Create data products

K Todd-Brown (ktoddbrown@gmail.com)

23 August 2018

```
library(tidyverse)
library(lubridate)
library(wordcloud) #make pretty word clouds
library(ncdf4) #read in netcdf files
library(raster) #read in geotiff files
library(gdalUtils) #helper for dealing with raster files
library(dplyr)
library(SoilDataR) #library(devtools); install_github("ktoddbrown/soilDataR") #access ISCN soil data

##Modify files below for your system
ISCN_filename <- '~/Documents/Datasets/ISCN_3/ISCN3.RData'
met_dir <- '~/Documents/Datasets/CPC'
workingTemp_dir <- '~/Documents/GitHubRepo/PrecipOnSOC/temp'
```

Soil data

Download and clean the International Soil Carbon Database (v3) data product.

```
#######ISCN3 - soil carbon, bulk density, clay percentage

if(file.exists('~/Documents/Datasets/ISCN_3/ISCN3.RData')){
  load(ISCN_filename)
} else {
  ISCN3 <- processData_ISCN3(dataDir=dirname(ISCN_filename), verbose=TRUE)
  save(ISCN3, file='~/Documents/Datasets/ISCN_3/ISCN3.RData')
}

#### Add new indecies for possible duplications  ####
ISCN3$field <- ISCN3$field %>%
  mutate(fieldID_byString = group_indices(., lat, lon,
                                          layer_bottom, layer_top, observation_date)) %>%
  mutate_at(c('layer_bottom', 'layer_top', 'observation_date'),
            funs(num=round(as.numeric(as.character(.)), 0))) %>%
  mutate_at(c('lat', 'lon'), funs(num=round(as.numeric(as.character(.)), 1))) %>%
  mutate(fieldID_byApprox = group_indices(., lat_num, lon_num,
                                          layer_bottom_num, layer_top_num,
                                       observation_date_num)) %>%
  dplyr::select(-ends_with('_num'))

#### Identify non-located entries ####
noLocation <- ISCN3$field %>%
  filter(is.na(lat), is.na(lon),
         is.na(layer_bottom), is.na(layer_top),
         is.na(observation_date)) %>%
  dplyr::select(ends_with('_name'), starts_with('fieldID')) %>%
  unique()
```

```
#### Identify conflicting entries ####
problematicFieldEntries <- ISCN3$field %>%
  filter(!is.na(lat), !is.na(lon), !is.na(layer_bottom), !is.na(layer_top),
         !is.na(observation_date)) %>% #only look at well defined sites
  dplyr::select(-lat, -lon, -layer_bottom, -layer_top, -observation_date,
         -fieldID, -fieldID_byApprox) %>% #look at non-identifying columns
  group_by(fieldID_byString) %>% #group by entry matching
  gather(key='header', value='value', -fieldID_byString, na.rm=TRUE) %>%
  group_by(fieldID_byString, header) %>%
  unique() %>%
  tally()  %>% #count up number of unique values for that header
  filter(n > 1) %>% #only care about headers with more then one value
  dplyr::select(fieldID_byString) %>% unique %>%
  left_join(ISCN3$field %>% dplyr::select(fieldID, fieldID_byString))
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
## Joining, by = "fieldID_byString"
```

```
#dim(noLocation)
#ISCN3$field$fieldID_byString %>% unique %>% length
#problematicFieldEntries$fieldID_byString %>% unique %>% length
#problematicFieldEntries %>% arrange(-n) %>% head

#### recast the lat-lon as numerics ####
ISCN3$field <- ISCN3$field  %>%
  mutate(lat=as.numeric(lat), lon=as.numeric(lon))

#### Add the hard coded units ####
hardUnits.df <- ISCN3$ISCNKey %>% filter(type == 'value', !is.na(hardUnit)) %>% dplyr::select(var, hardU
ISCN3$measure <-  ISCN3$measure %>%
  left_join(dplyr::select(filter(ISCN3$ISCNKey, type=='value'), var, hardUnit), by='var') %>%
  mutate(unit=if_else(grepl('\\w', unit) , as.character(unit), hardUnit))

#### Check country based on lat/lon ####
countryCheck <- ISCN3$field %>%
  filter(is.finite(lat+lon)) %>%
  dplyr::select(lat, lon, country) %>%
  mutate(country_renamed=recode(country, "United States"="USA",
                        "Federated States of Micronesia" = "Micronesia",
                        "Korea, Republic of"="South Korea",
                        "Congo (Democratic Republic of the)" =
                          "Democratic Republic of the Congo",
                        "Unknown"=as.character(NA))) %>%
  unique() %>%
  mutate(mapsID=as.factor(maps::map.where(y=lat, x=lon))) %>%
  mutate(mapsID=recode(mapsID,
                       'Puerto Rico'='USA:Puerto Rico',
                       'Virgin Islands, US:Saint Croix'=
                         'USA:Virgin Islands, US:Saint Croix',
                       'Virgin Islands, US:Saint John'=
                         'USA:Virgin Islands, US:Saint John',
                       'Virgin Islands, US:Saint Thomas'=
```

```
                          'USA:Virgin Islands, US:Saint Thomas',
                        'Guam'='USA:Guam')) %>%
  group_by(country_renamed, country, mapsID) %>%
  mutate(countryMatch = as.logical(grepl(as.character(country_renamed[1]),
                                         as.character(mapsID))),
         onlyOne = xor(is.na(country_renamed), is.na(mapsID)))

#countryCheck %>% filter(!countryMatch) %>% summary
#temp <- countryCheck %>% filter(!countryMatch & !onlyOne)
```

Filter for sites with bulk density and organic carbon percentage and remove sites that are gap filled or otherwise interpolated.

```
soilField <- ISCN3$field %>%
  filter(!is.na(lat), !is.na(lon), !is.na(layer_bottom), !is.na(layer_top),
         !is.na(observation_date)) %>% #only look at well defined sites
  filter(!(fieldID %in% problematicFieldEntries$fieldID) ) %>% #throw out non-unique field characterist
  inner_join(countryCheck %>%
               filter(countryMatch | onlyOne) %>%
               dplyr::select(-countryMatch, -onlyOne)) #force lat-lon to have well-defined countries
```

```
## Joining, by = c("lat", "lon", "country")
```

```
soilMeasures <- ISCN3$measure %>%
  filter(grepl('^bd', var) | grepl('^oc', var) | grepl('clay', var)) %>%
  filter(! (grepl('from horizon type', method) |
            grepl('from nearby sample', method) |
            grepl('predicted', method) |
            grepl('bd.*loi', method) |
              grepl('model', method) |
              grepl('similar horizons', method) |
              grepl('interpolated', method)))

soil.df <- ISCN3$sample %>%
  filter(fieldID %in% soilField$fieldID) %>%
  filter(measureID %in% soilMeasures$measureID)

uniquePairs <- soil.df %>%
  dplyr::select(-value) %>%
  group_by(fieldID, measureID) %>%
  tally %>%
  filter(n == 1)

soil.df <- soil.df %>%
  right_join(uniquePairs, by=c('fieldID', 'measureID'))

goodFieldID <- soil.df %>%
  group_by(fieldID) %>%
  summarize(hasClay= 100 %in% measureID,
            hasOC = 113 %in% measureID,
            n = length(measureID)) %>%
  filter(hasClay, hasOC, n > 2)

temp <- soil.df %>%
  filter(fieldID %in% goodFieldID$fieldID) %>%
```

```
  left_join(soilField) %>%
  left_join(soilMeasures %>% dplyr::select(measureID, var)) %>%
  unite(var_id, var, measureID) %>%
  spread(var_id, value)
```

```
## Joining, by = "fieldID"
```

```
## Joining, by = "measureID"
```

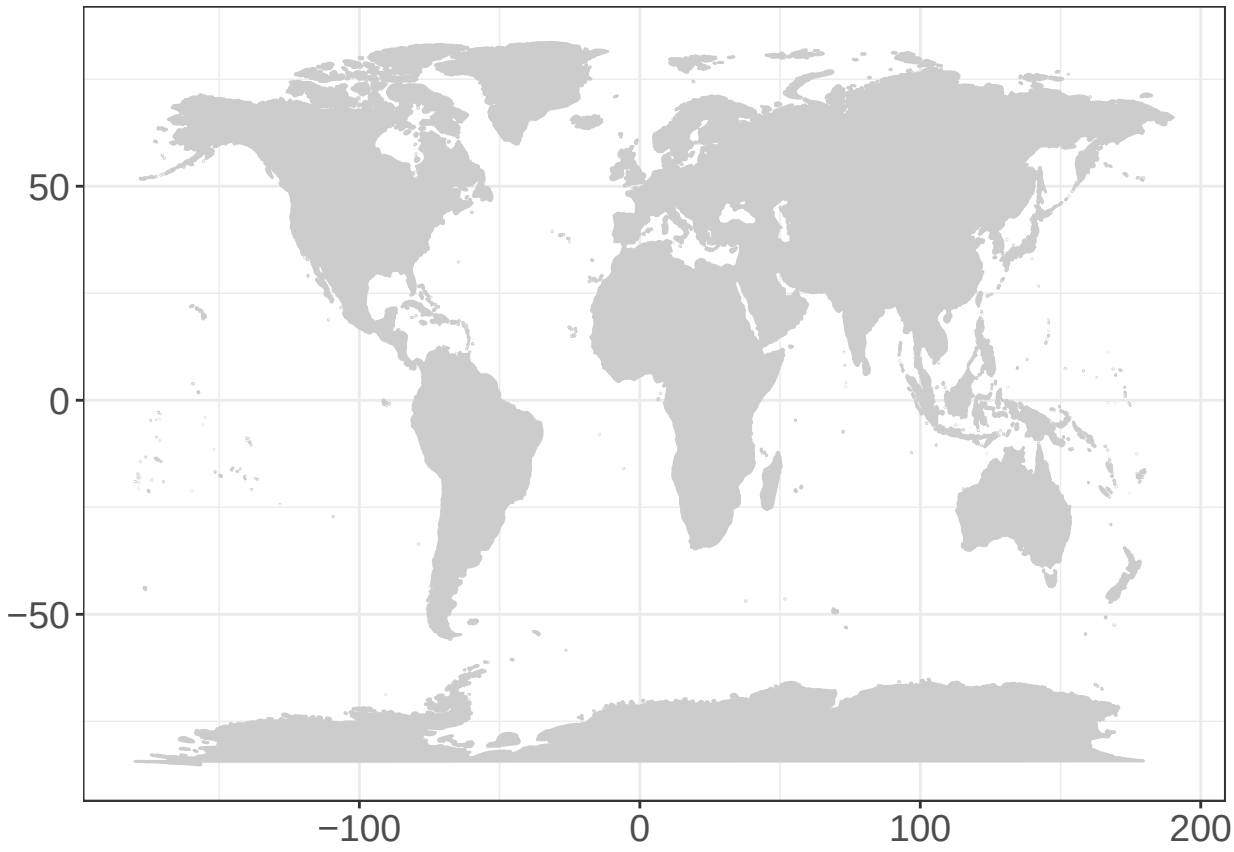
```
# Bulk density
#
#     bd_samp is the bulk density of the <2mm (fine earth) fraction, in which the mass is expressed on
#     bd_tot is the whole soil bulk density, includes fine earth fraction and rocks.
#     bd_whole is the bulk density of the whole soil (coarse frags, if present, included), expressed on
#     bd_other is, for data contributed by NRCS SSL, the bulk density of the fine earth fraction, but e

temp2 <-   temp %>%  dplyr::select(-n) %>%
  gather(key='bulk_density_type', value='bulk_density',
         starts_with('bd_'), na.rm=TRUE)

temp3 <- temp2 %>% filter(grepl('bd_sample', bulk_density_type))
temp4 <- temp2 %>% filter(grepl('bd_whole', bulk_density_type), !(fieldID %in% temp3$fieldID) )
temp5 <- temp2 %>% filter(grepl('bd_other', bulk_density_type), !fieldID %in% c(temp3$fieldID, temp4$fi
temp6 <- temp2 %>% filter(grepl('bd_tot', bulk_density_type), !fieldID %in% c(temp3$fieldID, temp4$fiel

soil.df <- temp3 %>% #bind_rows(temp3, temp4, temp5, temp6) %>%
  mutate_at(vars(layer_top, layer_bottom, observation_date), as.character) %>%
   mutate_at(vars(layer_top, layer_bottom, observation_date), as.numeric) %>%
  filter(layer_top == 0) #only look at the top layer
```

```
ggplot(soil.df) +
  borders("world", colour="gray80", fill="gray80") + # create a layer of borders
  geom_hex(aes(x=lon, y=lat), bins=250) +
  scale_fill_gradient(trans='log10') +
  theme_bw() +
  theme(text=element_text(size=18),
        axis.title.x = element_blank(), axis.title.y=element_blank())
```

```
##
## Attaching package: 'maps'
```

