// NASA\_NEXDCO30\_AtRiskSnow

// Calculate At-risk Snow (ARS)

// from NASA/NEX-DCP30 downscaled climate data

// Created 8 February 2017, David E. Rupp, david.rupp@oregonstate.edu

//

// Updated and modified by Gene Mar 8 June 2017

// Modified categories and color pallete and added legends: David Rupp 18 Sep 2017

// HUC selection added: David Rupp 18 Sep 2017

// HUC selection/extraction updaed by Gene Mar 28 Sep 2017

// Statistics calculation by HUC added: Gene Mar 29 Sep 2017

//

// References: https://cds.nccs.nasa.gov/nex/

// https://www.skepticalscience.com/rcp.php?t=1

var collectionPath = 'NASA/NEX-DCP30';

var skiResorts = ee.FeatureCollection('ft:1q\_x12r4gjHhPpnGB2krnlsKIEks6Yr7TzpCzY875');

var scenario1 = 'historical';

var scenario2 = 'rcp85';

var scenario1Updated = 0;

var scenario2Updated = 0;

var scenario1YearsUpdated = 0;

var scenario2YearsUpdated = 0;

var GCMUpdated = 0;

var GCM = 'ACCESS1-0';

var huc = 'None';

var WaterShedsFC;

var ARS1;

var ARS2;

var deltaARS;

var theText;

var layerText;

var initLon = -100;

var initLat = 38;

var nexdcp30 = {};

// Creates and styles 1 row of the legend.

var makeRow = function(color, name) {

// Create the label that is actually the colored box.

var colorBox = ui.Label({

style: {

backgroundColor: '#' + color,

// Use padding to give the box height and width.

padding: '8px',

margin: '0 0 4px 0'

}

});

// Create the label filled with the description text.

var description = ui.Label({

value: name,

style: {margin: '0 0 4px 6px'}

});

// return the panel

return ui.Panel({

widgets: [colorBox, description],

layout: ui.Panel.Layout.Flow('horizontal')

});

};

// 0 Snw white: FFFAFA - "Safe" snow

// 1 Orange: FFA500 - At-risk snow

// 2 Gray: B8B8B8 - Negligible snow

// Palette with the colors

var palette3ars =['FFFAFA', 'FFA500','B8B8B8'];

// name of the legend

var names3ars = ['Safe snow','At-risk snow','Negligible snow'];

// Palette with the colors

var palette =['FFFAFA', 'FF0000','FFA500','8B4513','F7E084','B8B8B8'];

//Map.addLayer(deltaARS, {min: 0, max: 6, palette: ['FFFAFA', 'FF0000','FFA500','8B4513','F7E084','000000','B8B8B8']}, 'Change');

// 0 White: FFFAFA - Safe snow to safe snow

// 1 Red: FF0000 - Safe snow to at-risk snow

// 2 Orange: FFA500 - At-risk snow to at-risk snow

// 3 Saddle brown: 8B4513 - Safe snow to negligible snow

// 4 Yellow/beige: F7E084 - At-risk to negligible snow

// 5 Black: 000000 - Not used

// 6 Gray: B8B8B8 - Negligible snow to negligible snow

nexdcp30.arsApply = function() {

Map.clear();

// Get ARS1

print(scenario1);

ARS1 = nexdcp30.ars(scenario1, 1);

// Get ARS2

print(scenario2);

ARS2 = nexdcp30.ars(scenario2, 0);

// Add ARS1 to ARS2

deltaARS = ARS2.add(ARS1);

// Reassign some category values

var blank = ee.Image(0);

var dummy = blank.where(ARS1.gt(1),-1);

ARS1 = ARS1.add(dummy);

blank = ee.Image(0);

dummy = blank.where(ARS2.gt(1),-1);

ARS2 = ARS2.add(dummy);

blank = ee.Image(0);

dummy = blank.where(deltaARS.gt(5),-1);

deltaARS = deltaARS.add(dummy);

Map.addLayer(ARS1, {min: 0, max: 2, palette: ['FFFAFA', 'FFA500','B8B8B8']}, scenario1);

// 0 White: FFFFFF - "Safe" snow

// 1 Orange: FFA500 - At-risk snow

// 2 Gray: B8B8B8 - Negligible snow

//var initLon = -100;

//var initLat = 38;

Map.setCenter(initLon, initLat);

Map.addLayer(ARS2, {min: 0, max: 2, palette: ['FFFAFA', 'FFA500','B8B8B8']}, scenario2);

// 0 Snow white: FFFAFA - "Safe" snow

// 1 Orange: FFA500 - At-risk snow

// 2 Gray: B8B8B8 - Negligible snow

//Map.setCenter(lon, lat, 4);

