KURUNJI VENKATRAMANA GOWDA POLYTECHNIC SULLIA-574327

5TH SEMESTER

AI/ML WEEK-3

Explore NumPy module:

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.NumPy stands for Numerical Python

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important

Import NumPy

Once NumPy is installed, import it in your applications by adding the import keyword:

```
import numpy
```

Now NumPy is imported and ready to use.

Example

```
import numpy

arr = numpy.array([1, 2, 3, 4, 5])

print(arr)
```

NumPy as np

NumPy is usually imported under the np alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the as keyword while importing:

```
import numpy as np
```

Now the NumPy package can be referred to as np instead of numpy.

Example

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

Checking NumPy Version

The version string is stored under __version__attribute.

```
import numpy as np

print(np.__version__)
```

1. Array Aggregation Functions with NumPy:

Python numpy Aggregate Functions

In the Python numpy module, we have many aggregate functions or statistical functions to work with a single-dimensional or multi-dimensional array. The Python numpy aggregate functions are sum, min, max, mean, average, product, median, standard deviation, variance, argmin, argmax, percentile, cumprod, cumsum, and corrcoef.

These are some of the Array Aggregation using NumPy:

1. numpy.sum(): Returns the sum of the array.

```
# Importing numpy module
import numpy as np
# Importing random module to generate array with random
number import random # Creating an array arr =
np.array([1,2,3,4,5])  print(np.sum(arr))
```

2. numpy.min() numpy.max() returns minimum element and maximum element respectively.

```
# Importing
numpy module
import numpy
as np
# Importing random module to generate array
with randomnumber import random # Creating an
array arr = np.array([1,2,3,4,5])
print(np.min(arr))
print(np.max(arr))
```

3. Getting largest element of an array from individual axis:

Syntax: numpy.max(array,axis = 1) [axis = 1 means x axis – row wise]
Syntax: numpy.max(array,axis = 0) [axis = 1 means y axis – column wise]

```
       import numpy as np
import random
```

```
array = np.random.randint(16,size=(4,4)) print("Input
Array: \n",array)
 print("largest number in x axis:
",np.max(array,axis=1)) print("largest number in y
axis: ",np.max(array,axis=0))
```

1. Getting the smallest element of an array from individual axis: Syntax: numpy.min(array,axis = 1) [axis = 1 means x axis – row wise]
   Syntax: numpy.min(array,axis = 0) [axis = 1 means y axis – column wise]

```
import numpy as np
import random
array = np.random.randint(16,size=(4,4)) print("Input
Array: \n",array)
# Smallest element of the x and y axis
```

```
print("Smallest number in x axis: ",np.min(array,axis=1)) print("Smallest
number in y axis: ",np.min(array,axis=0))
```

Here are some of the important Array aggregation functions:

Functions	Description
np.mean()	Compute the arithmetic mean along the specified axis.
np.std()	Compute the standard deviation along the specified axis.
np.var()	Compute the variance along the specified axis.
np.sum()	Sum of array elements over a given axis.

np.prod()	Return the product of array elements over a given axis.
np.cumsum()	Return the cumulative sum of the elements along a given axis.
np.cumprod()	Return the cumulative product of elements along a given axis.
np.min(), np.max()	Return the minimum / maximum of an array or minimum along an axis.
np.argmin(), np.argmax()	Returns the indices of the minimum / maximum values along an axis
np.all()	Test whether all array elements along a given axis evaluate to True.
np.any()	Test whether any array element along a given axis evaluates to True.

Map:

The map() function applies a given function to each item of an iterable (list, tuple etc.) and returns an iterator. map() Syntax Syntax: map(function, iterable)

Example of map()

```
def square(number):
    return number*number nums
= [1,2,3,4]
squared_numbers = list(map(square,nums)) print(squared_numbers)

# Using lambda function  nums1
= [1,2,3,4,5,6]
squareOfnums1 = list(map(lambda n:n*n,nums1)) print(squareOfnums1)
```

Filter:

The filter () method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

syntax:

filter(function, sequence) **Example:**

```
def is_even(n):
if n % 2 ==0:
return True
else:
return False nums =
[1,2,3,4]
evens = list(filter(is_even,nums)) print(evens)
```

OUTPUT:

[2, 4]

Reduce:

The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.This function is defined in “functools” module.

Example:

```
from functools import reduce def sum_all(a,b):
return a+b nums = [1,2,3,4] sum = reduce(sum_all,nums)
print(sum)
```

OUTPUT:

10

LAMBDA:

Python Lambda Functions are anonymous function means that the function is without a name. As we already know that the def keyword is used to define a normal function in Python.

Similarly, the lambda keyword is used to define an anonymous function in Python

Syntax: lambda arguments: expression Example:

```
nums = [1,2,3,4]
evens = list(filter(lambda n:n%2==0,nums))
print(evens)
```

Pandas Data Frames:

Pandas Data Frame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labelled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Pandas DataFrame consists of three principal components, the data, rows, and columns.

Creating dataframe using list:

```
# import pandas as pd
import pandas as pd

# list of strings
lst = ['Apple', 'Mango', 'Banana', 'Pine Apple',
            'Grapes', 'Watermelon']

# Calling DataFrame constructor on
list df = pd.DataFrame(lst) print(df)
```

Out Put:

	0
0	Apple
1	Mango
2	Banana
3	Pine Apple
4	Grapes
5	Watermelon

Creating DataFrame from dict:

```
import pandas as pd

# intialise data of lists. data =
{'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}

# Create DataFrame df =
pd.DataFrame(data)
```

Out Put:

	Name	Age
0	Tom	20
1	nick	21 2 krish 19
3	jack	18

Reading a csv file in Pandas:

```
# importing pandas package import
pandas as pd

# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving row by loc method first
= data.loc["Avery Bradley"] second =
data.loc["R.J. Hunter"]
    print(first, "\n\n\n",
second)
```

Output:

Team Boston Celtics

Number 0.0

Position PG

Age 25.0

Height 6-2

Weight 180.0

College Texas

Salary 7730337.0

Name: Avery Bradley, dtype: object

Team Boston Celtics

Number 28.0

Position SG

Age 22.0

Height 6-5

Weight 185.0

College Georgia State

Salary 1148640.0

Name: R.J. Hunter, dtype: object

DataFrame Methods:

FUNCTION	DESCRIPTION
index()	Method returns index (row labels) of the DataFrame
insert()	Method inserts a column into a DataFrame
add()	Method returns addition of dataframe and other, element-wise (binary operator add)
sub()	Method returns subtraction of dataframe and other, element-wise (binary operator sub)
mul()	Method returns multiplication of dataframe and other, elementwise (binary operator mul)
div()	Method returns floating division of dataframe and other, elementwise (binary operator truediv)
unique()	Method extracts the unique values in the dataframe
nunique()	Method returns count of the unique values in the dataframe

value_counts()	Method counts the number of times each unique value occurs within the Series
columns()	Method returns the column labels of the DataFrame
axes()	Method returns a list representing the axes of the DataFrame
isnull()	Method creates a Boolean Series for extracting rows with null values
notnull()	Method creates a Boolean Series for extracting rows with non-null values
between()	Method extracts rows where a column value falls in between a predefined range
isin()	Method extracts rows from a DataFrame where a column value exists in a predefined collection

dtypes()	Method returns a Series with the data type of each column. The result's index is the original DataFrame's columns
astype()	Method converts the data types in a Series
values()	Method returns a Numpy representation of the DataFrame i.e. only the values in the DataFrame will be returned, the axes labels will be removed
sort_values()- Set1, Set2	Method sorts a data frame in Ascending or Descending order of passed Column
sort_index()	Method sorts the values in a DataFrame based on their index positions or labels instead of their values but sometimes a data frame is made out of two or more data frames and hence later index can be changed using this method
loc[]	Method retrieves rows based on index label

iloc[]	Method retrieves rows based on index position
ix[]	Method retrieves DataFrame rows based on either index label or index position. This method combines the best features of the .loc[] and .iloc[] methods
rename()	Method is called on a DataFrame to change the names of the index labels or column names
columns()	Method is an alternative attribute to change the coloumn name
drop()	Method is used to delete rows or columns from a DataFrame
pop()	Method is used to delete rows or columns from a DataFrame
sample()	Method pulls out a random sample of rows or columns from a DataFrame
nsmallest()	Method pulls out the rows with the smallest values in a column

nlargest()	Method pulls out the rows with the largest values in a column
shape()	Method returns a tuple representing the dimensionality of the DataFrame
ndim()	Method returns an 'int' representing the number of axes / array dimensions. Returns 1 if Series, otherwise returns 2 if DataFrame
dropna()	Method allows the user to analyze and drop Rows/Columns with Null values in different ways
fillna()	Method manages and let the user replace NaN values with some value of their own
rank()	Values in a Series can be ranked in order with this method

query()	Method is an alternate string-based syntax for extracting a subset from a DataFrame
copy()	Method creates an independent copy of a pandas object
duplicated()	Method creates a Boolean Series and uses it to extract rows that have duplicate values
drop_duplicates()	Method is an alternative option to identifying duplicate rows and removing them through filtering
set_index()	Method sets the DataFrame index (row labels) using one or more existing columns
reset_index()	Method resets index of a Data Frame. This method sets a list of integer ranging from 0 to length of data as index
where()	Method is used to check a Data Frame for one or more condition and return the result accordingly. By default, the rows not satisfying the condition are filled with NaN value

Pivot and Melt Function in Pandas:

Pandas melt() function is used to change the DataFrame format from wide to long. It's used to create a specific format of the DataFrame object where one or more columns work as identifiers. All the remaining columns are treated as values and unpivoted to the row axis and only two columns – variable and value.

1. Pandas melt() Example

The use of melt() function is more clear when looked through an example.

```
import pandas as pd
 d1 = {"Name": ["Pankaj", "Lisa", "David"], "ID": [1, 2, 3], "Role": ["CEO", "Editor",
"Author"]}
 df =
pd.DataFrame(d1)

print(df)
 df_melted = pd.melt(df, id_vars=["ID"], value_vars=["Name",
"Role"])
 print(df_melted)
```

Output:

Name ID Role

0 Pankaj 1 CEO

1 Lisa 2 Editor

2 David 3 AuthorID

variable value

0 1 Name Pankaj

1 2 Name Lisa

2 3 Name David

3 1 Role CEO

4 2 Role Editor

5 3 Role Author

Pivot in Pandas:

pandas.pivot(index, columns, values) function produces pivot table based on 3 columns of the DataFrame. Uses unique values from index / columns and fills with values.

```
# importing pandas as pd import
pandas as pd

# creating a dataframe
df = pd.DataFrame({'A': ['John', 'Boby', 'Mina'],
    'B': ['Masters', 'Graduate', 'Graduate'],
    'C': [27, 23, 21]})
 print(df)
```

Output:

Out[9]:

	A	B	C
0	John	Masters	27
1	Boby	Graduate	23
2	Mina	Graduate	21

1. Data visualization with python

What is Data Visualization?

With so much information being collected through data analysis in the business world today, we must have a way to paint a picture of that data so we can interpret it. Data visualization gives us a clear idea of what the information means by giving it visual context through maps or graphs. This makes the data more natural for the human mind to comprehend and therefore makes it easier to identify trends, patterns, and outliers within large data sets.

Why is Data Visualization Important?

No matter what business or career you've chosen, data visualization can help by delivering data in the most efficient way possible. As one of the essential steps in the business intelligence process, data visualization takes the raw data, models it, and delivers the data so that conclusions can be reached. In advanced analytics, data scientists are creating machine learning algorithms to better compile essential data into visualizations that are easier to understand and interpret.

Specifically, data visualization uses visual data to communicate information in a manner that is universal, fast, and effective. This practice can help companies identify which areas need to be improved, which factors affect customer satisfaction and dissatisfaction, and what to do with specific products (where should they go and who should they be sold to). Visualized data gives stakeholders, business owners, and decision-makers a better prediction of sales volumes and future growth.

What Are The Benefits of Data Visualization?

Data visualization positively affects an organization's decision-making process with interactive visual representations of data. Businesses can now recognize patterns more quickly because they can interpret data in graphical or pictorial forms. Here are some morespecific ways that data visualization can benefit an organization:

- **Correlations in Relationships:** Without data visualization, it is challenging to identify the correlations between the relationship of independent variables. By making sense of those independent variables, we can make better business decisions.
- **Trends Over Time:** While this seems like an obvious use of data visualization, it is also one of the most valuable applications. It's impossible to make predictions without having the necessary information from the past and present. Trends over time tell us where we were and where we can potentially go.
- **Frequency:** Closely related to trends over time is frequency. By examining the rate,or how often, customers purchase and when they buy gives us a better feel for how potential new customers might act and react to different marketing and customer acquisition strategies.
- **Examining the Market:** Data visualization takes the information from different markets to give you insights into which audiences to focus your attention on and which ones to stay away from. We get a clearer picture of the opportunities within those markets by displaying this data on various charts and graphs.
- **Risk and Reward:** Looking at value and risk metrics requires expertise because, without data visualization, we must interpret complicated spreadsheets and numbers. Once information is visualized, we can then pinpoint areas that may or may not require action.
- **Reacting to the Market:** The ability to obtain information quickly and easily with data displayed clearly on a functional dashboard allows businesses to act and respond to findings swiftly and helps to avoid making mistakes.

2. Coordinate Systems and Axes

To make any sort of data visualization, we need to define position scales, which determine where in a graphic different data values are located. We cannot visualize data without placing different data points at different locations, even if we just arrange them next to each other along a line. For regular 2d visualizations, two numbers are required to uniquely specify a point, and therefore we need two positionscales. These two scales are usually but not necessarily the *x* and *y* axis of the plot. We also have to specify the relative geometric arrangement of these scales.

Conventionally, the *x* axis runs horizontally and the *y* axis vertically, but we could choose other arrangements. For example, we could have the *y* axis run at an acuteangle relative to the *x* axis, or we could have one axis run in a circle and the other run radially. The combination of a set of position scales and their relative geometric arrangement is called a coordinate system

Cartesian coordinates

The most widely used coordinate system for data visualization is the 2d *Cartesian coordinate system*, where each location is uniquely specified by an *x* and a *y* value. The *x* and *y* axes run orthogonally to each other, and data values are placed in an even spacing along both axes (Figure 3.1). The two axes are continuous position scales, and they can represent both positive and negative real numbers. To fully specify the coordinate system, we need to specify the range of numbers each axis covers. In Figure 3.1, the *x* axis runs from -2.2 to 3.2 and the *y* axis runs from -2.2 to
2.2. Any data values between these axis limits are placed at the respective location in the plot. Any data values outside the axis limits are discarded.

