04_trainingdata

Casey Menick

2022-11-19

Contents

Chapter 1

About

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports; for example, a math equation $a^2 + b^2 = c^2$.

1.1 Usage

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: `# A good chapter`, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: `## A short section` or `### An even shorter section`.

The `index.Rmd` file is required, and is also your first book chapter. It will be the homepage when you render the book.

1.2 Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and
2. Click on **Build Book**, then select your output format, or select "All formats" if you'd like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a `bookdown::pdf_book`, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

1.3 Preview book

As you work, you may start a local server to live preview this HTML book. This preview will update as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in "Preview book", or from the R console:

```
bookdown::serve_book()
```

Chapter 2

Set Up

```
library(tidyverse)
library(terra)
library(sf)
library(mapview)
library(raster)
library(rgeos)
library(lubridate)
library(ggplot2)
library(exactextractr)
library(patchwoRk)
library(gridExtra)
```

Chapter 3

Import Data

3.1 USDA National Forest Type Group Dataset (2004)

Conifer Forest Type Groups: Douglas-Fir, Fir-Spruce-Mountain Hemlock, Lodgepole Pine

```
# forest type groups and key
conus_forestgroup <- raster('data/forest_type/conus_forestgroup.tif')
forest_codes <- read_csv('data/forest_type/forestgroupcodes.csv')

# set crs
crs = crs(conus_forestgroup)
```

3.2 EPA level-3 Ecoregions

Canadian Rockies, Idaho Batholith, Middle Rockies, Columbian Mountains - Northern Rockies

```
# level 3 ecoregions
l3eco <- st_read('data/ecoregion/us_eco_l3.shp') %>%
  st_transform(., crs=crs)

# select northern rocky mountains from level3 ecoregions
eco_select <- l3eco %>%
  filter(NA_L3NAME %in% c('Canadian Rockies','Columbia Mountains/Northern Rockies','Middle Rockie
```

3.3 Mapping

3.3.1 Ecoregions

```
mapview(eco_select)
```

3.3.2 Forest Type Groups

```
forestgroup_crop <- crop(conus_forestgroup,l3eco)
mapview(forestgroup_crop)
```

Chapter 4

Define Fire Parameters

4.1 MTBS

Criteria:

-1988-1991

-500+ acres of high-severity

-Within selected ecoregions

->25% of selected forest types

```
# mtbs fire perimeters
mtbs_full <- st_read('data/mtbs/mtbs_perims_DD.shp') %>%
  st_transform(., crs=crs)

mtbs_select <- mtbs_full %>%
  mutate(state = str_sub(Event_ID,0,2),
         year = year(as.Date(Ig_Date))) %>%
  filter(state %in% c("WA","ID","MT","WY","SD"),
         between(Ig_Date, as.Date('1988-01-1'), as.Date('1991-12-31')))
```

