Pre-fire vegetation

Nils Rietze

2025-02-03

User functions

```
read_hls_data_frames <- function(index_name, UTM_TILE_ID,year,
                                 sample_points, dnbr_sample){
  filename <- sprintf("%s_sampled_%s_%s.csv",index_name,UTM_TILE_ID,year)

  df <- read.csv2(paste0(TABLE_DIR,filename)) %>%
    select(-1) %>%
    mutate(dNBR = dnbr_sample$dNBR) %>%
    as_tibble() %>%
    mutate(burn_date = sample_points$burn_date)

  if (index_name == "LST"){
    fmt = "X%Y.%m.%d"
  } else{
    fmt = "X%Y.%m.%d.%H.%M.%S"
  }

  df_long <- df %>%
    mutate(ObservationID = 1:nrow(df)) %>%  # Add a column for observation IDs (row numbers)
    gather(key = "Time", value = index_name,
           -c(ObservationID,dNBR,burn_date)) %>%
    mutate(Time = as.POSIXct(Time, format = fmt))

  return(df_long)
}
```

0. Load and prepare data

1. Descriptive statistics

a. Distribution of dNBR

How does the dNBR distribution look for the entire fire event perimeter? Below is the histogram and the corresponding dNBR raster:

```
binwidth <- 20

# Plot dNBR map
p1 <- ggplot() +
  geom_spatraster(data = dnbr_in_perimeter) +
```

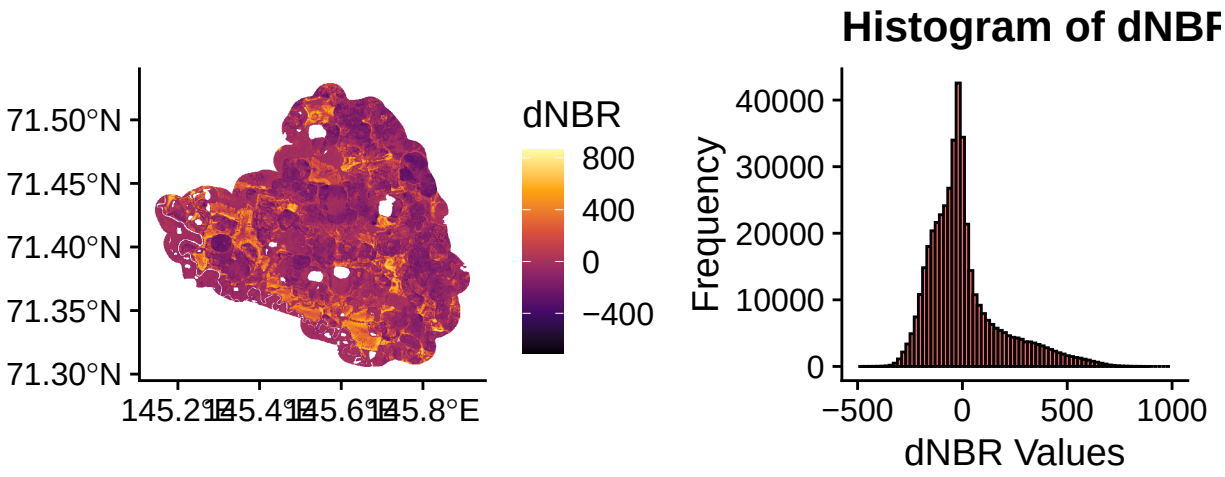
```
  scale_fill_viridis_c(option = "inferno",
                       na.value = "white",
                       name = "dNBR") +
  theme_cowplot()
```

```
## <SpatRaster> resampled to 500640 cells.
```

```
# Plot dNBR for this map
p2 <- ggplot(data = dnbr_in_perimeter, aes(x = dNBR)) +
  geom_histogram(binwidth = binwidth, color = "black", fill = "indianred1") +
  xlim(c(-500,1000)) +
  labs(
    title = "Histogram of dNBR",
    x = "dNBR Values",
    y = "Frequency"
  ) +
  theme_cowplot()

(pg <- p1 + p2)
```

```
## Warning: Removed 325769 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```




```
# ggsave2(pg,filename = "figures/expl_dNBR_hist.png",width = 8,height = 6,bg = "white")
```

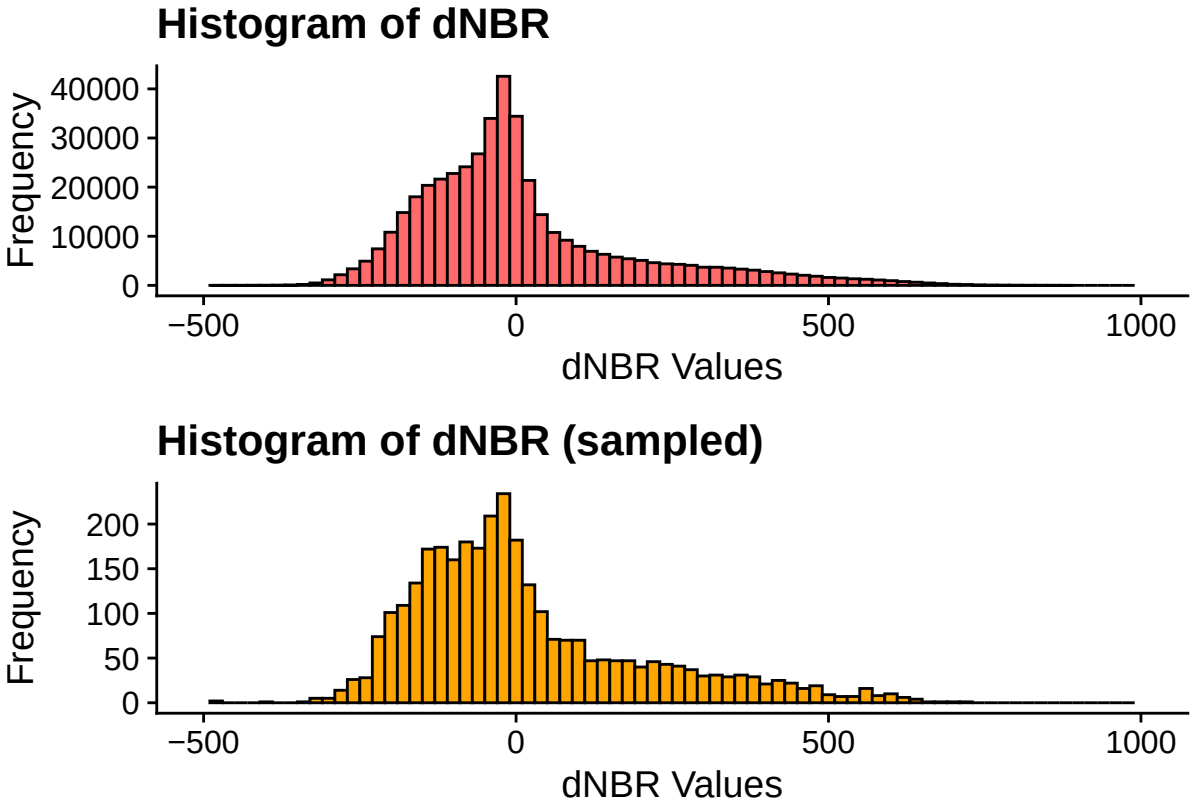
and what does it look like for the sampled points? Remember the points were sampled using a stratification from a binned dNBR histogram, so the histograms should be similar.

```
# Plot distribution of total raster vs. sample points
p3 <- ggplot(data = dnbr_sample, aes(x = dNBR)) +
  geom_histogram(binwidth = binwidth, color = "black", fill = "orange") +
  xlim(c(-500,1000)) +
  labs(
    title = "Histogram of dNBR (sampled)",
    x = "dNBR Values",
    y = "Frequency"
  ) +
  theme_cowplot()
```

```
(pg <- p2 / p3 )
```

```
## Warning: Removed 325769 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
## Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



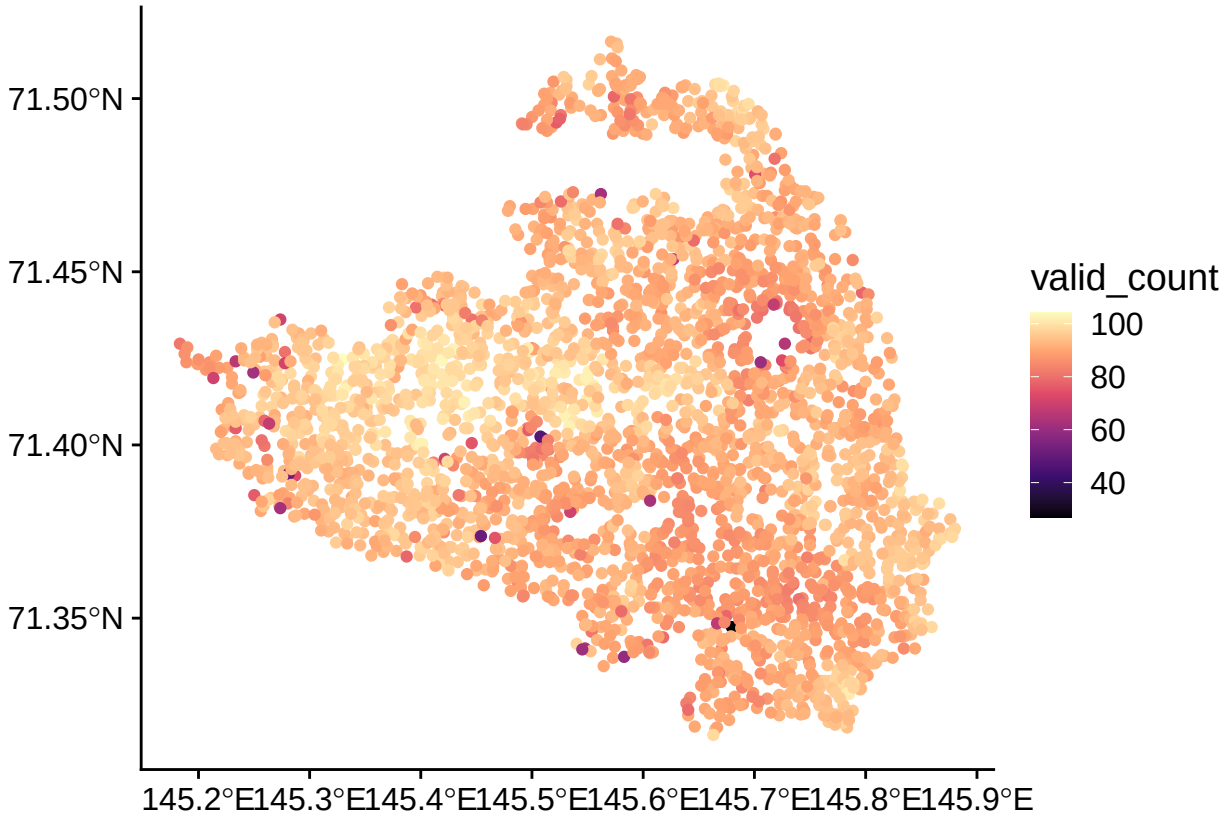

```
# ggsave2(pg, filename = "figures/expl_dNBR_hist_vs_sampled.png",
#         width = 8, height = 6,bg = "white")
```

b. Check nr. of available observations per point

Let's see how many non-NA observations each sampled point has:

```
# get nr. of valid observations per point
valid_counts_by_id <- df_ndmi %>%
  group_by(ObservationID) %>%
  summarise(valid_count = sum(!is.na(NDMI)))

# Add this info to the spatvect point layer
sample_points_filtered <- sample_points %>%
  merge(valid_counts_by_id, by = "ObservationID")

# Plot map and show number of observations per points
(p <- ggplot() +
    # geom_spatraster(data = dnbr_in_perimeter, aes(fill = dNBR)) +
```

```
    geom_spatvector(data = sample_points_filtered, aes(color = valid_count)) +
    scale_color_viridis_c(option = "magma",
                          na.value = "transparent") +
    labs(fill = "Number of non-masked observations") +
    theme_cowplot()
)
```




```
# ggsave2(p,
#         filename = "figures/Nobs_randompoints_map.png",
#         bg = "white",width = 10, height = 8)
```

