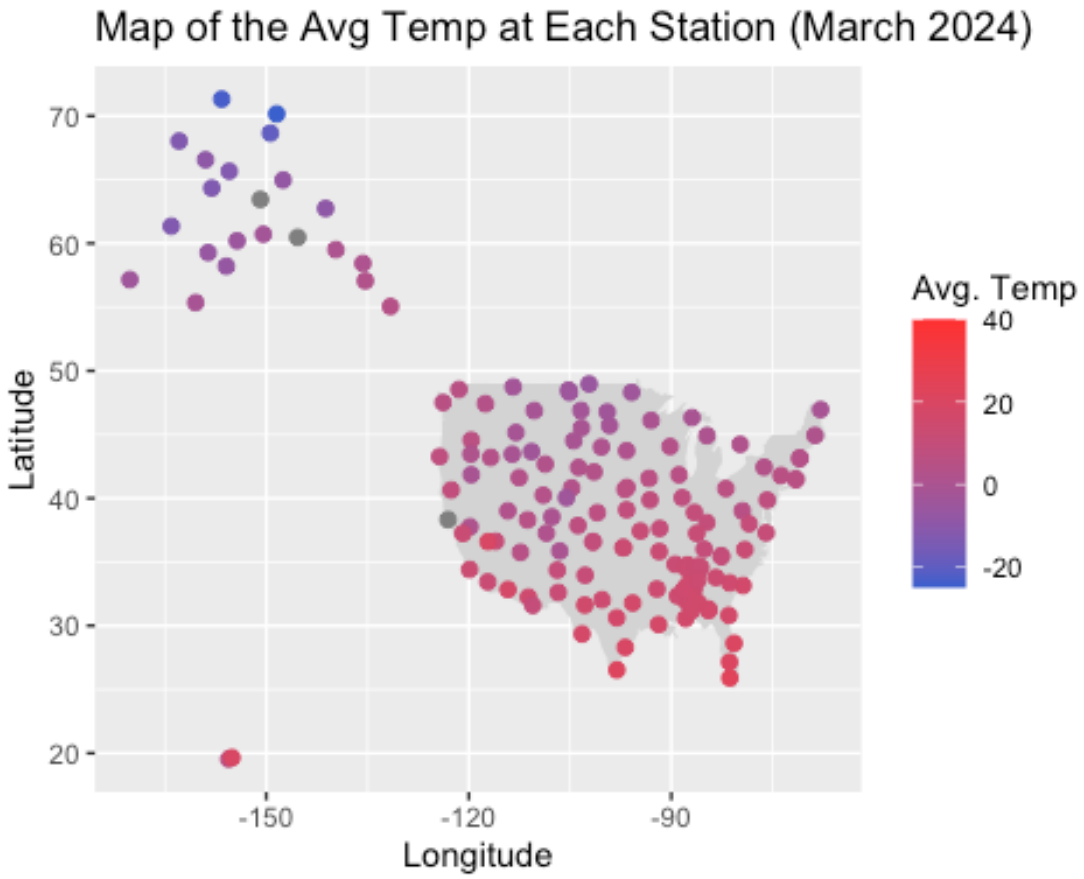
# climateAPI Vignette

```
library(climateAPI)
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
library(GpGp)
library(maps)
library(ggplot2)

data(daily_data)
data(stations)
globalVariables(colnames(daily_data))
#>  [1] "WBANNO"                  "LST_DATE"
#>  [3] "CRX_VN"                  "LONGITUDE"
#>  [5] "LATITUDE"                "T_DAILY_MAX"
#>  [7] "T_DAILY_MIN"             "T_DAILY_MEAN"
#>  [9] "T_DAILY_AVG"             "P_DAILY_CALC"
#> [11] "SOLARAD_DAILY"           "SUR_TEMP_DAILY_TYPE"
#> [13] "SUR_TEMP_DAILY_MAX"      "SUR_TEMP_DAILY_MIN"
#> [15] "SUR_TEMP_DAILY_AVG"      "RH_DAILY_MAX"
#> [17] "RH_DAILY_MIN"            "RH_DAILY_AVG"
#> [19] "SOIL_MOISTURE_5_DAILY"   "SOIL_MOISTURE_10_DAILY"
#> [21] "SOIL_MOISTURE_20_DAILY"  "SOIL_MOISTURE_50_DAILY"
#> [23] "SOIL_MOISTURE_100_DAILY" "SOIL_TEMP_5_DAILY"
#> [25] "SOIL_TEMP_10_DAILY"      "SOIL_TEMP_20_DAILY"
#> [27] "SOIL_TEMP_50_DAILY"      "SOIL_TEMP_100_DAILY"
globalVariables(colnames(stations))
#>  [1] "WBANNO"                  "LST_DATE"
#>  [3] "CRX_VN"                  "LONGITUDE"
#>  [5] "LATITUDE"                "T_DAILY_MAX"
#>  [7] "T_DAILY_MIN"             "T_DAILY_MEAN"
#>  [9] "T_DAILY_AVG"             "P_DAILY_CALC"
#> [11] "SOLARAD_DAILY"           "SUR_TEMP_DAILY_TYPE"
#> [13] "SUR_TEMP_DAILY_MAX"      "SUR_TEMP_DAILY_MIN"
#> [15] "SUR_TEMP_DAILY_AVG"      "RH_DAILY_MAX"
#> [17] "RH_DAILY_MIN"            "RH_DAILY_AVG"
#> [19] "SOIL_MOISTURE_5_DAILY"   "SOIL_MOISTURE_10_DAILY"
#> [21] "SOIL_MOISTURE_20_DAILY"  "SOIL_MOISTURE_50_DAILY"
#> [23] "SOIL_MOISTURE_100_DAILY" "SOIL_TEMP_5_DAILY"
#> [25] "SOIL_TEMP_10_DAILY"      "SOIL_TEMP_20_DAILY"
#> [27] "SOIL_TEMP_50_DAILY"      "SOIL_TEMP_100_DAILY"
```

