seesaw

```r
library(seesaw)
library(ggplot2)
library(sf)
#> Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
library(GpGp)
```

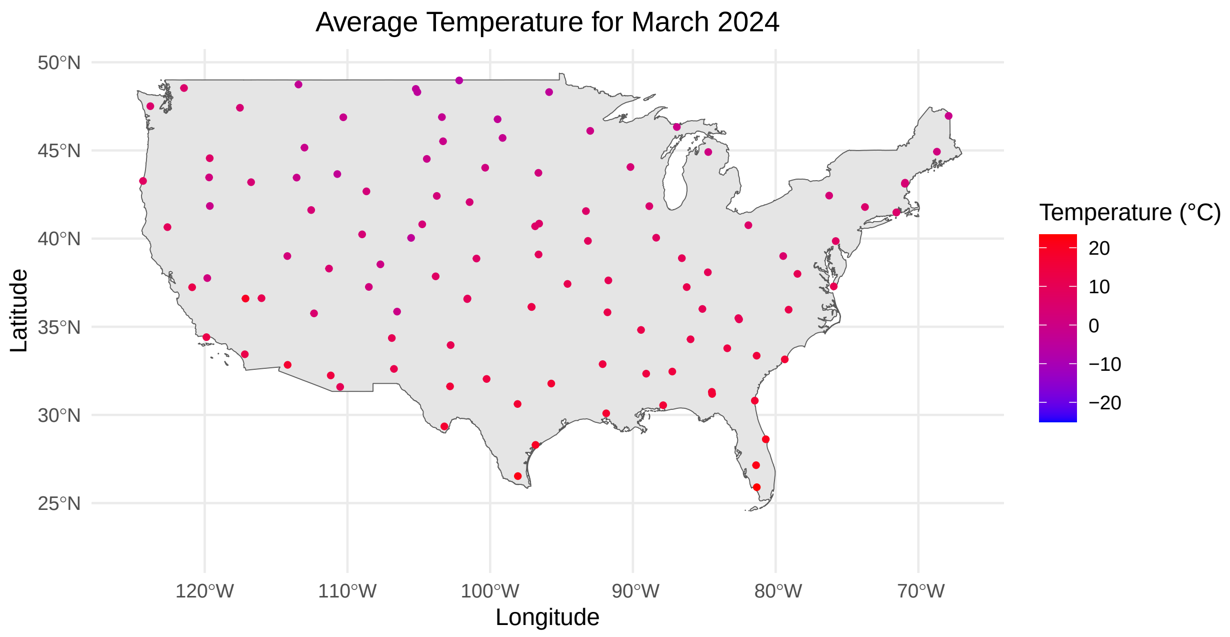
Load the Data

```r

data("full_table", package = "seesaw")
data("station_info", package = "seesaw")
data("shp_file", package = "seesaw")
data("elevation_grid", package = "seesaw")
data("training_X", package = "seesaw")

# remove canadian NOAA stations
station_info <- station_info[!is.na(station_info$state),]
```

1. Make a map of the average temperature at each station for the month of March
2.

```r
# Get list of station ids
station_ids <- station_info$station_id

# Filter data to include only March 2024, find average March 2024
# temperature for each station
march_data <- full_table[full_table$LST_DATE >= "2024-03-01"
                         & full_table$LST_DATE <= "2024-03-31", ]

# Remove NAs
march_data <- march_data[complete.cases(march_data), ]

average_temps <- aggregate(march_data$T_DAILY_AVG,
                           by = list( march_data$WBANNO ),
                           FUN = mean )

# Create a new data frame with station ids, average temperatures,lat and lon
df <- data.frame(station_id = unique(march_data$WBANNO),
                 average_temp = average_temps$x)
# Merge df with station_info to get lat and lon
df <- merge(df, station_info, by.x = "station_id", by.y = "station_id")

# Drop NAs
df <- df[complete.cases(df), ]
```

