Visualizations of ContinuousVariables

David W. Gerbing
The School of Business
Portland State University

April 6, 2026

Excerpted from Chapter 2 and Chapter 5 of the forthcoming 2nd edition:

Data Visualization: Derive Meaning from Data

CRC Press

```
Cramer's V: 0.415

Chi-square Test:  Chisq = 6.200, df = 4, p-value = 0.185
>>> Low cell expected frequencies, chi-squared approximation may not be accurate
```

Listing 2.3: `Chart()` statistical output to evaluate the relationship among two categorical variables.

2.3 Distribution of a Continuous Variable

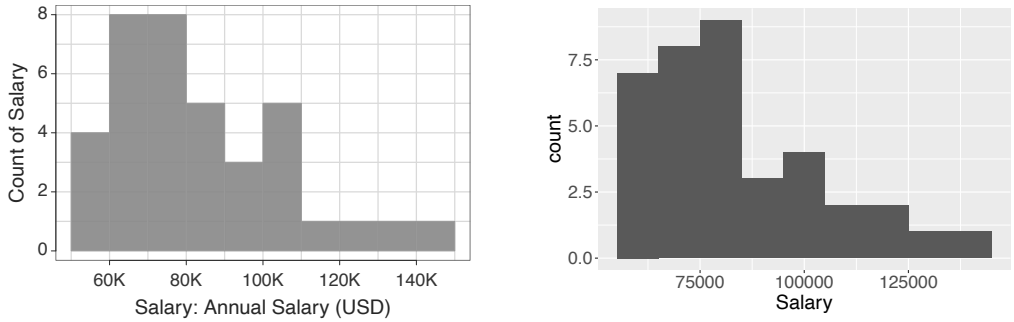
bin: An interval of similar values of a continuous variable.

The histogram visualizes the distribution of a continuous variable, such as *Age*, *Salary*, *MPG*, or *Height*. The histogram groups the underlying continuous distribution spanning the range of data values into intervals, generally of equal width, called *bins*. Place each data value in its corresponding bin, then count the number of values in each bin. To construct the histogram, usually place the bins on the horizontal axis. Then construct a bar over each bin to represent the corresponding count of the number of data values contained within that bin. Unlike a bar graph of a categorical variable, a line graph shows the underlying continuity of the numerical scale, with adjacent bins sharing a common border.

2.3.1 Default Histogram

histogram: Place each data value in its corresponding bin, represented by a bar with a height proportional to the frequency of its values.

Grayscale `lessR` and `ggplot2` histograms appear in Figure 2.4.



(a) lessR **(b)** ggplot2

Figure 2.4: Default lessR histogram, and ggplot2 histogram with set bin width.

Interpretation: The histogram of *Salary* reveals that among these 37 employees, the most frequently occurring salaries range from $60,000 to $80,000. In contrast, salaries exceeding $100,000 are relatively rare. The salaries range from about $55,000 to about $145,000.

workspace, Section 1.1.3, p. 8

R Input *Default histogram:* Salary *from* d *data frame*

data: `d <- Read("Employee")`

lessR: `X(Salary)`

ggplot2: `ggplot(d, aes(Salary)) + geom_histogram(binwidth=10000)`

`lessR Default histogram`

- ▷ Function `X()`: Visualize the distribution of a continuous variable, such as with a histogram.
  - Parameter `x`: Create the histogram for the specified continuous variable, *Salary*.
  - Optional parameter `type`: Specify `"histogram"`, which is not needed as that is the default value.
  - *Optional* parameter `data`: The default data frame name is *d*. Need to include this parameter for an input data frame with another name.
  - *Optional* parameter `bin_width`: Specify a specific value for the bin width.

`ggplot2 Default histogram except specified bin width`

- ▷ Function `ggplot()` : Specify the input data frame, *d*.
  - Function `aes()`: Specify the variable, *Salary*, for the *x*-axis variable.
- ▷ function `geom_histogram()`: Create the histogram.
  - Parameter `binwidth`: Specify the bin width for the created histogram, 10000. Presumably to encourage exploration of an optimal bin width, `ggplot2` requires to set this parameter without a default value.

The `lessR X()` output includes summary statistics and an outlier analysis, shown in Listing 2.4.

```
--- Salary: Annual Salary (USD) ---

     n   miss       mean         sd        min        mdn        max
    37      0   63795.56   21799.53   36124.97   59547.60  124419.23


(Box plot) Outliers: 1

Small      Large
-----      -----
            124419.2
```

Listing 2.4: Statistical summary analysis provided by the `lessR` function `X()`.

Also included in the histogram output is the frequency distribution table of the bins, bin midpoints, frequencies or counts, relative frequencies or proportions, and cumulative relative frequencies, as well as summary statistics, and an outlier analysis, shown in Listing 2.5.

```
--- Salary: Annual Salary (USD) ---

Bin Width: 10000
Number of Bins: 10

             Bin  Midpnt  Count    Prop  Cumul.c  Cumul.p
-------------------------------------------------------------
  30000 >  40000   35000      4    0.11        4     0.11
...
 110000 > 120000  115000      1    0.03       36     0.97
 120000 > 130000  125000      1    0.03       37     1.00
```

Listing 2.5: Frequency distribution analysis provided by the `lessR` function `X()`.

VBS Plot: Integrated violin, box and scatterplot for the display of the distribution of a continuous variable.

boxplot, Section 5.3, p. 130

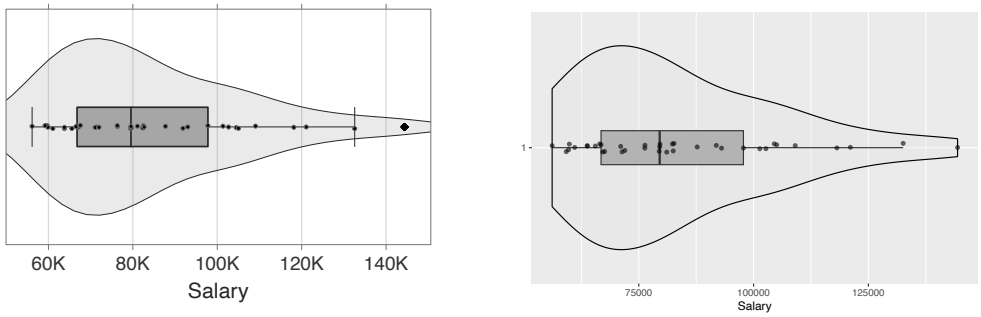
outlier, Section 5.16, p. 130

VBS plot, Section 5.4, p. 134

2.3.2 The VBS Plot

Modern computer technology provides more effective visualizations than the 19th-century histogram, such as the VBS plot for displaying the distribution of a continuous variable (Gerbing, 2024). The VBS plot integrates three different but complementary plots into a single visualization: a violin plot (Hintze & Nelson, 1998), a box plot (Tukey, 1977), and a 1-variable scatterplot called a strip plot that flags outliers, and then, when generated by `lessR`, automatically tunes the completed visualization in terms of the sample size and other characteristics of the distribution. Find a more extensive discussion of all three subplots and of the VBS plot in Chapter 5 beyond the following introduction.

Figure 2.5 illustrates the `lessR` and `ggplot2` VBS plots for a small data set of 37 annual salaries.



(a) lessR **(b)** ggplot2

Figure 2.5: Integrated Violin/Box/Scatterplot or VBS Plot.

Interpretation: The VBS plot of *Salary* shows that for these 37 employees, the most frequently occurring salaries range from \$60,000 to \$80,000, with relatively few salaries over \$100,000. The boxplot marks the highest value of *Salary*, over \$120,000, with the largest salary an outlier.

R Input *R Input VBS Plot of* Salary *from the* d *data frame*

data: `d <- Read("Employee")`

lessR: `X(Salary, type="vbs")`

ggplot2:
```
ggplot(d, aes(x=factor(1), y=Salary)) +
  geom_violin(fill="gray90", color="black", alpha=0.35) +
  geom_boxplot(width=0.15, fill="gray70", outlier.shape=NA) +
  geom_jitter(width=0.04, height=0, shape=16, alpha=0.6, size=1.8) +
  coord_flip() + labs(x=NULL, y="Salary")
```

lessR VBS plot

The VBS plot demonstrates a primary goal of `lessR`: Consider the most beneficial information, and then provide the relevant visualization and statistical analysis with a simple function call given appropriate defaults. Each layer of the plot is implicitly added by the calling function. Further, the function automatically adjusts characteristics of the plot according to the sample size and distribution type, minimizing user trial-and-error to discover the most appropriate parameter values.

- ▷ Function `X()`: Create a strip plot, a 1-dimensional scatterplot, in this example for a single continuous variable. Also superimpose a box plot and violin plot, integrated as a VBS plot.
  - Parameter `x`: Create the VBS plot for the specified continuous variable. *Salary*.
  - Parameter `type`: Specify `"vbs"`.

For additional explanation, see Section 5.4.1, Page 134.

ggplot2 VBS plot

Construct the VBS plot by explicitly adding each of the three components as separate layers. Define each layer with its own `geom` and then run successive visualizations manually tuning the parameters such as bandwidth and point size to obtain the desired result.

- ▷ Function `ggplot()` : Specify the input data frame, *d*.
  - Function `aes()`: Specify the y-variable, *Salary*, from which to create the VBS plot
  - Parameter `x`: The expression `factor(1)` creates a factor, that is, a categorical variable, with a single level, which places all observations into one group on the x-axis. The specific value inside `factor()` is arbitrary. The choice of 1 is simply a convenient convention to indicate a single category.
  - Parameter `y`: Specify the continuous variable from which to construct the VBS plot, *Salary*.
- ▷ Functions `geom_violin()`, `geom_box()`, and `geom_jitter()` and more: For additional explanation, see Section 5.4.1, Page 135.

To re-iterate a fundamental visualization theme: `lessR` provides the simplicity to obtain a pre-programmed result and `ggplot2` provides the flexibility, at the cost of additional complexity, to create what can be conceptualized.

ggplot2 VBS code detail, Section 5.20, p. 134

Chapter 5

Visualize a Continuous Variable

histogram, Section 2.4, p. 38

VBS plot, Section 5.4, p. 133

A visualization of the distribution of a continuous variable estimates the shape of the underlying population distribution from a sample of that population. Chapter 2 introduced two such visualizations: the histogram as well as a modern replacement, the VBS plot. This chapter expands upon this material for both and also shows other visualizations.

5.1 Histogram

5.1.1 Bin Continuous Data

Group data values of a continuous variable into bins. In contrast to the relatively few unique values of a categorical value, a continuous variable has potentially too many unique data values to plot individually. Consider annual USD salary for employees of a company. The salaries are in the ranges of tens of thousands of dollars, but each penny must be considered. Many, if not most, potential values never occur in the data, and for those that do occur, only once or a few times at most. For example, a *Salary* of \$69,424.79 would likely not occur except for a huge sample size.

bins: Sequence of adjacent intervals, each generally of the same size.

What should be done with the unique data values that cannot be meaningfully plotted individually? Modern computer technology offers solutions, such as the VBS plot (Gerbing, 2024). Older techniques, such as the classical histogram, emerged from 19th-century paper and pencil technology. The histogram divides the range of values into *bins*, or *classes*, each containing similar data values. Figure 5.1 illustrates a bin that includes values ranging from \$60,000 to \$70,000 for annual USD salaries.

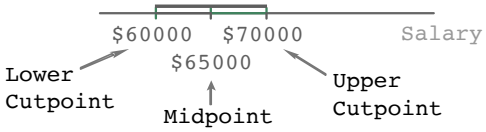


Figure 5.1: Example of a bin defined over the range of data values from \$60,000 up to \$70,000.

cutpoints: Lower, upper boundaries of a bin.

bin width: Width of a bin.

midpoint: Value in the middle of a bin, summary of all values in the bin.

Define each bin by its lower and upper boundary, its *cutpoints*, which sets the width of each bin, the *bin width*. The single point that most effectively summarizes the values of the bin is the *midpoint*, which is not necessarily a data value. In Figure 5.1, bin width is \$10,000 and the midpoint is \$65,000. Place each data value into its respective bin. Consistently assign values precisely equal to a cutpoint to either the adjacent lower bin or the adjacent higher bin.