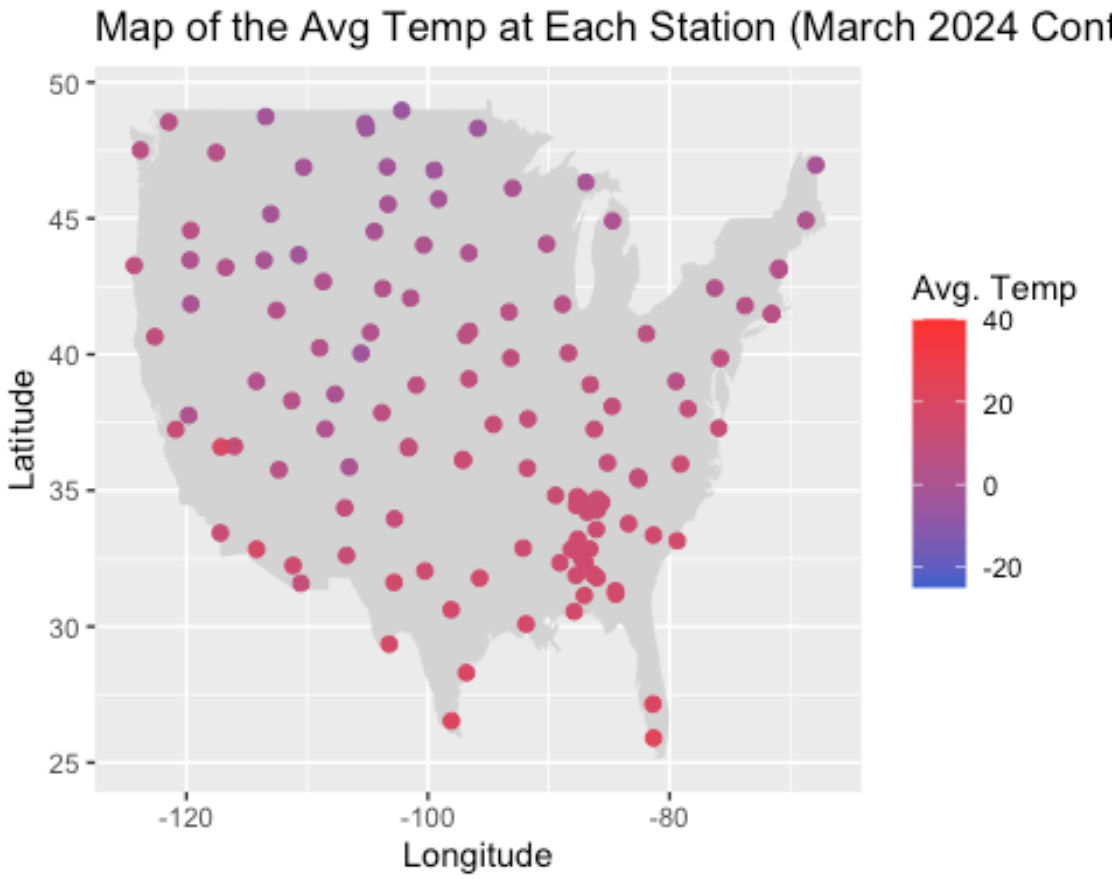
```
#> [29] "Station_Name"            "State"
#> [31] "Station_ID"              "Longitude"
#> [33] "Latitude"
```

1. Make a map of the average temperature at each station for the month of March 2024.

```
#Copy data
dat <- daily_data

#Create a variable yearmo to allow us to filter for just March 2024
dat$yearmo <- format(dat$LST_DATE, '%Y-%m')

#Summarize and groupby gets us average temperatures for each unique WBANNO
march.data <-
  dat |> filter(yearmo == '2024-03') |> group_by(WBANNO) |>
summarize(avg_temp = mean(T_DAILY_AVG, na.rm = TRUE))

#Ordering by station to allow us to cbind dataframes and have data match.
station.data <- stations[order(as.numeric(stations$Station_ID)), ]
station.data <-
  station.data[which(station.data$Station_ID %in% march.data$WBANNO), ]
station.data <- station.data[!duplicated(station.data$Station_ID), ]

#Dataframe of average temperatures at each station
march.station <-
  as.data.frame(cbind(
    as.numeric(station.data$Longitude),
    as.numeric(station.data$Latitude),
    march.data$avg_temp
  ))

#All stations (Inlcuding outside of Contiguous USA)
plot1 <- plot_predictions(march.station)
plot1 + labs(title = "Map of the Avg Temp at Each Station (March 2024)",
colour =
               'Avg. Temp')
```