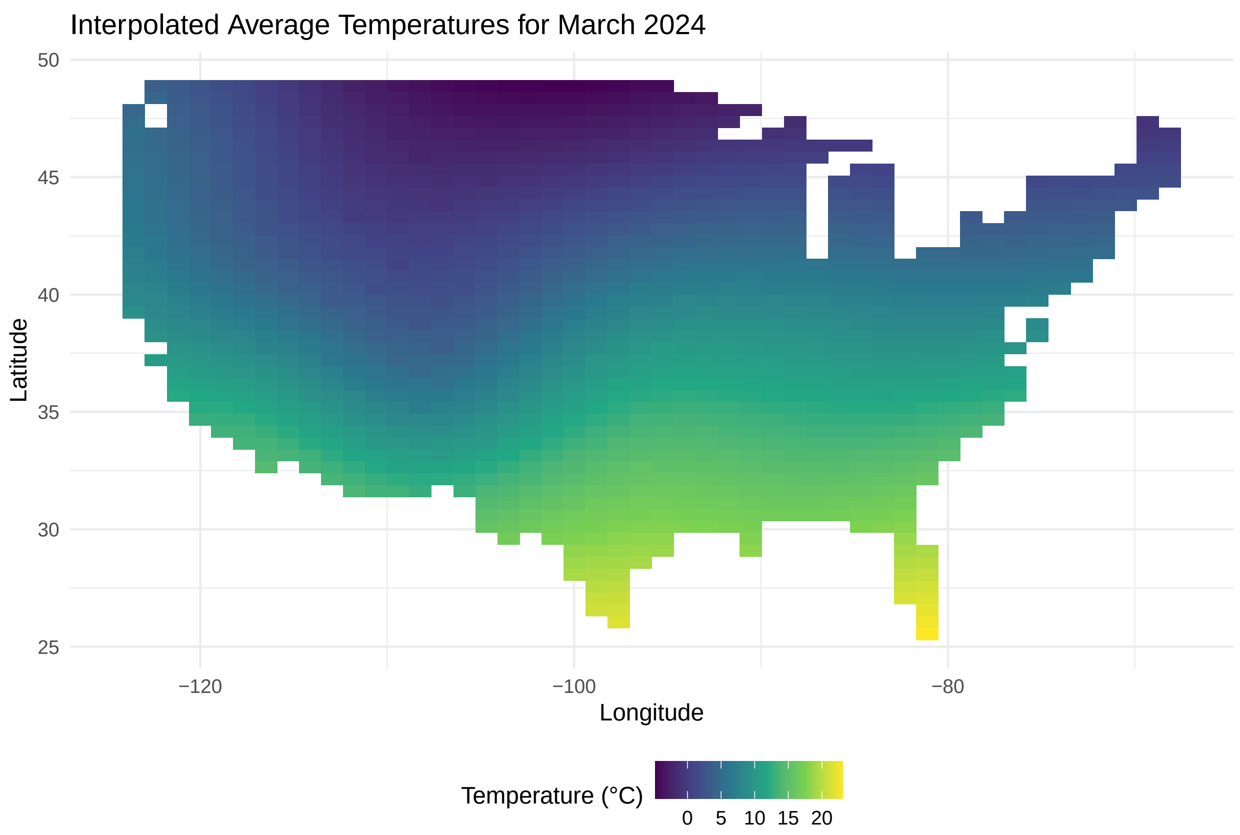
```r
# Plot the USA shapefile and add the average temperature data as points
usa_boundary <- st_crop(st_make_valid(shp_file), xmin = -125, xmax = -66.93457,
                    ymin = 22.396308, ymax = 49.384358)
#> Warning: attribute variables are assumed to be spatially constant throughout
#> all geometries

# Convert the df to the right coordinate reference system
df_sf <- st_as_sf(df, coords = c("longitude", "latitude"), crs =
                    st_crs(usa_boundary))
coordinates <- st_coordinates(df_sf)
df_sf$x <- coordinates[,1]
df_sf$y <- coordinates[,2]

# Use ggplot and sf to plot the map
ggplot() +
  geom_sf(data = usa_boundary) +
  geom_point(data = df_sf, aes(color = average_temp, x = x, y = y), size = 1) +
  coord_sf(xlim = c(-125, -66.93457), ylim = c(22.396308, 49.384358)) +
  theme_minimal() +
  labs(title = "Average Temperature for March 2024",
       x = "Longitude", y = "Latitude") +
  scale_color_gradient(name = "Temperature (°C)", low = "blue", high = "red") +
  theme(plot.title = element_text(hjust = 0.5))
```



2. Fit a spatial model and plot an interpolated map of average temperatures for March 2024. Consider including elevation in your model.

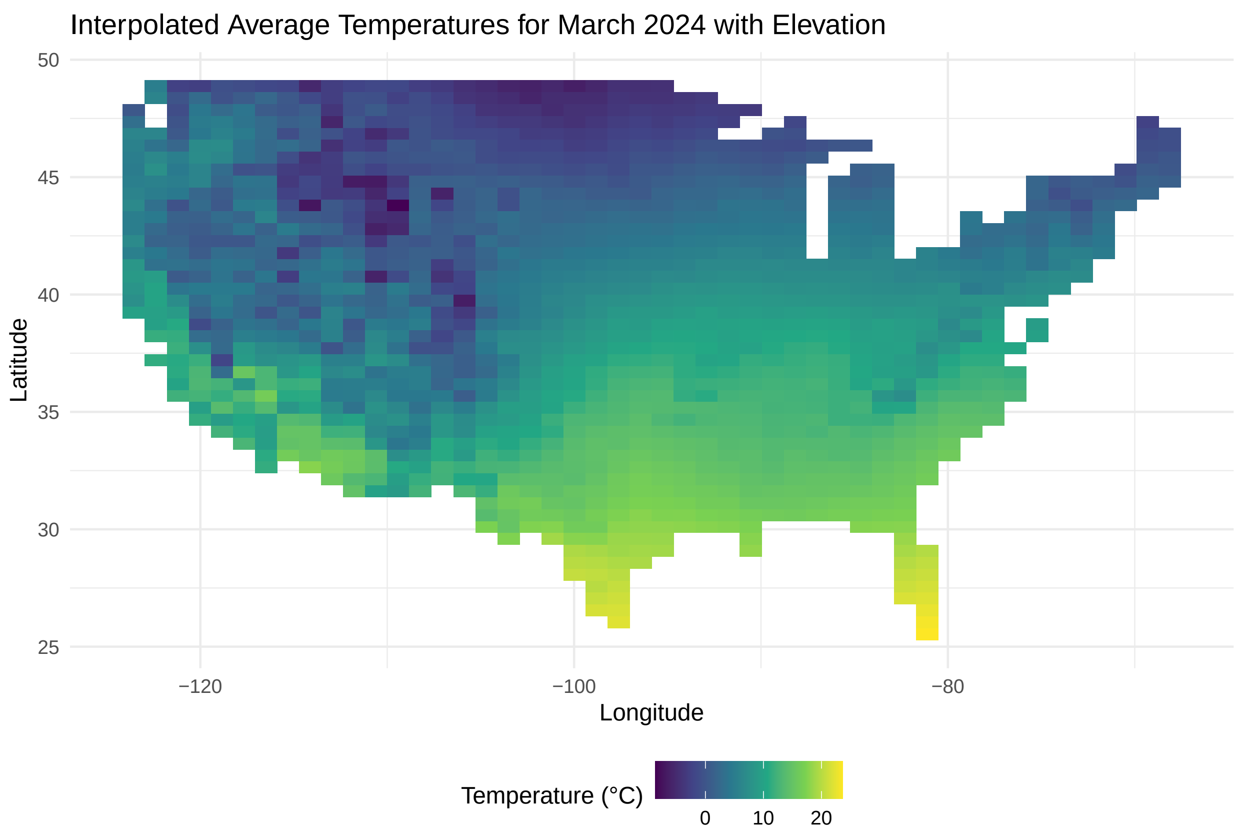
A model without elevation:

```r
# Set up the grid for interpolation
res <- 50
grid <- usagrid(res)

# Define the X and y to train the model
X <- as.data.frame(model.matrix(~ x+y , data = df_sf))
y <- df_sf$average_temp

# Use seesaw's interpolate_grid function to fit the spatial model
spatial_model <- interpolate_grid(X,y,resolution = res)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

graph_interp(spatial_model, grid) +
  labs( title = "Interpolated Average Temperatures for March 2024",
    x = "Longitude", y = "Latitude",
    fill = "Temperature (°C)")
```



