CHAPTER 5

Data Visualization

OBJECTIVE

This chapter is an introduction to data visualization techniques with R, using the `ggplot2` package. Here is the place where you will learn how to create a wide variety of graphs in R. You will learn to write simple and reusable R scripts for reading Excel data, creating good-looking graphs that can be exported to Excel.

Contents

5.1 Introduction

This chapter is devoted to the important topic of data visualization. Data visualization is an essential component in any data analysis project, and Excel offers several options for creating charts of various types. R however, is clearly unbeatable when it comes to graphics. The sophistication level of the graphs you can create in R will be limited only by your imagination. Several charts that are not part of Excel are readily available in R. Creating basic graphs in R will be relatively simple. Although creating more complex charts will require a bigger investment of your time, you will learn step by step, how to use the different features R has to offer to produce the best results.

I still use Excel for graphics, especially when it is a simple graph, which does not require substantial manual formatting. Excel can actually be quick and user-friendly if you have a dataset ready to be visualized and that the number of graphs to create is limited. Otherwise, I would prefer R and the `ggplot2` package if the dataset requires an initial and substantial data manipulation, or many graphs must be created, or the graphs to be created will need to be updated with new data at a later time. Using Excel in any of these situations will be time-consuming and error-prone.

Note that graphs created with R can be exported to Excel as discussed in chapters 7 and 8. However, a graph created in R cannot be modified from within Excel. You will need to make any change to the graph by modifying the original R script that created it.

5.2 Graphics with ggplot2

To create charts in R, you will need an R package called `ggplot2`, which is one the core packages included in `tidyverse`. Therefore, once `tidyverse` is installed and loaded, you will be ready to start using `ggplot2`. The `ggplot2` documentation is immense, and the focus of this chapter is on providing you

with the essentials and showing you step by step how to easily create and export the most commonly-used graphs for data analysts. However, if you need more details later on, you should be able to find them using the link `https://ggplot2.tidyverse.org/`. There is also a book by Wickham (2016), which provides a comprehensive coverage of `ggplot2`, although the material may overwhelm most beginners.

Although base R has some capability in graphics, what the `ggplot2` package has to offer is superior by a wide margin. Therefore, I strongly advise you to focus on `ggplot2` for graphics as it is one of the most flexible and versatile data visualization tool for data analysts.

You will first learn how to create some basic plots in the next few sections, and how to export them in various formats. To illustrate the creation of various plots with `ggplot2`, I will consider an example of the manager of a company who wants to compare the impact that 3 different promotional campaigns had on the sales volume of cracker boxes. The 3 promotional campaigns are following:

- Sampling of product by customers in store and regular shelf space.
- Additional shelf space in regular location.
- Special display shelves at ends of aisle in addition to regular shelf space.

For this experiment, 15 stores were selected and each of them was randomly assigned one of the 3 promotion types described above, resulting in 5 stores being assigned to each type of promotion. For each combination of store and promotion type, the following 2 measurements were taken:

- $y.post$: number of boxes of crackers sold during the promotional period,
- $x.pre$: number of boxes of crackers sold during the period preceding promotion.

The data collected from this experiment is presented in Table 5.1 .

Using the `ggplot2` package may appear confusing at first. It is actually quite simple once you get your feet wet. Learning by example is likely one of the most effective ways to master this popular data visualization tool. Creating any graph with `ggplot2` amounts to performing several small tasks, each of which is accomplished by using a specific function.

An initial skeleton `ggplot` object defined by the axes and a grid are first laid out with the `ggplot()` function. One or many subsequent "layers" using specific functions are added to this plot object. For example, using the `geom_point()` function, you can add a new scatterplot layer to the initial plot object.

Table 5.1 : Cracker Promotion Data[a]

storeno	Promotion	*x.pre*	*y.post*
S11	P1	21	38
S21	P1	26	39
S31	P1	22	36
S41	P1	28	45
S51	P1	19	33
S12	P2	34	43
S22	P2	26	38
S32	P2	29	38
S42	P2	18	27
S52	P2	25	34
S13	P3	23	24
S23	P3	29	32
S33	P3	30	31
S43	P3	16	21
S53	P3	29	28

[a]Source: Neter et al. (1985, Page 854).
Download this data in a `csv` file with the link: `https://bit.ly/3yWQO67`

Script 5.1 is a small script file, which creates the scatterplot of Figure 5.2. This script file is assumed to be located in a project root directory, whereas the input dataset `crackers.csv` would be in a subdirectory named `data/chap5`.

Script 5.1. Basic scatterplot of *y.post* versus *x.pre*

```
1 library(tidyverse)
2 crackers.df <- read_csv("./data/chap5/crackers.csv")
3 sp <- ggplot(data=crackers.df,mapping=aes(x=x.pre,y=y.post))
4 sp +
5  geom_point(shape=16,size=3)
```

End of Script

Line #1 of this script loads the `tidyverse` package to the R session. Line #2 reads the `csv` file containing Table 5.1 data to create the `crackers.df` data frame (update the project subdirectory to match the location of input dataset `crackers.csv`). In line #3, the `ggplot()` function is used to create the initial `sp` ggplot object shown in Figure 5.1. As you can see, the `ggplot()` function creates that empty shell on which you can put several layers of graphs. Line #4, indicates that a new layer will be added to the initial shell `sp`. In line #5, I use the `geom_point()` function to add a new scatterplot layer to Figure 5.1 to create the new Figure 5.2. You will see in the next few sections that you can actually add many layers using the `"+"` sign at the end of the last layer.

Let me now review what Script 5.1 actually does:

- The `ggplot()` function is called with 2 arguments: (1) The `"data="` argument, which specifies the input dataset containing the 2 variables to be plotted, (2) The `"mapping="` argument, defining a set of *aesthetic mappings* described using the `aes()` function[1]. For the `ggplot()` function, aesthetic mappings are generally limited to specifying the x and y variables with the parameters `x=` and `y=`.

- The `geom_point()` function is one of the many possible functions used for specifying the type of graphs you want (scatterplot, bar, line, ...), or the

[1]Note that the `ggplot()` function itself does not have the capability to read the content of the input dataset in its `data` argument. Therefore, the use of another function `aes()` is necessary to specify the x and y variables.

type of layer you want to add to the original plot shell. In `ggplot2` lingo, the types of graph are referred to as *geometric objects*. All functions in this family of functions start with the same prefix `geom_`.

