AV Profit

Bijesh Mishra, Ph.D.

2024-10-28

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

```
# #| echo: TRUE
rm(list = ls()) # Clean the environment.
options(
  warn=0, # Warnings. options(warn=-1) / options(warn=0)
  scipen=999 # No scientific notations.
  )
```

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)
```

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

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

```
[1] 814968     29
```

2.1.1 Calculate tavp_wocp

- 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.

```
# 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
```

```r
    ) {
  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
```

```r
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
```

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

2.1.2 TAV Profit > Tomato Alone

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

```r
# 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, ]
```

```r
    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
```

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

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

2.2 Strawberry AV

See tomato for variable descriptions.

sbav_profit = total profit from solar and strawberry.

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

```
[1] 814968     29
```

2.2.1 Calculate sbvp_wocp

- 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
# Calculate the profit:
# Step 1: Filter the dataframe to get the unique profit values for each price when yldva
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: Create the new variable sbavp_wocp by subtracting the unique profit from sqav_
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 -7509.045
32      4 -4434.045
53      5 -1359.045
74      6  1715.955
95      7  4790.955
116     8  7865.955
137     9 10940.955
```

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

2.2.2 SBAV Profit > Strawberry Alone

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
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); rm(split_data); rm(process_subset)
```

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

3 Tomato AV Results

3.1 tav_profit Crosstab

```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)
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, 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") # 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 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 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.
# Display the result matrix
rm(result_matrix); rm(sprop); rm(array); rm(height);
rm(elcprc); rm(price); rm(yldvar); rm(al_regs)
```

```
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
```

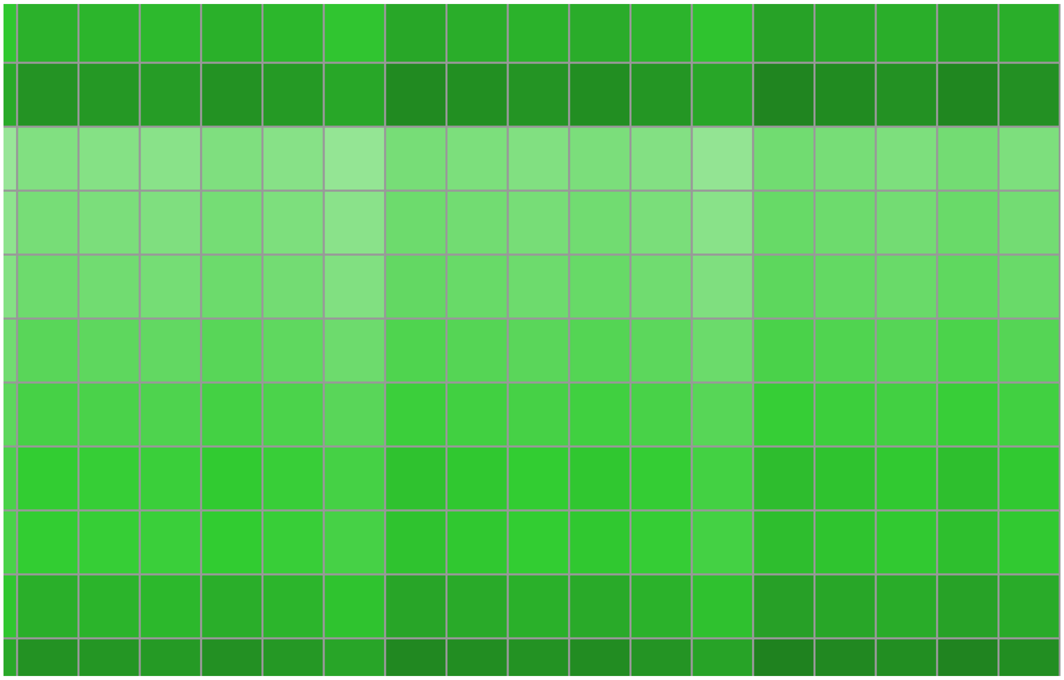
3.2 tav_profit Heatmap

- Heatmap of 324*30 dimension matrix
- Tomato profit.

```
# 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_format = "%.2f",
```

```
  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/TAV Profits CTab R50", ".png"))
#rm(colorcount); rm(heatmap_plot)
```

3.3 tav_profit manuscript

```
# Define the values for each variable
sprop <- c(0, 0.25, 0.50, 0.75, 1.00)
array <- c("Fixed", "Tracking") # Solar Array
```

```r
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

# 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,
```

```r
                                  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
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
```

```r
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)
# Display the result matrix
rm(result_matrix); rm(sprop); rm(array); rm(height);
rm(elcprc); rm(price); rm(yldvar); rm(al_regs)
```

3.4 tavp_wocp Crosstab

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

```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") # 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
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
```

```r
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 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],
```

