Some Visualizations that Relate Categorical Variables

David W. Gerbing
The School of Business
Portland State University

March 31, 2026

Excerpted from Chapter 4 of the forthcoming 2nd edition:

Data Visualization: Derive Meaning from Data

CRC Press

4.1 Visualize Two Categorical Variables

bar charts of one variable, Section 3.2.1, p. 55

The two-variable bar chart provides the most commonly encountered visualization of the relationship between two categorical variables. As with the one-variable bar chart, a categorical variable x displays its levels on the corresponding axis. Given this axis, the two-categorical variable bar chart visualizes the levels of the second categorical variable to express the relationship between the variables in one of three ways:

- *Stacked* bar chart: Divide each bar into separate rectangles within the bars.
- *Unstacked* bar chart: Split each bar into separate bars.
- *Trellis* or *Faceted* bar chart: Split the bar chart panel into separate panels.

Each of these three forms of a two-categorical variable bar chart express the same relationship, though with different emphasis on various attributes of the visualizations as discussed in the following sections.

4.1.1 The Joint Summary Data Table

summary or *pivot table*, Section 3.1, p. 52

As discussed in the previous chapter, the bar chart is constructed from the summary or pivot table computed by aggregating data across the groups to calculate the numerical y variable. One option computes the summary table separately from the function that plots the bar chart and then entered as the data. Or, have the bar chart function compute the summary table from the original, raw data of measurements.

Employee data set, Section 1.2.1, p. 8

The summary table for the bar chart of two categorical variables lists the value of the numerical y variable for each combination of the levels of the two categorical variables. As with the single categorical variable bar chart, the y variable can be any numerical variable computed from existing data, projected data, or random nonsense. In this example, each value of y is the count of the corresponding group defined by the levels of the two categorical variables, which results in the *joint frequency table* or *cross-tabulation table* shown in Figure 4.1 from the small Employee data table that references two genders.

Gender	ACCT	ADMN	FINC	MKTG	SALE
M	2	2	3	1	10
W	3	4	1	5	5

Figure 4.1: Joint summary table from Employee data.

joint frequency table: A joint summary table of counts.

cross-tabulation table: Table of joint frequencies for all combinations of levels of two or more categorical variables.

Whether computed explicitly as shown below or implicitly by the bar chart function, the summary table of y across all groups provides the needed information from which to construct the bar chart. From Figure 4.1, examine the relationship between the number of employees in each department with gender. The joint summary table shows, for example, that the company employs three women accountants and two men accountants. One group referenced here is defined by women accountants and the other group is defined by men accountants.

R Input *Calculate the cross-tabulation table, save in data frame* a
data: `d <- Read("Employee")`

lessR: `d |> pivot(table, by=c(Dept, Gender) -> a`
dplyr: `d |> group_by(Dept, Gender) |> summarise(n=n()) -> a`

`lessR Data aggregation of counts to obtain the joint frequency table`

▷ Function `pivot()`: Same as for the aggregation of counts for a single categorical variable, Section 3.1.1, Page 53, except that the `by` parameter is now a vector of two categorical variables, here *c(Dept, Gender)*.

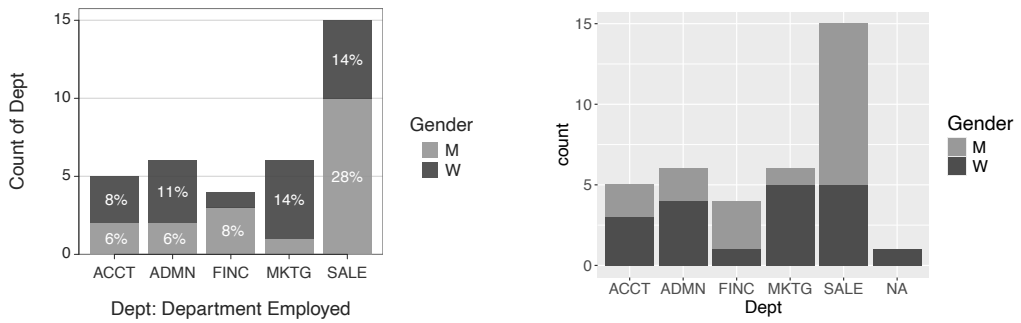
`ggplot2 Data aggregation of counts to obtain the joint frequency table`

▷ `dplyr` functions: Same as for the aggregation of counts for a single categorical variable, Section 3.1.1, Page 53, except that the `group_by` function now references two categorical variables, here *(Dept, Gender)*.