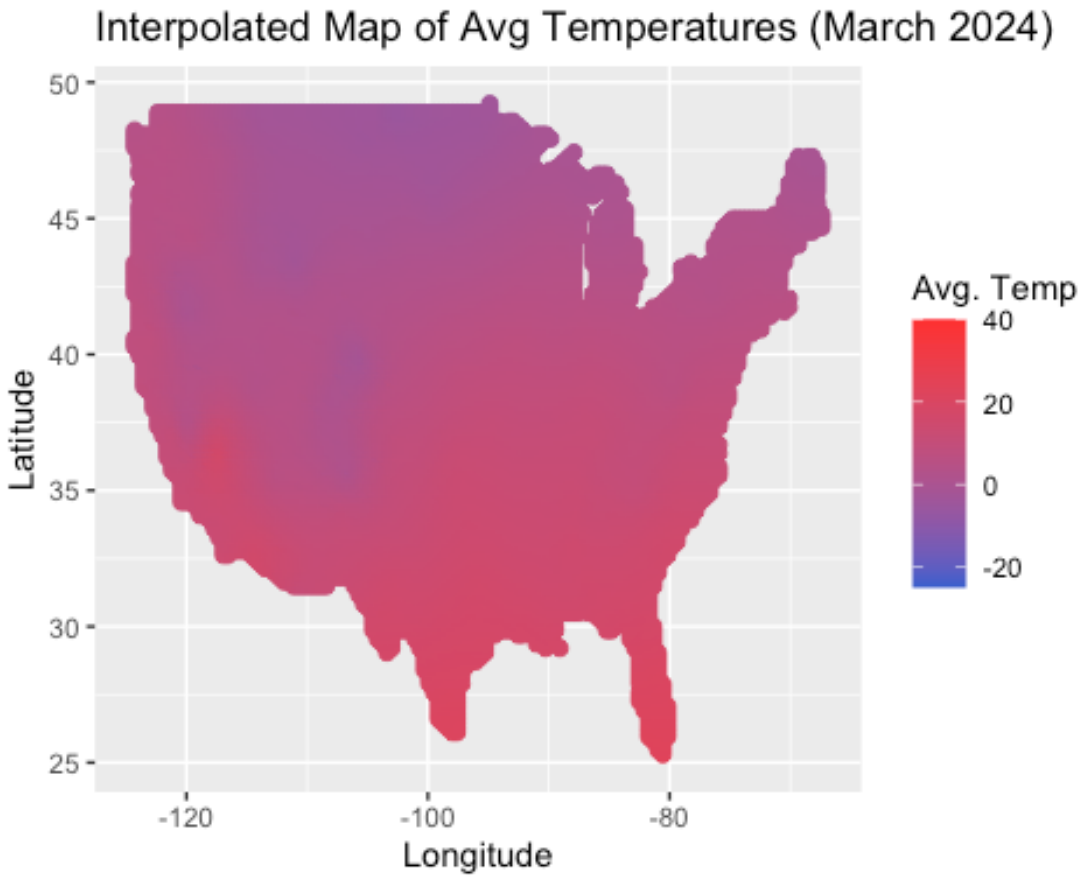



```
#Only Contiguous USA
inside <- map.where('usa', march.station[, 1:2])

#ifElse to create a logical vector for plotting indices
inside.logical <- ifelse(is.na(inside), FALSE, TRUE)

contig.data <- march.station[inside.logical, ]

plot2 <- plot_predictions(contig.data)
plot2 + labs(title = "Map of the Avg Temp at Each Station (March 2024
Contiguous)", colour =
               'Avg. Temp')
```




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

```
#Filter march data
march.data <- dat |> filter(yearmo == '2024-03')

#Interpolate
interpy <- interpolate_data('T_DAILY_AVG', march.data, 150)
#> 157 observations removed due to missingness or Inf
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

#Plot
plot3 <- plot_predictions(interpy)
plot3 + labs(title = 'Interpolated Map of Avg Temperatures (March 2024)',
colour =
               "Avg. Temp")
```




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.

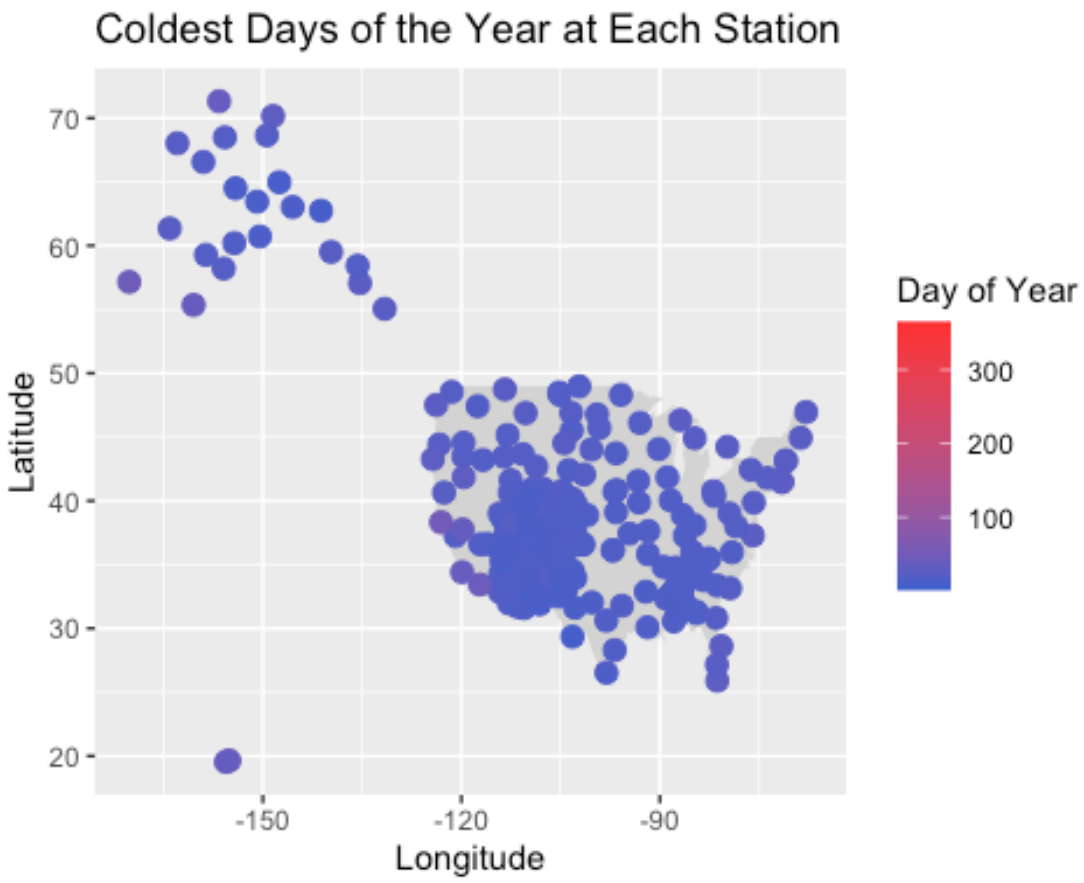
   In your report, describe the statistical analysis that you used for estimating the warmest and coldest days at each station, including writing down any statistical models in mathematical notation. Be sure to define all your symbols and assumptions.

