Week 3, Class 5: Practice Exercises - ANSWER KEY

Data Transformation and Variable Creation

2024-12-31

1 Non-AI Exercises

1.1 1. Understanding mutate()

1.1.1 1.1 Multiple Choice: mutate() Function

What does the `mutate()` function do?

a) Removes columns from a data frame
b) Creates new columns or modifies existing ones
c) Filters rows based on conditions
d) Sorts data by a variable

Answer: **b) Creates new columns or modifies existing ones**

1.1.2 1.2 Code Detective: Basic mutate()

What does this code create?

```
data %>%
  mutate(
    vote_margin = dem_votes - rep_votes,
    winner = if_else(vote_margin > 0, "Democrat", "Republican")
  )
```

Line 3 creates: **A numeric variable showing the difference between Democratic and Republican votes (vote margin)** Line 4 creates: **A character variable indicating which party won based on the vote margin**

1.1.3 1.3 Fill in the Blanks: Variable Types

When creating new variables, we often need to:

1. Calculate **differences** between existing variables
2. **Recode** text variables into categories
3. Create **logical** (TRUE/FALSE) indicators
4. Convert between **data** types
5. Handle **missing** values appropriately

1.2 2. Conditional Logic

1.2.1 2.1 Match: if_else() vs case_when()

Match each function with when to use it:

Functions: a) if_else() b) case_when()

Use cases: 1. Creating a variable with only two possible outcomes 2. Creating a variable with multiple categories 3. Simple yes/no binary coding 4. Complex multi-condition logic

Matches: a = **1 and 3**, b = **2 and 4**

1.2.2 2.2 Code Detective: case_when()

What categories does this code create?

```r
data %>%
  mutate(
    age_group = case_when(
      age < 30 ~ "Young",
      age < 50 ~ "Middle",
      age < 65 ~ "Older",
      TRUE ~ "Senior"
    )
  )
```

For someone age 25: **Young** For someone age 45: **Middle** For someone age 70: **Senior**

1.2.3 2.3 Spot the Error

What's wrong with this case_when() statement?

```r
mutate(
  income_cat = case_when(
    income < 30000 ~ "Low",
    income < 50000 ~ "Medium",
    income < 30000 ~ "Low",
    TRUE ~ "High"
  )
)
```

Problem: **Line 5 duplicates the condition from line 3 (income < 30000). This is redundant and the second instance will never be reached because case_when() evaluates conditions in order.**

1.3 3. Working with Proportions

1.3.1 3.1 Multiple Choice: Calculating Proportions

To calculate the proportion of Democrats in a dataset, you would:

a) Count Democrats and divide by Republicans
b) Count Democrats and divide by total observations
c) Count total and divide by Democrats
d) Use mean() on a logical variable

Answer: **b) Count Democrats and divide by total observations** (Note: d is also correct if you have a logical variable indicating Democrats)

1.3.2 3.2 True or False: summarise()

Mark each statement as True (T) or False (F):

T summarise() reduces multiple rows to a single summary row **T** You can calculate multiple statistics in one summarise() call **F** summarise() automatically groups by all variables **T** n() counts the number of rows in each group **F** summarise() can only calculate numeric summaries

1.3.3 3.3 Fill in the Code

Complete this code to calculate turnout rate:

```
data %>%
  summarise(
    total_voters = n(),
    total_voted = sum(voted == "Yes"),
    turnout_rate = total_voted / total_voters
  )
```

1.4 4. Advanced Transformations

1.4.1 4.1 Match: Functions and Purposes

Match each function with its purpose:

Functions: a) round() b) log() c) sqrt() d) abs() e) lag()

Purposes: 1. Remove decimal places 2. Transform skewed data 3. Calculate square root 4. Make negative values positive 5. Get previous row's value

Matches: a = **1**, b = **2**, c = **3**, d = **4**, e = **5**

1.4.2 4.2 Code Detective: Complex Transformation

What does this code calculate?

```
data %>%
  group_by(state) %>%
  mutate(
    state_avg = mean(income, na.rm = TRUE),
    income_diff = income - state_avg,
    above_avg = income_diff > 0
  )
```

This code calculates: **For each state, it calculates the average income, then for each person it calculates how their income differs from their state's average, and creates a logical variable indicating if they earn above their state's average**

2 AI Exercises

For each AI exercise: - Work with Claude to analyze the data - Record your prompts and key findings