//theText = scenario2.concat(' - ');

//theText = theText.concat(scenario1);

layerText = scenario2 + ' - ' + scenario1;

//Map.addLayer(deltaARS, {min: 0, max: 4, palette: ['FFFFFF', 'FF0000', '000000','F7E084']}, 'Change');

// Map.addLayer(deltaARS, {min: 0, max: 6, palette: ['FFFAFA', 'FF0000','FFA500','8B4513','F7E084','000000','B8B8B8']}, layerText);

// 0 Snow white: FFFAFA - Safe snow to safe snow

// 1 Red: FF0000 - Safe snow to at-risk snow

// 2 Orange: FFA500 - At-risk snow to at-risk snow

// 3 Saddle brown: 8B4513 - Safe snow to negligible snow

// 4 Yellow/beige: F7E084 - At-risk to negligible snow

// 5 Black: 000000 - Not used

// 6 Gray: B8B8B8 - Negligible snow to negligible snow

Map.addLayer(deltaARS, {min: 0, max: 5, palette: ['FFFAFA', 'FF0000','FFA500','8B4513','F7E084','B8B8B8']}, layerText);

// 0 Snow white: FFFAFA - Safe snow to safe snow

// 1 Red: FF0000 - Safe snow to at-risk snow

// 2 Orange: FFA500 - At-risk snow to at-risk snow

// 3 Saddle brown: 8B4513 - Safe snow to negligible snow

// 4 Yellow/beige: F7E084 - At-risk to negligible snow

// 5 Gray: B8B8B8 - Negligible snow to negligible snow

// name of the legend

var names = ['Safe (unchanged)','Safe to at-risk',

'At-risk (unchanged)','Safe to negligible',

'At-risk to negligible','Negligible snow (unchanged)'];

//Map.setCenter(lon, lat, 4);

if (huc != 'None') {

alert('Please select approximate location of watershed on map, but first\n\n \xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0Press OK to continue!');

// SELECT WATERSHED

WaterShedsFC = ee.FeatureCollection('USGS/WBD/2017/HUC' + huc);

Map.onClick(function(coords) {

print(coords);

var point = ee.Geometry.Point([coords.lon, coords.lat]);

var watershed = WaterShedsFC.filter(ee.Filter.bounds(point));

//print(watershed)

var shed = ee.Feature(watershed.first());

//print(shed);

var shedname = shed.get('name');

//print(shedname);

var watershedname = ee.String(shedname).getInfo();

print(watershedname);

//print(point);

print(coords.lon);

print(coords.lat);

Map.addLayer(watershed, {color: '000000'}, watershedname, 1, 0.5);

Map.setCenter(coords.lon, coords.lat, 6);

var roi = watershed.first().geometry();

var ws\_deltaARS = deltaARS.reduceRegion(ee.Reducer.fixedHistogram(0, 6, 6), roi, 1000);

var ws\_ARS1 = ARS1.reduceRegion(ee.Reducer.fixedHistogram(0, 3, 3), roi, 1000);

var ws\_ARS2 = ARS2.reduceRegion(ee.Reducer.fixedHistogram(0, 3, 3), roi, 1000);

/\*\*\*

var minmax\_deltaARS = deltaARS.reduceRegion(ee.Reducer.minMax(), roi, 1000);

var minmax\_ARS1 = ARS1.reduceRegion(ee.Reducer.minMax(), roi, 1000);

var minmax\_ARS2 = ARS2.reduceRegion(ee.Reducer.minMax(), roi, 1000);

print(ws\_deltaARS.getInfo());

print(ws\_ARS1.getInfo());

print(ws\_ARS2.getInfo());

print(minmax\_deltaARS.getInfo());

print(minmax\_ARS1.getInfo());

print(minmax\_ARS2.getInfo());

\*\*\*/

var bucket21 =[];

var bucket2 = [];

var bucket1 = [];

var sum21 = 0;

var sum2 = 0;

var sum1 = 0;

for (var i=0; i < 6; i++) {

bucket21[i] = ee.List(ws\_deltaARS.get('constant')).getInfo()[i][1];

sum21 += bucket21[i];

if (i < 3) {

bucket2[i] = ee.List(ws\_ARS2.get('constant')).getInfo()[i][1];

bucket1[i] = ee.List(ws\_ARS1.get('constant')).getInfo()[i][1];

sum2 += bucket2[i];

sum1 += bucket1[i];

