

---

# Geomorphological units (fishcamp)

---

## 9.1 Introduction

The purpose of this exercise is to: (1) generate and filter a DEM from point data, and use it to derive various DEM parameters; (2) extract landform classes using an objective procedure (fuzzy  $k$ -means algorithm); and (3) improve the accuracy of soil mapping units using an existing map. We will also use geostatistical tools to assess variogram parameters for various landform classes to see if there are significant differences between them. For geographical analysis and visualization, we will exclusively use SAGA GIS (Brenning, 2008); an equally good alternative to run a similar analysis is GRASS GIS<sup>1</sup> (Neteler and Mitasova, 2008).

We will use three standard elevation data sets common for contemporary geomorphometry applications: point-sampled elevations (LiDAR), contours lines digitized from a topo map, and a raster of elevations sampled using a remote sensing system. All three elevation sources (`lidar.shp`, `contours.shp` and `DEMSRTM1.asc`) refer to the same geographical area — a  $1 \times 2$  km case study `fishcamp` located in the eastern part of California (Fig. 9.4). This area is largely covered with forests; the elevations range from 1400 to 1800 meters. The data set was obtained from the USGS National Map seamless server<sup>2</sup>. The map of soil mapping units was obtained from the Natural Resources Conservation Service (NRCS) Soil Data Mart<sup>3</sup>. There are six soil mapping units: (1) Holland family, 35 to 65% slopes; (2) Chaix-chawanakee family-rock outcrop complex; (3) Chaix family, deep, 5 to 25% slopes; (4) Chaix family, deep, 15 to 45% slopes, (5) Holland family, 5 to 65% slopes, valleys; (6) Chaix-chawanakee families-rock outcrop complex, hilltops. The complete data set shown in this chapter is available via the geomorphometry.org website<sup>4</sup>; the scripts used to predict soil mapping units and extract landforms are available via the book's homepage.

There are basically two inputs to a supervised extraction of landforms (shown in Fig. 9.5): (1) raw elevation measurements (either points or un-filtered rasters); (2) existing polygon map i.e. the expert knowledge. The raw elevations are used to generate the initial DEM, which typically needs to be filtered for artifacts. An expert then also needs to define a set of suitable **Land Surface Parameters** (LSPs) that can be used to parameterize the features of interest. In practice, this is not trivial. On one hand, classes from the geomorphological or soil map legend are often determined by their morphology; hence we can easily derive DEM parameters that describe shape (curvatures, wetness index), hydrologic context (distance from the streams, height above the drainage network) or climatic conditions (incoming solar radiation). On the other hand, many classes are defined by land surface and sub-surface (geological) parameters that are difficult to obtain and often not at our disposal. Hence, the results of mapping soil and landform units will often be of limited success, if based only on the DEM and its derivatives. Please keep that in mind when running similar types of analysis with your own data.

This chapter is largely based on the most recent book chapter for the Geomorphological Mapping handbook by Seijmonsbergen et al. (2010). An introduction to some theoretical considerations connected with the

---

<sup>1</sup><http://grass.itc.it>

<sup>2</sup><http://seamless.usgs.gov>

<sup>3</sup><http://soildatamart.nrcs.usda.gov>

<sup>4</sup><http://geomorphometry.org/content/fishcamp>

1 geostatistical modeling of land surface topography is given in section 2.7.

## 2 9.2 Data download and exploration

3 Open the R script (fishcamp.R) and start preparing the data. First, download the fishcamp.zip complete  
4 data set and layers of interest:

```
> download.file("http://geomorphometry.org/system/files/fishcamp.zip",
+ destfile=paste(getwd(), "fishcamp.zip", sep="/"))
# LiDAR points:
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("lidar", j, sep=""),
+ zipname="fishcamp.zip")
>   file.copy(fname, paste("./lidar", j, sep=""), overwrite=TRUE)
> }
# contour lines:
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("contours", j, sep=""),
+ zipname="fishcamp.zip")
>   file.copy(fname, paste("./contours", j, sep=""), overwrite=TRUE)
> }
# streams:
> for(j in list(".shp", ".shx", ".dbf")){
>   fname <- zip.file.extract(file=paste("streams", j, sep=""),
+ zipname="fishcamp.zip")
>   file.copy(fname, paste("./streams", j, sep=""), overwrite=TRUE)
> }
# SRTM DEM:
> fname <- zip.file.extract(file="DEMSRTM1.asc", zipname="fishcamp.zip")
> file.copy(fname, "./DEMSRTM1.asc", overwrite=TRUE)
# soil map:
> fname <- zip.file.extract(file="soilmu.asc", zipname="fishcamp.zip")
> file.copy(fname, "./soilmu.asc", overwrite=TRUE)
```

5 where lidar.shp is a point map (LiDAR ground reflections), contours.shp is a map of contours (lines)  
6 digitized from a topo map, and DEMSRTM1.asc is the 1 arcsec (25 m) Shuttle Radar Topography Mission  
7 (SRTM) DEM.

8 To load LiDAR point data set, SRTM DEM, and the soil map to R, we use functionality of the rgdal package:

```
> lidar <- readOGR("lidar.shp", "lidar")

OGR data source with driver: ESRI Shapefile
Source: "lidar.shp", layer: "lidar"
with 273028 rows and 1 columns
Feature type: wkbPoint with 2 dimensions

> grids25m <- readGDAL("DEMSRTM1.asc")

DEMSRTM1.asc has GDAL driver AAIGrid
and has 40 rows and 80 columns

> grids5m <- readGDAL("soilmu.asc")

soilmu.asc has GDAL driver AAIGrid
and has 200 rows and 400 columns
```

9 this shows that lidar.shp contains 273,028 densely sampled points. The original LiDAR data set consist of,  
10 in fact, over 5 million of points; these points were subsampled for the purpose of this exercise, i.e. to speed  
11 up the processing. Next, we can estimate the approximate grid cell size<sup>5</sup> based on the density of points in the  
12 area of interest:

<sup>5</sup>We can take a rule of thumb that there should be at least 2 points per grid cell.

```
> pixelsize <- round(2*sqrt(areaSpatialGrid(grids25m)/length(lidar$Z)),0)
> pixelsize
[1] 5
```

and then also attach the correct coordinate system to each spatial object: 1

```
> proj4string(lidar) <- CRS("+init=epsg:26911")
> proj4string(lidar)
[1] "+init=epsg:26911 +proj=utm +zone=11 +ellps=GRS80
+ datum=NAD83 +units=m +no_defs +towgs84=0,0,0"
```

which is the UTM coordinate system with North American Datum 83 and GRS80 ellipsoid<sup>6</sup>. This allows us to obtain the geographical coordinates of the study area: 2  
3

```
> proj4string(grids25m) <- CRS("+init=epsg:26911")
# coordinates of the center:
> grids25m.ll <- spTransform(grids25m, CRS("+proj=longlat +ellps=WGS84"))
> grids25m.ll@bbox
```

```
      min      max
x -119.6232 -119.60060
y  37.4589  37.46817
```

```
> clon <- mean(grids25m.ll@bbox[1,])
> clat <- mean(grids25m.ll@bbox[2,])
```

## 9.3 DEM generation 4

### 9.3.1 Variogram modeling 5

The first data analysis step is generation of a DEM from the LiDAR point data. Here geostatistics can provide a lot of information. First, we can determine how smoothly elevation varies in space, are the measurements noisy, is the feature of interest anisotropic? This type of analysis can be run by using e.g. the `gstat` package. However, fitting a variogram model with such a large point data set will take a long time on standard desktop PC. Instead, an equally reliable variogram can be produced by taking a random sub-sample of the measurements: 6  
7  
8  
9  
10  
11

```
> lidar.sample <- lidar[runif(length(lidar$Z))<0.05,]
> varmap.plt <- plot(variogram(Z ~ 1, lidar.sample, map=TRUE, cutoff=50*pixelsize,
+ width=pixelsize), col.regions=grey(rev(seq(0,1,0.025))))
> Z.svar <- variogram(Z ~ 1, lidar.sample, alpha=c(45,135)) # cutoff=50*dem.pixelsize
> Z.vgm <- fit.variogram(Z.svar, vgm(psill=var(lidar.sample$Z), "Gau",
+ sqrt(areaSpatialGrid(grids25m))/4, nugget=0, anis=c(p=135, s=0.6)))
> vgm.plt <- plot(Z.svar, Z.vgm, plot.nu=F, cex=2, pch="+", col="black")
# plot the two variograms next to each other:
> print(varmap.plt, split=c(1,1,2,1), more=T)
> print(vgm.plt, split=c(2,1,2,1), more=F)
```

which results in Fig. 9.1. This variogram map provides a clear picture of how semivariances change in every compass direction. This allows one to more easily find the appropriate principal axis for defining the anisotropic variogram model. In this case, elevations are more spatially ‘continuous’ in the direction NW–SE. 12  
13  
14

```
> Z.vgm
      model      psill      range ang1 anis1
1  Nug      64.34422    0.0000   0    1.0
2  Gau 11309.80343  963.0828  135   0.6
```

The resulting variogram shows that the feature of interest varies smoothly in the area of interest (Fig. 9.1), which is typical for elevation data. Nugget variation is insignificant, but  $\neq 0$ . This information can help us determine the amount of filtering needed to reduce man-made objects and artifacts in the LiDAR DEM. 15  
16  
17

<sup>6</sup><http://spatialreference.org/ref/epsg/26911/>

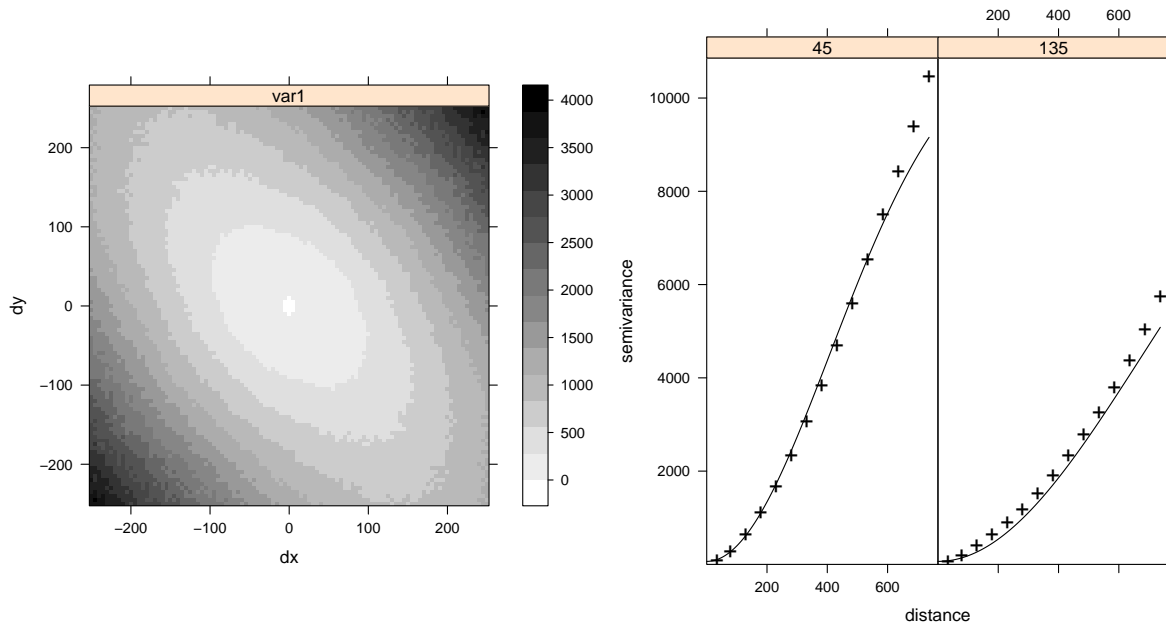


Fig. 9.1: Variogram map (left) and fitted anisotropic variogram model (right) for LiDAR-based elevations.