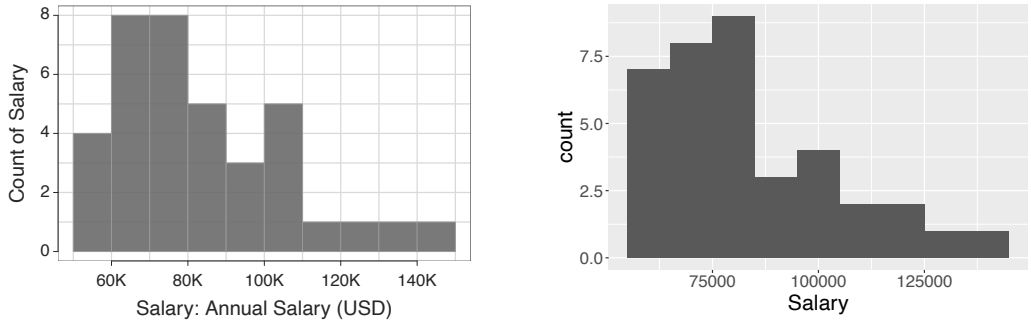
histogram, Section 2.4, p. 38

histogram: Place each data value of a continuous variable into its bin, plotted as a bar with height proportional to its frequency.

The histogram. The most common visualization of a continuous variable is the *histogram*. To create the histogram, first bin the data, that is, sort the individual data values into distinct intervals or bins. Like the bar graph, the histogram consists of bars. Unlike the bar graph, the adjacent bars of a histogram share a common side to indicate the underlying continuity of a continuous variable.

The default `lessR` histogram, and a `ggplot2` histogram, both from Figure 2.4, are repeated here in Figure 5.2 for convenience. By default, `ggplot2` plots a histogram

with a bin width that is deliberately too small to be meaningful, presumably to encourage exploration of different bin widths. Both histograms in Figure 2.4 have a bin width of $10,000 but differ because of different starting points.



(a) lessR **(b)** ggplot2

Figure 5.2: Histograms of the same data and bin width but different start values.

R Input *Default histogram*

data: `d <- Read("Employee")`

lessR: `X(Salary)`

ggplot2: `ggplot(d, aes(Salary)) + geom_plot(binwidth=10000)`

`lessR Histogram`

- ▷ Function `X()`: Visualize the distribution of of a continuous variable, such as with the histogram.
  - Parameter `x`: The variable from which to construct the histogram, here *Salary*.
  - Parameter `type`: Rely upon the default value of `"histogram"`.
  - Parameter `bin_width`: By default, Base R `hist()` assigns a value that is equal to a cutpoint to the lower bin, a rule also retained by `X()`. To customize, specify the width of the histogram bins.

`X()` extends the Base R histogram function, `hist()` and retains its defaults such as for computing bin width, as well as direct access to many of its parameters.

`ggplot2 Histogram`

- ▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it from which to construct the histogram, *Salary*.
- ▷ Function `geom_plot()`: Plot the histogram.
  - Parameter `binwidth`: Specify the width of the histogram bins.

As shown in Listing 2.5, `X()` generates a statistical analysis in addition to the visualization, which includes the frequency distribution table, summary statistics, and an outlier analysis.

X() output, Section 2.5, p. 40

Many options are available to customize the colors of the histograms. Display the bars in the same color or in a gradient of related colors, such as a sequential color pallet. Or, one or more bars can be displayed in colors distinct from the other bars.

fill and color parameters, Section 10.1.2, p. 281

sequential color palettes, Section 10.2.1, p. 284

5.1.2 Histogram Artifacts

bin width artifact: Shape of the histogram depends on the bin width

bin shift artifact: Shape of the histogram depends on the bin starting point

Each histogram's shape depends on both the starting point of the bins as well as the bin width. Change one or both of these values, and the shape of the histogram may change, as illustrated in Figure 5.2. For example, the histograms from `lessR` and `ggplot2` are of the same data with the same bin width of $10,000, yet they differ. Why? The `lessR` histogram has a default start point for the first bin at $30,000. Setting the `bin_start` parameter to $35,000 results in the same histogram computed by `ggplot2` in Figure 5.2b.

What are the correct bin width and bin starting values for the data? There are no correct answers in the sense of solving a mathematical equation as the selected values depend somewhat on personal preference. For the bin width, generally choose the smallest reasonable bin width supported by the available data to display the underlying distribution.

Unfortunately, the algorithms that choose a default bin width do not necessarily provide the preferred bin width. By trial and error, you may create a visualization more representative of the corresponding population distribution than the default bin width. Usually, the underlying population distribution resembles a smooth curve in one direction, or, as with the normal curve, a smooth curve to a maximum (or, inverted, minimum), and then a smooth curve in the other direction. The histogram in Figure 5.3, with a bin width of $15,000 and a starting value of $48,000, provides a histogram that follows this pattern for these employment data that better represents the underlying population distribution than do the defaults.

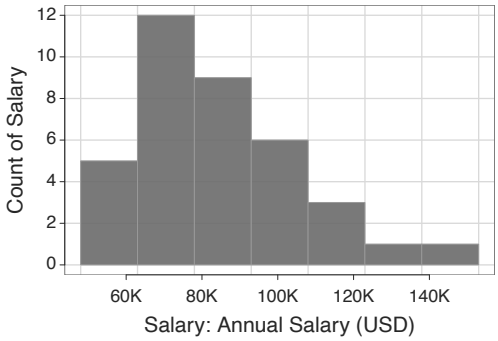
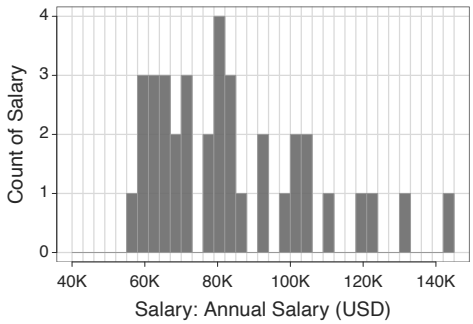
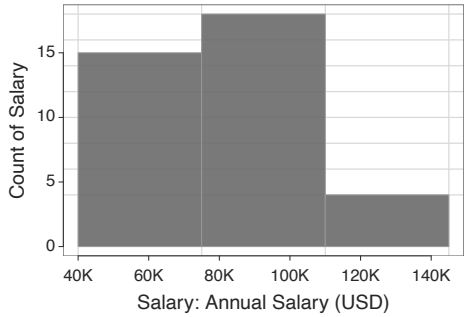


Figure 5.3: A more optimal histogram.

At the other extreme, consider the histograms in Figure 5.4 that have very narrow or very wide bins.



(a) Undersmoothed



(b) Oversmoothed

Figure 5.4: Under-smoothed and over-smoothed histograms that need adjustment of bin width.

The narrow binned histogram results from the `ggplot2` bin width default of approximately $3500, the bin width of the histogram in Figure 5.4a. The large number of ups and downs of the histogram bars more likely indicate too small of a bin width for the given sample size instead of a reflection of the actual population distribution. These excessive ups and downs indicate *under-smoothing*. The many fluctuations likely represent sampling error and so would not replicate with a new sample of the same size.

under-smoothing: Bin width too small relative to the available data so that too many bins result in too much detail.

The other extreme is *over-smoothing*, illustrated in Figure 5.4b. An over-smoothed histogram results in bins too wide given the sample size, obscuring some of the available information. Instead, narrower bins result in more detail for the portrayal of the underlying distribution.

over-smoothing: Not enough bins results in the bin width too large, obscuring properties of the underlying distribution.

Generate the histograms in Figure 5.4 with different bin widths follow.

> *under-smoothed*: `X(Salary, bin_width=3000)`
> *over-smoothed*: `X(Salary, bin_width=35000)`

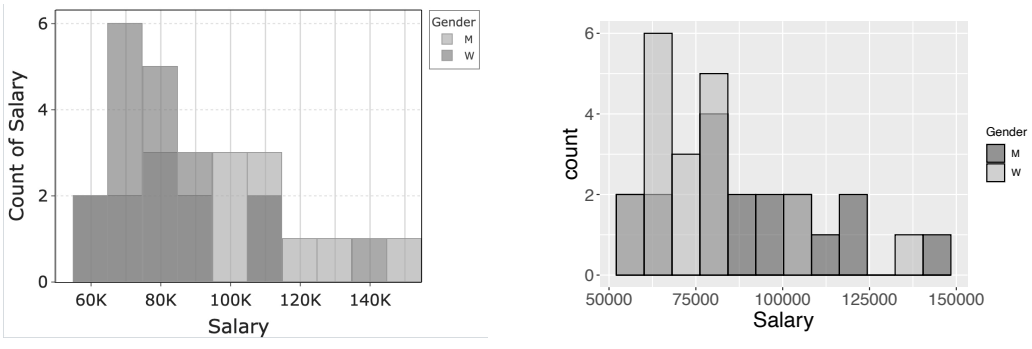
To obtain a more representative histogram may require adjusting both the bin widths and the starting point of the bins, such as with the `lessR` `bin_start` parameter.

5.1.3 Stratification

Stratification is the division of data into meaningful subgroups, or strata, according to the values of one or more categorical variables. Rather than viewing only the overall distribution, stratification makes it possible to examine the distribution within each subgroup. For example, the distribution of salary might be stratified by gender or department. The subgroup distributions can be displayed together on the same panel for direct comparison or separately on different panels, often called *facets* or *Trellis plots*. Stratification is useful because important differences among groups may be obscured when all the data are combined into a single overall distribution.

Overlapping Histograms

Figure 5.5 shows the overlapping histograms of *Salary* across *Gender* values M and F.



(a) lessR **(b)** ggplot2

Figure 5.5: Overlapping histograms of *Salary* across two genders.

R Input *Create overlapping histograms*

data: `d <- Read("Employee")`

lessR: `X(Salary, by=Gender)`

ggplot2:
```
d |> dplyr::filter(Gender %in% c("M", "W")) |>
     ggplot(aes(x=Salary, fill=Gender)) +
     geom_histogram(position="identity", alpha=0.6,
                    bins=12, color="black") +
     scale_fill_manual(values=c("M"="gray35", "W"="gray75"))
```

`lessR Overlapping Histograms`

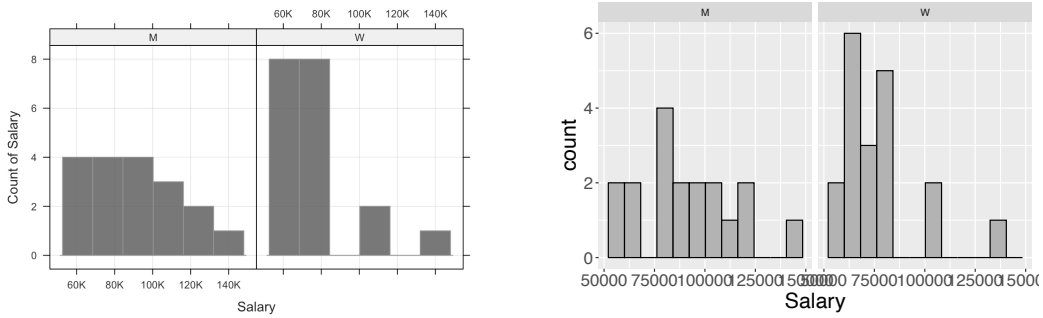
- Function `X()`: Rely upon the default value of `"histogram"` for optional parameter `type` for plotting the distribution of the specified variable *Salary* along the x-axis.
  - Parameter `by`: Indicate the categorical variable by which to plot the distributions for each level on the same panel.

`ggplot2 Overlapping Histograms`

- Function `aes()`: Specify the variables to map to the plot. Here, `x=Salary` maps the continuous variable *Salary* to the horizontal axis, and `fill=Gender` maps the categorical variable *Gender* to the fill aesthetic so that separate histograms are drawn for the two groups.
- Function `geom_histogram()`: Draw the histograms for the specified groups.
  - Parameter `position`: Set to `"identity"` so that the histograms are plotted in the same coordinate space and therefore overlap rather than stack.
  - Parameter `alpha`: Specify the transparency of the bars.
  - Parameter `bins`: Specify the number of histogram bins.
  - Parameter `color`: Specify the border color of the bars, here `"black"`.
- Function `scale_fill_manual()`: Specify the fill colors for the levels of *Gender*. Here, the two levels are assigned different shades of gray for a grayscale display.

Faceted Histograms

Figure 5.6 shows the facet histograms of *Salary* across *Gender* values M and F.



(a) lessR **(b)** ggplot2

Figure 5.6: Faceted histograms of *Salary* across two genders.

Faceting separates the distributions into different panels instead of overlapping them.

