Land cover map for multifunctional landscapes of Taita Taveta County, Kenya, based on Sentinel-1 radar, Sentinel-2 optical, and topoclimatic data

Temesgen Abera[1*, 2], Ilja Vuorinne[1,2], Martha Munyao[1,3], Petri Pellikka[1,2,4], Janne Heiskanen[1]

[1] Department of Geosciences and Geography, University of Helsinki, P.O. Box 68, FI-00014 Helsinki, Finland; ilja.vuorinne@helsinki.fi (I.V.); martha.munyao@helsinki.fi (M.M.); petri.pellikka@helsinki.fi (P.P.); janne.heiskanen@helsinki.fi (J.H.)
[2] Institute for Atmospheric and Earth System Research, Faculty of Science, University of Helsinki,P.O.Box 4, FI-00014 Helsinki, Finland
[3] Kenya Wildlife Service, P.O. Box 40241, 00100 Nairobi, Kenya
[4] State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China
* Correspondence: temesgen.abera@helsinki.fi

Supplementary material

The below script shows the Google Earth Engine (GEE) java script used to prepare Taita Taveta County land cover map.

```
//------------------------------------------------------------------------------//
// Code by Abera, T.
// Land cover map and reference database: https://data.mendeley.com/datasets/xv24ngy2dz/2
//-------------------------------------------------------------
// To run the code: provide your study area boundary, export intermediate results (median, quantiles, etc) to asset and load these together with reference database as input for the gradient tree boost classification
//-------------------------------------------------------------
// Study area boundary
var region = ee.FeatureCollection("users/...../study_area_boundary"); //path to study area boundary from asset
Map.centerObject(region);
//-----------------------------------------------------------------------------
// Sentinel-1 Analysis Ready Data (ARD) preparation in Google Earth Engine, based on
// Mullissa et al. (2021). Remote Sens., 13, 1954. https://doi.org/10.3390/rs13101954
//-----------------------------------------------------------------------------
var wrapper = require('users/adugnagirma/gee_s1_ard:wrapper'); // functions for radiometric terrain normalization, speckle filtering, and border-noise corrections
var helper = require('users/adugnagirma/gee_s1_ard:utilities');// scale conversion between db (decibel) and linear
// DEFINE PARAMETERS FOR SENTINEL-1
// Select short dry season (January and February)
var parameter = {//1. Data Selection
        START_DATE: "2020-01-01",
        STOP_DATE: "2020-02-29",
        POLARIZATION:'VVVH',
        ORBIT : 'ASCENDING',
        GEOMETRY: region
        //2. Additional Border noise correction
        APPLY_ADDITIONAL_BORDER_NOISE_CORRECTION: true,
        //3. Speckle filter
        APPLY_SPECKLE_FILTERING: true,
        SPECKLE_FILTER_FRAMEWORK: 'MULTI',
        SPECKLE_FILTER: 'LEE',
        SPECKLE_FILTER_KERNEL_SIZE: 9,
        SPECKLE_FILTER_NR_OF_IMAGES: 10,
        //4. Radiometric terrain normalization
        APPLY_TERRAIN_FLATTENING: true,
```

```
        DEM: ee.Image('USGS/SRTMGL1_003'),
        TERRAIN_FLATTENING_MODEL: 'VOLUME',
        TERRAIN_FLATTENING_ADDITIONAL_LAYOVER_SHADOW_BUFFER: 0,
        //5. Output
        FORMAT : 'DB',
        CLIP_TO_ROI: true,
        SAVE_ASSETS: false
}
// Preprocess the S1 collection
var s1_preprocces = wrapper.s1_preproc(parameter);
var s1 = s1_preprocces[0]
s1_preprocces = s1_preprocces[1]
print(s1_preprocces);
print(s1);

// Visulization of the first image in the collection in RGB for VV, VH, images
var visparam = {}
if (parameter.POLARIZATION=='VVVH'){
    if (parameter.FORMAT=='DB'){
   var s1_preprocces_view = s1_preprocces.map(helper.add_ratio_lin).map(helper.lin_to_db2);
