

12 ggplot2 Visualizations in R

While R has become one of the world's leading languages for statistical and data analysis [1, ch.1], its plots are rarely of high-enough quality for publication (see previous chapter). Enter Hadley Wickam's ggplot2, an aesthetically pleasing and logical approach to data visualization based on the **layered grammar of graphics** (see Section 5.4). In this chapter, we introduce ggplot2's **basic elements**, and present some examples illustrating how it is used in practice.

12.1 Basics of ggplot2's Grammar

Four graphical systems are frequently used with R.

- 1. The base graphics system, written by R. Ihaka, is included in every R installation.¹
- 2. The grid graphics system, written by Paul Murrell in 2011, is implemented through the grid package, which offers a lower-level alternative to the standard graphics system. The user can create arbitrary rectangular regions on graphics devices, define coordinate systems for each region, and use a rich set of drawing primitives to control the arrangement and appearance of graphic elements.²
- 3. The lattice package, written by D. Sarkar in 2008, implements trellis graphs, as outlined by W.S. Cleveland [122]. Basically, trellis graphs display the distribution of a variable or the relationship between variables, separately for each level of one or more other variables. Built using the grid package, the lattice package has grown beyond the original approach to visualizing multivariate data and now provides a comprehensive alternative system for creating statistical graphics in R.
- 4. Finally, the ggplot2 package, written by H. Wickham [123], provides a system for creating graphs based on the grammar of graphics described by L. Wilkinson [54] and expanded by Wickham [55]. The intention of the ggplot2 package is to provide a comprehensive, grammar-based system for generating graphs in a unified and coherent manner, allowing users to create new and innovative data visualizations. The power of this approach has led to ggplot2 becoming one of the most common R data visualization tool.

12.1 ggplot2's Grammar	235
Geometries and Aesthetics	236
Types of geoms	238
More About aes	242
12.2 ggplot2 Miscellenea	244
Facets	244
Multiple Graphs	245
Themes	246
Tidy Data	247
Saving Graphs	251
Summary	251
12.3 Examples	252
Algae Bloom Data	252
Gapminder Chart	260
Causes of Mortality	262
Conjugal Status Chart	263
March to Moscow	265
Cholera Outbreak Map	267
Census PUMS Data	268
ggplot2 Recipes	286

1: The graphs produced in Section 11, *Basic Visualizations in R*, rely on base graphics functions.

2: This flexibility makes grid a valuable tool for software developers. But the grid package doesn't provide functions for producing statistical graphics or complete plots. As a result, it is rarely used directly by data analysts and won't be discussed further (see Dr. Murrell's Grid website C²).



Figure 12.1: Schematics of the grammar of ggplot2 graphics [124].

Access to the four systems differs: they are all included in the base installation, except for ggplot2, and they must all be explicitly loaded, except for the base graphics system.

As we saw previously, visualization involves representing data using various elements, such as lines, shapes, colours, etc.. There is a structured relationship – a **mapping** – between the variables in the data and their representation in the displayed plot. We also saw that not all mappings make sense for all types of variables, and (independently), that some representations are harder to interpret than others.

ggplot2 provides a set of tools to map data to visual display elements and to specify the desired type of plot, and subsequently to control the fine details of how it will be displayed. Figure 12.1 shows a schematic outline of the process starting from data, at the top, down to a finished plot at the bottom.

The most important aspect of ggplot2 is the way it can be used to think about the logical structure of the plot. The code allows the user to explicitly state the connections between the variables and the plot elements that are seen on the screen – items such as points, colors, and shapes.

Geometries and Aesthetics

In ggplot2, the logical connections between the data and the plot elements are called **aesthetic mappings**, or simply **aesthetics**, referred to as an aes. After installing and loading the package, a plot is created by telling the ggplot() function what the data is, and how the variables in this data logically map onto the plot's aesthetics.

The next step is to specify what sort of plot is desired (scatterplot, boxplot, bar chart, etc), also known as a geom (short for "plot geometry"). Each geom is created by a specific function:

- geom_point() for scatterplots
- geom_bar() for barplots
- geom_boxplot() for boxplots,
- and so on.

These two components are combined, literally adding them together in an expression, using the "+" symbol. With these, ggplot2 has enough information to draw a plot – the other components (see Figure 12.1) provide additional design elements.

If no further details are specified, ggplot2 uses a set of **sensible default parameters**; usually, however, the user will want to be more specific about, say, the scales, the labels of legends and axes, and other guides that can improve the plot readability.

These additional pieces are added to the plot in the same manner as the geom_function() component, with specific arguments, again using the "+" symbol. Plots are built systematically in this manner, **piece by piece**.

Let's look at some illustrative ggplot2 code:

```
library(ggplot2)
theme_set(theme_bw()) # built-in minimalist theme
```

We create an artificial dataset.



What is going on here? This basic display call contains the following elements:

- ggplot(): creates a plotting object and specifies the data;
- geom_point(): selects a scatter plot as the "geometry" (this is called a "geom" in ggplot2 parlance);
- aes(): specifies the "aesthetic" chart elements a legend is automatically created;
- facet_grid(): specifies the "faceting" or panel layout of the chart.

Other components include **statistics**, **scales**, and **annotation** options. At a bare minimum, charts require a **dataset**, some **aesthetics**, and a **geometry**, combined, as above, with "+" symbols.

This non-standard approach has the advantage of allowing ggplot2 plots to be proper R objects, which can modified, inspected, and re-used (and they are compatible with the tidyverse and pipeline operations).

ggplot2's main plotting functions are qplot() and ggplot(); qplot() is short for "quick plot" and is meant to mimic the format of base R's plot(); it requires less syntax for many common tasks, but has limitations – it's essentially a **wrapper** for ggplot(), which is not itself that complicated to use. We will focus on this latter function.

Types of geoms

Whereas ggplot() specifies the data source and variables to be plotted, the various geom functions specify how these variables are to be visually represented (using points, bars, lines, and shaded regions).

There are currently 35+ available geometries. The tables below list the more common ones, along with frequently used options (most of the graphs shown in this report can be created using those geoms).

Table 12.1: Common ggplot2 geometries and options

Function	Geometries	Options
geom_bar()	bar chart	color, fill, alpha
<pre>geom_boxplot()</pre>	boxplot	color, fill, alpha, notch, width
<pre>geom_density()</pre>	density plot	color, fill, alpha, linetype
<pre>geom_histogram()</pre>	histogram	color, fill, alpha, linetype, binwidth
geom_hline()	horizontal lines	color,alpha,linetype,size
geom_line()	jittered points	color, size, alpha, shape
<pre>geom_jitter()</pre>	line graph	color, alpha, linetype, size
<pre>geom_point()</pre>	scatterplot	color, alpha, shape, size
geom_rug()	rug plot	color,side
geom_smooth()	fitted line	<pre>method, formula, color, fill, linetype, size</pre>
<pre>geom_text()</pre>	text annotations	many; see the help for this function
geom_violin()	violin plot	color, fill, alpha, linetype
geom_vline()	vertical lines	color, alpha, linetype, size

	Table 12.2: Common ggplot2 geometries options
Option	Specifies
color	colour of points, lines, and borders around filled regions
fill	colour of filled areas such as bars and density regions
alpha	transparency of colors, ranging from 0 (fully transparent) to 1 (opaque)
linetype	pattern for lines (1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash)
size	point size and line width
shape	point shapes (same as pch, with 0 = open square, 1 = open circle, 2 = open triangle, and so on)
position	position of plotted objects such as bars and points; for bars, dodge places grouped bar charts
	side by side, stacked, vertically stacks grouped bar charts, and fill, vertically stacks grouped
	bar charts and standardizes their heights to be equal;
	for points, jitter reduces point overlap
binwidth	bin width for histograms
notch	indicates whether boxplots should be notched (TRUE/FALSE)
sides	placement of rug plots on the graph ($b = bottom$, $l = left$, $t = top$, $r = right$,
	bl = both bottom and left, and so on)
width	width of boxplots

As an example, the next bit of code produces a histogram of the heights of singers in the 1979 edition of the New York Choral Society (Figure and a display of height by voice part for the same data.

```
library(ggplot2)
data(singer, package="lattice")
ggplot(singer, aes(x=height)) + geom_histogram()
```



ggplot(singer, aes(x=voice.part, y=height)) + geom_boxplot()



From the second of those (the boxplots), it appears that basses tend to be taller and sopranos tend to be shorter. Although the singers' gender was not recorded, it accounts for much of the variation seen in the diagram.

Note that only the x variable (height) was specified when creating the histogram, but that both the x (voice part) and the y (height) variables were specified for the boxplot – indeed, geom_histogram() defaults to counts on the y-axis when no y variable is specified.³

We examine the use of some of these options using the Salaries dataset, which contains information regarding the salaries of university professors collected during the 2008–2009 academic year.

```
Salaries = read.csv("Salaries.csv", head=TRUE)
```

Variables include rank (AsstProf, AssocProf, Prof), sex (Female, Male), yrs.since.phd (obvious), yrs.service (ditto), and salary (nine-month salary in US dollars).

