````
---
title: "FLUXNET-CH4 Upscaling"
author: "Gavin McNicol"
date: "2/22/2021"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Load packages and ggplot theme:

```{r message = F}
library(tidyverse)
library(lubridate)
library(raster)
library(oce) # for time series transformations
source("code/ggplot_theme.R")
```

**NOTE: This work flow uses many large files so most data is stored locally
and requires hard filepaths**
The local file path used for all large files is: `/Volumes/LACIE SHARE/
Stanford CH4/upch4_local/`.

Set the local head directory:

```{r}
loc <- "/Volumes/LACIE SHARE/Stanford CH4/upch4_local/"
```

#### Workflow

  1. Objective of Study
  2. Input Data
      + FLUXNET-CH4 Data
      + Gridded Data
  3. FLUXNET-CH4 Pre-Processing
      + FLUXNET-CH4 Data Preparation
          - Averaging
          - Variable Selection
  4. Gridded Data Pre-Processing
      + Grid Preparation
          - Potential Radiation (computed)
          - MODIS
      + Grid Extraction
          - Canopy Height
          - Computed (Rpot)
          - Compound Topographic Index
          - Earth Environment
          - N and S Deposition
          - SoilGrids
          - TerraClimate
````

- Fractional Vegetation Cover (VCF)
- Wetland Extent (WAD2M)
- WorldClim 2.0

+ Merge Gridded Data
+ Gridded Data Quality Control

5. Finalize Training Data
    + Subset Wetlands, Dates
    + Cluster Sites for Cross Validation
    + Finalize Data
        - Remove Extraneous
        - Impute Missing Data
        - Add Lagged Data
    + FLUXNET-CH4 Data Quality Control
    + Table of Final FLUXNET-CH4 Inputs
6. Forward Feature Selection (FFS)
    + Filter Weekly Data
    + Feature Subset Experiments
    + FFS
        - First Pair
        - Additional Stepwise Features
        - FFS Evolution Plots
    + Summarize FFS Performance
7. Cross Validation
    + ML Model Training
        - RF and Final Predictors or Subsets
        - XGB and Final Predictors or Subsets
        - ANN and Final Predictors or Subsets
        - RNN and Final Predictors or Subsets
    + Output Ensembles and Predictions
    + Validation
        - Global Performance
        - Site-means
        - Monthly Seasonal Cycles
        - Monthly Anomalies
8. Variable Importances
    + Variable Importance Rankings
    + Variable Responses
    + Partial Dependency Plots
    + ShapR
9. Upscaled Model with Monte Carlo (MC)
    + Forcing Data
        - Mapping
        - Member Product Choices
        - Evaluate Product Divergence
    + Data Preparation
        - Extract Gridded Data for MC
        - Pre-Process FLUXNET-CH4 for MC
    + MC Simulations
    + MC ML Model Training
    + MC ML Model Validation
10. Upscaling
    + Prepare Member Forcing Data
    + Run on Computing Cluster
        - Output Grids and Sums

        + Product Evaluation
            - Unweighted Wetland Fluxes (nmol)
            - Weighted Sums and Uncertainties (Tg)
            - Independent Validation
    11. Data Representativeness
        + Prepare Gridded Data
        + Global Dissimilarity
        + Tower Constituency
        + Extrapolation Errors
            - MC ML Model Training - Dissimilarity Only

[**Link to Workflow Figure**](https://drive.google.com/drive/folders/1jiFyqzoxxMpdtRLCwxCtzKpfILRNIW5K)

$~$

1. Objective of Study

The goal of `FLUXNET-CH4 Upscaling` is to implement FLUXCOM-like ML approaches (e.g. [Jung et al. 2020](10.5194/bg-17-1343-2020)) to train a machine learning model using eddy covariance data that can predict wetland methane (CH4) fluxes globally. The predictions should be readily comparable to the Global Carbon Project (GCP) bottom-up process model ensembles that inform the Global Methane Budget ([Saunois et al. 2020](10.5194/essd-12-1561-2020)). Wetland fluxes specifically, rather than methane fluxes from all terrestrial ecosystems, are the predictive goal of this study because 1) most eddy covariance data available are in wetlands, with limited coverage across the multitude of upland ecosystems, 2) methane fluxes are highest and most variable in wetlands, and 3) comparable bottom-up process models predict wetland fluxes, then scale predictions to a global grid-cell using a prescribed (diagnostic runs) or model-derived (prognostic runs) wetland extent. In the last GCP Global Methane Budget ([Saunois et al. 2020](10.5194/essd-12-1561-2020)), diagnostic runs used the WAD2M product ([Zhang et al. in review](10.5194/essd-2020-262)). WAD2M includes coastal wetlands, however, we exclude coastal wetlands from the upscaling because they are salt-influenced and we do not have consistent salinity coverage, thus their inclusion is likely to bias flux estimates low even in non-coastal wetlands. The final model will be forced with available globally gridded data. Final product specifications are:

  - Monthly time-step
  - Historic reconstruction: ca. 2000 - 2018
  - 0.25-degree grid cell resolution (as is WAD2M)
  - Propagates training data uncertainties using Monte Carlo simulations
  - Considers sensitivity to global forcing data product choices

