task7_0930

2025-09-30

```
library(ggplot2)  # Plotting with the grammar of graphics
library(readr)
library(dplyr)   # Data manipulation (filter, mutate, etc.)
set.seed(123)     # So that any random demo data will be the same across runs

csv_path <- file.path(rmd_dir, "magic_guys.csv")  # robust path anchored at the .Rmd folder

if (file.exists(csv_path)) {                                                # Check if th
e CSV actually exists
  raw_df <- read_csv(csv_path, show_col_types = FALSE)                      # Read the CS
V with readr (guess column types)
} else {
  # Fallback demo data (so the document still knits even without the CSV) -------
  # Assumption: two species with slightly different height distributions
  raw_df <- tibble::tibble(
    species = rep(c("Species_A", "Species_B"), each = 200),                 # Two groups
of 200 each
    height  = c(rnorm(200, mean = 170, sd = 7), rnorm(200, mean = 165, sd = 6))  # Heights in
cm (example)
  )
  message("NOTE: 'magic_guys.csv' not found. Using a simulated demo dataset.")    # Friendly m
essage in the console
}

# Peek at column names to help us map the right fields ---------------------------
names(raw_df)                                                               # Print the c
olumn names to the console
```

```
## [1] "uniqId"  "species" "length"  "weight"
```

```r
# Try to detect the height column (flexible to different naming) -----------------
# We look for a column name containing "height" or "body" and that is numeric
candidate_height_cols <- names(raw_df)[grepl("height|body", tolower(names(raw_df)))]  # Find possible matches by name
if (length(candidate_height_cols) == 0) {                                          # If we found none by name
  # If none by name, also consider numeric columns as a fallback -----------------
  numeric_cols <- names(raw_df)[sapply(raw_df, is.numeric)]                        # All numeric columns
  height_col <- numeric_cols[1]                                                    # Take the first numeric as a last resort
} else {
  # If there are name matches, prefer the first one ------------------------------
  height_col <- candidate_height_cols[1]                                           # Use the first matching column as height
}

# Try to detect the species/grouping column (flexible naming) --------------------
candidate_species_cols <- names(raw_df)[grepl("species|type|group|category", tolower(names(raw_df)))]  # Common labels
if (length(candidate_species_cols) == 0) {
  stop("Could not find a species/group column. Please rename a column to 'species' (or 'type'/'group').")  # Stop with a clear message
} else {
