AV Profit REAP50

Bijesh Mishra, Ph.D.

2024-12-11

Table of contents

NOTE: RUN “SUMULATION R50” BEFORE RUNNING THIS CODE FOR UPDATED INFORMATION.

Analysis in this file start by loading data saved after simulating tomato and strawberry AV profits. See simulation file for more details. The result tables I have here are quite big. Results are summarized in separate excel files.

1 Setting Up

1.1 Housekeeping

```
rm(list = ls())
options(
  warn=0,
  scipen=999
  )
```

1.2 Load libraries

```
library(tidyverse, warn.conflicts = FALSE, quietly = TRUE)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
library(psych, warn.conflicts = FALSE, quietly = TRUE)
library(likert,  warn.conflicts = FALSE, quietly = TRUE)
library(mice,  warn.conflicts = FALSE, quietly = TRUE)
library(openxlsx2, warn.conflicts = FALSE, quietly = TRUE)
library(ggpubr, warn.conflicts = FALSE, quietly = TRUE)
library(gmodels,  warn.conflicts = FALSE, quietly = TRUE)
library(reshape2, warn.conflicts = FALSE, quietly = TRUE)
library(arrow, warn.conflicts = FALSE, quietly = TRUE)
```

```r
library(plot3D, warn.conflicts = FALSE, quietly = TRUE)
library(plotly, warn.conflicts = FALSE, quietly = TRUE)
library(lattice, warn.conflicts = FALSE, quietly = TRUE)
library(purrr, warn.conflicts = FALSE, quietly = TRUE)
library(furrr, warn.conflicts = FALSE, quietly = TRUE)
library(pheatmap, warn.conflicts = FALSE, quietly = TRUE)
library(grid, warn.conflicts = FALSE, quietly = TRUE)
library(data.table, warn.conflicts = FALSE, quietly = TRUE)
library(parallel, warn.conflicts = FALSE, quietly = TRUE)
```

2 Import data

Import necessary data.

2.1 Tomato AV

Parameters defining agrivoltaic systems:

- sprop = proportion of solar in agrivoltaic system (0 to 1 in 0.5 increment.) Length = 21.
- panels = number of solar panels. Length = 16. Some sprop have same number of panels.
- al_regs = four regions of Alabama. Northern, Central, Black Belt, Southern. Length = 4.
- array = Solar array; Sun tracking (Tracking) and non-tracking (Fixed). Length = 2.
- elecprc = electricity price (1 cents to 6 cents). Length = 6.
- height = clearance height of solar panels. 4.6 ft., 6.4 ft., and 8.2 ft. Length = 3.
- yldvar = crop yield variation (10% to 200%) = Length 21.
- yield = crop yield variation based on yldvar. (same as yldvar) = Length = 21.

Calculated results using above parameters:

- dc_kw = DC system size (kW) See PVWatts® Calculator.
- energy = total energy generated from solar system. See: PVWatts® Calculator.
- capex = AV system capex per kW. See: Capex Cost for AV table 1 and table 3.
- ttlcost = total solar system cost in AV. See: Capex Cost for AV table 1 and table 3.

- anncost = annualized total cost.
- moncost = monthly total cost.
- price = crop yield price per bucket.
- eprofit = profit from electricity.

Result of Interests:

- eannprof = annualized total profit from electricity.
- emonprof = monthly total profit from electricity.
- profit = profit from crops.
- tav_profit = total profit from solar and tomato.

```r
tav_profit <- as.data.frame(
  read_feather(file = "Data/tav_profit R50.feather")
  )
dim(tav_profit)
```

```
[1] 814968     29
```

2.1.1 TAVP - Tomato Profit

- Profit at 100% crop yield at their respective price is subtracted from tav_profit.
- tavp_wocp = tav_profit - profit from 100% crop at their respective prices. This variable gives an idea where av profit stands in relation to crop profit. It helps to identify relative profitability of agrivoltaic system compared to crop only.

```r
# Calculate the profit:
# Step 1: Filter the dataframe to get the unique profit values for each price when yldva
unique_profits <- unique(tav_profit[tav_profit$yldvar == 1,
                                    c("price", "profit")])

# Step 2: Create a lookup table for unique profits by price
profit_lookup <- setNames(unique_profits$profit,
                          unique_profits$price)

# Step 3: Create the new variable tavp_wocp by subtracting the unique profit from tav_pr
tav_profit$tavp_wocp <- mapply(function(
    tav_profit,
    price
    ) {
  profit_to_subtract <- ifelse(
    price %in%
      names(profit_lookup),
    profit_lookup[as.character(price)], 0)
  return(tav_profit - profit_to_subtract)
}, tav_profit$tav_profit, tav_profit$price)
unique_profits # 7 Prices give 7 Profits at 100% Yield.
```

```
    price    profit
11     17  5539.383
32     18  6899.383
53     19  8259.383
74     20  9619.383
95     21 10979.383
116    22 12339.383
137    23 13699.383
```

```
tav_profit[1:21,] # Sample data.
```

```
   sprop    al_regs array dc_kw panels energy elcprc elcrev height    capex
1      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
2      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
3      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
4      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
5      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
6      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
7      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
8      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
9      0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
10     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
11     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
12     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
13     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
14     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
15     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
16     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
17     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
18     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
19     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
20     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
21     0 Black Belt Fixed     0      0      0   0.01      0    4.6 1.593333
   landlease ttlcost inscst recredit reap annlzcost annoftotcost monthlycost
```

```
1       1000       0      0        0    0         0            0           0
2       1000       0      0        0    0         0            0           0
3       1000       0      0        0    0         0            0           0
4       1000       0      0        0    0         0            0           0
5       1000       0      0        0    0         0            0           0
6       1000       0      0        0    0         0            0           0
7       1000       0      0        0    0         0            0           0
8       1000       0      0        0    0         0            0           0
9       1000       0      0        0    0         0            0           0
10      1000       0      0        0    0         0            0           0
11      1000       0      0        0    0         0            0           0
12      1000       0      0        0    0         0            0           0
13      1000       0      0        0    0         0            0           0
14      1000       0      0        0    0         0            0           0
15      1000       0      0        0    0         0            0           0
16      1000       0      0        0    0         0            0           0
17      1000       0      0        0    0         0            0           0
18      1000       0      0        0    0         0            0           0
19      1000       0      0        0    0         0            0           0
20      1000       0      0        0    0         0            0           0
21      1000       0      0        0    0         0            0           0
   opex taxcr anncost eannprof eannprofworeap eannprofwoincentives yldvar yield
1     0     0       0        0              0                    0    2.0  2720
2     0     0       0        0              0                    0    1.9  2584
3     0     0       0        0              0                    0    1.8  2448
4     0     0       0        0              0                    0    1.7  2312
5     0     0       0        0              0                    0    1.6  2176
6     0     0       0        0              0                    0    1.5  2040
7     0     0       0        0              0                    0    1.4  1904
8     0     0       0        0              0                    0    1.3  1768
9     0     0       0        0              0                    0    1.2  1632
10    0     0       0        0              0                    0    1.1  1496
11    0     0       0        0              0                    0    1.0  1360
12    0     0       0        0              0                    0    0.9  1224
13    0     0       0        0              0                    0    0.8  1088
14    0     0       0        0              0                    0    0.7   952
15    0     0       0        0              0                    0    0.6   816
16    0     0       0        0              0                    0    0.5   680
17    0     0       0        0              0                    0    0.4   544
18    0     0       0        0              0                    0    0.3   408
19    0     0       0        0              0                    0    0.2   272
20    0     0       0        0              0                    0    0.1   136
21    0     0       0        0              0                    0    0.0     0
```

```
   price      profit  tav_profit tavp_wocp
1     17  21679.3826  21679.3826     16140
2     17  20065.3826  20065.3826     14526
3     17  18451.3826  18451.3826     12912
4     17  16837.3826  16837.3826     11298
5     17  15223.3826  15223.3826      9684
6     17  13609.3826  13609.3826      8070
7     17  11995.3826  11995.3826      6456
8     17  10381.3826  10381.3826      4842
9     17   8767.3826   8767.3826      3228
10    17   7153.3826   7153.3826      1614
11    17   5539.3826   5539.3826         0
12    17   3925.3826   3925.3826     -1614
13    17   2311.3826   2311.3826     -3228
14    17    697.3826    697.3826     -4842
15    17   -916.6174   -916.6174     -6456
16    17  -2530.6174  -2530.6174     -8070
17    17  -4144.6174  -4144.6174     -9684
18    17  -5758.6174  -5758.6174    -11298
19    17  -7372.6174  -7372.6174    -12912
20    17  -8986.6174  -8986.6174    -14526
21    17 -10600.6174 -10600.6174    -16140
```

```
rm(unique_profits); rm(profit_lookup)
```

2.1.2 TAVP GE Tomato

Tomato yield where tomato AV start becoming more profitable than tomato alone.

```
# Convert the data frame to a data.table for faster operations
setDT(tav_profit)

# Function to process each subset
process_subset <- function(subset) {
  subset <- subset[order(-tavp_wocp)]

  # Find the row where yield changes from positive to negative
  change_row <- which(diff(sign(subset$tavp_wocp)) == -2)[1]

  # Check if change_row is not NA
  if (!is.na(change_row)) {
    result_row <- subset[change_row, ]
    return(result_row)
  } else {
    return(NULL)
  }
}

# Split data by unique combinations of the filtering criteria
split_data <- split(tav_profit,
                    by = c("al_regs", "array", "sprop",
                           "elcprc", "price", "height"))

# Apply the process_subset function sequentially using lapply
results <- lapply(split_data, process_subset)

# Combine all results into a single data.table
tav_be_yld <- rbindlist(results,
                        use.names = TRUE,
                        fill = TRUE) %>%
  select(al_regs, array, sprop, panels, elcprc, price,
         height, profit, yldvar, yield, tav_profit, tavp_wocp)
dim(tav_be_yld)
```

```
[1] 34027    12
```

```
# Dimension and Clean up
rm(results); rm(split_data); rm(process_subset)
```

```
write_xlsx(x = tav_be_yld,
           file = "Results/TAV Tomato Breakeven Yield R50.xlsx",
           as_table = TRUE)
```

2.1.3 Tax Credit for TAV

How much money should be spent as REAP or tax credit to make AV as profitable as crop?

```
tcc_tav_tomato <- tav_profit %>%
  filter(yldvar == 1,
         elcprc == 0.04,
         price == 20,
         sprop >= 0.1
         )
tcc_tav_tomato[which.max(tcc_tav_tomato$eannprofworeap),]
```

```
   sprop  al_regs    array dc_kw panels energy elcprc  elcrev height    capex
   <num>   <char>   <char> <num>  <num>  <num>  <num>   <num>  <num>    <num>
1:   0.1 Southern Tracking 28.25     59  47537   0.04 1901.48    4.6 1.733333
   landlease  ttlcost   inscst recredit     reap annlzcost annoftotcost
       <num>    <num>    <num>    <num>    <num>     <num>        <num>
1:      1000 55735.33 278.6767 313.7442 27857.91  2416.089     4782.678
   monthlycost     opex    taxcr anncost eannprof eannprofworeap
         <num>    <num>    <num>   <num>    <num>          <num>
1:    159.1859 143.4803 1434.803 2559.57 1090.458       -1132.65
   eannprofwoincentives yldvar yield price   profit tav_profit tavp_wocp
                  <num>  <num> <num> <num>    <num>      <num>     <num>
1:            -2881.198      1  1360    20 9619.383   10709.84  1090.458
```

```
cat("Minimum REAP Compensation to make
    TAV as profitable as Tomato: ",
    abs(max(tcc_tav_tomato$eannprofworeap)) + 9619.38,
    fill = TRUE)
```

```
Minimum REAP Compensation to make
    TAV as profitable as Tomato:  10752.03
```

```
tcc_tav_tomato[which.min(tcc_tav_tomato$eannprofworeap),]
```

```
   sprop  al_regs  array  dc_kw panels energy elcprc  elcrev height capex
   <num>   <char> <char>  <num>  <num>  <num>  <num>   <num>  <num> <num>
1:     1 Northern  Fixed 423.74    885 574020   0.04 22960.8    8.2  2.33
   landlease ttlcost   inscst recredit    reap annlzcost annoftotcost
       <num>   <num>    <num>    <num>  <num>     <num>        <num>
1:      1000 1123817 5619.086 3788.532 561712  48716.72     96435.34
   monthlycost    opex   taxcr  anncost eannprof eannprofworeap
         <num>   <num>   <num>    <num>    <num>          <num>
1:     3209.74 2893.06 28930.6 51609.78 4070.151      -40755.41
   eannprofwoincentives yldvar yield price   profit tav_profit tavp_wocp
                  <num>  <num> <num> <num>    <num>      <num>     <num>
1:            -73474.54      1  1360    20 9619.383   13689.53  4070.151
```

```
cat("Maximum REAP Compensation to make
    TAV as profitable as Tomato: ",
    abs(min(tcc_tav_tomato$eannprofworeap)) + 9619.38,
    fill = TRUE)
```

```
Maximum REAP Compensation to make
    TAV as profitable as Tomato:  50374.79
```

2.2 TAV = Tomato Scenarios

Which scenario produces same profit as or above tomato alone at 50% REAP?

```
tav_equals_tomato_r50 <- tav_profit %>%
  filter(yldvar == 0.5,
         elcprc == 0.04,
         price == 20,
         sprop >= 0.1,
         tav_profit == 9619.38
         )
```

2.3 Strawberry AV

See tomato for variable descriptions.

sbav_profit = total profit from solar and strawberry.

```
sbav_profit <- as.data.frame(
  read_feather(file = "Data/sbav_profit R50.feather")
  )
dim(sbav_profit)
```

```
[1] 814968     29
```

2.3.1 SBAVP - Strawberry Profit

- Profit at 100% crop at their respective price is subtracted from sbav_profit.