Here are the summary statistics for the nr. of valid observations

```
# Report summary stats for pre-fire observations
df_filtered %>%
  group_by(ObservationID) %>%
  filter(BeforeBurnDate) %>%
  summarise(
    n_observations = n()
  ) %>%
  summarise(
    mean_obs = mean(n_observations),
    median_obs = median(n_observations),
    min_obs = min(n_observations),
    max_obs = max(n_observations)
  )
```

```
## # A tibble: 1 x 4
##   mean_obs median_obs min_obs max_obs
```

```
##       <dbl>       <int>    <int>    <int>
## 1      39.0          40       25       43
```

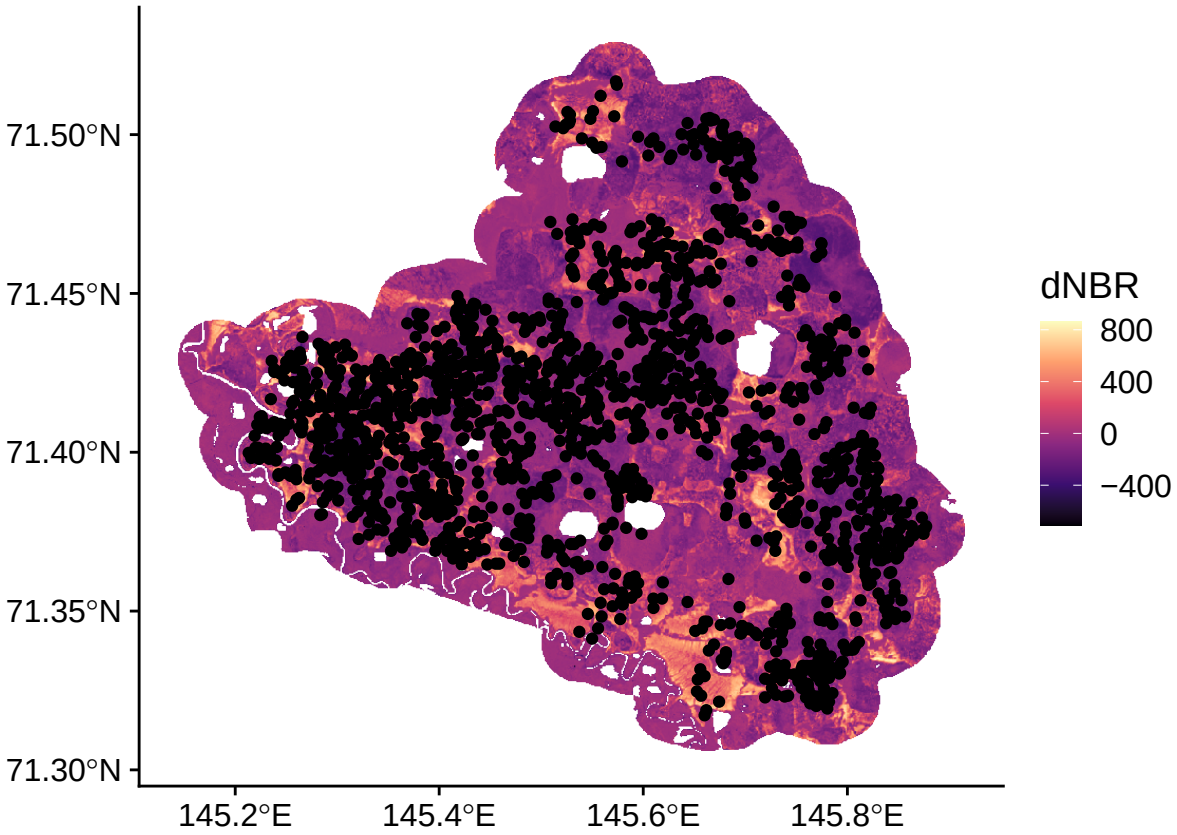
What does the histogram of nr. of non-na observations look like? I added a vertical line at a percentile-threshold.

b. Points filtered by nr. of non-na observations

Where are these filtered points? Here's a map of them:

```
# Plot where these filtered points sit
sample_points_filtered <- sample_points %>%
  filter(ObservationID %in% df_filtered$ObservationID)

(p <- ggplot() +
  geom_spatraster(data = dnbr_in_perimeter, aes(fill = dNBR)) +
  geom_spatvector(data = sample_points_filtered) +
  scale_fill_viridis_c(option = "magma",
                       na.value = "transparent") +
  labs(fill = "dNBR") +
  theme_cowplot()
)
```

```
## <SpatRaster> resampled to 500640 cells.
```



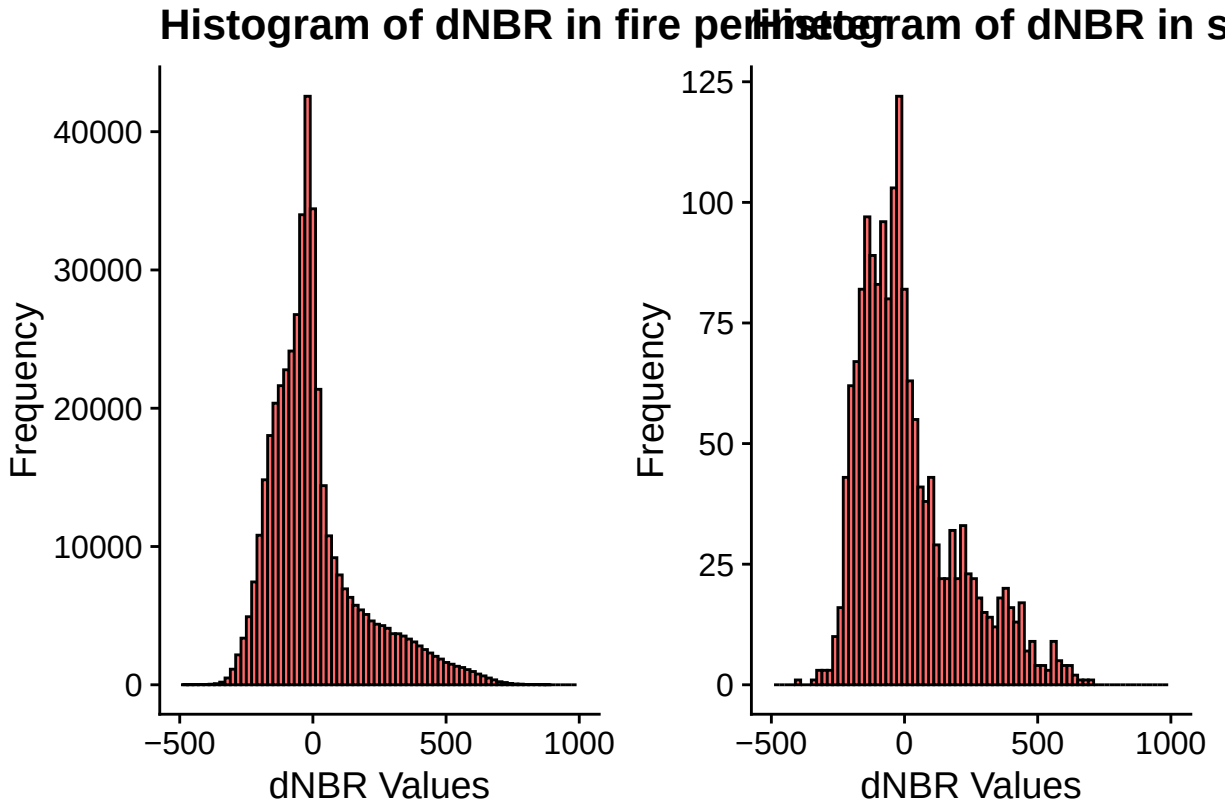

```
# ggsave2(p,
#         filename = sprintf("figures/Location_randompoints_%sth_pctile.png",pct_cutoff*100),
#         bg = "white",width = 10, height = 8)
```

Is the dNBR distribution for this final subset still similar to the original dNBR distribution?

```r
p1 <- ggplot(data = dnbr_in_perimeter, aes(x = dNBR)) +
  geom_histogram(binwidth = binwidth, color = "black", fill = "indianred1") +
  xlim(c(-500,1000)) +
  labs(
    title = "Histogram of dNBR in fire perimeter",
    x = "dNBR Values",
    y = "Frequency"
  ) +
  theme_cowplot()

p2 <- df_filtered %>%
  group_by(ObservationID) %>%
  summarise(dNBR = first(dNBR)) %>%
  ggplot(data = ., aes(x = dNBR)) +
  geom_histogram(binwidth = binwidth, color = "black", fill = "indianred1") +
  xlim(c(-500,1000)) +
  labs(
    title = "Histogram of dNBR in subset",
    x = "dNBR Values",
    y = "Frequency"
  ) +
  theme_cowplot()

(pg <- p1 + p2)
```

```
## Warning: Removed 325769 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
## Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



The sampled subset cutting off points with fewer than 80 observations seems to resemble the overall dNBR distribution but cuts off high-severity values. So probably, the cutoff is too conservative.

The question now is a trade-off of capturing the distribution better or avoiding more spatial autocorrelation.

c. Check spatial autocorrelation of final point sample

```
library(spatstat)
```

```
## Loading required package: spatstat.data
## Loading required package: spatstat.geom
## spatstat.geom 3.2-7
##
## Attaching package: 'spatstat.geom'
## The following object is masked from 'package:patchwork':
##
##     area
## The following objects are masked from 'package:terra':
##
##     area, delaunay, is.empty, rescale, rotate, shift, where.max,
##     where.min
## Loading required package: spatstat.random
## spatstat.random 3.2-1
## Loading required package: spatstat.explore
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## spatstat.explore 3.2-5

## Loading required package: spatstat.model

## Loading required package: rpart

## spatstat.model 3.2-8

## Loading required package: spatstat.linnet

## spatstat.linnet 3.1-3

##
## spatstat 3.0-6
## For an introduction to spatstat, type 'beginner'
```

```
library(spdep)
```

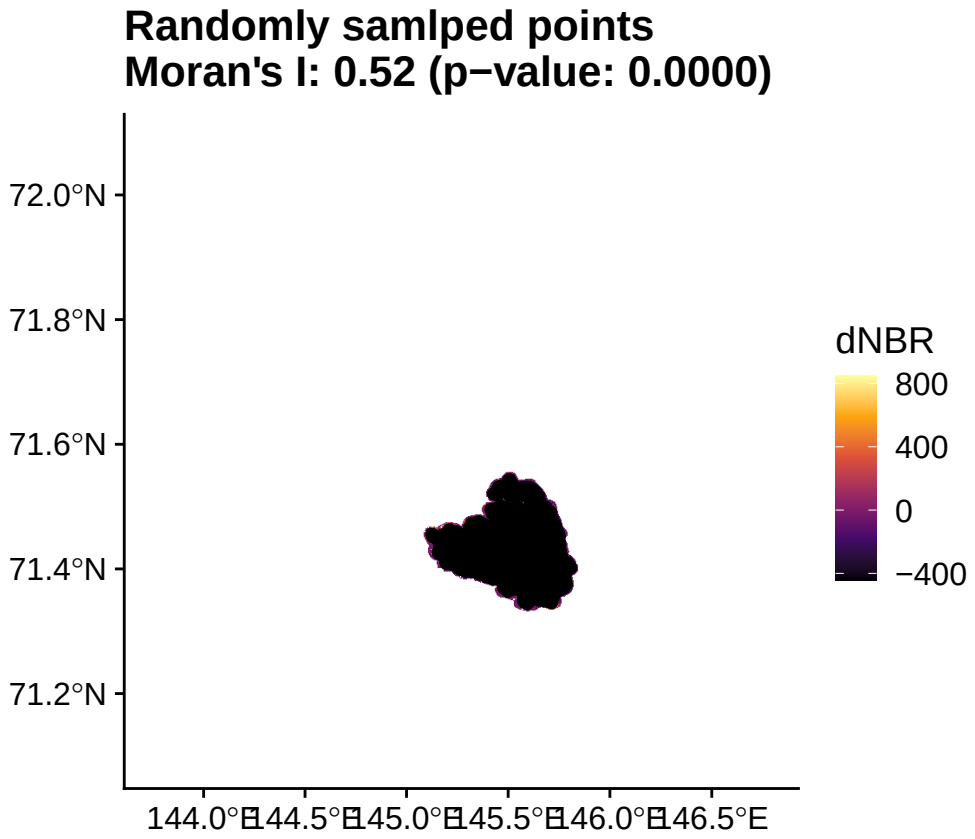
```
## Loading required package: spData