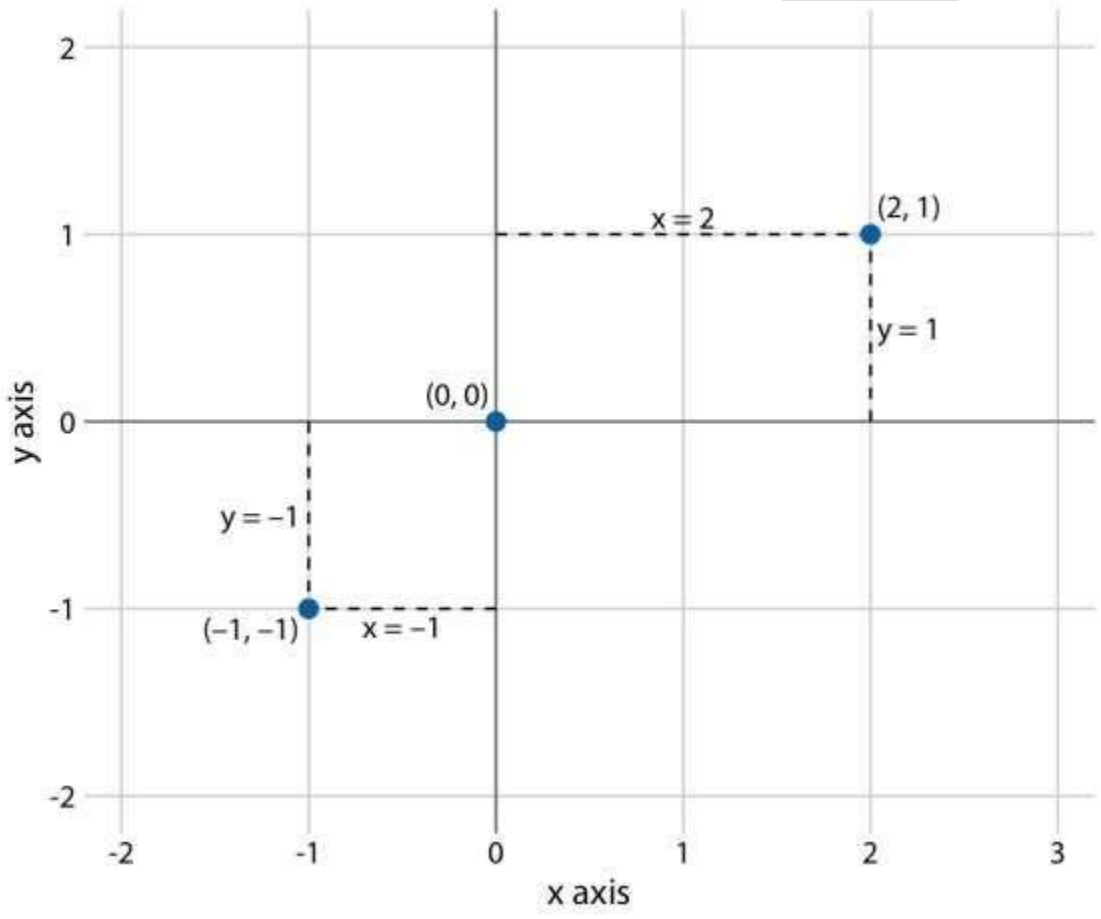


Figure 3.1: Standard cartesian coordinate system. The horizontal axis is conventionally called *x* and the vertical axis *y*. The two axes form a grid with equidistant spacing. Here, both the *x* and the *y* grid lines are separated by units of one. The point (2, 1) is located two *x* units to the right and one *y* unit above the origin (0, 0). The point (-1, -1) is located one *x* unit to the left and one *y* unit below the origin.

Data values usually aren't just numbers, however. They come with units. For example, if we're measuring temperature, the values may be measured in degrees

Celsius or Fahrenheit. Similarly, if we're measuring distance, the values may be measured in kilometers or miles, and if we're measuring duration, the values may be measured in minutes, hours, or days. In a Cartesian coordinate system, the spacing between grid lines along an axis corresponds to discrete steps in these data units. In a temperature scale, for example, we may have a grid line every 10 degrees Fahrenheit, and in a distance scale, we may have a grid line every 5 kilometers.

A Cartesian coordinate system can have two axes representing two different units. This situation arises quite commonly whenever we're mapping two different types of variables to *x* and *y*. For example, in Figure 2.3, we plotted temperature vs. days of the year. The *y* axis of Figure 2.3 is measured in degrees Fahrenheit, with a grid line every at 20 degrees, and the *x* axis is measured in months, with a grid line at the first of every third month. Whenever the two axes are measured in different units, we can stretch or compress one relative to the other and maintain a valid visualization of the data (Figure 3.2). Which version is preferable may

depend on the story we want to convey. A tall and narrow figure emphasizes change along the *y* axis and a short and wide figure does the opposite. Ideally, we want to choose an aspect ratio that ensures that any important differences in position are noticeable.

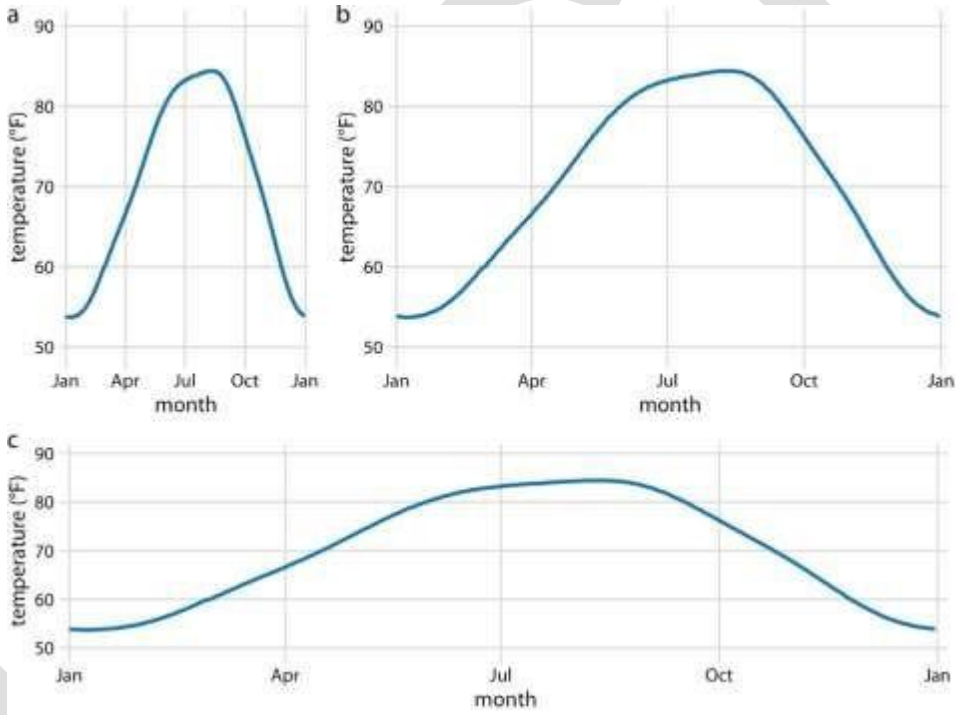


Figure 3.2: Daily temperature normals for Houston, TX. Temperature is mapped to the *y* axis and day of the year to the *x* axis. Parts (a), (b), and (c) show the same figure in different aspect ratios. All three parts are valid visualizations of the temperature data. Data source: NOAA.

On the other hand, if the *x* and the *y* axes are measured in the same units, then the grid spacings for the two axes should be equal, such that the same distance along the *x* or *y* axis corresponds to the same number of data units. As an example, we can plot the temperature in Houston, TX against the temperature in San Diego, CA,

for every day of the year (Figure 3.3a). Since the same quantity is plotted along bothaxes, we need to make sure that the grid lines form perfect squares, as is the case in Figure 3.3.

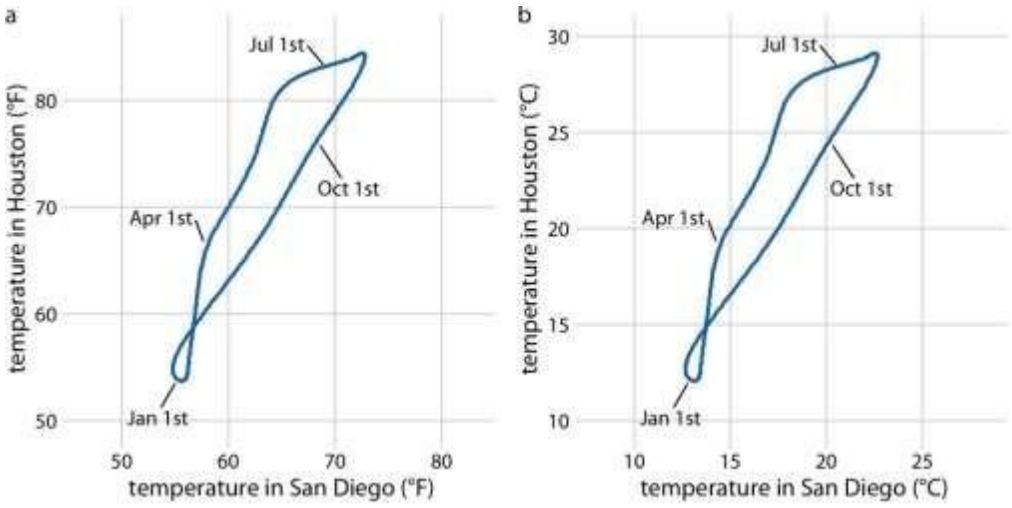


Figure 3.3: Daily temperature normals for Houston, TX, plotted versus the respective temperature normals of San Diego, CA. The first days of the months January, April, July, and October are highlighted to provide a temporal reference. (a) Temperatures are shown in degrees Fahrenheit. (b) Temperatures are shown in degrees Celsius. Data source: NOAA.

You may wonder what happens if you change the units of your data. After all, units are arbitrary, and your preferences might be different from somebody else's. A change in units is a linear transformation, where we add or subtract a number to or from all data values and/or multiply all data values with another number. Fortunately,Cartesian coordinate systems are invariant under such linear transformations.

Therefore, you can change the units of your data and the resulting figure will not change as long as you change the axes accordingly. As an example, compare Figures 3.3a and 3.3b. Both show the same data, but in part (a) the temperature units are degrees Fahrenheit and in part (b) they are degrees Celsius. Even though the grid lines are in different locations and the numbers along the axes are different,the two data visualizations look exactly the same.

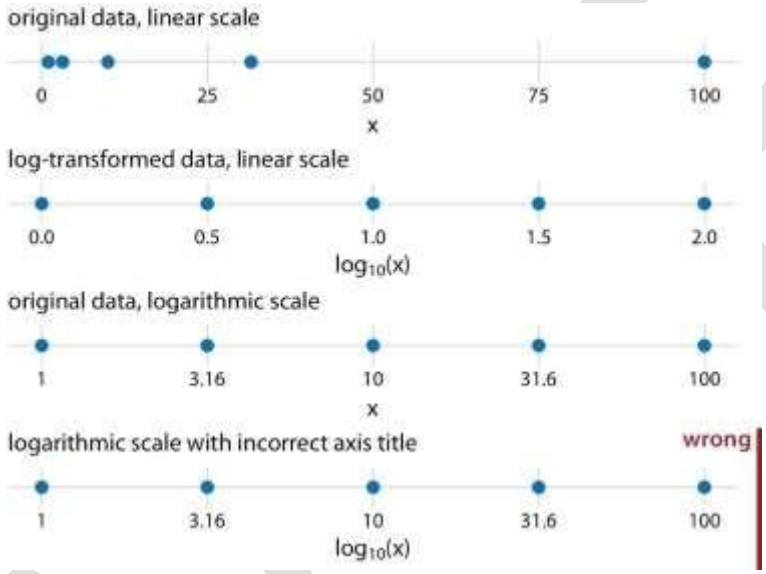
Nonlinear axes

In a Cartesian coordinate system, the grid lines along an axis are spaced evenly both in data units and in the resulting visualization. We refer to the position scales inthese coordinate systems as *linear*. While linear scales generally provide an accurate representation of the data, there are scenarios where nonlinear scales are preferred. In a nonlinear scale, even spacing in data units corresponds to uneven spacing in the visualization, or conversely even spacing in the visualization corresponds to uneven spacing in data units.

The most commonly used nonlinear scale is the *logarithmic scale* or *log scale* for short. Log scales are linear in multiplication, such that a unit step on the scale corresponds to multiplication with a fixed value. To create a log scale, we need to log-transform the data values while exponentiating the numbers that are shown along the axis grid lines. This process is demonstrated in Figure 3.4, which shows the numbers 1, 3.16, 10, 31.6, and 100 placed on linear and log scales. The numbers 3.16 and 31.6 may seem a strange choice, but they were chosen because they are exactly half-way between 1 and 10 and between 10 and 100 on a log scale.

We can see this by observing that $10^{0.5} = \sqrt{} \approx 3.16 10^{0.5} = 10 \approx 3.16$ and

10



equivalently $3.16 \times 3.16 \approx 10 3.16 \times 3.16 \approx 10$.

Similarly, $10^{1.5} = 10 \times 10^{0.5} \approx 31.6 10^{1.5} = 10 \times 10^{0.5} \approx 31.6$.

Figure 3.4: Relationship between linear and logarithmic scales. The dots correspondto data values 1, 3.16, 10, 31.6, 100, which are evenly-spaced numbers on a logarithmic scale. We can display these data points on a linear scale, we can log-

transform them and then show on a linear scale, or we can show them on a logarithmic scale. Importantly, the correct axis title for a logarithmic scale is the name of the variable shown, not the logarithm of that variable.

Mathematically, there is no difference between plotting the log-transformed data on a linear scale or plotting the original data on a logarithmic scale (Figure 3.4). The only difference lies in the labeling for the individual axis ticks and for the axis as a whole. In most cases, the labeling for a logarithmic scale is preferable, because it places less mental burden on the reader to interpret the numbers shown as the axis tick labels. There is also less of a risk of confusion about the base of the logarithm.
When working with log-transformed data, we can get confused about whether the data were transformed using the natural logarithm or the logarithm to base 10. And it's not uncommon for labeling to be ambiguous, e.g. "log(x)", which doesn't specify a base at all. I recommend that you always verify the base when working with log- transformed data. When plotting log-transformed data, always specify the base in the labeling of the axis.

Because multiplication on a log scale looks like addition on a linear scale, log scales are the natural choice for any data that have been obtained by multiplication or division. In particular, ratios should generally be shown on a log scale. As an example, I have taken the number of inhabitants in each county in Texas and have divided it by the median number of inhabitants across all Texas counties. The resulting ratio is a number that can be larger or smaller than 1. A ratio of exactly 1 implies that the corresponding county has the median number of inhabitants. When visualizing these ratios on a log scale, we can see clearly that the population numbers in Texas counties are symmetrically distributed around the median, and that the most populous counties have over 100 times more inhabitants than the median while the least populous counties have over 100 times fewer inhabitants (Figure 3.5). By contrast, for the same data, a linear scale obscures the differences between a county with median population number and a county with a much smaller population number than median (Figure 3.6).

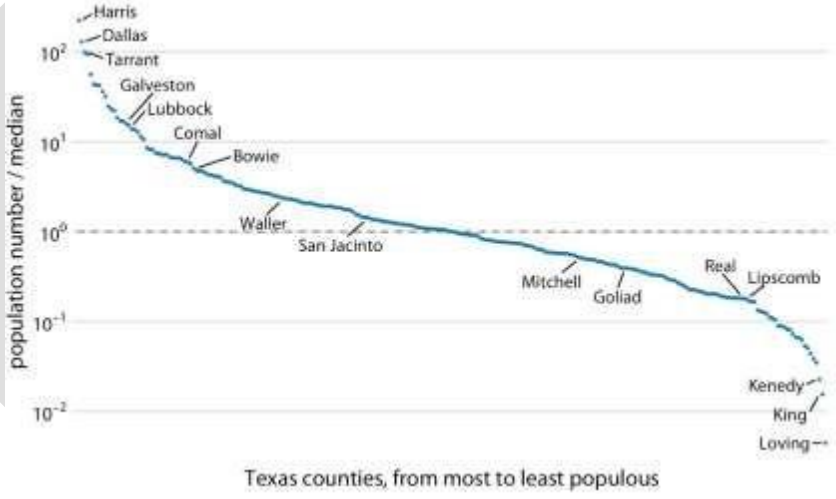


Figure 3.5: Population numbers of Texas counties relative to their median value. Select counties are highlighted by name. The dashed line indicates a ratio of 1, corresponding to a county with median population number. The most populous counties have approximately 100 times more inhabitants than the median county,

and the least populous counties have approximately 100 times fewer inhabitants than the median county. Data source: 2010 Decennial U.S. Census.