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

```
#Remove stations that do not have a full years worth of data to analyze
stations.to.analyze <- stations[c(-213, -231, -235,-236,-237), ]

#Initialize empty matrix
temp <- matrix(NA, nrow = nrow(stations.to.analyze), ncol = 5)

#Use yearly_cycle to get station data for those that we want to analyze
for (i in 1:nrow(stations.to.analyze)) {
  dat <- yearly_cycle(stations.to.analyze[i, 'Station_ID'])
  temps <- dat$Predicted_Temp
  temp[i, 5] <- dat[which.min(temps), 1]
  temp[i, 4] <- dat[which.max(temps), 1]
```

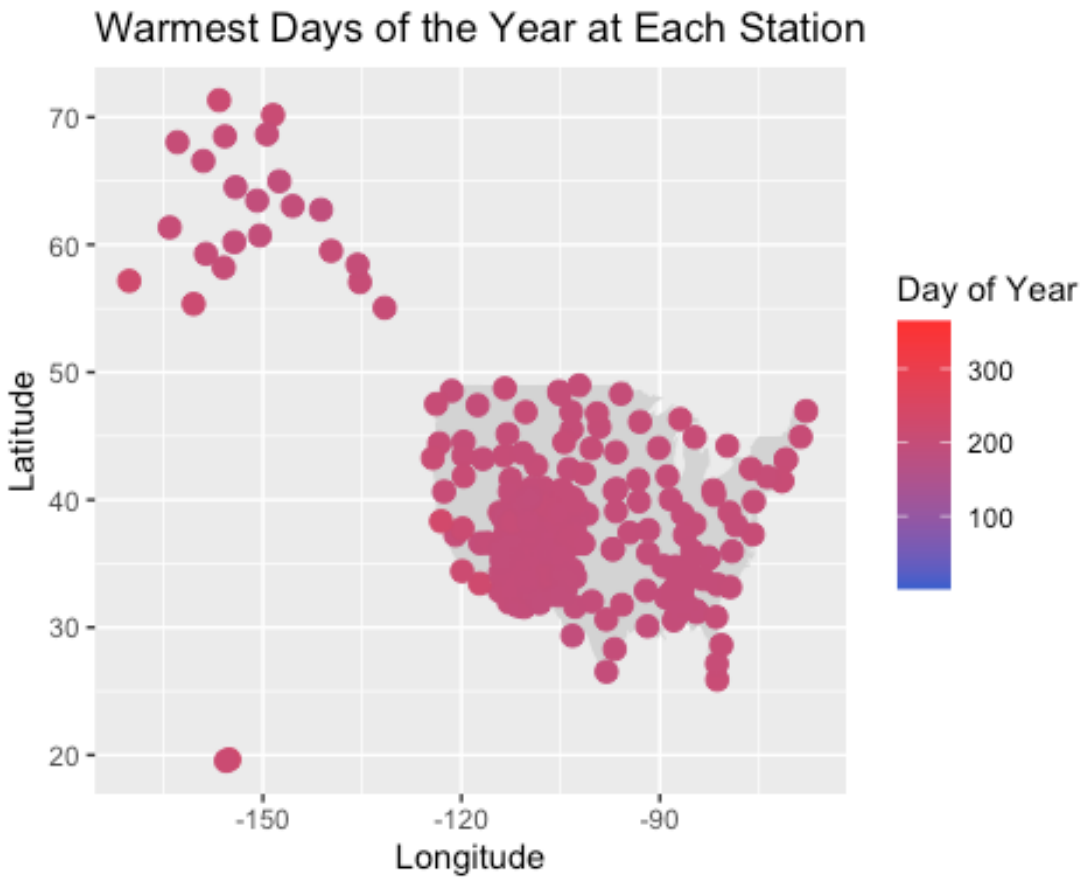
```r
  temp[i, 3] <- max(temps)
  temp[i, 2] <- min(temps)
  temp[i, 1] <- stations.to.analyze[i, 'Station_ID']
}

#Create dataframe of temperature data. Each row is a station. Add lon/lat
temp.data <-
  as.data.frame(cbind(
    temp,
    stations.to.analyze$Longitude,
    stations.to.analyze$Latitude
  ))

#Ensure all data is numeric
temp.data <- temp.data %>%  mutate_all(as.numeric)

#Assign column names for plotting and interpolating
colnames(temp.data) <-
  c("ID",
    "Min.Temp",
    "Max.Temp",
    'Max.Day',
    'Min.Day',
    "LONGITUDE",
    "LATITUDE")

#Acces data only for plot4
map.1 <- data.frame(temp.data$LONGITUDE,
                    temp.data$LATITUDE,
                    temp.data$Min.Temp)

#Create plot object
plot4 <- plot_predictions(map.1)

#plot
plot4 +
  labs(title = 'Coldest Days of the Year at Each Station',
              colour = 'Day of Year') +

  geom_point(data = temp.data,
  aes(x = LONGITUDE,
  y = LATITUDE,
  colour = Min.Day),
  size = 3) +

  scale_colour_gradient(
    low = "royalblue3",
    high = "firebrick1",
    limits = c(1, 365)
```

```
  )
#> Scale for colour is already present.
#> Adding another scale for colour, which will replace the existing scale.
```



