Gather Results

Dana Melamed

2023-04-02

Checkout modelling results for the entire growing season period of 2012. Then, apply the model to other years, and calculate a 5 year average.

## Model temperature with each input variable individually

Generate a model for each input raster, with the input raster as the independent and the growing seasons (April - November) GDD10 as the dependent variable.

This R code prepares a list of input rasters, resamples one of the rasters to match the extent of another raster, creates a multi-layer raster brick, applies some transformations to specific layers of the brick, assigns names to the layers based on a list of variable names, and then extracts the values of the raster layers at specified locations and adds them to a dataframe.

The in\_folder, resolution, var\_names, ext, and rasters\_list variables are used to set up a list of input rasters that are located in a specific folder, have a specific resolution, and have specific names that include a resolution suffix and extension. The lapply() function is used to create a list of raster objects based on these inputs.

The cp raster is extracted from the rasters\_list and then resampled using the raster::resample function to match the extent of the first raster in the list. The resulting resampled raster is then assigned back to the second element in the rasters\_list.

The rasters\_brick object is created by using the raster::brick function on the rasters\_list, which combines all of the individual raster layers into a single multi-layer object.

Two transformations are applied to specific layers of the rasters\_brick object using standard arithmetic operations. The second layer is raised to the power of 0.5, effectively taking the square root of the values, while the sixth layer is multiplied by the cosine of the seventh layer.

The names() function is used to assign the variable names to each layer of the rasters\_brick object.

Finally, the extract\_constant\_raster\_values() function is used to extract the values of the raster layers at specified locations and add them to a dataframe called df\_in. Overall, this code is a good example of how to prepare and manipulate raster data in R.

# Prepare the data  
# Make a list of input rasters  
in\_folder <- 'f:\\data\\rasters\\input\_rasters\\'  
resolution <- 100  
var\_names <- list('dem','cp','east','north','tpi5000m',  
 'asp','slp')  
ext = '.tif'  
rasters\_list <- lapply(X=paste0(in\_folder,var\_names,resolution,'m',ext),FUN=raster::raster)

## Warning in showSRID(SRS\_string, format = "PROJ", multiline = "NO", prefer\_proj  
## = prefer\_proj): Discarded datum NAD83 Canadian Spatial Reference System in Proj4  
## definition

# cp to the proper extent  
cp <- rasters\_list[[2]]  
rasters\_list[[2]] <- raster::resample(cp,rasters\_list[[1]])   
  
# brick rasters  
rasters\_brick <- raster::brick(rasters\_list)

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer\_proj  
## = prefer\_proj): Discarded datum Unknown based on GRS80 ellipsoid in Proj4  
## definition

