgs841	Prominence	vlsbl24000	vlsbl40000	vlsbl60000	vlsbl100000	vlsbl150000
Big Mountain	1177	1177.442993	82.442993	True	True	True	True	True
Bother Knob	1323	1322.696655	2.696655	<Null>	<Null>	<Null>	<Null>	<Null>
Brierpatch Mountain	1348	1348.251465	253.251465	True	True	True	True	True
Brushy Mountain	1004	1004.035522	164.035522	True	True	True	True	True
Brushy Mountain	1258	1258.400146	283.400146	True	True	True	True	True
Bulls Head	838	837.54071	177.54071	True	True	True	True	True
Castle Mountain	1032	1032.903076	132.903076	True	True	True	True	True
Castle Rock	709	706.523193	106.523193	True	True	True	True	True
Cave Knob	711	710.976868	65.976868	True	True	True	True	True
Cave Mountain	854	853.514648	418.514648	True	True	True	True	True
Cedar Knob	891	890.767578	80.767578	True	True	True	True	<Null>
Clifton Knob	866	865.501831	85.501831	True	True	True	True	<Null>
Cow Knob	1229	1228.712402	13.712402	<Null>	<Null>	<Null>	<Null>	<Null>
Day Knob	858	859.584717	34.584717	True	True	True	True	<Null>
Dayton Knob	620	619.559387	124.559387	True	True	True	True	True
Dug Knob	749	749.153748	44.153748	True	True	True	True	True
Dunkle Knob	849	845.474548	95.474548	True	True	True	True	True
Entry Mountain	813	813.316223	93.316223	True	True	True	True	True

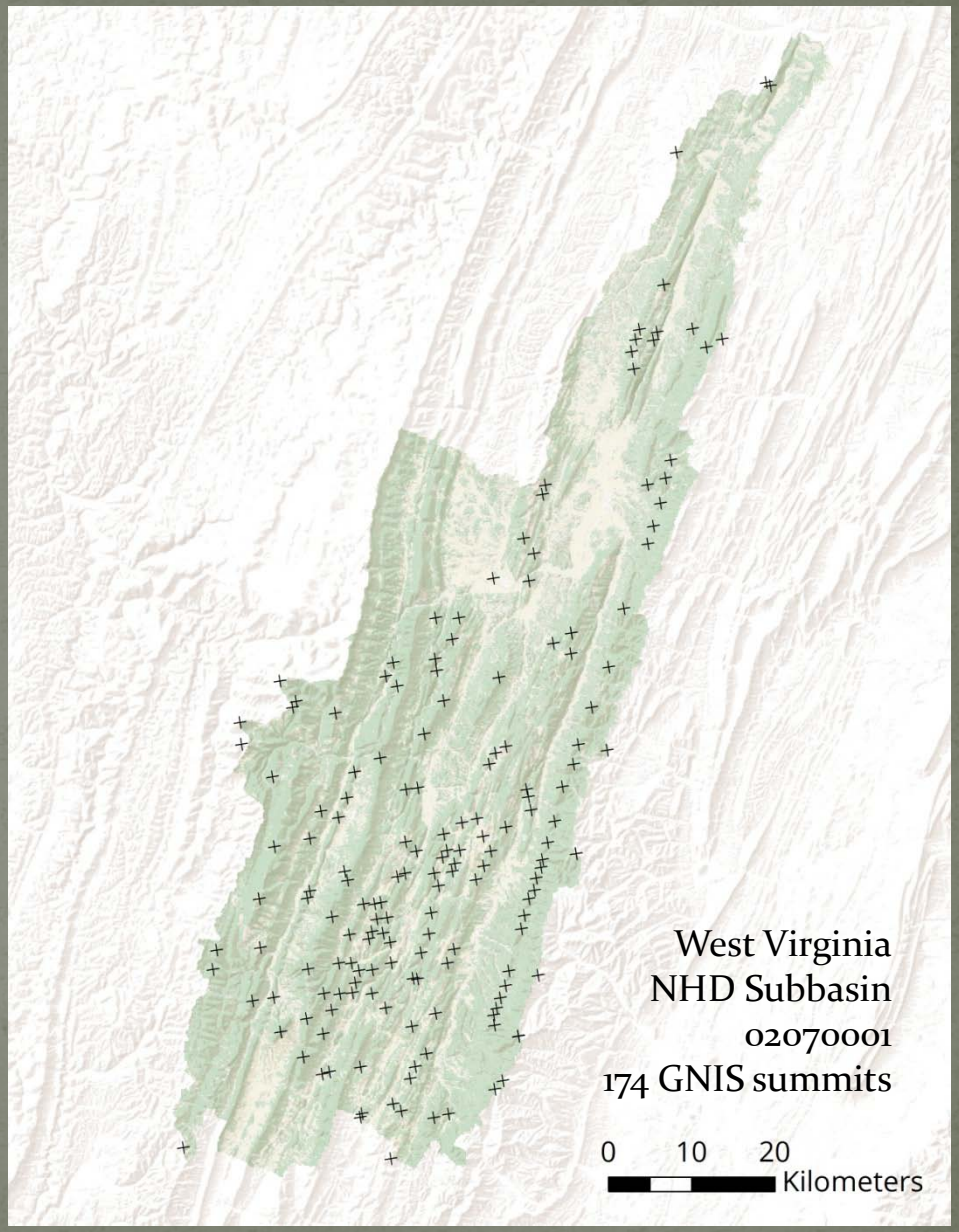
0 (0 out of 174 Selected)

