```
#### Script Description Header ####

# File Name: 2022 ACC v1b to NBT ECR.R

# File Location: "~/Desktop/Avoided Cost Calculator to Net Billing Tariff Export
Compensation Rate/Net Billing Tariff Export Compensation Rate Calculation"

# Project: Avoided Cost Calculator to Net Billing Tariff Export Compensation Rate

# Description: Calculates Net Billing Tariff export compensation rates
# based on Avoided Cost Calculator (version 2022 v1b).


#### User Inputs ####

# Weighted Average Distribution Capacity Avoided Cost by Installed Capacity in CZ
# If TRUE, take a weighted average of distribution capacity avoided costs based on
# Net Energy Metering/Net Billing Tariff distributed generation capacity (kW-AC)
# in each climate zone.
# If FALSE, take a simple average of distribution capacity avoided costs
# across all climate zones in a utility's service territory.
Weighted_Average_DCap_by_Capacity_in_CZ = FALSE


#### Load Packages ####
library(tidyverse)
library(lubridate)
library(openxlsx)

# Disable Scientific Notation
options(scipen = 999)

# Turn off a confusing message when summarizing data.
options(dplyr.summarise.inform = FALSE)

# Set Working Directories
setwd("~/Desktop/Avoided Cost Calculator to Net Billing Tariff Export Compensation
Rate/Net Billing Tariff Export Compensation Rate Calculation")
Code_WD <- getwd()

setwd("/Users/ryanmann/Desktop/Avoided Cost Calculator to Net Billing Tariff Export
Compensation Rate/2022 ACC v1b")
ACC_WD <- getwd()

setwd("/Users/ryanmann/Desktop/Avoided Cost Calculator to Net Billing Tariff Export
Compensation Rate/DGStats Installed Capacity by Climate Zone")
CZ_Weighting_WD <- getwd()


#### Define Function that Converts Export Compensation Rates to CEC Market Informed Demand
Automation Server (MIDAS) Format ####
# Note: this data is provided in the CEC MIDAS TOU format,
# with timestamps in local America/Los_Angeles timezone.
# The latest (and preferred) MIDAS format is (sub-) hourly streaming data,
# with timestamps in UTC (Greenwich Mean Time) timezone.
# The CEC is building software to convert TOU-formatted data
# to the streaming rate structure.

# DateStart & DateEnd - first day of calendar month to last day of calendar month
# TimeStart & TimeEnd - pad with leading zero for 00:00 - 09:00
# DayTypeStart & DayTypeEnd - 1 = Mon, ... , 5 = Fri, 6 = Sat, 7 = Sun, 8 = Holiday

Net_Billing_Tariff_ECR_MIDAS_Formatter <- function(Net_Billing_Tariff_ECR_Unformatted){
```

```
  Net_Billing_Tariff_ECR_Formatted <- Net_Billing_Tariff_ECR_Unformatted %>%
    mutate(DateStart = make_date(year = ACC_Year, month = Month, day = 1)) %>%
    mutate(TimeStart = paste0(str_pad(as.character(Hour_Beginning),
                                      width = 2, side = "left", pad = "0"),
                              ":00:00")) %>%
    mutate(DateEnd = make_date(year = ACC_Year, month = Month,
                               day = days_in_month(DateStart))) %>%
    mutate(TimeEnd = paste0(str_pad(as.character(Hour_Beginning),
                                    width = 2, side = "left", pad = "0"),
                            ":59:59")) %>%
    mutate(DayTypeStart = ifelse(Weekend_Holiday_Flag == TRUE, 6, 1)) %>%
    mutate(DayTypeEnd = ifelse(Weekend_Holiday_Flag == TRUE, 8, 5)) %>%
    mutate(Value = Export_Compensation_Rate) %>%
    mutate(Unit = "$/kWh") %>%
    select(DateStart, TimeStart, DateEnd, TimeEnd, DayTypeStart, DayTypeEnd, Value, Unit)

  return(Net_Billing_Tariff_ECR_Formatted)

}