   var s1_view = s1.map(helper.add_ratio_lin).map(helper.lin_to_db2);
   visparam = {bands:['VV','VH','VVVH_ratio'],min: [-20, -25, 1],max: [0, -5, 15]}
   }
   else {
   var s1_preprocces_view = s1_preprocces.map(helper.add_ratio_lin);
   var s1_view = s1.map(helper.add_ratio_lin);
   visparam = {bands:['VV','VH','VVVH_ratio'], min: [0.01, 0.0032, 1.25],max: [1, 0.31, 31.62]}
   }
}
else {
   if (parameter.FORMAT=='DB') {
   s1_preprocces_view = s1_preprocces.map(helper.lin_to_db);
   s1_view = s1.map(helper.lin_to_db);
   visparam = {bands:[parameter.POLARIZATION],min: -25,max: 0}
   }
   else {
   s1_preprocces_view = s1_preprocces;
   s1_view = s1;
   visparam = {bands:[parameter.POLARIZATION],min: 0,max: 0.2}
   }
}
Map.centerObject(parameter.GEOMETRY, 12);
Map.addLayer(s1_view.first(), visparam, 'First image in the input S1 collection', true);
Map.addLayer(s1_preprocces_view.first(), visparam, 'First image in the processed S1 collection', true);
Map.addLayer(s1_preprocces_view,  {color: '00FF00'}, 'SAR', true);
//print(s1_preprocces_view);
//-----------------------------------------------
// spectral metrics: median and quantile (Q1,Q3)
//-----------------------------------------------
var composite = s1_preprocces_view.select(['VV', 'VH', 'VVVH_ratio']).median().clip(region);
Map.addLayer(composite, visparam, 'S1 median composite');
var medianvv = s1_preprocces_view.select('VV').reduce(ee.Reducer.median()).clip(region);
var medianvh = s1_preprocces_view.select('VH').reduce(ee.Reducer.median()).clip(region);
var medianvvvh = s1_preprocces_view.select('VVVH_ratio').reduce(ee.Reducer.median()).clip(region);
var Q1vv = s1_preprocces_view.select('VV').reduce(ee.Reducer.percentile([25])).clip(region);
var Q1vh = s1_preprocces_view.select('VH').reduce(ee.Reducer.percentile([25])).clip(region);
var Q1vvvh = s1_preprocces_view.select('VVVH_ratio').reduce(ee.Reducer.percentile([25])).clip(region);
```

```
var Q3vv = s1_preprocces_view.select('VV').reduce(ee.Reducer.percentile([75])).clip(region);
var Q3vh = s1_preprocces_view.select('VH').reduce(ee.Reducer.percentile([75])).clip(region);
var Q3vvvh = s1_preprocces_view.select('VVVH_ratio').reduce(ee.Reducer.percentile([75])).clip(region);
// Visualize the first quantiles of vv polarization (Q1vv)
Map.addLayer(Q1vv,  {color: '00FF00'}, 'Q31vv', true);
//------------------------
// Export results to asset
//------------------------
// Example for Q1vv
Export.image.toAsset({
  image:Q1vv ,
 description: 'vv_Q1',
 region: region,
 scale: 20, // adjust for desired resolution
});
//-------------------------------------
// Sentinel-2 MSI pre-processing
// Input data using surface reflectance
//-------------------------------------
var s2Sr = ee.ImageCollection('COPERNICUS/S2_SR');
var s2Clouds = ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY');
var START_DATE = ee.Date('2020-01-01');
var END_DATE = ee.Date('2020-12-31');
var MAX_CLOUD_PROBABILITY = 20;
//var region = ee.FeatureCollection("users/...../study_area_boundary");
//Map.setCenter(region);
//Map.addLayer(region,  {color: '00FF00'}, 'AOI extent', true);
// Cloud mask
function maskClouds(img) {
  var clouds = ee.Image(img.get('cloud_mask')).select('probability');
  var isNotCloud = clouds.lt(MAX_CLOUD_PROBABILITY);
  return img.updateMask(isNotCloud);
}
// Additional masks to exclude bad data at scene edges
function maskEdges(s2_img) {
  return s2_img.updateMask(
      s2_img.select('B8A').mask().updateMask(s2_img.select('B9').mask()));
}
// Filter collections by desired data range and region.