WVsummits_scalethinned

Generalization

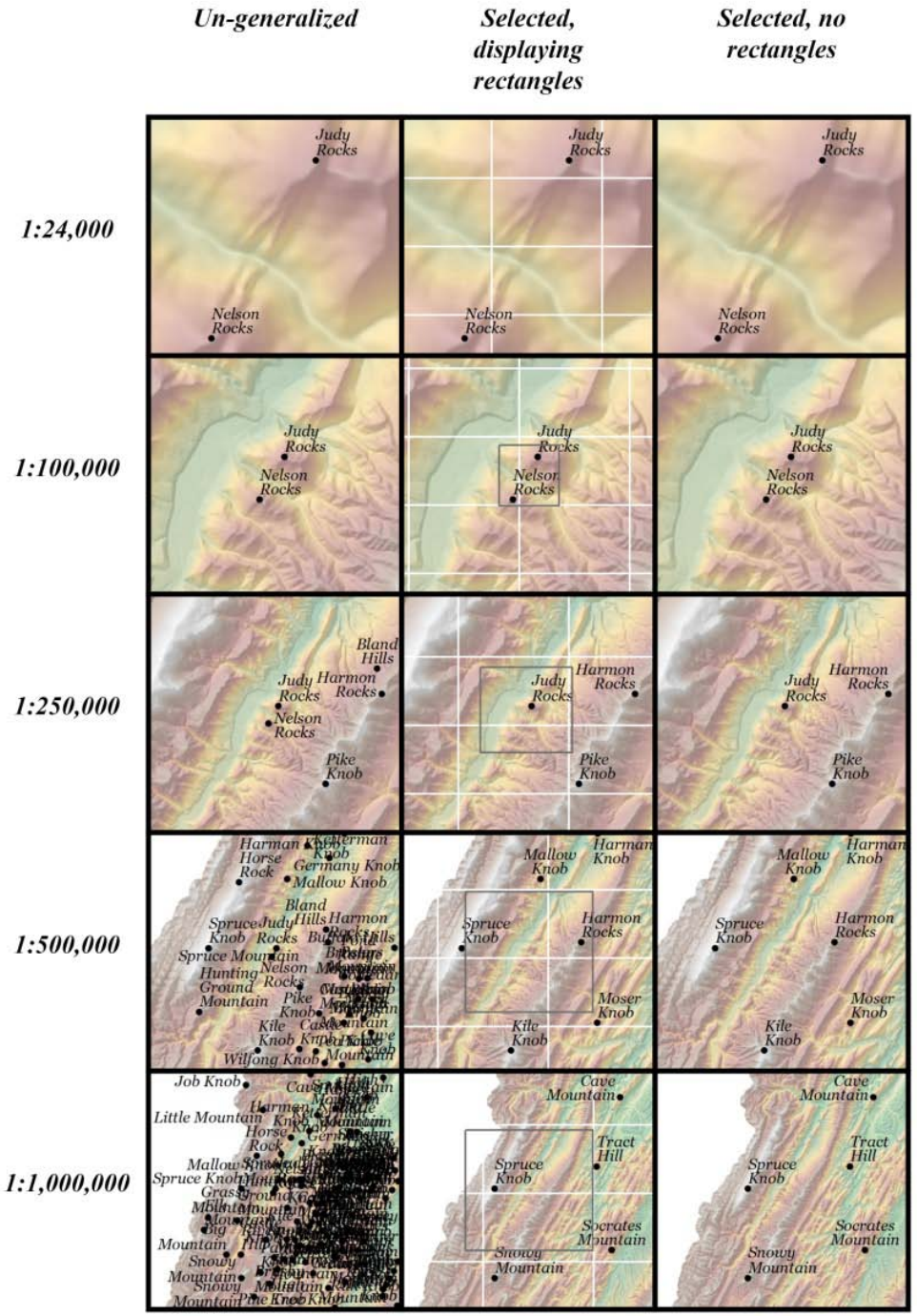
Implementation West Virginia

- 21 ladder "rungs" computed between 1:50,000 – 1:1,000,000, incrementing by 50,000
- Used standard workstation (3.00GHz, 2-core, 4GB) and ArcGIS 10.1:
  - Prominence processing ~ 5 min
  - Laddering processing ~ 20 min



Point selection through scale

- Use rectangles of size proportional to scale to tessellate map
- Choose summit of highest prominence in each rectangle
- Flag selection/elimination for each scale in a new attribute field
- Progress by ladder



```
    arcpy.AddMessage("Reading through spatial join product to isolate all point and rectangle tuples...")
    tupleList = []
    rows = arcpy.SearchCursor(ptsRect)
    for row in rows:
        # Tuples of type (unqPtID, importanceField, uRectID)
        aTuple = row.getValue("unqPtID"), row.getValue(importanceField), row.getValue("uRectID")
        tupleList.append(aTuple)
    del row, rows

    arcpy.AddMessage("Determining the most important point in each rectangle...")
    rectList = []                                   # list of unique uRectID values in tupleList.
    listOfHighest = []                              # list of the datasets's highest importance values, one in each rectangle.
    for eachTuple in tupleList:
        if eachTuple[2] not in rectList:
            rectList.append(eachTuple[2])
    for rect in rectList:
        rConsiderationList = []                     # list of all the tuples from tupleList where this rectangle is present.
        for entry in tupleList:
            if entry[2] == rect:
                rConsiderationList.append(entry)
        proms = []                                  # list of all importance values in this rectangle.
        for eTuple in rConsiderationList:
            proms.append(eTuple[1])
        highestIndex = proms.index(max(proms))
        listOfHighest.append(rConsiderationList[highestIndex][0])

    arcpy.AddMessage("Adding visibility field for this scale...")
    visFieldName = "visbl" + mapScale.replace(" meters", "")
    arcpy.AddField_management(outputPts, visFieldName, "TEXT")

    arcpy.AddMessage("Flaging visible points for this scale...")
    rows = arcpy.UpdateCursor(outputPts)
    for row in rows:
        if row.getValue("unqPtID") in listOfHighest:
            row.setValue(visFieldName, "True")
        rows.updateRow(row)
    del row, rows
```

```
    arcpy.AddMessage("Reading through spatial join product to isolate all point and rectangle tuples...")
    tupleList = []
    rows = arcpy.SearchCursor(ptsRect)
    for row in rows:
        # Tuples of type (unqPtID, importanceField, uRectID)
        aTuple = row.getValue("unqPtID"), row.getValue(importanceField), row.getValue("uRectID")
        tupleList.append(aTuple)
    del row, rows

    arcpy.AddMessage("Determining the most important point in each rectangle...")
    rectList = []                                   # list of unique uRectID values in tupleList.