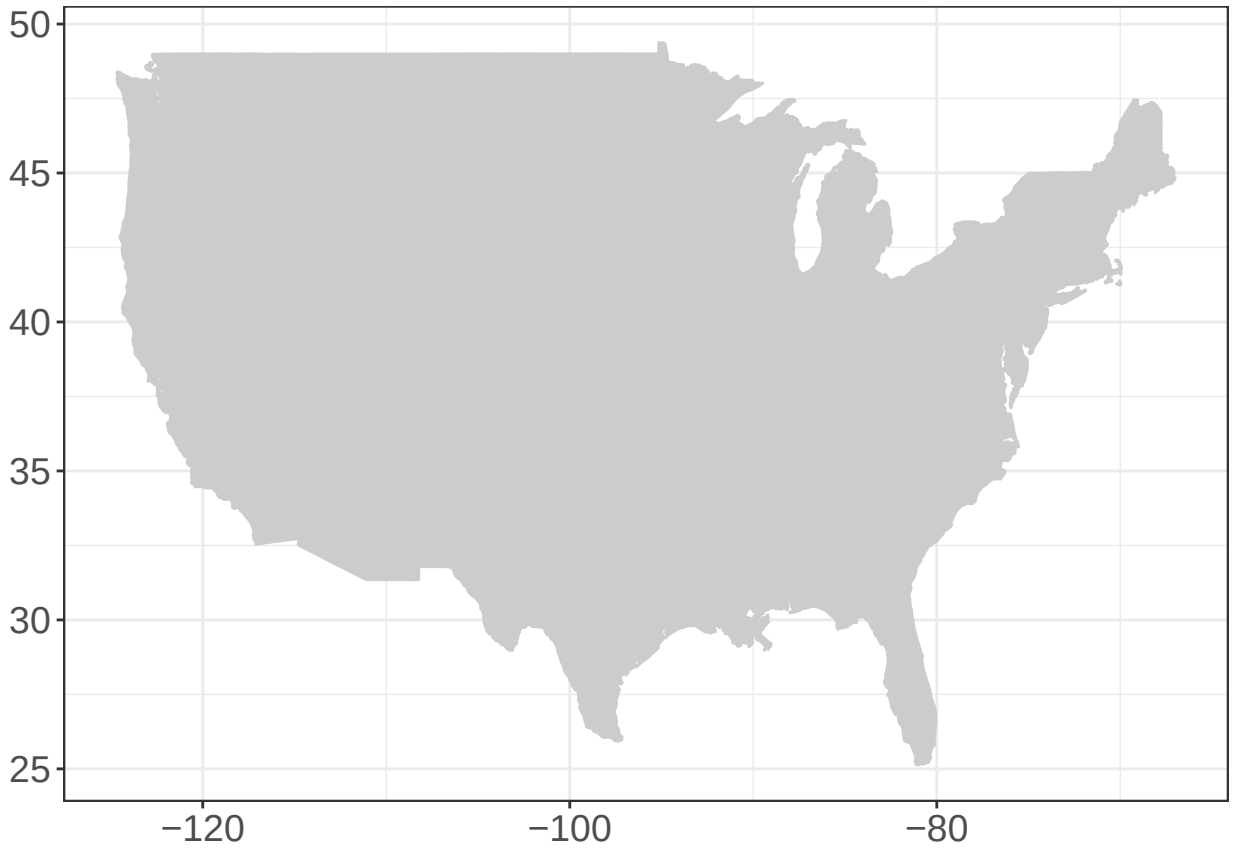
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
## Warning: Computation failed in `stat_binhex()`:
## Package `hexbin` required for `stat_binhex`.
## Please install and try again.
```

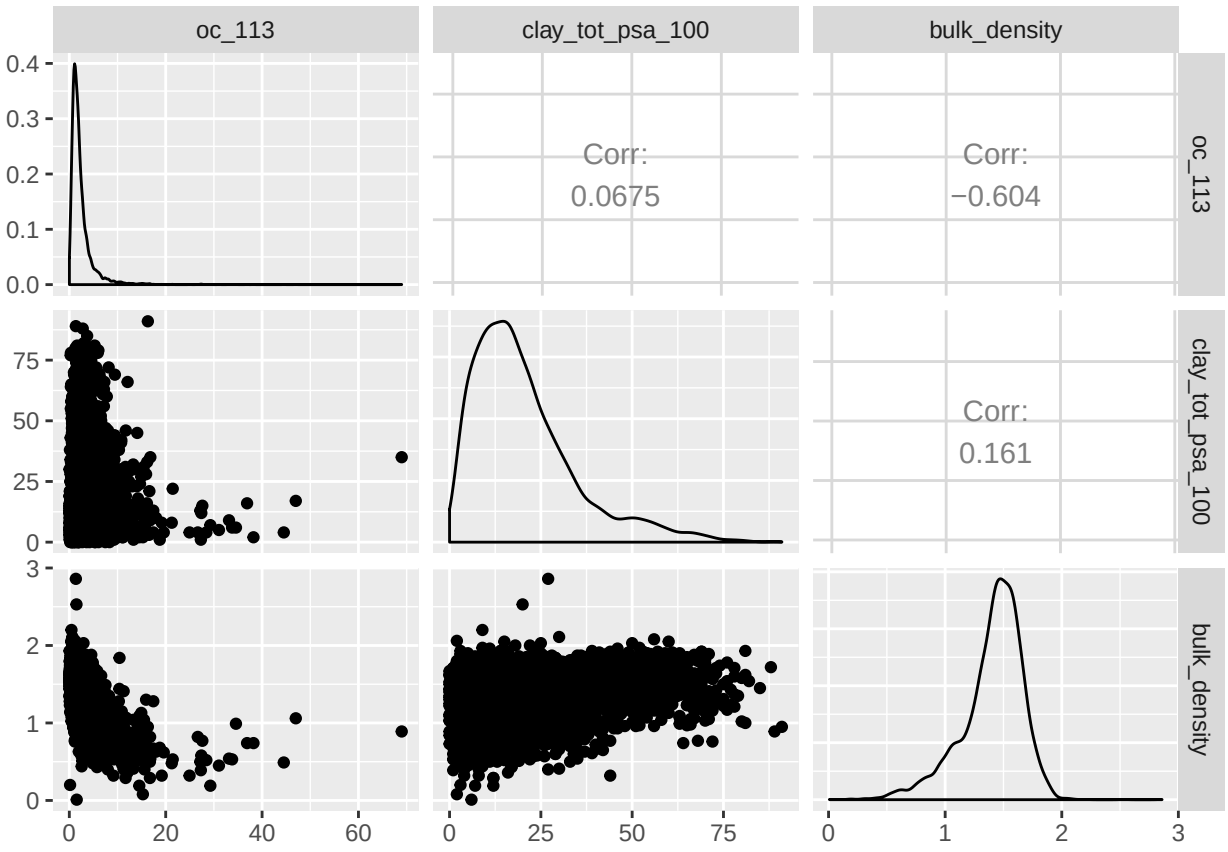



```
ggplot(soil.df %>% filter(mapsID == 'USA') )+
  borders("state", colour="gray80", fill="gray80") + # create a layer of borders
  geom_hex(aes(x=lon, y=lat), bins=200) +
  #scale_fill_gradient(trans='log10') +
  theme_bw() +
  theme(text=element_text(size=18),
        axis.title.x = element_blank(), axis.title.y=element_blank())
```

```
## Warning: Computation failed in `stat_binhex()`:
## Package `hexbin` required for `stat_binhex`.
## Please install and try again.
```




```
GGally::ggpairs(soil.df %>%
                  dplyr::select(oc_113, clay_tot_psa_100, bulk_density))
```



Meterological data

```
##This downloads 3.3Gb of data and might take a long time
downloadList <- c(
  sprintf('ftp://ftp.cdc.noaa.gov/Datasets/cpc_global_precip/precip.%d.nc', 1979:2018),
  #sprintf('ftp://ftp.cdc.noaa.gov/Datasets/cpc_global_temp/tmax.%d.nc', 1979:2018), Only need MAT so d
  #sprintf('ftp://ftp.cdc.noaa.gov/Datasets/cpc_global_temp/tmin.%d.nc', 1979:2018),
  'ftp://ftp.cdc.noaa.gov/Datasets/cpc_global_temp/tmax.day.ltm.nc',
  'ftp://ftp.cdc.noaa.gov/Datasets/cpc_global_temp/tmin.day.ltm.nc')

downloadList <- downloadList[!file.exists(file.path(met_dir, basename(downloadList)))]

for(downloadFile in downloadList){
  download.file(downloadFile, file.path(met_dir, basename(downloadFile)))
}
```

Load the mean anaual temperature and find the nearest neighbor to the soil sites.

```
loadFiles <- list(Tmin=file.path(met_dir, 'tmin.day.ltm.nc'),
                  Tmax=file.path(met_dir, 'tmax.day.ltm.nc'))

tmin.nc <- nc_open(loadFiles$Tmin)
tmin.lon <- ncvar_get(tmin.nc, varid = "lon")-360*(ncvar_get(tmin.nc, varid = "lon")>180)
tmin.lat <- ncvar_get(tmin.nc, varid = "lat")
tmin.value <- ncvar_get(tmin.nc, varid='tmin')
```