In Script 5.1, I specified the 2 parameters `shape=` and `size=` to describe the point shape and its size that I wanted to use for representing the data points. Section 5.3 provides a more detailed discussion of scatterplots.

- The `"data="` argument and aesthetic mappings defined with the `aes()` function, can be set in `ggplot()` as well as in individual layers, such as `geom_`. When they are not in the `geometric` layer, their values will be inherited from the `ggplot()` function.

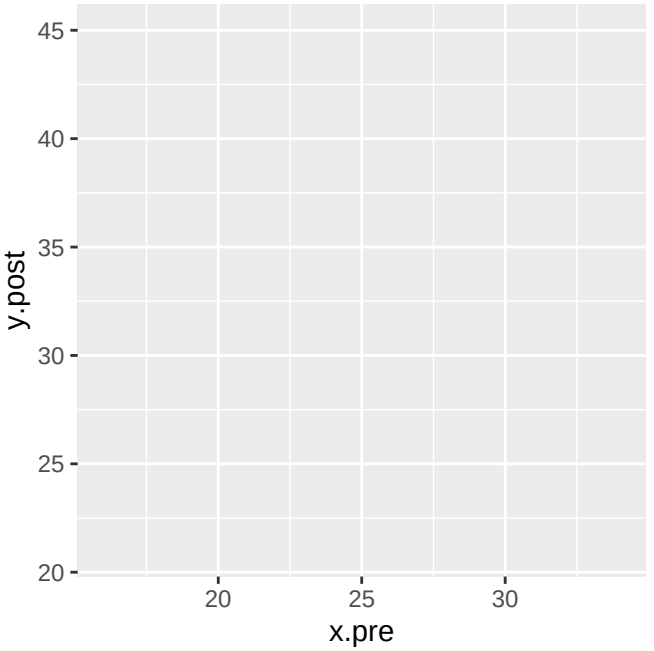

Figure 5.1: Chart grid produced by Script 5.1

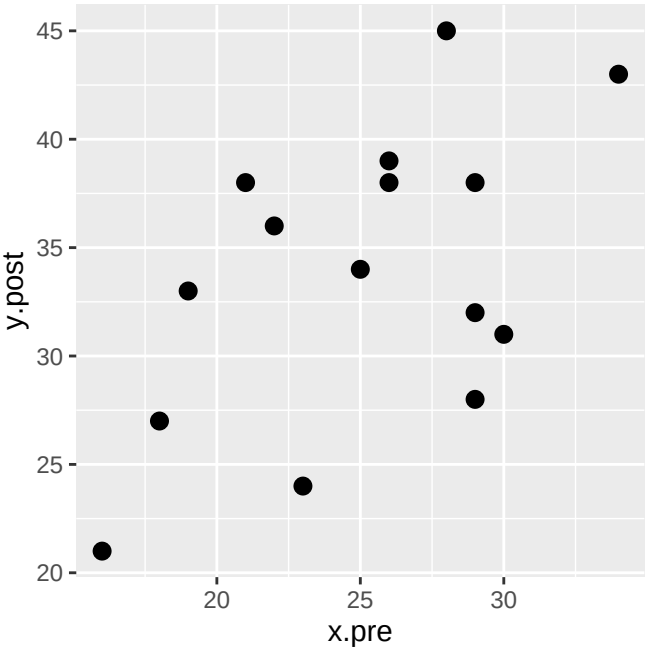

Figure 5.2: Scatterplot produced by Script 5.1

5.2.1 *Changing the Plot Theme and Adding Titles and Subtitles*

The plot theme generally refers to all non-data display that you may want to customize. This includes but is not limited to the grid lines and the background color. Figure 5.1 shows the default theme used by `ggplot2`, and which I personally do not like very much. `ggplot2` gives you several options for modifying this theme. One in particular that I like is obtained with the `theme_classic()` function used in Script 5.2. Additional options are reviewed in section 5.3 and a more detailed review of these options can be obtained with the link `https://ggplot2.tidyverse.org/reference/ggtheme.html`.

Other graph elements of interest with a weak link to data include main titles, subtitles and other labels found in a graph. All these labels can be added or customized using the `labs()` function. This function is used in Script 5.2 to

obtain the graph of Figure 5.3. You can see that the `labs()` function used here takes 2 arguments. The `title` argument defines the main title, whereas the `subtitle` argument defines the subtitle. Moreover, the plot theme was changed with the use of function `theme_classic()`.

Script 5.2. Basic scatterplot of *y.post* versus *x.pre*

```
library(tidyverse)
crackers.df <- read_csv("./data/chap5/crackers.csv")
sp <- ggplot(data=crackers.df,mapping=aes(x=x.pre,y=y.post))
sp +
  geom_point(shape=16,size=3) +
  labs(title =
    "Scatterplot of crackers sold before & during promotion",
    subtitle = "Data source:  Neter et al.  (1985)") +
  theme_classic()
```

End of Script

> *The default theme of ggplot2 is them_gray(), also known as them_grey(). I personally do not like this theme, and very few people do. My personal preference is theme_classic(). You can make it your default theme for an R session by executing the following command:*
>
> ```
> theme_set(
>   theme_classic()
> )
> ```