- sbavp_wocp = sbav_profit - profit from 100% crop at their respective prices. This variable gives an idea where av profit stands in relation to crop profit. It helps to identify relative profitability of agrivoltaic system compared to crop only.

```r
# Step 1: Unique Profit
unique_profits <- unique(sbav_profit[sbav_profit$yldvar == 1,
                                    c("price", "profit")])

# Step 2: Create a lookup table for unique profits by price
profit_lookup <- setNames(unique_profits$profit,
                          unique_profits$price)

# Step 3: sbavp_wocp = sqav_profit - unique profit
sbav_profit$sbavp_wocp <- mapply(function(sbav_profit, price) {
  profit_to_subtract <- ifelse(price %in%
                                 names(profit_lookup),
                               profit_lookup[as.character(price)], 0)
  return(sbav_profit - profit_to_subtract)
}, sbav_profit$sbav_profit, sbav_profit$price)

unique_profits # 7 Prices give 7 Profits at 100% Yield.
```

```
    price    profit
11      3 -5049.345
32      4 -1974.345
53      5  1100.655
74      6  4175.655
95      7  7250.655
116     8 10325.655
137     9 13400.655
```

```r
rm(unique_profits, profit_lookup)
```

2.3.2 SBAVP GE Strawberry Profit

Strawberry yield where strawberry AV profit start becoming more profitable than strawberry alone.

```r
# Convert the data frame to a data.table for faster operations
setDT(sbav_profit)

# Function to process each subset
```

```r
process_subset <- function(subset) {
  subset <- subset[order(-sbavp_wocp)]

  # Find the row where yield changes from positive to negative
  change_row <- which(diff(sign(subset$sbavp_wocp)) == -2)[1]

  # Check if change_row is not NA
  if (!is.na(change_row)) {
    result_row <- subset[change_row, ]
    return(result_row)
  } else {
    return(NULL)
  }
}

# Split data by unique combinations of the filtering criteria
split_data <- split(sbav_profit,
                    by = c("al_regs", "array", "sprop",
                           "elcprc", "price", "height"))

# Apply the process_subset function sequentially using lapply
results <- lapply(split_data, process_subset)

# Combine all results into a single data.table
sbav_be_yld <- rbindlist(results,
                         use.names = TRUE,
                         fill = TRUE) %>%
  select(al_regs, array, sprop, panels, elcprc, price,
         height, profit, yldvar, yield, sbav_profit, sbavp_wocp)

# Clean up
rm(results, split_data, process_subset)
```

```r
write_xlsx(x = sbav_be_yld,
           file = "Results/SBAV Strawberry Breakeven Yield R50.xlsx",
           as.table = TRUE)
```

```
Warning in standardize_case_names(params, arguments = arguments, return =
TRUE): unused arguments (as.table)

Warning in standardize_case_names(..., arguments = arguments): unused arguments
(as.table)
```

```r
dim(sbav_be_yld)
```

```
[1] 30615    12
```

2.3.3 Tax Credit for SBAV

How much money should be spent as REAP or tax credit to make AV as profitable as crop?

```r
tcc_sbav_stberry <- sbav_profit %>%
  filter(yldvar == 1,
         elcprc == 0.04,
         price == 6,
         sprop >= 0.1
         )
tcc_sbav_stberry[which.max(tcc_sbav_stberry$eannprofworeap)]
```

```
   sprop  al_regs    array dc_kw panels energy elcprc  elcrev height    capex
   <num>   <char>   <char> <num>  <num>  <num>  <num>   <num>  <num>    <num>
1:   0.1 Southern Tracking 28.25     59  47537   0.04 1901.48    4.6 1.733333
   landlease  ttlcost   inscst recredit     reap annlzcost annoftotcost
       <num>    <num>    <num>    <num>    <num>     <num>        <num>
1:      1000 55735.33 278.6767 313.7442 27857.91  2416.089     4782.678
   monthlycost     opex    taxcr anncost eannprof eannprofworeap
         <num>    <num>    <num>   <num>    <num>          <num>
1:    159.1859 143.4803 1434.803 2559.57 1090.458       -1132.65
   eannprofwoincentives yldvar yield price   profit sbav_profit sbavp_wocp
                  <num>  <num> <num> <num>    <num>       <num>      <num>
1:            -2881.198      1  3075     6 4175.655    5266.113   1090.458
```

```r
cat("Minimum REAP Compensation to make SBAV
    profitable as Strawberry: ",
    abs(max(tcc_sbav_stberry$eannprofworeap)) + 4175.66,
    fill = TRUE)
```

```
Minimum REAP Compensation to make SBAV
    profitable as Strawberry:  5308.31
```

```r
tcc_sbav_stberry[which.min(tcc_sbav_stberry$eannprofworeap)]
```

```
   sprop  al_regs  array  dc_kw panels energy elcprc  elcrev height capex
   <num>   <char> <char>  <num>  <num>  <num>  <num>   <num>  <num> <num>
1:     1 Northern  Fixed 423.74    885 574020   0.04 22960.8    8.2  2.33
   landlease ttlcost   inscst recredit   reap annlzcost annoftotcost
       <num>   <num>    <num>    <num>  <num>     <num>        <num>
1:      1000 1123817 5619.086 3788.532 561712  48716.72     96435.34
   monthlycost    opex   taxcr  anncost eannprof eannprofworeap
         <num>   <num>   <num>    <num>    <num>          <num>
1:     3209.74 2893.06 28930.6 51609.78 4070.151      -40755.41
   eannprofwoincentives yldvar yield price   profit sbav_profit sbavp_wocp
                  <num>  <num> <num> <num>    <num>       <num>      <num>
1:            -73474.54      1  3075     6 4175.655    8245.806   4070.151
```

```
cat("Maximum REAP Compensation to make
    SBAV profitable as Strawberry: ",
    abs(min(tcc_sbav_stberry$eannprofworeap)) + 4175.66,
    fill = TRUE)
```

```
Maximum REAP Compensation to make
    SBAV profitable as Strawberry:  44931.07
```

2.4 SBAV = Strawberry Scenarios

Which scenario produces same profit as or above tomato alone at 50% REAP?

```
sbav_equals_sberry_r50 <- sbav_profit %>%
  filter(yldvar == 0.5,
         elcprc == 0.04,
         price == 20,
         sprop >= 0.1,
         sbav_profit == 9619.38
         )
```

3 Tomato AV Results

3.1 TAV Profit CrossTab

```
# Define the values for each variable
sprop <- c(0.10, 0.20, 0.30, 0.40, 0.50,
          0.60, 0.70, 0.80, 0.90, 1.00)
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
yldvar <- c(0.5, 1, 1.5)
al_regs <- c("Northern", "Central",
             "Black Belt", "Southern") # Regions AL
price <- c(17, 20, 23) # Crop Price
elcprc <- c(0.04) # Electricity Price

# Define the required columns
required_columns <- c("sprop", "array", "height",
                      "al_regs", "yldvar", "price", "elcprc")

# Check if the columns exist in tav_profit
missing_columns <- setdiff(required_columns,
                           names(tav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in tav_profit: ",
       paste(missing_columns,
             collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, array, sprop), 1,
                   function(x) paste0(x[3], " ", x[2], " ", x[1]))

# Generate row names using reversed order of expand.grid (without elcprc)
row_names <- apply(expand.grid(price, yldvar, al_regs), 1,
                   function(x) paste0(x[3], " ", x[2], " ", x[1]))

# Create an empty matrix to store the results
result_matrix <- matrix(NA,
                        nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)

# Merge with tav_profit to get tav_profit values for each combination
merged_data <- merge(param_combinations,
                     tav_profit,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with
# reversed column and row names (excluding elcprc in row_name)
merged_data$col_name <- apply(
  merged_data[, c("sprop", "array", "height")], 1,
  function(x) paste0(x[1], " ", x[2], " ", x[3]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "yldvar", "price")], 1,
  function(x) paste0(x[1], " ", x[2], " ", x[3]))


# Fill the matrix with tav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[merged_data$row_name == row_condition, ]

  # Ensure that there are valid matches for col_name before assignment
  col_indices <- match(row_data$col_name,
                       colnames(result_matrix))
  valid_indices <- which(!is.na(col_indices))

  if (length(valid_indices) > 0) {
    result_matrix[i, col_indices[valid_indices]] <- round(row_data$tav_profit[valid_indices]
  }
}

ct_tav_pft <- as.data.frame(result_matrix) # Table in Excel.
rm(result_matrix, sprop, array, height, elcprc, price, yldvar, al_regs)
rm(col_indices, col_names, i, missing_columns, required_columns)
```

```r
rm(row_condition, row_names, valid_indices)
rm(param_combinations, row_data, merged_data)
```

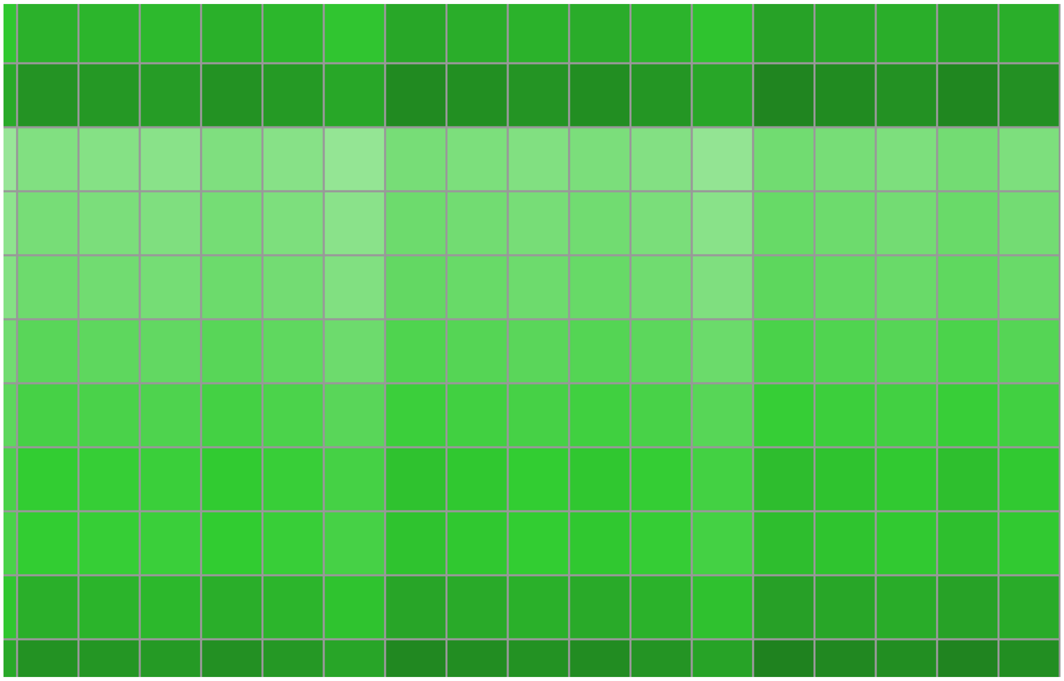
```r
write_xlsx(x = ct_tav_pft %>%
  dplyr::mutate(Row_Names = rownames(ct_tav_pft)) %>%
  dplyr::select(Row_Names, everything()),
           file = "Results/Profit Ctab TAV R50.xlsx",
           as_table = TRUE)
dim(ct_tav_pft)
```

```
[1] 36 60
```