var criteria = ee.Filter.and(
    ee.Filter.bounds(region), ee.Filter.date(START_DATE, END_DATE));
s2Sr = s2Sr.filter(criteria).map(maskEdges);
s2Clouds = s2Clouds.filter(criteria);
// Join S2 SR with cloud probability dataset to add cloud mask.
var s2SrWithCloudMask = ee.Join.saveFirst('cloud_mask').apply({
  primary: s2Sr,
  secondary: s2Clouds,
  condition:
      ee.Filter.equals({leftField: 'system:index', rightField: 'system:index'})
});
var bands = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7','B8','B8A','B11','B12'];
// Visualizing some composites
var Composites =
    ee.ImageCollection(s2SrWithCloudMask).map(maskClouds).median().divide(10000)
    .select(['B4', 'B3', 'B2']).clip(region);
var rgbVis = {min: 0, max: 0.3, bands: ['B4', 'B3', 'B2']};
Map.addLayer(Composites, rgbVis, 'MedianComposite', true);
```

```
// Mask the clouds
var s2CloudMasked =
    ee.ImageCollection(s2SrWithCloudMask).map(maskClouds)
    .select(bands);
var clip = function(image){
 return image.clip(region);
};
//---------------------------------------
// Spectral metrics: median and quantile
//---------------------------------------
var result = s2CloudMasked.reduce(ee.Reducer.median()).clip(region).divide(10000);
var Q1 = s2CloudMasked.reduce(ee.Reducer.percentile([25])).clip(region).divide(10000);
var Q3 = s2CloudMasked.reduce(ee.Reducer.percentile([75])).clip(region).divide(10000);
//Select layers for exporting(for example B2_p25)
var B2p25 = Q1.select('B2_p25');
//Select layers for exporting(for example B2_p75)
var B2 = Q3.select('B2_p75');
var B3 = Q3.select('B3_p75');
var B4 = Q3.select('B4_p75');
var B5 = Q3.select('B5_p75');
var B6 = Q3.select('B6_p75');
var B7 = Q3.select('B7_p75');
var B8 = Q3.select('B8_p75');
var B8A = Q3.select('B8A_p75');
var B11 = Q3.select('B11_p75');
var B12 = Q3.select('B12_p75');
// Calculate vegetation indices: NDVI, TCG(tasseled cap greeness), and TCW(tasseled cap wetness),
var ndvi = result.expression('(NIR - RED)/(NIR + RED)', {
    'NIR': result.select('B8'),
    'RED': result.select('B4')
});
// Tasseled cap wetness and greeness(coefficients from Lastovicka  et al. 2020, Remote Sensing)
   var tcw = result.expression('(0.1506*Blue + 0.1973*Green + 0.3279*Red + 0.3406*NIR - 0.7112*SWIRA- 0.4572*SWIRB)', {
    'Blue': result.select('B2'),
    'Green': result.select('B3'),
    'Red': result.select('B4'),
    'NIR': result.select('B8'),
    'SWIRA': result.select('B11'),
    'SWIRB': result.select('B12')
});
 var tcg = result.expression('(-0.28482*Blue - 0.24353*Green - 0.54364*Red + 0.72438*NIR + 0.084011*SWIRA -
0.180012*SWIRB)', {
    'Blue': result.select('B2'),
    'Green': result.select('B3'),
    'Red': result.select('B4'),
    'NIR': result.select('B8'),
    'SWIRA': result.select('B11'),
    'SWIRB': result.select('B12')
});
// Export the various layers to asset
// For example
 Export.image.toAsset({
  image:ndvi ,
 description: 'NDVI',
 region: region,
 scale: 20, // adjust for desired resolution
});
```

```
//------------------------
//Extract topoclimate data
//------------------------