3: Each function's documentation contains details and additional examples, but there's a lot of value to be found in playing around with data in order to determine the function's behaviour.



The rank is out of order;⁴ as the salary usually increases with the rank, they can be re-ordered as follows:

4: The progression should run from assistant to associate to regular professor.



The chart displays notched boxplots of salary by academic rank. The actual observations (teachers) are **overlaid** and given some transparency so they don't obscure the boxplots.

They are also **jittered** to reduce their overlap. Finally, a rug plot is provided on the left to indicate the general spread of salaries. We see that the salaries of assistant, associate, and full professors differ significantly from each other (there is no overlap in the notches).

Additionally, the variance in salaries increases with greater rank, with a larger range of salaries for full professors. In fact, at least one full professor earns less than all assistant professors. There are also three full professors whose salaries are so large as to make them **outliers** (as indicated by the black dots in the boxplot to the right).

All in all, we have managed to extract a fair amount of insight from this dataset *via* a fairly simple chart (and a fairly simple ggplot2 procedure).

More About aes

Aesthetics refer to the displayed attributes of the data. They map the data to an attribute (such as the size or shape of a marker) and generate an appropriate legend. Aesthetics are specified with the aes() function.

The aesthetics available for geom_point(), as an example, are:

- x
 y
 alpha
 color
 fill
 shape
- size

ggplot() tries to accommodate the user who has never "suffered" through base graphics before by using intuitive arguments like color, size, and linetype, but ggplot() also accepts arguments such as col, cex, and lty.

The ggplot2 documentation explains what aesthetic options exist for each geom (they're generally self-explanatory).

Aesthetics can be specified within the data function or within a geom; if they're specified **within** the data function call then they apply to all specified geoms in the call.

Note the important difference between specifying characteristics (like colour and shape) **inside** and **outside** the aes() function: those inside it are assigned characteristics **automatically**, based on the data. The characteristics defined outside the aes() function are **not mapped to the data**.

The following example, using the mpg dataset, illustrates the difference.



That is all there is to it! (well, not entirely, as we will see in the next section... but it's close enough).

12.2 ggplot2 Miscellenea

A few concepts will help take the basic ggplot2 charts to the next level.

Facets

5: The same chart, but plotted for various subsets of the data.

In ggplot2, small multiples⁵ are referred to as facets, implemented as:

- facet_wrap()
- facet_grid()

The former plots the panels in the order of the factor levels – when the end of a row is reached, the display wraps to the next one.⁶ The grid layout facet_grid() produces a grid with explicit x and y positions; if there are no observations in some of the factor levels, it produces an **empty plot**.

By default, the panels all share the same x and y axes. Note, however, that the various y-axes are allowed to vary *via* facet_wrap(scales = "free_y"), and that all axes are allowed to vary *via* facet_wrap(scales = free).

To specify the data frame columns that are mapped to the rows and columns of the facets, separate them with a tilde. Usually, only a row or a column is fed to facet_wrap() (what happens if both are fed to that component?).

Going back to the choral example, a faceted graph can be produced using the following code:





6: The number of columns and rows can be specified with nrow and ncol.

The resulting plot displays the distribution of singer heights by voice part; separating the height distribution into their own small, side-by-side plots makes them **easier to compare**.

As a second example, we re-visit the Salaries dataset – there is quite an interesting insight linking salary, sex and yrs.since.phd.⁷

Female Male 200000 rank salary 150000 AssocProf AsstProf Prof 100000 50000 40 20 20 40 yrs.since.phd

Multiple Graphs

In basic R, the graphic parameter mfrow and the base function layout() are used to combine two or more base graphs into a single plot. This approach will not work with plots created with the ggplot2 package, however.

The easiest way to place multiple ggplot2 graphs in a single figure is to use the grid.arrange() function found in the gridExtra package.

The following chunk of code places three ggplot2 charts based on the Salaries dataset onto a single graph.

```
p1 <- ggplot(data=Salaries, aes(x=rank)) + geom_bar()
p2 <- ggplot(data=Salaries, aes(x=sex)) + geom_bar()
p3 <- ggplot(data=Salaries, aes(x=yrs.since.phd, y=salary)) +
            geom_point()
gridExtra::grid.arrange(p1, p2, p3, ncol=3)</pre>
```

7: We forego the library(ggplot2) call from this point on in the interest of readability.



Each graph is saved as an object and then arranged into a single plot via grid.arrange(). 8

Themes

Themes allow the user to control the overall appearance of ggplot2 charts; theme() options are used to change fonts, backgrounds, colours, gridlines, and more. They can be used once or saved and applied to multiple charts.

```
mytheme <- theme(plot.title=element_text(face="bold",</pre>
                  size=14, color="brown"),
    axis.title=element_text(size=10, color="brown"),
    axis.text=element_text(size=9, color="black"),
    panel.background=element_rect(fill="white",color="black");
    panel.grid.major.y=element_line(color="grey",linetype=1),
    panel.grid.minor.y=element_line(color="grey",linetype=2),
    panel.grid.minor.x=element_blank(),legend.position="top")
ggplot(Salaries, aes(x=reorder(rank,salary), y=salary,
                     fill=sex)) +
    geom_boxplot() +
    labs(title="Salary by Rank and Sex",x="Rank",y="Salary")
ggplot(Salaries, aes(x=reorder(rank,salary), y=salary,
                     fill=sex)) +
  geom_boxplot() +
  labs(title="Salary by Rank and Sex",x="Rank",y="Salary") +
  mytheme
```

8: Note the difference between faceting and multiple graphs: faceting creates an array of plots based on one or more categorical variables, but the components of a multiple graph could be completely **independent** plots arranged into a single display.



Adding + mytheme to the plotting statement generates the second graph:

- plot titles are printed in brown 14-point bold;
- axis titles in brown 10pt;
- axis labels in black 9pt;
- the plot area should have a white fill and black borders;
- major horizontal grids should be solid grey lines;
- minor horizontal grids should be dashed grey lines;
- vertical grids should be suppressed, and
- the legend should appear at the top of the graph.

The theme() function allows great control over the look of the finished product (consult help(theme) to learn more about these options).

Tidy Data

ggplot2 is compatible with the **tidyverse** [125]. Social scientists will likely be familiar with the distinction between **wide data** and **long data**:

- in a long format table,⁹ every column represents a different (conceptual) variables, and every row represents an observation,
- in a **wide** format table, some variables are spread out across multiple columns, perhaps along some other characteristic such as the year, say.

Consider, for instance, the WorldPhones dataset, one of R's built-in dataset:

Year	N.Amer	Europe	Asia	S.Amer	Oceania	Africa	Mid.Amer
1951	45939	21574	2876	1815	1646	89	555
1956	60423	29990	4708	2568	2366	1411	733
1957	64721	32510	5230	2695	2526	1546	773
1958	68484	35218	6662	2845	2691	1663	836
1959	71799	37598	6856	3000	2868	1769	911
1960	76036	40341	8220	3145	3054	1905	1008
1961	79831	43173	9053	3338	3224	2005	1076

9: Also called a "tall" dataset.

This dataset records the number of telephones, in thousands, on each continent for several years in the 1950s and 1960s. Each column represents a different continent, and each row represents a different year.

This **wide** format seems like a reasonable way to store data, but suppose that we want to compare increases in phone usage between continents, with time on the horizontal axis. In that case, each point on the plot is going to represent a continent during one year – there are seven observations in each row, which makes it difficult to plot using ggplot2.

Fortunately, the tidyverse provides an easy way to convert this wide dataset into a **long** dataset, by **melting** the data. This can be achieved by loading the third-party package called *reshape2.*¹⁰ The WorldPhones dataset can now be melted from a wide to a long dataset *via* the melt() function. Let's assign the new melted data to an object called WorldPhones.m.¹¹

WorldPhones.m = reshape2::melt(WorldPhones)
str(WorldPhones.m)
head(WorldPhones.m)

'data.frame': 49 obs. of 3 variables: \$ Var1 : int 1951 1956 1957 1958 1959 1960 1961 1951 1956 1957 ... \$ Var2 : Factor w/ 7 levels "N.Amer", "Europe",..: 1 1 1 1 1 1 1 2 2 2 ... \$ value: num 45939 60423 64721 68484 71799 ... Var1 Var2 value 1 1951 N.Amer 45939 2 1956 N.Amer 60423 3 1957 N.Amer 64721 4 1958 N.Amer 68484 5 1959 N.Amer 71799 6 1960 N.Amer 76036

Note that while there were originally 7 columns, there are now only 3:

- Var1 represents the year;
- Var2 the continent, and
- value the number of phones.

Every data cell – every observation – every number of phones per year per continent – in the original dataset now has its own row in the melted dataset.

In 1951, in North America, for instance, there were 45,939,000 phones, which is the same value as in the original unmelted data – the data has not changed, it has just been **reshaped**.

Changing the column names might make the data more intuitive to read:

10: Note that this is not the only way to do so, see [1, sec 1.4].

