How to Spot a Front-Runner on The 'Bachelor' or 'Bachelorette'

Kimberly Greco

January 22, 2024

I. Introduction

The purpose of this file is to reproduce the following two figures from FiveThirtyEight's **How to Spot a Front-Runner on The 'Bachelor' or 'Bachelorette'**. We will describe each step of our workflow including importing the source data, variable cleaning, and figure generation.

Figure 1a. Original figure 1 to reproduce.

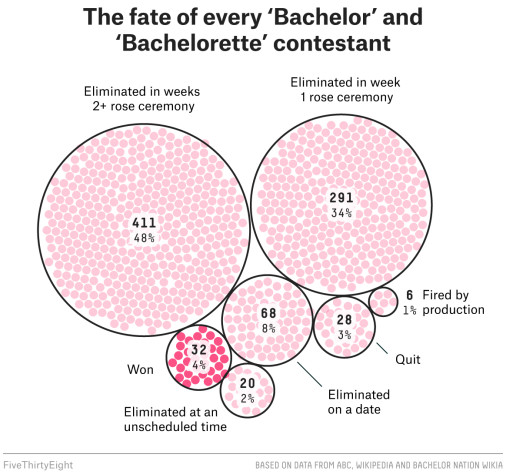
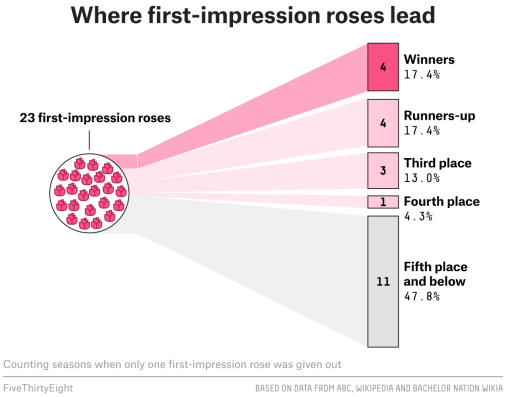


Figure 2a. Original figure 2 to reproduce.



II. Data Dictionary

Below, we've compiled a dictionary for all variables included in the replication of **Figure 1a** and **Figure 2a** from the source data `bachelorette.csv`.

Variable Name	Description
`SHOW`	Bachelor or Bachelorette
`SEASON`	Season
`CONTESTANT`	An identifier for the contestant in a given season
`ELIMINATION.1`	Who was eliminated in week 1
`ELIMINATION.2`	Who was eliminated in week 2
`ELIMINATION.3`	Who was eliminated in week 3
`ELIMINATION.4`	Who was eliminated in week 4
`ELIMINATION.5`	Who was eliminated in week 5
`ELIMINATION.6`	Who was eliminated in week 6
`ELIMINATION.7`	Who was eliminated in week 7
`ELIMINATION.8`	Who was eliminated in week 8
`ELIMINATION.9`	Who was eliminated in week 9
`ELIMINATION.10`	Who was eliminated in week 10

Within each `ELIMINATION` column, the following codes indicate the fate of each contestant each week:

- `R` contestant received rose
- `R1` contestant received first impression rose
- `E` contestant eliminated at a rose ceremony
- `ED` contestant eliminated on a date
- `EU` contestant eliminated at an unscheduled time
- `EQ` contestant quit
- `EF` contestant fired by production

III. Loading Packages and Data

We start by installing and loading all of the neccessary R packages.

```
# Install required packages
# install.packages("dplyr")
# install.packages("ggplot2")
# install.packages("knitr")

# Load the libraries
library(dplyr)
library(ggplot2)
library(knitr)
```

Next, we import the source data. For this step, ensure that `bachelorette.csv` (which can be downloaded via FiveThirtyEight's **Bachelorette GitHub Repository**) is stored in your working directory.

```
# Load original data
data <- read.csv("bachelorette.csv", header = TRUE)
```

IV. Data Cleaning

Here, we construct the necessary variables to generate each figure. For **Figure 1b**, we create `fate` which indicates how and when each contestant was eliminated from the show. For **Figure 2b**, we create `elimination` (indexes time of elimination), `ranking` (indexes ranking of elimination), and `place` (assigns a qualitative description to each ranking).