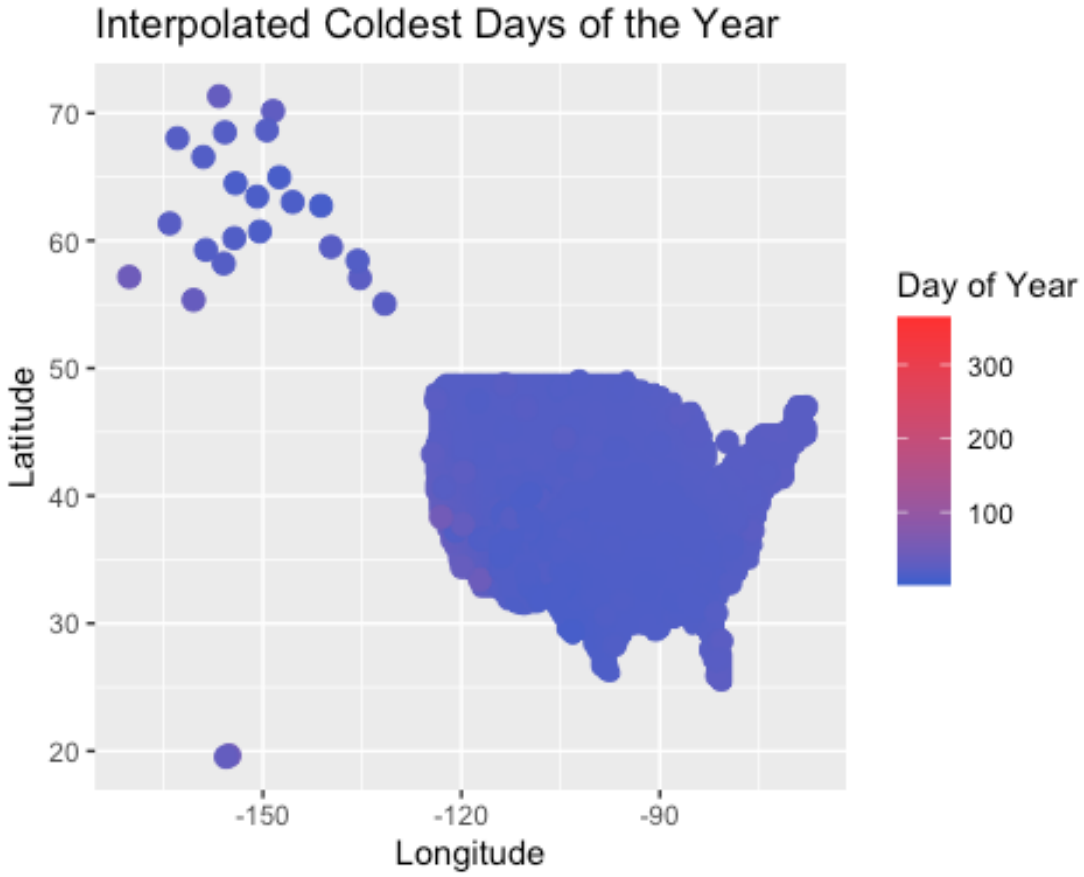

```
#Data for plot5
map.2 <-
  data.frame(temp.data$LONGITUDE,
             temp.data$LATITUDE,
             temp.data$Max.Temp)

#Create object and plot
plot5 <- plot_predictions(map.2)
plot5 +
  labs(title = 'Warmest Days of the Year at Each Station', colour =
                'Day of Year') +

  geom_point(data = temp.data,
             aes(x = LONGITUDE,
                 y = LATITUDE,
                 colour = Max.Day),
             size = 3) +
```

```
  scale_colour_gradient(
    low = "royalblue3",
    high = "firebrick1",
    limits = c(1, 365)
  )
#> Scale for colour is already present.
#> Adding another scale for colour, which will replace the existing scale.
```



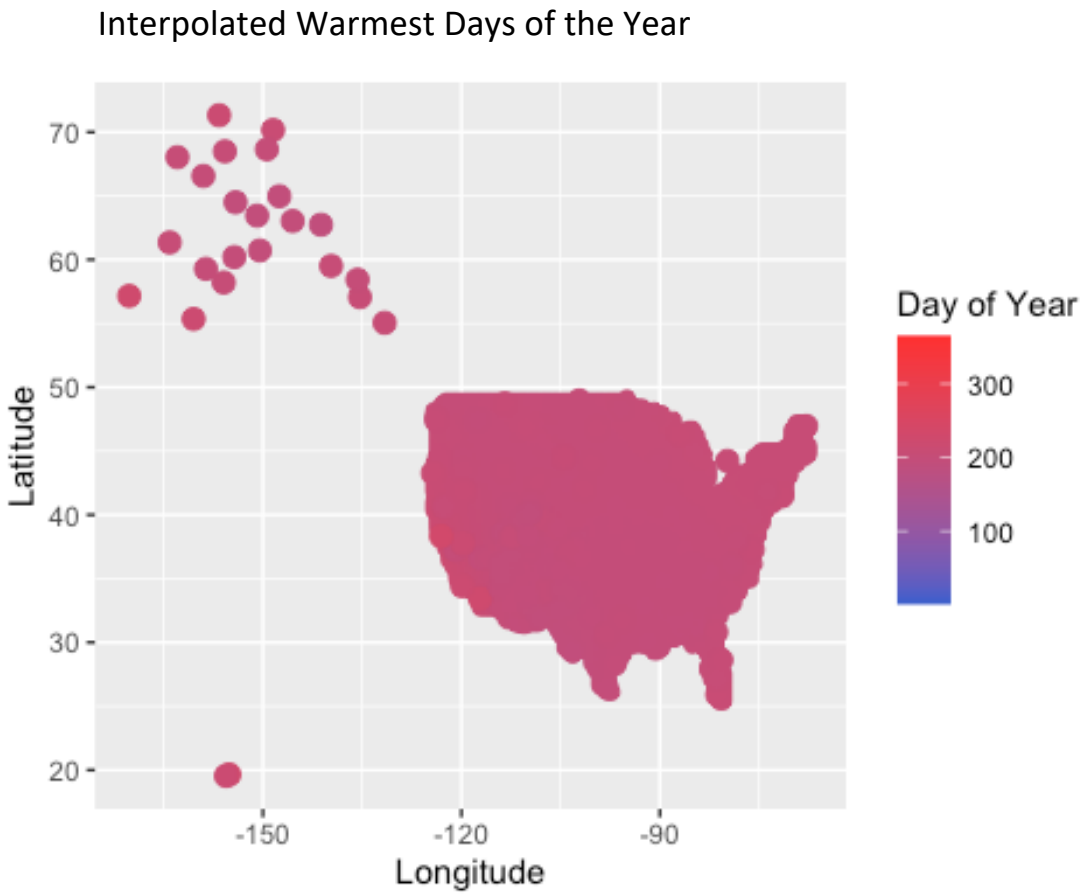

```
#Interpolate data for plot 6
map.3 <- interpolate_data('Min.Day', dat=temp.data)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

#Create
plot6 <- plot_predictions(map.3)

#Plot
plot6 + labs(title = 'Interpolated Coldest Days of the Year',
             colour ='Day of Year') +

  geom_point(data = temp.data,
  aes(x = LONGITUDE,
      y = LATITUDE,
```

```
      colour = Min.Day),
      size = 3) +

  scale_colour_gradient(
    low = "royalblue3",
    high = "firebrick1",
    limits = c(1, 365)
  )
#> Scale for colour is already present.
#> Adding another scale for colour, which will replace the existing scale.
```




```
#Interpolate for plot 7
map.4 <- interpolate_data("Max.Day", dat=temp.data)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

#Create & Plot
plot7 <- plot_predictions(map.4)
plot7 + labs(title = 'Warmest Days of the Year at Each Station',
             colour = 'Day of Year') +

  geom_point(data = temp.data,
  aes(x = LONGITUDE,
```

```
      y = LATITUDE,
      colour = Max.Day),
      size = 3) +

  scale_colour_gradient(
    low = "royalblue3",
    high = "firebrick1",
    limits = c(1, 365)
  )
#> Scale for colour is already present.
#> Adding another scale for colour, which will replace the existing scale.
```




