AV Profit

Bijesh Mishra, Ph.D.

Table of contents

1 Setting Up **2**
- 1.1 Housekeeping . . . 2
- 1.2 Working directory . . . 2
- 1.3 Load libraries . . . 2
- 1.4 Theme for plots . . . 3

2 Import data **5**
- 2.1 Tomato AV . . . 5
- 2.2 Strawberry AV . . . 6
- 2.3 Squash AV . . . 6

3 Tabulating Results **7**
- 3.1 Tomato AV . . . 7
  - 3.1.1 Plotting Tomato Profit . . . 10
- 3.2 Strawberry AV . . . 10
  - 3.2.1 Plotting Strawberry Profit . . . 13
- 3.3 Squash AV . . . 13
  - 3.3.1 Plotting Squash Profit . . . 16

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

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 Working directory

Codes and output are suppressed. Errors and warnings are visible. No warning and no error means code is working as it should.

```
path_mac = "/Users/bmishra/Library/CloudStorage/OneDrive-AuburnUniversity/Collaboration/Ngbe
path_office = "Users\\bzm0094\\OneDrive - Auburn University\\Collaboration\\Ngbede M\\Choice
ifelse(Sys.info()[6] == "bmishra",
       setwd(path_mac),
       setwd(path_office))
```

1.3 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 becor
```

```
library(psych, warn.conflicts = FALSE, quietly = TRUE)
library(likert,  warn.conflicts = FALSE, quietly = TRUE) # Likert Items
library(mice,  warn.conflicts = FALSE, quietly = TRUE)
library(openxlsx2, warn.conflicts = FALSE, quietly = TRUE)
library(ggpubr, warn.conflicts = FALSE, quietly = TRUE) # Scatter plot
library(gmodels,  warn.conflicts = FALSE, quietly = TRUE) # Crosstab
library(reshape2, warn.conflicts = FALSE, quietly = TRUE) # Reshape data
library(pacman,  warn.conflicts = FALSE, quietly = TRUE) # Package Management
library(progress, warn.conflicts = FALSE, quietly = TRUE) #progress bar
library(arrow, warn.conflicts = FALSE, quietly = TRUE) #feather
```

```
Some features are not enabled in this build of Arrow. Run `arrow_info()` for more informatior
The repository you retrieved Arrow from did not include all of Arrow's features.
You can install a fully-featured version by running:
`install.packages('arrow', repos = 'https://apache.r-universe.dev')`.
```

```
pacman::p_loaded()
```

```
 [1] "arrow"     "progress"  "pacman"    "reshape2"  "gmodels"   "ggpubr"
 [7] "openxlsx2" "mice"      "likert"    "xtable"    "psych"     "lubridate"
[13] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
[19] "tibble"    "ggplot2"   "tidyverse"
```

1.4 Theme for plots

Setting theme for plots:

```
####### Plotting Data: #####
# Map Theme:
plottheme <- ggplot() +
  theme_void() +
  # Mapping theme:
  theme(axis.title = element_blank(),
        axis.ticks = element_blank(),
        axis.text = element_blank(),
        panel.border = element_blank(),
```

```
        plot.margin = margin(t = 0,
                             r = 0,
                             b = 0,
                             l = 0,
                             unit = "cm"),
        plot.title = element_text(hjust = 0.5),
        plot.background = element_rect(fill = "white",
                                       color = "black",
                                       linewidth = 0),
        panel.background = element_rect(fill = "white",
                                        color = "black",
                                        linewidth = 0),
        panel.grid.major.x = element_line(color = "lightgrey",
                                          linetype = 2,
                                          linewidth = 0),
        panel.grid.minor.x = element_line(color = "lightgrey",
                                          linetype = 2,
                                          linewidth = 0),
        panel.grid.major.y = element_line(color = "grey",
                                          linetype = 2,
                                          linewidth = 0),
        panel.grid.minor.y = element_line(color = "grey",
                                          linetype = 2,
                                          linewidth = 0),
        axis.line.x.top = element_line(color = "white",
                                       linetype = 2,
                                       linewidth = 0),
        axis.line.y.right = element_line(color = "white",
                                         linetype = 2,
                                         linewidth = 0),
        axis.line.x.bottom = element_line(color = "black",
                                          linetype = 1,
                                          linewidth = 0),
        axis.line.y.left = element_line(color = "black",
                                        linetype = 1,
                                        linewidth = 0),
        # Text formatting:
        text = element_text(family = "serif", # font
                            size = 12, # font size
                            colour = "black"# font color
        ),
        legend.position = c(0.95, -0.05),
```

