Geodiv Vignette

Sydne Record, Annie Smith, Phoebe Zarnetske, Kyla Dahlin, Jennifer Costanza, and Adam Wilson

July 22, 2020

## *geodiv* R package vignette

*geodiv* is an R package that provides methods for calculating gradient surface metrics for continuous analysis of landscape features. There are a couple of ways to download and install the *geodiv* R package. You can install the released version of *geodiv* from CRAN with:

install.packages("geodiv")

And the development version from GitHub with:

install.packages("devtools")  
devtools::install\_github("bioXgeo/geodiv")

Note that Mac OS users may need to install the development tools here to get the package to install:

<https://cran.r-project.org/bin/macosx/tools/>

To begin, let’s load the necessary packages for the examples that follow.

library(geodiv)  
library(raster)  
library(mapdata)  
library(maptools)  
library(rgeos)  
library(ggplot2)  
library(tidyverse)  
library(parallel)  
library(sf)  
library(rasterVis)  
library(ggmap)  
library(corrplot)  
library(gridExtra)  
library(cowplot)  
library(factoextra)  
library(cluster)

# Example 1: Simple workflow with Landsat NDVI

The National Aeronautics and Space Administration (NASA) has been collecting Earth observing images with Landsat for decades. Normalized Difference Vegetation Index (NDVI) is a measure of vegetation or greeness that can be quantified from Landsat images by measuring the difference between near-infrared and red light, which vegetation strongly reflects and absorbs, respectively. For this first example, we generated an NDVI image over a small region in southwestern Oregon using Google Earth Engine. This image is available as a raster layer available as an R data object entitled ‘orforest’ when *geodiv* is installed. Google Earth Engine already has a number of tutorials and sample code and functions for this step that can be accessed on its site: earthengine.google.com. Given that the focus of this tutorial is on the use of the *geodiv* R package, we provide these data already generated in Google Earth Engine.

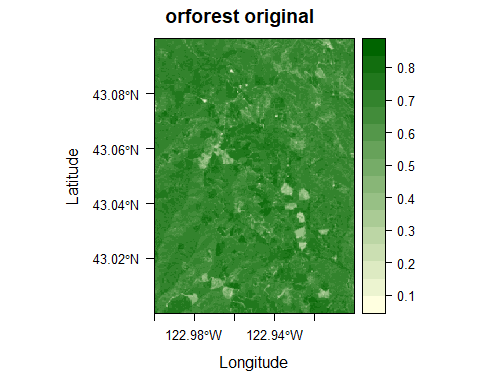
Let’s begin by loading the example data, which is a raster layer called ‘orforest’.

# Load the orforest data into your active session.  
data(orforest)  
# check out the properties of the orforest raster layer.  
orforest

## class : RasterLayer   
## dimensions : 371, 371, 137641 (nrow, ncol, ncell)  
## resolution : 0.0002694946, 0.0002694946 (x, y)  
## extent : -123, -122.9, 43.00002, 43.1 (xmin, xmax, ymin, ymax)  
## crs : +proj=longlat +datum=WGS84 +no\_defs +ellps=WGS84 +towgs84=0,0,0   
## source : memory  
## names : summer\_ndvi\_p45\_r30\_2000\_2016\_30m   
## values : 0.09595944, 0.8357643 (min, max)

Let’s first plot the data without any trends removed.

# plot without trend removed  
eviCols <- colorRampPalette(c('lightyellow1', 'darkgreen'))(100)  
eviTheme <- rasterTheme(region = eviCols)  
orig\_ndvi <- levelplot(orforest, margin = F, par.settings = eviTheme, xlab = 'Longitude', ylab = 'Latitude', main='orforest original')  
orig\_ndvi

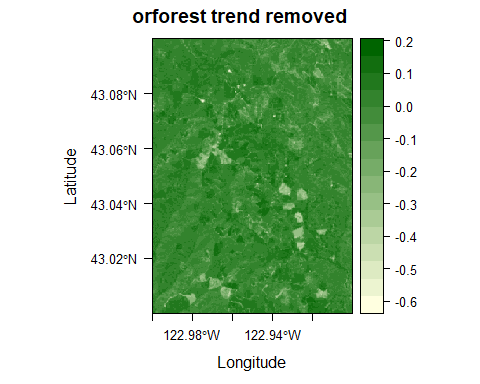


Use the ‘remove\_plane’ function of geodiv to remove the best fit polynomial plane from the image.

# plot with trend removed   
orfor\_rem <- remove\_plane(orforest)

## [1] "Order of polynomial that minimizes errors: 0"

orig\_rem\_plot <- levelplot(orfor\_rem, margin = F, par.settings = eviTheme, xlab = 'Longitude', ylab = 'Latitude', main='orforest trend removed')  
orig\_rem\_plot



Let’s also plot the raw satellite image (as opposed to the NDVI extracted from the Landsat image).

# Plot actual satellite image.  
lon <- mean(coordinates(orforest)[,1], na.rm = TRUE)  
lat <- mean(coordinates(orforest)[,2], na.rm = TRUE)  
api\_key <- NA # ADD AN API KEY HERE

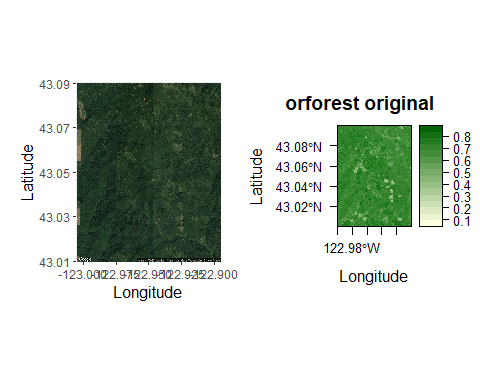
register\_google(api\_key)  
img <- get\_map(c(lon, lat), zoom = 13, maptype = 'satellite', source = "google")

## Source : https://maps.googleapis.com/maps/api/staticmap?center=43.050008,-122.950032&zoom=13&size=640x640&scale=2&maptype=satellite&language=en-EN&key=xxx-rlb90JyZ5xxnJYcNQIdKM

bbpoly <- data.frame(x = c(-122.9, -122.9, -123.00002, -123.00002), y = c(43.00002, 43.1, 43.1, 43.00002))  
satimg <- ggmap(img) + coord\_fixed(1.7) + xlab('Longitude') + ylab('Latitude') +  
 theme(axis.title.x = element\_text(size = 12),  
 axis.title.y = element\_text(size = 12),  
 plot.margin = unit(c(0,0,0,0), "cm"))

## Coordinate system already present. Adding new coordinate system, which will replace the existing one.

plot\_grid(satimg, orig\_ndvi, ncol = 2, nrow = 1, scale = c(0.84, 1))



### Calculate some surface gradient metrics for the orforest raster image.

The ‘sa’ function calculates the average roughness of a surface as the absolute deviation of surface heights from the mean surface height. The ‘sbi’ function calculates the surface bearing index, which is the ratio of the root mean square roughness (Sq) to height at 5% of bearing area (z05). The ‘std’ function calculates texture direction metrics (i.e., the angle of dominating texture and the texture direction of the Fourier spectrum image calculated from the orforest image). The ‘sfd’ function calculates the fractal dimension of the raster (i.e., the 3D fractal dimension using the triangular prism surface method).

sa <- sa(orforest)  
sbi <- sbi(orforest)  
std <- std(orforest, plot = FALSE)[[1]]  
sfd <- sfd(orforest)

## [1] "Warning: Raster will be converted to matrix format."

### Generate texture images for the orforest raster image.

Note that this step can be computationally intensive, so there is an option for running the code in parallel. Also, below this chunk we provide some code to read in files that contain the output rasters of these lines of code in case it is helpful for you.

sa\_img <- texture\_image(orforest, window\_type = 'square', size = 7, in\_meters = FALSE, epsg\_proj = 5070, metric = 'sa', args = NULL, parallel=FALSE)  
sbi\_img <- texture\_image(orforest, window\_type = 'square', size = 7, in\_meters = FALSE, epsg\_proj = 5070, metric = 'sbi', args = NULL)  
std\_img <- texture\_image(orforest, window\_type = 'square', size = 7, in\_meters = FALSE, epsg\_proj = 5070, metric = 'std', args = NULL)  
sfd\_img <- texture\_image(orforest, window\_type = 'square', size = 7, in\_meters = FALSE, epsg\_proj = 5070, metric = 'sfd', args = NULL)

Note that generating the texture images can take a while on a single processor, so we have provided the rasters generated for the following objects in case that is helpful: sa\_img, sbi\_img, std\_img, and sfd\_img. The code below reads these provided rasters into your R session. Note that you will need to ensure that the working directory for your session is set to the same directory where you have put these downloaded files if you do not run the texture image code yourself.

# The below line of code will need to be modified to match the directory on the user's machine that contains the vignette files if the texture\_image function has not been run in completion during the user's current session.  
sa\_img <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv\_paper/data/sa\_img\_450m.TIF')  
sbi\_img <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv\_paper/data/sbi\_img\_450m.TIF')  
std\_img <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv\_paper/data/std\_img\_450m.TIF')  
sfd\_img <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv\_paper/data/sfd\_img\_450m.TIF')

Now reproject the rasters back to their original projection.

# reproject back to original  
sa\_reproj <- projectRaster(sa\_img, orforest)  
sbi\_reproj <- projectRaster(sbi\_img, orforest)  
std\_reproj <- projectRaster(std\_img[[1]], orforest)  
sfd\_reproj <- projectRaster(sfd\_img, orforest)

Plot the reprojected texture image rasters.

