Easy Access to SST Time Series for Alaska's Management Areas

Jordan Watson (jordan.watson@noaa.gov), Matt Callahan (MCallahan@psmfc.org)

2/22/2021

Goal: This document describes easy access to daily SST data averaged across common spatial strata for Alaska

We would love feedback on how to make these products most useful. And we are happy to help answer questions!

Introduction

The AFSC and AKFIN are making satellite sea surface temperature (SST) data easier to access for commonly used spatial strata.

Satellite SST data from NOAA Coral Reef Watch (https://coralreefwatch.noaa.gov/product/5km/index_5km_sst.php) are downloaded daily for the entire spatial extent of the EEZ around Alaska. The time series begins 1985-01-01 and extends through the present, with a 1-2 day latency period (data sourced from Coral Reef Watch except Jan-Mar 1985 which were downloaded directly from NESDIS). The raw data (i.e., full gridded dataset) are available to users with an AKFIN database account and can be easily queried from Oracle, R, or Python.

Alternatively, we describe an easy data access tool that requires no accounts, no passwords, no VPN - just internet. We demonstrate using R. Daily SST data have been averaged within several dozen commonly used spatial strata, including the NMFS management areas, ecosystem regions (ESRs), BSIERP regions, and ADF&G statistical areas.

The greatest convenience of this tool is that no files need to be locally stored, nor do time series need to be updated and appended to existing data files. You will need the R package **httr** to pull data from a URL. Additional packages **tidyverse** and **lubridate** are used here to demonstrate plotting and manipulation but the object retrieved using **httr** can easily be manipulated using base R instead.

```
library(httr) # For pulling data via a URL
library(tidyverse) # Data manipulation
library(lubridate) # Date formatting
library(sp) # For maps
library(rgdal) # For maps
library(ggrepel) #For maps
require(gridExtra) #For maps
```

Data extraction using this AKFIN web service is as simple as the statement below, which will query the time series of daily temperatures for NMFS area 640. The data can be saved as an object for manipulation or piped directly into downstream functions. If you prefer base R instead of the tidyverse, you will still need to load **dplyr** for the bind_rows() function to work. Alternatively, you'll need a base R solution to reformat the input data.