```
nc_close(tmin.nc)
tmax.nc <- nc_open(loadFiles$Tmax)
tmax.value <- ncvar_get(tmax.nc, varid='tmax')
nc_close(tmax.nc)

met.df <- soil.df %>% dplyr::select(lon, lat) %>% unique %>%
  group_by(lon, lat) %>%
  mutate(nearLon = which.min(abs(lon-tmin.lon)),
         nearLat = which.min(abs(lat-tmin.lat))) %>%
  mutate(t_lon=tmin.lon[nearLon],
         t_lat=tmin.lat[nearLat],
         MAT=mean(tmin.value[nearLon, nearLat,]+tmax.value[nearLon, nearLat,])/2)
```

Precip is tricker then MAT. Below is a calculation for the Gini index* which is the area bounded by percentage of total annual rainfall (y) vs percentage of time period (x) and the 1:1 line. To calculate this we order the precipitation by day, take the cumulative sum of precip and days, normalize this to the total then integrate to come up with this bounded area.

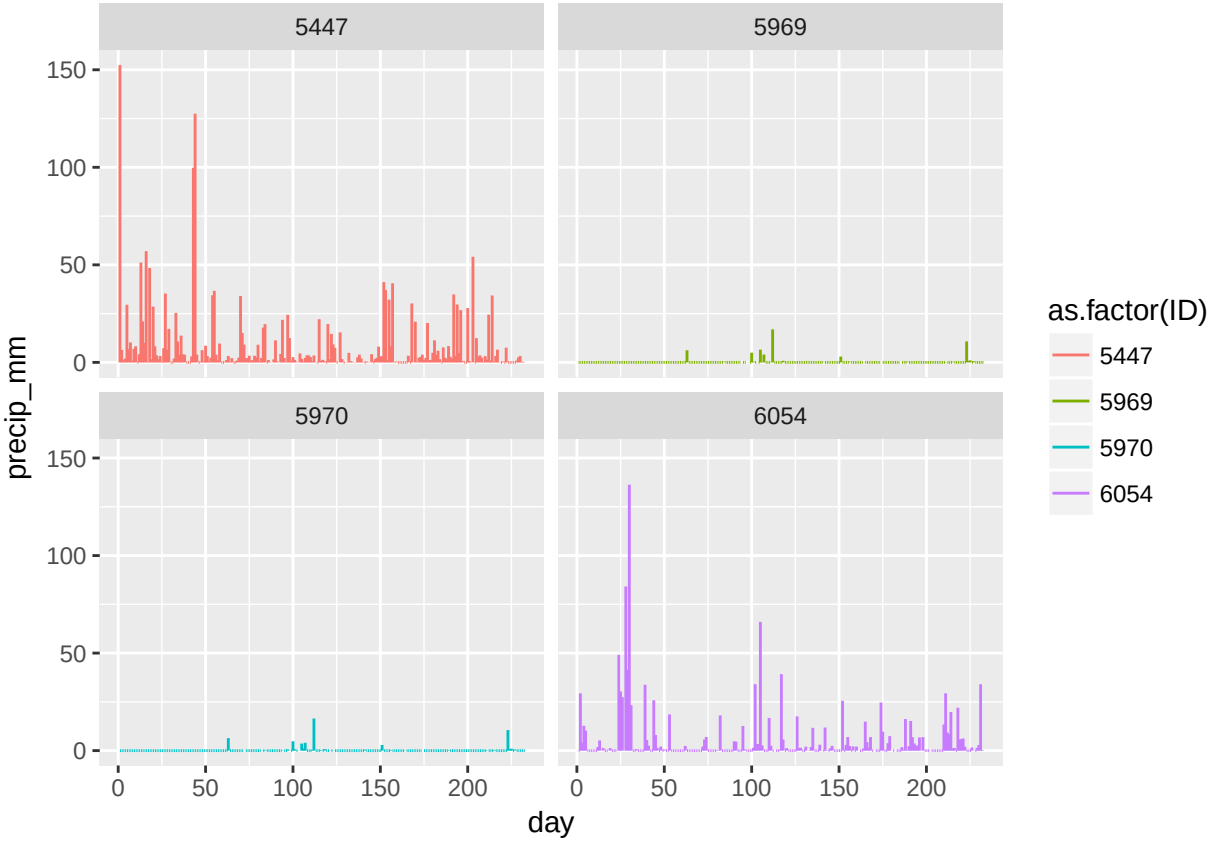
- Rajah et al. "Changes to the temporal distirbution of daily precipitation" GRL (2014) 10.1002/2014GL062156

```
if(file.exists(file.path(workingTemp_dir, 'giniPrecip.RData'))){
  load(file.path(workingTemp_dir, 'giniPrecip.RData'))
}else{
  gini.df <- data.frame()
  precip.raster <- brick(file.path(met_dir, 'precip.1980.nc'))
  sample.points <- SpatialPoints(coords=cbind(ifelse(met.df$lon < 0, 360 + met.df$lon, met.df$lon),
                           met.df$lat), proj4string = crs(precip.raster))
  for(yr in (1979:2018)){
    print(yr)

    precip.raster <- brick(file.path(met_dir, sprintf('precip.%d.nc', yr)))

    precip.value <- raster::extract(precip.raster, sample.points, method='simple', df=TRUE) %>%
      mutate(lon=met.df$lon, lat=met.df$lat) %>%
      group_by(ID, lon, lat) %>%
      gather(key='dateStr', value='precip_mm', starts_with('X')) %>%
      #separate(dateStr, into=c('year', 'month', 'mday', 'drop1', 'drop2', 'drop3'), sep='\\.') %>%
      #select(-starts_with('drop')) %>%
      dplyr::mutate(year=as.numeric(substr(dateStr, 2, 5)),
                    day=lubridate::yday(lubridate::ymd(substr(dateStr,2,11)))) %>%
      dplyr::select(-dateStr) %>%
      dplyr::arrange(ID, precip_mm, .by_group=TRUE) %>%
      dplyr::mutate(#cumPrecip = cumsum(ifelse(is.finite(precip_mm), precip_mm, 0)),
        #cumTime = cumsum(is.finite(precip_mm)),
        precipFrac= cumsum(ifelse(is.finite(precip_mm), precip_mm, 0))/sum(precip_mm, na.rm=TRUE),
        timeFrac = cumsum(is.finite(precip_mm))/sum(is.finite(precip_mm)))

    gini.df <-  precip.value %>%
      group_by(ID, lat, lon, year) %>%
      summarize(gini=sum(timeFrac - precipFrac, na.rm=TRUE)/sum(is.finite(precip_mm))*2,
                TotPrecip=sum(precip_mm, na.rm=TRUE),
                n=length(timeFrac),
                finiteCount = sum(is.finite(precip_mm))) %>%
      bind_rows(gini.df)
```

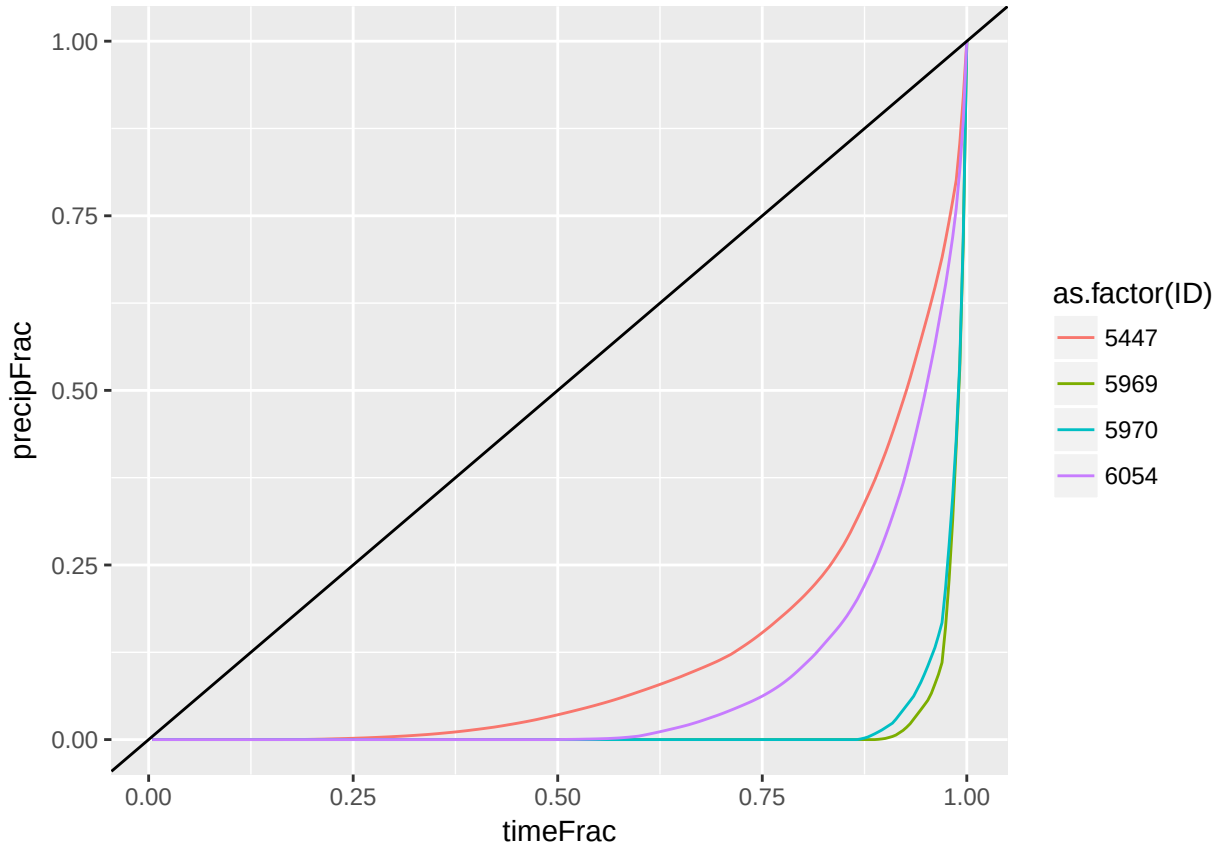
```
  }
  save(gini.df, precip.value, file=file.path(workingTemp_dir, 'giniPrecip.RData'))
  rm(precip.raster, sample.points)
}
```

```
shortIDS <- c(5447, 5969, 6054,  5970)
ggplot(precip.value %>% filter(ID %in% shortIDS)) +
  geom_segment(aes(x=day, y=precip_mm, xend=day, yend=0, color=as.factor(ID))) +
  facet_wrap(~ID)
```

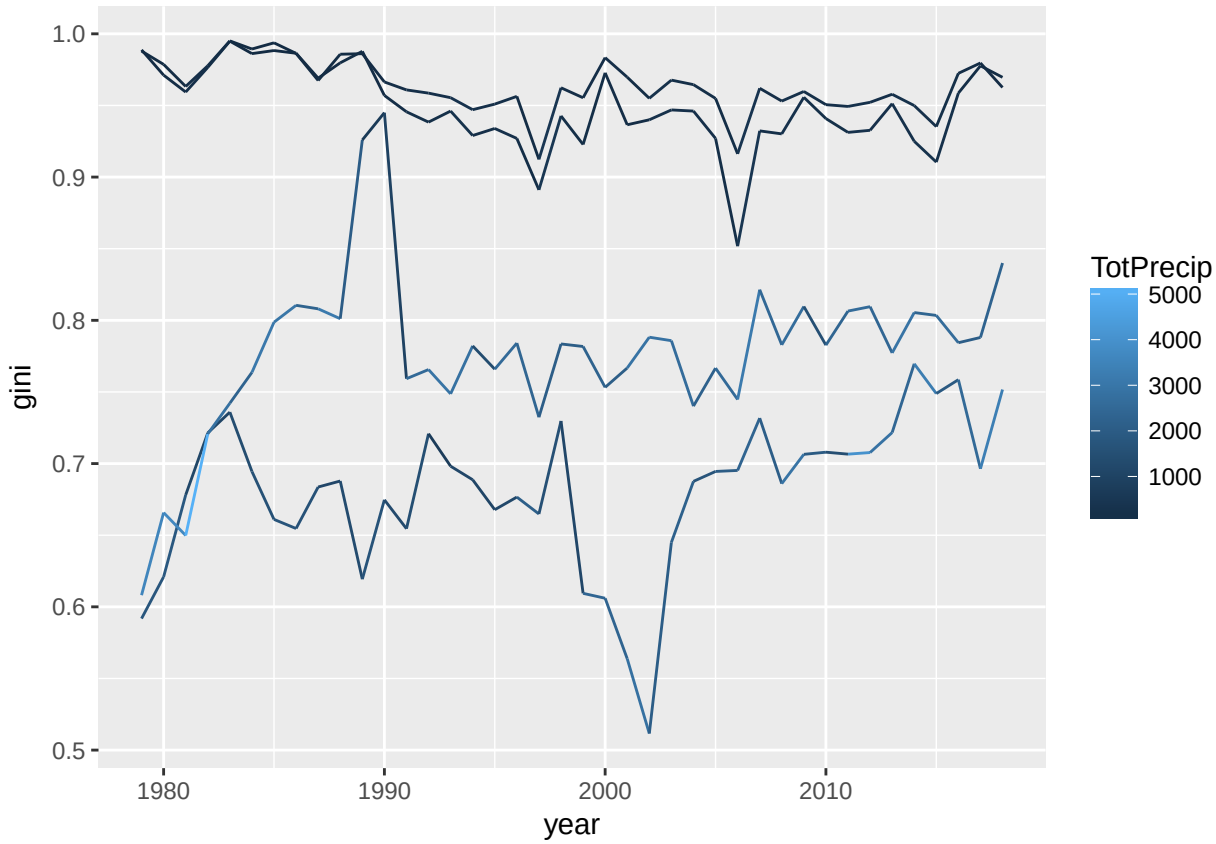
```
## Warning: Removed 2 rows containing missing values (geom_segment).
```