$~$

2. Input Data

FLUXNET-CH4 Data

The eddy covariance data used in this study are publicly available as part of the FLUXNET-CH4 community product V1.0 at [fluxnet.org](https://fluxnet.org/data/fluxnet-ch4-community-product/). The FLUXNET-CH4 synthesis activity is introduced in [Knox et al. 2019](10.1175/BAMS-D-18-0268.1) along with a detailed description of the eddy covariance post-processing steps including methane flux (FCH4) uncertainties. The first full (V1.0) dataset release (FLUXNET-CH4, hereafter) is described for 81 sites and is used in a wetland seasonality analysis in [Delwiche et al. in review](10.5194/essd-2020-307). **Data for this CH4 Upscaling Project were downloaded from [fluxnet.org](www.fluxnet.org) on Feb 22, 2021. [Download Manifest](https://docs.google.com/spreadsheets/d/1--_-XyBqsyiMIdc6JXqhilOOYFLeaY-UTMFhhI4Sd5M/edit#gid=0)**

Links:

  - [FLUXNET-CH4 Site (81) Metadata](https://docs.google.com/spreadsheets/d/1DN0huLs-vVM3g_XcF1hBQrTpkKaGhzuWwaacfbe4iCo/edit#gid=1384338468)
  - Permission was received via email on Feb 22, 2021, to use *Tier 2 Data Policy* sites in this study (SE-St1 and RU-Vrk; PI Thomas Friborg)
  - FLUXNET-CH4 data were used both for methane fluxes (FCH4; target variable) and tower-measured bio-meteorological variables (e.g., LE, GPP, TA; predictors)
      + Link to [FLUXNET.org variable descriptions](https://fluxnet.org/data/fluxnet-ch4-community-product/data-variables/)

$~$

Gridded Data (Predictors)

A summary of predictors (`Predictor Summary`) is available [here](https://docs.google.com/spreadsheets/d/1DN0huLs-vVM3g_XcF1hBQrTpkKaGhzuWwaacfbe4iCo/edit#gid=0), as well as an appendix table with all individual predictors (`Appendix: All Predictors`). Candidate predictors were drawn from a mix of source classes (EC tower measurements, global models, computations from observed data, or remote sensing (e.g. NASA MODIS)), different information content groups (Spatial-only or Spatio-temporal), and at different temporal frequencies (Static, Yearly, Monthly, Weekly, or Half-hourly). Using this information, we assigned each candidate predictor to a class (Generic, Climate, Biometeorological, Land Cover, Soil and Relief, or Greenness) to evaluate predictive power of particular classes, given the likelihood for redundancy in useful predictors across the full predictor set. For MODIS predictors, we computed the mean seasonal cycle (`_msc`), and yearly mean, min, max, and amplitude parameters, as in [Tramontana et al. (2016)](10.5194/bg-13-4291-2016) and [Jung et al. (2020)](10.5194/bg-17-1343-2020). For derived TerraClimate soil moisture and actual evapotranspiration (aet) we computed the interannual range and annual seasonality. More information on preprocessing of predictors is provided in the manuscript.

$~$

3. FLUXNET-CH4 Pre-Processing

Local machine steps:

  + Create a copy of FLUXNET-CH4 data and name it `/fluxnet-ch4-data-original`
  + Unzip all site flux files in `/fluxnet-ch4-data`
  + Reorganize into half-hourly (`/hh`) and `/daily` folders for easy access

Look at one site of **daily means** data.

```{r}
# setwd(loc)
# files <- list.files("fluxnet-ch4-data/daily/")
# one_site <- read_csv(paste(loc, "fluxnet-ch4-data/daily/", files[1], sep = ""))
# head(one_site)
```

No, but there is a quality flag `_QC`. `1`= data gap shorter than 2 months, `3` = gap exceeds 2 months.

Issue: **There is also no uncertainty (`_UNC`) estimate on the downloaded daily mean data.**

Clean up workspace:

```{r}
# rm(one_site)
```

Look at one site of **half-hourly** data.

```{r}
setwd(loc)
files <- list.files("fluxnet-ch4-data/hh/")
one_site <- read_csv(paste(loc, "fluxnet-ch4-data/hh/", files[1], sep = ""))
head(one_site)
```

  - Half-hourly data has `FCH4` uncertainty columns `FCH4_F_RANDUNC` and `FCH4_F_ANNOPTLM_UNC`. Can be averaged over day or week.
  - Missing values are filled with `-9999`

Get FLUXNET-CH4 site names:

```{r}
site.names <- paste( substr(files, 5, 6), substr(files, 8, 10), sep = "")
```

Get all **half-hourly** flux data: (this will take a few minutes)

```{r echo = F, message = F, warning = F}
hh <- lapply(paste(loc, "fluxnet-ch4-data/hh/", files, sep = ""), read_csv)
names(hh) <- site.names
```

str(hh)
```

##### Create a pristine replicate:

```{r}
hh2 <- hh
```

Look at all column names (including TIMESTAMP columns, to see date
and time format):

```{r}
names(hh[[1]])
head(hh[[1]]$TIMESTAMP_END)
```

Write and apply function to expand `TIMESTAMP_END` into`Year`,
`Month`, `Week`, `Day`, and `DOY` variables to facilitate merging with
other data:

```{r}
expand_date <- function(hh_data) {
  hh_data <- hh_data %>%
  mutate(Year = as.numeric(substr(TIMESTAMP_END, 1, 4)),
         Month = as.numeric(substr(TIMESTAMP_END, 5, 6)),
         Day = as.numeric(substr(TIMESTAMP_END, 7, 8)),
         Date = make_date(Year, Month, Day),
         DOY = yday(Date),
         Week = ceiling(DOY/7),
         Week = ifelse(Week == 53, 52, Week)) %>%
    group_by(Year, DOY) %>%
    mutate(HH = 1:n()) %>%
  dplyr::select(Year, Month, Week, Day, HH, DOY, everything(), - Date, -
TIMESTAMP_START, -TIMESTAMP_END)
}

hh <- lapply(hh, expand_date)
hh[[2]] # check it worked
```

Check if gap-filled FCH4 `CH4_F_ANNOPTLM` has already been pre-filled
with observations `FCH4`, where available.

