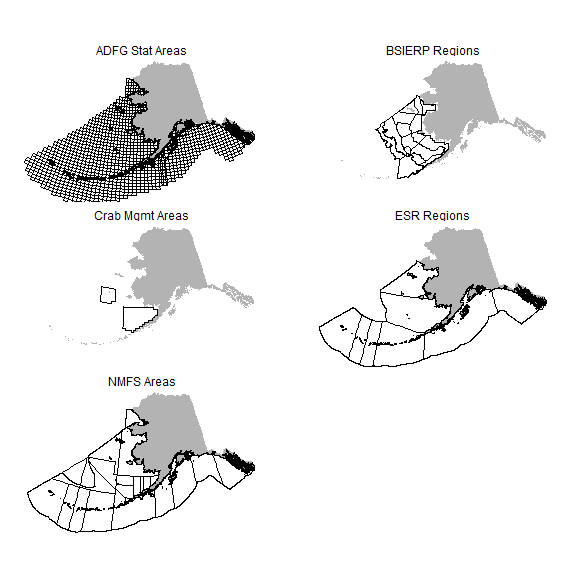
# Methods & Results

Methods & Results Satellite data For this study, two daily satellite sea surface temperature (SST) products were used. In both cases, data were accessed via NOAA ERDDAP servers (Simons [2020](#ref-Simons2020)) and downloaded as netcdf files within the Oracle database backend at the Alaska Fisheries Information Network (AKFIN), maintained by the Pacific States Marine Fisheries Commission. The SST data are publicly available but by ingesting them into the AKFIN backend, they can be seamlessly merged, behind the NOAA firewall, with confidential fishery-dependent data sets like observer data, vessel monitoring system (VMS) data, and fish tickets. Both of the SST products provide gap-free data each day. The MUR SST data set is provided by JPL NASA (JPL MUR MEaSUREs Project. 2015) and is available from June 2002 - present and are accessed via the NOAA CoastWatch West Coast Node ERDDAP server (coastwatch.pfeg.noaa.gov/erddap/griddap/jplMURSST41.html). These data are provided across a 0.01° x 0.01° (1 km) spatial grid. Meanwhile, the CRW SST data set covers a 0.05° x 0.05° (5 km) spatial grid and these data are obtained from the NOAA Coral Reef Watch Program (<https://coralreefwatch.noaa.gov/product/5km/index_5km_sst.php>) through the NOAA PacIOOS Program ERDDAP server (pae-paha.pacioos.hawaii.edu/erddap/griddap/dhw\_5km.html) from April 1985 – present. Additional data (January – March 1985) were downloaded via public ftp link from NESDIS (<ftp://ftp.star.nesdis.noaa.gov/pub/socd/mecb/crw/data/coraltemp/v1.0/nc/1985/>). Both the MUR and CRW data sets typically have a 1-2 day latency period. Both the MUR and the CRW data sets have native formats with longitudes ranging from -180 to +180. Because the spatial extent for Alaska waters includes the International Date Line, the daily data are downloaded via two separate operations each day. One operation downloads the negative longitude data from 46°N to 68.8°N and -180°E to -130°E and the second operation downloads the positive longitude data from 47.5°N to 60.0°N and 164°E to 180°E. These downloads are merged and then clipped to spatial regions of interest within the exclusive economic zone surrounding Alaska, yielding 212,813 SST records per day.