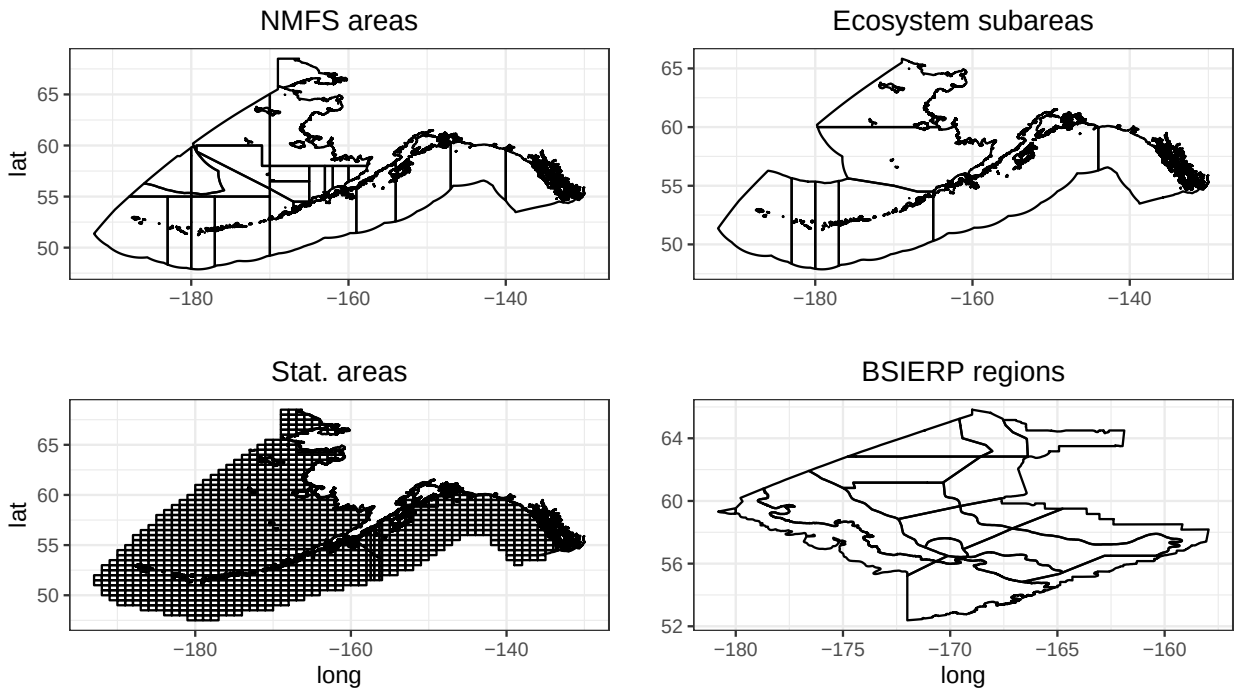


Figure 1: Map of four commonly used spatial strata for which SST data are averaged daily.

```
head(httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_a
  bind_rows)
```

```
## # A tibble: 6 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    6.06 640      1985-01-01T12:00:00Z 1985  001
## 2    5.89 640      1985-01-02T12:00:00Z 1985  002
## 3    5.78 640      1985-01-03T12:00:00Z 1985  003
## 4    5.76 640      1985-01-04T12:00:00Z 1985  004
## 5    5.78 640      1985-01-05T12:00:00Z 1985  005
## 6    5.92 640      1985-01-06T12:00:00Z 1985  006
```

With this tool, users can easily incorporate SST data into stock assessments and other processes. For example, one could plot a time series of average summer SST for NMFS areas 640 and 650.

```
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=6
  bind_rows %>%
  mutate(MONTH=month(as_date(READ_DATE))) %>% # Extract month from the date
  filter(MONTH==6 | MONTH==7 | MONTH==8) %>% # Filter summer months
  group_by(YEAR,NMFSAREA)%>%
    summarize(SST=mean(MEANSST))%>% # Average the data by year and area.
  ggplot(aes(as.numeric(YEAR),SST)) +
  geom_line() +
  facet_wrap(~NMFSAREA, nrow=2) +
  xlab("Year") +
  theme_bw()
```

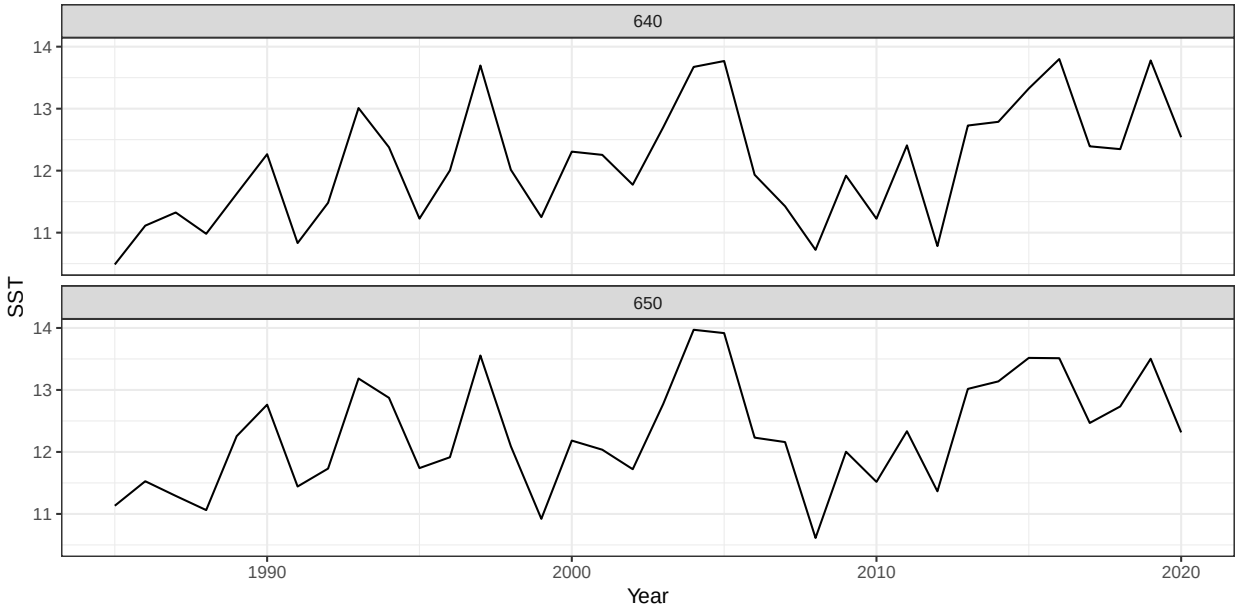


Figure 2: June-August SST 1985-2020 in EGOA NMFS areas.

