Complete Beginner's Guide to Three-Way Bar Plot R Code

What This Code Does

This R script creates a statistical bar chart that compares plant height across different stress conditions, treatments, and varieties. It performs advanced statistical tests and creates a publication-ready graph with error bars and significance letters.

Section 1: Comments and Documentation

```r
# ============================================================================
# THREE-WAY BAR PLOT WITH STATISTICAL ANALYSIS
```

What are comments?

- Lines starting with `#` are **comments** - they don't run as code
- Comments explain what the code does
- They help you (and others) understand the code later
- Always use comments to document your work!

Section 2: Loading Libraries

```r
library(ggplot2)      # For plotting
library(dplyr)        # For data manipulation
library(multcompView) # For statistical significance letters
```

What are libraries?

- **Libraries** (also called packages) are collections of pre-written functions
- Think of them as toolboxes with specialized tools
- You must load them before using their functions

The three libraries explained:

1. **ggplot2**: Creates beautiful, professional graphs
2. **dplyr**: Helps manipulate and summarize data easily
3. **multcompView**: Adds statistical significance letters to graphs

Installing packages (if needed):

```r
install.packages(c("ggplot2", "dplyr", "multcompView"))
```

- Only run this once per computer
- Like downloading an app - you only download once, but open it each time you use it

---

Section 3: Setting Up Your Workspace

```r
setwd("E:/aov")               # Set working directory
data_file <- "df.csv"         # Input CSV file
dv <- "PH"                    # Dependent variable
y_label <- "Plant Height (cm)"  # Y-axis label for plot
```

`setwd("E:/aov")`

- **setwd** = "set working directory"
- Tells R where to look for files and save outputs
- Like opening a specific folder on your computer
- **Change this path to match your computer!**

Variable Assignment (`<-`)

- `<-` is the assignment operator in R
- It stores values in named containers (variables)
- `data_file <- "df.csv"` means "store the text 'df.csv' in a variable called data_file"

Variables explained:

- `data_file`: Name of your CSV file containing the data
- `dv`: Dependent variable - what you're measuring (PH = Plant Height)

- `y_label`: The label that will appear on the y-axis of your graph

---

Section 4: Reading Data

```r
dt <- read.csv(data_file)
names(dt)
```

`read.csv()`

- Reads CSV (Comma Separated Values) files into R
- CSV files are like Excel spreadsheets saved in a simple format
- The data gets stored in a **data frame** (like a table)

`names(dt)`

- Shows the column names of your dataset
- Helps you see what variables you have to work with
- `dt` is the name we gave to our dataset

What your data should look like:

- Columns: stress, treat, var, PH (and possibly others)
- Each row represents one plant measurement
- Multiple measurements per treatment combination

Section 5: Statistical Analysis - ANOVA

```r
anova_formula <- as.formula(paste(dv, "~ stress * treat * var"))
anova <- aov(anova_formula, data = dt)
print(summary(anova))
```

Understanding ANOVA

- **ANOVA** = Analysis of Variance
- Tests if different groups have significantly different means
- **Three-way ANOVA** compares three factors simultaneously

Breaking down the code:

`paste(dv, "~ stress * treat * var")`

- `paste()` combines text together
- Creates the formula: "PH ~ stress * treat * var"
- `~` means "is explained by"
- `*` means "and all interactions between"

`as.formula()`

- Converts text into a formula that R understands
- Formulas tell R how variables relate to each other

`aov()`

- Performs the ANOVA test
- Takes the formula and dataset as inputs
- Returns statistical results

`print(summary(anova))`

- Shows the ANOVA results table
- Displays F-values, p-values, and significance

Section 6: Post-hoc Testing

```r
tukey <- TukeyHSD(anova)
cld <- multcompLetters4(anova, tukey)
cld <- as.data.frame.list(cld$`stress:treat:var`)
```

Why do we need post-hoc tests?

- ANOVA tells us "groups are different" but not "which groups are different"
- **Tukey's HSD** compares all pairs of treatments

- **HSD** = Honestly Significant Difference

`TukeyHSD(anova)`

- Performs all pairwise comparisons
- Controls for multiple testing (prevents false positives)

`multcompLetters4()`

- Converts p-values into letter codes
- Treatments with the same letter are NOT significantly different
- Treatments with different letters ARE significantly different

Example of letters:

- Treatment A gets letter "a"
- Treatment B gets letter "b" (significantly different from A)
- Treatment C gets letter "a" (not significantly different from A)

Section 7: Data Summarization

```
r

dt_summary <- dt %>%
  group_by(stress, treat, var) %>%
  summarise(
    mean_val = mean(.data[[dv]], na.rm = TRUE),
    se_val = sd(.data[[dv]], na.rm = TRUE) / sqrt(n()),
    sample_size = n(),
    .groups = "drop"
  ) %>%
  arrange(desc(mean_val)) %>%
  mutate(Tukey = cld$Letters)
```