R Input *Create overlapping histograms*

data: `d <- Read("Employee")`

lessR: `X(Salary, facet=Gender)`

ggplot2:
```
d |> dplyr::filter(Gender %in% c("M", "W")) |>
     ggplot(aes(x=Salary)) +
     geom_histogram(fill="gray70", color="black", bins=12)
     facet_grid(cols=vars(Gender)) +
     scale_fill_manual(values=c("M"="gray35", "W"="gray75"))
```

lessR Faceted Histograms

- Function `X()`: Rely upon the default value of `"histogram"` for optional parameter `type` for plotting the distribution of the specified variable *Salary* along the x-axis.
  - Parameter `facet`: Indicate the categorical variable by which to create a facet for each level.

ggplot2 Faceted Histograms

- Function `aes()`: Specify the variables to map to the plot. Here, `x=Salary` maps the variable *Salary* to the horizontal axis.
- Function `geom_histogram()`: Draw the histogram for each facet.
  - Parameter `fill`: Specify the fill color of the bars, here a shade of gray.
  - Parameter `color`: Specify the border color of the bars, here `"black"`.
  - Parameter `bins`: Specify the number of histogram bins.
- Function `facet_grid()`: Create a separate panel, or facet, for each level of the specified categorical variable.
  - Parameter `rows`: Arrange the facets in rows. Here, `rows=vars(Gender)` creates one row panel for each level of *Gender*.
  - Function `vars()`: Specify the variable, here *Gender*, that defines the facets.

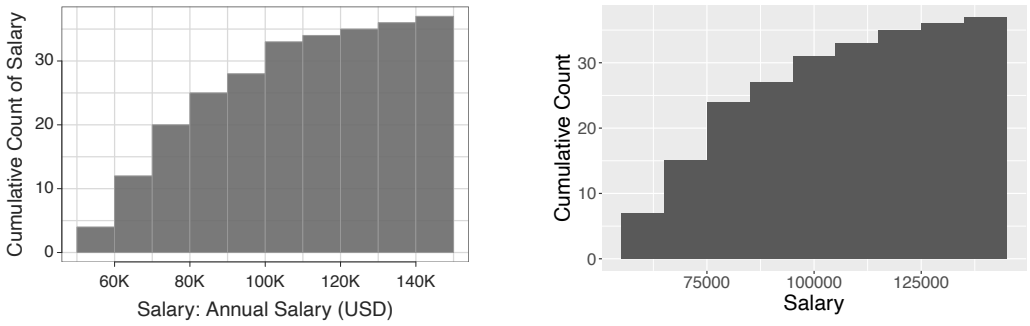
5.1.4 Cumulative Histogram

cumulative histogram: For each bin, sum the counts from the first bin to that bin.

A variation of the regular histogram is the *cumulative histogram*, which shows the running total up to each bin. For each bin, sum the counts from the first bin to that bin. The corresponding count for the second bin is added to the count for the first bin. And the third bin of the human of histogram is the sum of the count of the third bin plus the sum of the count of the two previous bins, and so forth. Figure 5.7 shows an example of the cumulative histogram based on the previously displayed histogram of *Salary*.

Superimposing the regular histogram over the cumulative version illustrates the relationship between the two histograms: The value of each cumulative histogram bar increases from the size of the previous bin. The cumulative histogram visualizes the number and percentage of values that fall below a specified threshold. Find these exact values in the default output of the `lessR` function `X()`, shown previously.

frequency and cumulative frequency distribution output, Section 2.5, p. 40



(a) lessR **(b)** ggplot2

Figure 5.7: Cumulative histogram.

R Input

data: `d <- Read("Employee")`

lessR: `X(Salary, cumulative="on")`

ggplot2:
```
p <- ggplot(d, aes(Salary)) +
       geom_plot(aes(y=cumsum(stat(count))), binwidth=10000) +
       labs(y="Cumulative Count")
     p
```

`lessR Cumulative histogram`

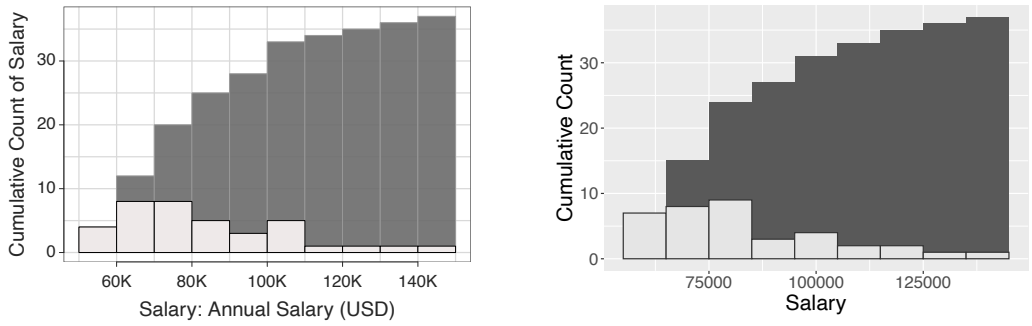
- ▷ Function `X()`: Plot the distribution of a continuous variable, Such as with a cumulative histogram, specifying the variable, *Salary*.
  - Parameter `type`: Rely upon the default value: `"histogram"`.
  - Parameter `cumulative`: Set to `"on"` to replace the regular histogram with the cumulative version.
  - Optional parameter `bin_width`: Specify the width of the histogram bins.

`ggplot2 Cumulative histogram`

Save the constructed visualization into the object named p for reference in a later visualization. To display `p`, as with any printable R object, enter its name at the console.

- ▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it from which to construct the histogram, *Salary*.
- ▷ Function `geom_plot()`: Plot the histogram.
  - Parameter `y` within `aes()`: Display the cumulative counts by invoking the Base R function `cumsum()` from the output of the `stat()` function evaluating the `count`. Because the counts are not from the data, but rather computed by `ggplot2`, reference them as the y variable with the `stat()` function.
  - Parameter `binwidth`: Specify the width of the histogram bins.

Figure 5.8 illustrates further development of the previous cumulative histogram. Here superimpose the regular histogram over the cumulative histogram.



(a) lessR **(b)** ggplot2

Figure 5.8: Cumulative histogram with a superimposed regular histogram.

Superimposing the regular histogram over the cumulative version illustrates the relationship between the two histograms: The value of each cumulative histogram bar increases from the size of the previous bin.

For `ggplot2`, explicitly add another histogram layer computed from the given data unmodified. From the saved object `p` from the last example, here add the layer to `p`.

R Input

data: `d <- Read("Employee")`

lessR: `X(Salary, type="histogram", cumulative="both")`

ggplot2: `p + geom_plot(binwidth=10000, fill="gray90", color="gray20")`

`lessR Cumulative histogram with superimposed regular histogram`

▷ Function `X()`: Plot the distribution of a continuous variable, such as a histogram.
- Parameter `x`: The variable from which to construct the histogram, *Salary*.
- Parameter `type`: Set to `"histogram"`.
- Parameter `cumulative`: Set to `"both"` to superpose the regular histogram on the cumulative version.
- Optional parameter `bin_width`: Specify the width of the histogram bins.

`ggplot2 Cumulative histogram with superimposed regular histogram`

Illustrated in this example is the flexibility of `ggplot2` to construct multiple layers for different analyses, each subsequent analysis building upon the previous analyses. Any `ggplot()` object can be saved, as in this example to the object `p`, displayed as is, and then another layer later added for a subsequent visualization.

▷ Object *p*: Visualization of the cumulative histogram saved from the previous analysis.

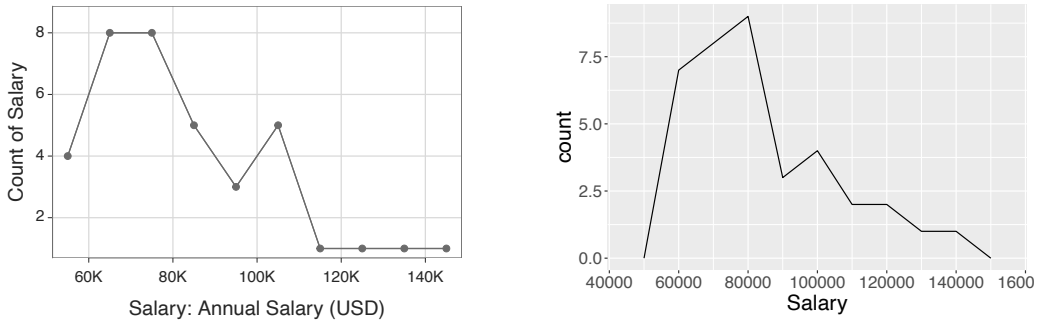
▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it from which to construct the histogram, *Salary*.

▷ Function `geom_plot()`: Plot the histogram as an additional layer to the previously derived visualization saved in *p*.
- Parameter `binwidth`: Specify the width of the histogram bins.

frequency polygon: Scatterplot of binned data of each bin midpoint with its corresponding frequency, plus the line segments that connect adjacent points.

5.1.5 Frequency Polygon

The primary distinction between the frequency polygons plotted in Figure 5.9 is the starting value of the bins. With the same starting value, the shape of the frequency polygons would be the same. The *frequency polygon* plots the bin midpoint of the binned data paired with its corresponding frequency, plus the connecting line segments, as shown in Figure 5.9. The bars of a histogram are replaced with the midpoints of the top of each bar and the connecting line segments.



(a) lessR **(b)** ggplot2

Figure 5.9: Default frequency polygons.

R Input *Frequency polygon*

data: `d <- Read("Employee")`

lessR: `X(Salary, type="freq_poly")`

ggplot2: `ggplot(d, aes(Salary)) + geom_freqpoly(binwidth=10000)`

faceted plot, Section 2.5.2, p. 49

`lessR Frequency polygon`

- ▷ Function `X()`: Plot the points where each point represents the midpoint of the top of each bar in the corresponding histogram.
  - Parameter `stat_x`: To indicate to plot the summary table of aggregated data in place of the original data of measurements, set the parameter to the value `"count"`. Then, connect adjacent points with line segments.
  - *Optional* parameter `size`: Specify a value of zero to remove the plotted points entirely, leaving only the line segments.

`ggplot2 Frequency polygon`

- ▷ Function `X()`: Plot a univariate distribution number continuous variable such as a frequency polygon.
  - Parameter `type`: Specify `"freq_poly"`.
  - Parameter `binwidth`: Specify the width of the bins.

The frequency polygon compared to the histogram can perhaps more clearly delineate the curve that represents the shape of the underlying distribution, particularly when comparing distributions across levels of a categorical variable with facets. Still, the

histogram and frequency polygon base the estimation of the likely smooth distribution on the approximation of bins. Perhaps best to provide the estimated smooth curve directly.

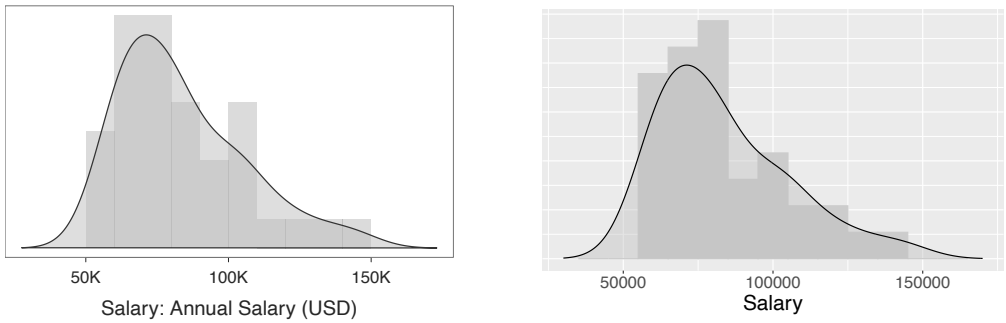
5.2 From Bins to Continuity

density curve: A smooth curve that visualizes the distribution of a numeric variable by showing where its values are more or less concentrated.

Histograms and frequency polygons are notable 19th century innovations. More recently developed algorithms dependent on computer technology (Silverman, 1986; Sheather & Jones, 1991) directly estimate the smooth curve, the *density curve*, that likely underlies the distribution of data sampled for a continuous variable. As with the histogram, the smooth density curve indicates where the values of the variable tend to occur more or less frequently than the other values but without the need to group data values into bins.

5.2.1 Density Curve with Histogram

General density curve. Figure 5.10 illustrates an enhanced, general density plot for `lessR` and `ggplot2`.



(a) lessR **(b)** ggplot2

Figure 5.10: General density plots with histogram.