3.2 TAV Profit HeatMap

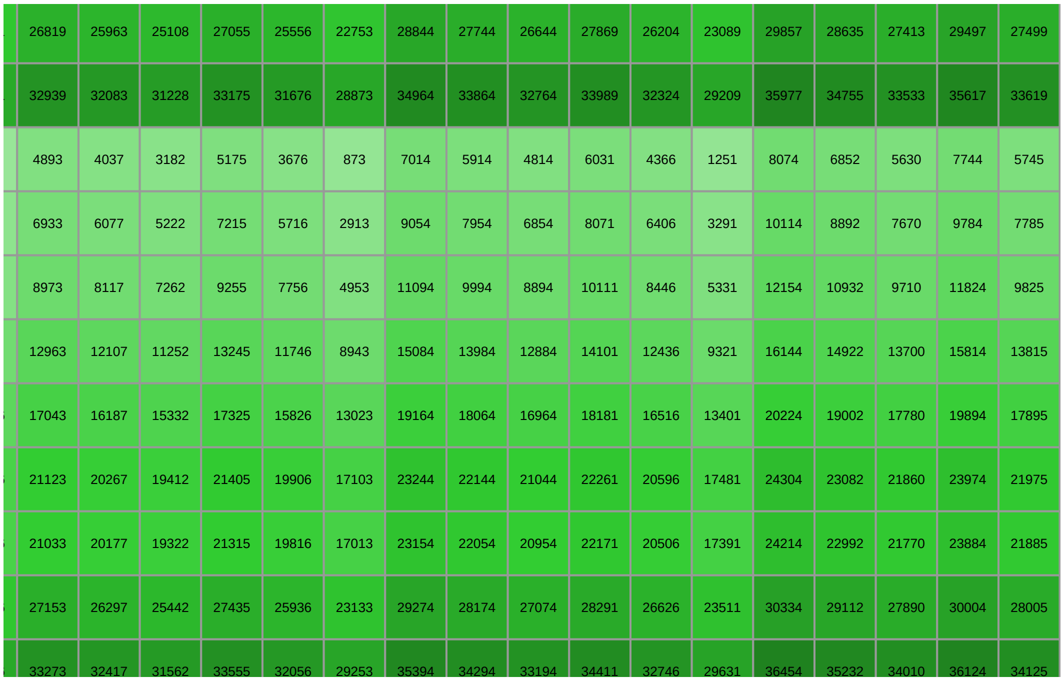
- Heatmap of 324*30 dimension matrix
- Tomato profit.
- Included in the manuscript.

```r
# Calculate color count based on unique values, excluding zero
colorcount <- length(unique(as.vector(as.matrix(ct_tav_pft[-1]))))

# Define custom breaks to ensure zero is distinctly marked
# Calculate min and max values to define the range
min_val <- min(ct_tav_pft, na.rm = TRUE)
max_val <- max(ct_tav_pft, na.rm = TRUE)

# Create breaks that ensure zero is in the middle
breaks <- seq(min_val, max_val, length.out = colorcount)

# Separate color palettes for negative and positive values
# Negative values: Shades of red
neg_colors <- colorRampPalette(c("#890800",
                                 "#FF1709",
                                 "#FF8F89"))(sum(breaks < 0))

# Define the color for zero separately
zero_color <- "#FF8F89"

# Positive values: Shades of green
pos_colors <- colorRampPalette(c("#99E699",
                                 "#32CD32",
                                 "#196719"))(sum(breaks > 0))

# Combine negative colors, zero, and positive colors
custom_colors <- c(neg_colors,
                   zero_color,
                   pos_colors)

# Generate heatmap with the custom color scheme
heatmap_plot <- pheatmap((ct_tav_pft),
                         clustering_distance_rows = "euclidean",
                         clustering_distance_cols = "euclidean",
                         clustering_method = "complete",
                         angle_col = 90,
                         na_col = "white",
                         color = custom_colors,
                         breaks = breaks,
                         cutree_rows = 5,
                         cutree_cols = 4,
                         cluster_rows = FALSE,
                         cluster_cols = FALSE,
                         show_rownames = TRUE,
                         show_colnames = TRUE,
                         display_numbers = FALSE,
                         number_color = "black",
                         fontsize_number = 5,
                         number_format = "%.0f",
                         cellheight = 24,
                         cellwidth = 23,
                         fontsize = 18,
                         fontsize_row = 22,
                         fontsize_col = 22)
```



```r
ggsave(heatmap_plot,
       height = 18,
       width = 24,
       units = "in",
       limitsize = FALSE,
       file = paste0("Plots/TAV Profits CTab R50", ".png"))
rm(breaks, colorcount, custom_colors, max_val, min_val, neg_colors)
rm(pos_colors, zero_color, heatmap_plot)
```

```r
# Calculate color count based on unique values, excluding zero
colorcount <- length(unique(as.vector(as.matrix(ct_tav_pft[-1]))))

# Define custom breaks to ensure zero is distinctly marked
# Calculate min and max values to define the range
min_val <- min(ct_tav_pft, na.rm = TRUE)
max_val <- max(ct_tav_pft, na.rm = TRUE)

# Create breaks that ensure zero is in the middle
breaks <- seq(min_val, max_val, length.out = colorcount)

# Separate color palettes for negative and positive values
# Negative values: Shades of red
neg_colors <- colorRampPalette(c("#890800",
                                 "#FF1709",
                                 "#FF8F89"))(sum(breaks < 0))

# Define the color for zero separately
zero_color <- "#FF8F89"

# Positive values: Shades of green
pos_colors <- colorRampPalette(c("#99E699",
                                 "#32CD32",
                                 "#196719"))(sum(breaks > 0))

# Combine negative colors, zero, and positive colors
custom_colors <- c(neg_colors,
                   zero_color,
                   pos_colors)

# Generate heatmap with the custom color scheme
heatmap_plot <- pheatmap((ct_tav_pft),
                         clustering_distance_rows = "euclidean",
                         clustering_distance_cols = "euclidean",
                         clustering_method = "complete",
                         angle_col = 90,
                         na_col = "white",
                         color = custom_colors,
                         breaks = breaks,
                         cutree_rows = 5,
                         cutree_cols = 4,
                         cluster_rows = FALSE,
                         cluster_cols = FALSE,
                         show_rownames = TRUE,
                         show_colnames = TRUE,
                         display_numbers = TRUE,
                         number_color = "black",
                         fontsize_number = 5,
                         number_format = "%.0f",
                         cellheight = 24,
                         cellwidth = 23,
                         fontsize = 18,
                         fontsize_row = 22,
                         fontsize_col = 22)
```



26819	25963	25108	27055	25556	22753	28844	27744	26644	27869	26204	23089	29857	28635	27413	29497	27499
32939	32083	31228	33175	31676	28873	34964	33864	32764	33989	32324	29209	35977	34755	33533	35617	33619
4893	4037	3182	5175	3676	873	7014	5914	4814	6031	4366	1251	8074	6852	5630	7744	5745
6933	6077	5222	7215	5716	2913	9054	7954	6854	8071	6406	3291	10114	8892	7670	9784	7785
8973	8117	7262	9255	7756	4953	11094	9994	8894	10111	8446	5331	12154	10932	9710	11824	9825
12963	12107	11252	13245	11746	8943	15084	13984	12884	14101	12436	9321	16144	14922	13700	15814	13815
17043	16187	15332	17325	15826	13023	19164	18064	16964	18181	16516	13401	20224	19002	17780	19894	17895
21123	20267	19412	21405	19906	17103	23244	22144	21044	22261	20596	17481	24304	23082	21860	23974	21975
21033	20177	19322	21315	19816	17013	23154	22054	20954	22171	20506	17391	24214	22992	21770	23884	21885
27153	26297	25442	27435	25936	23133	29274	28174	27074	28291	26626	23511	30334	29112	27890	30004	28005
33273	32417	31562	33555	32056	29253	35394	34294	33194	34411	32746	29631	36454	35232	34010	36124	34125

```r
ggsave(heatmap_plot,
       height = 18,
       width = 24,
       units = "in",
       limitsize = FALSE,
       file = paste0("Plots/TAV Profits CTab R50 Values", ".png"))
rm(breaks, colorcount, custom_colors, max_val, min_val, neg_colors)
rm(pos_colors, zero_color, heatmap_plot)
```

3.3 TAV Profit Manuscript

This table is summarized in the manuscript.

```r
# Define the values for each variable
sprop <- c(0, 0.25, 0.50, 0.75, 1.00)
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
yldvar <- c(1) # Yield Variability
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(20) # Crop Price
elcprc <- c(0.04) # Electricity Price
```

```r
# Define the required columns
required_columns <- c("sprop", "array", "height",
                      "al_regs", "yldvar", "price", "elcprc")

# Check if the columns exist in tav_profit
missing_columns <- setdiff(required_columns,
                           names(tav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in tav_profit: ",
       paste(missing_columns,
             collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, sprop), 1,
                   function(x) paste0(x[2], x[1]))

# Generate row names using reversed order of expand.grid
row_names <- apply(expand.grid(elcprc,
                               price,
                               yldvar,
                               al_regs,
                               array), 1,
                   function(x) paste0(x, collapse = ""))

# Create an empty matrix to store the results
result_matrix <- matrix(NA,
                        nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)
```

```r
# Merge with tav_profit to get tav_profit values for each combination
merged_data <- merge(param_combinations,
                     tav_profit,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with
# reversed column and row names
merged_data$col_name <- apply(
  merged_data[, c("sprop", "height")], 1,
  function(x) paste0(x[1], x[2]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "yldvar", "price",
                  "elcprc", "array")], 1,
  function(x) paste0(
                     x[4],
                     x[3],
                     x[2],
                     x[1],
                     x[5]))

# Fill the matrix with tav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[
    merged_data$row_name == row_condition, ]
  if (nrow(row_data) > 0) {
    result_matrix[i,
                  match(row_data$col_name,
                        colnames(result_matrix))] <- round(
                          row_data$tav_profit, 0)
  }
}
tav_prof_man <- as.data.frame(result_matrix) # Table in Excel.
# Display the result matrix
tav_prof_man
```

```
                          04.6 06.4 08.2 0.254.6 0.256.4 0.258.2 0.54.6 0.56.4
0.04201NorthernFixed      9619 9619 9619   11869   11369   10434  14865  13699
0.04201CentralFixed       9619 9619 9619   12063   11563   10628  15317  14151
0.04201Black BeltFixed    9619 9619 9619   12189   11690   10755  15612  14447
```

```
0.04201SouthernFixed      9619 9619 9619   12235   11736   10801  15720  14554
0.04201NorthernTracking   9619 9619 9619   12368   12001   11635  16031  15176
0.04201CentralTracking    9619 9619 9619   12658   12292   11925  16709  15853
0.04201Black BeltTracking 9619 9619 9619   12802   12435   12068  17043  16187
0.04201SouthernTracking   9619 9619 9619   12890   12523   12157  17249  16393
                          0.58.2 0.754.6 0.756.4 0.758.2  14.6  16.4  18.2
0.04201NorthernFixed       11519   17862   16030   12604 20860 18362 13690
0.04201CentralFixed        11971   18573   16741   13315 21829 19331 14659
0.04201Black BeltFixed     12266   19037   17205   13779 22462 19964 15292
0.04201SouthernFixed       12374   19206   17374   13948 22692 20194 15522
0.04201NorthernTracking    14320   19695   18351   17006 23359 21526 19693
0.04201CentralTracking     14998   20760   19415   18071 24810 22977 21144
0.04201Black BeltTracking  15332   21284   19940   18596 25526 23693 21860
0.04201SouthernTracking    15538   21608   20264   18920 25968 24135 22301
```

```
write_xlsx(x = tav_prof_man %>%
  dplyr::mutate(Row_Names = rownames(tav_prof_man)) %>%
  dplyr::select(Row_Names, everything()),
           file = "Results/Profit TAV Manuscript R50.xlsx",
           as_table = TRUE)
rm(result_matrix, sprop, array, height, elcprc, price, yldvar, al_regs)
rm(param_combinations, merged_data, row_data, col_names, i)
rm(missing_columns, required_columns, row_condition, row_names)
```