# plots  
sa\_plt <- levelplot(sa\_reproj, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'Sa')  
sbi\_plt <- levelplot(sbi\_reproj, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'Sbi')  
std\_plt <- levelplot(std\_reproj, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'Std')  
sfd\_plt <- levelplot(sfd\_reproj, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'Sfd')  
  
# export as one  
grid.arrange(sa\_plt, sbi\_plt, std\_plt, sfd\_plt, nrow = 2)

## Example 2: Applying all surface metrics across the entire state of Oregon, USA

### Motivation

By assessing heterogeneity using a variety of metrics, researchers can gain a more complete picture of heterogeneity than they would with a single metric (Dahlin, 2016). To demonstrate the utility of geodiv for this common application, in Example 2 we apply all surface metric functions to images across Oregon, USA and examine the patterns of, and relationships among, metrics. We will calculate metrics for both elevation data from the Shuttle Radar Topography Mission (SRTM) and a commonly-used measure of canopy greenness, Enhanced Vegetation Index (EVI) from NASA’s Moderate Resolution Imaging Spectroradiometer (MODIS). We then examine the correlations among metrics along a transect crossing the state, and determined how the metrics cluster using two methods, hierarchical clustering and Principal Components Analysis (PCA).

### Elevation and EVI Data

Elevation and EVI data that are available to use along with this vignette were prepared in Google Earth Engine (Gorelick et al., 2017) and analyzed in R. Void-filled SRTM data (Farr et al., 2007) and quality-filtered, maximum growing season, MODIS EVI data (Didan, 2015) were downloaded in Fall 2019. The code chunk below loads these data into the current R session.

# The below line of code will need to be modified to match the directory on the user's machine that contains the vignette files for this code chunk to run.  
elev <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv/oregon\_SRTM\_elevation\_2000.tif')  
evi <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv/oregon\_max\_growing\_season\_EVI\_2000\_2019.tif') \* 0.0001

Aggregate both datasets to ~2km resolution for comparison between datasets and to reduce computational time.

elev <- aggregate(elev, fact = 8)  
evi <- aggregate(evi, fact = 8)

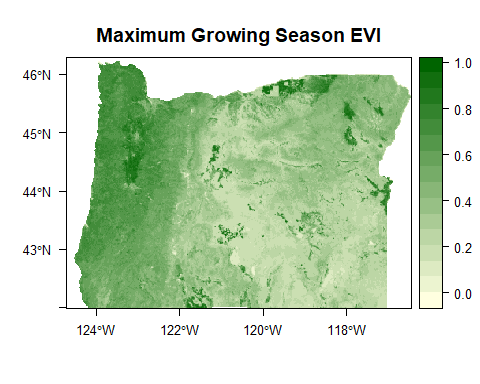
### Pre-processing of rasters

Begin by masking any values that are outside of the boundaries for the state of Oregon.

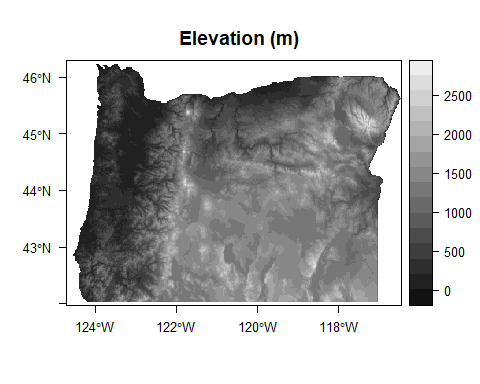
state <- maps::map(database = 'state', regions = 'oregon', fill = TRUE, plot = FALSE)  
statePoly <- map2SpatialPolygons(state, IDs = state$names, proj4string = CRS(proj4string(evi)))  
evi <- mask(x = evi, mask = statePoly)  
elev <- mask(x = elev, mask = statePoly)

Generate some plots to get a sense for the spatial patterns in the data.

# plot maximum growing season EVI for Oregon  
levelplot(evi, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'Maximum Growing Season EVI')



# plot elevation (in meters) for Oregon  
elevCols <- colorRampPalette(c('grey7', 'grey93'))(100)  
elevTheme <- rasterTheme(region = elevCols)  
levelplot(elev, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'Elevation (m)')



Remove any trends in the data with the ‘remove\_plane’ function and take a look at the plots of the images with the trends removed.

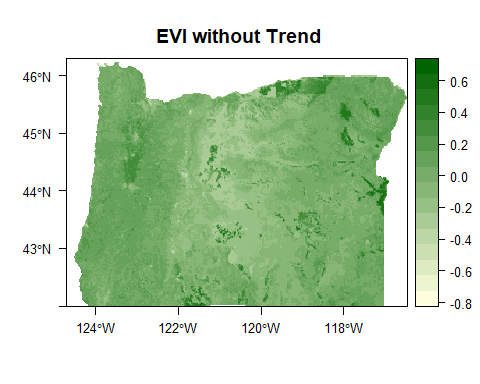
evi <- remove\_plane(evi)

## [1] "Order of polynomial that minimizes errors: 1"

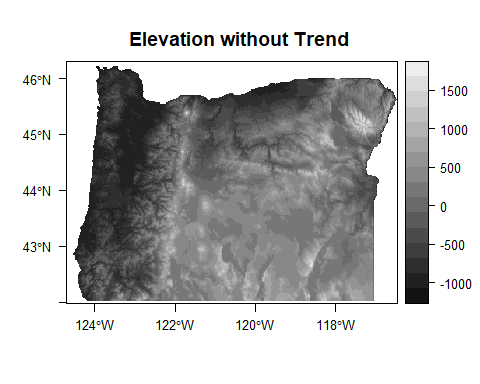
elev <- remove\_plane(elev) # there was no trend

## [1] "Order of polynomial that minimizes errors: 0"

# plot again to see what the new raster looks like  
levelplot(evi, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'EVI without Trend')



levelplot(elev, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'Elevation without Trend')



### Generate texture images of the state of Oregon

Below we generate a texture image for the state of Oregon using the sa metric for elevation. Note that the following step is computationally intensive and may take some time. We provide the output dataframe files along with this vignette for all metrics included in geodiv for elevation and EVI over ~30km x 30km square moving windows and scaled following calculation for the subsequent analyses in this vignette.

# Calculate elevation sa texture image for state of Oregon  
outrast <- texture\_image(elev, window\_type = 'square', size = 7, in\_meters = FALSE, epsg\_proj = 5070, metric = 'sa')

The code below illustrates how to convert the raster generated by the texture\_image function (i.e., outrast) to a dataframe for subsequent analyses.

# get raster values  
vals <- outrast[]  
  
# Convert output raster of sa metrics (outrast) to a dataframe for easier use in subsequent analyses  
sa\_data\_elev <- data.frame(x = coordinates(outrast)[,1], y = coordinates(outrast)[,2], v = vals)

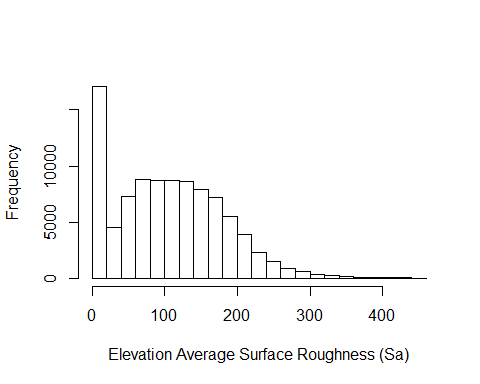
Generating these data frames can take a while, so we have provided .csv files for all gradient surface metrics calculated for elevation and EVI in case you find them useful for working with this vignette. The below code reads in these provided .csv files.

# Read in dataframes containing state of Oregon texture image outputs for EVI and elevation.  
data\_evi <- read.csv('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv\_paper/data/evi\_variables.csv', stringsAsFactors = FALSE)  
data\_elev <- read.csv('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv\_paper/data/elevation\_variables.csv', stringsAsFactors = FALSE)

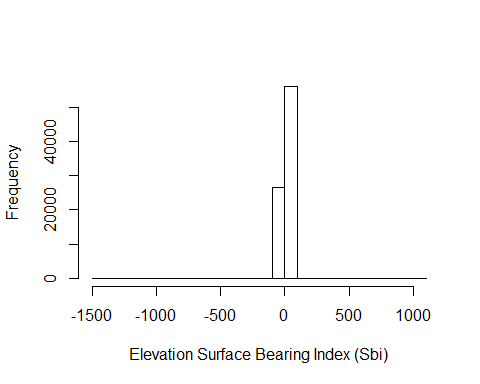
### Visualization of texture image outputs for elevation and EVI for the state of Oregon

Distributions of a few elevation variables:

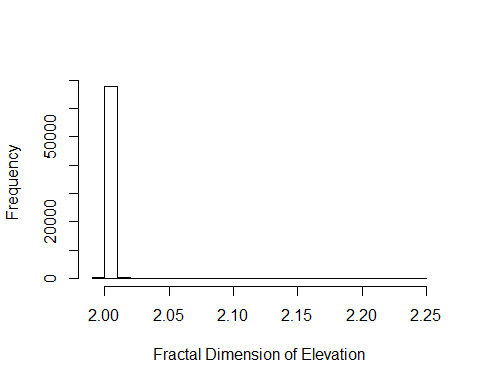
# Histogram of average roughness of a surface (Sa) of elevation  
hist(data\_elev$sa, breaks = 30, xlab='Elevation Average Surface Roughness (Sa)', main='')



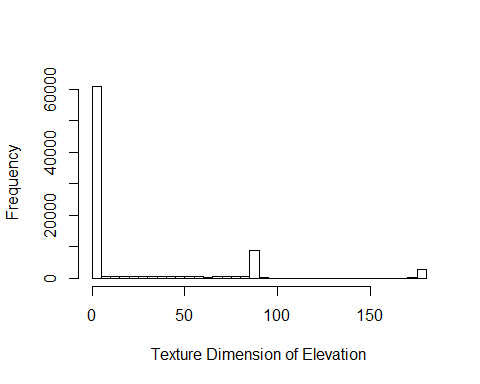
# Histogram of average roughness of the elevation surface (Sa)   
hist(data\_elev$sbi, breaks = 30, xlab='Elevation Surface Bearing Index (Sbi)', main='')



# Histogram of fractal dimension of the elevation raster   
hist(data\_elev$sfd, breaks = 30, xlab='Fractal Dimension of Elevation', main='')



# Histogram of texture dimension of elevation  
hist(data\_elev$std, breaks = 30, xlab='Texture Dimension of Elevation', main='')



The code below visualizes metrics over the entire state in order to capture different aspects of landscape heterogeneity. Individual metrics primarily distinguished mountainous versus flat terrain, and managed versus more natural areas; however, some metrics are difficult to interpret, or do not show very much variation over the region. The difficulty of interpreting metrics is a known complicating factor for their use. Others have addressed this issue and linked metrics with known ecosystem features or patch metrics (McGarigal et al., 2009; Kedron et al., 2018).

