vignette

Ethan Carlson, Xieqing Yu

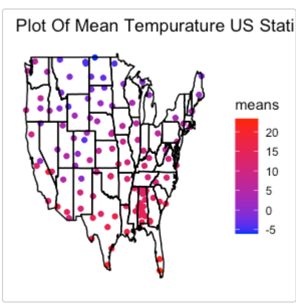
```r
library(MrSun)
```

##Loading Data

```r
# Load datafiles
merging <- merging
stations_and_names <- unique_stations
head(merging)
#   WBANNO state   station_name   LST_DATE CRX_VN LONGITUDE LATITUDE T_DAILY_MAX
# 1  53878    NC Asheville_13_S 2000-11-14     NA    -82.56    35.42          NA
# 2  53878    NC Asheville_13_S 2000-11-15     NA    -82.56    35.42          NA
# 3  53878    NC Asheville_13_S 2000-11-16     NA    -82.56    35.42          NA
# 4  53878    NC Asheville_13_S 2000-11-17     NA    -82.56    35.42          NA
# 5  53878    NC Asheville_13_S 2000-11-18     NA    -82.56    35.42          NA
# 6  53878    NC Asheville_13_S 2000-11-19     NA    -82.56    35.42          NA
#   T_DAILY_MIN T_DAILY_MEAN T_DAILY_AVG P_DAILY_CALC SOLARAD_DAILY
# 1          NA           NA          NA           NA            NA
# 2          NA           NA          NA           NA            NA
# 3          NA           NA          NA           NA            NA
# 4          NA           NA          NA           NA            NA
# 5          NA           NA          NA           NA            NA
# 6          NA           NA          NA           NA            NA
head(stations_and_names)
#     WBANNO      station_name state LONGITUDE LATITUDE
# 1    53878    Asheville_13_S    NC    -82.56    35.42
# 49   53877   Asheville_8_SSW    NC    -82.61    35.49
# 97   94060 Wolf_Point_29_ENE    MT   -105.10    48.31
# 109  94059  Wolf_Point_34_NE    MT   -105.21    48.49
# 851  54794        Durham_2_N    NH    -70.93    43.17
# 867  54795      Durham_2_SSW    NH    -70.95    43.11
```

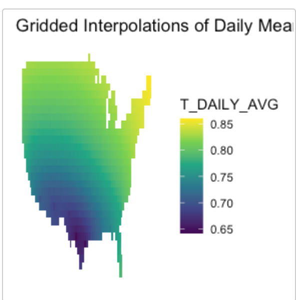
##Problem 1: Making a map of the average temperature at each station for the month of March 2024

```r
library(dplyr)
#
# Attaching package: 'dplyr'
# The following objects are masked from 'package:stats':
#
#     filter, lag
# The following objects are masked from 'package:base':
#
#     intersect, setdiff, setequal, union

# Extract all recorded data from March 2024
march_24 <- time.extraction(data = merging, start = "2024-03-01",
                            end = "2024-03-31")

avg_temps <- march_24 |>
  group_by(WBANNO, LONGITUDE, LATITUDE) |>
  summarise(means = mean(T_DAILY_AVG, na.rm = TRUE))
# `summarise()` has grouped output by 'WBANNO', 'LONGITUDE'. You can override
# using the `.groups` argument.

# Make March 2024 data only contain values within the contiguous US
usa_map <- ggplot2::map_data("usa") |> dplyr::filter(group==1)
inside <- sp::point.in.polygon(avg_temps$LONGITUDE, avg_temps$LATITUDE,
                               usa_map$long, usa_map$lat)
# The legacy packages maptools, rgdal, and rgeos, underpinning this package
# will retire shortly. Please refer to R-spatial evolution reports on
# https://r-spatial.org/r/2023/05/15/evolution4.html for details.
# This package is now running under evolution status 0
avg_temps <- avg_temps[as.logical(inside), ]

# Plot points onto a US map
library(ggplot2)
  ggplot() +
    geom_point(data = avg_temps, aes(x = LONGITUDE, y = LATITUDE,
                                     color = means)) +
    scale_color_gradient(low = "blue", high = "red") +
    labs(title = "Plot Of Mean Tempurature US Stations For March 2024") +
    theme_minimal() +
    geom_polygon(data = map_data("state"),
                 aes(x = long, y = lat, group = group),
                 fill = NA, color = "black") +
    theme(axis.text = element_blank(),
          axis.title = element_blank(),
          panel.grid = element_blank())
```