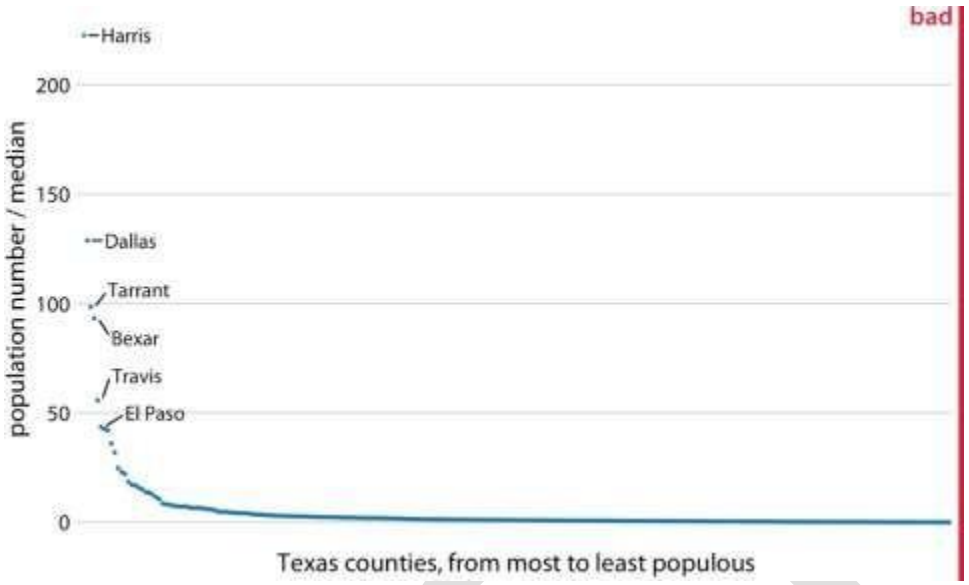


Figure 3.6: Population sizes of Texas counties relative to their median value. By displaying a ratio on a linear scale, we have overemphasized ratios > 1 and have obscured ratios < 1. As a general rule, ratios should not be displayed on a linear scale. Data source: 2010 Decennial U.S. Census.

On a log scale, the value 1 is the natural midpoint, similar to the value 0 on a linear scale. We can think of values greater than 1 as representing multiplications and values less than 1 divisions. For example, we can

write $10=1\times10$ and $0.1=1/10$. The value 0, on the other hand, can never appear on a log scale. It lies infinitely far from 1. One way to see this is to consider that $\log(0)=-\infty$. Or, alternatively, consider that to go from
1 to 0, it takes either an infinite number of divisions by a finite value
(e.g., $1/10/10/10/10/10\cdots=0$) or alternatively one division by infinity (i.e., $1/\infty=0$).

Log scales are frequently used when the data set contains numbers of very different magnitudes. For the Texas counties shown in Figures 3.5 and 3.6, the most populous one (Harris) had 4,092,459 inhabitants in the 2010 U.S. Census while the least populous one (Loving) had 82. So a log scale would be appropriate even if we hadn't divided population numbers by their median to turn them into ratios. But what would we do if there was a county with 0 inhabitants? This county could not be shown on the logarithmic scale, because it would lie at minus infinity. In this situation, the recommendation is sometimes to use a square-root scale, which uses a square root transformation instead of a log transformation
(Figure 3.7). Just like a log scale, a square-root scale compresses larger numbers into a smaller range, but unlike a log scale, it allows for the presence of 0.

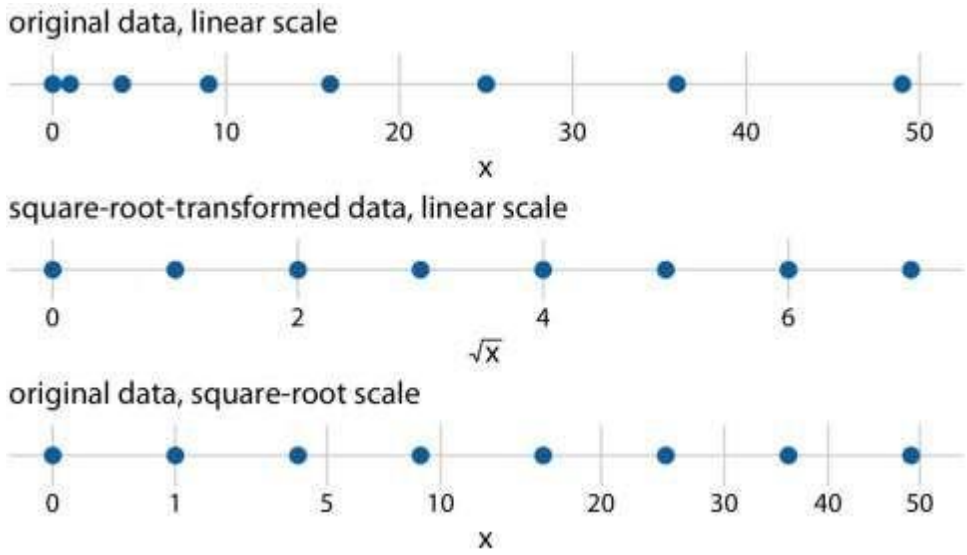


Figure 3.7: Relationship between linear and square-root scales. The dots correspondto data values 0, 1, 4, 9, 16, 25, 36, 49, which are evenly-spaced numbers on a square-root scale, since they are the squares of the integers from 0 to 7. We can display these data points on a linear scale, we can square-roottransform them and then show on a linear scale, or we can show them on a square-root scale.

I see two problems with square-root scales. First, while on a linear scale one unit step corresponds to addition or subtraction of a constant value and on a log scale it corresponds to multiplication with or division by a constant value, no such rule exists for a square-root scale. The meaning of a unit step on a square-root scale depends on the scale value at which we're starting. Second, it is unclear how to best place axis ticks on a square-root scale. To obtain evenly spaced ticks, we would have to place them at squares, but axis ticks at, for example, positions 0, 4, 25, 49, 81 (everysecond square) would be highly unintuitive. Alternatively, we could place them at linear intervals (10, 20, 30, etc), but this would result in either too few axis ticks near the low end of the scale or too many near the high end. In Figure 3.7, I have placed the axis ticks at positions 0, 1, 5, 10, 20, 30, 40, and 50 on the square-root scale.
These values are arbitrary but provide a reasonable covering of the data range.

Despite these problems with square-root scales, they are valid position scales and I do not discount the possibility that they have appropriate applications. For example, just like a log scale is the natural scale for ratios, one could argue that the square- root scale is the natural scale for data that come in squares. One scenario in which data are naturally squares are in the context of geographic regions. If we show the areas of geographic regions on a square-root scale, we are highlighting the regions' linear extent from East to West or North to South. These extents could be relevant, for example, if we are wondering how long it might take to drive across a region.
Figure 3.8 shows the areas of states in the U.S. Northeast on both a linear and a square-root scale. Even though the areas of these states are quite different (Figure 3.8a), the time it will take to drive across each state will more closely resemble the figure on the square-root scale (Figure 3.8b) than the figure on the linear scale (Figure 3.8a).

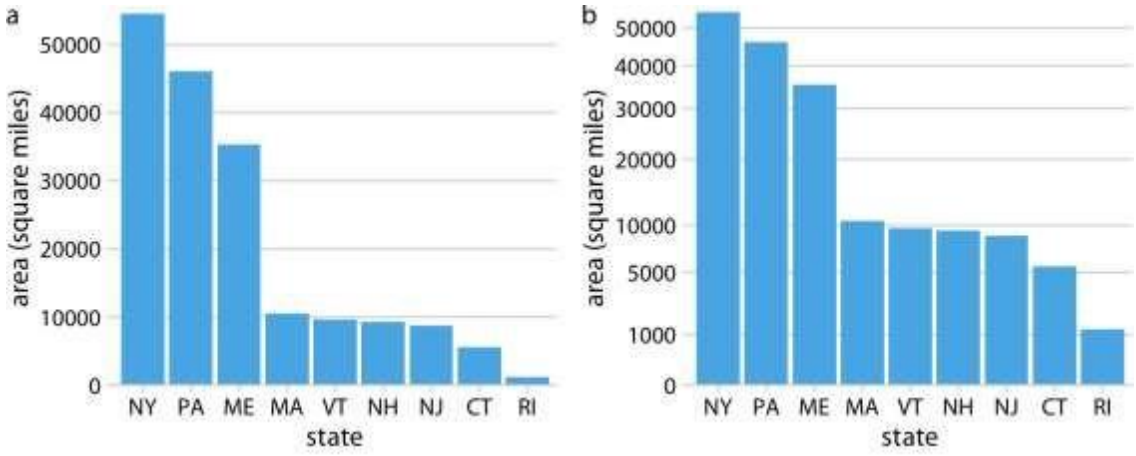


Figure 3.8: Areas of Northeastern U.S. states. (a) Areas shown on a linear scale. (b) Areas shown on a square-root scale. Data source: Google.

Coordinate systems with curved axes

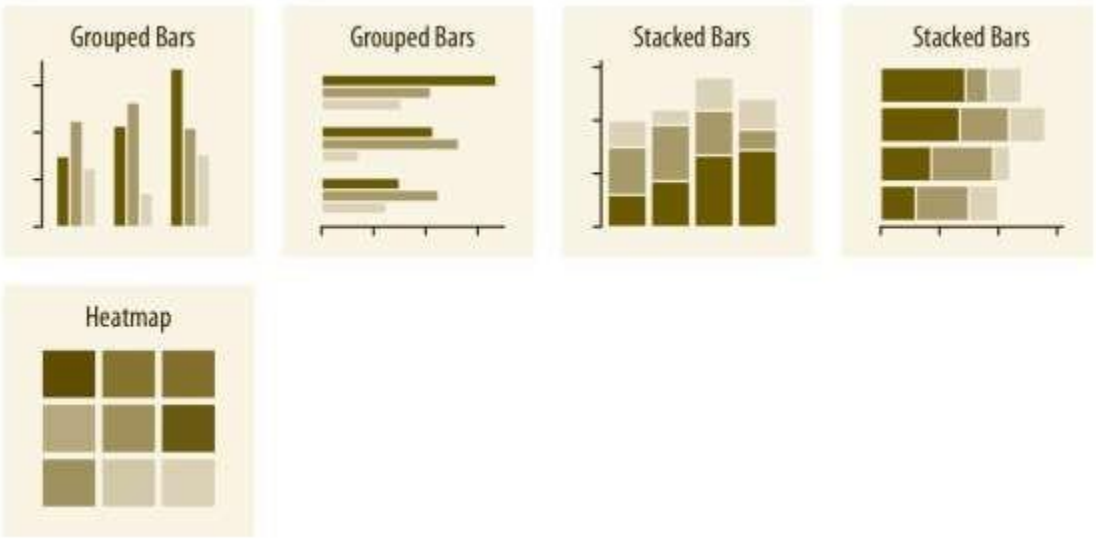
All coordinate systems we have encountered so far used two straight axes positioned at a right angle to each other, even if the axes themselves established a non-linear mapping from data values to positions. There are other coordinate systems, however, where the axes themselves are curved. In particular, in the *polar* coordinate system, we specify positions via an angle and a radial distance from the origin, and therefore the angle axis is circular (Figure 3.9).

3. Directory of visualizations

This chapter provides a quick visual overview of the various plots and charts that arecommonly used to visualize data. It is meant both to serve as a table of contents, in case you are looking for a particular visualization whose name you may not know, and as a source of inspiration, if you need to find alternatives to the figures you routinely make.

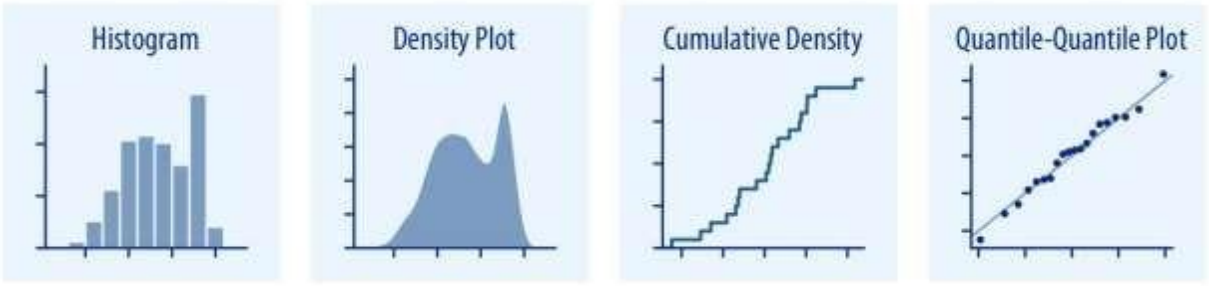
Amounts

The most common approach to visualizing amounts (i.e., numerical values shown for some set of categories) is using bars, either vertically or horizontally arranged (Chapter 6). However, instead of using bars, we can also place dots at the location where the corresponding bar would end (Chapter 6).

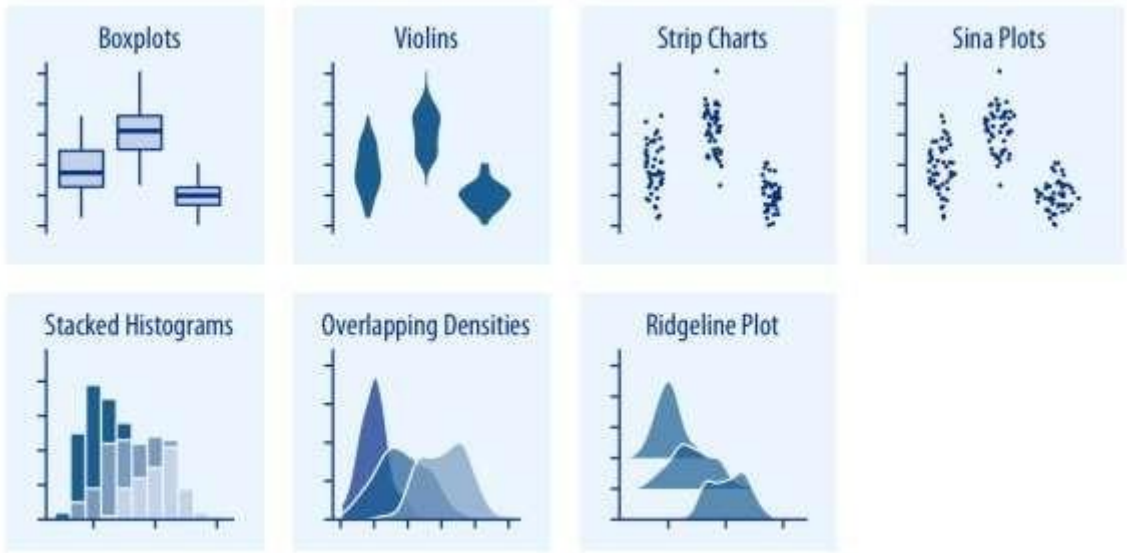


If there are two or more sets of categories for which we want to show amounts, we can group or stack the bars (Chapter 6). We can also map the categories onto the *x* and *y* axis and show amounts by color, via a heatmap (Chapter 6).

