myHomework2

chenpeng

2024-07-03

目录

1. Loading and cleaning

a.

```
ca_pa <- read.csv("../../data/calif_penn_2011.csv",
    header = T, encoding = "UTF-8")
```

b.

```
nrow(ca_pa)
```

```
## [1] 11275
```

```r
ncol(ca_pa)
```

```
## [1] 34
```

c.

```r
colSums(apply(ca_pa, c(1, 2), is.na))
```

```
##                            X                    GEO.id2
##                            0                          0
##                      STATEFP                   COUNTYFP
##                            0                          0
##                      TRACTCE                 POPULATION
##                            0                          0
##                     LATITUDE                  LONGITUDE
##                            0                          0
##            GEO.display.label          Median_house_value
##                            0                        599
##                 Total_units               Vacant_units
##                            0                          0
##                Median_rooms  Mean_household_size_owners
##                          157                        215
## Mean_household_size_renters         Built_2005_or_later
##                          152                         98
##           Built_2000_to_2004                 Built_1990s
##                           98                         98
##                 Built_1980s                 Built_1970s
##                           98                         98
##                 Built_1960s                 Built_1950s
##                           98                         98
##                 Built_1940s       Built_1939_or_earlier
##                           98                         98
##                  Bedrooms_0                  Bedrooms_1
##                           98                         98
##                  Bedrooms_2                  Bedrooms_3
##                           98                         98
##                  Bedrooms_4          Bedrooms_5_or_more
##                           98                         98
##                      Owners                     Renters
```

```
##                          100                          100
##     Median_household_income       Mean_household_income
##                          115                          126
```

Explanation: it counts NA values in each column of ca_pa.

d.

```
ca_pa.omit <- na.omit(ca_pa)
```

e.

```
nrow(ca_pa.omit)
```

```
## [1] 10605
```

```
nrow(ca_pa) - nrow(ca_pa.omit)
```