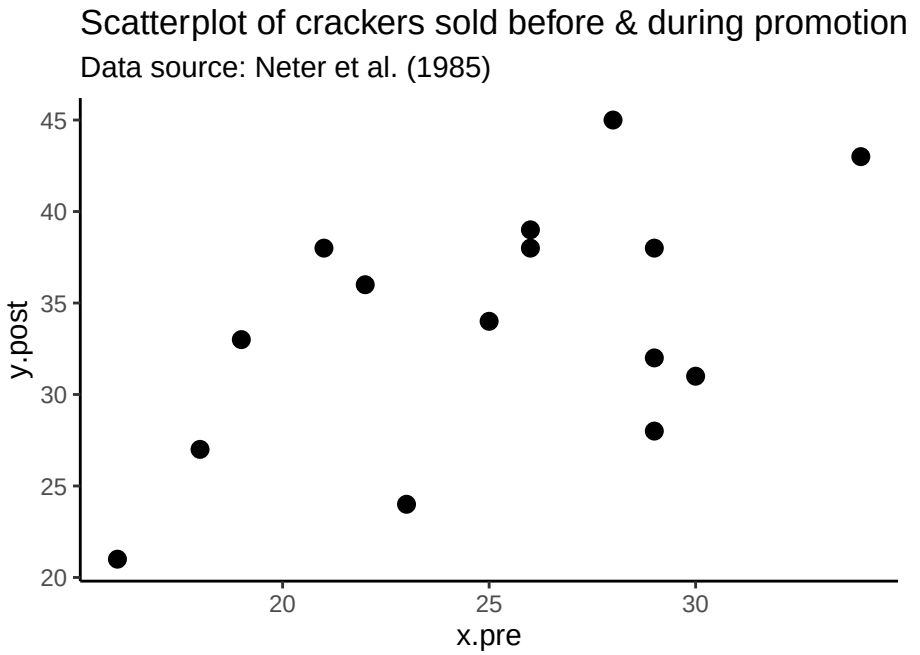

Figure 5.3: Revised scatterplot of Figure 5.2, with titles, and a classic theme

5.2.2 *Different Types of Plots*

Table 5.2 summarizes various plot types commonly used in Excel, and shows the R functions you would need to create them. The third column entitled "Section Number" provides for each type of graph, the section number where you can learn how it to create it. In these sections, you will see concrete examples of script files and a detailed discussion on how `ggplot2`'s functions build each graph and change its appearance. The `ggplot2` package offers the `ggsave()` function, which allows you to easily save your graph in various formats, including the `PDF` format.

In Excel, you have Column charts and Barcharts. Excel's barcharts are known in `ggplot2` as Horizontal barcharts, whereas Excel's Column charts are called Vertical barcharts in `ggplot2`'s lingo. Most graphs listed in Table 5.2 are expected to be familiar to most Excel users. Only continuous curves

discussed in section 5.6 are not often created in Excel. With `ggplot2`, you only need to describe the curve mathematical equation for it to be created.

Table 5.2 : List of Plot Types and Associated Functions

Plot Type	GGPlot2 Function	Section Number
Create a new plot	`ggplot()`	5.2
Scatterplot	`geom_point()`	5.3
Line Plot	`geom_line()`	5.3.2
Connected Scatterplot	`geom_line()` + `geom_point()`	5.3.3
Barcharts	`geom_bar()`	5.4[a]
Vertical / counts	`geom_bar()`	5.4.1
Vertical / percentages	`geom_bar()`	5.4.2
Horizontal / percentages	`geom_bar` + `coord_flip()`	5.4.3
Combined 2 charts	(`Figure1`+`Figure2`)	5.4.4
Stacked barcharts	`geom_bar`	5.4.5
Dodged barcharts	`geom_bar`	5.4.6
Pie Charts	`coord_polar()`	5.5
2D Pie Chart	`coord_polar()`	5.5.1
3D Pie Chart	`pie3D()`	5.5.2
Continuous curve	`geom_function()`	5.6

[a]Vertical & Horizontal Bar Charts

5.3 Scatterplot and Lineplots

In this section, I will show how you can create a scatterplot and a lineplot in R. You will also see how to save it in an external file. Scatterplots and lineplots are treated in this same section because of their similarities in appearance.

A typical scatterplot displays the relationship between 2 variables in the

form of a cloud of bullet points. Is that relationship linear? Quadratic? Increasing or decreasing? The primary objective of a scatterplot is to help you make a visual assessment of the nature of the relationship between the x and y variables.

The lineplot on the other hand, is a graph where the different (x, y) points are connected with a straight line. It helps you study how the y variable changes as x increases. Your primarily goal is to identify trends in your data, not the nature of the relationship between variables.

In section 5.3.1, you will see the creation of simple scatterplots in R with `ggplot2`. Examples will also show you how various plot elements can be customized to give the appearance you want to your graph. Section 5.3.2 is devoted to dot-free lineplots, where only a series of interconnected segment lines display the trend in your data. Finally, I will discuss connected scatterplots in section 5.3.3.

5.3.1 *Simple Scatterplots*

Figure 5.4 shows a scatterplot of the number cracker boxes sold before versus during a promotional campaign, with descriptive axis labels and a classic theme. Although the number of data points used to create it is small, one can see an upward trend that suggests that the promotional campaign may have increased the number of boxes of crackers sold.

I created Figure 5.4 by running Script 5.3. Let me review this script more closely to see what it does. You are already familiar with the first 2 lines, which load the `tidyverse` package and read the `csv` file `crackers.csv` into the `crackers.df` data frame respectively. Line `#03` creates the initial blank `ggplot` object waiting to receive one or several layers of graphs.

In line `#05`, I use the `geom_point()` function to add a scatterplot layer to `ggplot`. Look at the 3 arguments used with the `geom_point()` function. `shape=16` selects point shape `#16`, which is a bullet point as shown in Figure

5.5. Feel free to choose another point shape from this figure. The second argument (`size=3`) specifies the shape size, the default value of which is 1.5. You can increase or decrease it to change the bullet point's appearance. The third argument `color="black"` selects the black color for the bullet points[2].

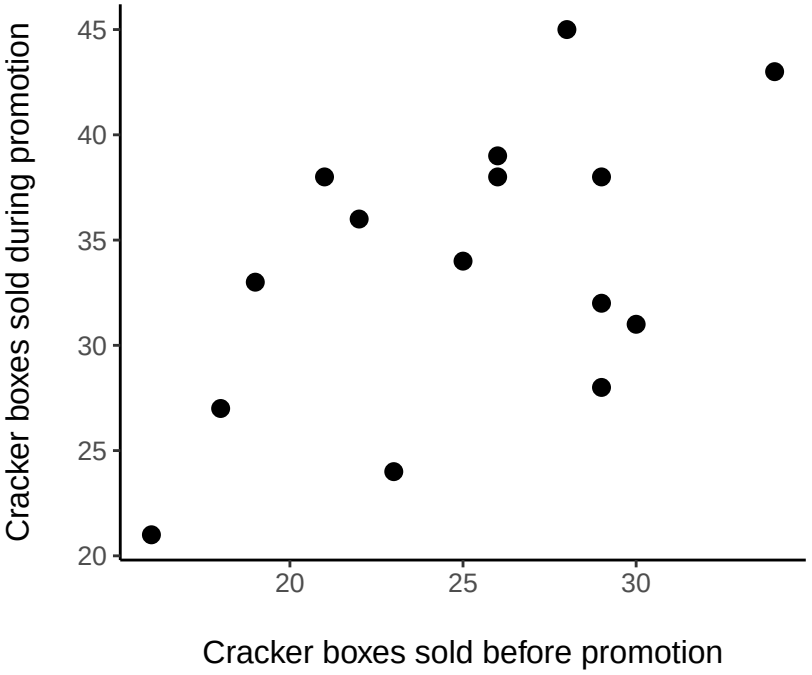

Figure 5.4: Scatterplot of cracker boxes sold before and during promotion, with descriptive axis labels and classic theme

Line #06 indicates that the graph will be based on the classic theme. If you are interested in other themes, check the link https://ggplot2.tidyverse.org/reference/ggtheme.html, where you will see many other options.