```
        legend.key = element_rect(color = "black",
                                  fill = NA,
                                  linewidth = 0.05,
                                  linetype = 1),
        legend.justification = "right",
        legend.direction = "horizontal")
```

```
Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
3.5.0.
i Please use the `legend.position.inside` argument of `theme()` instead.
```

2 Import data

Import necessary data.

2.1 Tomato AV

sprop = proportion of solar in agrivoltaic system (0 to 1 in 0.5 increment.)

al_regs = four regions of Alabama. Northern, Central, Black Belt, Southern.

array = Solar array; Sun tracking (Tracking) and non-tracking (Fixed).

dc_kw = DC system size (kW) See PVWatts® Calculator.

panels = number of solar panels.

energy = total energy generated from solar system. See: PVWatts® Calculator.

elecprc = electricity price (1 cents to 6 cents).

height = clearance height of solar panels. 4.6 ft., 6.4 ft., and 8.2 ft.

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.

eprofit = profit from electricity.

eannprof = annualized total profit from electricity.

emonprof = monthly total profit from electricity.

yldvar = crop yield variation (10% to 200%)

yield = crop yield variation based on yldvar.

price = crop yield price per bucket.

profit = profit from crops.

tav_profit = total profit from solar and tomato.

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

```
[1] 776160     21
```

```
#str(tav_profit)
#head(tav_profit); head(tav_profit)
```

2.2 Strawberry AV

See tomato for variable descriptions.

sbav_profit = total profit from solar and strawberry.

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

```
[1] 776160     21
```

```
#str(sbav_profit)
#head(sbav_profit); tail(sbav_profit)
```

2.3 Squash AV

See tomato for variable descriptions.

sqav_profit = total profit from solar and squash.

```
sqav_profit <- as.data.frame(read_feather(file = "sqav_profit.feather"))
dim(sqav_profit)
```

```
[1] 776160     21
```

```
#str(sqav_profit)
#head(sqav_profit); tail(sqav_profit)
```

3 Tabulating Results

3.1 Tomato AV

```
# Define the values for each variable
sprop <- c(0, 0.25, 0.50, 0.75, 1.00) # Land Proportion
array <- c("Fixed", "Tracking") # Solar Array
height <- c(4.6, 6.4, 8.2) # Panel height
yldvar <- c(0.10, 0.30, 0.50, 0.70, 1.00, 1.20, 1.50, 1.80, 2.00) # Crop Yield Variation
al_regs <- c("Northern", "Central", "Black Belt", "Southern") # Regions of AL
price <- c(17, 20, 23) # Crop Price
elcprc <- c(0.02, 0.03, 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
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], "_",
                     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$tav_profit, 2)
  }
}
tav_chtbl <- as.data.frame(result_matrix) #Table in Excel.
rm(result_matrix)
```

```
write.csv(as.data.frame(tav_chtbl),
          row.names = TRUE,
          col.names = TRUE,
          file = "tav_chtbl.csv")
```

```
Warning in write.csv(as.data.frame(tav_chtbl), row.names = TRUE, col.names =
TRUE, : attempt to set 'col.names' ignored
```

```
dim(tav_chtbl)
```

```
[1] 324  30
```

- 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 Tomato AV system.

```
# Display the result matrix
#head(tav_chtbl)
#tail(tav_chtbl)
names(tav_profit)
```

```
 [1] "sprop"      "al_regs"    "array"      "dc_kw"      "panels"
 [6] "energy"     "elcprc"     "elcrev"     "height"     "capex"
[11] "ttlcost"    "anncost"    "moncost"    "eprofit"    "eannprof"
[16] "emonprof"   "yldvar"     "yield"      "price"      "profit"
[21] "tav_profit"
```

3.1.1 Plotting Tomato Profit

- Result suppressed.