    listOfHighest = []                              # list of the datasets's highest importance values, one in each rectangle.
    for eachTuple in tupleList:
        if eachTuple[2] not in rectList:
            rectList.append(eachTuple[2])
    for rect in rectList:
        rConsiderationList = []                     # list of all the tuples from tupleList where this rectangle is present.
        for entry in tupleList:
            if entry[2] == rect:
                rConsiderationList.append(entry)
        proms = []                                  # list of all importance values in this rectangle.
        for eTuple in rConsiderationList:
            proms.append(eTuple[1])
        highestIndex = proms.index(max(proms))
        listOfHighest.append(rConsiderationList[highestIndex][0])

    arcpy.AddMessage("Adding visibility field for this scale...")
    visFieldName = "visbl" + mapScale.replace(" meters", "")
    arcpy.AddField_management(outputPts, visFieldName, "TEXT")

    arcpy.AddMessage("Flaging visible points for this scale...")
    rows = arcpy.UpdateCursor(outputPts)
    for row in rows:
        if row.getValue("unqPtID") in listOfHighest:
            row.setValue(visFieldName, "True")
        rows.updateRow(row)
    del row, rows
```

```
    arcpy.AddMessage("Reading through spatial join product to isolate all point and rectangle tuples...")
    tupleList = []
    rows = arcpy.SearchCursor(ptsRect)
    for row in rows:
        # Tuples of type (unqPtID, importanceField, uRectID)
        aTuple = row.getValue("unqPtID"), row.getValue(importanceField), row.getValue("uRectID")
        tupleList.append(aTuple)
    del row, rows

    arcpy.AddMessage("Determining the most important point in each rectangle...")
    rectList = []                                   # list of unique uRectID values in tupleList.
    listOfHighest = []                              # list of the datasets's highest importance values, one in each rectangle.
    for eachTuple in tupleList:
        if eachTuple[2] not in rectList:
            rectList.append(eachTuple[2])
    for rect in rectList:
        rConsiderationList = []                     # list of all the tuples from tupleList where this rectangle is present.
        for entry in tupleList:
            if entry[2] == rect:
                rConsiderationList.append(entry)
        proms = []                                  # list of all importance values in this rectangle.
        for eTuple in rConsiderationList:
            proms.append(eTuple[1])
        highestIndex = proms.index(max(proms))
        listOfHighest.append(rConsiderationList[highestIndex][0])

    arcpy.AddMessage("Adding visibility field for this scale...")
    visFieldName = "visbl" + mapScale.replace(" meters", "")
    arcpy.AddField_management(outputPts, visFieldName, "TEXT")

    arcpy.AddMessage("Flaging visible points for this scale...")
    rows = arcpy.UpdateCursor(outputPts)
    for row in rows:
        if row.getValue("unqPtID") in listOfHighest:
            row.setValue(visFieldName, "True")
        rows.updateRow(row)
    del row, rows
```