##Problem 2: Fiting a spatial model and plot an interpolated map of average temperatures for March 2024.

```r
# Create blank grid
blank_grid <- create_grid_points(resolution = 0.8)

# Interpolate daily average temperatures to blank grid from March 2024
grid_data <- interpolate_to_grid(grid = blank_grid, stations_data = march_24)
# Warning: package 'GpGp' was built under R version 4.3.1
#
# Attaching package: 'lubridate'
# The following objects are masked from 'package:base':
#
#     date, intersect, setdiff, union
# Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

# Display the interpolated values
plot_interpolations(grid_data)
```



##Problem 3: Estimating the warmest and coldest day of the year for each station.

In order to estimate the warmest and coldest temperatures for each station, we needed to estimate the yearly cycle for each station. That is, estimate the typical temperature cycle for each station based on all of the data provided from the data set of all recorded temperatures.

To do this, we created a function called `yearly_cycle()` which takes in said datasets and the id of the station of interest, and preforms the following:

Data Preparation For Each Station:

- Filter the dataset data to include only observations from the specified station_id.
- Select relevant columns `LST_DATE` (local standard time date) and `T_DAILY_AVG` (daily average temperature).
- Converts the LST_DATE to the day of the year, denoted as day_of_year. This is achieved by formatting the date and extracting the day of the year as a numeric value.

Model Fitting And Analysis:

- For a given station and data, we fit a linear regression model which will predicts `T_DAILY_AVG` from 0 to 365 days from January 1st (result of leap years) based on a sinusoidal function.
- We then predict the `T_DAILY_AVG` between the 1 to 365 days from Janaury 1st.
- We repeat this process for all station IDs.
- Finally, the minimum and maximum value is picked for the 365 predicted tempuratures for each station

Mathematical Notation Of Process:

`$$`

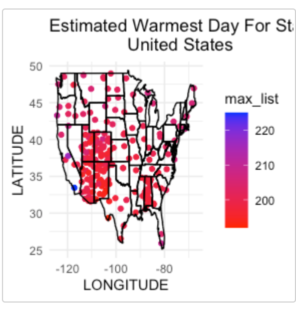
T_{DAILY}AVG_{min} = min(_0 + _1 sin() + _2 cos() +)

T_{DAILY}AVG_{max} = max(_0 + _1 sin() + _2 cos() +)

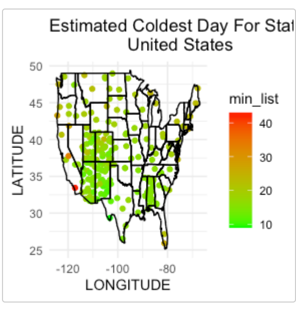
`$$` + Where: - T DAILY AVG: the daily average temperature for a given day of the year - β_0: the estimated intercept for the daily average temperature for a given day of the year (the day before January 1st). - β_1 and β_2: the coefficients estimated by the model. - ε is the error term.

```r
# Creating datasets
all_stations <- stations_and_names$WBANNO
temp_sta <- stations_and_names |> dplyr::select(WBANNO, LATITUDE, LONGITUDE)

n <- length(temp_sta$WBANNO)
temp_sta$min_list <- rep(NA, n)
temp_sta$max_list <- rep(NA, n)

for (x in 1:n) {
  y <- yearly_cycle(merging, all_stations[x])
  temp_sta$min_list[x] <- y$day_of_year[which.min(y$expected_temperature)]
  temp_sta$max_list[x] <- y$day_of_year[which.max(y$expected_temperature)]
}
usa_map <- ggplot2::map_data("usa") |> dplyr::filter(group==1)

inside <- sp::point.in.polygon(temp_sta$LONGITUDE, temp_sta$LATITUDE,
                               usa_map$long, usa_map$lat)

temp_sta <- temp_sta[as.logical(inside), ]

max_data <- temp_sta |> dplyr::select(-min_list)
min_data <- temp_sta |> dplyr::select(-max_list)
```

