Graphs and Data Visualization

- **Bar Chart geom_bar()**
  - Compares quantities across categories.
  - Uses rectangular bars to display discrete data.(their length represents the data value)
- **Density Plot geom_density()**
  - Visualizes data distribution over a continuous interval.
  - Indicates value concentration; smoothness adjusted by 'bw'.
  - Empirical Rule (68-95-99.7 rule): for normal distribution (e.g., centered near 0, standard deviation 15): ~95% of data within ±2 standard deviations (between -30 and 30).
- **Histogram geom_histogram()**
  - Similar to bar chart but for continuous data, shows frequency.
  - Adjust 'binwidth' or 'bins' for granularity.
  - distribution of numerical data
- **Scatter Plot geom_point()**
  - Plots two variables, ideal for showing their relationship.
  - Add trend line with geom_smooth to highlight correlations.
  - ideal for showing the relationship between two continuous variables

ggplot2

- Builds plots layer by layer: data, aesthetics (aes), geometries (geom_).
- Aesthetics assign variables to axes, color, size, shape.
- Geometries specify plot type (e.g., geom_point, geom_line).
- Facets (facet_wrap(), facet_grid()) create small multiples.
- Scales (e.g., scale_x_continuous()) adjust axis properties.
- Themes (e.g., theme(), theme_minimal()) modify appearance.

R Functions Tidyverse:

Dplyr:

- **select():** selects a subset of columns from a dataset
  - example: new_df <- df %>% select(x, y) creates a new dataset containing only the columns x and y from df and puts it in new_df
- **filter():** subsets rows based on a logical condition
  - example: df %>% filter(x > 0) returns a subset of df containing only rows where x is greater than 0
- **group_by():** groups data by one or more variables
  - example: df %>% group_by(color) treats rows with the same color as belonging to the same group for subsequent summaries or transformations
- **summarize():** generates summary statistics for groups or entire datasets
  - example: df %>% group_by(color) %>% summarize(max_s = max(s)) computes the maximum of s for each color group
- **mutate():** adds new variables or transforms existing ones
  - example: df %>% mutate(s = x + y) adds a new column s to df, containing the sum of x and y for each row
- **arrange():** orders rows by variables
  - example: df %>% arrange(desc(s)) arranges rows of df in descending order of s
- **join() functions:** merges different datasets by key variables
  - Example: joined_df <- df1 %>% inner_join(df2, by = "key") merges 'df1' with 'df2' based on matching 'key' column
- **tally() and count():** used to count the number of observations, optionally grouped by one or more variables
  - Example: count_df <- df %>% group_by(color) %>% tally() counts the number of observations in each 'color' group
- **slice():** selects rows by their row number
  - Example: sliced_df <- df %>% slice(1:3) Selects the first three rows from 'df'
- **slice_max():** selects rows with highest values of a given variable
  - example: df %>% slice_max(y, n = 1) selects the row(s) from df where y is at its maximum value. If there are ties, more than one row can be returned
- **across():** used within mutate() or summarize() to apply functions across multiple columns
  - Example: across_df <- df %>% summarize(across(c(x, y), mean)) applies the mean function across the columns 'x' and 'y'
- **nrow():** returns the number of rows in a df/matrix
  - Example: number_of_rows <- nrow(df) returns the number of rows in 'df'
- **ungroup():** removes grouping structure from a grouped data frame
  - ungrouped_df <- df %>% ungroup() removes grouping structure from 'df'
- **drop_na():** removes rows with missing values(NA) from a data frame
  - Example: clean_df <- df %>% drop_na() removes rows with NA values from 'df'
- **semi_join():** returns all rows from the left table that have a match in the right table
  - Example: semi_joined_df <- df1 %>% semi_join(df2, by = "key") returns all rows from 'df1' that have a matching key in 'df2'
- **inner_join():** combines rows from the left/right table where there are matching rows in the join columns
  - Example: inner_joined_df <- df1 %>% inner_join(df2, by = "key") combines rows from 'df1' and 'df2' where there are matching values in join columns
- **left/right_join():** combines all rows from the left/right table and the matched rows from the right/left table
  - Example: left_joined_df <- df1 %>% left_join(df2, by = "key") combines all rows from 'df1' (left table) with matched rows from 'df2' (right table)