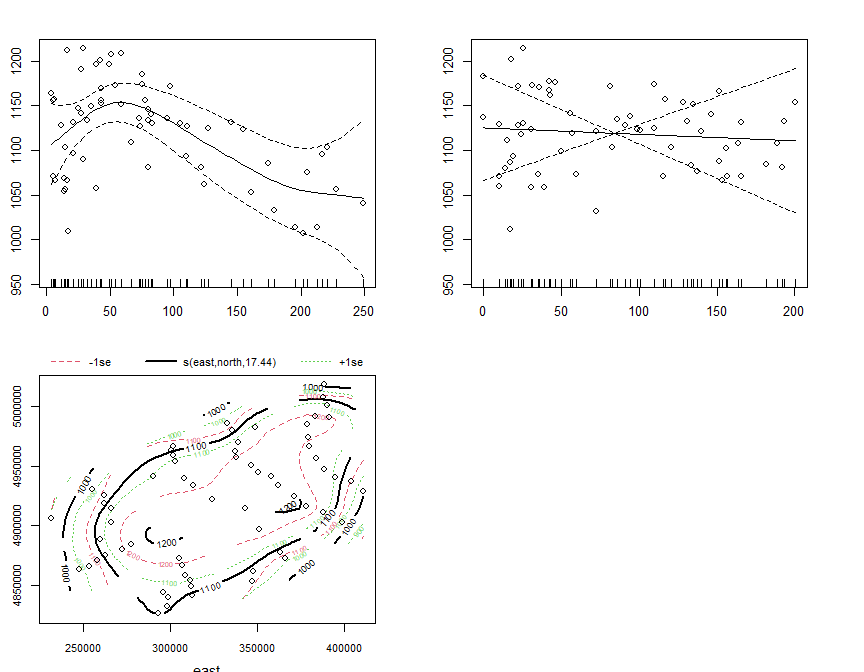
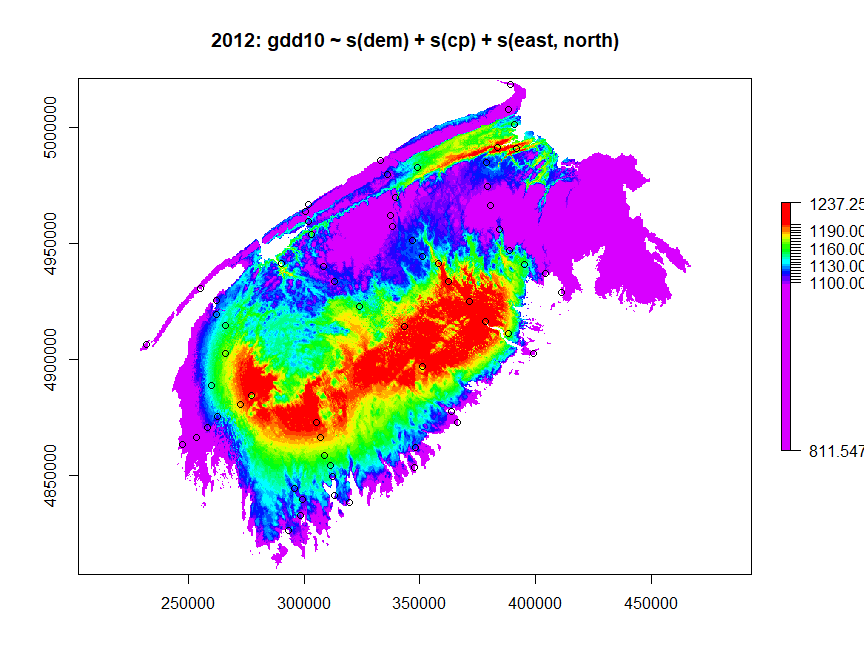
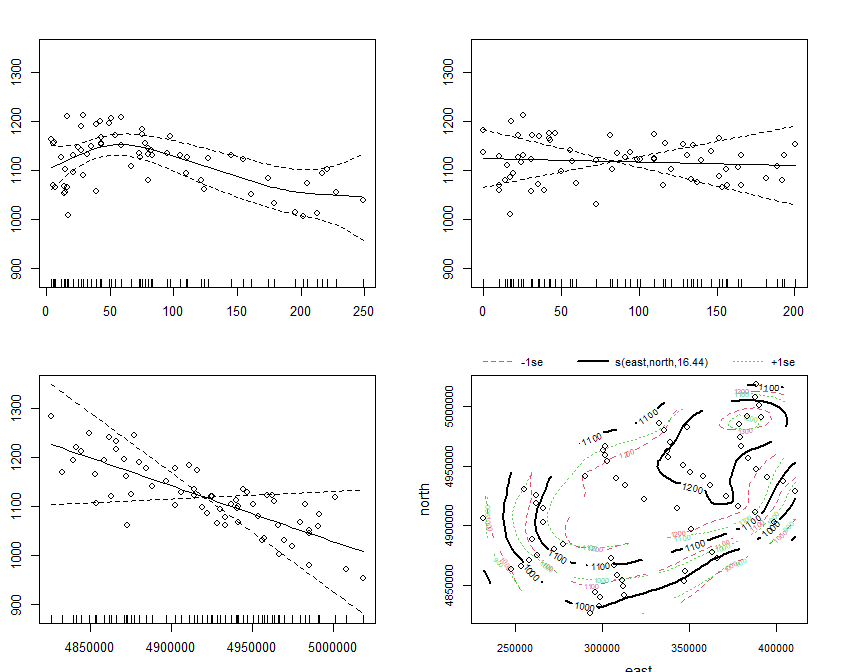
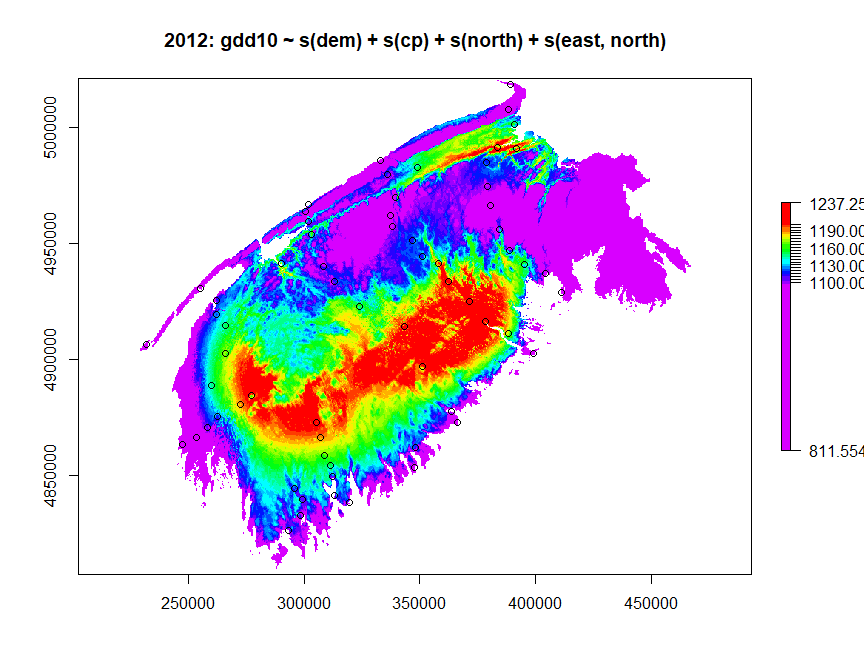
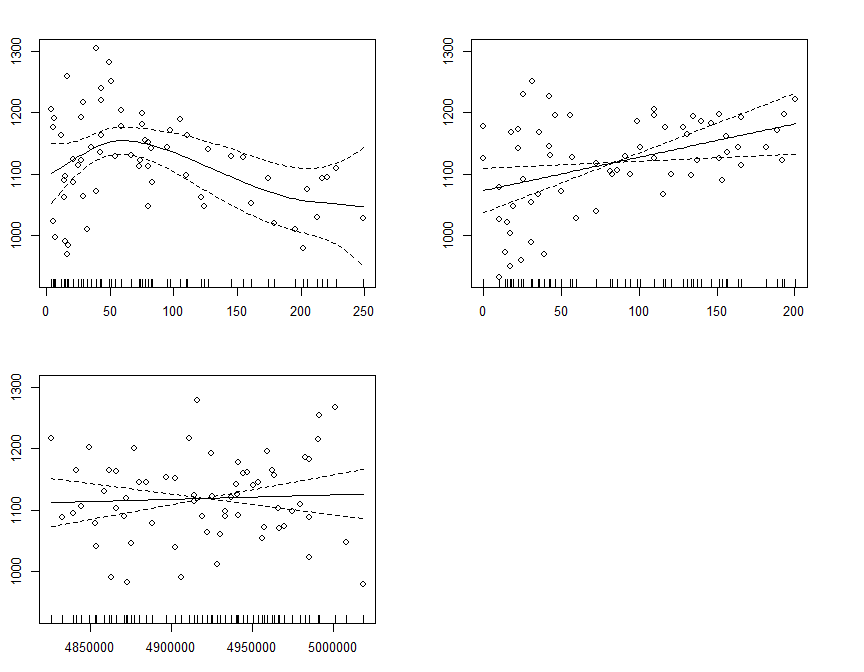