1

### 9.3.2 DEM filtering

2 Although the original LiDAR product is suppose to contain only ground reflection measurements, we can easily  
 3 notice that there are still many artificial spikes and isolated pixels, with much higher elevation values than the  
 4 neighboring pixels. For example, we can quickly generate a DEM in SAGA by converting the point map to a  
 5 raster map:

```
> rsaga.geoprocessor(lib="grid_gridding", module=0,
+   param=list(GRID="DEM5LIDAR.sgrd", INPUT="lidar.shp", FIELD=0, LINE_TYPE=0,
+   USER_CELL_SIZE=pixelsize, USER_X_EXTENT_MIN=grids5m@bbox[1,1]+pixelsize/2,
+   USER_X_EXTENT_MAX=grids5m@bbox[1,2]-pixelsize/2,
+   USER_Y_EXTENT_MIN=grids5m@bbox[2,1]+pixelsize/2,
+   USER_Y_EXTENT_MAX=grids5m@bbox[2,2]-pixelsize/2))
```

6 which will contain many missing pixels (Fig. 9.2a). In addition, this DEM will show many small pixels with  
 7 10–20 m higher elevations from the neighbors. Spikes, roads and similar artifacts are not really connected  
 8 with the geomorphology and need to be filtered before we can use the DEM for geomorphological mapping.  
 9 Spikes<sup>7</sup> can be detected using, for example, difference from the mean value, given a search radius (see ‘*residual*  
 10 *analysis*’ in SAGA):

```
> rsaga.geoprocessor(lib="geostatistics_grid", 0,
+   param=list(INPUT="DEM5LIDAR.sgrd", MEAN="tmp.sgrd", STDDEV="tmp.sgrd",
+   RANGE="tmp.sgrd", DEVMEAN="tmp.sgrd", PERCENTILE="tmp.sgrd", RADIUS=5,
+   DIFF="dif_lidar.sgrd"))
# read back into R and mask out all areas:
> rsaga.sgrd.to.esri(in.sgrd=c("dif_lidar.sgrd", "DEM5LIDAR.sgrd"),
+   out.grids=c("dif_lidar.asc", "DEM5LIDAR.asc"), out.path=getwd(), prec=1)
> grids5m$DEM5LIDAR <- readGDAL("DEM5LIDAR.asc")$band1
> grids5m$dif <- readGDAL("dif_lidar.asc")$band1
> lim.dif <- quantile(grids5m$dif, c(0.025,0.975), na.rm=TRUE)
> lim.dif
```

<sup>7</sup>These individual pixels are most probably dense patches of forest, which are very difficult for LiDAR to penetrate.

```
2.5% 97.5%
-3.9 3.4
```

```
> grids5m$DEM5LIDARf <- ifelse(grids5m$dif<=lim.dif[[1]]|grids5m$dif>=lim.dif[[2]],
+   NA, grids5m$DEM5LIDAR)
> summary(grids5m$DEM5LIDARf)[7]/length(grids5m@data[[1]])
# 15% pixels have been masked out
```

which will remove about 15% of ‘suspicious’ pixels. The remaining missing pixels can be filtered/re-interpolated<sup>8</sup> from the neighboring pixels (see ‘close gaps’ method in SAGA; the resulting map is shown in Fig. 9.2b):

```
rsaga.geoprocessor(lib="grid_tools", module=7, param=list(INPUT="DEM5LIDARf.sgrd",
+   RESULT="DEM5LIDARf.sgrd")) # we write to the same file!
```

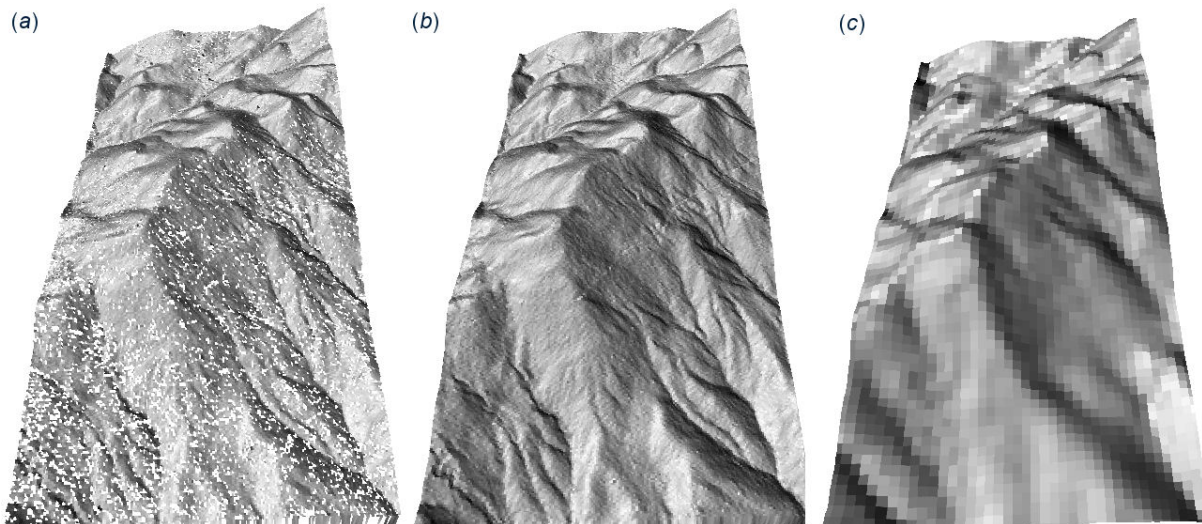


Fig. 9.2: Initial 5 m DEM (a) generated directly from the LiDAR points, and after filtering (b). In comparison with the 25 m DEM (c) derived from the contour lines. Seen from the western side.

### 9.3.3 DEM generation from contour data

We can also generate DEM surfaces from digitized contour lines (contours.shp) using a spline interpolation, which is often recommended as the most suited DEM gridding technique for contour data (Conrad, 2007; Neteler and Mitasova, 2008). In SAGA:

```
> rsaga.geoprocessor(lib="grid_spline", module=1, param=list(GRID="DEM25TPS.sgrd",
+   SHAPES="contours.shp", TARGET=0, SELECT=1, MAXPOINTS=10,
+   USER_CELL_SIZE=25, USER_FIT_EXTENT=T))
```

This looks for the nearest 10 points in a local search radius and fits the Thin Plate Spline<sup>9</sup> over a 25 grid. This initial DEM can be hydrologically adjusted using the deepen drainage route:

```
> rsaga.geoprocessor(lib="ta_preprocessor", module=1,
+   param=list(DEM="DEM25TPS.sgrd", DEM_PREPROC="DEM25TPSf.sgrd", METHOD=0))
```

The resulting DEM surface can be seen in Fig. 9.2(c). We can compare the LiDAR-based and topo-map based DEMs and estimate the accuracy<sup>10</sup> of the DEM derived from the contour lines. First, we need to aggregate the 5 m resolution DEM5LIDAR to 25 m resolution:

<sup>8</sup>This can then be considered a void filling type of DEM filtering (Hengl and Reuter, 2008, p.104–106).