Stringr: work with strings and regular expressions

- Quotes in strings: either use “” for string and ‘’ for quotes or use \” to specify “ within a string
- **Detecting Matches:** finding matches within a string
  - str_detect(string, pattern) → T/F if pattern in a string, str_count(string, pattern) → # of time string contains pattern
- **Subsetting:** selecting part of a string
  - str_sub(string, start, end) → extracts start to end substring, str_extract(string, pattern) → returns first match of pattern: NA if no match
- **Length Functions:** str_length(string) → returns length,
  - str_pad(string, width = , side = , pad =) → pads the string to width, adding pad character to side (left, right or both)
- **Mutating Strings:** can set str_string() = new_string → sets substring to new_string
  - str_replace(string, pattern, replacement) → replaces first pattern match with replacement
  - str_to_lower(string) and str_to_upper(string) → strings to lower or upper case
- **Joining/Splitting Strings:** str_c(str(s), sep =) → joins string together, separating by sep
  - str_split(string, pattern, n) → splits a string into vector objects about the pattern returning at max n elements
- **Regular Expressions:** Anchors: ^ → start of line, $ → end of line
  - Alternates: ab|d = ab or d, [abe] = a or b or e, [^abe] = anything but a, b or e, [a-c] = anything between a and c
  - Quantifiers: a? = 0 or 1 a’s, a* = 0+ a’s, a+ = 1+ a’s, a{n} = n a’s, a{n,} = n+ a’s, a{n,m} = >=n & <= m a’s
  - Groups: use parentheses to separate groups and set precedence
    - (ab|e)c = abc or ec, can us \\1 to refs first group (\\2 refs 2nd group, etc): (b|c)e\\1 = beb | ceb | bec | cec

Readr: importing data from different file types

- read_csv(): csv files, read_csv2(): vals separated by “.”, read_tsv(): tab separated
- read_delim(filepath, delim =): specify how vals are delimited, read_excel(), read_table()

Tidyr:

- **pivot_longer():** converts data from wide format to long format, gathering multiple columns into key-value pairs
  - Example: longer_df <- df %>% pivot_longer(cols = c(x, y), names_to = "variable", values_to = "value") converts 'df' from wide format to long format, considering columns 'x' and 'y'
- **pivot_wider():** transforms data from long format to wide format, spreading key-value pairs across multiple columns
  - Example: wider_df <- df %>% pivot_wider(names_from = variable, values_from = value) transforms 'df' from long format to wide format

Lubricate:

- **ymd(),dmy(),mdy(),etc:** functions used to parse character strings into date objects
  - Example: date_df <- df %>% mutate(date_column = ymd(date_string)) parses a string into a Date object assuming the format is year-month-day
- **detect():** chooses specific columns in a data frame

Data Handling

- **Missing Values:**
  - Detect: is.na()
  - Remove rows: filter() or na.omit()
  - Replace values: mutate() with ifelse() or c
- **Valid Names for an R Object:**
  - Start with: Letter or dot (if followed by a letter)
  - Can contain: Letters, numbers, underscores, dots
  - Avoid: Reserved words (e.g., if, else, TRUE, FALSE)
- **Data Filtering:**
  - Command: obesity %>% filter(!is.na(n), sex == "female", age != "05-17")
  - Keeps rows where n is not missing, sex is female, and age is not 05-17.
- **Pipe Operator %>%**
  - Passes output of one function to the next.

Stats Terms

- **Mean**: Average value; sensitive to outliers. Indicates data center.
- **Median**: Middle value when data is sorted. Less affected by outliers.
  - Right-skewed: Mean > Median
  - Left-skewed: Mean < Median
  - Symmetric: Mean ≈ Median
- **Mode**: Most frequent value(s). May have none, one, or several modes.
- **Standard Deviation (SD):** Measures data spread.
  - Low SD: Data close to the mean.
  - High SD: Data spread out.
  - Normal Distribution: ~68% of data within 1 SD, ~95% within 2 SD, ~99.7% within 3 SD.
- **Quartiles:**
  - Q1: Median of the first half.
  - Q2: Median.
  - Q3: Median of the second half.
  - IQR: Q3 - Q1. Helps identify outliers.
- **P-Values:** Probability that observed data would occur by chance.
  - < 0.05: Typically indicates significance.
- **Confidence Intervals:** Range likely containing the true value.
  - 95% confidence: Standard for reliability.

