Data Visualization

Unlike content in the preceding chapters, data visualization is more an *art* than a *science*. Data visualization is heavily dependent on the type of data being presented, key messages that need to be conveyed, the unique characteristics of the audience to whom information needs to be communicated, and various contextual features germane to a proper interpretation and understanding of material.

This chapter will provide general best practices as well as strategies specific to various types of data that will promote success when communicating data to technical and non-technical stakeholders alike. Since there is a considerable amount of code required to construct the plethora of graphics in this chapter, only the data prep code will be included. You can reference the "Appendix" for a curated set of fully reproducible code for each data visualization provided in this chapter.

Best Practices

While the type of visual depends on the nature of data being presented, there are general best practices for effectively communicating information that are applicable for all data types, audiences, and contexts.

Color Palette

There are several important considerations when choosing colors for data visualization.

Color Has Meaning
The meaning of color is not consistent across the world. For example, in Western cultures red generally has unfavorable connotations (e.g., off track, danger), while

C. Starbuck, *The Fundamentals of People Analytics*,
https://doi.org/10.1007/978-3-031-28674-2_15

green signals a favorable status (e.g., on track, cash flow positive). However, in Asian cultures red indicates success.

It is important to understand what color indicates in the contexts in which information is shared to ensure color choices are consistent with the messages that need to be conveyed.

Respect Color Blindness
Most estimates indicate that 1 in 12 men (8%) and 1 in 200 women (0.5%) are colorblind (~4.5% of the world population). Males are more likely to be colorblind because the X chromosome contains the genes responsible for the most common forms of color blindness. Trouble distinguishing between red and green is the most common manifestation of colorblindness (Gordon, 1998).

This has important implications for data visualization, and it is generally best to avoid the use of red and green. One alternative is to leverage orange to indicate unfavorable data points and blue for favorable. This strategy will be implemented throughout this chapter.

Adhere to Brand Colors
In most organizations, the marketing team defines a color palette consistent with the branding used for consumer products and services. In this case, analysts may be constrained to the use of colors within the branding palette.

Be sure to consult with marketing colleagues and adhere to any color palette requirements.

Use Color Consistently
To support a correct interpretation of information, it is important to use a consistent color scheme across the various visuals in dashboards, slides, and documents. For example, if blue is assigned to highlight how the Engineering department compares to other departments in a particular visual, blue should be assigned to the Engineering department in every visual in which department is a grouping dimension. The consistent use of color requires the audience to wield less effort to understand information, which in turn reduces the risk of incorrect interpretation.

Chart Borders

The goal in data visualization is to enable the most important data to take center stage. Formatting should support—not detract—from this objective. Chart borders are a usual suspect and prime example of formatting that can divert focus away from the data.

Figure 1 illustrates the use of heavy and light borders for a tabular presentation of data. Formatting takes center stage in the first table due to heavy borders, while formatting is not the focus in the second table given the minimal light gray border.

Dimension	Metric 1	Metric 2	Metric 3
Dimension A	x_1	y_1	z_1
Dimension B	x_2	y_2	z_2
Dimension C	x_3	y_3	z_3

Dimension	Metric 1	Metric 2	Metric 3
Dimension A	x_1	y_1	z_1
Dimension B	x_2	y_2	z_2
Dimension C	x_3	y_3	z_3

Fig. 1 Comparison of data tables with heavy (left) and light (right) borders

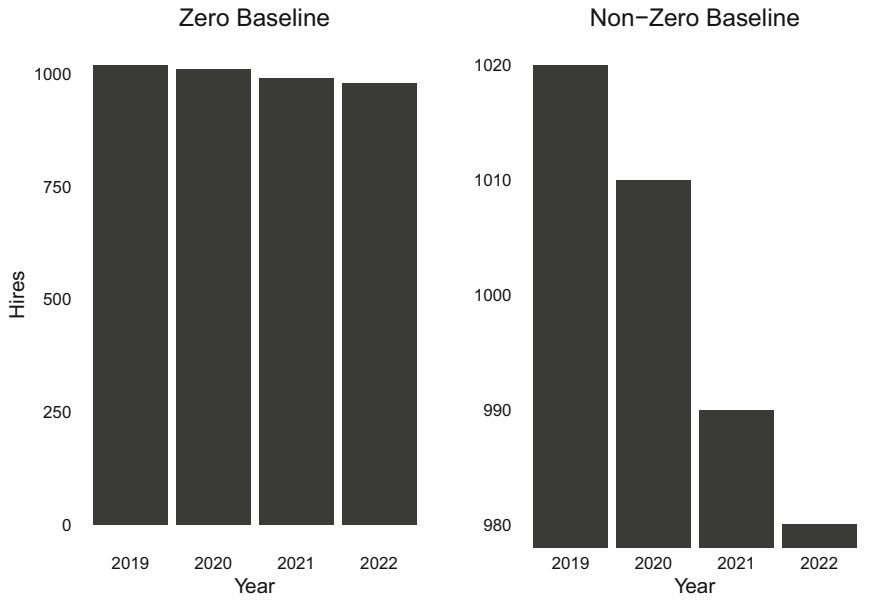

Fig. 2 Hires by year with zero baseline (left) and non-zero baseline (right)

Zero Baseline

The use of a non-zero baseline can exaggerate differences in metrics, resulting in misleading conclusions.

Figure 2 illustrates how minor differences are exaggerated when a non-zero baseline is applied to the *y*-axis. The average number of hires across these four years is 1000, and the small variation YoY is accurately reflected in the bar chart with a zero baseline.

Intuitive Layout

When visualizing data, it is best not to deviate from the way in which the audience will naturally interpret the data. Outlined below are a few important considerations:

- Numbers lower on a y-axis will naturally be interpreted to be smaller than numbers higher on the y-axis.
- When there are both positive and negative numbers along an x-axis, negative metrics are best placed on the left and positive on the right since people naturally consume content from left to right.
- Rotated axis labels require more time to read and interpret. If labels must be rotated to fit along an axis, a 45° angle is generally preferred over a more extreme 90° rotation.

Figure 3 shows murders committed using firearms in Florida across time. This visual has an inverted y-axis, which is highly misleading since what appears to be spikes in murders are actually dips. Irrespective of whether this was politically motivated or simply a failed attempt at creativity, this is a deceptive data visualization.

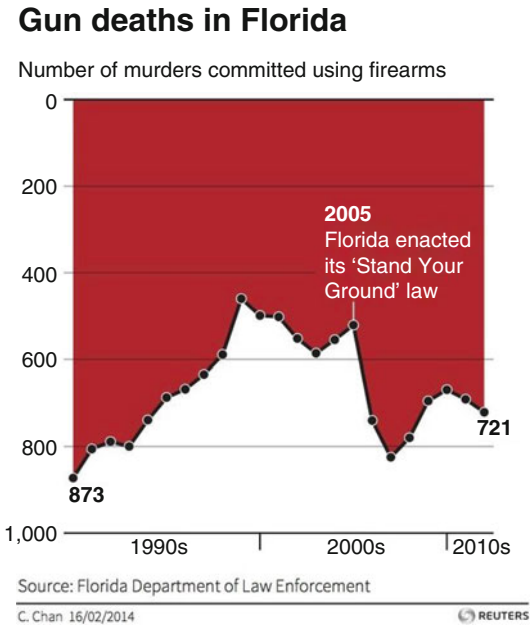

Fig. 3 Misleading inverted y-axis

Fig. 4 The application of preattentive attributes to highlight labor costs as a key driver of increased operating expenses

Preattentive Attributes

Preattentive attributes signal where to look and help our audience focus on content we wish to emphasize (Knaflic, 2015).

It is important to remember that data do not always need to be visualized with a chart. Sometimes communicating with text is a sufficient and effective way to convey information. In the case of text, preattentive attributes are often implemented using bold text and/or contrasting colors for keywords and phrases that need to take center stage. **Bolded** text is generally more effective than *italicizing* or underlining, as it highlights the chosen elements without unnecessary noise and compromised legibility.

Gray is an important color in the implementation of preattentive attributes as it facilitates a move of less important data and design elements to the background to make room for more predominate colors to highlight the most important information. Figure 4 illustrates the application of preattentive attributes to emphasize the key message in a sentence. This is accomplished by pushing the less important content to the background using gray text and using bold orange text to highlight the labor cost increase.

Preattentive attributes can also be applied to data visualizations to focus the audience's attention. Figure 5 illustrates the use of preattentive attributes by assigning a different color (orange) to the labor cost for the current year and muting the labor costs associated with prior years using a less predominate gray.

Simply put, if something is **really important**, make sure it is different from other content on the page, slide, or section.

Step-by-Step Visual Upgrade

Software knows neither what we wish to highlight nor the audience to whom we intend to communicate the information. Therefore, regardless of the software, design defaults are rarely best.

This section will implement data visualization best practices step by step to improve upon the design defaults for the `ggplot2` library.

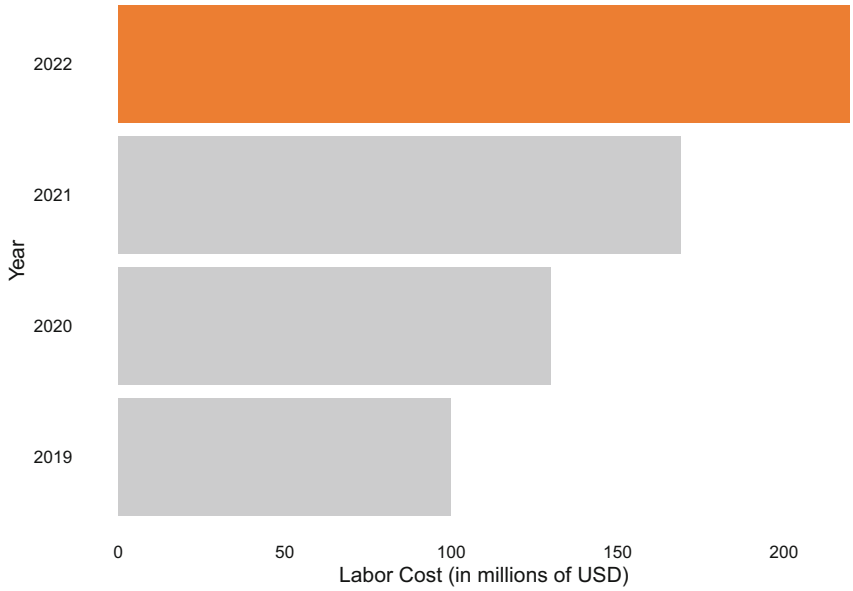

Fig. 5 The application of preattentive attributes to highlight labor costs for the current year relative to prior years

Step 1: Build Bar Chart with Defaults

We will begin by building a bar chart that shows the distribution of active employees' educational backgrounds. To simplify the data structure for data visualization, let us ingest our `employees` data and create an aggregated cube with counts by `ed_field` for active employees:

```
# Load libraries
library(peopleanalytics)
library(dplyr)

# Load data
data("employees")

# Create data cube for active employees
smmry_ed_field <- employees |>
                  dplyr::filter(active == 'Yes') |>
                  dplyr::count(ed_field)
```

A bar chart can be created using the `geom_bar()` function, and we will start with the `ggplot2` library's design defaults (Fig. 6):

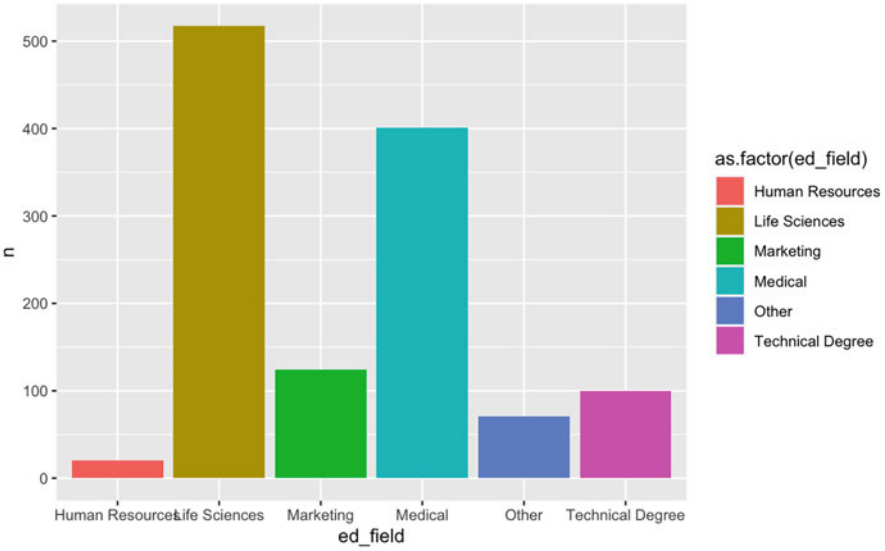

Fig. 6 Step 1: bar chart with defaults

Step 2: Remove Legend

Since the legend is not necessary for interpreting the education field to which each bar corresponds, we will remove it using `theme(legend.position = "none")`. Eliminating the legend also provides more real estate for the chart, which helps make the x-axis values more legible (Fig. 7).