3.4 TAVP - Tomato Profit CrossTab

- Heatmap of 324*30 dimension matrix.
- See tav_profit for variable naming convention.

```
# Define the values for each variable
sprop <- c(0, 0.05, 0.10, 0.15, 0.20, 0.25,
          0.30, 0.35, 0.40, 0.45, 0.50,
          0.55, 0.60, 0.65, 0.70, 0.75,
          0.80, 0.85, 0.90, 0.95, 1.00)
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
yldvar <- c(0, 0.10, 0.20, 0.30, 0.40,
            0.50, 0.60, 0.70, 0.80, 0.90,
            1.00, 1.10, 1.20, 1.30, 1.40,
            1.50, 1.60, 1.70, 1.80, 1.90, 2.00)
al_regs <- c("Northern", "Central", "Black Belt", "Southern") # Regions of AL
```

```r
price <- c(17, 18, 19, 20, 21, 22, 23) # Crop Price
elcprc <- c(0.03, 0.04, 0.05) # Electricity Price

# Define the required columns
required_columns <- c("sprop", "array", "height",
                  "al_regs", "yldvar", "price", "elcprc")

# Check if the columns exist in tav_profit
missing_columns <- setdiff(required_columns,
                       names(tav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in tavp_wocp: ",
       paste(missing_columns, collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, array, sprop), 1,
                   function(x) paste0(x[3], x[2], x[1]))

# Generate row names using reversed order of expand.grid
row_names <- apply(expand.grid(elcprc,
                               price,
                               yldvar,
                               al_regs), 1,
                   function(x) paste0(x, collapse = ""))

# Create an empty matrix to store the results
result_matrix <- matrix(NA, nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)
```

```r
# Merge with tav_profit to get tav_profit values for each combination
merged_data <- merge(param_combinations,
                     tav_profit,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with
# reversed column and row names
merged_data$col_name <- apply(
  merged_data[, c("sprop", "array", "height")], 1,
  function(x) paste0(x[1], x[2], x[3]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "yldvar", "price", "elcprc")], 1,
  function(x) paste0(x[4],
                     x[3],
                     x[2],
                     x[1]))

# Fill the matrix with tav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[
    merged_data$row_name == row_condition, ]
  if (nrow(row_data) > 0) {
    result_matrix[i,
                  match(row_data$col_name,
                        colnames(result_matrix))] <- round(
                          row_data$tavp_wocp, 2)
  }
}
ct_tavp_wocp <- as.data.frame(result_matrix) # Table in Excel.
rm(result_matrix, merged_data, param_combinations, row_data)
rm(al_regs, array, col_names, elcprc, height, i, missing_columns)
rm(price, required_columns, row_names, sprop, yldvar, row_condition)
```

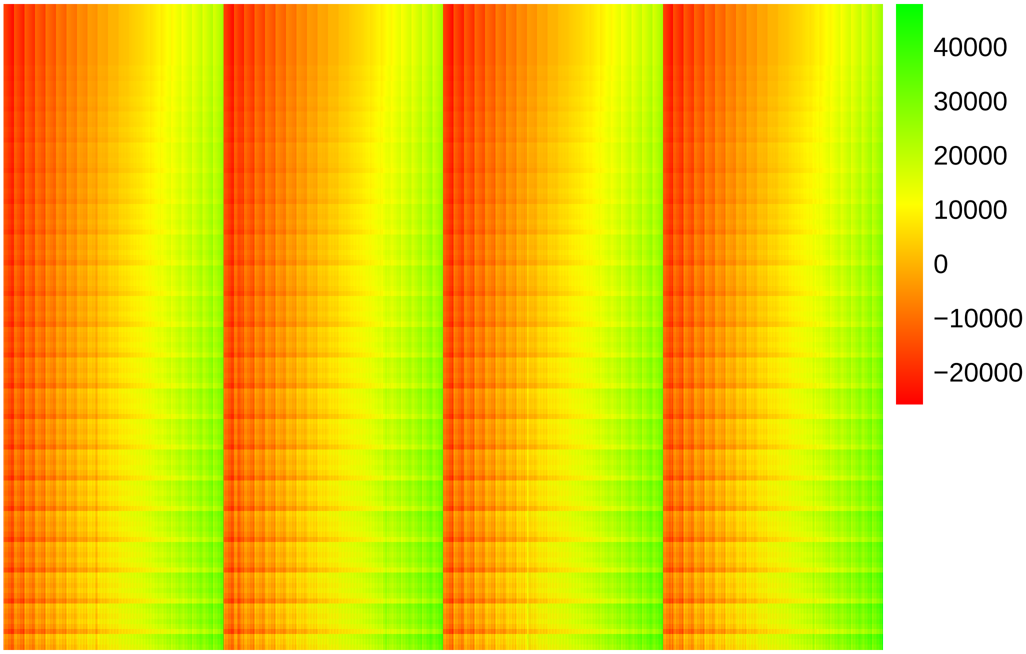
```r
write.csv(as.data.frame(ct_tavp_wocp),
          row.names = TRUE,
          file = "Results/ct_tavp_wocp R50.csv")
```

3.5 TAVP - Tomato Profit HeatMap

```
colorcount = length(unique(as.vector(as.matrix(ct_tavp_wocp[-1]))))
colorcount
```

```
[1] 150092
```

```
heatmap_plot <- pheatmap(t(ct_tavp_wocp),
                         #clustering_distance_rows = "correlation",
                         clustering_distance_rows = "euclidean",
                         clustering_distance_cols = "euclidean",
                         clustering_method = "complete",
                         color = colorRampPalette(c("red",
                                                    "yellow",
                                                    "green"))(colorcount),
                         #cutree_rows = 5,
                         #cutree_cols = 4,
                         cutree_rows = 5,
                         cutree_cols = 4,
                         cluster_rows = FALSE,
                         cluster_cols = FALSE,
                         show_rownames = FALSE,
                         show_colnames = FALSE,
                         display_numbers = FALSE,
                         number_format = "%.2f",
                         #cellheight = 3,
                         #cellwidth = 3
                         )
```



```
ggsave(heatmap_plot,
       height = 8,
       width = 12,
       units = "in",
       file = paste0("Plots/gp_tavp_wocp R50", ".png"))
rm(heatmap_plot); rm(colorcount)
```

3.6 TAV Breakeven Yield Crosstab

```
# Define the values for each variable
sprop <- c(0.05, 0.25, 0.50, 0.75,
           0.80, 0.85, 0.90, 1)
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(17, 20, 23) # Crop Price
elcprc <- c(0.02, 0.03, 0.04) # Electricity Price
#elcprc <- c(0.04) # Electricity Price
yldvar <- c(1)
# yldvar <- c(0, 0.10, 0.20, 0.30, 0.40,
#              0.50, 0.60, 0.70, 0.80, 0.90, 1.00,
#             1.10, 1.20, 1.30, 1.40, 1.50,
#             1.60, 1.70, 1.80, 1.90, 2.00)

# Define the required columns
required_columns <- c("sprop", "array", "height",
                      "al_regs", "price", "elcprc")

# Check if the columns exist in tav_profit
missing_columns <- setdiff(required_columns,
                           names(tav_be_yld))
if (length(missing_columns) > 0) {
  stop("Missing columns in tavp_be_yld: ",
       paste(missing_columns, collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, array, sprop), 1,
                   function(x) paste0(x[3] , x[2] , x[1]))

# Generate row names using reversed order of expand.grid
row_names <- apply(expand.grid(elcprc,
                               price,
                               #yldvar,
                               al_regs), 1,
                   function(x) paste0(x, collapse = ""))

# Create an empty matrix to store the results
result_matrix <- matrix(NA, nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  #yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)
```

```
# Merge with tav_be_yld to get tav_be_yld values for each combination
merged_data <- merge(param_combinations,
                     tav_be_yld,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with
# reversed column and row names
merged_data$col_name <- apply(
  merged_data[, c("sprop", "array", "height")], 1,
  function(x) paste0(x[1], x[2], x[3]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "price", "elcprc")], 1,
  function(x) paste0(x[3],
                     x[2],
                     x[1]))

# Fill the matrix with tav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[
    merged_data$row_name == row_condition, ]
  if (nrow(row_data) > 0) {
    result_matrix[i,
                  match(row_data$col_name,
                        colnames(result_matrix))] <- round(
                          row_data$yield, 0)
  }
}
ct_tav_be_yld <- as.data.frame(result_matrix) # Table in Excel.
rm(result_matrix, merged_data, row_data, param_combinations)
rm(al_regs, array, col_names, elcprc, height, i, missing_columns)
rm(price, required_columns, row_condition, row_names, sprop, yldvar)
```

```
write.csv(as.data.frame(ct_tav_be_yld),
          row.names = TRUE,
          file = "Results/TAV Breakeven Yield R50.csv")
```

3.7 TAV Breakeven Yield HeatMap

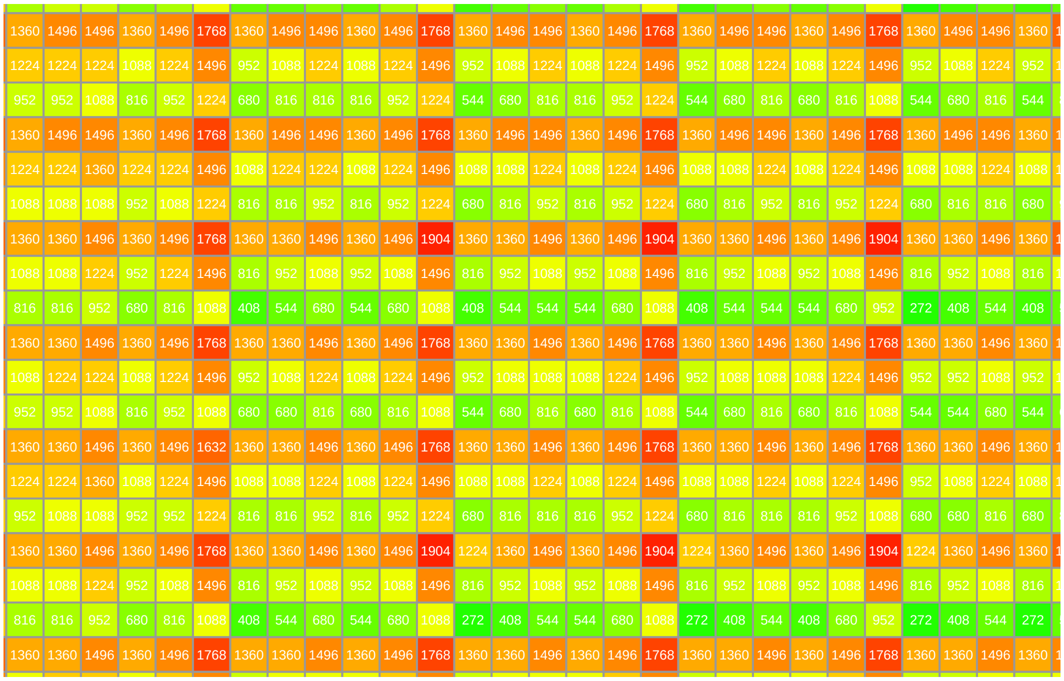
```
uniquevalue = unique(as.vector(as.matrix(ct_tav_be_yld[-1])))
uniquevalue
```

```
 [1]   NA 1496 1360 1224 1088  952 1768 1632  816  680 1904  544  408  272 2040
[16]  136
```

```
colorcount = length(unique(as.vector(as.matrix(ct_tav_be_yld[-1]))))
colorcount
```

```
[1] 16
```

```
heatmap_plot <- pheatmap((ct_tav_be_yld),
                         #clustering_distance_rows = "correlation",
                         clustering_distance_rows = "euclidean",
                         clustering_distance_cols = "euclidean",
                         clustering_method = "complete",
                         angle_col = 90,
                         na_col = "white",
                         color = colorRampPalette(
                           c("green", "yellow","red")
                           )(colorcount),
                         cellheight = 13,
                         cellwidth = 14,
                         fontsize = 12,
                         fontsize_row = 12,
                         fontsize_col = 12,
                         number_color = "white",
                         fontsize_number = 5,
                         cluster_rows = FALSE,
                         cluster_cols = FALSE,
                         show_rownames = TRUE,
                         show_colnames = TRUE,
                         display_numbers = TRUE,
                         number_format = "%.0f"
                         #legend_breaks = uniquevalue
                         )
```