11: Where the m reminds us that the data has been melted.

```
colnames(WorldPhones.m) = c("Year", "Continent", "Phones")
head(WorldPhones.m)
```

	Year	Continent	Phones
1	1951	N.Amer	45939
2	1956	N.Amer	60423
3	1957	N.Amer	64721
4	1958	N.Amer	68484
5	1959	N.Amer	71799
6	1960	N.Amer	76036

Now that the data has been melted into a long dataset, it is easy to create a plot with ggplot2, with the usual steps of a ggplot() call, but with WorldPhones.m instead of WorldPhones:



We place Year on the x-axis, in order to see how the numbers change over time, while Phones (the variable of interest) is displayed on the y-axis. The Continent factor is represented with colour. A scatterplot is obtained by adding a geom_point() layer.

We can also show trends over time, by drawing lines between points for each continent. This only require a change to geom_line().



Incidentally, one might expect the number of phones to increase exponentially over time, rather than linearly.¹² When that is the case, it can be a good idea to plot the vertical axis on a logarithmic scale.



1956

Year

1959

12: A fair number of observations are clustered at the bottom of the chart.

1e+02

1953

Most of the phone trends still look linear, and the lower values are spotted more easily; for example, it is now clear that Africa has overtaken Central America by 1956.

Note how easy it was to build this plot once the data was in the long format: one row for every point – that is, every combination of year and continent – on the graph.

Saving Graphs

Plots might look great on the screen, but they typically have to be embedded in other documents (Markdown, LaTeX, Word, etc.). They must first be saved in an appropriate format, with a specific resolution and size.

Default size settings can be saved within a .Rmd document by declaring them in the first chunk of code:

```
knitr::opts_chunk$set(fig.width=8, fig.height=5)
```

would tell knitr to produce 8 in. \times 5 in. charts.

Saving charts is quite convenient with ggsave(): options include which plot to save, where to save it, and in what format. For instance,

myplot <- ggplot(data=mtcars, aes(x=mpg)) + geom_histogram()
ggsave(file="mygraph.png", plot=myplot, width=5, height=4)</pre>

saves the myplot object as a 5-inch by 4-inch .png file named mygraph.png in the current working directory. The available formats include .ps, .tex, .jpeg, .pdf, .jpg, .tiff, .png, .bmp, .svg, or .wmf (the latter only being available on Windows machines).

Without the plot= option, the most recently created graph is saved. For instance, the following code would save the mtcars plot (the last plot before the call) to the current working directory (see the ggsave() helf file for additional details):

```
ggplot(data=mtcars, aes(x=mpg)) + geom_histogram()
ggsave(file="mygraph.pdf")
```

Summary

While ggplot2 and the tidyverse have proven popular and user-friendly, they do come with some drawbacks, however: the ggplot2 and *tidyverse* design teams have fairly strong opinions about how data should be visualized and processed. As a result, it can sometimes be difficult to produce charts that go against their design ideals.

13: Although that can prove fairly annoying, it is a really good idea for analysts to maintain and update their code regularly.

14: Please contact us if you discover missing or mis-attributed references.

15: We explored one of its subsets in Section 2.1.

16: Which was not available from CRAN anymore as of publication.

In the same vein, some package updates have been known not to **preserve the functionality** of working code, sending the analysts scurrying to figure how the new functions work, which can cause problems with legacy code.¹³ Still, the **versatility** and overall **simplicity** of **ggplot2** cannot be overstated.

The definitive ggplot2 reference is Wickham, Navarro, and Lin Pedersen's ggplot2: Elegant Graphics for Data Analysis [7] [123]; it contains explanations and examples (some of which we borrowed from), as well as the underlying theory behind ggplot2. Other useful examples and starting points can also be found in [124, 126].

The ggplot2 action flow is always the same: start with data in a table, map the display variables to various **aesthetics** (position, colour, shape, etc.), and select one or more **geometries** to draw the graph. This is accomplished in the code by first creating an object with the basic data and mappings information, and then by adding or layering additional information as needed.

Once this general way of thinking about plots is understood (especially the aesthetic mapping part), the drawing process is simplified significantly. There is no need to think about how to draw particular shapes or colours in the chart; the many (self-explanatory) geom_ functions do all the heavy lifting.

Similarly, learning how to use new geoms is easier when they are viewed as ways to display specific aesthetic mappings.

12.3 Examples

In this section, we provide a copious number of examples; some of them also highlight various aspects of ggplot2 that we did not touch upon in the preceding explanations.

The vast majority of these examples have been modified from already existing references; we have strived to cite the references when the required information was still extent.¹⁴

Algae Bloom Data

This example is adapted off of L. Torgo's excellent Data Mining with R Learning with Case Studies, Second Edition ☑ [31].¹⁵

The ability to monitor and perform early forecasts of various river algae blooms is crucial to control the ecological harm they can cause.

The dataset which is used to train the learning model consists of:

- chemical properties of various water samples of European rivers
- the quantity of seven algae in each of the samples, and
- the characteristics of the collection process for each sample.

It is available in the algae_blooms.csv file, or in Torgo's DMwR package.¹⁶

algae_blooms<-read.csv("algae_blooms.csv",sep=",",header=TRUE)</pre>

We get a sense for the data's structure by calling the ${\tt str()}$ function. 17

17: Evidently, algae_blooms is a data frame with 340 observations of 18 variables each.

str(algae_blooms)

'da	ata.fra	am	e':	340 obs. of 18 variables:
\$	seasor	1 :	chr	"winter" "spring" "autumn" "spring"
\$	size	:	chr	"small" "small" "small"
\$	speed	:	chr	"medium" "medium" "medium"
\$	mxPH	:	num	$8 \ 8.35 \ 8.1 \ 8.07 \ 8.06 \ 8.25 \ 8.15 \ 8.05 \ 8.7 \ 7.93 \$
\$	mn02	:	num	9.8 8 11.4 4.8 9 13.1 10.3 10.6 3.4 9.9
\$	Cl	:	num	60.8 57.8 40 77.4 55.4
\$	N03	:	num	6.24 1.29 5.33 2.3 10.42
\$	NH4	:	num	578 370 346.7 98.2 233.7
\$	oP04	:	num	105 428.8 125.7 61.2 58.2
\$	P04	:	num	170 558.8 187.1 138.7 97.6
\$	Chla	:	num	50 1.3 15.6 1.4 10.5
\$	al	:	num	$0 \ 1.4 \ 3.3 \ 3.1 \ 9.2 \ 15.1 \ 2.4 \ 18.2 \ 25.4 \ 17 \ \ldots$
\$	a2	:	num	0 7.6 53.6 41 2.9 14.6 1.2 1.6 5.4 0
\$	a3	:	num	$0 \ 4.8 \ 1.9 \ 18.9 \ 7.5 \ 1.4 \ 3.2 \ 0 \ 2.5 \ 0 \ \ldots$
\$	a4	:	num	0 1.9 0 0 0 0 3.9 0 0 2.9
\$	a5	:	num	34.2 6.7 0 1.4 7.5 22.5 5.8 5.5 0 0
\$	a6	:	num	8.3 0 0 0 4.1 12.6 6.8 8.7 0 0
\$	a7	:	num	0 2.1 9.7 1.4 1 2.9 0 0 0 1.7

We have seen that basic histograms can be constructed with the hist() function. For instance, the histogram of mnO2 measurements is:

hist(algae_blooms\$mn02)

Histogram of algae_blooms\$mnO2



Let's spruce up this histogram with ggplot2.

```
library(ggplot2)
ggplot(algae_blooms,aes(x=mn02)) +
   geom_histogram(aes(y=..density..)) +
   geom_density(color="blue") +
   geom_rug() +
   ggtitle("Histogram of mn02 among 340 observations") +
   xlab("") +
   ylab("")
```



Histogram of minimum value of O2 among 340 observations

In a nutshell, the code above:

0.1

0.0

- plots mn02 from the algae_blooms dataset ...
- as a histogram, where the vertical axis is the density ...
- on which will be layered a blue density curve ...
- and a rug (or comb) chart showing where the observations actually fall...

- with a title ...
- no x axis label ...
- and no y axis label.

We can do the same for a1.

```
ggplot(algae_blooms,aes(x=a1)) +
   geom_histogram(aes(y=..density..)) +
   geom_density(color="red") +
   geom_rug() +
   ggtitle("Histogram of al among 340 observations") +
   xlab("") + ylab("")
```



Histogram of minimum value of a1 among 340 observations

Due to the pronounced skew in both instances, we see that the normal distribution is not a good fit for either of mn02 or a1.

Now let's take a look at some plots involving NH4.¹⁸



18: We plot NH4 from the algae_blooms, as a boxplot with a rug on which the true values are shown and a horizontal line showing where the mean value of NH4 falls.

We can't really see the structure because of the suspected outliers.

Let us take a look at the distribution from another angle: we display the scatter plot of NH4 against the observation number, add a **solid line** at the mean value of NH4, a dashed line at the mean + sd value of NH4, and a tight dashed line at the median value of NH4.



Only a small number of values find themselves above the 3000 threshold – might these be **outliers**?