Summit selection

Table

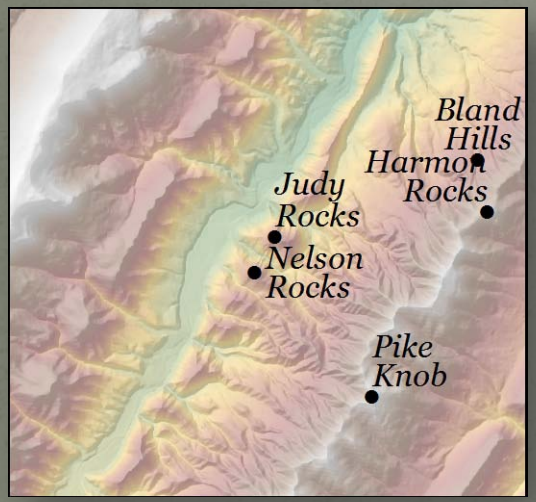
WVsummits_scalethinned

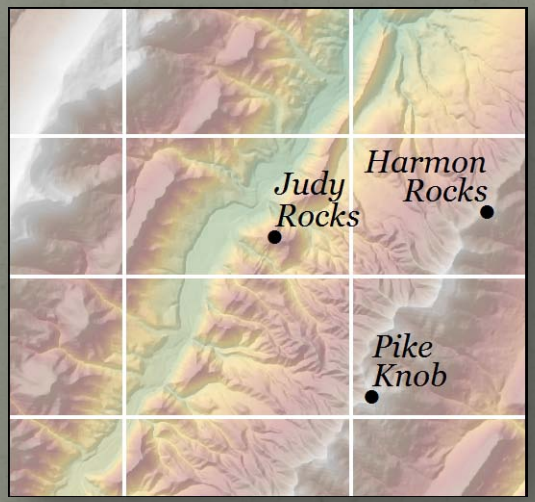
FEATURE_NAME	ELEV_IN_M	Wv_elev_wgs841	Prominence	vlsbl24000	vlsbl40000	vlsbl60000	vlsbl100000	vlsbl150000	vlsbl250000	vlsbl400000
Big Mountain	1177	1177.442993	82.442993	True	True	True	True	True	<Null>	<Null>
Bother Knob	1323	1322.696655	2.696655	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>
Brierpatch Mountain	1348	1348.251465	253.251465	True	True	True	True	True	True	True
Brushy Mountain	1004	1004.035522	164.035522	True	True	True	True	True	True	True
Brushy Mountain	1258	1258.400146	283.400146	True	True	True	True	True	True	True
Bulls Head	838	837.54071	177.54071	True	True	True	True	True	True	True
Castle Mountain	1032	1032.903076	132.903076	True	True	True	True	True	True	<Null>
Castle Rock	709	706.523193	106.523193	True	True	True	True	True	True	<Null>
Cave Knob	711	710.976868	65.976868	True	True	True	True	True	<Null>	<Null>
Cave Mountain	854	853.514648	418.514648	True	True	True	True	True	True	True
Cedar Knob	891	890.767578	80.767578	True	True	True	True	<Null>	<Null>	<Null>
Clifton Knob	866	865.501831	85.501831	True	True	True	True	<Null>	<Null>	<Null>
Cow Knob	1229	1228.712402	13.712402	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>
Day Knob	858	859.584717	34.584717	True	True	True	True	<Null>	<Null>	<Null>
Dayton Knob	620	619.559387	124.559387	True	True	True	True	True	True	<Null>
Dug Knob	749	749.153748	44.153748	True	True	True	True	True	<Null>	<Null>
Dunkle Knob	849	845.474548	95.474548	True	True	True	True	True	<Null>	<Null>
Entry Mountain	813	813.316223	93.316223	True	True	True	True	True	<Null>	<Null>
Evick Knob	813	812.719788	77.719788	True	True	True	True	True	<Null>	<Null>
Fisher Knob	878	877.67627	127.67627	True	True	True	True	True	True	<Null>
Fisher Mountain	918	915.908264	135.908264	True	True	True	True	True	True	True
Flint Knob	959	958.894348	148.894348	True	True	True	True	True	True	<Null>
Foremost Mountain	905	903.491028	153.491028	True	True	True	True	True	True	True
Frye Knob	960	958.61969	73.61969	True	True	True	True	True	True	<Null>
Germany Knob	838	833.417725	173.417725	True	True	True	True	True	True	<Null>