# Iterate through PG&E, SCE, and SDG&E ACC Outputs.
Utility_Names <- c("PG&E", "SCE", "SDG&E")

for(Utility_Name_Iter in Utility_Names){

  ##### Calculate Climate Zone Weighting ####
  # For weighted average, use amount of distributed generation capacity (kW-AC)
  # installed under NEM/Net Billing tariffs in each CZ to create weights.
  # For simple average, each CZ in utility service territory has the same weight.

  # Get relevant climate zones from ACC outputs.
  ACC_Filepath <- file.path(ACC_WD,
                            paste0("2022 ACC v1b Outputs - ",
                                   Utility_Name_Iter))
  ACC_Filenames <- list.files(ACC_Filepath)

  # Create data frame with climate zones parsed from ACC output filenames.
  CZ_In_Utility_Territory <- data.frame(`E3.ACC.Climate.Zone` = ACC_Filenames) %>%
    filter(grepl("CZ", `E3.ACC.Climate.Zone`)) %>%
    mutate(`E3.ACC.Climate.Zone` = gsub(".csv", "", `E3.ACC.Climate.Zone`)) %>%
    mutate(`E3.ACC.Climate.Zone` = gsub(paste0(Utility_Name_Iter,
                                               " ", "DCap CZ"), "",
                                        `E3.ACC.Climate.Zone`)) %>%
    mutate(`E3.ACC.Climate.Zone` = factor(`E3.ACC.Climate.Zone`,
                                          levels = c("1", "2", "3A", "3B",
                                                     as.character(seq(4,16))))) %>%
    arrange(`E3.ACC.Climate.Zone`) %>%
    mutate(In_ACC = TRUE)


  if(Weighted_Average_DCap_by_Capacity_in_CZ == TRUE){

    # Distribution Capacity Avoided Cost weighting for each climate zone is calculated
    # by dividing installed capacity in that climate zone
    # by installed capacity across all climate zones in a utility's service territory.
    CZ_Capacity_Filename <- paste0(gsub("&", "", Utility_Name_Iter),
                                   " Installed NEM & NBT Capacity by Climate Zone
2022.csv")

    CZ_Capacity <- read.csv(file.path(CZ_Weighting_WD,
                                      CZ_Capacity_Filename)) %>%
      mutate(`E3.ACC.Climate.Zone` = factor(`E3.ACC.Climate.Zone`,
                                            levels = c("1", "2", "3A", "3B",
                                                       as.character(seq(4,16)))))
```

```
  rm(CZ_Capacity_Filename)

  # Some of the project sites in the DGStats database
  # appear to have been labeled with the wrong ZIP Code/County information
  # and are therefore in CZs that are outside of their utility's service territory.
  # Installed NEM/NBT capacity associated with those climate zones is set to 0 kW-AC
  # before calculating weights used to average avoided distribution capacity costs.
  CZ_Weighting <- CZ_Capacity %>%
    left_join(CZ_In_Utility_Territory,
              by = c("E3.ACC.Climate.Zone")) %>%
    mutate(In_ACC = ifelse(is.na(In_ACC), FALSE, In_ACC)) %>%
    rename(NEM_NBT_Capacity = `Installed.NEM...NBT.Capacity..kW.AC.`) %>%
    mutate(NEM_NBT_Capacity = ifelse(In_ACC == FALSE, 0,
                                     NEM_NBT_Capacity)) %>%
    mutate(DCap_ACC_Weighting = NEM_NBT_Capacity/sum(NEM_NBT_Capacity)) %>%
    select(`E3.ACC.Climate.Zone`, DCap_ACC_Weighting)

  rm(CZ_Capacity)

}else if(Weighted_Average_DCap_by_Capacity_in_CZ == FALSE){