```
ggplot(precip.value %>% filter(ID %in% shortIDS)) +
  geom_line(aes(x=timeFrac, y=precipFrac, color=as.factor(ID))) +
  geom_abline(slope=1)
```




```
ggplot(gini.df %>% filter(ID %in% c(5447, 5969, 6054,  5970))) +
  geom_line(aes(x=year, y=gini, group=ID, color=TotPrecip))
```

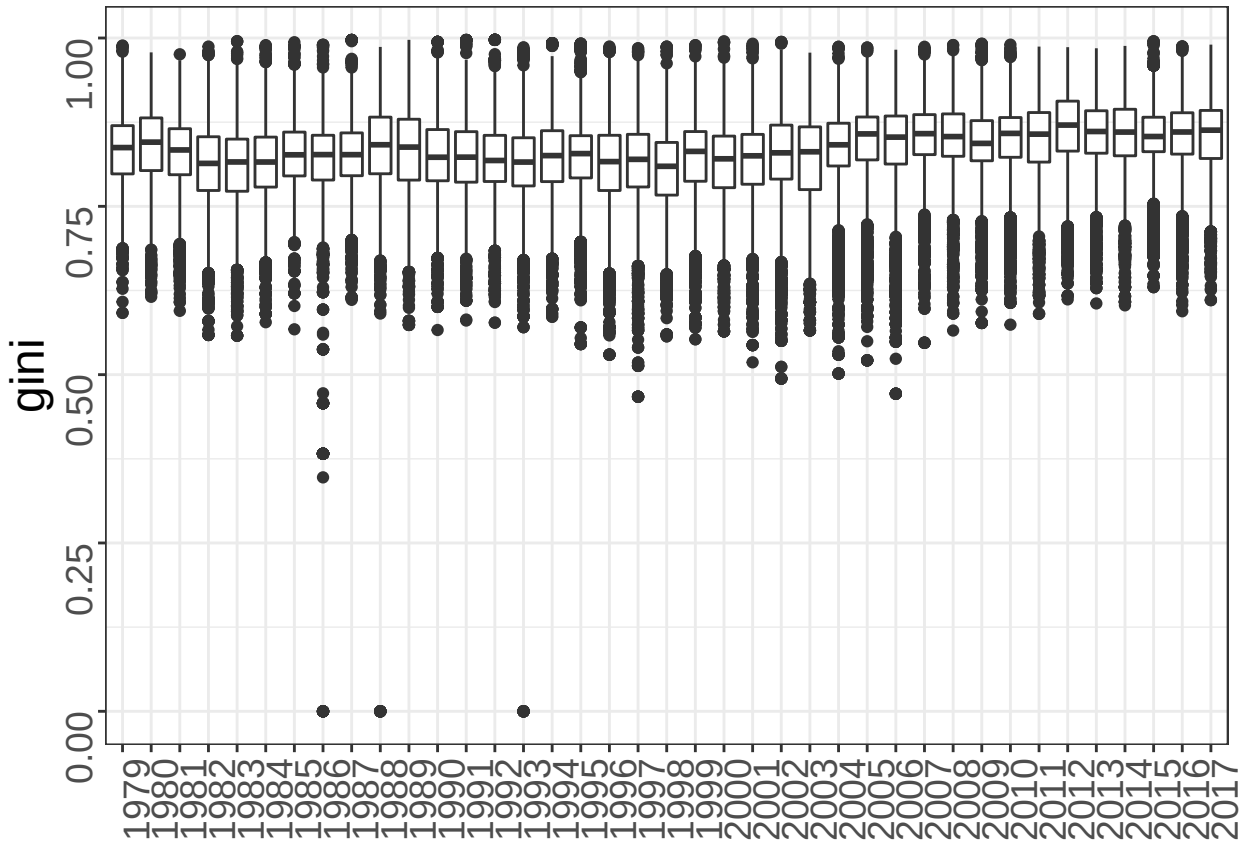



```
gini.df %>%
  filter(ID %in% shortIDS, year == precip.value$year[1])%>%
  print
```

```
## # A tibble: 4 x 8
## # Groups:   ID, lat, lon [4]
##      ID   lat    lon  year  gini TotPrecip     n finiteCount
##   <dbl> <dbl>  <dbl> <dbl> <dbl>     <dbl> <int>       <int>
## 1  5447 10.3   125.   2018 0.752    1910.    232         232
## 2  5969  1.94   44.8  2018 0.970      58.9   232         231
## 3  5970  2.75   45.3  2018 0.963      58.4   232         231
## 4  6054 17.6   -92.0  2018 0.840    1196.    232         232
```

```
met.df <- gini.df %>%
  filter(finiteCount > 0.95*365) %>% #only take years w/ 95%+ completeness
  summarize(MAP= mean(TotPrecip),
            precip_years = length(year),
            MAGini = mean(gini)) %>%
  full_join(met.df, by=c('lat', 'lon'))
```

```
ggplot(gini.df %>% filter(finiteCount > 0.95*365)) +
  geom_boxplot(aes(x=as.factor(year), y=gini)) + theme_bw() +
  theme(text=element_text(size=18, angle=90),
        axis.title.x = element_blank())
```




```
ggplot(gini.df) +
  geom_hex(aes(x=gini, y=TotPrecip), bins=100) +
  scale_fill_gradient(high='#CCCC00', low="#132B43", trans='log10')
```

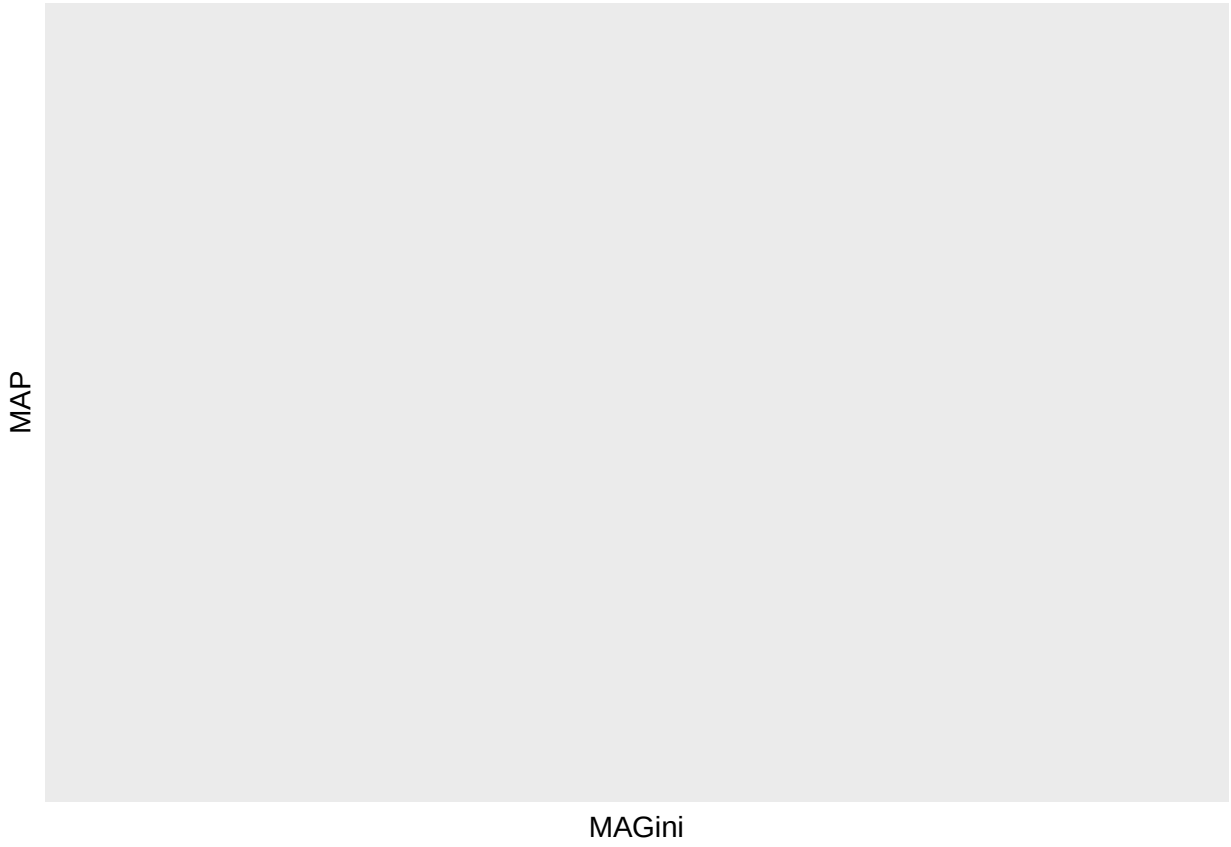
```
## Warning: Removed 712 rows containing non-finite values (stat_binhex).
```

```
## Warning: Computation failed in `stat_binhex()`:
## Package `hexbin` required for `stat_binhex`.
## Please install and try again.
```

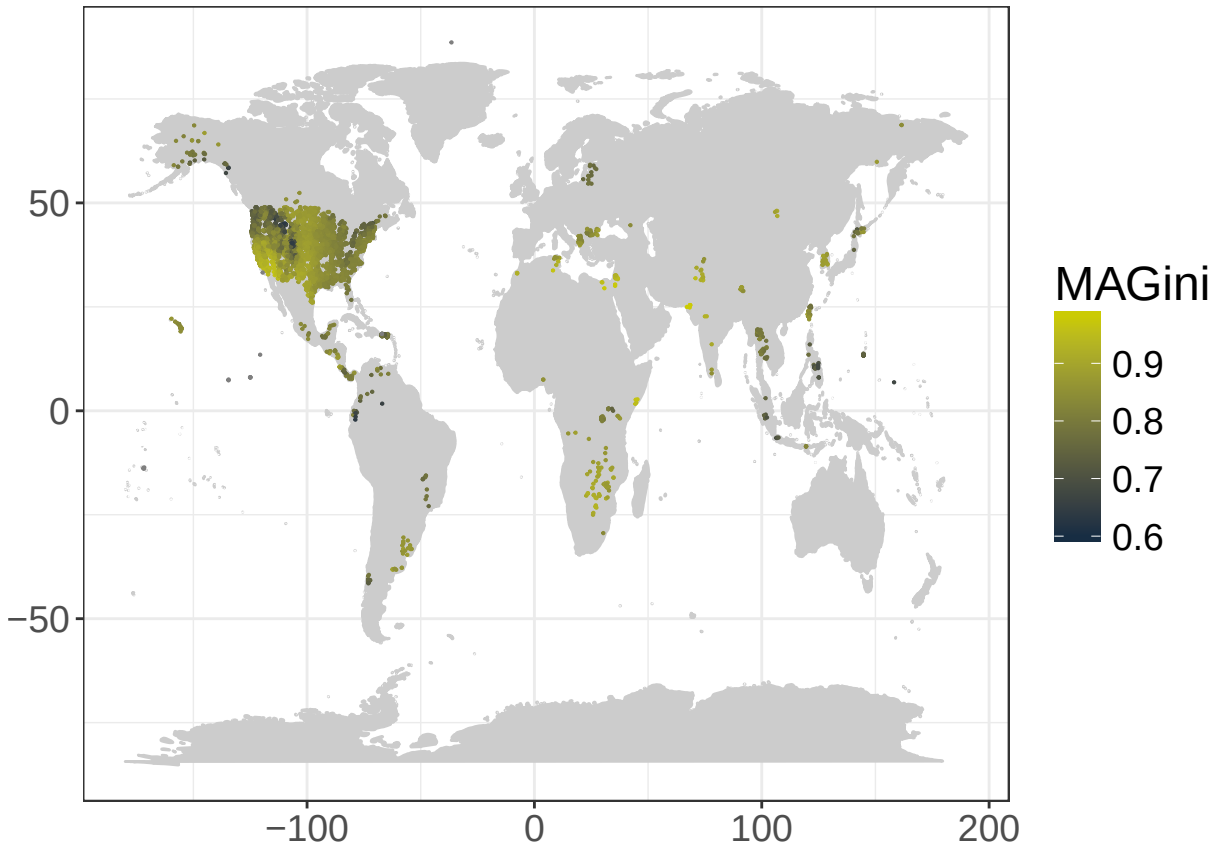

