heatwaveR

SM

## Libraries

shhh = suppressPackageStartupMessages # It's a library, so shhh!  
library(heatwaveR) # For detecting MHWs  
shhh(library(sf))  
shhh(library(lubridate)) # Useful functions for dealing with dates  
library(ggplot2) # The preferred library for data visualisation  
library(tidync) # For easily dealing with NetCDF data  
library(rerddap) # For easily downloading subsets of data  
library(doParallel) # For parallel processing

Loading required package: foreach

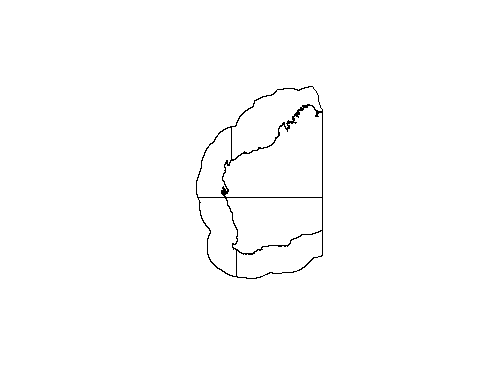
Loading required package: iterators

Loading required package: parallel

library(stringr) # for string wrapping

## Read in the DPIRD Bioregions shapefile

setwd(getwd()) # set to git wd  
wa\_sf = st\_read(".//Data//DPIRD\_Bioregions\_DPIRD\_095.shp", quiet = TRUE) %>%  
 st\_transform(crs = 4326) # OISST is WGS84  
plot(st\_geometry(wa\_sf)) # plot the geoms



bbox = st\_bbox(wa\_sf) # get the bounding box for heatwaveR  
print(bbox)

xmin ymin xmax ymax

109.23349 -38.52944 129.03041 -11.21521

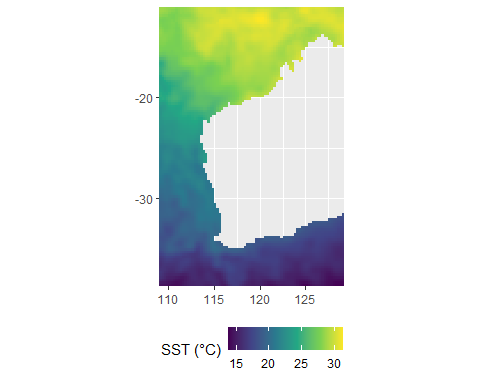
## Download OISST data

# Date download range by start and end dates per year  
dl\_years = data.frame(date\_index = 1:6, # can only dl in chunks of ~7 years  
 start = c("1982-01-01", "1990-01-01",   
 "1998-01-01", "2006-01-01", "2014-01-01", "2020-01-01"),  
 end = c("1989-12-31", "1997-12-31",   
 "2005-12-31", "2013-12-31", "2019-12-31", "2023-12-31"))

# This function downloads and prepares data based on user provided start and end dates  
  
#### MAY TAKE SOME TIME ###  
###########################  
OISST\_sub\_dl = function(time\_df){  
 OISST\_dat = rerddap::griddap(datasetx = "ncdcOisst21Agg\_LonPM180",  
 url = "https://coastwatch.pfeg.noaa.gov/erddap/",   
 time = c(time\_df$start, time\_df$end),   
 zlev = c(0, 0),  
 latitude = c(bbox[2], bbox[4]), # from bbox  
 longitude = c(bbox[1], bbox[3]), # from bbox  
 fields = "sst")$data %>%   
 dplyr::mutate(time = base::as.Date(stringr::str\_remove(time, "T12:00:00Z")))%>%  
 dplyr::rename(t = time, temp = sst, lon = longitude, lat = latitude) %>%   
 dplyr::select(lon, lat, t, temp) %>%   
 stats::na.omit()  
}

#year\_range = OISST\_sub\_dl(dl\_years[1,]) if you want to test a chunk  
  
## this will take some time to download##  
########################################  
OISST\_data = dl\_years %>%   
 dplyr::group\_by(date\_index) %>%   
 dplyr::group\_modify(~OISST\_sub\_dl(.x)) %>%   
 dplyr::ungroup() %>%   
 dplyr::select(lon, lat, t, temp)  
#base::saveRDS(OISST\_data, file = ".//Data//Downloads//WA\_temp.Rds")  
  
# Save the data as an .Rds file   
base::saveRDS(OISST\_data, file = ".//Data//Downloads//WA\_temp.Rds")

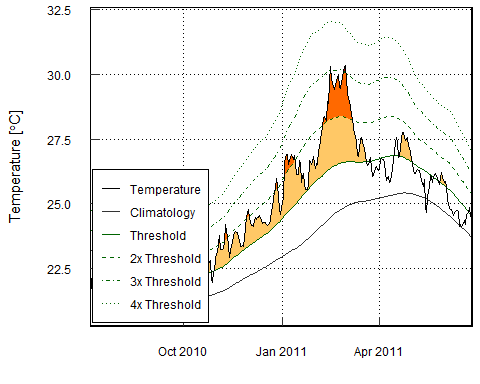
OISST\_data = readRDS(".//Data//Downloads//WA\_temp.Rds")  
  