// elevation, slope, aspect
var elevation = ee.Image("USGS/SRTMGL1_003");
var topo = ee.Algorithms.Terrain(elevation).clip(region);
var elevation = topo.select(['elevation']);
var slope = topo.select(['slope']);
var aspect = topo.select(['aspect']);
//Rainfall data is from WORLDCLIM
//var bio = ee.Image('WORLDCLIM/V1/BIO').select(['bio12']).rename(['RAINFALL']).clip(region);
// Export the various layers to asset
 //-------------------------------------------------------
// Input data for gradient boost machine (GBM) classification
//-------------------------------------------------------
// Load input data from previous steps saved in asset
// Medians
var B2 = ee.Image('users/temegis/TTCSentinel/B2').rename('B2');
var B3 = ee.Image('users/...../B3'); // path to median of band 3
var B4 = ee.Image('users/...../B4');
var B5 = ee.Image('users/...../B5');
var B6 = ee.Image('users/...../B6');
var B7 = ee.Image('users/...../B7');
var B8 = ee.Image('users/...../B8');
var B8A = ee.Image('users/...../B8A');
var B11 = ee.Image('users/...../B11');
var B12 = ee.Image('users/...../B12');
var ndvi = ee.Image('users/...../NDVI');
var tcw = ee.Image('users/...../TCW');
var tcg = ee.Image('users/...../TCG');
var vv = ee.Image('users/...../vv_median');
var vh = ee.Image('users/...../vh_median');
var vvvh = ee.Image('users/...../vvvh_ratio_median');
//25 Percentile, Q1
var C2 = ee.Image('users/...../B2_p25').rename('B2_p25');
var C3 = ee.Image('users/...../B3_p25').rename('B3_p25');
var C4 = ee.Image('users/...../B4_p25').rename('B4_p25');
var C5 = ee.Image('users/...../B5_p25').rename('B5_p25');
var C6 = ee.Image('users/...../B6_p25').rename('B6_p25');
var C7 = ee.Image('users/...../B7_p25').rename('B7_p25');
var C8 = ee.Image('users/...../B8_p25').rename('B8_p25');
var C8A = ee.Image('users/...../B8A_p25').rename('B8A_p25');
var C11 = ee.Image('users/...../B11_p25').rename('B11_p25');
var C12 = ee.Image('users/...../B12_p25').rename('B12_p25');
var Q1vv = ee.Image('users/...../vv_Q1').rename('Q1vv');
var Q1vh = ee.Image('users/...../vh_Q1').rename('Q1vh');
var Q1vvvh = ee.Image('users/...../vvvh_ratio_Q1').rename('Q1vvvh');
//75 Percentile, Q3
var D2=ee.Image('users/temegis/...../B2_p75').rename('B2_p75');
var D3=ee.Image('users/temegis/...../B3_p75').rename('B3_p75');
var D4=ee.Image('users/temegis/...../B4_p75').rename('B4_p75');
var D5=ee.Image('users/temegis/...../B5_p75').rename('B5_p75');
var D6=ee.Image('users/temegis/...../B6_p75').rename('B6_p75');
var D7=ee.Image('users/temegis/...../B7_p75').rename('B7_p75');
var D8=ee.Image('users/temegis/...../B8_p75').rename('B8_p75');
var D8A=ee.Image('users/temegis/...../B8A_p75').rename('B8A_p75');
var D11=ee.Image('users/temegis/...../B11_p75').rename('B11_p75');
```

```
var D12=ee.Image('users/temegis/...../B12_p75').rename('B12_p75');
var Q3vv=ee.Image('users/temegis/...../vv_Q3').rename('Q3vv');
var Q3vh=ee.Image('users/temegis/...../vh_Q3').rename('Q3vh');
var Q3vvvh=ee.Image('users/temegis/...../vvvh_Q3').rename('Q3vvvh');
//Interquartile range, Q2 to Q12 are from Sentinel-2.  Q13 to Q15 are from Sentinel-1.
