# Summary Statistics for Categorical Data

## Categorical Data Spread

Calculating a mean for ordinal variables would be inappropriate because the spacing between categories may be uneven. Since standard deviation and variance both depend on the mean, these statistics should not be used to summarize categorical data. Instead, measures of spread for ordinal categorical data include percentiles and range.

## Central Tendency of Ordinal Categorical Data

For ordinal categorical data, both the median and mode can be calculated as measures of central tendency; a mean is not appropriate because it assumes equal spacing between categories. For nominal categorical data, the mode is calculable and interpretable (as the most common recorded value), but the mean and median are not.

## Encoding Ordinal Categorical Data

In order to calculate summary statistics for ordinal categorical data (eg., a median or percentile), many functions, like `np.median` , require numeric inputs. It is therefore helpful to store category names both as strings and as numerical values (eg., integer or float data types). This can be done using label encoding (in Pandas, using the `cat.codes` attribute of a 'category' dataframe column). For example, suppose there is a variable named `response` in a dataframe named `df` that contains responses to the question "Rate your agreement with the statement: the wealthy should pay higher taxes," where the response options are "strongly disagree", "disagree", "neutral", "agree" and "strongly agree". The provided code can be used to calculate the median category for this data.

```
import pandas as pd

cagetories = ["strongly disagree", "disagree",
"neutral", "agree" and "strongly agree"]
df.response = pd.Categorical(df.response,
categories, ordered=True)
median_value
= np.median(df["response"].cat.codes)
median_text = categories[int(median_value)]
```

## Categorical Data Frequencies

One way to summarize a categorical variable is to compute the frequencies of the categories. For further summarization, the frequency of the modal category (most frequent category) is often reported. For example, when analyzing a dataset with an education level variable (highschool, associates, bachelors, masters, etc.), we could calculate the frequency of each category and report the most common category. For a pandas dataframe, we can use the `.value_counts()` method on a column of data to calculate the frequencies of the categories.

```
# calculate counts of values for a column in
a dataframe:
df['column_name'].value_counts()
```

## Proportions

When summarizing categorical data, proportions are often more useful than frequencies, especially for individual categories. For example, knowing that 75% of respondents said that they “strongly agree” with some statement provides more information than that 150 respondents said they “strongly agree” (which is only interpretable in context of how many total people responded).

## Misleading Proportions

Proportions make our data more interpretable but they can also be misleading, especially when the sample size is small. For example, would it be fair to compare a restaurant with a 5-star rating and only 1 review against a restaurant that has a 5-star rating out of 200 reviews? To avoid misleading reporting we can provide the proportions along with the total sample size by using the `.value_counts()` function in pandas and then subsequently converting those frequencies to proportions using `normalize=True` inside `value_counts()` .

```
df.value_counts(normalize = True)
```

## Calculating Proportions

Proportions are often used to summarize categorical data and can be calculated by dividing individual frequencies by the total number of responses. In Python/pandas, `df['column_name'].value_counts(normalize=True)` will ignore missing data and divide the frequency of each category by the total in any category. This can also be done explicitly with something like: `df['column_name'].value_counts()/df['column_name'].value_counts().sum()` . The interpretation is something like, “among people who answered the question, 40% said they strongly agree”.
Alternatively, in order to count missing data in the denominator, we could use: `df['column_name'].value_counts()/len(df['column_name'])` . The interpretation is something like, “among all people who were surveyed, 40% said they strongly agree”.

## Binary Variables

For binary variables coded as 1/0 or True/False (which get’s coerced to 1/0 in many programming languages), the mean of the variable is equal to the proportion of 1’s or True’s. Meanwhile, the sum of the variable is equal to the frequency of 1’s or True’s. For example, if we have a variable indicating whether or not a customer made a purchase and the contents of that variable looks something like `[True False True True False False False False True]` , we can calculate the proportion of values that are `True` by calculating the mean and the number of purchases by calculating the sum.

```
outcome = np.array([True, False, True, True,
False, False, False, False, True])

# proportion of true values:
print(np.mean(outcome)) #output: 0.44444

# number of true values:
print(np.sum(outcome)) #output: 4
```