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`

## Loading required package: sf

## Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
knn <- knn2nb(knearneigh(dnbr_sample[c("x","y")], k = 5))
```

```
## Warning in knearneigh(dnbr_sample[c("x", "y")], k = 5): knearneigh: identical
## points found

## Warning in knearneigh(dnbr_sample[c("x", "y")], k = 5): knearneigh: kd_tree not
## available for identical points
```

```
# Convert neighbors to weights
weights <- nb2listw(knn, style = "W")

# Compute Moran's I
moran_i <- moran.test(dnbr_sample$dNBR, listw = weights)

(ggplot() +
  geom_spatraster(data = mask(dnbr,fire_perimeter_buffered)) +
  scale_fill_viridis_c(option = "inferno",
                       na.value = "white",
                       name = "dNBR") +
  geom_spatvector(data = sample_points) +
  labs(title = sprintf("Randomly samlped points\nMoran's I: %.2f (p-value: %.4f)",
                       moran_i$estimate[1],moran_i$p.value)) +
  theme_cowplot())
```

```
## <SpatRaster> resampled to 501264 cells.
```




```
# ggsave2("figures/stratified_sample_points_moranI.png",bg = "white")
```

2. Explore pre-fire time series data

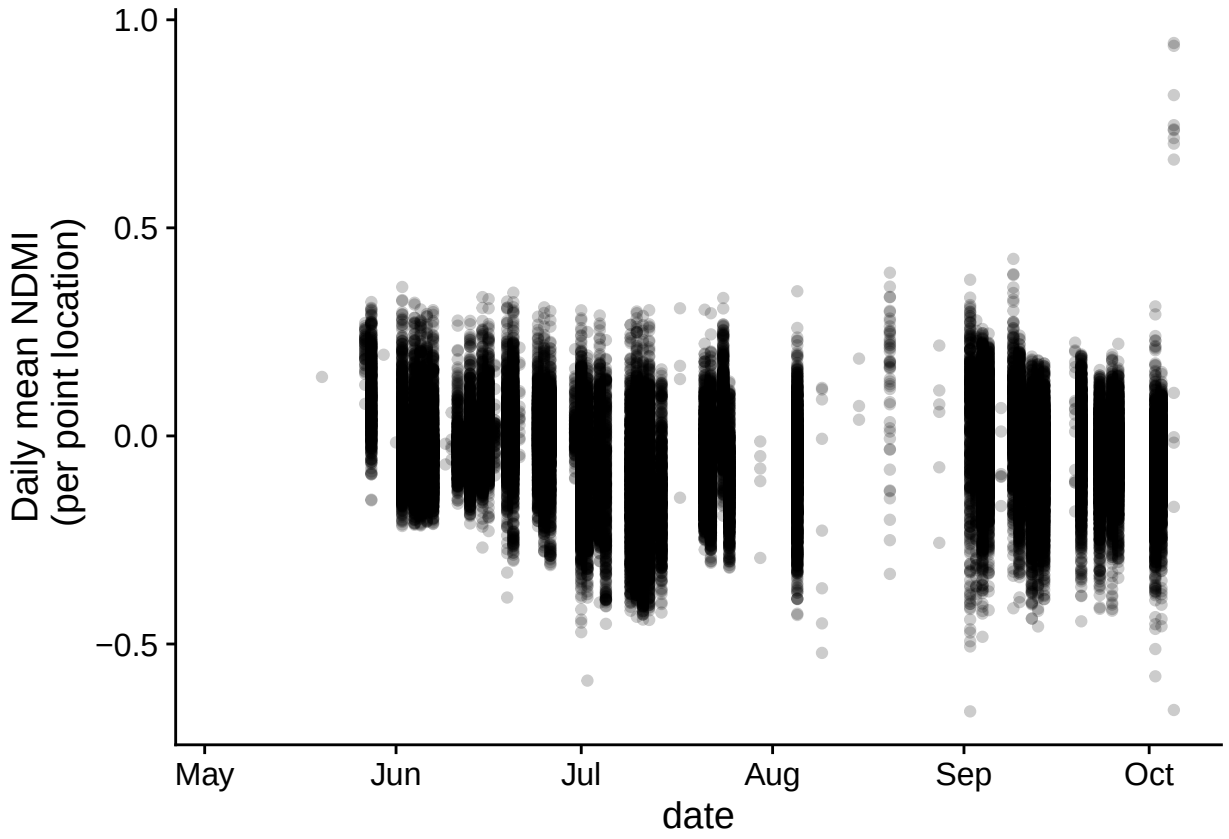
a. Spectral indices

i. Overall time series

The time series for all points looks like this:

```
# Time series for all points
ggplot(df_filtered) +
  geom_point(aes(x = date,y = DailyMeanNDMI),alpha = .2) +
  labs(y = "Daily mean NDMI\n (per point location)") +
  theme_cowplot()
```

```
## Warning: Removed 123611 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



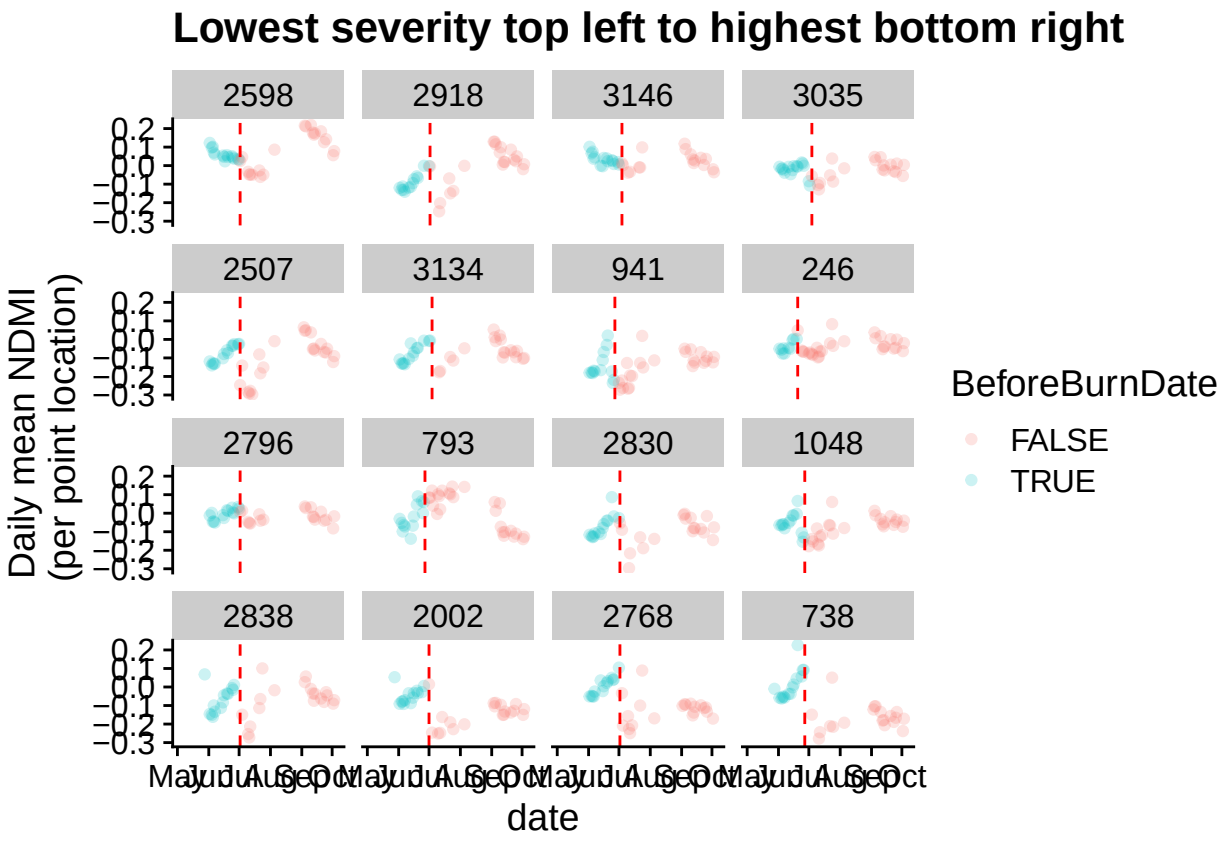

```
# ggsave2("figures/Timeseries_NDMI_randompoints.png",
#         bg = "white",width = 10, height = 8)
```

ii. individual point's TS

A look at individual time series reveals diverse trends before the fire:

```
# Time series for some random points
rand_id <- sample(unique(df_filtered$ObservationID), 16)
# rand_id <- c(546,1568,1620,1824,1892,2122,2391,2552,2558,2573,2645,2869,2885,2988,3017,3085)

index_name <- "NDMI"

df_filtered %>%
  filter(ObservationID %in% rand_id) %>%
  mutate(ObservationID = as.factor(ObservationID)) %>%
  mutate(ObservationID = fct_reorder(ObservationID, dNBR)) %>%
  ggplot() +
    geom_point(aes(x = date,y = !!sym(paste0("DailyMean",index_name)),
                   color = BeforeBurnDate),alpha = .2) +
    labs(y = sprintf("Daily mean %s\n (per point location)",index_name),
         title = "Lowest severity top left to highest bottom right") +
    geom_vline(aes(xintercept = as.numeric(burn_date)), color = "red", linetype = "dashed") +
    facet_wrap(~ObservationID) +
    theme_cowplot()
```

```
## Warning: Removed 1176 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# ggsave2(sprintf("figures/%s_perpoint.png",index_name),
#         bg = "white",width = 9, height = 8)
```