R Input *Density plots with histogram*

data: `d <- Read("Employee")`

lessR: `X(Salary, type="density")`

ggplot2:
```
ggplot(d, aes(Salary)) + xlim(30000,170000) +
  geom_plot(binwidth=10000, aes(y=stat(density)),
    color="gray80", fill="gray80") +
  geom_density(alpha=.4, fill="gray") +
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```

`lessR Density curve with histogram`

- Function `X()`: Plot the histogram, specifying the variable, *Salary*.
  - Parameter `type`: Set to `"density"` plot the corresponding density curve superimposed on the underlying histogram.
  - *Optional* parameter `by`: Plot a tend to the curve for each value of this categorical variable.
  - *Optional* parameter `bin_width`: Specify the width of the histogram bins.
  - *Optional* parameter `kind`: The default value is `"general"`. Other possible values are `"normal"` to estimate the best fitting normal curve or `"both"` to estimate both curves.
  - *Optional* parameter `bandwidth`: Specify the bandwidth for the density estimation. See the following Figure 5.11 and accompanying explanation.

`ggplot2 Density curve with histogram`

- Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it from which to construct the histogram, *Salary*.
- *Optional* function `xlim()`: By default, the density curve conservatively only ranges from the smallest data value to the largest data value. To extend the density curve to the x-axis in both directions, specify the beginning value and ending value of the density curve accordingly.
- Function `geom_plot()`: Plot the histogram.
  - Parameter `y` within `aes()`: Display the cumulative counts by invoking the Base R function `cumsum()` from the output of the `stat()` function evaluating the `count`. Because the counts are not from the data, but rather computed by `ggplot2`, reference them as the y variable with the `stat()` function.
  - Parameter `binwidth`: Specify the width of the histogram bins.
  - *Optional* parameters `color` and `fill`: Specify the color of the density curve and the color of the area underneath the curve, respectively.
- Function `geom_density()`: Plot the density curve.
  - Parameter `alpha`: Impose some transparency on the filled density curve so that the underlying histogram remains visible. The value can range from 0 to 1 with 0 representing complete transparency.
- *Optional* function `theme()`: With parameters `axis.title.y`, `axis.text.y`, and `axis.ticks.y`, remove the y-axis title, axis values, and tick marks, respectively.

As with most of `lessR` visualizations, a statistical analysis accompanies the visualization. The statistics for the density curve include the sample size, number of missing values, the density bandwidth for potential adjustment in later analyses, the Shapiro-Wilk normality test, and a box plot analysis of outliers described more fully in a following section. Listing 5.1 presents the analysis that accompanies Figure 5.10.

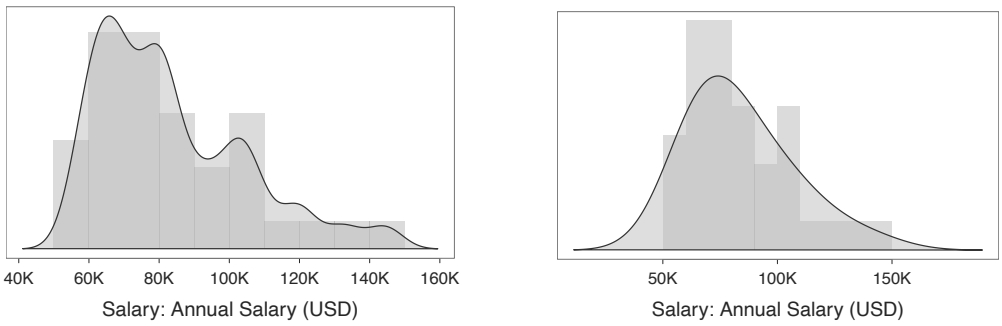
bandwidth: Determines the amount of smoothness of an estimated density plot.

Bandwidth. A key parameter in the estimation of a density curve is *bandwidth*, the value that determines its smoothness. As such, bandwidth is sometimes referred to as the *smoothing parameter*. A lower value yields more detail and a higher value yields more smoothing. If bandwidth is too low, there is too much noise so that the plot overfits the data, yielding an under-smoothed density curve. If bandwidth is too high, meaningful structure is lost so that the plot becomes too flat, over-smoothed.

```
--- Bandwidth ---      for general curve: 9529.0447

--- Salary ---

      n   miss         mean           sd          min          mdn          max
     37      0    83795.557    21799.533    56124.970    79547.600    144419.230


--- Outliers ---     from the box plot: 1

Small      Large
-----      -----
            144419.2
```

Listing 5.1: `Density()` statistical analysis.

Just as bin width may need to be adjusted from its initial default value when constructing a histogram, bandwidth may need to be adjusted from its initial value when estimating a density curve. As seen from Listing 5.1, the initial bandwidth provided by the R function `density()` is 9529, the bandwidth that results in Figure 5.10. Compare this result to the under-smoothed density curve in Figure 5.11a, with bandwidth set at 5000. And compare to the over-smoothed density curve in Figure 5.11b with bandwidth set at 15000.



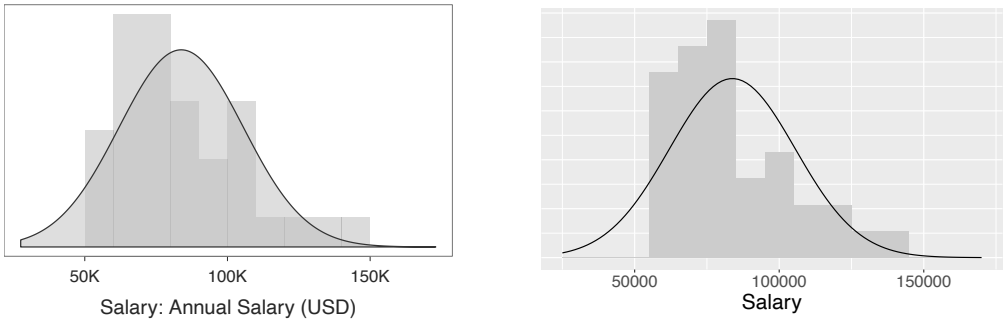
(a) Undersmoothed: Bandwidth set at 5000 **(b)** Oversmoothed: Bandwidth set at 15,000

Figure 5.11: Density plots with bandwidth set too low and set too high (from setting the `X()` parameter `bandwidth`).

The initial density estimate is usually reasonably appropriate, just as the initial bin width estimate for a histogram is usually reasonably appropriate. In general, however, minor adjustments may optimize the visualization. Experience has shown that if some minor adjustment is needed, it is usually to increase the bin width for a histogram or increase the band width for a density curve to show slightly less detail than provided by the default values. The problem with increasing the density curve bandwidth too much is that the curve plots increasingly lower as the bandwidth is increased, shown by comparing Figure 5.11b with Figure 5.11a or Figure 5.10.

Normal density curve. The density curves in Figure 5.12 are general density curves designed to estimate the shape of the underlying continuous distribution, whatever

that shape may be. We can also estimate a normal density function to obtain the best fitting normal curve that fits the data.



(a) lessR **(b)** ggplot2

Figure 5.12: Normal density plots with histogram.

R Input *Normal density curve with histogram*

data: `d <- Read("Employee")`

lessR: `X(Salary, type="density", kind="normal")`

ggplot2:
```
ggplot(d, aes(Salary)) + xlim(25000,170000) +
  geom_plot(binwidth=10000, aes(y=stat(density)),
                  fill="gray80") +
  stat_function(fun=dnorm, color="black",
        args = list(
          mean = mean(d$Salary, na.rm=TRUE),
          sd = sd(d$Salary, na.rm=TRUE) )
        ) +
   theme(axis.title.y=element_blank(),
         axis.text.y=element_blank(),
         axis.ticks.y=element_blank())
```

`lessR Normal density curve with histogram`

▷ Function `X()`: Create the histogram with normal density plot.
- Parameter `type`: Set to `"density"`.
- Parameter `kind`: Set to `"normal"` for the normal density or `"both"` for both the general and normal density estimates.

`ggplot2 Normal density curve with histogram`

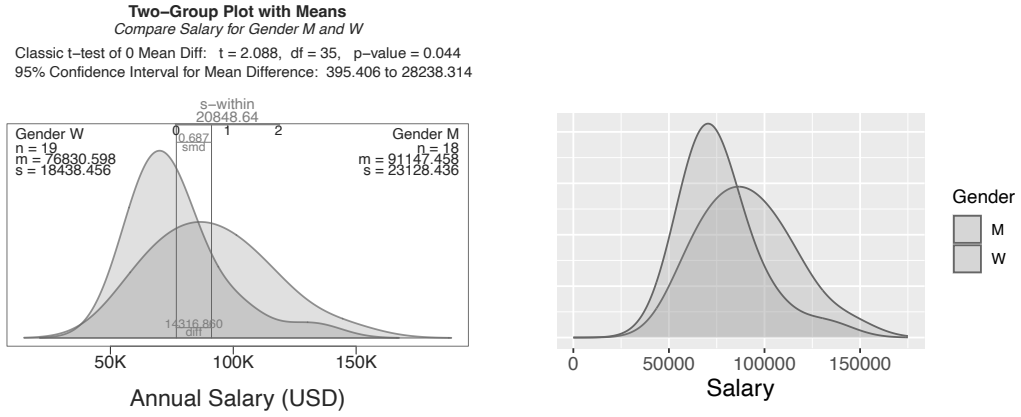
▷ Functions `ggplot()`, `aes()`, and `geom_histogram`: Same as in the previous example.

▷ Function `stat_function()`: There is no `ggplot2` normal curve `geom`, so use this function.
- Parameter `fun`: The function for which to compute densities, the Base R function `dnorm()`.
  - Parameter `args`: Obtain needed information to calculate the normal densities, the mean and standard deviation of the data.

5.2.2 Overlapping Density Curves

A common statistical analysis compares the population means of a continuous variable across two different groups. Is there a difference in average *Salary* for Men and Women at a given company that generalizes beyond the difference observed in the sample? The classic answer follows from the t-test for the mean difference, which provides both a p-value regarding the null hypothesis of no difference, as well as the corresponding confidence interval of the mean difference.

Visualize the overlapping density curves of *Salary* for Men and Women in Figure 5.13. As shown in Figure 5.13a for the `lessR` analysis, if there are two groups that define two density curves, obtain the descriptive and inferential analysis of the independent groups t-test of the mean difference as well as the corresponding visualization of the two density curves. Also provided is a visualization of the sample mean difference expressed in the units of analysis, here USD, and as the standardized effect size (*smd*), Cohen's d (J. Cohen, 1988).



(a) lessR **(b)** ggplot2

Figure 5.13: Overlapping density curves.

R Input *Overlapping density curves*

data: `d <- Read("Employee")`

lessR: `X(Salary, by=Gender, type="density")`

ggplot2:
```
ggplot(d, aes(Salary, fill=Gender)) + xlim(0, 175000) +
  geom_density(alpha=0.25, color="gray50", bw=14000) +
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()) +
  scale_fill_manual(values=c("gray60", "gray60"))
```

`lessR Overlapping density curves`

▷ Function `X()`: Plot the distribution of a continuous variable, x=Salary.
- Parameter `by`: Grouping variable, Gender, to paritition the data into groups, here two groups, M and W.

▷ Alternate Function `ttest(Salary ~ Gender)`: Call the *t*-test function directly if there are exactly two groups.

`ggplot2 Overlapping density curves`

▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it, *Salary*.
- Parameter `fill`: Specify the categorical variable for which a density curve will be plotted for each level of that variable, *Gender*. `ggplot2` creates as many density curves as levels of this variable.

▷ Function `geom_density()`: Plot the density functions.
- Parameter `bw`: The bandwidth of the density curves, 14000. The default bandwidth is so low that the density curves had a wavy appearance, which likely reflects random noise. Obtain this value with trial-and-error.

▷ *Optional* function `xlim()`: Specify the range of the values plotted over the x-axis. By default, `geom_density()` only plots over the range of obtained data, which often leaves the density curves truncated. Not extrapolating is the more conservative approach as the form of the data beyond the range of observation is speculative, but often a reasonable presumption.

▷ *Optional* function `scale_fill_manual()`: Specify the fill color of each density curve, here the same color of `"gray60"`.

A descriptive and influential statistical analysis accompanies the `ttest()` visualization. The first presentation is the descriptive statistics, shown in Listing 5.2.

```
Response Variable:  Salary, Annual Salary (USD)
Grouping Variable:  Gender, Male or Female

------ Description ------

Salary for Gender M:  n.miss = 0,  n = 18,  mean = 71147.458,  sd = 23128.436
Salary for Gender F:  n.miss = 0,  n = 19,  mean = 56830.598,  sd = 18438.456

Sample Mean Difference of Salary:  14316.860

Within-group Standard Deviation:   20848.636
```

Listing 5.2: `ttest()` descriptive statistics.

The next section of statistical output evaluates the assumptions of the *t*-test, shown in Listing 5.3.

The inferential tests are conducted with and without the assumption of homogeneity of variance. The analysis with the assumption of homogeneity appears in Listing 5.4.

Also provided are the effect size, and the bandwidth for each density curve that is adjusted with the `bw` parameter. There is also a version of the `ttest()` named `tt_brief()`, which provides the same visualization, but only the basic statistical

```
------ Assumptions ------

Note: These hypothesis tests can perform poorly, and the
      t-test is typically robust to violations of assumptions.
      Use as heuristic guides instead of interpreting literally.