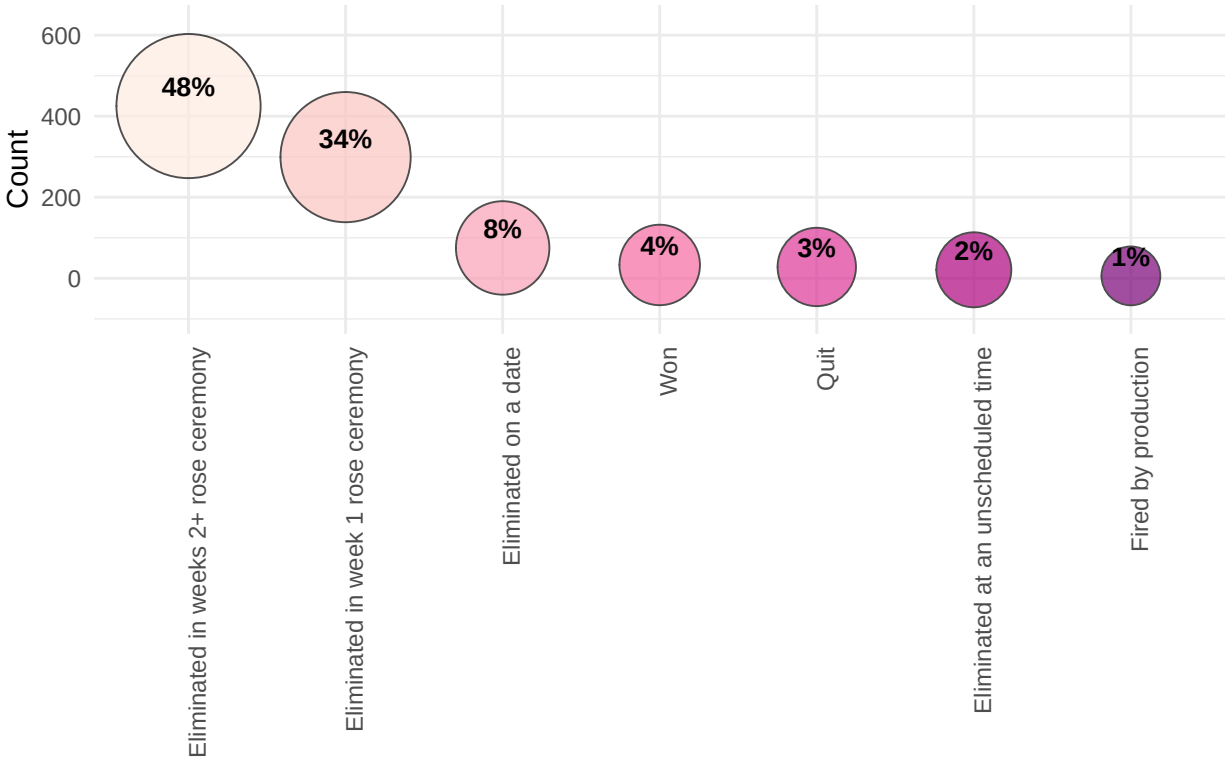
```
# Remove sub-headings to create cohesive dataframe
data <- data[data$CONTESTANT != "ID", ]

# Generate fate variable
data <- data %>%
  mutate(fate = ifelse(ELIMINATION.1 == "E",
                       "Eliminated in week 1 rose ceremony",
                       ifelse(apply(data[, paste0("ELIMINATION.", 2:10)], 1, function(x) any(grepl("^E$
                              "Eliminated in weeks 2+ rose ceremony",
                              ifelse(apply(data[, paste0("ELIMINATION.", 1:10)], 1, function(x) any(grep
                                     "Eliminated on a date",
                                     ifelse(apply(data[, paste0("ELIMINATION.", 1:10)], 1, function(x) a
                                            "Eliminated at an unscheduled time",
                                            ifelse(apply(data[, paste0("ELIMINATION.", 1:10)], 1, functi
                                                   "Fired by production",
                                                   ifelse(apply(data[, paste0("ELIMINATION.", 1:10)], 1
                                                          "Quit",
                                                          ifelse(any(grepl("W", data[paste0("ELIMINATIO
                                                                 "Won",
                                                                 NA_character_)))))))

# Generate elimination variable
data <- data %>%
  mutate(elimination = apply(select(., starts_with("ELIMINATION.")), 1, function(x) {
    last_occurrence <- max(which(x %in% c("E", "ED", "EU", "EF", "EQ", "W")))
    if (length(last_occurrence) == 0) NA else last_occurrence
  }))

# Re-index elimination for contestants who won
data <- data %>%
  mutate(elimination = ifelse(fate == "Won", 11, elimination))

# Assign elimination rank and place within each season for each show
data <- data %>%
  group_by(SHOW, SEASON) %>%
  arrange(desc(elimination)) %>%
  mutate(ranking = row_number(),
         place = case_when(
           ranking == 1 ~ "Won",
           ranking == 2 ~ "Runners-up",
           ranking == 3 ~ "Third place",
           ranking == 4 ~ "Fourth place",
           TRUE ~ "Fifth place and below"
         )) %>%
  ungroup()
```