After choosing what theme you want for your graph, you need to decide which of the many theme elements you want to customize. You can modify plot titles, axis labels, fonts, background, gridlines, and legends. In line #07,

[2]To see the other color options, run the command `> colors()` on RStudio console

I requested all non-data-related text elements on the graph to have a font size of 12, and a black color. There are so many ways you can customize these elements. For more options, check the link https://ggplot2.tidyverse.org/reference/theme.html.

In lines #08 and #09, I use the `labs()` function to modify the `x` and `y` labels. Alternatively, I could have used 2 functions `xlab("\nCracker boxes sold before promotion\n")` and `ylab("\nCracker boxes sold during promotion\n")` to label the 2 axes. More generally, the `labs()` function is used to write axis labels, legend captions, and plot titles, as you will see in subsequent examples in this section. For more information on the use of this function, check the link https://ggplot2.tidyverse.org/reference/labs.html.

In lines #10 and #11, I use the `ggsave()` function to export the graph currently in display to a PDF file. The use of this function is intuitive. Note that, it recognises the extensions eps/ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg and wmf (windows only). Simply change the file extension from `.pdf` to something elese, and the `ggsave()` function will take care of the rest.

Script 5.3. Basic scatterplot of *y.post* versus *x.pre* with classic theme

```
01 library(tidyverse)
02 crackers.df <- read_csv("./data/chap5/crackers.csv")
03 sp <- ggplot(data=crackers.df,aes(x=x.pre,y=y.post))
04 sp +
05   geom_point(shape=16,size=3,color="black") +
06   theme_classic() +
07   theme(text = element_text(size = 12, colour="black")) +
08   labs(x="\nCracker boxes sold before promotion\n",
09        y="\nCracker boxes sold during promotion\n")
10 ggsave(filename="./images/chap5/crackers.pdf", width = 4.5,
11   height = 3.8,units = "in")
```

End of Script

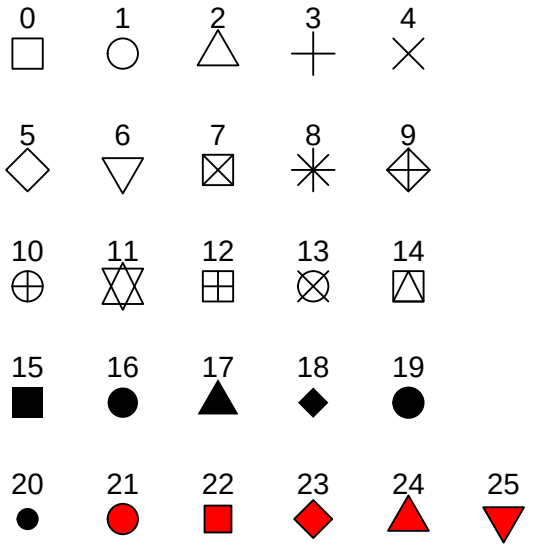

Figure 5.5: Point Shapes Available in R.

> *Note that in Figure 5.5, the point shape numbers 21 through 25 are empty, and you can use them to fill in any color you want. For example, the "fill=" argument of the geom_point() function will only work with these 5 point shapes.*

Figure 5.6 is similar to Figure 5.4, the only difference being the point shapes. In Figure 5.6, the point shapes used are empty diamonds (`shape=23`) filled in blue with red borders. This graph was created by running Script 5.3 after replacing line `#05` with the following:

```
05 geom_point(shape=23,size=5,color="red",fill="blue")
```

This new line `#05` will work with any of the `shape` numbers 21, 22, 23, 24, and 25.

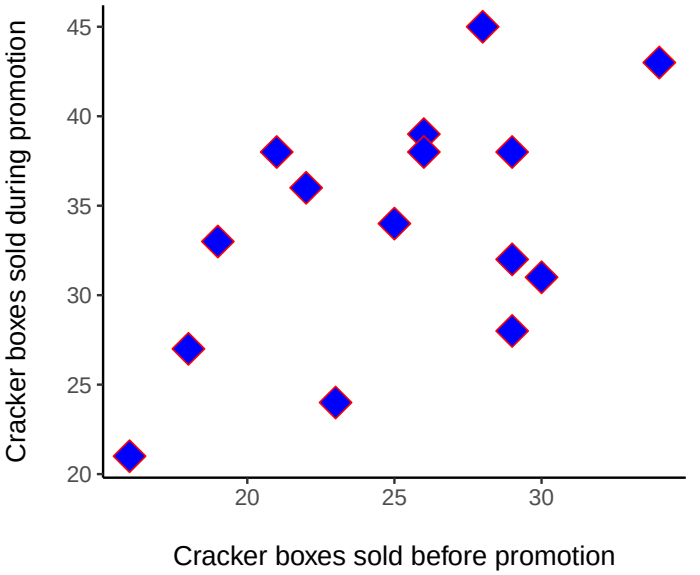

Figure 5.6: Scatterplot of cracker boxes sold before and during a promotional campaign, with an empty diamond point shape filled in blue with red borders.

Scatterplot with Several Theme Elements Customised

Figure 5.7 depicts another version of the cracker sales scatterplot with more customization of theme elements. While the classic theme is still used, you can notice that arrows have been added at the end of both axes. Such axes, which I personally like, are common in many graphs. Moreover, general titles, axis labels, tick labels now have different colors. The x axis tick labels have also been rotated and are printed at different intervals. Bullet points on the graph, which are associated with stores with a post-promotion sale volume below 25 or above 40 have been flagged with the store number.

Script 5.4 shows you all the commands you need to create such a graph. You can download this script file (see the script title), and modify it if needed. I will not review this script line by line. Instead, I will select specific lines of code and explain what they are needed for.