Web Service - How It Works

AKFIN has created web services to streamline access to average daily temperatures within each of the strata. In the database back-end at AKFIN, satellite SST files are downloaded daily as two separate netCDF files - one for positive and one for negative longitudes. The daily data are clipped spatially by merging them with a spatial look-up table that encompasses the exclusive economic zone (EEZ) of Alaska. The result is a gridded dataset that includes 212,813 SST records per day.

```
#View strata included in the lookup table
lkp <- readRDS("Data/crwsst_spatial_lookup_table.RDS")
str(lkp)
```

```
## 'data.frame':    212814 obs. of  11 variables:
##  $ id           : int  1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 ...
##  $ statefed     : Factor w/ 3 levels "FED","INTL","STATE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ stat_area    : num  686800 686800 686800 686800 686800 ...
##  $ depth        : num  -54 -54 -54 -53 -53 -51 -50 -50 -50 -50 ...
##  $ longitude    : num  -169 -169 -169 -169 -169 ...
##  $ latitude     : num  68.5 68.5 68.5 68.5 68.5 ...
##  $ nmfsarea     : Factor w/ 25 levels "400","508","509",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ bsierp_id    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ bsierp_name  : Factor w/ 16 levels "AK peninsula",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ Ecosystem    : chr  NA NA NA NA ...
##  $ Ecosystem_sub: chr  NA NA NA NA ...
```

Currently, daily mean SST for each NMFS area can only be queried individually. For the Bering Sea and Gulf of Alaska, the query filters only data where water depth is between 10 and 200m. For the Aleutian Islands, a depth filter is not implemented. Analysts that are interested in data for different depth ranges, custom spatial bounds, or aggregated NMFS areas can contact the authors and we will arrange for your request.

The web service enables a query using a URL, where the URL the query parameters. You could paste the URL below into a browser and view the output there if desired. Below, we query the URL "https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=640", where "nmfs_area_crw_avg_sst?" is the name of the dataset. This is the daily SST data averaged by **nmfs_area**. A "?" separates the dataset name from the query criteria. The default behavior is to pull the single most recent datum record. Here we tell R that the native format is json.

```
httr::content(httr::GET("https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=64
  bind_rows
```

```
## # A tibble: 1 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    4.75 640      2021-03-13T12:00:00Z 2021  072
```

```
#By default the URL returns a list. We use "%>% bind_rows" here to convert to a data frame (tibble)
```

Time Series

To query a date range, specify "start_date" and "end_date", "read_date", or "dates_back" parameters. Separate parameters with an "&". Most users will want the entire time series, which starts on 1985-01-01. To query the entire time series, specify "start_date" & "end_date". "end_date" must be included, but if you do not know the most recent date of the time series, you can choose an end date some time in the future and it will query all of the data that exist.

```
data <- httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs
  bind_rows

head(data)
```

```
## # A tibble: 6 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    6.06 640      1985-01-01T12:00:00Z 1985  001
## 2    5.89 640      1985-01-02T12:00:00Z 1985  002
## 3    5.78 640      1985-01-03T12:00:00Z 1985  003
## 4    5.76 640      1985-01-04T12:00:00Z 1985  004
## 5    5.78 640      1985-01-05T12:00:00Z 1985  005
## 6    5.92 640      1985-01-06T12:00:00Z 1985  006
```

The full time series yields more than 13,000 rows of data per area (i.e., daily data from 1985-01-01 to present).

```
dim(data)
```

```
## [1] 13217     5
```

Any time range can be chosen with "start_date" and "end_date". For example, SST in NMFS area 640 in 1987.

```
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=64
  bind_rows %>%
  mutate(date=as_date(READ_DATE)) %>%
  ggplot(aes(date,MEANSST)) +
  geom_line()+
  theme_bw()
```

Figure 3: SST for 1987 in NMFS area 640.