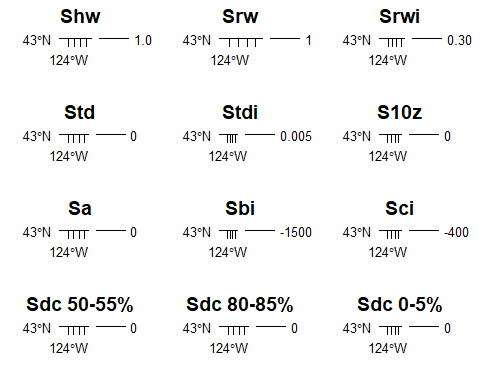
# Create plots of all possible surface gradient metrics that geodiv calculates for elevation and EVI.  
elev\_maps <- list()  
elev\_names <- c('Shw', 'Srw', 'Srwi', 'Std', 'Stdi', 'S10z', 'Sa', 'Sbi', 'Sci', 'Sdc 50-55%', 'Sdc 80-85%', 'Sdc 0-5%', 'Sdq6',   
 'Sdr', 'Sds', 'Sfd', 'Sk', 'Sku', 'Smean', 'Sp', 'Spk', 'Sq', 'Ssc', 'Ssk', 'Sv', 'Svi', 'Svk')  
for (i in seq(3, ncol(data\_elev))) {  
 cat('Creating map for var: ', names(data\_elev)[i], '\n', sep = '')  
 temp <- setValues(elev, data\_elev[,i])  
 temp[is.na(elev)] <- NA  
 goodname <- elev\_names[i-2]  
 elev\_maps[[i - 2]] <- levelplot(temp, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = goodname)  
 elev\_maps[[i-2]]$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
 elev\_maps[[i-2]]$aspect.fill <- TRUE  
}

## Creating map for var: shw  
## Creating map for var: srw  
## Creating map for var: srwi  
## Creating map for var: std  
## Creating map for var: stdi  
## Creating map for var: s10z  
## Creating map for var: sa  
## Creating map for var: sbi  
## Creating map for var: sci  
## Creating map for var: sdc05055  
## Creating map for var: sdc08085  
## Creating map for var: sdc0005  
## Creating map for var: sdq6  
## Creating map for var: sdr  
## Creating map for var: sds  
## Creating map for var: sfd  
## Creating map for var: sk  
## Creating map for var: sku  
## Creating map for var: smean  
## Creating map for var: sph  
## Creating map for var: spk  
## Creating map for var: sq  
## Creating map for var: ssc  
## Creating map for var: ssk  
## Creating map for var: sv  
## Creating map for var: svi  
## Creating map for var: svk

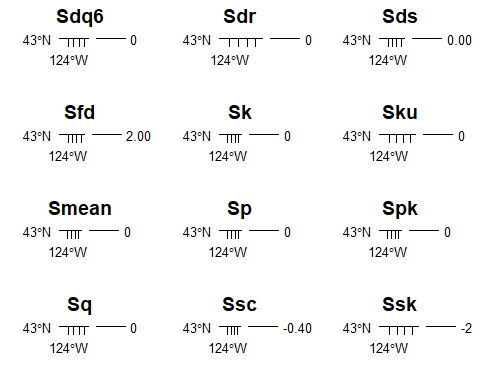
evi\_maps <- list()  
evi\_names <- c('S10z', 'Sa', 'Sbi', 'Sci', 'Sdc 50-55%', 'Sdc 80-85%', 'Sdc 0-5%', 'Sdq6', 'Sdr', 'Sds', 'Sfd', 'Shw', 'Sk', 'Sku', 'Smean',   
 'Sp', 'Spk', 'Sq', 'Srw', 'Srwi', 'Ssc', 'Ssk', 'Std', 'Stdi', 'Sv', 'Svi', 'Svk')  
for (i in seq(3, ncol(data\_evi))) {  
 cat('Creating map for var: ', names(data\_evi)[i], '\n', sep = '')  
 temp <- setValues(evi, data\_evi[,i])  
 temp[is.na(evi)] <- NA  
 goodname <- evi\_names[i-2]  
 evi\_maps[[i - 2]] <- levelplot(temp, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = goodname)  
 evi\_maps[[i-2]]$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
 evi\_maps[[i-2]]$aspect.fill <- TRUE  
}

## Creating map for var: s10z  
## Creating map for var: sa  
## Creating map for var: sbi  
## Creating map for var: sci  
## Creating map for var: sdc05055  
## Creating map for var: sdc08085  
## Creating map for var: sdc0005  
## Creating map for var: sdq6  
## Creating map for var: sdr  
## Creating map for var: sds  
## Creating map for var: sfd  
## Creating map for var: shw  
## Creating map for var: sk  
## Creating map for var: sku  
## Creating map for var: smean  
## Creating map for var: sph  
## Creating map for var: spk  
## Creating map for var: sq  
## Creating map for var: srw  
## Creating map for var: srwi  
## Creating map for var: ssc  
## Creating map for var: ssk  
## Creating map for var: std  
## Creating map for var: stdi  
## Creating map for var: sv  
## Creating map for var: svi  
## Creating map for var: svk

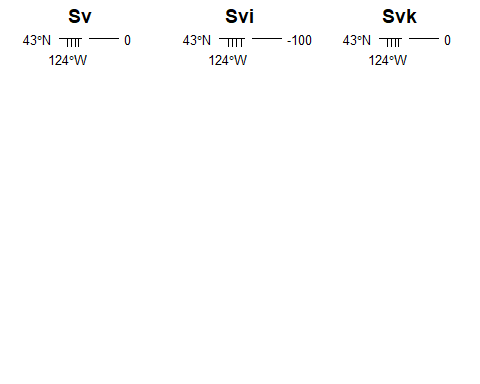
# create 3x4 panels (x3 for each)  
grid.arrange(grobs = elev\_maps[1:12], nrow = 4, ncol = 3) # 850x800



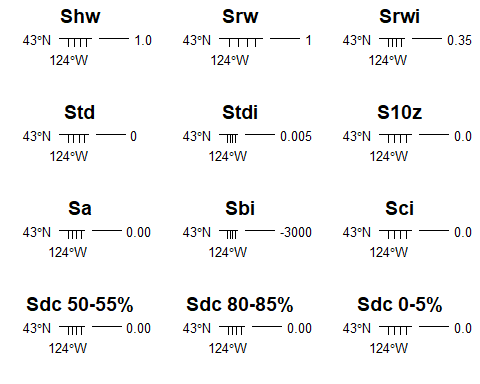
grid.arrange(grobs = elev\_maps[13:24], nrow = 4, ncol = 3) # 850x800



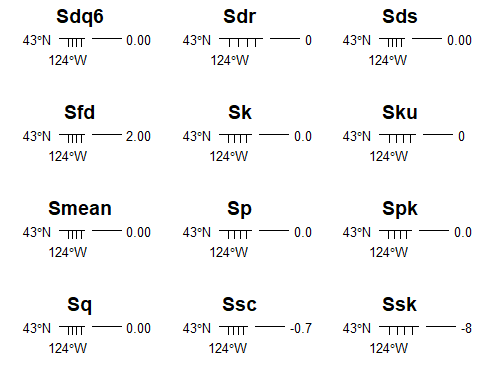
grid.arrange(grobs = elev\_maps[25:27], nrow = 4, ncol = 3) # 850x800



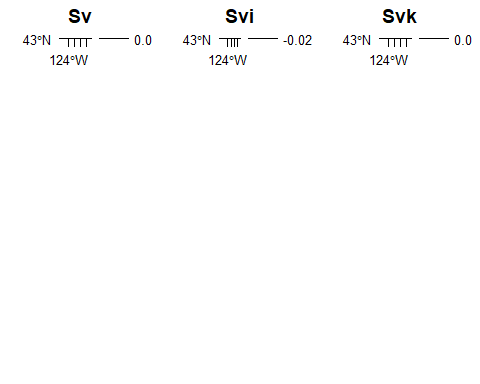
grid.arrange(grobs = evi\_maps[c(12, 19, 20, 23, 24, seq(1,7))], nrow = 4, ncol = 3) # 850x800



grid.arrange(grobs = evi\_maps[c(seq(8, 11), 13, 14, 15, 16, 17, 18, 21, 22)], nrow = 4, ncol = 3) # 850x800



grid.arrange(grobs = evi\_maps[c(25, 26, 27)], nrow = 4, ncol = 3) # 850x800



## Transect analysis

In the code below, We examine an example of local correlation and clustering among the surface gradient metrics by extracting values over a horizontal transect across central Oregon.