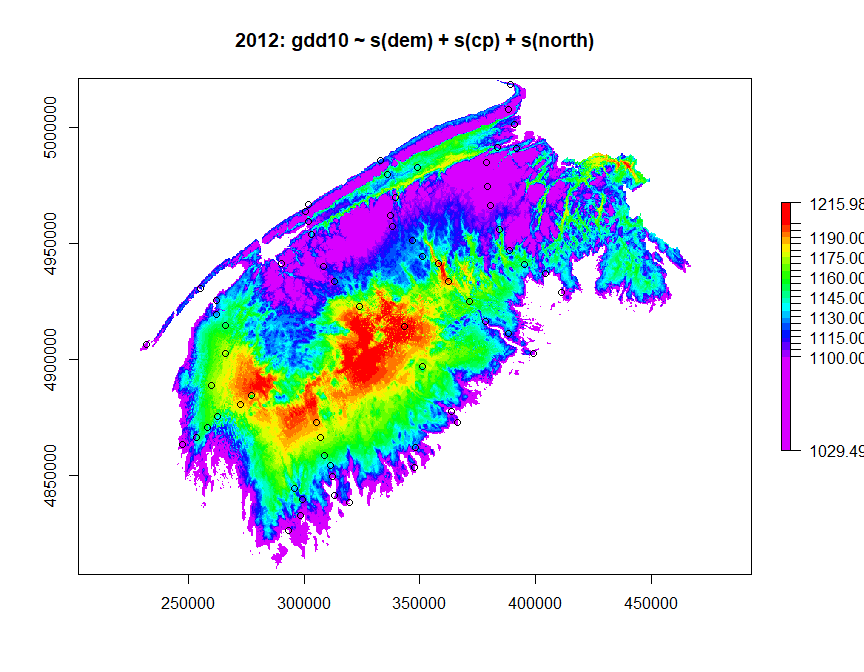
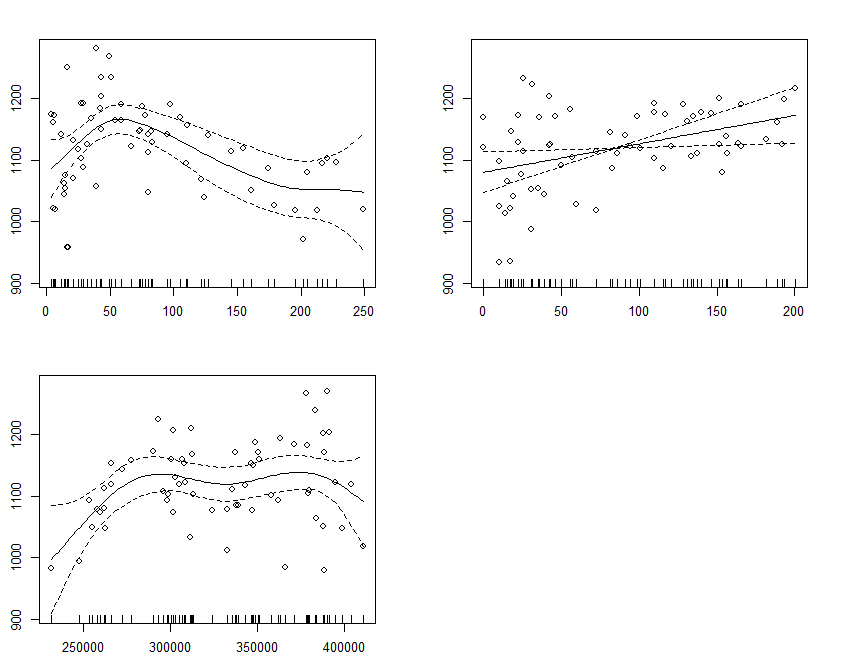
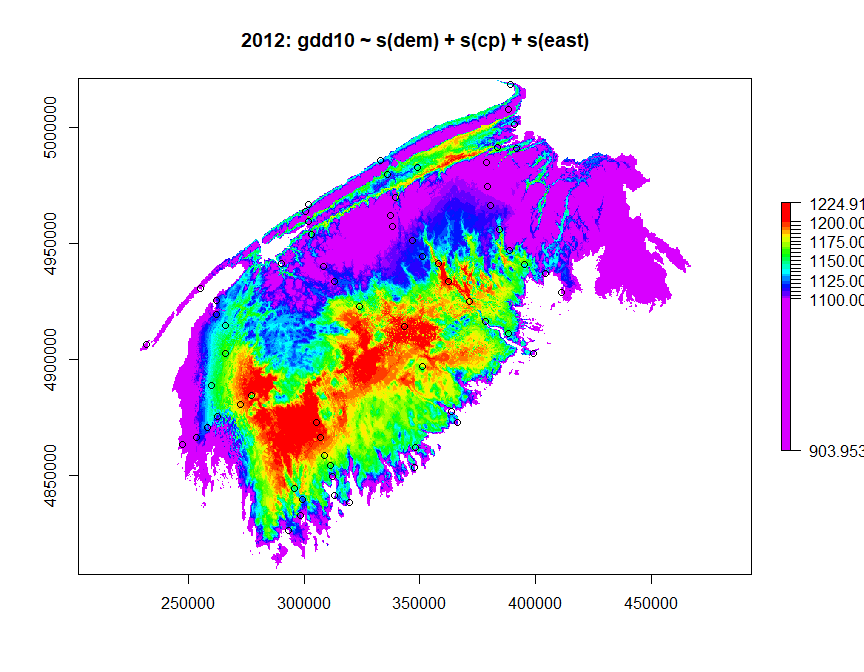
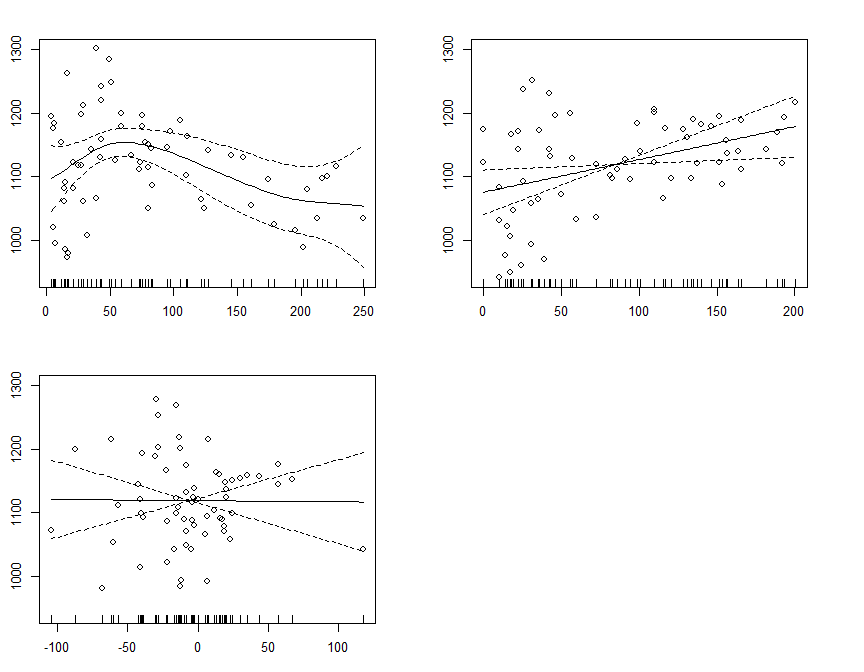
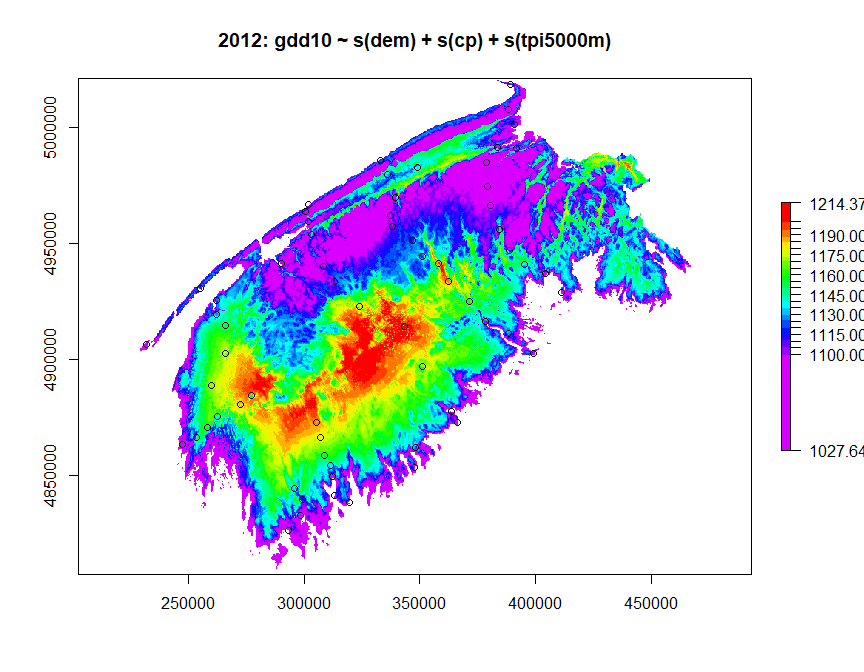
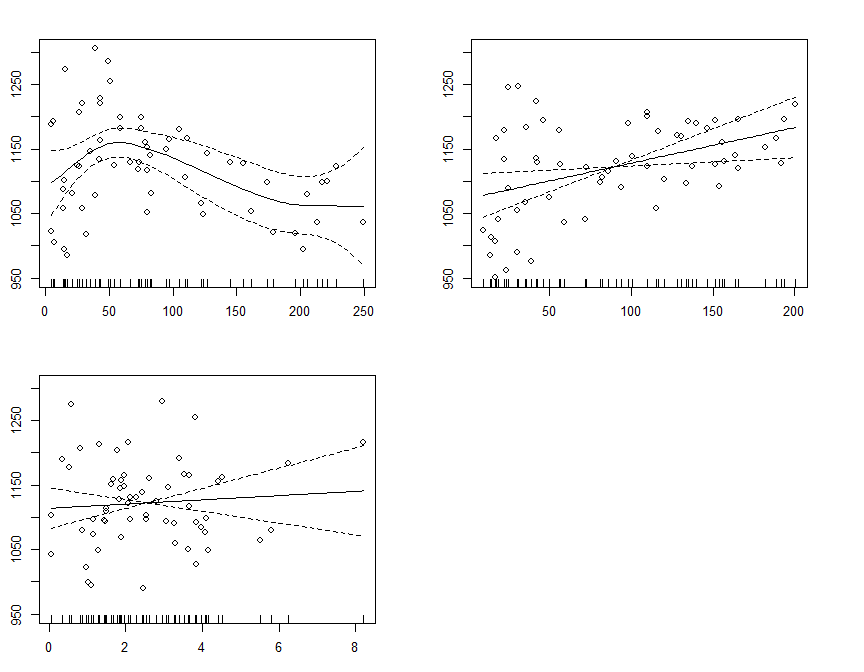