4.2 Group Adjacent Fires

```
# function to group adjoining fire polygons to ensure contiguous high-severity patches
group_fires <- function(mtbs_year) {

  # join the polygons with themselves, and remove those that do not join with any besides themsel
```

```
combined<- st_join(mtbs_year, mtbs_year, join=st_is_within_distance, dist = 180, left
  drop_na(Event_ID.y)%>%
  dplyr::select(Event_ID.x,Event_ID.y)

if(nrow(combined)>=1){ # if there are overlaps for this years fires...

  # partition data into that that has overlap, and that that does not
  overlap <- mtbs_year %>%
    filter(Event_ID %in% combined$Event_ID.x)
  no_overlap <- mtbs_year %>%
    filter(!(Event_ID %in% combined$Event_ID.x))

  print(paste0("there are ",nrow(overlap)," overlapping polygons"))

  # join all overlapping features, and buffer to ensure proper grouping
  overlap_union <- st_union(overlap) %>%
    st_buffer(190)

  # break apart the joined polygons into their individual groups
  groups <- st_as_sf(st_cast(overlap_union ,to='POLYGON',group_or_split=TRUE)) %>%
    mutate(year = mean(mtbs_year$year),
           Fire_ID = str_c("Fire_",c(1:nrow(.)),"_",year)) %>%
    rename(geometry = x)

  print(paste0("polygons formed into ",nrow(groups)," groups"))

  # join back with original dataset to return to unbuffered geometry
  grouped_overlap <- st_join(overlap,groups,left=TRUE)

  # arrange by the new grouping
  joined_overlap_groups <- grouped_overlap %>%
    group_by(Fire_ID) %>%
    tally()%>%
    st_buffer(1) %>%
    dplyr::select(Fire_ID) %>%
    mutate(year = mean(mtbs_year$year))

  # add new ID to the freestanding polygons
  no_overlap_groups <- no_overlap %>%
    mutate(Fire_ID = str_c("Fire_",nrow(groups)+c(1:nrow(no_overlap)),"_",year)) %>%
    dplyr::select(Fire_ID,year)

  # join the new grouped overlap and the polygons without overlap
  fires_export <- rbind(joined_overlap_groups,no_overlap_groups)
  return(fires_export)
```

```
    } else { # if there are no overlaps for this year...

      print("no overlapping polygons")

      fires_export <- mtbs_year %>%
        mutate(Fire_ID = str_c("Fire_",c(1:nrow(.)),"_",year)) %>%
        dplyr::select(Fire_ID,year)

      return(fires_export)
  }
}
```

```
# group adjacent polygons within each fire year
fires_88 <- group_fires(mtbs_select %>%  filter(year == 1988))
```

```
## [1] "there are 22 overlapping polygons"
## [1] "polygons formed into 7 groups"
```

```
fires_89 <- group_fires(mtbs_select %>%  filter(year == 1989))
```

```
## [1] "there are 2 overlapping polygons"
## [1] "polygons formed into 1 groups"
```

```
fires_90 <- group_fires(mtbs_select %>%  filter(year == 1990))
```

```
## [1] "there are 2 overlapping polygons"
## [1] "polygons formed into 1 groups"
```

```
fires_91 <- group_fires(mtbs_select %>%  filter(year == 1991))
```

```
## [1] "no overlapping polygons"
```

```
# join each fire year, filter by area
mtbs_grouped <- rbind(fires_88,fires_89,fires_90,fires_91)%>%
  mutate(area_ha = as.numeric(st_area(geometry))/10000,
         area_acres = area_ha*2.471)
```

4.3 Select Fires by Ecoregion and Forest Type

```
# assign ecoregion and proportions of forest type to each fire polygon
fires_join <- st_join(mtbs_grouped,eco_select,join=st_intersects,left=FALSE,largest=TRU
  left_join(., exact_extract(conus_forestgroup,mtbs_grouped, append_cols = TRUE, max_c
                             fun = function(value, coverage_fraction) {
                               data.frame(value = value,
                                          frac = coverage_fraction / sum(coverage_fract
                                 group_by(value) %>%
                                 summarize(freq = sum(frac), .groups = 'drop') %>%
                                 pivot_wider(names_from = 'value',
                                             names_prefix = 'freq_',
                                             values_from = 'freq')}) %>%
              mutate(across(starts_with('freq'), replace_na, 0)))

# remove unnecessary columns, cleanup names
# filter to ensure fire polygons are at least 25% type of interest
fires <- fires_join %>%
  dplyr::select("Fire_ID","year","area_ha","area_acres","US_L3NAME","freq_0","freq_200
  rename("ecoregion" = "US_L3NAME",
         "freq_df"="freq_200",
         "freq_pp"="freq_220",
         "freq_fs"="freq_260",
         "freq_lpp"="freq_280") %>%
  mutate(freq_allother = 1-(freq_0 + freq_df+freq_pp+freq_fs+freq_lpp),
         freq_forested = 1- freq_0,
         freq_ideal = freq_df+freq_fs+freq_lpp)%>%
  mutate(across(starts_with('freq'), round,2))%>%
  filter(freq_ideal > 0.25)
```

