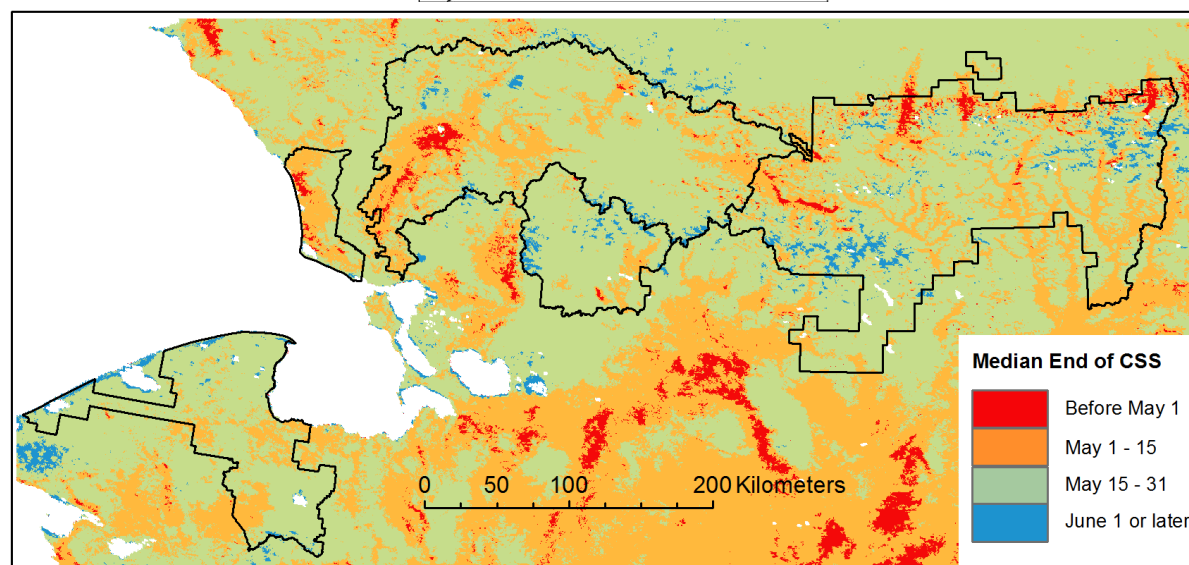
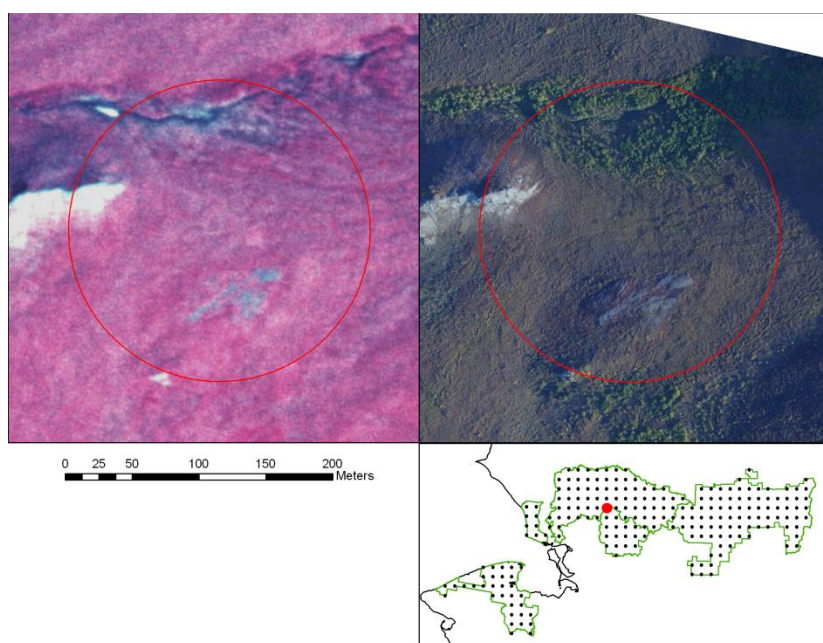




Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network

Standard Operating Procedures

Natural Resource Report NPS/ARC/NRR—2017/1381



ON THE COVER

Examples of monitoring projects for the ARCN Terrestrial Landscape Patterns and Dynamics Vital Sign. Above, Tall shrub increase on tundra. An area of mostly herbaceous and low shrub vegetation in 1979 (AHAP color-infrared photo, left) was colonized by tall shrubs by 2008 (small-format true color photo, right). The red dot on the inset map shows the location of the plot in the Noatak National Preserve, Alaska. Below, the median date of the end of the continuous snow season (CSS) in ARCN for the period 2001-2013.

Photograph Courtesy of NPS.

Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network

Standard Operating Procedures

Natural Resource Report NPS/ARC/NRR—2017/1381

David K. Swanson

National Park Service
4175 Geist Road
Fairbanks, Alaska 99709

The National Park Service, Natural Resource Stewardship and Science office in Fort Collins, Colorado, publishes a range of reports that address natural resource topics. These reports are of interest and applicability to a broad audience in the National Park Service and others in natural resource management, including scientists, conservation and environmental constituencies, and the public.

The Natural Resource Report Series is used to disseminate comprehensive information and analysis about natural resources and related topics concerning lands managed by the National Park Service. The series supports the advancement of science, informed decision-making, and the achievement of the National Park Service mission. The series also provides a forum for presenting more lengthy results that may not be accepted by publications with page limitations.

All manuscripts in the series receive the appropriate level of peer review to ensure that the information is scientifically credible, technically accurate, appropriately written for the intended audience, and designed and published in a professional manner.

This report received formal peer review by subject-matter experts who were not directly involved in the collection, analysis, or reporting of the data, and whose background and expertise put them on par technically and scientifically with the authors of the information.

Views, statements, findings, conclusions, recommendations, and data in this report do not necessarily reflect views and policies of the National Park Service, U.S. Department of the Interior. Mention of trade names or commercial products does not constitute endorsement or recommendation for use by the U.S. Government.

This report is available from the National Park Service, [Arctic Inventory and Monitoring Network](#) and the [Natural Resource Publications Management website](#). To receive this report in a format that is optimized to be accessible using screen readers for the visually or cognitively impaired, please email irma@nps.gov.

Please cite this publication as:

Swanson, D. K. 2017. Landscape patterns and dynamics monitoring protocol for the Arctic Network: Standard operating procedures. Natural Resource Report NPS/ARC/NRR—2017/1381. National Park Service, Fort Collins, Colorado.

Contents

	Page
Acknowledgments.....	ix
Introduction.....	1
A. Standard Operating Procedures for Vegetation Greenness Monitoring by Satellite	3
Standard Operating Procedure (SOP) #1: Summary of MODIS NDVI imagery	3
<i>A. Background.....</i>	<i>3</i>
<i>B. Acquire the NDVI data</i>	<i>4</i>
<i>C. Create the seasonal percentile summary rasters.</i>	<i>8</i>
<i>D. Compute the Dates of Half Greenup and Senescence.....</i>	<i>10</i>
<i>E. Test for Trends.....</i>	<i>14</i>
<i>F. Summary Statistics.....</i>	<i>15</i>
Section A Literature Cited	17
B. Standard Operating Procedures for Snow Cover Monitoring by Satellite	19
Standard Operating Procedure (SOP) # 1: Analysis of Snow Cover Metrics	19
<i>A. Background.....</i>	<i>19</i>
<i>B. Download and Clip the MODIS Snow Metrics</i>	<i>20</i>
<i>C. Create Single Band Images</i>	<i>21</i>
<i>D. Compute Multi-Year Medians.....</i>	<i>22</i>
<i>E. Compute Individual Year Deviations from the Median.....</i>	<i>23</i>
<i>F. Summarize trends for geographic regions</i>	<i>23</i>
<i>G. Regression on pixel-by-pixel basis.....</i>	<i>25</i>
Section B Literature Cited	26
C. Standard Operating Procedures for Vegetation and Snow Monitoring by Remote Automated Cameras.....	27

Contents (continued)

	Page
Standard Operating Procedure (SOP) # 1: Phenology Camera Setup and Operation	28
<i>A. Background</i>	28
<i>B. Wiring</i>	28
<i>C. Configure Laptop/Ethernet on field laptop</i>	28
<i>D. Test Camera before departure.</i>	28
<i>E. Field Setup</i>	29
<i>G. Retrieve images</i>	29
<i>H. Transfer images to network</i>	29
<i>I. Equipment</i>	29
Standard Operating Procedure (SOP) # 2: Greenness and Snow Phenology Photograph Analysis	30
<i>A. Background</i>	30
<i>B. Database Updates</i>	30
<i>C. Define the Analysis Zones</i>	33
<i>D. Derive Greenness Metrics from the Photo Data</i>	35
<i>E. Update Fog Table</i>	38
<i>F. Greenness Curve Fitting and Summary</i>	39
<i>G. Find greenness minimum due to redness.</i>	43
<i>H. Determine degree-day sum for green-up midpoint.</i>	43
<i>I. Snow Cover Analysis.</i>	44
<i>J. Analyze results for trends</i>	52
Standard Operating Procedure (SOP) # 3: Creation of time-lapse videos from times series of still photographs from phenology cameras.....	53
<i>A. Background</i>	53

Contents (continued)

	Page
<i>B. Reformatting the photographs with Python and Image Magick.....</i>	53
<i>C. Create a video using ffmpeg.....</i>	56
Section C Literature Cited	57
D. Standard Operating Procedures for Land Cover and Landform Monitoring with High-Resolution Aerial Photographs.....	58
Standard Operating Procedure (SOP) # 1: Ordering aerial photographs.....	59
<i>A Background.....</i>	59
<i>B. Specifications.....</i>	59
Standard Operating Procedure (SOP) # 2: Cataloguing and georeferencing aerial photographs.....	60
<i>A. Background.....</i>	60
<i>B. Create an index from Exif headers.....</i>	60
<i>C. Initial registration of aerial photographs</i>	61
<i>D. Exact georeferencing of aerial photographs</i>	63
Standard Operating Procedure (SOP) # 3: Change detection using high-resolution aerial photographs.....	65
<i>A. Background.....</i>	65
<i>B. Generate and view the sample circles in ArcMap.....</i>	65
<i>C. Photo-interpretation of change at the whole plot level.....</i>	66
<i>D. Land cover and ice-wedge polygons classification using 37 subplots.....</i>	69
<i>E. Stereo 3D viewing.....</i>	72
<i>F. Data Analysis and Reporting.....</i>	77
Section D Literature Cited	81
E. Standard Operating Procedures for Lake Surface Area Monitoring	83
Standard Operating Procedure (SOP) # 1: Image download and preparation	83

Contents (continued)

	Page
<i>A. Background</i>	83
<i>B. Download the Landsat images.</i>	83
<i>C. Convert the hdf file to tif using Multispec.</i>	85
Standard Operating Procedure (SOP) # 2: Water body mapping using Landsat Band 5.....	87
<i>A. Background</i>	87
<i>B. Computations</i>	87
Section E Literature Cited	91

Figures

	Page
Figure C1. Creating an analysis area mask for phenology analysis.	35
Figure C2. Example of sigmoid curve fit to greenness data.	42
Figure D1. Data entry form for aerial photograph change detection.	68
Figure D2. Data validation dialog box entries used in the aerial photo change data entry form.	69
Figure D3. Examples of ice-wedge polygon types. High-center (upper photo), flat (middle photo), and low-center (lower photo).	71
Figure D4. Arranging photos in Stereo Photo Maker.	73

Tables

	Page
Table A1. Deliverables schedule for summary of satellite greenness data	3
Table B1. Deliverables schedule for analysis of snow cover metrics	19
Table C1. Deliverables schedule for phenology cameras	27
Table D1. Deliverables schedule for systematic grid image analysis	58
Table D2. Land cover change types and mechanisms.	66
Table D3. Example cross-tabulation of ecotype change results ¹	79
Table E1. Deliverables schedule for analysis of surface water change by Landsat Imagery	83

Acknowledgments

Thanks to Matt Macander and M. Torre Jorgenson for helpful review comments. Jay Ver Hoef and Josh Schmidt provided helpful discussion about sampling design and statistical analysis, and JVH designed the trinomial analysis in Section D, SOP #3.

Introduction

The long-term monitoring vital sign “Terrestrial Landscape Patterns and Dynamics” addresses the “big-picture” of changes in ecosystems of the National Park Service (NPS) Arctic Inventory and Monitoring Network (ARCN; the five NPS units in northern Alaska). This monitoring program consists of five basic projects:

Vegetation greenness monitoring by satellite. Satellite measurements of reflected solar radiation in the red (620–670) and near-infrared (841–876 nm) bands is used to compute the Normalized Difference Vegetation Index (NDVI, an indicator of biomass). These NDVI measurements are used to track changes in the timing of the growing season and the overall greenness of the landscape across all of ARCN.

Snow cover monitoring by satellite. Satellite measurements of green (545-565 nm) and short-wavelength infrared (1628-1652 nm) light are used to track changes in the length and timing of the snow season across all of ARCN.

Vegetation and snow phenology monitoring by remote automated cameras. Cameras at five ARCN climate monitoring stations take five photographs each day through the year. These photographs are analyzed to determine the timing of the growing season and the snow season.

Land cover and landform monitoring with high-resolution aerial photographs. A systematic grid of aerial photographs taken at intervals of ten or more years to track changes in vegetation and landforms across all of ARCN.

Lake surface area monitoring by satellite. Satellite measurements of reflected short-wavelength infrared (1.55-1.75 μm) light are used to map surface water in lakes and ponds each year across all of ARCN.

Taken together these various projects provide us with broad picture of changes occurring in the dynamic landscape of Alaska’s arctic National Parks. This report contains the standard operating procedures for the Arctic Network landscape patterns and dynamics monitoring protocol.

A. Standard Operating Procedures for Vegetation Greenness Monitoring by Satellite

These Standard Operating Procedures are used to monitor vegetation greenness with MODIS satellite imagery as explained in Section 2 of the protocol narrative document (Swanson 2017).

Table A1. Deliverables schedule for summary of satellite greenness data

Identifier	Deliverable	Format	Schedule	Responsibility	Description
A1	NDVI metrics	Raster GIS file	Annually, for the preceding year	Cooperator funded by NPS; or PI	Growing-season metrics derived from the eMODIS composites by the method of Zhu et al. (2013) ¹
A2	Supplementary NDVI metrics	Raster GIS files	Annually for the preceding year, or just prior to 5-year report	PI	Additional metrics of interest derived from the eMODIS composites as outlined in SOP #1 part D
A3	Summary report	Document file (pdf)	Every 5 years (beginning 2017)	PI	Either an NPS Natural Resource Report or peer-reviewed journal article

¹These files are currently stored and served by UAF GINA

Standard Operating Procedure (SOP) #1: Summary of MODIS NDVI imagery Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

The MODIS (Moderate-resolution Imaging Spectroradiometer) satellite has been supplying multispectral imagery at a resolution of 250 to 1000 m (depending on the band) since 2000. These data are currently being used to compute weekly NDVI (normalized difference vegetation index) composites. A composite image incorporates cloud-free pixels from multiple days to provide a single image for the time period that simulates a single cloud-free scene. The MODIS composites are produced by the "eMODIS" project for all of Alaska (and the rest of the US), and are described in detail by Jenkerson and Schmidt (2009). The composites are further processed to extract growing season metrics by the GINA (the Geographic Information Network for Alaska) at the University of Alaska Fairbanks (Zhu et al. 2013). ARCN is summarizing NDVI data in order to monitor trends in

growing season characteristics. This SOP describes the procedure used by ARCN to analyze the MODIS data (Swanson 2010, 2016a). The schedule of deliverables for this project is given in Table A1.

The major steps in our analysis process are: download the data from USGS (composites) or GINA (metrics), clip out the portion of each image that occurs within ARCN, mask out the cloudy pixels, and compute summary statistics.

B. Acquire the NDVI data

The NDVI data are available for download from the USGS via their “Earth Explorer” website (<http://earthexplorer.usgs.gov>). The NDVI metrics data developed by UAF GINA are available from their website (<http://gina.alaska.edu/>) as a Web Coverage Service (WCS). Both are posted as statewide images, which are then clipped to the extent of ARCN and saved locally to make processing more efficient.

B1. Download and clip the eMODIS NDVI composites to the ARCN extent.

Start USGS Earth Explorer <http://earthexplorer.usgs.gov>. In the interactive map of the world, outline a rectangle in northwest Alaska (the location is not important, since all of Alaska comes as one scene). Under the Search Criteria tab, enter the date range (years) and search months (June-Sept). Note that if your search finds more than 100 images the result will be truncated, so you may have to limit the years or months and download several batches. Under the Data Sets tab, expand “Vegetation Monitoring” and select “eMODIS NDVI”. Hit “Results” and the Alaska-wide eMODIS 7-day composites will appear. Click on the “Add to Bulk Download” option. When all have been selected, hit “View Item Basket” then “Proceed to Checkout” and “Submit Order” and the images will be saved as a bulk download order. Install the USGS Bulk Download Application (<http://earthexplorer.usgs.gov/bulk/>). Download the images, consulting the directions supplied in http://earthexplorer.usgs.gov/resources/Bulk_tutorial_ee.pdf if needed.

The NDVI images come as one zip file per date containing the NDVI, acquisition dates (ACQ), and quality (QUAL) images in tif format. Use the following Python script to unzip, clip to the ARCN extent, mask out all pixels labeled as cloud or bad data in the QUAL image with “no data”, and save. Put all of the years of zipfiles from a particular compositing period into a subfolder named “zip” under a folder named after the period, e.g. “\155\zip” for the zipfiles for the day 155-161 composites. Create another subfolder named “clip” to hold the output of the script below, e.g. “\155\clip”. Edit the input path and output path names near the top of the script as needed. Note that in Python, path names must either have forward slashes “/” or double back slashes “\”.

```
# Script to unzip Earth Explorer downloaded eMODIS NDVI images
# clip ARCN extent, mask clouds and snow with No Data and save result to
local hard drive
#ArcGIS 10.2, Python 2.7
# -----
# User inputs below
inpath = inpath = "C:\GIS\GISdataNoBackup\MODIS\155\zip\" #path to
folder of eMODIS zipfiles
```

```

outpath = "C:\\GIS\\GISdataNoBackup\\MODIS\\155\\clip\\"
ARCNextent = "-633000, 1708000, 207000, 2105000" #clip extent in AK Albers,
x-min, y-min, x-max, y-max
# User edit section ends here-----

import zipfile, os
import arcpy
arcpy.CheckOutExtension("spatial")
from arcpy.sa import *
arcpy.env.workspace = outpath

infiles = os.listdir(inpath)

for zipfilenm in infiles:
    print zipfilenm #to track progress
    with zipfile.ZipFile(inpath + zipfilenm, 'r') as emodiszip:
        contents = emodiszip.namelist()
        emodiszip.extract(contents[4], path=outpath) #extract the ndvi image
        emodiszip.extract(contents[5], path=outpath) #extract the quality
image

        arcpy.Clip_management(contents[4], ARCNextent, "tmpndvi")
        arcpy.Clip_management(contents[5], ARCNextent, "tmpqual")
        goodNDVIRaster = Con(Raster("tmpqual") == 0, 1, Con(Raster("tmpqual") ==
4, 1)) * Raster("tmpndvi") #convert cloudy and bad pixels to no data
        outfile = outpath + zipfilenm[8:25] + "good.tif"
        goodNDVIRaster.save(outfile)

        arcpy.Delete_management("tmpndvi")
        arcpy.Delete_management("tmpqual")

        os.remove(outpath + contents[4])
        os.remove(outpath + contents[5])

```

B1.1. Note: if any of the composites are missing, problems will arise later in processing. Currently we are missing composites 218-224 and 225-231 from the year 2000, and 169-175 and 176-182 from the year 2001. These missing rasters have been filled by blank (no data) rasters of the correct size and coordinate system. You should copy and rename these blank rasters to substitute for any missing composites in the future.

B2. Download and clip the UAF GINA MODIS metrics WCS to the ARCN extent.

The MODIS-derived NDVI metrics stored at

<http://ndvi.gina.alaska.edu/metrics/emodis?>

Each year of metrics is stored in a file named “XXXX_oneyear_layer_subset_good_metrics_ver16m1_3” where “XXXX” is the year. To view the years of data available, copy the following into address into your web browser:

```
http://ndvi.gina.alaska.edu/metrics/emodis?request=getcapabilities&service=wcs&version=1.0.0
```

You will get a long xml file in code. Scroll down a short way until you see a list of filenames bounded by the tag “<Content Metadata>”. For example:

```
<name>2000_oneyear_layer_subset_good_metrics_ver16m1_3</name>
```

The WCS data is most efficiently processed in batches using the Python scripts, without adding them to an ArcGIS project.

B2.1 Create the input text file. The Python script for downloading the images uses a text file containing the list of images to process. Each line of the file must contain an NDVI metrics image name. For example:

```
2000_oneyear_layer_subset_good_metrics_ver16m1_3
2001_oneyear_layer_subset_good_metrics_ver16m1_3
2002_oneyear_layer_subset_good_metrics_ver16m1_3
```

Using the list of available metrics images obtained in the previous step, compose a list of files from the years you need and save in a file called “infile.txt”

B2.2 Clip the images to the area of interest and save locally. This step logs into the GINA NDVI metrics WCS, uses input list of coverages in “infile.txt” to download the Alaska-wide images, and then uses the “Clip” geoprocessing tool to clip out a rectangle, based on bounding coordinates. Edit the output directory and location of your input text file near the top of the script.

```
# Script to clip WCS coverages and save result to local hard drive
#ArcGIS 10.2, Python 2.7
# -----
# User inputs below
outpath = "C:\\GIS\\GISdataLocalBackup\\MODIS\\currentyr\\"
intxtfile = "C:\\GIS\\GISdataLocalBackup\\MODIS\\currentyr\\infile.txt"

WCS = "http://ndvi.gina.alaska.edu/metrics?" #URL of the WCS
ARCNextent = "-633000, 1708000, 207000, 2105000" #clip extent in AK Albers,
x-min, y-min, x-max, y-max
# User edit section ends here-----

# Import system modules
import arcpy

infilelist = open(intxtfile, 'r')
for linex in infilelist:
```

```

wcsimage = linex[0:-1] #delete the newline character
inputWCScoverage = WCS + "coverage=" + wcsimage
print inputWCScoverage #to track progress
outraster = outpath + wcsimage + ".tif"

# Make WCS Layer. The clip geoprocess will not work directly on a WCS
coverage
arcpy.MakeWCSLayer_management(inputWCScoverage, "templayer", ARCNextent,
"")

# Clip and delete the temporary layer to make room for the next one in
the list
arcpy.Clip_management("templayer", ARCNextent, outraster)
arcpy.Delete_management("templayer")

infilelist.close()

```

Occasionally this script will fail before it finishes due to problems with the internet connection. If this happens, you will need to determine how far it progressed before the failure, edit the input file (infile.txt) to remove the completed filenames, and restart the program.

If you need to work with the WCS data in ArcGIS, open an ArcMap project and zoom into a very small area (such that the scale is something like 1:100) before proceeding. Otherwise there will be a very long delay when you add the MODIS images and ArcMap attempts to draw them. Add the MODIS imagery to the project. Choose add data in ArcMap, in the “Look in” window choose GIS Servers, select “Add WCS server” (or choose the proper WCS server if you’ve added it previously: “WCS on ...”). In the “Add WCS Server” dialog box enter the URL for the server, which in our case is <http://ndvi.gina.alaska.edu/metrics/>. After delay you will see a list of images named as described above. Select the desired image(s) and hit “Add”. Expect a long delay as the images are added to your project. The display may be black because display settings are not made automatically for the WCS.

B2.3 Compute the long-term medians. Later you will need long-term medians from several of the NDVI metrics. These can be computed using ArcGIS or the R raster package. Use these steps to do it in R. Start R and load the "raster" and "rgdal" packages. This code was run with R version 3.0. Set the working directory to the location of the files you just clipped, and create a list called "infile" by matching the filename pattern as follows:

```

setwd("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\metrics\\")
infile <- dir(getwd(), pattern="_ver16m1_3.tif$")

```

Next create a stack of one band from these files. The ones we need are band 2 (NDVI at onset of green season), band 4 (NDVI at end of green season), and band 7 (maximum NDVI). This example stacks band 2. Also, "no data" in these rasters has a value of zero, which we prefer to replace with a value of "NA".

```

s2 <- stack(infile, bands=2)

```

```
s2[s2==0]<-NA
```

Calculate the median using the "calc" function, and save as an integer raster to save space, after multiplying by 1000 to preserve the details.

```
med2 <- calc(s2, function(x){median(x, na.rm=TRUE)})  
writeRaster(1000*med2, filename="b2median.tif", format="GTiff",  
datatype="INT2S")
```

Repeat for the other bands.

C. Create the seasonal percentile summary rasters.

In this step, we create rasters that summarizes greenness of the green-up season, the peak greenness, and the fall senescence season for each year.

C1. Spring Greenup NDVI.

C1.1. First we compute the percentile score of each year in the June composites (days 155-161, 162-168, 169-175, and 176-182). Be sure that the clipped composites are in subfolders named "\155\clip", "162\clip", "169\clip", and "176\clip".