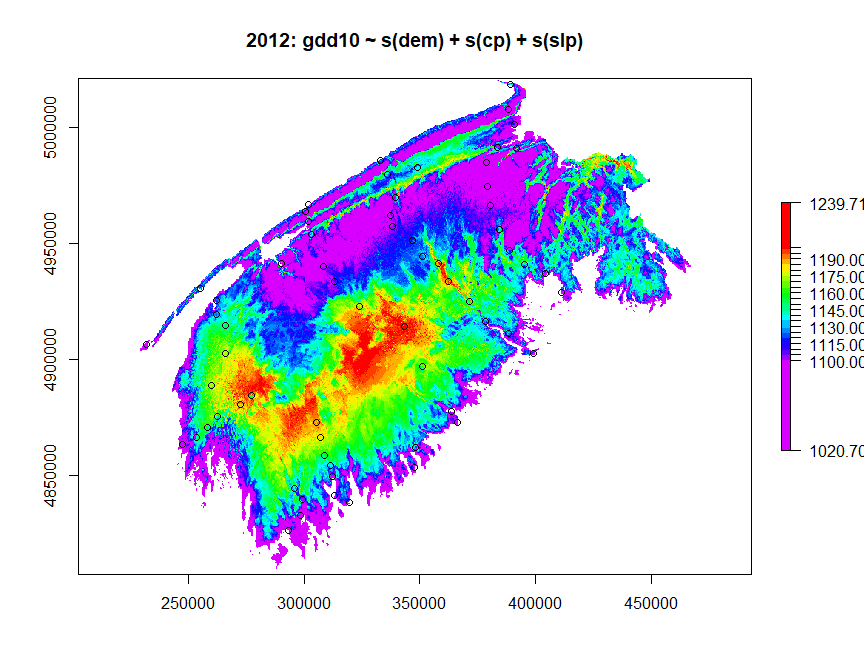
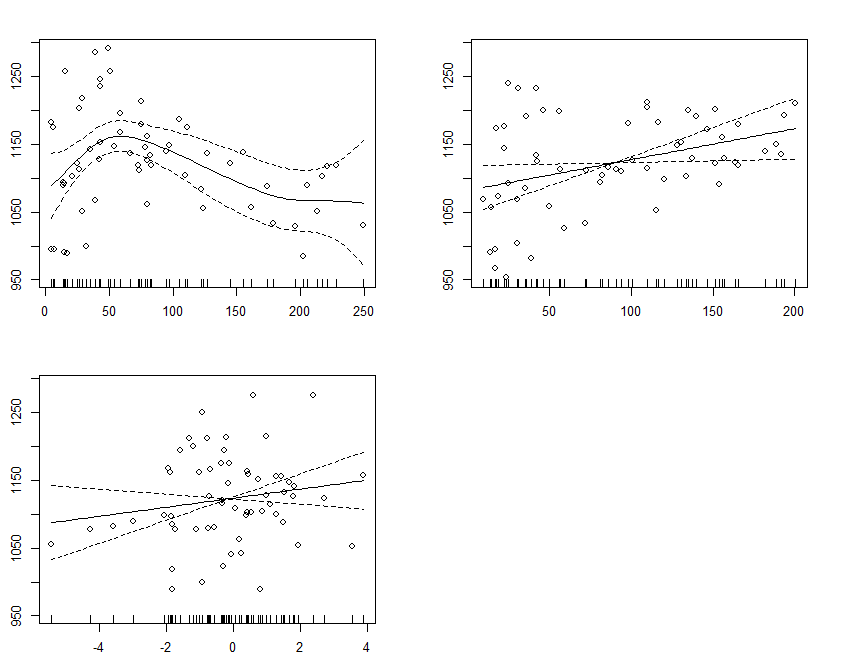
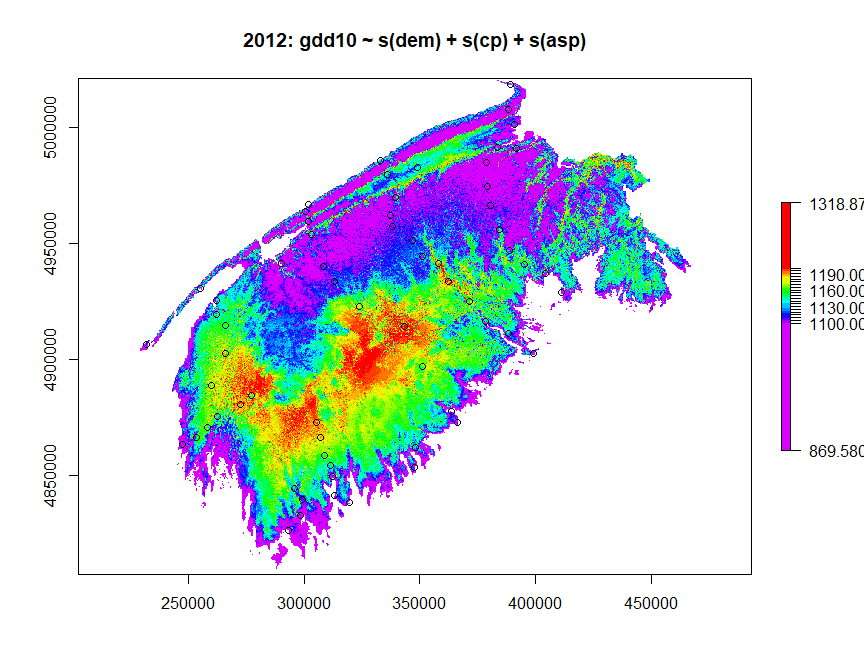
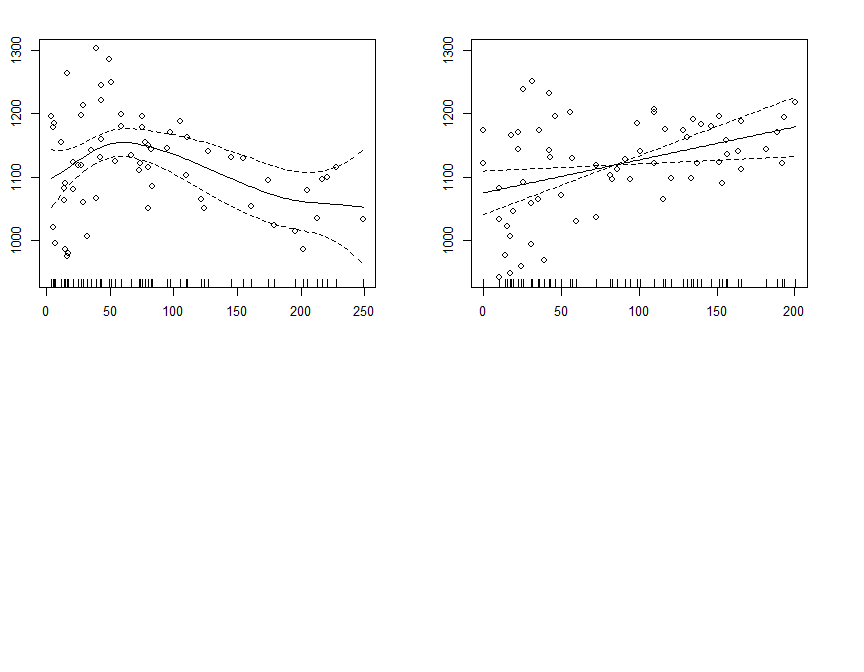
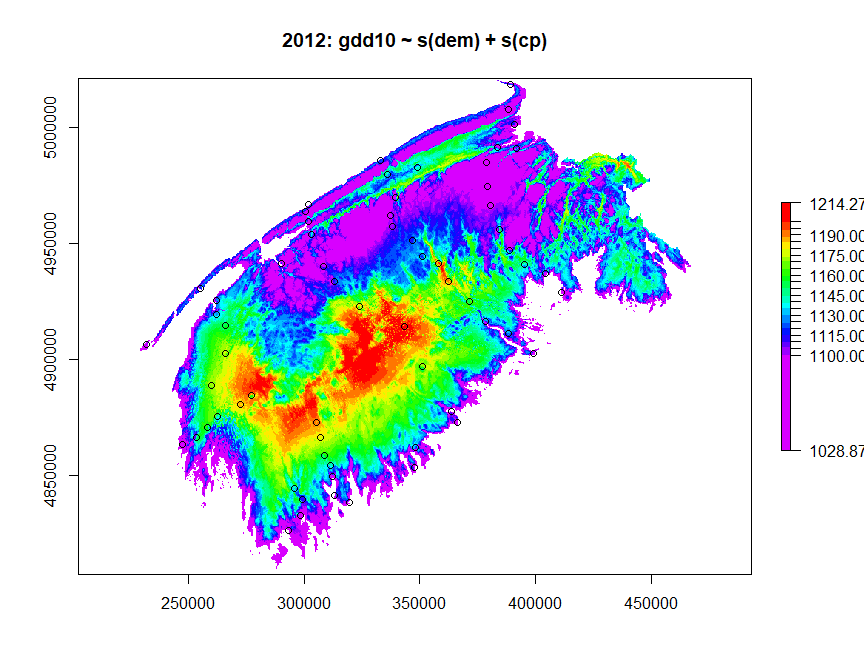