If we want to only keep the observations that have values of NH4 below 3000 (roughly all values below the long dashed line above), we obtain the following boxplot.¹⁹

19: Whether this is actually a good strategy or not is a discussion for another time.



It is a bit better than the previous boxplot, to be sure (the box structure has expanded, at the very least), but there still seems to be a few outlying.²⁰

Now we take a look at some of the algae levels: we plot a3 by season in a series of boxplots.

```
ggplot(algae_blooms,aes(x=season,y=a3)) +
geom_boxplot() +
xlab("Season") +
ylab("Algae Level a3")
```



20: Perhaps that is to be expected? How could we find out? [1]

We can re-arrange the factors' order, but it requires a bit of fancy footwork using the forcats'library fct_relevel() function, and dplyr's mutate().

```
library(forcats)
library(dplyr)
algae_blooms = mutate(algae_blooms,
    # factors should appear in the desired order
    size=fct_relevel(size,c("small","medium","large")),
    speed=fct_relevel(speed,c("low","medium","high")),
    season=fct_relevel(season,c("spring","summer",
                          "autumn","winter")))
```

21: A cyclic seasonal pattern is evident.

Note the difference:²¹

```
ggplot(algae_blooms,aes(x=season,y=a3)) +
   geom_boxplot() +
   xlab("Season") +
   ylab("Algae Level a3")
```



Violin plots are cousins of boxplots. Can we use them to get a bit more insight on the a3 trend, say?²²

```
ggplot(algae_blooms,aes(x=season,y=a3)) +
   geom_violin() +
   geom_jitter() +
   xlab("Season") +
   ylab("Algae Level a3")
```

22: What happens if the jitter option is turned off?



We can now take a look at possible interactions for NH4< 3000 and season, binning the observations with respect to the quartiles.²³

```
library(dplyr)
f.NH4.data <- filter(algae_blooms,!is.na(NH4)) |>
    filter(NH4<3000) |> mutate(q.NH4=cut(NH4,quantile(NH4,
           c(0,0.25,0.5,0.75,1)), include.lowest=TRUE))
```

The faceted chart is shown below - is there anything of interest in the chart?

```
ggplot(f.NH4.data,aes(x=a1,y=season,color=season)) +
    geom_point() + facet_wrap(~q.NH4) + guides(color=FALSE) +
    ggtitle("Algae Level a1 by Season and NH4 Quartiles")
```



Algae Level a1 by Season and Ammonium Quartiles NH4

23: We use the dplyr pipeline operator |> (sometimes written as %>%; see [1, sec1.4]).

Gapminder Chart

summary(gapminder_ds)

Southern Europe: 684

:5586

(Other)

The Gapminder dataset is available in R. Irizarry's dslabs package. We'll show how to produce a chart such as the one in Figure 1.4.

```
gapminder_ds <- dslabs::gapminder # for the dataset
library(tidyverse) # to access ggplot2 and dplyr
library(wesanderson) # for the colour palette
library(ggrepel) # for country names on chart
```

We start by getting a summary of the available data in the dslabs package, in particular the available years.

infant_mortality life_expectancy country year Albania ÷ 57 Min. :1960 Min. : 1.50 Min. :13.20 Algeria 1st Qu.: 16.00 1 57 1st Qu.:1974 1st Qu.:57.50 Angola 57 Median :1988 Median : 41.50 Median :67.54 3 Antigua 57 : 55.31 :64.81 3 Mean :1988 Mean Mean 57 3rd Qu.: 85.10 3rd Qu.:73.00 Argentina : 3rd Qu.:2002 Armenia 57 Max. :2016 Max. :276.90 Max. :83.90 1 (Other) :10203 NA's :1453 fertility population gdp continent Min. :0.840 Min. :3.124e+04 Min. :4.040e+07 Africa :2907 1st Qu.:2.200 1st Qu.:1.333e+06 1st Qu.:1.846e+09 Americas:2052 Median :3.750 Median :5.009e+06 Median :7.794e+09 Asia :2679 Mean :4.084 Mean :2.701e+07 Mean :1.480e+11 Europe :2223 3rd Qu.:6.000 3rd Qu.:1.523e+07 3rd Qu.:5.540e+10 Oceania : 684 Max. :9.220 Max. :1.376e+09 Max. :1.174e+13 NA's NA's NA's :2972 :187 :185 region Western Asia :1026 Eastern Africa : 912 Western Africa : 912 Caribbean : 741 South America : 684

Preparing the chart for 2012 would be a nice symmetry (see Figure 1.4), but the version of the data that we have does not contain all the required information for that year, so we will pick 2009 instead, while setting the possibility of **changing the year** if required.

Due to lack of space, we will not be able to label all the countries in the chart; instead, we label only 20 of them, with the probability that a name is selected proportional to the country populations.

The chart is then produced by the code below, using the Darjeeling1 colour palette from the wesanderson package.

```
f.gapminder |>
 ggplot(aes(x = gdppc, y=life_expectancy, size=pop_m)) +
 geom_point(aes(fill=continent), pch=21) +
 scale_fill_manual(values=wes_palette(n=5, name="Darjeeling1")) +
 scale_x_log10() +
 geom_label_repel(aes(label=country_display, size=sqrt(pop_m/pi)),
    alpha=0.9, fontface="bold", show.legend=FALSE,
   min.segment.length = unit(0, 'lines')) +
 gqtitle(chart_title) +
 theme(plot.title = element_text(size=14, face="bold")) +
 xlab('log GDP per capita ($/year)') +
 ylab('Life expectancy (years)') + ylim(45,85) +
 scale_size_continuous(range=c(1,40), limits = c(0, 1500),
    breaks = c(1,10,100,1000), labels = c("1","10","100","1000")) +
 guides(fill=guide_legend(override.aes = list(size=5))) +
 labs(fill="Continent", size="Population (M)") +
 theme_bw() +
 theme(plot.title = element_text(size=16, face="bold"))
```



Health & Wealth of Nations

Florence Nightingale's Causes of Mortality

24: We always assume that the tidyverse package has been loaded from this point on (to have access to all ggplot2 and dplyr functions.

Nightingale's celebrated rose charts (see Figure 1.1) can also be re-created using ggplot2 (with a big assist to N. Saunders' tutorial ^[7]).²⁴

```
library(HistData) # to get access to the Nightingale dataset
Nightingale |>
  select(Date, Month, Year, contains("rate")) |>
  pivot_longer(cols=4:6, names_to="Cause", values_to="Rate") |>
 mutate(Cause = gsub(".rate", "", Cause),
    period = ifelse(Date <= as.Date("1855-03-01"),</pre>
    "April 1854 to March 1855", "April 1855 to March 1856"),
    Month = fct_relevel(Month, "Jul", "Aug", "Sep", "Oct",
    "Nov", "Dec", "Jan", "Feb", "Mar", "Apr", "May", "Jun")) |>
 ggplot(aes(Month, Rate)) +
  geom_col(aes(fill=Cause), width=1, position="identity") +
  coord_polar() + facet_wrap(~period) +
  scale_fill_manual(values=c("skyblue3", "grey30",
  "firebrick")) +
  scale_y_sqrt() + theme_bw() +
  theme(axis.text.x = element_text(size = 9),
        strip.text = element_text(size = 11),
        legend.position = "bottom",
        plot.margin = unit(c(10, 10, 10, 10), "pt"),
        plot.title = element_text(vjust = 5)) +
 ggtitle("Diagram of the Causes of Mortality in the
                                      Army in the East")
```



Diagram of the Causes of Mortality in the Army in the East

W.E.B. Du Bois' Conjugal Status Chart

We follow statswithmatt's tutorial ◻ to recreate another W.E.B. Du Bois classic chart [5].²⁵

```
# creating the data
gender <- c("female", "male")</pre>
status <- c("single", "widowed", "married")</pre>
age_bins <- c("0-15", "15-20", "20-25", "25-30", "30-35",
  "35-45", "45-55", "55-65", "OVER 65")
marital <- expand.grid(age_bins, gender, status)</pre>
names(marital) <- c("age", "gender", "status")</pre>
marital$pct <- c(100, 84, 38, 18, 12, 8, 6, 4, 4, 100, 99,
  66, 30, 18, 10, 6, 4, 4, 0, 0, 4, 8, 10, 16, 28, 44, 66,
  0, 0, 1, 2, 3, 5, 9, 11, 20, 0, 16, 58, 74, 78, 76, 66,
  52, 30, 0, 1, 33, 68, 79, 85, 85, 85, 76)
marital$status <- factor(marital$status,levels = c("widowed",</pre>
    "married", "single"))
marital$age_numeric <- as.numeric(marital$age)</pre>
# creating the chart
ppmsca_33915 <- ggplot(data=marital, mapping = aes(</pre>
    x = age_numeric, y = ifelse(gender == "male", -pct, pct),
    fill=status)) +
  geom_bar(stat = "identity", width = 1) +
  scale_x_continuous(breaks = (1:9) + 0.5, labels = age_bins,
```