}

}

var maxBucket21 = 0;

var maxBucket2 = 0;

var maxBucket1 = 0;

for (i=0; i < 6; i++) {

bucket21[i] = (100\*bucket21[i]/sum21).toFixed(1);

if (bucket21[i] > maxBucket21) maxBucket21 = bucket21[i];

if (i < 3) {

bucket2[i] = (100\*bucket2[i]/sum2).toFixed(1);

if (bucket2[i] > maxBucket2) maxBucket2 = bucket2[i];

bucket1[i] = (100\*bucket1[i]/sum1).toFixed(1);

if (bucket1[i] > maxBucket1) maxBucket1 = bucket1[i];

}

}

//print(maxBucket);

for (i=0; i < 6; i++) {

if (maxBucket21 == 100) {

bucket21[i] = ('00' + bucket21[i]).slice(-5);

}

else {

bucket21[i] = ('00' + bucket21[i]).slice(-4);

}

if (i < 3) {

if (maxBucket2 == 100) {

bucket2[i] = ('00' + bucket2[i]).slice(-5);

}

else {

bucket2[i] = ('00' + bucket2[i]).slice(-4);

}

if (maxBucket1 == 100) {

bucket1[i] = ('00' + bucket1[i]).slice(-5);

}

else {

bucket1[i] = ('00' + bucket1[i]).slice(-4);

}

}

}

//Map.remove(legend);

var legend = ui.Panel({

style: {

position: 'bottom-right',

padding: '8px 15px'

}

});

legendTitle = ui.Label({

value: watershedname + ' : ' + layerText,

style: {

fontWeight: 'bold',

fontSize: '16px',

margin: '0 0 4px 0',

padding: '0'

}

});

legend.add(legendTitle);

//print(bucket);

names = [bucket21[0] + '% \xa0Safe (unchanged)',

bucket21[1] + '% \xa0Safe to at-risk',

bucket21[2] + '% \xa0At-risk (unchanged)',

bucket21[3] + '% \xa0Safe to negligible',

bucket21[4] + '% \xa0At-risk to negligible',

bucket21[5] + '% \xa0Negligible snow (unchanged)'];

//print(names);

for (i = 0; i < 6; i++) {

legend.add(makeRow(palette[i], names[i]));

}

Map.add(legend);

legend = ui.Panel({

style: {

position: 'bottom-left',

padding: '8px 15px'

}

});

legendTitle = ui.Label({

value: watershedname + ' : ' + scenario2,

style: {

fontWeight: 'bold',

fontSize: '16px',

margin: '0 0 4px 0',

padding: '0'

}

});

legend.add(legendTitle);

//print(bucket);

names = [bucket2[0] + '% \xa0Safe snow',

bucket2[1] + '% \xa0At-risk snow',

bucket2[2] + '% \xa0Negligible snow'];

//print(names);

for (i = 0; i < 3; i++) {

legend.add(makeRow(palette3ars[i], names[i]));

}

legendTitle = ui.Label({

value: watershedname + ' : ' + scenario1,

style: {

fontWeight: 'bold',

fontSize: '16px',

margin: '0 0 4px 0',

padding: '0'

}

});

legend.add(legendTitle);

//print(bucket);

names = [bucket1[0] + '% \xa0Safe snow',

bucket1[1] + '% \xa0At-risk snow',

bucket1[2] + '% \xa0Negligible snow'];

//print(names);

for (i = 0; i < 3; i++) {

legend.add(makeRow(palette3ars[i], names[i]));

}

Map.add(legend);

//alert('Hit \"Apply\" to select another watershed, but first\n\n \xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0Press OK to continue!');

});

}

else {

// LEGEND for At-risk snow //

// set position of panel

var legend = ui.Panel({

style: {

position: 'bottom-left',

padding: '8px 15px'

}

});

// Create legend title

var legendTitle = ui.Label({

value: 'At-risk snow',

style: {

fontWeight: 'bold',

fontSize: '16px',

margin: '0 0 4px 0',

padding: '0'

}

});

// Add the title to the panel

legend.add(legendTitle);

// Add color and and names

for (var i = 0; i < 3; i++) {

legend.add(makeRow(palette3ars[i], names3ars[i]));

}

// add legend to map

Map.add(legend);

// LEGEND for Change in at-risk snow //

// set position of panel

legend = ui.Panel({

style: {

position: 'bottom-right',

padding: '8px 15px'

}

});

// Create legend title

//var theText = scenario2.concat(' - ');

//theText = theText.concat(scenario1);

var legendText = scenario2 + ' - ' + scenario1;

legendTitle = ui.Label({

value: legendText,

style: {

fontWeight: 'bold',

fontSize: '16px',

margin: '0 0 4px 0',

padding: '0'