```r
                     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.
dim(ct_tavp_wocp);rm(result_matrix)
```

```
[1] 1764  126
```

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

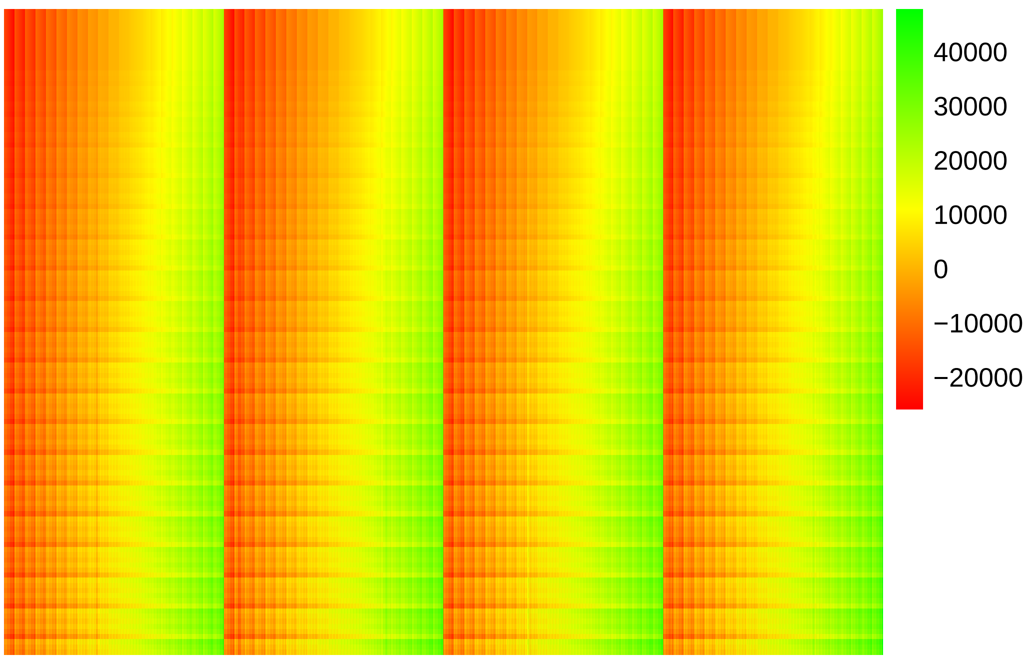
3.5 tavp_wocp Heatmap

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

```
[1] 150092
```

```r
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_be_yld 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))
```

```r
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.
dim(ct_tav_be_yld); rm(result_matrix)
```

```
[1] 36 48
```

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

3.7 tav_be_yld Heatmap

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

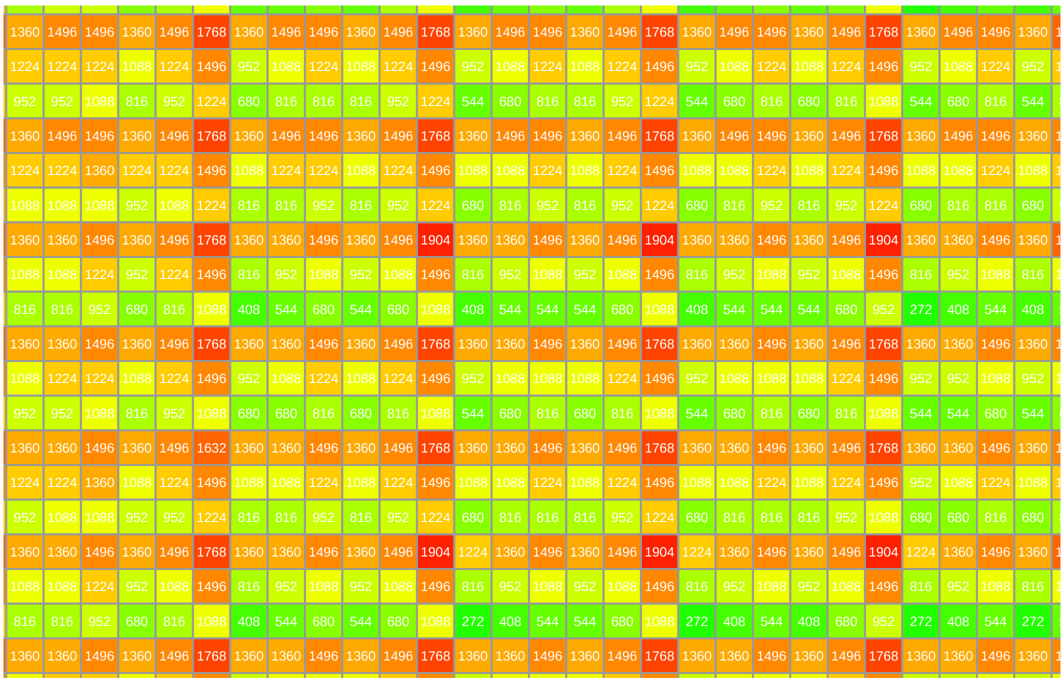
```
 [1]   NA 1496 1360 1224 1088  952 1768 1632  816  680 1904  544  408  272 2040
[16]  136
```

```
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
                    )
```




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

3.8 Plotting 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.