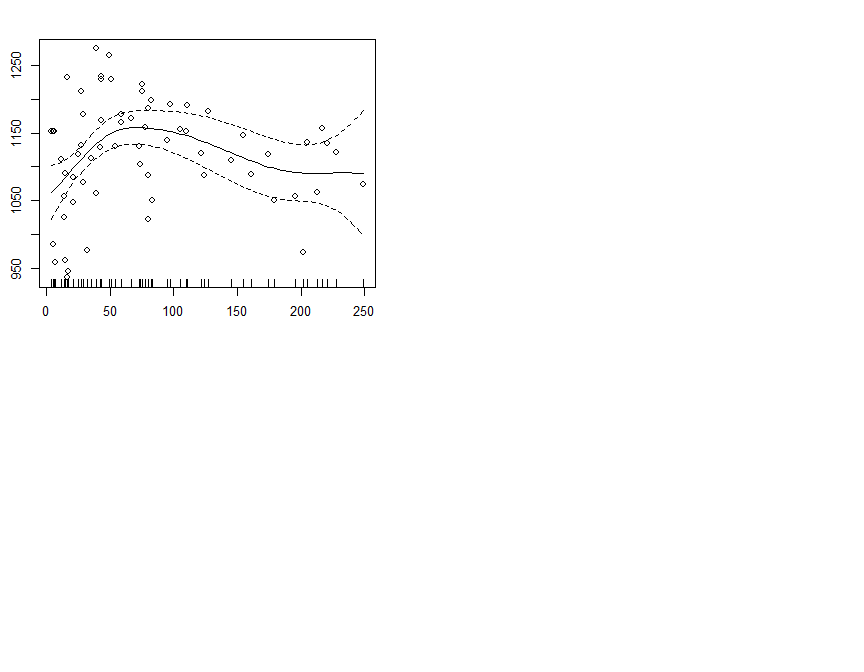
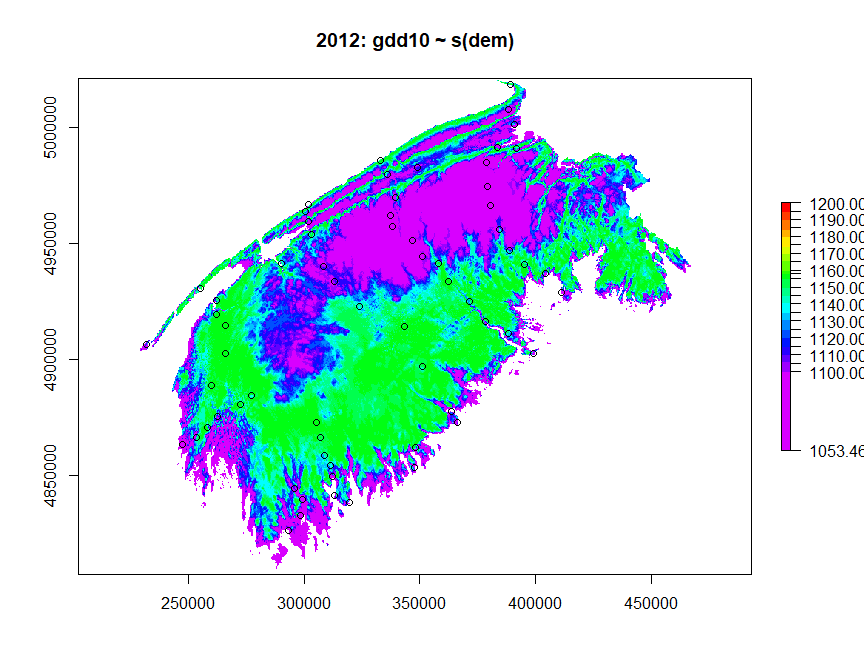
# transform  
rasters\_brick[[2]]<-rasters\_brick[[2]]^0.5  
rasters\_brick[[6]] <- rasters\_brick[[7]]\*cos(rasters\_brick[[6]])  
  