4.4 Select Fires by Burn Severity

```
# import all mtbs rasters via a list
rastlist <- list.files(path = "data/mtbs", pattern='.tif', all.files=TRUE, full.names=
allrasters <- lapply(rastlist, raster)
names(allrasters) <- str_c("y", str_sub(rastlist,22,25))

# create empty dataframe
severity_list <- list()

# loop through mtbs mosasics for 1988-1991
# extract mtbs burn severity raster for all selected fires
# calculate burn severity percentages for each fire
for (i in names(allrasters)){
```

```
  mtbs_year <- allrasters[[i]]
  fire_year <- filter(fires, year==str_sub(i,2,5))
  raster_extract <- exact_extract(mtbs_year,fire_year, max_cells_in_memory = 3e+09,coverage_area=
  names(raster_extract) <- fire_year$Fire_ID

  output_select <- bind_rows(raster_extract, .id = "Fire_ID")%>%
    group_by(Fire_ID , value) %>%
    summarize(total_area = sum(coverage_area)) %>%
    group_by(Fire_ID) %>%
    mutate(proportion = total_area/sum(total_area))%>%
    dplyr::select("Fire_ID","value","proportion") %>%
    spread(.,key="value",value = "proportion")

  severity_list[[i]] <- output_select
}

# combine extracted raster datasets
severity_df <- do.call(rbind, severity_list)

# join burn severity % to fires polygons
# fix naming
# filter dataset for 500 acres high severity
fires_severity <- left_join(fires,severity_df,by="Fire_ID")%>%
  rename(noburn= "1",lowsev = "2", medsev = "3", highsev = "4",regrowth = "5", error = "6") %>%
  dplyr::select(- "NaN",-"regrowth",-"error") %>%
  mutate(highsev_acres = area_acres*highsev)%>%
  filter(highsev_acres > 500)
```

4.5 Clean Up Dataset

```
# get the most common forest type within each polygon
fires_select <- fires_severity %>%
  left_join(.,exact_extract(conus_forestgroup,fires_severity, 'mode', append_cols = TRUE, max_cel

fires_select$mode <- as.factor(fires_select$mode)

fires_select <- fires_select %>%
    mutate(fire_foresttype = case_when(mode==200 ~ "Douglas-Fir",
                                       mode==220 ~ "Ponderosa",
                                       mode==260 ~ "Fir-Spruce",
                                       mode==280 ~ "Lodegepole Pine",
                                       TRUE ~ "Other"),
```

```
            Fire_ID = str_c("Fire_",c(1:nrow(.)),"_",year))
```

```
# join the grouped fires back to original mtbs boundaries
fires_mtbs <- st_join(mtbs_select,fires_select,left=FALSE,largest=TRUE) %>%
  filter(year.x==year.y)%>%
  dplyr::select("Event_ID","Incid_Name","Fire_ID","Ig_Date","year.y","state","BurnBndA
  rename(year= year.y)
```

Chapter 5

Final Fire Dataset

5.1 Selected fires by year

```
# plot
mapview(fires_select, zcol = "year")
```

5.2 Selected fires by majority forest type

```
# plot
mapview(fires_select, zcol = "fire_foresttype")
```

Chapter 6

Export Data

6.1 Final Cleanup for Export

```
# reformat and project
fires_export <- fires_select %>%
  mutate(year = as.integer(year)) %>%
  st_transform(., crs="EPSG:4326")

mtbs_export <- fires_mtbs %>%
  mutate(year = as.integer(year)) %>%
  st_transform(., crs="EPSG:4326")
```

6.2 Export

```
# st_write(fires_export, "data/fire_boundaries/", "fires_export.shp", driver = 'ESRI Shapefile')
# st_write(mtbs_export, "data/fire_boundaries/", "mbts_export.shp", driver = 'ESRI Shapefile')
```

Chapter 7

Set Up

7.1 Libraries

```
library(tidyverse)
library(terra)
library(patchwoRk)
library(sf)
library(mapview)
library(exactextractr)
library(lubridate)
```

7.2 Import Data

```
# import calculated RdNBR rasters for each fire boundary polygon
rast_list <- list.files(path = "data/rdnbr_rasters", pattern='.tif', all.files=TRUE, full.names=T
rast_all <- lapply(rast_list, rast)
rast_collection <- sprc(rast_all)

crs <- crs(rast_collection[1])
```

```
# import fire boundaries
mtbs_export <- st_read('data/fire_boundaries/mtbs_export.shp') %>%
  st_transform(., crs=crs)

fires_export <- st_read("data/fire_boundaries/fires_export.shp")%>%
```

```
  st_transform(., crs=crs)

# import forest type group raster
conus_forestgroup <- raster('data/forest_type/conus_forestgroup.tif')
forest_codes <- read_csv('data/forest_type/forestgroupcodes.csv')
```