Distributions



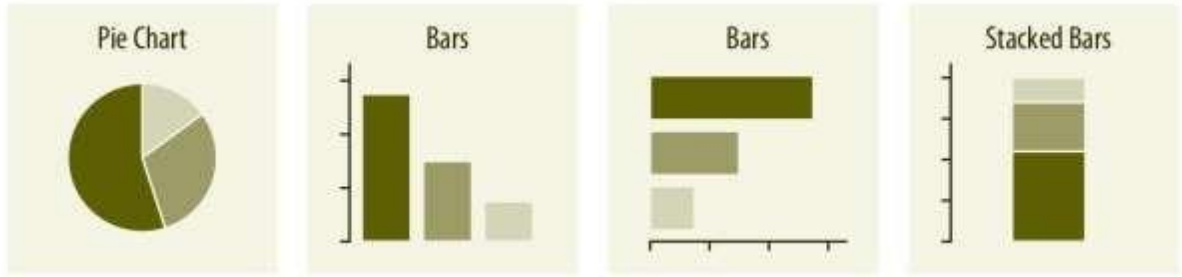
Histograms and density plots (Chapter 7) provide the most intuitive visualizations of a distribution, but both require arbitrary parameter choices and can be misleading. Cumulative densities and quantile-quantile (q-q) plots (Chapter 8) always represent the data faithfully but can be more difficult to interpret.



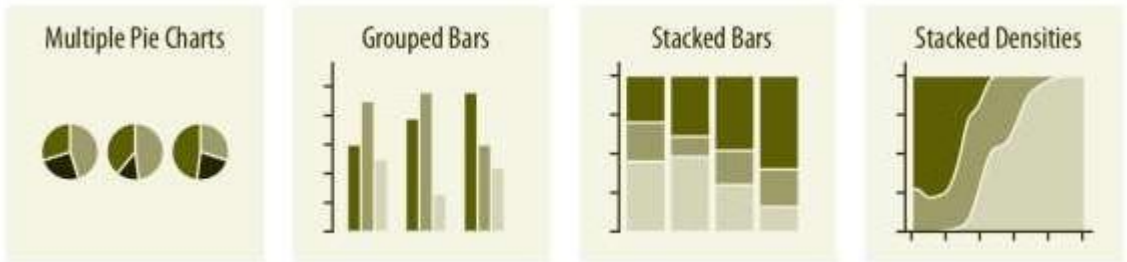
Boxplots, violins, strip charts, and sina plots are useful when we want to visualize many distributions at once and/or if we are primarily interested in overall shifts among the distributions (Chapter 9.1). Stacked histograms and overlapping densitiesallow a more in-depth comparison of a smaller number of distributions, though stacked histograms can be difficult to interpret and are best avoided (Chapter 7.2).

Ridgeline plots can be a useful alternative to violin plots and are often useful when visualizing very large numbers of distributions or changes in distributions over time(Chapter 9.2).

Proportions



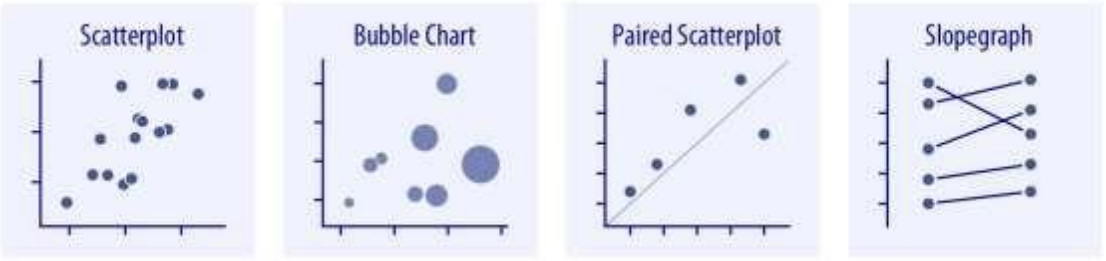
Proportions can be visualized as pie charts, side-by-side bars, or stacked bars (Chapter 10), and as in the case for amounts, bars can be arranged either verticallyor horizontally. Pie charts emphasize that the individual parts add up to a whole andhighlight simple fractions. However, the individual pieces are more easily compared in side-by-side bars. Stacked bars look awkward for a single set of proportions, but can be useful when comparing multiple sets of proportions (see below).



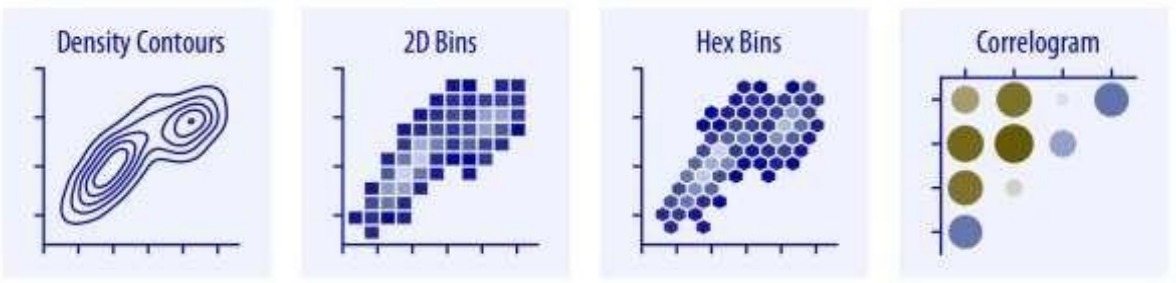
When visualizing multiple sets of proportions or changes in proportions across conditions, pie charts tend to be space-inefficient and often obscure relationships. Grouped bars work well as long as the number of conditions compared is moderate, and stacked bars can work for large numbers of conditions. Stacked densities (Chapter 10) are appropriate when the proportions change along a continuous variable.

When proportions are specified according to multiple grouping variables, then mosaic plots, treemaps, or parallel sets are useful visualization approaches (Chapter 11). Mosaic plots assume that every level of one grouping variable can be combined with every level of another grouping variable, whereas treemaps do not make such an assumption. Treemaps work well even if the subdivisions of one group are entirely distinct from the subdivisions of another. Parallel sets work better than either mosaic plots or treemaps when there are more than two grouping variables.

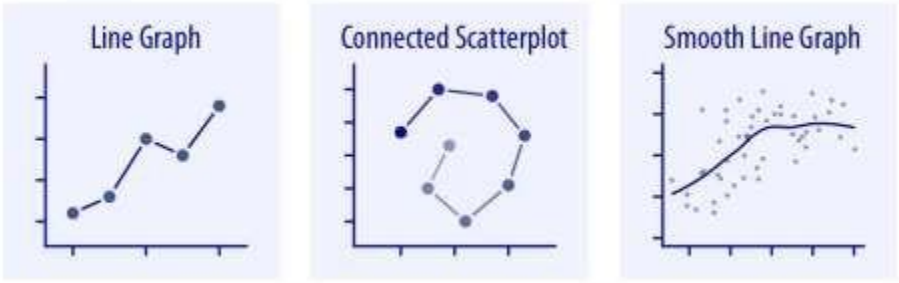
x–*y* relationships



Scatterplots represent the archetypical visualization when we want to show one quantitative variable relative to another (Chapter 12.1). If we have three quantitative variables, we can map one onto the dot size, creating a variant of the scatterplot called bubble chart. For paired data, where the variables along the *x* and the *y* axes are measured in the same units, it is generally helpful to add a line indicating $x = y$ (Chapter 12.4). Paired data can also be shown as a slope graph of paired points connected by straight lines (Chapter 12.4).

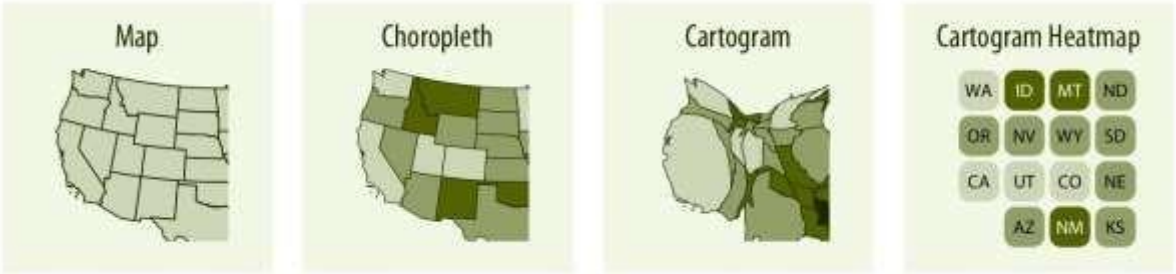


For large numbers of points, regular scatterplots can become uninformative due to overplotting. In this case, contour lines, 2D bins, or hex bins may provide an alternative (Chapter 18). When we want to visualize more than two quantities, on the other hand, we may choose to plot correlation coefficients in the form of a correlogram instead of the underlying raw data (Chapter 12.2).



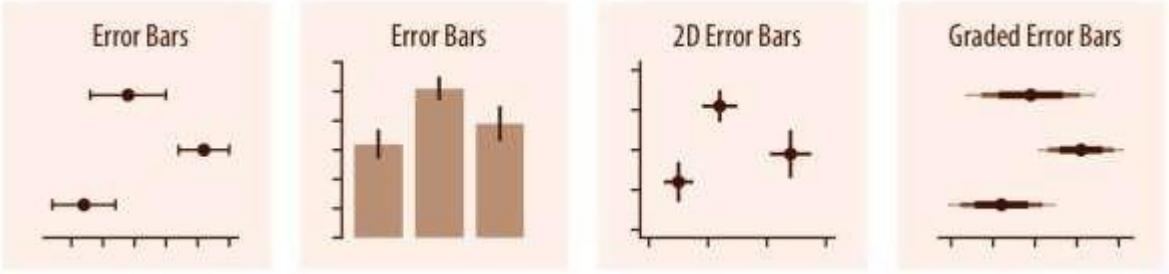
When the *x* axis represents time or a strictly increasing quantity such as a treatment dose, we commonly draw line graphs (Chapter 13). If we have a temporal sequenceof two response variables, we can draw a connected scatterplot where we first plot the two response variables in a scatterplot and then connect dots corresponding to adjacent time points (Chapter 13.3). We can use smooth lines to represent trends ina larger dataset (Chapter 14).

Geospatial data

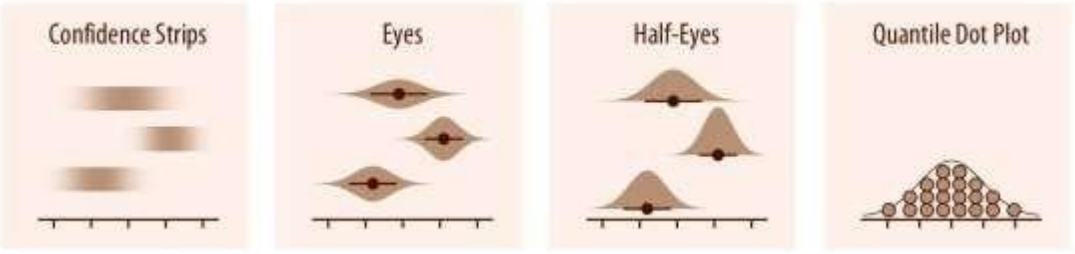


The primary mode of showing geospatial data is in the form of a map (Chapter 15). A map takes coordinates on the globe and projects them onto a flat surface, such that shapes and distances on the globe are approximately represented by shapes and distances in the 2D representation. In addition, we can show data values in different regions by coloring those regions in the map according to the data. Such a map is called a choropleth (Chapter 15.3). In some cases, it may be helpful to distort the different regions according to some other quantity (e.g., population number) or simplify each region into a square. Such visualizations are called cartograms.

Uncertainty



Error bars are meant to indicate the range of likely values for some estimate or measurement. They extend horizontally and/or vertically from some reference point representing the estimate or measurement (Chapter 16). Reference points can be shown in various ways, such as by dots or by bars. Graded error bars show multiple ranges at the same time, where each range corresponds to a different degree of confidence. They are in effect multiple error bars with different line thicknesses plotted on top of each other.



To achieve a more detailed visualization than is possible with error bars or graded error bars, we can visualize the actual confidence or posterior distributions (Chapter 16). Confidence strips provide a clear visual sense of uncertainty but are difficult to read accurately. Eyes and half-eyes combine error bars with approaches to visualize distributions (violins and ridgelines, respectively), and thus show both precise rangesfor some confidence levels and the overall uncertainty distribution. A quantile dot plotcan serve as an alternative visualization of an uncertainty distribution (Chapter 16.1).By showing the distribution in discrete units, the quantile dot plot is not as precise but can be easier to read than the continuous distribution shown by a violin or ridgeline plot.

For smooth line graphs, the equivalent of an error bar is a confidence band (Chapter16.3). It shows a range of values the line might pass through at a given confidence level. As in the case of error bars, we can draw graded confidence bands that show

multiple confidence levels at once. We can also show individual fitted draws in lieu ofor in addition to the confidence bands.

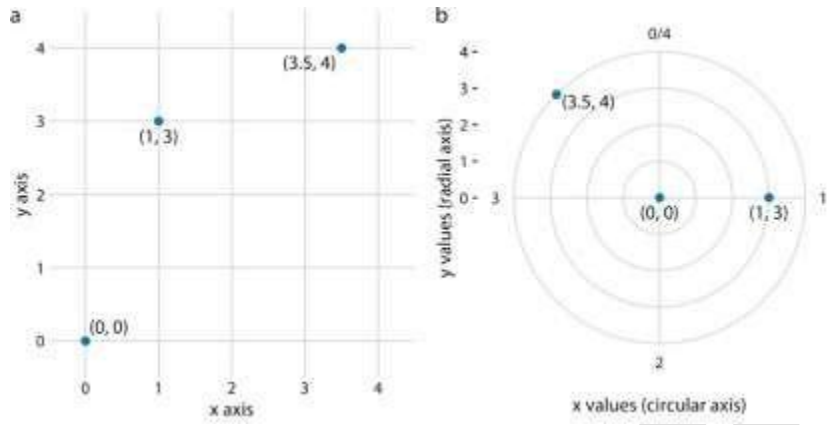


Figure 3.9: Relationship between Cartesian and polar coordinates. (a) Three data points shown in a Cartesian coordinate system. (b) The same three data points shown in a polar coordinate system. We have taken the *x* coordinates from part (a) and used them as angular coordinates and the *y* coordinates from part (a) and usedthem as radial coordinates. The circular axis runs from 0 to 4 in this example, and therefore $x = 0$ and $x = 4$ are the same locations in this coordinate system.

Polar coordinates can be useful for data of a periodic nature, such that data values at one end of the scale can be logically joined to data values at the other end. For example, consider the days in a year. December 31st is the last day of the year, but it is also one day before the first day of the year. If we want to show how some quantity varies over the year, it can be appropriate to use polar coordinates with the angle coordinate specifying each day. Let's apply this concept to the temperature normals of Figure 2.3. Because temperature normals are average temperatures that are not tied to any specific year, Dec. 31st can be thought of as 366 days later than Jan. 1st (temperature normals include Feb. 29) and also one day earlier. By plotting the temperature normals in a polar coordinate system, we emphasize this cyclical property they have (Figure 3.10). In comparison to Figure 2.3, the polar version highlights how similar the temperatures are in Death Valley, Houston, and San Diego from late fall to early spring. In the Cartesian coordinate system, this fact is obscured because the temperature values in late December and in early January are shown inopposite parts of the figure and therefore don't form a single visual unit.