1360	1496	1496	1360	1496	1768	1360	1496	1496	1360	1496	1768	1360	1496	1496	1360	1496	1768	1360	1496	1496	1360	1496	1768	1360	1496	1496	1360
1224	1224	1224	1088	1224	1496	952	1088	1224	1088	1224	1496	952	1088	1224	1088	1224	1496	952	1088	1224	1088	1224	1496	952	1088	1224	952
952	952	1088	816	952	1224	680	816	816	816	952	1224	544	680	816	816	952	1224	544	680	816	680	816	1088	544	680	816	544
1360	1496	1496	1360	1496	1768	1360	1496	1496	1360	1496	1768	1360	1496	1496	1360	1496	1768	1360	1496	1496	1360	1496	1768	1360	1496	1496	1360
1224	1224	1360	1224	1224	1496	1088	1224	1224	1088	1224	1496	1088	1088	1224	1088	1224	1496	1088	1088	1224	1088	1224	1496	1088	1088	1224	1088
1088	1088	1088	952	1088	1224	816	816	952	816	952	1224	680	816	952	816	952	1224	680	816	952	816	952	1224	680	816	816	680
1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1904	1360	1360	1496	1360	1496	1904	1360	1360	1496	1360	1496	1904	1360	1360	1496	1360
1088	1088	1224	952	1224	1496	816	952	1088	952	1088	1496	816	952	1088	952	1088	1496	816	952	1088	952	1088	1496	816	952	1088	816
816	816	952	680	816	1088	408	544	680	544	680	1088	408	544	544	544	680	1088	408	544	544	544	680	952	272	408	544	408
1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360
1088	1224	1224	1088	1224	1496	952	1088	1224	1088	1224	1496	952	1088	1088	1088	1224	1496	952	1088	1088	1088	1224	1496	952	952	1088	952
952	952	1088	816	952	1088	680	680	816	680	816	1088	544	680	816	680	816	1088	544	680	816	680	816	1088	544	544	680	544
1360	1360	1496	1360	1496	1632	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360
1224	1224	1360	1088	1224	1496	1088	1088	1224	1088	1224	1496	1088	1088	1224	1088	1224	1496	1088	1088	1224	1088	1224	1496	952	1088	1224	1088
952	1088	1088	952	952	1224	816	816	952	816	952	1224	680	816	816	816	952	1224	680	816	816	816	952	1088	680	680	816	680
1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1904	1224	1360	1496	1360	1496	1904	1224	1360	1496	1360	1496	1904	1224	1360	1496	1360
1088	1088	1224	952	1088	1496	816	952	1088	952	1088	1496	816	952	1088	952	1088	1496	816	952	1088	952	1088	1496	816	952	1088	816
816	816	952	680	816	1088	408	544	680	544	680	1088	272	408	544	544	680	1088	272	408	544	408	680	952	272	408	544	272
1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360	1496	1768	1360	1360	1496	1360

```r
ggsave(heatmap_plot,
       height = 8,
       width = 12,
       units = "in",
       file = paste0("Plots/TAV Breakeven Yield R50", ".png"))
rm(heatmap_plot, colorcount, uniquevalue)
```

3.8 Plot Tomato Profits by Panels

You can see plot breakdown based on yield variation, crop price, and electricity price. You can see variation for all solar proportion in one facet of the chart. Each facet of the chart contain av profit three heights of solar panels, four regions of AL, two array types.

```r
combinations <- expand.grid(
  yldvar = c(0, 0.1, 0.3, 0.5, 0.7,
             1, 1.20, 1.5, 1.80, 2), # Yield
  price = c(17, 20, 23), # Tomato price
  elcprc = c(0.03, 0.04, 0.05) #Electricity price
)

# Iterate over the combinations and create the plots
for (combo in seq_len(nrow(combinations))) {
```

```r
filtered_data <- tav_profit %>%
  filter(
    yldvar == combinations$yldvar[combo],
    price == combinations$price[combo],
    elcprc == combinations$elcprc[combo]
  )
# If by panel, put panels below in color and group.
tav_sp_plot <- ggplot(data = filtered_data,
                      mapping = aes(x = al_regs,
                                    y = tav_profit,
                                    color = factor(panels),
                                    group = factor(panels))) +
  geom_line() +
  geom_point() +
  facet_grid(height ~ array,
             labeller = as_labeller(
               c(
                 "4.6" = "4.6 ft. Height",
                 "6.4" = "6.4 ft. Height",
                 "8.2" = "8.2 ft. Height",
                 Tracking = "Single Axis Rotation",
                 Fixed = "Fixed Open Rack"
                 ))) +
  guides(color = guide_legend(ncol = 1,
                              reverse = TRUE)) +
  scale_x_discrete(limits = c("Northern", "Central",
                              "Black Belt", "Southern"),
                   labels = c("North", "Center",
                              "B Belt", "South")) +
  guides(color = guide_legend(ncol = 2,
                              reverse = TRUE)) +
  labs(x = "Regions of Alabama",
       y = "Profit ($) from Tomato Agrivoltaic System",
       color = "Number of Solar \n Panels per Acre",
       title = (list(combinations[combo,]))
       ) +
  theme(strip.background = element_blank())

# Add horizontal line at y = 0 if y has both positive and negative values
if (min(filtered_data$tav_profit) < 0 &
    max(filtered_data$tav_profit) > 0) {
  tav_sp_plot <- tav_sp_plot +
      geom_hline(yintercept = 0,
                 linewidth = 0.30,
                 linetype = "dashed",
                 color = "black")
  }
  print(combinations[combo,])
  print(tav_sp_plot)
  ggsave(file = paste0("Plots/tav_sp_ R50", combo, ".png"))
  #break
}
rm(combinations, combo)
```

3.9 Plot Tomato Profits by Yields

You can see plot breakdown based on solar proportion, crop price, and electricity price. You can see variation for all crop yield variation in one facet of the chart. Each facet of the chart contain av profit three heights of solar panels, four regions of AL, two array types.

```
combinations <- expand.grid(
  sprop = c(0, 0.25, 0.50, 0.75, 1.00), # Solar proportion
  price = c(17, 20, 23), # Tomato price
  elcprc = c(0.03, 0.04, 0.05) #Electricity price
)

# Iterate over the combinations and create the plots
for (combo in seq_len(nrow(combinations))) {
  filtered_data <- tav_profit %>%
    filter(
      sprop == combinations$sprop[combo],
      price == combinations$price[combo],
      elcprc == combinations$elcprc[combo]
    )
  # If by yield, put yield below in color and group.
  tav_yv_plot <- ggplot(data = filtered_data,
                        mapping = aes(x = al_regs,
                                      y = tav_profit,
                                      color = factor(yield),
                                      group = factor(yield))) +
    geom_line() +
    geom_point() +
    facet_grid(height ~ array,
               labeller = as_labeller(
                 c(
                   "4.6" = "4.6 ft. Height",
                   "6.4" = "6.4 ft. Height",
                   "8.2" = "8.2 ft. Height",
                   Tracking = "Single Axis Rotation",
                   Fixed = "Fixed Open Rack"
                   ))) +
    guides(color = guide_legend(ncol = 1,
                                reverse = TRUE)) +
    scale_x_discrete(limits = c("Northern", "Central",
                                "Black Belt", "Southern"),
                     labels = c("North", "Center",
                                "B Belt", "South")) +
    guides(color = guide_legend(ncol = 2,
                                reverse = TRUE)) +
    labs(x = "Regions of Alabama",
         y = "Profit ($) from Tomato Agrivoltaic System",
         color = "Tomato Yield \n (25 Lb Buckets)",
         title = (list(combinations[combo,]))
         ) +
    theme(strip.background = element_blank())
  # Add horizontal line at y = 0 if y has both positive and negative values
  if (min(filtered_data$tav_profit) < 0 &
      max(filtered_data$tav_profit) > 0) {
    tav_yv_plot <- tav_yv_plot +
      geom_hline(yintercept = 0,
                 linewidth = 0.30,
                 linetype = "dashed",
                 color = "black")
  }
  print(combinations[combo,])
  print(tav_yv_plot)
  ggsave(file = paste0("Plots/tav_yv_ R50", combo, ".png"))
  #break
}
rm(combinations, combo)
```

4 Strawberry AV Results

4.1 SBAV Profit CrossTab

```
# Define the values for each variable
sprop <- c(0.10, 0.20, 0.30, 0.40, 0.50,
           0.60, 0.70, 0.80, 0.90, 1.00)
array <- c("Fixed", "Tracking")
height <- c(4.6, 6.4, 8.2)
# yldvar <- c(0, 0.10, 0.20, 0.30, 0.40, 0.50,
#             0.60, 0.70, 0.80, 0.90, 1.00,
#             1.10, 1.20, 1.30, 1.40, 1.50,
#             1.60, 1.70, 1.80, 1.90, 2.00)
yldvar <- c(0.5, 1, 1.5)
al_regs <- c("Northern", "Central",
             "Black Belt", "Southern")
price <- c(3, 6, 9)
elcprc <- c(0.04) # Electricity Price

# Define the required columns
required_columns <- c("sprop", "array", "height",
                      "al_regs", "yldvar", "price", "elcprc")

# Check if the columns exist in sbav_profit
missing_columns <- setdiff(required_columns, names(sbav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in sbav_profit: ", paste(missing_columns, collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, array, sprop), 1,
                   function(x) paste0(x[3], " ", x[2], " ", x[1]))

# Generate row names using reversed order of expand.grid (without elcprc)
row_names <- apply(expand.grid(price, yldvar, al_regs), 1,
                   function(x) paste0(x[3], " ", x[2], " ", x[1]))

# Create an empty matrix to store the results
result_matrix <- matrix(NA, nrow = length(row_names), ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names
```