A model with elevation:

```
# Fit a spatial model with elevation
locs <- X[, c("x", "y")]

Xpred <- as.data.frame(model.matrix(~ x+y+elevation, data = elevation_grid))
# Create model matrix
preds <- interpolate_grid(training_X, y, Xpred = Xpred)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees
graph_interp(preds,grid) +
  labs( title = "Interpolated Average Temperatures for March 2024 with Elevation",
    x = "Longitude", y = "Latitude",
    fill = "Temperature (°C)")
```

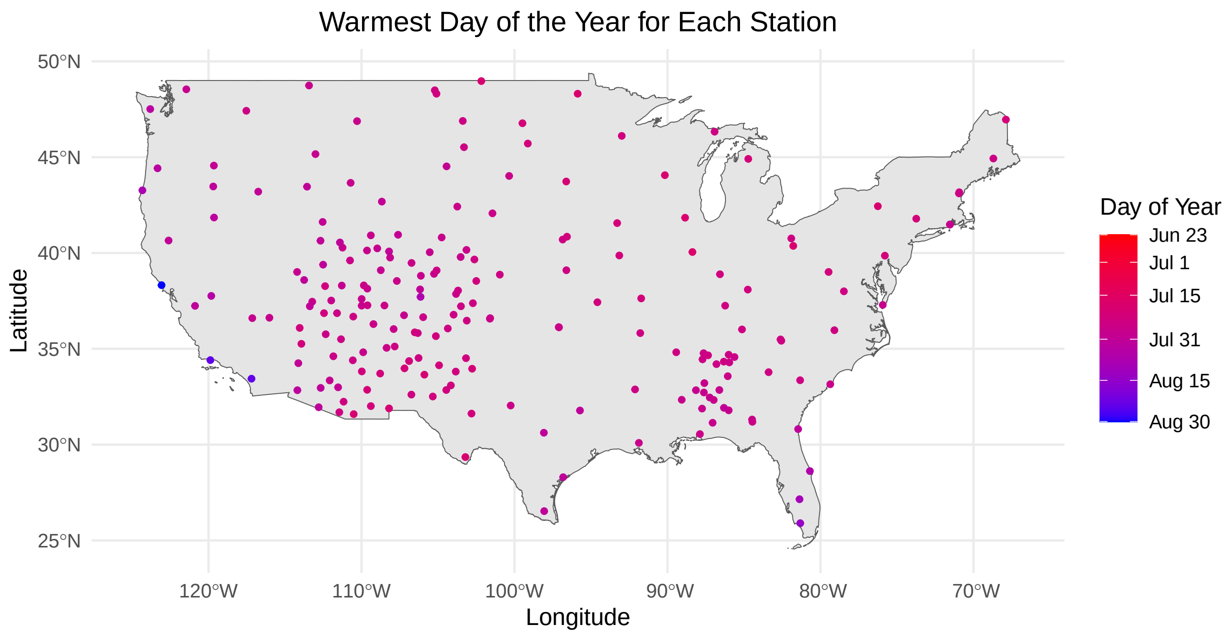

Interpolated Average Temperatures for March 2024 with Elevation
Latitude
50
45
40
35
30
25
Longitude
−120
−100
−80
Temperature (°C)
0
10
20

3. Estimate the warmest and coldest day of the year for each station, and plot those days on two maps. Think carefully about how to represent the days numerically.

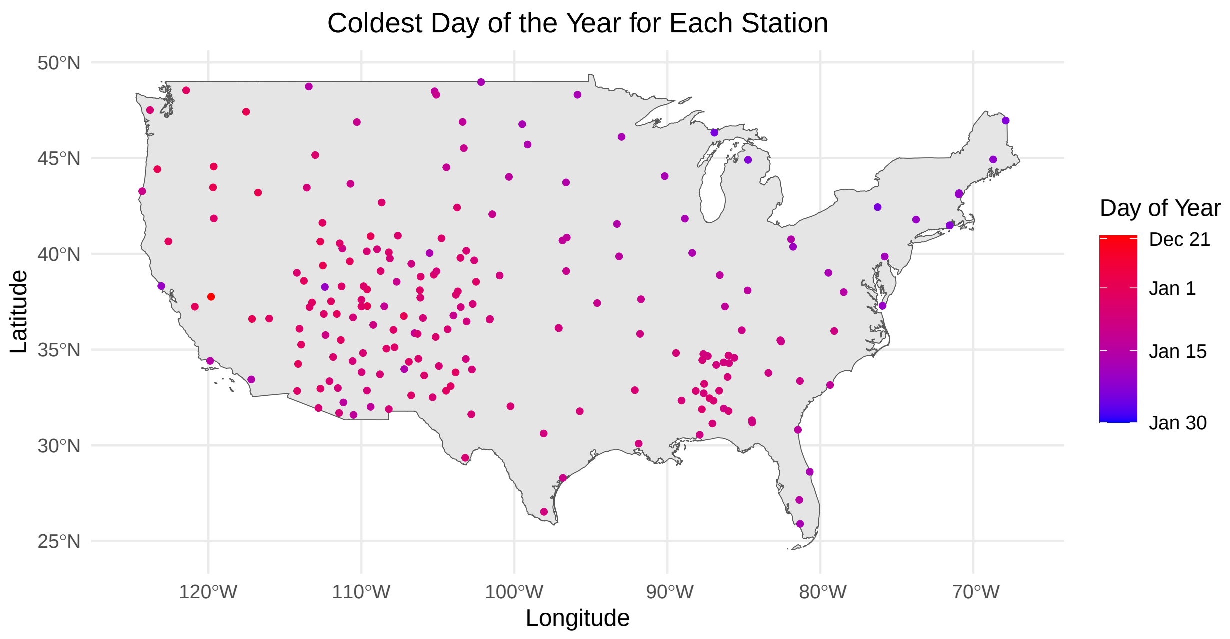
In your report, describe the statistical analysis that you used for estimating the warmest and coldest days at each station, including writing down any statistical models in mathematical notation. Be sure to define all your symbols and assumptions.