The Pipe Operator (`%>%`)

- Reads as "then do"
- Passes the result of one function to the next
- Makes code easier to read (like a recipe with steps)

Breaking down each step:

`group_by(stress, treat, var)`

- Groups data by all combinations of the three factors
- Like sorting data into separate piles

`summarise()`

- Calculates summary statistics for each group
- Creates one row per treatment combination

Inside summarise():

- `mean_val`: Average value for each group
- `se_val`: Standard error (measure of uncertainty)
- `sample_size`: Number of observations per group
- `.data[[dv]]`: Refers to the column specified in `dv`
- `na.rm = TRUE`: Ignores missing values
- `sqrt(n())`: Square root of sample size (for standard error calculation)

`arrange(desc(mean_val))`

- Sorts results by mean value (highest to lowest)
- `desc()` means descending order

`mutate(Tukey = cld$Letters)`

- Adds the significance letters to our summary table

Section 8: Setting Graph Limits

```
r

y_max <- max(dt_summary$mean_val + 2 * dt_summary$se_val) * 1.15
```

What this does:

- Finds the highest point on the graph (mean + 2 standard errors)

- Adds 15% buffer space above
- Ensures significance letters fit on the graph
- `max()` finds the maximum value
- `*` means multiplication

Section 9: Printing Results

```r
print(data.frame(Grand_Mean = mean(dt[[dv]], na.rm = TRUE)))
print(data.frame(CV_Percent = (sd(residuals(anova), na.rm = TRUE) / mean(dt[[dv]], na.rm = TRUE
print(dt_summary)
```

These print important statistics:

Grand Mean:

- Overall average across all treatments
- `mean()` calculates the average

CV% (Coefficient of Variation):

- Measures how much variation exists in your data
- `sd()` calculates standard deviation
- `residuals()` gets the unexplained variation from ANOVA
- Lower CV% = more consistent data

Summary Table:

- Shows means, standard errors, and significance letters for each treatment

---

Section 10: Tukey HSD Critical Value

```r
mse <- summary(anova)[[1]]$`Mean Sq`[nrow(summary(anova)[[1]])]
n_groups <- nrow(dt_summary)
n_replicates <- mean(dt_summary$sample_size)
q_crit <- qtukey(0.95, n_groups, anova$df.residual)
hsd_value <- q_crit * sqrt(mse / n_replicates)
print(data.frame(HSD_value = hsd_value))
```

This calculates the minimum difference needed for significance:

- `mse`: Mean Square Error from ANOVA (measure of variability)
- `n_groups`: Number of treatment combinations
- `n_replicates`: Average number of plants per treatment
- `qtukey()`: Gets the critical value from Tukey distribution
- `hsd_value`: The actual critical difference

What HSD value means:

- If two treatments differ by more than this value, they're significantly different
- If they differ by less, they're not significantly different

---

Section 11: Creating the Graph

```r
p <- ggplot(dt_summary, aes(x = treat, y = mean_val, fill = stress)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(aes(ymax = mean_val + se_val, ymin = mean_val - se_val),
                position = position_dodge(0.9),
                width = 0.25,
                color = "Gray25") +
  geom_text(aes(label = Tukey, y = mean_val + se_val + 0.05 * y_max),
            size = 2.5,
            color = "Gray25",
            position = position_dodge(1)) +
  labs(y = y_label) +
  ylim(0, y_max) +
  facet_grid(~factor(var, levels = c("v1", "v2"))) +
  theme_bw() +
  theme(
    axis.title.x = element_blank(),
    legend.position = "none",
    text = element_text(size = 12),
    axis.title = element_text(size = 10),
    axis.text = element_text(size = 10)
  )
```

Understanding ggplot2 Grammar

`ggplot(dt_summary, aes(...))`

- `ggplot()`: Main plotting function
- `dt_summary`: The data to plot
- `aes()`: "aesthetics" - tells ggplot which variables to use for what
  - `x = treat`: Treatment goes on x-axis
  - `y = mean_val`: Mean values go on y-axis
  - `fill = stress`: Stress levels determine bar colors

The `+` operator:

- In ggplot, `+` adds layers to your graph
- Each `geom_` function adds a different visual element

Graph Layers Explained:

`geom_bar(stat = "identity", position = "dodge")`

- `geom_bar()`: Creates bar chart
- `stat = "identity"`: Use the actual values (not counts)
- `position = "dodge"`: Place bars side-by-side (not stacked)

`geom_errorbar(...)`

- Adds error bars showing standard error
- `ymax/ymin`: Top and bottom of error bars
- `position_dodge(0.9)`: Aligns with bars
- `width = 0.25`: How wide the error bar caps are
- `color = "Gray25"`: Color of error bars