##

## Statistical Analysis Used To Estimate Warmest/Coldest Days

We use a linear regression model with sine and cosine terms as predictors. This approach is commonly used in modeling periodic data, such as seasonal temperature variations.

### Linear Regression Model

The linear regression model assumes a linear relationship between the independent variables (years) and the dependent variable (days). We also transformed our data into two different cos and sin terms.

$$Y = Intercept + \beta_1 X_{sin} + \beta_2 X_{cos} + \varepsilon$$

- $Y$: Response variable (temperature).
- $X_1, X_2$: Transformed day predictor variables.
- $\beta_0, \beta_1, \beta_2$: Coefficients to be estimated.
- $\varepsilon$: Error term.

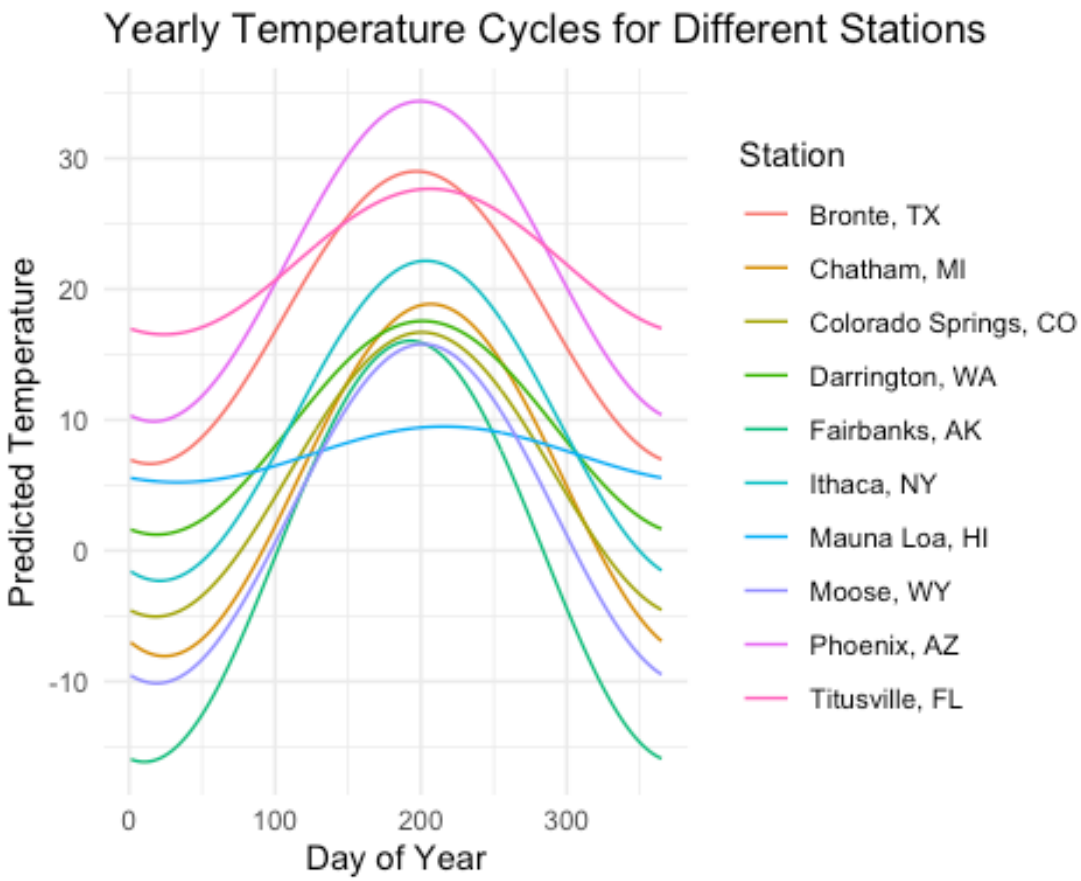
### Interpretation of Coefficients

The coefficients associated with the sine and cosine terms indicate the magnitude and phase of the periodic components of temperature variation. They provide insights into the amplitude and phase shift of the seasonal variation in temperature.