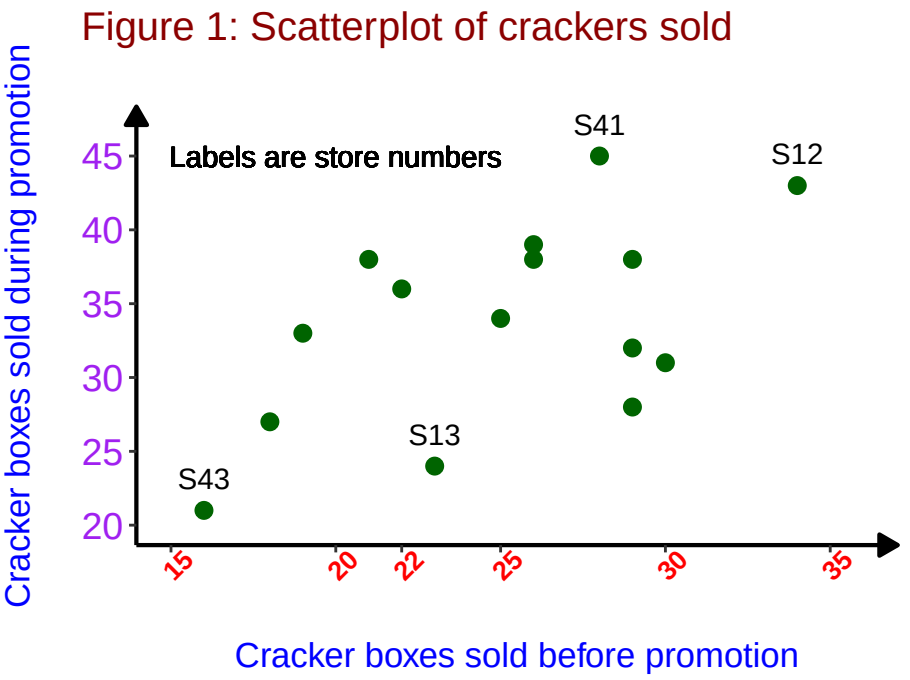

Figure 5.7: Scatterplot of cracker boxes sold with additional formatting.

Here is a review of the most important lines of code of Script 5.4:

- In line `#03`, I needed to create the `extreme.df` data frame. It contains only records where post-promotion sales exceeds 40 boxes or fall below 25 boxes. This dataset is needed to single out those stores and flag them with store numbers as shown in Figure 5.7.
- In line `06`, I added the scatterplot with dark-green points.
- Line `#07` is where stores with extreme sales numbers are flagged. Note that the `data=extremes.df` argument is specified, which means that it is that dataset that will be displayed. However, the aesthetics x and y are not specified. Since they are required in the `geom_text()` function, they will be inherited from the `ggplot()` call of line `#04` (i.e. `x=x.pre,`

`y=y.post`). The `label` aesthetics is also required `geom_text()`. *Each time you use geom˙text(), remember that x, y, and label are 3 required aesthetics to be specified as argument with the aes() function.*

With the `geom_text()` function, I also used the argument `vjust=-1` to slightly move the labels upwards for a better display.

- In lines `#08`, the `geom_text()` is used again to display the text `"Labels are store numbers"`.
- Lines `#11` through `#21` show you how you can use `theme()` to customize almost any component of your plot. I only used a few arguments that this powerful function can take. These arguments are `text=` (for general text elements), `axis.text.x=` (for tick labels along the x-axis), `axis.text.y=` (for tick labels along the y-axis), `plot.title=` (for general plot title), and `axis.line=` (for axis lines). If you want to do something special not considered here, check the link https://ggplot2.tidyverse.org/reference/theme.html for more information regarding the `theme()` function.
- The 2 functions `scale_x_continuous()` and `scale_y_continuous()` are used in lines `#22-#25` to determine what tick points should be displayed on the axes.

Script 5.4. Basic scatterplot of *y.post* versus *x.pre* with customized theme elements. *Download this script:* ***https://bit.ly/3IQL8z7***

```
01 library(tidyverse)
02 crackers.df <- read_csv("./data/chap5/crackers.csv")
03 extremes.df <- filter(crackers.df,y.post>40 | y.post<25)
04 sp <- ggplot(data=crackers.df,aes(x=x.pre,y=y.post))
05 sp +
06    geom_point(shape=16,size=3,color="darkgreen") +
07    geom_text(data=extremes.df,aes(label=storeno),vjust=-1) +
08    geom_text(aes(x=20,y=45,label="Labels are store numbers"),
```

```
09              color="black") +
10    theme_classic() +
11    theme(text = element_text(size = 13, colour="blue"),
12          axis.text.x= element_text(face="bold",colour="red",
13                                    angle=45),
14          axis.text.y= element_text(colour="purple",size=14),
15          plot.title = element_text(size = 15,color="darkred",
16                                    hjust=-0.5),
17          axis.line = element_line(
18            size=0.75,
19            linetype = "solid",
20            color = "black",
21            arrow = arrow(type="closed",length=unit(8,'pt')))) +
22    scale_x_continuous(limits = c(15,36),
23                       breaks = c(15,20,22,25,30,35))+
24    scale_y_continuous(limits = c(20,47),
25                       breaks = seq(20,45,5)) +
26    labs(title = "Figure 1: Scatterplot of crackers sold\n",
27         x="\nCracker boxes sold before promotion\n",
28         y="\nCracker boxes sold during promotion\n")
29  ggsave(filename="./images/chap5/crackersfancy.pdf",
30         width = 5.0,height = 3.8,units = "in")
```

End of Script

Scatterplot with Conditional Color

In the previous example, the focus was on customizing various elements of the graph. All points on the scatterplot were colored in dark green. In the next example, you will see how the color associated with the bullet points can be made conditional upon the promotion program `P1`, `P2` or `P3`.

Figure 5.8 shows the same scatterplot of number of cracker boxes sold before and after a promotional program. The main difference from previous graphs is the bullet point color that now identifies the promotional program associated with the store, and the associated legend located on the right side of the graph.

This graph was created by running Script 5.5.