Start R, install the "raster" and "rgdal", packages if you haven't already, and load the packages:

```
library(raster)  
library(rgdal)
```

Set the working directory to one of the subdirectories with clipped June rasters:

```
setwd("C:\\GIS\\GISdataNoBackup\\MODIS\\169\\clip\\")
```

Create an image stack from all the rasters in that folder:

```
infiles <- dir(getwd(), pattern="good.tif$")  
s <- stack(infiles)
```

C1.2. Compute the percentile score of greenness in each image stack. With the image stack in memory as "s", run these commands to compute the rank of each pixel relative to all of the years:

```
pt <- calc(s, function(x){rank(x, na.last="keep")/sum(!is.na(x))})  
writeRaster(pt, filename="pt.tif", format="GTiff", bylayer=TRUE)
```

Repeat for all 4 spring compositing periods.

C1.3. Now merge the percentiles of all 4 spring compositing periods into one by computing their median percentile. First, load the four spring percentile rasters from the previous step into memory

```
setwd("C:\\GIS\\GISdataNoBackup\\MODIS\\")  
pt155 <- stack(paste0(getwd(), "/155/clip/pt.tif"))  
pt162 <- stack(paste0(getwd(), "/162/clip/pt.tif"))  
pt169 <- stack(paste0(getwd(), "/169/clip/pt.tif"))
```

```
pt176 <- stack(paste0(getwd(), "/176/clip/pt.tif"))
```

Now run this loop:

```
for (i in 1:16){
s <- stack(pt155[[i]], pt162[[i]], pt169[[i]], pt176[[i]]) #spring
#s <- stack(pt225[[i]], pt232[[i]], pt239[[i]], pt246[[i]]) #fall
md <- calc(s, function(x){median(x, na.rm=TRUE)}) #compute the median
outfile <- paste0("md_jun",1999+i,".tif")
#outfile <- paste0("md_fal",1999+i,".tif")
writeRaster(md, filename=outfile, format="GTiff")}
```

This loop runs through 16 years in each stack (edit this number in the future; remember that all the percentile rasters must have the same number of matching years, see step B1.1 above). A stack is created that contains all four June rasters for a given year *i*, and computed the median using the "calc" function of the raster package. The result is saved as a tif file named "md_junYYYY.tif" where YYYY is the year. The two lines blocked out by comments above are alternate commands to use for the fall summary, which is essentially the same.

Move the output files into a subfolder called "june_med".

C2. Fall senescence NDVI.

Run steps C1.1 and C1.2 above for the four fall senescence compositing periods: 225-231, 232-238, 239-245, and 246-252. Repeat step C1.3 but with the fall composite percentiles loaded into memory:

```
pt225 <- stack(paste0(getwd(), "/225/clip/pt.tif"))
pt232 <- stack(paste0(getwd(), "/232/clip/pt.tif"))
pt239 <- stack(paste0(getwd(), "/239/clip/pt.tif"))
pt246 <- stack(paste0(getwd(), "/246/clip/pt.tif"))
```

And run the loop with the comment signs changed to expose "fall" variants of the "s" and "outfile" lines:

```
for (i in 1:16){
#s <- stack(pt155[[i]], pt162[[i]], pt169[[i]], pt176[[i]]) #spring
s <- stack(pt225[[i]], pt232[[i]], pt239[[i]], pt246[[i]]) #fall
md <- calc(s, function(x){median(x, na.rm=TRUE)}) #compute the median
#outfile <- paste0("md_jun",1999+i,".tif")
outfile <- paste0("md_fal",1999+i,".tif")
writeRaster(md, filename=outfile, format="GTiff")}
```

Move the result into a subdirectory called "fall_med".

C3. Summer maximum NDVI.

Compute the maximum NDVI from the 6 summer composites with the following R code:

```

periods <- c("190-196", "197-203", "204-210", "211-217", "218-224", "225-
231")
max_yr <- c()      #vector to hold max value raster for each year
for (yr in 2000:2015){ #loop through all years
  rsum <- c()      #vector hold rasters of different composite periods in one year
  for (per in periods){ #loop to assemble summer rasters from a single year
    into vector of rasters rsum
  }
  path = paste0("C:\\GIS\\GISdataNoBackup\\MODIS\\", substr(per,1,3),
  "\\clip\\")
  filenm = paste0("NDVI.", yr, ".", per, "good.tif")
  if (file.exists(paste0(path, filenm))){ #some composite files are missing
    rsum <- c(rsum, raster(paste0(path, filenm)))}
  }
  max_yr <- c(max_yr, max(stack(rsum), na.rm=TRUE))}

```

This code contains two loops, the outer one iterates through the available years, the inner one through the six compositing periods. The code assembles raster file names that match the pattern "NDVI.2000.197-203good.tif" ("NDVI", years, compositing period, and "good" from the Python clip routine above), and loads them using the "raster()" command, makes a stack from all six in a single year, and finds the maximum. You must edit the path name in the line that starts with "path" to match yours.

The result is a raster stack "max_yr" with one layer per year. To write these to files, one per year, use:

```

writeRaster(max_yr[[1]], filename="annmax00.tif", format="GTiff")
writeRaster(max_yr[[2]], filename="annmax01.tif", format="GTiff")
writeRaster(max_yr[[3]], filename="annmax02.tif", format="GTiff")

```

Continue through all the years (layers) of data. The output will be a raster for each year; move them into a subdirectory called "max_local".

You will want to find the percentile score of each year of maximum NDVI data too. Point R to that working directory:

```

setwd("C:\\GIS\\GISdataNoBackup\\MODIS\\max_local\\")

```

Create an image stack from all the rasters there:

```

infiles <- dir(getwd(), pattern=".tif$")
s <- stack(infiles)

```

Compute the percentile score of greenness in the images in the stack.

```

pt <- calc(s, function(x){rank(x, na.last="keep")/sum(!is.na(x))})
writeRaster(pt, filename="pt.tif", format="GTiff", bylayer=TRUE)

```

D. Compute the Dates of Half Greenup and Senescence

Load the "raster" and "rgdal" packages if you haven't already.

```
library(raster)
library(rgdal)
```

First we need to make a slight modification of the eMODIS date of acquisition rasters *.dacq.tif. These store a 5-digit number that contains the date plus the pass number on that date; more convenient for us is the date alone. From the download step above the *.dacq.tif files should be in subdirectories named after the first 3 digits of the compositing period and then "\clip\". The files are named "NDVI.YYYY.XXX-XXX.dacq.tif" where YYYY is the year and XXX-XXX is the compositing period. First create a list of the subdirectories that need processing:

```
periods <- c("099", "106", "113", "120", "127", "134", "141", "148", "155",
"162", "169", "176", "183", "190", "197", "204", "211", "218", "225", "232",
"239", "246", "253", "260", "267", "274", "281", "288", "295", "302")
```

Then run this loop through these periods. Modify the "path =" line near the top to match the location of the files on your machine. The script will find all the *.dacq.tif files in those subdirectories, pull out the date information, and save into a new raster.

```
for (per in periods){
  path = paste0("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\", per,
"\clip\\")
  setwd(path)

  infiles <- dir(getwd(), pattern="dacq.tif$")
  for (infile in infiles) {
    r1 <- raster(infile)
    r2 <- trunc(r1/100) #date is actually date+pass so need to trim off
the two right chars
    outfile <- infile
    substr(outfile, 19, 22) <- "ate" #change 'dacq' to 'date' in file
name
    writeRaster(r2, filename=outfile , format="GTiff", datatype="INT2S")
  }
}
```

Now there will be rasters named "NDVI.YYYY.XXX-XXX.date.tif" in all the folders. In the future this operation could be folded into main script below if desired.

Now compute the half green-up dates. First set the working directory to the location where the output files will go.

```
setwd("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\spr\\")
```

Load the long-term median values for the metrics NDVI at the start of the green season (band 2) and the maximum NDVI (band 7) created in step B2.3 above. Compute the mean, which is the midpoint greenness value to be located in the time series for each year.

```
#spring half greenup value
```

```

b2med <-
raster("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\metrics\\b2median.tif")
#start of season NDVI
b7med <-
raster("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\metrics\\b7median.tif")
#max NDVI
half <- 10*(b2med+b7med)/2 #long-term value at half greenup, adjusted by 10x
to match the eMODIS scale

```

To locate the half greenup we need composites from late winter through mid summer. These are stored as a list to that corresponds to the folders where the clipped eMODIS composites are:

```

#spring eMODIS periods
periods <- c("099-105", "106-112", "113-119", "120-126", "127-133", "134-
140", "141-147", "148-154", "155-161", "162-168", "169-175", "176-182", "183-
189", "190-196", "197-203", "204-210")

```

The following script has an outer loop through years 2000 to 2015; modify as needed. The first inner loop locates all of the rasters from the periods listed above for each year and reads them into R. The "path =" line must be modified to locate your setup. It is currently written to search in root directory "C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\", subfolders named after the first day of the compositing period (i.e. the first three characters, e.g. "106" from period "106-112"), subfolder "clip". It searches for file names of the pattern "NDVI.YYYY.XXX-XXX.good.tif" and "NDVI.YYYY.XXX-XXX.date.tif" where YYYY is the year and XXX-XXX is the compositing period as listed above. Next the script creates a function called "f1" that will interpolate the date when the half-greenup value is reached given the NDVI values and dates of the composite images that bracket it. The second inner loop scans through the composites in a year and searches for the pair that it can use to interpolate the half green-up date. The NDVI data has local minima caused by clouds that must be ignored. This is accomplished by scanning the composites for each year in order and storing the maximum NDVI value to date; if a value lower than the running maximum is encountered later in the spring it is assumed to be cloud-related and ignored. When composites are located with NDVI values that bracket the long-term half greenup value, the date of the half-greenup is interpolated using the "f1" function and saved to a temporary raster called "spr". Note that the function "overlay" is the raster package's function for doing these complex raster calculations. The results for each year are saved into the working directory under file names "sprXXXX.tif", where XXXX is the year.

```

for (yr in 2000:2015){ #loop through years

list_ndvi <- c() #vector hold rasters of different composite periods in one
year
list_date <- c() #vector hold rasters of different composite periods in one
year

for (per in periods){ #loop to assemble summer rasters from a single year
into vector of rasters

```

```

path = paste0("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\", substr(per,1,3),
"\clip\\")
file_ndvi <- paste0("NDVI.", yr, ".", per, "good.tif")
file_date <- paste0("NDVI.", yr, ".", per, "date.tif")
if (file.exists(paste0(path, file_ndvi))) { #some composite files are missing
list_ndvi <- c(list_ndvi, raster(paste0(path, file_ndvi)))
list_date <- c(list_date, raster(paste0(path, file_date)))}
}
sn <- stack(list_ndvi, na.rm=TRUE)
sd <- stack(list_date, na.rm=TRUE)
crs(half) <- crs(sn) #used different ellipsoids WGS84 and GRS80, v. similar

interp <- function(x,y1,y2,x1,x2){round(y1+(x-x1)*(y2-y1)/(x2-x1))} #linear
interpolation, rounded to integer
#if test value is between two existing values, then interpolate; note:for
some reason output is 0 not NA for missing input data
f1<-function(x,y1,y2,x1,x2){ifelse (x1<x, ifelse (x<=x2,
interp(x,y1,y2,x1,x2), NA), NA)}

#initialize maxes to first raster, set up output stack
maxn <- sn[[1]]
maxd <- sd[[1]]
spr <- setValues(sd[[1]],NA)

for (i in 2:dim(sn)[3]){ #iterate through stack, interpolate date where
appropriate
tmp <- overlay(half,maxd,sd[[i]],maxn,sn[[i]], fun=f1) #pixels that were
successfully interpolated go here
spr <- sum(spr,tmp, na.rm=TRUE) #collapse results into one layer using
sum; only one value should be present
maxn <- max(maxn, sn[[i]], na.rm = TRUE) #for next iteration, left value
in interpolation is running max
maxd <- overlay(maxn,sn[[i]],maxd,sd[[i]],
fun=function(x1,x2,y1,y2){ifelse(is.na(x2),y1, ifelse(x1==x2,y2,y1))})
}

spr[spr==0] <- NA #fix the zeros, they are actually NA
writeRaster(spr, filename=paste0("spr", yr, ".tif"), format="GTiff",
datatype="INT2S")
}

```

To process the fall composites for half-senescence, replace the initial "#spring half greenup values" and "#spring eMODIS periods" with fall equivalents, including a fall output directory:

```

#fall processing
setwd("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\fal\\")
b4med <-
raster("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\metrics\\b4median.tif")
#end of season NDVI

```

```

b7med <-
raster("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\metrics\\b7median.tif")
#max NDVI
half <- 10*(b4med+b7med)/2 #long-term value at half greenup, adjusted by 10x
to match the eMODIS scale

#fall eMODIS periods; order backwards so local minima are screened properly
periods <- c("302-308", "295-301", "288-294", "281-287", "274-280", "267-
273", "260-266", "253-259", "246-252", "239-245", "232-238", "225-231", "218-
224", "211-217", "204-210", "197-203", "190-196")

```

Note that the fall eMODIS periods are listed backwards so that they are scanned for the undesired cloud minima by storing the maximum NDVI moving from winter back toward summer. Everything else is the same except for the final write statement that names the outfiles "fallXXXX.tif"

```

writeRaster(spr, filename=paste0("fal", yr, ".tif"), format="GTiff",
datatype="INT2S")

```

E. Test for Trends

Run the Mann_Kendall test of monotonic trend for the spring half-greenup dates, spring greenup median percentiles, the fall half-senscence dates, fall senescence median percentiles, and the maximum NDVI. Start R and load the "raster" and "rgdal" packages as before. Install the "rkt" package and set the working directory to the location of the data desired, in this example, the spring half-greenup date rasters.

```

library(rkt)
setwd("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\spr\\")

```

Now run the following commands to create an image stack:

```

infiles <- dir(getwd(), pattern="spr[0-9][0-9][0-9][0-9].tif$")
s <- stack(infiles)

```

The Mann-Kendall/Theil-Sen test requires at least 4 years of data and will end in error at a pixels with less, so we have to locate the pixels with less than the minimum number of years, assign dummy data to them (all zeros) so they don't cause errors, and later create a mask to remove the dummy results from the output. Unfortunately, the R raster package "mask" function has memory limits that do not allow us to run our entire study area at once, so we must crop the study area into 3 pieces to run the analysis. Raster "sc" is the cropped piece, "n" is the count of years of data, "msk" is the mask that gives a no data "NA" value to pixels with 5 or less years of data, "msk" is the cropped stack with pixels lacking enough data given all zeros, "k" is the results of the Mann-Kendall test, and "kmsk" is these results with the nonsense pixels assigned "NA". This result is called "kmskA" to keep it separate from the other two study area pieces to follow ("kmskB and kmskC").

```

sc <- crop(s, extent(s, 1, 1589, 1, 1120))
n <- calc(sc, function(x){sum(!is.na(x))})
msk <- reclassify(n, c(-1,5,NA,5,16,1))

```

```
smsk <- mask(sc, msk, updatevalue=0, updateNA=TRUE)
k <- calc(smsk, function(x){unlist(rkt(1:16,x)[c(1:4,12)]))}) #keep p-value,
S, slope, var, tau
kmsk <- mask(k, msk) #mask out nonsense values returned by rkt for no-data
areas that were coded as 0
kmskA<-kmsk
```

Repeat these commands for the other two pieces of the study area by replacing the first and last lines with:

```
sc <- crop(s, extent(s, 1, 1589, 1121, 2240))
kmskB<-kmsk
```

and then:

```
sc <- crop(s, extent(s, 1, 1589, 2241, 3361))
kmskC<-kmsk
```

Merge the 3 pieces back together and save the output file:

```
kmskAll<-merge(kmskA, kmskB, kmskC)
writeRaster(kmskAll, filename="mk.tif", format="GTiff", bylayer=TRUE)
```

The write raster command creates 5 output files, one for each layer in the output raster "kmskAll". They are Kendall's p-value, Kendall's S, the Theil-Sen slope, Kendall's variance, and Kendall's tau..

Repeat for the summer maximum and fall senescence data.

F. Summary Statistics

F1. View the Trend analysis.

When the processing is complete, view the Mann-Kendall output. Most useful are the probability value and the slope. Display and compare the two rasters to locate the places where trends are up or down and significant. Display the probability raster with “Symbology” set to “Classified” and 4 classes with breaks of 0.05, 0.1, 0.2. For publication purposes it may be useful to apply median smoothing (3 X 3 or 5 X 5 window) to the output rasters. Keep in mind that with a large number of trend analyses, some will be statistically significant even if overall trends are very weak. If the proportion of significant regressions is low compared to the significance criterion (e.g., less than 5% of the regressions were significant at the 0.05 level), and the “significant” pixels are scattered thinly around the study area, they probably represent noise rather than real trends.

F2. Summarize annual variation for geographic regions

The pixel-by-pixel yearly data (the seasonal percentile scores, dates of green-up or senescence, or maximum NDVI) for the various metrics can be summarized for geographic regions to determine which years had early greenup, high maximum NDVI, or late senescence. Load the "raster" and "rgdal" libraries, and the raster containing the zones of interest. Set the working directory to where the yearly rasters are located on your system.

```
library(raster)
```

```
library(rgdal)
zone_raster <- raster("C:\\GIS\\GISdata\\Sections\\NDVIZones250.tif")
setwd("C:\\GIS\\GISdataLocalBackup\\MODIS_NDVI\\spr\\")
```

Next create a stack of all the years of data. The easiest way to do this is to use the "dir" command to find all the file names that match a pattern and then create a stack from them:

```
infiles <- dir(getwd(), pattern="spr[0-9][0-9][0-9][0-9].tif$")
r<-stack(infiles)
```

We will use the R "quantile" function to find the median and other percentile values of each zone by year. This line defines which percentiles we want to view:

```
pctiles <- c(0.1, 0.33, 0.5, 0.67, 0.9)
```

The next few lines determine the number of zones and years for the loop that follows, and sets up the output results table:

```
zones <- unique(zone_raster)
zn_cnt <- length(zones)
yr_cnt <- dim(r)[3]
out <- matrix(ncol=length(pctiles)+2, nrow=(zn_cnt*yr_cnt))
colnames(out) <- c("yr", "zone", pctiles)
```

This loop runs the quantile function for the zones and writes them to a temporary table called "out":

```
for (j in 1:yr_cnt){
  for (i in 1:zn_cnt){
    out[(j-1)*zn_cnt+i,]<- c(j, zones[i],
quantile(r[[j]][zone_raster==zones[i]], pctiles, na.rm=TRUE))
  }
}
```

Further plotting can be done in R, or the table exported and plots made elsewhere. The export script is:

```
write.table(out, file = "spr_zonal_stats.txt", sep = "\t")
```

Note that in this output table the "year" column is just a number from 1 to the count of available years. To plot these data, reload the zonal statistics output if it isn't still in memory from the previous step:

```
infile <- "spr_zonal_stats.txt"
data <- read.table(infile, header=TRUE, sep="\t", na.strings="M", dec=".",
strip.white=TRUE)
```

To shorten the plot commands, read the various columns into variables x, y, and z:

```
x<-data[,1]+1999 #year, adjusted to convert to real year number
y<-data[,5] #median value for metric
z<-data[,2] #zones
```

So to plot the data for zones 1 and 2:

```
plot(x[z==1],y[z==1], lty=1, type="l", xlab = "Year", ylab = "Median",  
ylim=c(min(data[,5]), max(data[,5])))  
lines(x[z==2],y[z==2], lty=2)
```

To output the plot as a pdf for printing, place the "plot" and "lines" commands above between these two lines:

```
pdf(file="plot.pdf", paper = "special", bg = "white", width=5, height=5)  
dev.off()
```

The same zonal statistics operation can be performed using ArcGIS, the Spatial Analyst tool “Zonal Statistics as Table. Run in batch mode and request *.dbf file output; name the output files after the year. This method produces numerous output *.dbf files that must be merged before plots can be made, a laborious step that makes the R process preferable.

Section A Literature Cited

- Jenkerson, C. B., and G. L. Schmidt. 2009. “eMODIS Alaska.” In *American Society for Photogrammetry & Remote Sensing Annual Conference*. American Society for Photogrammetry and Remote Sensing.
<http://info.asprs.org/publications/proceedings/baltimore09/0043.pdf>.
- Swanson, D. K. 2010. “Satellite Greenness Data Summary for the Arctic Inventory and Monitoring Network, 1990-2009.” Natural Resource Data Series NPS/ARC/NRDS—2010/124. Fort Collins, Colorado: National Park Service.
<https://irma.nps.gov/DataStore/Reference/Profile/2166935>.
- Swanson, D. K. 2016. “Satellite Greenness Data Summary for the Arctic Inventory and Monitoring Network, 2000-2015.” Natural Resource Data Series NPS/ARC/NRDS—2016/XXX. Fort Collins, Colorado: National Park Service.
<https://irma.nps.gov/DataStore/Reference/Profile/>.
- Swanson, D. K. 2017. “Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network: Narrative.” Natural Resource Report NPS/ARC/NRR-2017/1380. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/DataStore/Reference/Profile/2238342>.
- Zhu, J., A. E. Miller, C. Lindsay, D. Broderson, T. Heinrichs, and P. Martyn. 2013. “MODIS NDVI Products and Metrics User Manual. Version 1.0.” Geographic Information Network for Alaska, University of Alaska. <http://www.gina.alaska.edu/projects/modis-derived-ndvi-metrics>.

B. Standard Operating Procedures for Snow Cover Monitoring by Satellite

These Standard Operating Procedures are used to monitor the timing and duration of snow cover derived from MODIS satellite imagery, as described in Section 3 of the protocol narrative document (Swanson 2017).

Table B1. Deliverables schedule for analysis of snow cover metrics

Ident- ifier	Deliverable	Format	Schedule	Respon- sibility	Description
B1	MODIS snow metrics	Raster GIS file	Annually, for the preceding year	Cooperator funded by NPS; or PI	Growing-season metrics derived from the daily by the method of Zhu and Lindsay (2013) and Lindsay (2015) ¹
B2	Summary report	Document file (pdf)	Every 5 years (beginning 2017)	PI	Either an NPS Natural Resource Report or peer-reviewed journal article

¹These files are currently stored and served by UAF GINA

Standard Operating Procedure (SOP) # 1: Analysis of Snow Cover Metrics Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

This SOP describes how to download the MODIS Snow metrics images created for NPS by UAF GINA (Lindsay et al. 2015, Zhu and Lindsay 2013), clip out ARCN from the statewide image, and compute summary statistics from the images. There is one snow metrics images for each year. Each annual image contains multiple bands, and each band stores some information of interest that GINA has extracted from that year's set of daily snow classification images (which in turn were produced by the National Snow and Ice Data center). For example, band 5 contains the date of the end of the continuous snow season, stored as an integer number. In the first calendar year of a given winter these are ordinal dates from 1 to 365. Dates in the second calendar year of the winter (after 1 Jan) continue counting from 365, e.g. a value of 465 would be April 10, day 100. These summary procedures were developed for Swanson (2014). To troubleshoot the R scripts below, see Hijmans et al. (2015).

B. Download and Clip the MODIS Snow Metrics

B1. Clip from Web Coverage Service (WCS).

Create ArcMap Project. Zoom very close into a small area to speed up the display. Click on add data. Choose “GIS Servers”, “Add WCS Server”. Copy the URL where the snow metrics images reside (<http://snow.proto.gina.alaska.edu/metrics?>) into the dialog box (note, the “?” is part of the URL) and hit Open. The Add Data dialog box appears with the WCS on it. Click “Add” to see the individual years available. Add the desired years to the ArcMap project. Clip out a rectangle of data that covers your area of interest; for all of ARCN this is -620200, 1719000, 200300, 2100000 (clip extent in Alaska Albers project, x-min, y-min, x-max, y-max). The following Python script automates this task for a list of files:

```
# -----
# MODIS_SnowClipBatch.py
# Script to clip WCS coverages and save result to local hard drive
# ArcGIS 10.2, Python 2.7
# -----
# User inputs below
WCS = "http://snow.proto.gina.alaska.edu/metrics?" #URL of the WCS
outpath = "C:\\GIS\\GISdataLocalBackup\\MODISSnow\\"
ARCNextent = "-620000, 1720000, 200000, 2100000" #clip extent in AK Albers,
x-min, y-min, x-max, y-max
intxtfile = "C:\\GIS\\GISdataLocalBackup\\MODISSnow\\infile.txt"
# User edit section ends here-----

# Import system modules
import arcpy
from arcpy import env
env.workspace = outpath

infilelist = open(intxtfile, 'r')
for line in infilelist:
    wcsimage = line[0:-1] #delete the newline character
    inputWCScoverage = WCS + "coverage=" + wcsimage
    print inputWCScoverage #to track progress
    outraster = outpath + wcsimage + ".tif"

    # Make WCS Layer. The clip geoprocess will not work directly on a WCS
    coverage
    arcpy.MakeWCSLayer_management(inputWCScoverage, "templayer", ARCNextent,
    "")

    # Clip and delete the temporary layer to make room for the next one in
    the list
    arcpy.Clip_management("templayer", ARCNextent, outraster)
    arcpy.Delete_management("templayer")

infilelist.close()
```

This script loads a file called “infile.txt” (see the end of the user inputs section; modify the path as needed) that contains a list of the filenames, for example:

```
2001_snowyear_metrics_v7
2002_snowyear_metrics_v7
2003_snowyear_metrics_v7
2004_snowyear_metrics_v7
    etc.
```

B2. Alternate method: download whole state and clip.