# names  
names(rasters\_brick) <- var\_names  
  
# extract rasters to dataframe  
df\_in <- stations\_df %>%   
 extract\_constant\_raster\_values(var\_names,rasters\_brick) %>% add\_date\_columns()

This R code creates a separate data frame for each year between 2012 and 2016. The code begins by filtering the data frame df\_in to only include data from certain stations listed in nscc\_stations\_list and removing some unwanted stations. The resulting data frame df is then used to create separate data frames for each year using the dplyr::filter function to select observations between April and November of each year. The group\_by function is used to group the data by station ID, and the dplyr::filter function is used to select only the observation with the latest date within each group. Finally, the resulting data frames are named df\_12, df\_13, df\_14, df\_15, and df\_16 for the years 2012 to 2016, respectively. The resulting data frames contain only the latest observation for each station for each year, and can be used for further analysis or modeling.

#Create a dataframe for each year  
# calculate gdd10 in dataframe  
df <- df\_in %>%  
 dplyr::filter(stationid %in% nscc\_stations\_list|  
 # stationid == '47187' | # halifax  
 stationid == '6354'   
 )%>%  
 dplyr::filter(stationid!='S100') %>% # 2012  
 dplyr::filter(stationid!='S160') %>% # 2012, early in season/cold  
 dplyr::filter(stationid!='S20') %>% #2012  
 dplyr::filter(stationid != 'S60') %>% #2012  
 dplyr::filter(stationid != 'S80') %>% #2012  
 dplyr::filter(stationid != 'CL1')  
modelling\_stations <- df %>% filter(date\_time == ymd('2012-06-01'))  
coordinates(modelling\_stations) <- ~ EASTING+NORTHING  
  
df\_list <- list()  
df\_list[[1]] <- df %>% dplyr::filter(year=='2012'&between(month,4,11)) %>%  
 group\_by(stationid)%>%  
 dplyr::filter(date\_time == max(date\_time))  
  
df\_list[[2]] <- df %>% dplyr::filter(year=='2013'&between(month,4,11)) %>%  
 group\_by(stationid)%>%  
 dplyr::filter(date\_time == max(date\_time))  
  
df\_list[[3]] <- df %>% dplyr::filter(year=='2014'&between(month,4,11)) %>%  
 group\_by(stationid)%>%  
 dplyr::filter(date\_time == max(date\_time))  
  
df\_list[[4]] <- df %>% dplyr::filter(year=='2015'&between(month,4,11)) %>%  
 group\_by(stationid)%>%  
 dplyr::filter(date\_time == max(date\_time))  
  
df\_list[[5]] <- df %>% dplyr::filter(year=='2016'&between(month,4,11)) %>%  
 group\_by(stationid)%>%  
 dplyr::filter(date\_time == max(date\_time))  
  
avg\_df <- df %>%  
 filter(between(year(date\_time),2012,2016)&  
 month(date\_time)==11 &   
 day(date\_time)==30) %>%  
 group\_by(stationid) %>%  
 mutate(gdd10\_5ya = mean(gdd10)) %>%  
 ungroup()  
df\_test <- df\_list[[1]]

### Forward stepwise modelling

# Create models using the land-surface variables  
m\_list <- list()  
m\_list[[1]]<- gam(gdd10~s(dem),data=df\_test,method='REML')  
m\_list[[2]]<- gam(gdd10~s(dem)+s(cp),data=df\_test,method='REML')  
m\_list[[3]]<- gam(gdd10~s(dem)+s(cp)+s(asp),data=df\_test,method='REML')  
m\_list[[4]]<- gam(gdd10~s(dem)+s(cp)+s(slp),data=df\_test,method='REML')  
m\_list[[5]]<- gam(gdd10~s(dem)+s(cp)+s(tpi5000m),data=df\_test,method='REML')  
m\_list[[6]]<- gam(gdd10~s(dem)+s(cp)+s(east),data=df\_test,method='REML')  
m\_list[[7]]<- gam(gdd10~s(dem)+s(cp)+s(north),data=df\_test,method='REML')  
m\_list[[8]]<- gam(gdd10~s(dem)+s(cp)+s(north)+s(east,north),data=df\_test,method='REML')  
m\_list[[9]]<- gam(gdd10~s(dem)+s(cp)+s(east,north),data=df\_test,method='REML')  
  
  
for(m in m\_list){  
 gam\_raster <- raster::predict(rasters\_brick,m)  
 plot(gam\_raster,  
 main=paste0('2012: ',paste(deparse(m$formula),collapse=''))   
 ,col=rev(rainbow(26)[1:22])  
 , breaks = c(gam\_raster@data@min,seq(1100,1200,by=5),gam\_raster@data@max)  
 )  
 points(modelling\_stations)  
 par(mfrow=c(2,2))  
 plot.gam(m, residuals = TRUE, pch = 1, cex = 1, shift = coef(m)[1])  
 par(mfrow=c(1,1))  
}