  # Distribution Capacity Avoided Cost weighting for each climate zone
  # is calculated by dividing 1 by the number of CZs in a utility's service territory.
  CZ_Weighting <- data.frame(E3.ACC.Climate.Zone = c("1", "2", "3A", "3B",
                                                     as.character(seq(4,16)))) %>%
    mutate(`E3.ACC.Climate.Zone` = factor(`E3.ACC.Climate.Zone`,
                                          levels = c("1", "2", "3A", "3B",
                                                     as.character(seq(4,16))))) %>%
    left_join(CZ_In_Utility_Territory,
              by = c("E3.ACC.Climate.Zone")) %>%
    mutate(In_ACC = ifelse(is.na(In_ACC), FALSE, In_ACC)) %>%
    mutate(DCap_ACC_Weighting = as.numeric(In_ACC)) %>%
    mutate(DCap_ACC_Weighting = DCap_ACC_Weighting/sum(DCap_ACC_Weighting)) %>%
    select(`E3.ACC.Climate.Zone`, DCap_ACC_Weighting)

}

rm(CZ_In_Utility_Territory)


##### Calculate Total Delivery and Generation Avoided Costs ####
# To reduce amount of data stored in memory simultaneously,
# load ACC files one at a time.
# If ACC filename includes "CZ"
# (i.e. Climate-Zone-specific distribution capacity avoided costs),
# apply weighting before adding to the Delivery dataframe.

# Start by creating blank Delivery and Generation dataframes.

# Timestamps are in Pacific Standard Time (no Daylight Savings Time),
# and include Leap Day 2020 but not Dec. 31, so as to have 365 days.
Date_Times_2020 <- seq.POSIXt(as.POSIXct("2020-01-01 00:00", tz = "Etc/GMT+8"),
                              as.POSIXct("2020-12-30 23:00", tz = "Etc/GMT+8"),
                              by = "1 hour")

ACC_Years <- seq(2023, 2052)


# ACC components assigned to Delivery include:
# Transmission Capacity
# Distribution Capacity
# Greenhouse Gas Adder
# Greenhouse Gas Portfolio Rebalancing
# Methane Leakage
```

```
# (See Attachment B (starting PDF pg. 55) of "SCE 4961-E NBT.pdf")
Delivery_ACC_Components <- c("TCap", "DCap", "GHGAdder", "GHGRebalance", "Methane")

ACC_Delivery <- expand.grid(list(Date_Time = Date_Times_2020,
                                 ACC_Year = ACC_Years)) %>%
  mutate(ECR_Component = "Delivery",
         Total_Avoided_Cost = 0)


# ACC components assigned to Generation include:
# Energy
# Cap and Trade
# Generation Capacity
# Losses
# Avoided Ancillary Services Procurement

# See Attachment B (starting PDF pg. 55) of "SCE 4961-E NBT.pdf".
Generation_ACC_Components <- c("Energy", "CapTrade", "GenCap", "Losses", "AS")

ACC_Generation <- expand.grid(list(Date_Time = Date_Times_2020,
                                   ACC_Year = ACC_Years)) %>%
  mutate(ECR_Component = "Generation",
         Total_Avoided_Cost = 0)

rm(Date_Times_2020, ACC_Years)


for(ACC_Filename in ACC_Filenames){

  ACC_File_Raw <- read.csv(file.path(ACC_Filepath, ACC_Filename))

  # Convert ACC files from "wide" (each column is an ACC Year) to "long".
  # ACC timestamps are in Pacific Standard Time (no Daylight Savings Time).
  ACC_File_Clean <- ACC_File_Raw %>%
    pivot_longer(X2023:X2052, names_to = "ACC_Year", names_prefix = "X",
                 values_to = "Avoided_Cost") %>%
    mutate(Date_Time = as.POSIXct(Date_Time, tz = "Etc/GMT+8")) %>%
    mutate(ACC_Year = as.numeric(ACC_Year)) %>%
    arrange(ACC_Year, Date_Time)

  rm(ACC_File_Raw)

  # Get Avoided Cost Component Name and Climate Zone Number from Filename.
  ACC_Component_Name <- gsub(".csv", "", ACC_Filename)
  ACC_Component_Name <- gsub(paste0(Utility_Name_Iter, " "), "",
                             ACC_Component_Name)

  E3_ACC_Climate_Zone <- if(grepl("DCap", ACC_Component_Name)){
    gsub("DCap CZ", "",  ACC_Component_Name)
  }else NA

  E3_ACC_Climate_Zone <- if(grepl("DCap", ACC_Component_Name)){
    factor(E3_ACC_Climate_Zone,
           levels = c("1", "2", "3A", "3B",
                      as.character(seq(4,16))))
  }else NA

  ACC_Component_Name <- if(grepl("DCap", ACC_Component_Name)){
    "DCap"
  }else ACC_Component_Name


  if(any(grepl(ACC_Component_Name, Delivery_ACC_Components))){
```