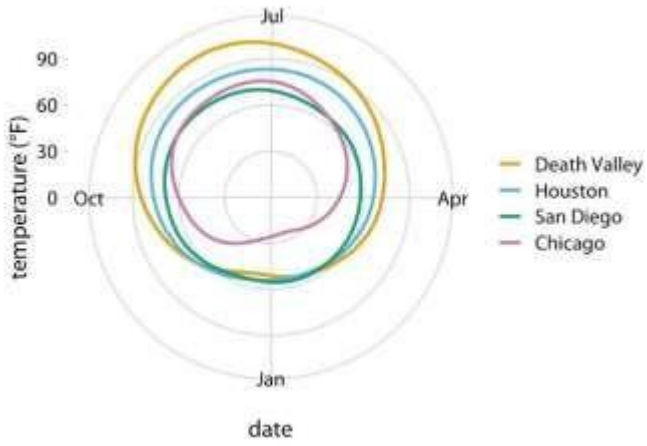


Figure 3.10: Daily temperature normals for four selected locations in the U.S., shown in polar coordinates. The radial distance from the center point indicates the daily temperature in Fahrenheit, and the days of the year are arranged counterclockwise starting with Jan. 1st at the 6:00 position.

A second setting in which we encounter curved axes is in the context of geospatial data, i.e., maps. Locations on the globe are specified by their longitude and latitude. But because the earth is a sphere, drawing latitude and longitude as Cartesian axes is misleading and not recommended (Figure 3.11). Instead, we use various types of non-linear projections that attempt to minimize artifacts and that strike different balances between conserving areas or angles relative to the true shape lines on the globe (Figure 3.11).

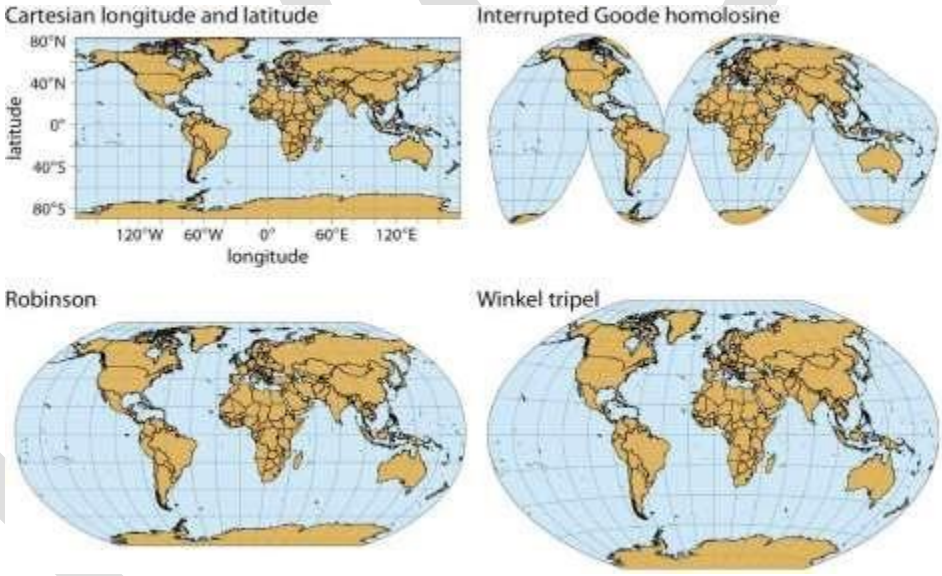


Figure 3.11: Map of the world, shown in four different projections. The Cartesian longitude and latitude system maps the longitude and latitude of each location onto a regular Cartesian coordinate system. This mapping causes substantial distortions in both areas and angles relative to their true values on the 3D globe. The interrupted Goode homolosine projection perfectly represents true surface areas, at the cost of dividing some land masses into separate pieces, most notably Greenland and Antarctica. The Robinson projection and the Winkel tripel projection both strike a

balance between angular and area distortions, and they are commonly used for maps of the entire globe.

Basics of python visualization with Matplotlib

Importing Datasets

In this article, we will use two freely available datasets. The Iris and WineReviews dataset, which we can both load into memory using pandas read_csvmethod.

```
import pandas as pd

iris = pd.read_csv('iris.csv', names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])

print(iris.head())
```

Python

Copy

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figure 1: Iris dataset head

```
wine_reviews = pd.read_csv('winemag-data-130k-v2.csv', index_col=0)

wine_reviews.head()
```

Python

Copy

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

Figure 2: Wine Review dataset head

Matplotlib

Matplotlib is the most popular Python plotting library. It is a low-level library with a Matlablike interface that offers lots of freedom at the cost of having to write more code.

To install Matplotlib, pip, and conda can be used.

```
pip install matplotlib or
conda install matplotlib
```

Bash

Copy

Matplotlib is specifically suitable for creating basic graphs like line charts, bar charts,histograms, etc. It can be imported by typing:

```
import matplotlib.pyplot as plt
```

Python

Copy

Scatter Plot

To create a scatter plot in Matplotlib, we can use the scattermethod. We will also create afigure and an axis using plt.subplotsto give our plot a title and labels.

```
# create a figure and axis fig,
ax = plt.subplots()

# scatter the sepal_length against the sepal_width
ax.scatter(iris['sepal_length'], iris['sepal_width'])
# set a title and labels ax.set_title('Iris
Dataset') ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
```

Python

Copy

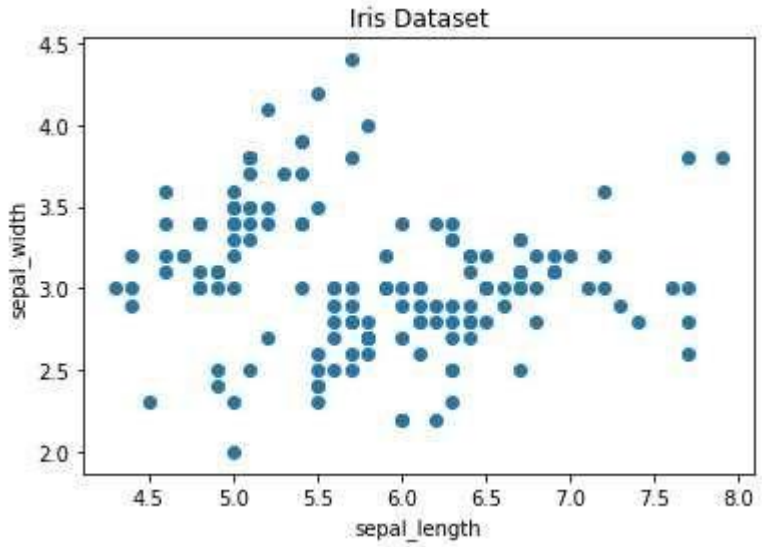


Figure 3: Matplotlib Scatter plot

We can give the graph more meaning by coloring each data point by its class. This can be done by creating a dictionary that maps from class to color and then scattering each point on its own using a for-loop and passing the respective color.

```
# create color dictionary colors = {'Iris-setosa':'r',
'Iris-versicolor':'g', 'Irisvirginica':'b'}
# create a figure and axis fig, ax =
plt.subplots() # plot each data-point for
i in range(len(iris['sepal_length'])):
    ax.scatter(iris['sepal_length'][i],
iris['sepal_width'][i],color=colors[iris['class'][i]])
# set a title and labels ax.set_title('Iris
Dataset') ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
```

Python

Copy

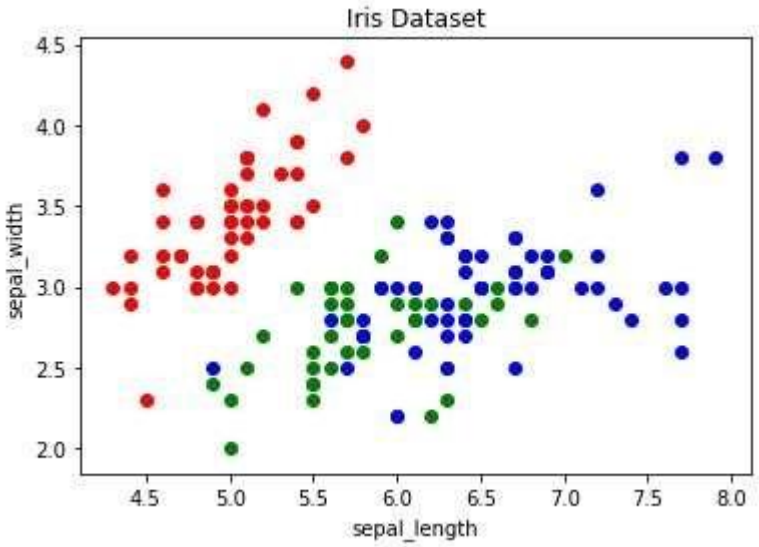


Figure 4: Scatter Plot colored by class

Line Chart

In Matplotlib, we can create a line chart by calling the plot method. We can also plot multiple columns in one graph by looping through the columns we want and plotting eachcolumn on the same axis.

```
# get columns to plot columns =
iris.columns.drop(['class'])
# create x data x_data =
range(0, iris.shape[0])
# create figure and axis fig, ax
= plt.subplots() # plot each
column for column in columns:
ax.plot(x_data, iris[column])
# set title and legend
ax.set_title('Iris Dataset')
ax.legend()
```

Python

Copy

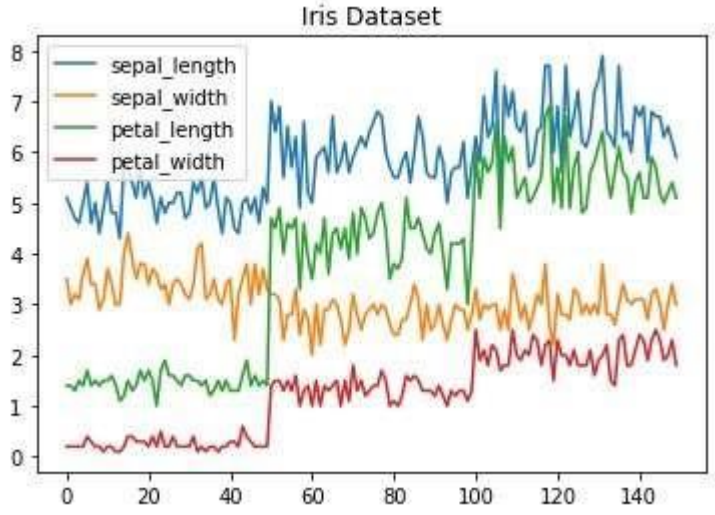


Figure 5: Line Chart

Histogram

In Matplotlib, we can create a Histogram using the hist method. If we pass categorical data like the points column from the wine-review dataset, it will automatically calculate how often each class occurs.

```
# create figure and axis fig,
ax = plt.subplots()
# plot histogram ax.hist(wine_reviews['points'])
# set title and labels ax.set_title('Wine
Review Scores') ax.set_xlabel('Points')
ax.set_ylabel('Frequency')
```

Python

Copy

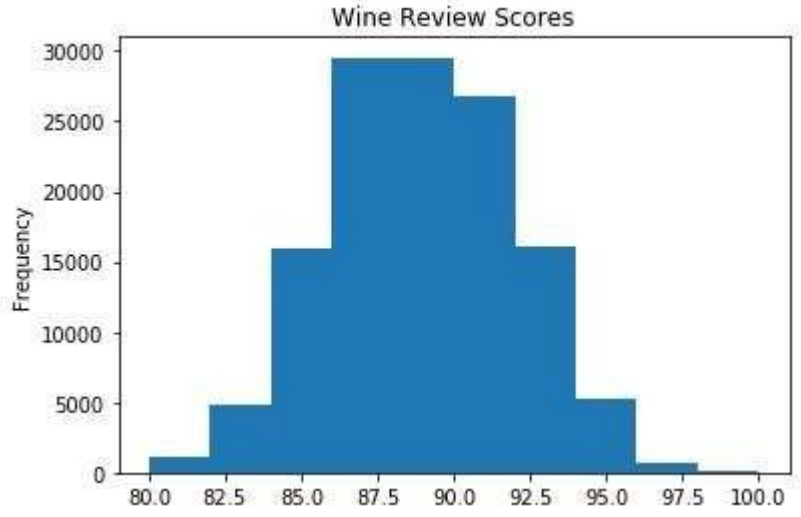


Figure 6: Histogram

Bar Chart

A bar chart can be created using the bar method. The bar chart isn't automatically calculating the frequency of a category, so we will use pandas value_counts method to do this. The bar chart is useful for categorical data that doesn't have a lot of different categories
(less than 30) because else it can get quite messy.

```
# create a figure and axis fig,
ax = plt.subplots()
# count the occurrence of each class data =
wine_reviews['points'].value_counts()
# get x and y data points =
data.index  frequency =
data.values # create bar chart
ax.bar(points, frequency) # set
title and labels
ax.set_title('Wine Review Scores')
ax.set_xlabel('Points')
ax.set_ylabel('Frequency')
```

Python

Visualizing amounts

In many scenarios, we are interested in the magnitude of some set of numbers. For example, we might want to visualize the total sales volume of different brands of cars, or the total number of people living in different cities, or the age of olympians performing different sports. In all these cases, we have a set of categories (e.g., brands of cars, cities, or sports) and a quantitative value for each category. I refer to these cases as visualizing amounts, because the main emphasis in these visualizations will be on the magnitude of the quantitative values. The standard visualization in this scenario is the bar plot, which comes in several variations, including simple bars as well as grouped and stacked bars.
Alternatives to the bar plot are the dot plot and the heatmap.

Bar plots

To motivate the concept of a bar plot, consider the total ticket sales for the most popular movies on a given weekend. Table 6.1 shows the top-five weekend gross ticket sales on the Christmas weekend of 2017. The movie "Star Wars: The Last Jedi" was by far the most popular movie on that weekend, outselling the fourth- andfifth-ranked movies "The Greatest Showman" and "Ferdinand" by almost a factor of10.

Table 6.1: Highest grossing movies for the weekend of December 22-24, 2017. Data source: Box Office Mojo (http://www.boxofficemojo.com/). Used with permission

Rank	Title	Weekend gross
1	Star Wars: The Last Jedi	$71,565,498
2	Jumanji: Welcome to the Jungle	$36,169,328
3	Pitch Perfect 3	$19,928,525
4	The Greatest Showman	$8,805,843
5	Ferdinand	$7,316,746

This kind of data is commonly visualized with vertical bars. For each movie, we drawa bar that starts at zero and extends all the way to the dollar value for that movie's weekend gross (Figure 6.1). This visualization is called a *bar plot* or *bar chart*.

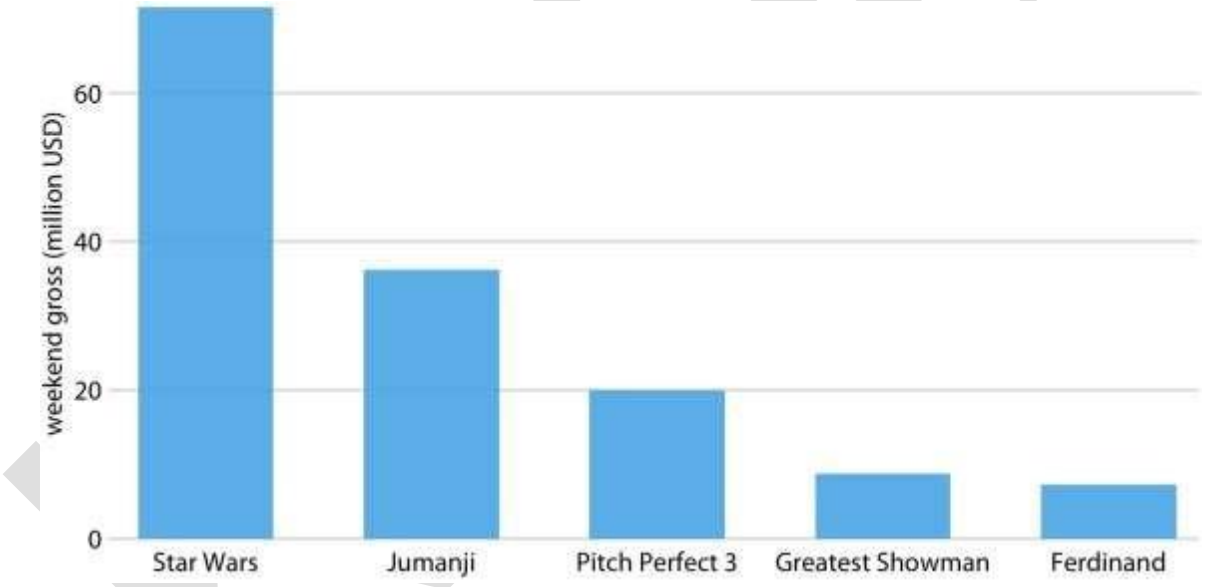


Figure 6.1: Highest grossing movies for the weekend of December 22-24, 2017, displayed as a bar plot. Data source: Box Office Mojo (http://www.boxofficemojo.com/). Used with permission

One problem we commonly encounter with vertical bars is that the labels identifyingeach bar take up a lot of horizontal space. In fact, I had to make Figure 6.1 fairly wide and space out the bars so that I could place the movie titles underneath. To save horizontal space, we could place the bars closer together and rotate the labels(Figure 6.2). However, I am not a big proponent of rotated labels. I find the resultingplots awkward and difficult to read. And, in my experience, whenever the labels are too long to place horizontally they also don't look good rotated.