## quick plot of a day to check  
OISST\_data %>%   
 dplyr::filter(t == "1994-12-01") %>%   
 ggplot2::ggplot(aes(x = lon, y = lat)) +  
 ggplot2::geom\_tile(aes(fill = temp)) +  
 # ggplot2::borders() + # Activate this line to see the global map  
 ggplot2::scale\_fill\_viridis\_c() +  
 ggplot2::coord\_quickmap(expand = F) +  
 ggplot2::labs(x = NULL, y = NULL, fill = "SST (°C)") +  
 ggplot2::theme(legend.position = "bottom")



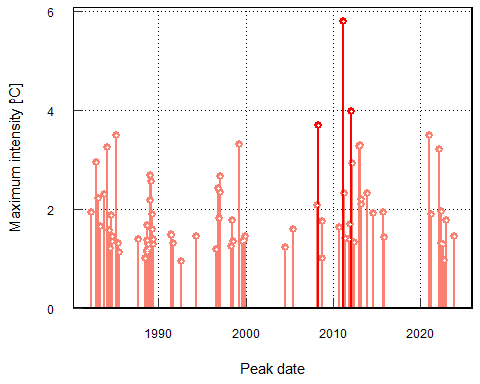
## Heatwave detection at sharkbay

selected\_data = OISST\_data %>%  
 dplyr::filter(lon == 112.875 & lat == -25.625) # for one coordinate  
  
plt\_clim = heatwaveR::ts2clm(data = selected\_data, # percentiles is 90% default  
 climatologyPeriod = c("1982-01-01", "2011-01-01"))  
  
plt\_event = heatwaveR::detect\_event(data = plt\_clim, # duration is 5 days default  
 categories = TRUE, climatology = TRUE)   
  
suppressMessages(event\_line(plt\_event, spread = 100, start\_date = "1982-01-01",  
 end\_date = "2023-12-31", category = TRUE))

Warning: The `guide` argument in `scale\_\*()` cannot be `FALSE`. This was deprecated in  
ggplot2 3.3.4.  
ℹ Please use "none" instead.  
ℹ The deprecated feature was likely used in the heatwaveR package.  
 Please report the issue at  
 <https://github.com/robwschlegel/heatwaveR/issues>.



lolli\_plot(plt\_event)



## Heatwave detection for WA gridded data

## heatwaveR function to apply across gridded data ##  
  
## this is the event only function ##  
event\_only = function(df){  
 # First calculate the percentiles (90th by default)  
 clim = heatwaveR::ts2clm(data = df,   
 climatologyPeriod = c("1982-01-01", "2011-01-01"))  
 # Then the events  
 event = heatwaveR::detect\_event(data = clim, ## default is 5 day duration  
 categories = TRUE, climatology = TRUE)  
 # Return event metric dataframe of results  
 return(event$event)  
}  
  
## for events blocked by year  
block\_av = function(df){  
 # First calculate the percentiles (90th by default)  
 clim = heatwaveR::ts2clm(data = df,   
 climatologyPeriod = c("1982-01-01", "2011-01-01"))  
 # Then the events  
 event = heatwaveR::detect\_event(data = clim, ## default is 5 day duration  
 categories = TRUE, climatology = TRUE)  
 block = block\_average(event)  
 # Return only the event metric dataframe of results  
 return(block$event)  
}

# if you want to read it in  
#OISST\_data = readRDS(".//Data//Downloads//WA\_temp.Rds")

# if you want parrellel, use your cores -1  
#registerDoParallel(cores = 7)  
  
# Detect events  
  
MHW\_result = suppressMessages(plyr::ddply(.data = OISST\_data,   
 .variables = c("lon", "lat"),  
 .fun = event\_only, .parallel = FALSE))  
# Save the results as an .Rds file   
base::saveRDS(MHW\_result, file = ".//Results//WA\_MHW.Rds")  
#write.csv(MHW\_result , ".//Results//WA\_MHW.csv')

# read the mhw rds  
MHW\_result = readRDS(".//Results//WA\_MHW.Rds")

## Convert MHW results to spatial

MHW\_sf = MHW\_result %>%  
 st\_as\_sf(coords = c("lon", "lat"), crs = 4326) %>% # convert to sf  
 st\_intersection(wa\_sf) # clip by the DPIRD bioregions

Warning: attribute variables are assumed to be spatially constant throughout  
all geometries

# Add lat/lon columns to the sf object  
MHW\_sf = MHW\_sf %>%  
 dplyr::mutate(  
 lon = st\_coordinates(.)[, "X"],  
 lat = st\_coordinates(.)[, "Y"]  
 )  
  
MHW = dplyr::tibble(MHW\_sf) # and convert back to tibble

## Area calculation of OISST grid cells

### Assuming length and width at equator

# Define the area of one 0.25-degree grid cell in km²  
grid\_cell\_width\_km = 0.25 \* 111 # Width of the grid cell in km  
grid\_cell\_height\_km = 0.25 \* 111 # Height of the grid cell in km  
grid\_cell\_area\_km2 = grid\_cell\_width\_km \* grid\_cell\_height\_km # Area of the grid # cell in km²