Figure YY: AKFIN data flow diagram

*Spatial strata* State and Federal waters of Alaska include numerous spatial strata that are relevant to fisheries management, ecology, and individual species distributions. For example, the Alaska Department of Fish & Game (ADF&G) divides Alaskan waters into nearly 1,800 statistical areas, many of which are 0.5° latitude by 1.0° longitude boxes. Meanwhile, the National Marine Fisheries Service (NMFS) divides the same waters into only 25 management areas. However, these regulatory strata are inconsistent with ecological stratifications (Eastern Bering Sea, Gulf of Alaska, and the Aleutian Islands) identified for the same waters. These ecosystem regions, even when subdivided, do not necessarily align with spatial strata identified for individual fish or crab stocks, so stock assessment scientists and fishery managers are often interested in yet further customized spatial boundaries. Thus it is not surprising that different users of environmental information like SST may want those data aggregated or clipped to a different (or multiple) spatial boundaries. To develop operational data products across Alaska’s suite of spatial strata, we undertook extensive point-in-polygon geoprocessing operations to apportion the individual latitude-longitude coordinates for both the MUR and CRW SST spatial grids to each of the polygons from a suite of shapefiles (ADF&G management areas, NMFS management areas, Ecosystem regions [from NMFS Ecosystem Status Reports], Bering Sea Integrated Ecosystem Research Program [BSIERP] regions, Bristol Bay red king crab management areas, St. Matthew blue king crab management areas) (Fig. YY). The spatial extent of Alaska includes more than 200,000 data records daily for the CRW data set and more than 1 million records daily for the MUR data set. To avoid repeating the computationally intensive point-in-polygon operations, we created spatial lookup tables that are stored in the backend of the AKFIN Oracle database system. Thus, as data are downloaded daily from ERDDAP servers across the spatial extent of Alaska’s waters, each SST record is matched via a database join to the spatial strata in which it falls instead of via repeated point-in-polygon operations.



Accessing the data We demonstrate two general methods for accessing the data stored in AKFIN. The first method, customized web services (web APIs), is ideal for accessing time series of aggregated data (e.g. daily SST averaged across a spatial stratum or multiple spatial strata) and for queries less than about 100,000 records. This approach leverages a simplified data access point (url) that is outside of the AKFIN firewall and requires no user login.. The second method, direct database access, requires a login to the AKFIN database backend and relies on SQL to extract either aggregated data summaries or larger gridded data sets (e.g., millions of data records). In the sections that follow, we demonstrate data queries using custom web services and by using direct SQL and R access. For each case, we illustrate the utility of operational data workflows by piping these data into R functions for calculating marine heatwaves (MHWs). Customized Web Service (Web API) For queries that are likely to be repeated often or to become part of an automated process, customized web services offer a particularly efficient data access option. Note that the queries demonstrated here are simply examples, but additional customized queries can be developed for users by contacting this study’s authors. These web services require no accounts, no passwords, no VPN - just internet. For example, a user could type the url <https://apex.psmfc.org/akfin/data_marts/akmp/nmfs_area_crw_avg_sst?nmfs_area=640> into a web browser and they will be served with the most recent day’s average SST data for NMFS area 640. Such urls can be readily embedded into programming applications (e.g., R, Python) and for multiple or different spatial strata as well as for specified time series. An additional convenience is that web services allow users to query time series without storing data locally which is particularly helpful for operations that would typically append data to existing files. To access web services using R Statistical Software, 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.

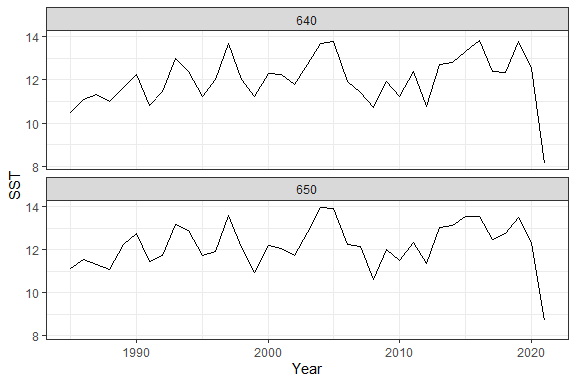
Data extraction using this AKFIN web service is as simple as the statement below, which will query the time series of daily temperatures for NMFS area 640. The data can be saved as an object for manipulation or piped directly into downstream functions. If you prefer base R instead of the tidyverse, you 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.