4.1.2 Two-Variable Bar Chart

Stacked Bar Chart

Figure 4.2 shows the default grayscale `lessR` and `ggplot2` bar charts that portray the relation between the number of observations for two categorical variables, *Dept* and *Gender*, from the joint frequency table in Figure 4.1. The bar colors indicate the corresponding proportions of the second categorical variable for each level of the `x` variable. Each bar displays the proportion of M (men) and W (women) for the corresponding department.



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

Figure 4.2: Default two-variable bar chart.

By default, R alphabetizes the presentation of the categories as M before W. To override the default ordering for any categorical variable, convert the variable to a factor variable with the specified custom ordering of the levels.

factor variables, Section 1.2.5, p. 20

R Input *Default two-variable bar chart with implicit aggregation of counts*

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

lessR: `Chart(Dept, by=Gender)`

ggplot2: `ggplot(d, aes(Dept, fill=Gender)) + geom_bar()`

`lessR Stacked two-variable bar chart`

As always, for grayscale, do `style("gray")`, before calling the visualization function.

- ▷ Function `Chart()`: Construct the bar chart, relying upon the default value of parameter `type` set to `"bar"`.
  - Parameter `x`: Specify the first categorical variable, the x-axis variable *Dept*.
  - Parameter `by`: Specify the second categorical variable, *Gender*, of which the proportion of each level is shown within each bar of the x-variable.

`ggplot2 Stacked two-variable bar chart`

- ▷ Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variables within it.
- ▷ Function `geom_bar()`: Create a bar chart from the specified variables.
  - Parameter `x`: Specify the first categorical variable, here *Dept*, which appears on the x-axis.
  - Parameter `fill`: Specify the second categorical variable, here *Gender*, of which the proportion of each level is shown within each bar of the x-variable.
- ▷ *Optional* function `scale_fill_grey()`: Indicate to plot the bars in grayscale, such as with the `start` value of 0.6 and the `end` value of 0.3 illustrated in Figure 4.1. The default first two colors are a variation of orange and then aqua.

Consistent with the `lessR` data visualization functions, `Chart()` provides an accompanying statistical analysis, partially shown in Listing 4.1. The output begins with the variable names and, if present, variable labels. The primary output is the joint frequency table from which the bar chart is constructed, as shown in Figure 4.1. Construct the bar chart from this tabulation of the joint occurrences of the levels of the two variables across all combinations of levels.

```
Dept: Department Employed
  - by levels of -
Gender: Man or Woman

Joint and Marginal Frequencies
------------------------------

       Dept
Gender   ACCT ADMN FINC MKTG SALE Sum
  M         2    2    3    1   10  18
  W         3    4    1    5    5  18
  Sum       5    6    4    6   15  36
```

Listing 4.1: `Chart()` two-way cross-tabulation table.

marginal frequency: Row or column sum from the table of joint frequencies.

In addition to the joint frequencies, `Chart()` provides the frequency distribution of each variable by itself, obtained for each level of one variable summed across the levels of the other variable. Refer to each sum in this context as a *marginal frequency*. These marginal frequencies appear under the Row or Column labeled Sum. Find the

grand total, the sum of all joint frequencies, or, equivalently, the sum of the marginal frequencies for either one of the variables. From Listing 4.1, the grand total equal 36.

As shown in Listing 4.2, `Chart()` also provides two statistics to evaluate the relationship between the variables. *Cramer's V* indicates the magnitude of association between nominal variables, a type of correlation coefficient between two categorical variables, the Pearson chi-square statistic rescaled with values between 0 and 1. Based on that chi-square statistic, the corresponding chi-square statistic tests the null hypothesis of no relationship between the two variables. Here the null hypothesis could not be rejected because the corresponding *p-value* exceeds the traditional $\alpha = 0.05$ threshold.

Cramer's V: Type of correlation for two nominal (categorical) variables that varies from -1 to 1.

p-value: Probability of obtaining a sample result as far from the null value of 0 given the population assumption of 0.