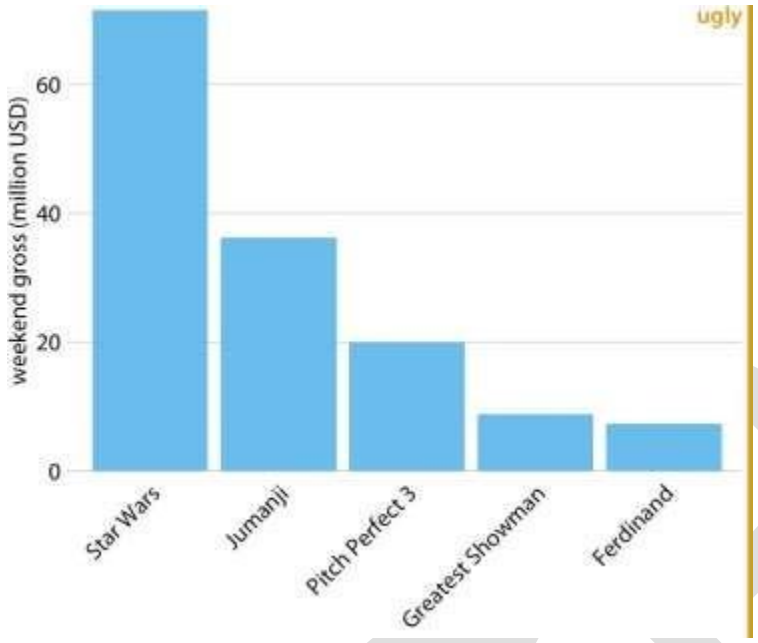


Figure 6.2: Highest grossing movies for the weekend of December 22-24, 2017, displayed as a bar plot with rotated axis tick labels. Rotated axis tick labels tend to be difficult to read and require awkward space use undearneath the plot. For these reasons, I generally consider plots with rotated tick labels to be ugly. Data source: Box Office Mojo (http://www.boxofficemojo.com/). Used with permission

The better solution for long labels is usually to swap the *x* and the *y* axis, so that the bars run horizontally (Figure 6.3). After swapping the axes, we obtain a compact figure in which all visual elements, including all text, are horizontally oriented. As a result, the figure is much easier to read than Figure 6.2 or even Figure 6.1.

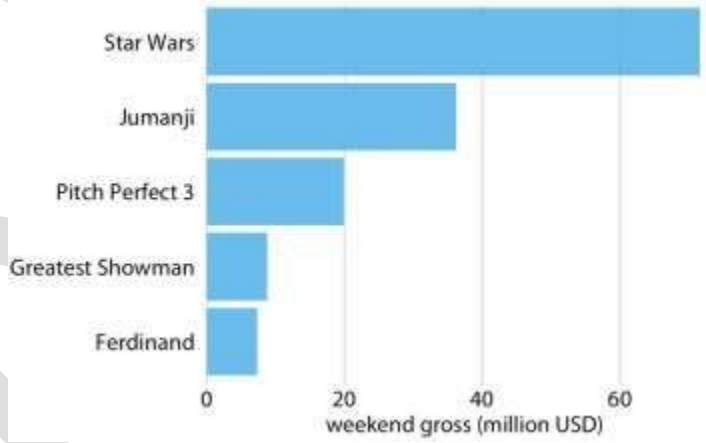


Figure 6.3: Highest grossing movies for the weekend of December 22-24, 2017, displayed as a horizontal bar plot. Data source: Box Office Mojo (http://www.boxofficemojo.com/). Used with permission

Regardless of whether we place bars vertically or horizontally, we need to pay attention to the order in which the bars are arranged. I often see bar plots where the bars are arranged arbitrarily or by some criterion that is not meaningful in the context of the figure. Some plotting programs arrange bars by default in alphabetic order of the labels, and other, similarly arbitrary arrangements are possible (Figure 6.4). In

general, the resulting figures are more confusing and less intuitive than figures where bars are arranged in order of their size.

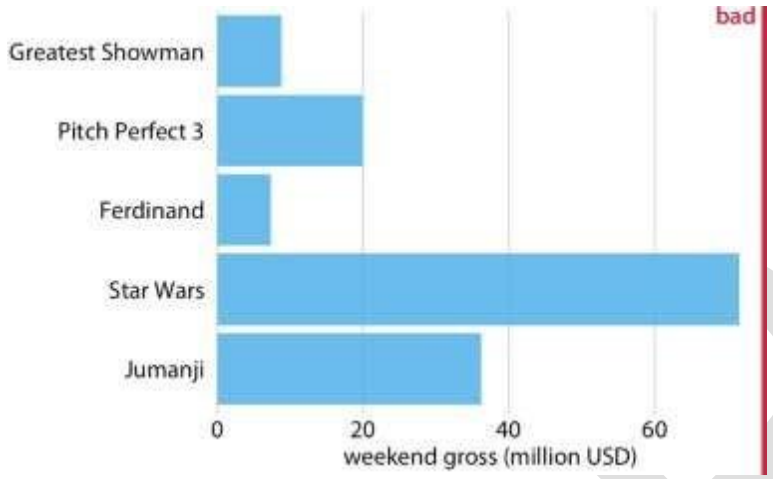


Figure 6.4: Highest grossing movies for the weekend of December 22-24, 2017, displayed as a horizontal bar plot. Here, the bars have been placed in descendingorder of the lengths of the movie titles. This arrangement of bars is arbitrary, it doesn't serve a meaningful purpose, and it makes the resulting figure much less intuitive than Figure 6.3. Data source: Box Office Mojo (http://www.boxofficemojo.com/). Used with permission

We should only rearrange bars, however, when there is no natural ordering to the categories the bars represent. Whenever there is a natural ordering (i.e., when our categorical variable is an ordered factor) we should retain that ordering in the visualization. For example, Figure 6.5 shows the median annual income in the U.S. by age groups. In this case, the bars should be arranged in order of increasing age.Sorting by bar height while shuffling the age groups makes no sense (Figure 6.6).

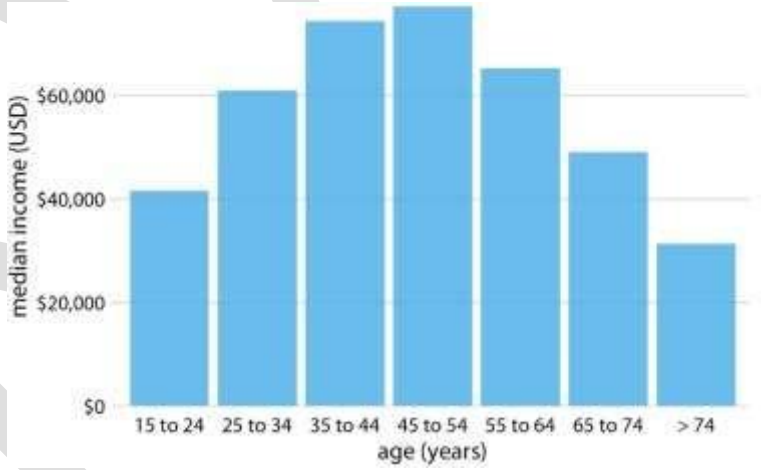


Figure 6.5: 2016 median U.S. annual household income versus age group. The 45–54 year age group has the highest median income. Data source: United StatesCensus Bureau

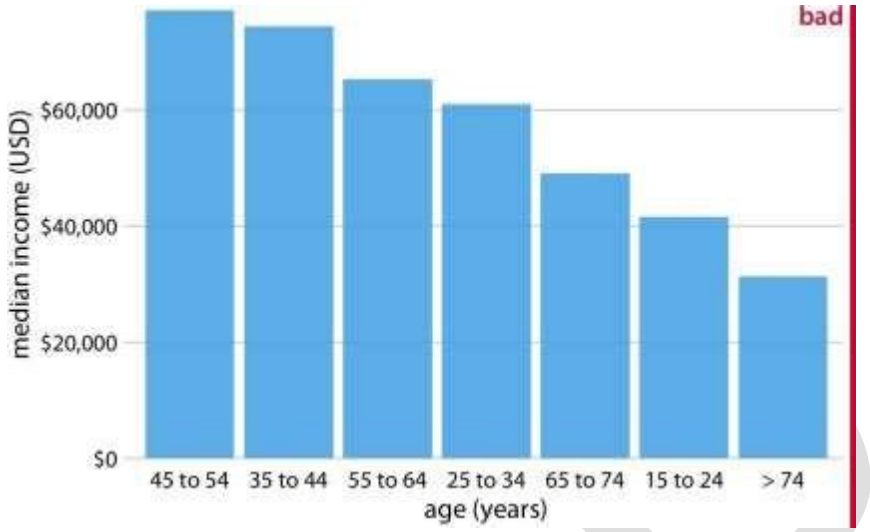


Figure 6.6: 2016 median U.S. annual household income versus age group, sorted by income. While this order of bars looks visually appealing, the order of the age groups is now confusing. Data source: United States Census Bureau

Pay attention to the bar order. If the bars represent unordered categories, order them by ascending or descending data values.

Grouped and stacked bars

All examples from the previous subsection showed how a quantitative amount varied with respect to one categorical variable. Frequently, however, we are interested in two categorical variables at the same time. For example, the U.S. Census Bureau provides median income levels broken down by both age and race. We can visualize this dataset with a *grouped bar plot* (Figure 6.7). In a grouped bar plot, we draw a group of bars at each position along the x axis, determined by one categorical variable, and then we draw bars within each group according to the other categorical variable.

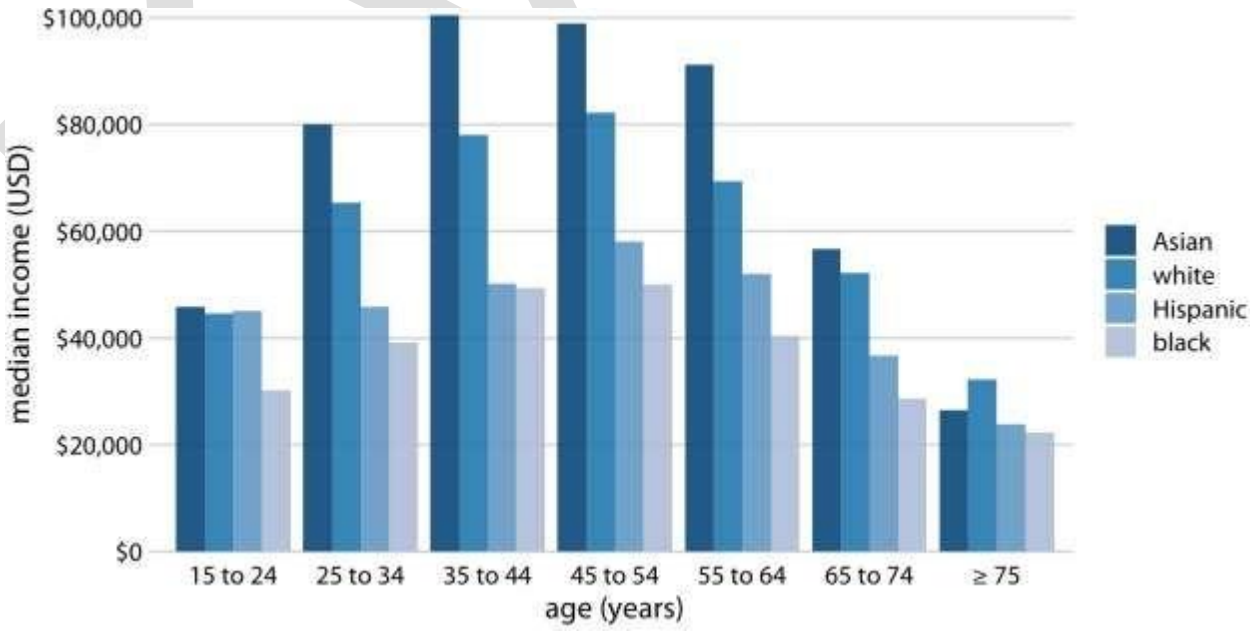


Figure 6.7: 2016 median U.S. annual household income versus age group and race. Age groups are shown along the *x* axis, and for each age group there are four bars, corresponding to the median income of Asian, white, Hispanic, and black people, respectively. Data source: United States Census Bureau

Grouped bar plots show a lot of information at once and they can be confusing. In fact, even though I have not labeled Figure 6.7 as bad or ugly, I find it difficult to read. In particular, it is difficult to compare median incomes across age groups for a given racial group. So this figure is only appropriate if we are primarily interested in the differences in income levels among racial groups, separately for specific age groups. If we care more about the overall pattern of income levels among racial groups, it may be preferable to show race along the *x* axis and show ages as distinctbars within each racial group (Figure 6.8).

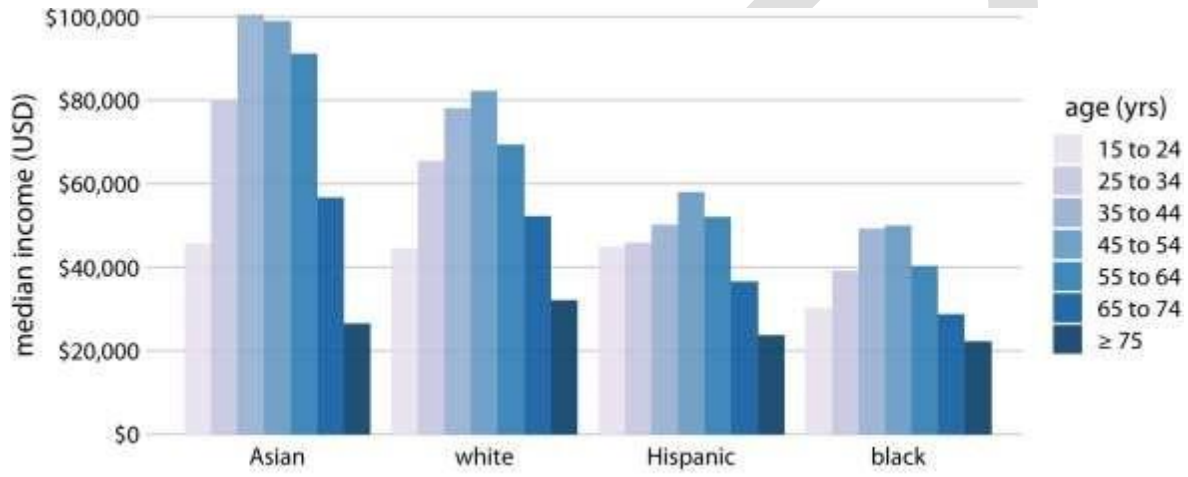


Figure 6.8: 2016 median U.S. annual household income versus age group and race. In contrast to Figure 6.7, now race is shown along the *x* axis, and for each race we show seven bars according to the seven age groups. Data source: United States Census Bureau

Both Figures 6.7 and 6.8 encode one categorical variable by position along the *x* axis and the other by bar color. And in both cases, the encoding by position is easy to read while the encoding by bar color requires more mental effort, as we have to mentally match the colors of the bars against the colors in the legend. We can avoidthis added mental effort by showing four separate regular bar plots rather than one grouped bar plot (Figure 6.9). Which of these various options we choose is ultimately a matter of taste. I would likely choose Figure 6.9, because it circumventsthe need for different bar colors.