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.

```
#Stations we'd like to analyze
station_ids <-
  c(26494, 53156, 4223, 64756, 4131, 3072, 54810, 53007, 92821, 21514)

#Assigning station names for legend
names(station_ids) <-
  c(
    "Fairbanks, AK",
    "Phoenix, AZ",
    "Darrington, WA",
    "Ithaca, NY", #Hello Ithaca!
    "Moose, WY",
    "Bronte, TX",
    "Chatham, MI",
    "Colorado Springs, CO",
    "Titusville, FL",
    "Mauna Loa, HI"
  )

#initialize a dataframe
all_data <- data.frame()

# Loop through each station ID, get the yearly cycle data, and append to the
dataframe
for (i in 1:length(station_ids)) {
  yearly_cycle_data <- yearly_cycle(station_ids[i])
  yearly_cycle_data$Station <- names(station_ids)[i]
  all_data <- rbind(all_data, yearly_cycle_data)
}

# Plot the yearly cycles grouped by station
ggplot(all_data, aes(x = Day, y = Predicted_Temp, color =
as.factor(Station))) +
```

```
  geom_line() +
  labs(title = "Yearly Temperature Cycles for Different Stations",
       x = "Day of Year",
       y = "Predicted Temperature") +
  scale_color_discrete(name = "Station") +
  theme_minimal()
```



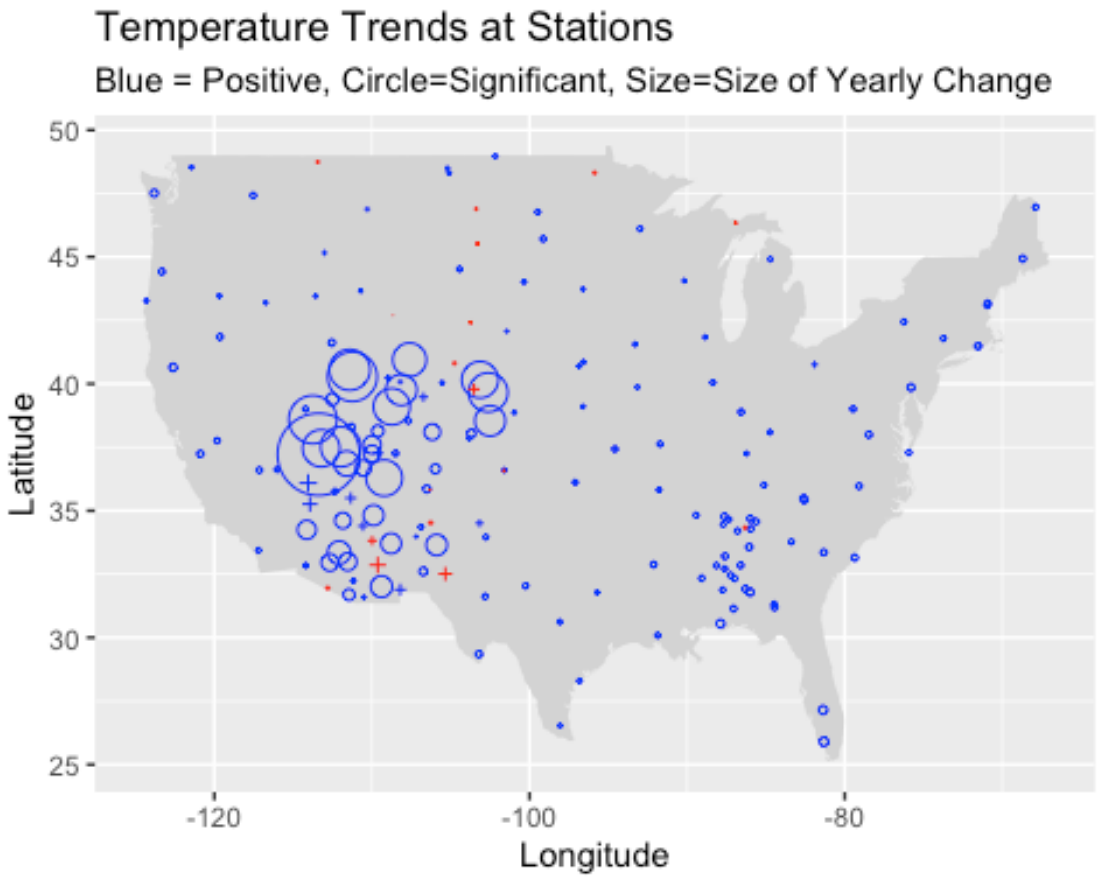

5. Estimate the trend over the years for each station, in units of degrees Fahrenheit per year, and plot the trend values on a map. Indicate visually on your map which of the trends are statistically significant.

   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.