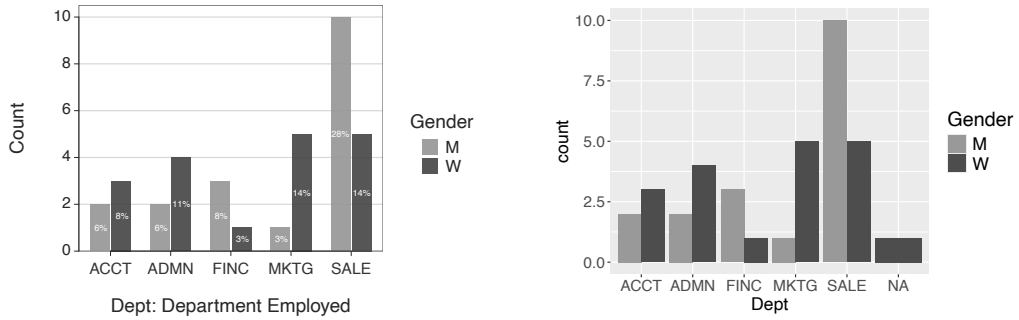
```
Cramer's V: 0.415

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

Listing 4.2: `Chart()` statistical output to evaluate the relationship among two categorical variables.

The default stacked form of the bar chart compares the relative proportion of each level of the second categorical variable within each level of the first categorical variable. Other versions of the two categorical variable bar chart highlight other features.

Unstacked Bar Chart

The unstacked bar chart portrays the levels of the second categorical variable as separate bars adjacent to each other within each level of the first categorical variable. As in Figure 4.3 illustrates, this form directly compares the levels of the second categorical variable to each other at each level of the first variable.



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

Figure 4.3: Unstacked two-variable bar chart.

R Input *Unstacked two-variable bar chart*

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

lessR: `Chart(Dept, by=Gender, beside=TRUE)`

ggplot2: `ggplot(d, aes(Dept, fill=Gender)) + geom_bar(position="dodge")`

lessR Unstacked two-variable bar chart

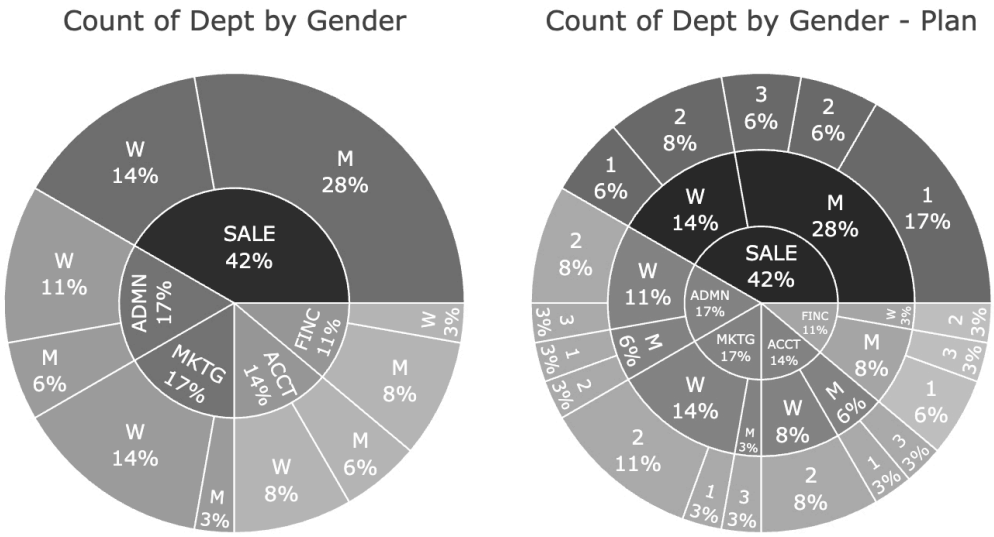
▷ Function `Chart()`: Same construction as for the default stacked bar chart except with the addition of the parameter `beside` set to `TRUE`.

ggplot2 Unstacked two-variable bar chart

▷ Functions `ggplot()` and `aes()`: Same construction as for the default stacked bar chart except with the addition of the `geom_bar()` parameter `position` set to `"dodge"`.

4.1.3 The Sunburst Chart

A pie chart displays the distribution of a single categorical variable, with each slice sized by its count or some other summary statistic. The *sunburst* chart extends this idea to two or more categorical variables by adding concentric rings to the pie chart, one per additional categorical variable. Figure 4.4 shows sunburst charts of two and three levels. The sunburst charge visualizes hierarchical, categories, sub-categories within categories. Sunburst charts stack categorical variables in concentric rings, the inner ring showing the primary variable and each outer ring adding a grouping level.



(a) lessR two-level **(b)** lessR three-level

Figure 4.4: Gray-scale two-level and three-level sunburst charts.

In this example, the first variable, the variable Dept, occupies the inner ring, exactly as in a plain pie chart. Each additional categorical variable adds an outer ring that subdivides its parent sector. Adding `by=Gender` adds a second ring showing the male/female split within each department, and `by=c(Gender, Plan)` adds a third ring further subdividing by health plan. Each outer ring is a finer partition of the one inside it, and the progressive lightening of color from ring to ring visually reinforces that nesting.