First we convert the raw elevation and EVI data from the NASA’s SRTM and MODIS mission, respectively, to a dataframe and add those raw values to the dataframes for EVI and elevation containing the gradient surface metrics we’ve calculated across the state of Oregon.

elev <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv/oregon\_SRTM\_elevation\_2000.tif')  
evi <- raster('C:/Users/lsth490/Box/Smith, Annie/papers/geodiv/oregon\_max\_growing\_season\_EVI\_2000\_2019.tif') \* 0.0001  
  
# Coarsen resolution to 2km  
elev <- aggregate(elev, fact = 8)  
evi <- aggregate(evi, fact = 8)  
  
# convert to data frame  
elev\_pixdf <- as(elev, "SpatialPixelsDataFrame")  
elev\_df <- as.data.frame(elev\_pixdf)  
colnames(elev\_df) <- c("value", "x", "y")  
evi\_pixdf <- as(evi, "SpatialPixelsDataFrame")  
evi\_df <- as.data.frame(evi\_pixdf)  
colnames(evi\_df) <- c("value", "x", "y")  
  
# add raw values to other metric dataframe  
data\_evi$evi <- evi\_df$value  
data\_elev$elev <- elev\_df$value

Now let’s extract the data along a latitudinal transect going across the state of Oregon.

# Calculate the closest y value to the mean and then select all pixels along that line.  
evi\_crop <- crop(evi, extent(evi, round(nrow(evi) / 2) - 7, round(nrow(evi) / 2) + 7, 1, ncol(evi)))  
elev\_crop <- crop(elev, extent(elev, round(nrow(elev) / 2) - 7, round(nrow(elev) / 2) + 7, 1, ncol(elev)))  
central\_y\_evi <- unique(coordinates(evi\_crop)[,2])[8]  
central\_y\_elev <- unique(coordinates(elev\_crop)[,2])[8]  
central\_y\_evi <- unique(data\_evi$y[near(data\_evi$y, central\_y\_evi, 0.01)])[1]  
central\_y\_elev <- unique(data\_elev$y[near(data\_elev$y, central\_y\_elev, 0.01)])[1]  
  
# Extract mean EVI and elevation values by column  
evi\_means <- colMeans(as.matrix(evi\_crop), na.rm = TRUE)  
elev\_means <- colMeans(as.matrix(elev\_crop), na.rm = TRUE)  
  
# Now limit the dataframe to those points in space  
transect\_evi <- data\_evi[data\_evi$y == central\_y\_evi,]  
transect\_elev <- data\_elev[data\_elev$y == central\_y\_elev,]  
  
# Add column means to dataframe  
transect\_evi$evi <- evi\_means  
transect\_elev$elev <- elev\_means

The code below places standardizes all metrics by placing them on the same scale from 0-1.

# Get all metrics on same scale (0-1)  
for (i in 3:ncol(transect\_evi)) {  
 transect\_evi[,i] <- (transect\_evi[,i] - min(transect\_evi[,i], na.rm = TRUE)) / (max(transect\_evi[,i], na.rm = TRUE) - min(transect\_evi[,i], na.rm = TRUE))  
}  
for (i in 3:ncol(transect\_elev)) {  
 transect\_elev[,i] <- (transect\_elev[,i] - min(transect\_elev[,i], na.rm = TRUE)) / (max(transect\_elev[,i], na.rm = TRUE) - min(transect\_elev[,i], na.rm = TRUE))  
}

Final data wrangling to take data from wide to long version for the clustering analysis.

# save full spread out (i.e., wide) version of dataframe  
spread\_evi <- transect\_evi  
spread\_elev <- transect\_elev  
  
# gather data to create long version of dataframe  
transect\_evi <- transect\_evi %>%  
 tidyr::gather(key = 'var', value = 'value', names(transect\_evi[,seq(3, ncol(transect\_evi))]))  
transect\_elev <- transect\_elev %>%  
 tidyr::gather(key = 'var', value = 'value', names(transect\_elev[,seq(3, ncol(transect\_elev))]))  
  
# order variables the same way for evi and elevation  
transect\_evi <- transect\_evi[order(transect\_evi$var),]  
transect\_elev <- transect\_elev[order(transect\_elev$var),]

#### Clustering Analysis

For the transect analysis, we will perform hierarchical clustering on metric values using the function ‘eclust’ in the package factoextra (Kassambara & Mundt, 2020). But, first some additional data wrangling to prepare the data for the clustering analysis below.

# Remove NA values  
sapply(spread\_evi[, c(3:ncol(spread\_evi))], function(x) sum(is.na(x)))

## s10z sa sbi sci sdc05055 sdc08085 sdc0005 sdq6   
## 0 0 40 40 40 40 40 0   
## sdr sds sfd shw sk sku smean sph   
## 42 0 73 67 40 40 53 40   
## spk sq srw srwi ssc ssk std stdi   
## 40 40 67 67 80 40 40 60   
## sv svi svk evi   
## 40 40 40 0

for (i in 3:ncol(spread\_evi)) {  
 spread\_evi <- spread\_evi[!is.na(spread\_evi[, i]),]  
}  
sapply(spread\_elev[, c(3:ncol(spread\_elev))], function(x) sum(is.na(x)))

## shw srw srwi std stdi s10z sa sbi   
## 63 63 63 35 55 0 0 36   
## sci sdc05055 sdc08085 sdc0005 sdq6 sdr sds sfd   
## 35 35 35 35 0 38 0 68   
## sk sku smean sph spk sq ssc ssk   
## 37 36 153 35 37 36 180 36   
## sv svi svk elev   
## 35 35 37 0

for (i in 3:ncol(spread\_elev)) {  
 spread\_elev <- spread\_elev[!is.na(spread\_elev[, i]),]  
}  
  
# Check for constant columns  
sapply(spread\_evi[, c(3:ncol(spread\_evi))], function(x) length(unique(x)))

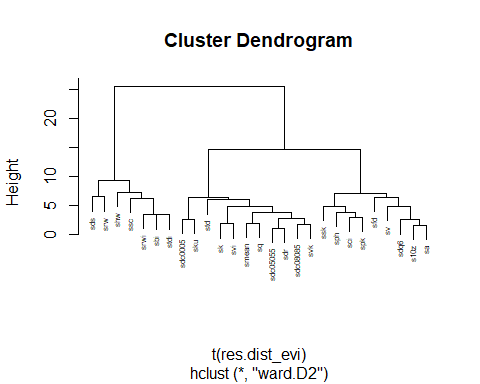
## s10z sa sbi sci sdc05055 sdc08085 sdc0005 sdq6   
## 356 360 360 360 360 360 349 360   
## sdr sds sfd shw sk sku smean sph   
## 360 285 360 64 360 360 347 143   
## spk sq srw srwi ssc ssk std stdi   
## 360 360 200 360 345 360 57 360   
## sv svi svk evi   
## 194 360 355 360

sapply(spread\_elev[, c(3:ncol(spread\_elev))], function(x) length(unique(x)))

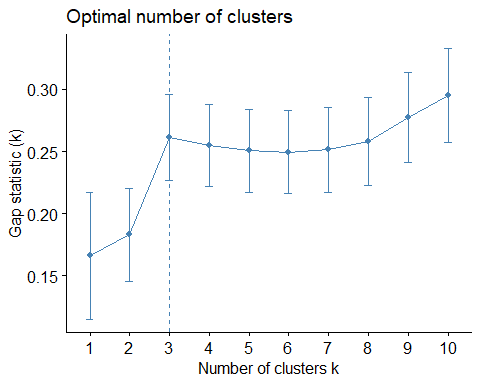
## shw srw srwi std stdi s10z sa sbi   
## 17 82 245 37 245 225 245 245   
## sci sdc05055 sdc08085 sdc0005 sdq6 sdr sds sfd   
## 245 245 244 228 245 245 164 245   
## sk sku smean sph spk sq ssc ssk   
## 245 245 245 169 245 245 245 245   
## sv svi svk elev   
## 200 245 244 245

The code below performs the clustering analysis on the EVI surface gradient metrics.

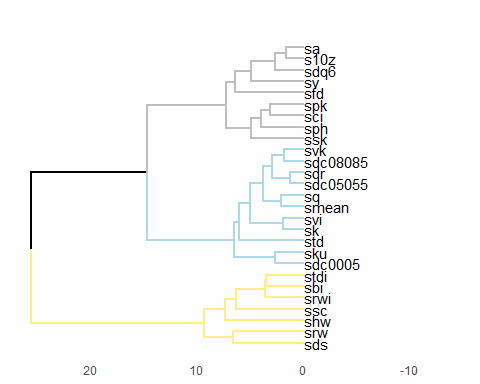
### EVI  
# Compute dissimilarity matrix  
res.dist\_evi <- dist(t(spread\_evi)[3:(ncol(spread\_evi) - 1),], method = "euclidean")  
# Compute hierarchical clustering  
res.hc\_evi <- hclust(t(res.dist\_evi), method = "ward.D2")  
# Visualize  
plot(res.hc\_evi, cex = 0.5)



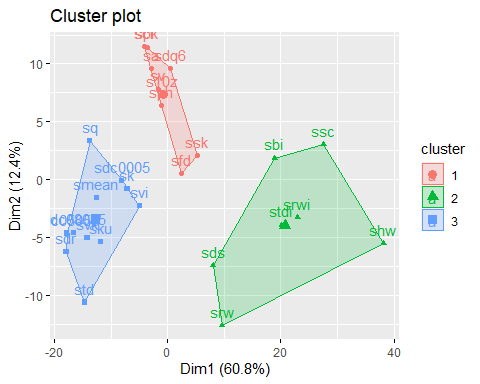
# Enhanced k-means clustering  
res.km\_evi <- clusGap(t(spread\_evi)[3:(ncol(spread\_evi) - 1),], stats::kmeans, K.max = 10, B = 100, nstart = 25)  
# Gap statistic plot  
factoextra::fviz\_gap\_stat(res.km\_evi)