Interpolate maps of the warmest and coldest days, and plot the interpolated maps of warmest and coldest days.

```r
# Create a function for finding the estimated warmest and coldest days of year
# for a given station
warmest_coldest <- function(station_id){
  cycle <- yearly_cycle_station(station_id, "T_DAILY_AVG")
  #warmest_day <- cycle$DOY[which.max(cycle$Expected_T_DAILY_AVG)]
  warmest_day <- which.max(cycle$Expected_T_DAILY_AVG)
  #coldest_day <- cycle$DOY[which.min(cycle$Expected_T_DAILY_AVG)]
  coldest_day <- which.min(cycle$Expected_T_DAILY_AVG)
  return(c(warmest_day, coldest_day))
}

# Preallocate the data frame
warmest_coldest_days <- data.frame(station_id = station_ids,
                                   warmest_day = integer(length(station_ids)),
                                   coldest_day = integer(length(station_ids)))

# Loop through each station
for (i in 1:length(station_ids)){
  # Assign values directly without using c()
  warmest_coldest_days[i, c("warmest_day", "coldest_day")] <-
    warmest_coldest(station_ids[i])
}

# Use sf to get the correct coordinate reference system to plot
df_warmest_coldest <- merge(warmest_coldest_days, station_info,
                    by.x = "station_id", by.y = "station_id")
df_warm_cold_sf <- st_as_sf(df_warmest_coldest,
                          coords = c("longitude", "latitude"),
                          crs = st_crs(usa_boundary))

coordinates <- st_coordinates(df_warm_cold_sf)
df_warm_cold_sf$x <- coordinates[,1]
df_warm_cold_sf$y <- coordinates[,2]

# Plot the warmest day
ggplot() +
  geom_sf(data = usa_boundary) +
  coord_sf(lims_method = "geometry_bbox") +
  geom_point(data = df_warm_cold_sf,
             aes(color = warmest_day, x = x, y = y), size = 1) +
  theme_minimal() +
  labs(title = "Warmest Day of the Year for Each Station",
       x = "Longitude", y = "Latitude") +
  scale_color_gradient(name = "Day of Year",
                     labels = c("Jun 23", "Jul 1", "Jul 15", "Jul 31",
                                "Aug 15", "Aug 30"),
                     breaks = c(174, 184, 196, 212, 227, 242),
                     trans = "reverse", low = "blue", high = "red") +
  guides(colour = guide_colourbar(title.vjust = 1.5)) +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# Function to map days from December 1st to January 31st to a continuous scale
day_to_continuous_scale <- function(day) {
  if (day <= 31) {
    return(day + 31)
  } else {
    return(31 - (365-day)/31)
  }
}

# Apply the function to the coldest day data
df_warm_cold_sf$continuous_day <- sapply(df_warm_cold_sf$coldest_day,
                                         day_to_continuous_scale)

# Plot the shapefile and add the coldest day data as points
ggplot() +
  geom_sf(data = usa_boundary) +
  coord_sf(lims_method = "geometry_bbox") +
  geom_point(data = df_warm_cold_sf,
             aes(color = continuous_day, x = x, y = y), size = 1) +
  scale_color_gradient(name = "Day of Year",
                       labels = c("Dec 21", "Jan 1", "Jan 15", "Jan 30"),
                       breaks = c(21, 32, 46, 62),
                       trans = "reverse", low = "blue", high = "red") +
  theme_minimal() +
  labs(title = "Coldest Day of the Year for Each Station",
       x = "Longitude", y = "Latitude") +
  guides(colour = guide_colourbar(title.vjust = 1.5)) +
  theme(plot.title = element_text(hjust = 0.5))
```



Statistical Model:

The estimated average warmest and coldest day of the year was calculated by fitting a sinusoidal model to the daily average temperature data for each station. The model is given by:

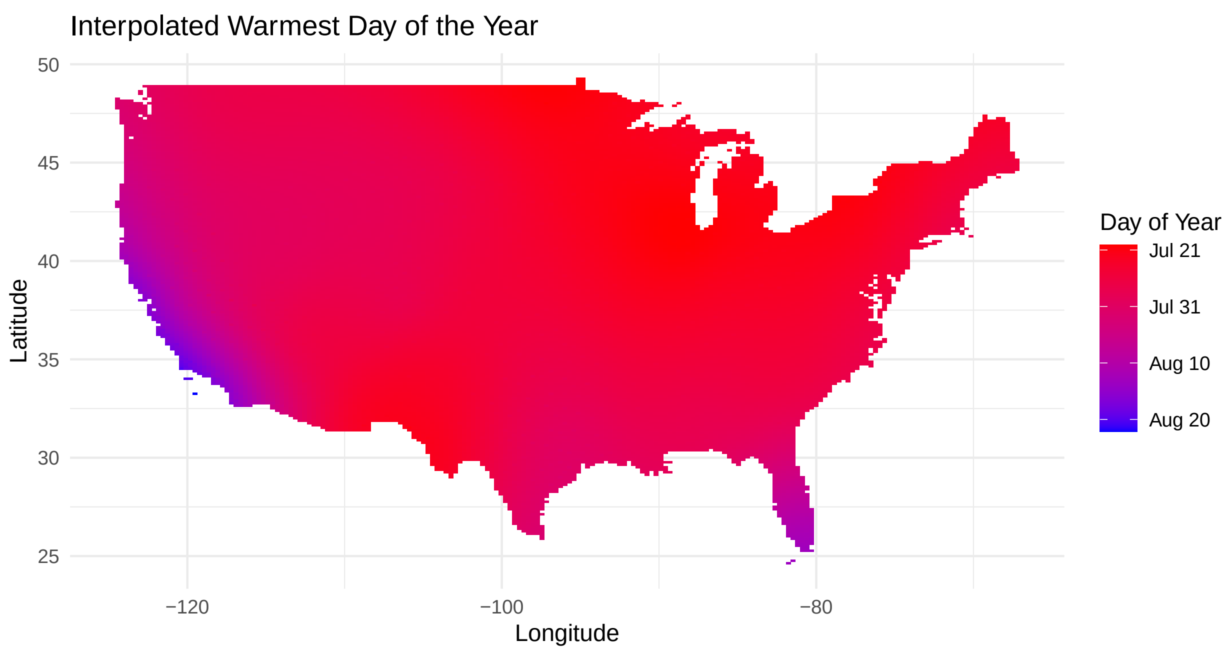
$Y_i = \beta_0 + \beta_1 * \sin(2\pi d/365) + \beta_2 * \cos(2\pi d/365) + \epsilon_i$