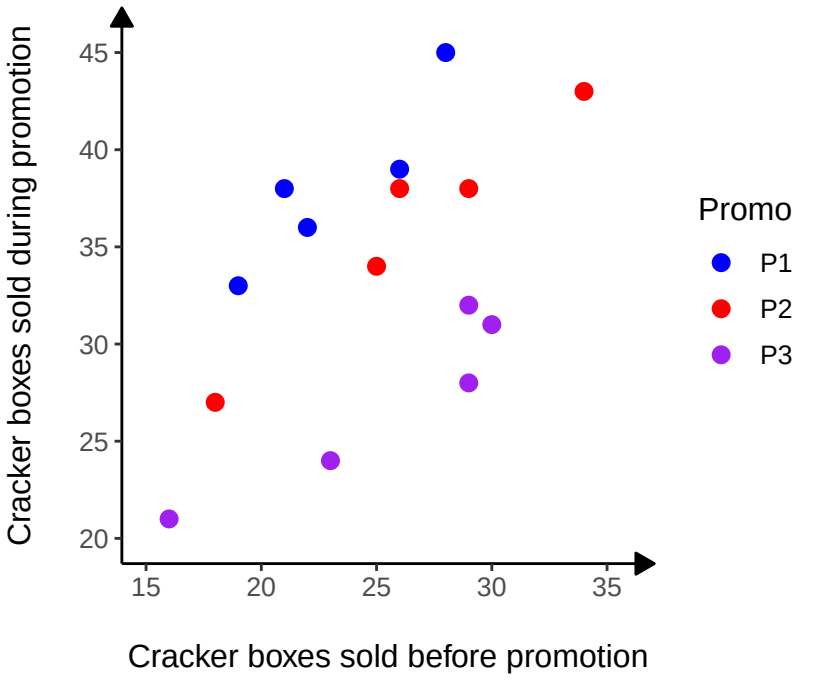

Figure 5.8: Scatterplot of cracker boxes sold: analysis by promotion.

Let me briefly review some key lines of code of Script 5.5. You are encouraged to download this script file and to play with it in order to master the functions that were used.

- Line `#06` is where the `geom_point()` function is used to add the scatterplot layer to the `ggplot` object `sp`. Take a closer look at the `color` aesthetic in `aes(color=promotion)`. Here is where you tell `ggplot2` to use a different color for each value of the `promotion` variable.

- In lines `#13-#15`, I use the `labs()` function, whose *role is to change axis labels as well as legend titles*. Since I requested a different color by promotion in line `#06`, then a legend will automatically be added to the plot, and `"Promotion"` will be the legend title. However, I wanted to use

a shorter name for the legend title. Therefore, I renamed the legend as `"Promo"` in line `#15`.

- Sometimes, colors randomly selected by default do not look good. You can manually overwrite them as I did in line `#16`.

Script 5.5. Basic scatterplot of $y.post$ versus $x.pre$.
(Download this script: https://bit.ly/3yY4H40)

```
01  #-- crackers data: scatterplot analysis by promotion ---
02  library(tidyverse)
03  crackers.df <- read_csv("./data/chap5/crackers.csv")
04  sp <- ggplot(data=crackers.df,aes(x=x.pre,y=y.post))
05  sp +
06    geom_point(shape=16,size=3,aes(color=promotion)) +
07    theme_classic() +
08    theme(text = element_text(size = 12, colour="black"),
09          axis.line = element_line(
10            arrow = arrow(type="closed",length = unit(8,'pt')))) +
11    scale_x_continuous(limits = c(15,36)) +
12    scale_y_continuous(limits = c(20,46),breaks=seq(20,45,5)) +
13    labs(x="\nCracker boxes sold before promotion\n",
14         y="\nCracker boxes sold during promotion\n",
15         color="Promo") +
16    scale_color_manual(values = c("blue","red","purple"))
17  ggsave(filename="./images/chap5/crackers2promo.pdf",
18         width = 4.5, height = 3.8,units = "in")
```

End of Script

5.3.2 *Lineplots*

The lineplot also known as line chart or line graph, is primarily used to display the evolution of one or several numeric y-variables as the reference x-variable increases. Since successive data points are connected by straight line segments, it is essential for the dataset to be sorted in ascending order of the x

variable. Otherwise, the lineplot will not be readable. In `ggplot2`, the sorting is generally done automatically. No need to worry about it.

You may have a single lineplot or multiple lineplots on the same figure. I will now show how both types of graphs can be created.

Single lineplots on a figure

Figure 5.9 shows a lineplot based on the cracker boxes sales data of Table 5.1 . Lineplots are best used for depicting time series, where trends can be interpreted in a more naturally way. You can see that post-promotion sales tend to be higher than the pre-promotion levels. The improvement is sometimes marginal, but can be significant most of the times.

Script 5.6 contains the R code used for creating this simple lineplot. This script file uses the `geom_line()` function to add a line plot to the `ggplot` object. It is the only new function used in this script file that has not been covered before. The argument `size=1` defines the line size (you may increase or decrease it) and the aesthetics needed are inherited from the `ggplot()` call.

Script 5.6. Basic line plot of *y.post* (Cracker boxes sold during promotion) versus *x.pre* (Cracker boxes sold before promotion).
(Download this script: https://bit.ly/3yZxtRw)

```
01 library(tidyverse)
02 crackers.df <- read_csv("./data/chap5/crackers.csv")
03 sp <- ggplot(data=crackers.df,aes(x=x.pre,y=y.post))
04 sp +
05   geom_line(size=1) +
06   theme_classic() +
07   theme(text = element_text(size = 12, colour="black"),
08 axis.line = element_line(
09   arrow = arrow(type="closed",length = unit(8,'pt')))) +
10   scale_x_continuous(limits = c(15,36)) +
11   scale_y_continuous(limits = c(15,52),breaks=seq(15,50,5)) +
```

```
12   labs(x="\nCracker boxes sold before promotion\n",
13    y="\nCracker boxes sold during promotion\n")
14 ggsave(filename="./images/chap5/lineplot.pdf", width = 4.5,
15    height = 3.8,units = "in")
```

End of Script

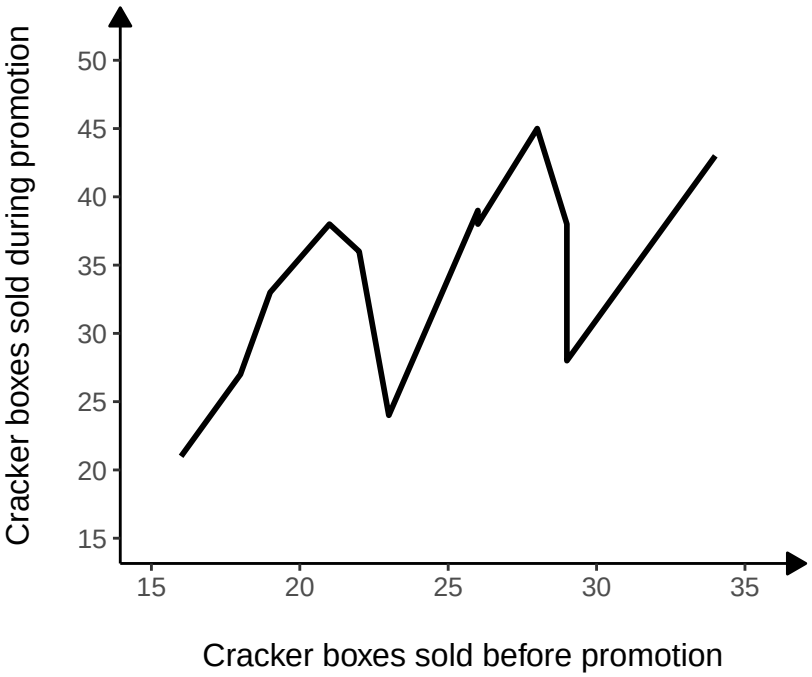

Figure 5.9: Line plot of cracker boxes sold.