```
    if(ACC_Component_Name == "DCap"){
      # Apply CZ Weighting to Avoided Distribution Capacity Cost data,
      # if current dataframe corresponds to a "DCap" file.
      ACC_File_Clean <- ACC_File_Clean %>%
        mutate(Climate_Zone = E3_ACC_Climate_Zone) %>%
        left_join(CZ_Weighting,
                  by = c("Climate_Zone" = "E3.ACC.Climate.Zone")) %>%
        mutate(Avoided_Cost = Avoided_Cost * DCap_ACC_Weighting) %>%
        select(Date_Time, ACC_Year, Avoided_Cost)
    }

    # Add avoided costs from selected ACC file to total Delivery costs,
    # if current dataframe corresponds to a Delivery avoided cost component.
    ACC_Delivery <- ACC_Delivery %>%
      left_join(ACC_File_Clean, by = c("Date_Time", "ACC_Year")) %>%
      mutate(Total_Avoided_Cost = Total_Avoided_Cost + Avoided_Cost) %>%
      select(Date_Time, ACC_Year, ECR_Component, Total_Avoided_Cost)
  }


  if(any(grepl(ACC_Component_Name, Generation_ACC_Components))){

    # Add avoided costs from selected ACC file to total Generation costs,
    # if current dataframe corresponds to a Generation avoided cost component.
    ACC_Generation <- ACC_Generation %>%
      left_join(ACC_File_Clean, by = c("Date_Time", "ACC_Year")) %>%
      mutate(Total_Avoided_Cost = Total_Avoided_Cost + Avoided_Cost) %>%
      select(Date_Time, ACC_Year, ECR_Component, Total_Avoided_Cost)
  }

  rm(ACC_File_Clean, ACC_Component_Name, E3_ACC_Climate_Zone)


}

rm(ACC_Filepath, ACC_Filenames, ACC_Filename)
rm(Delivery_ACC_Components, Generation_ACC_Components, CZ_Weighting)

# Concatenate Generation and Delivery Dataframes
# Note: the values from these two cost categories aren't yet being added together,
# because negative values of each component need to be set to $0/kWh later.
ACC_Combined = rbind(ACC_Delivery,
                     ACC_Generation)

rm(ACC_Delivery, ACC_Generation)


#### Shift Time & Hour Labels to Account for DST, and Identify Weekends & Holidays ####

# Convert from "Etc/GMT+8" (Pacific Standard Time - no Daylight Savings Time)
# to "America/Los_Angeles" (Pacific Prevailing Time - includes DST).
ACC_Combined <- ACC_Combined %>%
  mutate(Date_Time = with_tz(Date_Time, tzone = "America/Los_Angeles"))

# Identify Weekends
ACC_Combined <- ACC_Combined %>%
  mutate(Day_of_Week = weekdays(Date_Time)) %>%
  mutate(Weekend_Flag = Day_of_Week %in% c("Saturday", "Sunday"))


# Identify Holidays

# PG&E Holiday List 2020: https://www.pge.com/tariffs/toudates.shtml

# SCE Rate Tariffs (which include holiday definition):
```

```
# https://www.sce.com/regulatory/tariff-books/rates-pricing-choices
# "Holidays are
# New Year's Day (January 1),
# Presidents' Day (third Monday in February),
# Memorial Day (last Monday in May),
# Independence Day (July 4),
# Labor Day (first Monday in September),
# Veterans Day (November 11),
# Thanksgiving Day (fourth Thursday in November),
# and Christmas (December 25).
# When any holiday listed above falls on Sunday,
# the following Monday will be recognized as a holiday.
# No change will be made for holidays falling on Saturday."

# SDG&E Holiday List 2020:
# https://www.sdge.com/sites/default/files/2020%20Billing%20Cycle%20Schedule_1.pdf
# SDG&E observed Independence Day on July 3rd (July 4th is a Saturday),
# and does not observe Veterans Day.
# https://www.sdge.com/residential/pricing-plans/about-our-pricing-plans/whenmatters
# mentions Veterans Day as a holiday, but 2020 Billing Cycle document doesn't.
# The rate tariffs mention holidays but do not define them.

Holidays_2020 <- read.csv(file.path(Code_WD, "CA_IOU_Holidays_2020.csv")) %>%
  pivot_longer(New.Years.Day:Christmas,
               names_to = "Holiday_Name", values_to = "Date") %>%
  filter(Date != "") %>% # Filter out SDG&E Veterans Day (empty cell in CSV).