```{r}
hh[[1]]
```

FCH4 in row 7 in `hh[[1]]` (ID BRNpw) = 10.71, which **matches**
`CH4_F_ANNOPTLM` = 1.071e+01.
`CH4_F_ANNOPTLM` has been pre-filled with observations.

Write and apply function to create `imputed`, a flag variable where:
```

  - `1` == `CH4_F_ANNOPTLM` was imputed
  - `2` == `CH4_F_ANNOPTLM` was observed

```{r}
create_imputed <- function(hh_data) {
  hh_data <- hh_data %>%
    mutate(imputed = ifelse(FCH4 == -9999, 1, 0))
}

hh <- lapply(hh, create_imputed)
# hh[[1]] check it worked
```

Create ID column:

```{r}
for (i in 1:length(hh)){
  hh[[i]] <- hh[[i]] %>%
    mutate(ID = site.names[i]) %>%
    dplyr::select(ID, everything())
}
head(hh[[1]])
```

Convert missing values (`-9999`) to `NA`:

```{r}
for (i in 1:length(hh)){
  hh[[i]][hh[[i]] == -9999] <- NA
}
head(hh[[1]])
```

Get `u` and `v` wind components:

Identify which sites lack WD:

```{r}
find_wd <- function(hh_data) {
  sum(names(hh_data) == "WD") == 0
}

unlist(lapply(hh, find_wd))
sum(unname(unlist(lapply(hh, find_wd))))
```

OK, `CASCB` and `RUVrk` and missing `WD`.

Create dummy `WD` variables:

```{r}
hh$CASCB$WD <- NA
hh$RUVrk$WD <- NA

unlist(lapply(hh, find_wd))
sum(unname(unlist(lapply(hh, find_wd))))
```

Now compute wind direction components:

```{r}
compute_uv <- function(hh_data) {
  hh_data <- hh_data %>%
    mutate(U = -WS_F * sin(2 * pi * WD/360),
           V = -WS_F * cos(2 * pi * WD/360))
}

hh <- lapply(hh, compute_uv)

head(hh[[1]]) # check it worked
```

Compute daily means of everything, including vector average WD and speed, excluding precip. (sums)

Create function to compute daily means:

```{r echo = F}
compute_daily1 <- function(hh_data) {
  hh_data <- hh_data %>%
    group_by(ID, Year, DOY) %>%
    summarize_all(list(~mean(., na.rm = T))) %>%
    mutate(WD_mean = (atan2(U, V) * 360/2/pi) + 180,
           WS_mean = ((U^2 + V^2)^0.5)) %>%
    dplyr::select(ID, Year, Month, Week, Day, DOY, everything(), -HH)
}
```

Check run time for one site:

```{r warning = F}
system.time(
 compute_daily1(hh[[1]])
)
```

Apply to all sites using lapply:

```{r warning = F}
```

daily <- lapply(hh, compute_daily1)
```

Get sum of precip.

```{r warning = F, message = F}
sum_precip <- function(hh_data) {
  hh_data <- hh_data %>%
    group_by(ID, Year, DOY) %>%
    summarize(P_F_sum = sum(P_F))
}

daily_precip <- lapply(hh, sum_precip)
```

Rejoin precip. to main data frame:

```{r}
for (i in 1:length(daily)) {
daily[[i]] <- daily[[i]] %>%
  left_join(daily_precip[[i]], by = c("ID", "Year", "DOY"))
}
names(daily[[1]])
```

Output each "complete" daily site as a separate .csv file:

```{r}
for (i in 1:length(daily)) {
  daily[[i]] %>% write.csv(paste(loc, "fluxnet-ch4-data/daily_upch4/",
site.names[i], "_daily_upch4.csv", sep = ""),
                           row.names=FALSE)
}
```

3. FLUXNET-CH4 Pre-Processing (cont.)
(re)Load Daily Data

```{r message = F, warning = F, echo = F}
files <- list.files(paste(loc, "fluxnet-ch4-data/daily_upch4/", sep = ""))
daily <- lapply(paste(loc,"fluxnet-ch4-data/daily_upch4/",files, sep = ""),
read_csv)
```

Bind rows

```{r}
daily_flat <- daily %>% bind_rows() %>% as_tibble()
head(daily_flat)
```

Compute FCH4 uncertainty, and subset gap-filled (and best) predictors

Notes on variable selection:
```