Step 3: Assign Colors Strategically

There is a lot competing for attention due to this vibrant color palette. Let us assume that the objective is to highlight the education field pursued by the largest number of employees. Preattentive attributes lend well to this, and we can assign specific hex color codes to the education categories (Fig. 8).

Since Life Sciences is the field pursued by the most employees, let us highlight the corresponding bar in blue and move the remaining bars to the background through the assignment of light gray. One method of accomplishing this is via the following explicit category assignments:

```
  scale_fill_manual(values = c("Human Resources" = "#BFBFBF",
"Life Sciences" = "#0070C0", "Marketing" = "#BFBFBF", "Medical"
= "#BFBFBF", "Other" = "#BFBFBF", Technical Degree" = "#BFBFBF"))
```

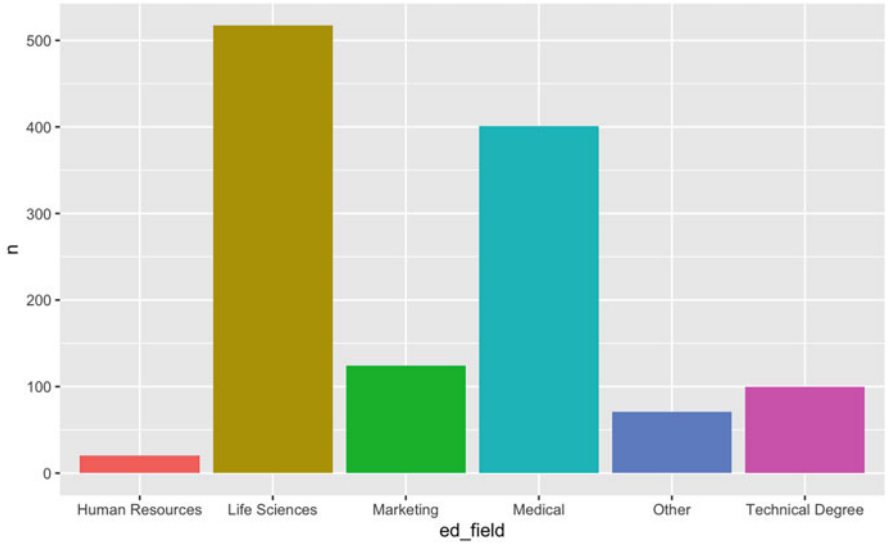

Fig. 7 Step 2: remove legend

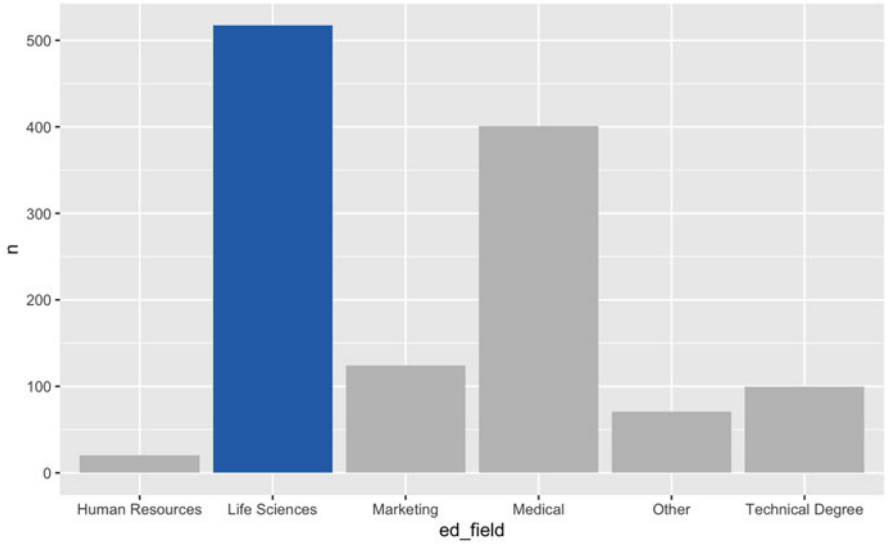

Fig. 8 Step 3: assign colors strategically

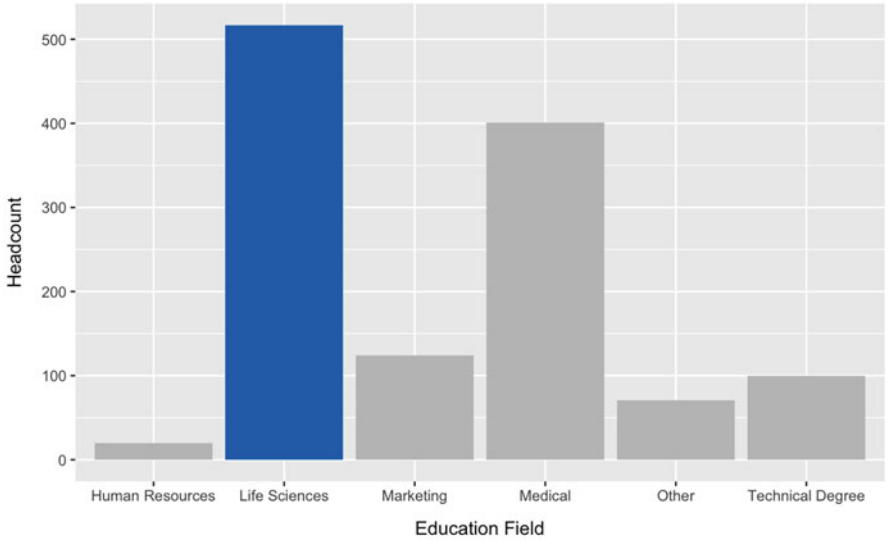

Fig. 9 Step 4: add mixed case axis titles and margins

Step 4: Add Axis Titles and Margins

Axis titles are the column names by default, which is usually not the most user-friendly option. We can assign new mixed case axis titles by chaining `labs(x = 'Education Field', y = 'Headcount')` to the visualization code.

Spacing can also be added between the axis titles and labels to improve upon the default formatting and reduce text congestion. The `ggplot2::element_text(margin = margin(t = 0, r = 0, b = 0, l = 0))` parameter can be defined for the *x* and *y* axes via `axis.title.x` and `axis.title.y`, respectively, where:

- **t** = space on the top of axis title
- **r** = space on the right of axis title
- **b** = space on the bottom of axis title
- **l** = space on the left of axis title

Let us create a margin of white space above the *x*-axis title and to the right of the *y*-axis title by defining the following parameters (Fig. 9):

- `axis.title.x = ggplot2::element_text(margin = margin(t = 10, r = 0, b = 0, l = 0))`
- `axis.title.y = ggplot2::element_text(margin = margin(t = 0, r = 10, b = 0, l = 0))`

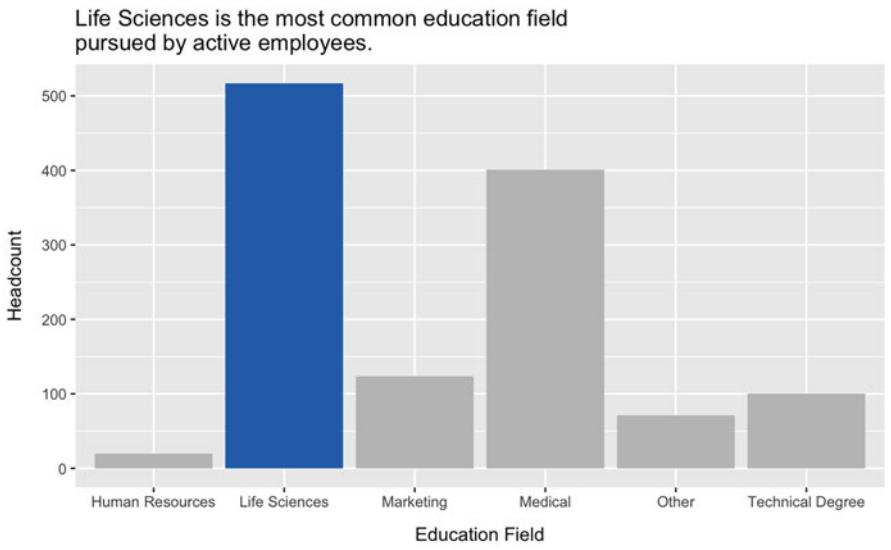

Fig. 10 Step 5: add title

Step 5: Add Left-Justified Title

We can add a chart title via `labs(title = 'This is a chart title.')`. This title can be centered via `theme(plot.title = ggplot2::element_text(hjust = 0.5)`, left justified via `theme(plot.title = ggplot2::element_text(hjust = 0)`, or right justified via `theme(plot.title = ggplot2::element_text(hjust = 1)`.

It is a best practice to left justify titles since the readers consume information beginning with the left side of the page (like reading a book), and left justifying the title increases the probability that the audience will read the title and understand its purpose before engaging with the visual. Left justification is the default for `ggplot2` titles.

It is also a best practice to assign a descriptive title to charts to highlight the key message(s) we want to convey to the audience through the visual. For longer titles, the new line character `\n` can be used to break titles into multiple lines (Fig. 10).

Step 6: Remove Background

The default gray background is a distraction from the data we need to take center stage. We can remove this background using `panel.background = element_blank()` to achieve a cleaner aesthetic and allow the bars in this chart to become more pronounced (Fig. 11).

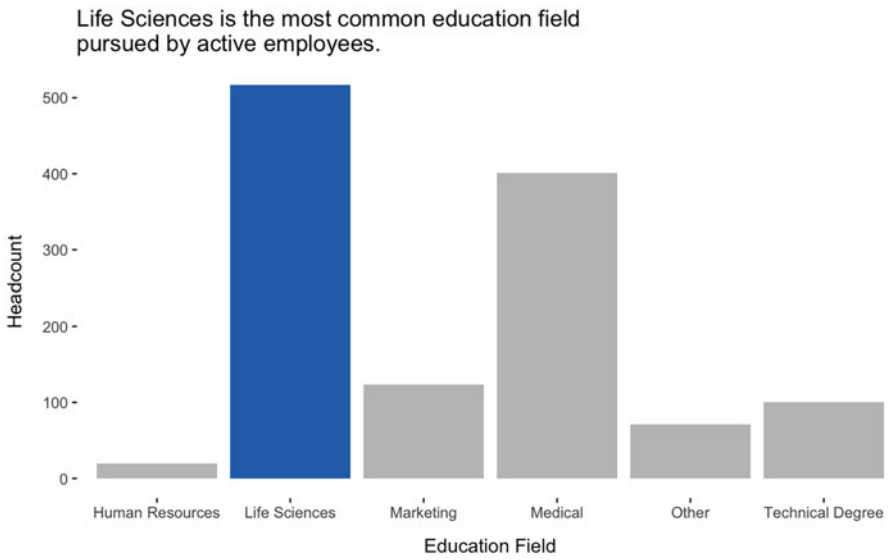

Fig. 11 Step 6: remove background

Step 7: Remove Axis Ticks

Axis ticks add noise to this visual and are not necessary to ascertain to which values along the axes the data align. Axis ticks can be removed for each axis independently with `axis.ticks.x = element_blank()` for the x-axis and `axis.ticks.y = element_blank()` for the y-axis (Fig. 12).