2.1 5. Creating Political Variables

Dataset: legislative_votes.csv

2.1.1 5.1 Initial Data Exploration

```r
# Load the dataset
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.2
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
legislators <- read_csv("legislative_votes.csv")
```

```
Rows: 535 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): name, party, state
dbl (7): legislator_id, district, ideology_score, vote_attendance, bills_spo...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Examine the data
glimpse(legislators)
```

```
Rows: 535
Columns: 10
$ legislator_id   <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,~
$ name            <chr> "Legislator_1", "Legislator_2", "Legislator_3", "Legis~
$ party           <chr> "Democrat", "Democrat", "Republican", "Democrat", "Dem~
$ state           <chr> "MN", "OK", "MD", "FL", "NE", "VA", "VA", "AZ", "NM", ~
$ district        <dbl> 30, 9, 37, 17, 44, 20, 36, 38, 34, 3, 50, 17, 24, 50, ~
$ ideology_score  <dbl> -0.22499579, -0.07570425, 0.60032405, -0.32344155, 0.3~
$ vote_attendance <dbl> 43.05888, 87.71777, 70.22108, 91.00167, 85.69489, 59.1~
$ bills_sponsored <dbl> 4, 5, 7, 1, 8, 5, 5, 6, 4, 7, 6, 7, 6, 6, 8, 3, 10, 7,~
$ years_service   <dbl> 20, 12, 4, 30, 28, 24, 8, 5, 13, 11, 21, 28, 25, 6, 11~
$ committee_count <dbl> 3, 2, 3, 3, 1, 4, 7, 5, 3, 7, 2, 4, 1, 2, 4, 2, 4, 4, ~
```

2.1.2 5.2 Creating Categorical Variables

```r
# Create ideology categories based on terciles
legislators <- legislators %>%
  mutate(
    # Calculate tercile cutpoints for ideology
    ideology_category = case_when(
      ideology_score < quantile(ideology_score, 1/3, na.rm = TRUE) ~ "left",
      ideology_score < quantile(ideology_score, 2/3, na.rm = TRUE) ~ "center",
      TRUE ~ "right"
    ),

    # Create effectiveness score
    # Assume effective if sponsored more than 4 bills AND serves on committees
    effectiveness = if_else(bills_sponsored > 4 & committee_count > 0,
                           "effective", "not effective"),

    # Create seniority category
    seniority = if_else(years_service > 20, "senior", "junior")
  )

# Check the results
legislators %>%
  count(ideology_category)
```

```
# A tibble: 3 x 2
  ideology_category     n
  <chr>              <int>
1 center               178
2 left                 178
3 right                179
```

```
legislators %>%
  count(effectiveness)
```

```
# A tibble: 2 x 2
  effectiveness     n
  <chr>         <int>
1 effective       317
2 not effective   218
```

```
legislators %>%
  count(seniority)
```

```
# A tibble: 2 x 2
  seniority     n
  <chr>     <int>
1 junior      351
2 senior      184
```

2.1.3 5.3 Calculating Party Metrics

```
# Calculate party-level statistics for the created variables
party_stats <- legislators %>%
  group_by(party) %>%
  summarise(
    # Ideology distribution
    pct_left = mean(ideology_category == "left", na.rm = TRUE) * 100,
    pct_center = mean(ideology_category == "center", na.rm = TRUE) * 100,
    pct_right = mean(ideology_category == "right", na.rm = TRUE) * 100,

    # Effectiveness
    pct_effective = mean(effectiveness == "effective", na.rm = TRUE) * 100,

    # Seniority
    pct_senior = mean(seniority == "senior", na.rm = TRUE) * 100,
    n = n()
  ) %>%
  mutate(across(where(is.numeric) & !n, ~round(., 1)))

party_stats
```

```
# A tibble: 3 x 7
  party       pct_left pct_center pct_right pct_effective pct_senior     n
  <chr>          <dbl>      <dbl>     <dbl>         <dbl>      <dbl> <int>
1 Democrat        68.9       28.9       2.2          54.8       35.5   228
2 Independent     25         50        25            58.3       41.7    24
3 Republican       5.3       35.3      59.4          62.9       32.9   283
```