+ Total FCH4 uncertainty `FCH4_F_UNC` is random `FCH4_F_RANDUNC` and gap-filling `FCH4_F_ANNOPTLM_UNC` uncertainty summed in quadrature. I also drop the `FCH4_F_ANNOPTLM_QC` Quality control flag because I implement a stricter filtering criteria later of at least one observed flux per day.
+ The output of the daytime method `_DT` is used for gross primary production (GPP) and ecosystem respiration (RECO). Although both methods are subject to bias due to light-inhibition of leaf respiration, [Keenan et al. 2019](https://doi.org/10.1038/s41559-019-0809-2) show that the biases for the DT method only impact night-time respiration, but did not impact apparent photosynthesis or daytime respiration.
+ LE and NEE were gap-filled according to [Knox et al. 2019](10.1175/BAMS-D-18-0268.1) - same overall method to FCH4.
+ Other micro-meteorology (e.g., PPFD_IN) was gap-filled using ERA interim data (see [Delwiche et al.](10.5194/essd-2020-307)).
+ Only PPFD_IN (not PPFD_OUT) was selected as it can be approximated from SW_IN.
+ The shallowest available soil temperature is taken (`TS_1`)

```{r}
daily_subset <- daily_flat %>%
  mutate(FCH4_F_UNC = sqrt(FCH4_F_ANNOPTLM_UNC^2 + FCH4_F_RANDUNC^2)) %>%
  dplyr::select(ID, Year, Month, Week, Day, DOY,    # descriptive data
         FCH4 = FCH4_F_ANNOPTLM, FCH4_F_UNC, imputed,   # methane fluxes
         NEE = NEE_F_ANNOPTLM,  GPP = GPP_DT, RECO = RECO_DT,  # ecosystem C fluxes
         PPFD_IN = PPFD_IN_F, SW_IN = SW_IN_F, LW_IN = LW_IN_F, NETRAD = NETRAD_F,   # radiation
         LE = LE_F_ANNOPTLM, H = H_F,  # ecosystem energy fluxes
         TA = TA_F, PA = PA_F, RH = RH_F, VPD = VPD_F, P = P_F_sum, USTAR = USTAR, WS = WS_mean, # meteorology
         TS = TS_1, SWC = SWC_F, WTD = WTD_F) # soil properties
head(daily_subset)
```

Append site metadata

Pulled from [FLUXNET-CH4 Site Metadata](https://docs.google.com/spreadsheets/d/1DN0huLs-vVM3g_XcF1hBQrTpkKaGhzuWwaacfbe4iCo/edit#gid=1384338468)

```{r}
metadata <- read_csv(paste(loc, "fluxnet-ch4-data/metadata/fluxnet-ch4-site-metadata.csv", sep = ""))
daily_subset_meta <- metadata %>%
  mutate(ID = paste(substr(ID,1,2),substr(ID,4,6), sep="")) %>%
  right_join(daily_subset, by = ("ID"))
```
Save subset, flattened, and metadata-appended daily FLUXNET-CH4 data

```{r}
write.csv(daily_subset_meta,
          paste(loc, "fluxnet-ch4-data/daily_flat/daily_subset_meta.csv", sep = ""),
```

          row.names = F)
```

#### 4. Gridded Data Pre-Processing

#### *Potential Radiation (RPot)*

**Description**
Get the mean seasonal cycle from Zutao Ouyang's 2001-2018 MATLAB output.
Original citation for `Rpot` is [Peltola et al. 2019](10.5194/essd-11-1263-2019) where it was found to be a useful predictor of high latitude wetland FCH4.

Define a function to take a number of years and return an index of months:

```{r}
create_msc_index <- function(years) {
  msc_index <- list()
  for (i in 1:12){
      msc_index[[i]] <- seq(i, ((years-1)*12+i), by = 12)
  }
  msc_index
}
# create_msc_index(15) # test it out
```

Load in Rpot and create msc index for 19 years.

```{r}
Rpot <- brick(paste(loc, "grids/computed/Rpot.nc", sep = ""))
msc_index <- create_msc_index(19)
```

Get `_msc`:

```{r}
Rpot_msc <- list()
for (i in 1:12) {
  Rpot_msc[[i]] <- calc(Rpot[[msc_index[[i]]]], fun = mean)
}
```

Create raster stack:

```{r}
Rpot_msc <- stack(Rpot_msc)
```

Output `Rpot_msc.nc`:

```{r}
# writeRaster(Rpot_msc, paste(loc, "grids/computed/Rpot_msc.nc", sep = ""),
format="CDF", overwrite = T)
```
```

MODIS

Description
Zutao Ouyang (Stanford University) extracted MODIS pixels at the FLUXNET-CH4 sites in April 2020, for 9 products:

  + Daytime Land Surface Temperature (`LSTD` from **MOD11A2**)
      - *(not used as it is a correlate of nighttime temp., and nighttime is more applicable to soil conditions)*
  + Nighttime Land Surface Temperature (`LSTN` from **MOD11A2**)
  + Normalized Difference Vegetation Index (`NDVI` from **MOD09A1**)
  + Enhanced Vegetation Index (`EVI` from **MOD09A1**)
  + Leaf Area Index (`LAI` from **MCD15A2H**)
      - This is modeled, not directly measured.
  + Long Short Water Index (`LSWI` from **MOD09A1**)
  + Simple Ratio Water Index (`SRWI` from **MOD09A1**)
  + Normalized Difference Water Index (`NDWI` from **MOD09A1**)
  + Normalized Difference Snow Index (`NDSI` from **MOD11A2**)

NOTE Tables docmenting MODIS processing step effects on data are here under [MODIS Processing](https://docs.google.com/spreadsheets/d/1DN0huLs-vVM3g_XcF1hBQrTpkKaGhzuWwaacfbe4iCo/edit#gid=1971246167) and QC figures are output to `upch4_local/modis/modis_qc`.