<sup>9</sup>SAGA implements the algorithm of Donato and Belongie (2003).

<sup>10</sup>Assuming that the LiDAR DEM is the ‘true’ DEM.

```
# create empty grid:
> rsaga.geoprocessor(lib="grid_tools", module=23,
+   param=list(GRID="DEM25LIDAR.sgrd", M_EXTENT=0,
+   XMIN=grids5m@bbox[1,1]+pixelsize/2, YMIN=grids5m@bbox[2,1]+pixelsize/2,
+   NX=grids25m@grid@cells.dim[1], NY=grids25m@grid@cells.dim[2], CELLSIZE=25))
# resample to 25 m:
> rsaga.geoprocessor(lib="grid_tools", module=0,
+   param=list(INPUT="DEM5LIDARf.sgrd", GRID="DEM25LIDAR.sgrd",
+   GRID_GRID="DEM25LIDAR.sgrd", METHOD=2, KEEP_TYPE=FALSE, SCALE_UP_METHOD=5))
```

1 The difference between the two DEMs is then:

```
> rsaga.sgrd.to.esri(in.sgrd=c("DEM25LIDAR.sgrd", "DEM25TPS.sgrd"),
+   out.grids=c("DEM25LIDAR.asc", "DEM25TPS.asc"), out.path=getwd(), prec=1)
> grids25m$DEM25LIDAR <- readGDAL("DEM25LIDAR.asc")$band1
> grids25m$DEM25TPS <- readGDAL("DEM25TPS.asc")$band1
> sqrt(sum((grids25m$DEM25LIDAR-grids25m$DEM25TPS)^2)/length(grids25m$DEM25LIDAR))

[1] 5.398652
```

2 which means that the average error of the DEM derived using contour lines (topo map) is about 5 m, which is  
3 well within the accuracy standards for this scale.

#### 4 9.4 Extraction of Land Surface Parameters

5 We proceed with the extraction of LSPs that will be used to explain the distribution of soil mapping units.  
6 SAGA can derive over 100 LSPs given an input DEM. There is no need of course to use all of them; instead  
7 we need to try to list the LSPs that are relevant to the mapping objectives, study area characteristics and  
8 scale of application. For example, because the area is of high relief we can derive some representative DEM  
9 parameters that can explain hydrological, climatic and morphological properties of a terrain. For example: (1)  
10 SAGA Topographic Wetness Index (TWI), (2) Valley depth (VDEPTH), (3) Solar Insolation (INSOLAT), and (4)  
11 Convergence index (CONVI):

```
# Topographic Wetness Index:
> rsaga.geoprocessor(lib="ta_hydrology", module=15,
+   param=list(DEM="DEM5LIDARf.sgrd", C="catharea.sgrd", GN="catchslope.sgrd",
+   CS="modcatharea.sgrd", SB="TWI.sgrd", T=10))
# valley depth:
> rsaga.geoprocessor(lib="ta_morphometry", module=14,
+   param=list(DEM="DEM5LIDARf.sgrd", HO="tmp.sgrd", HU="VDEPTH.sgrd",
+   NH="tmp.sgrd", SH="tmp.sgrd", MS="tmp.sgrd", W=12, T=120, E=4))
# incoming solar radiation:
> rsaga.geoprocessor(lib="ta_lighting", module=2,
+   param=list(ELEVATION="DEM5LIDARf.sgrd", INSOLAT="INSOLAT.sgrd",
+   DURATION="durat.sgrd", LATITUDE=clat, HOUR_STEP=2, TIMESPAN=2, DAY_STEP=5))
# convergence index:
> rsaga.geoprocessor(lib="ta_morphometry", module=2,
+   param=list(ELEVATION="DEM5LIDARf.sgrd", RESULT="CONVI.sgrd", RADIUS=3,
+   METHOD=0, SLOPE=TRUE))
```

12 Note that, because the features of interest (soil mapping units) are geographically continuous and smooth,  
13 we should use a wide search radius to derive the LSPs. In this case we use arbitrary parameters to derive  
14 specific LSPs — these are not as easy to determine objectively<sup>11</sup>.

15 Now that we have prepared a list of DEM parameters that can be used to describe geomorphology of the  
16 terrain, we can proceed with the extraction of land-surface objects. Before we can proceed, we need to read  
17 the maps into R:

<sup>11</sup>Note also that the resulting LSPs can also differ largely for different combination of parameters

```
> LSP.list <- c("TWI.asc", "VDEPTH.asc", "INSOLAT.asc", "CONVI.asc")
> rsaga.sgrd.to.esri(in.sgrds=set.file.extension(LSP.list, ".sgrd"),
+   out.grids=LSP.list, prec=1, out.path=getwd())
> for(i in 1:length(LSP.list)){
>   grids5m@data[strsplit(LSP.list[i], ".asc")[[1]]] <- readGDAL(LSP.list[i])$band1
> }
```

```
TWI.asc has GDAL driver AAIGrid
and has 200 rows and 400 columns
...
CONVI.asc has GDAL driver AAIGrid
and has 200 rows and 400 columns
```

## 9.5 Unsupervised extraction of landforms

### 9.5.1 Fuzzy $k$ -means clustering

Geomorphological classes can be optimally extracted using, for example, the **fuzzy  $k$ -means clustering** approach as implemented in that stats package (Venables and Ripley, 2002). This will optimally assign each individual pixel to an abstract class; the class centers will be selected in such way that the within groups sum of squares is minimized. In statistical terms, this is the cluster analysis approach to extraction of features.