You may also download the entire (statewide) image for each year to your computer using ftp. Check the header (*.hdr) file and see what format the accompanying data file is. You may need to add a file extension (e.g. *.bil) to the data file for ArcGIS to recognize it. Add the image to an ArcMap project and clip it to the ARCN extent, using the “Clip” geoprocessing tool (under Data Management Tools, Raster, Raster Processing) and the clip extent given above; then delete the statewide image(s).

C. Create Single Band Images

This step can be omitted if further processing is done in R. ArcMap geoprocessing is easier to automate when the data are in single-band images.

C1. Band definitions.

The MODIS snow metrics images contains 12 bands, defined as follows:

1. first_snow_day, first day of the full snow season (FSS start day)
2. last_snow_day, last day of the full snow season (FSS end day)
3. fss_range, last_snow_day-first_snow_day +1
4. longest_css_first_day, first day of the longest continuous snow season segment (CSS start day)
5. longest_css_last_day, last day of the longest continuous snow season segment (CSS end day)
6. longest_css_day_range, longest_css_last_day-longest_css_first_day +1
7. snow_days, the number of snow days
8. no_snow_days, the number of no snow days
9. css_segment_num, the number of continuous snow season segments
10. mflag, pixel type (ocean, land, or lake/inland water) and type of snow (no snow, broken snow, or continuous snow)
11. cloud_days, number of cloud days
12. tot_css_days, total number of all days within continuous snow season segments

C2. Write the individual bands to separate single-band images.

There is currently no tool specifically for this purpose in ArcMap, but you can add each individual band to an ArcMap project and then use the “Copy Raster” tool (under Data Management Tools, Raster, Raster Dataset) to copy the band as into its own image file. Add the band identifier to the file name and save as a tif file, e.g., *b4.tif. Put all the years for a single band into one folder.

D. Compute Multi-Year Medians

D1. R method. Compute the multi-year median value for each pixel across all years for band 1 (first snow day), band 2 (last snow day), band 4 (first day of longest continuous snow season), and band 5 (last day of continuous snow season). Start R, load the "raster" and "rgdal" libraries, and set the working directory to where the downloaded MODIS metrics images reside (edit the "setwd" path below as needed; note the double back slashes).

```
library(raster)
library(rgdal)
setwd("C:\\GIS\\GISdataLocalBackup\\MODISsnow\\")
```

Next use the "dir" command to create a list of all the snow metrics files in the working directory, and create image stacks of each band of interest (bands 1, 2, 4, and 5):

```
infiles <- dir(getwd(), pattern="_snowyear_metrics_v7.tif$")
s1 <- stack(infiles, bands=1)
s2 <- stack(infiles, bands=2)
s4 <- stack(infiles, bands=4)
s5 <- stack(infiles, bands=5)
```

Now compute the medians using the raster "calc" function:

```
s1_median <- calc(s1, function(x){median(x, na.rm=TRUE)})
s2_median <- calc(s2, function(x){median(x, na.rm=TRUE)})
s4_median <- calc(s4, function(x){median(x, na.rm=TRUE)})
s5_median <- calc(s5, function(x){median(x, na.rm=TRUE)})
```

Save the results to files using the "writeRaster" command. Note that for bands 2 and 5 we subtract 365 from the result because the spring snow dates are counted as cumulative through the snow season from the previous year, e.g., a value of 465 for band 2 would be April 10, day 100. The results are stored as integer rasters (INT2U) to save space.

```
writeRaster(s1_median, filename="b1med.tif", format="GTiff",
datatype="INT2U")
writeRaster(s2_median-365, filename="b2med.tif", format="GTiff",
datatype="INT2U")
writeRaster(s4_median, filename="b4med.tif", format="GTiff",
datatype="INT2U")
writeRaster(s5_median-365, filename="b5med.tif", format="GTiff",
datatype="INT2U")
```

D2. ArcGIS method.

This is an alternative method using ArcGIS. Use the "Cell Statistics" geoprocessing tool (under Spatial Analyst Tools, Local). Navigate to the folder where all the years of data for a single band reside (from step C2 above) and choose all the images. Select MEDIAN as the Overlay statistic. Save the output as a ".tif" file with the band number, e.g. "b4med.tif"

E. Compute Individual Year Deviations from the Median

The difference between each individual year and the long-term median is a useful statistic because large areas tend to be early or late by a similar amount in any given year.

E1. R method. With the s1, s1, s4, and s5 raster stacks and the s1_median, s2_median, s4_median, and s5_median rasters in memory from step D1 above, compute the differences and save the output

```
s1dif <- s1-s1_median
writeRaster(s1dif, filename="b1dif.tif", format="GTiff", datatype="INT2S")
s2dif <- s2-s2_median
writeRaster(s2dif, filename="b2dif.tif", format="GTiff", datatype="INT2S")
s4dif <- s4-s4_median
writeRaster(s4dif, filename="b4dif.tif", format="GTiff", datatype="INT2S")
s5dif <- s5-s5_median
writeRaster(s5dif, filename="b5dif.tif", format="GTiff", datatype="INT2S")
```

E2. ArcGIS method.

The difference is computed using the Raster Calculator geoprocessing tool (under Spatial Analyst Tools, Map Algebra, Raster Calculator). Construct your map algebra expression as in this example for band 5, year 2005:

```
"b5_2005.tif" - "b5Median.tif"
```

Save the output as a tif with the year, band, and "dev" in the file name. Put all the years' deviations for a single band into one folder. Run for bands 1, 2, 4, and 5.

Keep in mind when interpreting the results that detection of fall establishment of snow cover is a problem in some years (Swanson 2014). Very late dates for the establishment of snow cover (e.g. December dates for bands 1 and 4) may be erroneous and result, for example, from late October snow establishment coupled with cloudy weather that prevents its detection before winter twilight arrives in November. Patches of erroneous data usually have shapes that do not conform to natural features such as lowlands.

F. Summarize trends for geographic regions

The metrics or their deviations from the median obtained in step E above can be summarized for geographic regions, such as ecological sections, NPS units, or wildlife study areas. This is accomplished using R "quantile" function in the "raster" package or the Spatial Analyst tool "Zonal Statistics as Table".

F1. Method using R.

Set the working directory to where the data are, use "dir" to create a list of files to load, and make them into a raster stack. This example shows only band 5 (end of continuous snow season).

```
setwd("C:\\GIS\\GISdataLocalBackup\\MODISSnow\\")
infiles <- dir(getwd(), pattern="[0-9][0-9][0-9][0-9]_snowyear_metrics_v7.tif$")
st <- stack(infiles, bands=5) #band 5 is end of CSS
```

Load the shapefile for the zone of interest and convert it to a raster labeled "1" (in the zone) or "NA" (out of the zone).

```
sh <-  
shapefile("C:\\GIS\\GISdata\\SheepSurveyAreas\\Itkillik_GAAR_SurveyArea_2014-  
16_AKAlbers.shp")  
  
m <- raster(st) #blank mask the size of the snow data  
m <- setValues(m, NA)  
zn <- rasterize(sh, m) # zone transferred onto mask  
zn[!is.na(zn)] <- 1 #make sure zone mask is all 1 or NA
```

Multiply the zone raster times the snow metrics raster so that only data within the zone survives. Then run the "quantile" function to summarize the data in the zone:

```
znst <- zn*st #retain data for just the zone  
pctiles <- c(0.1, 0.25, 0.5, 0.75, 0.9)  
qday <- quantile(znst, pctiles) - 365 #-365 because css is counted from  
previous year
```

Label the rows in the output and save:

```
rownames(qday) <- substr(infiles, 0, 4)  
write.csv(qday, file = "qday.csv")
```

For a zone file with multiple zones in it, the procedure is a little more complicated. Make a raster of the zones if they are not in this format already. This is probably easiest to do in ArcGIS with the Polygon to Raster geoprocessing tool (Conversion tools, to Raster). It can also be done using R's "rasterize" function in the "raster" library. Next, for an example zone file called "NDVIZones500.tif", load and crop to match the data stack:

```
tmp <- raster("C:\\GIS\\GISdata\\Sections\\NDVIZones500.tif")  
zone_raster <- crop(tmp, st)
```

Everything should be in the Alaska Albers coordinate system, but you may have to check this using the "crs" command. Next define the percentiles to be located, count the zones and years, and set up an output matrix called "out".

```
pctiles <- c(0.1, 0.33, 0.5, 0.67, 0.9)  
zones <- unique(zone_raster)  
zn_cnt <- length(zones)  
yr_cnt <- dim(r)[3]  
out <- matrix(ncol=length(pctiles)+2, nrow=(zn_cnt*yr_cnt))  
colnames(out) <- c("yr", "zone", pctiles)
```

Now run this loop through the years and zones, running the "quantile" function to find the median and other quantiles.

```

for (j in 1:yr_cnt){
  for (i in 1:zn_cnt){
    out[(j-1)*zn_cnt+i,]<- c(j, zones[i],
quantile(r[[j]][zone_raster==zones[i]], pctiles, na.rm=TRUE))
  }
}

```

Last, write the results to a file.

```
write.table(out, file = "sno_day_zonal_stats.txt", sep = "\t")
```

F2. Method using ArcGIS.

Use the batch option of the "Zonal Statistics as Table" geoprocessing tool. The result is a table of zonal statistics for each input image. Before merging these tables to compare years, you need to add the year label to each record. The column COUNT will accommodate a 4 digit year and is not needed. After you have filled in the year in all table rows, merge these with the Merge geoprocessing command in ArcMap (Data Management Tools, General). Sort by zone and plot the mean with standard deviation vs. year for each zone, and fit a regression to the means if a linear trend is apparent.

G. Regression on pixel-by-pixel basis

G1. Regression with "Curve Fit".

The ArcMap extension "Curve Fit" by the USGS Upper Midwest Environmental Sciences Center; (http://www.umesc.usgs.gov/management/dss/curve_fit.html; Jager and Fox 2013) takes a time series of images and fits a curve to each pixel. Download "Curve Fit" and install according to the directions included with the download package. Create an ArcMap project, add the curve fit tool according to the included directions and also add a time series of images, such as the band 5 rasters created in step C above. Launch "Curve Fit" and under the "Input" tab load the images as the Y value, then type the year as the X value. Under the "Function" tab you will usually choose "Linear" but you may want to experiment with others based on inspection of the trends in the previous section. From the "Output" tab choose minimally the a coefficient (slope) and P-value for the a coefficient. Check the "Settings Summary" tab for errors and run. Be prepared for a run covering all of ARCN and more than 10 years to take an hour to complete.

When the processing is complete, check a , the slope parameter, for areas with a positive or negative trend, and also the P-value for the slope. To display, you will first need to calculate statistics for the new rasters. Open Arc Catalog, right-click on each new raster and choose "Calculate Statistics" then accept the defaults and run. Choose a threshold P-value of 0.1 or 0.05 and display all of the slopes with a P value lower than this. A good way to accomplish this is to display the probability raster with "Symbology" set to "Classified" with two classes (i.e., 0 to 0.05 and 0.05 to 1 for $P = 0.05$). Set the color for the 0 to 0.05 range to "No Color" and the color for 0.05 to 1 as black. Next display the actual slope value underneath this and it will only show through where the slopes are significant. Set the slope symbology as "Classified" with 2 classes that separate at 0, and give them contrasting colors.

Keep in mind that with such a large number of regressions, some will be statistically significant even if trends are very weak. To see how many of the pixels meet a particular P-value, use the “Reclassify” geoprocessing tool (Spatial Analyst Tools, Reclass). Reclassify the range of “significant” P-values (i.e. 0 to 0.05) to 1 and the rest of the values to zero. Run and open the “vat” table of the resulting raster to see how many pixels re-classified as “1”, as a proportion of the total. If the proportion of significant regressions was low compared to the significance criterion (e.g., less than 5% of the regressions were significant at the 0.05 level), and the “significant” pixels are scattered thinly around the study area, they probably represent noise rather than important trends.

G2. Mann-Kendall trend test.

As an alternative to linear regression using "Curve Fit", you may test for trend on a pixel-by-pixel basis using the Mann-Kendall test, a non-parametric test for monotonic trend of any shape as described on Section A, SOP #1 step E.

Section B Literature Cited

- Hijmans, R. J. 2015. *Raster: Geographic Data Analysis and Modeling* (version R package version 2.3-40). <http://CRAN.R-project.org/package=raster>.
- Jager, N. R., and T. J. Fox. 2013. “Curve Fit: A Pixel-Level Raster Regression Tool for Mapping Spatial Patterns.” *Methods in Ecology and Evolution* 4 (8): 789–92.
- Lindsay, C., J. Zhu, A. E. Miller, P. Kirchner, and T. L. Wilson. 2015. “Deriving Snow Cover Metrics for Alaska from MODIS.” *Remote Sensing* 7 (10): 12961–85. doi:10.3390/rs71012961.
- Swanson, D. K. 2014. “Snow Cover Monitoring with MODIS Satellite Data in the Arctic Inventory and Monitoring Network, Alaska, 2000-2013.” Natural Resource Data Series NPS/ARC/NRDS—2014/634. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/App/Reference/Profile/2208992>.
- Swanson, D. K. 2017. “Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network: Narrative.” Natural Resource Report NPS/ARC/NRR-2017/1380. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/DataStore/Reference/Profile/2238342>.
- Zhu, J., and C. Lindsay. 2013. “MODIS-Derived Snow Metrics Algorithm, Version 1.1.” Fairbanks, Alaska: University of Alaska Fairbanks, Geographic Information Network for Alaska. http://static.gina.alaska.edu/NPS_products/MODIS_snow/MODIS_derived_snow_metrics/version_1.0/MODIS_derived_snow_metrics_ver1_0.pdf.

C. Standard Operating Procedures for Vegetation and Snow Monitoring by Remote Automated Cameras

These Standard Operating Procedures are used to monitor the timing and duration of the vegetation green season and snow cover using remote automated cameras, as described in Section 4 of the protocol narrative document (Swanson 2017).

Table C1. Deliverables schedule for phenology cameras

Ident- ifier	Deliverable	Format	Schedule	Respon- sibility	Description
C1	Phenology camera images	Photo files (jpg)	Annually, within a month of the return from the field visit	NPS climate scientists making annual maintenance visit	Five images per day for the preceding year from each ARCN phenology camera
C2	Phenology camera time-lapse videos	Video file (mp4)	Annually before the end of the calendar year	PI	Still images processed into a time-lapse video. Created annually for the preceding calendar year
C3	Annual photo catalog	Data in SQL Server database	Annually before the end of the calendar year	PI	Annual update of list of available phenology camera photos, stored in SQL Server database as outlined in SOP #2 part B
C4	Data summary updates	Data in SQL Server database	Every 5 years (beginning 2016)	PI	Database elements updated as a part of analysis. These are described in the following sections of SOP #1: any new analysis zones (C), greenness metrics (D), fog estimates (E), fitted curve parameters (F), and snow results (I: visual estimates, fog, and final snow cover estimates)
C5	Summary report	Document file (pdf)	Every 5 years (beginning 2016)	PI	Either an NPS Natural Resource Report or peer-reviewed journal article

Standard Operating Procedure (SOP) # 1: Phenology Camera Setup and Operation Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

The camera setup is based on the Campbell Scientific manual for the CC5MPX camera (Campbell Scientific 2012). Camera settings and data download are made using an Ethernet cable and internet browser.

B. Wiring

B1. Connect the red wire to power, and the black, white, and clear to ground.

C. Configure Laptop/Ethernet on field laptop

C1. Connect the camera to the computer with an Ethernet cable.

C2. Open Control Panel-Network Connections-Local Area Connection-Internet Protocol (TCP/IP), IP Address 192.168.1.91, Subnet mask 255.255.255.0, Default Gateway 192.168.1.1

C3. Right Click Local Area Connection and click Repair

C4. Open Web Browser and go to <http://192.168.1.90>

D. Test Camera before departure.

D1. Update firmware

D2. Format SD Card (use one with extended temperature testing) and install.

D3. Configure Power Modes to maximum power saving settings (OFF state and FULL POWER SAVE MODE)

D4. Configure image capture intervals: 5 images/day at hourly intervals from 10:00-14:00, adjusted to local solar noon. Approximate solar noon values (daylight savings time) are: Kugururok 14:50, Mt. Noak 15:00, Pamichtuk 14:10, Salmon River 14:40, Serpentine Hot Springs 15:00)

D5. Configure still image settings (date, time, and station name on the outside of the image). The filename scheme is XXX00_YYYY_MM_DD_HH_NN where XXX is the 3-letter AICC station code (Table 1) followed by 2 zeros, followed by the year, month, date, hour, and minutes.

D6. Set still images to high resolution

E. Field Setup

E1. Align the scene (generally north-facing)

E2. Zoom and Focus

E3. Maximize Focusing Number

E4. Take a test image from home screen

G. Retrieve images

G1. Open in Windows Explorer (NOT Internet Explorer) and navigate to ftp://192.168.1.90:85

G2. Copy all photos to folder named with year and site.

G3. Back up photos to USB device on computer.

G3. Double-check that photos are copied and delete from camera SD card

H. Transfer images to network

H1. Upon returning to the office, copy all photos from field laptop to network folder labeled by the site and year. Verify photo transfer and delete photos from the laptop.

H2. Keep photos on the backup USB device until backup of the network drive has occurred.

I. Equipment

- Camera with sun shield
- Power cable (in conduit)
- Protective Conduit for power cable
- Battery with enclosure mount
- Solar panel with mount and U-bolts
- Wind pipe (Aluminum 1’')
- Camera mounting hardware and U-bolts

**Standard Operating Procedure (SOP) # 2: Greenness and Snow Phenology
Photograph Analysis
Version 1.0 (Jan 2017)**

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

This SOP describes the procedure used to analyze photographs taken by the phenology cameras. The process is automated using scripts written for the Python programming language. These scripts run on Python version 2.7. The Python libraries “PIL” (Python Imaging Library; Lundh and Ellis 2002), “numpy” (<http://www.numpy.org/>), and “pyodbc” must also be installed. These are the procedures used in Swanson (2015).

B. Database Updates

B1. Connect to the database.

The data for ARCN phenology cameras as of Dec 2016 is stored in the SQL Server INPYUGAMS08SQL\NUNA, database ARCN_PhenologyCameras. Users familiar with Microsoft Access may set up an Access “front end” database to query the SQL Server database, using the following information, saved in the file “phenocams_odbc.dsn”

```
[ODBC]
DRIVER=SQL Server Native Client 10.0
UID=DKSwanson
DATABASE=ARCN_PhenologyCameras
WSID=INPYUGA22311
APP=Microsoft Office 2010
Trusted_Connection=Yes
SERVER=INPYUGAMS08SQL\NUNA
```

In Access, choose “External Data”, “ODBC Database”, “Link to the data source...”, then navigate to this dsn file and select it to connect.

B2. Create photo filename list.

After returning from the field each year, populate the “photos” database table with the list of the new photo taken in that year. To speed up this process, use the following Python script. Edit the 4 lines specifying the station, start and end year-month, and output path. The script contains the paths to all the photos (edit if these change). The script searches the subfolders under these paths that match the

range of months specified, locates all photos (jpg files) and write the result to a text file. The output file is named by the station and range of months plus “photos.txt”.

```
#ListPhotos.py
#Python 2.7 script to create list of all phenology photos in specified
directory
#DKS Jan 2015
#edit the station, date range, and output file location below
#output file name is created from station, start and end dates, plus
"photos.txt"

#edit the next 4 lines
station = "MNO" #station abbreviation
startyrmon = "2013_07" #starting year and month, matches subfolder names
endyrmon = "2013_09" #ending year and month
outpath = "O:\\Monitoring\\Vital Signs\\Terrestrial Landscape Patterns and
Dynamics\\Results\\Phenocams\\Scripts\\" #path for output file
#outpath must have double back slashes and end with double back slash

import os
#these are the current photo path locations, may need to be edited in the
future
KUG = "O:\\Monitoring\\Vital Signs\\Climate and
Weather\\Photos\\Kugururok\\CC5MPX Photos"
MNO = "O:\\Monitoring\\Vital Signs\\Climate and Weather\\Photos\\Mt
Noak\\CC5MPX Photos"
PAM = "O:\\Monitoring\\Vital Signs\\Climate and
Weather\\Photos\\Pamichtuk\\CC5MPX Photos"
SRW = "O:\\Monitoring\\Vital Signs\\Climate and Weather\\Photos\\Salmon
River\\CC5MPX Photos"
SRT = "O:\\Monitoring\\Vital Signs\\Climate and Weather\\Photos\\Serpentine
HS\\CC5MPX Photos"

outfile = outpath + station + startyrmon + "_" + endyrmon + "photos.txt"
#output path and filename
photopath = eval(station) #replace station string with path

outdata = [] #output data list
dirs = os.walk(photopath).next()[1] #create a list of subfolders for the
station, one layer down only
for subfold in dirs[dirs.index(startyrmon):(dirs.index(endyrmon)+1)]:
#iterate only through folders in the year_month range specified
    photodir = photopath + "\\\" + subfold + "\\\"
    print photodir #to track progress
    #Create list of images in target directory
    jpgs = filter(lambda x: x.endswith("jpg"),os.listdir(photodir)) #create
list with just jpg file names; lambda is shortcut for fn call
    outdata = outdata + jpgs
```

```
#write outdata to text file
f = open(outfile, "w")
for item in outdata:
    f.write(item + '\n')
f.close
```

The filename lists from several stations can be merged before import into the database. This is most easily done by concatenating the files using a batch command in a bat file located with the files to be joined:

```
copy *photos.txt newphotos.txt
```

B3. Import photo filenames into the database and append.

Next, import the text file into the database and append the new filenames to the “photos” database table. Usually the first month of photos will contain some names that were appended in the previous year, so a simple append will result in some duplicates. To avoid duplicates, use a query that test for duplicates as follows:

```
INSERT INTO dbo_photos (filename)
SELECT newphotos.filename FROM newphotos
LEFT JOIN dbo_photos ON newphotos.filename = dbo_photos.filename
WHERE dbo_photos.filename Is Null;
```

The new photo filenames are in a table called “newphotos” column “filename”. It is joined to the database photos table and checks to make sure the filename is not there yet.

B4. Populate missing columns.

The “photos” table has columns “station”, “year”, and “daynum” that will be blank next to the new filenames. The following update query takes 3-letter station abbreviation from the photo filename and writes it into the “station” column (if “station” is blank):

```
UPDATE dbo_photos SET dbo_photos.station = Left([filename],3)
WHERE dbo_photos.station Is Null;
```

This query does the same for the “year” column:

```
UPDATE dbo_photos SET dbo_photos.[year] = Mid([filename],7,4)
WHERE (((dbo_photos.year) Is Null));
```

Filling in the “daynum” (ordinal date) column is more difficult because the date is stored in MM_DD format in the file name, not ordinal dates. The ordinal date for each MM_DD string is found in the table “daynum_lookup”. In Access the simplest way to fill in the blank “daynum” values is to save the following query as “DaynumUpdate1”:

```
SELECT t.filename, dbo_daynum_lookup.daynum
FROM (SELECT dbo_photos.filename, Mid([filename],12,5) AS datestr
FROM dbo_photos) AS t
```

```
INNER JOIN dbo_daynum_lookup ON t.datestr = dbo_daynum_lookup.datestr;
```

This gives you a table with the filenames and corresponding ordinal dates. Then create a second query that joins this table to the “photos” table and fills in the blank “daynums”

```
UPDATE DaynumUpdate1  
INNER JOIN dbo_photos ON DaynumUpdate1.filename = dbo_photos.filename  
SET dbo_photos.daynum = [DaynumUpdate1].[daynum]  
WHERE dbo_photos.daynum Is Null;
```

C. Define the Analysis Zones

The analysis zones (or “windows”) are one or more rectangular areas identified by pixel coordinates on images taken from a securely mounted camera so that the zones encompass the same area in all photos (see Fig. 7 in section 4 of the protocol narrative, Swanson 2017). A greenness analysis zone should encompass an area of uniform vegetation with uniform illumination (i.e. uniform slope and aspect, lacking terrain shadows) and at relatively close range. Snow analysis zones can be either a foreground area representing snow conditions in the immediate vicinity of the climate station, or a more panoramic view of distant slopes that represents snow cover across the larger landscape. An additional mask layer often must be created to remove sky or fragments of foreground from the distant analysis zones. Be sure that no sky is included in computations. Terrain shadows are usually not a problem as a result of our north-facing camera orientation, but in future setups may also need masking.

For existing cameras the analysis zones have been established and are stored in the “windows” database table. In this case simply check to make sure the camera has not shifted and, if not, then skip to step D below. If a camera shifted or was reset after shifting, define new analysis windows by the procedure described below, matching as closely as possible to the previous windows.