25: For the historical examples, it would be a great idea to take the time to explore the data before trying to plot it.

```
expand = c(0, 0), sec.axis = dup_axis()) +
  scale_y_continuous(breaks = seq(-100, 100, by = 10),
      labels = abs, expand = c(0, 0),
      minor_breaks = seq(-100, 100, by = 2)) +
  scale_fill_manual(values=c("seagreen4", "firebrick3",
  "royalblue3"),
    labels = c("WIDOWED", "MARRIED", "SINGLE")) +
  labs(title = "Conjugal condition of American Negroes
                according to age periods",
    subtitle = "Done by Atlanta University", x = "AGES",
                y = "PERCENT") +
  coord_flip(clip = "off")
# annotating the chart
ppmsca_33915 + annotate("text",label = rep(c("SINGLE",
       "MARRIED", "WIDOWED"), each = 2),
   y = c(-35, 35, -55, 55, -92, 92),
    angle = c(45, -45, 45, -45, 60, -60),
    x = c(2, 2, 6, 6, 8.5, 7.5),
    size = c(4, 4, 4, 4, 3, 3),
    fontface = "bold") +
  annotate("text", label = c("MALES", "FEMALES"),
    y = c(-50, 50), x = Inf,
    vjust = -0.4, size=2.5, fontface = "bold") +
  theme(text = element_text(face = "bold"),
    panel.background = element_blank(),
    plot.title = element_text(size = 8, vjust = 2),
    plot.subtitle = element_text( size = 6, vjust = 2),
    axis.title = element_text(size = 8),
    axis.ticks = element_blank(),
    panel.grid.major = element_line(color = "black",
      size = 0.1),
    panel.grid.minor.x = element_line(color = "black",
      size = 0.05),
    panel.grid.minor.y = element_blank(),
    legend.background = element_blank(),
    legend.position = "none", legend.key = element_blank(),
    panel.ontop = TRUE,
    panel.border = element_rect(fill=NA, color = "black"),
    axis.text.x = element_text(size = 8),
    axis.title.y = element_text(angle = 0, vjust = 1),
    axis.title.y.right = element_text(angle = 0, vjust = 1),
    axis.text.y = element_text(vjust = 2,size=8))
```



Minard's March to Moscow

We recreate Minard's famous *March to Moscow* chart (see Figure 1.1 and Section 1.4) by following very closely the code provided in M. Friendly's tutorial ²⁷.²⁶

```
# getting the data
data(Minard.troops,Minard.cities,Minard.temp,package="HistData")
# required packages
library(ggplot2)
library(scales)
                       # additional formatting for scales
library(grid)
                       # combining plots
library(gridExtra)
                       # combining plots
library(dplyr)
                       # tidy data manipulations
# the troops/cities upper chart
breaks <- c(1, 2, 3) * 10<sup>5</sup>
plot.troops.cities <- Minard.troops |> ggplot(aes(long, lat)) +
  geom_path(aes(size = survivors, colour = direction,
                group = group), lineend="round") +
  scale_size("Survivors", range = c(0,20),
             breaks=breaks, labels=scales::comma(breaks)) +
  scale_colour_manual("Direction",
                     values=c("#E80000", "#1F1A1B"),
                     labels=c("Advance", "Retreat")) +
  ylim(53.8,56.0) + coord_cartesian(xlim = c(24, 38)) +
  labs(x = NULL, y = NULL) + theme_bw() +
  guides(color = "none",size="none") +
```

26: We would have had a hard time coming up with this stuff on our own, not going to lie...

```
geom_point(data=Minard.cities, size=10, pch=22,
             color = "black", fill="gray") +
  geom_label_repel(data=Minard.cities, aes(label = city),
                   size = 3, vjust = 0.5, alpha=0.8)
# replacing a missing value in the temperature data
Minard.temp$date = factor(Minard.temp$date,
  levels=c(levels(Minard.temp$date), "Unknown"))
Minard.temp$date[is.na(Minard.temp$date)] <- "Unknown"</pre>
# the temperature lower chart
plot.temp <- Minard.temp |>
  mutate(label = paste0(temp, "° ", date)) |>
  ggplot(aes(long, temp)) +
  geom_path(color="grey", size=2, group=1) +
  geom_point(size=1) +
  geom_label_repel(aes(label=label), size=4) +
  coord_cartesian(xlim = c(24, 38)) +
  labs(x = NULL, y="Temperature") +
  theme_bw() + theme(panel.grid.major.x = element_blank(),
   panel.grid.minor.x = element_blank(),
   panel.grid.minor.y = element_blank(),
   axis.text.x = element_blank(), axis.ticks = element_blank(),
   panel.border = element_blank())
# combining both charts
grid.arrange(plot.troops.cities, plot.temp, nrow=2,
    heights=c(3.5, 1.2))
grid.rect(width = .95, height = .95, gp = gpar(lwd = 0,
    col = "white", fill=NA))
```



John Snow's Cholera Outbreak Map

We recreate John Snow's historical map of the 1854 London cholera outbreak (adapted from V.B. Lanzetta's R Data Visualization Recipes ☑) [127].²⁷

```
# getting the data and required packages
data(Snow.streets, Snow.deaths, Snow.pumps, package="HistData")
library(ggplot2)
library(ggrepel)
# street map, death locations, water pump locations
ggplot(data=Snow.streets) +
  geom_path(aes(x=x, y=y, group = street)) +
  geom_point(data=Snow.deaths, aes(x=x, y=y,
             colour="black", shape="15")) +
  geom_point(data=Snow.pumps, aes(x=x, y=y,
             colour="red", shape="16"), size=4) +
  scale_colour_manual("Locations", values=c("black","red"),
                      labels=c("deaths","water pumps")) +
  scale_shape_manual("Locations", values=c(16,15),
                      labels=c("deaths","water pumps")) +
 geom_label_repel(data=Snow.pumps, aes(x=x, y=y, label=label),
        colour="black", size=3, vjust=0.5, alpha=0.8) +
 gqtitle("John Snow's London Cholera Outbreak Map (1854)") +
  theme_bw() +
 theme(plot.title = element_text(size=16, face="bold"),
    axis.title.x=element_blank(), axis.text.x=element_blank(),
    axis.ticks.x=element_blank(), axis.title.y=element_blank(),
    axis.text.y=element_blank(), axis.ticks.y=element_blank())
```

John Snow's London Cholera Outbreak Map (1854)



27: The culprit (contaminated pump) is easily identifiable in this version of the chart.

Census PUMS Data

The following example is taken from N. Zumel and J. Mount's excellent Practical Data Science with R ☑ [128].

The custdata.tsv file is derived from U.S. Census PUMS data. The business objective is to predict whether a customer has health insurance. This synthetic dataset contains customer information for individuals whose health insurance status is **known**.

We start by importing the data into a data frame using the read.delim() function (to handle the odd file format).

```
df <- read.delim(here::here("data", "custdata.tsv"))
dim(df)</pre>
```

[1] 1000 11

Evidently, there are 1000 observations of 11 variables. The first few observations look like:

head(df,5)

Married TRUE
Married TRUE
Married FALSE
Married FALSE
Married TRUE
ve num.vehicles age state.of.res
SE 2 49 Michigar
JE 3 40 Florida
JE 3 22 Georgia
GE 0 22 New Mexico
JE 1 31 Florida

We obtain a data summary *via* summary().

summary(df)

custid	sex	is.employed	income
Min. : 2068	Length:1000	Mode :logical	Min. : -8700
1st Qu.: 345667	Class :character	FALSE:73	1st Qu.: 14600
Median : 693403	Mode :character	TRUE :599	Median : 35000
Mean : 698500		NA's :328	Mean : 53505
3rd Qu.:1044606			3rd Qu.: 67000
Max. :1414286			Max. :615000

```
marital.stat
                   health.ins
                                   housing.type
                                                       recent.move
Length:1000
                   Mode :logical
                                   Length:1000
                                                       Mode :logical
Class :character
                   FALSE:159
                                   Class :character
                                                       FALSE:820
Mode :character
                   TRUE :841
                                   Mode :character
                                                       TRUE :124
                                                       NA's :56
 num.vehicles
                                state.of.res
                     age
Min.
       :0.000
                                Length:1000
                Min.
                       : 0.0
                1st Qu.: 38.0
1st Qu.:1.000
                                Class :character
Median :2.000
                Median : 50.0
                                Mode :character
Mean
       :1.916
                Mean
                       : 51.7
3rd Qu.:2.000
                3rd Qu.: 64.0
Max.
       :6.000
                       :146.7
                Max.
NA's
       :56
```

Right off the bat, we see that there are some problems with the data (NAs, impossible ranges, etc.). We can also provide the number of NAs per variable in an easier-to-read format using the following code block:

```
mv <- colSums(is.na(df))
# cbind(mv) # cbind to display as column
mv</pre>
```

custid sex is.employed income marital.stat health.ins 0 0 328 0 0 0 housing.type recent.move num.vehicles age state.of.res 56 56 56 0 0

The fact that three of the variables have the same (relatively high) number of missing values means that it is possible that the same 56 observations have no measurement for housing.type, recent.move, and num.vehicles.²⁸