We can start by converting the LSPs to independent components by using Principal Component analysis (Fig. 9.3):

```
> pc.dem <- prcomp( ~ DEM5LIDARf+TWI+VDEPTH+
+   INSOLAT+CONVI, scale=TRUE, grids5m@data)
> biplot(pc.dem, arrow.len=0.1,
+   xlabs=rep(".", length(pc.dem$x[,1])),
+   main="PCA biplot")
```

which shows that the LSPs are relatively independent. To be statistically correct, we will proceed with clustering the Principal Components instead of using the original predictors. Next we can try to obtain the optimal number of classes for fuzzy  $k$ -means clustering by using (Venables and Ripley, 2002)<sup>12</sup>:

```
> demdata <- as.data.frame(pc.dem$x)
> wss <- (nrow(demdata)-1)*sum(apply(demdata,2,var))
> for (i in 2:20) {wss[i] <- sum(kmeans(demdata, centers=i)$withinss)}
```

Warning messages:

```
1: did not converge in 10 iterations
```

which unfortunately did not converge<sup>13</sup>. For practical reasons, we will assume that 12 classes are sufficient:

```
> kmeans.dem <- kmeans(demdata, 12)
> grids5m$kmeans.dem <- kmeans.dem$cluster
> grids5m$landform <- as.factor(kmeans.dem$cluster)
> summary(grids5m$landform)
```

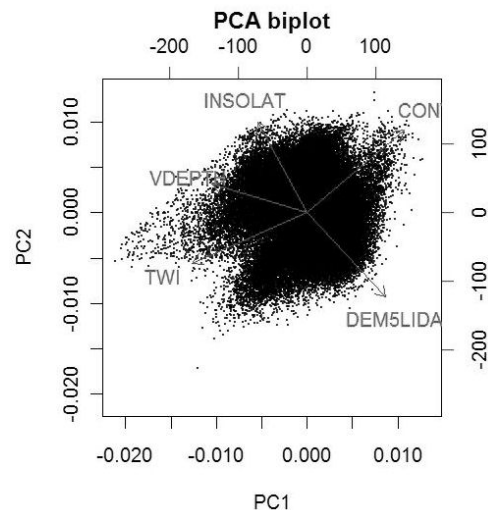


Fig. 9.3: The PCA biplot showing first two components derived using five LSPs.

<sup>12</sup>An alternative to  $k$ -means clustering is to use the Partitioning Around Medoids (PAM) method, which is generally more robust to 'messy' data, and will always return the same clusters.

<sup>13</sup>Which also means that increasing the number of classes above 20 will still result in smaller within groups sum of squares.

1	2	3	4	5	6
2996	3718	6785	7014	4578	7895
7	8	9	10	11	12
7367	13232	2795	6032	7924	9664



Fig. 9.4: Results of unsupervised classification (12 classes) visualized in Google Earth.

1 The map of predicted classes can be seen in Fig. 9.4. The size of polygons is well distributed and the  
 2 polygons are spatially continuous. The remaining issue is what do these classes really mean? Are these  
 3 really geomorphological units and could different classes be combined? Note also that there is a number  
 4 of object segmentation algorithms that could be combined with extraction of (homogenous) landform units  
 5 (Seijmonsbergen et al., 2010).

### 6 9.5.2 Fitting variograms for different landform classes

7 Now that we have extracted landform classes, we can see if there are differences between the variograms for  
 8 different landform units (Lloyd and Atkinson, 1998). To be efficient, we can automate variogram fitting by  
 9 running a loop. The best way to achieve this is to make an empty data frame and then fill it in with the results  
 10 of fitting:

```
> lidar.sample.ov <- overlay(grids5m["landform"], lidar.sample)
> lidar.sample.ov$Z <- lidar.sample$Z
# number of classes:
> landform.no <- length(levels(lidar.sample.ov$landform))
# empty dataframes:
> landform.vgm <- as.list(rep(NA, landform.no))
> landform.par <- data.frame(landform=as.factor(levels(lidar.sample.ov$landform)),
+   Nug=rep(NA, landform.no), Sill=rep(NA, landform.no),
+   range=rep(NA, landform.no))
# fit the variograms:
> for(i in 1:length(levels(lidar.sample.ov$landform))) {
>   tmp <- subset(lidar.sample.ov,
+     lidar.sample.ov$landform==levels(lidar.sample.ov$landform)[i])
>   landform.vgm[[i]] <- fit.variogram(variogram(Z ~ 1, tmp, cutoff=50*pixelsize),
+     vgm(psill=var(tmp$Z), "Gau", sqrt(areaSpatialGrid(grids25m))/4, nugget=0))
>   landform.par$Nug[i] <- round(landform.vgm[[i]]$psill[1], 1)
>   landform.par$Sill[i] <- round(landform.vgm[[i]]$psill[2], 1)
>   landform.par$range[i] <- round(landform.vgm[[i]]$range[2], 1)
> }
```



and we can print the results of variogram fitting in a table:

```
> landform.par
      landform Nug   Sill range
1         1  4.0 5932.4 412.6
2         2  0.5  871.1 195.1
3         3  0.1 6713.6 708.2
4         4  8.6  678.7 102.0
5         5  0.0 6867.0 556.8
6         6  7.6 2913.6 245.5
7         7  3.3 1874.2 270.9
8         8  5.8  945.6 174.1
9         9  6.4  845.6 156.0
10        10  5.0 3175.5 281.2
11        11  5.1  961.6 149.5
12        12  7.4 1149.9 194.4
```

which shows that there are distinct differences in variograms between different landform classes (see also Fig. 9.4). This can be interpreted as follows: the variograms differ mainly because there are differences in the surface roughness between various terrains, which is also due to different tree coverage.

Consider also that there are possibly still many artificial spikes/trees that have not been filtered. Also, many landforms are *'patchy'* i.e. represented by isolated pixels, which might lead to large differences in the way the variograms are fitted. It would be interesting to try to fit local variograms<sup>14</sup>, i.e. variograms for each grid cell and then see if there are real discrete jumps in the variogram parameters.

## 9.6 Spatial prediction of soil mapping units

### 9.6.1 Multinomial logistic regression

Next, we will use the extracted LSPs to try to improve the spatial detail of an existing *traditional*<sup>15</sup> soil map. A suitable technique for this type of analysis is the multinomial logistic regression algorithm, as implemented in the `multinom` method of the `nnet` package (Venables and Ripley, 2002, p.203). This method iteratively fits logistic models for a number of classes given a set of training pixels. The output predictions can then be evaluated versus the complete geomorphological map to see how well the two maps match and where the most problematic areas are. We will follow the iterative computational framework shown in Fig. 9.5. In principle, the best results can be obtained if the selection of LSPs and parameters used to derive LSPs are iteratively adjusted until maximum mapping accuracy is achieved.