}

});

// Add the title to the panel

legend.add(legendTitle);

// Add color and and names

for (i = 0; i < 6; i++) {

legend.add(makeRow(palette[i], names[i]));

}

// add legend to map

Map.add(legend);

}

Map.addLayer(skiResorts, {'color':'FF00FF'}, 'Ski Resorts');

};

nexdcp30.filters = {

scenario1YearRange : ui.Textbox('Scenario1YearRange','1970 : 1999', function(){scenario1YearsUpdated = 1; return;}),

scenario2YearRange : ui.Textbox('Scenario2YearRange','2070 : 2099', function(){scenario2YearsUpdated = 1; return;}),

applyButton: ui.Button('Apply', nexdcp30.arsApply)

};

nexdcp30.filters.panel = ui.Panel({

widgets:[

ui.Label({

value: 'At-risk Snow Calculator',

style: {fontWeight: 'bold'}

}),

ui.Panel([

ui.Label({

value:'GCM :',

style: {fontWeight: 'bold'}}),

ui.Select({

items: [

{label: 'ACCESS1-0', value: 'ACCESS1-0'},

{label: 'BNU-ESM', value: 'BNU-ESM'},

{label: 'CCSM4', value: 'CCSM4'},

{label: 'CESM1-BGC', value: 'CESM1-BGC'},

{label: 'CESM1-CAM5', value: 'CESM1-CAM5'},

{label: 'CMCC-CM', value: 'CMCC-CM'},

{label: 'CNRM-CM5', value: 'CNRM-CM5'},

{label: 'CSIRO-Mk3-6-0', value: 'CSIRO-Mk3-6-0'},

{label: 'CanESM2', value: 'CanESM2'},

{label: 'FGOALS-g2', value: 'FGOALS-g2'},

{label: 'FIO-ESM', value: 'FIO-ESM'},

{label: 'GFDL-CM3', value: 'GFDL-CM3'},

{label: 'GFDL-ESM2G', value: 'GFDL-ESM2G'},

{label: 'GFDL-ESM2M', value: 'GFDL-ESM2M'},

{label: 'GISS-E2-H-CC', value: 'GISS-E2-H-CC'},

{label: 'GISS-E2-R-CC', value: 'GISS-E2-R-CC'},

{label: 'GISS-E2-R', value: 'GISS-E2-R'},

{label: 'HadGEM2-AO', value: 'HadGEM2-AO'},

{label: 'HadGEM2-CC', value: 'HadGEM2-CC'},

{label: 'HadGEM2-ES', value: 'HadGEM2-ES'},

{label: 'IPSL-CM5A-LR', value: 'IPSL-CM5A-LR'},

{label: 'IPSL-CM5A-MR', value: 'IPSL-CM5A-MR'},

{label: 'IPSL-CM5B-LR', value: 'IPSL-CM5B-LR'},

{label: 'MIROC-ESM-CHEM', value: 'MIROC-ESM-CHEM'},

{label: 'MIROC-ESM', value: 'MIROC-ESM'},

{label: 'MIROC5', value: 'MIROC5'},

{label: 'MPI-ESM-LR', value: 'MPI-ESM-LR'},

{label: 'MPI-ESM-MR', value: 'MPI-ESM-MR'},

{label: 'MRI-CGCM3', value: 'MRI-CGCM3'},

{label: 'NorESM1-M', value: 'NorESM1-M'},

{label: 'bcc-csm1-1-m', value: 'bcc-csm1-1-m'},

{label: 'bcc-csm1-1', value: 'bcc-csm1-1'},

{label: 'inmcm4', value: 'inmcm4'}

],

placeholder: 'ACCESS1-0',

value: 'ACCESS1-0',

onChange: function(value) {

GCM = value;

GCMUpdated = 1;

}

})

], ui.Panel.Layout.flow('horizontal')),

ui.Label({

value:'Scenario1 :',

style: {fontWeight: 'bold'}}),

ui.Select({

items: [

{label: 'Historical (1950 - 2005)', value: 'historical'},

{label: 'RCP 2.6 (2006 - 2099)', value: 'rcp26'},

{label: 'RCP 4.5 (2006 - 2099)', value: 'rcp45'},

{label: 'RCP 6.0 (2006 - 2099)', value: 'rcp60'},

{label: 'RCP 8.5 (2006 - 2099)', value: 'rcp85'}

],

placeholder: 'Historical (1950 - 2005)',

onChange: function(value) {

scenario1 = value;

scenario1Updated = 1;