Get file local file names/paths for 8-day extracted MODIS data:

```{r}
setwd(paste(loc, "/modis/modis-extracted", sep = ""))
files <- list.files()
```

Set MODIS product/file names (for gather values):

```{r}
modis.names <- c("LSTD", "EVI", "LAI", "LSWI", "NDSI", "NDVI", "NDWI",
"LSTN", "SRWI")
```

Read files:

```{r echo = F, warning = F, message = F}
modis <- lapply(paste(loc, "/modis/modis-extracted/", files, sep = ""),
read_csv)
names(modis) <- modis.names
str(modis)
```

Look at head for first file:

```{r}
head(modis[[1]])
```

Files are not tidy (short and wide). There are **86** extracted sites (extra sites than in FLUXNET-CH4 V1.0).

Use gather to make long and narrow, remove hyphen from ID:

```{r}
for (i in 1:length(files)) {
  modis[[i]] <- modis[[i]] %>%
    gather(key = ID, value = "modis.name", 2:87) %>%
    mutate(ID = paste(substr(ID, 1, 2), substr(ID, 4, 6), sep = "")) %>%
    as_tibble()
}
names(modis) <- modis.names
```

Check data are same length:

```{r}
str(modis)
```

1) `NDSI` is longer, 2) `LAI` is slightly shorter. Based on the dates, `NDSI` is daily.

Will not be able to bind columns with different lengths

Remove `LAI` and `NDSI` then rejoin:

```{r}
modis.1.names <- c("LSTD", "EVI", "LSWI", "NDVI", "NDWI", "LSTN", "SRWI")
modis.1 <- modis[modis.1.names]
```

Select only first two columns (Date and ID) then data columns:

```{r}
modis.1 <- modis.1 %>%
  bind_cols() %>%
  dplyr::select(1, 2, 3,6,9,12,15,18,21) %>% # subset only data columns
  as_tibble()
names(modis.1) <- c("Date", "ID", modis.1.names)
```

Rejoin `LAI` using `Date`:

```{r}
modis.2 <- modis.1 %>% left_join(modis$LAI)
names(modis.2)[10] <- "LAI"
```

Rejoin `NDSI`:

```{r}
modis <- modis.2 %>% right_join(modis$NDSI)
```

names(modis)[11] <- "NDSI"
```

##### Convert `Date` into `Year` and `DOY`, correct `MARC` name (should be `US-MAC`), remove `LSTD`:

```{r}
modis <- modis %>%
  mutate(Date = as.Date(Date, format = "%m/%d/%Y")) %>%
  mutate(ID = ifelse(ID == "MARC", "USMAC",ID),
         DOY = yday(Date),
         Date = as_date(Date),
         Week = ceiling(DOY/7),
         Week = ifelse(Week == 53, 52, Week),
         Week = as.factor(Week),
         Year = as.integer(substr(Date, 1,4 )),
         Month = as.numeric(substr(Date, 6, 7))) %>%
  dplyr::select(ID, Date, Year, Month, Week, DOY, NDSI, NDVI, EVI, LAI,
NDWI, SRWI, LSWI, LSTN)
```

Count `NAs` before despiking:

```{r}
modis_nas <- modis %>%
  group_by(ID) %>%
  summarize_all(list(~sum(is.na(.))))
modis_nas
```
Despike outlier values for each product:

```{r}
modis_dsp <- modis %>%
  mutate(NDSI_dsp = despike(NDSI, reference = 'trim', min = 0, max = 100,
replace = "NA"),
         NDVI_dsp = despike(NDVI, reference = 'trim', min = -1, max = 1,
replace = "NA"),
         EVI_dsp = despike(EVI, reference = 'trim', min = -1, max = 1,
replace = "NA"),
         LAI_dsp = despike(LAI, reference = 'trim', min = 0, max = 5,
replace = "NA"),
         NDWI_dsp = despike(NDWI, reference = 'trim', min = 0, max = 1,
replace = "NA"),
         SRWI_dsp = despike(SRWI, reference = 'trim', min = -1, max = 3,
replace = "NA"),
         LSWI_dsp = despike(LSWI, reference = 'trim', min = 0, max = 1,
replace = "NA"),
         LSTN_dsp = despike(LSTN, reference = 'trim', min = -60, max = 50,
replace = "NA")) %>%
dplyr::select(ID, Date, Year, Month, Week, DOY, NDSI, NDVI, EVI, LAI, NDWI,
SRWI, LSWI, LSTN,
              NDSI_dsp, NDVI_dsp, EVI_dsp, LAI_dsp, NDWI_dsp, SRWI_dsp,
LSWI_dsp, LSTN_dsp)
```
```

