STAT 240: The Bob Ross Case Study

Solutions

Fall 2024

1. Elements by Episode

```r
all_elements <- read_csv("../../data/elements-by-episode.csv")
head(all_elements)
```

1.1 Are Any Titles Repeated?

```r
# Write code to determine if which values of TITLE, if any, appear multiple times.
all_elements %>%
  count(TITLE) %>%
  filter(n > 1)

all_elements %>% filter(TITLE == "\"LAKESIDE CABIN\"" |
                          TITLE == "\"MOUNTAIN WATERFALL\"") %>% select(EPISODE, TITLE)
```

1.2 Which Features Appear The Most?

```r
# Write code here to remove the frame and host columns and filter to Bob Ross' episodes.
# Save this dataframe, which has just EPISODE, TITLE, and the painting-based features, to painting_elem
painting_elements <- all_elements %>%
  filter(GUEST == 0) %>%
  select(-c(DIANE_ANDRE, STEVE_ROSS, GUEST, contains("FRAME")))

# painting_elements should have 381 rowsand 50 columns.
dim(painting_elements)
```

```r
# Approach 1: Using across()
painting_elements %>%
  summarize(across(AURORA_BOREALIS:WINTER, mean))
```

```r
# Approach 2: Using pivoting, and then group_by

# We want to have each row be a different painting-feature combination.
# There should be a column for the feature and a column for whether or not it appears.

painting_elements %>%
  pivot_longer(-c(EPISODE, TITLE),
               names_to = "feature", values_to = "appeared")

# We should end up with 381 Bob Ross episodes x 48 features = 18,288 rows.
# Now, we can sum up the "appeared" column separately for each different "feature".
painting_element_counts = painting_elements %>%
  pivot_longer(-c(EPISODE, TITLE),
               names_to = "feature", values_to = "appeared") %>%
  group_by(feature) %>%
  summarize(percentage = mean(appeared)) %>%
  arrange(desc(percentage))

painting_element_counts
```

1.3 Visualization

```r
# Graph base: features on the left, with horizontal bars for percentages
ggplot(painting_element_counts, aes(y = feature, x = percentage)) +
  geom_col()
```

```r
# Take out all the very rare features and reorder bars
painting_element_counts %>%
  filter(percentage > 0.02) %>%
ggplot(aes(y = reorder(feature, percentage), x = percentage)) +
  geom_col()
```

```r
# Add the percentage labels through geom_text()
painting_element_counts %>%
  filter(percentage > 0.02) %>%
ggplot(aes(y = reorder(feature, percentage), x = percentage)) +
  geom_col() +
  # label should be set within aes() because it is a variable aesthetic
  geom_text(aes(label = round(percentage*100)), hjust = "left")
```

```r
# From here it's all purely visual modifications
painting_element_counts %>%
  filter(percentage > 0.02) %>%
  # Change the feature names from "THIS_FORMAT" to "This format"
  mutate(feature = str_to_sentence(feature),
         feature = str_replace(feature, "_", " ")) %>%

ggplot(aes(y = reorder(feature, percentage), x = percentage)) +
  # Fill the bars dodgerblue
  geom_col(fill = "dodgerblue") +
  geom_text(aes(label = round(percentage*100)), hjust = "left", size = 3) +
  # Reduce that space between the feature labels and the bars
  scale_x_continuous(
    expand = expansion(mult = c(0, 0.1))
  ) +

  # Get rid of the grid and x axis labels, make the background all light gray
  theme(
    panel.grid = element_blank(),
    axis.text.x = element_blank(),
    plot.background = element_rect(fill = "gray96", color = "gray96"),
    plot.title.position = "plot",
    plot.title = element_text(face = "bold")
  ) +
  # Add plot labels and take away axis titles
  labs(
    title = "The Paintings of Bob Ross",
    subtitle = "Percentage containing each element",
    x = "",
    y = ""
  )


# new command: ggsave! Saves the last ggplot you created.
ggsave("MyBobRossPlot.png")
```