Step 8: Mute Titles

While the chart and axis titles are important for clarifying what information is represented in the visual, these should not be the focus. Just as we pushed the less important education categories to the background using gray text, we can mute the chart and axis titles with gray text to help draw attention to the data.

We can change the color of the chart title to light gray with `plot.title = ggplot2::element_text(colour = "#404040")`. The axis titles can be changed to the same color using `axis.title.x = ggplot2::element_text (colour = "#404040")` for the x-axis and `axis.title.y = ggplot2::element_text(colour = "#404040")` for the y-axis (Fig. 13).

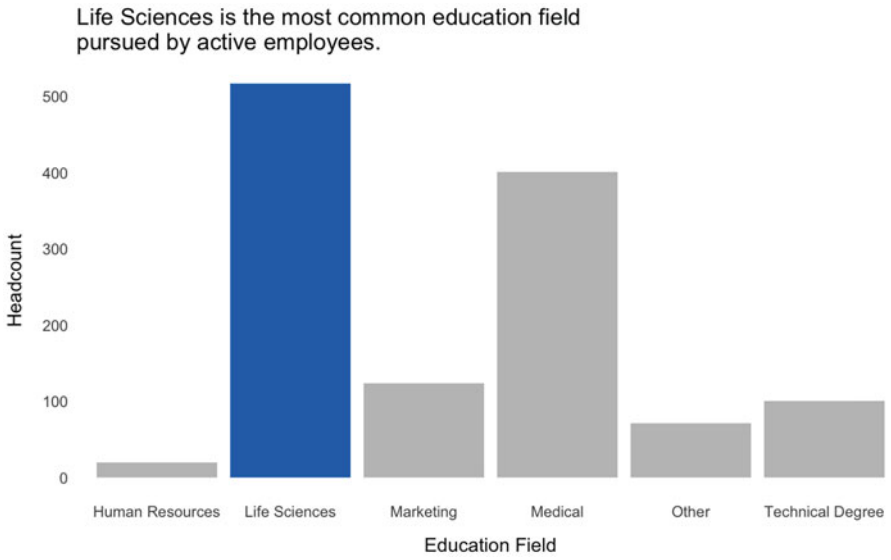

Fig. 12 Step 7: remove axis ticks

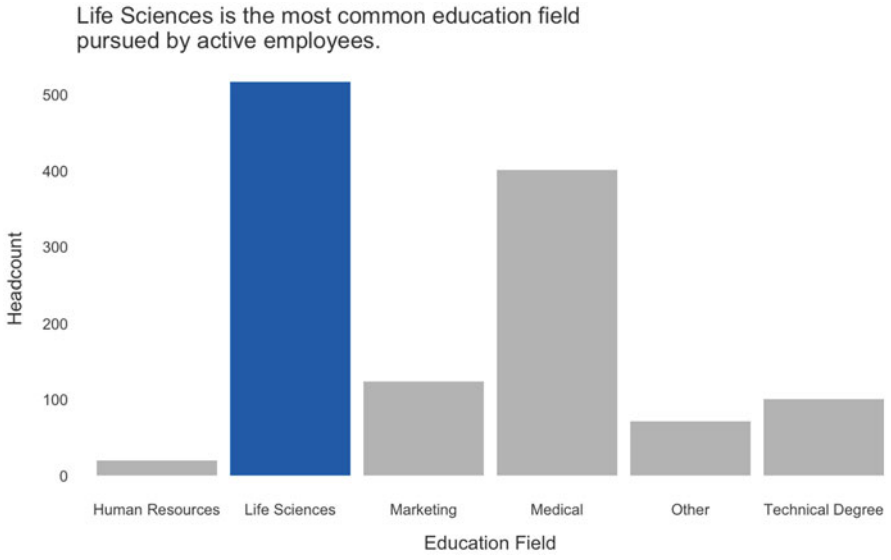

Fig. 13 Step 8: mute titles

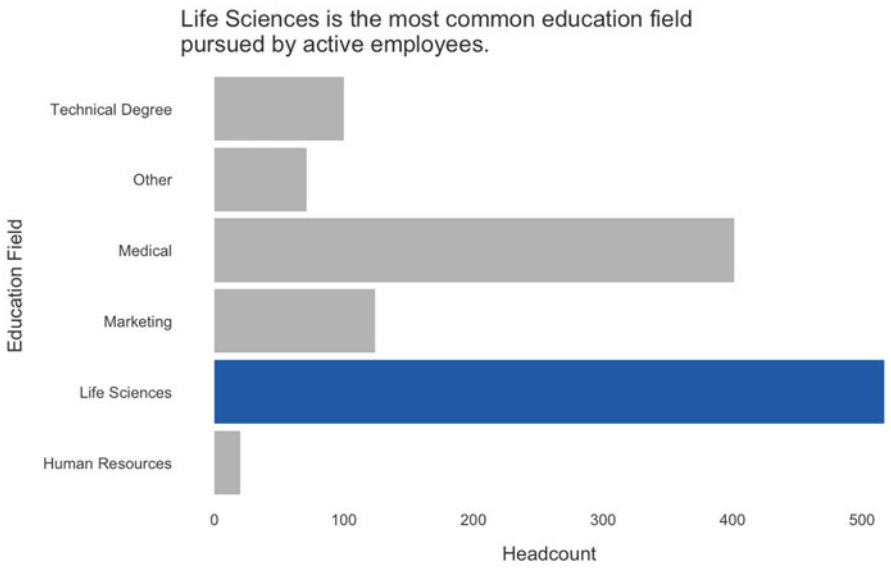

Fig. 14 Step 9: flip axis

Step 9: Flip Axes

Flipping the coordinates of axes to convert the default vertical bar chart into a horizontal bar chart allows the audience to more easily scan down the right side of the visual to quickly identify and understand the relative frequencies of education categories (Fig. 14).

Coordinates can be flipped using `coord_flip()`.

Step 10: Sort Data

This visual can be further simplified by sorting the bars from highest to lowest value. With sorted bars, the audience can more easily ascertain the relative ranking of each education field.

We can pass `reorder(ed_field, n)` into the `aes()` function to sort the education field bars from highest to lowest *n*-count. If we needed to sort in the opposite direction, `reorder(ed_field, -n)` will reverse the sort direction (Fig. 15).

As shown in Fig. 16, the final visual is a marked improvement over the initial design defaults.

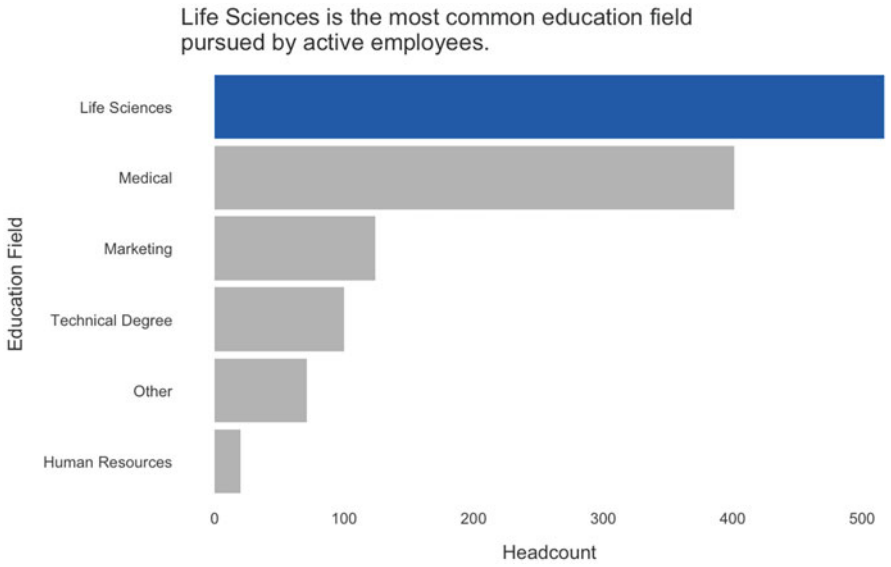

Fig. 15 Step 10: sort data

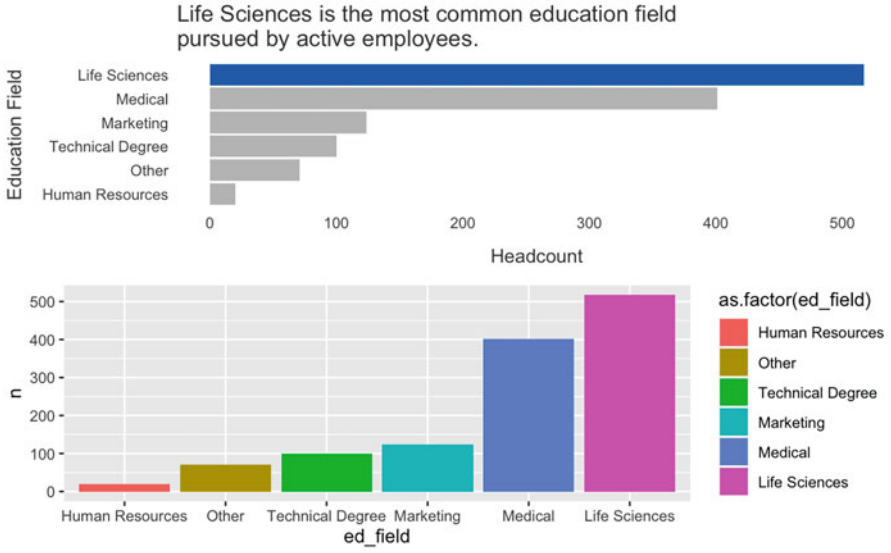

Fig. 16 Enhanced design (top). Design defaults (bottom)

Visualization Types

There are some highly advanced and interactive data visualizations that can be built using JavaScript (JS) libraries such as D3. With some exceptions, JS libraries are generally out of scope for this book. `ggplot2` is capable of building very elegant data visualizations, and this will be the tool used to implement most of the data visualizations in this chapter.

All visuals will have a set of common parameters for design aesthetics, consistent with the themes used to produce the enhanced chart design shown in Fig. 16. Therefore, each section will highlight the differences needed to achieve the respective data visualization. For production applications with many visualizations, it may be helpful to wrap common `ggplot2` design elements in a function to simplify chart building.

Tables

Tables are the most basic way to organize data. Since tables generally contain many metrics, they are usually better situated as reference material in the Appendix of a doc/deck or within a metric drill-through in dashboards rather than occupying prime real estate that should be leveraged strategically to focus the audience's attention on key messages.