where Y_i is the daily average temperature, d is the day of the year, and ϵ_i is the error term. The estimated warmest day of the year is the day with the highest expected temperature, and the estimated coldest day of the year is the day with the lowest expected temperature.

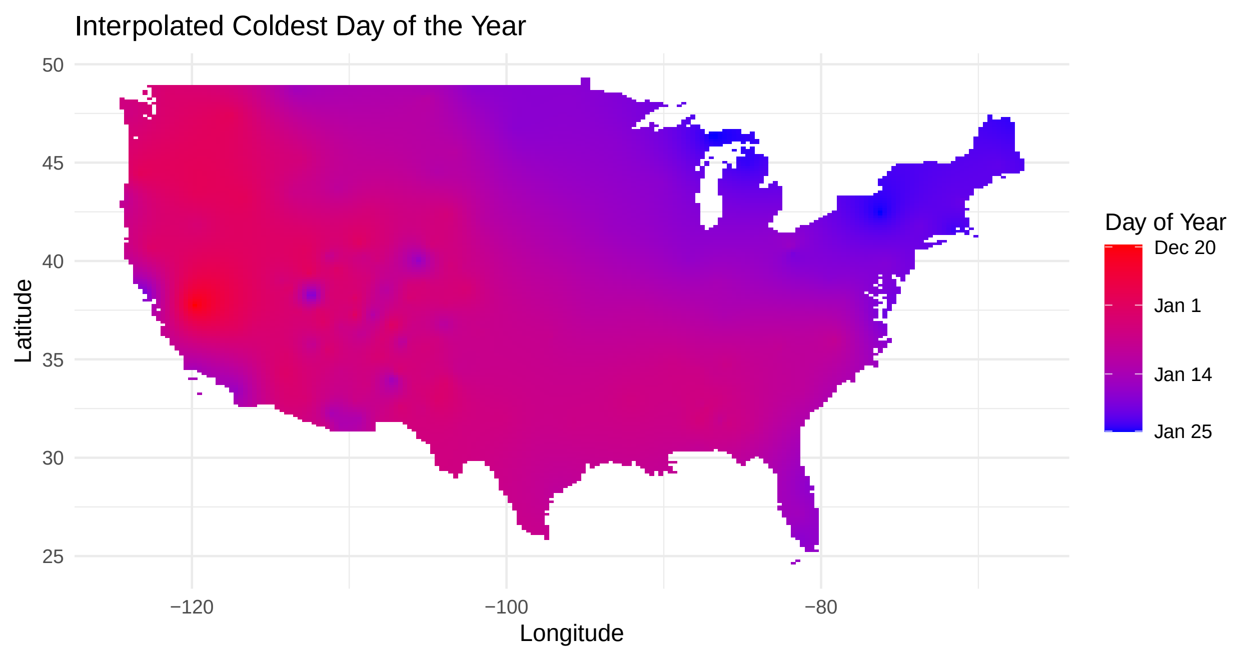
Assumptions: 1. The daily average temperature data is sinusoidal with a period of 365 days. 2. The daily average temperature data is stationary over time. 3. The error term is normally distributed with mean 0 and constant variance.

```
# Create higher resolution grid
res <- 200
grid <- usagrid(res)
```

```r
# Ensure correct types
df_warmest_coldest$rescaled_coldest <-
  as.numeric(df_warm_cold_sf$continuous_day)
df_warmest_coldest$longitude <- as.numeric(df_warmest_coldest$longitude)
df_warmest_coldest$latitude <- as.numeric(df_warmest_coldest$latitude)
locs <- df_warmest_coldest[, c("longitude", "latitude")]

# Make a dataframes of locations and y_cold/y_warm
cold_dataset <- df_warmest_coldest[,c("longitude", "latitude",
                                      "rescaled_coldest")]
warm_dataset <- df_warmest_coldest[,c("longitude", "latitude", "warmest_day")]

# Drop repeated rows
cold_dataset <- cold_dataset[!duplicated(cold_dataset),]
warm_dataset <- warm_dataset[!duplicated(warm_dataset),]

# Set up X and y for the interpolation
X_cold <- as.data.frame(cbind(1, cold_dataset[, c("longitude", "latitude")]))
names(X_cold) <- c("intercept", "x", "y")
X_warm <- as.data.frame(cbind(1, warm_dataset[, c("longitude", "latitude")]))
names(X_warm) <- c("intercept", "x", "y")
y_cold <- cold_dataset$rescaled_coldest
y_warm <- warm_dataset$warmest_day

# Run the interpolations
warm_pred <- interpolate_grid(X_warm, y_warm, resolution = res)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees
cold_pred <- interpolate_grid(X_cold,y_cold, resolution = res)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

# Plot interpolations
graph_interp(warm_pred, grid) +
  labs( title = "Interpolated Warmest Day of the Year",
    x = "Longitude", y = "Latitude",
    fill = "Day of Year") +
scale_fill_gradient(name = "Day of Year",
                    labels = c("Jul 21", "Jul 31", "Aug 10", "Aug 20"),
                       breaks = c(202, 212, 222, 232),
                    low = "blue", high = "red",
                    trans = "reverse") +
  guides(colour = guide_colourbar(title.vjust = 1.5)) +
  theme(legend.position = "right")
#> Scale for fill is already present.
#> Adding another scale for fill, which will replace the existing scale.
```