Count `NAs` after despiking:

```{r}
modis_dsp_nas <- modis_dsp %>%
  group_by(ID) %>%
  summarize_all(list(~sum(is.na(.))))
modis_dsp_nas
```
Calculate removed values during despiking and output to `upch4_local/modis/modis_qc/`

```{r}
modis_dsp_nas %>%
  mutate(NDSI = NDSI - NDSI_dsp,
         NDVI = NDVI - NDVI_dsp,
         EVI = EVI - EVI_dsp,
         LAI = LAI - LAI_dsp,
         NDWI = NDWI - NDWI_dsp,
         SRWI = SRWI - SRWI_dsp,
         LSWI = LSWI - LSWI_dsp,
         LSTN = LSTN - LSTN_dsp) %>%
  dplyr::select(-ID) %>%
  summarize_all(list(~sum(.))) %>%
  write.csv(paste(loc, "/modis/modis-qc/despiking_na_effects.csv", sep =
""))
```


Rename without `_dsp` suffix:

```{r}
modis_dsp <- modis_dsp %>%
  dplyr::select(ID, Date, Year, Month, Week, DOY,
                NDSI = NDSI_dsp,
                NDVI = NDVI_dsp,
                EVI = EVI_dsp,
                LAI = LAI_dsp,
                NDWI = NDWI_dsp,
                SRWI = SRWI_dsp,
                LSWI = LSWI_dsp,
                LSTN = LSTN_dsp)
```


Calculate mean seasonal cycle (msc), then weekly means, then fill weekly gaps with msc:

```{r}
modis_gapfilled <- modis_dsp %>%
  group_by(ID, Month) %>%  # this section computes  monthly values averaged
over multiple years
  mutate(NDSI_msc = mean(NDSI, na.rm = TRUE),
         NDVI_msc = mean(NDVI, na.rm = TRUE),
```

```
         EVI_msc = mean(EVI, na.rm = TRUE),
         LAI_msc = mean(LAI, na.rm = TRUE),
         NDWI_msc = mean(NDWI, na.rm = TRUE),
         SRWI_msc = mean(SRWI, na.rm = TRUE),
         LSWI_msc = mean(LSWI, na.rm = TRUE),
         LSTN_msc = mean(LSTN, na.rm =TRUE)) %>%
  group_by(ID, Year, Month, Week) %>% # this section compute weekly means
  summarize(NDSI = mean(NDSI, na.rm = TRUE),
            NDVI = mean(NDVI, na.rm = TRUE),
            EVI = mean(EVI, na.rm = TRUE),
            LAI = mean(LAI, na.rm = TRUE),
            NDWI = mean(NDWI, na.rm = TRUE),
            SRWI = mean(SRWI, na.rm = TRUE),
            LSWI = mean(LSWI, na.rm = TRUE),
            LSTN = mean(LSTN, na.rm = TRUE),
            NDSI_msc = NDSI_msc[1],    # this section selects the monthly
(msc) value corresponding to the weekly data
            NDVI_msc = NDVI_msc[1],
            EVI_msc = EVI_msc[1],
            LAI_msc = LAI_msc[1],
            NDWI_msc = NDWI_msc[1],
            SRWI_msc = SRWI_msc[1],
            LSWI_msc = LSWI_msc[1],
            LSTN_msc = LSTN_msc[1]) %>%
  mutate(NDSI_F = ifelse(is.na(NDSI), NDSI_msc, NDSI), # this section fills
any missing weekly values with the msc (monthly average)
         NDVI_F = ifelse(is.na(NDVI), NDVI_msc, NDVI),
         EVI_F = ifelse(is.na(EVI), EVI_msc, EVI),
         LAI_F = ifelse(is.na(LAI), LAI_msc, LAI),
         NDWI_F = ifelse(is.na(NDWI), NDWI_msc, NDWI),
         SRWI_F = ifelse(is.na(SRWI), SRWI_msc, SRWI),
         LSWI_F = ifelse(is.na(LSWI), LSWI_msc, LSWI),
         LSTN_F = ifelse(is.na(LSTN), LSTN_msc, LSTN))
```

If snow is on the ground, set water indices to the 5% quantile (frozen):

```{r}
modis_frozen <- modis_gapfilled %>%
  group_by(ID,Year) %>%
  mutate(NDWI_msc = ifelse(NDSI_msc > 0, quantile(NDWI_msc, 0.05,
na.rm=TRUE), NDWI_msc),
         SRWI_msc = ifelse(NDSI_msc > 0, quantile(SRWI_msc, 0.05,
na.rm=TRUE), SRWI_msc),
         LSWI_msc = ifelse(NDSI_msc > 0, quantile(LSWI_msc, 0.05,
na.rm=TRUE), LSWI_msc),

         NDWI_F = ifelse(NDSI_F > 0, quantile(NDWI_F, 0.05, na.rm=TRUE),
NDWI_F),
         SRWI_F = ifelse(NDSI_F > 0, quantile(SRWI_F, 0.05, na.rm=TRUE),
SRWI_F),
```

         LSWI_F = ifelse(NDSI_F > 0, quantile(LSWI_F, 0.05, na.rm=TRUE),
LSWI_F))
```

##### Compute mean, min, max, amplitude:

```{r message = F, warning = F}
modis_frozen <- modis_frozen %>%
  group_by(ID, Year) %>%
  mutate(NDSI_mean = mean(NDSI_F, na.rm=TRUE),
         NDVI_mean = mean(NDVI_F, na.rm=TRUE),
         EVI_mean = mean(EVI_F, na.rm=TRUE),
         LAI_mean = mean(LAI_F, na.rm=TRUE),
         NDWI_mean = mean(NDWI_F, na.rm=TRUE),
         SRWI_mean = mean(SRWI_F, na.rm=TRUE),
         LSWI_mean = mean(LSWI_F, na.rm=TRUE),
         LSTN_mean = mean(LSTN_F, na.rm=TRUE),