A simple cube containing employee counts by department and tenure band will be constructed for demonstrating how to display tabular output:

```
# Append new tenure band column
employees$tenure_band <- dplyr::case_when(
  employees$org_tenure < 1 ~ "Under 1 Year",
  employees$org_tenure < 2.5 ~ "1-2 Years",
  employees$org_tenure < 5.5 ~ "3-5 Years",
  employees$org_tenure <= 10 ~ "6-10 Years",
  TRUE ~ "Over 10 Years"
)

# Store aggregate measures to cube
dept_tenure <- employees |>
               dplyr::filter(active == 'Yes') |>
               dplyr::group_by(dept, tenure_band) |>
               dplyr::summarise(cnt = dplyr::n())

# Specify ordered factor
dept_tenure$tenure_band <- ordered(dept_tenure$tenure_band,
↪  levels = c("Under 1 Year", "1-2 Years", "3-5 Years", "6-10
↪  Years", "Over 10 Years"))
```

The default display of a tibble (data frame produced by `dplyr`) is very basic:

```
# Display output using default settings
dept_tenure
```

```
## # A tibble: 14 x 3
## # Groups:   dept [3]
##    dept                  tenure_band     cnt
##    <chr>                 <ord>         <int>
##  1 Human Resources       1-2 Years         7
##  2 Human Resources       3-5 Years        19
##  3 Human Resources       6-10 Years       17
##  4 Human Resources       Over 10 Years     8
##  5 Research & Development 1-2 Years      148
##  6 Research & Development 3-5 Years      262
##  7 Research & Development 6-10 Years     252
##  8 Research & Development Over 10 Years  148
##  9 Research & Development Under 1 Year    18
## 10 Sales                 1-2 Years        57
## 11 Sales                 3-5 Years        93
## 12 Sales                 6-10 Years      124
## 13 Sales                 Over 10 Years    70
## 14 Sales                 Under 1 Year     10
```

While these data could easily be copied and pasted into presentation software for formatting, additional libraries exist in R for formatting tables. For example, R Markdown scripts can leverage the `DT` package to provide filtering, pagination, sorting, search, and other interactive features for HTML output. Field names can also be changed to proper case via the `dplyr::rename()` function (Fig. 17).

```
# Assign proper case field names
dept_tenure_proper <- dept_tenure |>
                      rename('Department' = dept,
                             'Tenure' = tenure_band,
                             'Count' = cnt)
```

Heatmaps

Heatmaps use a shading scheme to highlight the relative magnitude of numbers in a tabular format.

The `geom_tile()` function can be used to build a heatmap with `ggplot2`. The range of colors can be specified via the `scale_fill_continuous(low =`

Show 10 entries Search:

Department	Tenure	Count
Human Resources	1-2 Years	7
Human Resources	3-5 Years	19
Human Resources	6-10 Years	17
Human Resources	Over 10 Years	8
Research & Development	1-2 Years	148
Research & Development	3-5 Years	262
Research & Development	6-10 Years	252
Research & Development	Over 10 Years	148
Research & Development	Under 1 Year	18
Sales	1-2 Years	57

Showing 1 to 10 of 14 entries Previous 1 2 Next

Fig. 17 Data table

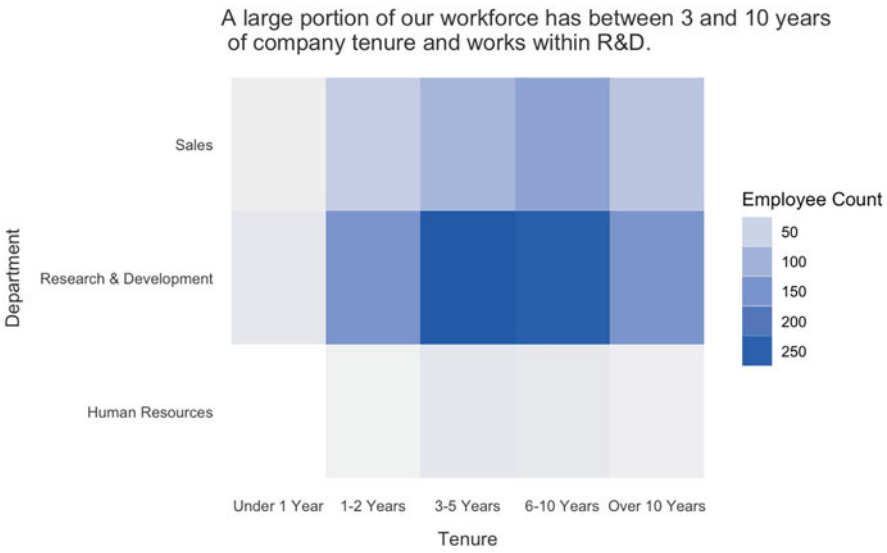

Fig. 18 Heatmap showing the concentration of employees within departments and tenure bands

`'minimum value color', high = 'maximum value color')` function since the fill variable defined by `aes(fill = cnt)` is measured on a continuous scale (Fig. 18).

This heatmap is excellent for focusing attention on the department + tenure segments with highest and lowest employee counts. However, if the specific employee counts are required, complimenting this heatmap with a basic table of metrics is a good option to avoid cluttering the heatmap with $3 * 5 = 15$ additional numbers.

Fig. 19 Scatterplot showing the relationship between work experience and YTD sales for active salespeople

Scatterplots

Scatterplots are useful for visualizing relationships between numeric variables.

Let us build a scatterplot to visualize the relationship between `work_exp` and `ytd_sales`, with a goal of focusing the audience's attention on salespeople whose `ytd_sales` meets or exceeds the full year sales quota of 150,000 USD. Let us first subset `employees` data to active salespeople and append a flag to indicate sales quota attainment for use in shading data points in the scatterplot.

```
# Subset df to active sales employees
sales <- subset(employees, dept == 'Sales' & active == 'Yes')

# Set quote attainment flag for data viz coloring
sales$quota_flg <- ifelse(sales$ytd_sales >= 150000, 1, 0)
```

The `geom_point()` function is used to create a scatterplot. The `scale_y_continuous()` function can be used in conjunction with the `scales` library to override the default scale for the y-axis (scientific notation) with more intuitive values.

While the basic scatterplot in Fig. 19 is effective in visualizing the relationship between `work_exp` and `ytd_sales`, additional design elements are needed to highlight the data points that meet or exceed the full year sales quota of 150,000 USD.

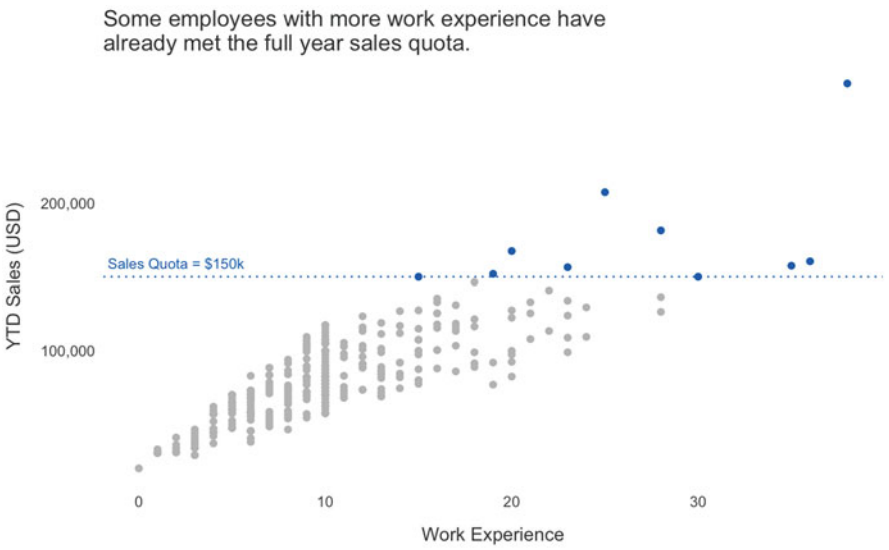

Fig. 20 Scatterplot with data points colored relative to the full year sales quota of 150k USD

The `color` argument for the `geom_point()` function is used to apply preattentive attributes to this visual by specifying the field used for conditional data point shading. We can define the color for each of the field's values using the `scale_color_manual()` function. Additionally, the `geom_hline()` function is used to add a dotted horizontal line at a specified position on the y-axis (at 150,000 USD), and annotation is added at a specified pair of x and y coordinates using the `annotate()` function (Fig. 20).

Line Graphs

Line graphs are used for visualizing continuous data across time.

When visualizing trended data, it is important to avoid cumulative trends because they suggest an upward trajectory (positive slope) even when the corresponding non-cumulative metrics indicate a declining trend.

Consider the following data frame with decreasing hire counts by year:

```
# Print hires df
print(hires_dat[order(hires_dat$year, decreasing = FALSE), ])
```

```
##   year hires cum_hires
## 4 2019  1020       980
## 3 2020  1010      1970
```

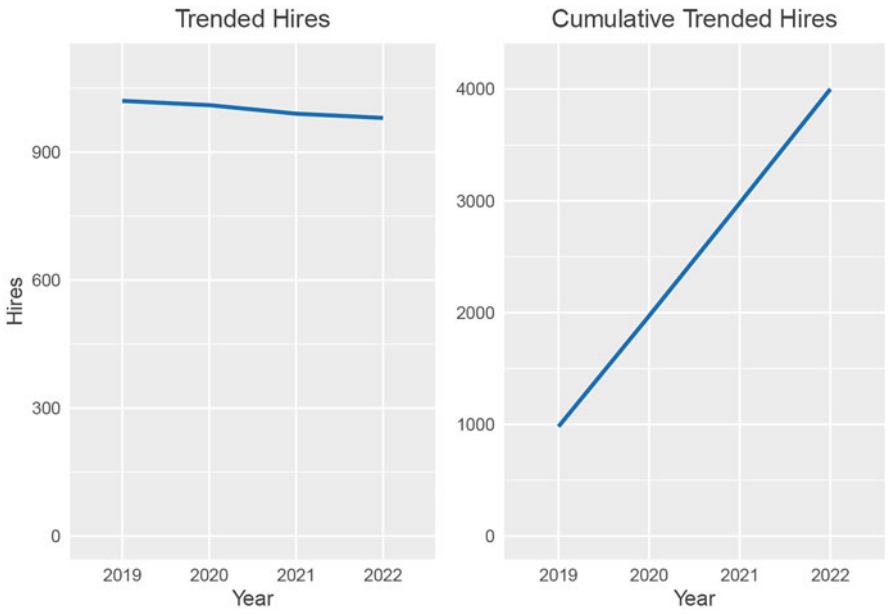

Fig. 21 Hires by year (left) and cumulative hires by year (right)

```
## 2 2021    990         2980
## 1 2022    980         4000
```

Figure 21 juxtaposes a trended line chart with hires by year (left) against a cumulative version of the same (right). Since hires at each year are additive in the cumulative line chart, the slope is positive and misleading.

Single Series

The most basic type of line graph is a **single series line graph**, which reflects a trend for a single group.

Let us generate some attrition data for illustrating various types of line graphs:

```
# Set seed for reproducibility
set.seed(1234)

# Create data
months = 1:24
eng_rt = 5 - runif(1, 2.7, 2.9) + 2.41*months - .41*months^2 +
↪   .02*months^3
fin_rt = runif(1, 5, 8) - 6.97 + 15*months - .53*months^2
ppl_rt = 3 - runif(1, 5, 8) - 6.97 + 12*months - .4*months^2
prd_rt = runif(1, 5, 8) - 6.97 + 13*months - .53*months^2
```

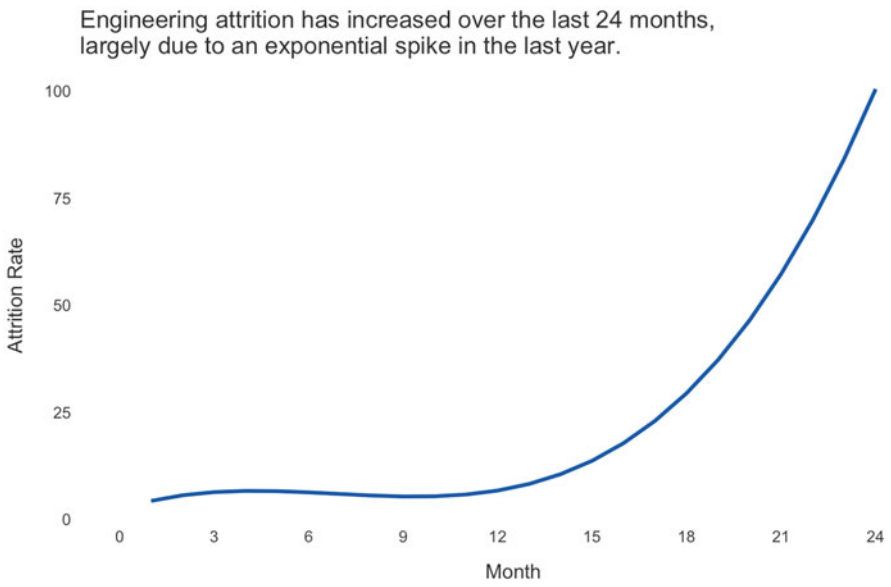

Fig. 22 Single series line graph

```
# Combine dimensions and metrics within df
attrition_dat <- data.frame(month = rep(months, 4),
                            dept = c(rep('Engineering',
                             ↪  length(months)),
                                     rep('Finance',
                                      ↪  length(months)),
                                     rep('People',
                                      ↪  length(months)),
                                     rep('Product',
                                      ↪  length(months))),
                            rate = c(eng_rt,
                                     fin_rt,
                                     ppl_rt,
                                     prd_rt))
```

Line graphs can be constructed using the `geom_line()` function in `ggplot2`. Figure 22 shows a single series line graph built in `ggplot2`.