### 9.6.2 Selection of training pixels

Because the objective here is to refine the existing soil map, we use a selection of pixels from the map to fit the model. A simple approach would be to randomly sample points from the existing maps and then use them to train the model, but this has a disadvantage of (wrongly) assuming that the map is absolutely the same quality in all parts of the area. Instead, we can place the training pixels along the medial axes for polygons of interest. The medial axes can be derived in SAGA, but we need to convert the gridded map first to a polygon map, then extract lines, and then derive the buffer distance map:

```
# convert the raster map to polygon map:
> rsaga.esri.to.sgrd(in.grids="soilmu.asc", out.sgrd="soilmu.sgrd",
+   in.path=getwd())
> rsaga.geoprocessor(lib="shapes_grid", module=6, param=list(GRID="soilmu.sgrd",
+   SHAPES="soilmu.shp", CLASS_ALL=1))
# convert the polygon to line map:
> rsaga.geoprocessor(lib="shapes_lines", module=0,
+   param=list(POLYGONS="soilmu.shp", LINES="soilmu_l.shp"))
```

<sup>14</sup>Local variograms for altitude data can be derived in the Digeman software provided by Bishop et al. (2006).

<sup>15</sup>Mapping units drawn manually, by doing photo-interpretation or following some similar procedure.

```

# derive the buffer map using the shapefile:
> rsaga.geoprocessor(lib="grid_gridding", module=0,
+   param=list(GRID="soilmu_r.sgrd", INPUT="soilmu_l.shp", FIELD=0, LINE_TYPE=0,
+   TARGET_TYPE=0, USER_CELL_SIZE=pixelsize,
+   USER_X_EXTENT_MIN=grids5m@bbox[1,1]+pixelsize/2,
+   USER_X_EXTENT_MAX=grids5m@bbox[1,2]-pixelsize/2,
+   USER_Y_EXTENT_MIN=grids5m@bbox[2,1]+pixelsize/2,
+   USER_Y_EXTENT_MAX=grids5m@bbox[2,2]-pixelsize/2))
# buffer distance:
> rsaga.geoprocessor(lib="grid_tools", module=10,
+   param=list(SOURCE="soilmu_r.sgrd", DISTANCE="soilmu_dist.sgrd",
+   ALLOC="tmp.sgrd", BUFFER="tmp.sgrd", DIST=sqrt(areaSpatialGrid(grids25m))/3,
+   IVAL=pixelsize))
# surface specific points (medial axes!):
> rsaga.geoprocessor(lib="ta_morphometry", module=3,
+   param=list(ELEVATION="soilmu_dist.sgrd", RESULT="soilmu_medial.sgrd",
+   METHOD=1))

```

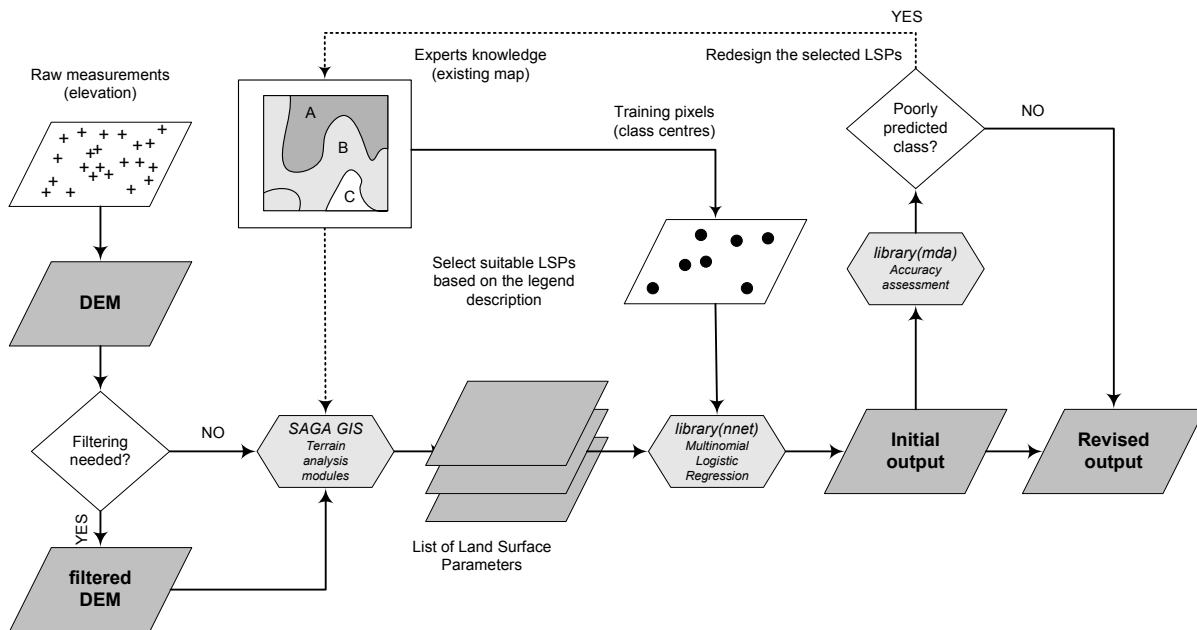


Fig. 9.5: Data analysis scheme and connected R packages: supervised extraction of geomorphological classes using the existing geomorphological map — a hybrid expert/statistical based approach.

- 1 The map showing medial axes can then be used as a weight map to randomize the sampling (see further
- 2 Fig. 9.6a). The sampling design can be generated using the `rpoint` method<sup>16</sup> of the `spatstat` package:

```

# read into R:
> rsaga.sgrd.to.esri(in.sgrds="soilmu_medial.sgrd",
+   out.grids="soilmu_medial.asc", prec=0, out.path=getwd())
> grids5m$soilmu_medial <- readGDAL("soilmu_medial.asc")$band1
# generate the training pixels:
> grids5m$weight <- abs(ifelse(grids5m$soilmu_medial>=0, 0, grids5m$soilmu_medial))
> dens.weight <- as.im(as.image.SpatialGridDataFrame(grids5m["weight"]))
# image(dens.weight)

```

<sup>16</sup>This will generate a point pattern given a prior probability i.e. a mask map.

```

> training.pix <- rpoint(length(grids5m$weight)/10, f=dens.weight)
# plot(training.pix)
> training.pix <- data.frame(x=training.pix$x, y=training.pix$y,
+   no=1:length(training.pix$x))
> coordinates(training.pix) <- ~ x+y

```

This reflects the idea of sampling the class centers, at least in the geographical sense. The advantage of using the medial axes is that also relatively small polygons will be represented in the training pixels set (or in other words — large polygons will be under-represented, which is beneficial for the regression modeling). The most important is that the algorithm will minimize selection of transitional pixels that might well be in either of the two neighboring classes.