iii. Histogram of spectral index values

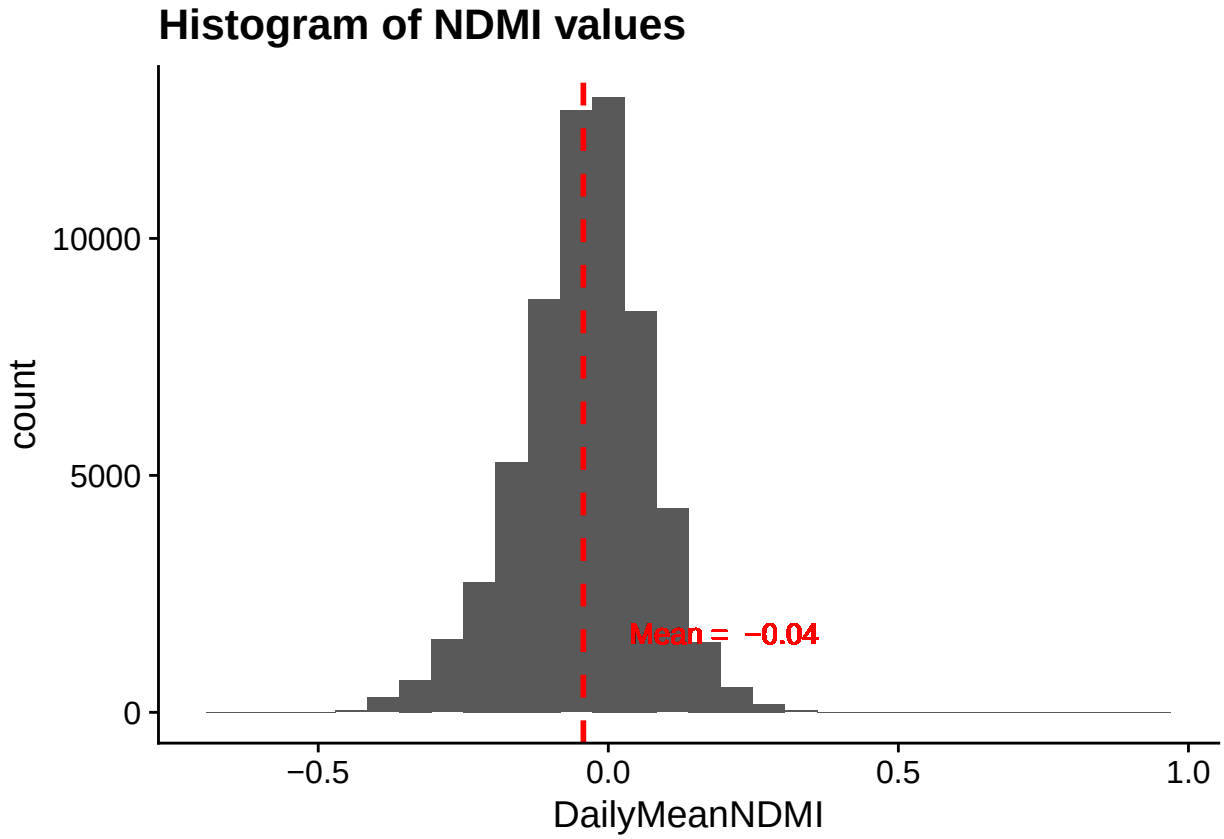
The histogram for this spectral index:

```
mean_index <- df_filtered %>%
  select(contains(index_name)) %>%
  pull() %>% mean(.,na.rm = T)

# Histogram of index values
ggplot(df_filtered) +
  geom_histogram(aes(x = !!sym(paste0("DailyMean",index_name)))) +
  geom_vline(xintercept = mean_index,
             color = "red", linetype = "dashed", size = 1) +
  geom_text(x = 0.2, y = 1e3,
                label = paste("Mean = ",
                              round(mean_index, 2)),
            color = "red", vjust = -1) +
  labs(title = sprintf("Histogram of %s values",index_name)) +
  theme_cowplot()
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 123611 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

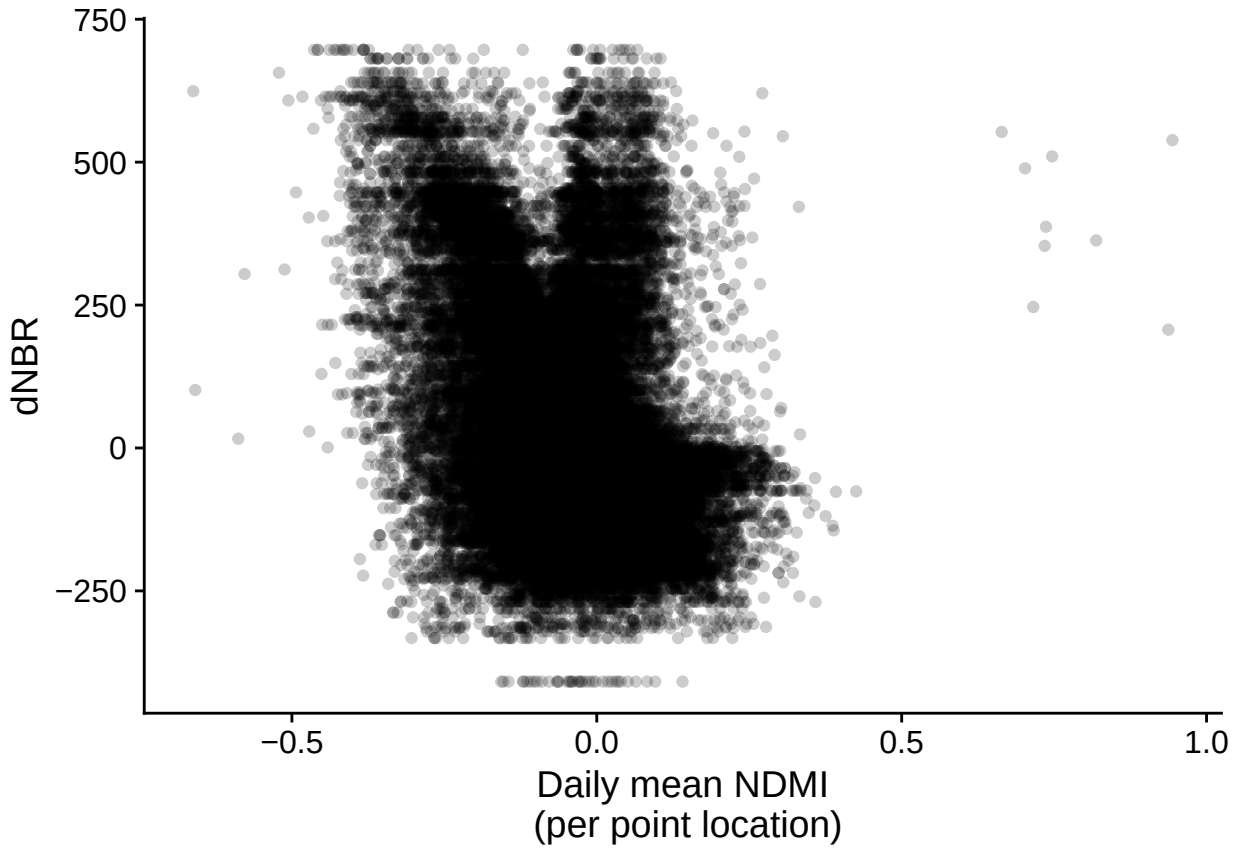



```
# ggsave2(sprintf("figures/Histogram_%s.png",index_name),
#         bg = "white",width = 10, height = 8)
```

iv. Scatter dNBR vs. spectral index

```
# Scatterplot spectral index vs.dNBR
(ggplot(df_filtered) +
  geom_point(aes(x = !!sym(paste0("DailyMean",index_name)),y = dNBR),alpha = .2) +
  labs(x = sprintf("Daily mean %s\n (per point location)",index_name),
       y = "dNBR") +
 theme_cowplot())
```

```
## Warning: Removed 123611 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

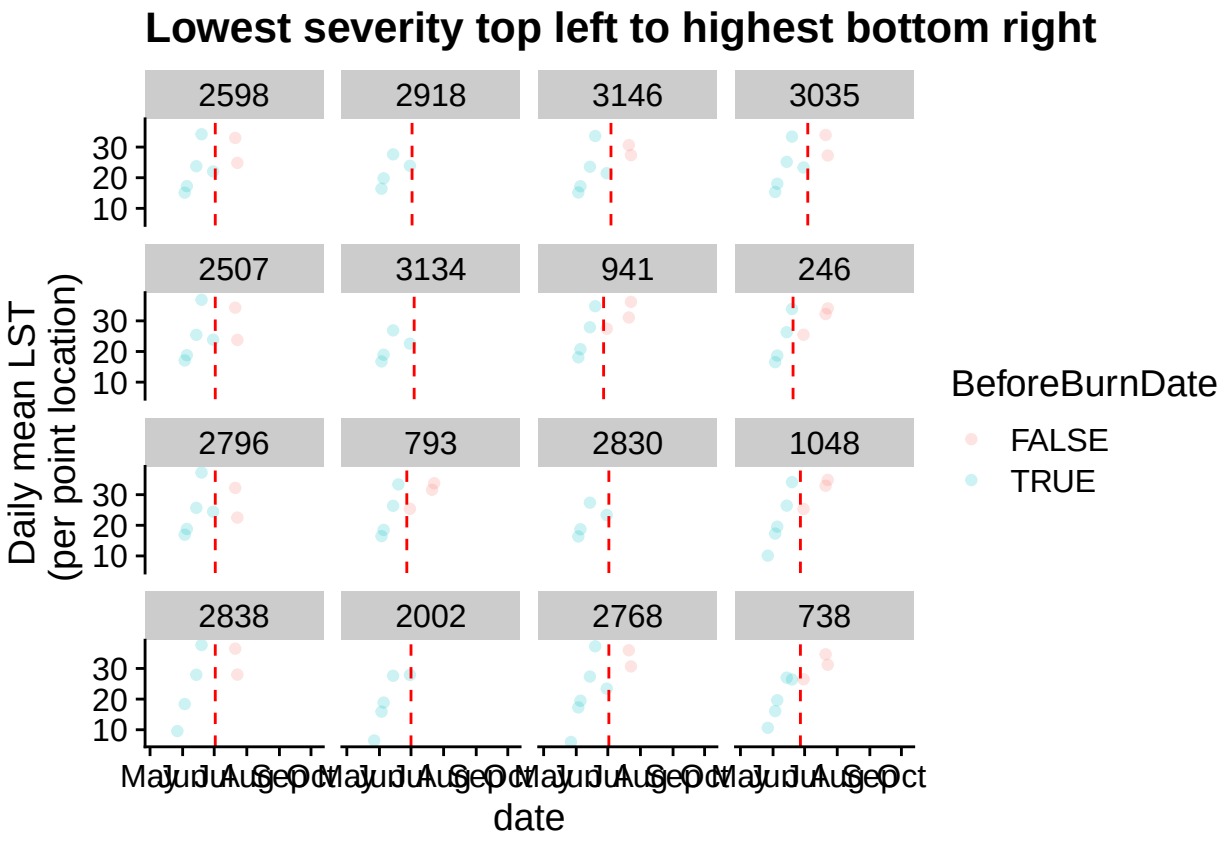


b. LST

The time series of LST for this fire event looks like this:

```
# Plot LST time series
df_filtered %>%
  filter(ObservationID %in% rand_id) %>%
  mutate(ObservationID = as.factor(ObservationID)) %>%
  mutate(ObservationID = fct_reorder(ObservationID, dNBR)) %>%
  ggplot() +
  geom_point(aes(x = date,y = LST, color = BeforeBurnDate),alpha = .2) +
  labs(y = sprintf("Daily mean %s\n (per point location)","LST"),
       title =  "Lowest severity top left to highest bottom right") +
  geom_vline(aes(xintercept = as.numeric(burn_date)), color = "red", linetype = "dashed") +
  facet_wrap(~ObservationID) +
  theme_cowplot()
```

```
## Warning: Removed 1641 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# ggsave2("figures/LST_perpoint.png",bg = "white",width = 10, height = 8)
```

3. Fit curves to pre-fire data

a. Fit linear regression to pre-fire data

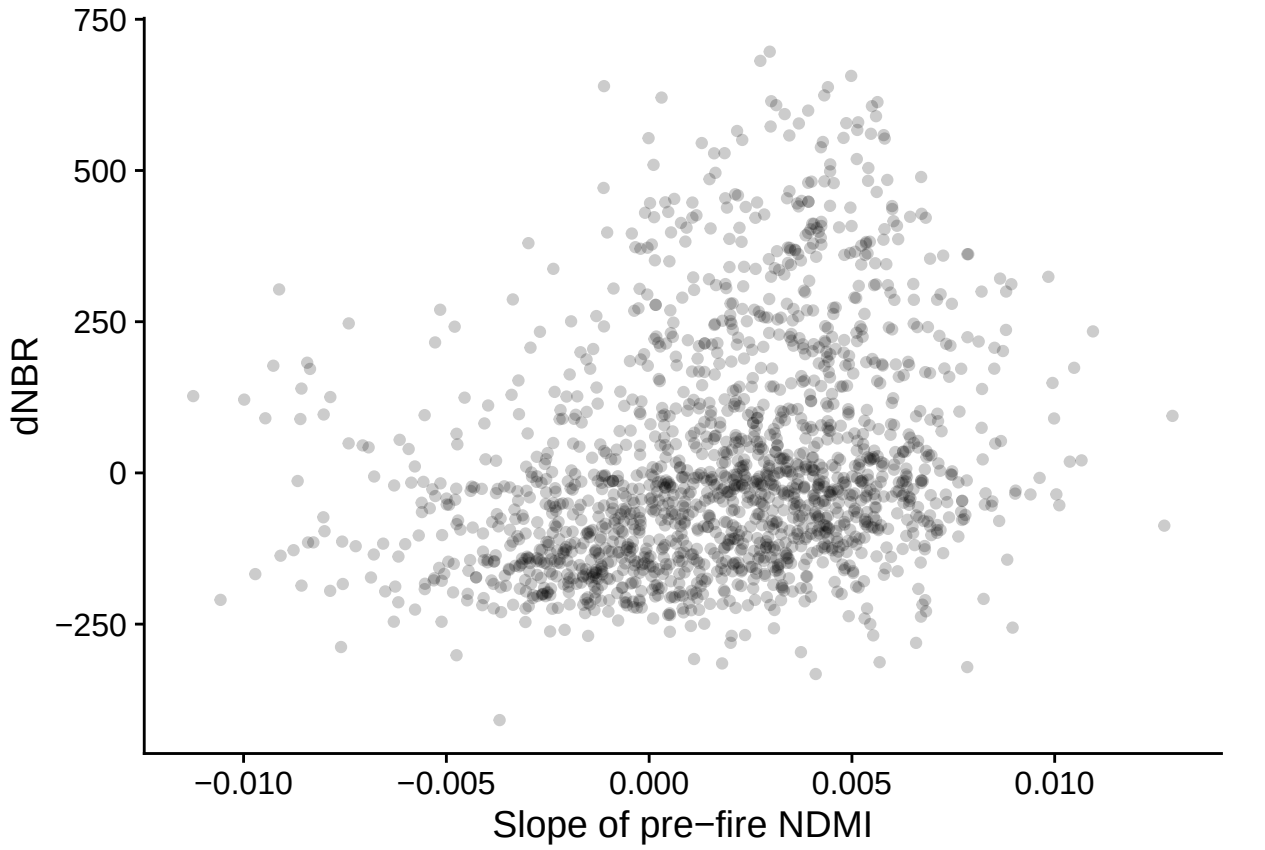
```
# Apply OLS to each time series point
df_mod <- df_filtered %>%
  group_by(ObservationID) %>%
  filter(BeforeBurnDate) %>%
  do(mod = lm(!!sym(paste0("DailyMean",index_name)) ~ date, data = .)) %>%
  mutate(Slope = summary(mod)$coeff[2],
         Intercept = summary(mod)$coeff[1]) %>%
  select(-mod)

df_mod <- df_filtered %>%
  group_by(ObservationID) %>%
  select(dNBR) %>%
  summarize(dNBR = first(dNBR)) %>%
  select(-ObservationID) %>%
  cbind(df_mod)
```

```
## Adding missing grouping variables: `ObservationID`
```