  filter(Utility == Utility_Name_Iter) %>%
  mutate(Date = as.Date(Date, tz = "America/Los_Angeles")) %>%
  mutate(Holiday_Flag = TRUE) %>%
  select(Date, Holiday_Flag)

# After performing a join with 2020 holiday calendar,
# rows corresponding to holidays have value of TRUE, and all others are NA.
# The final step is to replace NA values with FALSE.
ACC_Combined <- ACC_Combined %>%
  mutate(Date = as.Date(Date_Time, tz = "America/Los_Angeles")) %>%
  left_join(Holidays_2020, by = "Date") %>%
  mutate(Holiday_Flag = replace_na(Holiday_Flag, FALSE))

rm(Holidays_2020)

# Create Weekend/Holiday Flag for days that are weekends or holidays.
ACC_Combined <- ACC_Combined %>%
  mutate(Weekend_Holiday_Flag = Weekend_Flag | Holiday_Flag)

# Remove columns that will not be used in future steps.
ACC_Combined <- ACC_Combined %>%
  select(Date_Time, ACC_Year, ECR_Component,
         Total_Avoided_Cost, Weekend_Holiday_Flag)


#### Calculate Average Avoided Costs by ACC Year, Month, Weekday/Weekend, and Hour. ####
# Note: This is a simple arithmetic-mean average.
# Averaging across climate zones has already been completed.
# Generation and Delivery cost components are still being kept separate.

Net_Billing_Tariff_ECR <- ACC_Combined %>%
  mutate(Month = lubridate::month(Date_Time),
         Hour_Beginning = lubridate::hour(Date_Time)) %>%
  group_by(ACC_Year, ECR_Component, Month,
           Weekend_Holiday_Flag, Hour_Beginning) %>%
  summarize(Export_Compensation_Rate = mean(Total_Avoided_Cost)) %>%
  ungroup()

rm(ACC_Combined)
```

```
#### Set Any Negative Generation or Delivery Component Values to $0/MWh. ####

# Calculate parallel/vectorized maximum of averaged ACC values and $0/MWh,
# eliminating negative values.
Net_Billing_Tariff_ECR <- Net_Billing_Tariff_ECR %>%
  mutate(Export_Compensation_Rate = pmax(Export_Compensation_Rate, 0))


#### Convert from $/MWh to $/kWh, Round to 5 Digits ####

Net_Billing_Tariff_ECR <- Net_Billing_Tariff_ECR %>%
  mutate(Export_Compensation_Rate = Export_Compensation_Rate/1000) %>%
  mutate(Export_Compensation_Rate = round(Export_Compensation_Rate,
                                          digits = 5))


#### Format Bundled Export Compensation Rates & Save as CSV ####

# Combine Delivery and Generation ECR Components
# This new grouping does not include ECR_Component,
# unlike the one above, so the two components are added.
Net_Billing_Tariff_ECR_Bundled <- Net_Billing_Tariff_ECR %>%
  group_by(ACC_Year, Month, Weekend_Holiday_Flag, Hour_Beginning) %>%
  summarize(Export_Compensation_Rate = sum(Export_Compensation_Rate)) %>%
  ungroup()

# Convert to Standardized MIDAS Format Using Function Defined Above
Net_Billing_Tariff_ECR_Bundled <-
  Net_Billing_Tariff_ECR_MIDAS_Formatter(Net_Billing_Tariff_ECR_Bundled)

# Save as CSV File
NBT_ECR_Filename_Utility <- paste0(Utility_Name_Iter,
                                   " Net Billing Tariff Export Compensation Rate")

if(Weighted_Average_DCap_by_Capacity_in_CZ == TRUE){
  NBT_ECR_Filename_DCap_Weighting <- " - Weighted Average DCap"
} else if(Weighted_Average_DCap_by_Capacity_in_CZ == FALSE){
  NBT_ECR_Filename_DCap_Weighting <- " - Simple Average DCap"
}

NBT_ECR_Filename_Components_Bundled <- " - Bundled.csv"

NBT_ECR_Filename_Bundled <- paste0(NBT_ECR_Filename_Utility,
                                   NBT_ECR_Filename_DCap_Weighting,
                                   NBT_ECR_Filename_Components_Bundled)

write.csv(Net_Billing_Tariff_ECR_Bundled,
          file.path(Code_WD, NBT_ECR_Filename_Bundled), row.names = F)