2.2 6. Education and Political Participation

Dataset: civic_engagement.csv

2.2.1 6.1 Loading and Initial Transformation

```
# Load the dataset
civic <- read_csv("civic_engagement.csv")
```

```
Rows: 1000 Columns: 9
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): education, voted_2020
dbl (7): respondent_id, age, income, political_interest, volunteer_hours, do...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Check the structure
glimpse(civic)
```

```
Rows: 1,000
Columns: 9
$ respondent_id          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ~
$ education              <chr> "Master's", "High School", "Some College", "Hig~
$ age                    <dbl> 42, 36, 62, 81, 77, 77, 21, 56, 71, 24, 24, 66,~
$ income                 <dbl> 54146.01, 25746.07, 29484.80, 11927.46, 33049.0~
$ voted_2020             <chr> "Yes", "Yes", "No", "Yes", "Yes", "No", "Yes", ~
$ political_interest     <dbl> 8, 9, 7, 9, 6, 7, 8, 2, 5, 4, 1, 7, 2, 3, 4, 6,~
$ volunteer_hours        <dbl> 3, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 5, 2, 2, 2, 1,~
$ donations              <dbl> 2417.21218, 169.08434, 15.33118, 0.00000, 62.12~
$ social_media_political <dbl> 7.8416122, 1.8931274, 7.4243669, 4.5703181, 2.0~
```

2.2.2 6.2 Education and Engagement Index

```r
# Calculate medians for thresholds
median_donations <- median(civic$donations, na.rm = TRUE)
median_volunteer <- median(civic$volunteer_hours, na.rm = TRUE)

# Create new variables
civic <- civic %>%
  mutate(
    # Recode education into binary
    education_binary = if_else(
      education %in% c("High School", "Some College", "Bachelor", "Graduate"),
      "High School +",
      "Less than High School"
    ),

    # Create activist variable
    # TRUE if donated > median AND volunteered > median AND voted in 2020
    activist = (donations > median_donations) &
               (volunteer_hours > median_volunteer) &
               (voted_2020 == "Yes")
  )

# Check the results
civic %>%
  count(education_binary)
```

```
# A tibble: 2 x 2
  education_binary          n
  <chr>                 <int>
1 High School +           555
2 Less than High School   445
```

```
civic %>%
  count(activist)
```

```
# A tibble: 2 x 2
  activist     n
  <lgl>    <int>
1 FALSE      949
2 TRUE        51
```

```
# Analyze relationship between education and activism
civic %>%
  group_by(education_binary) %>%
  summarise(
    pct_activist = mean(activist, na.rm = TRUE) * 100,
    avg_donations = mean(donations, na.rm = TRUE),
    avg_volunteer = mean(volunteer_hours, na.rm = TRUE),
    n = n()
  )
```

```
# A tibble: 2 x 5
  education_binary      pct_activist avg_donations avg_volunteer     n
  <chr>                        <dbl>         <dbl>         <dbl> <int>
1 High School +                 3.60          81.9          1.92   555
2 Less than High School         6.97          83.5          2.02   445
```

2.3 7. Campaign Finance Transformations

Dataset: campaign_finance_2024.csv

2.3.1 7.1 Data Preparation

```
# Load the dataset
finance <- read_csv("campaign_finance_2024.csv")
```

```
Rows: 2000 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (7): candidate, party, office, contributor_type, state, employer, occup...
```

```
dbl  (2): contribution_id, amount
date (1): date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Explore the data
glimpse(finance)
```

```
Rows: 2,000
Columns: 10
$ contribution_id  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
$ candidate        <chr> "Wilson (I)", "Jones (R)", "Brown (D)", "Wilson (I)",~
$ party            <chr> "Independent", "Republican", "Democrat", "Independent~
$ office           <chr> "House", "Governor", "House", "Senate", "Senate", "Ho~
$ amount           <dbl> 1000.000, 50.000, 1693.038, 4685.861, 500.000, 500.00~
$ date             <date> 2024-11-04, 2024-09-17, 2024-08-10, 2024-10-08, 2024~
$ contributor_type <chr> "PAC", "Union PAC", "Individual", "Individual", "Indi~
$ state            <chr> "NM", "NY", "NM", "SC", "MN", "ND", "DE", "CO", "AR",~
$ employer         <chr> "Education", "Not Employed", "Education", "Tech Corp"~
$ occupation       <chr> "Business Owner", "Business Owner", "Engineer", "Cons~
```