As for the ranges, something is definitely fishy with income and age:

```
summary(df$income)
summary(df$age)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-8700	14600	35000	53505	67000	615000
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0	38.0	50.0	51.7	64.0	146.7

What does it mean for incomes to be negative? For a customer to be 0 years old? Or worse, 146.7?

We use ggplot2 to visually explore the data.²⁹ We will begin by providing a number of univariate visualizations, starting with the age variable.

28: That is not a guarantee, of course. We still need to check (try it!).

29: Recall that:

- the ggplot() function works only on data frames and/or tibbles;
- it does not create a graph, it creates an **object**;
- graphs are produced from layers, which are *added* to the object, and
- aesthetics are the visual elements of the graph, e.g., the *x* and *y* variables, the size of markers, colors, etc.





What happens if we use different bin widths?



ggplot(df) + geom_histogram(aes(x=age),binwidth=10,fill="gray")

We definitely seem to find ourselves in a Goldilocks situation.

We can also get some (univariate) information about the income variable:

library(scales)		
<pre>ggplot(df) + geom_histogram(aes(x=income),</pre>	binwidth	= 10000)
<pre>scale_x_continuous(labels=dollar)</pre>		
<pre>ggplot(df) + geom_histogram(aes(x=income),</pre>	binwidth	= 5000) +
<pre>scale_x_continuous(labels=dollar)</pre>		


We have already seen that there are negative income values in the dataset. We will restrict the data to those customers with positive income, and display them using a logarithmic scale.

```
df.2 <- subset(df, income > 0)
ggplot(df.2) +
    geom_histogram(aes(x=income), binwidth = 5000) +
    scale_x_log10(breaks=10^(1:6), labels=dollar)
```



Whoa, that's ... quite possible the most useless chart we have ever seen. The problem here is that binwidth refers to the powers, not to the raw numbers.

```
ggplot(df.2) +
   geom_histogram(aes(x=income), binwidth = 0.05) +
   scale_x_log10(breaks=10^(1:6), labels=dollar)
```



Much better!

A density plot can also provide an estimate of the probability distribution function.





The tail is a bit long/heavy: as before, it might be more illuminating to use a logarithmic scale.



What can we say about the distribution of marital status?



Nothing is too surprising so far (although, as mentioned, something is definitely off with some of the age and income measurements).

If we try to get information about a variable with 10+ levels (state.of.res), we see that the charts can get busy.





The flipped chart is clearly easier to read, due to text overlap in the original chart.

The latter displays the states ordered alphabetically; to order according to the number of observations in each state, we first need to modify the data using transform(), which actually re-orders the levels for state.of.res by population in the dataset (which are presumably representative of the order in the full US population).

```
tbl <- as.data.frame(table(df$state.of.res))
colnames(tbl) <- c("state.of.res","count")
tbl <- transform(tbl,state.of.res=reorder(state.of.res,count))
ggplot(tbl) +
    geom_bar(aes(x=state.of.res, y=count), stat="identity") +
    coord_flip() +
    theme(axis.text.y=element_text(size=rel(0.6)))</pre>
```



How can we find the average number of vehicles per customer in each state? For instance, in Delaware and Alaska, it is:

```
with(df,mean(num.vehicles[state.of.res=="Delaware"],na.rm=TRUE))
with(df,mean(num.vehicles[state.of.res=="Alaska"],na.rm=TRUE))
```

[1] 2[1] 2.333333

(Note the na.rm=TRUE to avoid issues with computations involving observations with no measurement)

We could repeat the process 50 times (once for each state), or we could use either a **split/apply/combine** approach (in base R) or a **tidyverse** approach (using plyr).

```
# split
pieces <- split(df, df$state.of.res)
# apply
result <- lapply(pieces, function(p) data.frame(
    state.of.res=p$state.of.res[[1]],
    state.avg.vehicles=mean(p$num.vehicles, na.rm=TRUE)))
(result <- do.call("rbind", result))</pre>
```

<pre>state.of.res</pre>	state.avg.vehicles	state.of.res	state.avg.vehicles
Alabama	2.100000	Montana	2.500000
Alaska	2.333333	Nebraska	1.500000
Arizona	1.888889	Nevada	2.000000
Arkansas	1.833333	New Hampshire	1.800000
California	2.098901	New Jersey	1.555556
Colorado	1.636364	New Mexico	2.333333
Connecticut	2.000000	New York	1.928571
Delaware	2.000000	North Carolina	1.666667
Florida	1.866667	North Dakota	3.000000
Georgia	1.708333	Ohio	1.836735
Hawaii	1.750000	Oklahoma	1.818182
Idaho	1.666667	Oregon	2.285714
Illinois	2.183673	Pennsylvania	1.938462
Indiana	2.000000	Rhode Island	2.000000
Iowa	2.000000	South Carolina	1.785714
Kansas	1.750000	South Dakota	1.600000
Kentucky	1.933333	Tennessee	1.571429
Louisiana	1.533333	Texas	1.833333
Maine	2.200000	Utah	1.750000
Maryland	2.750000	Vermont	1.666667
Massachusetts	1.833333	Virginia	1.884615
Michigan	1.843137	Washington	2.235294
Minnesota	2.350000	West Virginia	1.666667
Mississippi	1.500000	Wisconsin	1.692308
Missouri	1.950000	Wyoming	2.000000

The tidyverse-like solution is much more elegant, however:

When it comes to *univariate* representations:

- we may consider using a histogram or density plot to look for outliers or incorrect values in numerical variables,
- which will also give a sense of the distribution is it symmetric, normal, lognormal, etc;
- but we may consider using a bar chart to compare frequencies for categorical variables.

We can also look at bivariate charts; perhaps one involving age and income, say. We start by removing the odd observations.

df.3 <- with(df, subset(df, age>0 & age < 100 & income > 0))

We can prepare a scatterplot:

```
ggplot(df.3, aes(x=age, y=income)) +
geom_point() +
scale_y_continuous(labels=dollar)
```



Or colour the dots according to the health insurance status.

```
ggplot(df.3, aes(x=age, y=income, colour = health.ins)) +
geom_point() +
scale_y_continuous(labels=dollar)
```



The relationship between age and income is not linear, so adding the line of best-fit might not provide much in the way of insight, but it can be done nonetheless.

```
ggplot(df, aes(x=age, y=income)) +
  geom_point() +
  geom_smooth(method="lm") +
  scale_y_continuous(labels=dollar)

ggplot(df, aes(x=age, y=income, colour = health.ins)) +
  geom_point() +
  geom_smooth(method="lm") +
  scale_y_continuous(labels=dollar)
```



A heat map (where a cell's colour represents the number of observations in the cell) might be more *à* propos.³⁰

```
ggplot(df, aes(x=age, y=income)) +
geom_bin2d() +
scale_y_continuous(labels=dollar)
ggplot(df, aes(x=age, y=income)) +
geom_bin2d() +
scale_y_continuous(labels=dollar) +
geom_smooth()
```

30: Is the smoothing curve a bit too much, here?



How about a hexbin heat map?

```
library(hexbin)
ggplot(df, aes(x=age, y=income)) + geom_hex(binwidth=c(5, 20000)) +
    scale_y_continuous(labels=dollar) + geom_smooth()
```





Other plots can be produced: here is a plot of health.ins vs. age.

31: As one gets older, one is more likely to get health insurance, we suppose.

That's not really surprising, is it?³¹ It doesn't seem as though there are that many more people with insurance than there are without, but that's because all the observations with the same age are represented by a single dot.

A heatmap could incorporate the number of observations into the picture.



Mmhhh... that's not nearly as insightful as could have been expected. One of the geoms can come in handy: geom_jitter.



Now we can clearly see that there are substantially fewer customers without life insurance.

Why stop at only 2 variables when we could add income to the picture?



ggplot(df.3, aes(x=age, y=health.ins, colour=log10(income))) + geom_jitter(height=0.2) Is there anything insightful in there?

We could also try to link marital status to health insurance status?



ggplot(df) + geom_bar(aes(x=marital.stat, fill=health.ins))

Stacked bar charts are the pie charts of bar charts – it is much better to put the bars side-by-side.





One exception could be made for proportion stacked bar charts:

But we do lose the sense of the size of each sub-categories' population. Some jitter functionality comes to the rescue once again!



Might there be a link between housing type and marital status?