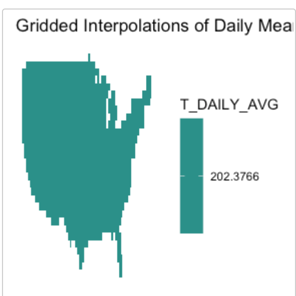
```r
# Plot exact points
library(ggplot2)

plot_max <- ggplot() +
  geom_point(data = temp_sta, aes(x = LONGITUDE, y = LATITUDE,
                                  color = max_list)) +
  scale_color_gradient(low = "red", high = "blue") +
  theme_minimal() +
  geom_polygon(data = map_data("state"), aes(x = long, y = lat, group = group),
               fill = NA, color = "black") +
  labs(title = "Estimated Warmest Day For Stations Inside The Contiguous
                United States")

plot_min <- ggplot() +
  geom_point(data = temp_sta, aes(x = LONGITUDE, y = LATITUDE,
                                  color = min_list)) +
  scale_color_gradient(low = "green", high = "red") +
  theme_minimal() +
  geom_polygon(data = map_data("state"), aes(x = long, y = lat, group = group),
               fill = NA, color = "black") +
  labs(title = "Estimated Coldest Day For Stations Inside The Contiguous
                United States")

print(plot_max)
```



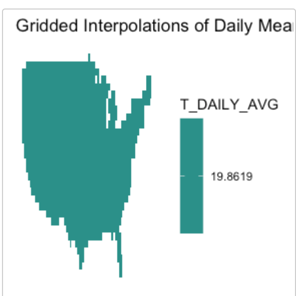

```r
print(plot_min)
```




```r
# Interpolate maps

max_data$T_DAILY_AVG <- max_data$max_list
min_data$T_DAILY_AVG <- min_data$min_list

blank_grid_2 <- create_grid_points(resolution = 0.8)

mmax <- interpolate_to_grid(blank_grid_2, stations_data = max_data)
# Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees
mmin <- interpolate_to_grid(blank_grid_2, stations_data = min_data)
# Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

plot_interpolations(mmax)
```

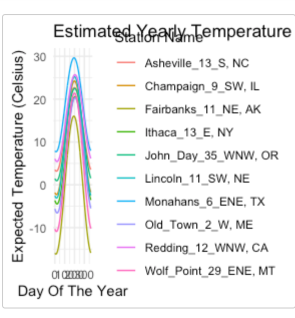



```r
plot_interpolations(mmin)
```



Problem 4: Estimateing yearly cycles for 10 different stations

```r
stations_10 <- c("Asheville_13_S", "Wolf_Point_29_ENE", "Fairbanks_11_NE",
                 "Old_Town_2_W", "Champaign_9_SW", "Lincoln_11_SW",
                 "John_Day_35_WNW", "Redding_12_WNW", "Monahans_6_ENE",
                 "Ithaca_13_E")
the_10 <- stations_and_names[stations_and_names$station_name %in% stations_10, ]
id_stations_10 <- unique(the_10[, c("WBANNO", "station_name", "state")])

result_list <- vector("list", length(id_stations_10$WBANNO))

for (x in seq_along(id_stations_10$WBANNO)) {
  y <- yearly_cycle(merging, id_stations_10$WBANNO[x])
  result_list[[x]] <- cbind(id_stations_10$station_name[x],
                            id_stations_10$state[x], y)
}

combined_df <- do.call(rbind, result_list)
combined_df$station <- paste(combined_df$`id_stations_10$station_name[x]`,
                             combined_df$`id_stations_10$state[x]`, sep = ", ")

# Create the plot
plot <- ggplot(combined_df, aes(x = day_of_year, y = expected_temperature,
                                color = station)) +
  geom_line(stat = "identity", position = "dodge") +
  labs(title = "Estimated Yearly Temperature Cycle By Station",
       color = "Station Name", x = "Day Of The Year",
       y = "Expected Temperature (Celsius)") +
  theme_minimal()

plot
# Warning: Width not defined
# i Set with `position_dodge(width = ...)`
```



##Problem 5: Estimating the trend over the years for each station.*

In order to estimate the warmest and coldest temperatures for each station, we needed to estimate the yearly cycle for each station. That is, estimate the typical temperature cycle for each station based on all of the data provided from the data set of all recorded temperatures.