Multiple lineplots with a legend on the same figure

Table 5.3 show strength measurements of an adhesive at different pressure and temperature levels. Figure 5.10 displays 3 lineplots (one for each temperature level) showing the measurements as a function of pressure. Script 5.7 shows you the code I used to create this figure.

Table 5.3 : Shear strength of an adhesive at different pressure levels and different temperatures[a]

	Temperature (°F)		
Pressure (lb/in.^{2})	250	260	270
120	9.60	11.28	9.00
130	9.69	10.10	9.57
140	8.43	11.01	9.03
150	9.98	10.44	9.80

[a]Source: Douglas C. Montgomery (1984). *Design and Analysis of Experiments*, 2nd Edition, John Wiley & Sons.

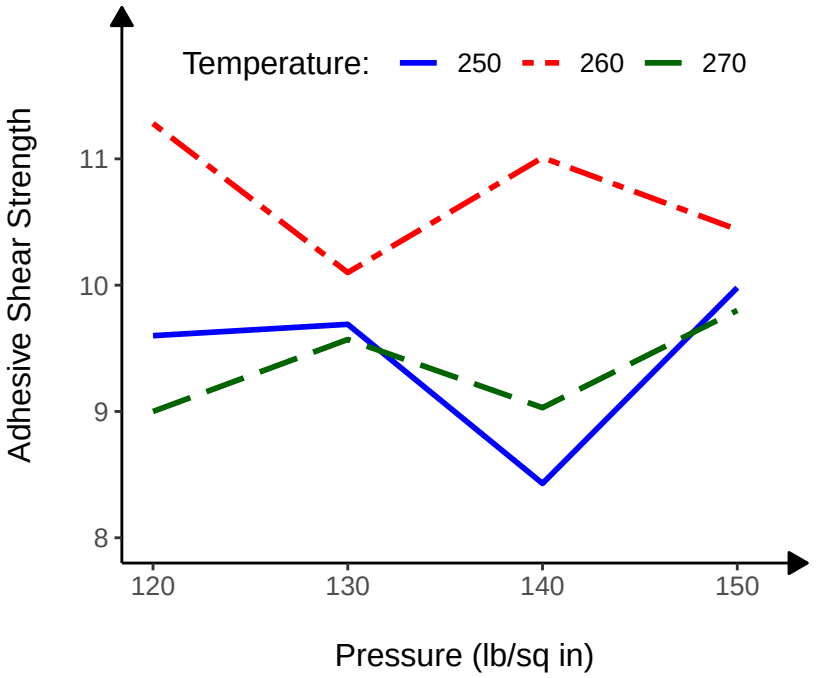

Figure 5.10: Line plots of cracker boxes sold: comparing post-promotion sales to the average of pre- and post-promotion sales.

I am now going to review some key lines of code from Script 5.7 to see how

Figure 5.10 is created.

- In line `#05`, I define the 3 types of lines I will use for the 3 lineplots. See Figure 5.11 for a more comprehensive list of line types available to you.
- In line `#06`, I define the 3 colors associated with the 3 lineplots. The ordering of colors must match that of line types in line `#05`. That is, it is the `solid` line that is blue, the `twodash` is red, and the `longdash`, darkgreen.
- What is being done in lines `#07-#09` is essential for a successful creation of multiple lineplots. You need to reshape your dataset in the long format as follows:

```
pressure temperature strength
  <dbl>     <chr>       <dbl>
   120       250         9.6
   120       260        11.3
   120       270         9
   130       250         9.69
   130       260        10.1
   130       270         9.57
   140       250         8.43
   140       260        11.0
   140       270         9.03
   150       250         9.98
   150       260        10.4
   150       270         9.8
```

  Luckily for you, the `tidyverse` package offers the `pivot_longer()` function that does that. This function was discussed extensively in section 4.2.5 of chapter 4. Note that the x variable will be `pressure`, whereas the y variable will be `strength`, and `temperature` will determine a particular lineplot.

- In lines `#12` and `#13`, the `geom_line()` function is used to display the 3 lineplots. Here is where you specify the aesthetics that tell `ggplot2`

that the line type and color must vary by temperature. Not wanting to leave it up to `ggplot2` to determine the line type and colors, I used the `scale_linetype_manual()` and `scale_color_manual()` functions in lines `#14` and `#15` respectively to impose my own choices. My line types are defined by the `mylinetypes` variables, whereas the colors are defined by the `mycolors` variable.

- Line `#20` is where I decided that the legend will be positioned horizontally. Line `#21` is where I define the legend box location on the figure. `legend.position = c(0.50,0.90))` means the legend box x coordinate is `x`=0.5×`(x-axis width)` and its y coordinate is `y`=0.90×`(y-axis height)`.

Script 5.7. Script file for creating the multiple lineplots of Figure 5.10. *(Here is the download link: https://bit.ly/3uZIjWE)*

```
01 #-- Adhesive shear strength data: multiple line plots  ---
02
03 library(tidyverse)
04 strength.df <- read_csv("./data/chap5/shearstrength.csv")
05 mylinetypes <- c("solid","twodash","longdash")
06 mycolors <- c("blue","red","darkgreen")
07 strength.df1 <- strength.df %>%
08   pivot_longer(cols = c('250','260','270'),
09                names_to = "temperature",values_to = "strength")
10 sp <- ggplot(data=strength.df1,aes(x=pressure,y=strength))
11 sp +
12   geom_line(size=1, aes(linetype=temperature,
13                         color=temperature)) +
14   scale_linetype_manual(values = mylinetypes) +
15   scale_color_manual(values = mycolors) +
16   theme_classic() +
17   theme(text = element_text(size = 12, colour="black"),
18         axis.line = element_line(
19           arrow = arrow(type="closed",length = unit(8,'pt'))),
20         legend.direction = "horizontal",
```

```
21         legend.position = c(0.50,0.90)) +
22   scale_x_continuous(limits = c(120,152)) +
23   scale_y_continuous(limits = c(8,12),breaks=seq(7,11,1)) +
24   labs(x="\nPressure (lb/sq in)\n",
25        y="\nAdhesive Shear Strength\n",
26        linetype="Temperature: ",color="Temperature: ")
27 ggsave(filename="./images/chap5/mlineplots.pdf", width = 4.5,
28        height = 3.8,units = "in")
```