Two Series

A **two series line graph** reflects trends for two groups, as shown in Fig. 23.

The `aes(color = group)` parameter defines the group by which lines are stratified.

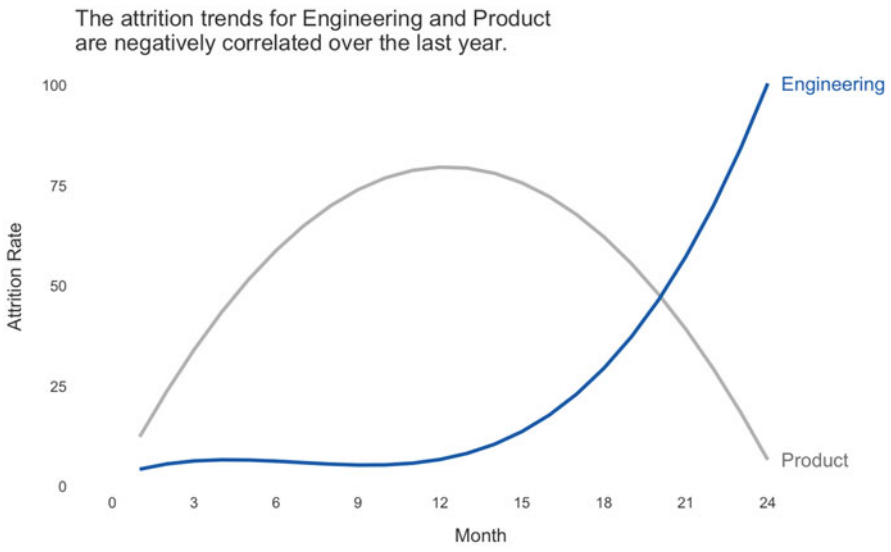

Fig. 23 Two series line graph

Multiple Series

A **multiple series line graph** reflects trends for three or more groups.

Note the preattentive attributes applied in Fig. 24. If the focus is on how a single group (Engineering in this case) compares to two or more other groups, there is no need to differentiate the other groups with respect to color—only label.

Slopegraphs

Slopegraphs are helpful in illustrating relative changes between two points in time.

Common applications for slopegraphs in people analytics include survey variable score changes between two time periods, pre/post changes to outcome variables in an experimental context, and various metrics (e.g., headcount, TTM attrition) for which changes need to be evaluated MoM, QoQ, or YoY when data points between the start and end points are unimportant.

The data structure needed to support a slopegraph is consistent with a line graph. To illustrate how to construct a slopegraph in R, a data frame will be constructed that holds engagement scores for two points in time for both a treatment and a control group.

```
# Build data frame with YoY headcount metrics by department
prepost_scores <- data.frame(date = c(rep('Time 1', 2),
 ↪  rep('Time 2', 2)),
```

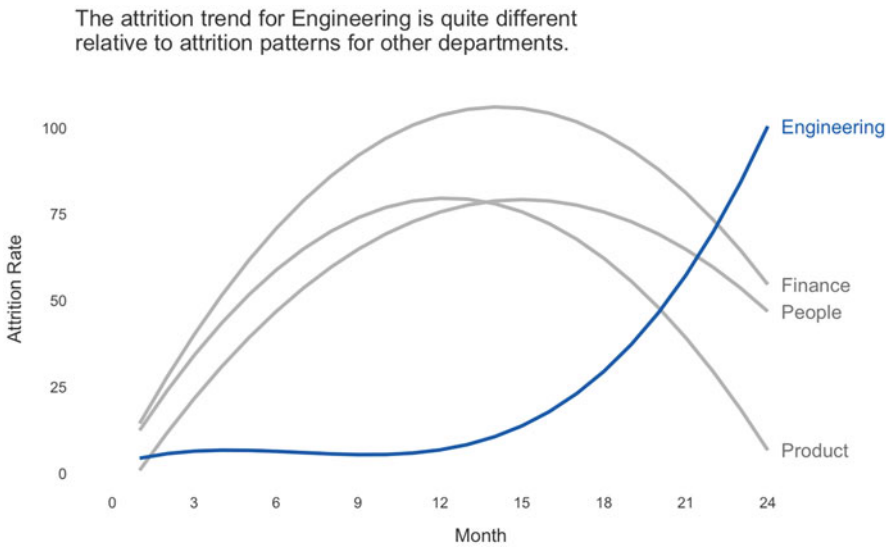

Fig. 24 Multiple series line graph

```
                                    group = rep(c('Treatment',
                                    ↪  'Control'), 2),
                                    score = c(50, 53, 75, 56))
```

In `ggplot2`, a slopegraph is simply a line graph with only two values on the x-axis and some additional formatting including annotations and y-axis removal. Figure 25 is a slopegraph comparing the engagement score changes for treatment and control groups over an observation period, with preattentive attributes applied to focus attention on the treatment group.

Bar Charts

Bar charts are used to display categorical data. Four common types of bar charts are *vertical*, *horizontal*, *stacked*, and *bidirectional*.

Vertical

A **vertical bar chart** is the most basic and pervasive method of visualizing categorical data. Like line charts, bar charts can be single series, two series, or multiple series based on the data that need to be displayed.

To demonstrate how to build bar charts, departmental engagement data will be simulated with some rank variables to support preattentive attributes.

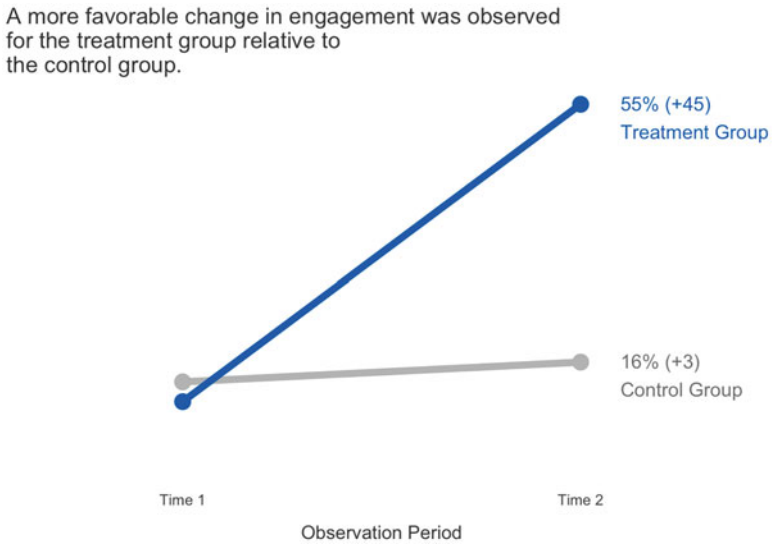

Fig. 25 Slopegraph

```
### Data Prep ###

# Set seed for reproducibility
set.seed(1234)

# Generate favorability distributions
fav_pct <- round(runif(7, 20, 45), 0)
neu_pct <- round(runif(7, 20, 45), 0)
unfav_pct <- 100 - fav_pct - neu_pct
scores <- c(fav_pct, neu_pct, unfav_pct)

# Average top box (favorable) score
topbox_avg <- round(mean(fav_pct), 0)

# Build data frame with YoY headcount metrics by department
engagement_scores <- data.frame(dept = rep(c('Engineering',
↪  'Finance', 'Legal', 'Marketing', 'People', 'Product',
↪  'Sales'), 3),
                                favorability =
                                ↪  c(rep('Favorable', 7),
                                ↪  rep('Neutral', 7),
                                ↪  rep('Unfavorable', 7)),
                                pct = scores)
```

```
# Rank departments by top box score to support sorting
dept_rank <- engagement_scores |>
             dplyr::filter(favorability == 'Favorable') |>
             dplyr::arrange(desc(pct)) |>
             dplyr::mutate(rank = dense_rank(desc(pct))) |>
             dplyr::select(dept, rank)

# Flag top department records in df to support conditional
↪  coloring
engagement_scores$top_score = ifelse(engagement_scores$dept ==
↪  dept_rank[1, 'dept'], 1, 0)

# Add department rank based on top box scores
engagement_scores <- left_join(engagement_scores, dept_rank,
↪  by = "dept")
```

The `geom_bar()` function can be leveraged to construct a bar chart using ggplot2. The `geom_hline()` and `annotate()` functions can be added to include a reference line with the average top box score. Preattentive attributes can also be applied by setting the derived `top_score` variable as the `fill` parameter in the `aes()` function (Fig. 26).

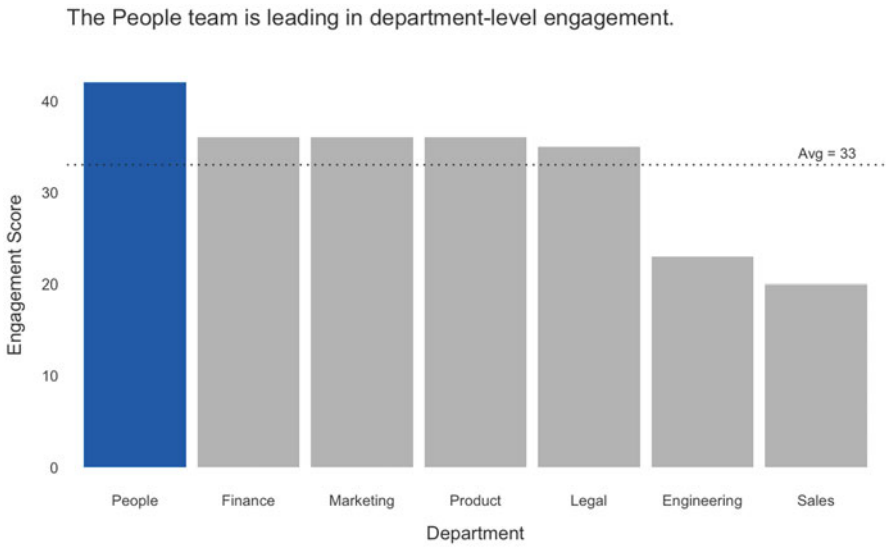

Fig. 26 Vertical bar chart

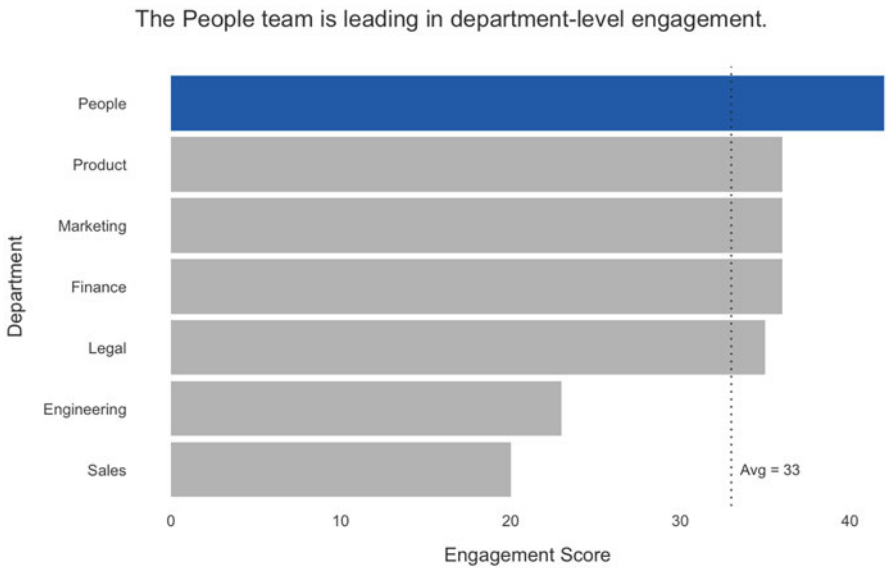

Fig. 27 Horizontal bar chart

Horizontal

The **horizontal bar cart** is a horizontal version of the vertical bar chart, and it tends to be easier to read.