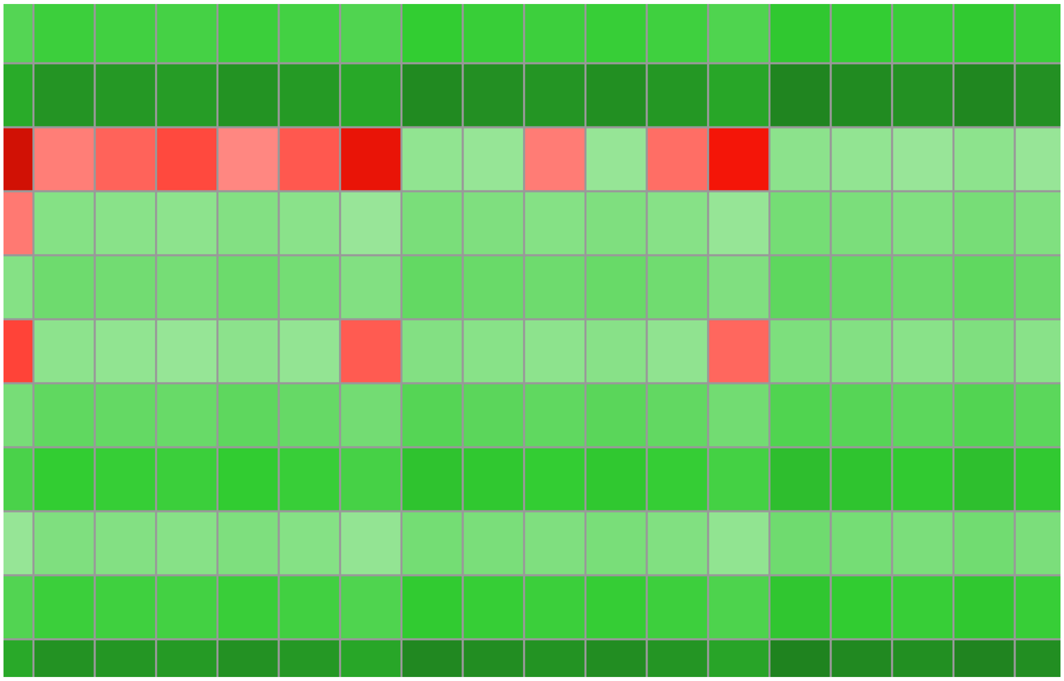
```r
# Create a data frame with all combinations of parameters in reversed order (including elcpr
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)

# Merge with sbav_profit to get sbav_profit values for each combination
merged_data <- merge(param_combinations,
                     sbav_profit,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with reversed column and row names (excluding el
merged_data$col_name <- apply(
  merged_data[, c("sprop", "array", "height")], 1,
  function(x) paste0(x[1], " ", x[2], " ", x[3]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "yldvar", "price")], 1,
  function(x) paste0(x[1], " ", x[2], " ", x[3]))

# Fill the matrix with sbav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[merged_data$row_name == row_condition, ]

  # Ensure that there are valid matches for col_name before assignment
  col_indices <- match(row_data$col_name,
                       colnames(result_matrix))
  valid_indices <- which(!is.na(col_indices))

  if (length(valid_indices) > 0) {
    result_matrix[i, col_indices[valid_indices]] <- round(row_data$sbav_profit[valid_indices]
  }
}
ct_sbav_pft <- as.data.frame(result_matrix) #Table in Excel.
rm(result_matrix, sprop, array, height, elcprc, price, yldvar, al_regs)
rm(col_indices, col_names, i, missing_columns, required_columns)
rm(row_condition, row_names, valid_indices)
rm(param_combinations, row_data, merged_data)
```

```r
write_xlsx(x = ct_sbav_pft %>%
  dplyr::mutate(Row_Names = rownames(ct_sbav_pft)) %>%
  dplyr::select(Row_Names, everything()),
           file = "Results/Profit Ctab SBAV R50.xlsx",
           as_table = TRUE)
```

4.2 SBAV Profit HeatMap

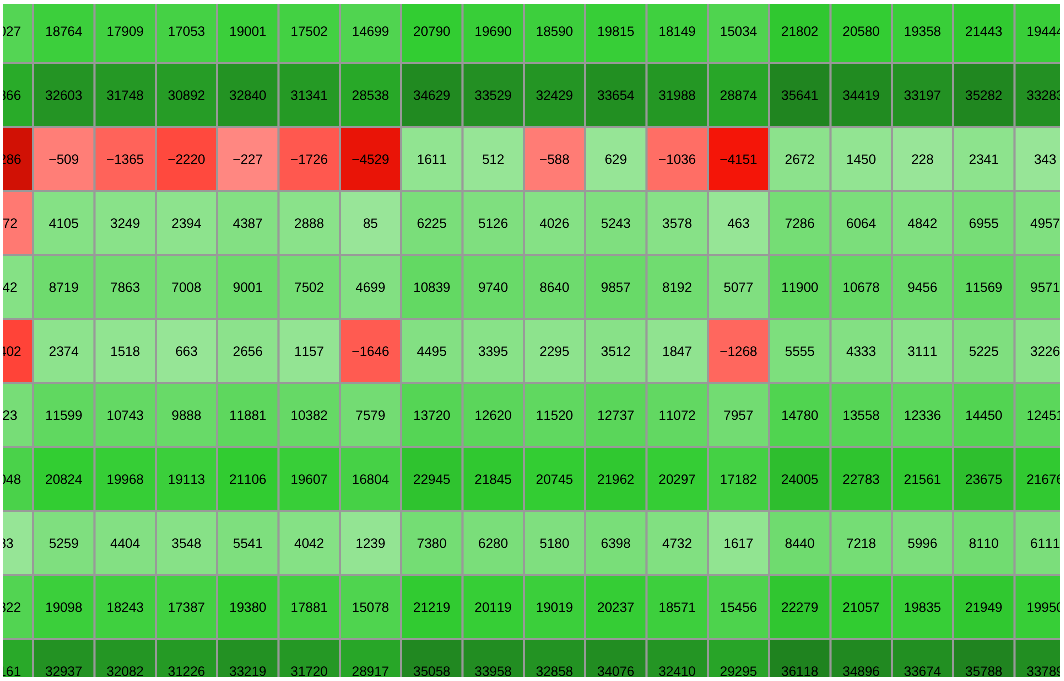
```r
# Calculate color count based on unique values, excluding zero
colorcount <- length(unique(as.vector(as.matrix(ct_sbav_pft[-1]))))

# Define custom breaks to ensure zero is distinctly marked
# Calculate min and max values to define the range
min_val <- min(ct_sbav_pft, na.rm = TRUE)
max_val <- max(ct_sbav_pft, na.rm = TRUE)

# Create breaks that ensure zero is in the middle
breaks <- seq(min_val, max_val, length.out = colorcount)

# Separate color palettes for negative and positive values
# Negative values: Shades of red
neg_colors <- colorRampPalette(c("#890800",
                                 "#FF1709",
                                 "#FF8F89"))(sum(breaks < 0))

# Define the color for zero separately
zero_color <- "#FF8F89"

# Positive values: Shades of green
pos_colors <- colorRampPalette(c("#99E699",
                                 "#32CD32",
                                 "#196719"))(sum(breaks > 0))

# Combine negative colors, zero, and positive colors
custom_colors <- c(neg_colors,
                   zero_color,
                   pos_colors)
```

```r
# Generate heatmap with the custom color scheme
heatmap_plot <- pheatmap(
  (ct_sbav_pft),
  clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean",
  clustering_method = "complete",
  angle_col = 90,
  na_col = "white",
  color = custom_colors,
  breaks = breaks,
  cutree_rows = 5,
  cutree_cols = 4,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  show_rownames = TRUE,
  show_colnames = TRUE,
  display_numbers = FALSE,
  number_color = "black",
  fontsize_number = 5,
  number_format = "%.0f",
  cellheight = 24,
  cellwidth = 23,
  fontsize = 18,
  fontsize_row = 22,
  fontsize_col = 22
)
```



```
ggsave(heatmap_plot,
       height = 18,
       width = 24,
       units = "in",
       limitsize = FALSE,
       file = paste0("Plots/SBAV Profits Ctab R50", ".png"))
rm(breaks, colorcount, custom_colors, max_val, min_val, neg_colors)
rm(pos_colors, zero_color, heatmap_plot)
```

```
# Calculate color count based on unique values, excluding zero
colorcount <- length(unique(as.vector(as.matrix(ct_sbav_pft[-1]))))

# Define custom breaks to ensure zero is distinctly marked
# Calculate min and max values to define the range
min_val <- min(ct_sbav_pft, na.rm = TRUE)
max_val <- max(ct_sbav_pft, na.rm = TRUE)

# Create breaks that ensure zero is in the middle
breaks <- seq(min_val, max_val, length.out = colorcount)

# Separate color palettes for negative and positive values
# Negative values: Shades of red
neg_colors <- colorRampPalette(c("#890800",
                                 "#FF1709",
                                 "#FF8F89"))(sum(breaks < 0))

# Define the color for zero separately
zero_color <- "#FF8F89"

# Positive values: Shades of green
pos_colors <- colorRampPalette(c("#99E699",
                                 "#32CD32",
                                 "#196719"))(sum(breaks > 0))

# Combine negative colors, zero, and positive colors
custom_colors <- c(neg_colors,
                   zero_color,
                   pos_colors)

# Generate heatmap with the custom color scheme
heatmap_plot <- pheatmap(
  (ct_sbav_pft),
  clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean",
  clustering_method = "complete",
  angle_col = 90,
  na_col = "white",
  color = custom_colors,
  breaks = breaks,
  cutree_rows = 5,
  cutree_cols = 4,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  show_rownames = TRUE,
  show_colnames = TRUE,
  display_numbers = TRUE,
  number_color = "black",
  fontsize_number = 5,
  number_format = "%.0f",
  cellheight = 24,
  cellwidth = 23,
  fontsize = 18,
  fontsize_row = 22,
  fontsize_col = 22
)
```



027	18764	17909	17053	19001	17502	14699	20790	19690	18590	19815	18149	15034	21802	20580	19358	21443	19444
366	32603	31748	30892	32840	31341	28538	34629	33529	32429	33654	31988	28874	35641	34419	33197	35282	33283
286	−509	−1365	−2220	−227	−1726	−4529	1611	512	−588	629	−1036	−4151	2672	1450	228	2341	343
72	4105	3249	2394	4387	2888	85	6225	5126	4026	5243	3578	463	7286	6064	4842	6955	4957
42	8719	7863	7008	9001	7502	4699	10839	9740	8640	9857	8192	5077	11900	10678	9456	11569	9571
402	2374	1518	663	2656	1157	−1646	4495	3395	2295	3512	1847	−1268	5555	4333	3111	5225	3226
23	11599	10743	9888	11881	10382	7579	13720	12620	11520	12737	11072	7957	14780	13558	12336	14450	12451
048	20824	19968	19113	21106	19607	16804	22945	21845	20745	21962	20297	17182	24005	22783	21561	23675	21676
33	5259	4404	3548	5541	4042	1239	7380	6280	5180	6398	4732	1617	8440	7218	5996	8110	6111
322	19098	18243	17387	19380	17881	15078	21219	20119	19019	20237	18571	15456	22279	21057	19835	21949	19950
161	32937	32082	31226	33219	31720	28917	35058	33958	32858	34076	32410	29295	36118	34896	33674	35788	33789

```
ggsave(heatmap_plot,
       height = 18,
       width = 24,
       units = "in",
       limitsize = FALSE,
       file = paste0("Plots/SBAV Profits Ctab R50 Values", ".png"))
rm(breaks, colorcount, custom_colors, max_val, min_val, neg_colors)
rm(pos_colors, zero_color, heatmap_plot)
```