Chapter 8

Create High-Severity Patches

8.1 PatchMorph

```
# loop through RdNBR rasters, assign >640 to high severity category
# utilize patchmorph to act as 3x3 cell majority filter
patch_df <- list()
for (i in 1:length(rast_all)){
  # print(i)
  rast_fire <- raster(rast_collection[i])
  rast_fire[rast_fire < 640] <- 0
  rast_fire[rast_fire >= 640] <- 1

  patch <- patchMorph(rast_fire, spurThresh = 3, gapThresh = 3)
  patch_poly <- as.polygons(rast(patch)) %>%
    st_as_sf()
  df_union_cast <- patch_poly %>%
    st_cast(., "POLYGON") %>%
    filter(layer == 1)
  patch_df[[i]] <- df_union_cast}

patch_poly_all <- do.call(rbind,patch_df)
```

8.2 Refine Patches

```
# filter small patches
patches_full <- patch_poly_all %>%
  mutate(patch_area_ha = as.numeric(st_area(.))/10000) %>%
  filter(patch_area_ha > 2.25)
```

```
# join patches back to grouped fires
patches_joined <- st_join(patches_full,mtbs_export,join = st_intersects,left= FALSE,la
  dplyr::select(-layer,-BurnBndAc) %>%
  left_join(.,exact_extract(conus_forestgroup,., 'mode', append_cols = TRUE, max_cells
  mutate(patch_foresttype = case_when(mode==200 ~ "Douglas-Fir",
                                      mode==220 ~ "Ponderosa",
                                      mode==260 ~ "Fir-Spruce",
                                      mode==280 ~ "Lodegepole Pine",
                                      mode==0 ~ "Unforested",
                                      TRUE ~ "Other"))
```

8.3 Mapping

```
mapview(patches_joined,col.regions = "red") + mapview(fires_export, alpha.regions = 0,
```

Chapter 9

Export Data

```
patches <- patches_joined %>%
  st_transform(crs = crs)

# st_write(patches, "data/patches/", "highsev_patches.shp",driver = 'ESRI Shapefile')
```

Chapter 10

Set Up

10.1 Libraries

```
library(elevatr)
library(tidyverse)
library(sf)
library(terra)
library(mapview)
```

10.2 Import Data

```
# data import
patches <- st_read("data/patches/highsev_patches.shp") %>%
  st_transform(crs="EPSG:4326")
crs <- crs(patches)

patch_interiors<- st_read("data/patches/highsev_patches_interior.shp") %>%
  st_transform(crs=crs)
patch_exteriors<- st_read("data/patches/highsev_patches_exterior.shp") %>%
  st_transform(crs=crs)

mtbs_export <- st_read('data/fire_boundaries/mtbs_export.shp') %>%
  st_transform(crs=crs)

fires_export <- st_read("data/fire_boundaries/fires_export.shp")%>%
  st_transform(crs=crs)
```