library(httr) # For pulling data via a URL  
library(tidyverse) # Data manipulation  
library(lubridate) # Date formatting  
  
head(httr::content(httr::GET('https://apex.psmfc.org/akfin/data\_marts/akmp/nmfs\_area\_crw\_avg\_sst?nmfs\_area=640&start\_date=19850101&end\_date=20220101'), type = "application/json") %>%   
 bind\_rows)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 6.06 | 640 | 1985-01-01T12:00:00Z | 1985 | 001 |
| 5.89 | 640 | 1985-01-02T12:00:00Z | 1985 | 002 |
| 5.78 | 640 | 1985-01-03T12:00:00Z | 1985 | 003 |
| 5.76 | 640 | 1985-01-04T12:00:00Z | 1985 | 004 |
| 5.78 | 640 | 1985-01-05T12:00:00Z | 1985 | 005 |
| 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=640,650&start\_date=19850101&end\_date=20220101'), type = "application/json") %>%   
 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()



June-August SST 1985-2020 in EGOA NMFS areas.

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=640"), type = "application/json")%>%   
 bind\_rows #By default the URL returns a list. "bind\_rows" convert to data frame/tibble

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 8.62 | 640 | 2021-06-07T12:00:00Z | 2021 | 158 |

# 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\_area=640&start\_date=19850101&end\_date=20220101'), type = "application/json") %>%   
 bind\_rows  
  
head(data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 6.06 | 640 | 1985-01-01T12:00:00Z | 1985 | 001 |
| 5.89 | 640 | 1985-01-02T12:00:00Z | 1985 | 002 |
| 5.78 | 640 | 1985-01-03T12:00:00Z | 1985 | 003 |
| 5.76 | 640 | 1985-01-04T12:00:00Z | 1985 | 004 |
| 5.78 | 640 | 1985-01-05T12:00:00Z | 1985 | 005 |
| 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] 13307 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=640&start\_date=19870101&end\_date=19880101'), type = "application/json") %>%   
 bind\_rows %>%   
 mutate(date=as\_date(READ\_DATE)) %>%   
 ggplot(aes(date,MEANSST)) +   
 geom\_line()+  
 theme\_bw()



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 day's reading.  
httr::content(httr::GET('https://apex.psmfc.org/akfin/data\_marts/akmp/nmfs\_area\_crw\_avg\_sst?nmfs\_area=640&read\_date=20000101'), type = "application/json") %>%   
 bind\_rows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 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=640&read\_date=20000101&days\_back=2'), type = "application/json") %>%   
 bind\_rows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 5.43 | 640 | 1999-12-29T12:00:00Z | 1999 | 363 |
| 5.13 | 640 | 1999-12-30T12:00:00Z | 1999 | 364 |
| 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=640&days\_back=2'), type = "application/json") %>%   
 bind\_rows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 8.20 | 640 | 2021-06-05T12:00:00Z | 2021 | 156 |
| 8.34 | 640 | 2021-06-06T12:00:00Z | 2021 | 157 |
| 8.62 | 640 | 2021-06-07T12:00:00Z | 2021 | 158 |

# 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=640,650"), type = "application/json")%>%   
 bind\_rows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | NMFSAREA | READ\_DATE | YEAR | JULIAN |
| 8.62 | 640 | 2021-06-07T12:00:00Z | 2021 | 158 |
| 9.08 | 650 | 2021-06-07T12:00:00Z | 2021 | 158 |

The **Ecosystem\_sub** fields available for query include the regions within the Eastern Bering Sea, Aleutian Islands, and Gulf of Alaska.