```
tavp_plot <- tav_profit
ggplot(data = tav_plot,
       mapping = aes(x = pprop,
                     y = tav_profit,
                     color = factor(yldvar),
                     group = factor(yield))) +
  geom_line() +
  geom_point() +
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "black") +
  guides(color = guide_legend(ncol = 2,
                              reverse = TRUE))
```

```
Error in ggplot(data = tav_plot, mapping = aes(x = pprop, y = tav_profit, : object 'tav_plot
```

3.2 Strawberry AV

```
# Define the values for each variable
sprop <- c(0, 0.25, 0.50, 0.75, 1.00)
array <- c("Fixed", "Tracking")
height <- c(4.6, 6.4, 8.2)
yldvar <- c(0.10, 0.30, 0.50, 0.70, 1.00, 1.20, 1.50, 1.80, 2.00)
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(3, 6, 9)
elcprc <- c(0.02, 0.03, 0.04)

# 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]))
```

```r
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$sbav_profit, 2)
  }
}
sbav_chtbl <- as.data.frame(result_matrix) #Table in Excel.
rm(result_matrix)
```

```r
write.csv(as.data.frame(sbav_chtbl),
          row.names = TRUE,
          col.names = TRUE,
          file = "sbav_chtbl.csv")
```

```
Warning in write.csv(as.data.frame(sbav_chtbl), row.names = TRUE, col.names =
TRUE, : attempt to set 'col.names' ignored
```

```r
dim(sbav_chtbl)
```

```
[1] 324  30
```

- 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
# Display the result matrix
#head(sbav_chtbl)
#tail(sbav_chtbl)
names(sbav_profit)
```

```
 [1] "sprop"       "al_regs"     "array"       "dc_kw"       "panels"
 [6] "energy"      "elcprc"      "elcrev"      "height"      "capex"
[11] "ttlcost"     "anncost"     "moncost"     "eprofit"     "eannprof"
[16] "emonprof"    "yldvar"      "yield"       "price"       "profit"
[21] "sbav_profit"
```

3.2.1 Plotting Strawberry Profit

- Result suppressed.

3.3 Squash AV

```r
# Define the values for each variable
sprop <- c(0, 0.25, 0.50, 0.75, 1.00)
array <- c("Fixed", "Tracking")
height <- c(4.6, 6.4, 8.2)
yldvar <- c(0.10, 0.30, 0.50, 0.70, 1.00, 1.20, 1.50, 1.80, 2.00)
al_regs <- c("Northern", "Central", "Black Belt", "Southern")
price <- c(11, 14, 17)
elcprc <- c(0.02, 0.03, 0.04)

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

# Check if the columns exist in sqav_profit
missing_columns <- setdiff(required_columns,
                           names(sqav_profit))
if (length(missing_columns) > 0) {
  stop("Missing columns in sqav_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 sqav_profit values for each combination
merged_data <- merge(param_combinations,
                     sqav_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 sqav_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$sqav_profit, 2)
  }
}
sqav_chtbl <- as.data.frame(result_matrix) #Table in Excel.
rm(result_matrix)
```

```
write.csv(as.data.frame(sqav_chtbl),
          row.names = TRUE,
          file = "sqav_chtbl.csv")
dim(sqav_chtbl)
```

```
[1] 324  30
```

- 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 Squash AV system.

```
# Display the result matrix
#head(sqav_chtbl)
#tail(sqav_chtbl)
names(sqav_profit)
```

```
 [1] "sprop"      "al_regs"    "array"      "dc_kw"      "panels"
 [6] "energy"     "elcprc"     "elcrev"     "height"     "capex"
[11] "ttlcost"    "anncost"    "moncost"    "eprofit"    "eannprof"
[16] "emonprof"   "yldvar"     "yield"      "price"      "profit"
[21] "sqav_profit"
```

3.3.1 Plotting Squash Profit

- Result suppressed.