2. Colors by Episode

```r
painting_colors <- read_csv("../../data/bob_ross_paintings.csv")

nrow(painting_colors)
```

```r
painting_colors %>%
  head()
```

2.1 Visualization

```r
# Write code here to visualize the distribution of num_colors.
ggplot(painting_colors) +
  geom_bar(aes(x = num_colors),
               color = "black", fill = "dodgerblue")  +

  # Improved axis labeling
  scale_x_continuous(
    breaks = 0:15
  )

# A histogram or density plot would be a worse choice.
```

2.2 What's that Outlier?

```r
# Investigate outlier here!
painting_colors %>%
  filter(num_colors == 1) %>%
  relocate(img_src, youtube_src, season, episode, painting_title)
```

3. Joining the Two Datasets

```r
# Approach 1: format episode and season into EPISODE
# This combines boolean logic and string manipulation
painting_colors %>%
  mutate(seasonChar = ifelse(season < 10, str_c("S0", season), str_c("S", season)),
         episodeChar = ifelse(episode < 10, str_c("E0", episode), str_c("E", episode)),
         EPISODE = str_c(seasonChar, episodeChar)) %>%
  select(season, episode, seasonChar, episodeChar, EPISODE)

paintings_joined <- painting_colors %>%
  mutate(seasonChar = ifelse(season < 10, str_c("S0", season), str_c("S", season)),
         episodeChar = ifelse(episode < 10, str_c("E0", episode), str_c("E", episode)),
         EPISODE = str_c(seasonChar, episodeChar)) %>%
  # The type of join we use doesn't matter since they have the same 403 rows
  full_join(all_elements, join_by(EPISODE))
```

```r
# Approach 2: Split EPISODE into episode and season
all_elements %>%
  mutate(season = as.numeric(str_sub(EPISODE, 2, 3)),
         episode = as.numeric(str_sub(EPISODE, 5, 6))) %>%
  # Just to display what season and episode are
  relocate(season, episode)

all_elements %>%
    mutate(season = as.numeric(str_sub(EPISODE, 2, 3)),
          episode = as.numeric(str_sub(EPISODE, 5, 6))) %>%
  full_join(painting_colors) %>%
  select(EPISODE, TITLE, painting_title, num_colors, TREE)
```

```r
# Approach 3: Trying to coerce TITLE and painting_title to be the same
elements_temp <- all_elements %>%
  mutate(TITLE = str_remove_all(TITLE, "\""))

colors_temp <- painting_colors %>%
  mutate(painting_title = str_to_upper(painting_title))

elements_temp %>% select(TITLE) %>% head(n = 7)
colors_temp %>% select(painting_title) %>% head(n = 7)

# We should only have 403 rows!
full_join(elements_temp, colors_temp, join_by(TITLE == painting_title))
```

4. Statistical Inference

We will use `paintings_joined` to investigate some interesting inference questions.

```r
# Some light cleaning: getting rid of some columns and moving identifying columns to the front

paintings_joined = paintings_joined %>%
  # neat little trick; you can rename and reorder columns within select!
  select(ID = EPISODE, season, episode, painting_title, num_colors, AURORA_BOREALIS:WINTER, Black_Gesso

paintings_joined
```

4.1 Single Proportion

```r
# Write your code here!
structure_summary = paintings_joined %>%
  summarize(structures = sum(STRUCTURE),
    n = n())
```

```
structure_summary

# Three inputs; successes, sample size, confidence level
x <- structure_summary$structures
n <- structure_summary$n


# Calculate AC confidence interval
x_ac <- x + 2
n_ac <- n + 4

phat <- x_ac/n_ac
se <- sqrt(phat * (1 - phat)/n_ac)

cv <- qnorm(0.975)

c(phat - se*cv, phat + se*cv)
```

```
# Write code here!
gbinom(403, 0.2, scale = TRUE) +
  geom_vline(xintercept = 85)

dbinom(85, 403, 0.2)

gbinom(403, 0.2, a = 72, b = 86) +
  geom_hline(yintercept = dbinom(85, 403, 0.2))
```

```
# Write code here!
pbinom(85 - 1, 403, 0.2, lower.tail = F) + pbinom(75, 403, 0.2)
```