         NDSI_min = min(NDSI_F, na.rm=TRUE),
         NDVI_min = min(NDVI_F, na.rm=TRUE),
         EVI_min = min(EVI_F, na.rm=TRUE),
         LAI_min = min(LAI_F, na.rm=TRUE),
         NDWI_min = min(NDWI_F, na.rm=TRUE),
         SRWI_min = min(SRWI_F, na.rm=TRUE),
         LSWI_min = min(LSWI_F, na.rm=TRUE),
         LSTN_min = min(LSTN_F, na.rm=TRUE),

         NDSI_max = max(NDSI_F, na.rm=TRUE),
         NDVI_max = max(NDVI_F, na.rm=TRUE),
         EVI_max = max(EVI_F, na.rm=TRUE),
         LAI_max = max(LAI_F, na.rm=TRUE),
         NDWI_max = max(NDWI_F, na.rm=TRUE),
         SRWI_max = max(SRWI_F, na.rm=TRUE),
         LSWI_max = max(LSWI_F, na.rm=TRUE),
         LSTN_max = max(LSTN_F, na.rm=TRUE),

         NDSI_amp = NDSI_max-NDSI_min,
         NDVI_amp = NDVI_max-NDVI_min,
         EVI_amp = EVI_max-EVI_min,
         LAI_amp = LAI_max-LAI_min,
         NDWI_amp = NDWI_max-NDWI_min,
         SRWI_amp = SRWI_max-SRWI_min,
         LSWI_amp = LSWI_max-LSWI_min,
         LSTN_amp = LSTN_max-LSTN_min)
```

Get site IDs:

```{r}
site.names <- modis_frozen %>%
  ungroup() %>%
  mutate(ID = factor(ID)) %>%
  dplyr::select(ID) %>%
  pull() %>% unique()

````
```

Get variable names:

```{r}
names <- names(modis)[7:14]
names_F <- paste(names, "_F", sep = "")
names_msc <- paste(names, "_msc", sep = "")
```

Create modis qcqa figures:

```{r message = F, warning = F}
for (i in 1:length(names)){

  # visualize 1-45
  modis_frozen %>%
    filter(ID %in% site.names[1:45]) %>%
    dplyr::select(Month, value_F = names_F[i], value_msc = names_msc[i])
%>%
    ggplot(aes(Month, value_F)) +
    geom_point(size = 1, alpha = 0.3) +
    geom_line(aes(Month, value_msc), col = 'purple', size = 2) +
    facet_wrap(~ID, scales = 'free', ncol = 9) +
    my_theme
  ggsave(paste(loc, "/modis/modis-qc/", names[i], "_1.pdf", sep = ""),
         width = 50, height = 30, units = c("cm"), dpi = 300)

  # visualize 46-86
  modis_frozen %>%
    filter(ID %in% site.names[46:86]) %>%
    dplyr::select(Month, value_F = names_F[i], value_msc = names_msc[i])
%>%
    ggplot(aes(Month, value_F)) +
    geom_point(size = 1, alpha = 0.3) +
    geom_line(aes(Month, value_msc), col = 'purple', size = 2) +
    facet_wrap(~ID, scales = 'free', ncol = 9) +
    my_theme
  ggsave(paste(loc, "/modis/modis-qc/", names[i], "_2.pdf", sep = ""),
         width = 50, height = 30, units = c("cm"), dpi = 300)

}
```

Output processed MODIS data:

```{r}
write.csv(modis_frozen, paste(loc, "/modis/modis-processed/modis-
processed.csv", sep = ""),
          row.names = FALSE)
```

Grid Extraction
````
````

````
**Extract geospatial data at FLUXNET-CH4 sites and output a single
geospatial .csv.**

##### Load site metadata (ID, Latitude, Longitude), applicable to all
extractions:

```{r}
sites <- read_csv(paste(loc, "fluxnet-ch4-data/metadata/fluxnet-ch4-site-
metadata.csv", sep = "")) %>%
  mutate(ID = paste(substr(ID, 1, 2), substr(ID, 4, 6), sep = ""))
site.coords <- cbind(sites$Longitude, sites$Latitude) # (x, y)
site.num <- length(site.coords[,1])
```

##### Global Canopy Height

```{r}
raster.names <- list.files(paste(loc, "grids/global-canopy-height/", sep =
""), pattern = "tif$", full.names = FALSE)
stack <- stack(paste(loc, "grids/global-canopy-height/", raster.names, sep
= ""))
canopyht <- as_tibble(raster::extract(stack, site.coords))
canopyht <- cbind(sites, canopyht) %>%
  rename(canopyht = Simard_Pinto_3DGlobalVeg_JGR)
head(canopyht)
```

##### Computed (Rpot)

```{r}
stack <- stack(paste(loc, "grids/computed/Rpot_msc.nc", sep = ""))
Rpot <- as_tibble(raster::extract(stack, site.coords))
Rpot <- cbind(sites, Rpot) %>%
  gather(key = "Month", value = "Rpot", 15:26) %>%
  mutate(Month = str_remove(Month, "X"))
head(Rpot)
```

##### Compound Topographic Index