Null hypothesis, for each group, is a normal distribution of Salary.
Group M  Shapiro-Wilk normality test:  W = 0.962,  p-value = 0.647
Group F  Shapiro-Wilk normality test:  W = 0.828,  p-value = 0.003

Null hypothesis is equal variances of Salary, i.e., homogeneous.
Variance Ratio test:  F = 534924536.348/339976675.129 = 1.573,
   df = 17;18,  p-value = 0.349
Levene's test, Brown-Forsythe:  t = 1.302,  df = 35,  p-value = 0.201
```

Listing 5.3: `ttest()` evaluation of assumptions.

```
------ Inference ------

--- Assume equal population variances of Salary for each Gender

t-cutoff: tcut =  2.030
Standard Error of Mean Difference: SE =  6857.494

Hypothesis Test of 0 Mean Diff:  t = 2.088,  df = 35,  p-value = 0.044

Margin of Error for 95% Confidence Level:  13921.454
```

Listing 5.4: `ttest()` inferential analysis, assumption of homogeneity of variance.

output of the summary statistics, hypothesis test and confidence interval of the mean difference.

5.2.3 Rug Plot

Another version of a density plot is a *rug plot*, a scatterplot for a single variable that plots each data value as a tick mark instead of a dot. Typically display the rug along the variable axis of a density plot, as in Figure 5.14.

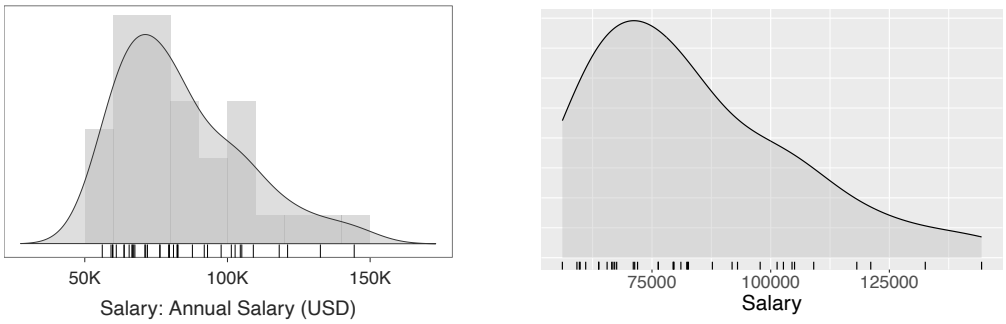
rug plot: A type of scatterplot in which a small tick mark represents each individual data value.

R Input *Density plots with rug*

data: `d <- Read("Employee")`

lessR:
```
X(Salary, type="density", rug=TRUE)
```

ggplot2:
```
ggplot(d, aes(Salary)) +
         geom_density(alpha=.4, fill="gray") + geom_rug()
         theme(axis.title.y=element_blank(),
               axis.text.y=element_blank(),
               axis.ticks.y=element_blank())
```



(a) lessR **(b)** ggplot2

Figure 5.14: Density plots with rugs.

`lessR Rug plot`

- ▷ Function `X()`: Visualize a Univar distribution such as with the histogram and density curve with the rug plot.
  - Parameter `type`: Set to `"density"`.
  - Parameter `rug`: Set to `TRUE` to add the rug plot beneath the density curve.
  - *Optional* parameter `color_rug`: Change the color of the rug tick marks.
  - *Optional* parameter `size_rug`: Change the width of the rug tick marks, depending and the number of data values.

`ggplot2 Rug plot`

- ▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it, *Salary*.
- ▷ Function `geom_density()`: Create the density curve for *Salary*.
- ▷ Function `geom_rug`: Create the rug plot.
  - *Optional* parameter `color`: Change the color of the rug tick marks.
  - *Optional* parameter `size`: Change the width of the rug tick marks, depending and the number of data values.

violin plot: Mirrored density plot, usually longer than wide (or reverse).

5.2.4 Violin Plot

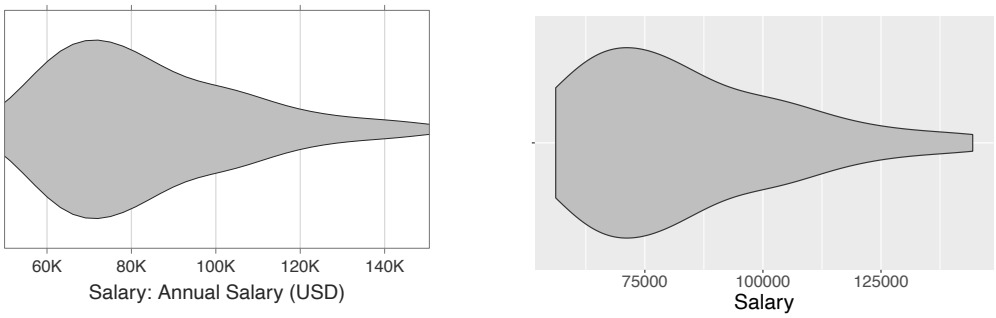
The density plot displays density with the vertical height from the origin. The related *violin plot* rotates the density plot to the mirror image of the densities, symmetric about its axis. See Figure 5.15 for an example. The violin plot displays density according to the width of the mirrored densities, which more emphasizes changes in density along the continuum than a single density plot by itself.

R Input *Violin plots*

data:
```
d <- Read("Employee")
```

lessR:
```
X(Salary, type="violin")
```

ggplot2:
```
ggplot(d, aes(Salary)) +
         geom_violin(fill="gray75") + coord_flip()
```



(a) lessR **(b)** ggplot2

Figure 5.15: Violin plots.