  species_col <- candidate_species_cols[1]        # Use the first matching species-like column
}

# Standardize to a clean, consistent data frame
df <- raw_df %>%
  dplyr::rename(                                  # Rename to standard names used below
    species = !!species_col,                      # Rename detected species column to 'species'
    height  = !!height_col                       # Rename detected height column to 'height'
  ) %>%
  dplyr::select(species, height) %>%            # Keep only the two columns we need
  dplyr::mutate(
    species = as.factor(species)              # Ensure species is a factor (categorical)
  )

# Remove rows with missing or non-finite height values
df <- df %>% dplyr::filter(is.finite(height))     # Drop NA/Inf/NaN heights to avoid plotting issues

# Confirm that we have (at least) two species
length(levels(df$species))
```

```
## [1] 2
```

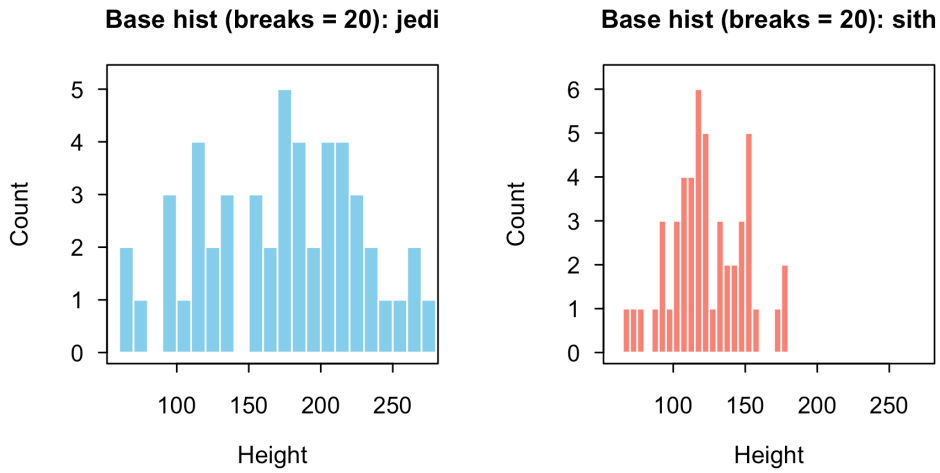
```r
# If there are more than two species, keep only the first two for this assignment
if (length(levels(df$species)) > 2) {          # If more than two species exist
  keep_lvls <- levels(df$species)[1:2]          # Take the first two levels
  df <- df %>% dplyr::filter(species %in% keep_lvls) %>% droplevels()    # Filter to those two
and drop unused levels
  message("NOTE: More than two species detected. Keeping only the first two: ",
          paste(keep_lvls, collapse = ", "))            # Let the user know what we kept
}

# Quick sanity check of the final data
summary(df)                                 # See summary stats for heights by species
```

```
##  species       height
##  jedi:50   Min.   : 60.1
##  sith:50   1st Qu.:111.3
##            Median :138.4
##            Mean   :147.8
##            3rd Qu.:178.7
##            Max.   :272.9
```

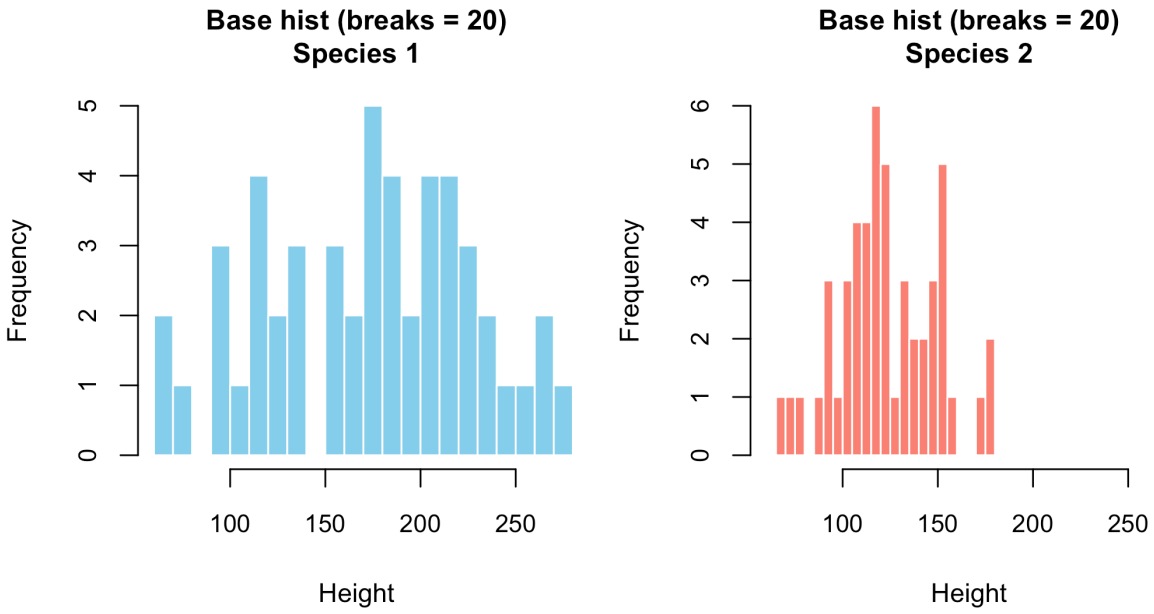
```r
# Compute a data-driven bin width using the Freedman–Diaconis rule ----------------
# FD rule: binwidth = 2 * IQR(x) / n^(1/3)  (works well for continuous data)
fd_binwidth <- function(x) {                                                    # Define a s
mall helper function
  x <- x[is.finite(x)]                                                          # Remove non
-finite values
  n <- length(x)                                                                # Sample siz
e
  if (n < 2) return(NA_real_)                                                   # Not enough
data for a bin width
  bw <- 2 * IQR(x, na.rm = TRUE) / (n^(1/3))                                    # Freedman–D
iaconis formula
  if (!is.finite(bw) || bw <= 0) return(NA_real_)                               # Guard agai
nst weird cases