You can query a specific date with "read_date". For example SST in MFS 640 on Y2K.

```
#Query the day after your date of interest because omitting the time component in read_date misses that
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=64
  bind_rows
```

```
## # A tibble: 1 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    4.94 640      1999-12-31T12:00:00Z 1999  365
```

You can specify a number of days prior to any date using a "days_back" parameter specification. For example the three days before Y2K.

```
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=64
  bind_rows
```

```
## # A tibble: 3 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    5.43 640      1999-12-29T12:00:00Z 1999  363
## 2    5.13 640      1999-12-30T12:00:00Z 1999  364
## 3    4.94 640      1999-12-31T12:00:00Z 1999  365
```

If “read_date” is not specified, “days_back” returns the most recent SSTs. Here are SSTs for the last three days in NMFS 640.

```
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=64
  bind_rows
```

```
## # A tibble: 3 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    4.79 640      2021-03-11T12:00:00Z 2021  070
## 2    4.74 640      2021-03-12T12:00:00Z 2021  071
## 3    4.75 640      2021-03-13T12:00:00Z 2021  072
```

Spatial Extents

To query multiple areas, separate the values by a comma. For example to query NMFS areas 640 and 650 (Southeast Alaska outside waters).

```
httr::content(httr::GET("https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=64
  bind_rows
```

```
## # A tibble: 2 x 5
##   MEANSST NMFSAREA READ_DATE            YEAR  JULIAN
##     <dbl> <chr>    <chr>                <chr> <chr>
## 1    4.75 640      2021-03-13T12:00:00Z 2021  072
## 2    5.57 650      2021-03-13T12:00:00Z 2021  072
```

The **Ecosystem_sub** field in our look-up table contains each of the ecosystem regions. Points outside these ecosystem regions are listed as NA for this field.

```
unique(lkp$Ecosystem_sub)
```

```
## [1] NA                          "Northern Bering Sea"
## [3] "Western Gulf of Alaska"    "Eastern Gulf of Alaska"
## [5] "Southeastern Bering Sea"   "Eastern Aleutians"
## [7] "Central Aleutians"         "Western Aleutians"
```

To query the data for the “Southeastern Bering Sea”, for example, add “ecosystem_sub=Southeastern%20Bering%20Sea”, where spaces are filled by “%20”.

```
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/ecosystem_sub_crw_avg_sst?ecosyst
  bind_rows
```

```
## # A tibble: 1 x 5
##   MEANSST ECOSYSTEM_SUB           READ_DATE            YEAR  JULIAN
##     <dbl> <chr>                   <chr>                <chr> <chr>
## 1    1.06 Southeastern Bering Sea 2021-03-13T12:00:00Z 2021  072
```

Putting the pieces together - data can be queried directly from AKFIN and saved, manipulated, or visualized directly. Here we query and plot the full time series for the Eastern Gulf of Alaska and for the Eastern Aleutian Islands.

```
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/ecosystem_sub_crw_avg_sst?ecosyst
  bind_rows %>%
  mutate(date=as_date(READ_DATE)) %>%
  ggplot(aes(date,MEANSST)) +
  geom_line() +
  facet_wrap(~ECOSYSTEM_SUB)
```