var Q2 = D2.subtract(C2).rename('IQ2');
var Q3 = D3.subtract(C3).rename('IQ3');
var Q4 = D4.subtract(C4).rename('IQ4');
var Q5 = D5.subtract(C5).rename('IQ5');
var Q6 = D6.subtract(C6).rename('IQ6');
var Q7 = D7.subtract(C7).rename('IQ7');
var Q8 = D8.subtract(C8).rename('IQ8');
var Q8A = D8A.subtract(C8A).rename('IQ8A');
var Q11 = D11.subtract(C11).rename('IQ11');
var Q12 = D12.subtract(C12).rename('IQ12');
var Q13 = Q3vv.subtract(Q1vv).rename('IQ13');
var Q14 = Q3vh.subtract(Q1vh).rename('IQ14');
var Q15 = Q3vvvh.subtract(Q1vvvh).rename('IQ15');
//stack input datasets
var result = B2.addBands(B3.rename('B3')).addBands(B4.rename('B4'))
.addBands(B5.rename('B5')).addBands(B6.rename('B6')).addBands(B7.rename('B7'))
.addBands(B8.rename('B8')).addBands(B8A.rename('B8A')).addBands(B11.rename('B11'))
.addBands(B12.rename('B12'))
.addBands(ndvi.rename('NDVI'))
.addBands(tcw.rename('TCW')).addBands(tcg.rename('TCG')).addBands(topo.rename('Elevation','Slope','Aspect'))
.addBands(bio.rename('Precip')).addBands(WoodyCover)
.addBands(C2).addBands(C3).addBands(C4).addBands(C5).addBands(C6)
.addBands(C7).addBands(C8).addBands(C8A).addBands(C11).addBands(C12)
.addBands(D2).addBands(D3).addBands(D4).addBands(D5).addBands(D6).addBands(D7)
.addBands(D8).addBands(D8A).addBands(D11).addBands(D12).addBands(Q2).addBands(Q3)
.addBands(Q4).addBands(Q5).addBands(Q6).addBands(Q7).addBands(Q8)
.addBands(Q8A).addBands(Q11).addBands(Q12).addBands(vv.rename('vv')).addBands(vh.rename('vh'))
.addBands(vv.rename('vvvh')).addBands(Q1vv).addBands(Q1vh).addBands(Q1vvvh).addBands(Q3vv)
.addBands(Q3vh).addBands(Q3vvvh).addBands(Q13).addBands(Q14).addBands(Q15).clip(region);
// Load training sample points from reference database for classification
// sample points were buffered at 18m
var newfc = ee.FeatureCollection("users/...../reference_database");
// Select the bands for training
// Exclude B11 as it has high correlation with B12 and other bands
var predics = ['B2','B3', 'B4', 'B5', 'B6', 'B7','B8','B8A','B12','NDVI','TCW','TCG','Elevation',
'Slope','Aspect','Precip','B2_p25','B3_p25', 'B4_p25', 'B5_p25', 'B6_p25', 'B7_p25','B8_p25',
'B8A_p25','B11_p25','B12_p25','B2_p75','B3_p75', 'B4_p75', 'B5_p75', 'B6_p75', 'B7_p75','B8_p75',
'B8A_p75','B11_p75','B12_p75','IQ2','IQ3', 'IQ4', 'IQ5', 'IQ6', 'IQ7','IQ8','IQ8A','IQ11','IQ12',
'vv','vh','Q1vv','Q3vv','Q3vvvh','IQ13','IQ14','IQ15'];
var training = result.select(predics).sampleRegions({
  collection: newfc,
  properties: ['Landcover'],
 scale: 20,
  //numPixels: 5000
});
// Split the data for training (70%) and validation (30%)
var withRandom = training.randomColumn('random');
var split = 0.7;  // Roughly 70% training, 30% testing.