```
        theme(axis.title.y=element_blank())
```

lessR Violin plot

- ▷ Function `X()`: Visualize the distribution of a continuous variable such as a violin plot.
  - Parameter `x`: Specify the continuous variable, *Salary*.

ggplot2 Violin plot

- ▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it, *Salary*.
- ▷ Function `geom_violin()`: Create the violin plot.
  - *Optional* parameter `bw`: The smoothing parameter. The bandwidth of the density curve and its mirror image used to construct the violin plot. A lower value yields more detail and a higher value yields more smoothing.
  - *Optional* parameter `fill`: Interior color of the violin plot, `"gray75"`.
  - *Optional* parameter `alpha`: Impose some transparency on the filled density curve. The value can range from 0 to 1 with 0 representing complete transparency.
- ▷ *Optional* function `coord_flip()`: The default displays the violin plot vertically. Flip the axes for a horizontal orientation.
- ▷ *Optional* function `theme()`): Remove the *y*-axis label by setting the parameter `axis.title.y` to the function call `element_blank()`.

5.3 More Non-Density Visualizations

IQR: The interquartile range, the middle 50% of the values of the sorted distribution, the difference between the 3rd and 1st quartiles.

quartile: The three quartiles divide a sorted distribution into quarters.

box plot: A box around the 1st and 3rd quartiles, a line for the median, and lines (whiskers) out from the box to include all values within 1.5 IQR of each edge.

5.3.1 Classic Box Plot

The boxplot (Tukey, 1977) follows from the *interquartile range (IQR)*. Sort the values from smallest to largest. The *quartiles* are the three values that divide the sorted distribution of values into quarters, four equal-size groups. The first one-quarter of the values are less than the first quartile. The second quartile is the median, the value that occupies the middle position between the smallest and largest values in the sorted distribution. The third quartile separates the largest 25% of the values from the smaller values. The IQR is the range of values between the 3rd and 1st quartiles.

The box's length is the long edge of the box defined by the 1st and 3rd quartiles, as shown in Figure 5.16. The width of the box is irrelevant. This is the same box plot from the full VBS plot shown in Figure 2.5. The box plot bins the distribution into quartiles. The distribution in Figure 5.16 skews to the right, indicated by the distance from the median to the right-and edge of the box.

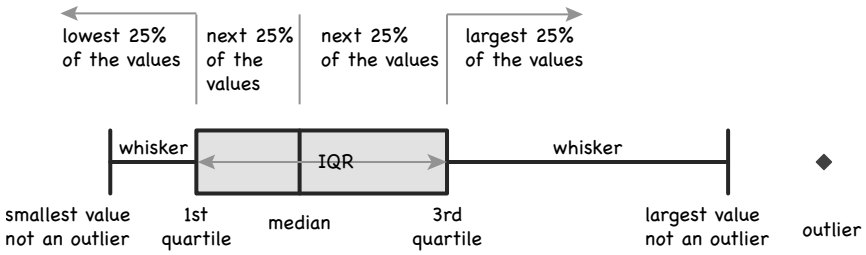


Figure 5.16: Box Plot.

outlier: A data value located far from most of the data values.

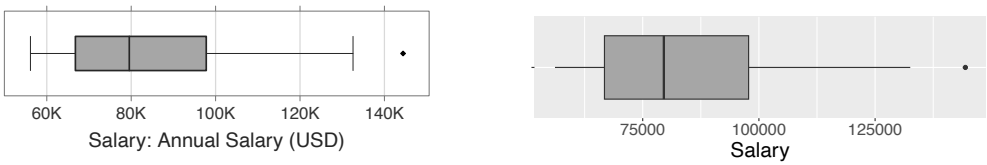
A primary use of the box plot identifies potential *outliers*, necessary to understand the properties of a distribution (Belsley, Kuh, & Welsch, 1980; Osborne & Overbay, 2004). Figure 5.16 shows an outlier as the largest value in the distribution as outliers are typically plotted as individual points. Label values far from the edges of the box as outliers. For a symmetric distribution, define a potential outlier as a data value that lies 1.5 IQR's beyond the 1st or 3rd quartiles. Define an actual outlier as a value that lies 3.0 IQR's or more beyond these quartiles.

When an outlier is identified, the analyst should understand the process that generated the anomalous value. An outlier could indicate a data collection or transcription error. Or, an outlier could be a data value sampled from a population distinct from the population that generated the remaining data values. A data value sampled from a different population would bias the analysis regarding generalizations to the intended population of interest.

The box plots for the *Salary* data are shown in Figure 5.17.

R Input *Box plot of* Salary *from* d *data frame*

data: `d <- Read("Employee")`



(a) lessR **(b)** ggplot2

Figure 5.17: Box plot.

lessR: `X(Salary, type="box")`

ggplot2:
```
ggplot(d, aes(x="", y=Salary)) +
  geom_boxplot(outlier.color="black", fill="gray75") +
  coord_flip()
```

lessR Boxplot

- Function `X()`: Visualize the distribution of a continuous variable, such as with the box plot. Plot points as dark red for potential outliers and then a brighter red for actual outliers, unless grayscale is selected. Figure 5.17 identifies the largest salary as an outlier.
  - Parameter `x`: Specify the continuous variable, *Salary*.
  - Parameter `type`: Set to `"box"`.

ggplot2 Boxplot

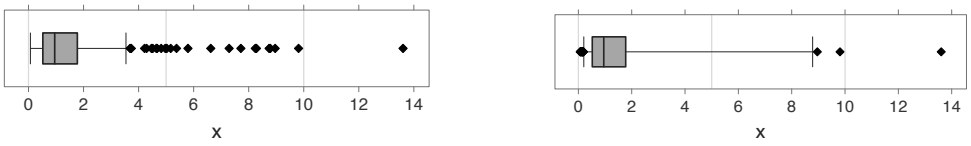
- Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it, *Salary*.
- Function `geom_boxplot()`: Create the box plot.
  - *Optional* parameter `outlier.color`: Specify the color of the outlier points.
  - *Optional* parameter `fill`: Interior color of the violin plot, `"gray75"`.
- *Optional* function `coord_flip()`: The default displays the box plot vertically. Flip the axes for a horizontal orientation.

5.3.2 Box Plot Adjusted for Asymmetry

The IQR cutoffs of 1.5 and 3.0 more appropriately apply to symmetric distributions (Vandervieren & Hubert, 2004; Hubert & Vandervieren, 2008). Compared to a symmetric distribution, more values in the skewed tail of an asymmetric distribution are expected, perhaps even extreme values. Values in the opposite tail are expected to vary less compared to a symmetric distribution.

To illustrate, Figure 5.18a applies the classic boxplot definition of outliers to 250 randomly sampled values, all from the same asymmetric distribution, the standardized log-normal output from the Base R `rlnorm()` function. The distribution is severely right-tail skewed, indicated by the 22 values classified as outliers according to the rule of more than 1.5 IQR's larger than the 3rd quartile. These values are outliers in

the sense that they are far from most other values, yet they are expected given the characteristics of the underlying distribution.



(a) classic boxplot **(b)** adjusted boxplot

Figure 5.18: Box plot and adjusted box plot for a right-tail skewed distribution.

To compensate, Hubert and Vandervieren (2008) generalize the classic 1.5 IQR's definition to also consider the distribution's skewness. Introduce two new parameters, `a` and `b`, whose values adjust the amount of change of the whiskers on both sides of the box. With non-zero values of `a` and `b`, the whiskers on both sides of the box plot are shifted toward the direction of skew. Without skew, the adjusted formula simplifies to the classic definition of an outlier. As shown in Figure 5.18b, for the adjusted box plot the right whisker becomes larger, reducing the number of detected outliers.

The R package `robustbase` (Maechler et al., 2024) provides for this box plot adjustment. The `lessR` function `X()` can implicitly call functions from `robustbase`. Set the `X()` parameter `box_adj` to `TRUE`. Perhaps change the parameters `a` and `b` from their respective default values of -4 and 3 to change the impact of the adjustment of the whiskers on each side of the box.

5.3.3 One-Dimensional Scatterplot

strip plot: A one-dimensional scatterplot, that is, for a single variable.

jitter: Random perturbation of the coordinates of plotted points in a scatterplot.

The scatterplot is a direct visualization of the data values and corresponding sample size. The scatterplot for a single variable is called a *strip plot*. Each data value plots as a point, according to its location along the corresponding value axis. Points corresponding the same value overlap. Plot overlapping points with random perturbations of their coordinates called *jitter* to distinguish between multiple points of the same value, shown in Figure 5.19.

(a) lessR **(b)** ggplot2

Figure 5.19: One-dimensional scatterplot.

R Input *Strip plot of Salary from* d *data frame*

data: `d <- Read("Employee")`

lessR: `X(Salary)`

ggplot2:
```
ggplot(d, aes(x="", y=Salary)) +
  geom_jitter(width=0.02, height=0) + coord_flip() +
  theme(axis.title.y=element_blank())
```

lessR Strip plot

- Function `X()`: Create the 1-D scatterplot, with jitter added automatically if needed. More overlap of plotted points results in more jitter, vertical jitter if possible to retain the value of the plotted point, but horizontal jitter is applied if there are too many points with the same value.
  - Parameter `x`: Specify the continuous variable, *Salary*.
  - Parameter `type`: Set to `"strip"`.
  - *Optional* parameter `jitter_x`: Specify the amount of horizontal jitter.
  - *Optional* parameter `jitter_y`: Specify the amount of vertical jitter.

ggplot2 Strip plot

- Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it, *Salary*.
  - Parameter `x`: Before flipping the axes, there is no x-axis variable, indicated by an empty character string, `""`. Once flipped, this will become the y-axis.
  - Parameter `y`: Specify the continuous variable, *Salary*.
- Function `geom_jitter()`: Create the scatterplot with jittered points.
  - Parameter `height`: Specify the amount of vertical jitter, or horizontal jitter if the coordinate axes are flipped. Here, there is no horizontal jitter.
  - Parameter `width`: Specify the amount of horizontal jitter, or vertical jitter if the coordinate axis are flipped. Here, there is a small amount of vertical jitter added.
- *Optional* function `coord_flip()`: The default displays the strip plot vertically. Flip the axes for a horizontal orientation.

5.4 Integrated Violin/Box/Scatterplot

X(), Chapter 5, p. 111

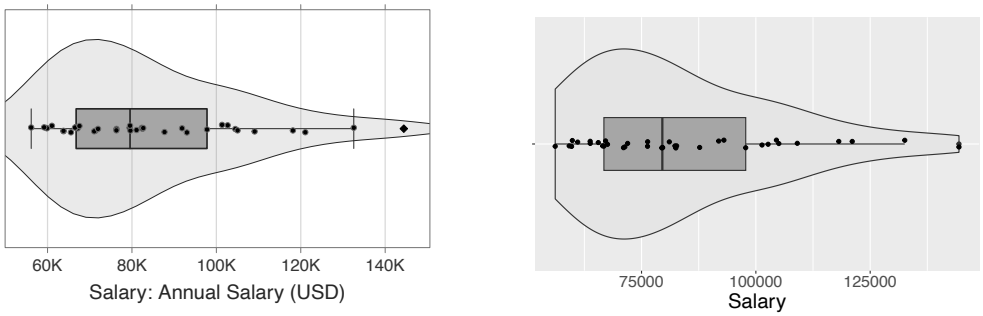
Since the 19th century, the histogram, such as in Figure 2.4, has been the standard visualization to display the frequencies of a continuous variable such as Age, *Salary*, MPG, or Height. The problem is that the histogram follows from a simple technology developed well before the first computer was ever built and the first computer data visualization ever generated. Its uses jagged, discontinuous bars to represent an underlying continuity, does not detect outliers, does not provide a visualization of sample size, and does not easily stack to compare against different groups (Gerbing, 2024). No surprise that modern computer technology provides a means for which to develop more informative displays of the distribution of the values of a continuous variable than the histogram. We can do better with modern tools.

5.4.1 VBS Plot

VBS plot, Chapter 2.5, p. 40

The VBS plot (Gerbing, 2024) integrates three streamlined visualizations of the distribution of a continuous variable: violin, box and scatter plots. The VBS plot of the *Salary* data was shown in Figure 2.5, but is repeated here in Figure 5.20 for continuity and to provide additional detail. Obtain the `lessR` VBS plot of the variable *Salary* with `X(Salary, type="VBS")`.

ggplot2 code, VBS plot, Chapter 5.4.1, p. 134



(a) lessR **(b)** ggplot2

Figure 5.20: Integrated Violin/Box/Scatterplot or VBS Plot.

The *smoothing parameter* relates to the bandwidth of the density curve and its mirror image used to construct the violin plot.

jitter: Random perturbation of the coordinates of each plotted point to lessen over-plotting.

R Input *R Input VBS Plot of* Salary *from the* d *data frame*

data: `d <- Read("Employee")`

lessR: `X(Salary, type="vbs")`

ggplot2:
```
ggplot(d, aes(x=factor(1), y=Salary)) +
       geom_violin(fill="gray90", color="black", alpha=0.35) +
       geom_boxplot(width=0.15, fill="gray70", outlier.shape=NA) +
       geom_jitter(width=0.02, height=0, shape=16, alpha=0.6, size=1.8) +
       coord_flip() + labs(x=NULL, y="Salary")
```

`lessR VBS plot`

The VBS plot demonstrates a primary goal of `lessR`: Conceive of the most beneficial information, and then provide the relevant visualization and statistical analysis with a simple function call given appropriate defaults. Each layer of the plot is implicitly added by the calling function. Further, the function automatically adjusts characteristics of the plot according to the sample size and distribution type, minimizing user trial-and-error to discover the most appropriate parameter values such as the amount of needed jitter and the size of the plotted points.

- ▷ Function `X()`: Create a 1-dimensional scatterplot for a single continuous variable. In this situation, also superimpose a box plot and violin plot, integrated as a VBS plot.
  - Parameter `x`: Create the VBS plot for the specified continuous variable, *Salary*.
  - Parameter `type`: Set to `"vbs"` to simultaneously visualize all three components of the plot.

- *Optional* parameter `bw`: Specify the bandwidth of the density curves for the violin plot. Larger values yield smoother curves.

`X()` also provides the statistical analysis in Listing 5.5 accompanied by the visualization. These statistics include summary statistics including those of the box plot, labeling of the outliers, and the computed parameter values to construct the visualization. Knowing the values of these computed parameters is useful to implement customizations manually.

`ggplot2 VBS plot`

Construct the VBS plot, or any subset of its components, by explicitly adding the violin plot, box plot, and strip plot as separate layers. Define each layer with its own `geom` function, then iteratively adjust the relevant parameters, such as bandwidth, box width, and point size, to obtain the desired display. The construction of each of these three component layers is discussed in the respective sections referenced below.

- ▷ Functions `ggplot()` and `aes()`: See Page 129. In this example, specify `x=factor(1)` to define a one-level categorical variable, which places all observations into a single group on the categorical axis. The specific value inside `factor()` is arbitrary, so `factor(1)`, `factor(999)`, or `factor("All")` would all serve the same purpose.
- ▷ Function `geom_violin()`: See Page 129. Trial different values of the `bw` parameter to obtain the desired smoothness of the density curve. Setting `trim=FALSE` allows the smoothed violin shape to extend beyond the minimum and maximum observed data values instead of being cut off at those points.
- ▷ Function `geom_boxplot()`: See Page 131. Trial different values of the `width` parameter to obtain the desired width of the plotted box plot. Specify `outlier.shape=NA` to suppress the separate plotting of box plot outliers because all data values, including outliers, are already displayed by the strip plot.
- ▷ Function `geom_jitter()`: See Page 133. Trial different values of the `width` parameter to obtain the desired amount of jitter along the categorical axis after flipping the coordinates. If needed, add jitter in the other direction with `height`, though for this application it is usually set to 0.
- ▷ *Optional* function `theme()`: See Page 129. Use to customize axis labels, tick marks, text, and other non-data display features.
- ▷ *Optional* function `coord_flip()`: See Page 129. Flip the coordinate axes to display the graph horizontally rather than vertically.

The trade-off between `lessR` and `ggplot2` is clear. The simpler `lessR` function calls produce pre-defined visualizations, which, for many situations provides the intended visualization, along with a corresponding statistical analysis. For example, with the VBS plot, all three components provide unique, but complementary information, in not much more space than is required for any one of the components. In contrast, although the `ggplot2` construction is more involved with multiple function calls, many `geom` functions are available that provide the analyst with an incredibly versatile tool kit for customizing visualizations.

```
--- Salary ---
Present: 37
Missing: 0
Total  : 37

Mean         : 63795.557
Stnd Dev     : 21799.533
IQR          : 31012.560
Skew         : 0.190   [medcouple, -1 to 1]

Minimum      : 36124.970
Lower Whisker: 36124.970
1st Quartile : 46772.950
Median       : 59547.600
3rd Quartile : 77785.510
Upper Whisker: 112563.380
Maximum      : 124419.230