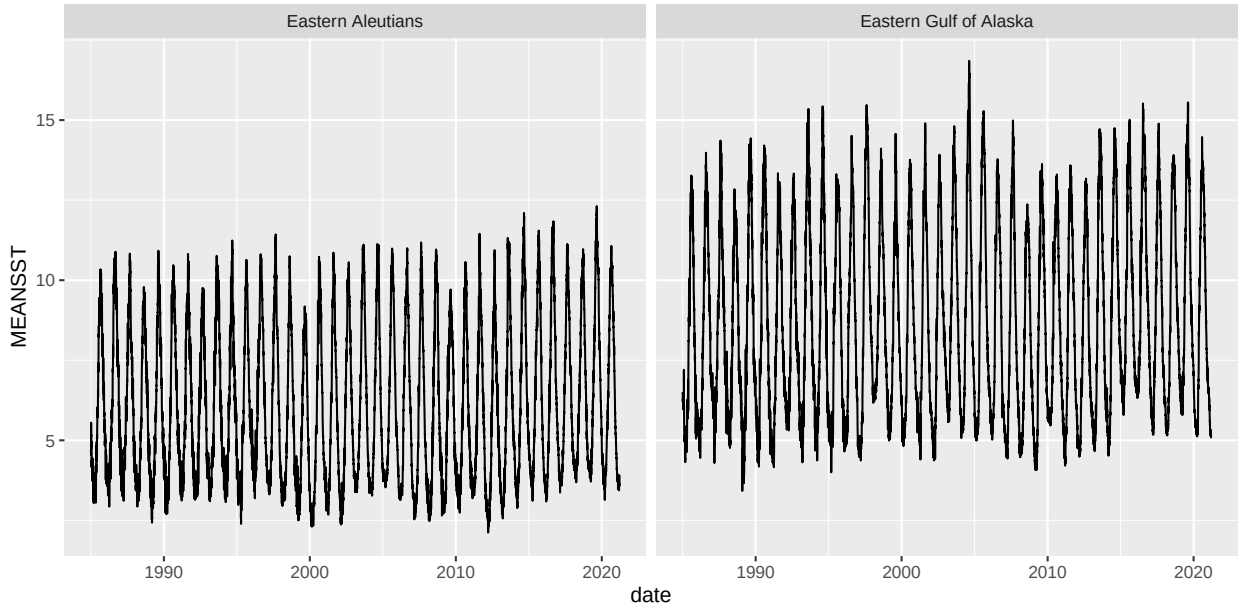


Figure 4: Eastern GOA and Eastern Aleutians SST from 1985 - Present.

One could query and summarize data by month (week, year, etc) by simply grouping and summarizing the data in-line. In this case, we should have removed the most recent year, whose data are incomplete.

```
#Note that year is a character so it needs to be converted to an integer for continuous plotting.
httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=6
  bind_rows %>%
  mutate(date=as_date(READ_DATE),
         YEAR=as.numeric(YEAR)) %>%
  group_by(YEAR,NMFSAREA) %>%
  summarise(meansst=mean(MEANSST)) %>%
  ggplot(aes(YEAR,meansst)) +
  geom_line() +
  geom_smooth() +
  facet_wrap(~NMFSAREA, nrow=2)+
  theme_bw()
```

Marine Heatwave Bonus Material

With the daily time series for a spatial stratum it's easy to use the heatwaveR package to characterize marine heatwaves (MHWs) in your data. The package vignette is fantastic (https://robwschlegel.github.io/heatwaveR/). We demonstrate a few quick examples below using our time series for NMFS area 640 (Eastern