# Enhanced hierarchical clustering  
res.hc\_evi <- factoextra::eclust(t(spread\_evi)[3:(ncol(spread\_evi) - 1),], "hclust") # compute hclust  
# Dendrogram plot  
fviz\_dend(res.hc\_evi, rect = FALSE, k\_colors = c('lightgoldenrod1', 'lightblue', 'grey'), lwd = 1, label\_cols = 'black',  
 cex = 0.8, main = "", ylab = "", type = 'rectangle', horiz = TRUE, labels\_track\_height = 14) +  
 theme(axis.text.y = element\_blank(),  
 axis.ticks = element\_blank())

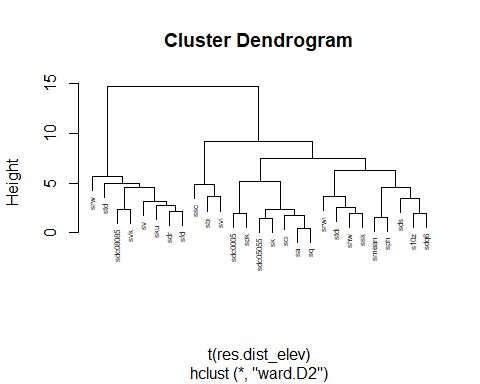


# Scatterplot  
fviz\_cluster(res.hc\_evi) # scatter plot

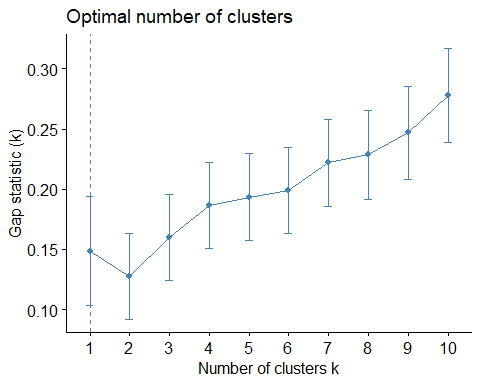


The code below performs the clustering analysis on the elevation surface gradient metrics.

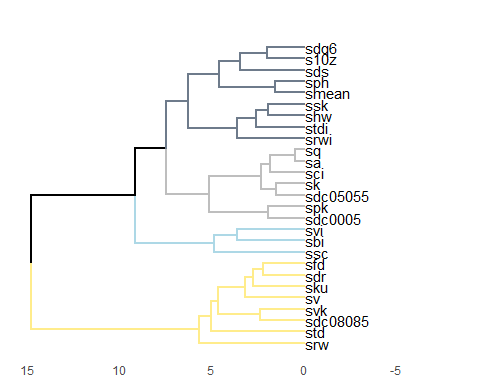
### ELEV  
# Compute dissimilarity matrix  
res.dist\_elev <- dist(t(spread\_elev)[3:(ncol(spread\_elev) - 1),], method = "euclidean")  
# Compute hierarchical clustering  
res.hc\_elev <- hclust(t(res.dist\_elev), method = "ward.D2")  
# Visualize  
plot(res.hc\_elev, cex = 0.5)



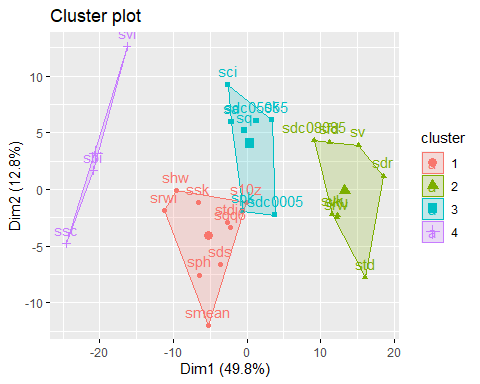
# Enhanced k-means clustering  
res.km\_elev <- clusGap(t(spread\_elev)[3:(ncol(spread\_elev) - 1),], stats::kmeans, K.max = 10, B = 100, nstart = 25)  
# Gap statistic plot  
factoextra::fviz\_gap\_stat(res.km\_elev)



# Enhanced hierarchical clustering  
res.hc\_elev <- factoextra::eclust(t(spread\_elev)[3:(ncol(spread\_elev) - 1),], "hclust", k = 4) # compute hclust  
# Plot Dendrogram  
fviz\_dend(res.hc\_elev, rect = FALSE, k\_colors = c('lightgoldenrod1', 'lightblue', 'grey', 'lightsteelblue4'), lwd = 1, label\_cols = 'black',  
 cex = 0.8, main = "", ylab = "", horiz = TRUE, type = 'rectangle', labels\_track\_height = 8) +  
 theme(axis.text.y = element\_blank(),  
 axis.ticks = element\_blank())

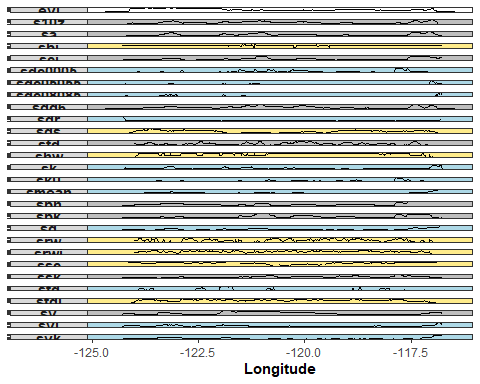


# Scatterplot  
fviz\_cluster(res.hc\_elev)

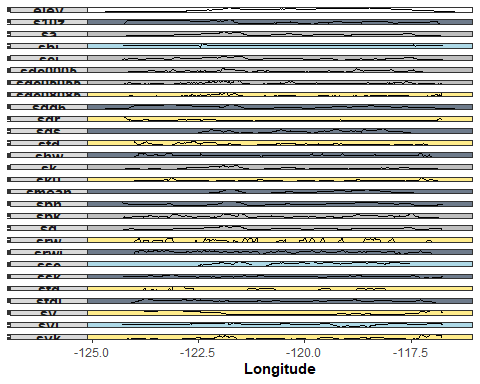


Now, let’s generate plots that show the EVI and elevation surface gradient metrics along the Oregon state transect.

# EVI clusters  
evi\_c1 <- c('sds', 'srw', 'shw', 'ssc', 'srwi', 'sbi', 'stdi')  
evi\_c2 <- c('sdc0005', 'sku', 'std', 'sk' , 'svi', 'smean', 'sq', 'sdc05055', 'sdr', 'sdc08085', 'svk')  
evi\_c3 <- c('ssk', 'sph', 'sci', 'spk', 'sfd', 'sv', 'sdq6', 's10z', 'sa')  
  
# Elevation clusters  
elev\_c1 <- c('srw', 'std', 'sdc08085', 'svk', 'sv', 'sku', 'sdr', 'sfd')  
elev\_c2 <- c('ssc', 'sbi', 'svi')  
elev\_c3 <- c('sdc0005', 'spk', 'sdc05055', 'sk', 'sci', 'sa', 'sq')  
elev\_c4 <- c('srwi', 'stdi', 'shw', 'ssk', 'smean', 'sph', 'sds', 's10z', 'sdq6')  
  
# Apply colors by cluster  
transect\_evi$cluster <- NA  
transect\_evi$cluster[transect\_evi$var %in% evi\_c1] <- 1  
transect\_evi$cluster[transect\_evi$var %in% evi\_c2] <- 2  
transect\_evi$cluster[transect\_evi$var %in% evi\_c3] <- 3  
transect\_evi$cluster[is.na(transect\_evi$cluster)] <- 'EVI'  
  
transect\_elev$cluster <- NA  
transect\_elev$cluster[transect\_elev$var %in% elev\_c1] <- 1  
transect\_elev$cluster[transect\_elev$var %in% elev\_c2] <- 2  
transect\_elev$cluster[transect\_elev$var %in% elev\_c3] <- 3  
transect\_elev$cluster[transect\_elev$var %in% elev\_c4] <- 4  
transect\_elev$cluster[is.na(transect\_elev$cluster)] <- 'Elevation'  
  
# Plot EVI (cropped evi metrics on top of tiny version of entire state showing transect location)  
ggplot(transect\_evi, aes(x = x, y = value)) +   
 geom\_rect(aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf, fill = as.factor(cluster))) +  
 geom\_line(lwd = 0.7) +  
 xlab('Longitude') +  
 facet\_grid(var~., switch = 'y') +  
 scale\_fill\_manual(values = c("1" = "lightgoldenrod1", "2" = "lightblue", "3" = "grey", 'EVI' = 'white'), name = 'Cluster') +  
 theme\_bw() +  
 theme(axis.title.y = element\_blank(),  
 axis.text.y = element\_blank(),  
 strip.text.y.left = element\_text(face = 'bold', size = 11, angle = 0),  
 legend.position = 'none',  
 axis.title.x = element\_text(face = 'bold', size = 11))



# Plot elevation (cropped elevation metrics on top of tiny version of entire state showing transect location)  
ggplot(transect\_elev, aes(x = x, y = value)) +   
 geom\_rect(aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf, fill = as.factor(cluster))) +  
 geom\_line(lwd = 0.7) +  
 xlab('Longitude') +  
 facet\_grid(var~., switch = 'y') +  
 scale\_fill\_manual(values = c("1" = "lightgoldenrod1", "2" = "lightblue", "3" = "grey", "4" = "lightsteelblue4", 'Elevation' = 'white'), name = 'Cluster') +  
 theme\_bw() +  
 theme(axis.title.y = element\_blank(),  
 axis.text.y = element\_blank(),  
 strip.text.y.left = element\_text(face = 'bold', size = 11, angle = 0),  
 legend.position = 'none',  
 axis.title.x = element\_text(face = 'bold', size = 11))



### Summary of transect cluster analysis

Overall trends along the transect were similar among metrics, with more variation at smaller intervals. Using hierarchical clustering, we found four clusters of metrics for elevation, and three for EVI. The metrics fell into different combinations based on the variable considered (elevation or EVI). For example, Sdq6 and S10z grouped together for both variables, but Std and Srw were in the same group for elevation, and different groups for EVI.