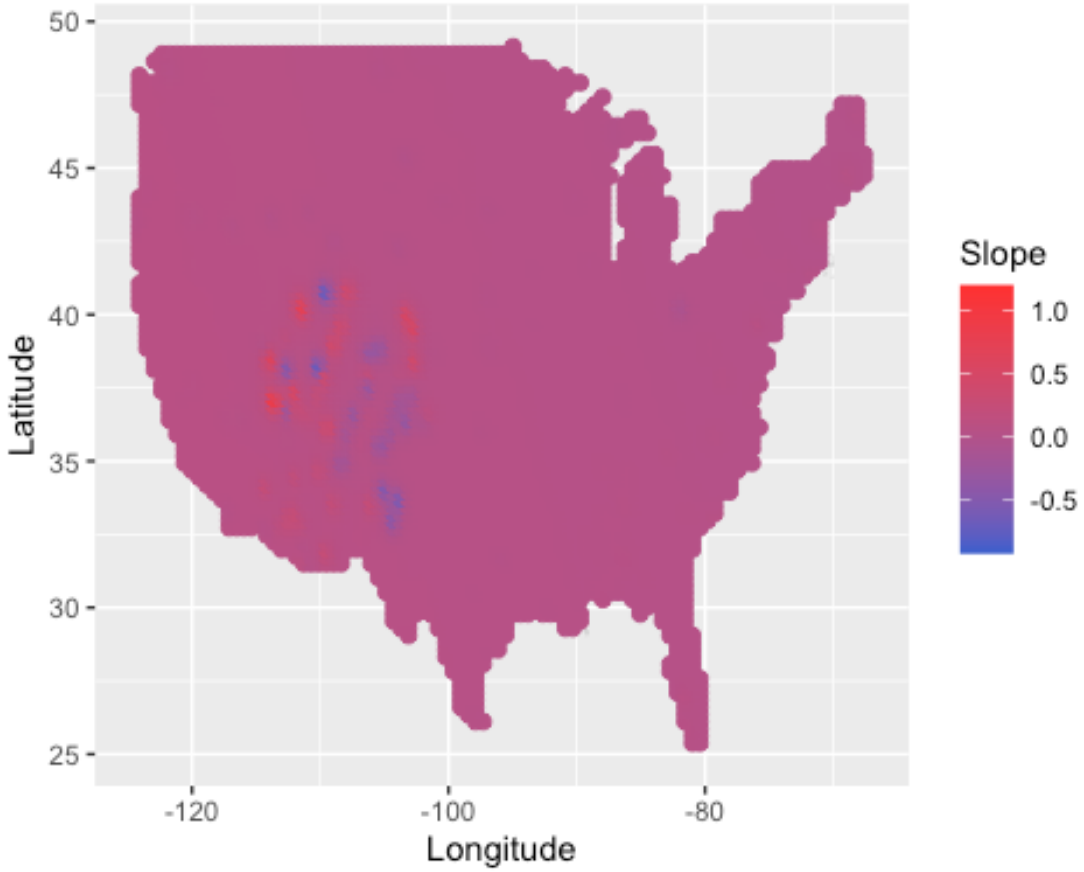
```
# Loop through each station ID, get the yearly cycle data, and append to the
dataframe
stations.analyze <- stations[-c(213,235,236,237),] #Remove incomplete data

#Initialize a dataframe
```

```r
all_data_temp <- data.frame(matrix(NA,ncol=4,nrow=nrow(stations.analyze)))

for (i in 1:nrow(stations.analyze)) {
  trend_data <- temp_trends(stations.analyze[i, 'Station_ID'])
  all_data_temp[i,] <- trend_data
}

#Assign column names
colnames(all_data_temp) <- c("Station_ID","Intercept","Slope", "PValue")

#Add Lat and Lon for interpolation
all_data_temp$LONGITUDE <-stations.analyze$Longitude
all_data_temp$LATITUDE <- stations.analyze$Latitude

#Only Contiguous USA
us_map <- map_data("usa")
inside <- map.where('usa', all_data_temp[,5:6])

inside.logical <- ifelse(is.na(inside), FALSE, TRUE)
contig.data.temp <- all_data_temp[inside.logical,]

#Differentiate between significant or not
contig.data.temp$Point_Shape <- ifelse(contig.data.temp$PValue <= 0.05, 1, 3)

contig.data.temp <- contig.data.temp %>%  mutate_all(as.numeric)

contig.data.temp$Point_Shape <- ifelse(contig.data.temp$PValue <= 0.05, 1, 3)

# Create a ggplot object for the US map
ggplot() +
  geom_polygon(data = us_map,
               aes(x = long, y = lat, group = group),
               fill = "lightgray") +

  geom_point(data = contig.data.temp,
             aes(x = as.numeric(LONGITUDE), y = as.numeric(LATITUDE)),
             col = ifelse(sign(contig.data.temp$Slope) > 0, "blue", "red"),
             size = 10 * contig.data.temp$Slope,
             pch = contig.data.temp$Point_Shape) +

  scale_color_identity() +

  labs(title ='Temperature Trends at Stations',
    x = "Longitude",
    y = "Latitude",
    subtitle = "Blue = Positive, Circle=Significant, Size=Size of Yearly
Change"
  )
```




```
#Significant data only for interpolation
significant.data <- contig.data.temp |> filter(PValue < 0.05 & Slope >-1)

#Data for interpolation (Original dataframe doesn't work)
significant.data.2 <- as.data.frame(
  cbind(as.numeric(significant.data$LONGITUDE),
        as.numeric(significant.data$LATITUDE),
        as.numeric(significant.data$Slope)))

colnames(significant.data.2) <- c("LONGITUDE","LATITUDE","Slope")

#Interpolate data, create plot, and plot
map.5 <- interpolate_data('Slope', significant.data.2)
#> Assuming columns 1 and 2 of locs are (longitude,latidue) in degrees

#Cluster of datapoints that have far too low of slopes. Assume due to vacancy
#of data and remove
plot8 <- plot_predictions(map.5)

plot8 +   scale_colour_gradient(
```

```
    low = "royalblue3",
    high = "firebrick1",
    limits = c(-0.92, 1.2), name="Slope"
  )
#> Scale for colour is already present.
#> Adding another scale for colour, which will replace the existing scale.
```




## Statistical Analysis Used To Estimate Temp Trends

We use a linear regression model with year and month as predictors.

### Linear Regression Model

The linear regression model assumes a linear relationship between the independent variables (year/month) and the dependent variable (temperature). Our data itself follows a cyclical pattern, but we are expecting a linear change in temperatures per month/year.

$$Y = Intercept + \beta_{Year}X_1 + \beta_{Jan}X_2 + \cdots + \beta_{Nov}X_{13} + \beta_{Dec}X_{14} + \cdots + \varepsilon$$

- $Y$: Response variable (temperature).
- $X_1, X_2, \ldots, X_3, X_{14}$: Predictor variables for year, and month. (Month is 1 or 0)
- $\beta_0, \beta_{Year}, \beta_{Jan}, \ldots, \beta_{Dec}$: Coefficients to be estimated.
- $\varepsilon$: Error term.

### Interpretation of Coefficients

Our coefficients imply that with all other variables held constant, we expect a change in the dependent variable equal to that of the coefficient if the dependent variable increases by one (continuous), or if the dependent variable is equal to 1.

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.

*Data from National Oceanic and Atmospheric Administration's National Centers for Environmental Information.*

*We have an average rate of change of 0.03 degrees celsius per year over the past 20 years across the Contiguous United States. This value was obtained by converting the Data from the NOAA to fahrenheit, differencing each year, and taking the average of the values of said differences.*

*Compared to our estimate of the yearly change, obtained using column means for the {code}{map.5} dataframe, we have an average rate of change of 0.01 degrees celsius per year. Compared to our value obtained from the NOAA, we are relatively close, only being 0.02 degrees/year off.*