2.3.2 7.2 Creating Contribution Categories

```r
# Calculate mean for threshold
mean_amount <- mean(finance$amount, na.rm = TRUE)

# Create donor categories
finance <- finance %>%
  mutate(
    # Donor size categories based on mean
    donor_size = if_else(
      amount < mean_amount,
      "small_donor",
      "major_donor"
    ),

    # In-state vs out-of-state (assuming we're in New York)
    contribution_source = if_else(
      state == "NY",
      "in_state",
      "out_of_state"
    )
  )

# Summarize results
finance %>%
  count(donor_size) %>%
  mutate(pct = n / sum(n) * 100)
```

```
# A tibble: 2 x 3
  donor_size      n   pct
  <chr>       <int> <dbl>
1 major_donor   426  21.3
2 small_donor  1574  78.7
```

```
finance %>%
  count(contribution_source) %>%
  mutate(pct = n / sum(n) * 100)
```

```
# A tibble: 2 x 3
  contribution_source     n   pct
  <chr>               <int> <dbl>
1 in_state               38   1.9
2 out_of_state         1962  98.1
```

```
# Analyze by party
finance %>%
  group_by(party, donor_size) %>%
  summarise(
    total_raised = sum(amount, na.rm = TRUE),
    avg_contribution = mean(amount, na.rm = TRUE),
    n_contributions = n(),
    .groups = "drop"
  )
```

```
# A tibble: 6 x 5
  party       donor_size  total_raised avg_contribution n_contributions
  <chr>       <chr>              <dbl>            <dbl>           <int>
1 Democrat    major_donor     1568273.            8477.             185
```

```
2 Democrat    small_donor      228476.          331.           690
3 Independent major_donor      509194.         8933.            57
4 Independent small_donor       70699.          314.           225
5 Republican  major_donor     1546399.         8404.           184
6 Republican  small_donor      203424.          309.           659
```

2.3.3 7.3 Time-Based Transformations

```
library(lubridate)

# Convert date and create time variables
finance <- finance %>%
  mutate(
    date = as.Date(date),
    month = month(date, label = TRUE),
    month_num = month(date),

    # Create campaign period categories
    campaign_period = case_when(
      month_num <= 3 ~ "early",
      month_num <= 7 ~ "middle",
      TRUE ~ "late"
    )
  )

# When was the most money given?
finance %>%
  group_by(campaign_period) %>%
  summarise(
    total = sum(amount, na.rm = TRUE),
    avg = mean(amount, na.rm = TRUE),
    n = n()
  ) %>%
  arrange(desc(total))
```

```
# A tibble: 3 x 4
  campaign_period   total  avg     n
  <chr>             <dbl> <dbl> <int>
1 late           1720636. 2041.   843
2 middle         1502924. 2204.   682
3 early           902904. 1901.   475
```

```r
# Monthly totals
monthly_totals <- finance %>%
  group_by(month) %>%
  summarise(
    total_amount = sum(amount, na.rm = TRUE),
    n_contributions = n(),
    avg_contribution = mean(amount, na.rm = TRUE)
  )

monthly_totals
```

```
# A tibble: 12 x 4
   month total_amount n_contributions avg_contribution
   <ord>        <dbl>           <int>            <dbl>
 1 Jan        299699.             165            1816.
 2 Feb        321696.             150            2145.
 3 Mar        281509.             160            1759.
 4 Apr        391615.             165            2373.
 5 May        408361.             176            2320.
 6 Jun        353240.             169            2090.
 7 Jul        349708.             172            2033.
 8 Aug        279707.             160            1748.
 9 Sep        305594.             182            1679.
10 Oct        389474.             167            2332.
11 Nov        346315.             154            2249.
12 Dec        399546.             180            2220.
```

```r
# Cumulative contributions over time
finance %>%
  arrange(date) %>%
  mutate(cumulative_total = cumsum(amount)) %>%
  group_by(month) %>%
  summarise(
    cumulative = max(cumulative_total),
    month_total = sum(amount)
  )
```

```
# A tibble: 12 x 3
   month cumulative month_total
   <ord>      <dbl>       <dbl>
 1 Jan      299699.     299699.
 2 Feb      621395.     321696.
 3 Mar      902904.     281509.
 4 Apr     1294519.     391615.
 5 May     1702880.     408361.
 6 Jun     2056120.     353240.
 7 Jul     2405828.     349708.
 8 Aug     2685535.     279707.
 9 Sep     2991129.     305594.
10 Oct     3380603.     389474.
11 Nov     3726918.     346315.
12 Dec     4126464.     399546.
```