End of Script

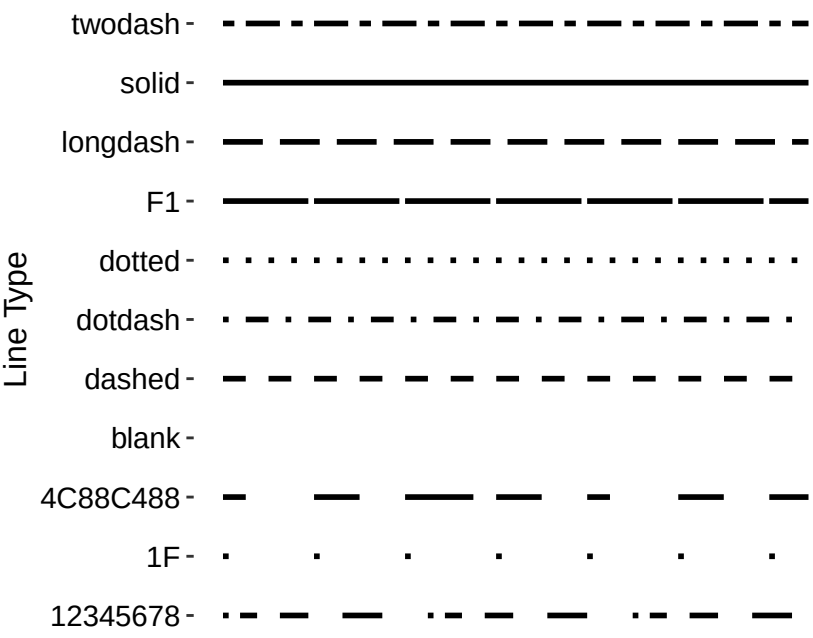

Figure 5.11: Line types available in the `ggplot2` package

Legend position can also be defined as: "left", "top", "right", "bottom", "none." Script 5.7 defined it as a vector `c(x,y)`*, with* $0 \leq x, y \leq 1$*. c(0,0) corresponds to the "bottom left" and c(1,1) to the "top right" position.*

5.3.3 *Connected Scatterplots*

A connected scatterplot is essentially a hybrid plot between a scatterplot and a lineplot. It is obtained by adding a lineplot on top of the scatterplot, and can be seen as a lineplot on which input data points are displayed. The x-variable will generally be sorted in ascending order to obtain a connected scatterplot that can be interpreted[3].

Figure 5.12 shows an example of a connected scatterplot based on the cracker boxes sales data of Table 5.1 . Script 5.8 shows you the lines of code needed to create it. Lines `#04` and `#05` are the 2 lines of code used to create the scatterplot and the lineplot respectively.

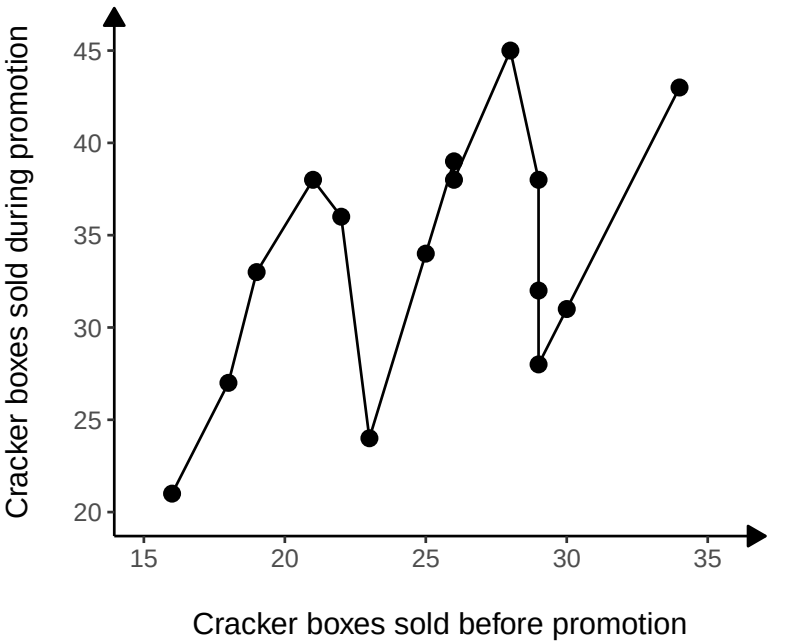

Figure 5.12: Connected scatterplot of cracker boxes sold.

[3]Note that the sorting is automatically done in R, not in Excel

Script 5.8. Connected scatterplot of *y.post* (Cracker boxes sold during promotion) versus *x.pre* (Cracker boxes sold before promotion). Here is the download link: https://bit.ly/3b4fPEp

```
01 crackers.df <- read_csv("./data/chap5/crackers.csv")
02 sp <- ggplot(data=crackers.df,aes(x=x.pre,y=y.post))
03 sp +
04   geom_point(shape=16,size=3) +
05   geom_line() +
06   theme_classic() +
07   theme(text = element_text(size = 12, colour="black"),
08   axis.line = element_line(
09     arrow = arrow(type="closed",length = unit(8,'pt')))) +
10   scale_x_continuous(limits = c(15,36)) +
11   scale_y_continuous(limits = c(20,46),breaks=seq(20,45,5)) +
12   labs(x="\nCracker boxes sold before promotion\n",
13        y="\nCracker boxes sold during promotion\n")
14 ggsave(filename="./images/chap5/cted@scatterplot.pdf",
15        width = 4.5, height = 3.8,units = "in")
```

End of Script

Connected Scatterplot by Promotion

You can also have multiple connected scatterplots on the same figure as shown in Figure 5.13. The color associated with each lineplot was automatically determined by `ggplot` and may not be appealing to you. However, you can manually select these colors (c.f. Figure 5.10).

Script 5.9 shows you the code used to create this figure. Line `#07` creates the scatterplot by promotion, whereas line `#08` adds a new layer of lineplots for each of the 3 promotional programs.