```{r}
raster.names <- list.files(paste(loc, "grids/geomorpho90m/", sep = ""),
pattern = "tif$", full.names = FALSE)
stack <- stack(paste(loc, "grids/geomorpho90m/", raster.names, sep = ""))
cti <- as_tibble(raster::extract(stack, site.coords))
cti <- cbind(sites, cti) %>%
  rename(cti = dtm_cti_merit.dem_m_250m_s0cm_2018_v1.0)
head(cti)
```

##### Earth Environment Texture, Land Cover and Topography

Texture (14 variables)
Land Cover (2 variables)
````

Topography (10 variables)
See: [Appendix: All Predictors](https://docs.google.com/spreadsheets/d/1DN0huLs-vVM3g_XcF1hBQrTpkKaGhzuWwaacfbe4iCo/edit#gid=0). Search for `EarthEnv` under column `Gridded Product`.

First do texture:

```{r}
raster.names <- list.files(paste(loc, "grids/earthenv/texture/", sep = ""),
pattern = "tif$", full.names = FALSE)
stack <- stack(paste(loc, "grids/earthenv/texture/", raster.names, sep =
""))
earthenv_texture <- as_tibble(raster::extract(stack, site.coords))
names(earthenv_texture) <-
c("cont","corr","cv","diss","entr","even","homo","max","rang","shan","simp","std","unif"
```

Then land cover (commented code for first-run creates `HD` - human development - as the sum of LC7 and LC9):

```{r}
raster.names <- list.files(paste(loc, "grids/earthenv/cover/", sep = ""),
pattern = "tif$", full.names = FALSE)
stack <- stack(paste(loc, "grids/earthenv/cover/", raster.names, sep = ""))

# # create `HD` (human development)
# HD <- stack[[1]] + stack[[2]]
# writeRaster(HD, paste(loc, "/grids/earthenv/cover/HD.tiff", overwrite =
T))

earthenv_cover <- as_tibble(raster::extract(stack, site.coords))
names(earthenv_cover) <- c("HD7", "HD9", "HD")
```

Then Topography:

```{r}
raster.names <- list.files(paste(loc, "grids/earthenv/topography/", sep =
""), pattern = "tif$", full.names = FALSE)
stack <- stack(paste(loc, "grids/earthenv/topography/", raster.names, sep =
""))
earthenv_topography <- as_tibble(raster::extract(stack, site.coords))
names(earthenv_topography) <-
c("east","elev","flat","hollow","north","pcurv","rough","slope","tcurv","tpi")
```

Now combine:

```{r}
earthenv <- cbind(sites, earthenv_texture, earthenv_cover,
earthenv_topography)
```

Look at data histograms:

```{r}
earthenv %>%
  gather(key = "var", value = "value", 15:41) %>%
  ggplot(aes(value)) +
  geom_histogram() +
  facet_wrap(~var, scales = 'free')
```

`cont`, `diss`, and `var` have extreme outliers (xmax = 10e+09)

Replace extreme outliers with median

```{r}
earthenv <- earthenv %>%
  gather(key = "var", value = "value", 15:41) %>%
  group_by(var) %>%
  mutate(value = ifelse(value > 10^9, median(value, na.rm = TRUE), value))
%>%
  ungroup() %>%
  spread(key = "var", value = "value")
```

Visualize again above to check they are corrected.

Also check for missing data:

```{r}
earthenv %>%
  gather(key = "var", value = "value", 15:41) %>%
  mutate(missing = is.na(value)) %>%
  group_by(ID) %>%
  summarize(total = n(),
            missing = sum(missing)) %>%
  arrange(desc(missing))
```

There are two sites (USDPW and USWPT) with missing values. Plot the sites:

```{r}
plot(stack[[1]])
points(site.coords[sites$ID %in% c("USDPW", "USWPT"), ])
```

Fill with nearby site values.

USDPW from USLA1

USWPT from USOWC

```{r}
```

```
earthenv[earthenv$ID == "USDPW", c(is.na(earthenv[earthenv$ID == "USDPW",
])) ] <- earthenv[earthenv$ID == "USLA1", c(is.na(earthenv[earthenv$ID ==
"USDPW", ])) ]
earthenv[earthenv$ID == "USWPT", c(is.na(earthenv[earthenv$ID == "USWPT",
])) ] <- earthenv[earthenv$ID == "USOWC", c(is.na(earthenv[earthenv$ID ==
"USWPT", ])) ]
```

N and S Deposition

From [Lamarque et al. 2013](10.5194/acp-13-7997-2013):

  - accmip_nhx_acchist_2000.nc
  - accmip_noy_acchist_2000.nc
  - accmip_sox_acchist_2000.nc

```{r}
raster.names <- list.files(paste(loc, "grids/ns-deposition/", sep = ""),
pattern = "nc$", full.names = FALSE)
stack <- stack(paste(loc, "grids/ns-deposition/", raster.names, sep = ""))
plot(stack[[3]])
ns_depo <- as_tibble(raster::extract(stack, site.coords))
ns_depo <- cbind(sites, ns_depo) %>%
  rename(nx_depo = Dry.deposition.NH3.NH4,
         ny_dep = Dry.deposition.NOy,
         s_dep = Dry.deposition.SO2.SO4)
head(ns_depo)
```

SoilGrids
TerraClimate
Fractional Vegetation Cover (VCF)
Wetland Extent (WAD2M)
WorldClim 2.0