```
ggplot(df.3) + geom_bar(aes(housing.type, fill=marital.stat),
position="dodge")
last_plot() + coord_flip()
ggplot(subset(df.3, !is.na(housing.type))) +
geom_bar(aes(housing.type, fill=marital.stat),
position="dodge") +
theme(axis.text.x=element_text(angle=15))
```





It's easy to see how some fine-tuning can make the charts easier to read (which can only help when it comes to extracting actionable insights).

We end our data exploration of by showing how to build a small multiple chart, by housing.type:





Is there any link with with health insurance status?



ggplot2 Recipes and Examples

We end with examples showcasing additional functionality.³²

Smoothing Lines

```
options(scipen=999) # turn-off scientific notation
theme_set(theme_bw()) # pre-set the bw theme.
data("midwest", package = "ggplot2")
ggplot(midwest, aes(x=area, y=poptotal)) +
geom_point(aes(col=state, size=popdensity)) +
geom_smooth(method="loess", se=F) +
xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
labs(subtitle="Area Vs Population", y="Population", x="Area",
title="Scatterplot", caption = "Source: midwest")
```

32: We will not be adding the ggplot2 and theme_set calls in the code chunks anymore; unless otherwise stated, assume that they have been compiled at an earlier stage.



Scatterplots and Jitter Charts



Marginal Distributions

```
library(ggExtra)
data(mpg, package="ggplot2")

mpg_select <- mpg[mpg$hwy >= 35 & mpg$cty > 27, ]
g <- ggplot(mpg, aes(cty, hwy)) + geom_count() +
   geom_smooth(method="lm", se=F)

plot(g)
ggMarginal(g, type = "histogram", fill="transparent")</pre>
```



Diverging Bar Charts

```
mtcars_new <- mtcars |>
  tibble::rownames_to_column(var = "car_name") |>
  dplyr::mutate(car_name = as.factor(car_name),
    mpg_z = round(scale(mpg), 2),
    mpg_type = ifelse(mpg_z < 0, "below", "above")) |>
  dplyr::arrange(mpg_z) # sort
# Diverging Barcharts
ggplot(mtcars_new, aes(x= car_name, y=mpg_z, label=mpg_z)) +
  geom_bar(stat='identity', aes(fill=mpg_type), width=.5) +
  scale_fill_manual(name="Mileage",
    labels = c("Above Average", "Below Average"),
    values=c("above"="#00ba38", "below"="#f8766d")) +
  labs(subtitle="Normalised mileage from 'mtcars'",
    title= "Diverging Bars", x = "Car Names") + coord_flip()
```



Area Charts

```
library(quantmod)
data("economics", package = "ggplot2")
# Compute % Returns
economics$returns_perc <- c(0, diff(economics$psavert) /</pre>
     economics$psavert[-length(economics$psavert)])
# Create break points and labels for axis ticks
brks <- economics$date[seq(1, length(economics$date), 12)]</pre>
#install.packages("lubridate")
lbls <- lubridate::year(economics$date[seq(1,</pre>
    length(economics$date), 12)])
# Plot
ggplot(economics[1:100, ], aes(date, returns_perc)) +
  geom_area() +
  scale_x_date(breaks=brks, labels=lbls) +
  theme(axis.text.x = element_text(angle=90)) +
  labs(title="Area Chart",
       subtitle = "Perc Returns for Personal Savings",
       y="% Returns for Personal savings",
       caption="Source: economics")
```



Funnel Charts

```
library(ggthemes)
library(ggfortify)
# Read data
email_campaign_funnel <- read.csv("https://raw.</pre>
      githubusercontent.com/selva86/datasets/
      master/email_campaign_funnel.csv")
# X Axis Breaks and Labels
brks <- seq(-15000000, 15000000, 5000000)</pre>
lbls = paste0(as.character(c(seq(15, 0, -5),
                              seq(5, 15, 5))), "m")
# Plot
ggplot(email_campaign_funnel, aes(x = Stage, y = Users,
        fill=Gender)) +
    geom_bar(stat = "identity", width = .6) +
    scale_y_continuous(breaks = brks,
        labels = lbls) +
    coord_flip() +
    labs(title="Email Campaign Funnel") +
    theme_tufte() + # Tufte theme from ggfortify
    theme(plot.title = element_text(hjust = .5),
        axis.ticks = element_blank()) +
    scale_fill_brewer(palette = "Dark2")
```



Calendar Heatmaps

```
year yearmonthf monthf week monthweek weekdayf VIX.Close
1 2012
         Jan 2012
                                      1
                                             Tue
                                                     22.97
                     Jan
                            1
2 2012
         Jan 2012
                     Jan
                            1
                                      1
                                             Wed
                                                     22.22
3 2012
        Jan 2012
                     Jan
                            1
                                      1
                                             Thu
                                                     21.48
4 2012
        Jan 2012
                     Jan
                            1
                                      1
                                             Fri
                                                     20.63
5 2012
                                      2
         Jan 2012
                     Jan
                            2
                                             Mon
                                                     21.07
6 2012
        Jan 2012
                     Jan
                            2
                                      2
                                             Tue
                                                     20.69
```



Time-Series Calendar Heatmap



Ordered Bar Charts

```
# Prepare data: group mean city mileage by manufacturer.
cty_mpg <- aggregate(mpg$cty, by=list(mpg$manufacturer), FUN=mean)
colnames(cty_mpg) <- c("make", "mileage")
cty_mpg <- cty_mpg[order(cty_mpg$mileage), ]
cty_mpg$make <- factor(cty_mpg$make, levels = cty_mpg$make)
head(cty_mpg, 4)
```

	make	mileage
9	lincoln	11.33333
8	land rover	11.50000
3	dodge	13.13514
10	mercury	13.25000

```
# Draw plot
ggplot(cty_mpg, aes(x=make, y=mileage)) +
  geom_bar(stat="identity", width=.5, fill="tomato3") +
  labs(title="Ordered Bar Chart",
      subtitle="Make Vs Avg. Mileage",
      caption="source: mpg") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```



source: mpg

Correlograms

```
library(ggcorrplot)
# Correlation matrix
corr <- round(cor(mtcars), 1)
# Plot
ggcorrplot(corr, hc.order = TRUE,
    type = "lower",
    lab = TRUE,
    lab_size = 3,
    method="circle",
    colors = c("tomato2", "white", "springgreen3"),
    title="Correlogram of mtcars",
    ggtheme=theme_bw)</pre>
```



Treemaps

library(devtools)
<pre>#devtools::install_github("wilkox/treemapify")</pre>
library(treemapify)
data(G20)
head(G20)

		region	(country	gdp_mil_usd	hdi
1		Africa	South	Africa	384315	0.629
2	North	America	United	States	15684750	0.937
3	North	America		Canada	1819081	0.911
4	North	America		Mexico	1177116	0.775
5	South	America		Brazil	2395968	0.730
6	South	America	Arg	gentina	474954	0.811

econ_classification hemisphere

1	Developing	Southern
2	Advanced	Northern
3	Advanced	Northern
4	Developing	Northern
5	Developing	Southern
6	Developing	Southern

```
ggplot(G20, aes(area = gdp_mil_usd, fill=region,
        label = country)) +
   geom_treemap() +
   geom_treemap_text(grow = T, reflow = T, colour = "black") +
   facet_wrap( ~ econ_classification) +
   scale_fill_brewer(palette = "Set1") +
   theme(legend.position = "bottom") +
   labs(
     title = "The G-20 major economies",
     caption = "The area of each country is proportional
        to its relative GDP
within the economic group (advanced or developing)",
     fill="Region"
   )
```

The G-20 major economies



The area of each country is proportional to its relative GDP within the economic group (advanced or developing)

Parallel Coordinates

```
# set an id string that denotes the value combination
df_grouped <- df_grouped |> mutate(id = factor(paste(q1_d1,
       q1_d2, q1_d3, sep = '-')))
order.freq <- order(df_grouped[,4],decreasing=TRUE)</pre>
# sort by count and select top rows
df_grouped <- df_grouped[order.freq[1:25],]</pre>
library(reshape2)
# create long format
df_pcp <- melt(df_grouped, id.vars = c('id', 'freq'))</pre>
df_pcp$value <- factor(df_pcp$value)</pre>
y_levels <- levels(factor(1:7))</pre>
ggplot(df_pcp, aes(x = variable, y = value, group = id)) +
  geom_path(aes(size = freq, color = id),
            alpha = 0.5,
            lineend = 'round', linejoin = 'round') +
  scale_y_discrete(limits = y_levels, expand = c(0.5, 0)) +
  scale_size(breaks = NULL, range = c(1, 7))
```



```
Time Series and Variants
```

```
library(ggfortify)
theme_set(theme_classic())
autoplot(AirPassengers) + labs(title="AirPassengers") +
theme(plot.title = element_text(hjust=0.5))
```



theme_set(theme_classic())



Number of unemployed in thousands from 'Economics-US' Dataset



library(lubridate)