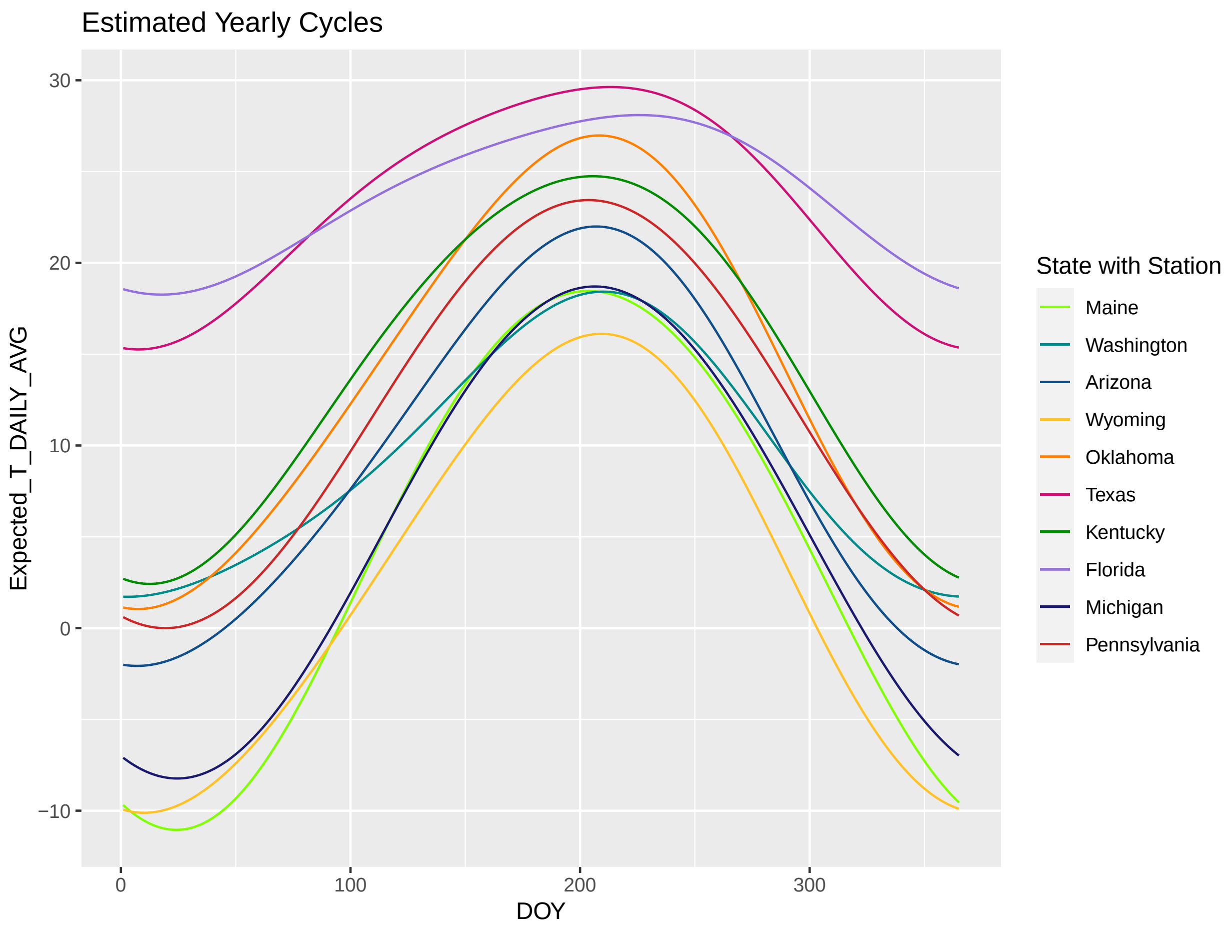


```
graph_interp(cold_pred, grid) +
  labs( title = "Interpolated Coldest Day of the Year",
    x = "Longitude", y = "Latitude",
    fill = "Day of Year") +
  scale_fill_gradient(name = "Day of Year",
                      labels = c("Dec 20", "Jan 1", "Jan 14", "Jan 25"),
                       breaks = c(21, 32, 45, 55.8),
                      low = "blue", high = "red", trans = "reverse")  +
  guides(colour = guide_colourbar(title.vjust = 1.5)) +
  theme(legend.position = "right")
#> Scale for fill is already present.
#> Adding another scale for fill, which will replace the existing scale.
```



4. Make a single plot of the estimated yearly cycles for 10 different stations, highlighting a diversity of climates around the contiguous USA. Your plot should clearly indicate which cycle is from which station.

```r
# Select 10 stations from around the contiguous USA
# ( Maine, Washington, Arizona, Wyoming, Oklahoma, Texas, Kentucky, Florida,
#   Michigan, Pennsylvania )
stations <- c("94645", "04223", "53169", "04131", "53182", "12987", "63849",
              "92826", "54810", "03761")
states <- c("Maine", "Washington", "Arizona", "Wyoming", "Oklahoma", "Texas",
           "Kentucky", "Florida", "Michigan", "Pennsylvania")
# Select 10 colors
cols <- c("chartreuse1", "darkcyan", "dodgerblue4", "goldenrod1", "darkorange1",
          "deeppink3", "green4", "mediumpurple", "midnightblue", "firebrick3")
# Initialize plot
plt <- ggplot() +
  labs(title = "Estimated Yearly Cycles",
       x = "DOY", y = "Expected_T_DAILY_AVG")
# Calculate and plot estimated yearly cycle for each station
for (i in 1:10){
  cycle <- yearly_cycle_station(stations[i])
  cycle$state <- states[i]
  plt <- plt + geom_line(data = cycle,
                                aes(x = DOY, y = Expected_T_DAILY_AVG,
                                    color = state))
}
# Add legend
plt <- plt + scale_color_manual(name = "State with Station", breaks = states,
                                 values = setNames(cols, states))
plt
```