C1. Determine the analysis zone box coordinates.

Pixel coordinates are numbered starting at x, y = (0, 0) at the upper left corner of the photograph. Any photo editing software can be used to draw a box and determine its corner coordinates. In Irfanview (www.Irfanview.com, free software) simply drag the cursor to define a box, and read the coordinates from the window header. For example, “Selection 1000, 1000; 800 x 600;” is a box with upper left x, y coordinates of (1000, 1000), width 800 pixels and height 600 pixels. Hence the lower right x, y coordinates are (1800, 1600). These four numbers (1000, 1000, 1800, 1600) are entered into the ARCN phenology cameras database, table “windows”, with a unique name for the window (e.g., “foreground”; window names must be unique at a site but can be repeated at other sites), and used as input in the next step.

To check the position of your box and generate a figure showing its position on a sample photo; you can use Image Magick (open source graphics software; <http://www.imagemagick.org>) to accomplish this. The basic tool “convert.exe” is used to draw a rectangle on a photo. Put the photo and “convert.exe” into the same directory and create a text file named “RunCovert.bat” or similar containing the commands:

```
convert input.jpg -fill none -stroke red -strokewidth 3 -draw "rectangle
1000,1000 1800,1600" output.jpg
```

This command draws a red rectangle with upper left coordinate 1000, 1000 and lower right coordinate 1800, 1600) on a photo called “input.jpg” and writes the result to “output.jpg”. Change the input and output file names as needed, and also the box coordinates. Note that there is no comma between the two sets of corner coordinates.

C2. Create an analysis zone mask.

Sometimes the desired analysis zone does not fit neatly into a rectangular box. This is a problem for background snow analysis zones that include patches of sky or foreground. In these cases we will define a window that encompasses the entire area of interest plus some non-target areas, and create a mask that removes the undesired portions. We use the GIS to create a mask image that has the same dimensions as the rectangular analysis zone and contains zeros in pixels with “good” data and ones in pixels with data to be masked out. The mask is created as follows.

Add several typical photos from throughout the snow year to an ArcMap project; this is necessary because the camera shifts through the course of the year. Use the “Clip” geoprocessing tool (Data Management, Raster, Raster Processing) to clip the rectangular window from the full image, using the 4 coordinates obtained in the previous section. Note: the y-coordinates in ArcMap are entered as negative numbers, to signify counting down from the upper left corner. After cropping, turn off the full image from the display so you only see the smaller, rectangular cropped portion.

Create a new polygon shapefile and digitize polygons that mask out the unwanted areas. First make a single rectangular polygon that is little larger than the rectangular window from your photo. Then select this polygon and use the “Cut Polygons” tool to split out sub-polygons that outline the unwanted areas (Fig. C1). Don’t worry about what happens outside of the image boundaries, this will be deleted. Make your lines a little ways into the “good” area to avoid having camera shift bring sky into the analysis area. To be sure, check images from each month to make sure the camera shift doesn’t bring sky into the window. When you are done and before you “Stop editing” the mask, open its attribute table and type “1” in the “Id” field of all polygons that will be masked out and “0” in the analysis area.

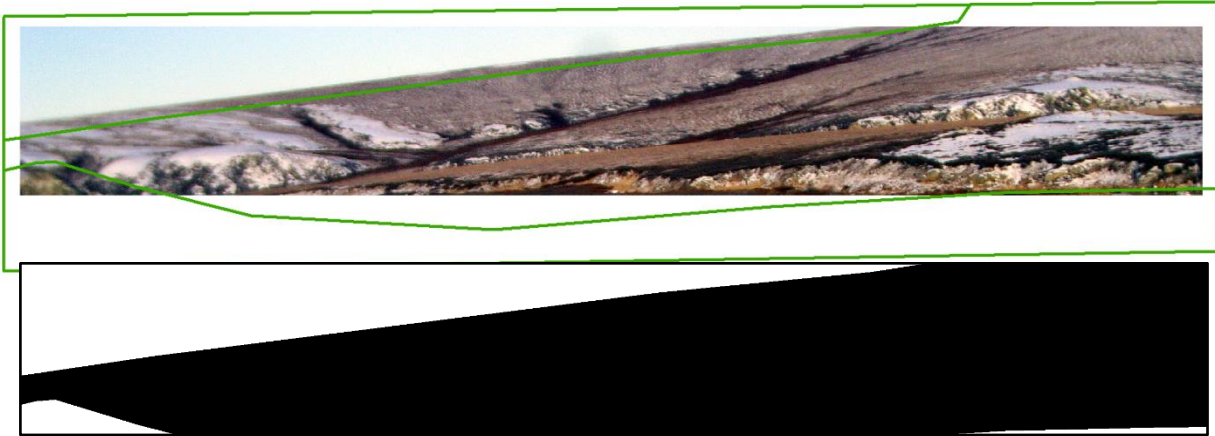


Figure C1. Creating an analysis area mask for phenology analysis. When no rectangular area can be created that encompasses the desired analysis area, we choose a larger rectangle and digitize a mask to remove the unwanted areas. The image above is the window that was initially cropped from the photo. In GIS we create polygons (displayed in green) that identify areas to mask out and areas to analyze. The sky and foreground polygons are coded as “ID = 1” while the central polygon is coded as “0”. The polygon layer is then converted to a raster and cropped to match the photo. Note that the lines are a short distance inside of the “good” area, to allow for camera shift. The resulting mask is shown in black and white below the photo.

Convert the mask polygons to a raster using the “Feature to Raster” tool (under Conversion, To Raster). Be sure the “Id” is listed in the “Field” box. Make your output a tif file. Change the cell size to “1”. Open the “Environments” settings at the bottom of the tool and choose “Processing Extent”. Set the extent from the dropdown to be the same as the clipped image.

D. Derive Greenness Metrics from the Photo Data

The following Python script iterates through all photos in a specified range of months for a station, extracts data from a specified analysis zone on each photo, and computes the mean of each band (R, G, and B). It also computes the “percent greenness” $G/(R + G + B)$ and “excess green” $2G - (R + B)$ indices for each pixel, averages them for the analysis zone, and writes the results to a text file. The user specifies the station, range of months, and window, and the script finds the path name for the photos and the coordinates of the window from the database. Thus the window must be defined and entered into the database as described in step C2 above. You must carefully edit the 5 lines near the top of the program, just below the comment “#Edit the next 5 lines”, before running. Be careful that the station abbreviation (“stanm”) and window name (“windnm”) exactly match the spelling of these items in the database. The date format “YYYY_MM” must match the subdirectories where the photos are stored.

```
# phenophotos.py
# Python 2.7 (numpy 1.8, Python image library 1.1) program to iterate through
all jpg files from a specified station and time interval, crop out specified
window,
# compute mean percent greenness and excess greenness in the window

import sys, os, time, pyodbc
```

```

from PIL import Image
from numpy import array, mean

#Edit the next 5 lines
stanm = "SRW" #station abbreviation
startyrmon = "2013_07" #starting year and month, matches subfolder names
endyrmon = "2013_07" #ending year and month
windnm = "foreground" #window name, must be present with coordinates in the
"windows" database table
outpath = "O:\\Monitoring\\Vital Signs\\Terrestrial Landscape Patterns and
Dynamics\\Results\\Phenocams\\Scripts\\" #path for output file

outfile = outpath + stanm + windnm + startyrmon + "_" + endyrmon + ".txt"
#output path and filename

#This next section connects to the database and gets the window coordinates
and paths to photos
DB_PRODUCTION_SERVER = r'INPYUGAMS08SQL\NUNA'
DATABASE_NAME = r'ARCN_PhenologyCameras'

def conn_string(server_name, database_name):
    return("ODBC;Provider=SQLNCLI10;" +
           "Driver=SQL Server Native Client 10.0;" +
           "Server=" + server_name + ";" +
           "Database=" + database_name + ";" +
           "Trusted_Connection=Yes;")

conn_str = conn_string(DB_PRODUCTION_SERVER, DATABASE_NAME)

conn = pyodbc.connect(conn_str)

SQLpath = "SELECT photopath FROM dbo.photopaths WHERE station=\'" + stanm +
"\';"
photopath = str(conn.cursor().execute(SQLpath).fetchall()[0][0])

SQLwindow = "SELECT ulx, uly, lrx, lry FROM dbo.windows WHERE (station=\'" +
stanm + "\') AND (dbo.windows.window=\'" + windnm + "\');"
windcoords = conn.cursor().execute(SQLwindow).fetchall()

# Coordinates of the analysis zone on the photos, upper left and lower right
corner, x and y; point (0,0) is the upper left corner
ulx = int(windcoords[0][0])
uly = int(windcoords[0][1])
lrx = int(windcoords[0][2])
lry = int(windcoords[0][3])

dirs = os.walk(photopath).next()[1] #create a list of subfolders for the
station, one layer down only
outdata = [] #output data list

```

```

for subfold in dirs[dirs.index(startyrmon):(dirs.index(endyrmon)+1)]:
#iterate only through folders in the year_month range specified
    photodir = photopath + "\\\" + subfold + "\\\"
    print photodir #to track progress
    #Create list of images in target directory
    jpgs = filter(lambda x: x.endswith(\"jpg\"),os.listdir(photodir)) #create
list with just jpg file names; lambda is shortcut for fn call

    for file in jpgs:
        print file #to track progress
        infile = photodir + file
        try:
            line_ = [] #initialize list to store one line of data, the info
about one photo file
            indata = array(Image.open(infile).crop((ulx, uly, lrx, lry)),
dtype=\"float\") #read data for cropped area into array, float data type to
allow future calcs

            line_.append(file) #write jpg filename to output
            line_.append(windnm) #write window name to output
            mtime = os.stat(infile).st_mtime #file last modified date-time
stamp
            line_.append(time.localtime(mtime).tm_year)
            line_.append(time.localtime(mtime).tm_yday)
            line_.append(indata[:, :, 0].mean()) #mean of red, green, and blue
bands
            line_.append(indata[:, :, 1].mean())
            line_.append(indata[:, :, 2].mean())
            greenpct = 100*indata[:, :, 1]/(indata[:, :, 0] + indata[:, :, 1] +
indata[:, :, 2]) #compute pct green; differs slightly if computed from band
means
            line_.append(greenpct.mean())
            exgrn = 2*indata[:, :, 1] - (indata[:, :, 0] + indata[:, :, 2])
#compute excess green; this could be done with means too
            line_.append(exgrn.mean())
            outdata.append(\"\".join(map(str, line_)) + \"\\n\") #map all values
in list to strings, then join them to one string with commas in between and
new line at end

        except IOError as e:
            print (\"File handling error. \" +
                infile + ': ' + repr(e))

f = open(outfile, \"w\")
f.writelines(outdata)
f.close()

```

The script output is a comma-delimited text table with the following columns (no headers):

Column 1: filename, the image file name (which contains station, date, and time information)
 Column 2: window, the analysis area
 Column 3: yr, the photo year
 Column 4: daynum, the photo ordinal date (number from 1 to 365)
 Column 5: red, the mean red value for the analysis zone
 Column 6: green, the mean green value for the analysis zone
 Column 7: blue, the mean blue value for the analysis zone
 Column 8: pctgrn, the mean green percent value for the analysis zone
 Column 9: exgrn, the mean excess green value for the analysis zone

The data from several stations can be merged before import into the database. This is most easily done by concatenating the files using a batch command in a bat file located with the files to be joined:

```
copy XXX*.txt XXXall.txt
```

Copy this text into the first line of the text file to get column headings:

```
filename,window,yr,daynum,red,green,blue,pctgrn,exgrn
```

Import the text file into the database and append to the “greenness_results” table. Since photos are usually downloaded sometime in mid-month, the database will likely already contain data for some days from the first month that were downloaded the previous year. This will cause errors due to duplicates if you try to append the new data. You can remove the duplicate records from the new data before appending, or use an insert query that first tests for duplicates:

```
INSERT INTO dbo_greenness_results ( filename, window, red, green, blue,
pctgrn, exgrn )
SELECT NewGrnData.filename, NewGrnData.window, NewGrnData.red,
NewGrnData.green, NewGrnData.blue, NewGrnData.pctgrn, NewGrnData.exgrn
FROM NewGrnData
LEFT JOIN dbo_greenness_results
ON (NewGrnData.window = dbo_greenness_results.window)
AND (NewGrnData.filename = dbo_greenness_results.filename)
WHERE (dbo_greenness_results.window Is Null)
AND (dbo_greenness_results.filename Is Null);
```

This example works if the new data are in a table called “NewGrnData” produced by the previous Python script and with column names defined as above.

E. Update Fog Table.

Before further processing can occur, you must flag the photos on which a particular window is obscured by fog. This may be meteorological clouds/fog on windows that view an area distant from the camera, or fog on the camera lens. Fog is rarely present on summer foreground images used for greenness analysis, but it is often present on background images used to analyze snow cover and occasionally on mid-range greenness windows, such as the Salmon River “tundra” window.

Append all new photo filename-window combinations in the “greenness_results” database table to the “fog” table. The following query checks for any records in “greenness_results” that lack a match in “fog” and appends them:

```
INSERT INTO dbo_fog (window, filename)
SELECT dbo_greenness_results.window, dbo_greenness_results.filename
FROM dbo_greenness_results LEFT JOIN dbo_fog
ON (dbo_greenness_results.filename = dbo_fog.filename)
AND (dbo_greenness_results.window = dbo_fog.window)
WHERE (dbo_fog.filename Is Null) AND (dbo_fog.window Is Null);
```

The same append operation is done with the “snow_results” table, after it is generated as described below. The following query updates the “fog” table to contain all window-date combinations in the “snow_results” table:

```
INSERT INTO dbo_fog (window, filename)
SELECT dbo_snow_results.window, dbo_snow_results.filename
FROM dbo_snow_results LEFT JOIN dbo_fog
ON (dbo_snow_results.filename = dbo_fog.filename)
AND (dbo_snow_results.window = dbo_fog.window)
WHERE (dbo_fog.filename Is Null) AND (dbo_fog.window Is Null);
```

Fill in the “fog” column with a 0 for clear and 1 for foggy. You may use an update query to fill in all blanks with zeros after flagging the few records with clouds to speed the process.

F. Greenness Curve Fitting and Summary

We use the nonlinear model fitting procedure “nlm” in R to fit a sigmoid curve to green-up and senescence. The multiple daily greenness values are reduced to one median value per day by SQL code in the database views “dbo.median_greenness_pct”, “dbo.median_excess_green”, and “dbo.median_snow_pct”. R draws the data directly from these database views.

The user edits the first few lines of the R code below with the correct station, window, year, start day, and end day. (Note: most failures in this procedure result from entering a combination of these values that does not exist in the data.) Next, use the comment marker (#) to block out two of the three possible data types to fit the curve to and leave one: median percent green (data from database view 'dbo.median_greenness_pct'), median excess green (from view 'dbo.median_excess_green'), or median percent snow (from view 'dbo.median_snow_pct'). If you will be generating plots as graphics files to use in a report, you will need to also edit the “path” variable.

```
#R 3.0 code to read data from the ARCN SQL Server Database
"phenology_cameras",
#Views "dbo.median_excess_green", "dbo.median_greenness_pct", or
"dbo.median_snow_pct.daily_sum"
#Plot the greenness or percent snow against day number and fit a sigmoid
curve
#The user must edit the input values station, window, year, start day, end
day, and data type (input view)
```

```

#Choose one data type (median percent green, excess green, or snow cover) and
use the comment sign "#" to mark out the other two
#Choose the start and end days to have about 10 days flat on either end of
the transition
#Run script down to the "plot(...)" command to view the data without curve
fitting

station <- "'MNO'" #note: double quoting of station and window are required
for SQL command
window <- "'foreground_tilted'"
yr <- 2013
daystart <- 197
dayend <- 263

#use the comment marker to bypass two of these lines and run just one
summary <- "'dbo.median_greenness_pct'"
#summary <- "'dbo.median_excess_green'"
#summary <- "'dbo.median_snow_pct'"

path <- "O:/Monitoring/Vital Signs/Terrestrial Landscape Patterns and
Dynamics/Results/Phenocams/Scripts/" #output file path for plot

#create no-quote versions of these variables
station_noquo <- substr(station, 2, 4)
window_noquo <- substr(window, 2, nchar(window)-1)
summary_noquo <- substr(summary, 2, nchar(summary)-1)

#connect to the database and query out the data
library(RODBC)
con.text <- paste("ODBC;",
"Provider=SQLNCLI10;",
"Driver=SQL Server Native Client 10.0;",
"Server=INPYUGAMS08SQL\\NUNA;",
"Database=ARCN_PhenologyCameras;",
"Trusted_Connection=Yes;")

ch <- odbcDriverConnect(con.text)

SQLstr <- paste("SELECT daynum, value FROM ", summary_noquo,
"WHERE station =", station, "AND window =", window,
"AND year =", yr, "AND daynum >", daystart,
"AND daynum <", dayend, ";")

dataset <- sqlQuery(ch, SQLstr)

close(ch)

gooddata <- na.omit(dataset) #remove records with no data

```

```

x <- gooddata$daynum
y <- gooddata$value

#set up plot graphic output
filename <- paste(path, paste(station_noquo, window_noquo, yr, daystart,
dayend, summary_noquo, ".png", sep = "_"), sep = "")
png(file=filename,width=800,height=800,res=200)

#convert summary code to a plot label
inlabel <- c("dbo.median_greenness_pct", "dbo.median_excess_green",
"dbo.median_snow_pct")
outlabel <- c("Percent Green", "Excess Green", "Percent Snow")

plot(x, y, xlab="Ordinal Date", ylab=outlabel[inlabel == summary_noquo])

#initial curve fit parameter values: p1 starting level, p2 amplitude
(negative for decreasing curve),
#p3: the approx length of transition in days is 2/p3, p4 center day of
transition
init = vector()
init[1] = mean(y[1:10])
init[2] = mean(y[length(y)]:y[length(y)-10]) - mean(y[1:10])
init[3] = 0.2
init[4] = mean(x)

#define the function and perform the fit
fn <- function(p) sum((y - (p[1]+p[2]/(1+exp(p[3]*(p[4]-x))))))^2)
sig <- nlm(fn, p=init)

#plot the fitted curve
xfit <-seq(daystart,dayend,2) #plot curve for length of "season", 2-day
point interval
yfit <-
sig$estimate[1]+sig$estimate[2]/(1+exp(sig$estimate[3]*(sig$estimate[4]-
xfit)))
lines(xfit,yfit)

#compute r-squared
ssqtot = var(y)*(length(y)-1)
rsq = (ssqtot-sig$minimum)/ssqtot

#compose a report variable with a summary of output
if (sig$estimate[4]<200) season <- "spring" else season <- "fall" #season
goes in the database; spring is before day 200, fall after
report <- paste(station_noquo, window_noquo, season, summary_noquo, yr,
daystart, dayend, toString(sig$estimate), rsq, sep=", ")

#parameters of curve (start value, amplitude, steepness, midpoint day
number, rsq) can be printed on the plot

```

```
dev.off() #completes writing of the plot to a file
```

First enter start and end days of 1 and 365 to see all of the data and help choose the season (green-up or senescence) to fit the curve to. Comment out (using the pound sign “#”) the line that starts with “png(file=”) so that the output goes to your computer screen (and not a file), then run the code down to the “plot” command. Examine the plot and determine the ordinal date range that starts about 10 days prior to the start of the transition (green-up or senescence) and ends about 10 days after it is over. Run the whole script except for the last line (“dev.off”, used for file output). Examine the results (Fig. C2) and adjust the start/end dates if needed. The curve parameters (a, b, c, d; see the formula in section 4 of the protocol narrative) and the r-squared correlation coefficient are given across the top of the plot. Type “report” at the R prompt to get a summary of all the coefficients at full precision; the comma-separated values are: station, window, season, the phenology measure (the source view name), year, start date, end date, a, b, c, d, and r-squared. Copy this data into the database table “curve_parameters”. The full R report for the ‘nlm’ procedure can be obtained by typing “sig”, the variable name that was assigned to the output of the curve fitting.

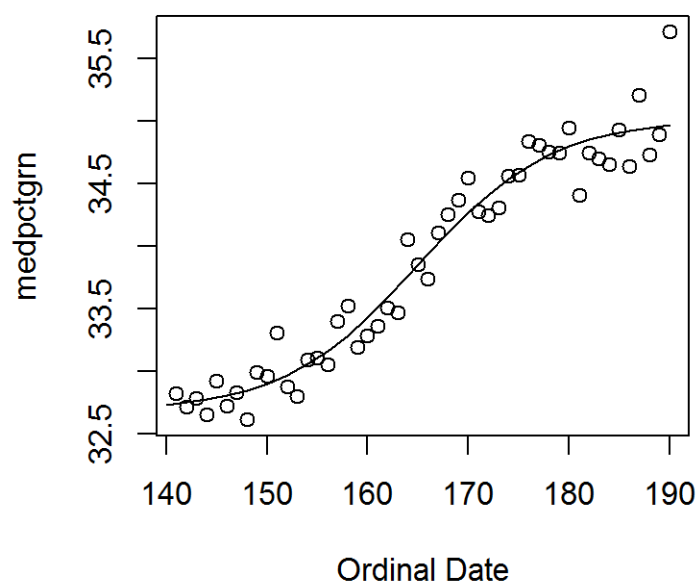


Figure C2. Example of sigmoid curve fit to greenness data.

For sites with strong red fall colors, as shown by a deep dip in greenness in the fall, we rely mainly on the minimum determined as described in the next step. To locate the midpoint of the senescence period, we can omit the anomalous points from the "red" period before fitting the curve. From the plot of greenness vs. date, note the range of dates that dip below the winter greenness value. To determine the exact day numbers, look at a sorted table of day number and greenness:

```
gooddata[order(gooddata$daynum), ]
```

Then omit the range of dates using the "subset" command. This example omits days 247 through 257:

```
gooddata <- subset(gooddata, gooddata$daynum<247 | gooddata$daynum>257)
```

Then run as before. Other anomalous date ranges (e.g. false spring green-up signal that sometimes occurs during snowmelt, or a temporary drop in greenness due to an early fall snowfall) can be screened out by this command.

Switch the comment sign to change the definition of “summary” to the other greenness measure and run the above procedure again. The process is the same to fit a curve to the snow cover data, simply move the comment marker to activate the line “summary <- "dbo.median_snow_pct” and run as before.

G. Find greenness minimum due to redness.

For windows with prominent red fall colors, our key date is the date of minimum greenness (which corresponds to maximum redness). To find this date, we will export the fall greenness data to a spreadsheet and use a 7-day moving linear regression to smooth the data.

First, use a simple select query on the database views “dbo.median_greenness_pct” and then again on “dbo.median_excess_green” to obtain daily data for one station, window, and year. Copy this data into a spreadsheet (you can simply select the whole query output table, copy, and paste as text into a blank spreadsheet). The first row is column names, “daynum” should be in column D, and greenness value in column E. To compute a smoothed value by 7-day moving regression, skip the first three days of data (because we assign the 7-day smoothed value to the middle day of the seven) and in cell F5 paste this formula:

```
=SLOPE(E2:E8,D2:D8)*D5+INTERCEPT(E2:E8,D2:D8)
```

Label column F “smoothgrn” or similar. Plot a scatter diagram of the smoothed greenness values (column F) vs. date (column D) to verify that indeed there is a fall minimum. To determine the exact date of the minimum, you can zoom on the plot and select the minimum point, or find the approximate range of dates where the minimum is and scan for it in column F. To make it easier to spot, you can use the MIN spreadsheet function. For example, if the minimum is somewhere in the range of cells F30 to F60, in cell G30 paste:

```
=IF(G30=MIN(F$30:F$60),"min","_")
```

Then ditto this formula down the column. It will print “min” in the row with the lowest value.

Repeat for the Excess Green data and enter the results in the database table “phenology_events”.

H. Determine degree-day sum for green-up midpoint.

We will create a table of degree-day sums through the year at each station, and then look up the degree-day value for the greenup date. First we need to populate the database table “daily_mean_t” for each station with the daily mean temperature for each climate monitoring station. Contact the

ARCN climate vital sign monitoring lead for the latest information on how to access the climate data. The raw data has hourly mean, maximum, and minimum temperatures. The daily mean can be obtained by computing the mean of the hourly means (preferred), or by averaging the highest hourly maximum and lowest hourly minimum for each day. The date format of these temperatures must also be converted to year and day number (ordinal date, a number between 1 and 365) so they can be inserted into the table “daily_mean_t”.

Next compute the cumulative degree-days above 0 °C for each day with a query. The example query works on the SRTtmean column (station SRT); edit all references to XXXtmean with other station names to obtain other stations. Also, to get other years, edit the year in the final line. For green-up we are not interested in adding mid-winter thaw days to the total, so this query has a condition to begin the sum after day 100 (April 10); edit this day number if a more appropriate value is needed.

```
SELECT dbo_daily_mean_t.Year, dbo_daily_mean_t.daynum,
dbo_daily_mean_t.SRTtmean, (SELECT sum(SRTtmean)
    FROM dbo_daily_mean_t AS t2
    WHERE t2.daynum <= dbo_daily_mean_t.daynum
    AND t2.[year] = dbo_daily_mean_t.[year]
    AND t2.SRTtmean>0
    AND t2.daynum > 100
    ) AS cumulative
FROM dbo_daily_mean_t
WHERE dbo_daily_mean_t.Year=2014;
```

Look up the degree-day sum for the green-up midpoint day. The cumulative degree-days can also be computed from daily means using a spreadsheet.