  bw                                                                             # Return th
e bin width
}

bw_fd <- fd_binwidth(df$height)                                                 # Compute FD
bin width for the whole dataset
bw_fd                                                                           # Print it s
o you can see the value
```

```
## [1] 29.03101
```

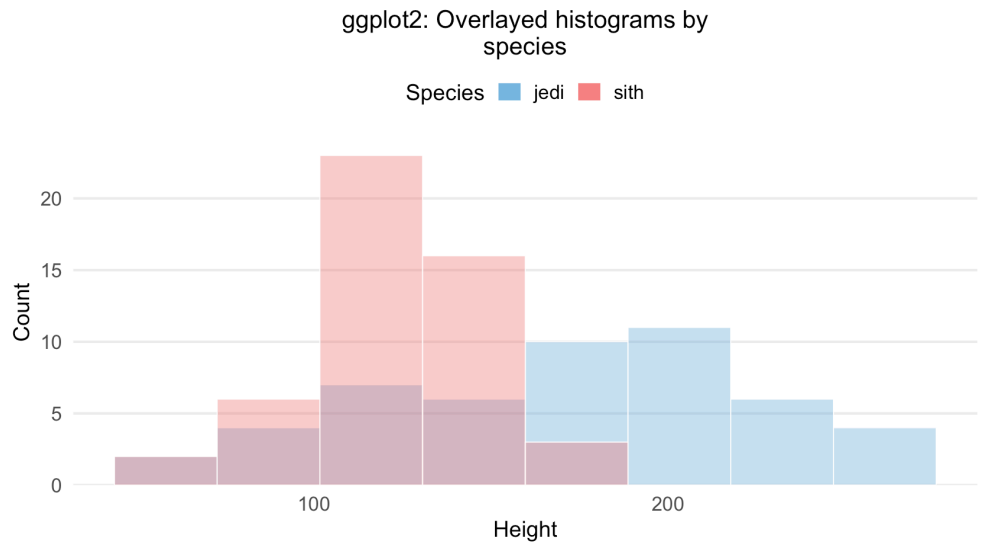
```r
# Split by species ------------------------------------------------------------
x1 <- df$height[df$species == levels(df$species)[1]]
x2 <- df$height[df$species == levels(df$species)[2]]
rng <- range(df$height, na.rm = TRUE)

# Safer layout & margins (no pin): mgp controls label-to-axis spacing ----------
op <- par(mfrow = c(1, 2),                    # 1 row × 2 columns
          mar = c(4.2, 4.2, 2.8, 0.8),       # margins (bottom, left, top, right) in text line
s
          mgp = c(2.2, 0.7, 0),              # axis title, tick labels, tick line distances
          tcl = -0.25,                       # tick length (negative = inward)
          cex.main = 0.92, cex.lab = 0.88, cex.axis = 0.85)

# Left panel ------------------------------------------------------------------
h1 <- hist(x1, breaks = 20, plot = FALSE)    # compute first to get y-limits
plot(h1, col = "skyblue", border = "white",
     xlim = rng, ylim = c(0, max(h1$counts) * 1.05),
     main = paste0("Base hist (breaks = 20): ", levels(df$species)[1]),
     xlab = "Height", ylab = "Count", axes = FALSE)  # turn off default axes
axis(1)                                      # draw x-axis
axis(2, las = 1)                             # draw y-axis; las=1 makes labels horizontal
box()                                        # frame around plot

# Right panel -----------------------------------------------------------------
h2 <- hist(x2, breaks = 20, plot = FALSE)
plot(h2, col = "salmon", border = "white",
     xlim = rng, ylim = c(0, max(h2$counts) * 1.05),
     main = paste0("Base hist (breaks = 20): ", levels(df$species)[2]),
     xlab = "Height", ylab = "Count", axes = FALSE)
axis(1)
axis(2, las = 1)
box()

par(op)                                      # restore
```



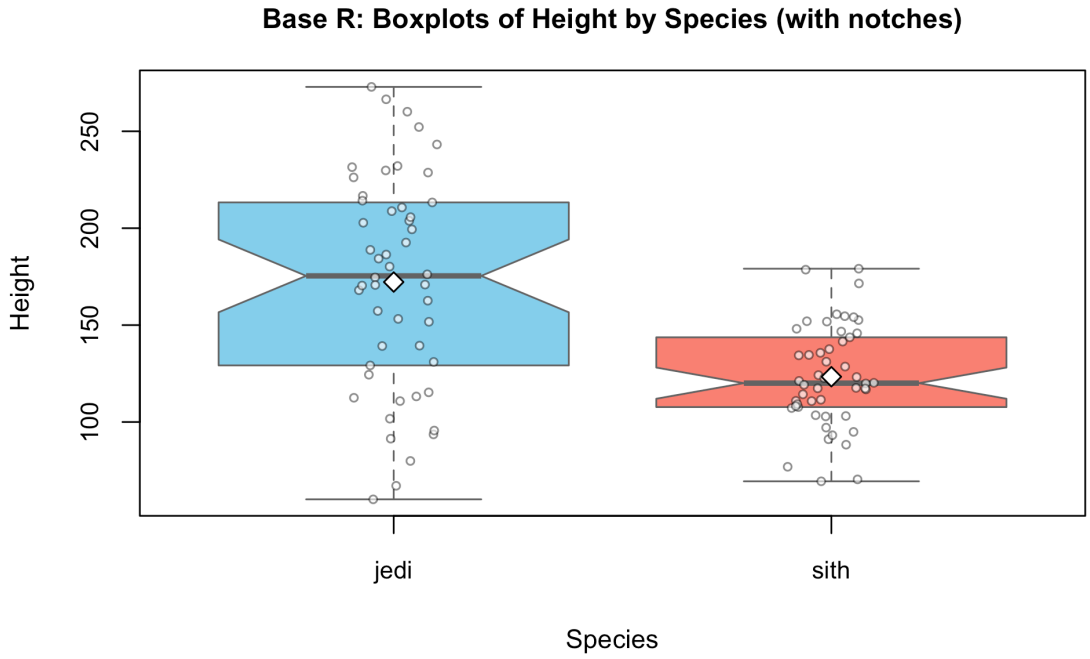
```r
# fig.width/fig.height: physical size in inches of the plotting device
# out.width: scale the rendered image in HTML (e.g., 80% of page width)
# dpi: resolution for crisp text/lines
# fig.align: center the figure

op <- par(mfrow = c(1, 2),                    # 1 row x 2 columns layout
          mar   = c(4, 4, 3.2, 1),            # margins: bottom, left, top, right (lines)
          cex.main = 0.95, cex.lab = 0.9, cex.axis = 0.85)  # shrink fonts a bit