# Note: NBT_ECR_Filename_Utility and NBT_ECR_Filename_DCap_Weighting
# are reused to name the CSV files for the unbundled components.
rm(Net_Billing_Tariff_ECR_Bundled,
   NBT_ECR_Filename_Components_Bundled,
   NBT_ECR_Filename_Bundled)


#### Format Unbundled Delivery Export Compensation Rates & Save as CSV ####

# Filter to just Delivery ECR component values
Net_Billing_Tariff_ECR_Unbundled_Delivery <- Net_Billing_Tariff_ECR %>%
  filter(ECR_Component == "Delivery") %>%
  select(ACC_Year, Month, Weekend_Holiday_Flag,
         Hour_Beginning, Export_Compensation_Rate)
```

```
  # Convert to Standardized MIDAS Format Using Function Defined Above
  Net_Billing_Tariff_ECR_Unbundled_Delivery <-
    Net_Billing_Tariff_ECR_MIDAS_Formatter(Net_Billing_Tariff_ECR_Unbundled_Delivery)

  # Save as CSV File
  NBT_ECR_Filename_Components_Unbundled_Delivery <- " - Unbundled Delivery.csv"

  NBT_ECR_Filename_Unbundled_Delivery <-
    paste0(NBT_ECR_Filename_Utility,
           NBT_ECR_Filename_DCap_Weighting,
           NBT_ECR_Filename_Components_Unbundled_Delivery)

  write.csv(Net_Billing_Tariff_ECR_Unbundled_Delivery,
            file.path(Code_WD, NBT_ECR_Filename_Unbundled_Delivery), row.names = F)

  rm(Net_Billing_Tariff_ECR_Unbundled_Delivery,
     NBT_ECR_Filename_Components_Unbundled_Delivery,
     NBT_ECR_Filename_Unbundled_Delivery)


  #### Format Unbundled Generation Export Compensation Rates & Save as CSV ####

  # Filter to just Generation ECR component values
  Net_Billing_Tariff_ECR_Unbundled_Generation <- Net_Billing_Tariff_ECR %>%
    filter(ECR_Component == "Generation") %>%
    select(ACC_Year, Month, Weekend_Holiday_Flag,
           Hour_Beginning, Export_Compensation_Rate)

  # Convert to Standardized MIDAS Format Using Function Defined Above
  Net_Billing_Tariff_ECR_Unbundled_Generation <-
    Net_Billing_Tariff_ECR_MIDAS_Formatter(Net_Billing_Tariff_ECR_Unbundled_Generation)

  # Save as CSV File
  NBT_ECR_Filename_Components_Unbundled_Generation <- " - Unbundled Generation.csv"

  NBT_ECR_Filename_Unbundled_Generation <-
    paste0(NBT_ECR_Filename_Utility,
           NBT_ECR_Filename_DCap_Weighting,
           NBT_ECR_Filename_Components_Unbundled_Generation)

  write.csv(Net_Billing_Tariff_ECR_Unbundled_Generation,
            file.path(Code_WD, NBT_ECR_Filename_Unbundled_Generation), row.names = F)

  rm(Net_Billing_Tariff_ECR_Unbundled_Generation,
     NBT_ECR_Filename_Components_Unbundled_Generation,
     NBT_ECR_Filename_Unbundled_Generation)


  # Remove remaining variables, print progress information to console
  rm(Net_Billing_Tariff_ECR,
     NBT_ECR_Filename_Utility,
     NBT_ECR_Filename_DCap_Weighting)

  print(paste0("Completed NBT Export Compensation Rate calculation for ",
               Utility_Name_Iter, "."))

}

rm(Utility_Names, Utility_Name_Iter)
```