#View strata included in the lookup table  
lkp <- readRDS("Data/crwsst\_spatial\_lookup\_table.RDS")   
  
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"

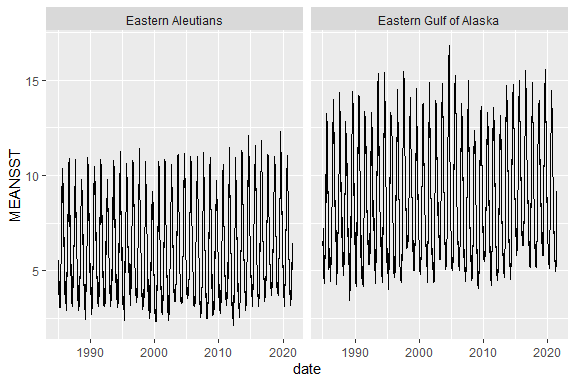
To query the data for the “Southeastern Bering Sea”, for example, add “ecosystem\_sub=Southeastern%20Bering%20Sea”, where spaces are filled by “%20”.

httr::content(httr::GET('https://apex.psmfc.org/akfin/data\_marts/akmp/ecosystem\_sub\_crw\_avg\_sst?ecosystem\_sub=Southeastern%20Bering%20Sea'), type = "application/json") %>%  
 bind\_rows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEANSST | ECOSYSTEM\_SUB | READ\_DATE | YEAR | JULIAN |
| 4.89 | Southeastern Bering Sea | 2021-06-07T12:00:00Z | 2021 | 158 |

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.

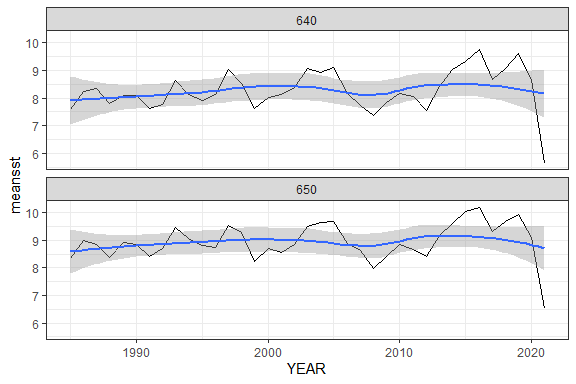
httr::content(httr::GET('https://apex.psmfc.org/akfin/data\_marts/akmp/ecosystem\_sub\_crw\_avg\_sst?ecosystem\_sub=Eastern%20Gulf%20of%20Alaska,Eastern%20Aleutians&start\_date=19850101&end\_date=20220101'), type = "application/json") %>%   
 bind\_rows %>%   
 mutate(date=as\_date(READ\_DATE)) %>%   
 ggplot(aes(date,MEANSST)) +   
 geom\_line() +   
 facet\_wrap(~ECOSYSTEM\_SUB)



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=640,650&start\_date=19850101&end\_date=20220101'), type = "application/json") %>%   
 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()



Annual average SST for NMFS areas 640 and 650.

# Piping web service queries into marine heatwave calculations

With the daily time series for a spatial stratum, implementation of the heatwaveR package is straightforward. We encourage readers to explore the functionality described in the heatwaveR vignettes, from which the following examples are generated (**???**). We demonstrate a few simple examples below using web services to query the SST time series for NMFS area 640 (Eastern GOA).

For illustration purposes, we save SST data as an object in the first code chunk but alternatively, the data query could be piped directly into the MHW code chunk that follows to avoid creating an intermediate object.

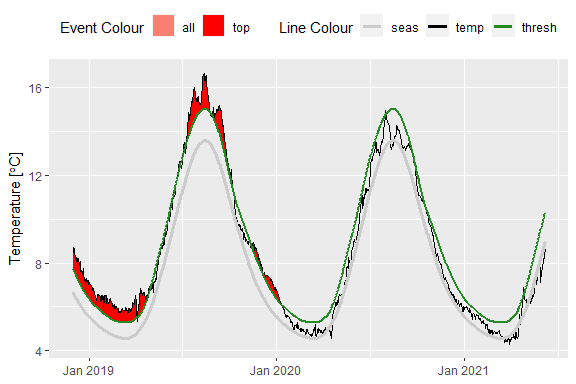
updateddata <- httr::content(httr::GET('https://apex.psmfc.org/akfin/data\_marts/akmp/nmfs\_area\_crw\_avg\_sst?nmfs\_area=640&start\_date=19850101&end\_date=20211231'), type = "application/json") %>%   
 bind\_rows %>%   
 mutate(date=as\_date(READ\_DATE)) %>%   
 data.frame %>%   
 dplyr::select(date,meansst=MEANSST,NMFSAREA) #simplify the data frame for clarity.