(Box plot) Outliers: 1

Small      Large
-----      -----
           18 124419.23

Number of duplicated values: 0

Parameter values (can be manually set)
------------------------------------------------------
size: 0.61      size of plotted points
jitter\_y: 0.45  random vertical movement of points
jitter\_x: 0.00  random horizontal movement of points
bw: 9529.04     set bandwidth higher for smoother edges
```

Listing 5.5: VBS statistical analysis.

5.4.2 VBS Plot of Likert data

Mach IV scale, Section 1.2.5, p. 23

Likert scales are typically defined with about four to seven response possibilities such as Strongly Disagree, Disagree, etc. The example here in Figure 5.21 is from the analysis of Machiavellianism with 351 responses (Hunter, Gerbing, & Boster, 1982) to the original Mach IV scale (Christie & Geis, 1970).

VBS plot of Salary, Section 5.20, p. 134

The underlying attitude varies along a continuum of disagreement/agreement, but is only assessed with 6 unique responses from Strongly Disagree (0) to Strongly Agree (5). Figure 5.21 visualizes sample data for item *m06* with a VBS plot.

`X()` detects a large number repeated data values with only 6 unique values distributed over 351 responses. In addition to the previous example that included only vertical jitter, `X()` automatically and appropriately adds horizontal jitter, which moves the point off of its coordinate but fits the scatterplot within the enclosing violin plot.

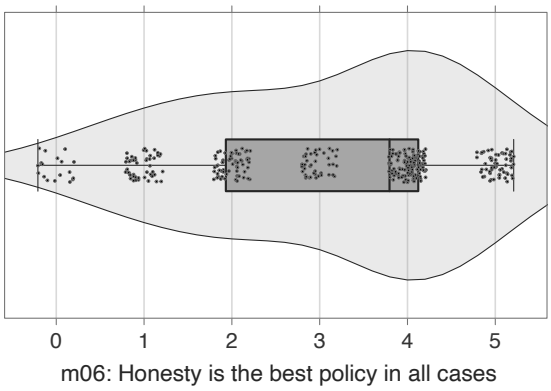


Figure 5.21: VBS plot of 6-pt Likert scale responses to the first item on the Mach IV scale.

From the limited number of unique data values, the default `X()` densities can estimate the underlying continuity. The densities exhibit a monotonically smooth decrease in agreement from the Strongly Agree (5) response, the mode, through the Strongly Disagree (0) response. This later response, along with the responses to Disagree (1), are outliers relative to the entire distribution. In contrast to the histogram, the VBS plot better resolves the dialectic tension of highly discrete measurements that represent an underlying continuity. The histogram would be limited to at most six bins. The VBS plot approximates the continuity, identifies outliers, and provides a visual representation of sample size and responses for each response category. The code to generate this visualization follows.

R Input *VBS plot of Likert data*

data:
```
d <- Read("Mach4")
l <- Read("Mach4_lbl", var_labels=TRUE)
```

lessR:
```
X(m06)
```

Unlike the previous example of the Likert responses to the Mach IV items that converted the integer data values to factors to create the stacked bubble plots, here we analyze the data numerically. The variable labels, the Mach IV items, display on the statistical output to the console and visualizations.

stacked bubble plot (BPFM), Section 3.7, p. 61

lessR Read with variable labels

▷ Function `Read()`: Read data from an external data file with identical syntax across a wide variety of available formats.

- Parameter `from`: The location of the file to be read, enclosed in quotes. Empty quotes, `""`, indicate to browse for the data file on your computer system.
- Parameter `var_labels`: Set to `TRUE` to indicate the file to be read contains the variable labels, two columns with the first column the variable name and the second column the variable label. Read into the data frame named `l`.

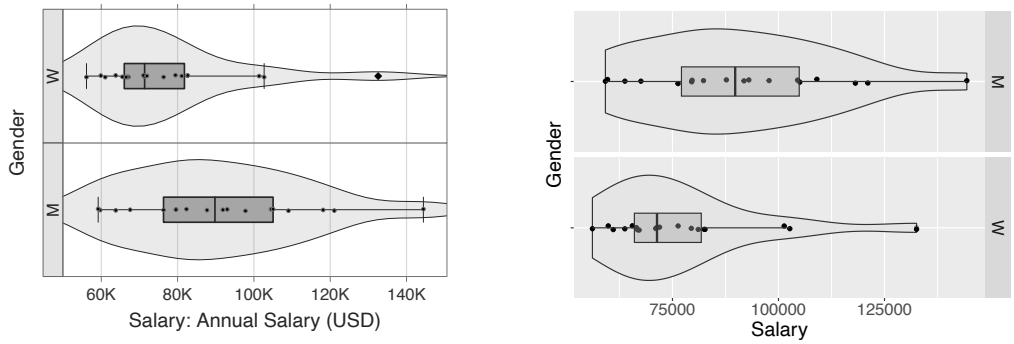
variable labels, Section 1.2.4, p. 19

5.4.3 Faceted VBS Plots

Another analysis of interest is to compare distributions of a continuous variable across the levels of a categorical variable. One possibility creates a VBS plot on a single panel, a space defined by a set of coordinate axes. On that single panel, plot the points with different shapes and/or colors for the different groups defined by the different levels of the categorical variable. Another possibility is faceted visualizations, historically referred to as trellis plots (Cleveland, 1993), that plot the distribution on a different panel, a facet, for each level of the categorical variable.

One facet variable

Following is a simple illustration of faceted graphics: Generate a VBS plot separately for the levels of *Gender* in the data set: Men and Women. Figure 5.22 shows a separate panel for each *Gender*.



(a) lessR **(b)** ggplot2

Figure 5.22: Faceted VBS plot, one panel each for Men and Women.

Interpretation. We conclude that a higher percentage of women occupy the lower salary ranges. The pattern of these visualization encourages further investigation to establish any significant differences in *Salary*, and perhaps a pattern of discrimination.

R Input *VBS facet plot of* Salary *from* d *data frame*
data: `d <- Read("Employee")`

lessR: `X(Salary, facet1=Gender, type="vbs")`

ggplot2: *Construct the plot as from the VBS plot on Page 135, save as object* p.

```
p <- p + facet_grid(cols=vars(Gender)) + labs(x="Gender")
p
```

lessR Faceted VBS chart

▷ Function `X()`: Create the VBS plot from the specified variable, *Salary*.
- Parameter `type`: Set to `"vbs"`.
- Parameter `facet1`: Specify the categorical variable, here *Gender*, that defines the trellises or facets implicitly constructed with functions from the `lattice` (Sarkar, 2008) visualization package. Explicitly set the panel orientation with the number of rows or columns with the `nrow` or `ncol` parameter.

- *Optional* parameter `facet2`: Provide a second categorical variable from which to build a faceted plot, forming a rectangular faceted grid for all the combinations of levels for the two categorical variables.

`ggplot2 Faceted VBS chart`

The approach here is to save a previously constructed VBS plot as an object named p, then add the facet layer to p to yield the facet plot according to *Gender*. `ggplot2` visualizations can be created incrementally.

▷ Object `p`: Save the VBS plot from Page 135 as p. As with any printable R object, display its contents by entering its name at the R command prompt.

▷ Function `facet_grid()`: Create a facet plot.

- Parameter `cols` with function `vars()`: Specify the categorical variable from which to create vertical facets from its levels, *Gender*, which facilitates the comparison of the Salaries for men and women. To stack the panels horizontally, that is, in rows, specify the `rows` parameter instead of `cols`.

Two facet variables and within-panel grouping variable

A VBS visualization of the distribution of a continuous variable can be expanded to multiple grouping variables. In this example, specify one grouping variable, USA or non-USA manufactured cars, to be plotted within each panel. That is, plot the points for the different values of *MPGcity* in different colors for the different levels of *Source*. Include two other grouping variables as facet variables, the number of cylinders in the cars engine and the type of transmission, manual or automatic.

Figure 5.23 illustrates the distribution of continuous variable *MPGcity* across these three grouping variables.

In the original data set the categorical variables are represented numerically. To provide more meaningful labels, convert the numbers to labeled categories. The created factor variable *Source* has levels USA and non-USA. The created Factor variable *Trans* has values Auto and Manual. For consistency, the numeric variable *Cylinders* was also transformed to a factor variable, though here the levels are character representations of the corresponding numeric values.

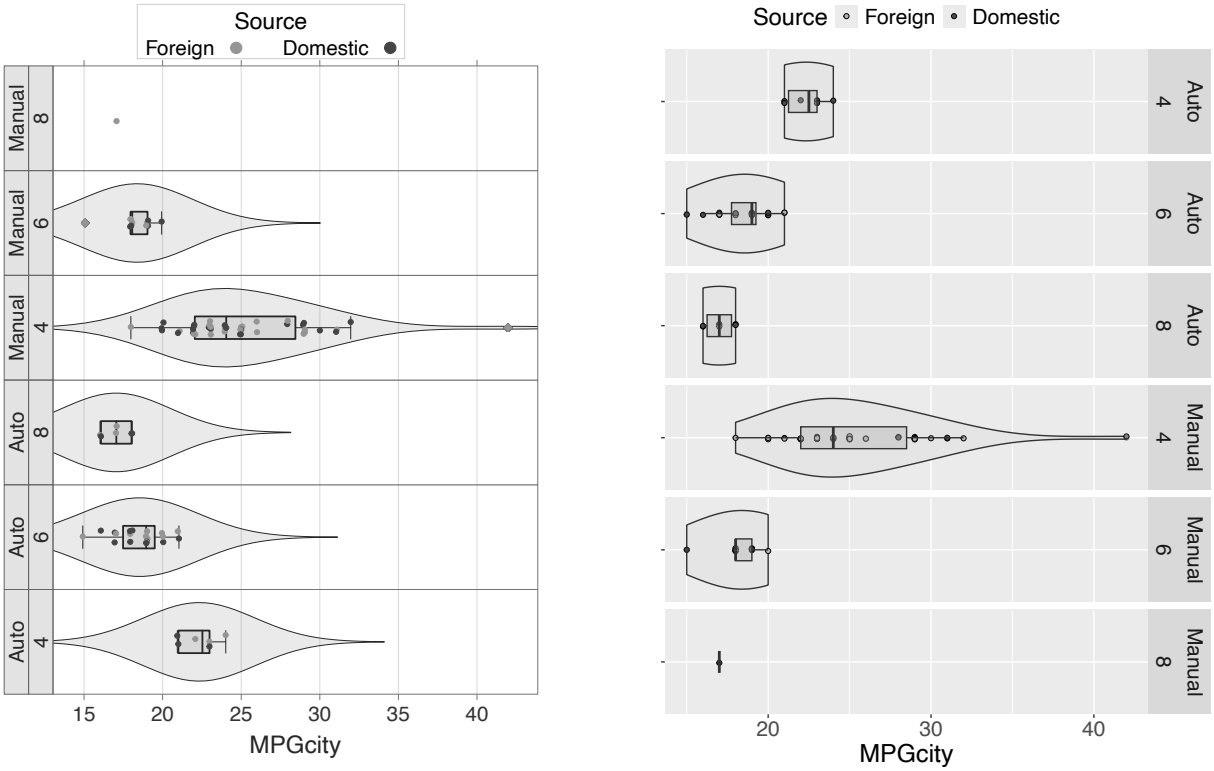
Interpretation. Several conclusions are apparent from examining this multifaceted VBS faceted visualization of 1993 cars.

▷ There is only one manual transmission, eight cylinder car, and it is manufactured outside of the USA.

▷ The most common type of car (in 1993) has a manual transmission and four-cylinders.

▷ The car with the best gas mileage has a manual transmission with a four-cylinder engine, and is an outlier from the rest of the distribution. This one is manufactured outside of the USA.

▷ The cars with the worst gas mileage are automatic transmission, six cylinder cars, which is also the most second most common combination of transmission type and number of cylinders.



(a) lessR **(b)** ggplot2

Figure 5.23: Faceted scatterplots of a car's city miles per gallon across facet variables transmission type and number of engine cylinders and within-panel grouping variable foreign versus domestic manufacturing.

R Input *VBS 1-within panel grouping variable and 2-facet scatterplot*

data:
```
d <- Read("Cars93")
d$Trans <- factor(d$Manual, levels=0:1, labels=c("Auto", "Manual"))
d$Source <- factor(d$Source, levels=0:1, labels=c("Foreign", "USA"))
d$Cylinders <- factor(d$Cylinders, levels=c(4,6,8))
```

lessR:
```
X(MPGcity, by=Source, facet1=Cylinders, facet2=Trans)
```

`lessR VBS plot of 1 continuous variable across 3 categorical variables`

▷ Function `X()`: visualize the distribution of a continuous variable, such as with a `vbs` plot.

- Parameter `x`: Continuous variable to plot, *MPGcity*.
- Parameter `type`: Set to `"vbs"`.
- Parameter `by`: Within-panel categorical grouping variable, *Source*, so that cars from different sources are plotted on the same panel but in different colors.
- Parameter `facet1`: First facet categorical variable, *Cylinders*, defines a different panel for each number of cylinders.
- Parameter `facet2`: Second facet categorical variable, *Trans*, the type of transmission, defines a different panel for each type of transmission.

- *Optional* parameter `n_row`: Specifies how to arrange the panels individualization. The default control is `n_col=1`. Output can also be ordered by the specified number of rows.
- *Optional* parameter `size`: Specify a size of the points larger or smaller than the default size of 0.9.
- *Optional* parameter `n_min_pivot`: Applies to the output pivot table. Specifies the minimum group sample size for the corresponding row to be displayed. The default value is 1 for all rows to be displayed except for those groups that do not contain data. Set to 0 to view all groups or a larger number to exclude rows.

In addition to the list of relevant parameter settings shown in Listing 5.5, `X()` also automatically provides the summary pivot table that corresponds to the plotted VBS Visualizations in Figure 5.23. Listing 5.6 displays the resulting pivot table.