```
ggplot(met.df) +
  geom_hex(aes(x=MAGini, y=MAP)) +
  scale_fill_gradient(high='#CCCC00', low="#132B43", trans='log10')
```

```
## Warning: Removed 17 rows containing non-finite values (stat_binhex).
```

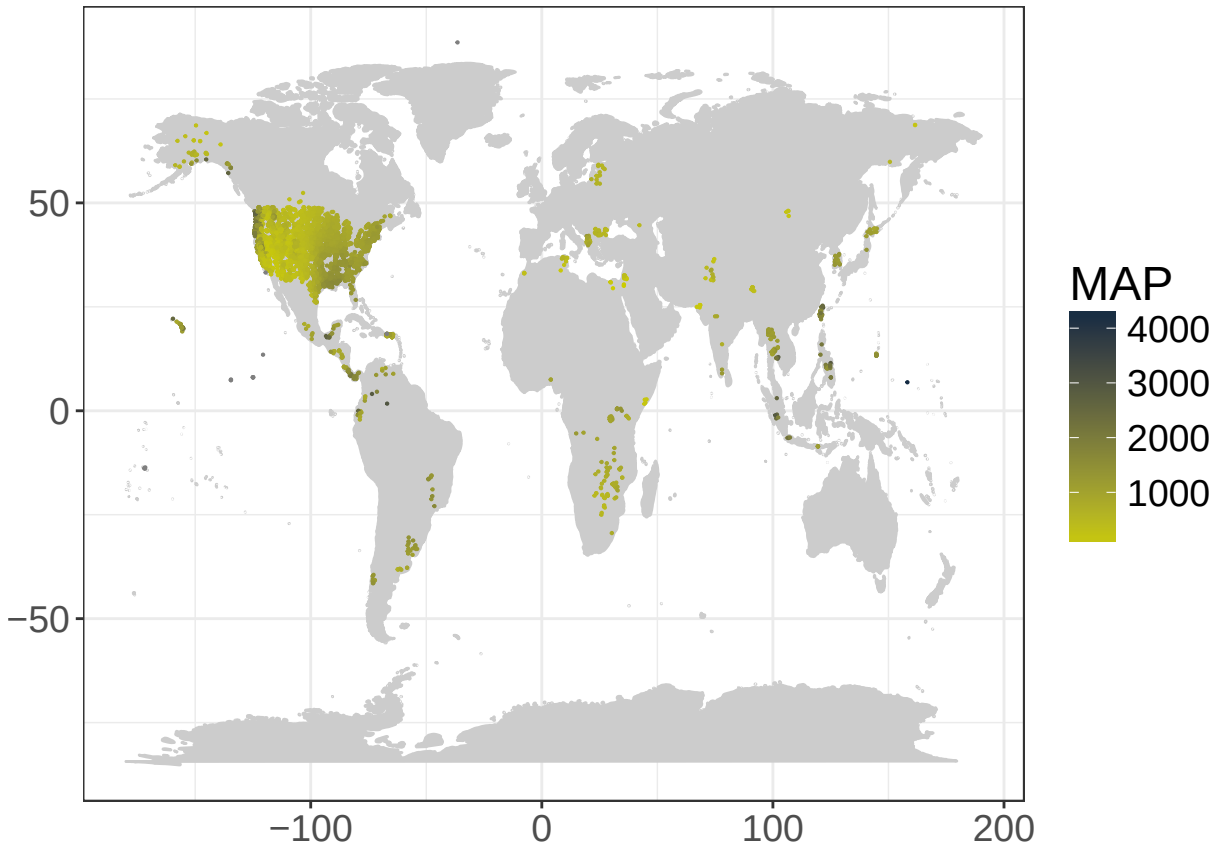
```
## Warning: Computation failed in `stat_binhex()`:
## Package `hexbin` required for `stat_binhex`.
## Please install and try again.
```




```
ggplot(met.df) +
  borders("world", colour="gray80", fill="gray80") + # create a layer of borders
  geom_point(aes(x=lon, y=lat, color=MAGini), size=0.1) +
  scale_color_gradient(high='#CCCC00', low="#132B43") +
  theme_bw() +
  theme(text=element_text(size=18),
        axis.title.x = element_blank(), axis.title.y=element_blank())
```




```
ggplot(met.df) +
  borders("world", colour="gray80", fill="gray80") + # create a layer of borders
  geom_point(aes(x=lon, y=lat, color=MAP), size=0.1) +
  scale_color_gradient(low='#CCCC00', high="#132B43") +
  theme_bw() +
  theme(text=element_text(size=18),
        axis.title.x = element_blank(), axis.title.y=element_blank())
```

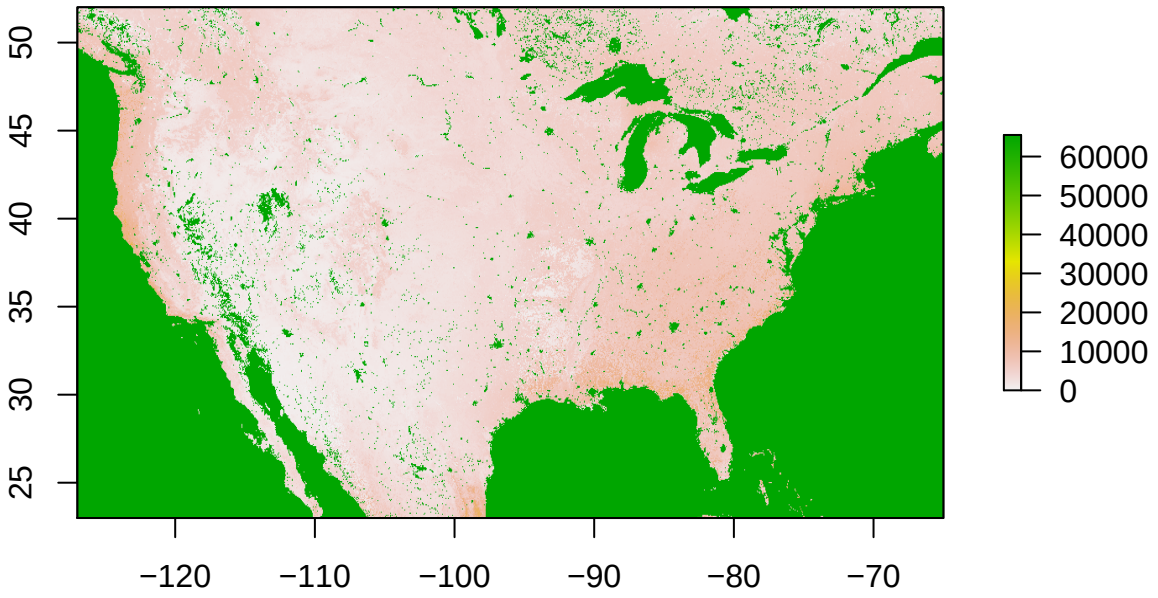


Net primary production

Please acknowledge Dr William Kobly Smith (University of Arizona) for their help identifying and interperating the correct data product.

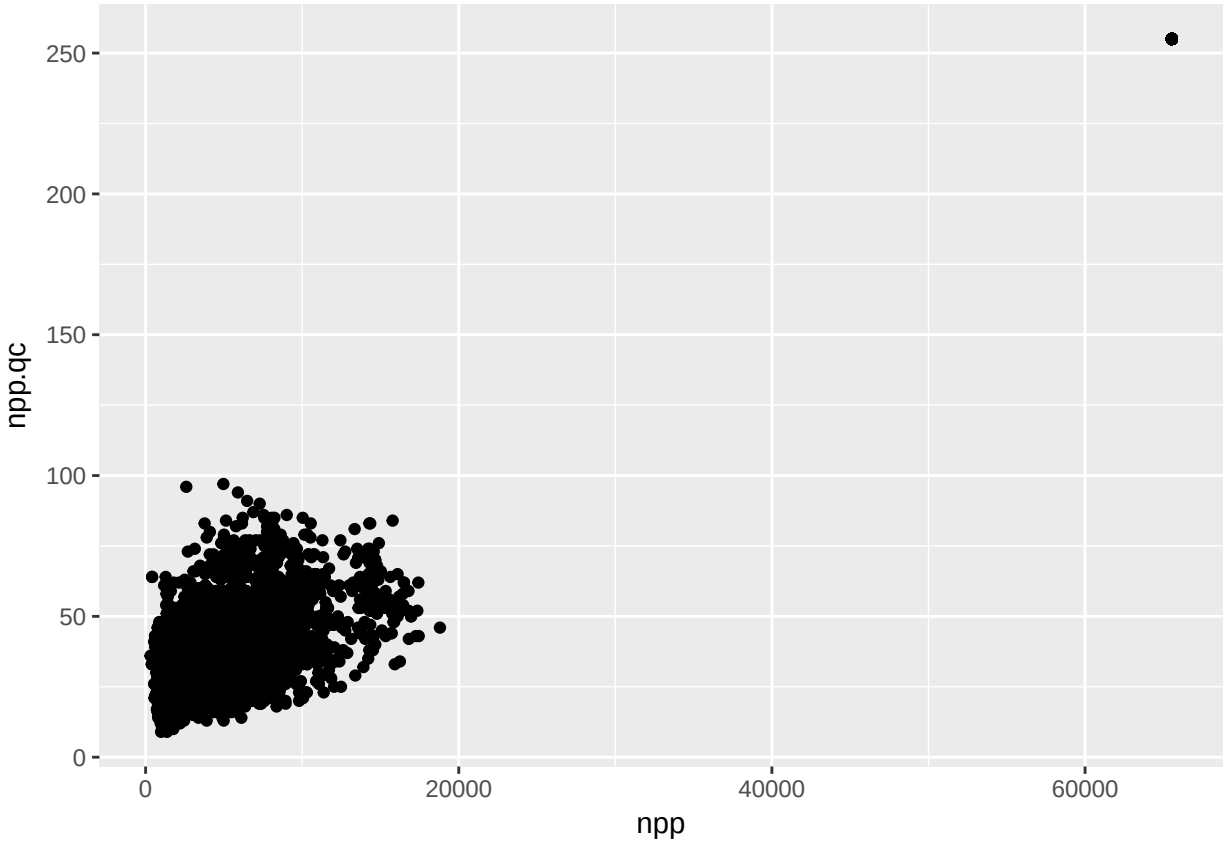
```
npp.filename <- 'http://files.ntsg.umt.edu/data/NTSG_Products/MOD17/GeoTIFF/MOD17A3/GeoTIFF_30arcsec/MO
nppqc.filename <- 'http://files.ntsg.umt.edu/data/NTSG_Products/MOD17/GeoTIFF/MOD17A3/GeoTIFF_30arcsec/
NPPFile <- file.path(workingTemp_dir, "MOD17A3_Science_NPP_mean_00_15.tif")
QCFile <- file.path(workingTemp_dir, "MOD17A3_Science_NPP_Qc_mean_00_15.tif")

if(!file.exists(NPPFile)) download.file(npp.filename, NPPFile)
if(!file.exists(QCFile)) download.file(nppqc.filename, QCFile)
#file.exists(NPPFile)
npp.raster <- raster(x=NPPFile)
qc.raster <- raster(x=QCFile)
#npp.raster@file@nodatavalue <- 65535 #reset the -Inf no data value

soil.df <- soil.df %>%
  mutate(npp = raster::extract(npp.raster, SpatialPoints(cbind(lon, lat), proj4string=crs(npp.raster)))
         npp.qc = raster::extract(qc.raster, SpatialPoints(cbind(lon, lat), proj4string=crs(npp.raster)