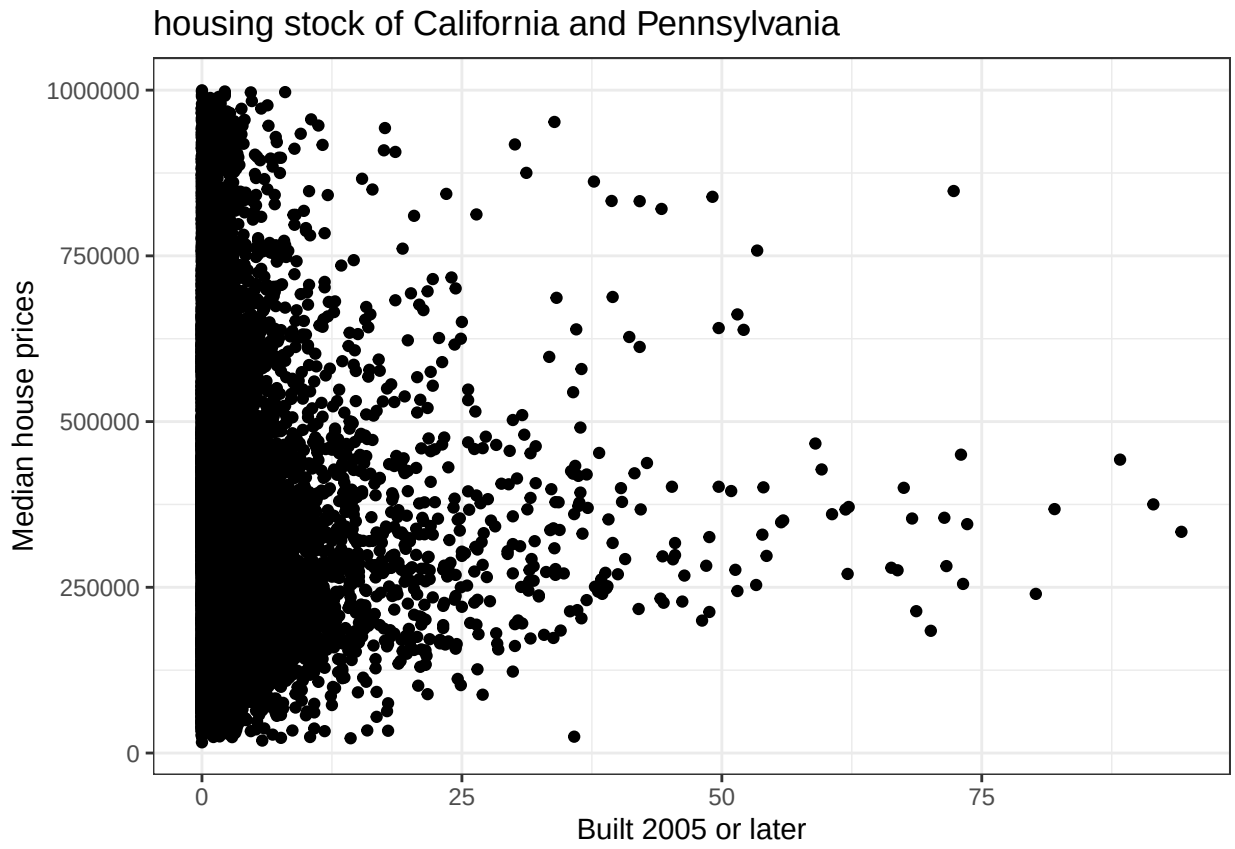
```
## [1] 670
```

f. Answer: Yes. Because some rows have different type of NA value in columns.

2. This Very New House

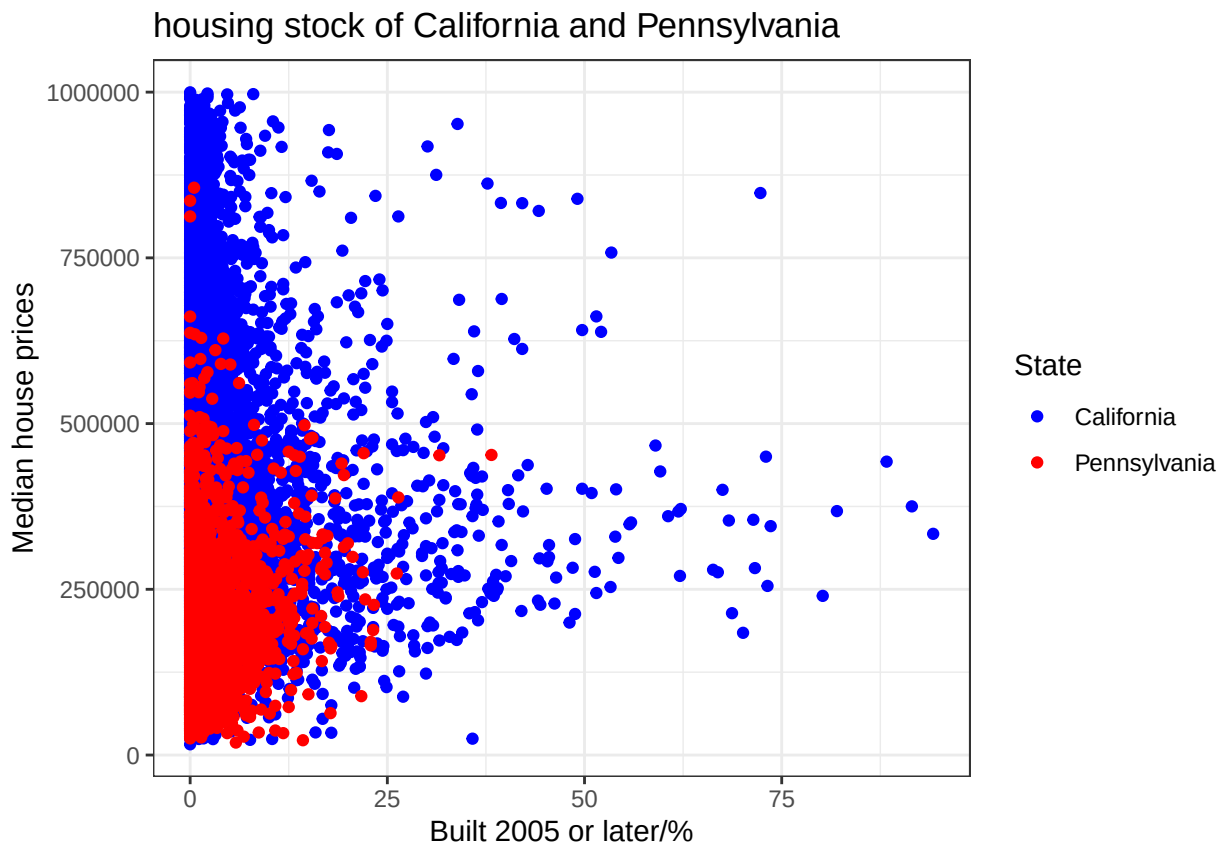
a.

```
ggplot(data = ca_pa.omit) + geom_point(aes(x = Built_2005_or_later,
    y = Median_house_value)) + labs(x = "Built 2005 or later",
    y = "Median house prices", title = "housing stock of California and Pennsylvania") +
    theme_bw()
```



b.

```
ggplot(data = ca_pa.omit) + geom_point(aes(x = Built_2005_or_later,
    y = Median_house_value, color = as.factor(STATEFP))) +
    labs(x = "Built 2005 or later/%", y = "Median house prices",
        title = "housing stock of California and Pennsylvania",
        color = "State") + scale_color_manual(values = c(`6` = "blue",
    `42` = "red"), labels = c(`6` = "California",
    `42` = "Pennsylvania")) + theme_bw()
```