The sum of degree-days above 5 °C at the green-up midpoint is also of interest. This query is very similar:

```
SELECT dbo_daily_mean_t.Year, dbo_daily_mean_t.daynum,
dbo_daily_mean_t.SRTtmean, (SELECT sum(SRTtmean-5)
    FROM dbo_daily_mean_t AS t2
    WHERE t2.daynum <= dbo_daily_mean_t.daynum
    AND t2.[year] = dbo_daily_mean_t.[year]
    AND t2.SRTtmean>5
    AND t2.daynum > 100
    ) AS cumulative
FROM dbo_daily_mean_t
WHERE (((dbo_daily_mean_t.Year)=2014));
```

I. Snow Cover Analysis.

I1. Snow cover analysis Python script.

We use a Python script to compute the percent snow cover through the year in a particular window. You will need to install the Python packages “PIL” (Python Imaging Library), “numpy”, and “pyodbc”. The script is reproduced below:

```

# Python 2.7 (PIL 1.1, numpy 1.8) program to iterate through all jpg files in
a directory, crop out specified window,
# apply mask, compute snow cover in the window, print classified image as png
# DKS Jan 2015

import sys, os, time, pyodbc
from PIL import Image
from numpy import array, zeros, where, mean, var, std, percentile
import numpy.ma

stanm = "SRW" #station abbreviation
startyrmon = "2013_10" #start year and month in YYYY_DD format, same as
photo subfolders
endyrmon = "2013_10" #end year and month
windnm = "foreground" #window name
outpath = "snowtest\\" #path to write results
mapprint = "nomap" #set this as "map" to get a pixel map of snow, any other
string will result in no map printing
#maskimg = "PAMmask.tif" #if a mask is needed, place it in the same location
as the Python script and name it here
maskimg = "nomask" #set masking = "nomask" to skip use of a mask

outpathfile = outpath + stanm + windnm + startyrmon + "_" + endyrmon + ".txt"

DB_PRODUCTION_SERVER = r'INPYUGAMS08SQL\NUNA'
DATABASE_NAME = r'ARCN_PhenologyCameras'

def conn_string(server_name, database_name):
    return("ODBC;Provider=SQLNCLI10;" +
           "Driver=SQL Server Native Client 10.0;" +
           "Server=" + server_name + ";" +
           "Database=" + database_name + ";" +
           "Trusted_Connection=Yes;")

conn_str = conn_string(DB_PRODUCTION_SERVER, DATABASE_NAME)

conn = pyodbc.connect(conn_str)

SQLpath = "SELECT photopath FROM dbo.photopaths WHERE station=\'" + stanm +
"\';"
photopath = str(conn.cursor().execute(SQLpath).fetchall()[0][0])

SQLwindow = "SELECT ulx, uly, lrx, lry FROM dbo.windows WHERE (station=\'" +
stanm + "\') AND (dbo.windows.window=\'" + windnm + "\');"
windcoords = conn.cursor().execute(SQLwindow).fetchall()
# Coordinates of the analysis zone on the photos, upper left and lower right
corner, x and y; point (0,0) is the upper left corner
ulx = int(windcoords[0][0])
uly = int(windcoords[0][1])

```

```

lrx = int(windcoords[0][2])
lry = int(windcoords[0][3])

if masking == "nomask":
    maskar = zeros((lry-uly, lrx-ulx)) #make a blank mask
else:
    maskar = array(Image.open(maskimg)) #make a mask from the file supplied

def concat_calc_results(infilenm, pt1, pctexplained, pctsnow):
    line_ = [] #initialize list to store one line of data, the info about
one photo file
    line_.append(stanm)
    line_.append(windnm)
    line_.append(infilenm) #write jpg filename to output
    line_.append(pt1) #append the break point between the snow/no-snow
groups
    line_.append(pctexplained) #append the pct of variance explained by the
classification
    line_.append(pctsnow) #append the final percent snow
    #map all values in list to strings, then join them to one string with
commas in between and new line at end
    return ",".join(map(str, line_)) + "\n"

#This is the list of trial breakpoints
breakpts = range(64,255,16)
breakpts.extend([248, 252, 254]) #starting set of breakpoints to choose best
from; min 64 max 254, need these high ones for smokey scenes with little snow

outdatasum = [] #output data list
line_ = ["station", "window", "filename", "snowbrkpt", "snowpctvar",
"snowpctmeas"] #create column headers
outdatasum.append(",".join(map(str, line_)) + "\n")

dirs = os.walk(photopath).next()[1] #this gives you only the immediate
subfolders
for subfold in dirs[dirs.index(startyrmon):(dirs.index(endyrmon)+1)]:
    photodir = photopath + "\\\" + subfold + "\\\"
    print photodir

    # Create list of images in target directory; errors when bad images not
in db are found in directory
    jpgs = filter(lambda x: x.endswith("jpg"),os.listdir(photodir)) #create
list with just jpg file names; lambda is shortcut for fn call

    for infilenm in jpgs:
        print infilenm #to track progress
        inpathfile = photodir + infilenm

        #query the estimated pct snow cover for the photo/window

```

```

SQLestsn = "SELECT snowpctest FROM dbo.snow_results WHERE window=\'"
+ windnm + "\' AND filename=\'" + infilesnm + "\';"
try:
    estsnow = conn.cursor().execute(SQLestsn).fetchall()[0][0]
except IndexError: #catches bad photos missing from the database
    outdatasum.append(concat_calc_results(infilesnm, '-99', '-99', '-99'))
    continue

    if estsnow == 0: #skip computations, append no data for breakpt and
% of var explained, 0 for % snow
        outdatasum.append(concat_calc_results(infilesnm, '-99', '-99', 0))
        continue

    if estsnow == 100: #skip computations, append no data for breakpt
and % of var explained, 100 for % snow
        outdatasum.append(concat_calc_results(infilesnm, '-99', '-99',
100))
        continue

    rgbar = array(Image.open(inpathfile).crop((ulx, uly, lrx, lry)))
#read data for cropped area into array
    bluar = rgbar[:, :, 2] #single out the blue band
    analar = numpy.ma.masked_array(bluar, maskar)

    if estsnow < 40: #if small snow cover, ignore part of the dark
pixels
        lowtile = 100-(2*estsnow+5)
        lowblu = percentile(analar.compressed(), lowtile) #compression
needed to activate mask
        analar2 = numpy.ma.masked_where(analar<lowblu, analar)
    elif estsnow > 60: #if high snow cover, ignore part of the bright
pixels
        hightile = 2*(100-estsnow)+5
        highblu = percentile(analar.compressed(), hightile)
        analar2 = numpy.ma.masked_where(analar>highblu, analar)
    else: analar2 = analar #analyze the whole image if about half snow

    outdata = zeros((int(rgbar.shape[0]), int(rgbar.shape[1]),2),
dtype="float")
    nall = analar.compressed().shape[0] #compressed method removes
masked pixels; n for img masked by area but not by pixel brightness condition
    n = analar2.compressed().shape[0] #n for subset of data after
removal by pixel brightness condition

    msi1 = 10**16 #initialize the best in-groups mean sum of squares so
far
    msi2 = 10**16 #initialize 2nd best in-groups mean ssq
    sst = var(analar2)*n #total sum of square deviations from mean

```

```

pt1 = 0 #initialize the breakpoint with best ssi so far
pt2 = 0 #initialize the breakpoint with the 2nd best ssi so far
aprx = 0 #initialize the counter of approximations

for pt in breakpts: #find the two best breakpoint from list
    lows = analar2[analar2[:, :]<pt].compressed() #pixels below the
current break point, compressed method removes masked pixels
    highs = analar2[analar2[:, :]>=pt].compressed() #pixels at or
above the current break point
    if lows.shape[0] < 5: #don't bother with computations if low or
high group is empty
        continue
    if highs.shape[0] < 5:
        continue

    ssi = var(highs)*highs.shape[0] + var(lows)*lows.shape[0]
    msi = ssi/(highs.shape[0] + lows.shape[0])

    if msi < msil:
        msil = msi #former best one becomes 2nd
        pt2 = pt1
        msil = msi #new best one
        pt1 = pt
    elif msi < msil2: #or if its better than the current 2nd place
        msil2 = msi #it becomes the new 2nd place
        pt2 = pt

    if pt2 == 0: #smoke/fog and little snow can cause all pixels to be
in top group, and no pt2 is found, exit
        outdatasum.append(concat_calc_results(infilenm, '-99', '-99', '-
99'))

        continue

while abs(pt1-pt2)>1: #refine the breakpoint by Newton's method
    aprx = aprx + 1
    pt = int((pt1+pt2)/2)
    lows = analar2[analar2[:, :]<pt].compressed() #pixels below the
current break point, compressed method removes masked pixels
    highs = analar2[analar2[:, :]>=pt].compressed() #pixels at or
above the current break point
    ssi = var(highs)*highs.shape[0] + var(lows)*lows.shape[0]
    msi = ssi/(highs.shape[0] + lows.shape[0])

    if msi < msil:
        msil = msi #former best one becomes 2nd
        pt2 = pt1
        msil = msi #new best one
        pt1 = pt

```

```

        elif msi < msi2: #or if its better than the current 2nd place
            msi2 = msi #it becomes the new 2nd place
            pt2 = pt

    if aprx > 6: #break out of loop if something keeps it from
converging
        break

    if mapprint == "map":
        snowmapar = where(maskar[:, :] == 1, 125, where(bluar[:, :] > pt1, 255, 0))
        snowmapimg = Image.fromarray(snowmapar[:, :])
        snowmapimg.convert("L").save(outpath + infiles[0:-4] + "map.png")

    #Make final calculations and write to output
    if sst > 0:
        pctexplained = int(100*(1-(ssi/sst))) #percent of variation
explained by the two groups (in the subset masked by brightness)
    else: pctexplained = '-99'
        snowclass = where(analar[:, :] > pt1, 1, 0) #snow presence/absence
array from (only zone-masked) array, for 5 cover calcs
        pctsnow = round(100*mean(snowclass), 1) #estimated percent snow cover
        outdatasum.append(concat_calc_results(infiles, pt1, pctexplained,
pctsnow))

conn.close()

f = open(outpathfile, "w")
f.writelines(outdatasum)
f.close()

```

I2. Script input values.

Edit the input values near the beginning of the script just below the “import” statements `st anm`, `st ar t yr mon`, `endyr mon`, `wi ndnm`, `out pat h`, `mappr i nt`, and `maski ng`. See the comments on the script for an explanation. Note that if `mappr i nt` = “nomap”, the script will print only an output table; if `mappr i nt` is equal to anything else, a snow map will be printed for each photo window. Some analysis windows require additional masking (see step C2 above). If this is the case, put the mask in the same subdirectory as the Python script and give its name in the line `maski ng = XXXmask.tif`. If there is no mask, the statement should be `maski ng = “nomask”`.

I3. Enter snow cover estimates.

The snow mapping script requires prior estimates of snow cover. These are entered into the “snow_results” table, but first, blank records (containing only the photo file names) must be appended to the “snow_results” table to accommodate the estimates. The following query appends all of the photos from the specified station (PAM) and year (2013) to the “snow_results” table, after first

checking for duplicates; it also fills in the window column with a value specified in the query (“background” in this case):

```
INSERT INTO dbo_snow_results ( window, filename )
SELECT "background" AS window, dbo_photos.filename
FROM dbo_photos
LEFT JOIN dbo_snow_results ON dbo_photos.filename = dbo_snow_results.filename
WHERE ((dbo_photos.station="PAM")
AND (dbo_photos.year=2013) AND (dbo_snow_results.filename Is Null));
```

Edit the 2nd line **SELECT “ background” AS wi ndow** to match your window name (e.g., “foreground” instead of “background”), exactly as it appears elsewhere in the database. Edit station and year as needed and run.

If the window you are using for percent snow estimates changed during mid-year, you will need to edit the "window" in the snow results table accordingly.

Estimate the snow cover in the window on each photo, and enter them in the “snowpctest” column of the “snow_results” table. Picture viewing software that allows you to iterate through all the photos in a directory using the arrow key or similar is useful here. Estimate the snow cover as 0, 1, 2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 95, 98, 99, or 100 percent. Interpolate across foggy days and don’t worry about these, as they will be screened out anyway. You will probably want to combine this step with the next one (entering fog flags), since they both require scanning through the photos.

Since the snow cover goes for long time intervals with little change, you can enter just the "snowpctest" each time it changes and use a Visual Basic (VB) script to ditto the other values. Create a query called "editEstSnow" that selects a single camera station and monitoring window:

```
SELECT dbo_snow_results.window, dbo_snow_results.filename,
dbo_snow_results.snowpctest
FROM dbo_snow_results
WHERE (((dbo_snow_results.window)="foreground_right") AND
((dbo_snow_results.filename) Like "SRTA2_*"))
ORDER BY dbo_snow_results.filename;
```

Enter the snow cover values each time it changes. If you hav edited the window and station information used to select records, be sure to save the query before proceeding. Then use the following VB module to ditto the last snow cover across any blanks that follow:

```
Public Sub DittoPrevious()
    Dim rst As DAO.Recordset
    Dim db As DAO.Database
    Dim varSnowPctLast As Variant
    Dim varSnowPctEst As Variant

    Set db = Access.CurrentDb

    Set rst = db.OpenRecordset("editEstSnow")
```

```

varSnowPCTemp = Null

While Not rst.EOF
    varSnowPctEst = rst.Fields("snowpctest").Value
    If Not IsNull(varSnowPctEst) Then
        varSnowPctLast = varSnowPctEst
    Else
        rst.Edit
        rst.Fields("snowpctest").Value = varSnowPctLast
        rst.Update
    End If
    rst.MoveNext
Wend
Debug.Print "Done!"

rst.Close
Set rst = Nothing
Set db = Nothing

End Sub

```

I4. Update the “fog” database table.

Background snow analysis windows are often obscured by clouds or fog, which are indistinguishable from snow on a simple color digital photo. We screen out the foggy photos by flagging photos with a “1” in the “fog” database table as described in step E above. Remember that records must first be added to the “fog” table before they can be edited, as described in step E above.

I5. Run the snow cover estimation script (section I1 above) and append the results to the database.

The output file is a text file named after the station, window, and dates. The column names match the columns in “snow_results” where the data will go. Use an INSERT query to fill in the new data; this query checks to make sure there are blanks where the new data will go before inserting:

```

UPDATE dbo_snow_results
RIGHT JOIN newdata
ON (dbo_snow_results.filename = newdata.filename)
AND (dbo_snow_results.window = newdata.window)
SET dbo_snow_results.snowbrkpt = newdata.snowbrkpt,
dbo_snow_results.snowpctmeas = newdata.snowpctmeas,
dbo_snow_results.snowpctvar = newdata.snowpctvar
WHERE ((dbo_snow_results.snowbrkpt Is Null)
AND (dbo_snow_results.snowpctmeas Is Null)
AND (dbo_snow_results.snowpctvar Is Null));

```

I6. Additional details on the snow classification algorithm.

The script is used to find the optimal digital count value between 0 and 255 to be used on the blue color band of an image to separate snow and non-snow. We note that if an image is approximately

half snow and half non-snow, it will separate into two natural groups where snow is the lighter group and non-snow the darker group. The visual estimate of percent snow cover is used to screen out the darkest pixels (in a mostly non-snow scene) or lightest pixels (in a mostly snow-covered scene) so that the remaining pixels have an approximately balance of snow and non-snow and include the snow/non-snow breakpoint. If you estimate 0 or 100% snow cover, the script will accept the estimate and move onto the next image. Otherwise the script will screen out dark or light pixels to create a subset with approximately 50-50 snow and non-snow. For example, if you estimate 5% snow cover, the script will double that and add 5% more just in case the estimate was off, and then select the brightest 15% of all pixels in the window to search for a breakpoint brightness to separate snow and non-snow. If you estimate 90% snow cover, the script will double the complement to obtain 20% and add 5%, then select the darkest 25% for the breakpoint search. If the snow estimate is 40% to 60% (i.e. nearly balanced already), no pixels are screened.

Next the script iterates through a list of trial breakpoints and selects the two best. The starting list is 64, 80, 96, 112, ..., 224, 240, plus 248, 252, and 254. The “best” breakpoint is the one that yields the lowest within-groups mean square deviation from the mean (the same criterion used in a one-way analysis of variance) for the two groups. The script then refines the “best” breakpoint by choosing a point midway between the “best” and “2nd best” points; if this point provides a better result than either of the current best or 2nd best points, they are replaced and another midpoint is chosen. The process continues until the best and 2nd best points are adjacent. Then the best breakpoint is used to classify the entire window.

The percent snow pixels produced by the final breakpoint is written to the output, along with the actual breakpoint value and the percent of the variance explained by the 2-group classification. If requested, a map (classified image) is also saved, and the script continues on to the next photo.

J. Analyze results for trends

Plot the day number for each of the phenology events vs. year at each site. Test for long-term trends with a Mann-Kendall trend test or linear regression. If abrupt “stepwise” changes are apparent, years may be grouped and difference between them analyzed by t-test.

Compute the long-term mean, median, and standard deviation in the number of thaw degree-days at the dates corresponding to the midpoint of snowmelt and midpoint of green-up at all stations. Compare the stations to see if there are patterns in the heat requirement for green-up of the various plant communities.

The dates for midpoint of snow cover loss, midpoint of green-up, midpoint of fall senescence, and date of snow cover establishment obtained from the cameras can be used as verification of the analogous dates obtained from satellite observations. Intersect the coordinates of the cameras with the MODIS-derived metrics images using the ArcGIS geoprocessing tool "Extract Multi Values to Points" (Spatial Analyst Tools, Extraction). Plot the MODIS dates vs. the camera dates. For comparison, plot a 1:1 line also to show where points would plot if agreement were perfect, and note if any systematic bias is present. Keep in mind that the two methods sample different geographic areas and use similar but slightly different measures of when a phenological event occurs.

Standard Operating Procedure (SOP) # 3: Creation of time-lapse videos from times series of still photographs from phenology cameras

Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

This SOP describes the procedure used to make a time series of photographs taken by a phenology camera into a video. Our photo-to-video conversion consists of two steps: 1) reformatting of the photos to add the location, date, and NPS logo to the margins of the photo, and resize to 1920 x 1080 pixels (standard high-definition video format); and 2) conversion to video. For step 1 we use the open-source Python language to manage files and run the image processing "convert.exe" process by Image Magick (www.imagemagick.org). For step 2 we use the open source software ffmpeg (<http://ffmpeg.org>) to convert from the time series of jpg images to a video in mp4 format. The protocol was tested using ffmpeg version 2015-07-24 git-da8b70b.

Improved time-lapse videos could be created by correcting for minor camera movements that occurs between frames. Precise alignment of successive photos can be accomplished with panorama-stitching software such as "hugin" (<http://hugin.sourceforge.net/>). Due to the considerable complexity of this process it is not included in this SOP.

B. Reformatting the photographs with Python and Image Magick

We use a Python script below to format the photos with Image Magick. The output files are named "imageXXXXX.jpg", where "XXXXX" is a sequential number padded with zeros. This numbered file series is readily ingested by ffmpeg to create a video in the next step. Besides Python you must have the Python Image Library (PIL) installed, as required by the previous SOPs for phenology analysis. PIL is used to extract the original image date from the exif metadata header on the image file. This is slightly more involved but more reliable than taking the date from the file system.

```
#Python 2.7 (PIL 1.1) script to prepare phenocam photos for video creation
#The script searches a chosen directory for a chosen range of subdirectories
with format YYYY_MM
#makes a list of all the jpg files in these subdirectories
#runs Image Magick to: resample to half resolution,
#crop off the date-time strip, center on an HD video frame (1920x1080) with
black background
```

```

#write the location string chose by the user and the day-month-year taken
from the photo's exif header
#add NPS logo and output into sequentially numbered files for video creation

#User inputs-----

##Choose the directory containing the subfolders to process, script will
process all jpgs in order
#inpath = "O:\\Monitoring\\Vital Signs\\Climate and
Weather\\Photos\\Kugururok\\CC5MPX Photos\\"
#inpath = "O:\\Monitoring\\Vital Signs\\Climate and Weather\\Photos\\Mt
Noak\\CC5MPX Photos\\"
#inpath = "O:\\Monitoring\\Vital Signs\\Climate and
Weather\\Photos\\Pamichtuk\\CC5MPX Photos\\"
#inpath = "O:\\Monitoring\\Vital Signs\\Climate and Weather\\Photos\\Salmon
River\\CC5MPX Photos\\"
inpath = "O:\\Monitoring\\Vital Signs\\Climate and
Weather\\Photos\\Serpentine HS\\CC5MPX Photos\\"
#inpath = "J:\\Monitoring\\Climate\\Photos\\Gates glacier\\CC5MPX Photos\\"
#inpath = "J:\\Monitoring\\Climate\\Photos\\Upper Charley River\\CC5MPX
Photos\\"

##Choose the location text to print below the photo; space limit 65
characters (unless adjust the font to be smaller)
#location = "Kugururok, Noatak National Preserve"
#location = "Mt Noak, Cape Krusenstern National Monument"
#location = "Pamichtuk, Gates of the Arctic National Park and Preserve"
#location = "Salmon River, Kobuk Valley National Park"
location = "Serpentine Hot Springs, Bering Land Bridge National Preserve"

##Directory where output files will be written
##and where the Image Magick "convert.exe" and NPS logo "NPSlogo.png" (300 x
320 px) are located; end with \\
outpath = "C:\\Documents and Settings\\DKSwanson\\My
Documents\\VitalSigns\\Landscape\\PhenologyCameras\\video\\"

startyrmon = "2013_08" #starting year and month, YYYY_MM, matches subfolder
names
endyrmon = "2015_06" #ending year and month

startnum = 1 #normally = 1; adjust if needed to avoid overwriting files
#End of user inputs-----

import subprocess, os, glob
from PIL import Image
#dictionary to look up month abbreviations
mondict =
{"01":"Jan","02":"Feb","03":"Mar","04":"Apr","05":"May","06":"Jun","07":"Jul"
,"08":"Aug","09":"Sep","10":"Oct","11":"Nov","12":"Dec"}

```

```

imconvert = outpath+"convert.exe" #location of Image Magick convert.exe
infiles = [] #Create list of images in target directory
dirs = os.walk(inpath).next()[1] #create a list of subfolders for the
station, one layer down only
for subfold in dirs[dirs.index(startyrmon):(dirs.index(endyrmon)+1)]:
#iterate only through folders in the year_month range specified
    photodir = inpath + subfold + "\\\"
    #list subdirectory contents and filter out jpgs, append list of jpg file
names in target subdirectory
    infiles.extend(glob.glob(photodir+"*.jpg"))
n = len(infiles)
print str(n)+" photos to process"
for i in range(0,n):
    num = str(i+startnum).zfill(5) #compute the output image number and pad
so it can be part of a sorted output file name
    #use PIL to get image date from the exif data
    img = Image.open(infiles[i])
    exif = img._getexif()
    dto = exif[36867] #the original date and time are stored in this
dictionary entry
    datestr = dto[8:10]+" "+mondict[dto[5:7]]+" "+dto[0:4] #compose the date
string from the exif dictionary

    #compose Image Magick command
    process = ([imconvert,
                infiles[i], #to skip averaging
                "-filter", "box", "-resize", "50%",
                "-crop", "1296X968+0+24", #for date strip at top
                "-set", "page", "1920X1080+312+20", "-background", "black",
"-flatten",
                "-pointsize", "30", "-fill", "white", "-annotate",
"+312+1035", location,
                "-pointsize", "50", "-fill", "white", "-annotate",
"+1300+1040", datestr,
                "NPSlogo.png", "-composite",
                outpath+"image"+num+".jpg"])

    subprocess.call(process)

```

B1. User inputs.

Several lines near the top of the Python script must be edited with a text editor. First, use the comment "#" symbol to mask all the "inpath =" lines except the one desired. Next use comment symbols in front of the "location =" lines to chose the proper location text to be printed on each photo. Edit the "outpath =" statement as needed to point to the place you want the output photos to go. Image Magick executable file "convert.exe" and the NPS logo file "NPSlogo.png" (300 x 320 pixels with black background) must also be in this location. Fill in the start and end year and months in the next two lines, in format YYYY_MM to match the directories where the photos are stored in the climate subdirectory on the NPS network. We usually create a video for each calendar year.

Finally, there is an option to start numbering the output photos with some number other than 1; this may be required if you are processing photos in two batches that you want to joined into one video.

Most of our cameras are set to write a small date-time strip at the top of the frame; this is cropped off by the "-crop" Image Magick commnd near the end of the script (and replaced with date info in larger text). For a camera set to put the date-time strip at the bottom, substitute this line:

```
"-crop", "1296x972+0+0",
```

B2. Image smoothing.