Scatterplot of pre-fire NDMI slope vs. dNBR

```
# Plot slope of index vs. dNBR
ggplot(df_mod) +
```

```
  geom_point(aes(x = Slope,y = dNBR),alpha = .2) +
  labs(x = sprintf("Slope of pre-fire %s",index_name),
       y = "dNBR") +
  theme_cowplot()
```

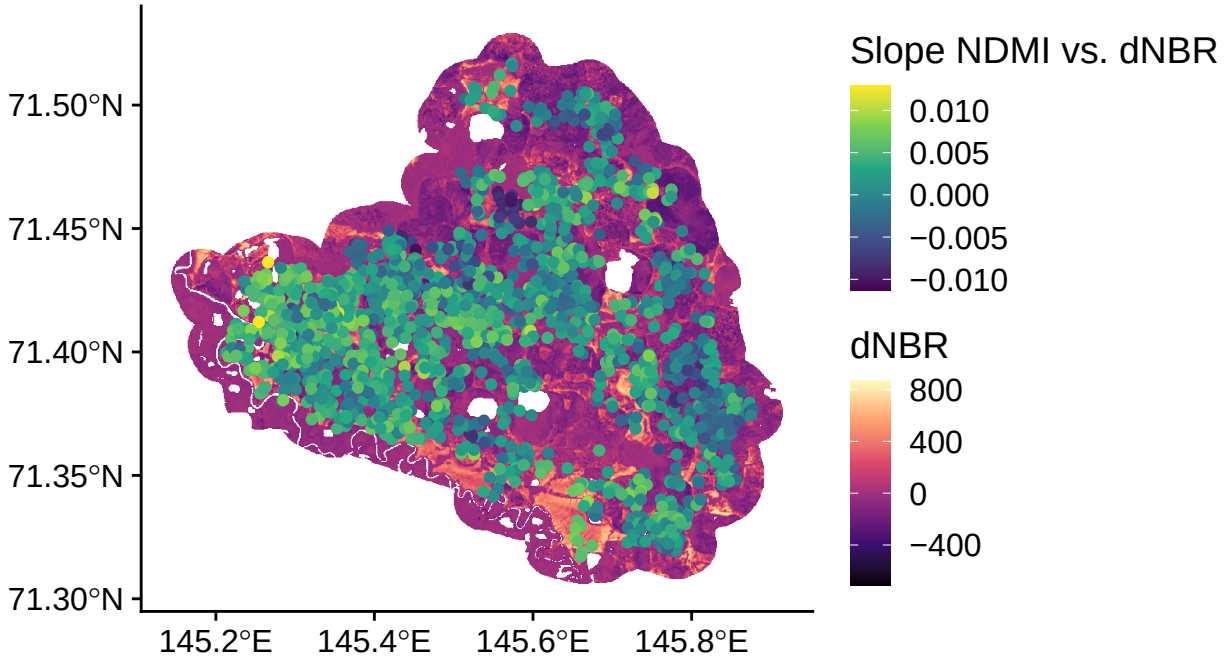



```
# ggsave2(sprintf("figures/%s_vs_dNBR.png",index_name),
#         bg = "white",width = 10, height = 8)
```

Is there a spatial pattern in the OLS coefficients?

```
sample_points_merged <- sample_points %>%
  filter(ObservationID %in% df_mod$ObservationID) %>%
  merge(., df_mod, by = "ObservationID")

(p <- ggplot() +
    geom_spatraster(data = dnbr_in_perimeter,aes(fill = dNBR)) +
    geom_spatvector(data = sample_points_merged, aes(color = Slope)) +
    scale_fill_viridis_c(option = "magma",
                         na.value = "transparent") +
    scale_color_viridis_c(option = "viridis",
                         na.value = "transparent") +
    labs(fill = "dNBR",
         color = sprintf("Slope %s vs. dNBR",index_name)) +
    theme_cowplot()
)
```

```
## <SpatRaster> resampled to 500640 cells.
```




```
# ggsave2(p,
#         filename = sprintf("figures/Map_Slope_per_Point_%s_vs_dNBR_%sth.png",
#                            pct_cutoff*100,index_name),
#         bg = "white",width = 10, height = 8)
```

b. Splines

Setting up user functions.

```
model_fit_smoothedspline <- function(x,y,spar = 0.5) {
  # Using a spline smoother
  smooth.spline(x = x, y = y, spar = spar)
}

model_index_smoothedspline <- function(x,y,full_data, spar = 0.5) {

  # Use function to fit model
  model <- model_fit_smoothedspline(x,y,spar)

  # Generate predictions for curve plotting (for time-period doy 130-300)
  pred <- predict(model, data.frame(doy = 130:300))
  # pred <- unlist(pred$y)

  # use function to find vertex (linear model)
  # vertex <- find_vertex(model)
  # find vertex based on predictions (spline smoother)
  vertex <- data.frame(
    x = pred$x[pred$y == max(pred$y)],
    y = pred$y[pred$y == max(pred$y)]
  )

  # Write necessary values back to df
  data <- suppressMessages(full_join(full_data,
```

```
                                        data.frame(
                                          doy = 130:300,
                                          spline.max = vertex$y,
                                          spline.max.doy = vertex$x,
                                          spline.pred = pred))
                                )
  return(data)
}
```

We can fit a spline for the day of year and spectral index

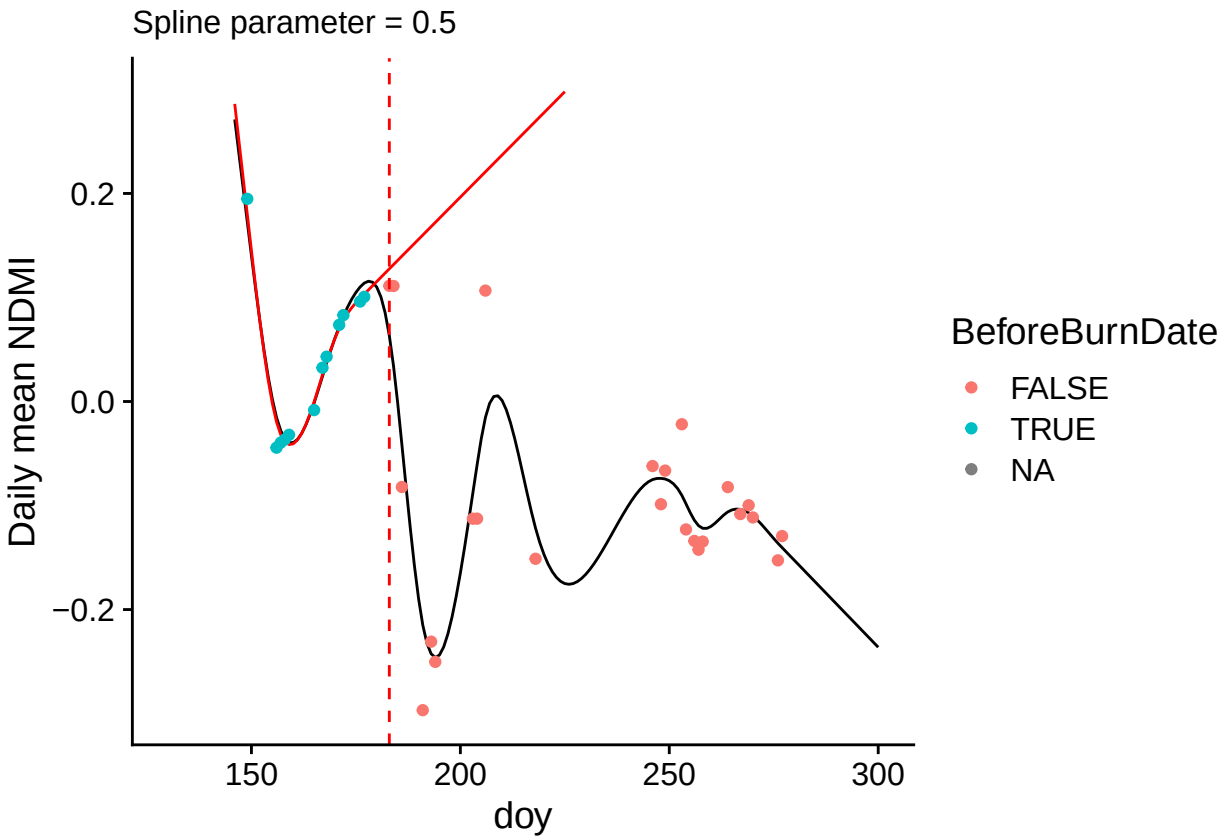
```
df_filtered$doy <- lubridate::yday(df_filtered$date)

df <- df_filtered %>%
  filter(ObservationID == 2391,
         !is.na(DailyMeanNDMI))

test_smooth <- model_index_smoothedspline(df$doy,df$DailyMeanNDMI,df)

df <- df_filtered %>%
  filter(ObservationID == 2391,
         BeforeBurnDate == TRUE,
         !is.na(DailyMeanNDMI))

test_smooth_prefire <- model_index_smoothedspline(df$doy,df$DailyMeanNDMI,df) %>%
  rename(spline.pred.prefire = 'spline.pred.doy.1')

test_smooth_join <- test_smooth %>%
  rename(spline.pred = 'spline.pred.doy.1') %>%
  left_join(test_smooth_prefire %>% select(spline.pred.doy,spline.pred.prefire),
            by = "spline.pred.doy")

ggplot(test_smooth_join) +
  geom_line(aes(x = doy, y = spline.pred)) +
  geom_line(aes(x = doy, y = spline.pred.prefire),color = "red") +
  geom_point(aes(x = doy, y = DailyMeanNDMI,color = BeforeBurnDate)) +
  labs(y = sprintf("Daily mean %s",index_name),
       subtitle = sprintf("Spline parameter = %s",0.5)) +
  geom_vline(aes(xintercept = yday(burn_date)), color = "red", linetype = "dashed") +
  ylim(c(-0.3,0.3)) +
  theme_cowplot()
```

```
## Warning: Removed 16 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 91 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 135 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 135 rows containing missing values or values outside the scale range
## (`geom_vline()`).
```



As we can see, the spline struggles to deal with the outlier at the edge. This outlier from 28 May 2020 is due to the pixel being treated as a valid one (not masked out by Fmask) even though a quick check revealed it was partly snow covered. So maybe an additional snow mask is needed to remove this issue. For now, I'll continue with the outlier in the data.