The vertical bar chart can be converted to a horizontal bar chart by adding `coord_flip()` to swap the axes (Fig. 27).

Stacked

The **stacked bar chart** is useful for illustrating the relative contribution of subcomponents to a whole. In a people analytics setting, a 100% stacked bar chart is an excellent tool for visualizing the favorability distribution across survey items and various categorical dimensions (e.g., departments, locations, job profiles).

The only adjustment needed to build a stacked area chart is to specify the variable containing the favorability categories as an ordered factor for the `fill` parameter. Colors can be specified for each favorability category via the `scale_fill_manual()` parameter (Fig. 28).

Bidirectional

The **bidirectional bar cart** is an effective visual for comparing two metrics side by side across values of a categorical variable. The bidirectional bar chart is sometimes referred to as a **divergent bar chart**, **back-to-back bar chart**, or **mirror bar chart**.

Let us illustrate two levers of departmental headcount change—hires and terminations—using a bidirectional bar chart. While the visualization code is

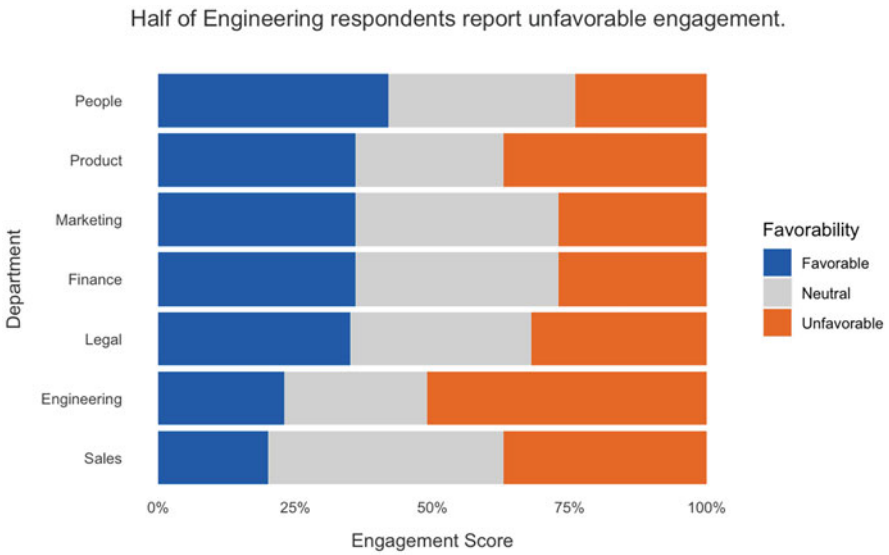

Fig. 28 100% stacked bar chart

consistent with that of horizontal bar charts, the data need to be transformed such that losses (terms) are negative numbers and gains (hires) are positive numbers.

```
# Set seed for reproducibility
set.seed(1234)

# Build data frame with hire and term metrics by department
hires_terms <- data.frame(dept = c('Engineering', 'Finance',
↪  'Legal', 'Marketing', 'People', 'Product', 'Sales'),
                          metric = rep(c('Hires', 'Terms'),
                           ↪  7),
                          cnt = round(runif(14, 5, 150), 0))

# Append transformed count column to support bidirectional bar
↪  charts
hires_terms$cnt_trans <- ifelse(hires_terms$metric == 'Terms',
↪  0 - hires_terms$cnt, hires_terms$cnt)
```

Using the `cnt_trans` field containing both negative and positive integers, we can visualize net changes in departmental headcount (Fig. 29).

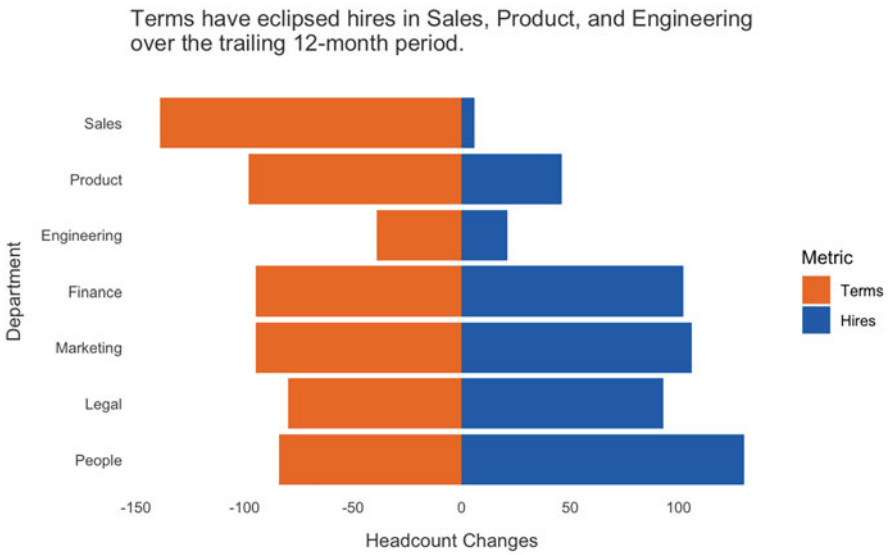

Fig. 29 Bidirectional bar chart

Combination Charts

Combination charts display data using different types of visualizations within the same chart.

In a people analytics context, we may wish to highlight differences between regrettable (bad) and non-regrettable (good) turnover trends relative to total voluntary turnover rates. It is usually more difficult to compare the magnitude of regrettable and non-regrettable rates across time using a stacked bar chart, and a combination chart is often a more intuitive method of presenting this information.

As illustrated in Fig. 30, we can leverage a two-series line chart for monthly regrettable and non-regrettable rates relative to the total voluntary turnover rate visualized with a light gray vertical bar chart in the background.

Waterfall Charts

Waterfall charts are alternatives to stacked bar charts that aid in understanding events between two points in time that explain a change in a starting and ending period value. Explaining drivers of headcount changes over time is a common use case for waterfall charts in people analytics.

Building a waterfall chart requires a bit more data prep relative to the data structure requirements for a stacked bar chart. Unlike a bar chart, a waterfall chart

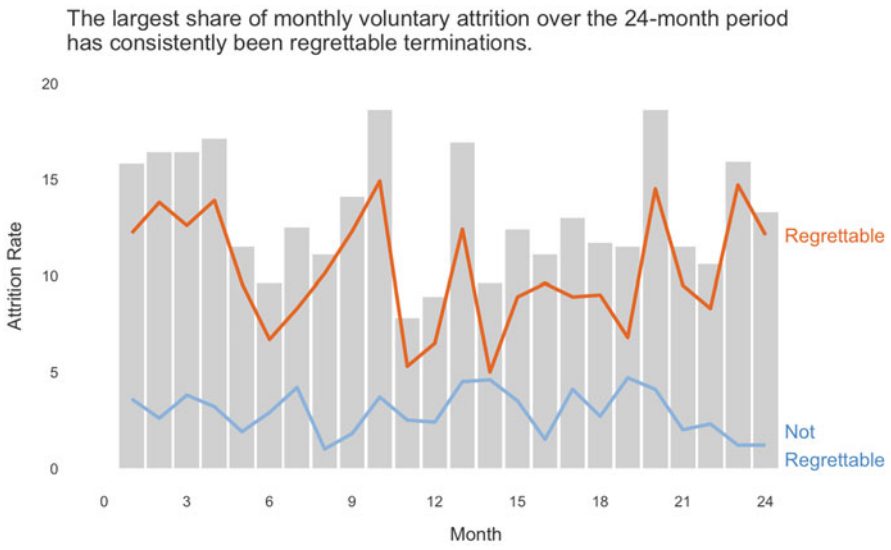

Fig. 30 Combination chart

needs a start and an end value for each event type so that each bar beyond the first begins where the previous bar ends.

```
# Generate headcount data with id field to define bar order
hc_dat <- data.frame(id = 1:6,
                     event = c("Starting HC", "Hires",
                     ↪  "Transfers In", "Transfers Out",
                     ↪  "Exits", "Ending HC"),
                     type = c("Headcount", "Growth", "Growth",
                     ↪  "Loss", "Loss", "Headcount"),
                     count = c(100, 50, 10, -10, -20, 130),
                     start = NA,
                     end = NA)

# Define start and end values to support waterfall chart
hc_dat$end <- cumsum(hc_dat$count)
hc_dat$end <- c(head(hc_dat$end, -1), 0)
hc_dat$start <- c(0, head(hc_dat$end, -1))

# Swap start/end values for last record (Ending HC)
hc_dat[nrow(hc_dat), "end"] <- hc_dat[nrow(hc_dat), "start"]
hc_dat[nrow(hc_dat), "start"] <- 0
```

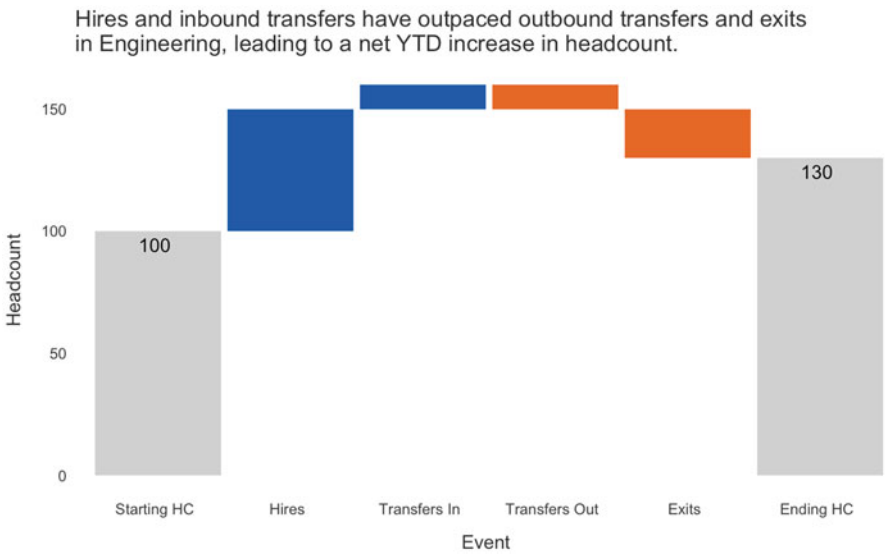

Fig. 31 Waterfall chart

With data properly structured, we can use the `geom_rect()` function to build the waterfall chart. Strategies such as labeling only the Beginning HC and Ending HC reduces clutter and lends to a cleaner design aesthetic (Fig. 31).

Waffle Charts

Waffle charts, also known as **square area charts**, are well-suited for illustrating parts of a whole.

A common application in people analytics is illustrating candidate movement through the recruiting funnel. It is often helpful to visualize a normalized applicant pool (e.g., per 100 applicants) to identify where bottlenecks exist across funnel stages and how pass-through rates are compared across business areas.