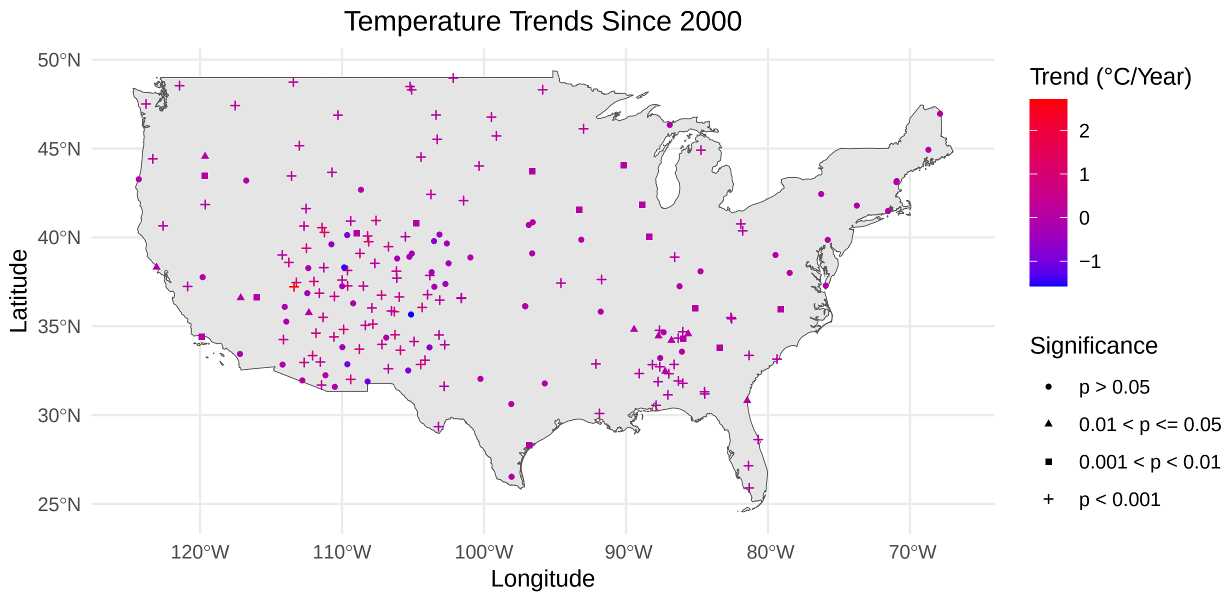
5. Estimate the trend over the years for each station, in units of degrees Fahrenheit per year, and plot the trend values on a map. Indicate visually on your map which of the trends are statistically significant.

In your report, write the statistical model that you used in mathematical notation. Be sure to define all your symbols and assumptions.

Interpolate the estimated trends to a grid, and plot them on a map. For the interpolations, you may consider using only the trend estimates whose standard errors are sufficiently small.

```
# Find the temperature trends for each station (since 2000)
temp_trends <-  sapply(station_info$station_id, seesaw::trend_of_temps)
temp_trends <- t(temp_trends)

# Create a new data frame with station ids, temperature trend, lat and long
station_trends <- data.frame(station_id = rownames(temp_trends),
```

```r
                              trend = temp_trends[,1], SE = temp_trends[,2])
# Merge df with station_info to get lat and lon
station_trends <- merge(station_trends, station_info, by.x = "station_id",
                        by.y = "station_id")

# Determine which trends are statistically significant
sd <- var(station_trends$trend, na.rm = TRUE)
n <- length(station_trends$trend)
test_statistics <- (station_trends$trend - 0) / ( sd/sqrt(n) )
p_vals <- pt( test_statistics, df = n - 1, lower.tail = FALSE )

determine_significance <- function(pval){

  # Assume all pvals are greater than 0.05
  signif <- " "
  if (is.na(pval) || pval > 0.05){
    return(signif)
  }
  # Greater than 0.01, less than 0.05
  if (pval <= 0.05 & pval > 0.01) {
    signif <- "*"
  } else if (pval <= 0.01 & pval > 0.001) {
  # Greater than 0.001, less than 0.01
    signif <- "**"
  } else if (pval <= 0.001) {
  # Less than 0.001
    signif <- "***"
  }
  return(signif)
}

station_trends$signif <- sapply(p_vals, determine_significance)


# Plot the USA shapefile
usa_boundary <- st_crop(st_make_valid(shp_file), xmin = -125, xmax = -66.93457,
                    ymin = 22.396308, ymax = 49.384358)

station_trends_sf <- st_as_sf(station_trends,
                              coords = c("longitude", "latitude"),
                              crs = st_crs(usa_boundary))
coordinates <- st_coordinates(station_trends_sf)
station_trends_sf$x <- coordinates[,1]
station_trends_sf$y <- coordinates[,2]

# Plot overall trends
signifs = c(" ", "*", "**", "***")
signif.codes <- c("p > 0.05", "0.01 < p <= 0.05", "0.001 < p < 0.01",
                  "p < 0.001")
ggplot() +
  geom_sf(data = usa_boundary) +
  coord_sf(lims_method = "geometry_bbox") +
  geom_point(data = station_trends_sf,
             aes(color = trend, x = x, y = y, shape = signif),
             size = 1) + theme_minimal() +
  labs(title = "Temperature Trends Since 2000",
       x = "Longitude", y = "Latitude") +
  scale_color_gradient(name = "Trend (°C/Year)", low = "blue", high = "red") +
  scale_shape_discrete(name = "Significance", breaks = signifs,
                                labels = signif.codes) +
  theme(plot.title = element_text(hjust = 0.5))
```



Statistical Model:

The estimated trend over the years for each station was calculated by fitting a linear model for each month of the year and taking the average of the slope coefficients from each linear model. This is given by:

$estimated\ trend = \frac{\sum_{i=1}^{12} \beta_{1i}}{12}$

where β_{1i} denotes the slope coefficient for ith month, and is derived from the model:

$Y_i = \beta_0 + \beta_1 * d + \epsilon_i$

where Y_i is the daily average temperature, d is the day of the year, and ϵ_i is the error term.