In `lessR`, `type="pie"` handles both cases: a single variable produces a standard pie chart, and adding a `by` variable automatically generates the corresponding sunburst. The chart type can also be specified explicitly as `type="sunburst"`, but the analogy with bar charts is instructive. Just as `type="bar"` produces either a simple bar chart or a grouped/stacked chart depending on whether parameter `by=` is present, `type="pie"` scales from one ring to many. The number of rings equals the number of variables: one for `x` alone, two for `x` plus one `by` variable, three for `x` plus two `by` variables, and so on.

R Input *Sunburst charts*

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

lessR: `Chart(Dept, by=Gender, type="pie")`

lessR: `Chart(Dept, by=c(Gender, Plan), type="pie")`

lessR Sunburst charts

- Function `Chart()` for any available chart with previously defined variables for parameter `x` and `by`.
  - Parameter `by`: Set to a single variable, Gender, for a two-ring sunburst chart, or a vector, `c(`Gender, Plan`)`, for a three ring chart.
  - Parameter `type`: Set to `"pie"` or `"sunburst"`.

Plotly automatically lightens the color of each outer ring relative to its parent sector, using the visual channel of lightness to encode hierarchical depth — the deeper the ring, the paler the shade. This works well with qualitative palettes, where each top-level category has a distinct hue, and the progressive lightening of outer rings reads as a natural visual hierarchy. With sequential palettes, such as grayscale, the situation differs. Grayscale already uses lightness to separate categories, so Plotly's additional per-ring lightening further lightens the lighter sectors, pushing them toward white. The result is that some wedges appear somewhat washed out across all rings at once. The grayscale versions shown here reflect that constraint. Create these visualizations without first specifying `style("gray")` to view the more desirable color sunburst chart with each wedge a different color (hue).

4.1.4 Hierarchical Treemap and Icicle Chart

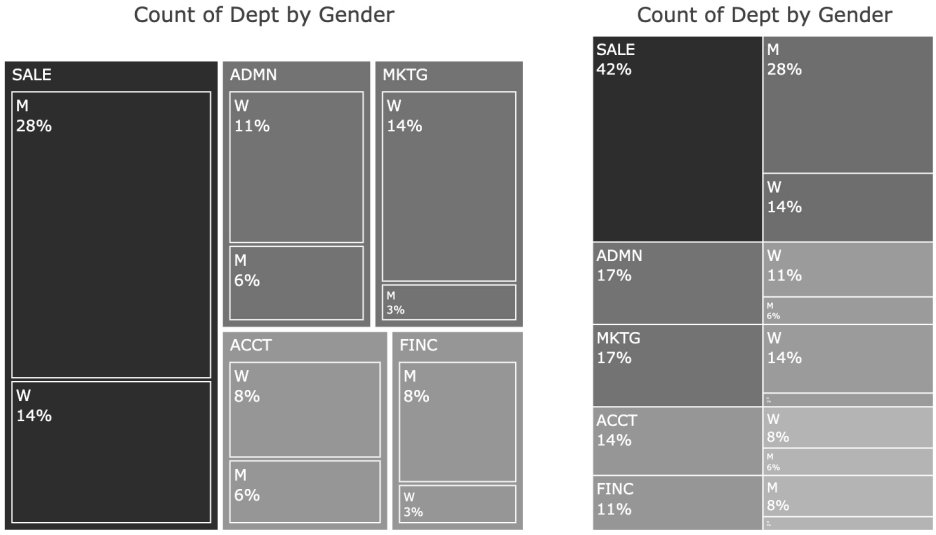
treemap: A potentially hierarchical display of nested rectangles, with each rectangle subdivided into smaller rectangles for its subcategories.

icicle chart: A potentially hierarchical display of rectangular bars arranged in levels, with subcategories shown as adjacent bars extending from their parent level.

A treemap displays categories as rectangles and, if hierarchical, subcategories as nested rectangles. Each rectangle's area represents a quantitative value such as count, revenue, or size. It is designed to show part-to-whole relationships within a hierarchy while using space very efficiently.

An icicle plot is also a hierarchical chart, but it shows the hierarchy as stacked rectangular bands arranged in levels from the root outward. Each band represents a node in the tree, and its width or height reflects its value, depending on orientation. Icicle charts visualize hierarchical data from leaves and/or outer branches toward the root, using rectangles.

Figure 4.5 presents a tree map and icicle chart of counts, each with one level of hierarchy. Both charts are interactive when displayed in the RStudio `Viewer` window, or when saved as a webpage from the `Export` tab in that window.