## Principal Components Analysis (PCA)

Next, we will determine the statewide elevation and EVI variance explained by metrics using PCA. First, we need to get the data ready for the PCA. In the code below, we remove several variables due to their large number of NA values, caused either by windows containing too few values, or windows lacking ‘peaks’ or ‘valleys’ (pixels surrounded by lower or higher values, respectively). After cleaning the data, 21 metrics remain in the analysis.

# Get data ready for PCA   
# Identify metrics with large numbers of NA values  
sapply(data\_elev, function(x) sum(is.na(x)))

## x y shw srw srwi std stdi s10z   
## 0 0 23372 23372 23372 12000 20362 0   
## sa sbi sci sdc05055 sdc08085 sdc0005 sdq6 sdr   
## 0 12263 12000 12000 12000 12000 0 13018   
## sds sfd sk sku smean sph spk sq   
## 0 26252 12379 12263 34767 11996 12379 12263   
## ssc ssk sv svi svk elev   
## 39530 12263 11996 12000 12379 0

sapply(data\_evi, function(x) sum(is.na(x)))

## x y s10z sa sbi sci sdc05055 sdc08085   
## 0 0 0 0 14912 14631 14631 14631   
## sdc0005 sdq6 sdr sds sfd shw sk sku   
## 14631 0 15719 0 29741 26609 15010 14912   
## smean sph spk sq srw srwi ssc ssk   
## 21555 14618 15010 14912 26609 26609 28087 14912   
## std stdi sv svi svk evi   
## 14631 23343 14618 14631 15010 0

#Remove metrics with large numbers of NA values  
data\_elev <- data\_elev[, c(-3, -4, -5, -21, -25, -18)] # remove shw, srw, srwi, smean, ssc, sfd  
data\_evi <- data\_evi[, c(-13, -14, -21, -22, -23, -17)] # remove shw, srw, srwi, smean, ssc, sfd  
data\_elev\_noNA <- na.omit(data\_elev) # removes most additional NA values beyond those in metrics targeted in above two lines of code  
data\_evi\_noNA <- na.omit(data\_evi)

In the code below, the PCA is performed with the remaining metrics using the ‘prcomp’ function in the stats package.

# Calculate the principal components  
elev\_prc <- prcomp(data\_elev\_noNA[,3:23], center = TRUE, scale = TRUE)  
evi\_prc <- prcomp(data\_evi\_noNA[,3:23], center = TRUE, scale = TRUE)  
summary(elev\_prc)

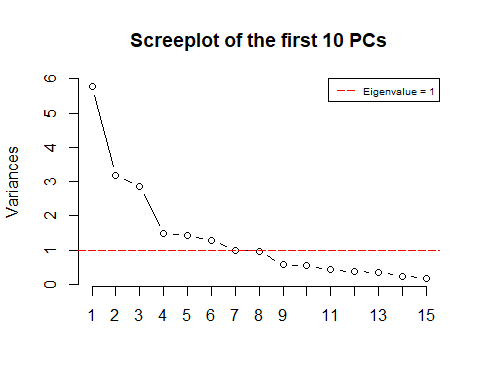
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 2.4045 1.7841 1.6919 1.21353 1.19505 1.13416 1.00019  
## Proportion of Variance 0.2753 0.1516 0.1363 0.07013 0.06801 0.06125 0.04764  
## Cumulative Proportion 0.2753 0.4269 0.5632 0.63332 0.70133 0.76258 0.81022  
## PC8 PC9 PC10 PC11 PC12 PC13 PC14  
## Standard deviation 0.98427 0.75362 0.73849 0.65243 0.6046 0.59616 0.49301  
## Proportion of Variance 0.04613 0.02704 0.02597 0.02027 0.0174 0.01692 0.01157  
## Cumulative Proportion 0.85635 0.88340 0.90937 0.92964 0.9470 0.96397 0.97554  
## PC15 PC16 PC17 PC18 PC19 PC20 PC21  
## Standard deviation 0.41766 0.3723 0.31335 0.24444 0.17253 0.0912 0.06780  
## Proportion of Variance 0.00831 0.0066 0.00468 0.00285 0.00142 0.0004 0.00022  
## Cumulative Proportion 0.98385 0.9905 0.99512 0.99797 0.99939 0.9998 1.00000

summary(evi\_prc)

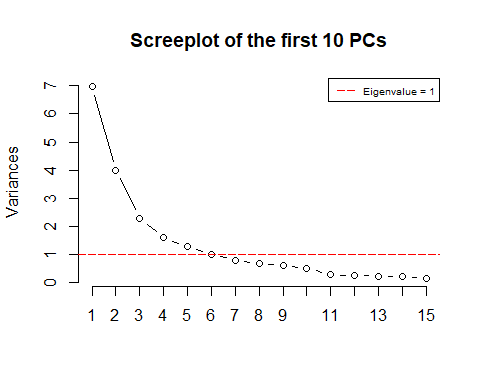
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 2.6422 1.9977 1.5084 1.25770 1.12477 0.99939 0.88731  
## Proportion of Variance 0.3325 0.1900 0.1083 0.07532 0.06024 0.04756 0.03749  
## Cumulative Proportion 0.3325 0.5225 0.6308 0.70615 0.76640 0.81396 0.85145  
## PC8 PC9 PC10 PC11 PC12 PC13 PC14  
## Standard deviation 0.8248 0.77923 0.70343 0.51598 0.49401 0.47486 0.43800  
## Proportion of Variance 0.0324 0.02891 0.02356 0.01268 0.01162 0.01074 0.00914  
## Cumulative Proportion 0.8839 0.91276 0.93633 0.94900 0.96062 0.97136 0.98050  
## PC15 PC16 PC17 PC18 PC19 PC20 PC21  
## Standard deviation 0.37444 0.3368 0.27901 0.21529 0.13515 0.09959 0.05939  
## Proportion of Variance 0.00668 0.0054 0.00371 0.00221 0.00087 0.00047 0.00017  
## Cumulative Proportion 0.98717 0.9926 0.99628 0.99849 0.99936 0.99983 1.00000

Now let’s look at some diagnostic plots for the principal components. Scree plots indicate the importance of the principal components with a broken stick criterion. The point at which the scree plot curve crosses the broken stick model distribution, which we will plot in red is considered to indicate the maximum number of components to retain.

# Take a look at the components for elevation using a screeplot  
screeplot(elev\_prc, type = "l", npcs = 15, main = "Screeplot of the first 10 PCs")  
abline(h = 1, col = "red", lty = 5)  
legend("topright", legend = c("Eigenvalue = 1"),  
 col = c("red"), lty = 5, cex = 0.6)

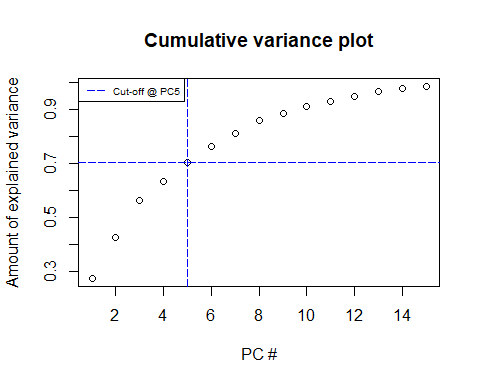


# Take a look at the components for EVI using a screeplot  
screeplot(evi\_prc, type = "l", npcs = 15, main = "Screeplot of the first 10 PCs")  
abline(h = 1, col = "red", lty = 5)  
legend("topright", legend = c("Eigenvalue = 1"),  
 col = c("red"), lty = 5, cex = 0.6)

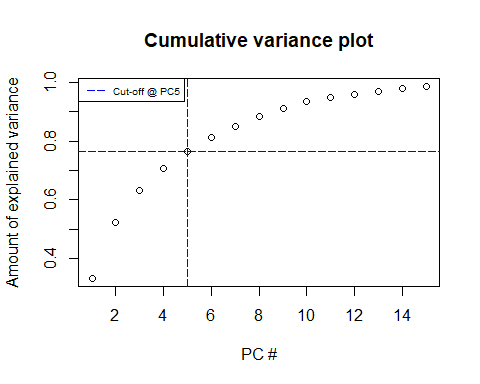


We can also take a look at the components for elevation and EVI to see how much variance the surface metrics explain with cumulative variance plots.