3. Nobody Home

a.

```
ca_pa.omit$Vacancy_rate <- ca_pa.omit$Vacant_units/ca_pa.omit$Total_units

min_vacancy <- min(ca_pa.omit$Vacancy_rate)
max_vacancy <- max(ca_pa.omit$Vacancy_rate)
mean_vacancy <- mean(ca_pa.omit$Vacancy_rate)
median_vacancy <- median(ca_pa.omit$Vacancy_rate)

# Print the summary statistics
cat("Minimum Vacancy Rate:", min_vacancy,
    "\n")
```

```
## Minimum Vacancy Rate: 0
```

```
cat("Maximum Vacancy Rate:", max_vacancy,
    "\n")
```

```
## Maximum Vacancy Rate: 0.965311
```

```
cat("Mean Vacancy Rate:", mean_vacancy, "\n")
```

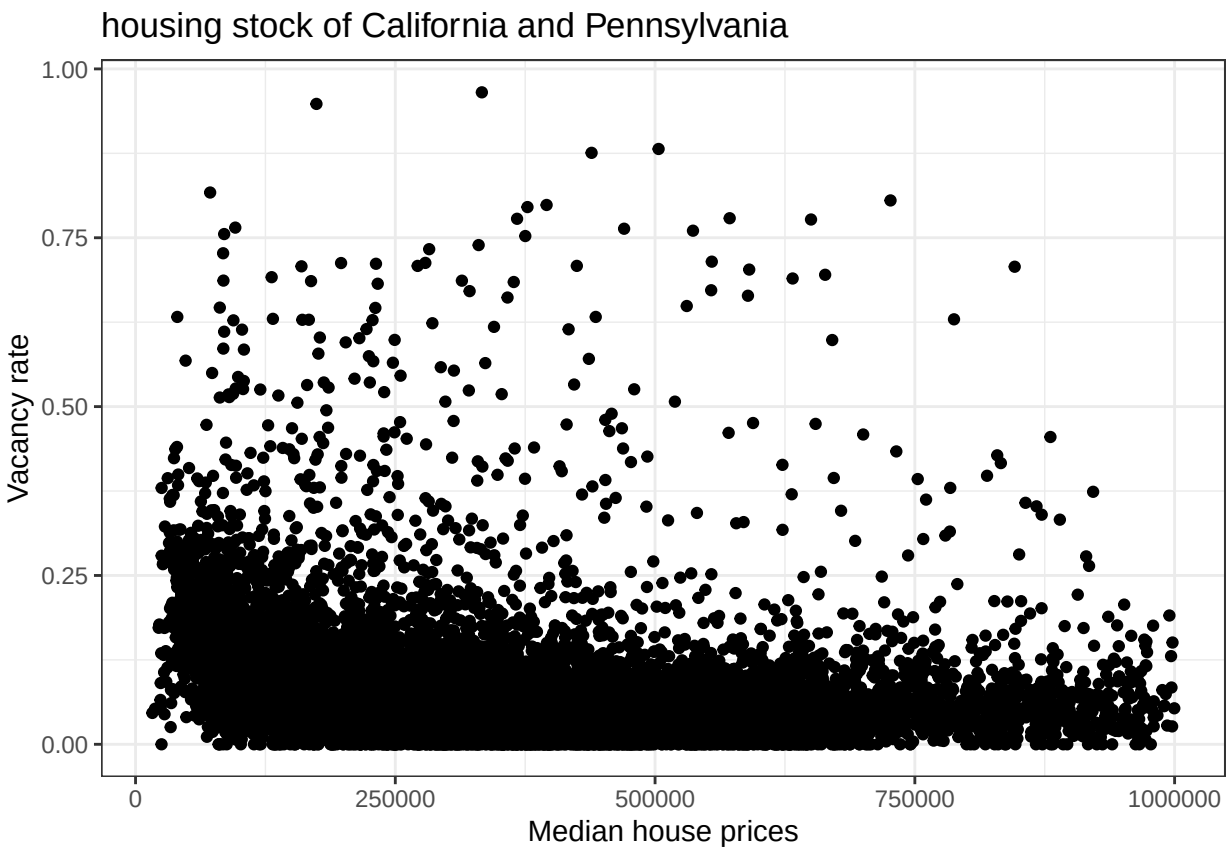
```
## Mean Vacancy Rate: 0.08888789
```

```
cat("Median Vacancy Rate:", median_vacancy,
    "\n")
```

```
## Median Vacancy Rate: 0.06767283
```