#NPP units are in g-C m-2 yr-1
npp.df <- soil.df %>% dplyr::select(lat, lon, mapsID) %>% unique %>%
  mutate(npp = raster::extract(npp.raster, SpatialPoints(cbind(lon, lat), proj4string=crs(npp.raster)))
         npp.qc = raster::extract(qc.raster, SpatialPoints(cbind(lon, lat), proj4string=crs(npp.raster)
```

```
plot(npp.raster, xlim=c(-127, -65), ylim=c(23, 52))
```



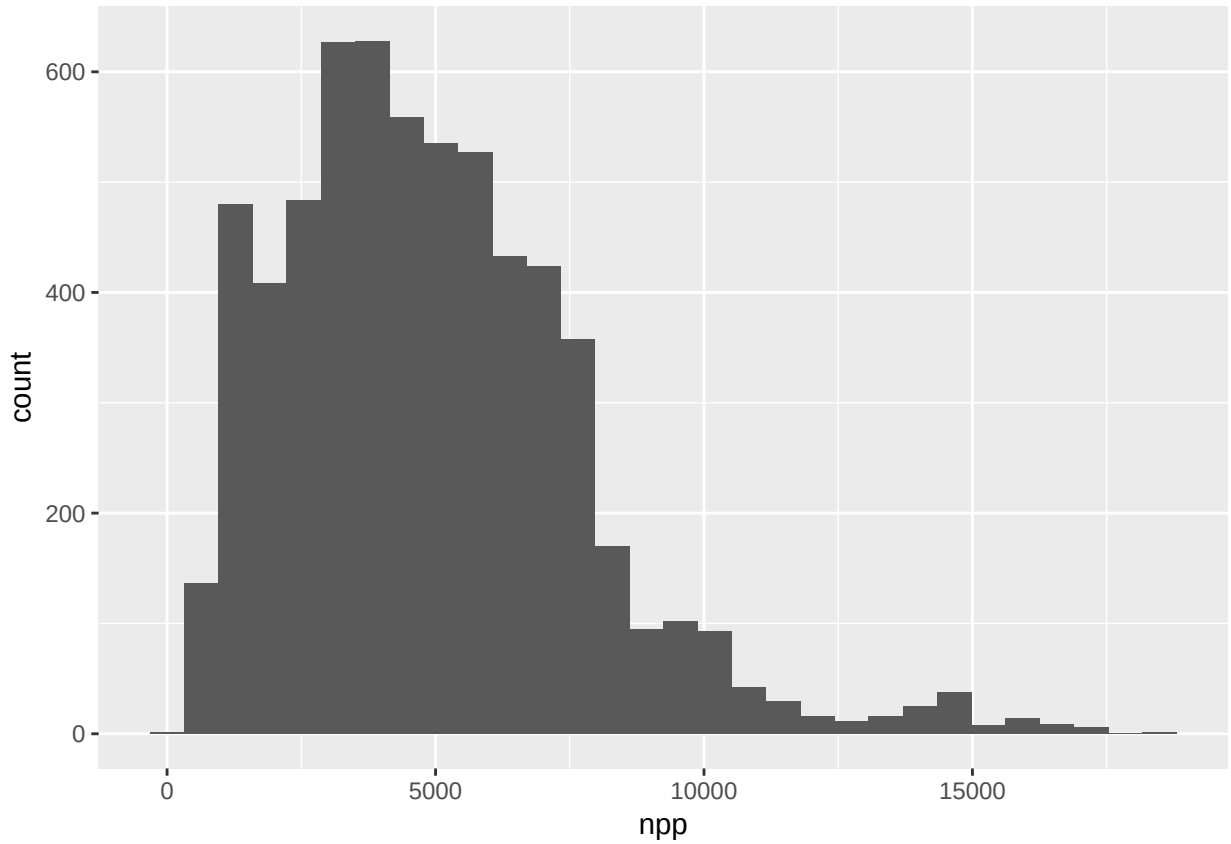

```
ggplot(npp.df) + geom_point(aes(x=npp, y=npp.qc))
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

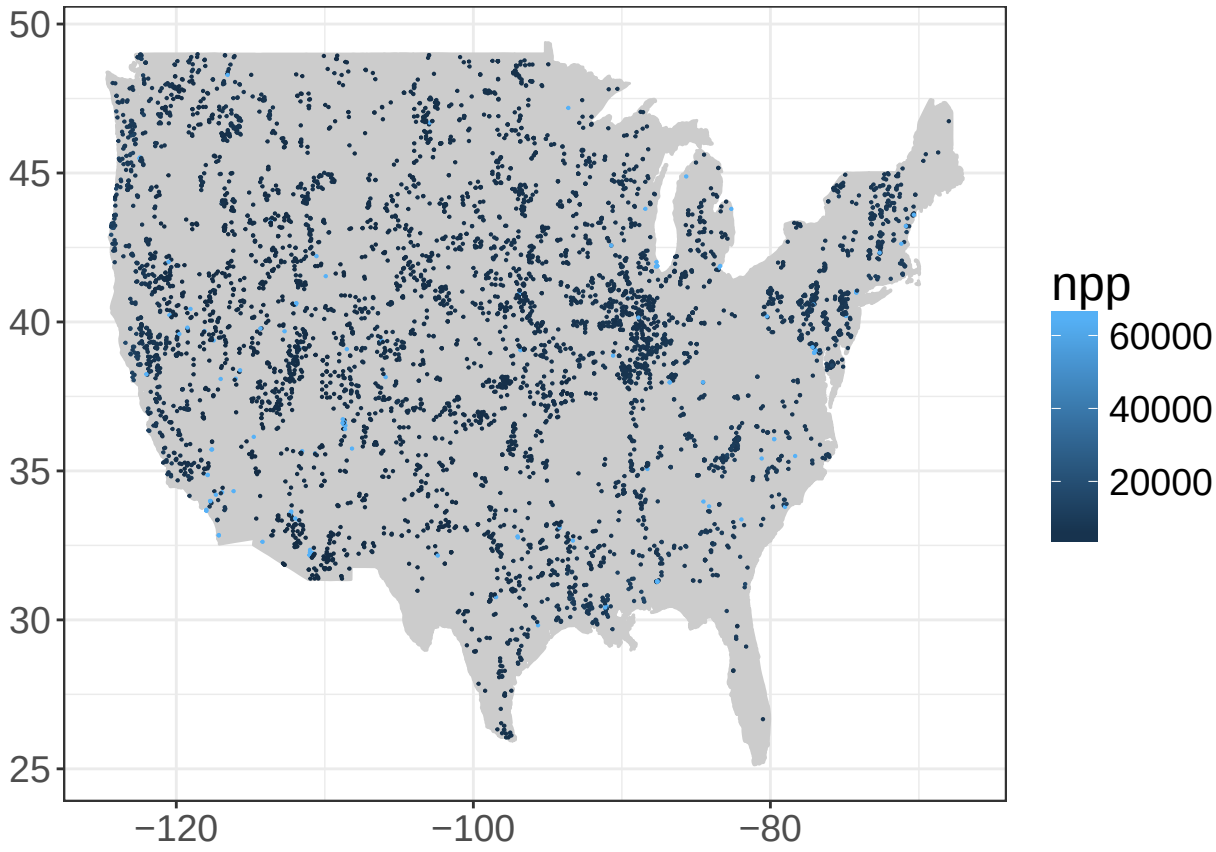



```
ggplot(npp.df %>% filter(npp < 6e4)) + geom_histogram(aes(x=npp))
```

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




```
ggplot(npp.df %>% filter(mapsID == 'USA') )+
  borders("state", colour="gray80", fill="gray80")  +
  #borders("world", colour="gray80", fill="gray80") +# create a layer of borders
  geom_point(aes(x=lon, y=lat, color=npp), size=0.1) +
  #geom_point(data=(npp.df %>% filter(NPP > 6e4 | is.na(NPP))), aes(x=lon, y=lat), color='red', size=0.
  theme_bw() +
  theme(text=element_text(size=18),
        axis.title.x = element_blank(), axis.title.y=element_blank())
```