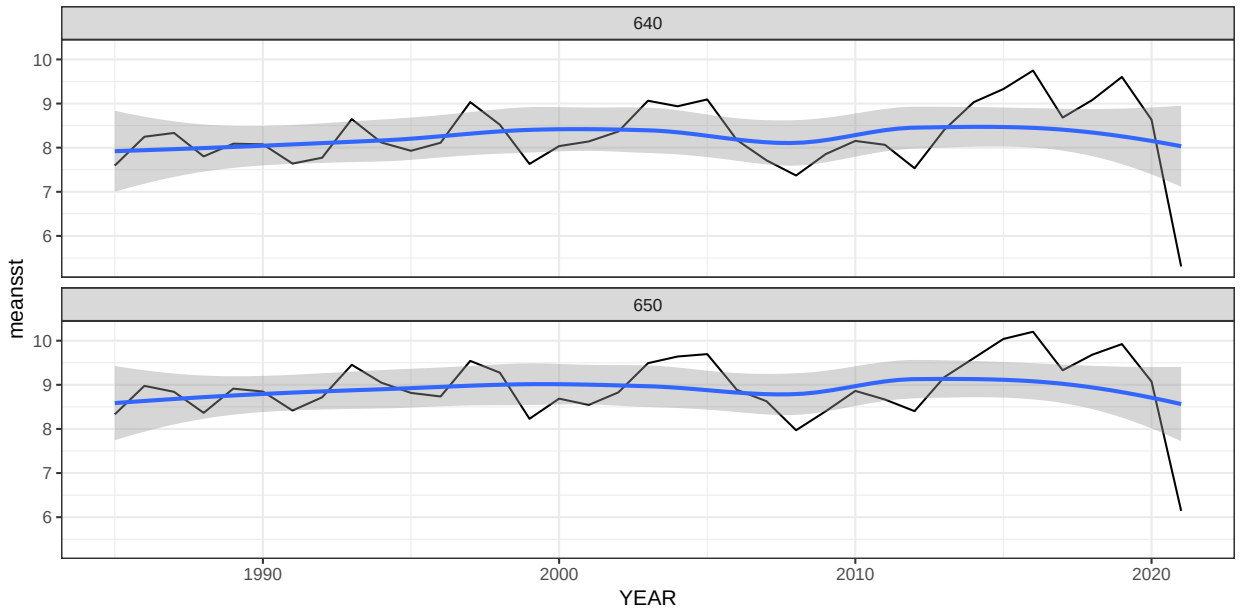


Figure 5: Annual average SST for NMFS areas 640 and 650.

GOA). To keep this document short we only demonstrate plotting a single stratum but please contact us for additional examples with multiple strata if you are interested. We're happy to help.

In this example we save SST data as an object in the first code chunk but you could alternatively embed the data query into the MHW code chunk that follows and avoid creating an intermediate object.

```
# Save the SST data to an object
updateddata <- httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_s
  bind_rows %>%
  mutate(date=as_date(READ_DATE)) %>%
  data.frame %>%
  dplyr::select(date,meansst=MEANSST,NMFSAREA) #I just simplify my data frame
```

The detect_event() and the ts2clm() functions from **heatwaveR** will calculate MHW status status relative to your baseline period. We specify the earliest 30-years as a baseline by convention. The ts2clm() function generates marine heatwave thresholds from your baseline period. The detect_event() function creates a list of 2 data frames. The first, **climatology**, adds a series of columns to your data frame, including the seasonal climatology and MHW thresholds as well as flags for whether or not a record falls within a MHW. The second list object, **event**, includes summary information about each of the MHW events (e.g., max intensity, duration, start and end dates, etc.).

```
library(heatwaveR)
mhw <- detect_event(ts2clm(updateddata %>%
                             rename(t=date,temp=meansst) %>%
                             arrange(t), climatologyPeriod = c("1985-01-01", "2014-12-31"))) #Specify

str(mhw)
```

```
## List of 2
##  $ climatology:Classes 'data.table' and 'data.frame':    13221 obs. of  9 variables:
##   ..$ doy             : int [1:13221] 1 2 3 4 5 6 7 8 9 10 ...
```