A time-lapse video made from photos widely separated in time flickers and appears jerky because of very different illumination conditions between successive frames. A smoother video is obtained by creating the video from a time series of photos that are a running average of 5 successive frames. The disadvantage of this technique is that the viewer cannot inspect individual frames that match the original photos for rapidly changing features such as wildlife, and some blurring occurs due to camera movements. If smoothing is desired, change the loop definition line `for i in range(0, n):` to `for i in range(2, n-2):` and change the line `"infile[i],"` to

```
infile[i-2], infile[i-1], infile[i], infile[i+1], infile[i+2], "-  
evaluate-sequence", "mean", .
```

B3. Run the script.

Save the edits and double-click the Python file to run.

C. Create a video using ffmpeg

Copy the executable file "ffmpeg.exe" into the same directory as the photo time series created in the previous step. Open a command prompt and type the drive name (e.g. C:) if it differs from the default, and then type `cd` plus the path name to the directory where the photo time series is. Then copy the following command to the command prompt:

```
ffmpeg -f image2 -framerate 8 -i image%5d.jpg -crf 30 -s 1920x1080 output.mp4
```

Hit return to run. This command uses ffmpeg to convert a series of jpg files into a video with 8 frames per second (i.e. each day – 5 photos – takes just over half a second) and resolution of 1920 x 1080 pixels, and saves it in mp4 format. The input file names have the format "imageXXXXX.jpg", where "XXXXX" is a sequential number starting with one and padded with zeros (symbolized by the "%5d" symbol). The "crf" ("constant rate factor") adjusts the file compression, thereby making the compromise between quality and file size. This number can range from 0 (lossless compression, huge file) to 51 (the worst possible quality, tiny file). The ffmpeg default is 23, which results in pretty big files. A crf of 30, recommended here, reduces the file size to less than 1/4th of the default with no obvious loss in quality.

If your photos don't start at number 00001, use this modified command:

```
ffmpeg -f image2 -framerate 8 -start_number 01366 -i image%5d.jpg -crf 30 -s  
1920x1080 output.mp4
```

Edit the number after `-start_number` to match the first image in your list.

You may also put several ffmpeg commands into a *.bat text file, save, and double-click to run. This is convenient if you have several groups of images in different subfolders to make into several videos at once. For example:

```
ffmpeg -f image2 -framerate 8 -i MNO13\image%%5d.jpg -crf 30 -s 1920x1080  
mno13.mp4  
ffmpeg -f image2 -framerate 8 -i MNO14\image%%5d.jpg -crf 30 -s 1920x1080  
mno14.mp4  
ffmpeg -f image2 -framerate 8 -i MNO15\image%%5d.jpg -crf 30 -s 1920x1080  
mno15.mp4
```

This bat file looks for ffmpeg.exe in the same directory as itself, then goes into subdirectories MNO13, MNO14, and MNO15 (i.e. 3 different years at station MNO) to read in the jpg images and make the 3 videos, and saves the videos in the same directory as the batfile. Note that in an MS DOS batfile you need to put in two percent signs "%%" in the wildcard number string.

Section C Literature Cited

- Campbell Scientific. 2012. "Instruction Manual, CC5MPX and CC5MPXWD Digital Network Cameras, Revision 3/12." Campbell Scientific, Logan, Utah.
<http://s.campbellsci.com/documents/us/manuals/cc5mpx.pdf>.
- Lundh, F., and M. Ellis. 2002. "Python Imaging Library Overview."
www.pythonware.com/media/data/pil-handbook.pdf.
- Swanson, D. K. 2015. "Monitoring of Greenness and Snow Phenology by Remote Automated Cameras in the NPS Arctic Inventory and Monitoring Network, 2013-14." Natural Resource Data Series NPS/ARCN/NRDS—2015/798. Fort Collins, Colorado: National Park Service.
<https://irma.nps.gov/DataStore/Reference/Profile/2222147>.
- Swanson, D. K. 2017. "Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network: Narrative." Natural Resource Report NPS/ARCN/NRR-2017/1380. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/DataStore/Reference/Profile/2238342>.

D. Standard Operating Procedures for Land Cover and Landform Monitoring with High-Resolution Aerial Photographs

These Standard Operating Procedures are used to monitor changes in vegetation, land cover, and landforms using high-resolution imagery, as described in Section 5 of the protocol narrative document (Swanson 2017).

Table D1. Deliverables schedule for systematic grid image analysis

Ident- ifier	Deliverable	Format	Schedule	Respon- sibility	Description
D1	Images	Photo files (jpg or RAW); or satellite images	Within a month of the return from the imaginig flight; every 10 or 15 years beginning in 2008	PI contracts the photography flights, or acquires satellite images	High-resolution images (pixel size 1 m or less) that cover the plots on the systematic 20-km grid
D2	Image georeferencing	World files	Within 6 months of image acquisition	PI	World files that locate the images within 2 m accuracy. These are stored with the image files. Procedure described in SOP #2. Future satellite images may be delivered with georeferencing and not require a separate deliverable.
D3	Data recording spreadsheets	Excel or other spreadsheet	Within 9 months of image acquisition	PI	Data forms with data on plot-level change as described in SOP #3 part C
D4	Data from image interpretation	Data in SQL Server database	Within 9 months of image acquisition	PI	Import data from spreadsheets with plot-level change into the database (SOP #3 part C). Import data from the GIS file of subplot (37 cell) data in the database (SOP #3 part D).
D5	Summary report	Document file (pdf)	Within 1 year of image acquisition	PI	Either an NPS Natural Resource Report or peer-reviewed journal article

Standard Operating Procedure (SOP) # 1: Ordering aerial photographs

Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A Background

This SOP describes the specifications for small-format vertical aerial photos used for change detection on the ARCN 20-km grid.

B. Specifications

Photo format: we prefer to receive both jpg and a lossless format, but jpg alone is adequate. Conventional cameras that record visible red, green, and blue light are adequate. If images with near infrared light become available in the future these would be useful also.

Photo orientation: the camera should be oriented so that the short side of the camera frame is parallel to the flight path. The flight direction of the aircraft may be chosen for convenience and safety.

Photo footprint: Approximately 400 by 600 m. Our 4-ha sample plot circle (diameter 225 m) should fit easily within the short dimension of a photo.

Adjacent photo overlap: 75 to 80%. This will allow for creation of orthophotos by photogrammetry if desired, and it ensures that the 4-ha circular plot will always be accommodated on a single photo.

Photo resolution: pixel size as measured on the ground should be 20 cm or less.

Photo season: photos should be in midsummer or later so that only semi-permanent snowbanks remain. Photos should be taken during the leaf-on season. Photos taken when fall vegetation colors are visible can be useful for distinguishing plant species, but care must be taken to avoid imaging part of the study area after leaf-drop has begun.

Sun angle: photographs should be taken with a sun angle of at least 20° above the horizon. See ESRL Global Monitoring Division (2016) for solar angle calculations.

Photo locations: camera coordinates of at least mapping grade are required (5 to 10 m minimum horizontal accuracy). Highly accurate coordinates are preferred (less than 50 cm) because georeferenced orthomosaics may be produced. Coordinates should be written into the photos exif metadata and supplied in table form.

Standard Operating Procedure (SOP) # 2: Cataloguing and georeferencing aerial photographs

Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

This SOP describes how to manage a new group of small-format vertical aerial photos. The steps include reading the location information out of the file headers and using it to create an approximate world file for the photos, so that they will display at the approximately proper scale and location in a GIS. Then more exact georeferencing is performed to allow accurate location of systematic sample points onto the photos. These are the procedures that were used in Swanson (2013)

The metadata for photographs taken with a digital camera in the jpg and tif formats is often stored in the header according to the Exchangeable Image File format (Exif) specification. This header information contains a large amount of information, such as the type of camera and time/date of exposure. It will also contain location information if the camera was integrated with GPS, as in ARCN's 2008-2011 small-format photographs. This Exif information can be used to create a photo index layer for the GIS, and to create initial georeferencing information (world files) that can be used to load the photos into the GIS so they will appear near the correct location and at the approximately appropriate scale. ArcGIS is then used to refine the georeferencing information.

If the coordinates are already available in tabular form as an index file, then step B below can be skipped and you can proceed directly to step C.

B. Create an index from Exif headers

B1. Background.

Software that can read Exif headers from numerous images and output them to a file in a batch operation is needed to extract information efficiently. This SOP describes the use of the open-source utility "ExifTool" (<http://www.sno.phy.queensu.ca/~phil/exiftool/>).

B2. Write Exif header information to a file.

The file "exiftool.exe" should be downloaded and copied into the directory where the photos reside. The exiftool utility can be run at a DOS command prompt, but it is easier to run it from a *.bat file. To do this, open your text editor (i.e. Notepad) and copy this command:

```
exiftool -n -fast2 -DateTimeOriginal -GPSLatitude -GPSLongitude -GPSAltitude  
-csv *.jpg >exif_out.csv
```

This command looks in the folder where the *.bat file is for all jpg files and writes the exif items listed in the command to a csv file (comma-separated text table). The "-n" option keeps the data in its original form, which in the case of latitude and longitude is what we want (decimal degrees). The "-fast2" option speeds up the process by skipping some information that we don't need. Note that any other exif information could be added to this command, such as "-Aperture" or "-ShutterSpeed".

B3. Create an index file with Alaska Albers coordinates.

Add the table obtained in step B3 to an ArcGIS project, right-click on the table name in the Table of Contents and choose "Display XY Data", and use the dialog box to indicate the coordinate fields and coordinate system (usually Geographic for GPS data). Convert the resulting event layer to a shapefile by right-click and choose "Data", "Export Data". Add the shapefile to a project with Alaska Albers as the current coordinate system, open the attribute table, add columns for "Xcoord" and "Ycoord". Then right-click on the new Ycoord column and choose "Calculate Geometry" and a dialog box opens. Check to make sure that it is set to compute the Y coordinate of the point, then select "Use coordinate system of the data frame" and units of meters. Repeat for the "Xcoord" column. This shapefile shows the photo locations and its attribute table can be used to generate world files.

C. Initial registration of aerial photographs

C1. Background.

World files are simple text files with a jgw or tfw extension (for jpg and tif, files respectively) that contain 6 numbers to locate and scale an image in the GIS. It is useful to batch-create world files for a whole photography project. Then each photograph will appear in approximately the right place and scale when displayed in a GIS. The following instructions refer to the creation of world files to be displayed on an Alaska Albers map projection. This procedure is useful for approximate registration of photographs only. For more precise registration multiple tie points between the two images must be collected using ArcMap georeferencing, as described in the next section.

An approximate world file can be constructed if the following are known: 1) the x,y coordinates of the photo center. These were computed from the GPS coordinates of the aircraft when the photo was shot in section B above; and 2) the pixel resolution of the image (i.e. size of the pixels on the ground in meters). The pixel resolution can be determined by georeferencing a few sample photos, or from the footprint size (in meters) divided by the image dimensions (in pixels).

For initial georeferencing we will construct approximate world files with six lines containing:

- The pixel dimension in meters
- 0 (Zero)
- 0 (Zero)
- Negative the pixel dimension in meters
- X coordinate of the photo center, in meters (Alaska Albers)
- Y coordinate of the photo center, in meters (Alaska Albers)

This will place the upper left corner of the photo at the point where its center point should be and place the top of the photo to the north.

C2. Generate approximate world files.

Create a spreadsheet with the photo filenames and center x and y coordinates (in meters of the Alaska Albers coordinate system) obtained in part B above. Add the approximate pixel size to a fourth column and save as comma-delimited text (*.csv). The file should look like this:

```
Photo,Xcoord,Ycoord,pixel_size
Photo1.jpg,-200111,1907537,0.2
Photo2.jpg,-202035,1908125,0.2
Photo3.jpg,-203945,1908701,0.2
Photo4.jpg,-205842,1909267,0.2
...etc.
```

This file will be the input file for a Python script that creates world files for the corresponding photos. The row of column header names must be present in the file, but the exact names are not important as this line will be skipped over. Create a subdirectory called "wfs" in the same directory as the input coordinates table. Edit the first 2 lines of the Python script below to match your input path and file name. Edit the worldfile extension variable "wf_ext" to match your file type: "jgw" for jpeg images, "tfw" for tiff images, pgw for png images. Then run.

```
#Python 2.7 script to create approximate worldfiles from a table of photo
names
#and photo center coordinates
#The input file must be comma-delimited, have a header line, and 4 columns:
#the photo file name, the pixels size in m, the Alaska Albers x coordinate,
#and the Alaska Albers y coordinate
#Edit the input path and file with table of coordinates (note double
backslashes)
#The world files will be written to the folder "wfs" in the specified
directory
#This subdirectory must be present for the script to run
#Also supply the appropriate world file extension (wf_ext) for your images:
#"jgw" for jpeg images, "tfw" for tiff images, pgw for png images

inpath = "C:\\GIS\\GISdata\\USGSphotos\\"
infilenm = "test.csv"
wf_ext = ".jgw"

import csv
infile = open(inpath+infilenm, 'r')
data = csv.reader(infile, delimiter=',')
data.next() #read past header line

for line in data:
    wf = [float(line[3]), 0, 0, -float(line[3]), float(line[1]),
float(line[2])]
```

```

outfile = open(inpath+"wfs\\"+line[0].split('.')[0]+wf_ext, 'wb')
tmp = csv.writer(outfile, delimiter='\n')
tmp.writerow(wf)
outfile.close()

```

Copy the world files to the location of the corresponding photos, and the GIS will automatically read them and display the photos at approximately the right location.

D. Exact georeferencing of aerial photographs

D1. Exact georeferencing of small-format aerial photographs

D1.1. Obtain the best available orthorectified high-resolution imagery available and add it to an ArcMap project.

D1.2. Add the photos to the ArcMap project. If you have generated approximate world files for your photos as described above, they will appear near their proper position in the GIS, but may be rotated from their proper alignment.

D1.3. At each sampling grid point, choose the small-format image that is best centered over permanent monitoring grid point. Only this one must be georeferenced. It may, however, be necessary to georeference two photos to determine which is best centered over the grid point.

D1.4. Add the Georeferencing Toolbar (from Customize-Toolbars) to the view. Select the layer (photo file) to georeference in the box in the georeferencing toolbar. Add tie points by clicking on the X—X icon and successively clicking on points on the photo and the orthoimage, clicking the photo on and off as needed to make the orthoimage visible. Choose distinctive, small ground features such as large rocks, compact shrubs, and sharp patterned ground features. As you are working the photo registration should improve. If it suddenly deteriorates, open the link table (the table icon next to the X—X icon) and delete the last point. This table should be checked as georeferencing continues to follow your progress. The first-order polynomial (affine) transformation that we are using requires a minimum of 3 points, but we will typically use 6 to 10 points, often deleting one or two that are a poor fit. Shoot for a root mean square error equal to or less than the pixel size of the reference image (1 m in the case of an IKONOS background image). Remember that we are most interested in the area within 100 m of the grid point. The “swipe” tool (available under Customize-Toolbars, Effects) is useful when looking for control points.

Note: if you did not produce a preliminary world file for the photo, it will initially plot far away from the proper position and not be visible. In this case, select the photo in the Layer choice box and then select “Fit to Display” under the Georeferencing drop-down menu. This will place the small-format image roughly over the reference image. Or you can right-click on the photo in the table of content and choose “Zoom to Layer”. Then proceed as before with tie points.

D1.5. When the high resolution photo has been adequately georeferenced, hit Update Georeferencing on the Georeferencing toolbar dropdown menu, keeping the default 1st-order polynomial (affine) transformation. This will write a new world file to the folder where your photo resides. Its extension

will be *.JGwx, and your original worldfile will persist. We save these worldfiles as a project deliverable and record of the georeferencing used for sampling.

D1.6. Finally, open the attribute table of the ARCN20k_GridPts shapefile (the layer containing the locations of the 20-km grid point centers) and add the name of the photo file that you just georeferenced to the column labeled PtCntPhoto# (where # is the sampling repetition number).

D2. Exact georeferencing of large-format aerial photographs.

The procedure for georeferencing large-format aerial photographs, such as scans of standard 9” by 9” aerial photos, is similar. For our purposes we only need good registration near the sample point. So we generally define 2 or 3 tie points near the perimeter of the photograph to orient and scale it approximately, then zoom into the sample area and define 3 to 5 more points there. One of two of the early (distant) points may be deleted to improve the fit to the local sampling area. As in the case of small-format photos, when the high resolution photo has been adequately georeferenced, hit Update Georeferencing on the Georeferencing toolbar dropdown menu, keeping the default 1st-order polynomial (affine) transformation. This will write a new world file to the folder where your photo resides. Its extension will be *.JGwx, and your original worldfile will persist. We save these worldfiles as a project deliverable and record of the georeferencing used for sampling

Standard Operating Procedure (SOP) # 3: Change detection using high-resolution aerial photographs
Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

This SOP describes how to photo-interpret change in vegetation and other landscape features on high-resolution aerial photographs. Photographs from two dates are georeferenced as described in SOP #2. Circles 4 ha in size centered on the 20-km grid points are projected on the photos and changes between the two dates are recorded on a form. The land cover of the circle is also classified by the system used in the ecological land survey of the Arctic Network (Jorgenson et al. 2009). These are the procedures that were used in Swanson (2013).

B. Generate and view the sample circles in ArcMap

B1. The original 20-km grid system was created with an arbitrary starting point. A shapefile consisting of circles with an area of 4 ha around each 20-km grid point was created using the “buffer” geoprocessing tool (under “Analysis Tools”, “Proximity”). The input feature was the 20-km grid points; and the output was named “GridPt4haCircles.shp”. Under “Distance [value or field]” we chose “Linear unit”, “meters”, and entered 112.84 as the linear distance (this is the radius of a 4 ha circle). This same shapefile can be used for all future sampling events. Display the resulting shapefile with no fill to allow viewing of the circle centers.

B2. Create an ArcMap project with the following layers: the sample circles, the sample points in the circles (see step D below), the photograph index shapefiles (used to select the photographs to display), the photographs from both dates that will be compared, the high-resolution ortho-image used to georeference the photos, a medium-resolution satellite image (e.g. 30 m Landsat image), and the ARCN land cover map (Jorgenson et al. 2009). It is helpful to create a group layer for each photo date and add multiple photos to the group.

B3. View both the landscape setting of the sample point (e.g. at 1:10,000 scale) with the Landsat image and the ARCN land cover map, and also zoom in until the sample circle fills most of the window. Use the “swipe” tool to sweep the top photo on and off. The “swipe” tool is available on the

“Effects:” toolbar: “Customize”, “Toolbars”, “Effects”. On this toolbar you may select the layer or group of layers to swipe.

C. Photo-interpretation of change at the whole plot level

C1. Background.

This section describes how to photo-interpret and record changes for the 4-ha plot as a whole. Data are recorded on a spreadsheet form with drop-down menu choices that present the choices given in Table 3 (in the narrative section of this protocol). We use a single spreadsheet file for each park unit, with multiple sheets for convenience in the two large parks (NOAT and GAAR). The menu choices are hierarchical. For each change category (shrubs, trees, wetlands, etc.) the analyst first determines the direction of change (“no change”, “increase”, or “decrease”). The outcome of this choice determines the next level of choice, the mode of change; for example, under “shrub increase” the choices are “expansion on tundra”, “post-fire succession”, etc. (see Table D2). We also record the vegetation types and presence of ice-wedge polygons; this procedure is outlined in section D below, but in practice will be performed at the same time as the change detection.

Table D2. Land cover change types and mechanisms.

Change type	Change mechanism
Shrub_increase	increase on tundra post-fire succession floodplain succession increase on wetland other primary succession
Shrub_decrease	fire floodplain erosion succession to trees wetland expansion
Tree_increase	increase on tundra post-fire succession floodplain succession increase on wetland other primary succession infilling below treeline
Tree_decrease	fire floodplain erosion wetland expansion
Lichen_increase	succession

Table D2 (continued). Land cover change types and mechanisms.

Change type	Change mechanism
Lichen_decrease	fire erosion wetland expansion shrub increase herbivory/other
Barren_increase	fire floodplain erosion snowfield/icing expansion mass movement
Barren_decrease	floodplain vegetation increase vegetation increase on snowfield/icing upland vegetation increase
Wetland_increase	thermokarst non-thermokarst water level rise paludification
Wetland_decrease	thermokarst non-thermokarst water level drop
Water_increase	thermokarst non-thermokarst water level rise stream channel migration
Water_decrease	outlet incision non-thermokarst water level drop stream channel migration

C2. Data entry form.

The first column contains row labels. Each plot (circle) occupies a column on the data form. After each plot is entered we scroll left to right with the buttons in the first column and frozen on the left pane (“View, Freeze Panes”) so that the row labels remain conveniently displayed (Fig. D1).

C2.1. The dropdown lists are stored on a second worksheet called “Lists”. The lists are defined by highlighting them, then selecting “Formulas”, “Define Name”. First name the list used for most features (containing “no change”, “increase”, and “decrease”) as “trend”. In the first cell to the right of the row labeled “Shrub_”, choose “Data”, “Data Validation”, and in the dialog box that opens under “Allow” select “List”, and under “Source:” enter “=trend”. This will cause the “trend” list to be displayed here as a dropdown.

The hierarchical feature of the choice lists is used to program the choice list for the next line, the mechanism (“mode” for short) of change: “shrub mode”. First create the lists for mechanisms of

shrub increase, shrub decrease, and the default for no change (Table 3) and name these lists as “Shrub_increase”, “Shrub_decrease”, and Shrub_nochange”, respectively using “Define Name” as above. Now in the row next to “shrub mode”, choose “Data”, “Data Validation”, and then enter the formula into the dialog box as shown in Fig. D2.

This causes Excel to choose the choice list named by merging the word “Shrub_” from cell A4 (the row label) with the choice you made above “increase”, “decrease”, or “nochange”. Analogous formulas are entered for all cells, and then copied to the right into other columns.

C2.2. The ice-wedge change feature has a simple dropdown list with no hierarchical feature. The choices are “no change”, “low-center/flat to high-center”, and “high-center to higher-center”.

	A	F	G	H
1	Point	BELA_-43_177	BELA_-43_179	BELA_-43_181
2	Photo1	AB580002909ROLL_5918	AB580002909ROLL_5978	AB578002620ROLL_5399
3	photo2	BELA_GP06-2_20080820	BELA_GP09-2_20080820	BELA_GP19-1_20080820
4	Shrub_	nochange	increase	nochange
5	Shrub mode	NA	expansion on tundra	NA
6	Tree_	nochange	nochange	nochange
7	Tree mode	NA	NA	NA
8	Lichen_	nochange	nochange	nochange
9	Lichen mode	NA	NA	NA
10	Barren_	nochange	nochange	nochange
11	Barren mode	NA	NA	NA
12	IceWedges_	nochange	nochange	nochange
13	Wetland_	nochange	nochange	nochange
14	Wetland mode	NA	NA	NA
15	Water_	increase	nochange	nochange
16	Water mode	thermokarst	NA	NA
17	Note	original shrub water track now has canopy break over channel exposing some water surface, plus some very small ponds (5x2 m) have formed	You can see some new alders on the hi-res.	

Figure D1. Data entry form for aerial photograph change detection.

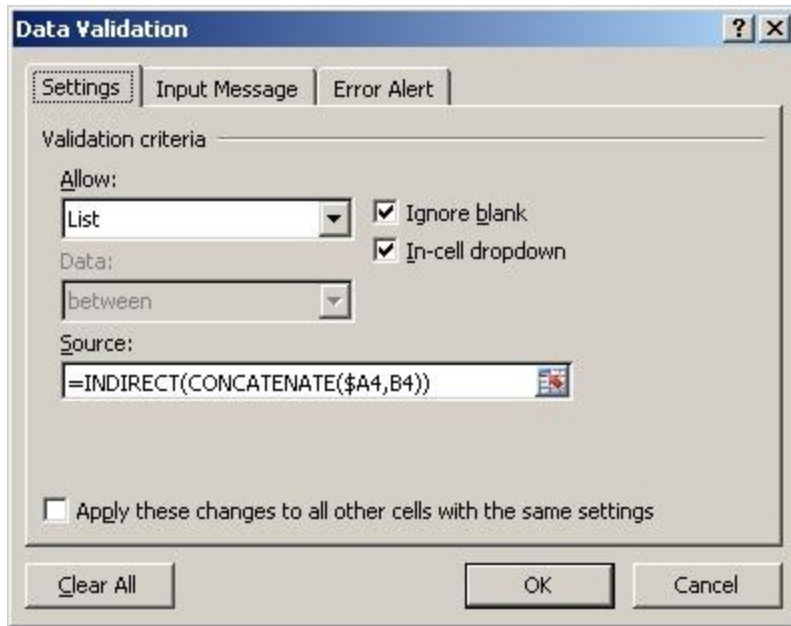


Figure D2. Data validation dialog box entries used in the aerial photo change data entry form. In rows where the mode of environmental change is recorded (see Fig. D1), the “INDIRECT” statement shown will cause the correct dropdown choice list to appear based on the type of change recorded

C3. Data summary.