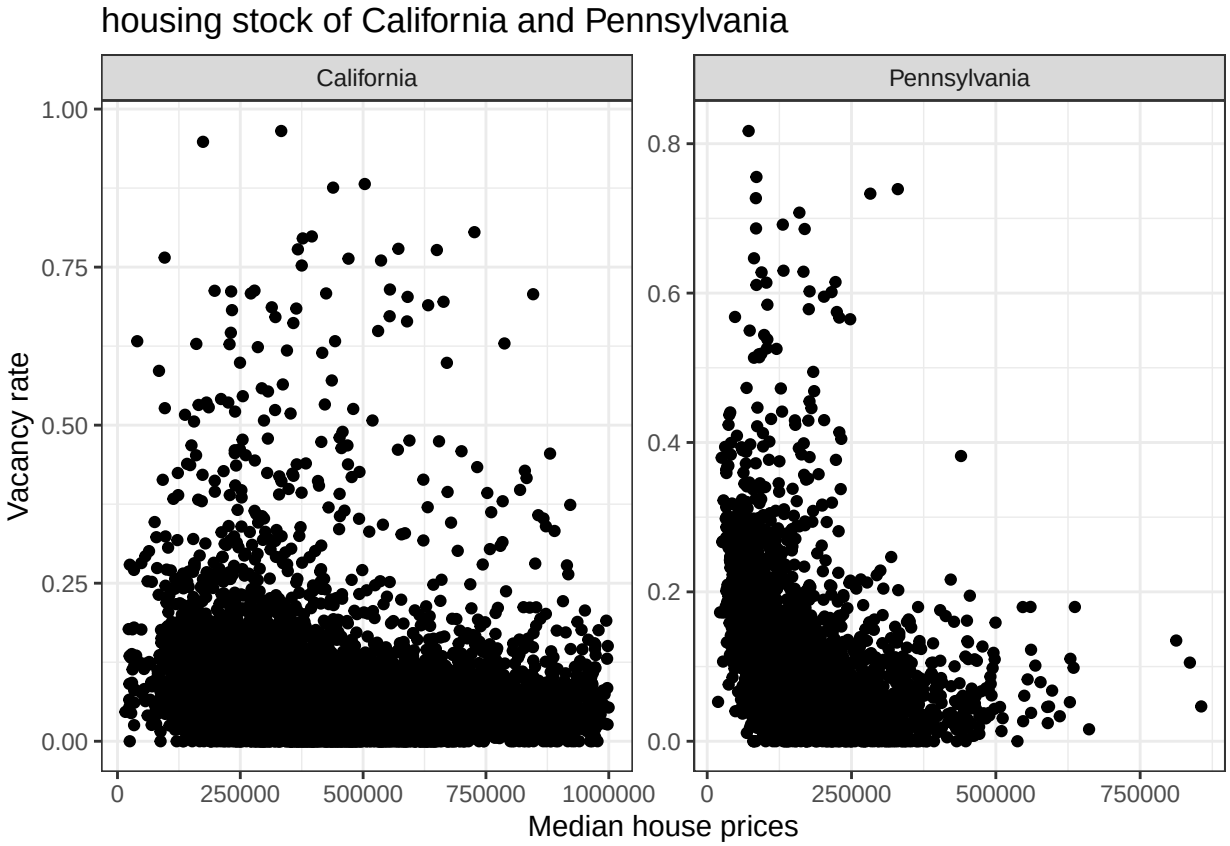
b.

```
ggplot(data = ca_pa.omit) + geom_point(aes(x = Median_house_value,
    y = Vacancy_rate)) + labs(x = "Median house prices",
    y = "Vacancy rate", title = "housing stock of California and Pennsylvania") +
    theme_bw()
```



c.

```
ggplot(data = ca_pa.omit) + geom_point(aes(x = Median_house_value,
    y = Vacancy_rate)) + labs(x = "Median house prices",
    y = "Vacancy rate", title = "housing stock of California and Pennsylvania") +
    facet_wrap(~STATEFP, scales = "free",
        labeller = labeller(STATEFP = c(`6` = "California",
            `42` = "Pennsylvania"))) + theme_bw()
```



Difference: In California, the higher the prices, the lower the Vacancy rate. In Pennsylvania, the Median house prices are lower than those of California.