`geom_text(...)`

- Adds significance letters above bars
- `label = Tukey`: Uses our significance letters
- `y = mean_val + se_val + 0.05 * y_max`: Positions letters above error bars
- `size = 2.5`: Text size
- `position_dodge(1)`: Aligns with bars

`labs(y = y_label)`

- Sets axis labels
- `y_label` contains "Plant Height (cm)"

`ylim(0, y_max)`

- Sets y-axis limits from 0 to our calculated maximum

`facet_grid(~factor(var, levels = c("v1", "v2")))`

- Creates separate panels for each variety
- `factor(..., levels = ...)` controls the order of panels

`theme_bw()`

- Applies a clean, black-and-white theme
- Professional appearance for publications

`theme(...)`

- Customizes specific theme elements
- `axis.title.x = element_blank()`: Removes x-axis title
- `legend.position = "none"`: Removes legend
- Various text size specifications

Section 12: Display and Save

```
r

print(p)

ggsave(
  filename = paste0(dv, "_plot.tiff"),
  plot = p,
  width = 3,
  height = 2,
  dpi = 300,
  compression = "lzw",
  bg = "white"
)
```

`print(p)`

- Displays the graph on screen
- In R, you need to print ggplot objects to see them

`ggsave()`

- Saves the graph as a file
- `filename`: Name of output file (e.g., "PH_plot.tiff")
- `plot = p`: Which graph to save
- `width/height`: Dimensions in inches
- `dpi = 300`: High resolution for publication
- `compression = "lzw"`: Reduces file size
- `bg = "white"`: White background

Key R Concepts for Beginners

1. Data Types

- **Numeric**: Numbers (1.5, 3.14, 100)
- **Character**: Text ("hello", "PH", "treatment")
- **Logical**: TRUE or FALSE
- **Factor**: Categories (stress levels, varieties)

2. Data Structures

- **Vector**: Single column of data
- **Data Frame**: Table with rows and columns (like Excel)
- **List**: Container that can hold different types of data

3. Functions

- Functions perform specific tasks
- Format: `function_name(arguments)`
- Example: `mean(c(1, 2, 3))` calculates the average
- Arguments go inside parentheses, separated by commas

4. Operators

- `<-`: Assigns values to variables
- `+`, `-`, `*`, `/`: Mathematical operations
- `==`: Tests if equal
- `!=`: Tests if not equal
- `%>%`: Pipe operator (passes data forward)

5. Getting Help

- `?function_name`: Get help for a specific function
- `??search_term`: Search for functions
- `help.search("keyword")`: Broader search

Understanding Your Output

ANOVA Table

- **Df**: Degrees of freedom
- **Sum Sq**: Variation explained by each factor
- **Mean Sq**: Average variation per degree of freedom
- **F value**: Test statistic
- **Pr(>F)**: P-value (probability of seeing this result by chance)
- **Significance codes**: * = p < 0.05, ** = p < 0.01, *** = p < 0.001

Summary Statistics

- **Grand Mean**: Overall average across all treatments
- **CV%**: Coefficient of variation (lower = more consistent)
- **HSD Value**: Minimum difference for significance

Graph Interpretation

- **Bar height**: Mean value for each treatment
- **Error bars**: Show uncertainty (standard error)
- **Letters**: Treatments with same letters are not significantly different
- **Panels**: Separate graphs for each variety

Common Beginner Mistakes and Solutions

1. File Path Issues

Problem: `Error: cannot change working directory` **Solution**: Check that your file path exists and use forward slashes (/) or double backslashes (\)

2. Missing Packages

Problem: `Error: there is no package called 'ggplot2'` **Solution**: Install the package first: `install.packages("ggplot2")`

3. Data Reading Errors

Problem: `Error: object 'dt' not found` **Solution**: Make sure your CSV file is in the working directory and has the correct name

4. Column Name Errors

Problem: `Error: object 'PH' not found` **Solution**: Check column names with `names(dt)` and ensure `dv` matches exactly

5. Factor Level Issues

Problem: Bars appear in wrong order **Solution**: Use `factor(variable, levels = c("order", "you", "want"))`

Next Steps for Learning R

1. **Practice with your own data**: Modify the `dv` and `y_label` variables
2. **Experiment with colors**: Change the `fill` aesthetic or add `scale_fill_manual()`
3. **Try different graph types**: Replace `geom_bar()` with `geom_point()` or `geom_boxplot()`
4. **Learn more statistics**: Explore other tests like t-tests or regression
5. **Improve graphs**: Add titles, change themes, adjust colors

Remember: Everyone starts as a beginner! The key is to practice regularly and don't be afraid to experiment. R has excellent documentation and a helpful community online.