When a park unit is completed, copy the table and past to a new sheet using the “Paste Special” feature with “Values” and “Transpose” selected. In this form the table may be imported into the database for further processing. This will usually consist of summary queries using the “Group by” and “Count” functions.

D. Land cover and ice-wedge polygons classification using 37 subplots

D1. Background.

The land cover composition and condition of ice-wedge polygons in the sample circle is determined by overlaying a dot grid on the circle. The dots mark the centers of the 37 hexagonal cells depicted in Fig. 13 of the narrative section 5 (Swanson 2017); we do not actually plot the hexagons. We classify the vegetation in the vicinity of each dot and record the class in the GIS attribute table of the dot grid. The classification of each point is easily depicted in ArcMap to track progress. When classification is complete, the proportion of dots in each class provides an estimate of the composition of the sample circle. A hexagonal dot grid that fills the sample circles has been generated for the ARCN 20-km grid points. Normally the land cover and ice wedge polygon classifications are performed at the same time as the change detection.

D2. Setup.

Create an ArcMap project as described above in step B2. Add the shapefile containing the hexagonal dot-grid of points in the 4-ha circles four times; this allows us to display 4 different point label schemes (vegetation from the two photo dates and ice-wedge polygons from the two photo dates). If you label one of your photo dates with underlines and the other plain, you can readily compare the two dates. (To label the hexagonal dot grid points, right-click on the layer, choose “Label features”;

then double-click on the layer, choose the “Labels” tab, and select the desired field). Right-click the dot grid layer and select “Edit features”, “Start Editing”.

On the editing toolbar, click on the “Attributes” icon (resembles a written page) to open a small table where attributes may be entered. In the ArcMap Table of Contents, choose the “List by Selection” format and make only the dot grid layer selectable. Open the database or spreadsheet containing the ecotype choice list.

D3. Classifying the land cover.

Use the ecotype legend from Jorgenson et al. (2009). This publication contains several classification levels that result from aggregating small or similar classes. The one we will use is “Regional Map Ecotype (short)” as listed in Appendix 8 of Jorgenson et al. (2009). This matches the ecotype legend in the GIS layer. Additional information that can be used to identify these types is found in Wells and Macander (2010). In complex areas, choose the dominant ecotype within the hexagonal area represented by the point. Keep in mind that the hexagonal area may be classified as a tree or shrub type even if the point itself falls in a small canopy gap.

Look at the previous photo date and its classification when choosing ecotypes. If no change is obvious, the ecotype from both dates should be the same, even if it is uncertain. If any point changes ecotype between sample dates, there should be a corresponding change recorded on the form in part C above. The reverse is not necessarily true, however: a detectable change as noted on the form for part B may occur without changing the ecotype. For example, a white spruce forest may become thicker (tree increase) but is nonetheless the vegetation class is white spruce forest on both dates.

D4. Data recording.

Select groups of points from the dot grid using the arrow from the editor toolbar (drag the arrow around groups of points). The points will appear in the “Attributes” window. Click on the layer name in the “Attributes” window (to allow you to edit all the points at once), and copy the ecotype symbol into the row (which corresponds to a column in the attribute table) that matches the image you are classifying. The new labels should appear on the display. Continue until all are labeled correctly and hit “Save Edits” on the editing toolbar dropdown.

D5. Classifying the ice-wedge polygons.

Ice-wedge polygons are placed into one of the following classes (Fig. D3):

None – polygons not visible.

Low-centered – the polygon centers are lower than the polygon rims (though not necessarily lower than the troughs between the polygons). Normally the relief on polygons is too subtle to see the low centers even on high-resolution stereo photos, so low-centered polygons are inferred from the wetter conditions observed in the center versus the rims. This is usually manifested in vegetation differences (graminoids in the centers, shrubs on the rims) and/or darker colors due to wetness or standing water in the centers.

Flat – the polygons appear uniform except for narrow lines marking the ice wedge troughs.

High-centered – the polygon centers are higher than the rims and troughs. You can often see the relief on high-resolution photos in stereo. Also, high-centered polygons can have wide troughs and visible concentric zones, with the centers appearing drier (lighter-colored) than the troughs.

Obviously, any point that falls within a closed ice-wedge polygon should be classified into one of the 3 polygon types. In areas where polygon boundaries are discontinuous, examine the path between each point and its 6 neighbors; if an ice wedge trough (i.e. a linear feature) is visible between any two neighboring points, classify the polygon type for both points, otherwise list “none”. For points on the edge of the circle, visualize the path to missing neighbors outside of the circle and classify polygon type of the point if the path crosses a wedge.

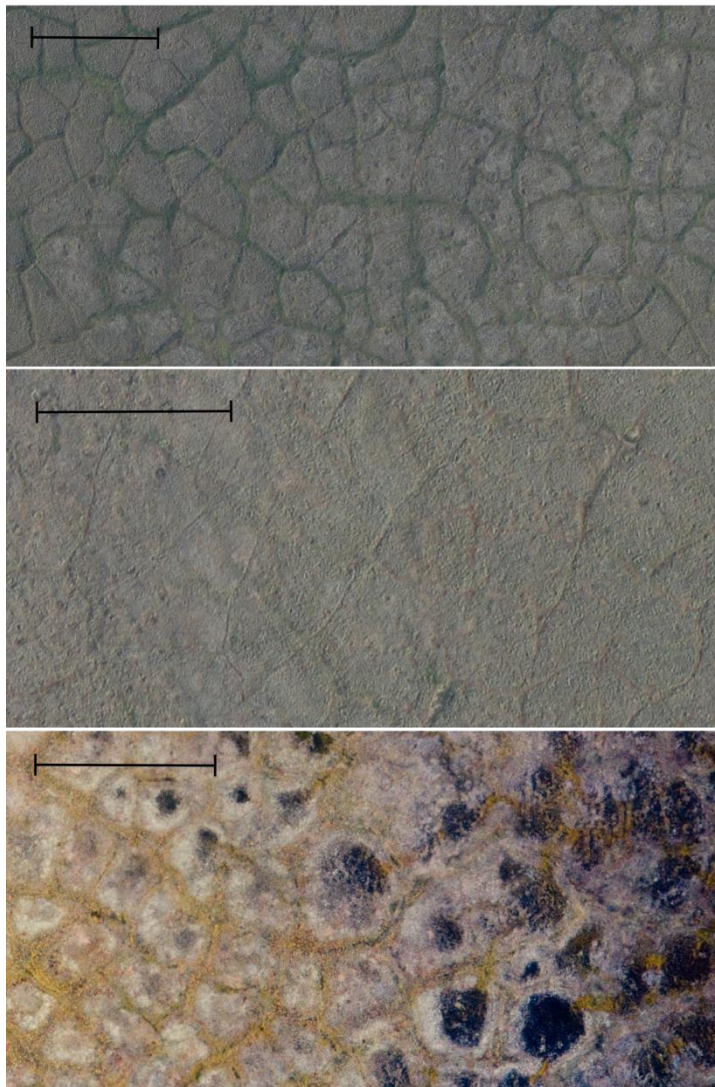


Figure D3. Examples of ice-wedge polygon types. High-center (upper photo), flat (middle photo), and low-center (lower photo). The scale bar in each photo is 50 m long.

D6. Database updates.

When all data recording for the 37 hexagonal cells is complete, import the dbf table into the database. Add new columns to the table "ARCNGrid37subplots" to accommodate the new data. Join the new data to this table on columns "ARC�_ID" and "ptid" and update the new column with your new data.

E. Stereo 3D viewing.

E1. Background.

Stereo 3D viewing is helpful to identify certain landforms and plants. In most cases, it is adequate to simply view in stereo the photographs that cover the general area of the 4-ha sample circle to familiarize oneself with the features present. If the exact location of the sample circle on the stereophotos is desired, the circle may be written on one of the stereo photographs so that it is visible when they are viewed in stereo.

E2. Viewing in Stereo

E2.1. An excellent stereo viewer, Stereo Photo Maker, is available free from <http://stereo.jpn.org/eng/stphmkr/>. This protocol was tested with Stereo Photo maker 4.3.7. Open the Stereo Photo Maker executable file (stphmkre.exe) to start. Choose File, Open Left/Right Images, and navigate to the location of your stereo photos. The two photos should appear on the screen uncropped and side-by-side; if not, select "Stereo", "Side-by-side", "Side-by-side"; an icon is available for this. Orient the photos as they would be if the aircraft flew from left to right across the screen (Fig. D4). This will typically require a rotation ("Edit, Rotate/Flip Image, Rotate to Right or Rotate to Left"; see shortcut key information on the menu), and sometimes also a left-right swap ("View, Swap Left/Right"; also available as an icon).

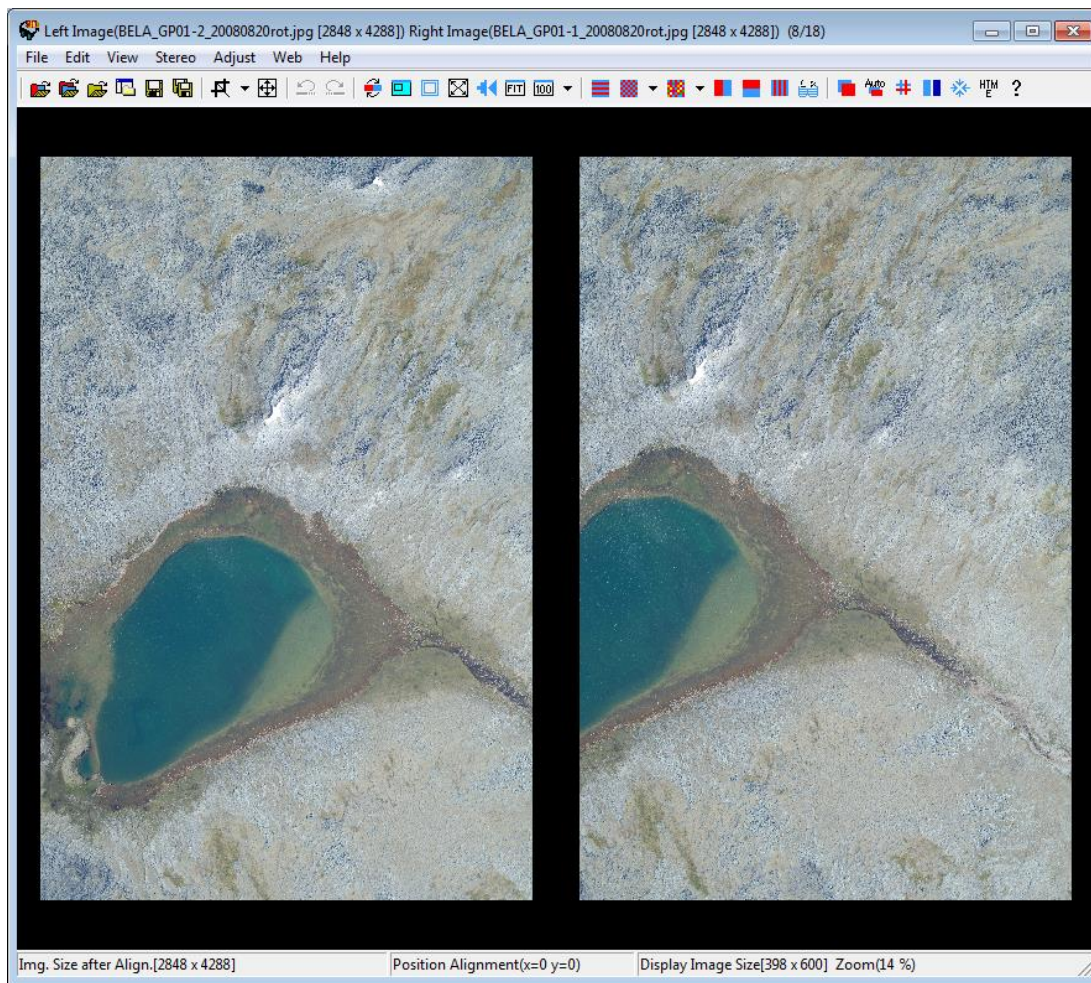


Figure D4. Arranging photos in Stereo Photo Maker. Photos should be arranged as they would be if the aircraft flew from left to right across the screen.

E2.2. Next run auto alignment (“Adjust, Auto Alignment”; also available as an icon). This will crop the photos to the overlap region, align them, and provide output parameters about the alignment (all parameters should be close to zero except for the Position V and H values, which represent simple change in position).

E2.3. If the automated alignment fails, manually align the photos: choose “Adjust, Alignment Mode” (also available as an icon), then click on a common landmark between the two photos. Sometimes after automated alignment the stereo quality in the sampling area can be improved slightly by manual alignment using a landmark in the sample area, because auto alignment is based on the entire photo.

E2.4. Next select the row interlaced view (“View, Interlaced, Row Interlaced”; also available as an icon). The two images are now superposed and can be viewed in 3 dimensions with left-right polarized glasses (the display is blurry otherwise). If the topography appears reversed, try swapping left/right again. Stereo viewing is best if the room is darkened to reduce glare.

E2.5. You may open Stereo Photo Maker a second time and set up a view of another photo date to allow comparisons.

E2.6. When exiting or loading new photos answer "no" the Stereo Photo Maker's question about saving a modified image of the previous stereo pair.

E3. Write the 4-ha circle to a photo for stereo viewing.

E3.1. If desired, the sampling circle can be written on the image in a new file and opened along with an adjacent photo in the stereo viewer. The size and location of the circle on the photo is determined from the data in the world file created by georeferencing in the previous SOP. The world file contains six numbers that are coefficient of an affine transformation in the order A, D, B, E, C, F, with one coefficient per line. These coefficients are used in the following formula:

$$x' = Ax + By + C$$

$$y' = Dx + Ey + F$$

where x' and y' are the transformed coordinates of the photo in map coordinates, x and y are the original pixel coordinates of the photo, A and E are the width of the pixels in the x and y directions (in map units), B and D are rotation terms (also in map units per pixel) and C and F are the map coordinates of the center of the lower left-hand corner pixel of the new image.

To convert from map coordinates to photo coordinates requires us to rearrange and solve for x and y :

$$x = (x' - C + B(F - y')/E)/(A - (BD/E))$$

$$y = (x' - C + A(F - y')/D)/(B - (AE/D))$$

We then use the "convert" utility from the open source graphics package ImageMagick to draw the sample circle on one photograph in the stereo pair.

E3.2. Write circles on photos for stereo viewing. The following Python program accepts as input the Alaska Albers coordinates of the plot center and the photo file name; it reads the transformation information of the associated world file for the photo (the *.jgwx world file created in ArcMap in the previous SOP), generates the 4-ha sample circle and the plot center location, and writes the result to a new file in the same directory as the original, the original file name but with "*Circle.jpg" added to the end. This file can be loaded into the stereo viewer (Stereo Photo Maker) and viewed for point sampling.

DrawCircleOnPhoto.py

```
#Python 2.7 program to print photographs with 4 ha circular plot for change
analysis
#D. Swanson 14 Feb 2012

# Import system modules
import sys, string, os

#Note: the program uses the world file generated by ArcMap geoprocessing with
*.jgwx extension
```

```

#If the good coefficients are in the standard *jgw file, either copy the file
and add the "x" or modify the line below that defines "worldfilenm"
#All numeric variables are defined as floats except those that refer to
pixels on the photos
#The latter are integers corresponding to discrete pixel numbers
#They result from the affinex and affine functions and some further
computations from these results

#Accept command line inputs:the input image and AK Albers coordinates (x,
y)of the central grid point
inimage = sys.argv[1] #input jpg image file; must have world file in same
directory
centx = float(sys.argv[2]) #x coordinate of center grid point, m AK Albers
(integer)
centy = float(sys.argv[3]) #y coordinate of center grid point, m AK Albers
(integer)

#Other parameter definitions
r4m = 112.84 #radius of 4 ha circle, m

#Derive other file names
outimage = inimage[:-4] + "Circle.jpg"
worldfilenm = inimage[:-4] + ".jgwx"

#Read the worldfile
worldfile = open(worldfilenm, 'r')
a = float(worldfile.readline())
d = float(worldfile.readline())
b = float(worldfile.readline())
e = float(worldfile.readline())
c = float(worldfile.readline())
f = float(worldfile.readline())
worldfile.close()

def affinex(x0, y0): #converts AK Albers x coordinate to image x coordinate
using affine parameters and Albers x,y
    return int(round((x0 - c + b * (f - y0) / e)/(a - (b * d / e))))

def affiney(x0, y0): #converts AK Albers y coordinate to image x coordinate
using affine parameters and Albers x,y
    return int(round((x0 - c + a * (f - y0) / d)/(b - (a * e / d))))

#Convert circle center Albers coordinates to image coordinates
icentx = affinex(centx, centy)
icenty = affiney(centx, centy)

#Compute circle radii in image coordinates using affine transformations
#on points displaced in x direction (case A) and y direction (case B) from
center

```

```

#Note the scales are slightly different in x and y direction, so we use an
average
#Numbers 1 and 4 still refer to the circle area in ha
#First the 1 ha circle
ppAx = float(affinex(centx + r4m, centy)) #x,y of perimeter point (pp)
offset r4m in x direction from the center
ppAy = float(affiney(centx + r4m, centy))
ppBx = float(affinex(centx, centy + r4m)) #x,y of perimeter point offset r4m
in y direction from the center
ppBy = float(affiney(centx, centy + r4m))
rAp = ((icentx - ppAx)**2 + (icenty - ppAy)**2)**0.5 #pythagorean theorem to
get distance from center to the offset point
rBp = ((icentx - ppBx)**2 + (icenty - ppBy)**2)**0.5
r4p = int(round((rAp + rBp)/2)) #average the two radii

#Assemble the ImageMagick command; first define some useful string variables
sp = ' '
co = ', '
qu = '"'

#Compose start of the convert command line; circles are red and 2 pixels wide
command = 'convert ' + inimage + ' -stroke red -fill none -strokewidth 2'

#Draw 4 ha circle
command = command + ' -draw "circle ' + str(icentx) + co + str(icenty) + sp
+ str(icentx) + co + str(icenty + r4p) + qu

#Draw small circle to mark center location
command = command + ' -draw "circle ' + str(icentx) + co + str(icenty) + sp +
str(icentx) + co + str(icenty + 5) + qu

#Finish out the command line
command = command + sp + outimage

#Run the convert command
os.system(command)

```

Typically we will georeference a group of photographs together and then write circles on them in a batch process. To run the above Python script for a group of photographs at once, copy the image file name, and Alaska Albers x and y coordinates for the 20-km grid point out of the attribute table for the GIs layer “ARCN20k_GridPts” into a text file with *.bat extension, e.g. “RunDrawCirclePy.bat”. For example, the following file will run the program “DrawCircleOnPhoto.py” for 5 photos with the plot circle centered at the given coordinates:

```

DrawCircleOnPhoto.py BELA_GP01-2_20080820.JPG -413985 1731160
DrawCircleOnPhoto.py BELA_GP02-2_20080820.JPG -453985 1751160
DrawCircleOnPhoto.py BELA_GP03-2_20080820.JPG -433985 1751160
DrawCircleOnPhoto.py BELA_GP04-2_20080820.JPG -413985 1751160

```

DrawCircleOnPhoto.py BELA_GP05-2_20080820.JPG -453985 1771160

Place the Python script, this batch file, and the “convert.exe” ImageMagick utility (<http://www.imagemagick.org>) into the directory where the photos are and run the batch file. The files containing circles should appear in the same directory with the same names but ending with *Circles.jpg

F. Data Analysis and Reporting

F1. Test for significance of whole plot-level changes.

Here we test for significance of the qualitative, whole 4-ha plot changes from the perspective of all of ARCN. Typically observations of "no change" dominate and we are primarily interested in whether the observed counts of "increase" and "decrease" for any given feature differ significantly. Our three change types ("increase", "decrease", and "no change") are modeled by a trinomial distribution. In a Bayesian formulation, the natural prior distribution for a trinomial variable is the Dirichlet distribution, and the posterior distribution is again the Dirichlet. Using an uninformative prior (Jeffrey's) adds 0.5 count to each category, and then the posterior distribution of the probability of each category is the counted (plus 0.5) proportion in that category. Confidence intervals on estimated probabilities, and difference in probabilities can be obtained by sampling from the posterior distribution. Variances (and confidence intervals) on posterior probabilities are estimated with 10,000 independent draws using the "rdirichlet" function in the MCMC pack (Martin et al 2012) of R statistical software (R Core Team 2014). The 95% confidence interval for the estimated probability of each outcome "increase", "no change", and "decrease"), and for the difference between "increase" and "decrease", are computed using quantiles from the 10,000 random draws from the posterior distribution. We would conclude that our observed count of "increase" plots for some feature differs from the number of "decrease" plots if the confidence interval for the difference in their estimated probabilities does not contain zero. R code that performs this test is:

```
#R 3.0 code Function to compute trinomial probabilities, by Jay Ver Hoef, 29
Feb 2012
#This requires R to have the MCMCpack package installed.
```

```
photoChangeTrinomial <-function(nDecrease, nNoChange, nIncrease) {
  require(MCMCpack)
  rdirichOut <- rdirichlet(10000,c(nDecrease + 0.5, nNoChange + 0.5,
    nIncrease + 0.5))
  ProbFits <- cbind(
    apply(rdirichOut,2,mean),
    apply(rdirichOut,2,sd),
    apply(rdirichOut,2,quantile, p = .025),
    apply(rdirichOut,2,quantile, p = .05),
    apply(rdirichOut,2,quantile, p = .975)
  )
  colnames(ProbFits) <- c("Est", "SE", "2.5%", "median", "97.5%")
  rownames(ProbFits) <- c("Pr(Decrease)", "Pr(NoChange)", "Pr(Increase)")
  DifFitsIncMinusDec <- c(
```

```

      mean(rdirichOut[,3] - rdirichOut[,1]),
      sd(rdirichOut[,3] - rdirichOut[,1]),
      quantile(rdirichOut[,3] - rdirichOut[,1],p = c(.025, .5, .975))
    )
    names(DifFitsIncMinusDec) <- c("Est", "SE", "2.5%", "median", "97.5%")
    list(ProbFits = ProbFits, DifFitsIncMinusDec = DifFitsIncMinusDec)
  }

```

To run this function, if we had results for 206 plots (the number sampled in Swanson 2013) where shrubs decreased on 2 plots, had no change on 181 plots, and increased on 23 plots, we would enter:

```
photoChangeTrinomial(nDecrease = 2, nNoChange = 181, nIncrease = 23)
```

which gives the results

```

$ProbFits
              Est              SE          2.5%          median          97.5%
Pr(Decrease) 0.01216101 0.007585356 0.002108705 0.002942116 0.03123865
Pr(NoChange) 0.87431496 0.022873137 0.826077924 0.834684309 0.91558121
Pr(Increase) 0.11352403 0.021919116 0.074761503 0.079848851 0.15984461

$DifFitsIncMinusDec
              Est              SE          2.5%          median          97.5%
0.10136301 0.02351149 0.05848592 0.10035616 0.14989795

```

In this example, the 95% confidence interval for the difference between the probability of increase and decrease (the final line of the output) is 0.058 to 0.150, which does not include zero, so we conclude that the probability of an increase is truly higher than the probability of a decrease.

We are normally interested in testing whether a particular change mechanism is important or not. For example, we may be interested in whether shrubs increased significantly on floodplains due to stream erosion and plant succession. To accomplish this, we must restrict our sample to just plots that contain floodplains; this is determined by the ecotypes present, see part D above. We also must limit ourselves to the appropriate changes types and change mechanisms (Table D2): "shrub increase" by "floodplain succession" and "shrub decrease" by "floodplain erosion". In our most recent sampling (Swanson 2013), 21 plots had floodplains, one had shrub decrease by floodplain erosion and 5 had shrub increase by floodplain succession; thus, 15 had no change in shrubs. We would run our R "photoChangeTrinomial" function with these numbers:

```
photoChangeTrinomial(nDecrease = 1, nNoChange = 15, nIncrease = 5)
```

This produces the output:

```

$ProbFits
              Est              SE          2.5%          median          97.5%
Pr(Decrease) 0.06612908 0.05089600 0.004784837 0.007958379 0.1979475
Pr(NoChange) 0.68926562 0.09532858 0.485360755 0.521578705 0.8575326
Pr(Increase) 0.24460530 0.08848320 0.095097657 0.114368654 0.4382617

```

```
$DiffFitsIncMinusDec
      Est      SE      2.5%      median      97.5%
0.17847621 0.10840582 -0.03088873 0.17617666 0.39750619
```

The 95% confidence interval for the difference between the probably of increase and decrease runs from -0.031 to 0.176, i.e. it overlaps zero. Thus we cannot conclude with 95% confidence that shrubs increased on floodplains. Shrub increases were clearly more common than decreases, but with the fairly small sample size our confidence is reduced. Note that in the definition of the "photoChangeTrinomial" R function above we could test for a less stringent 90% confidence interval: replace both occurrences of "0.025" with "0.05" and both occurrences of "0.975" with "0.95" to obtain the 90% confidence intervals (also, to prevent confusion, we should replace the corresponding labels "2.5%" and "97.5%" with "5%" and "95%"). The 90% confidence interval for the difference between increase and decrease ranges from 0.005 to 0.360, i.e. it does not overlap zero and with 90% confidence we detect a shrub increase on floodplains.