4.3 SBAV Profit Manuscript

This table is in manuscript.

```
# Define the values for each variable
sprop <- c(0, 0.25, 0.50, 0.75, 1.00)
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
yldvar <- c(1) # Yield Variability
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(6) # Crop Price
elcprc <- c(0.04) # Electricity Price
```

```
# Define the required columns
required_columns <- c("sprop", "array", "height",
                      "al_regs", "yldvar", "price", "elcprc")

# Check if the columns exist in tav_profit
missing_columns <- setdiff(required_columns,
                           names(sbav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in tav_profit: ",
       paste(missing_columns,
             collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, sprop), 1,
                   function(x) paste0(x[2], x[1]))

# Generate row names using reversed order of expand.grid
row_names <- apply(expand.grid(elcprc,
                               price,
                               yldvar,
                               al_regs,
                               array), 1,
                   function(x) paste0(x, collapse = ""))

# Create an empty matrix to store the results
result_matrix <- matrix(NA,
                        nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)
```

```r
# Merge with tav_profit to get tav_profit values for each combination
merged_data <- merge(param_combinations,
                     sbav_profit,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with
# reversed column and row names
merged_data$col_name <- apply(
  merged_data[, c("sprop", "height")], 1,
  function(x) paste0(x[1], x[2]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "yldvar", "price",
                  "elcprc", "array")], 1,
  function(x) paste0(
                     x[4],
                     x[3],
                     x[2],
                     x[1],
                     x[5]))

# Fill the matrix with tav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[
    merged_data$row_name == row_condition, ]
  if (nrow(row_data) > 0) {
    result_matrix[i,
                  match(row_data$col_name,
                        colnames(result_matrix))] <- round(
                          row_data$sbav_profit, 0)
  }
}
sbav_prof_man <- as.data.frame(result_matrix) # Table in Excel.
sbav_prof_man
```

```
                         04.6 06.4 08.2 0.254.6 0.256.4 0.258.2 0.54.6 0.56.4
0.0461NorthernFixed      4176 4176 4176    6425    5925    4991   9421   8255
0.0461CentralFixed       4176 4176 4176    6619    6119    5185   9873   8708
0.0461Black BeltFixed    4176 4176 4176    6746    6246    5312  10169   9003
0.0461SouthernFixed      4176 4176 4176    6792    6292    5357  10276   9110
```

```
0.0461NorthernTracking   4176 4176 4176    6924    6558    6191   10588   9732
0.0461CentralTracking    4176 4176 4176    7215    6848    6481   11265  10410
0.0461Black BeltTracking 4176 4176 4176    7358    6991    6625   11599  10743
0.0461SouthernTracking   4176 4176 4176    7446    7080    6713   11805  10950
                         0.58.2 0.754.6 0.756.4 0.758.2   14.6   16.4   18.2
0.0461NorthernFixed        6075   12419   10587    7160  15416  12918   8246
0.0461CentralFixed         6527   13130   11298    7871  16386  13887   9215
0.0461Black BeltFixed      6823   13594   11762    8335  17018  14520   9848
0.0461SouthernFixed        6930   13762   11930    8504  17249  14750  10078
0.0461NorthernTracking     8877   14251   12907   11563  17915  16082  14249
0.0461CentralTracking      9554   15316   13972   12627  19367  17534  15700
0.0461Black BeltTracking   9888   15841   14496   13152  20082  18249  16416
0.0461SouthernTracking    10094   16165   14820   13476  20524  18691  16858
```

```r
write_xlsx(x = sbav_prof_man %>%
  dplyr::mutate(Row_Names = rownames(sbav_prof_man)) %>%
  dplyr::select(Row_Names, everything()),
           file = "Results/Profit SBAV Manuscript R50.xlsx",
           as_table = TRUE)
rm(result_matrix, sprop, array, height, elcprc, price, yldvar, al_regs)
rm(param_combinations, merged_data, row_data, col_names, i)
rm(missing_columns, required_columns, row_condition, row_names)
```

4.4 SBAVP - Strawberry Profit CrossTab

- Row naming: Electricity Price_Crop Price_Solar Proportion_Alabama Regions
- Column naming: Solar Proportion_Array Types_Solar Panel Height.
- Solar Proportion can be converted to total number of panels.
- Only selected values from each variables are extracted for tabulation purpose.
- Values displayed in the table are profit from Strawberry AV system.

```r
# Define the values for each variable
sprop <- c(0, 0.05, 0.10, 0.15, 0.20, 0.25,
           0.30, 0.35, 0.40, 0.45, 0.50,
           0.55, 0.60, 0.65, 0.70, 0.75,
           0.80, 0.85, 0.90, 0.95, 1.00)
array <- c("Fixed", "Tracking")
height <- c(4.6, 6.4, 8.2)
yldvar <- c(0, 0.10, 0.20, 0.30, 0.40, 0.50,
            0.60, 0.70, 0.80, 0.90, 1.00,
            1.10, 1.20, 1.30, 1.40, 1.50,
            1.60, 1.70, 1.80, 1.90, 2.00)
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(3, 4, 5, 6, 7, 8, 9)
elcprc <- c(0.03, 0.04, 0.05)

# Define the required columns
required_columns <- c("sprop", "array", "height",
                      "al_regs", "yldvar", "price", "elcprc")

# Check if the columns exist in sbav_profit
missing_columns <- setdiff(required_columns,
                           names(sbav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in sbav_profit: ",
       paste(missing_columns, collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, array, sprop), 1,
                   function(x) paste0(x[3], x[2], x[1]))

# Generate row names using reversed order of expand.grid
row_names <- apply(expand.grid(elcprc,
                               price,
                               yldvar,
                               al_regs), 1,
                   function(x) paste0(x, collapse = ""))

# Create an empty matrix to store the results
result_matrix <- matrix(NA, nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  yldvar = yldvar,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)

# Merge with tav_profit to get sbav_profit values for each combination
merged_data <- merge(param_combinations,
                     sbav_profit,
                     by = required_columns,
                     all.x = TRUE)

# Reshape merged_data to fill result_matrix with
# reversed column and row names
merged_data$col_name <- apply(
  merged_data[, c("sprop", "array", "height")], 1,
  function(x) paste0(x[1],
                     x[2],
                     x[3]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "yldvar", "price", "elcprc")], 1,
  function(x) paste0(x[4],
                     x[3],
                     x[2],
                     x[1]))

# Fill the matrix with sbav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[
    merged_data$row_name == row_condition, ]
  if (nrow(row_data) > 0) {
    result_matrix[i,
                  match(row_data$col_name,
                        colnames(result_matrix))] <- round(
                          row_data$sbavp_wocp, 2)
  }
}
ct_sbavp_wocp <- as.data.frame(result_matrix) #Table in Excel.
rm(result_matrix, merged_data, param_combinations, row_data)
rm(al_regs, array, col_names, elcprc, height, i, missing_columns)
rm(price, required_columns, row_names, sprop, yldvar, row_condition)
```

```
write.csv(as.data.frame(ct_sbavp_wocp),
          row.names = TRUE,
          #col.names = TRUE,
          file = "Results/ct_sbavp_wocp R50.csv")
dim(ct_sbavp_wocp)
```

```
[1] 1764  126
```

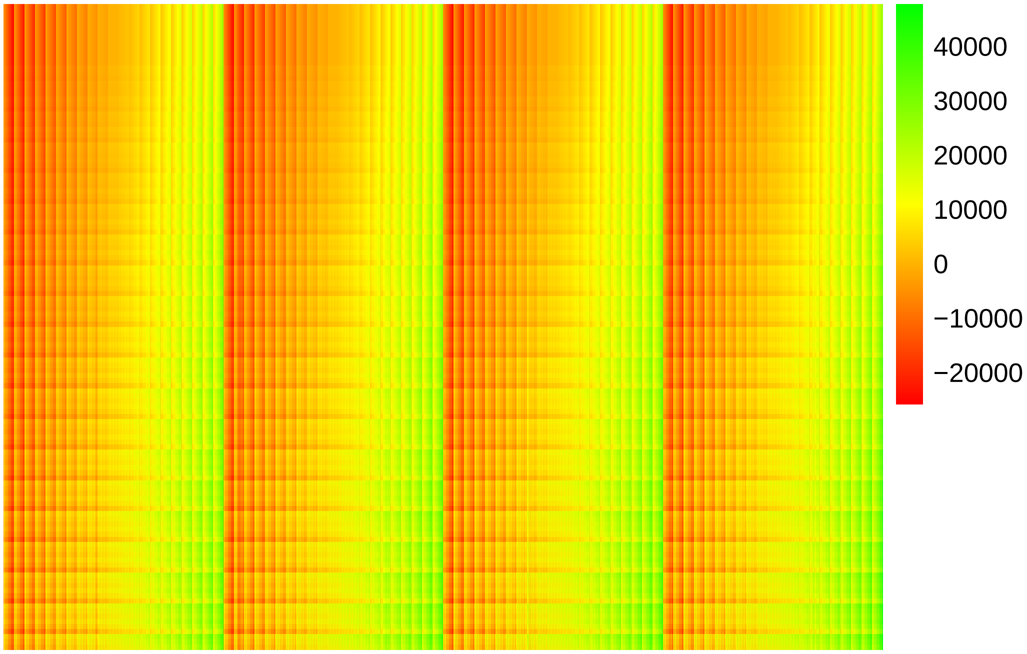
4.5 SBAVP - Strawberry Profit HeatMap

- Heatmap of 324*30 dimension matrix.

```
colorcount = length(unique(as.vector(as.matrix(ct_sbavp_wocp[-1]))))
colorcount
```

```
[1] 149564
```

```
heatmap_plot <- pheatmap(t(ct_sbavp_wocp),
                         #clustering_distance_rows = "correlation",
                         clustering_distance_rows = "euclidean",
                         clustering_distance_cols = "euclidean",
                         clustering_method = "complete",
                         color = colorRampPalette(c("red",
                                                    "yellow",
                                                    "green"))(colorcount),
                         #cutree_rows = 5,
                         #cutree_cols = 4,
                         cluster_rows = FALSE,
                         cluster_cols = FALSE,
                         show_rownames = FALSE,
                         show_colnames = FALSE,
                         display_numbers = FALSE,
                         number_format = "%.2f",
                         #cellheight = 3,
                         #cellwidth = 3
                         )
```



```
ggsave(heatmap_plot,
       height = 8,
       width = 12,
       units = "in",
       file = paste0("Plots/gp_sbavp_wocp R50", ".png"))
rm(heatmap_plot, colorcount)
```

4.6 SBAV Breakeven Yield Crosstab

```
sprop <- c(0.05, 0.25, 0.50, 0.75, 0.80, 0.85, 0.90, 1)
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(3, 6, 9)
elcprc <- c(0.02, 0.03, 0.04) # Electricity Price
yldvar <- c(0, 0.10, 0.20, 0.30, 0.40,
            0.50, 0.60, 0.70, 0.80, 0.90, 1.00,
            1.10, 1.20, 1.30, 1.40, 1.50, 1.60,
            1.70, 1.80, 1.90, 2.00)

# Define the required columns
```

```
required_columns <- c("sprop", "array", "height",
                      "al_regs", "price", "elcprc")

# Check if the columns exist in sbav_profit
missing_columns <- setdiff(required_columns,
                           names(sbav_be_yld))
if (length(missing_columns) > 0) {
  stop("Missing columns in sbav_be_yld: ",
       paste(missing_columns, collapse = ", "))
}

# Generate column names using reversed order of expand.grid
col_names <- apply(expand.grid(height, array, sprop), 1,
                   function(x) paste0(x[3], x[2], x[1]))

# Generate row names using reversed order of expand.grid
row_names <- apply(expand.grid(elcprc,
                               price,
                               al_regs), 1,
                   function(x) paste0(x, collapse = ""))

# Create an empty matrix to store the results
result_matrix <- matrix(NA, nrow = length(row_names),
                        ncol = length(col_names))
colnames(result_matrix) <- col_names
rownames(result_matrix) <- row_names

# Create a data frame with
# all combinations of parameters in reversed order
param_combinations <- expand.grid(elcprc = elcprc,
                                  price = price,
                                  al_regs = al_regs,
                                  height = height,
                                  array = array,
                                  sprop = sprop)

# Merge with tavp_be_yld to get tavp_be_yld values for each combination
merged_data <- merge(param_combinations,
                     sbav_be_yld,
                     by = required_columns,
                     all.x = TRUE)
```

```r
# Reshape merged_data to fill result_matrix with
# reversed column and row names
merged_data$col_name <- apply(
  merged_data[, c("sprop", "array", "height")], 1,
  function(x) paste0(x[1], x[2], x[3]))

merged_data$row_name <- apply(
  merged_data[, c("al_regs", "price", "elcprc")], 1,
  function(x) paste0(x[3],
                     x[2],
                     x[1]))

# Fill the matrix with sbav_profit values
for (i in seq_len(nrow(result_matrix))) {
  row_condition <- rownames(result_matrix)[i]
  row_data <- merged_data[
    merged_data$row_name == row_condition, ]
  if (nrow(row_data) > 0) {
    result_matrix[i,
                  match(row_data$col_name,
                        colnames(result_matrix))] <- round(
                          row_data$yield, 0)
  }
}
ct_sbav_be_yld <- as.data.frame(result_matrix) # Table in Excel.
rm(result_matrix, merged_data, row_data, param_combinations)
rm(al_regs, array, col_names, elcprc, height, i, missing_columns)
rm(price, required_columns, row_condition, row_names, sprop, yldvar)
```

```r
write.csv(as.data.frame(ct_sbav_be_yld),
          row.names = TRUE,
          file = "Results/ct_sbav_be_yld R50.csv")
```