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 baseline period

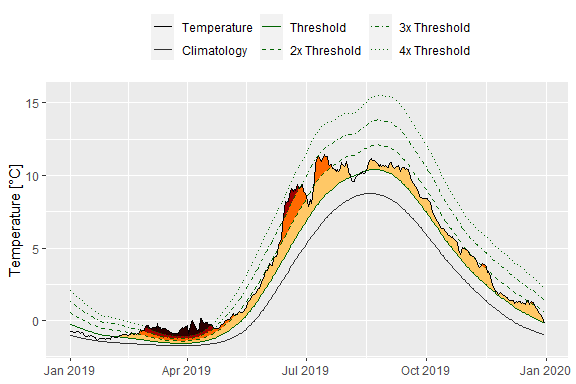
To create the common “flame” plots for MHWs, extract the climatology object from the list and plot it. Below we will filter the data since 2018-12-01 for NMFS area 640.

# Plotting code directly from 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. 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 embedding it into the ts2clm() function.  
  
clim\_cat <- (detect\_event(ts2clm(  
httr::content(httr::GET('https://apex.psmfc.org/akfin/data\_marts/akmp/ecosystem\_sub\_crw\_avg\_sst?ecosystem\_sub=Northern%20Bering%20Sea&start\_date=19850101&end\_date=20211231'),   
type = "application/json") %>%   
 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"))))$climatology %>% #Specify the baseline period.  
 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.  
  
# Plotting code directly from heatwaveR vignette.  
# Set line colours  
lineColCat <- c(  
 "Temperature" = "black",  
 "Climatology" = "gray20",  
 "Threshold" = "darkgreen",  
 "2x Threshold" = "darkgreen",  
 "3x Threshold" = "darkgreen",  
 "4x Threshold" = "darkgreen"  
 )  
  
# 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")



**Oracle database queries**

Some data users prefer the flexibility and transparency of querying raw gridded data directly from the Oracle server. This is particularly useful for larger queries (e..g, millions of records) or for exploring data across a suite of different spatial extents (e.g., custom depth ranges, shapefiles, etc.). As we note in the web services section, such custom queries can also be automated by working with AFSC and AKFIN staff. However, for users that prefer to code their own queries, we provide several examples here. Notably, to query directly from the database, users will need an AKFIN database account, which can be provided by contacting the authors of this document.

This section is not meant to serve as a SQL tutorial. Rather, its purpose is to orient users to the structure of the database related to the SST data and lookup tables. We assume that users interested in querying the database directly via Oracle (e.g., SQL Developer) or through odbc connections from R or Python are already acquainted with the coding and configuration settings. However, interested users can contact the authors for assistance establishing such connections or custom SQL queries.

The gridded SST data (Fig. BB) are stored within the AFSC schema on the AKFIN Oracle database and the primary key linking the lookup tables (Fig. AA) with the gridded data is the ID field. In the lookup table, it is simply “ID” and in the data table it is “CRW\_ID”.

select \* from afsc.erddap\_crw\_sst\_spatial\_lookup   
 where rownum<=5

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | STATE\_FED | STAT\_AREA | DEPTH | LONGITUDE | LATITUDE | NMFSAREA | BSIERP\_ID | BSIERP\_NAME | ECOSYSTEM | ECOSYSTEM\_SUB | CRAB |
| 58040 | FED | 525902 | -126 | -152.175 | 59.175 | 630 | NA | NA | Gulf of Alaska | Western Gulf of Alaska | NA |
| 58041 | FED | 525902 | -130 | -152.125 | 59.175 | 630 | NA | NA | Gulf of Alaska | Western Gulf of Alaska | NA |
| 58042 | FED | 525902 | -119 | -152.075 | 59.175 | 630 | NA | NA | Gulf of Alaska | Western Gulf of Alaska | NA |
| 58043 | FED | 525902 | -108 | -152.025 | 59.175 | 630 | NA | NA | Gulf of Alaska | Western Gulf of Alaska | NA |
| 58044 | STATE | 515906 | -115 | -151.975 | 59.175 | 630 | NA | NA | Gulf of Alaska | Western Gulf of Alaska | NA |