Problem 1. Which type of graph is typically most effective at displaying the relationship between two quantitative variables. (

(a) bar chart (b) density plot (c) histogram (d) scatter plot

(d) scatter plot

Problem 2. Which of the following commands calculate the proportion of missing values in a vector X? **Circle all correct an**

a. `mean(is.na(X))`
b. `mean(X == NA)`
c. `sum(X == NA) / count(X)`
d. `sum(is.na(X)) / length(X)`

(a) and **(d)**

Problem 4. Which command keeps all rows of the a data set `obesity` where the variable `n` is not missing, the variable `sex` is "female", and the variable `age` is not "05-17"? **Circle one answer.**

a. `obesity %>% filter(n != NA, sex = "female", age != "05-17")`
b. `obesity %>% filter(n != NA, sex == "female", age == !"adult")`
c. `obesity %>% filter(!is.na(n), sex == "female", age == !"05-17")`
d. `obesity %>% filter(!is.na(n), sex == "female", age != "05-17")`

(d)

Problem 5. The plot of a histogram of a variable `X` from a data set `df` appears to be symmetric, bell-shaped, and centered near 0. The value of `df %>% summarize(sd = sd(X))` is `15.0`. Which numerical value will be closest to the result of `df %>% summarize(p = mean( between(X, -30, 30)))` ? **Circle one answer.**

(a) 0 (b) 0.68 (c) 0.95 (d) 99

(c)

Problem 6. What is the output of the following command, (suppressing variable names), where March 8, 2023 is a Wednesday and August 3, 2023 is a Thursday? **Circle one answer.**

```
tibble(date = mdy("3/8/2023")) %>%
  summarize(day = day(date),
            wday = wday(date, label = TRUE),
            yday = yday(date),
            month = month(date, label = TRUE))
```

a. `8  Wed  67 Mar`
b. `8  67  Wed Mar`
c. `3  Thu  215 Aug`
d. `Thu 3  215 Aug`

(a)

Problem 5. A box plot of a variable `x` of length 100 has a box with edges at the values 10 and 50, a line inside the box at 20, whiskers which extend from below the box to 5 and from above the box to 90. There are three potential outlier values plotted between 120 and 150. Which values could possibly be equal to the 90th percentile (0.9 quantile) of `x` ? **Circle all correct answers, cross out incorrect answers.**

(a) 70 ~~(b) 100~~ ~~(c) 130~~

The 0.9 quantile is not one of the three largest values out of 100 and greater than or equal to the 0.75 quantile. The value must be in the whisker, between 50 and 90. 70 is the only possibility.

Problem 6. What is the value of the output of the following code? Note that October 20, 2023 is a Friday. **Circle all correct answers, cross out incorrect answers.**

```
ymd("2023/10/20") %>% day()
```

~~(a) 5~~ (b) 20 ~~(c) "Fri"~~

The day is 20. `wday()` would have returned `6` for the sixth day of the week.

Problem 7. A data set `grocery_items` has 20 rows and variables named `item`, `type`, and `price`. Every `item` value is unique. A data set named `grocery_list` has 8 rows and variables named `item` and `n`. Every `item` value is unique and each matches an `item` value in `grocery_items`. Which commands produce a data frame with 8 rows? **Circle all correct answers, cross out incorrect answers.**

a. `grocery_items %>% inner_join(grocery_list)`
~~(b) `grocery_list %>% left_join(grocery_items)`~~
b. `grocery_list %>% semi_join(grocery_items)`

There are 8 matches in common and all 8 rows from `grocery_ list` match rows in `grocery_items`.

The `obesity` data set includes one row for each `zip/sex/age` combination, a total of 7740 rows, and variables:

- `zip` is a zip code (a 5-digit a categorical variable);
- `sex` is either "female" or "male";
- `age` is an age range from "05-17", "18-34", "35-54", "55-74", and "75+";
- `obese` is the number of sampled individuals who are obese;
- `n` is the number of sampled individuals;
- `pop` is the population of individuals in the zip code/sex/age range combination; and
- `obese_pop = pop * (obese / n)` is an estimate of the number of individuals in `pop` who are obese: `obese`, `n`, and `obese_pop` contain some missing data.