4.7 SBAV Breakeven Yield HeatMap

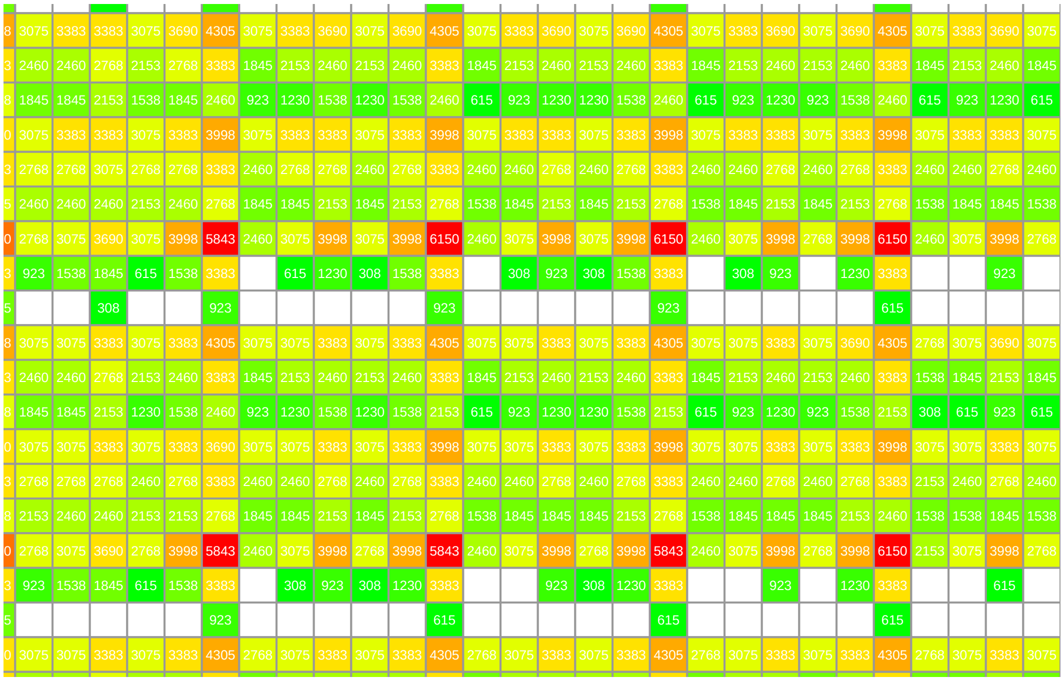
```r
uniquevalue <- unique(as.vector(as.matrix(ct_sbav_be_yld[-1])))
uniquevalue
```

```
 [1]   NA 3383 2768 2153 3075 2460 1845 3998 1538 3690  308  923  615 4920 1230
[16] 4305 6150 5843 4613
```

```
colorcount <- length(unique(as.vector(as.matrix(ct_sbav_be_yld[-1]))))
colorcount
```

```
[1] 19
```

```
heatmap_plot <- pheatmap((ct_sbav_be_yld),
                         #clustering_distance_rows = "correlation",
                         clustering_distance_rows = "euclidean",
                         clustering_distance_cols = "euclidean",
                         clustering_method = "complete",
                         angle_col = 90,
                         na_col = "white",
                         color = colorRampPalette(c("green",
                                                    "yellow",
                                                    "red"))(colorcount),
                         cellheight = 13,
                         cellwidth = 14,
                         fontsize = 12,
                         fontsize_row = 12,
                         fontsize_col = 12,
                         number_color = "white",
                         fontsize_number = 5,
                         cluster_rows = FALSE,
                         cluster_cols = FALSE,
                         show_rownames = TRUE,
                         show_colnames = TRUE,
                         display_numbers = TRUE,
                         number_format = "%.0f",
                         legend_breaks = uniquevalue
                         )
```



8	3075	3383	3383	3075	3690	4305	3075	3383	3690	3075	3690	4305	3075	3383	3690	3075	3690	4305	3075	3383	3690	3075	3690	4305	3075	3383	3690	3075
3	2460	2460	2768	2153	2768	3383	1845	2153	2460	2153	2460	3383	1845	2153	2460	2153	2460	3383	1845	2153	2460	2153	2460	3383	1845	2153	2460	1845
8	1845	1845	2153	1538	1845	2460	923	1230	1538	1230	1538	2460	615	923	1230	1230	1538	2460	615	923	1230	923	1538	2460	615	923	1230	615
0	3075	3383	3383	3075	3383	3998	3075	3383	3383	3075	3383	3998	3075	3383	3383	3075	3383	3998	3075	3383	3383	3075	3383	3998	3075	3383	3383	3075
3	2768	2768	3075	2768	2768	3383	2460	2768	2768	2460	2768	3383	2460	2460	2768	2460	2768	3383	2460	2460	2768	2460	2768	3383	2460	2460	2768	2460
5	2460	2460	2460	2153	2460	2768	1845	1845	2153	1845	2153	2768	1538	1845	2153	1845	2153	2768	1538	1845	2153	1845	2153	2768	1538	1845	1845	1538
0	2768	3075	3690	3075	3998	5843	2460	3075	3998	3075	3998	6150	2460	3075	3998	3075	3998	6150	2460	3075	3998	2768	3998	6150	2460	3075	3998	2768
3	923	1538	1845	615	1538	3383		615	1230	308	1538	3383		308	923	308	1538	3383		308	923		1230	3383			923	
5			308			923						923						923						615				
8	3075	3075	3383	3075	3383	4305	3075	3075	3383	3075	3383	4305	3075	3075	3383	3075	3383	4305	3075	3075	3383	3075	3690	4305	2768	3075	3690	3075
3	2460	2460	2768	2153	2460	3383	1845	2153	2460	2153	2460	3383	1845	2153	2460	2153	2460	3383	1845	2153	2460	2153	2460	3383	1538	1845	2153	1845
8	1845	1845	2153	1230	1538	2460	923	1230	1538	1230	1538	2153	615	923	1230	1230	1538	2153	615	923	1230	923	1538	2153	308	615	923	615
0	3075	3075	3383	3075	3383	3690	3075	3075	3383	3075	3383	3998	3075	3075	3383	3075	3383	3998	3075	3075	3383	3075	3383	3998	3075	3075	3383	3075
3	2768	2768	2768	2460	2768	3383	2460	2460	2768	2460	2768	3383	2460	2460	2768	2460	2768	3383	2460	2460	2768	2460	2768	3383	2153	2460	2768	2460
8	2153	2460	2460	2153	2153	2768	1845	1845	2153	1845	2153	2768	1538	1845	1845	1845	2153	2768	1538	1845	1845	1845	2153	2460	1538	1538	1845	1538
0	2768	3075	3690	2768	3998	5843	2460	3075	3998	2768	3998	5843	2460	3075	3998	2768	3998	5843	2460	3075	3998	2768	3998	6150	2153	3075	3998	2768
3	923	1538	1845	615	1538	3383		308	923	308	1230	3383			923	308	1230	3383			923		1230	3383			615	
5						923						615						615						615				
0	3075	3075	3383	3075	3383	4305	2768	3075	3383	3075	3383	4305	2768	3075	3383	3075	3383	4305	2768	3075	3383	3075	3383	4305	2768	3075	3383	3075

```r
ggsave(heatmap_plot,
       height = 8,
       width = 12,
       units = "in",
       file = paste0("Plots/SBAV Breakeven Yield R50", ".png"))
rm(heatmap_plot); rm(colorcount); rm(uniquevalue)
```

4.8 Plot Strawberry Profit by Panels

You can see plot breakdown based on yield variation, crop price, and electricity price. You can see variation for all solar proportion in one facet of the chart. Each facet of the chart contain av profit three heights of solar panels, four regions of AL, two array types.

```r
combinations <- expand.grid(
  yldvar = c(0, 0.1, 0.3, 0.5, 0.7, 1, 1.20, 1.5, 1.80, 2), # Yield
  price = c(3, 6, 9), # Strawberry price
  elcprc = c(0.03, 0.04, 0.05) # Electricity price
)

# Iterate over the combinations and create the plots
for (combo in seq_len(nrow(combinations))) {
  filtered_data <- sbav_profit %>%
    filter(
      yldvar == combinations$yldvar[combo],
      price == combinations$price[combo],
      elcprc == combinations$elcprc[combo]
    )
  # If by panel, put panels below in color and group.
  sbav_sp_plot <- ggplot(data = filtered_data,
                        mapping = aes(x = al_regs,
                                      y = sbav_profit,
                                      color = factor(panels),
                                      group = factor(panels))) +
    geom_line() +
    geom_point() +
    facet_grid(height ~ array,
               labeller = as_labeller(
                 c(
                   "4.6" = "4.6 ft. Height",
                   "6.4" = "6.4 ft. Height",
                   "8.2" = "8.2 ft. Height",
                   Tracking = "Single Axis Rotation",
                   Fixed = "Fixed Open Rack"
                   ))) +
    guides(color = guide_legend(ncol = 1,
                                reverse = TRUE)) +
    scale_x_discrete(limits = c("Northern", "Central",
                                "Black Belt", "Southern"),
                     labels = c("North", "Center",
                                "B Belt", "South")) +
    guides(color = guide_legend(ncol = 2,
                                reverse = TRUE)) +
    labs(x = "Regions of Alabama",
         y = "Profit ($) from Strawberry Agrivoltaic System",
         color = "Number of Solar \n Panels per Acre",
         title = (list(combinations[combo,]))
         ) +
    theme(strip.background = element_blank())
  # Add horizontal line at y = 0 if y has both positive and negative values
  if (min(filtered_data$sbav_profit) < 0 &
      max(filtered_data$sbav_profit) > 0) {
    sbav_sp_plot <- sbav_sp_plot +
      geom_hline(yintercept = 0,
                 linewidth = 0.30,
                 linetype = "dashed",
                 color = "black")
  }
  print(combinations[combo,])
  print(sbav_sp_plot)
  ggsave(file = paste0("Plots/sbav_sp_R50", combo, ".png"))
  #break
}
rm(combinations, combo)
```

4.9 Plot Strawberry Profit by Yields

You can see plot breakdown based on solar proportion, crop price, and electricity price. You can see variation for all crop yield variation in one facet of the chart. Each facet of the chart contain av profit three heights of solar panels, four regions of AL, two array types.

```
combinations <- expand.grid(
  sprop = c(0, 0.25, 0.50, 0.75, 1.00), # Solar proportion
  price = c(3, 6, 9), # Strawberry price
  elcprc = c(0.03, 0.04, 0.05) #Electricity price
)

# Iterate over the combinations and create the plots
for (combo in seq_len(nrow(combinations))) {
  filtered_data <- sbav_profit %>%
    filter(
      sprop == combinations$sprop[combo],
      price == combinations$price[combo],
      elcprc == combinations$elcprc[combo]
    )
  # If by yield, put yield below in color and group.
  sbav_yv_plot <- ggplot(data = filtered_data,
                        mapping = aes(x = al_regs,
                                      y = sbav_profit,
                                      color = factor(yield),
                                      group = factor(yield))) +
    geom_line() +
    geom_point() +
    facet_grid(height ~ array,
               labeller = as_labeller(
                 c(
                   "4.6" = "4.6 ft. Height",
                   "6.4" = "6.4 ft. Height",
                   "8.2" = "8.2 ft. Height",
                   Tracking = "Single Axis Rotation",
                   Fixed = "Fixed Open Rack"
                   ))) +
    guides(color = guide_legend(ncol = 1,
                                reverse = TRUE)) +
    scale_x_discrete(limits = c("Northern", "Central",
                                "Black Belt", "Southern"),
                     labels = c("North", "Center",
                                "B Belt", "South")) +
    guides(color = guide_legend(ncol = 2,
                                reverse = TRUE)) +
    labs(x = "Regions of Alabama",
         y = "Profit ($) from Strawberry Agrivoltaic System",
         color = "Strawberry Yield \n (25 Lb Buckets)",
         title = (list(combinations[combo,]))
         ) +
    theme(strip.background = element_blank())
  # Add horizontal line at y = 0 if y has both positive and negative values
  if (min(filtered_data$sbav_profit) < 0 &
      max(filtered_data$sbav_profit) > 0) {
    sbav_yv_plot <- sbav_yv_plot +
      geom_hline(yintercept = 0,
                 linewidth = 0.30,
                 linetype = "dashed",
                 color = "black")
  }
  print(combinations[combo,])
  print(sbav_yv_plot)
  ggsave(file = paste0("Plots/sbav_yv_R50", combo, ".png"))
  #break
}
rm(combinations, combo)
```