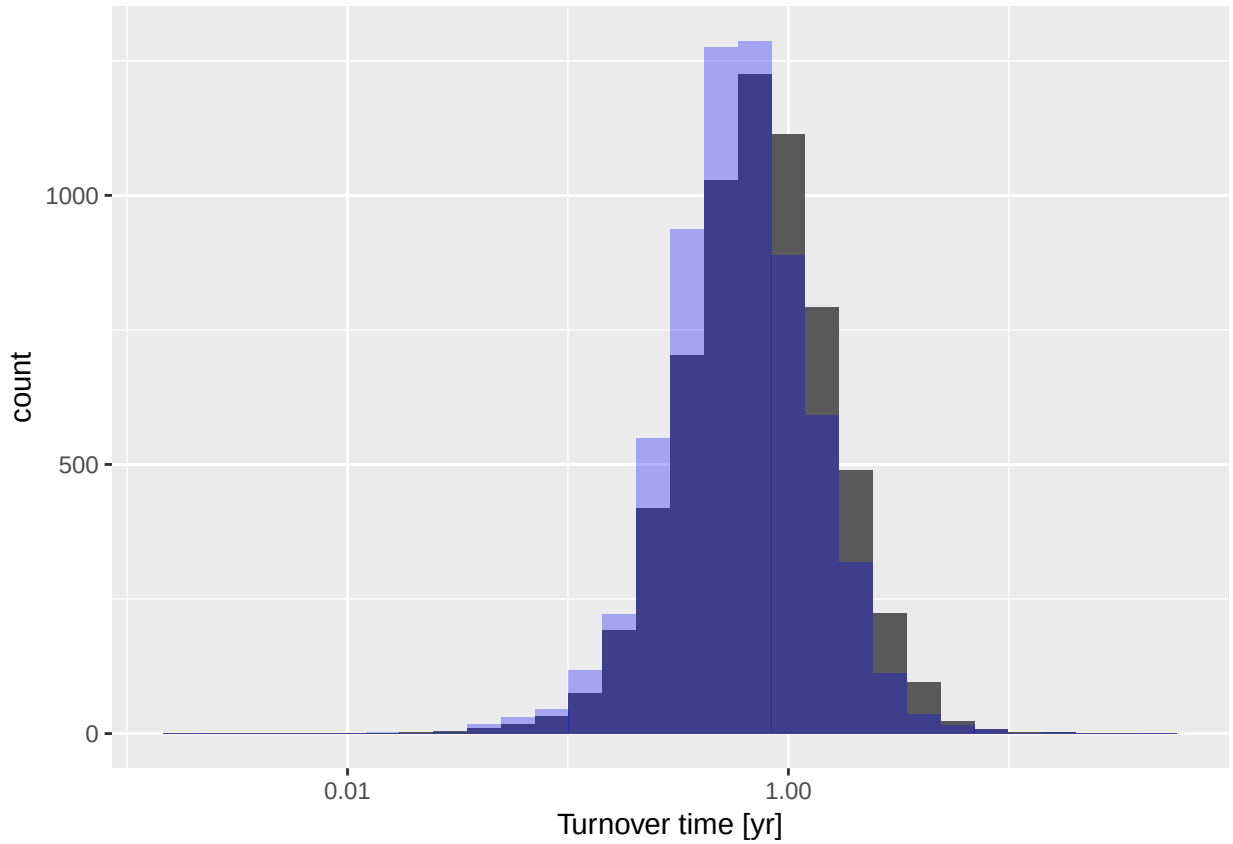


Data analysis

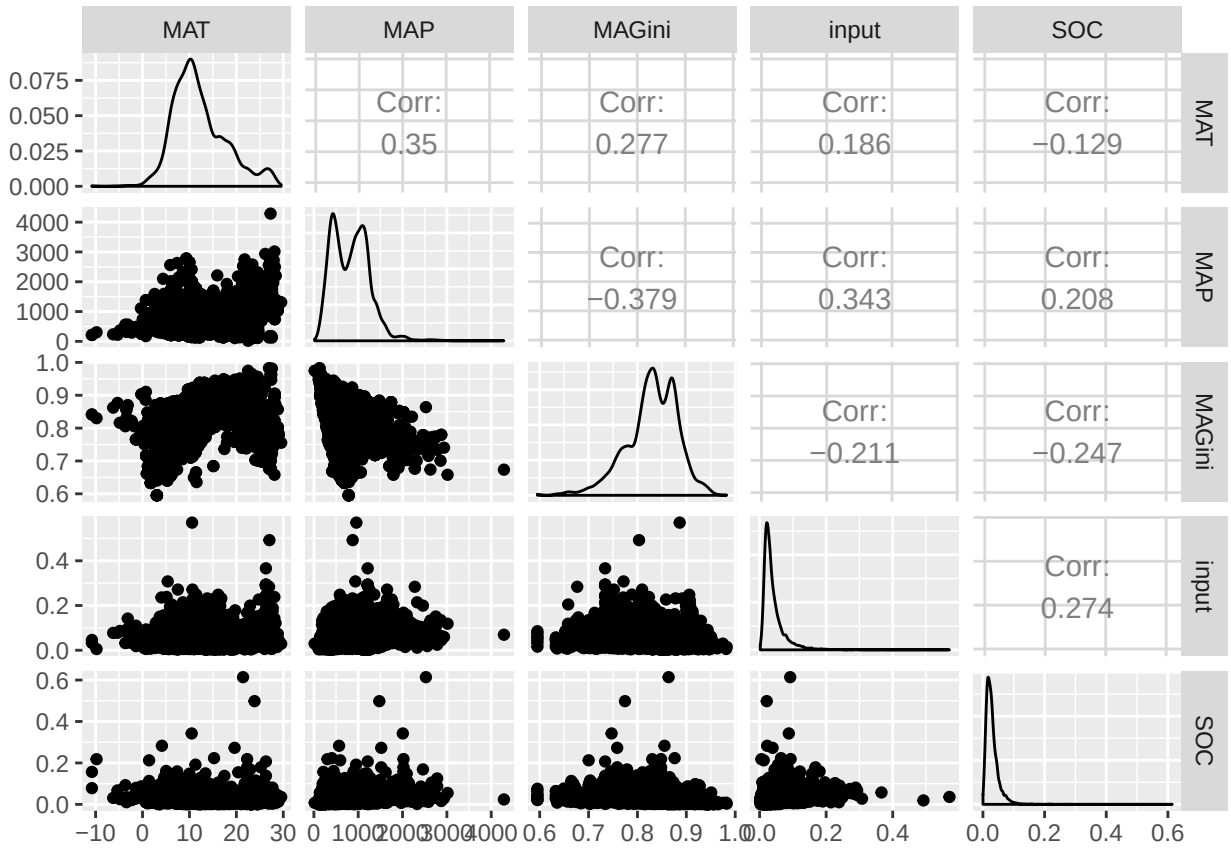
```
data.df <- soil.df %>%
  dplyr::full_join(met.df, by=c('lon', 'lat')) %>%
  mutate(SOC = oc_113/100*bulk_density) %>%
  filter(is.finite(npp) & npp < 6e4 , is.finite(SOC), is.finite(MAT), is.finite(MAP), is.finite(MAGini)
  mutate(input=npp*1e-4/(layer_bottom-layer_top)) %>% #convert from gC m-2 yr to gC cm-2 yr and then ad
  mutate(k_in=input/SOC,
         k_inQ10 = input/(SOC*2^((MAT-15)/10)))
```

```
ggplot(data.df) +
  geom_histogram(aes(x=1/k_in)) +
  geom_histogram(aes(x=1/k_inQ10), fill='blue', alpha=0.3) +
  scale_x_log10() +
  labs(x='Turnover time [yr]')
```

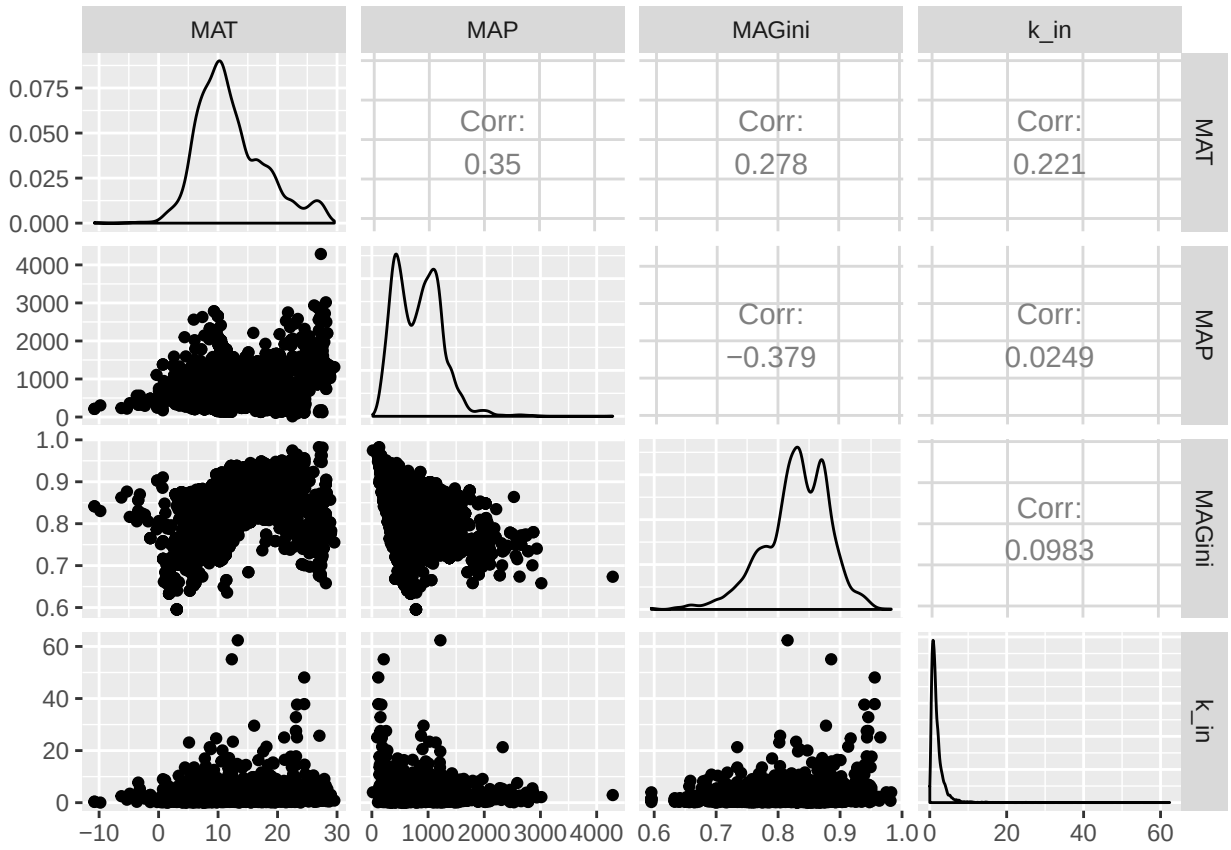
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



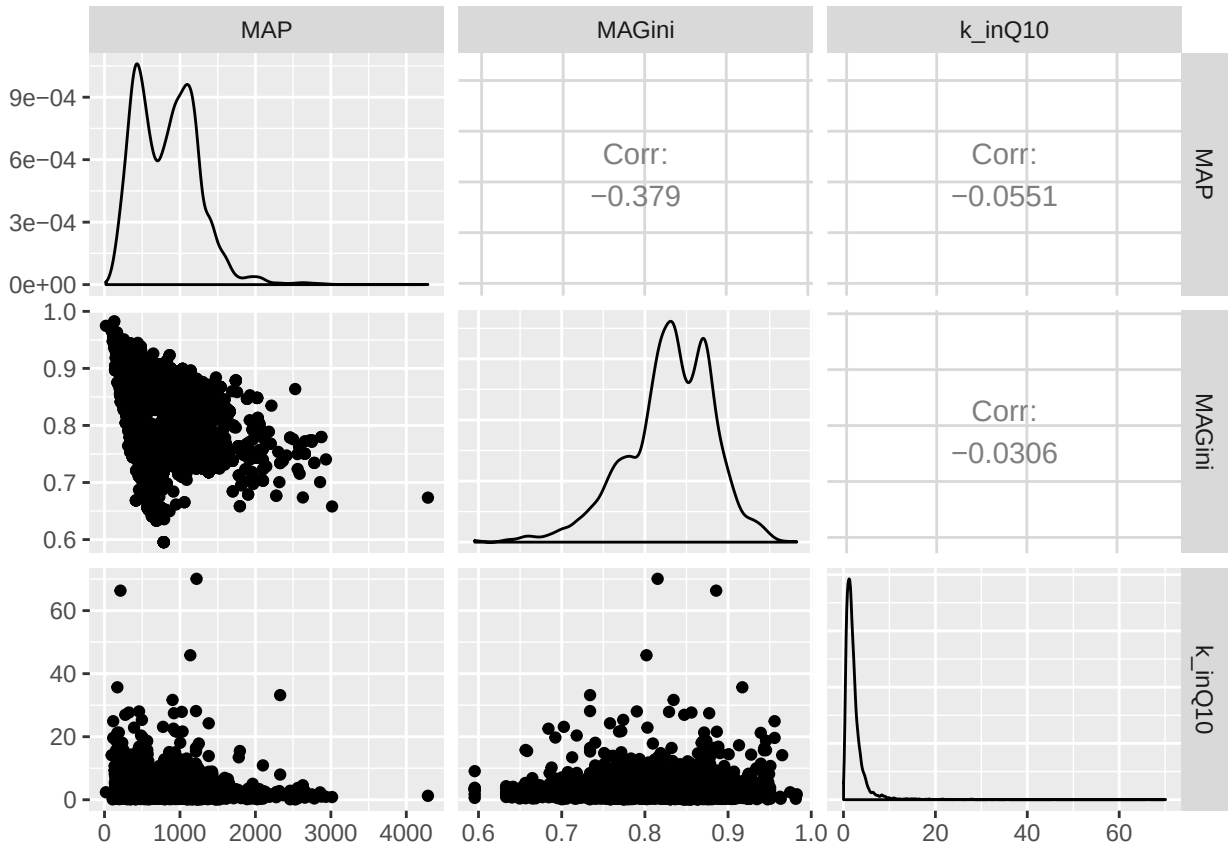

```
GGally::ggpairs(data.df, columns=c('MAT', 'MAP', 'MAGini', 'input', 'SOC'))
```




```
GGally::ggpairs(data.df %>% filter(k_in < 100), columns=c( 'MAT', 'MAP', 'MAGini','k_in'))
```




```
GGally::ggpairs(data.df %>% filter(k_in < 100), columns=c('MAP', 'MAGini', 'k_inQ10'))
```




```
print(head(data.df$soil_taxon[5:10]))
```

```
## [1] Fine, smectitic, frigid Typic Hapluderts
## [2] Fine, smectitic, frigid Typic Argiborolls
## [3] Fine, smectitic, frigid Borollic Natrargids
## [4] Fine, montmorillonitic, frigid Udorthentic Chromusterts
## [5] Fine-loamy, mixed, superactive, frigid Typic Argiborolls
## [6] Fine-loamy, mixed Aridic Haploborolls
## 18362 Levels: . Acid Typic Medisaprists ... XERUMBREPT
```

```
print(paste('Missing soil taxon: ', sum(is.na(data.df$soil_taxon)), 'of', nrow(data.df)))
```

```
## [1] "Missing soil taxon:  562 of 6468"
```

```
soilTaxon_words <- data.frame(words=unlist(strsplit(as.character(unique(data.df$soil_taxon )), split=',
  group_by(words) %>%
  tally %>% arrange(desc(n)) %>%
  mutate(clay_dominated=grepl('argi', words),
         shrink_swell=grepl('(montmor)|(smectit)', words))

print(soilTaxon_words[1:10,])
```

```
## # A tibble: 10 x 4
##    words            n clay_dominated shrink_swell
##    <fct>        <int> <lgl>          <lgl>
##  1 mixed         2882 FALSE          FALSE
##  2 mesic         1473 FALSE          FALSE
##  3 Typic         1361 FALSE          FALSE
##  4 superactive   1112 FALSE          FALSE
```