So maybe a different spline parameter enabling a broader segmentation might help. So I tried spar = 0.7

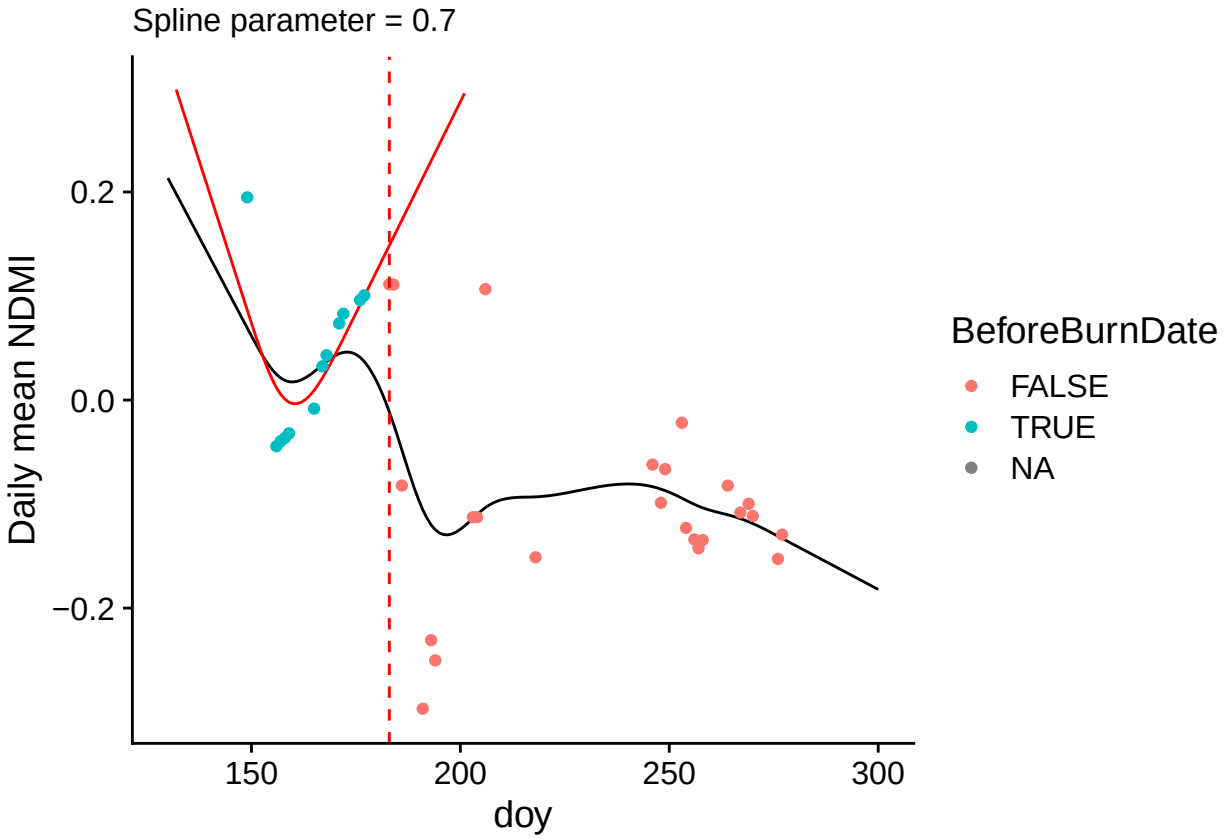
```
spar <- 0.7

df <- df_filtered %>%
  filter(ObservationID == 2391,
         !is.na(DailyMeanNDMI))

test_smooth <- model_index_smoothedspline(df$doy,df$DailyMeanNDMI,df,spar = spar)

df <- df_filtered %>%
  filter(ObservationID == 2391,
         BeforeBurnDate == TRUE,
         !is.na(DailyMeanNDMI))

test_smooth_prefire <- model_index_smoothedspline(df$doy,df$DailyMeanNDMI,df,spar = spar) %>%
  rename(spline.pred.prefire = 'spline.pred.doy.1')

test_smooth_join <- test_smooth %>%
  rename(spline.pred = 'spline.pred.doy.1') %>%
  left_join(test_smooth_prefire %>% select(spline.pred.doy,spline.pred.prefire),
            by = "spline.pred.doy")

ggplot(test_smooth_join) +
```

```
  geom_line(aes(x = doy, y = spline.pred)) +
  geom_line(aes(x = doy, y = spline.pred.prefire),color = "red") +
  geom_point(aes(x = doy, y = DailyMeanNDMI,color = BeforeBurnDate)) +
  labs(y = sprintf("Daily mean %s",index_name),
       subtitle = sprintf("Spline parameter = %s",spar)) +
  geom_vline(aes(xintercept = yday(burn_date)), color = "red", linetype = "dashed") +
  ylim(c(-0.3,0.3)) +
  theme_cowplot()
```

```
## Warning: Removed 101 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 135 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 135 rows containing missing values or values outside the scale range
## (`geom_vline()`).
```



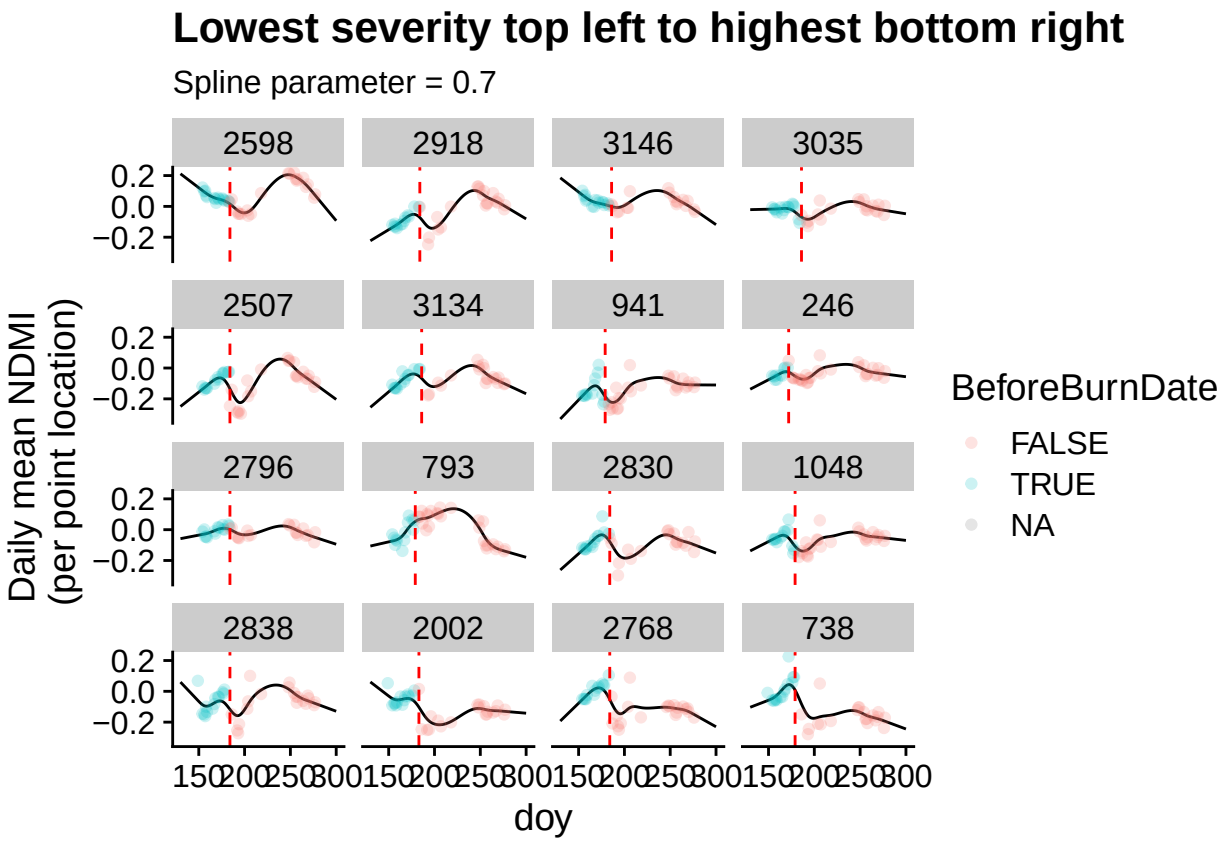

```
spar <- 0.7

ndmi_smooth <- df_filtered %>%
  filter(ObservationID %in% rand_id,
         !is.na(DailyMeanNDMI)) %>%
  group_by(ObservationID) %>%
  group_modify(~model_index_smoothedspline(.x$doy,.x$DailyMeanNDMI,.x,spar = spar))

ndmi_smooth <- ndmi_smooth %>%
  rename(spline.pred = 'spline.pred.doy.1')
```

```r
# Plot multiple time series with spline
(ndmi_smooth %>%
  ungroup() %>%
  mutate(ObservationID = as.factor(ObservationID)) %>%
  mutate(ObservationID = fct_reorder(ObservationID, dNBR)) %>%
  ggplot() +
  geom_line(aes(x = doy, y = spline.pred)) +
  geom_point(aes(x = doy,y = DailyMeanNDMI, color = BeforeBurnDate),alpha = .2) +
  labs(y = sprintf("Daily mean %s\n (per point location)",index_name),
       title = "Lowest severity top left to highest bottom right",
       subtitle = sprintf("Spline parameter = %s",spar)) +
  geom_vline(aes(xintercept = yday(burn_date)), color = "red", linetype = "dashed") +
  facet_wrap(~ObservationID) +
  theme_cowplot())
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `ObservationID = fct_reorder(ObservationID, dNBR)`.
## Caused by warning:
## ! `fct_reorder()` removing 2168 missing values.
## i Use `.na_rm = TRUE` to silence this message.
## i Use `.na_rm = FALSE` to preserve NAs.
```

```
## Warning: Removed 2168 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 2168 rows containing missing values or values outside the scale range
## (`geom_vline()`).
```



```r
# ggsave2(sprintf("figures/%s_perpoint_splines_%s.png",index_name,spar),
#          bg = "white",width = 10, height = 8)
```

It's much better than with the default parameter of 0.5 but still suffers from edge effects and could be too smooth. Some curves look like they're cut short at the time of burning.

So I tried LOESS smoothing, which can deal with edge effects.

c. LOESS

First, I ran a first order LOESS, so it's fitting a linear regression in each local window:

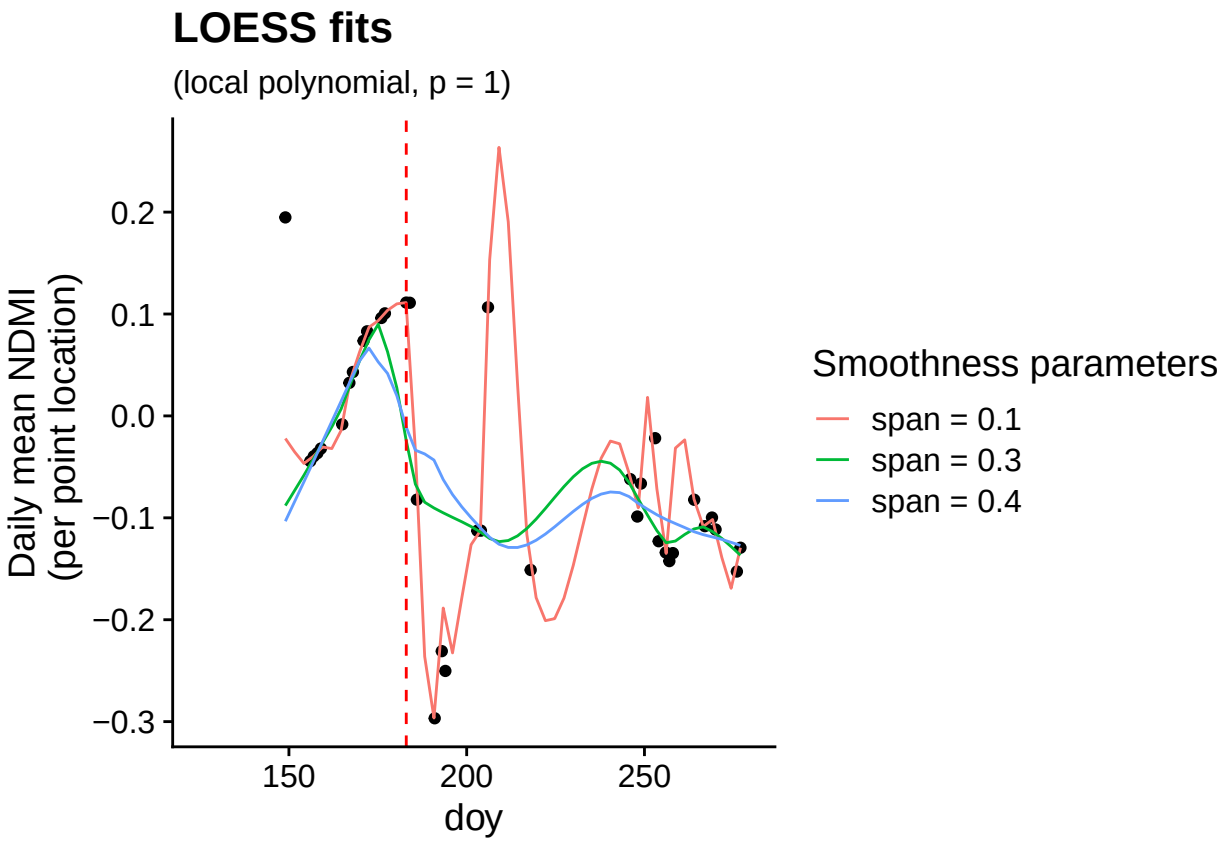
```r
df <- df_filtered %>% filter(ObservationID == 2391)

spans <- c(0.1,0.3,0.4)
lobj1 <- loess.smooth(df$doy,df$DailyMeanNDMI,span = spans[1])
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 157

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number -0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 9

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 157

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number -0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```