2.4 8. Demographic Transformations

Dataset: census_political.csv

2.4.1 8.1 Initial Exploration

```
# Load the dataset
census <- read_csv("census_political.csv")
```

```
Rows: 500 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): county_name, state
dbl (8): county_id, population, median_age, pct_college, median_income, unem...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Look at the data
glimpse(census)
```

```
Rows: 500
Columns: 10
$ county_id          <dbl> 10001, 10002, 10003, 10004, 10005, 10006, 10007, 1~
$ county_name        <chr> "County_1", "County_2", "County_3", "County_4", "C~
$ state              <chr> "OK", "CA", "VT", "AZ", "IN", "NY", "NJ", "CT", "D~
$ population         <dbl> 36033, 531847, 35045, 45525, 12389, 46815, 129419,~
$ median_age          <dbl> 35.42766, 53.41098, 43.02727, 37.21216, 43.95637, ~
$ pct_college         <dbl> 38.560397, 22.644499, 34.894835, 28.512275, 20.995~
$ median_income       <dbl> 51562.02, 51764.40, 47877.09, 70675.36, 60225.79, ~
$ unemployment_rate   <dbl> 2.4026045, 5.0597188, 7.2472078, 2.6662320, 12.508~
$ dem_vote_share_2020 <dbl> 31.24898, 38.56374, 49.43067, 34.59972, 45.45324, ~
$ turnout_2020        <dbl> 66.74962, 87.45555, 71.17116, 55.59097, 63.28583, ~
```

2.4.2 8.2 Creating Composite Indicators

```
# Calculate median population for urban/rural classification
median_pop <- median(census$population, na.rm = TRUE)

# Create classifications
census <- census %>%
  mutate(
    # Urban/rural classification based on population (proxy for density)
    urban_rural = if_else(
      population > median_pop,
      "urban",
      "rural"
    ),

    # Unemployment quartiles
    unemployment_quartile = case_when(
      unemployment_rate <= quantile(unemployment_rate, 0.25, na.rm = TRUE) ~ "Q1 (lowest)",
      unemployment_rate <= quantile(unemployment_rate, 0.50, na.rm = TRUE) ~ "Q2",
      unemployment_rate <= quantile(unemployment_rate, 0.75, na.rm = TRUE) ~ "Q3",
      TRUE ~ "Q4 (highest)"
    )
  )

# Check distributions
census %>%
  count(urban_rural) %>%
  mutate(pct = n / sum(n) * 100)
```

```
# A tibble: 2 x 3
  urban_rural     n   pct
  <chr>       <int> <dbl>
1 rural         250    50
2 urban         250    50
```

```r
census %>%
  count(unemployment_quartile)
```

```
# A tibble: 4 x 2
  unemployment_quartile     n
  <chr>                 <int>
1 Q1 (lowest)             125
2 Q2                      125
3 Q3                      125
4 Q4 (highest)            125
```

```r
# Analyze political patterns by urban/rural
census %>%
  group_by(urban_rural) %>%
  summarise(
    avg_dem_share = mean(dem_vote_share_2020, na.rm = TRUE),
    avg_turnout = mean(turnout_2020, na.rm = TRUE),
    avg_college = mean(pct_college, na.rm = TRUE),
    avg_income = mean(median_income, na.rm = TRUE),
    n = n()
  )
```

```
# A tibble: 2 x 6
  urban_rural avg_dem_share avg_turnout avg_college avg_income     n
  <chr>               <dbl>       <dbl>       <dbl>      <dbl> <int>
1 rural                50.7        68.4        27.7     56459.   250
2 urban                52.0        71.4        29.3     53449.   250
```

```r
# Analyze by unemployment quartiles
census %>%
  group_by(unemployment_quartile) %>%
  summarise(
    avg_dem_share = mean(dem_vote_share_2020, na.rm = TRUE),
    avg_turnout = mean(turnout_2020, na.rm = TRUE),
    avg_income = mean(median_income, na.rm = TRUE),
    avg_college = mean(pct_college, na.rm = TRUE),
    n = n()
  ) %>%
  arrange(unemployment_quartile)
```

```
# A tibble: 4 x 6
  unemployment_quartile avg_dem_share avg_turnout avg_income avg_college     n
  <chr>                         <dbl>       <dbl>      <dbl>       <dbl> <int>
1 Q1 (lowest)                    50.1        68.9     54113.        29.0   125
2 Q2                             53.5        70.5     53541.        29.3   125
3 Q3                             52.4        69.9     58192.        29.8   125
4 Q4 (highest)                   49.5        70.2     53969.        25.9   125
```