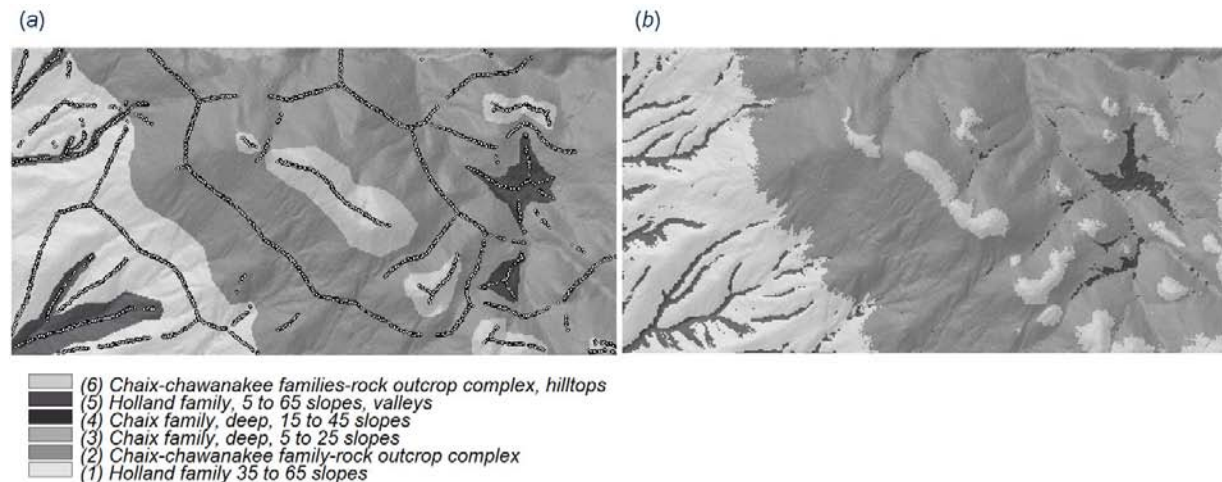


Fig. 9.6: Results of predicting soil mapping units using DEM-derived LSPs: (a) original soil mapping units and training pixels among medial axes, (b) soil mapping units predicted using multinomial logistic regression.

Once we have allocated the training pixels, we can fit a logistic regression model using the `nnet` package, and then predict the mapping units for the whole area of interest:

```

# overlay the training points and grids:
> training.pix.ov <- overlay(grids5m, training.pix)
> library(nnet)
> mlr.soilmu <- multinom(soilmu.c ~ DEM5LIDARf+TWI+VDEPTH+INSOLAT+CONVI, training.pix.ov)

# weights: 42 (30 variable)
initial value 14334.075754
iter 10 value 8914.610698
iter 20 value 8092.253630
...
iter 100 value 3030.721321
final value 3030.721321
stopped after 100 iterations

# make predictions:
> grids5m$soilmu.mlr <- predict(mlr.soilmu, newdata=grids5m)

```

Finally, we can compare the map generated using multinomial logistic regression versus the existing map (Fig. 9.6). To compare the overall fit between the two maps we can use the `mda` package:

```

> library(mda) # kappa statistics
> sel <- !is.na(grids5m$soilmu.c)
> Kappa(confusion(grids5m$soilmu.c[sel], grids5m$soilmu.mlr[sel]))

```

	value	ASE
Unweighted	0.6740377	0.002504416
Weighted	0.5115962	0.003207276

1 which shows that the matching between the two maps is 51–67%. A relatively low kappa is typical for soil  
 2 and/or geomorphological mapping applications<sup>17</sup>. We have also ignored that these map units represent suites  
 3 of soil types, stratified by prominence of rock outcroppings and by slope classes and NOT uniform soil bodies.  
 4 Nevertheless, the advantage of using a statistical procedure is that it reflects the experts knowledge more  
 5 objectively. The results will typically show more spatial detail (small patches) than the hand drawn maps.  
 6 Note also that the multinom method implemented in the nnet package is a fairly robust technique in the  
 7 sense that it generates few artifacts. Further refinement of existing statistical models (regression-trees and/or  
 8 machine-learning algorithms) could also improve the mapping of landform categories.

## 9.7 Extraction of memberships

10 We can also extend the analysis and extract member-  
 11 ships for the given soil mapping units, following the  
 12 fuzzy *k*-means algorithm described in Hengl et al.  
 13 (2004c). For this purpose we can use the same train-  
 14 ing pixels, but then associate the pixels to classes just  
 15 by standardizing the distances in feature space. This  
 16 is a more trivial approach than the multinomial logis-  
 17 tic regression approach used in the previous exercise.

18 The advantage of using membership, on the  
 19 other hand, is that one can observe how crisp cer-  
 20 tain classes are, and where the confusion of classes  
 21 is the highest. This way, the analyst has an oppor-  
 22 tunity to focus on mapping a single geomorpholog-  
 23 ical unit, adjust training pixels where needed and  
 24 increase the quality of the final maps (Fisher et al.,  
 25 2005). A supervised fuzzy *k*-means algorithm is not  
 26 implemented in any R package (yet), so we will de-  
 27 rive memberships step-by-step.