F2. Analysis of the areal cover data from the 37 hexagonal cells.

Ecotype changes between sample dates can be portrayed by cross-tabulating the total counts of each type by the two dates. An example from Swanson (2013) is given in Table D3. Of the 259 hexagonal cells of Lowland Black Spruce Forest in 2010, 85 were Lowland Birch-Ericaceous-Willow Low Shrub in 1980 and the remaining 174 were Lowland Black Spruce Forest in the preceding sample also. Thus $85/259 = 0.33$ or 33% of the Lowland Black Spruce Forest was newly created from low shrub over the 30-year period. The count of plots is included to provide a picture of how widespread the change was, since the 37 hexagonal grid cells at a plot are close together (i.e. spatially autocorrelated).

Table D3. Example cross-tabulation of ecotype change results¹

1980 Ecotype	2010 Ecotype	Lowland Birch Ericaceous- Willow Low Shrub	Lowland Black Spruce Forest
Lowland Birch Ericaceous-Willow Low Shrub		174 (19)	85 (7)
Lowland Black Spruce Forest			174 (12)
Total			259 (13)

¹Data from Swanson (2013). The numbers are counts of hexagonal sub-plots followed by 20-km grid plots in parentheses

The significance of a change in cover for a particular ecotype is determined using a paired-sample test on cover data summarized by plot. Here we are treating the plots as independent sample units

with cover by each ecotype obtained from the hexagonal cell data. For example, to determine whether the cover of tall shrub ecotypes has increased, we first query for the appropriate ecotypes (there are three: Lowland Alder-Willow Tall Shrub, Riverine Alder-Willow Tall Shrub, and Upland Alder-Willow Tall Shrub) and sum the number of cells with any of these ecotypes at each plot at a particular sample date.

```
SELECT ARCN_ID, COUNT(ptid) AS CountOfptid
FROM (
SELECT ARCN_ID, ecotypeAHAP, ptid
FROM dbo_ARCNgrid37subplots
WHERE (ecotypeAHAP Like "*tallshr"))
GROUP BY ARCN_ID;
```

Name this query "q1". Next do the same for the other sample date:

```
SELECT ARCN_ID, COUNT(ptid) AS CountOfptid
FROM (
SELECT ARCN_ID, ecotype_hires08, ptid
FROM dbo_ARCNgrid37subplots
WHERE (ecotype_hires08 LIKE "*tallshr"))
GROUP BY ARCN_ID;
```

Name this query "q2". Next create a list of all plots with data for both dates:

```
SELECT ARCN_ID
FROM (
SELECT ARCN_ID, ptid, ecotype_hires08, ecotypeAHAP
FROM dbo_ARCNgrid37subplots
WHERE ecotype_hires08 Is Not Null AND ecotypeAHAP Is Not Null)
GROUP BY ARCN_ID;
```

Name this query "q3". Now join q1, q2, and q3 together to get a table with columns for plot ID, count of cells with any tall shrub ecotype at "time 1", and count of cells with any tall shrub ecotype at "time 2":

```
SELECT q3.ARCN_ID, Nz([q1.CountOfptid],0) AS time1, Nz([q2.CountOfptid],0) AS
time2
FROM (q3 LEFT JOIN q2 ON q3.ARCN_ID = q2.ARCN_ID)
LEFT JOIN q1 ON q3.ARCN_ID = q1.ARCN_ID;
```

The "Nz" function fills zeros into blanks in the table. The result will look like this:

ARCN_ID	time1	time2
CAKR_-38_194	0	0
CAKR_-40_194	7	19
CAKR_-40_196	0	3
CAKR_-40_198	0	1
CAKR_-40_200	0	0
CAKR_-42_194	0	0

The numbers under *time1* and *time2* are counts of hexagonal cells with tall shrubs at that time. Export as text, import into R and run a Wilcoxon signed rank test to see if the counts on the two dates differ significantly. Edit the script below to match your input file path and name, and the names of your data columns assigned to *x* and *y*.

```
infile <- "O:\\Monitoring\\Vital Signs\\Terrestrial Landscape Patterns and
Dynamics\\Results\\20kmGridSampling\\ShrWilxnTest.txt"
data <- read.table(infile, header=TRUE, sep="\t", na.strings="NA", dec=".",
strip.white=TRUE)
x <- data$time1
y <- data$time2
wilcox.test(x, y, alternative = "two.sided", paired = TRUE)
```

A paired sample test such as this has more power than an unpaired test would. We use the non-parametric paired test (Wilcoxon signed rank) rather than a paired t-test because the many zeros in the data make it non-normally distributed. Many zeros are present because no change is common, and also we typically will include the many plots where the ecotype(s) were not observed on either sample date, to provide us with park-wide inference.

When three years of data are available, we can repeat the above test for any pair of years. For four or more years of data, we may want to test for trend with a linear model. Power will be improved if we treat the plots as a random effect in a general linear mixed model (Laird and Ware 1982, Hedeker 2005).

Section D Literature Cited

- ESRL Global Monitoring Division, G. R. G. 2016. "Solar Calculation Details." *Solar Calculation Details*. Accessed December 6.
<http://www.esrl.noaa.gov/gmd/grad/solcalc/calcdetails.html>.
- Hedeker, D. 2005. "Generalized Linear Mixed Models." In *Encyclopedia of Statistics in Behavioral Science*. John Wiley & Sons, Ltd. <http://bstt513.class.uic.edu/GLMM.pdf>.
- Jorgenson, M. T., J. E. Roth, P. F. Miller, M. J. Macander, M. S. Duffy, A. F. Wells, G. V. Frost, and E. R. Pullman. 2009. "An Ecological Land Survey and Landcover Map of the Arctic Network." Natural Resource Technical Report ARCN/NRTR-2009/270. Fort Collins (CO): National Park Service. <https://irma.nps.gov/App/Reference/Profile/663934>.
- Laird, N. M., and J. H. Ware. 1982. "Random-Effects Models for Longitudinal Data." *Biometrics* 38 (4): 963–74. doi:10.2307/2529876.
- Martin, A. D., K. M. Quin, and J. H. Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R." *Journal of Statistical Software* 42 (9): 1–21.
- R Core Team. 2014. "R: A Language and Environment for Statistical Computing." Vienna, Austria: R Foundation for Statistical Computing. <http://www.R-project.org>.

Swanson, D. K. 2013. “Three Decades of Landscape Change in Alaska’s Arctic National Parks: Analysis of Aerial Photographs, C. 1980–2010.” Natural Resource Technical Report NPS/ARC/NRTR—2013/668. Fort Collins (CO): National Park Service. <https://irma.nps.gov/App/Reference/Profile/2192701>.

Swanson, D. K. 2017. “Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network: Narrative.” Natural Resource Report NPS/ARC/NRR-2017/1380. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/DataStore/Reference/Profile/2238342>.

Wells, A. F., and M. J. Macander. 2010. “An IKONOS Satellite Image Library of Ecosystems in the Arctic Network of National Parks.” Natural Resource Technical Report NPS/ARC/NRTR—2010/397. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/DataStore/Reference/Profile/2166089>.

E. Standard Operating Procedures for Lake Surface Area Monitoring

These Standard Operating Procedures are used to monitor the area of surface water in lakes and ponds using Landsat satellite imagery, as described in Section 6 of the protocol narrative document (Swanson 2017).

Table E1. Deliverables schedule for analysis of surface water change by Landsat Imagery

Identifier	Deliverable	Format	Schedule	Responsibility	Description
E1	Zonal statistics summary tables	Text or dbf format	Every 5 years beginning in 2013	PI	Merged zonal statistics data files of water area by subsection
E2	Summary report	Document file (pdf)	Every 5 years beginning in 2013	PI	Either an NPS Natural Resource Report or peer-reviewed journal article

Standard Operating Procedure (SOP) # 1: Image download and preparation Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

In this procedure the Landsat image mosaics for one year covering all of ARCN are prepared for analysis. The main steps are to download the image tiles, extract band 5 and save into geotiff format, set the coordinate system, and join the images into a single image covering all of ARCN. These are the procedures that were used in Swanson (2013). Our procedure is based on use of the WELD Landsat data, which currently includes composite images from 2003 through 2012. At the present time we are not archiving the downloaded raw or water-classified images locally after data analysis is complete.

B. Download the Landsat images.

The WELD Landsat image composites are available from <http://e4ftl01.cr.usgs.gov/WELD/>. For more information, see <http://landsat.usgs.gov/WELD.php>. The Alaska seasonal products are in <http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/>, with the summer images in the directories

labeled YYYY.06.01 (i.e. the year and months starting in June. Inside this directory are the files, "Alaska.summer.YYYY.h##v##.doy###to###.v1.5.hdf" where the tile numbers are "h##v##" and "doy" gives the first a last dates of pixels in the composite. Complete coverage of ARCN requires the following tiles: h01v04, h02v03, h02v04, h03v02, h03v03, h03v04, h04v02, h04v03, h05v02, h05v03, h05v04, h06v02, h06v03. Create a folder named after each year and save all of the images there.

Downloading the WELD data requires a NASA Earthdata Login username and password. To obtain a NASA Earthdata Login account, visit <https://urs.earthdata.nasa.gov/users/new/>.

Saving each file using an Internet browser is very time-consuming. To automate the process, use the open source utility "Wget" (<http://www.gnu.org/software/wget/wget.html>). Download a Windows binary version of the program "Wget.exe" or "Wget64.exe" and place it in an empty directory. In the same directory, create a text file containing the batch commands shown below. You will have to replace "your_user_name" and "your_password" below with the user name and password that you set up with NASA Earthdata Login above. The user name and password do not need quotes.

```
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h01v04.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h02v03.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h02v04.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h03v02.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h03v03.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h03v04.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h04v02.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h04v03.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h05v02.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h05v03.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
```

```
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h05v04.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h06v02.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
wget64 --http-user=your_user_name --http-password=your_password -r -A
"Alaska.summer.????h06v03.*.hdf" -nd -np -nc
http://e4ftl01.cr.usgs.gov/WELD/WELDAKSE.001/????06.01/
```

Name the file "run_wget.bat" or similar and run. You can also run the "wget" lines one at a time from the command prompt. Wget will search through the WELD directory structure and find the summer files with the ARCN tiles (defined by the "h**v**" part of the file names in the command). The tiles are about 200 MB each so you will need almost 3 GB of storage per year of data. The options used with wget are: -r recursive download, -A accept files with names that match the pattern given, -nd no directory structure copied, -np no parent directories (i.e. don't follow links upward in the hierarchy), -nc no clobber (skip files with names that duplicate files that have been downloaded already).

Further processing is much easier to automate if the unpredictable "doy*to*" part of the file names is deleted. This can be done using the DOS rename command. Open a command prompt and navigate to the folder where all of the newly downloaded WELD composites were saved. Run these two commands:

```
ren "Alaska.summer.??????????.*.hdf.xml" "Alaska.summer.??????????.hdf.xml"
ren "Alaska.summer.??????????.*.hdf" "Alaska.summer.??????????.hdf"
```

These command also removes the "v1.5" string from the file names. Our analysis involves a new download of the data each time (i.e. we do not archive the data locally), so all further analysis will be with the latest version of the data.

C. Convert the hdf file to tif using Multispec.

This is the method used in Swanson (2013). At the present time, Arc GIS geoprocessing with Python scripts appears to be unable to access individual subdatasets (bands) in an hdf raster. So we are forced to extract the band 5 data and save it as an image (tif) for later processing. This SOP describes how to do this using an open-source software package, but it could be done manually in Arc GIS by adding each image individually to a project and using the "Copy" geoprocessing tool (under Data Management, Raster, Raster Dataset).

C1. Install and run Multispec. The public software package "Multispec" was produced by researchers at Purdue University (Landgrebe and Biehl 2011; <https://engineering.purdue.edu/~biehl/MultiSpec/index.html>). Download the self-extracting zipfile and extract. Run MultiSpecWin32z.exe. Choose "File", "Open Image", navigate to the hdf file and select it. Accept the defaults in the File Format Specifications dialog box, Display Specifications dialog box, and Histogram Specifications dialog box. Next choose "Processor", "Reformat", "Change Image File Format". In the "Set Image File Format Change Specifications" dialog box, set "Channels" to "Subset" and highlight only "5", make sure the header is "GeoTIFF format" and hit

“OK”. For the file name, choose the original file, replaced the characters from “doy” to the end with “Band5.tif”, and save.

C2. Set the spatial reference information.

Copy the following information into a file called "WELD_Alaska.prj"

```
PROJCS["WELD_Alaska",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Albers"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",-154.0],PARAMETER["Standard_Parallel_1",55.0],PARAMETER["Standard_Parallel_2",65.0],PARAMETER["Latitude_Of_Origin",50.0],UNIT["Meter",1.0]]
```

Save this file in the folder with other North American project coordinate systems. It will be similar to:

C:\Program Files\ArcGIS\Desktop10.0\Coordinate Systems\Projected Coordinate Systems\Continental\North America

Navigate to each new tif created above in Arc Catalog, right-click on the name, "Properties", scroll down to "Spatial Reference", click on "Edit", "Select" (select a pre-defined coordinate system"), then navigate the directories list above (Projected Coordinate Systems\Continental\North America) and select WELD_Alaska. This will re-write the Geotif headers with the correct coordinate information.

You can now add the image to an ArcMap project if desired. To improve the display, right-click the image in the ArcMap table of contents, choose "Symbology", and in the resulting dialog box choose "Stretched" and Stretch Type "Standard Deviations", answer Yes to the question about computing statistics.

Standard Operating Procedure (SOP) # 2: Water body mapping using Landsat Band 5 Version 1.0 (Jan 2017)

Revision History Log:

Version No.	Revision Date	Author	Changes Made	Reason for Change

A. Background

Change in surface water is detected using the Landsat middle infrared band 5. This band has low values for water (<250) and high values for all other surface types (>1000) except terrain shadow. Thus, we can make a good estimate of the area of surface water by summing the area of image pixels with values below a chosen threshold (e.g., 600) and screening out mountainous areas with terrain shadows and very few lakes. Because of problems with image quality (clouds, and banding due to the Landsat 7 malfunction), a water area map computed for each individual year would contain numerous errors. To reduce these errors, for a particular year we compute the median value for each pixel across 3 consecutive years (the current year, the previous year, and the following year) and then classify as water those pixels with median values less than 600. The median greatly reduces the effects of a single year of erroneous high band 5 values, at the cost of some loss of ability to resolve year-to-year variation.

The procedures in this SOP are based on ArcGIS geoprocessing and they were used in Swanson (2013). All of the computations could be also implemented with R using the "raster" package (Hijmans 2015). This might improve efficiency and should be considered in future revisions of this SOP.

B. Computations

B1. Classify pixels as water or land.

The ArcMap “Cell Statistics” geoprocessing tool is used (under Spatial Analyst Tools, Local) to compute the median value for band 5 in each pixel across 3 years of images. The “Con” function is then used to classify as water those pixels with median values of less than 600. This class is then assigned to an image named after the middle year of the 3, where “1” signifies water and “0” land. The following Python script runs these functions. Edit path and file names and the 3 years as needed.

```
#Script to compute median Landsat band 5 value for running 3-yr window
#using WELD Landsat composites, then classify result as water 1 or land 0
#Python 2.7, ArcGIS 10.2
```

```
# Import system modules
```

```

import arcpy
from arcpy import env
from arcpy.sa import *

# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")

outpath = "C:/GIS/GISdataLocalBackup/Landsat/Analysis/3yrMedian600"
arcpy.env.workspace = outpath
arcpy.env.overwriteOutput = "True"

year1 = "2009"
year2 = "2010"
year3 = "2011"

scenes =
["h01v04","h02v03","h02v04","h03v02","h03v03","h03v04","h04v02","h04v03","h05
v02","h05v03","h05v04","h06v02","h06v03"]

for scene in scenes:
    print scene #to track progress
    inpath1 = "C:/GIS/GISdataLocalBackup/Landsat/" + year1 + "/"
    inpath2 = "C:/GIS/GISdataLocalBackup/Landsat/" + year2 + "/"
    inpath3 = "C:/GIS/GISdataLocalBackup/Landsat/" + year3 + "/"
    inRaster1 = inpath1 + "Alaska.summer." + year1 + "." + scene +
"band5.tif"
    inRaster2 = inpath2 + "Alaska.summer." + year2 + "." + scene +
"band5.tif"
    inRaster3 = inpath3 + "Alaska.summer." + year3 + "." + scene +
"band5.tif"

    # Compute median band 5 value for 3 year window
    median = CellStatistics([inRaster1, inRaster2, inRaster3], "MEDIAN",
"DATA")

    #threshold the median raster to get water vs land, then save result
    water = Con(median, "1", "0", "VALUE < 600")
    outfile = outpath + "/" + year2 + "water3yr" + scene + ".tif"
    water.save(outfile)

```

B2. Merge files.

Further processing is easier if these scenes are all joined into one mosaic. The following Python code accomplishes this. Edit the path and years as needed. The script searches for images with names composed according to the conventions in the previous script.

```

#Script to mosaic WELD 3yr water classified images into one image
#that covers all of ARCN

```

```

# Import system modules
import arcpy
from arcpy import env
from arcpy.sa import *

# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")

path = "C:/GIS/GISdataLocalBackup/Landsat/Analysis/3yrMedian600"
arcpy.env.workspace = path
arcpy.env.overwriteOutput = "True"

years = ["2004", "2005", "2006", "2007", "2008", "2009", "2010"]
scenes =
["h01v04", "h02v03", "h02v04", "h03v02", "h03v03", "h03v04", "h04v02", "h04v03", "h05
v02", "h05v03", "h05v04", "h06v02", "h06v03"]

for year in years:
    infiles = "" #initialize list of infiles
    for scene in scenes:
        print year, scene #to track progress
        infiles = infiles + year + "water3yr" + scene + ".tif;"

    # Compute median band 5 value for 3 year window
    outfile = year + "water3yr.tif"
    arcpy.MosaicToNewRaster_management(infiles[0:-1], path, outfile, "",
"8_BIT_UNSIGNED", "", "1", "MAXIMUM", "")

    print infiles
    print outfile

```

B3. Summarize water area by Subsection.

Subsections are useful units to summarize water area. Mountainous subsections have significant areas of terrain shadow (which is mistakenly classified as water due to low values for all color bands, including band 5). Fortunately, these subsections contain very little surface water and thus can be screened out of the analysis with little effect on the results. Lowland subsections dominated by marine water bodies or rivers are also screened out; rivers vary in size greatly through the year and should be avoided to the degree possible. We use the subsections chosen in Swanson (2013). For ease of processing these selected subsections have been extracted out from the others and merged into a single layer "ARCNlakeZones.shp", saved in the project folder on the network. The ArcMap "Zonal Statistics as Table" geoprocessing tool is used with the SUM option to sum the number of water pixels in each ecological subsection (since each water pixel has a value of 1, the sum will provide a count of water pixels). Use the field "PARK_SUBS" (concatenation of NPS unit and subsection code) as the zone field. Run this for each year. Before merging the yearly tables, you will need to add a year identifier to each record. The existing column COUNT will accommodate a 4 digit year and is not needed. You may automate this task using the "Calculate Field" geoprocessing tool

(Data Management Tools, Fields). After the year has been added to each record, merge the resulting tables using the Merge geoprocessing command in ArcMap (Data Management Tools, General). Convert the sum of water pixels in each zone to area in square kilometers. For 30 m Landsat TM data, the conversion is: (sum of pixels in zone) x 900 m²/pixel x 10⁻⁶ km²/m². Look for trends in water for each subsection by plotting and regressing the water area in each subsection vs. year.

B4. Identify lake drainage events.

Update the Appendix Table 2 from Swanson (2013) that lists major lake drainage or re-filling/expansion events. A good way to find these are to run a linear regression on a pixel-by-pixel basis for the water/land classified images over the period since the last analysis was run. Positive slope indicates an area that went from land (0) to water (1) and negative slope from water (1) to land (0). Don't be concerned about significance of the regression, since we are simply using regression to identify areas that will then be verified by inspection of the actual images to determine the exact year of drainage.

One option is to use the ArcMap extension "Curve Fit" by the USGS Upper Midwest Environmental Sciences Center (http://www.umesc.usgs.gov/management/dss/curve_fit.html), which takes a time series of images and fits a curve to each pixel (see the SOP in Section B above for instructions on the use of "Curve Fit". Unfortunately, at the resolution of Landsat data, "Curve Fit" will not complete a run over our study area in a suitable length of time. As an alternative, the following Python script computes the regression slope of 5 rasters and saves the output as "slope.tif".

```
## Python script to compute slope and regression r for single band tif images
in a time series
#Python 2.7, ArcGIS 10.2
## -----
-

## Import system modules
import arcpy
from arcpy import env
from arcpy.sa import *
arcpy.CheckOutExtension("spatial")

##-----User Input Section-----
-

# Be sure that paths are double-slashed and end with double-slash
inpath = "C:\\GIS\\GISdata\\LakeHistory\\WELD3yrMedian600\\"
outpath = "C:\\GIS\\GISdata\\LakeHistory\\WELD3yrMedian600\\"
n = 5 #number of images (years of data); if this changes need to modify sy,
syy, sxy below to accomodate r6, r7, etc.

# list the rasters to be analyzed; include the path starting from "inpath"
defined above
r1 = Raster(inpath + "spr2002median.tif")
r2 = Raster(inpath + "spr2003median.tif")
```

```

r3 = Raster(inpath + "spr2004median.tif")
r4 = Raster(inpath + "spr2005median.tif")
r5 = Raster(inpath + "spr2006median.tif")

##-----End of User Inputs-----
-
env.workspace = outpath #sets the location for any temp files generated

# Sums used to compute slope and correlation coefficient r
sx = float(sum(range(1,n+1))) #number: sum of x values, which are assumed
to be consecutive years so simplified to be 1 to n
sxx = sum(float(i)**2 for i in range(1,n+1)) #number: sum of x^2
sy = r1 + r2 + r3 + r4 + r5 #raster: sum of y, the pixel values through the
years
syy = r1**2 + r2**2 + r3**2 + r4**2 + r5**2 #raster: sum of y^2
sxy = r1 + 2*r2 + 3*r3 + 4*r4 + 5*r5 #raster: sum of x*y

num = n*sxy - sx*sy #raster: numerator in slope and rsq equations
den1 = n*sxx - sx**2 #number: denominator in slope equation and part of the
den in rsq equation

slope = num/den1
#rxy = num/((den1*(n*syy-(sy**2)))**0.5)

slope.save("slopetest.tif")
#rxy.save("rxytest.tif")

```

Note: the lines near the end that begin with "#rxy" compute rasters with the correlation coefficient "; they are inactivated by comment signs because they are not needed here to and more than double the computation time, but by removing the comment sign they may be used in other applications.

Display the output "slope.tif" using a stretch color scheme where positive and negative values differ greatly. Negative slopes indicate water area loss (lake drainage) during the 5-year period and positive slope indicate water area gain. Verify that these areas actually represent lake drainage or filling events by inspecting the images from the individual years, and identify what year the effect occurred in. Display the National Hydrographic Dataset (USGS NHD 2014) and find the NHD code for lake involved. Enter this information into the database.

If you are familiar with R statistical software, the raster package (Hijmans 2015) may be used to run a linear regression or a Mann-Kendall trend test. The Mann-Kendall test is a non-parametric test for monotonic trend of any shape that is described on Section A, SOP #1 step D. Computer processing speed may be an issue with a Landsat image covering all of ARCN.

Section E Literature Cited

Hijmans, R. J. 2015. *Raster: Geographic Data Analysis and Modeling* (version R package version 2.3-40). <http://CRAN.R-project.org/package=raster>.

- Landgrebe, D., and L. Biehl. 2011. "An Introduction & Reference For MultiSpec, Version 9.2011." School of Electrical and Computer Engineering, Purdue University. <https://engineering.purdue.edu/~biehl/MultiSpec/index.html>.
- Swanson, D. K. 2013. "Surface Water Area Change in the Arctic Network of National Parks, Alaska, 1985-2011: Analysis of Landsat Data." Natural Resource Data Series NPS/ARC/NRDS—2013/445. Fort Collins, Colorado. <https://irma.nps.gov/App/Reference/Profile/2193078>.
- Swanson, D. K. 2017. "Landscape Patterns and Dynamics Monitoring Protocol for the Arctic Network: Narrative." Natural Resource Report NPS/ARC/NRR-2017/1380. Fort Collins, Colorado: National Park Service. <https://irma.nps.gov/DataStore/Reference/Profile/2238342>.
- USGS NHD. 2014. "Hydrography: National Hydrography Dataset, Watershed Boundary Dataset." U.S. Geological Survey. <http://nhd.usgs.gov/>.

The Department of the Interior protects and manages the nation's natural resources and cultural heritage; provides scientific and other information about those resources; and honors its special responsibilities to American Indians, Alaska Natives, and affiliated Island Communities.

NPS 953/136177, February 2017

National Park Service
U.S. Department of the Interior



Natural Resource Program Center
1201 Oakridge Drive, Suite 150
Fort Collins, CO 80525

www.nature.nps.gov