To do this, we created a function called `estimate_temp_trend()` which takes in said datasets and the id of the station of interest, and preforms the following:

Data Preparation For Each Station:

- Filter the dataset data to include only observations from the specified station_id.
- Select relevant columns `LST_DATE` (local standard time date) and `T_DAILY_AVG` (daily average temperature).
- Converts the LST_DATE to the day of the year, denoted as day_of_year. This is achieved by formatting the date and extracting the day of the year as a numeric value.

Model Fitting And Analysis:

- For a given station and data, we fit a linear regression model which will predicts `T_DAILY_AVG` from 0 to 365 days from January 1st (result of leap years) based on a sinusoidal function.
- We then predict the `T_DAILY_AVG` between the 1 to 365 days from Janaury 1st.
- We repeat this process for all station IDs.

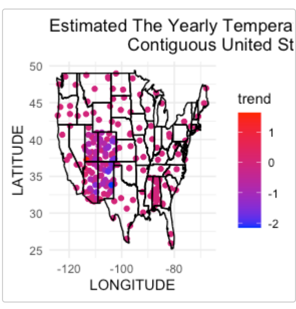
Mathematical Notation Of Process:

`$$`

T_{DAILY}AVG = min(_0 + _1 sin() + _2 cos() +)

`$$` + Where: - T DAILY AVG: the daily average temperature for a given day of the year - β_0: the estimated intercept for the daily average temperature for a given day of the year (the day before January 1st). - β_1 and β_2: the coefficients estimated by the model. - ε is the error term.

```r
est_sta <- stations_and_names |> dplyr::select(WBANNO, LONGITUDE, LATITUDE)
n <- length(est_sta$WBANNO)
est_sta$trend <- rep(NA,n)
est_sta$is_sig <- rep(NA,n)

for (x in 1:n) {
  e <- estimate_temp_trend(merging, all_stations[x])
  est_sta$trend[x] <- e[1]
  est_sta$is_sig[x] <- ifelse(e[3] > 0.05, 0, 1)
}

inside_est <- sp::point.in.polygon(est_sta$LONGITUDE, est_sta$LATITUDE,
                                   usa_map$long, usa_map$lat)

est_sta <- est_sta[as.logical(inside_est), ]

est_not_sig <- dplyr::filter(est_sta, is_sig == 0)
est_sig <- dplyr::filter(est_sta, is_sig == 1)

# Plot for trends
plot_esta <- ggplot() +
  geom_point(data = est_sta, aes(x = LONGITUDE, y = LATITUDE, color = trend)) +
  scale_color_gradient(low = "blue", high = "red") +
  theme_minimal() +
  geom_polygon(data = map_data("state"), aes(x = long, y = lat, group = group),
               fill = NA, color = "black") +
  geom_point(data = est_not_sig, aes(x = LONGITUDE, y = LATITUDE)) +
  labs(title = "Estimated The Yearly Temperature Trend For Stations Inside The
                Contiguous United States")
plot_esta
```




```r
# Interpolated trends
# est_sig$T_DAILY_AVG <- est_sig$trend
# est <- interpolate_to_grid(blank_grid, stations_data = est_sig)
# plot_interpolations(est)
```

##Problem 6: Comparing our results to a reputable source To compare our results, we used data from the National Center for Environmental Information (NCEI) between the years 2000 and 2024, and used their data to calculate a overall trend for temperature rising in the United States.

We found that our overall trend we estimated and from this source are not equal to each other. This is likely due to our methods of estimating the overall trend are different from the NCEI.

```r
source_data <- read.csv("/Users/ethancarlson/Desktop/data.csv")
source_data$Date <- as.Date(source_data$Date)
source_data$Date <- 1:243
model <- lm(Value ~ Date, data=source_data)
summary(model)
#
# Call:
# lm(formula = Value ~ Date, data = source_data)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -17.421  -8.007   3.895   7.918  11.620
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept) 46.751502   1.150394  40.640   <2e-16 ***
# Date         0.004459   0.008175   0.546    0.586
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 8.939 on 241 degrees of freedom
# Multiple R-squared:  0.001233,    Adjusted R-squared:  -0.002911
# F-statistic: 0.2976 on 1 and 241 DF,  p-value: 0.5859

mean(est_sig$trend)
# [1] NaN
```