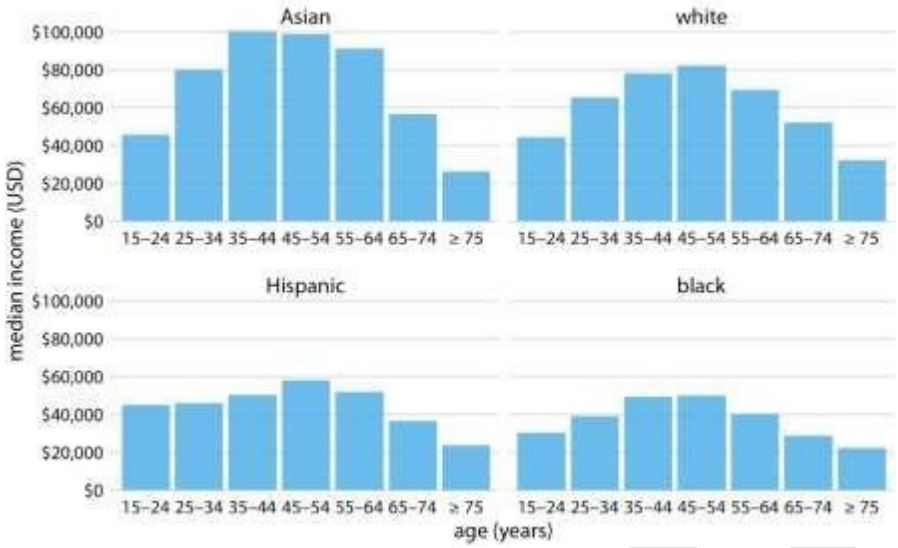


Figure 6.9: 2016 median U.S. annual household income versus age group and race. Instead of displaying this data as a grouped bar plot, as in
Figures 6.7 and 6.8, we now show the data as four separate regular bar plots. This choice has the advantage that we don't need to encode either categorical variable bybar color. Data source: United States Census Bureau

Instead of drawing groups of bars side-by-side, it is sometimes preferable to stack bars on top of each other. Stacking is useful when the sum of the amounts represented by the individual stacked bars is in itself a meaningful amount. So, whileit would not make sense to stack the median income values of Figure 6.7 (the sum of two median income values is not a meaningful value), it might make sense to stack the weekend gross values of Figure 6.1 (the sum of the weekend gross values of two movies is the total gross for the two movies combined). Stacking is also appropriate when the individual bars represent counts. For example,in a dataset of people, we can either count men and women separately or we can count them together. If we stack a bar representing a count of women on top of a bar representing a count of men, then the combined bar height represents the total count of people regardless of gender.

I will demonstrate this principle using a dataset about the passengers of the transatlantic ocean liner Titanic, which sank on April 15, 1912. On board were approximately 1300 passengers, not counting crew. The passengers were traveling in one of three classes (1st, 2nd, or 3rd), and there were almost twice as many maleas female passengers on the ship. To visualize the breakdown of passengers by class and gender, we can draw separate bars for each class and gender and stack the bars representing women on top of the bars representing men, separately for each class (Figure 6.10). The combined bars represent the total number of passengers in each class.

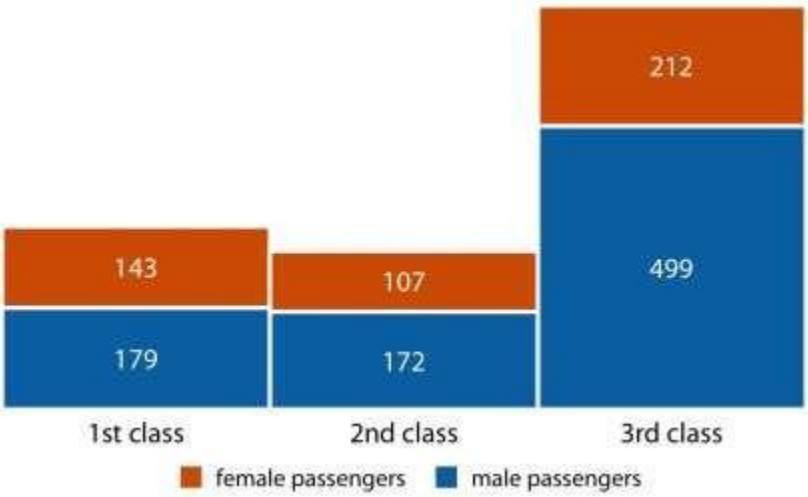


Figure 6.10: Numbers of female and male passengers on the Titanic traveling in 1st, 2nd, and 3rd class.

Figure 6.10 differs from the previous bar plots I have shown in that there is no explicit *y* axis. I have instead shown the actual numerical values that each bar represents. Whenever a plot is meant to display only a small number of different values, it makes sense to add the actual numbers to the plot. This substantially increases the amount of information conveyed by the plot without adding much visual noise, and it removes the need for an explicit *y* axis.

Dot plots and heatmaps

Bars are not the only option for visualizing amounts. One important limitation of bars is that they need to start at zero, so that the bar length is proportional to the amount shown. For some datasets, this can be impractical or may obscure key features. In this case, we can indicate amounts by placing dots at the appropriate locations along the *x* or *y* axis.

Figure 6.11 demonstrates this visualization approach for a dataset of life expectancies in 25 countries in the Americas. The citizens of these countries have life expectancies between 60 and 81 years, and each individual life expectancy value is shown with a blue dot at the appropriate location along the *x* axis. By limiting the axis range to the interval from 60 to 81 years, the figure highlights the key features of this dataset: Canada has the highest life expectancy among all listed countries, and Bolivia and Haiti have much lower life expectancies than all other countries. If we had used bars instead of dots (Figure 6.12), we'd have made a much less compelling figure. Because the bars are so long in this figure, and they all have nearly the same length, the eye is drawn to the middle of the bars rather than to their end points, and the figure fails to convey its message.

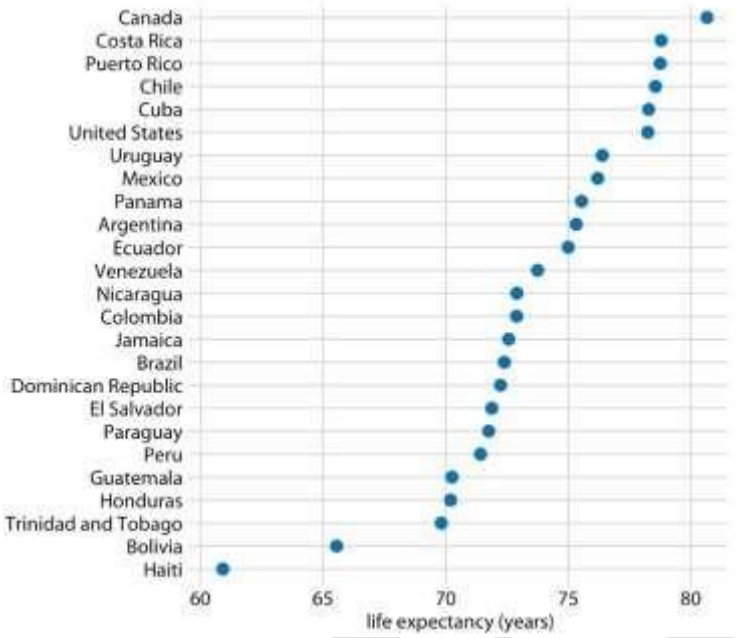


Figure 6.11: Life expectancies of countries in the Americas, for the year 2007. Data source: Gapminder project

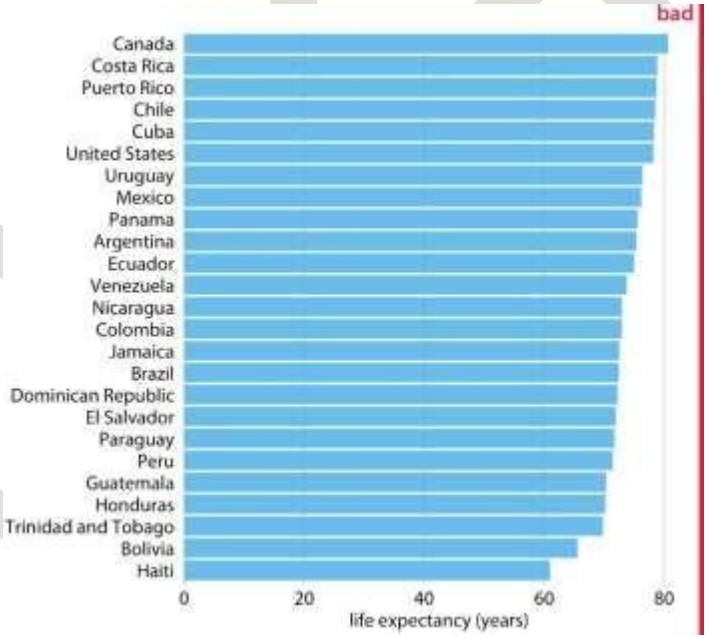


Figure 6.12: Life expectancies of countries in the Americas, for the year 2007, shown as bars. This dataset is not suitable for being visualized with bars. The bars are too long and they draw attention away from the key feature of the data, the differences in life expectancy among the different countries. Data source: Gapminder project

Regardless of whether we use bars or dots, however, we need to pay attention to the ordering of the data values. In Figures 6.11 and 6.12, the countries are ordered in descending order of life expectancy. If we instead ordered them alphabetically, we'd end up with a disordered cloud of points that is confusing and fails to convey a clear message (Figure 6.13).

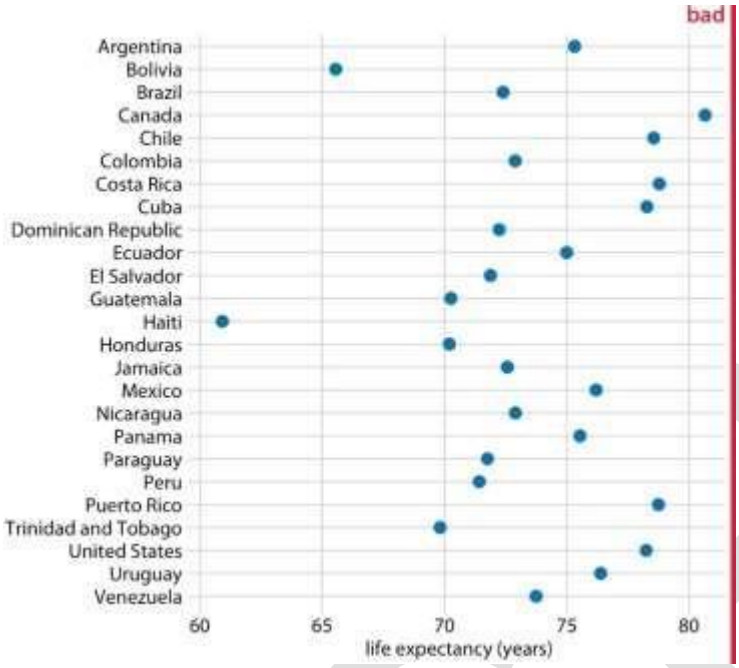


Figure 6.13: Life expectancies of countries in the Americas, for the year 2007. Here, the countries are ordered alphabetically, which causes a dots to form a disordered cloud of points. This makes the figure difficult to read, and therefore it deserves to be labeled as "bad." Data source: Gapminder project

All examples so far have represented amounts by location along a position scale, either through the end point of a bar or the placement of a dot. For very large datasets, neither of these options may be appropriate, because the resulting figure would become too busy. We had already seen in Figure 6.7 that just seven groups offour data values can result in a figure that is complex and not that easy to read. If wehad 20 groups of 20 data values, a similar figure would likely be highly confusing.

As an alternative to mapping data values onto positions via bars or dots, we can mapdata values onto colors. Such a figure is called a *heatmap*. Figure 6.14 uses this approach to show the percentage of internet users over time in 20 countries and for 23 years, from 1994 to 2016. While this visualization makes it harder to determine the exact data values shown (e.g., what's the exact percentage of internet users in the United States in 2015?), it does an excellent job of highlighting broader trends. We can see clearly in which countries internet use began early and which it did not, and we can also see clearly which countries have high internet penetration in the final year covered by the dataset (2016).

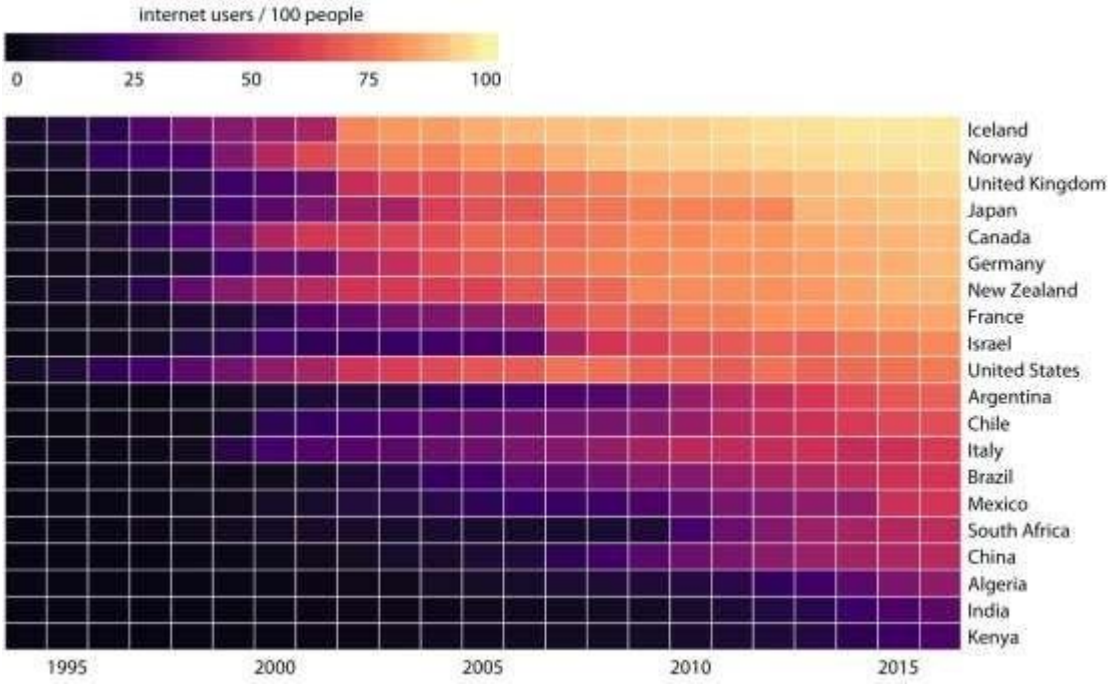


Figure 6.14: Internet adoption over time, for select countries. Color represents the percent of internet users for the respective country and year. Countries were ordered by percent internet users in 2016. Data source: World Bank

As is the case with all other visualization approaches discussed in this chapter, we need to pay attention to the ordering of the categorical data values when making heatmaps. In Figure 6.14, countries are ordered by the percentage of internet users in 2016. This ordering places the United Kingdom, Japan, Canada, and Germany above the United States, because all these countries have higher internet penetration in 2016 than the United States does, even though the United States saw significant internet use at an earlier time. Alternatively, we could order countries by how early they started to see significant internet usage. In
Figure 6.15, countries are ordered by the year in which internet usage first rose to above 20%. In this figure, the United States falls into the third position from the top, and it stands out for having relatively low internet usage in 2016 compared to how early internet usage started there. A similar pattern can be seen for Italy. Israel and France, by contrast, started relatively late but gained ground rapidly.