var trainingPartition = withRandom.filter(ee.Filter.lt('random', split));
var testingPartition = withRandom.filter(ee.Filter.gte('random', split));
// Gradient tree boost classification
var trainedClassifier = ee.Classifier.smileGradientTreeBoost(100);
```

```
    .train({
     features: trainingPartition,
      classProperty: 'Landcover',
     inputProperties: predics
});
 // Classify the input imagery.
var classified = result.select(predics).classify(trainedClassifier);
var trainAccuracy = trainedClassifier.confusionMatrix();
//print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());
// Classify using testing dataset
var test = testingPartition.classify(trainedClassifier);
// Print the confusion matrix.
var confusionMatrix = test.errorMatrix('Landcover', 'classification');
print('GBM testing error matrix', confusionMatrix);
print('Validated accuracy', confusionMatrix.accuracy());
// Variable importance
var dict = trainedClassifier.explain();
var variable_importance = ee.Feature(null, ee.Dictionary(dict).get('importance'));
var chart =
  ui.Chart.feature.byProperty(variable_importance)
   .setChartType('ColumnChart')
    .setOptions({
     title: 'Gradient Tree Boost Variable Importance',
      legend: {position: 'none'},
      hAxis: {title: 'Bands'},
      vAxis: {title: 'Importance'}
});
print(chart);
//Map.addLayer(classified.randomVisualizer(), {}, 'Land Use Classification');
 //Post classification smoothing
var smoothedResult = classified.focal_mode({radius:1.5});
Map.addLayer(smoothedResult.randomVisualizer(), {}, 'SmoothedLC');
```

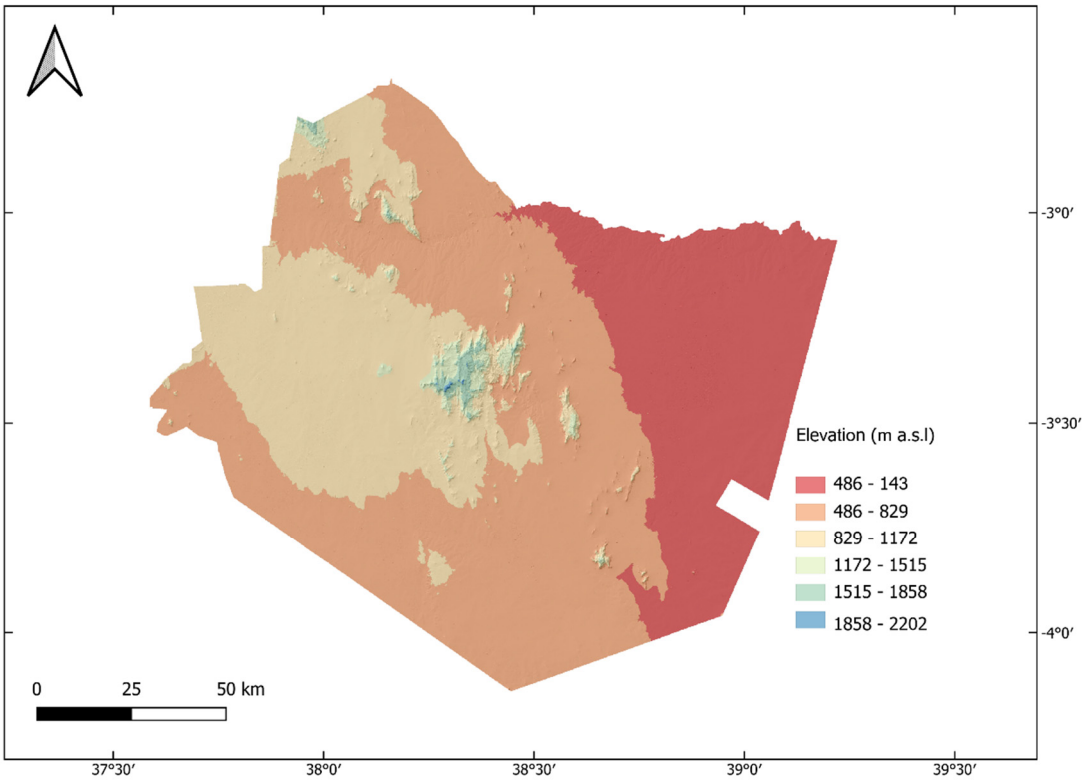

Figure S1. Elevation map of the study area (JAXA DEM 30 m resolution).