Figure BB. Screenshot of the AFSC.ERDDAP\_CRW\_SST table.

select \* from afsc.erddap\_crw\_sst   
 where rownum<=5

|  |  |  |
| --- | --- | --- |
| READ\_DATE | CRW\_ID | TEMP |
| 1985-04-03 12:00:00 | 66728 | -1.72 |
| 1985-04-03 12:00:00 | 66729 | -1.72 |
| 1985-04-03 12:00:00 | 66730 | -1.71 |
| 1985-04-03 12:00:00 | 66731 | -1.70 |
| 1985-04-03 12:00:00 | 66732 | -1.69 |

Figure AA. Screenshot of a CRW SST query within SQL Developer. Several columns reveal ‘NA’ because the particular latitude - longitude coordinates shown do not fall within any spatial strata represented by those columns.

The following query demonstrates the primary key relationship between the data and lookup tables. In this case, we query SST (“TEMP”) data that fall within a crab management area and we add a field for “Year” (Fig. CC).

select read\_date,  
 temp,  
 to\_char(read\_date,'YYYY') as Year,  
 crab  
from afsc.erddap\_crw\_sst a  
 INNER JOIN (select \* from afsc.erddap\_crw\_sst\_spatial\_lookup  
 where crab <> 'NA') b  
 ON a.crw\_id =b.id  
where rownum<=5

|  |  |  |  |
| --- | --- | --- | --- |
| READ\_DATE | TEMP | YEAR | CRAB |
| 1993-01-03 12:00:00 | 1.54 | 1993 | stmatts |
| 1993-01-02 12:00:00 | 1.73 | 1993 | stmatts |
| 1993-01-01 12:00:00 | 1.72 | 1993 | stmatts |
| 1993-01-04 12:00:00 | 1.65 | 1993 | stmatts |
| 1993-01-05 12:00:00 | 1.85 | 1993 | stmatts |

Figure CC. Screenshot of a CRW SST query for records that fall within a crab management area.

dbFetch(dbSendQuery(con,  
 paste0("select read\_date,  
 round(avg(temp),2) as sst,  
 crab  
 from afsc.erddap\_crw\_sst a  
 INNER JOIN (select \*   
 from afsc.erddap\_crw\_sst\_spatial\_lookup  
 where crab = 'bb') b  
 ON a.crw\_id = b.id   
 group by   
 crab,  
 read\_date"))) %>%   
 ggplot(aes(READ\_DATE,SST)) +   
 geom\_line() +   
 geom\_smooth()

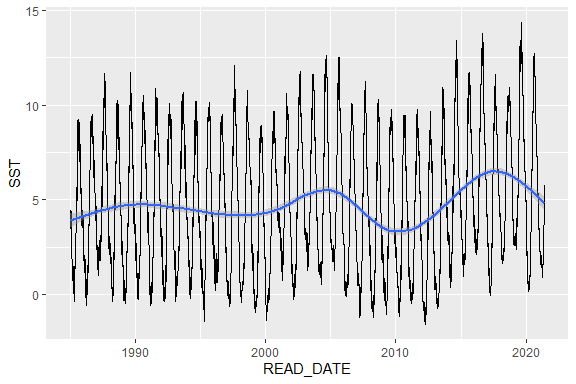


Figure DD. Screenshot of a plotted query of Bristol Bay crab management area SST data averaged daily and plotted with default smoothing.

# References

Simons B. 2020. ERDDAP..