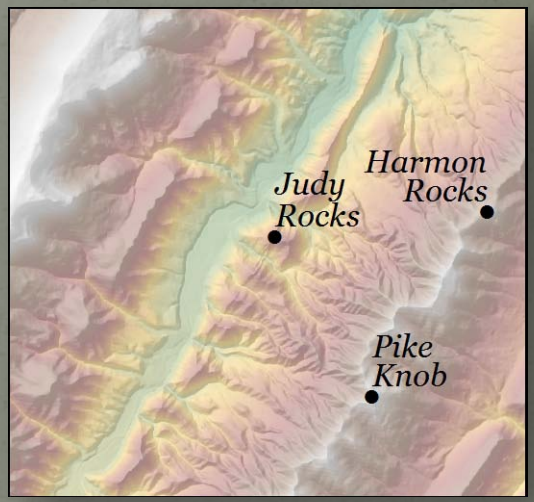
0 (0 out of 174 Selected)

WVsummits_scalethinned

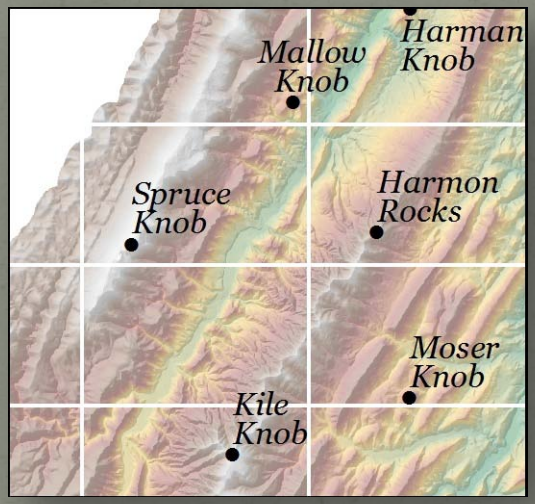
Point selection through scale

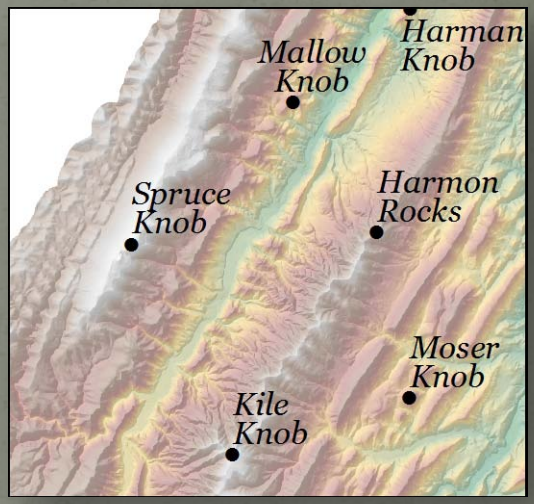






Point selection through scale





Acknowledgements

ICA Commission on Gen. & Multi Rep.
CEGIS, USGS, Penn State

The coffee machine

Thanks!

Descriptive stats