(a) lessR hierachical tree map **(b)** lessR hierachical icicle chart

Figure 4.5: Gray-scale two-level treemap and icicle charts.

The main distinction is the layout. A treemap fills a rectangular region with nested boxes, so it is compact and space-efficient, but sometimes harder to read the exact hierarchical path because the nesting can become visually dense. An icicle plot lays the hierarchy out in ordered levels, which can make parent-child structure easier to follow. This difference is one of geometry and readability tradeoff, which can sometimes favor treemaps and sometimes favor icicle charts, depending on your preference and aesthetic judgment.

`lessR treemap and icicle charts`

- ▷ Function `Chart()` for any available chart with previously defined variables for parameter `x` and `by`.
  - Parameter `by`: For a flat tree map or icicle chart with no hierarchy, do not specify a `by` variable. For a one level hierarchy, set to a single variable, Gender. For a two-level hierarchy, set `by` to a vector, such as `c(`Gender, Plan`)`.
  - Parameter `type`: Set to `"treemap"` or `"icicle"`.

The two plots are similar because both are potentially hierarchical, both encode magnitude with rectangular area, and both show part-to-whole structure across multiple levels of grouping. Both charts belong to the family of hierarchical charts with sunburst charts.

4.1.5 Faceted Plots of a Categorical Variable

trellis plot, Chapter 2, p. 140

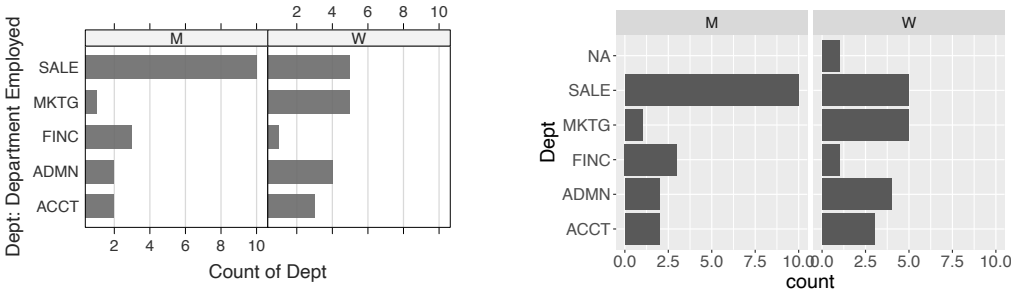
Developed and named by Cleveland (1993), a trellis plot forms a rectangular grid by displaying a visualization on a separate panel for each level of a categorical variable. Cleveland chose the name trellis for the resemblance of these data visualizations to garden trellises, such as shown in Figure 4.6. However, more recently trellis plots are typically referred to as faceted plots (Wickham, 2016). Each individual square in the trellis grid is an individual facet, which becomes a separate visualization according to the levels of one or more categorical variables.

Figure 4.6: Image of a garden trellis generated with DALL · E (OpenAI) via ChatGPT.

Faceted bar chart

factor variable, Chapter 1.2.5, p. 21

Figure 4.7 displays a faceted plot. The bar chart of `Dept` is plotted separately for each level of `Gender`. Because there are only two genders represented in this small data set, the corresponding faceted plot displays two facets, labeled as M and W for men and women. As always, to obtain more descriptive labels than shown here, convert *Gender* from a character variable as read into the data frame to a factor variable, assigning labels such as Men and Women.



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

Figure 4.7: Faceted bar charts (facets) of counts of *Dept* across levels of *Gender*.

R Input *Faceted bar charts of counts*

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

lessR: `Chart(Dept, facet=Gender)`

ggplot2: `ggplot(d, aes(Dept)) + geom_bar() +`