4.2 Difference in Proportions

```
# Write code here!
paintings_joined %>%
  group_by(GUEST) %>%
  summarize(X = sum(STRUCTURE),
            n = n())

x1 <- 84
n1 <- 381
x2 <- 1
n2 <- 22

n1_ac <- n1 + 2
n2_ac <- n2 + 2
p1_ac <- (x1+1)/n1_ac
p2_ac <- (x2+1)/n2_ac

pt_est <- p1_ac - p2_ac

se <- sqrt( p1_ac*(1-p1_ac)/n1_ac + p2_ac*(1-p2_ac)/n2_ac )

cv <- qnorm(0.95)
```

```r
c(pt_est - cv*se, pt_est + cv*se)
```

```r
# Write code here!
phat <- (x1 + x2)/(n1+n2)

se <- sqrt((phat*(1-phat)/n1) + (phat*(1-phat)/n2))

test_stat <- (x1/n1 - x2/n2) / se

test_stat

2 * pnorm(test_stat, lower.tail = F)
```

4.3 Single Mean

```r
# Write code here!
colors_summary <- paintings_joined %>%
  summarize(xbar = mean(num_colors),
            s = sd(num_colors),
            n = n())

colors_summary

xbar <- colors_summary$xbar
s <- colors_summary$s
n <- colors_summary$n

se <- s / sqrt(n)

cv <- qt(0.975, df = n-1)

c(xbar - cv*se, xbar + cv*se)
```

```r
# Write code here!
test_stat <- (xbar - 8) / se

test_stat

2 * pt(test_stat, df = n - 1, lower.tail = F)

t.test(paintings_joined$num_colors, mu = 8)
```

Difference in Means

```r
# Write code here!
colors_summary <- paintings_joined %>%
  group_by(GUEST) %>%
  summarize(xbar = mean(num_colors),
            s = sd(num_colors),
            n = n())

colors_summary
```

```r
xbar1 <- colors_summary$xbar[1]
xbar2 <- colors_summary$xbar[2]
s1 <- colors_summary$s[1]
s2 <- colors_summary$s[2]
n1 <- colors_summary$n[1]
n2 <- colors_summary$n[2]

se <- sqrt(s1^2/n1 + s2^2/n2)

test_stat <- ((xbar1-xbar2) - 0)/se
test_stat

w = (s1^2/n1 + s2^2/n2)^2 / (s1^4/(n1^2*(n1-1)) + s2^4/(n2^2*(n2-1)))

2 * pt(test_stat, df = w, lower.tail = F)

t.test(num_colors ~ GUEST, paintings_joined)
```

```r
# Write code here!
pt_est = xbar1 - xbar2


# We can use the standard error from above.

cv <- qt(0.975, df = 2)

c(pt_est - cv*se, pt_est + cv*se)
```

Linear Regression

```r
total_features_vector <- paintings_joined %>%
  select(AURORA_BOREALIS:WINTER) %>%
  select(!contains("FRAME")) %>%
  rowSums()

paintings_joined <- paintings_joined %>%
  mutate(num_features = total_features_vector)

# Write code here!
x <- paintings_joined$num_colors
y <- paintings_joined$num_features

beta_hat_1 <- cor(x, y) * sd(y) / sd(x)
beta_hat_1

# The summary p-value is for a two sided test
summary(lm(num_features ~ num_colors, paintings_joined))
```

```r
se <- 0.04748
n <- nrow(paintings_joined)

test_stat <- (beta_hat_1 - 0)/se
```

```r
pt(test_stat, df = n - 2, lower.tail = F)
```

```r
# Write code here!
cv <- qt(0.975, df = n-2)

c(beta_hat_1 - cv*se, beta_hat_1 + cv*se)
```