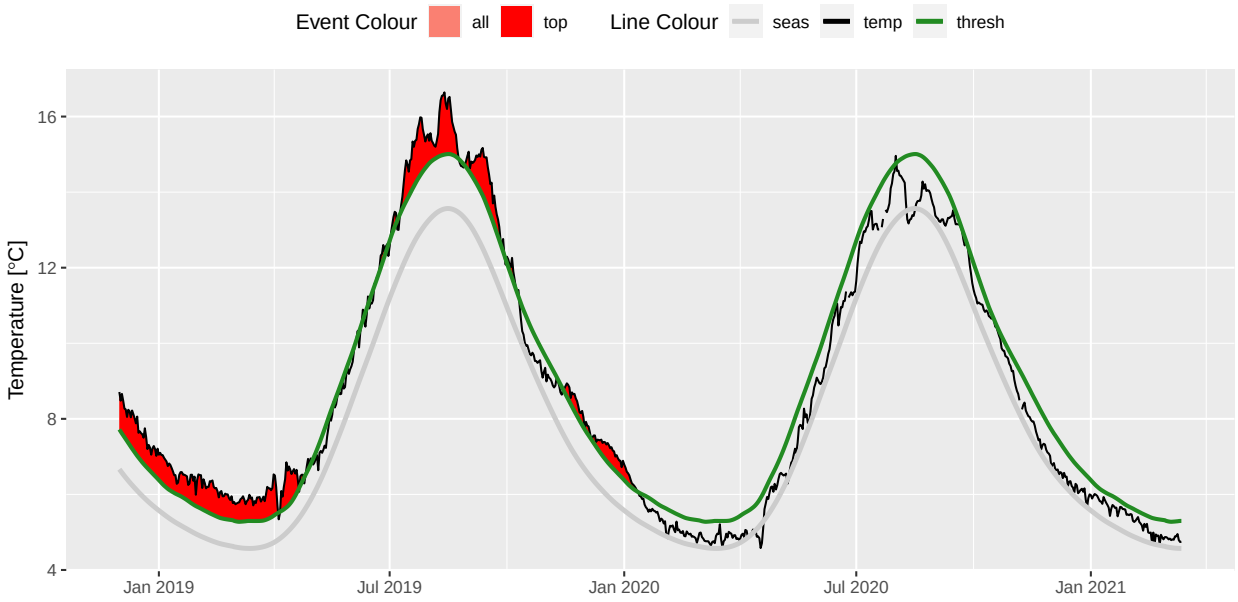
```
##   ..$ t                : Date[1:13221], format: "1985-01-01" "1985-01-02" ...
##   ..$ temp             : num [1:13221] 6.06 5.89 5.78 5.76 5.78 5.92 6 5.92 5.85 5.83 ...
##   ..$ seas             : num [1:13221] 5.57 5.55 5.52 5.5 5.47 ...
##   ..$ thresh           : num [1:13221] 6.38 6.34 6.3 6.27 6.24 ...
##   ..$ threshCriterion  : logi [1:13221] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..$ durationCriterion: logi [1:13221] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..$ event            : logi [1:13221] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..$ event_no         : int [1:13221] NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, ".internal.selfref")=<externalptr>
##  $ event      :Classes 'data.table' and 'data.frame':    54 obs. of  22 variables:
##   ..$ event_no                      : int [1:54] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ index_start                   : int [1:54] 26 675 718 736 771 806 1141 1153 1726 1742 ...
##   ..$ index_peak                    : int [1:54] 31 688 722 740 781 808 1143 1155 1727 1744 ...
##   ..$ index_end                     : int [1:54] 36 693 729 759 784 816 1145 1157 1730 1749 ...
##   ..$ duration                      : num [1:54] 11 19 12 24 14 11 5 5 5 8 ...
##   ..$ date_start                    : Date[1:54], format: "1985-01-26" "1986-11-06" ...
##   ..$ date_peak                     : Date[1:54], format: "1985-01-31" "1986-11-19" ...
##   ..$ date_end                      : Date[1:54], format: "1985-02-05" "1986-11-24" ...
##   ..$ intensity_mean                : num [1:54] 1.13 1.58 1.11 1.12 0.94 ...
##   ..$ intensity_max                 : num [1:54] 1.76 2.08 1.28 1.46 1.4 ...
##   ..$ intensity_var                 : num [1:54] 0.325 0.299 0.107 0.232 0.251 ...
##   ..$ intensity_cumulative          : num [1:54] 12.4 30 13.3 26.8 13.2 ...
##   ..$ intensity_mean_relThresh      : num [1:54] 0.392 0.414 0.251 0.359 0.25 ...
##   ..$ intensity_max_relThresh       : num [1:54] 1.026 0.929 0.414 0.709 0.72 ...
##   ..$ intensity_var_relThresh       : num [1:54] 0.3268 0.3012 0.0972 0.2359 0.2556 ...
##   ..$ intensity_cumulative_relThresh: num [1:54] 4.31 7.86 3.01 8.61 3.5 ...
##   ..$ intensity_mean_abs            : num [1:54] 6.07 9.13 6.9 6.31 5.66 ...
##   ..$ intensity_max_abs             : num [1:54] 6.7 9.54 7.17 6.81 6.08 ...
##   ..$ intensity_var_abs             : num [1:54] 0.314 0.32 0.188 0.281 0.243 ...
##   ..$ intensity_cumulative_abs      : num [1:54] 66.8 173.5 82.8 151.6 79.2 ...
##   ..$ rate_onset                    : num [1:54] 0.1761 0.0686 0.0814 0.1654 0.0769 ...
##   ..$ rate_decline                  : num [1:54] 0.1909 0.1899 0.0527 0.0424 0.1855 ...
##   ..- attr(*, ".internal.selfref")=<externalptr>
```