```r
lobj2 <- loess.smooth(df$doy,df$DailyMeanNDMI,span = spans[2])
lobj3 <- loess.smooth(df$doy,df$DailyMeanNDMI,span = spans[3])

ggplot() +
  geom_point(data = df,aes(x = doy, y = DailyMeanNDMI)) +
  geom_line(aes(x = lobj1$x,y = lobj1$y,
                color = sprintf("span = %s",spans[1]))) +
  geom_line(aes(x = lobj2$x,y = lobj2$y,
                color = sprintf("span = %s",spans[2]))) +
  geom_line(aes(x = lobj3$x,y = lobj3$y,
                color = sprintf("span = %s",spans[3]))) +
  geom_vline(aes(xintercept = yday(df$burn_date)), color = "red", linetype = "dashed") +
  labs(y = sprintf("Daily mean %s\n (per point location)",index_name),
       color = "Smoothness parameters",
       title = "LOESS fits",
       subtitle = "(local polynomial, p = 1)") +
  theme_cowplot()
```

```
## Warning: Removed 73 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# ggsave2(sprintf("figures/%s_LOESS.png",index_name),
#         bg = "white",width = 10, height = 8)
```

Would a second order polynomial be better?

```
df <- df_filtered %>% filter(ObservationID == 2391)

spans <- c(0.1,0.3,0.4)
lobj1 <- loess.smooth(df$doy,df$DailyMeanNDMI,span = spans[1], degree = 2)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## span too small.  fewer data values than degrees of freedom.
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 58.37
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## span too small.  fewer data values than degrees of freedom.
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36
```

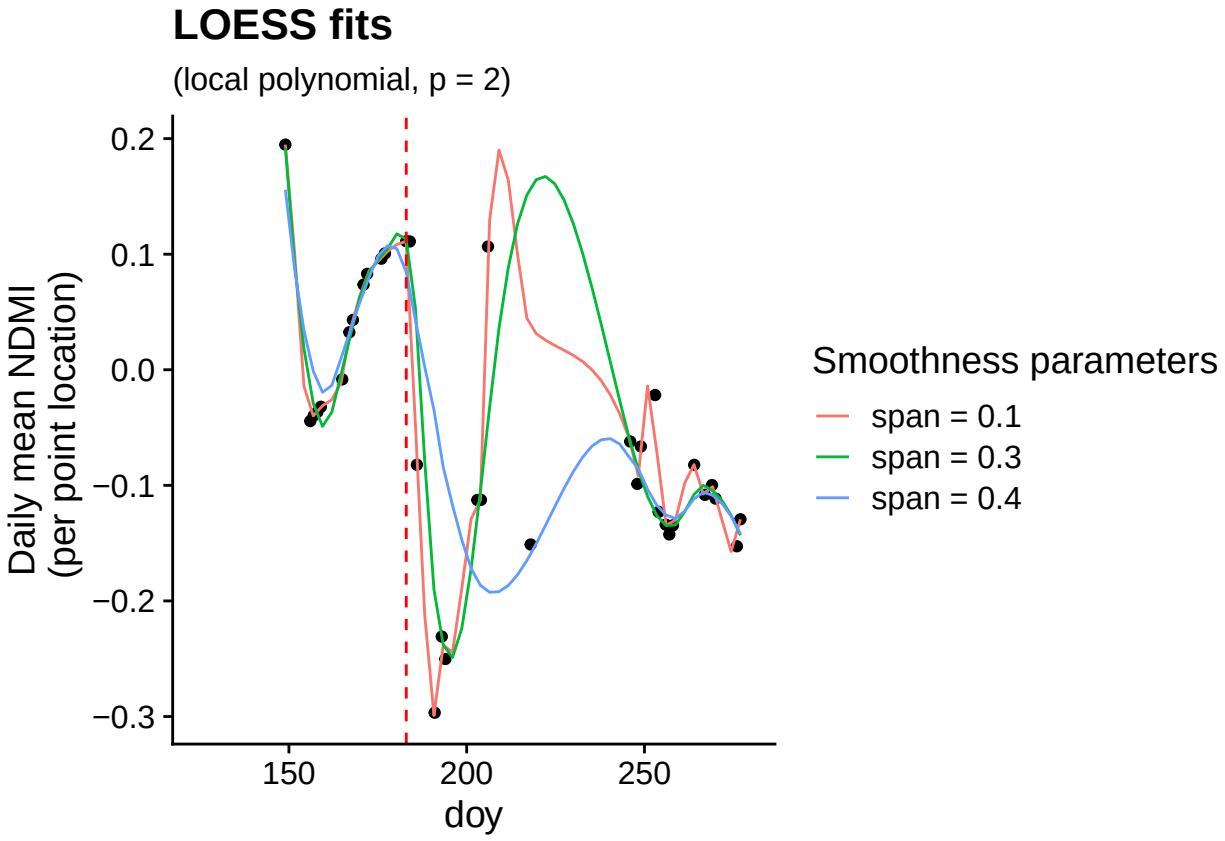
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
```

```
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 58.37

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## span too small.  fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 58.37

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## span too small.  fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 58.37

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## span too small.  fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 148.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 8.64

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 58.37
```

```r
lobj2 <- loess.smooth(df$doy,df$DailyMeanNDMI,span = spans[2], degree = 2)
lobj3 <- loess.smooth(df$doy,df$DailyMeanNDMI,span = spans[3], degree = 2)

ggplot() +
  geom_point(data = df,aes(x = doy, y = DailyMeanNDMI)) +
  geom_line(aes(x = lobj1$x,y = lobj1$y,
                color = sprintf("span = %s",spans[1]))) +
  geom_line(aes(x = lobj2$x,y = lobj2$y,
                color = sprintf("span = %s",spans[2]))) +
  geom_line(aes(x = lobj3$x,y = lobj3$y,
```

```
                color = sprintf("span = %s",spans[3]))) +
  geom_vline(aes(xintercept = yday(df$burn_date)), color = "red", linetype = "dashed") +
  labs(y = sprintf("Daily mean %s\n (per point location)",index_name),
       color = "Smoothness parameters",
       title = "LOESS fits",
       subtitle = "(local polynomial, p = 2)") +
  theme_cowplot()
```

```
## Warning: Removed 73 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



4. Retrieve time-integrated metric

I use the full-season splines (spar = 0.5) to predict NDMI values at different time intervals before the fire and then calculate their integrals.

```
df_filtered$doy <- lubridate::yday(df_filtered$date)

df <- df_filtered %>%
  filter(ObservationID == 2391,
         !is.na(DailyMeanNDMI))

test_smooth <- model_index_smoothedspline(df$doy,df$DailyMeanNDMI,df) %>%
  rename(spline.pred = 'spline.pred.doy.1')

model <- model_fit_smoothedspline(df$doy,df$DailyMeanNDMI,spar=0.5)
pred <- data.frame(predict(model, data.frame(doy = 130:300))) %>%
```

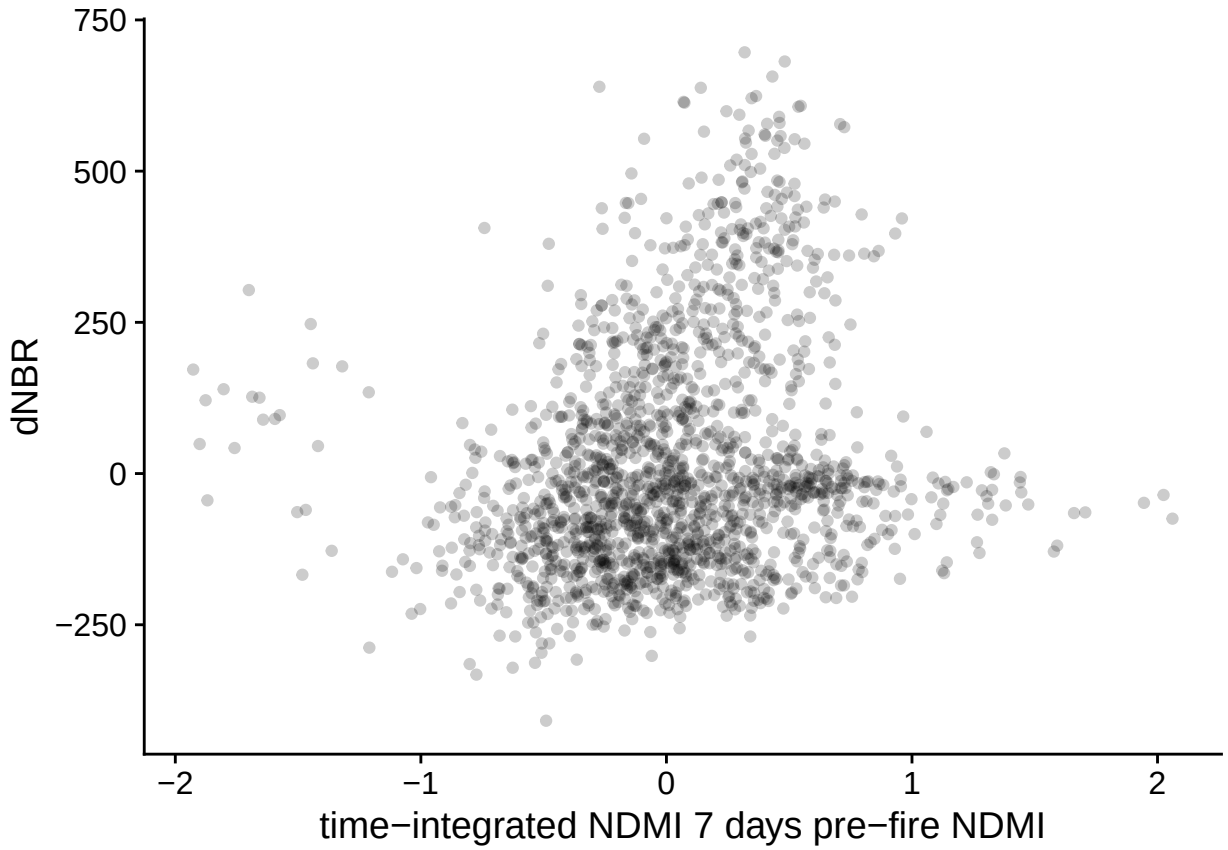
```
  rename(doy = doy, index = doy.1)

# Get DOY of burn
doy_burn <- yday(df$burn_date[1])

pred_before_burn <- pred %>%
  filter(doy < doy_burn) %>%
  mutate(TI_index = revcumsum(index),
         days_before_fire = doy_burn - doy)

TI_1week <- pred_before_burn %>%
  filter(days_before_fire ==7)
```

```
# Restructured function to return vector of predictions
model_index_smoothedspline <- function(x,y,full_data, spar = 0.5) {

  # Use function to fit model
  model <- model_fit_smoothedspline(x,y,spar)

  # Generate predictions for curve plotting (for time-period doy 130-300)
  pred <- data.frame(predict(model, data.frame(doy = 130:300))) %>%
  rename(doy = doy, index = doy.1)

  # Get DOY of burn
  doy_burn <- yday(full_data$burn_date[1])

  pred_before_burn <- pred %>%
    filter(doy < doy_burn) %>%
    mutate(TI_index = revcumsum(index),
           days_before_fire = doy_burn - doy)