V. Figure & Table Generation

Here, we generate a bubble plot that provides the same insights as **Figure 1a**. This is not an exact replication; it is likely that some graphic design was utilized to generate **Figure 1a**, and I am *just* a statistician. Nevertheless, the counts and percentages in **Figure 1b** approximately match **Figure 1a**, with some slight discrepancies likely due to data cleaning decisions that were made but not documented.

```
# Step 1: Figure generation
# Generate counts and percentages
fate_counts <- data %>%
  count(fate, name = "count") %>%
  mutate(percentage = round(count / sum(count) * 100,0))

# Order based on count
fate_counts$fate <- factor(fate_counts$fate, levels = fate_counts$fate[order(-fate_counts$count)])

# Generate bubble plot
figure1b <- ggplot(fate_counts, aes(x = fate, y = count, size = count, fill = fate)) +
  geom_point(alpha = 0.7, shape = 21) +
  geom_text(aes(label = paste0(round(percentage, 1), "%")), vjust = -0.5, color = "black", size = 3.5, f
  scale_size_continuous(name = "Count", range = c(10, 25)) +
  scale_fill_brewer(palette = "RdPu") +
  labs(title = "", x = "", y = "Count") +
  theme_minimal() +
  theme(legend.position = "none", axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_y_continuous(limits = c(-100, max(fate_counts$count) * 1.5))

# Step 2: Table generation
# Generate and order counts for table
fate_counts <- data %>%
  count(fate, name = "count") %>%
  arrange(desc(count))

# Generate formatted knitr table
colnames(fate_counts) <- c("Fate", "Count")
table1b <- kable(fate_counts, caption = "", align = 'lcc')
```