# Elevation Cumulative Variance Plot  
cumpro <- cumsum(elev\_prc$sdev ^ 2 / sum(elev\_prc$sdev ^ 2))  
plot(cumpro[0:15], xlab = "PC #", ylab = "Amount of explained variance", main = "Cumulative variance plot")  
abline(v = 5, col = "blue", lty = 5)  
abline(h = 0.70133, col = "blue", lty = 5)  
legend("topleft", legend = c("Cut-off @ PC5"),  
 col = c("blue"), lty = 5, cex = 0.6)

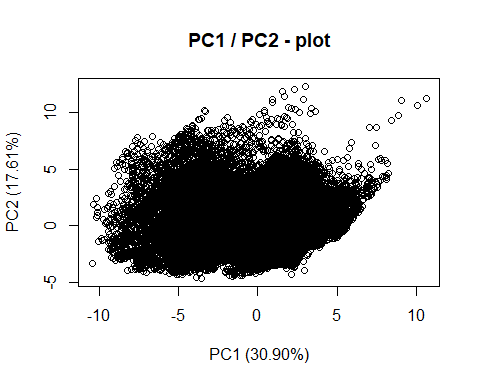


#EVI Cumulative Variance Plot  
cumpro <- cumsum(evi\_prc$sdev ^ 2 / sum(evi\_prc$sdev ^ 2))  
plot(cumpro[0:15], xlab = "PC #", ylab = "Amount of explained variance", main = "Cumulative variance plot")  
abline(v = 5, col = "blue", lty = 5)  
abline(h = 0.76640, col = "blue", lty = 5)  
legend("topleft", legend = c("Cut-off @ PC5"),  
 col = c("blue"), lty = 5, cex = 0.6)

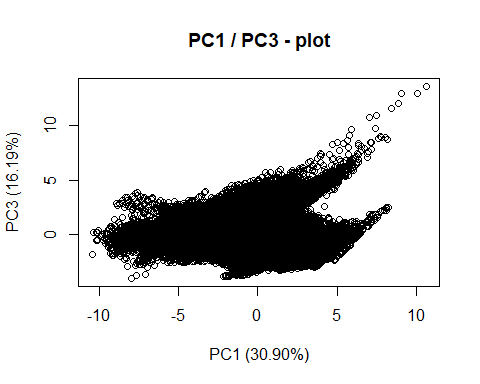


For both elevation and EVI, the first 5 principal components explained >70% of the variation. Now, let’s plot the first components for elevation.

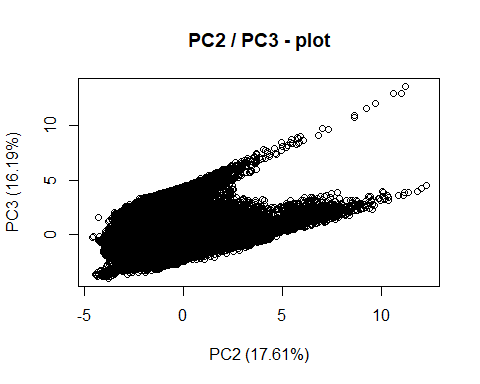
plot(elev\_prc$x[, 1], elev\_prc$x[, 2], xlab = "PC1 (30.90%)", ylab = "PC2 (17.61%)", main = "PC1 / PC2 - plot")



plot(elev\_prc$x[, 1], elev\_prc$x[, 3], xlab = "PC1 (30.90%)", ylab = "PC3 (16.19%)", main = "PC1 / PC3 - plot")



plot(elev\_prc$x[, 2], elev\_prc$x[, 3], xlab = "PC2 (17.61%)", ylab = "PC3 (16.19%)", main = "PC2 / PC3 - plot")



In the code below, let’s also map these elevation components to see if there are any spatial patterns.

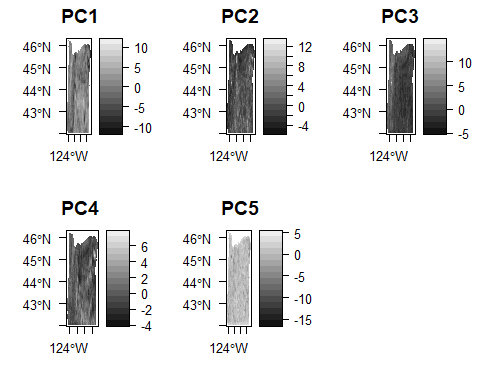
data\_elev\_noNA$prc1 <- elev\_prc$x[, 1]  
data\_elev\_noNA$prc2 <- elev\_prc$x[, 2]  
data\_elev\_noNA$prc3 <- elev\_prc$x[, 3]  
data\_elev\_noNA$prc4 <- elev\_prc$x[, 4]  
data\_elev\_noNA$prc5 <- elev\_prc$x[, 5]  
data\_evi\_noNA$prc1 <- evi\_prc$x[, 1]  
data\_evi\_noNA$prc2 <- evi\_prc$x[, 2]  
data\_evi\_noNA$prc3 <- evi\_prc$x[, 3]  
data\_evi\_noNA$prc4 <- evi\_prc$x[, 4]  
data\_evi\_noNA$prc5 <- evi\_prc$x[, 5]  
  
# Plot pca results   
# Add pca results back to raster data  
elev\_pca\_dat <- data\_elev %>%  
 left\_join(data\_elev\_noNA)

## Joining, by = c("x", "y", "std", "stdi", "s10z", "sa", "sbi", "sci", "sdc05055", "sdc08085", "sdc0005", "sdq6", "sdr", "sds", "sk", "sku", "sph", "spk", "sq", "ssk", "sv", "svi", "svk", "elev")

evi\_pca\_dat <- data\_evi %>%  
 left\_join(data\_evi\_noNA)

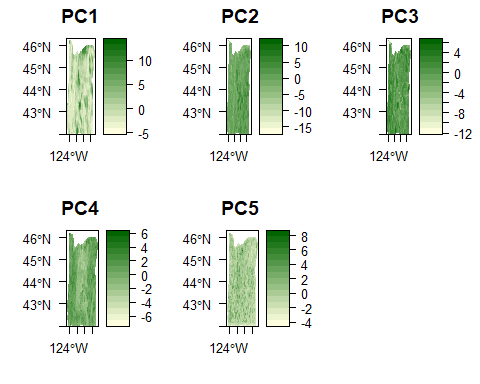
## Joining, by = c("x", "y", "s10z", "sa", "sbi", "sci", "sdc05055", "sdc08085", "sdc0005", "sdq6", "sdr", "sds", "sk", "sku", "sph", "spk", "sq", "ssk", "std", "stdi", "sv", "svi", "svk", "evi")

elev\_prc1 <- setValues(elev, elev\_pca\_dat$prc1)  
elev\_prc2 <- setValues(elev, elev\_pca\_dat$prc2)  
elev\_prc3 <- setValues(elev, elev\_pca\_dat$prc3)  
elev\_prc4 <- setValues(elev, elev\_pca\_dat$prc4)  
elev\_prc5 <- setValues(elev, elev\_pca\_dat$prc5)  
  
evi\_prc1 <- setValues(evi, evi\_pca\_dat$prc1)  
evi\_prc2 <- setValues(evi, evi\_pca\_dat$prc2)  
evi\_prc3 <- setValues(evi, evi\_pca\_dat$prc3)  
evi\_prc4 <- setValues(evi, evi\_pca\_dat$prc4)  
evi\_prc5 <- setValues(evi, evi\_pca\_dat$prc5)  
  
## Elevation plots for the first five components as indicated as appropriate by the scree plot.  
elev\_pc1\_map <- levelplot(elev\_prc1, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'PC1')  
elev\_pc2\_map <- levelplot(elev\_prc2, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'PC2')  
elev\_pc3\_map <- levelplot(elev\_prc3, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'PC3')  
elev\_pc4\_map <- levelplot(elev\_prc4, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'PC4')  
elev\_pc5\_map <- levelplot(elev\_prc5, margin = F, par.settings = elevTheme, ylab = NULL, xlab = NULL, main = 'PC5')  
  
elev\_pc1\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
elev\_pc1\_map$aspect.fill <- TRUE  
elev\_pc2\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
elev\_pc2\_map$aspect.fill <- TRUE  
elev\_pc3\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
elev\_pc3\_map$aspect.fill <- TRUE  
elev\_pc4\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
elev\_pc4\_map$aspect.fill <- TRUE  
elev\_pc5\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
elev\_pc5\_map$aspect.fill <- TRUE  
grid.arrange(elev\_pc1\_map, elev\_pc2\_map, elev\_pc3\_map, elev\_pc4\_map, elev\_pc5\_map, nrow = 2, ncol = 3)



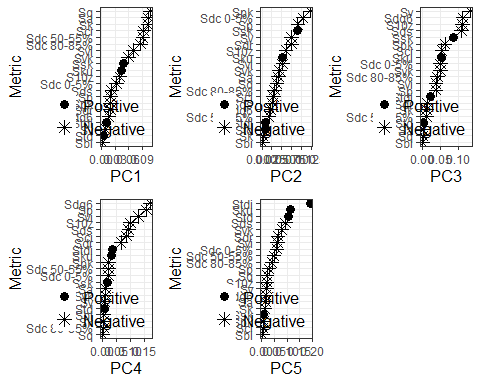
Let’s map the EVI components in space, too.