4.

a.

```
acca <- c()
for (tract in 1:nrow(ca_pa.omit)) {
    if (ca_pa.omit$STATEFP[tract] == 6) {
```

```
        if (ca_pa.omit$COUNTYFP[tract] ==
            1) {
            acca <- c(acca, tract)
        }
    }
}
accamhv <- c()
for (tract in acca) {
    accamhv <- c(accamhv, ca_pa.omit[tract,
        10])
}
median(accamhv)
```

```
## [1] 474050
```

The block of code provided is intended to calculate the median house value for certain tracts within Alameda County (California) based on a dataset `ca_pa`. Let's break down how it achieves this:

1. **Initialization**:

   ```
   acca <- c()
   ```

   - `acca` is initialized as an empty vector. It will store the indices (or row numbers) of the tracts that belong to Alameda County.

2. **First `for` loop**:

   ```
   for (tract in 1:nrow(ca_pa)) {
     if (ca_pa$STATEFP[tract] == 6) {
       if (ca_pa$COUNTYFP[tract] == 1) {
         acca <- c(acca, tract)
       }
     }
   }
   ```

   - This loop iterates through each row (`tract`) of the dataset `ca_pa`.
   - It checks if the `STATEFP` value (state FIPS code) at the current row (`tract`) equals 6, which corresponds to California.
   - If the state is California (`STATEFP == 6`), it further checks if the `COUNTYFP` value (county FIPS code) at that row equals 1. County FIPS code 1 typically corresponds to Alameda County.

- If both conditions are satisfied (tract is in California and specifically in Alameda County), it adds the index (`tract`) to the `acca` vector.

Purpose: This loop identifies all the rows (tracts) in `ca_pa` that belong to Alameda County (California) and stores their indices in the `acca` vector.

3. **Second `for` loop**:

```
accamhv <- c()
for (tract in acca) {
  accamhv <- c(accamhv, ca_pa[tract, 10])
}
```

- After identifying the tracts belonging to Alameda County, this loop iterates through each index (`tract`) in the `acca` vector.
- For each index (`tract`), it retrieves the value in the 10th column (`ca_pa[tract, 10]`), which presumably represents the median house value for that tract.
- It appends each median house value to the `accamhv` vector.

Purpose: This loop extracts the median house values of all tracts within Alameda County from the dataset `ca_pa` and stores them in the `accamhv` vector.

4. **Calculate the median**:

```
median(accamhv)
```

- Finally, `median(accamhv)` calculates the median of all the house values stored in the `accamhv` vector.

Purpose: This computes and returns the median house value for all tracts within Alameda County.

Summary: The entire block of code focuses on extracting and computing the median house value for tracts specifically within Alameda County (California) from the dataset `ca_pa`. It uses two loops: the first to identify the relevant tracts and the second to gather their median house values, culminating in the calculation of the median value itself.

b.

```
median(ca_pa.omit$Median_house_value[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 1])
```

```
## [1] 474050
```

c.

```
mean.acc <- mean(ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 1])
mean.sc <- mean(ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 85])
mean.acp <- mean(ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    42 & ca_pa.omit$COUNTYFP == 3])

cat("average percentages of housing built since 2005:\n")
```

```
## average percentages of housing built since 2005:
```

```
cat("Alameda County:", mean.acc, "\n")
```

```
## Alameda County: 2.820468
```

```
cat("Santa Clara:", mean.sc, "\n")
```

```
## Santa Clara: 3.200319
```

```
cat("Allegheny County:", mean.acp, "\n")
```

```
## Allegheny County: 1.474219
```

d. To compute correlations using the column names from your dataset `ca_pa`, you should modify the code accordingly. Here's how you can do it using the `names()` function to reference columns by name:

1. **Whole dataset**:

```
cor(ca_pa.omit$Median_house_value, ca_pa.omit$Built_2005_or_later)
```

```
## [1] -0.01893186
```

2. **California** (assuming STATEFP identifies California as 6):

```
cor(ca_pa.omit$Median_house_value[ca_pa.omit$STATEFP ==
    6], ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    6])
```

```
## [1] -0.1153604
```

3. **Pennsylvania** (assuming STATEFP identifies Pennsylvania as 42):

```
cor(ca_pa.omit$Median_house_value[ca_pa.omit$STATEFP ==
    42], ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    42])
```

```
## [1] 0.2681654
```

4. **Alameda County, California** (assuming COUNTYFP identifies Alameda County as 1 within STATEFP = 6):

```
cor(ca_pa.omit$Median_house_value[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 1], ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 1])
```

```
## [1] 0.01303543
```

5. **Santa Clara County, California** (assuming COUNTYFP identifies Santa Clara County as 85 within STATEFP = 6):

```
cor(ca_pa.omit$Median_house_value[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 85], ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 85])
```

```
## [1] -0.1726203
```

6. **Allegheny County, Pennsylvania** (assuming COUNTYFP identifies Allegheny County as 3 within STATEFP = 42):

```
cor(ca_pa.omit$Median_house_value[ca_pa.omit$STATEFP ==
    42 & ca_pa.omit$COUNTYFP == 3], ca_pa.omit$Built_2005_or_later[ca_pa.omit$STATEFP ==
    42 & ca_pa.omit$COUNTYFP == 3])
```

```
## [1] 0.1939652
```

Adjust the column names (`Median_house_value` and `Built_2005_or_later` in this example) according to the actual names in your dataset obtained from `names(ca_pa)`. This approach ensures that you're correctly referencing columns based on their names as listed in your dataset.