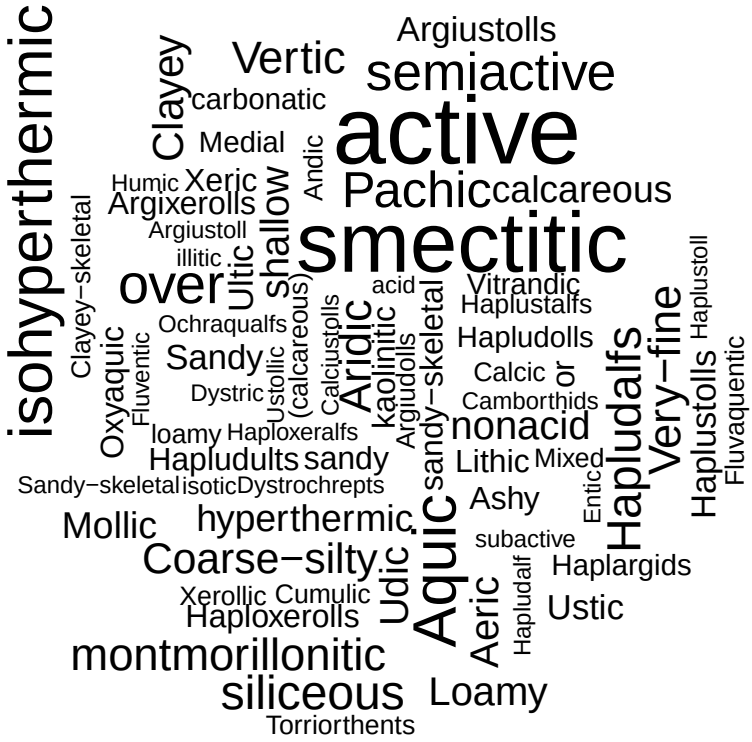
```
##  5 Fine           895 FALSE          FALSE
##  6 Fine-loamy     823 FALSE          FALSE
##  7 thermic        756 FALSE          FALSE
##  8 frigid         569 FALSE          FALSE
##  9 Fine-silty     552 FALSE          FALSE
## 10 Coarse-loamy   535 FALSE          FALSE
```

```
print(soilTaxon_words %>% filter(clay_dominated | shrink_swell))
```

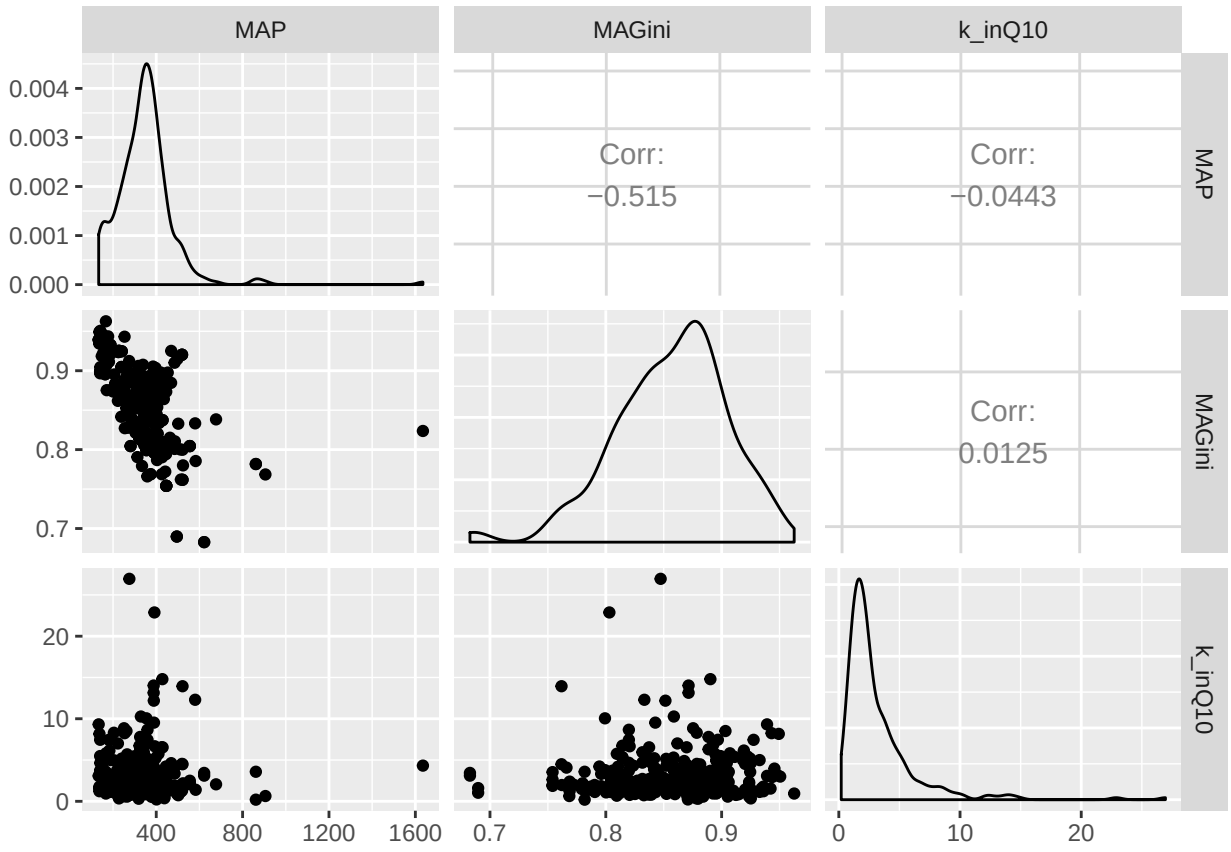
```
## # A tibble: 15 x 4
##    words                n clay_dominated shrink_swell
##    <fct>            <int> <lgl>          <lgl>
##  1 smectitic          388 FALSE          TRUE
##  2 montmorillonitic   164 FALSE          TRUE
##  3 Haplargids          76 TRUE           FALSE
##  4 Natrargids          38 TRUE           FALSE
##  5 Haplargid           30 TRUE           FALSE
##  6 Durargids           17 TRUE           FALSE
##  7 Paleargids          17 TRUE           FALSE
##  8 Natrargid           12 TRUE           FALSE
##  9 Calciargids          8 TRUE           FALSE
## 10 Nadurargids          7 TRUE           FALSE
## 11 Calciargid           6 TRUE           FALSE
## 12 Paleargid            6 TRUE           FALSE
## 13 Durargid             5 TRUE           FALSE
## 14 Natrargidic          2 TRUE           FALSE
## 15 Petroargids          2 TRUE           FALSE
```

```
wordcloud(words=soilTaxon_words$words[-(1:10)], freq=soilTaxon_words$n[-(1:10)],
          min.freq=10, max.words=75, scale=c(3, 0.5), rot.per=0.4)
```

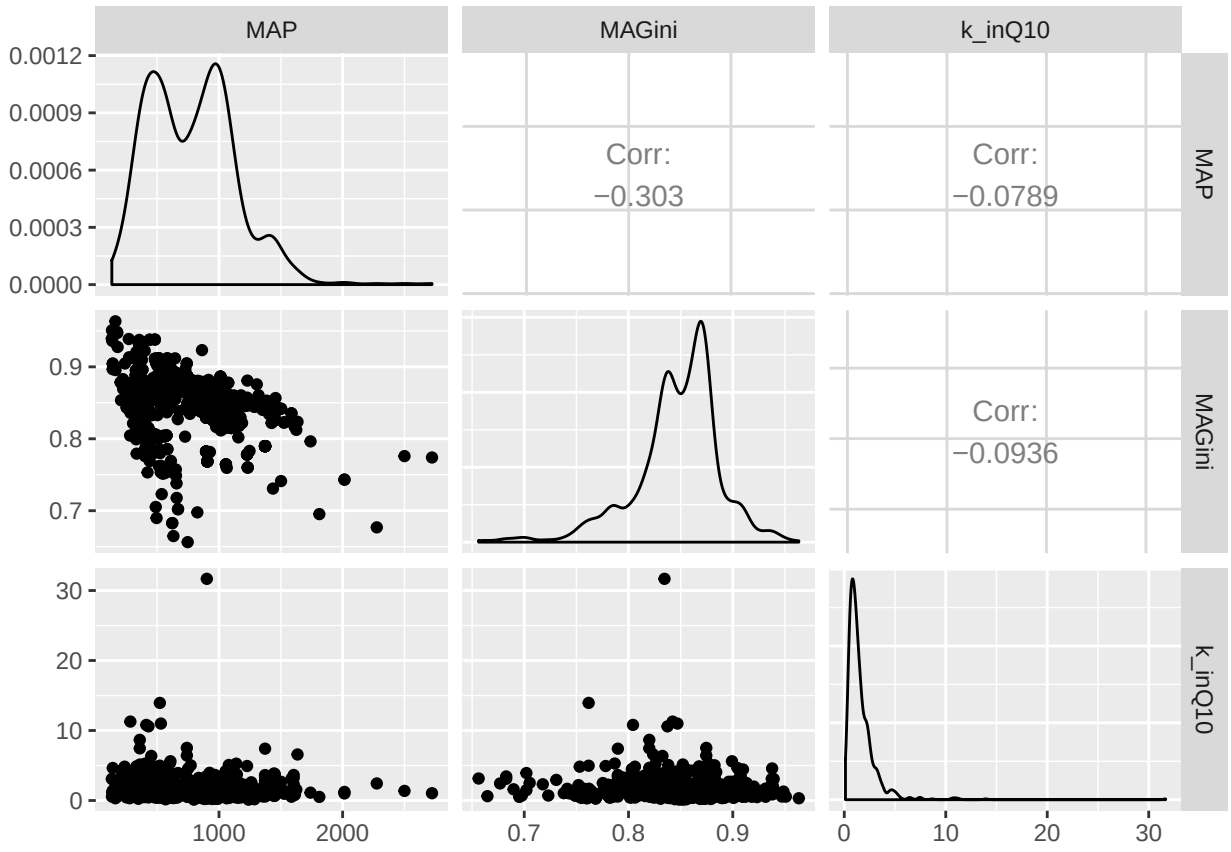
```
## Warning in wordcloud(words = soilTaxon_words$words[-(1:10)], freq =
## soilTaxon_words$n[-(1:10)], : Loamy-skeletal could not be fit on page. It
## will not be plotted.
```




```
#data.df %>% mutate(clay_dominated=grepl('argi', soil_taxon),
#                    shrink_swell=grepl('(montmor)|(smectit)', soil_taxon))
GGally::ggpairs(data.df %>% filter(k_inQ10 < 3e6,
                                   grepl('argi', soil_taxon)), columns=c('MAP', 'MAGini', 'k_inQ10'))
```




```
GGally::ggpairs(data.df %>% filter(k_inQ10 < 3e6,
                                   grepl('(montmor)|(smectit)', soil_taxon)), columns=c('MAP', 'MAGini'
```




```
#data.df %>% filter(grepl('(argi)|(montmor)|(smectit)', soil_taxon)) %>% dplyr::select(soil_taxon) %>% 
```