Figure 1b. The fate of every "Bachelor" and "Bachelorette" contestant

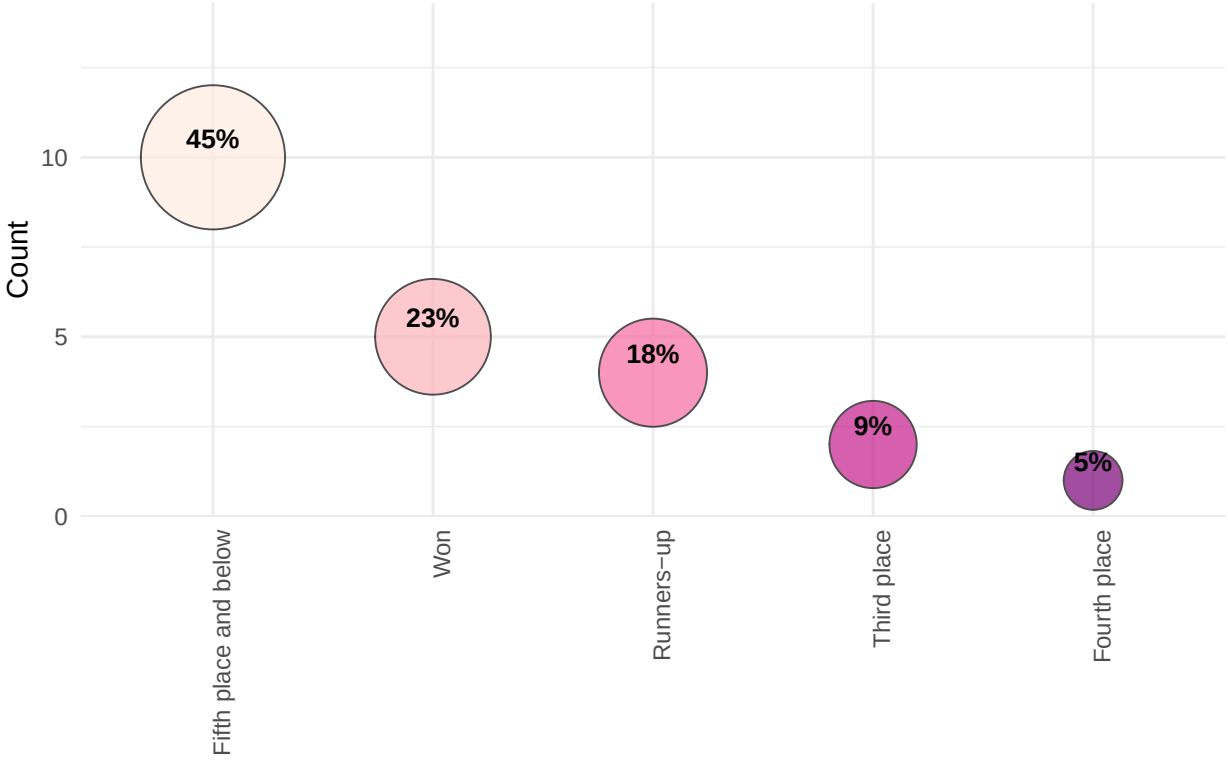


Fate	Count
Eliminated in weeks 2+ rose ceremony	425
Eliminated in week 1 rose ceremony	299
Eliminated on a date	75
Won	33
Quit	28
Eliminated at an unscheduled time	21
Fired by production	6

Here, we generate a bubble plot that provides the same insights as **Figure 2a**. Again, the counts and percentages in **Figure 2b** approximately match **Figure 2a**, with some slight discrepancies likely due to data cleaning decisions that were made but not documented.

```r
# Step 1: Figure generation
# Count the number of first impression roses (R1) given in each SHOW and SEASON combination
# Keep only seasons where 1 first impression rose is given (based on FiveThirtyEight's data cleaning st
r1_counts <- data %>%
  group_by(SHOW, SEASON) %>%
  summarise(r1_count = sum(ELIMINATION.1 == "R1", na.rm = TRUE)) %>%
  filter(r1_count == 1)

  # Join the original data with this summary to filter out the required rows
  filtered_data <- data %>%
    inner_join(r1_counts, by = c("SHOW", "SEASON")) %>%
    filter(ELIMINATION.1 == "R1")

  # Count the number of occurrences for each place
  place_counts <- filtered_data %>%
    count(place)

  # Order the factor levels of 'place' based on the count
  place_counts$place <- factor(place_counts$place, levels = place_counts$place[order(-place_counts$n)])

  # Calculate percentages
  place_counts <- place_counts %>%
    mutate(percentage = round(n / sum(n) * 100,0))

# Generate bubble plot
figure2b <- ggplot(place_counts, aes(x = place, y = n, size = n, fill = place)) +
  geom_point(alpha = 0.7, shape = 21) +
  geom_text(aes(label = paste0(round(percentage, 1), "%")), vjust = -0.5, color = "black", size = 3.5, 
  scale_size_continuous(name = "Count", range = c(10, 25)) +
  scale_fill_brewer(palette = "RdPu") +
  scale_y_continuous(expand = expansion(mult = c(0, 0.1)), limits = c(0, max(place_counts$n) * 1.3)) +
  labs(title = "", x = "", y = "Count") +
  theme_minimal() +
  theme(legend.position = "none", axis.text.x = element_text(angle = 90, hjust = 1))

# Step 2: Table generation
# Generate and order counts for table
place_counts <- filtered_data %>%
  count(place, name = "count") %>%
  arrange(desc(count))

# Generate formatted knitr table
colnames(place_counts) <- c("Fate", "Count")
table2b <- kable(place_counts, caption = "", align = 'lcc')
```

Figure 2b. Where first-impression roses lead



Fate	Count
Fifth place and below	10
Won	5
Runners-up	4
Third place	2
Fourth place	1

Note: Data for Figures 2a-b are restricted to seasons in which only one first-impression rose was given out.