  # Prepare output vector of 40-day predictions
  out_data <- pred_before_burn %>%
    filter(days_before_fire <= 40) %>%
    select(days_before_fire, TI_index) %>%
    pivot_wider(
      names_from = days_before_fire,
      values_from = TI_index,
      names_prefix = "d_prefire_"
      )

  return(out_data)
}

spar <- 0.5

# Apply spline to each time series point
ndmi_smooth <- df_filtered %>%
  filter(!is.na(DailyMeanNDMI)) %>%
  group_by(ObservationID) %>%
  group_modify(~model_index_smoothedspline(.x$doy,.x$DailyMeanNDMI,.x,spar = spar))


# Merge dataframes
```

```
data <- suppressMessages(df_filtered %>%
                           # filter(ObservationID %in% rand_id) %>%
                           select(ObservationID,dNBR,burn_date) %>%
                           group_by(ObservationID) %>%
                           unique() %>%
                           ungroup() %>%
                           right_join(ndmi_smooth, by = "ObservationID")
                         )
```

Let's look at the relationship (scatter) between dNBR and the cumulative NDMI from 1-7 days before the fire:

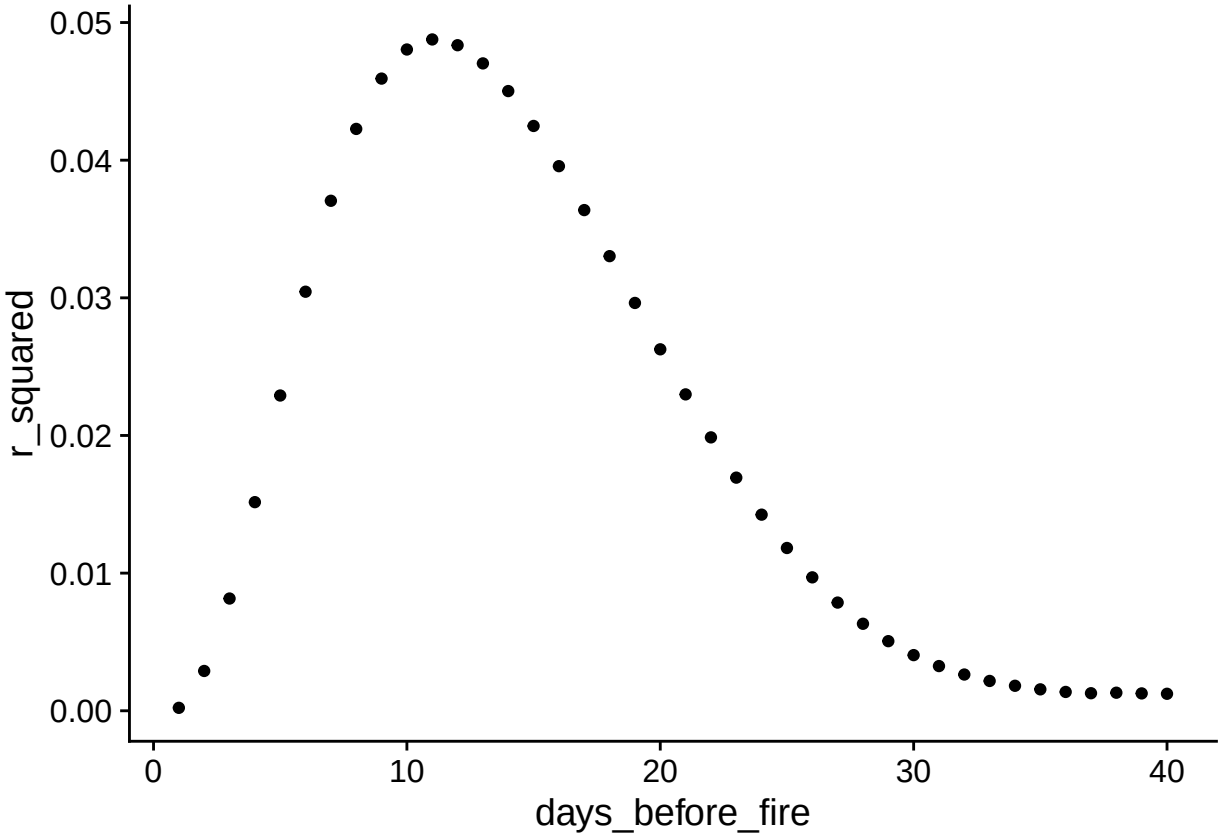
```
# Plot time-integrated NDMI 7 days before fire vs. dNBR
ggplot(data) +
  geom_point(aes(x = d_prefire_7 ,y = dNBR),alpha = .2) +
  labs(x = sprintf("time-integrated NDMI 7 days pre-fire %s",index_name),
       y = "dNBR") +
  theme_cowplot()
```



Now, let's see which time before the fire has the highest R2:

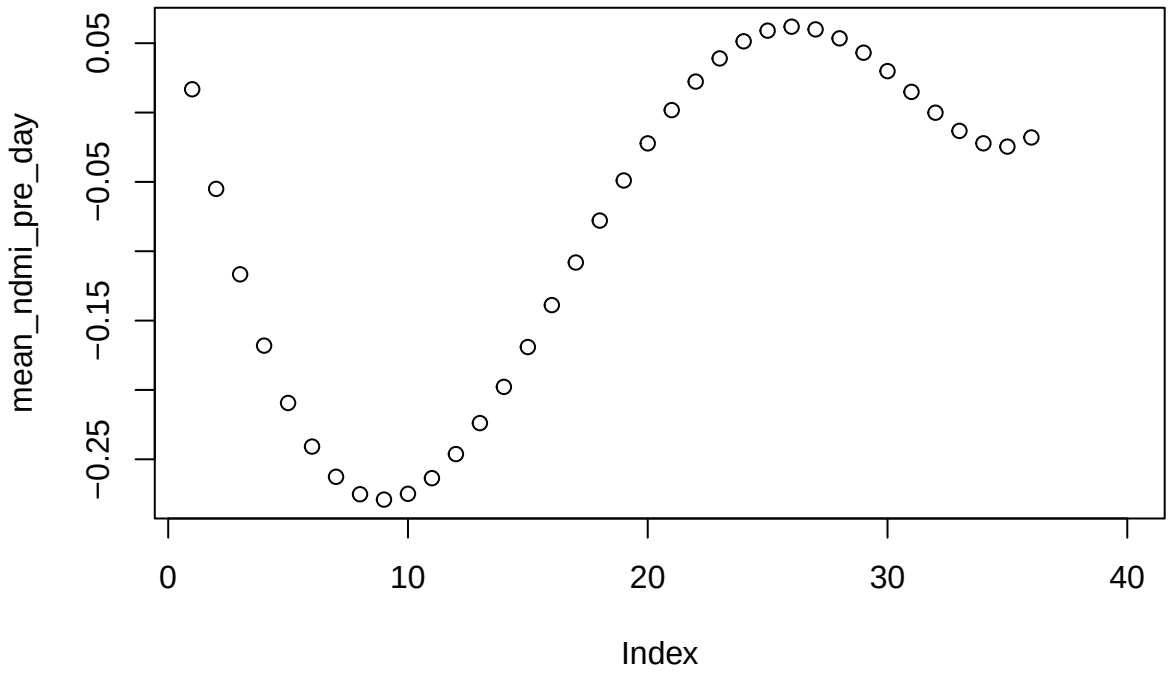
```
d_prefire_cols <- grep("^d_prefire_", names(data), value = TRUE)

# Function to run lm and extract statistics
run_lm <- function(y_var, x_var, data) {
  formula <- as.formula(paste(y_var, "~", x_var))
  model <- lm(formula, data = data)
  summary <- summary(model)
```

```
  # get days before fire as numeric
  days_before <- as.numeric(str_extract(x_var, "\\d+"))

  tibble(
    predictor = x_var,
    days_before_fire = days_before,
    r_squared = summary$r.squared,
    adj_r_squared = summary$adj.r.squared,
    coefficient = coef(model)[2],
    p_value = summary$coefficients[2, 4]
  )
}

# Run lm for each d_prefire column and combine results
results <- map_dfr(d_prefire_cols, ~run_lm("dNBR", .x, data))

results <- results %>% arrange(days_before_fire)

ggplot(results) +
  geom_point(aes(x = days_before_fire, y = r_squared)) +
  theme_cowplot()
```



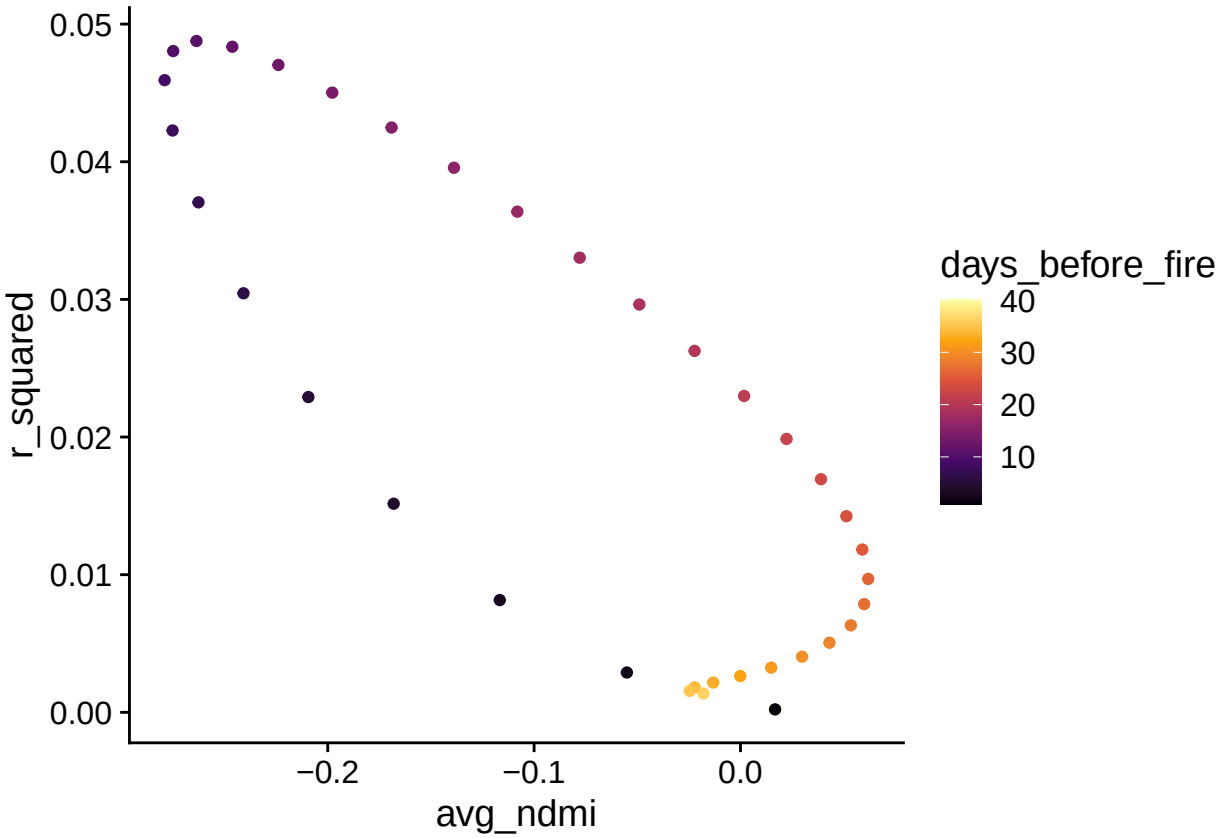
Does it look like the average of predicted NDMI before the fire? A bit inversed, yes.

```
mean_ndmi_pre_day <- colMeans(data[d_prefire_cols])

plot(mean_ndmi_pre_day)
```




```
results %>%
  mutate(avg_ndmi = mean_ndmi_pre_day) %>%
  ggplot() +
  geom_point(aes(x = avg_ndmi, y = r_squared, color = days_before_fire)) +
  scale_color_viridis_c(option = "inferno") +
  theme_cowplot()
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```


0.05
0.04
0.03
0.02
0.01
0.00
r_squared
−0.2
−0.1
0.0
avg_ndmi
days_before_fire
40
30
20
10