Chapter 11

Create Sampling Quadrants

11.1 Create Quadrants of North/South Aspects and Patch Interior/Exterior

```
# create list of fire IDs
fire_list <- unique(patches$Evnt_ID)

quadrants_df = list()

for(i in fire_list){

  # filter patch interiors/exteriors to the selected fire
  patch_fire <- patches %>%
    filter(Evnt_ID == i)

  mapview(patch_fire)

  patches_interior <- patch_interiors %>%
    filter(Evnt_ID == i)%>%
    st_make_valid() %>%
    st_union()

  patches_exterior <- patch_exteriors %>%
    filter(Evnt_ID_1 == i)%>%
    st_make_valid()%>%
    st_union()

  # set event and fire id to the selected fire
```

```
Evnt_ID <- i
Fire_ID <-names(which.max(table(patch_fire$Fire_ID)))

print(paste0("starting event ",Evnt_ID," in fire group ", Fire_ID))

# get and calculate cosine corrected aspect
dem <- get_elev_raster(patch_fire,z=11)
aspect <- terrain(dem, opt = "aspect",unit = "radians")
ccaspect <- cos(aspect)

# positive aspects are north-facing, negative are south-facing
ccaspect[ccaspect>0] <- 1
ccaspect[ccaspect<0] <- -1
ccaspect_poly <- as.polygons(rast(ccaspect)) %>%
  st_as_sf()

pos_aspect <- ccaspect_poly %>%
  filter(layer==1)%>%
  st_make_valid()
neg_aspect <- ccaspect_poly %>%
  filter(layer==-1) %>%
  st_make_valid()

# get quadrants as the intersection of interior/exterior and pos/neg aspect
pos_ext <- st_intersection(patches_exterior,pos_aspect)%>%
  st_make_valid() %>%
  st_union() %>%
  st_as_sf()%>%
  mutate(quadrant = "pos_ext",
         Evnt_ID = i,
         quad_id_event = paste0(Evnt_ID,"-",quadrant),
         Fire_ID = Fire_ID,
         quad_id_fire = paste0(Fire_ID,"-",quadrant))

pos_int <- st_intersection(patches_interior,pos_aspect)%>%
  st_make_valid() %>%
  st_union()%>%
  st_as_sf()%>%
  mutate(quadrant = "pos_int",
         Evnt_ID = i,
         quad_id_event = paste0(Evnt_ID,"-",quadrant),
         Fire_ID = Fire_ID,
         quad_id_fire = paste0(Fire_ID,"-",quadrant))

neg_ext <- st_intersection(patches_exterior,neg_aspect)%>%
```

```
    st_make_valid() %>%
    st_union() %>%
    st_as_sf()%>%
    mutate(quadrant = "neg_ext",
           Evnt_ID = i,
           quad_id_event = paste0(Evnt_ID,"-",quadrant),
           Fire_ID = Fire_ID,
           quad_id_fire = paste0(Fire_ID,"-",quadrant))

  neg_int <- st_intersection(patches_interior, neg_aspect)%>%
    st_make_valid() %>%
    st_union() %>%
    st_as_sf()%>%
    mutate(quadrant = "neg_int",
           Evnt_ID = i,
           quad_id_event = paste0(Evnt_ID,"-",quadrant),
           Fire_ID = Fire_ID,
           quad_id_fire = paste0(Fire_ID,"-",quadrant))

  # combine, export quadrants
  all_quadrants <- rbind(neg_int,pos_int,neg_ext,pos_ext) %>%
    st_transform(crs=crs)

  quadrants_df[[i]] <- all_quadrants

  print(paste0("completed"))
}

# bind list together
quadrants_fullset <- do.call(rbind,quadrants_df) %>%
  st_as_sf()
```

11.2 Clean Quadrants

```
# removes erroneous polygons created from irregular fire boundary shapes
# removes small border mismatched fire
quadrants_clean <- quadrants_fullset %>%
  mutate(area=as.numeric(st_area(x))) %>%
  filter(area > 1) %>%
  group_by(Evnt_ID) %>%
  mutate(n=n()) %>%
  filter(n == 4)
```

```
# clean up for export
quadrants_export <- quadrants_clean %>%
  st_make_valid() %>%
  st_as_sf() %>%
  dplyr::select(-"area")%>%
  st_transform(crs=crs)
```

Chapter 12

Export Data

```
# st_write(quadrants_export,"data/patches/","quadrants_export.shp",driver = "ESRI Shapefile")
```

Chapter 13

Set Up

13.1 Libraries

```
library(tidyverse)
library(sf)
library(terra)
```

13.2 Import Data

```
patches <- st_read("data/patches/highsev_patches.shp") %>%
  st_transform(crs="EPSG: 4326")

crs <- crs(patches)

quadrants <- st_read("data/patches/quadrants_export.shp") %>%
  st_transform(crs=crs)
```

Chapter 14

Sampling Points

14.1 Import and Combine Training Points

```
points_list <- list.files(path = "data/points/individual_fire_points/", pattern='.shp', all.files
points_all <- lapply(points_list, st_read)

points <- do.call(rbind,points_all) %>%
  st_transform(crs=crs)
```

14.2 Assign Points and Clean Data

```
# join points dataset back to fires to fill out dataset
points_joined <- st_join(points,patches,left=TRUE,largest=TRUE) %>%
  st_join(.,quadrants,left=TRUE,largest=TRUE)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries

## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
points_cleaned <- points_joined %>%
  dplyr::select("class","ptch_r_","Evnt_ID.x","Incd_Nm","Fire_ID.x","year","ecoregn","ptch_fr","c
  rename(patch_area_ha = ptch_r_,
         Event_ID = Evnt_ID.x,
```

```
        Incid_Name = Incd_Nm,
        Fire_ID = Fire_ID.x,
        patch_frtype = ptch_fr,
        quad = quadrnt,
        quad_event_id= qd_d_vn,
        quad_fire_id=qd_d_fr) %>%
  st_transform(crs=crs)
```

Chapter 15

Export

```
# st_write(points_cleaned, "data/points/", "points_export.shp",driver = 'ESRI Shapefile')
```