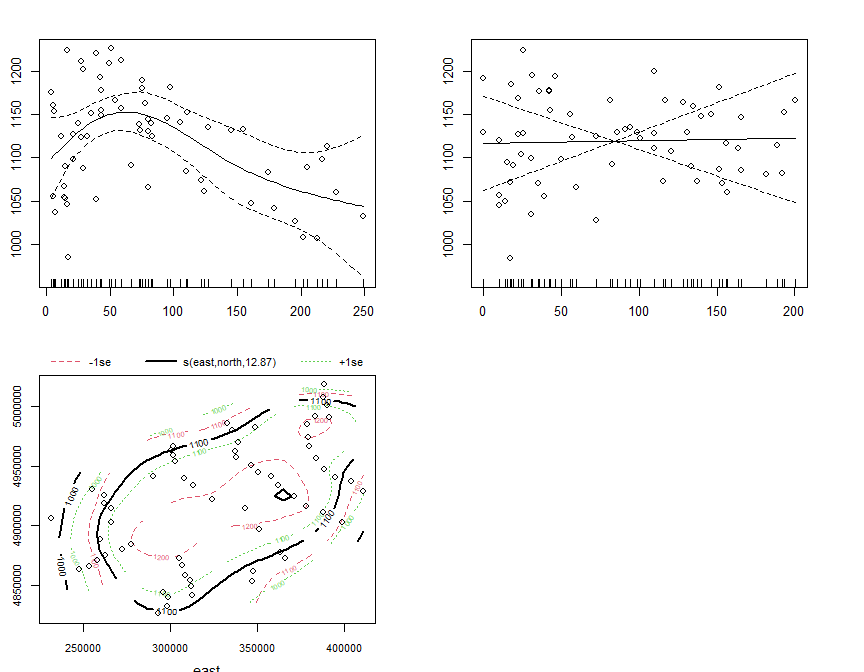
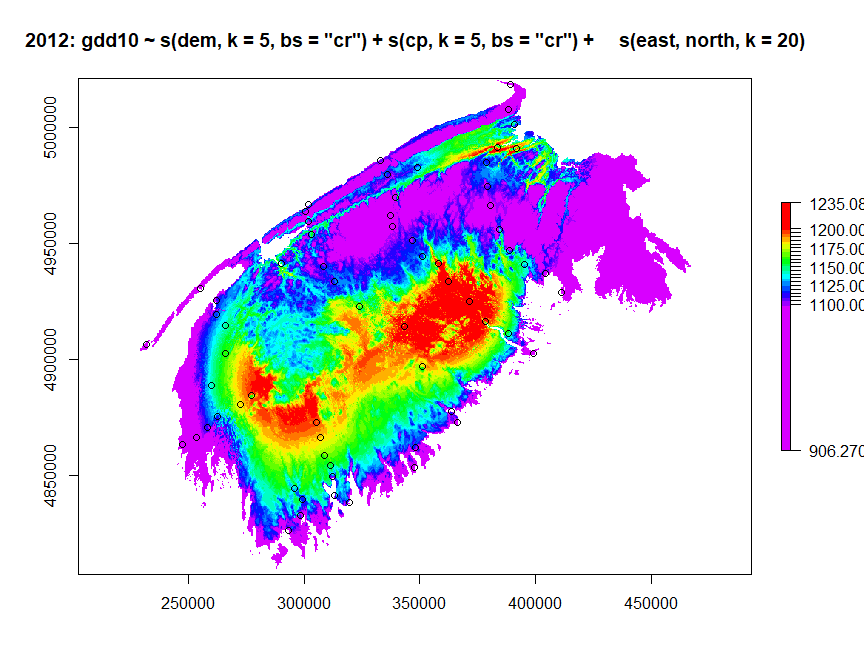
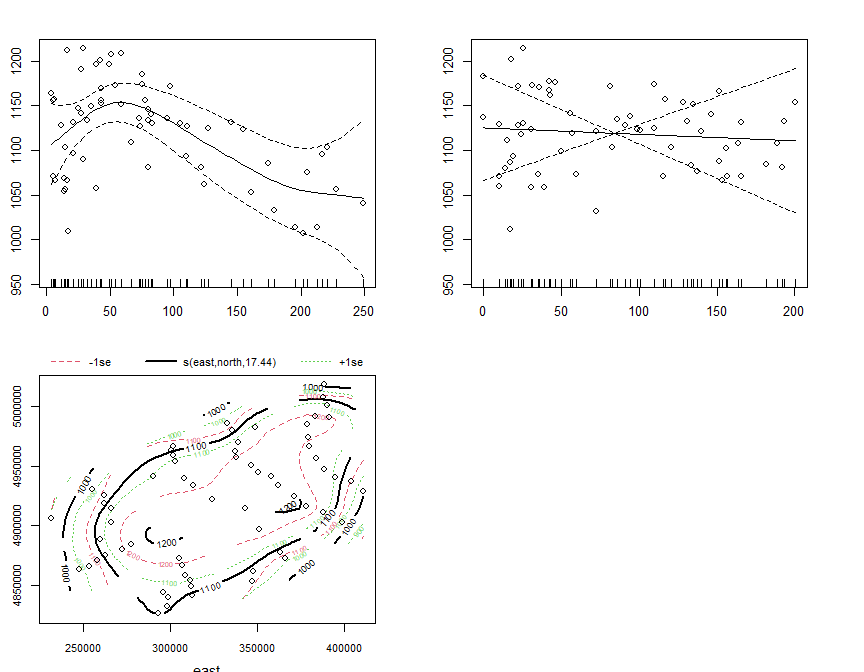
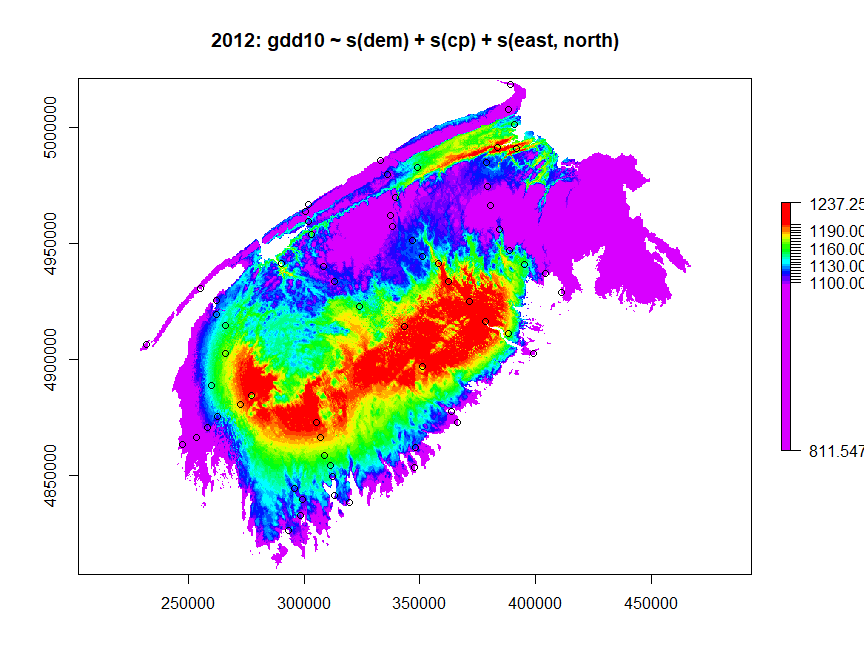