```
----- Pivot table for MPGcity

  Source  Cylinders   Trans   n  na    Mean  Median      SD    IQR  Min  Max
Domestic          4    Auto   6   0  22.333    22.5   1.211   1.75   21   24
 Foreign          6    Auto   5   0  18.600    18.0   1.517   1.00   17   21
Domestic          6    Auto  15   0  18.467    19.0   1.642   2.00   15   21
 Foreign          8    Auto   1   0  17.000    17.0      NA   0.00   17   17
Domestic          8    Auto   5   0  17.000    17.0   1.000   2.00   16   18
 Foreign          4  Manual  27   0  25.222    25.0   4.909   6.50   18   42
Domestic          4  Manual  16   0  25.188    23.5   3.270   5.25   22   31
 Foreign          6  Manual   6   0  18.500    18.0   0.837   0.75   18   20
Domestic          6  Manual   5   0  18.000    19.0   1.732   1.00   15   19
Domestic          8  Manual   1   0  17.000    17.0      NA   0.00   17   17
 Foreign         NA  Manual   6   0  28.667    26.5  12.437  19.75   17   46
```

Listing 5.6: VBS pivot table output of *MPGcity* over the `by`, `facet1`, and `facet2` variables that shows a variety of some statistics for each group.

Interpretation. This pivot table allows us to more precisely summarize the patterns we perceive from the visualization. For example, we see that, on average, four-cylinder, manual transmission cars show little difference in city Miles per gallon between foreign and domestic sources, 25.222 vs. 25.188. We also see that the highest city gas mileage, 46 miles per gallon, is for a foreign, manual transmission car with an unknown number of cylinders, as the value of *Cylinders* is `NA` for not available, that is, missing. Because this value of 46 MPG is from a role of data with missing data, it does not appear in Figure 5.23.

To facilitate comparisons among the levels of each grouping variable from which the visualization is derived, the statistics for the levels of each source separately are also provided in pivot the tables. Listing 5.7 shows these three pivot tables, one each for each of the categorical variables in the analysis: *Source*, *Cylinders*, and *Trans*.

Moreover, the output components of `lessR` functions typically are named and can be retrieved for later use such as generating markdown documents. To access, save the

```
  Source   n  na    Mean  Median     SD  IQR  Min  Max
 Foreign  45   0  23.867      22  6.673    7   17   46
Domestic  48   0  20.958      20  3.994    5   15   31

Cylinders   n  na    Mean  Median      SD    IQR  Min  Max
        4  49   0  24.857    24.0   4.178   6.00   18   42
        6  31   0  18.419    19.0   1.455   1.00   15   21
        8   7   0  17.000    17.0   0.816   1.00   16   18
       NA   6   0  28.667    26.5  12.437  19.75   17   46

 Trans   n  na    Mean  Median     SD   IQR  Min  Max
  Auto  32   0  18.938      19  2.228  3.25   15   24
Manual  61   0  24.164      23  6.025  8.00   15   46
```

Listing 5.7: VBS pivot tables output of *MPGcity* over the `by`, `facet1`, and `facet2` variables for each of the grouping variables in the analysis.

output of the function call to an object, which is automatically saved as an R list. For example, the output pivot table from this `X()` analysis is named `out_pivot`.

```
result <- X(MPGcity, by=Source, facet1=Cylinders, facet2=Trans)
```

Retrieve that pivot table with the name of the list, a `$`, followed by the name of the output component, `result$out_pivot`. Listing that name at the R console or in a markdown document will display the pivot table.

Overview of VBS plots

Our focus is a contemporary replacement for the pre-computer technology histogram. You can plot a single VBS plot for continuous variable or distribute the plots for levels of one or two categorical variables over one or more panels. Following is a summary of these different but related visualizations for `lessR`.

lessR VBS plot overview

- Function `X(X)` for a VBS plot of variable *X*.
- Function `X(X, by=Xcat)` for a VBS plot of different groups of *X* according to the levels of the categorical variable *Xcat* plotted on the same panel.
- Function `X(X, facet1=Xcat)` for a faceted VBS plot of variable *X* with facet variable *XCat*.
- Function `X(X, facet1=Xcat1, facet2=Xcat2, facet1=Xcat)` for a faceted VBS plot of variable *X* with facets defined by a categorical variables *XCat1* and *XCat2*.

5.5 Pareto Chart

At the beginning of the 20th century, the economist Vilfredo Pareto (1906) observed that about 20% of the Italian population owned 80% of the land, an observation that generalizes to a variety of types of wealth across many countries. To generalize further, a key observation in implementing quality control of a manufacturing process is that usually a small number of different types of defects lead to most of the quality issues. Generally, about 20% of defects lead to about 80% of the defective product. Or, about 20% of the reasons for returning merchandise leads to about 80% of the returns. About 80% of sales come from only about 20% of customers, a group of loyal, high-value customers that drive the majority of revenue, so marketing and customer relations should likely differ between these two

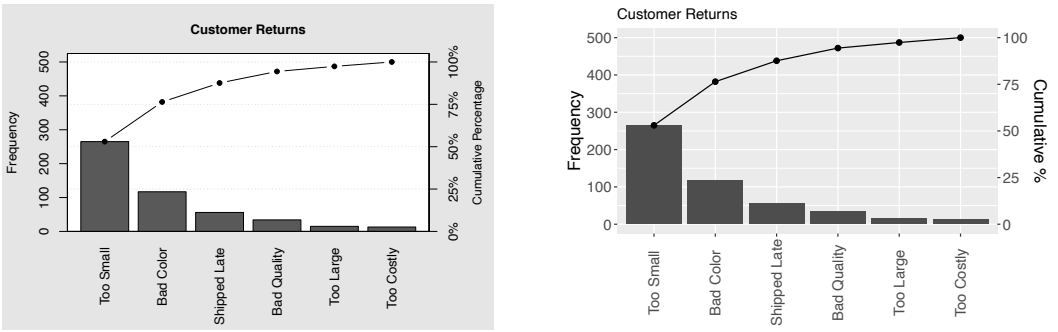
bar chart, Section 3.2.1, p. 55

cumulative distribution, Section 5.1.4, p. 117

The statement that 80% of the effects originate from 20% of the causes is known as the *Pareto principle* (Koch, 1998). The *Pareto chart* consists of two separate visualizations of a categorical variable on the same panel. One plot, with values shown on the left vertical axis, is a bar chart of the counts of the categories of interest. The bars are sorted in descending order. The second plot cumulates the same counts with a frequency polygon.

Pareto principle: Approximately 80% of effects come from 20% of causes.

The Pareto chart, depicted in Figure 5.24, visualizes the influential factors that contribute to the corresponding outcome. The first Pareto chart, in Figure 5.24a, is from the quality control chart package `qcc`. The second version, in Figure 5.24b, results from `ggplot2`.



(a) qcc **(b)** ggplot2

Figure 5.24: The Pareto chart.

Interpretation. Figure 5.24 reveals two primary reasons for the customer returning the purchased clothing: Too Small and Bad Color, which together account for 76.4% of the returns. From this analysis, the retailer understands that either the sizing of the clothing should be adjusted or a notice placed on the website product pages that sizing is a little smaller than many customers expect. Second, the photography on the website should be adjusted to more accurately display colors so that customers receive clothing with the colors that they expect.

The data consists of a column of 500 rows of data, the data value for each row a reason for returning the clothing. Read the data table into the data frame *d*. The

first entry in this table, in Row 1, is *Reason*, the variable name. The Base R function call `table(d$Reason)` yields the counts for each value of *Reason* in in Listing 5.8.

```
> myCount <- table(d$Reason)
> myCount

   Bad Color  Bad Quality Shipped Late   Too Costly    Too Large    Too Small
         117           34           56           13           15          265
```

Listing 5.8: Tabulated data input into the `qcc` Pareto chart function, saved into the object named *myCount* for later reference.

R Input *qcc Pareto chart*

data:
```
d <- Read("http://dgerbing/github.io/data/Pareto.csv")
myCount <- table(d$Reason)
```

qcc:
```
pareto.chart(myCount, main="Customer Returns", col="gray30")
```

stat_pareto(), `ggQC` package: Function to construct a Pareto chart.

ggQC control chart, Section 7.2, p. 196

qcc Pareto chart

- Base R function `table()`: Compute the frequencies, the counts, for each type of return.
- Function `pareto.chart()`: Create the Pareto chart. The frequencies for each category are entered as the data to the function call, so there must be first computed. The function sorts the tabulated values, calculates the cumulative sums, and plots the separate layers, the bar chart and the frequency polygon.
  - Base R parameter `main`: The chart title.
  - Base R parameter `col`: Color of the chart.

The `stat_pareto()` function in the `ggQC` package provides a variety of quality control functions and visualizations. The `ggQC` control chart, which relies upon `ggplot2`, is illustrated in a later chapter. To obtain the Pareto chart directly from `ggplot2` requires more work, but another example of how `ggplot2` has the potential to create a wide variety of visualizations from its basic function calls. The data preparation section is presented first, followed by the instructions for the Pareto chart itself.

The input data frame, *d*, contains a single variable, a categorical variable named *Reason*. As with the `qcc` function `pareto.chart()`, begin the `ggplot2` analysis by tabulating the counts for each of the reasons for returning an item of clothing. However, the data must be further manipulated than just obtaining a table of the accounts. The data must be converted to a data frame with the levels of *Reason* sorted in descending order, with the two new variables to be plotted, *Freq* and *Cml*, the counts and cumulative counts. The data frame of the aggregated data, *a*, shown in Listing 5.9, results from this data wrangling. This is the data entered into the `ggplot()` function.

The data wrangling instructions to obtain the data in Listing 5.9 follow.

```
> a
        Reason Freq Cml
1    Too Small  265 265
2    Bad Color  117 382
3 Shipped Late   56 438
4  Bad Quality   34 472
5    Too Large   15 487
6   Too Costly   13 500
```

Listing 5.9: Aggregated data by tabulation prepared to input into the `ggplot2` Pareto chart function calls.

R Input *Data preparation for `ggplot2` Pareto chart*

data:
```
d <- Read("https://dgerbing.github.io/data/Pareto.csv")
table(d$Reason, dnn="Reason") |> data.frame() |>
arrange(desc(Freq)) |>
mutate(
  Reason = factor(Reason, levels=Reason),
  Cml = cumsum(Freq)
) -> a
```

Data preparation for ggplot2 analysis

- ▷ Base R function `table()`: Compute the summary table by aggregating the data by counts, that is, tabulation. Retain the variable name *Reason* for the levels of the categorical variable. By default, create a new variable, *Freq*, for frequencies or counts.
  - Parameter `x`: The categorical variable from which two obtain the counts, *Reason* in data frame *d*. This function does not have a data parameter, so instead specify the data frame which contains the variable, followed by a `$`, followed by the variable name.
  - Parameter `dnn`: The name of the created count variable, *Reason*. By default, the frequencies are named *Freq*.
- ▷ Base R function `data.frame()`: Transform the table to a data frame.
- ▷ `dplyr` function `arrange()`: Sort the tabulated values in decreasing order. The ydesc() function specifies a descending sort.
- ▷ `dplyr` function `mutate()`: Do two transformations. Preserve this sorted ordered, accomplished by ordering according to the existing ordering, codified as the levels of a factor. Next, compute the cumulated sums with Base R `cumsum()`, stored in the variable *Cml*. The Base R function `transform()` with identical syntax works equally well here.

ordered factor levels, Section 1.2.5, p. 21

To create the `ggplot2` Pareto chart in Figure 5.24b, first consider the component layers of the plot: a bar chart, a cumulative frequency distribution indicated by corresponding points on the plot, and a set of line segments to connect the points of the cumulative frequencies. Each layer is explicitly plotted as a separate `geom`.

R Input *`ggplot2` Pareto chart*

data: *see previous code block for construction of the data source a*

ggplot2:
```
ggplot(a, aes(x=Reason)) +
  geom_bar(aes(y=Freq), stat="identity, fill="gray30"") +
  geom_point(aes(y=Cml)) +
  geom_line(aes(y=Cml, group=1), size=0.5) +
  theme(axis.text.x = element_text(angle=90, vjust=0.5)) +
    scale_y_continuous(name="Frequency",
      sec.axis=sec_axis(~ ./5, name="Cumulative %")) +
  labs(title="Customer Returns", x="", y="Frequency")
```

ggplot2 Pareto chart

Plot these three layers with `geom_bar()` (or `geom_col()`), `geom_point()`, and `geom_line()` for the line that connects the points that plot the cumulative frequencies. Plot the bars with the y-coordinate of variable *Freq*. Plot the cumulative frequency points and corresponding line with the y-coordinates of the variable *Cml*. Include two y-axes.

- ▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *a*, and the variable within it, *Reason*, for the x-axis variable to be applied to each following `geom`.
- ▷ Function `geom_bar()`: Plot the bar chart with the y-axis variable `Freq`.
  - Parameter `stat`: Specify `"identity"` to indicate that the input data are already aggregated.
- ▷ Function `geom_point()`: Plot the points of the cumulative frequency distribution with the y-axis variable `Freq`.
- ▷ Function `geom_line()`: Plot the line segments that connect the points of the cumulative frequency distribution with the y-axis variable `Freq`.
  - Function `aes()`: Specify the y-axis variable *Cml*. The parameter setting `group=1` specifies to treat the points to be connected by the line segments as a single group so as to connect them with a single line.
- ▷ Function `theme()`: Customize the text for the x-axis and y-axis.
  - Parameter `axis.text.x`: Use the function `element_text()` to present the category values perpendicular to the x-axis by specifying `angle=90` with a slight vertical adjustment according to `vjust=0.5`.
  - Function `scale_y_continuous()`: Annotate both y-axes.
    - Parameter `name`: Name the primary, left y-axis.
    - Parameter `sec.axis` with function `sec_axis`: Create and name the secondary y-axis. Create a transformed version of the vertical axis on the left with a parallel axis on the right. The total sample size of the original data is 500, so to compute the cumulative frequencies as percentages, divide the values on the left-axis by 5. The period, `.`, represents the primary y-axis values.
- ▷ Function `labs()`: Label the visualization with the parameter `title`. Remove the label for the x-axis. Label the left y-axis with parameter `y`.