Monthly Time Series Personal consumption expenditures, in billions of dollars



Source: Economics

library(lubridate) economics_y <- economics[1:90,] # labels and breaks for X axis text brks <- economics_y\$date[seq(1, length(economics_y\$date),12)] lbls <- lubridate::year(brks) # plot</pre>

```
ggplot(economics_y, aes(x=date)) +
  geom_line(aes(y=psavert)) +
  labs(title="Yearly Time Series",
      subtitle="Personal savings rate",
      caption="Source: Economics",
      y="psavert") +
  scale_x_date(labels = lbls, breaks = brks) +
  theme(axis.text.x = element_text(angle = 90, vjust=0.5),
      panel.grid.minor = element_blank())
```



```
data(economics_long, package = "ggplot2")
head(economics_long)
library(lubridate)
df <- economics_long[economics_long$variable %in% c("psavert",</pre>
                                   "uempmed"), ]
df <- df[lubridate::year(df$date) %in% c(1967:1981), ]</pre>
# labels and breaks for X axis text
brks <- df$date[seq(1, length(df$date), 12)]</pre>
lbls <- lubridate::year(brks)</pre>
# plot
ggplot(df,aes(x=date)) + geom_line(aes(y=value,col=variable)) +
  labs(title="Time Series of Returns Percentage",
       subtitle="Drawn from Long Data format",
       caption="Source: Economics",
       y="Returns %", color=NULL) +
  scale_x_date(labels = lbls, breaks = brks) +
```







```
library(lubridate)
df <- economics[, c("date", "psavert", "uempmed")]</pre>
```

```
df <- df[lubridate::year(df$date) %in% c(1967:1981), ]</pre>
```

```
# labels and breaks for X axis text
brks <- df$date[seq(1, length(df$date), 12)]
lbls <- lubridate::year(brks)</pre>
```

plot

```
ggplot(df, aes(x=date)) +
geom_area(aes(y=psavert+uempmed, fill="psavert")) +
geom_area(aes(y=uempmed, fill="uempmed")) +
labs(title="Area Chart of Returns Percentage",
    subtitle="From Wide Data format",
    caption="Source: Economics",
    y="Returns %") +
scale_x_date(labels = lbls, breaks = brks) +
scale_fill_manual(name="", values=c("psavert"="#00ba38",
        "uempmed"="#f8766d")) +
theme(panel.grid.minor = element_blank())
```







Clusters

```
# devtools::install_github("hrbrmstr/ggalt")
library(ggalt)
library(ggfortify)
theme_set(theme_classic())
```

```
# Compute data with principal components
df <- iris[c(1, 2, 3, 4)]
pca_mod <- prcomp(df) # compute principal components</pre>
# Data frame of principal components -
df_pc <- data.frame(pca_mod$x, Species=iris$Species)</pre>
df_pc_vir <- df_pc[df_pc$Species == "virginica", ]</pre>
df_pc_set <- df_pc[df_pc$Species == "setosa", ]</pre>
df_pc_ver <- df_pc[df_pc$Species == "versicolor", ]</pre>
# Plot
ggplot(df_pc, aes(PC1, PC2, col=Species)) +
  geom_point(aes(shape=Species), size=2) +
  labs(title="Iris Clustering",
       subtitle="With principal components PC1 and PC2
                 as X and Y axis",
       caption="Source: Iris") +
  coord_cartesian(xlim=1.2*c(min(df_pc$PC1),max(df_pc$PC1)),
                ylim=1.2*c(min(df_pc$PC2),max(df_pc$PC2))) +
  geom_encircle(data=df_pc_vir, aes(x=PC1, y=PC2)) +
  geom_encircle(data=df_pc_set, aes(x=PC1, y=PC2)) +
  geom_encircle(data=df_pc_ver, aes(x=PC1, y=PC2))
```

```
Iris Clustering
```

With principal components PC1 and PC2 as X and Y axis



Dumbbell Charts

```
# for right ordering of the dumbells
health$Area <- factor(health$Area,</pre>
    levels=as.character(health$Area))
# health$Area <- factor(health$Area)</pre>
gg <- ggplot(health, aes(x=pct_2013, xend=pct_2014, y=Area,</pre>
    group=Area)) +
    geom_dumbbell(color="#a3c4dc",
                  size=0.75,
                   point.colour.l="#0e668b") +
    scale_x_continuous(label=waiver()) +
    labs(x=NULL,
         v=NULL,
         title="Dumbbell Chart",
         subtitle="Pct Change: 2013 vs 2014",
         caption="Source: https://github.com/hrbrmstr/ggalt") +
    theme(plot.title = element_text(hjust=0.5, face="bold"),
          plot.background=element_rect(fill="#f7f7f7"),
          panel.background=element_rect(fill="#f7f7f7"),
          panel.grid.minor=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.major.x=element_line(),
          axis.ticks=element_blank(),
          legend.position="top",
          panel.border=element_blank())
plot(gg)
```



Slope Charts

```
library(dplyr)
theme_set(theme_classic())
source_df <- read.csv("https://raw.githubusercontent.com/</pre>
    jkeirstead/r-slopegraph/master/cancer_survival_rates.csv")
# Define functions
tufte_sort <- function(df, x="year", y="value", group="group",</pre>
         method="tufte", min.space=0.05) {
    ## First rename the columns for consistency
    ids <- match(c(x, y, group), names(df))</pre>
    df <- df[,ids]
    names(df) <- c("x", "y", "group")</pre>
    ## Expand grid so all combinations have a value
    tmp <- expand.grid(x=unique(df$x), group=unique(df$group))</pre>
    tmp <- merge(df, tmp, all.y=TRUE)</pre>
    df <- mutate(tmp, y=ifelse(is.na(y), 0, y))</pre>
    ## Cast into a matrix shape and arrange by first column
    require(reshape2)
    tmp <- dcast(df, group ~ x, value.var="y")</pre>
    ord <- order(tmp[,2])</pre>
    tmp <- tmp[ord,]</pre>
    min.space <- min.space*diff(range(tmp[,-1]))</pre>
    yshift <- numeric(nrow(tmp))</pre>
    for (i in 2:nrow(tmp)) {
      mat <- as.matrix(tmp[(i-1):i, -1])</pre>
      d.min <- min(diff(mat))</pre>
      yshift[i] <- ifelse(d.min < min.space, min.space-d.min,0)</pre>
    }
    tmp <- cbind(tmp, yshift=cumsum(yshift))</pre>
    scale <- 1
    tmp <- melt(tmp,id=c("group","yshift"), variable.name="x"</pre>
                 value.name="y")
    tmp <- transform(tmp, ypos=y + scale*yshift)</pre>
    return(tmp)
}
plot_slopegraph <- function(df) {</pre>
    ylabs <- subset(df, x==head(x,1))$group</pre>
    yvals <- subset(df, x==head(x,1))$ypos</pre>
    fontSize <- 3
    gg <- ggplot(df,aes(x=x,y=ypos)) +</pre>
      geom_line(aes(group=group),colour="grey80") +
      geom_point(colour="white",size=8) +
      geom_text(aes(label=y), size=fontSize) +
```

```
scale_y_continuous(name="", breaks=yvals, labels=ylabs)
    return(gg)
}
## Prepare data
df <- tufte_sort(source_df,</pre>
                  x="year",
                  y="value",
                  group="group",
                  method="tufte",
                  min.space=0.05)
df <- transform(df,</pre>
                 x=factor(x, levels=c(5,10,15,20),
                             labels=c("5 years","10 years",
                                       "15 years", "20 years")),
                 y=round(y))
## Plot
plot_slopegraph(df) + labs(title="Estimates of % survival rates") +
                 theme(axis.title=element_blank(),
                           axis.ticks = element_blank(),
                           plot.title = element_text(hjust=0.5,
                                family = "American Typewriter",
                                face="bold"),
                           axis.text = element_text(family =
                                "American Typewriter",
                                face="bold"))
```

Estimates of % survival rates



Dendrograms

```
library("ggdendro")
hc <- hclust(dist(USArrests), "ave") # hierarchical clust.
ggdendrogram(hc, rotate = TRUE,size=2)</pre>
```



Density Plots

```
theme_set(theme_classic())
ggplot(mpg, aes(cty)) + geom_density(aes(fill=factor(cyl)),
    alpha=0.8) + labs(title="Density Plot",
    subtitle="City Mileage Grouped by Number of cylinders",
    caption="Source: mpg", x="City Mileage", fill="# Cylinders")
```


Boxplots

```
theme_set(theme_classic())
ggplot(mpg, aes(class, cty)) + geom_boxplot(varwidth=T,
    fill="plum") + labs(title="Boxplot",
    subtitle="City Mileage grouped by Class of vehicle",
    caption="Source: mpg", x="Class of Vehicle", y="City Mileage")
```



Boxplots and Dotplots



Waffle Charts

```
var <- mpg$class # the categorical data
nrows <- 10
df <- expand.grid(y = 1:nrows, x = 1:nrows)</pre>
(categ_table <- round(table(var)*((nrows*nrows)/(length(var)))))</pre>
 2seater compact midsize minivan pickup subcompact suv
       2
               20
                       18
                                5
                                      14
                                                 15 26
df$category <- factor(rep(names(categ_table), categ_table))</pre>
ggplot(df,aes(x=x,y=y,fill=category)) +
  geom_tile(color="black", size=0.5) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand = c(0, 0), trans = 'reverse') +
  scale_fill_brewer(palette = "Set3") +
  labs(title="Waffle Chart", subtitle="'Class' of vehicles",
  caption="Source: mpg") +
  plot.title = element_text(size = rel(1.2)),
    legend.position = "right",
 axis.text = element_blank(),
  axis.title = element_blank(),
  axis.ticks = element_blank(),
  legend.title = element_blank())
```





Source: mpg