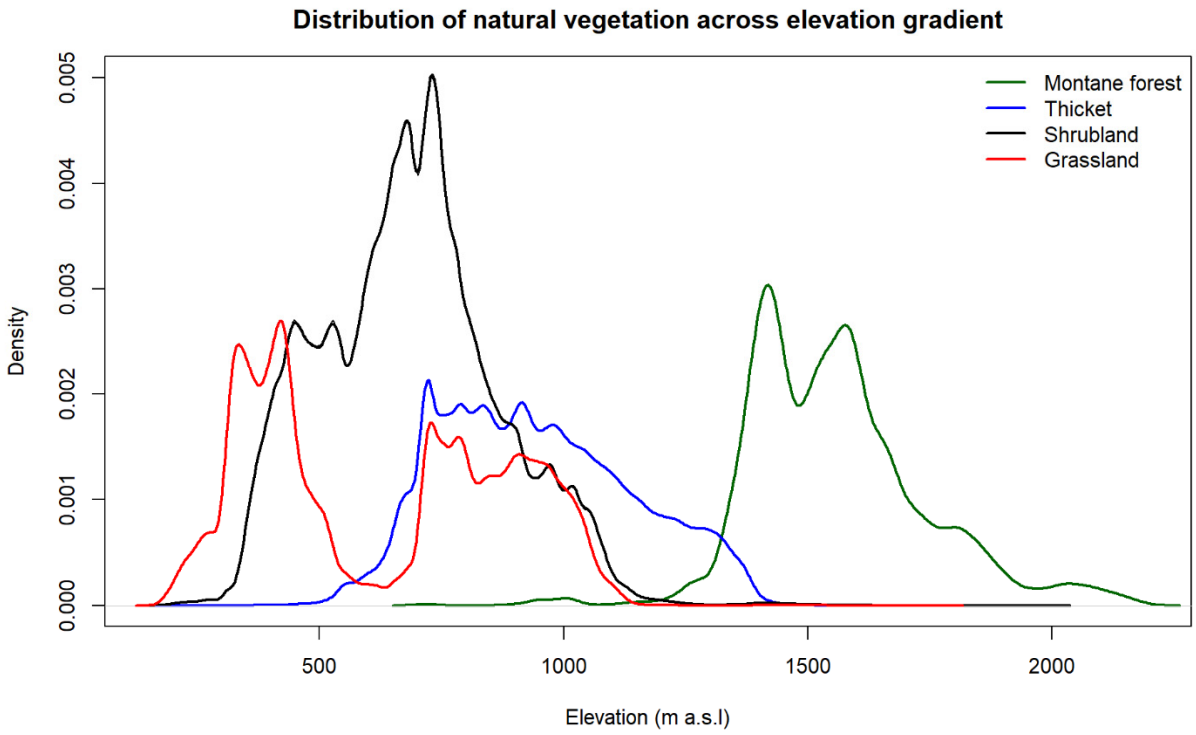

Figure S2. Probability density plot showing distribution of natural vegetation across elevation gradient in the study area.