To create the common "flame" plots for MHWs, extract the climatology object from the list and plot it. Below we will filter the data since 2018-12-01 for NMFS area 640. This code is directly from the heatwaveR vignette.

```
mhw_clim <- mhw$climatology %>% filter(t>=as.Date("2018-12-01"))  #Extract the MHW data

ggplot(data = mhw_clim, aes(x = t)) +
  geom_flame(aes(y = temp, y2 = thresh, fill = "all"), show.legend = T) +
  geom_flame(data = mhw_clim, aes(y = temp, y2 = thresh, fill = "top"),  show.legend = T) +
  geom_line(aes(y = temp, colour = "temp")) +
  geom_line(aes(y = thresh, colour = "thresh"), size = 1.0) +
  geom_line(aes(y = seas, colour = "seas"), size = 1.2) +
  scale_colour_manual(name = "Line Colour",
                      values = c("temp" = "black",
                                 "thresh" =  "forestgreen",
                                 "seas" = "grey80")) +
  scale_fill_manual(name = "Event Colour",
                    values = c("all" = "salmon",
                               "top" = "red")) +
  scale_x_date(date_labels = "%b %Y") +
```

```
  guides(colour = guide_legend(override.aes = list(fill = NA))) +
  labs(y = expression(paste("Temperature [", degree, "C]")), x = NULL) +
  theme(legend.position="top")
```



Tweak the image slightly to add the intensity categories of the MHWs. Again, the plotting code is verbatim from the heatwaveR vignettes. To illustrate the categories better here we cherry-pick an example from the Northern Bering Sea. We have also consolidated some code to reduce the number of intermediate objects and we display data from 2019-01-01 to 2019-12-31.

```
#  Here we'll use an example where we do not save the SST data as a separate object first, simply embed
clim_cat <- (detect_event(ts2clm(
  httr::content(httr::GET('https://apex.psmfc.org/akfin/data_marts/akmp/ecosystem_sub_crw_avg_sst?ecosy
                                  bind_rows %>%
                                  mutate(date=as_date(READ_DATE)) %>%
                                  data.frame %>%
                                  dplyr::select(t=date,temp=MEANSST) %>%
                                  arrange(t), climatologyPeriod = c("1985-01-01", "2014-12-31"))))$cli
  dplyr::mutate(diff = thresh - seas,
                thresh_2x = thresh + diff,
                thresh_3x = thresh_2x + diff,
                thresh_4x = thresh_3x + diff) %>%
  filter(t>=as.Date("2019-01-01") & t<=as.Date("2019-12-31")) # Select the time period to display.

# Set line colours
lineColCat <- c(
  "Temperature" = "black",
  "Climatology" = "gray20",
  "Threshold" = "darkgreen",
  "2x Threshold" = "darkgreen",
  "3x Threshold" = "darkgreen",
  "4x Threshold" = "darkgreen"
  )
```

```r
# Set category fill colours
fillColCat <- c(
  "Moderate" = "#ffc866",
  "Strong" = "#ff6900",
  "Severe" = "#9e0000",
  "Extreme" = "#2d0000"
  )


ggplot(data = clim_cat, aes(x = t, y = temp)) +
  geom_flame(aes(y2 = thresh, fill = "Moderate")) +
  geom_flame(aes(y2 = thresh_2x, fill = "Strong")) +
  geom_flame(aes(y2 = thresh_3x, fill = "Severe")) +
  geom_flame(aes(y2 = thresh_4x, fill = "Extreme")) +
  geom_line(aes(y = thresh_2x, col = "2x Threshold"), size = 0.7, linetype = "dashed") +
  geom_line(aes(y = thresh_3x, col = "3x Threshold"), size = 0.7, linetype = "dotdash") +
  geom_line(aes(y = thresh_4x, col = "4x Threshold"), size = 0.7, linetype = "dotted") +
  geom_line(aes(y = seas, col = "Climatology"), size = 0.7) +
  geom_line(aes(y = thresh, col = "Threshold"), size = 0.7) +
  geom_line(aes(y = temp, col = "Temperature"), size = 0.6) +
  scale_colour_manual(name = NULL, values = lineColCat,
                      breaks = c("Temperature", "Climatology", "Threshold",
                                 "2x Threshold", "3x Threshold", "4x Threshold")) +
  scale_fill_manual(name = NULL, values = fillColCat, guide = FALSE) +
  scale_x_date(date_labels = "%b %Y") +
  guides(colour = guide_legend(override.aes = list(linetype = c("solid", "solid", "solid",
                                                                "dashed", "dotdash", "dotted"),
                                                   size = c(0.6, 0.7, 0.7, 0.7, 0.7, 0.7)))) +
  labs(y = "Temperature [°C]", x = NULL) +
  theme(legend.position="top")
```