```
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))) {
  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")) +
```

```r
    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
}
```

3.9 Plotting 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.

```r
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
}
```

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]))
```

```r
# 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 (including elcprc
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.
# Display the result matrix
ct_sbav_pft <- as.data.frame(result_matrix) # Table in Excel.
rm(result_matrix); rm(sprop); rm(array); rm(height);
rm(elcprc); rm(price); rm(yldvar); rm(al_regs)
```

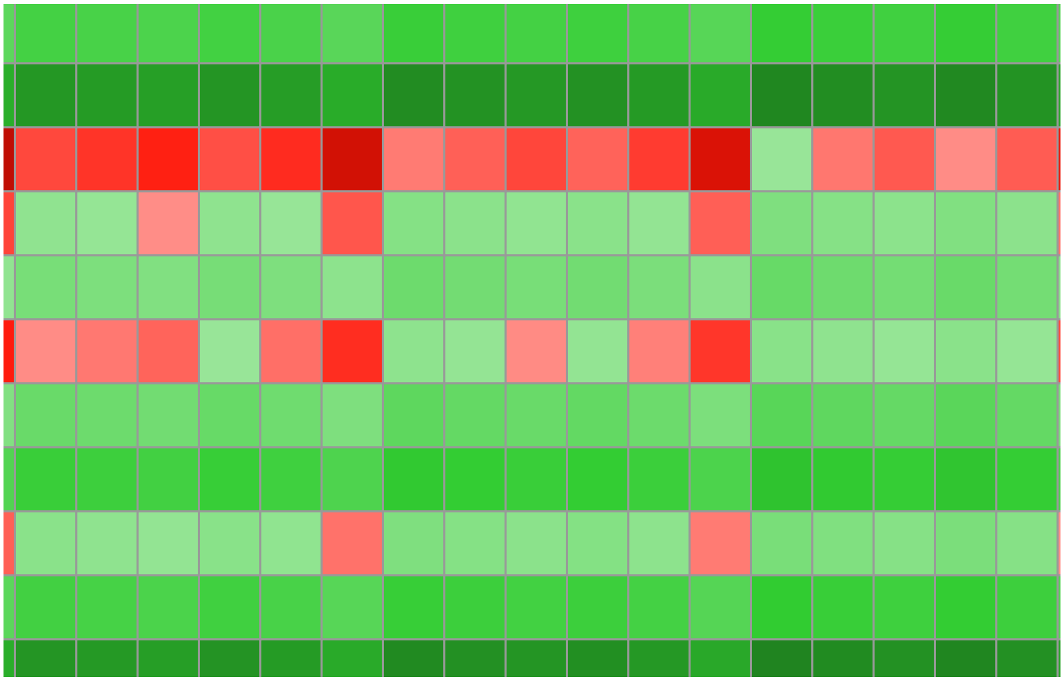
```
write_xlsx(x = ct_sbav_pft %>%
  dplyr::mutate(Row_Names = rownames(ct_sbav_pft)) %>%
  dplyr::select(Row_Names, everything()),
           file = "Results/Profit Ctab SBAV R50.xlsx",
           as_table = TRUE)
dim(ct_sbav_pft)
```

```
[1] 36 60
```

4.2 sbav_profit Heatmap

```
# 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
```

```r
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 = FALSE,
  number_format = "%.2f",
  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"))
```

4.3 sbav_profit 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(9) # Crop Price
elcprc <- c(0.04) # Electricity Price

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

```r
# 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)

# Merge with tav_profit to get tav_profit values for each combination
merged_data <- merge(param_combinations,
                     sbav_profit,
                     by = required_columns,
```

```r
                     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.

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)
# Display the result matrix
rm(result_matrix); rm(sprop); rm(array); rm(height);
rm(elcprc); rm(price); rm(yldvar); rm(al_regs)
```

4.4 sbavp_wocp 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.

```
# 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)
```

```
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)
```