Waffle charts require some data prep to visualize:

```
# Create df with TA funnel metrics
ta_dat <- data.frame(stage = c("Apply", "Phone Screen",
↪  "Interview", "Offer Extend", "Offer Accept"),
                     cnt = c(60, 20, 10, 6, 4))

# Set depth of waffle chart (# of y-axis rows)
depth <- 10
```

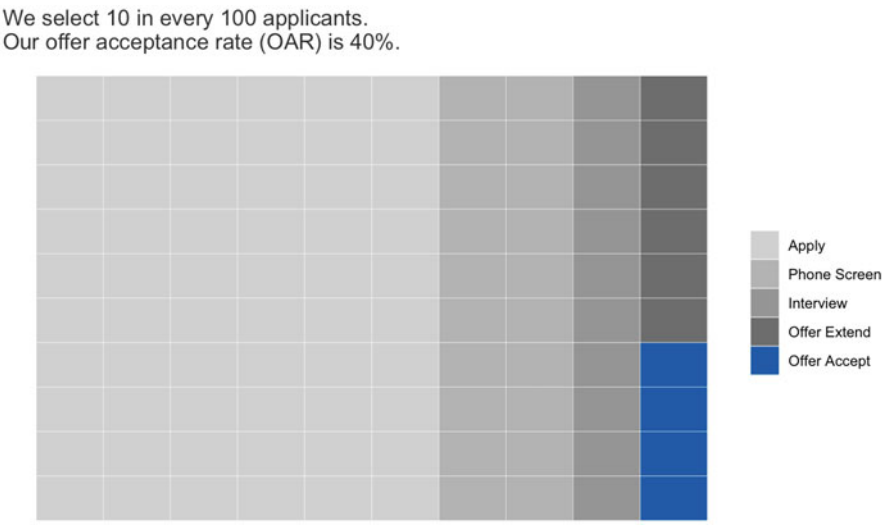

Fig. 32 Waffle chart visualizing pass-through rates across recruitment stages per 100 job applicants

```
# Each observation needs an x and y coordinate, and y needs to
↪  be specified first for a waffle chart with horizontal
↪  accumulation
waffle_dat <- expand.grid(y = 1:depth,
                          x = seq_len(ceiling(sum(ta_dat$cnt)
                          ↪  / depth)))

# Expand the applicant counts into a vector of stage labels
stages <- rep(ta_dat$stage, ta_dat$cnt)

# Integrate stages and fill any extra tiles with NA
waffle_dat$stage <- c(stages, rep(NA, nrow(waffle_dat) -
↪  length(stages)))
```

With this data structure, a waffle chart can be produced using the `geom_tile()` function from `ggplot2`.

The waffle chart in Fig. 32 illustrates how candidates for a set of filled requisitions move from application through the phone screen, interview, offer extend, and offer acceptance stages of the recruiting lifecycle. Based on this chart, 60% of applicants are rejected without a conversation, 40% receive a phone screen, 20% land an interview, 10% receive an offer, and 4% accept an offer.

Sankey Diagrams

Sankey diagrams, or alternatives such as **chord diagrams** and **alluvial plots**, are flow diagrams that are particularly effective in depicting a many-to-many relationship between two domains.

Sankey diagrams have many applications in people analytics. For example, sankeys may be used to understand internal transfers over a period of time (i.e., inflows and outflows among departments) or recruiting sources by which employees in various departments have been hired. To demonstrate how to implement a sankey diagram, let us use the `sankeyNetwork()` function from the `networkD3` library to visualize employee transfers between departments. D3 is an advanced JavaScript framework for creating interactive visualizations, and the `networkD3` library provides an easy interface for constructing sankey diagrams with interactive components. Building a sankey diagram using `networkD3` requires data to be structured within two data frames:

- **Nodes**: Defines the source and destination node names (i.e., the departments employees transfer into and out of)
- **Links**: Connections between pairs of source and destination nodes using an index beginning at 0 to represent the corresponding node from the nodes data frame

```
# Set seed for reproducibility
set.seed(1234)

# Create nodes df
nodes <- data.frame(name = c('Engineering', 'Finance',
↪  'Legal', 'Marketing', 'People', 'Product', 'Sales', #
↪  source
                            'Engineering', 'Finance',
                            ↪  'Legal', 'Marketing',
                            ↪  'People', 'Product',
                            ↪  'Sales')) # destination

# Create links df
links <- expand.grid(source = 0:6, target = 7:13)

# Append employee transfer counts per department pair to links
↪  df
links$value <- ifelse(links$source == links$target-7,
↪  round(rnorm(1000, 150, 50), 0), round(rnorm(1000, 30, 5),
↪  0))
```

With data properly structured within the `nodes` and `links` data frames, these data can be passed into the `sankeyNetwork()` function to construct the sankey

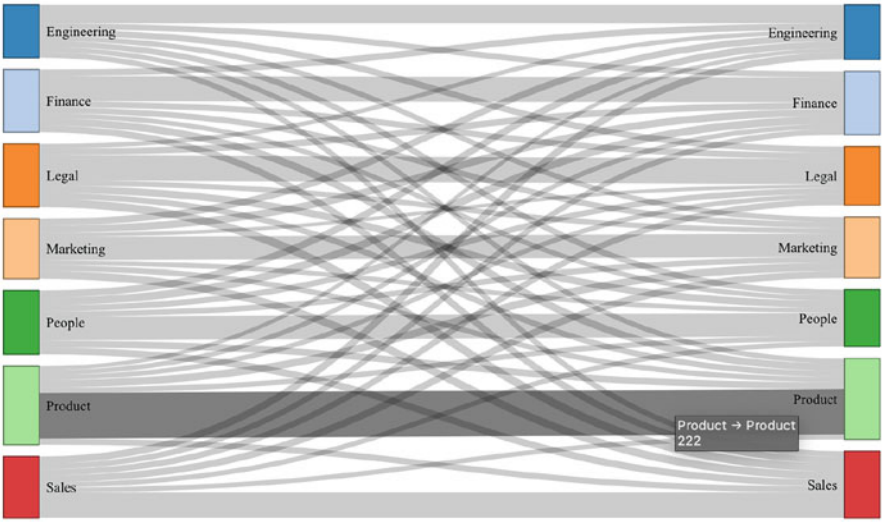

Fig. 33 Sankey diagram showing employee transfers between departments with default colors

diagram. As illustrated in Fig. 33, hovering over a connection displays the count of employee transfers between the corresponding departments ($n = 222$ moves within Product).

Though it can quickly become noisy as the number of nodes increases, coloring connections based on the source departments from which each group flows can be helpful in tracing the employee flow to the various destination departments.

A department variable needs to be defined and added to the `links` data frame in order to color connections based on the source department (transfers out). The `NodeGroup` and `LinkGroup` parameters in the call to the `sankeyNetwork()` function define the column in the `nodes` and `links` data frames, respectively, that specify the group values by which the nodes and links should be colored.

```
# Append source department variable for colored connections
links$dept <- dplyr::case_when(
  links$source == 0 ~ "Engineering",
  links$source == 1 ~ "Finance",
  links$source == 2 ~ "Legal",
  links$source == 3 ~ "Marketing",
  links$source == 4 ~ "People",
  links$source == 5 ~ "Product",
  links$source == 6 ~ "Sales",
  TRUE ~ "NA"
)
```

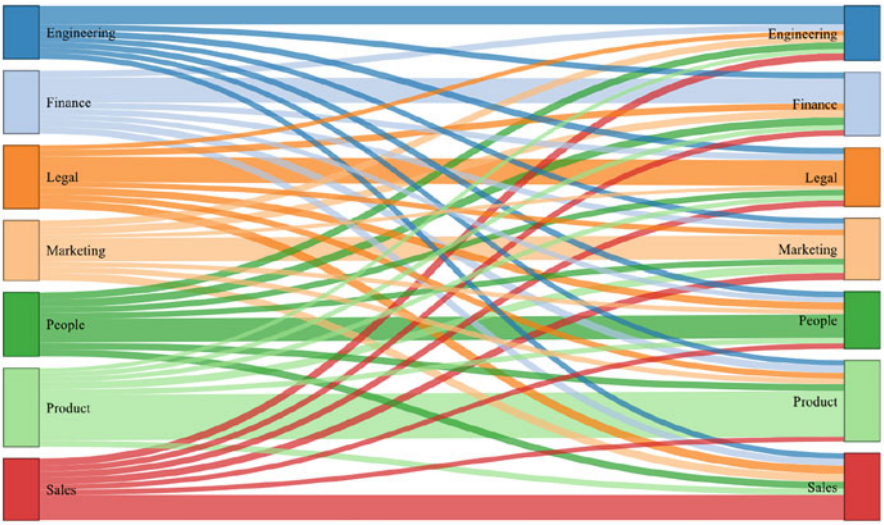

Fig. 34 Sankey diagram showing employee transfers between departments with connections shaded based on a source department color scheme

We can add another grouping variable to support coloring intradepartmental (within) moves differently from interdepartmental (outside) moves (Fig. 34).

```
# Append within/outside department transfer variable for
↪  colored connections
links$wiout_dept <- dplyr::case_when(
  (links$source == 0 & links$target == 7) | (links$source == 1
↪  & links$target == 8) | (links$source == 2 & links$target
↪  == 9) | (links$source == 3 & links$target == 10) |
↪  (links$source == 4 & links$target == 11) | (links$source
↪  == 5 & links$target == 12) | (links$source == 6 &
↪  links$target == 13) ~ "Within",
  TRUE ~ "Outside"
)
```

If the focus is on a single department's transfers, preattentive attributes can be applied to help focus the audience's attention and move the less important information to the background to reduce clutter (Figs. 35 and 36).

To facilitate this, a grouping variable can be defined to differentiate a single department, such as Product, from remaining departments. The color of each group can then be specified using valid D3 code: `d3.scaleOrdinal(["group_1_color", "group_2_color"])`:

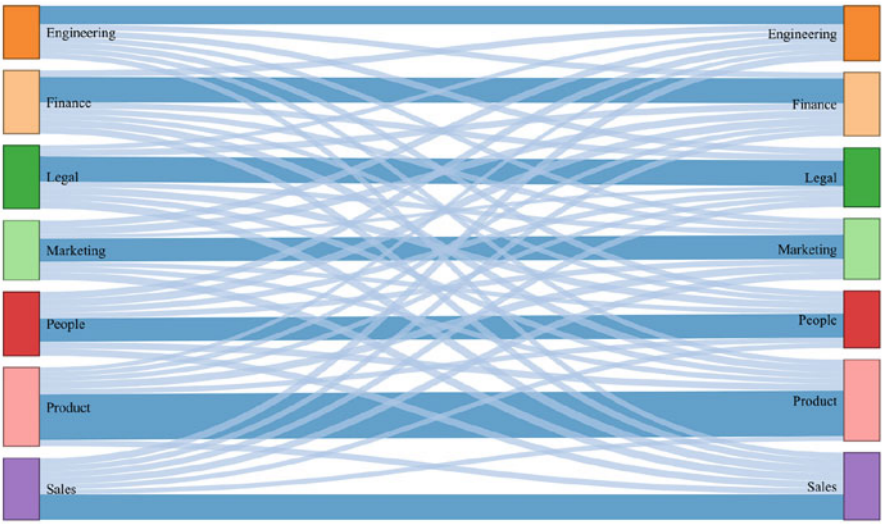

Fig. 35 Sankey diagram showing employee transfers between departments with connections shaded based on an intradepartmental vs. interdepartmental color scheme

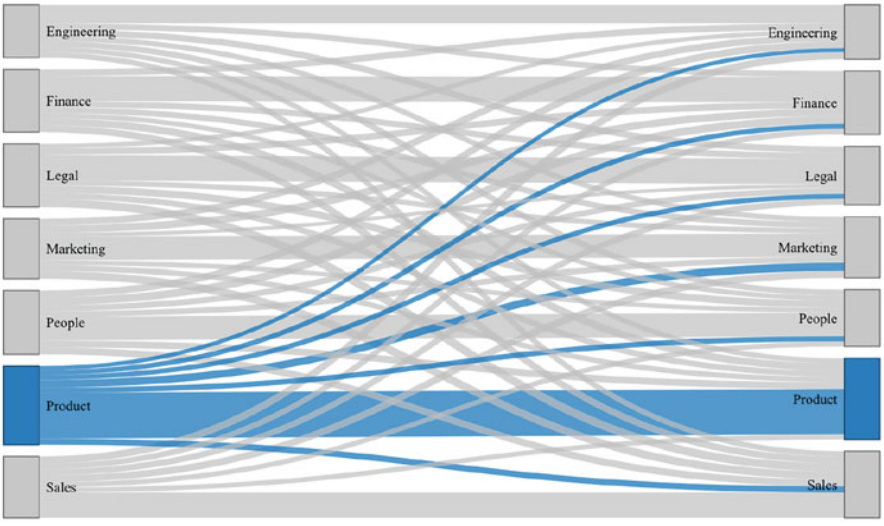

Fig. 36 Sankey diagram with preattentive attributes to highlight employee transfers out of the product department

```
# Append dichotomous product/other indicator to links and
↪  nodes data frames
links$prod <- ifelse(links$dept == 'Product', 'Product',
↪  'Other')
nodes$prod <- ifelse(nodes$name == 'Product', 'Product',
↪  'Other')
```

Pie Charts

Pie charts are rarely an effective way to visualize data, and this book does not generally endorse them. Pie charts can be appropriate in cases where there are two or three mutually exclusive and collectively exhaustive groups and we need to visualize the relative contribution of each to the whole. However, it quickly becomes difficult to ascertain relative size beyond a few groups with a pie chart, and labeling many slices adds a lot of clutter and noise. Proportions should always sum to 1, so if data are not mutually exclusive parts of a whole, a pie chart is not appropriate.