fixed_breaks <- 20                             # same breaks for both panels

hist(x1,
     breaks = fixed_breaks,
     col = "skyblue", border = "white", xlim = rng,
     main = "Base hist (breaks = 20)\nSpecies 1",
     xlab = "Height")

hist(x2,
     breaks = fixed_breaks,
     col = "salmon", border = "white", xlim = rng,
     main = "Base hist (breaks = 20)\nSpecies 2",
     xlab = "Height")

par(op)                                        # restore par
```



```r
# Fixed-size, compact overlayed histogram (smaller & cleaner)

# 1) Keep your bin settings ----------------------------------------------------
use_binwidth <- is.finite(bw_fd)                                  # TRUE if FD bin width is avai
lable
chosen_bins  <- 30                                                # Fallback number of bins
chosen_bw    <- if (use_binwidth) bw_fd else NA_real_             # FD bin width or NA

# 2) Libraries for nice title wrapping & sizing --------------------------------
library(stringr)                                                  # str_wrap() for wrapping long
titles
library(grid)                                                     # unit() for legend key sizes

# 3) Draw a compact overlayed histogram ----------------------------------------
ggplot(df, aes(x = height, fill = species)) +                     # Map height to x; species con
trols fill color
  geom_histogram(
    position = "identity",                                        # Overlay instead of stacking
    alpha    = 0.35,                                              # Lighter transparency for cle
aner overlap
    bins     = if (!use_binwidth) chosen_bins else NULL,          # Use bins if no binwidth
    binwidth = if (use_binwidth) chosen_bw else NULL,             # Otherwise use FD bin width
    color    = "white",                                           # Thin white bin edges
    linewidth = 0.2                                               # Slim edge lines to avoid hea
viness
  ) +
  scale_fill_manual(values = c("#4EA5D9", "#F25F5C")) +           # Pleasant, high-contrast pale
tte
  scale_y_continuous(expand = expansion(mult = c(0, 0.05))) +     # Tighten top/bottom padding
  labs(
    title = str_wrap("ggplot2: Overlayed histograms by species", width = 35),  # Wrapped title
fits small size
    x = "Height", y = "Count", fill = "Species"                   # Axis/legend labels
  ) +
  theme_minimal(base_size = 10) +                                 # Smaller global base font for
compact look
  theme(
    plot.title   = element_text(size = 10, hjust = 0.5, margin = margin(b = 4)), # Smaller cen
tered title