4.5 sbavp_wocp Heatmap

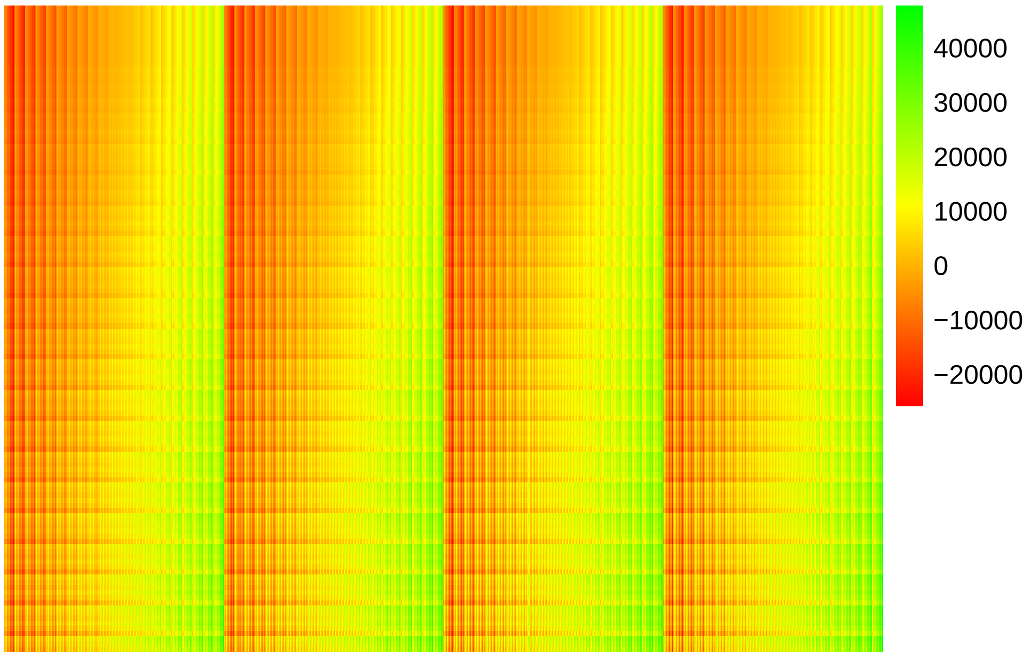
- Heatmap of 324*30 dimension matrix.

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

```
[1] 149651
```

```
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)
rm(colorcount)
```

4.6 sbav_be_yld 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
```

```r
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,
```

```r
                                   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)

# 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.
```

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

4.7 sbav_be_yld Heatmap

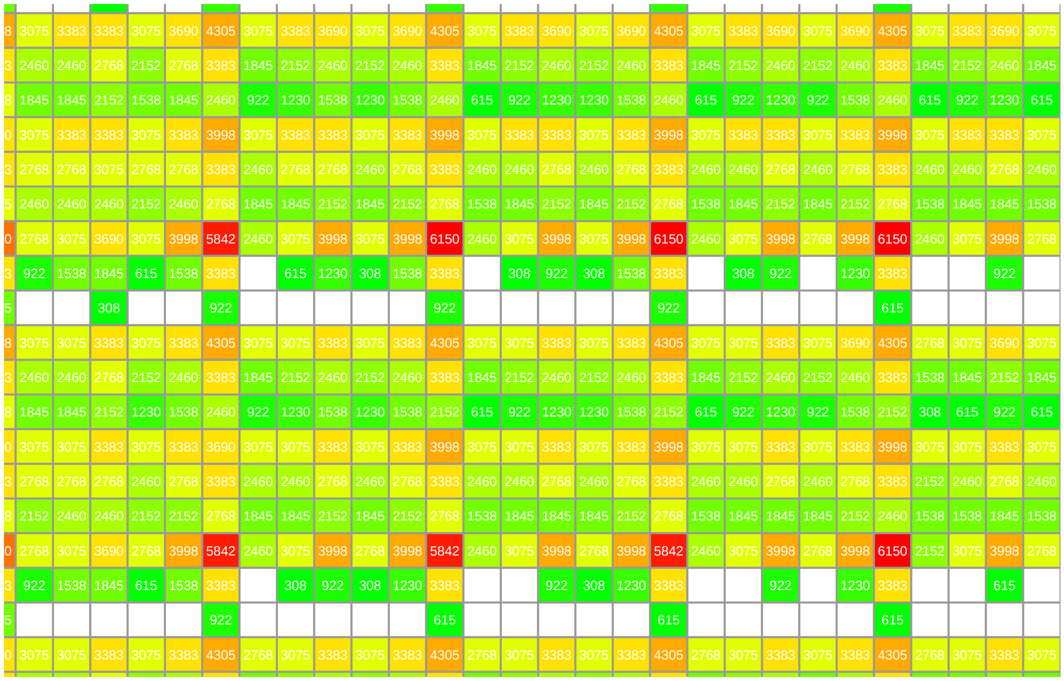
```
uniquevalue <- unique(as.vector(as.matrix(ct_sbav_be_yld[-1])))
uniquevalue
```

```
 [1]   NA 3383 2768 2152 3075 2460 1845 3998 1538 3690  308  922  615 4920 1230
[16] 4305 6150 5842 4612
```

```
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
                         )
```



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

4.8 Plotting 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
}
```

4.9 Plotting 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
}
```