# EVI plots for first 5 components as indicated by scree plots.  
evi\_pc1\_map <- levelplot(evi\_prc1, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'PC1')  
evi\_pc2\_map <- levelplot(evi\_prc2, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'PC2')  
evi\_pc3\_map <- levelplot(evi\_prc3, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'PC3')  
evi\_pc4\_map <- levelplot(evi\_prc4, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'PC4')  
evi\_pc5\_map <- levelplot(evi\_prc5, margin = F, par.settings = eviTheme, ylab = NULL, xlab = NULL, main = 'PC5')  
  
evi\_pc1\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
evi\_pc1\_map$aspect.fill <- TRUE  
evi\_pc2\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
evi\_pc2\_map$aspect.fill <- TRUE  
evi\_pc3\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
evi\_pc3\_map$aspect.fill <- TRUE  
evi\_pc4\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
evi\_pc4\_map$aspect.fill <- TRUE  
evi\_pc5\_map$par.settings$layout.heights[  
 c( 'bottom.padding',  
 'top.padding',  
 'key.sub.padding',  
 'axis.xlab.padding',  
 'key.axis.padding',  
 'main.key.padding') ] <- 1  
evi\_pc5\_map$aspect.fill <- TRUE  
grid.arrange(evi\_pc1\_map, evi\_pc2\_map, evi\_pc3\_map, evi\_pc4\_map, evi\_pc5\_map, nrow = 2, ncol = 3)

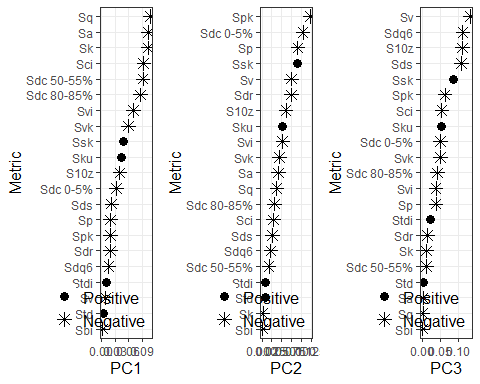


What are the pc loadings for elevation?

# Elevation  
# Rotation  
elev\_loadings <- elev\_prc$rotation[, 1:5]  
# Elevation components' relative loadings  
aload <- abs(elev\_loadings)  
elev\_rel <- sweep(aload, 2, colSums(aload), "/")  
elev\_rel <- as.data.frame(elev\_rel)  
elev\_rel$var <- c('Std', 'Stdi', 'S10z', 'Sa', 'Sbi', 'Sci', 'Sdc 50-55%', 'Sdc 80-85%', 'Sdc 0-5%', 'Sdq6', 'Sdr', 'Sds', 'Sk', 'Sku', 'Sp', 'Spk',  
 'Sq', 'Ssk', 'Sv', 'Svi', 'Svk')  
# Importance plots for elevation  
elev\_rel1 <- elev\_rel  
elev\_rel1$sign <- factor(sapply(elev\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
elev\_rel1 <- elev\_rel1[order(elev\_rel1$PC1),]  
elev\_rel1$var <- factor(elev\_rel1$var, levels = elev\_rel1$var)  
elev\_pc1\_plt <- ggplot(elev\_rel1, aes(x = PC1, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC1') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
elev\_rel2 <- elev\_rel  
elev\_rel2$sign <- factor(sapply(elev\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
elev\_rel2 <- elev\_rel2[order(elev\_rel2$PC2),]  
elev\_rel2$var <- factor(elev\_rel2$var, levels = elev\_rel2$var)  
elev\_pc2\_plt <- ggplot(elev\_rel2, aes(x = PC2, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC2') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
elev\_rel3 <- elev\_rel  
elev\_rel3$sign <- factor(sapply(elev\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
elev\_rel3 <- elev\_rel3[order(elev\_rel3$PC3),]  
elev\_rel3$var <- factor(elev\_rel3$var, levels = elev\_rel3$var)  
elev\_pc3\_plt <- ggplot(elev\_rel3, aes(x = PC3, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC3') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
elev\_rel4 <- elev\_rel  
elev\_rel4$sign <- factor(sapply(elev\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
elev\_rel4 <- elev\_rel4[order(elev\_rel4$PC4),]  
elev\_rel4$var <- factor(elev\_rel4$var, levels = elev\_rel4$var)  
elev\_pc4\_plt <- ggplot(elev\_rel4, aes(x = PC4, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC4') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
elev\_rel5 <- elev\_rel  
elev\_rel5$sign <- factor(sapply(elev\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
elev\_rel5 <- elev\_rel5[order(elev\_rel5$PC5),]  
elev\_rel5$var <- factor(elev\_rel5$var, levels = elev\_rel5$var)  
elev\_pc5\_plt <- ggplot(elev\_rel5, aes(x = PC5, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC5') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
grid.arrange(elev\_pc1\_plt, elev\_pc2\_plt, elev\_pc3\_plt, elev\_pc4\_plt, elev\_pc5\_plt, nrow = 2, ncol = 3)

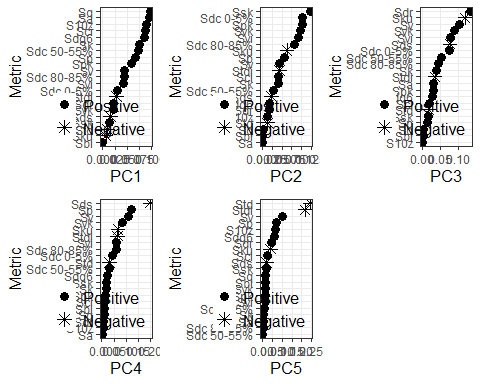


grid.arrange(elev\_pc1\_plt, elev\_pc2\_plt, elev\_pc3\_plt, ncol = 3) # 850x350

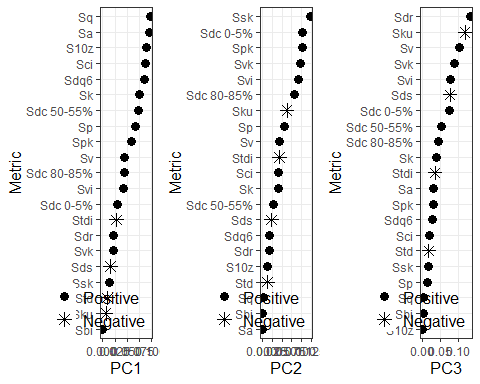


What are the pc loadings for EVI?

# EVI  
# rotation  
evi\_loadings <- evi\_prc$rotation[, 1:5]  
# EVI components' relative loadings  
aload <- abs(evi\_loadings)  
evi\_rel <- sweep(aload, 2, colSums(aload), "/")  
evi\_rel <- as.data.frame(evi\_rel)  
evi\_rel$var <- c('S10z', 'Sa', 'Sbi', 'Sci', 'Sdc 50-55%', 'Sdc 80-85%', 'Sdc 0-5%', 'Sdq6', 'Sdr', 'Sds', 'Sk', 'Sku', 'Sp', 'Spk',  
 'Sq', 'Ssk', 'Std', 'Stdi', 'Sv', 'Svi', 'Svk')  
  
# EVI importance plots  
evi\_rel1 <- evi\_rel  
evi\_rel1$sign <- factor(sapply(evi\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
evi\_rel1 <- evi\_rel1[order(evi\_rel1$PC1),]  
evi\_rel1$var <- factor(evi\_rel1$var, levels = evi\_rel1$var)  
evi\_pc1\_plt <- ggplot(evi\_rel1, aes(x = PC1, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC1') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
evi\_rel2 <- evi\_rel  
evi\_rel2$sign <- factor(sapply(evi\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
evi\_rel2 <- evi\_rel2[order(evi\_rel2$PC2),]  
evi\_rel2$var <- factor(evi\_rel2$var, levels = evi\_rel2$var)  
evi\_pc2\_plt <- ggplot(evi\_rel2, aes(x = PC2, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC2') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
evi\_rel3 <- evi\_rel  
evi\_rel3$sign <- factor(sapply(evi\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
evi\_rel3 <- evi\_rel3[order(evi\_rel3$PC3),]  
evi\_rel3$var <- factor(evi\_rel3$var, levels = evi\_rel3$var)  
evi\_pc3\_plt <- ggplot(evi\_rel3, aes(x = PC3, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC3') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
evi\_rel4 <- evi\_rel  
evi\_rel4$sign <- factor(sapply(evi\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
evi\_rel4 <- evi\_rel4[order(evi\_rel4$PC4),]  
evi\_rel4$var <- factor(evi\_rel4$var, levels = evi\_rel4$var)  
evi\_pc4\_plt <- ggplot(evi\_rel4, aes(x = PC4, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC4') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
evi\_rel5 <- evi\_rel  
evi\_rel5$sign <- factor(sapply(evi\_loadings[,1], FUN = function(x) x / abs(x)), levels = c(-1, 1))  
evi\_rel5 <- evi\_rel5[order(evi\_rel5$PC5),]  
evi\_rel5$var <- factor(evi\_rel5$var, levels = evi\_rel5$var)  
evi\_pc5\_plt <- ggplot(evi\_rel5, aes(x = PC5, y = var)) +  
 geom\_point(size = 3, aes(pch = sign)) +  
 scale\_shape\_manual(name = element\_blank(),  
 breaks = c(1, -1),  
 values = c(19, 8),  
 labels = c("Positive", "Negative")) +  
 xlab('PC5') +  
 ylab('Metric') +  
 theme\_bw() +  
 theme(panel.grid.minor = element\_blank(),  
 legend.justification = c(1, 0),   
 legend.position = c(1, 0),  
 legend.background = element\_blank(),  
 legend.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12))  
  
grid.arrange(evi\_pc1\_plt, evi\_pc2\_plt, evi\_pc3\_plt, evi\_pc4\_plt, evi\_pc5\_plt, nrow = 2, ncol = 3)



grid.arrange(evi\_pc1\_plt, evi\_pc2\_plt, evi\_pc3\_plt, ncol = 3)



### Summary of PCA

Looking at the PCA components and loadings, the first component for both elevation and EVI described general surface heterogeneity, whereas the second was related to the shape of the regional value distribution. The metric groupings observed match previous findings, and the first two components represent the same groupings (roughness and distribution) found by McGarigal et al. (2009).

## Overall summary

The contrast in metric groupings between the transect hierarchical clustering analysis and statewide PCA demonstrate that there may be differences in metric information depending on the landscape size and scale. As the PCA results were more in line with previous results in the literature, this suggests that those results show the broader metric grouping habits. The transect results demonstrate that this grouping does not always hold across different regions.