Figure 37 shows a donut chart—which is simply a pie chart with the center cut out—that is intended to show Kane Williamson's outsized contribution to New

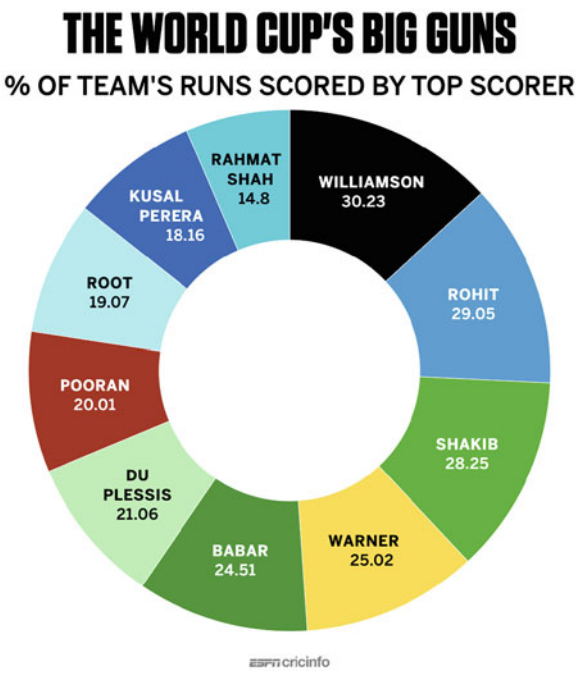

Fig. 37 Improper use of a donut chart

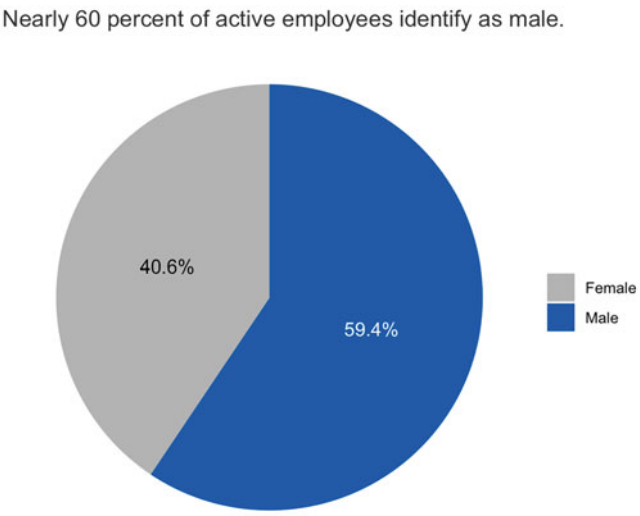

Fig. 38 Pie chart showing gender representation for active employees

Zealand's cricket runs. However, top scorers from other countries are also included in this visual, so the metrics do not sum to 100%. Pie and donut charts are not appropriate for these data.

There is no specific `geom()` function for building pie charts using `ggplot2`. Therefore, we need to create a bar chart and make it circular by adding a `coord_polar("y", start = 0)` transformation. The `coord_polar()` transformation complicates the positioning of labels, but we can add a `position = position_stack(vjust = 0.5))` argument within the `geom_text()` function to easily achieve metric centering within the respective category. We can also easily improve the aesthetics with the `theme_void()` function, which removes the default background, grid, and labels (Fig. 38).

The data need to be structured consistent with the requirements for a bar chart. A basic data cube with descriptive statistics by gender category will simplify the construction of the pie chart in `ggplot2`:

```
# Create data cube for gender representation among active
↪  employees
smmry_gender <- employees |>
                dplyr::filter(active == 'Yes') |>
                dplyr::group_by(gender) |>
                dplyr::summarise(cnt = dplyr::n()) |>
                dplyr::mutate(pct = round(cnt / sum(cnt) *
↪  100, 1)) |>
                dplyr::arrange(desc(pct))
```

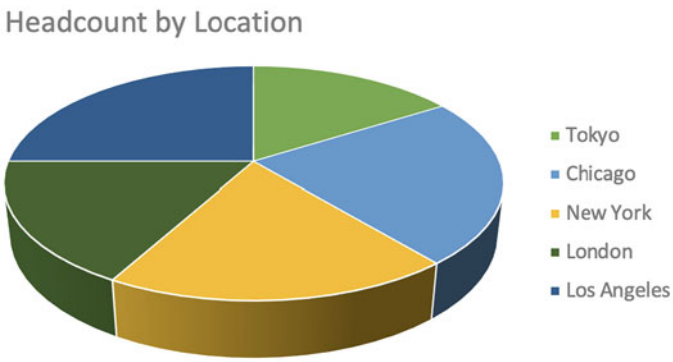

Fig. 39 Headcount by location misrepresented with a 3D pie chart

3D Visuals

An element of uncertainty is inherent in analytics, which is why this book avoids the use of categorical terms such as *always* and *never*. However, an exception will be made for this section. 3D visualizations are *never* appropriate. 3D visuals often misrepresent the data they are intended to visualize and are unnecessarily difficult to interpret.

Figure 39 shows the distribution of headcount across locations using a 3D pie chart. Based on this visual, it may not be apparent that Los Angeles (25%) is considerably larger than New York (19%). The misleading perspective of this tilted visual causes locations on the back side of the pie chart to appear smaller relative to locations on the front side.

A 3D bar chart is not an improvement over a 3D pie chart. As illustrated in Fig. 40, bars do not appear to align with the actual corresponding values (e.g., Los Angeles = 25%). This is because software determines the height of bars in a 3D bar chart using an invisible tangent plane to intersect the bars at the correct y-axis location.

As shown in Fig. 41, by leveraging a horizontal bar chart with labeled bars and preattentive attributes, it becomes easy to understand relative headcount across locations.

Elegant Data Visualization

This chapter has covered the fundamentals of effective data visualization. Applying these best practices will elevate impact when presenting results of analyses to stakeholders.

Fig. 40 Headcount by location visualized with a misleading 3D bar chart

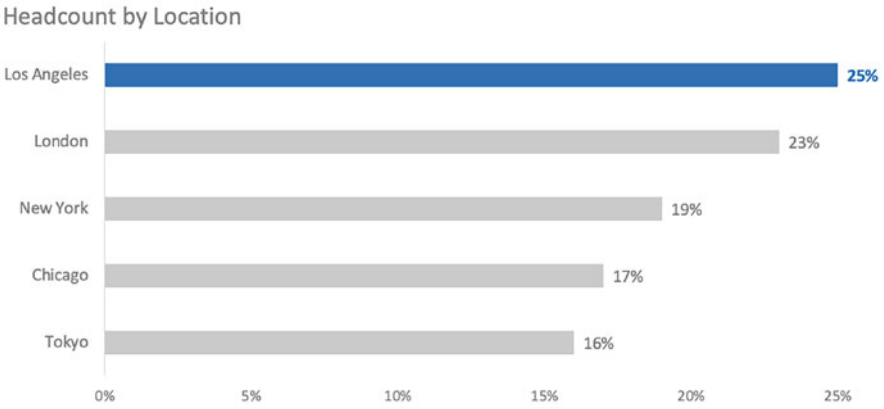

Fig. 41 Headcount by location properly visualized with a horizontal bar chart

Figure 42 is an example of an excellent data visualization that brings together many of the design elements promoted in this chapter. This visual shows seasonally adjusted jobless claims over a period of two decades to highlight the significance of the Coronavirus's impact on the labor market using the following design elements:

- Larger and darker text for the headline
- Smaller and lighter text for the commentary
- Gray axis labels and chart annotations to avoid competing with *current* statistics
- Low-profile dotted horizontal lines to help connect bars across the long x-axis to their corresponding y-axis values
- Orange coloring to represent jobless claims (an unfavorable event)
- Creative use of a bar chart resembling an area chart to represent the weekly interval for jobless claims over the 20-year period

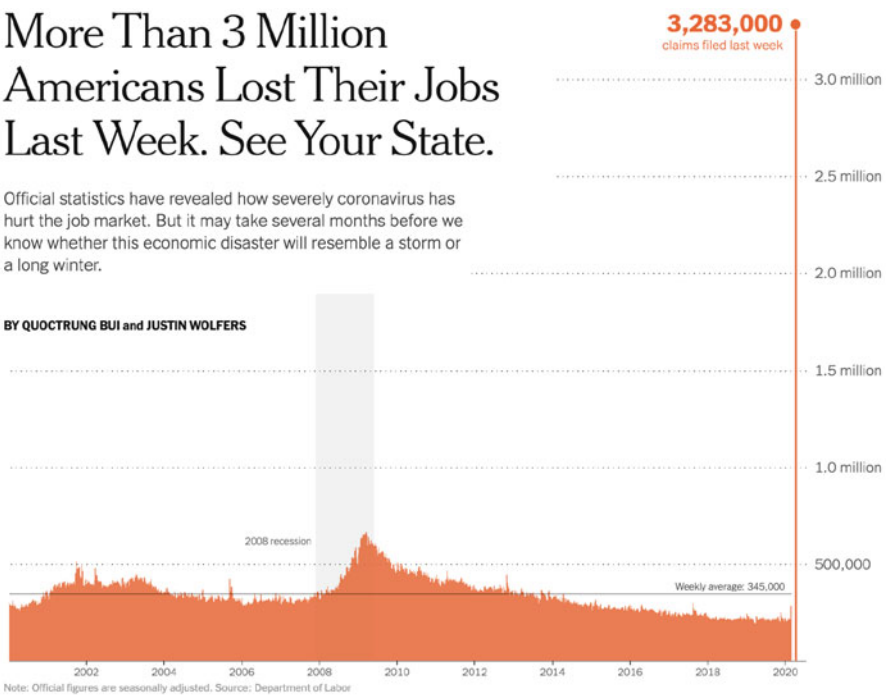

Fig. 42 An archetype of data visualization elegance

- Two helpful reference points to provide perspective: (1) 20-year weekly average and (2) cumulative jobless claims during the last significant US recession in 2008

Review Questions

1. What is the function of preattentive attributes in data visualization?
2. Why is the use of red and green in data visualization potentially problematic?
3. Why are horizontal bar charts easier to interpret than pie charts?
4. What types of data are visualized in scatterplots?
5. What advantages do heatmaps provide over tables?
6. What are some people analytics use cases for slopegraphs?
7. What are some people analytics use cases for sankey diagrams?
8. Why is it important to use a zero baseline for y-axes?
9. Why is it a best practice to avoid rotating axis labels?
10. What is the primary purpose of applying a consistent color palette across visuals?