    axis.title   = element_text(size = 9),                        # Smaller axis titles
    axis.text    = element_text(size = 8),                        # Smaller tick labels
    panel.grid.minor = element_blank(),                           # Remove minor grid for clarit
y
    panel.grid.major.x = element_blank(),                         # Remove vertical grid lines
    legend.position = "top",                                      # Legend on top
    legend.title    = element_text(size = 9),                     # Smaller legend title
    legend.text     = element_text(size = 8),                     # Smaller legend text
    legend.key.height = unit(8, "pt"),                            # Compact legend key height
    legend.key.width  = unit(10, "pt"),                           # Compact legend key width
    plot.margin = margin(t = 4, r = 6, b = 4, l = 6)              # Tight outer margins
  ) +
  guides(fill = guide_legend(nrow = 1, byrow = TRUE,              # Put legend in a single compa
ct row
                             override.aes = list(alpha = 0.8))) # Make legend swatches more vi
sible
```

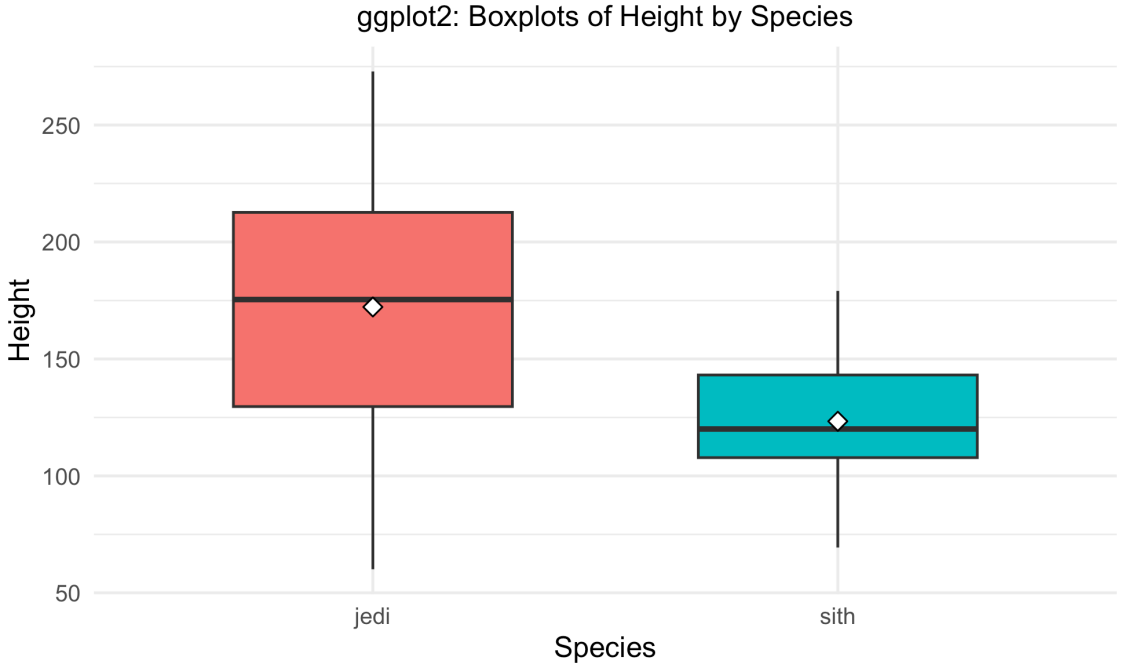



```
# Ensure the prepared data.frame 'df' exists and has the required columns --------
stopifnot(exists("df"), all(c("species","height") %in% names(df)))  # Fail early if not availa
ble

# Tweak margins and font sizes so long titles are not clipped --------------------
op <- par(mar = c(4.2, 4.8, 3.0, 1.0),         # Outer margins in lines: bottom, left, top, ri
ght
          cex.main = 0.95, cex.lab = 0.9,      # Slightly smaller title and axis labels
          cex.axis = 0.85)                     # Slightly smaller tick labels

# Draw side-by-side boxplots of height by species --------------------------------
boxplot(height ~ species, data = df,           # Formula: response ~ group
        notch = TRUE,                          # Notches visualize median CI (rough)
        col = c("skyblue", "salmon"),          # Fill colors for the two species
        border = "gray40",                     # Box borders
        outline = TRUE,                        # Draw outlier points (can set FALSE to hide)
        main = "Base R: Boxplots of Height by Species (with notches)",  # Plot title
        xlab = "Species",                      # X-axis label
        ylab = "Height")                       # Y-axis label

# Overlay raw data as jittered points for distribution insight -------------------
stripchart(height ~ species, data = df,        # Same formula as boxplot
           vertical = TRUE,                    # Keep vertical orientation
           method = "jitter",                  # Jitter to avoid overplotting