}

}),

ui.Label({

value:'Scenario 1 Winter Year Range : ',

style: {fontWeight: 'bold'}}), nexdcp30.filters.scenario1YearRange,

ui.Label({

value:'Scenario2 :',

style: {fontWeight: 'bold'}}),

ui.Select({

items: [

{label: 'Historical (1950 - 2005)', value: 'historical'},

{label: 'RCP 2.6 (2006 - 2099)', value: 'rcp26'},

{label: 'RCP 4.5 (2006 - 2099)', value: 'rcp45'},

{label: 'RCP 6.0 (2006 - 2099)', value: 'rcp60'},

{label: 'RCP 8.5 (2006 - 2099)', value: 'rcp85'}

],

placeholder: 'RCP 8.5 (2006 - 2099)',

onChange: function(value) {

scenario2 = value;

scenario2Updated = 1;

}

}),

ui.Label({

value:'Scenario 2 Winter Year Range : ',

style: {fontWeight: 'bold'}

}), nexdcp30.filters.scenario2YearRange,

ui.Panel([

ui.Label({

value:'HUC Watershed Level : ',

style: {fontWeight: 'bold'}

}),

ui.Select({

items: [

{label: 'None', value: 'None'},

{label: '02', value: '02'},

{label: '04', value: '04'},

{label: '06', value: '06'},

{label: '08', value: '08'},

{label: '10', value: '10'},

{label: '12', value: '12'}

],

placeholder: 'None',

onChange: function(value) {

huc = value;

}

})

], ui.Panel.Layout.flow('horizontal')),

nexdcp30.filters.applyButton

]

});

nexdcp30.ars = function(scenario, first) {

// Scenario1 years

var years;

if (first == 1) {

years = nexdcp30.filters.scenario1YearRange.getValue().replace(/\s+/g,'').split(':');

} else {

years = nexdcp30.filters.scenario2YearRange.getValue().replace(/\s+/g,'').split(':');

}

var startYear = years[0];

var endYear = years[1];

//print(startYear, endYear);

// PROCESS TIME PERIOD

var Year0 = parseInt(startYear) - 1;

var Year1 = parseInt(endYear) + 1;

var startDate = Year0 + '-12-01';

var endDate = Year1 + '-02-28';

// Filter collection by scenario and GCM

//print(scenario, GCM);

var collectionDec = ee.ImageCollection(collectionPath)

.filterDate(startDate, endDate)

.filter(ee.Filter.eq('scenario', scenario))

.filter(ee.Filter.eq('model', GCM))

.filter(ee.Filter.eq('month', 12));

var collectionJan = ee.ImageCollection(collectionPath)

.filterDate(startDate, endDate)

.filter(ee.Filter.eq('scenario', scenario))

.filter(ee.Filter.eq('model', GCM))

.filter(ee.Filter.eq('month', 1));

var collectionFeb = ee.ImageCollection(collectionPath)

.filter(ee.Filter.eq('scenario', scenario))

.filter(ee.Filter.eq('model', GCM))

.filter(ee.Filter.eq('month', 2));

//print(collectionDec, collectionJan, collectionFeb);

var meanDec = collectionDec.reduce(ee.Reducer.mean());

var meanJan = collectionJan.reduce(ee.Reducer.mean());

var meanFeb = collectionFeb.reduce(ee.Reducer.mean());

var meanWinter = (meanDec.add(meanJan).add(meanFeb)).divide(3);

var tas = (meanWinter.select('tasmin\_mean')

.add(meanWinter.select('tasmax\_mean')))

.divide(2);

var blank = ee.Image(0);

// Where tas > 273.15 K

var ARS = blank.where(tas.gt(273.15),1);

//if (first == 1) {

var ARS0 = blank.where(tas.gt(279.15),2);

ARS = ARS.add(ARS0)

//} else {

// var ARS0 = blank.where(tas.gt(279.15),1);

// ARS = ARS.add(ARS0)

//}

// Output contains 0s and 1s.

ARS = ARS.updateMask(tas);

return ARS;

};

/\*\* Creates the application interface. \*/

nexdcp30.boot = function() {

///////////////////////// DATE RANGE //////////////////////////////////

// First, create variables for the date range.

// There are many ways to do this, and I have not figured out the best way yet.

// To change the date range of analysis, simply change the date below.

// Set filter variables.

var main = ui.Panel({

widgets: [

nexdcp30.filters.panel

]

});

ui.root.insert(0, main);

alert('Please select GCM, Scenarios, Year Ranges and/or HUC Level, but first\n\n \xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0Press OK to continue!');

};

nexdcp30.boot();

// Set the default map's cursor to a "crosshair".

Map.style().set('cursor', 'crosshair');