## Five-year average

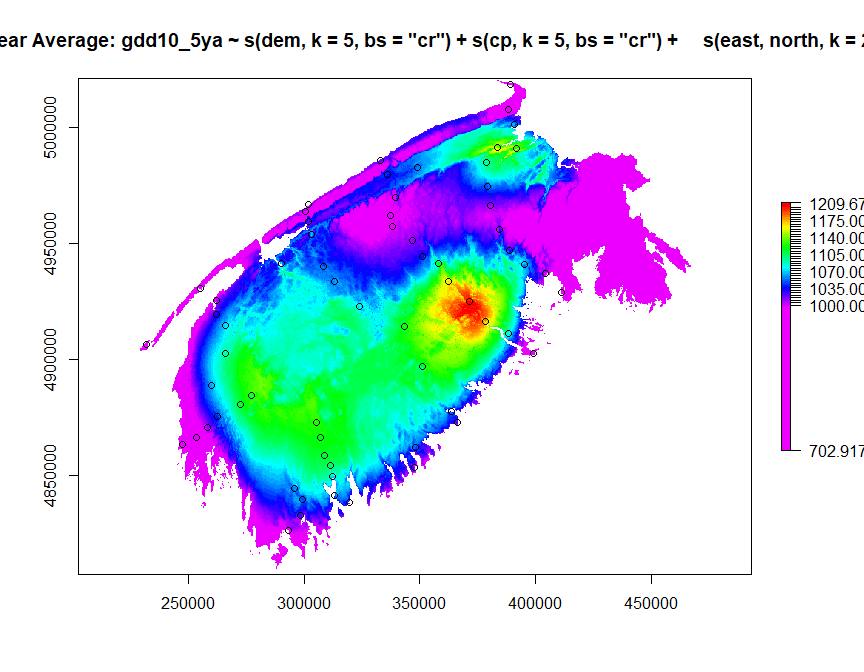
avg\_df <- df %>%  
 filter(between(year(date\_time),2012,2016)&  
 month(date\_time)==11 &   
 day(date\_time)==30) %>%  
 group\_by(stationid) %>%  
 mutate(gdd10\_5ya = mean(gdd10)) %>%  
 ungroup()

## Experiment with GAM parameters

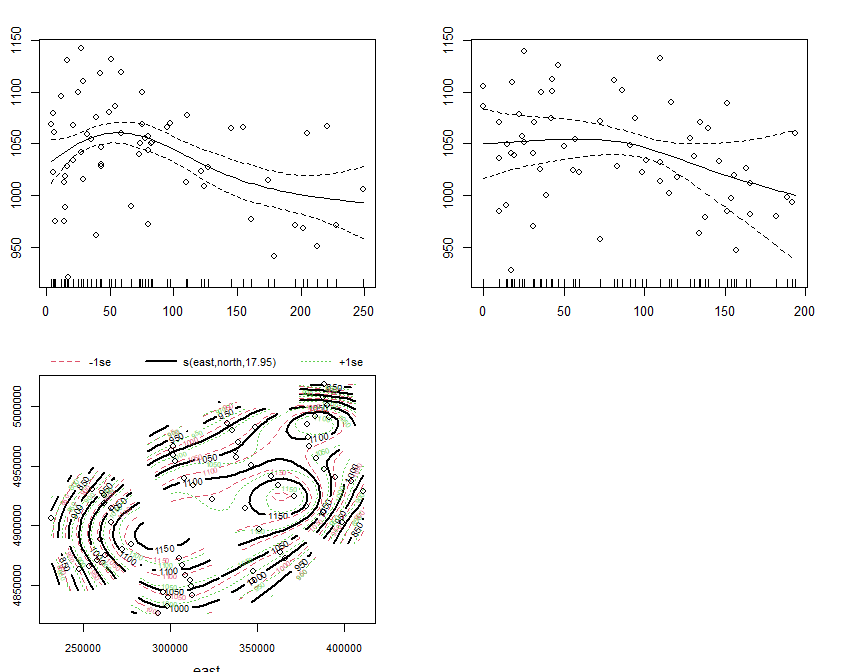
m\_list <- list()  
m\_list[[1]]<-gam(gdd10~s(dem)+s(cp)+s(east,north),data=df\_test,method='REML')  
m\_list[[2]]<- gam(gdd10~s(dem,k=5,bs='cr')+s(cp,k=5,bs='cr')+s(east,north,k=20),data=df\_test,method='REML')  
for(m in m\_list){  
 gam\_raster <- raster::predict(rasters\_brick,m)  
 plot(gam\_raster,  
 main=paste0('2012: ',paste(deparse(m$formula),collapse=''))  
 ,col=rev(rainbow(26)[1:22])  
 , breaks = c(gam\_raster@data@min,seq(1100,1200,by=5),gam\_raster@data@max))  
 points(modelling\_stations)  
 par(mfrow=c(2,2))  
 plot.gam(m, residuals = TRUE, pch = 1, cex = 1, shift = coef(m)[1])  
 par(mfrow=c(1,1))  
}



m <- gam(gdd10\_5ya~s(dem,k=5,bs='cr')+s(cp,k=5,bs='cr')+s(east,north,k=20),data=avg\_df,method='REML',family=gaussian())  
  
  
  
gam\_raster <- raster::predict(rasters\_brick,m)  
plot(gam\_raster,   
 main=paste0('Five Year Average: ',paste(deparse(m$formula),collapse='')),   
 col=rev(rainbow(50)[1:42]),  
 breaks = c(gam\_raster@data@min,seq(1000,1200,by=5),gam\_raster@data@max)  
 )  
points(modelling\_stations)



par(mfrow=c(2,2))  
plot.gam(m, residuals = TRUE, pch = 1, cex = 1, shift = coef(m)[1])  
par(mfrow=c(1,1))



## Other years