The `education` data set has one row for each zip code and variables:

- `zip` as above;
- `pct_f_bach` which is the percentage of women aged 25 and older with a bachelors degree; and
- `pct_m_bach` which is the percentage of men aged 25 and older with a bachelors degree

```
summary_2 = education_2 %>%
  group_by(sex) %>%
  summarize(U = 100*sum(obese_pop) / sum(pop),
            V = 100*sum(z) / sum(pop))
```

```
summary_2
```

```
## # A tibble: 2 × 3
##   sex       U     V
##   <chr>  <dbl> <dbl>
## 1 female  39.8  31.8
## 2 male    41.3  30.0
```

roblem 13 What do the values in columns U and V represent for each sex in `summary_2`?

The values in column U are the percentages of obese adults by sex (among zip codes with no missing obesity data for adults). The values in column U are the percentages of adults with a bachelors degree by sex (among zip codes with no missing obesity data for adults).

```
summary_1 = obesity %>%
  drop_na(obese) %>%
  group_by(sex, age) %>%
  summarize(v = 100 * sum(n) / sum(pop)) %>%
  ungroup() %>%
  mutate(sex = recode(sex,
                      "female" = "A",
                      "male" = "B")) %>%
  pivot_wider(names_from = sex, values_from = v)
```

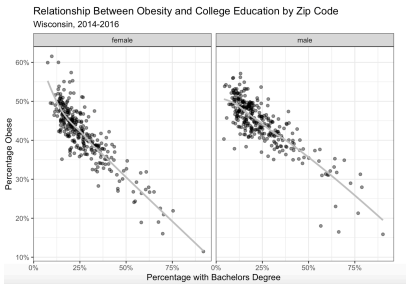
```
summary_1
```

```
## # A tibble: 5 × 3
##   age       A     B
##   <chr> <dbl> <dbl>
## 1 05-17  43.4  42.8
## 2 18-34  38.8  29.3
## 3 35-54  42.0  35.5
## 4 55-74  50.8  47.1
## 5 75+    56.3  59.6
```

roblem 12 What do the values in column A and column B represent for each age group in `summary_1`?

The table values are the proportion of individuals from each population included in the sample data with a row for each age group, column A for females, and column B for males.

```
g = ggplot(education_2, aes(x = w/100, y = y/100)) +
  geom_point(alpha = 0.5) +
  geom_smooth(se = FALSE, color = "gray") +
  xlab("") +
  ylab("") +
  scale_x_continuous(labels = scales::percent) +
  scale_y_continuous(labels = scales::percent) +
  facet_grid(cols = vars(sex)) +
  theme_bw()
```



Problem 7. A data set `grocery_items` has variables named `item`, `type`, and `price`. A data set named `grocery_list` has variables named `item` and `n`. The values in the columns `item` match if the same item is part of both data sets. Some items in `grocery_items` may not by in `grocery_list` and some items in `grocery_list` may not by in `grocery_items`. Which description matches the contents of `df` after executing the following code? Neither individual data set has multiple copies of the same item. **Circle one answer.**

```
df = grocery_list %>%
  full_join(grocery_items, by = "item")
```

a. A data frame with one row for each item in both data sets and columns `item` and `n` only.
b. A data frame with one row for each item in both data sets and columns `item`, `n`, `type` and `price`.
c. A data frame with one row for each item in `grocery_list` and columns `item`, `n`, `type` and `price`.
d. A data frame with one row for each item in either data set, columns `item`, `n`, `type` and `price`, and the value `NA` in columns `n`, `type`, and `price` in rows where this information was missing.

(d)

Problem 8

A data frame `sfo` has no missing data, $7 \times 24 = 168$ rows, and columns named `Day`, `Hour`, and `n` where `Day` is an abbreviated day of the week (one of `Sun`, , `Sat`), `Hour` is one of `0000-0100` to `2300-0000`, and `n` is an integer count. Each possible combination of `Day` and `Hour` is included in the data frame.