           pch = 21,                           # Point shape (filled circle)
           col = rgb(0, 0, 0, 0.5),            # Point border color (semi-transparent black)
           bg  = rgb(1, 1, 1, 0.6),            # Point fill color (semi-transparent white)
           cex = 0.6,                          # Point size
           add = TRUE)                         # Add on top of the boxplot

# Add species-wise means as diamond markers --------------------------------------
means <- tapply(df$height, df$species, mean, na.rm = TRUE)  # Compute mean per species
points(x = seq_along(means), y = means,        # X positions 1,2,...; Y at means
       pch = 23, cex = 1.2, bg = "white")      # Diamond (pch=23) with white fill
```




```
# Optional: flip axes to horizontal boxplots (uncomment next line) ---------------
# boxplot(height ~ species, data = df, horizontal = TRUE)

# Restore previous par settings --------------------------------------------------
par(op)
```

```
# Load ggplot2 in this chunk (safe even if already loaded) -----------------------
library(ggplot2)                                  # Grammar of graphics plotting

# Build a boxplot comparing height distributions by species ----------------------
ggplot(df, aes(x = species, y = height, fill = species)) +  # Map species to x/fill, height to
y
  geom_boxplot(width = 0.6,                       # Box width (relative to spacing)
               outlier.shape = 21,                # Outlier point shape (filled circle)
               outlier.alpha = 0.6) +             # Outlier transparency
  stat_summary(fun = mean, geom = "point",        # Add group means as points
               shape = 23, size = 2.6, fill = "white") +  # Diamond marker with white fill
  labs(title = "ggplot2: Boxplots of Height by Species",   # Plot title
       x = "Species", y = "Height", fill = "Species") +    # Axis/legend labels
  theme_minimal(base_size = 12) +                 # Clean theme with slightly smaller base font
  theme(plot.title = element_text(size = 12,      # Control title size and centering
                                  hjust = 0.5,
                                  margin = margin(b = 6)),
        legend.position = "none",                 # Hide redundant legend (colors match species
on x)
        plot.margin = margin(t = 8, r = 8, b = 8, l = 8))  # Add outer margins to avoid clippi
ng
```




```
# Optional: show raw data points (uncomment the next line) ---------------------
# + geom_jitter(width = 0.12, alpha = 0.5, size = 0.9)    # Jittered points over boxes
# Optional: flip axes if labels are long (uncomment the next line) ------------
# + coord_flip()
```