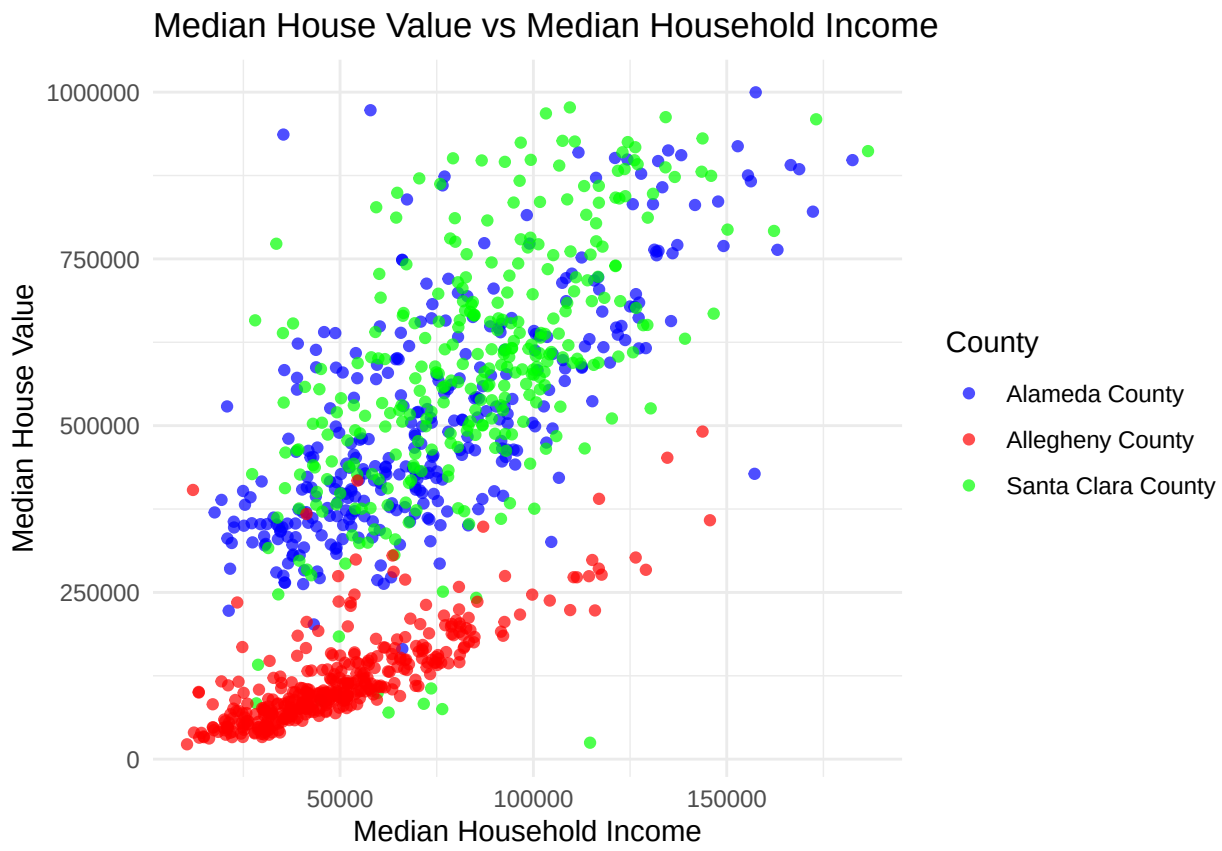
e.

```
# Filter data for Alameda County,
# California (STATEFP = 6, COUNTYFP =
# 1)
alameda_data <- ca_pa.omit[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 1, ]

# Filter data for Santa Clara County,
# California (STATEFP = 6, COUNTYFP =
# 85)
santa_clara_data <- ca_pa.omit[ca_pa.omit$STATEFP ==
    6 & ca_pa.omit$COUNTYFP == 85, ]

# Filter data for Allegheny County,
# Pennsylvania (STATEFP = 42, COUNTYFP
# = 3)
allegheny_data <- ca_pa.omit[ca_pa.omit$STATEFP ==
    42 & ca_pa.omit$COUNTYFP == 3, ]

# Create a combined plot
ggplot() + geom_point(data = alameda_data,
    aes(x = Median_household_income, y = Median_house_value,
        color = "Alameda County"), alpha = 0.7) +
    geom_point(data = santa_clara_data, aes(x = Median_household_income,
        y = Median_house_value, color = "Santa Clara County"),
        alpha = 0.7) + geom_point(data = allegheny_data,
    aes(x = Median_household_income, y = Median_house_value,
        color = "Allegheny County"), alpha = 0.7) +
    labs(x = "Median Household Income", y = "Median House Value",
        title = "Median House Value vs Median Household Income") +
    scale_color_manual(name = "County", values = c(`Alameda County` = "blue",
        `Santa Clara County` = "green", `Allegheny County` = "red")) +
    theme_minimal()
```