28 First, we need to estimate the class centers (mean  
 29 and standard deviation) for each class of interest:

```
# mask-out classes with <5 points:
mask.c <- as.integer(attr(summary(training.pix.ov$soilmu.c
+   [summary(training.pix.ov$soilmu.c)<5]), "names"))
# fuzzy exponent:
> fuzzy.e <- 1.2
# extract the class centroids:
> class.c <- aggregate(training.pix.ov@data[c("DEM5LIDARf", "TWI", "VDEPTH",
+   "INSOLAT", "CONVI)], by=list(training.pix.ov$soilmu.c), FUN="mean")
> class.sd <- aggregate(training.pix.ov@data[c("DEM5LIDARf", "TWI", "VDEPTH",
+   "INSOLAT", "CONVI)], by=list(training.pix.ov$soilmu.c), FUN="sd")
```

30 which allows us to derive diagonal/standardized distances between the class centers and all individual pixels:

```
# derive distances in feature space:
> distmaps <- as.list(levels(grid5m$soilmu.c)[mask.c])
> tmp <- rep(NA, length(grid5m@data[[1]]))
> for(c in (1:length(levels(grid5m$soilmu.c)))[mask.c]){
+   distmaps[[c]] <- data.frame(DEM5LIDARf=tmp, TWI=tmp, VDEPTH=tmp,
+   INSOLAT=tmp, CONVI=tmp)
>   for(j in list("DEM5LIDARf", "TWI", "VDEPTH", "INSOLAT", "CONVI")){
>     distmaps[[c]][j] <- ((grid5m@data[j]-class.c[c,j])/class.sd[c,j])^2
```



Fig. 9.7: Membership values for soil mapping unit: Chaix family, deep, 15 to 45% slopes.

<sup>17</sup>An in-depth discussion can be followed in Kempen et al. (2009).

```

> }
> }
# sum up distances per class:
> distsum <- data.frame(tmp)
> for(c in (1:length(levels(grid5m$soilmu.c)))[mask.c]){
>   distsum[paste(c)] <- sqrt(rowSums(distmaps[[c]], na.rm=T, dims=1))
> }
> str(distsum)

'data.frame':  80000 obs. of  6 variables:
 $ 1: num  1.53 1.56 1.75 2.38 3.32 ...
 $ 2: num  4.4 4.41 4.53 4.88 5.48 ...
 $ 3: num 37.5 37.5 37.5 37.5 37.6 ...
 $ 4: num  10 10 10.1 10.2 10.5 ...
 $ 5: num  2.64 2.54 3.18 4.16 5.35 ...
 $ 6: num  8.23 8.32 8.28 8.06 7.58 ...

# total sum of distances for all pixels:
> totsum <- rowSums(distsum^(-2/(fuzzy.e-1)), na.rm=T, dims=1)

```

Once we have estimated the standardized distances, we can then derive memberships using formula (Sokal and Sneath, 1963):

$$\mu_c(i) = \frac{[d_c^2(i)]^{-\frac{1}{(q-1)}}}{\sum_{c=1}^k [d_c^2(i)]^{-\frac{1}{(q-1)}}} \quad c = 1, 2, \dots, k \quad i = 1, 2, \dots, n \quad (9.7.1)$$

$$\mu_c(i) \in [0, 1] \quad (9.7.2)$$

where  $\mu_c(i)$  is a fuzzy membership value of the  $i$ th object in the  $c$ th cluster,  $d$  is the similarity (*diagonal*) distance,  $k$  is the number of clusters and  $q$  is the fuzzy exponent determining the amount of fuzziness. Or in R syntax:

```

> for(c in (1:length(levels(grid5m$soilmu.c)))[mask.c]){
>   grids5m@data[paste("mu_", c, sep="")] <-
+     (distsum[paste(c)]^(-2/(fuzzy.e-1))/totsum)[,1]
> }

```

The resulting map of memberships for class “*Chaix family, deep, 15 to 45% slopes*” is shown in Fig. 9.7. Compare also with Fig. 9.6.

### Self-study exercises:

- (1.) How much is elevation correlated with the TWI map? (derive correlation coefficient between the two maps)
- (2.) Which soil mapping unit in Fig. 9.6 is the most correlated to the original map? (HINT: convert to indicators and then correlate the maps.)
- (3.) Derive the variogram for the filtered LIDAR DEM and compare it to the variogram for elevations derived using the (LiDAR) point measurements. (HINT: compare nugget, sill, range parameter and anisotropy parameters.)

- 1 (4.) Extract landform classes using the pam algorithm as implemented in the cluster package and compare it  
2 with the results of kmeans. Which polygons/classes are the same in >50% of the area? (HINT: overlay  
3 the two maps and derive summary statistics.)
- 4 (5.) Extract the landform classes using unsupervised classification and the coarse SRTM DEM and identify if  
5 there are significant differences between the LiDAR based and coarse DEM. (HINT: compare the average  
6 area of landform unit; derive a correlation coefficient between the two maps.)
- 7 (6.) Try to add five more LSPs and then re-run multinomial logistic regression. Did the fitting improve and  
8 how much? (HINT: Compare the resulting AIC for fitted models.)
- 9 (7.) Extract membership maps (see section 9.7) for all classes and derive the confusion index. Where is the  
10 confusion index the highest? Is it correlated with any input LSP?

#### 11 Further reading:

- 12 ★ Brenning, A., 2008. Statistical Geocomputing combining R and SAGA: The Example of Landslide suscep-  
13 tibility Analysis with generalized additive Models. In: J. Böhrner, T. Blaschke & L. Montanarella (eds.),  
14 **SAGA — Seconds Out** (Hamburger Beiträge zur Physischen Geographie und Landschaftsökologie, 19),  
15 23–32.
- 16 ★ Burrough, P. A., van Gaans, P. F. M., MacMillan, R. A., 2000. High-resolution landform classification  
17 using fuzzy k-means. *Fuzzy Sets and Systems* 113, 37–52.
- 18 ★ Conrad, O., 2007. **SAGA — Entwurf, Funktionsumfang und Anwendung eines Systems für Automa-**  
19 **tisierte Geowissenschaftliche Analysen**. Ph.D. thesis, University of Göttingen, Göttingen.
- 20 ★ Fisher, P. F., Wood, J., Cheng, T., 2005. Fuzziness and ambiguity in multi-scale analysis of landscape mor-  
21 phometry. In: Petry, F. E., Robinson, V. B., Cobb, M. A. (Eds.), *Fuzzy Modeling with Spatial Information*  
22 for Geographic Problems. Springer-Verlag, Berlin, pp. 209–232.
- 23 ★ Smith, M. J., Paron, P. and Griffiths, J. (eds) 2010. **Geomorphological Mapping: a professional hand-**  
24 **book of techniques and applications**. *Developments in Earth Surface Processes*, Elsevier, in prepara-  
25 tion.