The data frame `df` created with the following code will have how many rows, how many columns, and what will the column names be? **Circle one answer.**

```
df = sfo %>%
  pivot_wider(names_from = Day, values_from = n)
```

a. 168 rows and 3 columns named `Hour`, `Day`, and `n`
b. 24 rows and 8 columns named `Hour`, `Sun`, , `Sat`
c. 24 rows and 7 columns named `Sun`, , `Sat`
d. 7 rows and 25 columns named `Day`, `0000-0100`, , `2300-0000`

(b)

Problem 1. A scatter plot displays the relationship between the percentages of obesity and attainment of a college degree across zip codes in Wisconsin. How could the graph be modified to show this relationship separately for females and males? **Circle all correct answers, cross out incorrect answers.**

a. Use different colors for each sex.
b. Use faceting to show each sex in a different plot panel.
~~(c) Use a density plot for one sex and a histogram for the other.~~

Density plots and histograms visualize a single quantitative variable, not two.

Problem 2. A data frame `mw` has one row per day for over 150 years of official Madison weather data and includes a column `tavg` with the average daily temperature and a column `year` with the year. Which commands contain correct code to find the average annual temperature for each year, accounting for having a few dates with missing data? **Circle all correct answers, cross out incorrect answers.**

Problem 8 Identify each true expression for the data frame `df` shown below. The function `nrow()` returns the number of rows in a data frame. **Circle all correct answers, cross out incorrect answers.**

```
##         color
## 1        gold
## 2 chartreuse4
## 3      gray67
## 4        snow
## 5     sienna3
```

a. `df %>% filter(str_detect(color, "^g")) %>% nrow() == 2`
~~(b) `df %>% mutate(f = str_sub(color, 1, 1)) %>% filter(f == "s") %>% nrow() == 3`~~
~~(c) `df %>% filter(str_detect(color, "[:digit:]+")) %>% nrow() == 3`~~

2 colors start with `g`; only 2 colors start with `s`; all 5 colors have 0 or more consecutive digits.

Problem 3. Base R vectors `x` and `y` contain numerical values and are the same length. Identify correct code to find the mean of their squared differences. (The mean of $(x_i - y_i)^2$.)

Circle all correct answers, cross out incorrect answers.

~~(a) `mean(x-y)^2`~~
(b) `mean( (x-y)^2 )`
(c) `sum( (x-y)^2 ) / length(x)`

Need to square before taking the mean.

Problem 4. A data frame `df` has 60 rows and columns `period`, `month`, and `tavg` with a label for a 30-year period of time, the month, and the average monthly temperature, respectively. There is one row for each month and five 30-year period. Which code will correctly reshape the data table to contain one row for each month and one column for each 30-year period? **Circle all correct answers, cross out incorrect answers.**

```
## first four rows of the input data frame df:
```

```
##      period month tavg
## 1 1871-1900   Jan 27.8
## 2 1871-1900   Feb 26.6
## 3 1871-1900   Mar 36.7
## 4 1871-1900   Apr 38.9
```

~~(a) `df %>% pivot_longer(names_to = month, values_to = tavg)`~~
(b) `df %>% pivot_wider(names_from = period, values_from = tavg)`
~~(c) `df %>% pivot_wider(period, names_from = month, values_from = tavg)`~~

`pivot_wider()`, new column names from `period` and values from `tavg`

```
df %>%
  select(-x) %>%
  filter(day != "Tue") %>%
  group_by(color, day) %>%
  summarize(min = min(y))
```

```
## # A tibble: 3 × 3
## # Groups:   color [2]
##   color day     min
##   <chr> <chr> <dbl>
## 1 blue  Mon       0
## 2 blue  Wed       5
## 3 red   Mon     -10
```

```
df %>%
  mutate(m = x*y) %>%
  group_by(day) %>%
  slice_max(m, n = 1)
```

```
## # A tibble: 3 × 5
## # Groups:   day [3]
##       x     y color day       m
##   <dbl> <dbl> <chr> <chr> <dbl>
## 1     2     0 blue  Mon       0
## 2     4     0 red   Tue       0
## 3     5    10 blue  Wed      50
```

Problem 9. Write the result of the following code.

```
##   x  y color day
## 1 1 -1   red Mon
## 2 2  0  blue Mon
## 3 3 -5  blue Tue
## 4 4  0   red Tue
## 5 5 10  blue Wed
## 6 6  5  blue Wed
```

```
df %>%
  count(color) %>%
  arrange(desc(n))
```

```
df %>%
  count(color) %>%
  arrange(desc(n))
```

```
## # A tibble: 2 × 2
##   color     n
##   <chr> <int>
## 1 blue      4
## 2 red       2
```