MB.Ch1.11.

```
gender <- factor(c(rep("female", 91), rep("male",
    92)))
table(gender)
```

```
## gender
## female   male
##     91     92
```

In this part, you created a factor variable gender with 91 "female" entries and 92 "male" entries.

```
gender <- factor(gender, levels = c("male",
    "female"))
table(gender)
```

```
## gender
```

```
##   male female
##     92     91
```

Here, you correctly reordered the levels of gender to prioritize “male” first and “female” second. Now the table shows 92 males and 91 females.

```
gender <- factor(gender, levels = c("Male",
    "female"))
# Note the mistake: 'Male' should be
# 'male'
table(gender)
```

```
## gender
##   Male female
##      0     91
```

The mistake here is that you tried to relabel “male” as “Male” (with an uppercase ‘M’), but since factors are case-sensitive in R, it created a new level “Male” which didn’t exist previously. Hence, all entries previously labeled as “male” (lowercase ‘m’) are now labeled as because they don’t match any of the specified levels.

```
table(gender, exclude = NULL)
```

```
## gender
##   Male female   <NA>
##      0     91     92
```

To see all levels including , you can use table(gender, exclude=NULL). This shows that there are 0 “Male”, 91 “female”, and 92 (missing) entries.

```
rm(gender)  # Remove gender
```

Finally, rm(gender) removes the gender variable from your workspace.

MB.Ch1.12.

Let’s proceed with writing a function that calculates the proportion of values in a vector `x` that exceed a specified `cutoff`.

Part (a): Function Definition

Here's the function that calculates the proportion of values in a vector `x` that exceed a specified `cutoff`:

```
proportion_exceed <- function(x, cutoff) {
    if (length(x) == 0) {
        stop("Vector x is empty.")
    }

    # Calculate proportion exceeding
    # cutoff
    prop_exceed <- mean(x > cutoff)

    return(prop_exceed)
}
```

Explanation:

- **Function Name**: `proportion_exceed`
- **Parameters**:
  - `x`: Vector of numeric values.
  - `cutoff`: Numeric value against which elements of `x` are compared.
- **Returns**: Proportion of values in `x` that exceed `cutoff`.

Testing with Sequence 1 to 100

Let's use the sequence of numbers 1 to 100 to verify our function:

```
# Create sequence from 1 to 100
sequence <- 1:100

# Test the function with cutoff of 50
proportion_exceed(sequence, 50)
```

```
## [1] 0.5
```

```
# Output: 0.5 (since half of the values
# in sequence exceed 50)
```

Part (b): Applying to Escape Times Data

Now, we'll apply this function to the `ex01.36` dataset from the `Devore6` package, which contains escape times from an oil platform drill.

```
# Load the Devore7 package (if not
# already installed, install it first)
# install.packages('Devore7')
library(Devore7)
```

```
## 载入需要的程序包: lattice
```

```
# Load the dataset
data(ex01.36)

# View the dataset
head(ex01.36)
```

```
##    C1
## 1 389
## 2 356
## 3 359
## 4 363
## 5 375
## 6 424
```

```
# Calculate the proportion of escape
# times that exceed 7 minutes (420
# seconds)
prop_exceed_7min <- proportion_exceed(ex01.36,
    420)

# Show the proportion
prop_exceed_7min
```

```
## [1] 0.03846154
```

Explanation:

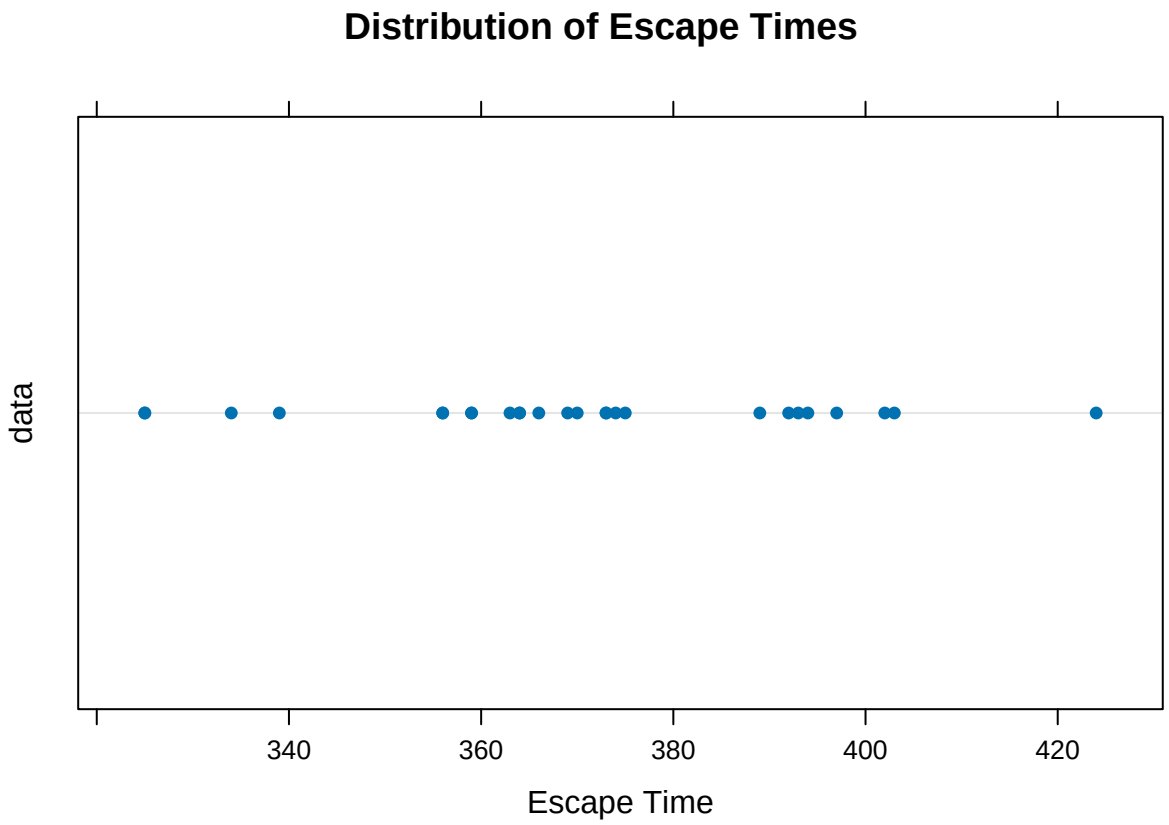
- **Loading Package**: `Devore6` package is loaded to access the `ex01.36` dataset.

- **Dataset**: We load the dataset and view its structure.
- **Calculation**: We calculate the proportion of escape times (`ex01.36`) that exceed 7 minutes (420 seconds).
- **Result**: `prop_exceed_7min` will give us the proportion of escape times exceeding 7 minutes.

Visualization with `dotplot()`

To visualize the distribution of escape times, we can use `dotplot()`:

```
# Visualize the distribution of escape
# times
dotplot(ex01.36$C1, main = "Distribution of Escape Times",
    xlab = "Escape Time", ylab = "data")
```



MB.Ch1.18.

```r
# Load the Rabbit data
data(Rabbit)

Dose <- unstack(Rabbit, Dose ~ Animal)[,
    1]
Treatment <- unstack(Rabbit, Treatment ~
    Animal)[, 1]
BPchange <- unstack(Rabbit, BPchange ~ Animal)
data.frame(Treatment, Dose, BPchange)
```

```
##    Treatment   Dose    R1    R2    R3    R4   R5
## 1    Control   6.25  0.50  1.00  0.75  1.25  1.5
## 2    Control  12.50  4.50  1.25  3.00  1.50  1.5
## 3    Control  25.00 10.00  4.00  3.00  6.00  5.0
## 4    Control  50.00 26.00 12.00 14.00 19.00 16.0
## 5    Control 100.00 37.00 27.00 22.00 33.00 20.0
## 6    Control 200.00 32.00 29.00 24.00 33.00 18.0
## 7        MDL   6.25  1.25  1.40  0.75  2.60  2.4
## 8        MDL  12.50  0.75  1.70  2.30  1.20  2.5
## 9        MDL  25.00  4.00  1.00  3.00  2.00  1.5
## 10       MDL  50.00  9.00  2.00  5.00  3.00  2.0
## 11       MDL 100.00 25.00 15.00 26.00 11.00  9.0
## 12       MDL 200.00 37.00 28.00 25.00 22.00 19.0
```