```
        facet_grid(cols=vars(Gender) + coord_flip()
```

lessR Faceted bar chart of counts

- Function `Chart()`: Create the bar chart from the specified variables.
  - Parameter `facet`: Specifies the second categorical variable, here *Gender*, that defines the trellises or facets implicitly constructed with functions from the `lattice` (Sarkar, 2008) visualization package. The default facet orientation is horizontal. Explicitly set the panel orientation with the `nrow` or `ncol` parameter.

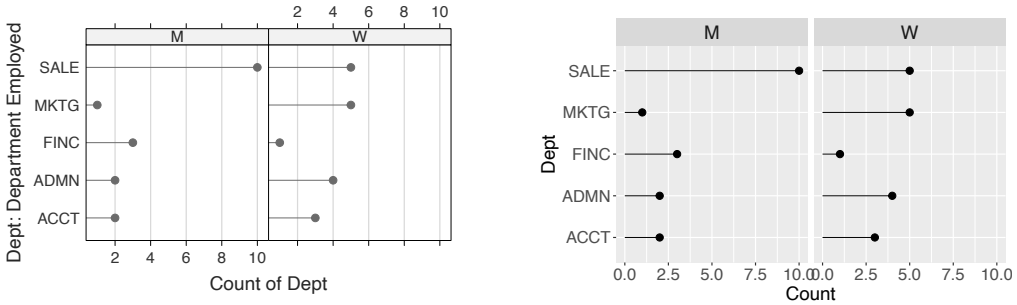
ggplot2 Faceted bar chart

- Functions `ggplot()` and `aes()`: Specify the input data frame, *d*, and the variable within it, *Dept*, which appears on the *x*-axis.
- 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*. To stack the panels horizontally, that is, in rows, specify the `rows` parameter instead of `cols`.
- Function `coord_flip()`: Orient the bars horizontally as a means to provide room to display the axis labels, the names of the categories.
- *Optional* function `facet_grid(XCat1 ~ XCat2)`: Specify two categorical variables to define two facets. The rows of the facet grid are defined by the levels of the first categorical variable, `XCat1`, and the columns are defined by the levels of the second categorical variable, `XCat2`.

Faceted Cleveland dot plot of counts

one-panel dot plot, Section 3.5, p. 58

The Cleveland dot plot generalizes the one-panel visualization in Figure 3.5 to the multi-panel faceted plot, such as shown in Figure 4.8. The second categorical variable in this example is *Gender*. Compare the number count of men and women employed in each department, coded with the values of M and W.



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

Figure 4.8: Faceted Cleveland dot plots of counts for *Dept*.

R Input *Faceted Cleveland dot plot of Dept counts by* Gender

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

lessR: `Chart(Dept, facet=Gender, type="dot")`

ggplot2:
```
d |>
      group_by(Dept, Gender) |> summarise(Count=n()) |> na.omit() |>
  ggplot(aes(Dept, Count)) +
    geom_segment(aes(xend=Dept, y=0, yend=Count), linewidth=0.25) +
    geom_point() +
    facet_grid(cols=vars(Gender)) + coord_flip()
```

lessR Faceted Cleveland dot plot

▷ Function `Chart()`: Create a faceted Cleveland dot plot with the same syntax in the previous bar chart example with `Chart()`. Specify the categorical variable of interest, *Dept*.
- Parameter `facet`: Specify the categorical variable, here *Gender*, from which to create the trellis or facets.

ggplot2 Faceted Cleveland dot plot

By computing the summary table and then passing to `ggplot()` as data, the created variable *Count* is available to all the `ggplot()` layers. The last function in the chain of functions related by the Base R pipe operator, `|>`, is `ggplot()` itself, so do not explicitly identify the input summary data frame in the call to `ggplot()`.

pipe operator, Section 1.2.6, p. 23

▷ `dplyr` functions `group_by()`, `summarise()`, and `n()`: Compute group counts. See Section 3.1.1, Page 53. Choose to remove missing data from the computed summary table, here with the Base R function `na.omit()`.

▷ Function `geom_point()`: Create the scatterplot of variables, *Dept* and *Count*.

▷ Function `geom_segment()`: Draw the connecting line segments from the corresponding axis to each corresponding point according to the starting parameters `x` and `y` and ending parameters `xend` and `yend`.
- *Optional* parameter `size`: Specify the width of the line segments.

▷ Function `facet_grid()`: Create the facet grid.
- Parameter `cols`: Place the facets into a column. Set its value to `vars(Gender)`.

Later we explore the Cleveland dot plot of a general statistic in which the numeric y variable is defined as an aggregation over the specified groups. In this section, the y variable is defined as the count of the number of observations in each group.

bar chart of a general statistic, Section 4.2.4, p. 98

Other faceted charts

Follow the previous `Chart()` function calls but replace parameter `by` with parameter `facet`.

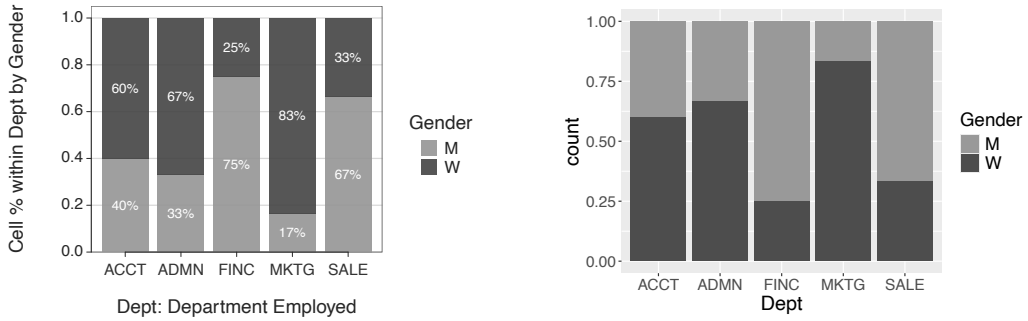
For example:

```
Chart(Dept, facet=Gender, type="sunburst")
Chart(Dept, facet=Gender, type="radar")
```

4.1.6 100% Stacked Bar Chart

100% stacked bar chart: Proportion of each level of the second categorical variable within each level of the `x` variable.

What percentage of men and women are employed within each department? Answer this question with an analysis of the distribution of *Gender within* each level of *Dept*, the 100% stacked bar chart. Each bar's height (length) for this analysis accounts for the full 100%, the full percentage of all elements within each group (category) for variable `x`. Figure 4.9 illustrates the corresponding *100% stacked bar chart*.



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

Figure 4.9: 100% stacked bar chart.

The stacked two-categorical variable bar chart plots two categorical variables: The x-axis variable, plus the variable that determines the shaded areas on each bar. The 100% stacked bar chart is a version of the stacked two-categorical variable bar chart by comparing the percentage of the distribution of the second categorical variable *within* each level of the `x`-variable instead of across all the data.

R Input *100% stacked bar chart*

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

lessR:
```
Chart(Dept, by=Gender, stack100=TRUE)
```

ggplot2:
```
d |> filter(!is.na(Dept), !is.na(Gender)) |>
    ggplot(aes(Dept, fill=Gender)) + geom_bar(position="fill")
```

`lessR 100% stacked bar chart`

- ▷ Function `Chart()`: Create the bar chart.
  - Parameter `x`: Specify the first categorical variable that defines the x-axis, here *Dept*.
  - Parameter `by`: Specify the second categorical variable that determines the fill colors of each bar, here according to the groups defined by *Gender*.
  - Parameter `stack100`: Set to `TRUE` to visualize the 100% stacked bar chart.

`ggplot2 100% stacked bar chart`

- ▷ *Optional* `dplyr` function `filter()`: Choose to subset rows of data by employing the Base R function `na.omit()` to remove any row of data that is missing the data value for *Dept* or *Gender*.
- ▷ Functions `ggplot()` and `aes()`: Specify variables *Dept* on the x-axis and *Gender* for the fill color of the bars.