The standard error was found by converting the standard error of each month's slope coefficient into a variance, pooling the variances, and calculating the new standard error. That is,

$Pooled\ Variance = \frac{\sum_{i=1}^{12} SE_i\ *\ n_i}{N-12}$ «««< HEAD

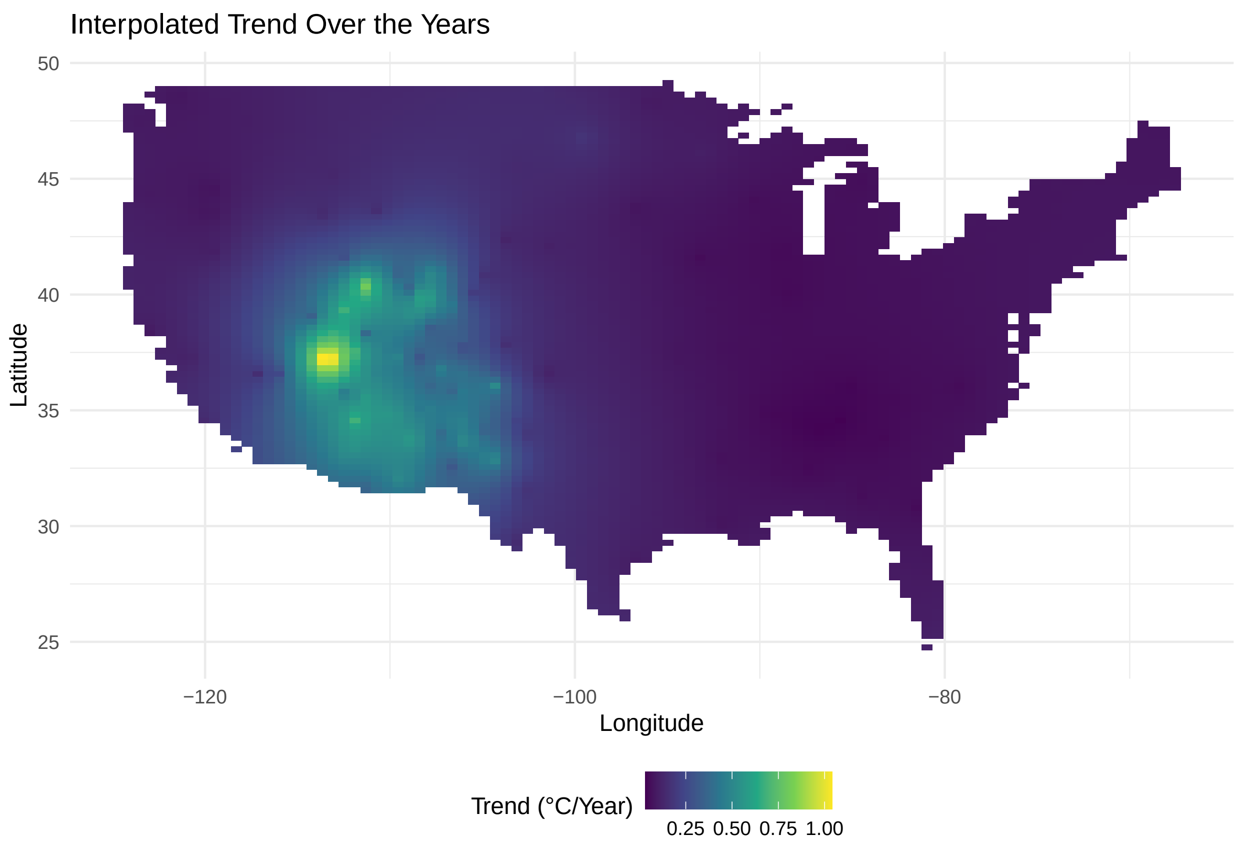
======= »»»> 7b390d4cc8e0fcfc9eb54c3fce5fa3b2484b7a8d $SE = \sqrt{\frac{Pooled\ Variance}{N}}$

where SE_i is the standard error for the ith month, n_i is the number of available readings for the 1st of that month at that station, and N is the total sample size (total number of available readings for the 1st of all months between the start and end dates at that station).

Assumptions: 1. The change in temperature over the years can be modeled linearly. Further, we assume that the the change in monthly temperatures overall the years follows a linear trend as well. 2. The overall trend for each month contributes equal to the station's trend overall (that is, each month's trend is equally indicative of the trend overall) 3. The error term is normally distributed with mean 0 and constant variance.

To determine statistical significance, we assume that the temperature trends of stations are normally distributed with mean 0, indicating the temperature has not changed over the years. We run a t-test with significance level $\alpha = 0.05$ to determine which trends are significant.

```r
res <- 100

# Subset station_trends to only include significant trends
significant_trends <- station_trends[station_trends$signif != " ",]

# Interpolate trends for contiguous US
y <- significant_trends$trend
locs <- significant_trends[, c("longitude", "latitude")]
names(locs) <- c("x", "y")
locs$x <- as.numeric(locs$x)
locs$y <- as.numeric(locs$y)
X <- as.data.frame(cbind(1, locs))

grid <- usagrid(res)
# Make predictions
trend_preds <- interpolate_grid(X,y,resolution = res)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

# Plot interpolations
graph_interp(trend_preds, grid) +
  labs( title = "Interpolated Trend Over the Years",
    x = "Longitude", y = "Latitude",
    fill = "Trend (°C/Year)")
```



6. Find a reputable source for the average temperature trend in the contiguous USA over the past 20 years, and compare your results to the source's.

Reputable Source: US EPA https://www.epa.gov/climate-indicators/climate-change-indicators-us-and-global-temperature

```r
median(significant_trends$trend)*10
#> [1] 0.7349643
```

The EPA reports that the average temperature in the contiguous US has risen by 0.55°F per decade since 1981. This is equivalent to 0.306°C per decade. Our model estimates the average temperature trend in the contiguous US to be roughly 0.73°C per decade. However, this is largely due to the fact that we are only considering the temperature trends since 2000, where according to the EPA, temperature has been increasing even faster. Additionally, it is the case that the vast majority of the US is increasing at a rate lower than 0.73°C per decade, while stations in the Southwest are increasing at a much greater rate.