## Merging the data based on each event detected

category\_rank = c(  
 "I Moderate" = 1,  
 "II Strong" = 2,  
 "III Severe" = 3,  
 "IV Extreme" = 4  
)  
  
# Perform the summary statistics  
OISST\_summary = MHW %>%  
 dplyr::mutate(  
 category\_rank = category\_rank[category])%>% ## adding an ordinal rank to category  
 dplyr::group\_by(event\_name) %>%  
 tidyr::drop\_na() %>% # this will drop moderate events  
 dplyr::summarise(  
 num\_events = dplyr::n(), # Count the number of rows (events) within each event  
 intensity = mean(intensity\_mean, na.rm = TRUE), # the mean intensity of the event  
 duration = max(duration, na.rm = TRUE), # Maximum duration  
 area\_km2 = num\_events \* grid\_cell\_area\_km2, # Total area covered in km²  
 category = category[which.max(category\_rank)], # category of events  
 .groups = "drop"  
 ) %>%  
   
 dplyr::arrange(event\_name) %>% # Arrange the results by season  
 dplyr::mutate(label = str\_remove(event\_name, "^Event ")) %>%  
 dplyr::mutate(  
 category = dplyr::recode(category, ## adding labels based on category  
 "I Moderate" = "Moderate",  
 "II Strong" = "Strong",  
 "III Severe" = "Severe",  
 "IV Extreme" = "Extreme"  
 ),  
 category\_colors = dplyr::recode(category,  
 "Moderate" = "gold2",  
 "Strong" = "orange",  
 "Severe" = "red",  
 "Extreme" = "darkred")  
 )  
  
print(OISST\_summary)

# A tibble: 186 × 8  
 event\_name num\_events intensity duration area\_km2 category label  
 <fct> <int> <dbl> <dbl> <dbl> <chr> <chr>  
 1 Event 1999 1246 1.72 141 959498. Severe 1999   
 2 Event 2000a 517 1.77 76 398122. Severe 2000a  
 3 Event 2000b 517 1.94 79 398122. Severe 2000b  
 4 Event 2010a 707 1.65 84 544434. Severe 2010a  
 5 Event 2010b 707 1.74 92 544434. Extreme 2010b  
 6 Event 2011 594 2.27 174 457417. Extreme 2011   
 7 Event 2012a 1075 2.06 140 827817. Severe 2012a  
 8 Event 2012b 1075 1.93 102 827817. Severe 2012b  
 9 Event 2012c 360 1.94 74 277222. Severe 2012c  
10 Event 2013 965 1.86 111 743110. Extreme 2013   
# ℹ 176 more rows  
# ℹ 1 more variable: category\_colors <chr>

## Filtering the data

# Calculate thresholds for filtering  
thresholds = OISST\_summary %>%  
 dplyr::summarise(  
 intensity\_thresh = quantile(intensity, 0.90, na.rm = TRUE),  
 duration\_thresh = quantile(duration, 0.90, na.rm = TRUE),  
 area\_km2\_thresh = quantile(area\_km2, 0.90, na.rm = TRUE)  
 )  
  
# Filter for one of the thresholds above   
OISST\_results = OISST\_summary %>%  
 dplyr::filter(  
 #intensity >= thresholds$intensity\_thresh   
 duration >= thresholds$duration\_thresh   
 #area\_km2 >= thresholds$area\_km2\_thresh   
 )

## Bubble plots

library(ggrepel) # For better label placement  
category\_colors = c(  
 "Moderate" = "gold2",  
 "Strong" = "orange",  
 "Severe" = "red",  
 "Extreme" = "darkred"  
)   
   
  
# Plot with ggplot2  
ggplot(OISST\_results, aes(x = duration, y = intensity, size = area\_km2, fill = category)) +  
 geom\_point(alpha = 1, stroke = 1.5,   
 shape = 21) + # Add bubbles with transparency  
 scale\_size\_continuous(name = "Area (km²)",   
 range = c(3, 20)) + # Adjust bubble size range  
 scale\_fill\_manual(values = category\_colors,   
 name = "Category") + # Custom fill color scale  
 labs(  
 title = "WA Heatwaves",  
 x = "Duration (Days)",  
 y = "Mean Intensity",  
 size = "Area (km²)"  
 ) +  
 theme\_minimal() +  
 xlim(80,180)+  
 theme(  
 panel.grid.major = element\_blank(), # Remove major grid lines  
 panel.grid.minor = element\_blank(), # Remove minor grid lines  
 panel.border = element\_blank(), # Remove panel border  
 axis.line = element\_line(color = "black"), # Add axis lines for clarity  
 legend.key = element\_rect(fill = NA, color = NA),  
 ) +  
 guides(  
 fill = guide\_legend(  
 override.aes = list(shape = 22, size =10) # Use rectangle (shape = 22)  
 )) +  
 ggrepel::geom\_text\_repel(aes(label = label), size = 3)