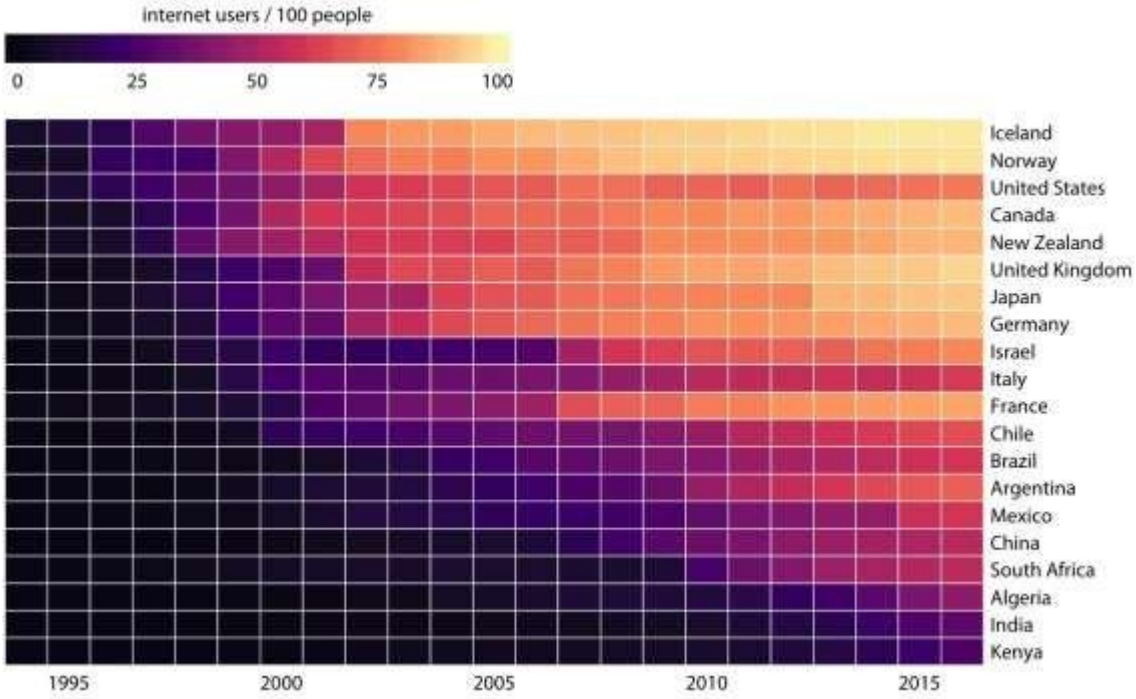
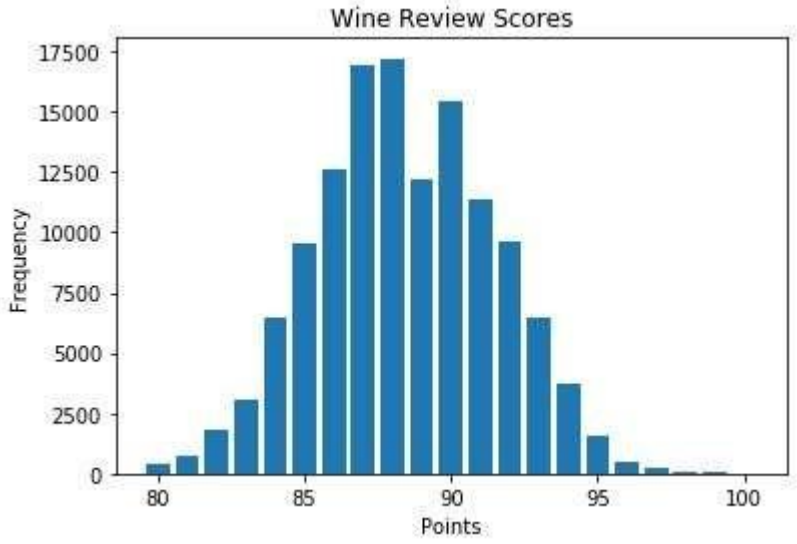


Figure 6.15: Internet adoption over time, for select countries. Countries were ordered by the year in which their internet usage first exceeded 20%. Data source: World Bank

Both Figures 6.14 and 6.15 are valid representations of the data. Which one is prefered depends on the story we want to convey. If our story is about internet usage in 2016, then Figures 6.14 is probably the better choice. If, however, our



Visualizing distributions: Histograms and density plots

We frequently encounter the situation where we would like to understand how a particular variable is distributed in a dataset. To give a concrete example, we will consider the passengers of the Titanic, a data set we encountered already in Chapter 6. There were approximately 1300 passengers on the Titanic (not counting crew), and we have reported ages for 756 of them. We might want to know how many passengers of what ages there were on the Titanic, i.e., how many children,

young adults, middle-aged people, seniors, and so on. We call the relative proportions of different ages among the passengers the *age distribution* of thepassengers.

Visualizing a single distribution

We can obtain a sense of the age distribution among the passengers by grouping allpassengers into bins with comparable ages and then counting the number of passengers in each bin. This procedure results in a table such as Table 7.1.

Table 7.1: Numbers of passenger with known age on the Titanic.

Age range	Count	Age range	Count	Age range	Count
0–5	31	31–35	61	61–65	16
6–10	36	36–40	66	66–70	3
11–15	41	41–45	71	71–75	3
16–20	46	46–50			
21–25	1	51–55	2		
26–30	12	56–60	22		

We can visualize this table by drawing filled rectangles whose heights correspondto the counts and whose widths correspond to the width of the age bins (Figure 7.1). Such a visualization is called a histogram. (Note that all bins must have the same width for the visualization to be a valid histogram.)

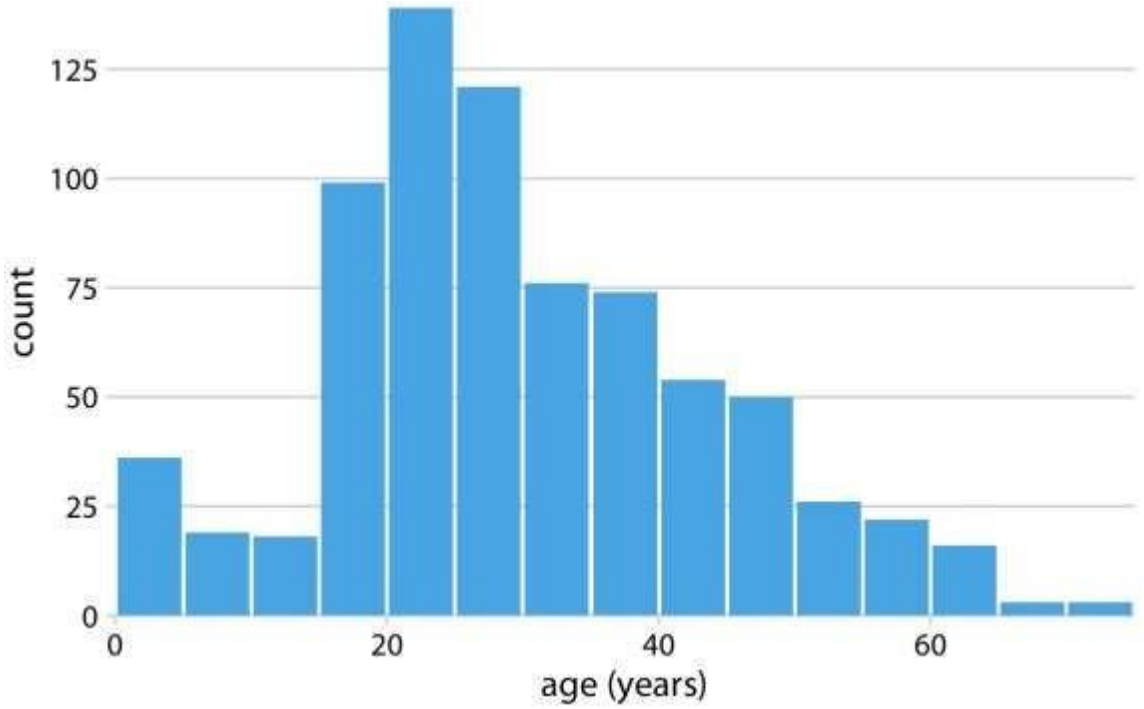


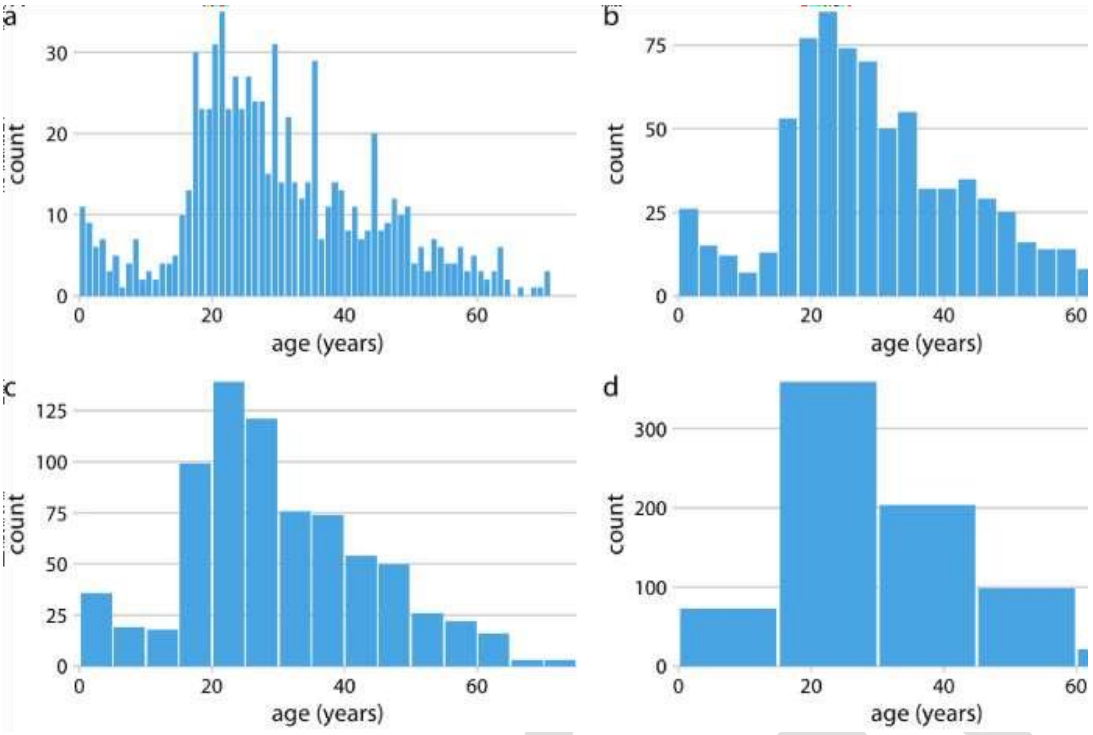
Figure 7.1: Histogram of the ages of Titanic passengers.

Because histograms are generated by binning the data, their exact visual appearance depends on the choice of the bin width. Most visualization programs thatgenerate histograms will choose a bin width by default, but chances are that bin width is not the most appropriate one for any histogram you may want to make. It is therefore critical to always try different bin widths to verify that the resulting histogram reflects the underlying data accurately. In general, if the bin width is too small, then the histogram becomes overly peaky and visually busy and the main trends in the data may be obscured. On the other hand, if the bin width is too large, then smaller features in the distribution of the data, such as the dip around age 10, may disappear.

For the age distribution of Titanic passengers, we can see that a bin width of oneyear is too small and a bin width of fifteen years is too large, whereas bin widths between three to five years work fine (Figure 7.2).

Figure 7.2: Histograms depend on the chosen bin width. Here, the same age distribution of Titanic passengers is shown with four different bin widths: (a) one year; (b) three years; (c) five years; (d) fifteen years.

When making a histogram, always explore multiple bin widths.



Histograms have been a popular visualization option since at least the 18th century, in part because they are easily generated by hand. More recently, as extensive computing power has become available in everyday devices such as laptops and cell phones, we see them increasingly being replaced by density plots. In a density plot, we attempt to visualize the underlying probability distribution of the data by drawing an appropriate continuous curve (Figure 7.3). This curve needs to be estimated from the data, and the most commonly used method for this estimation procedure is called *kernel density estimation.* In kernel density estimation, we draw acontinuous curve (the kernel) with a small width (controlled by a parameter called *bandwidth*) at the location of each data point, and then we add up all these curves toobtain the final density estimate. The most widely used kernel is a Gaussian kernel (i.e., a Gaussian bell curve), but there are many other choices.

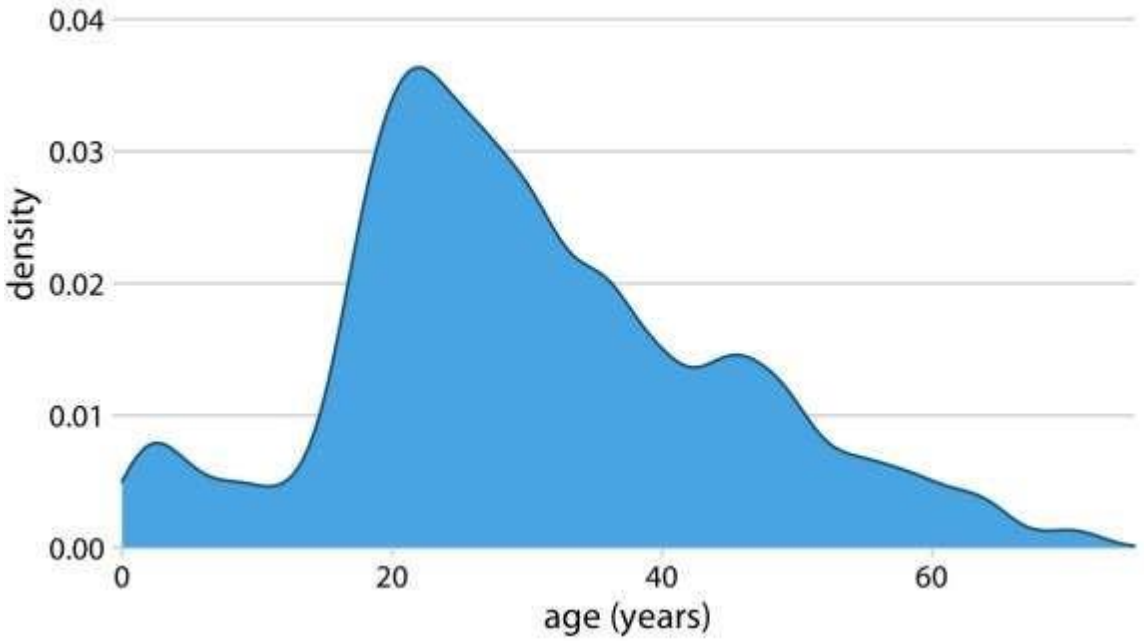
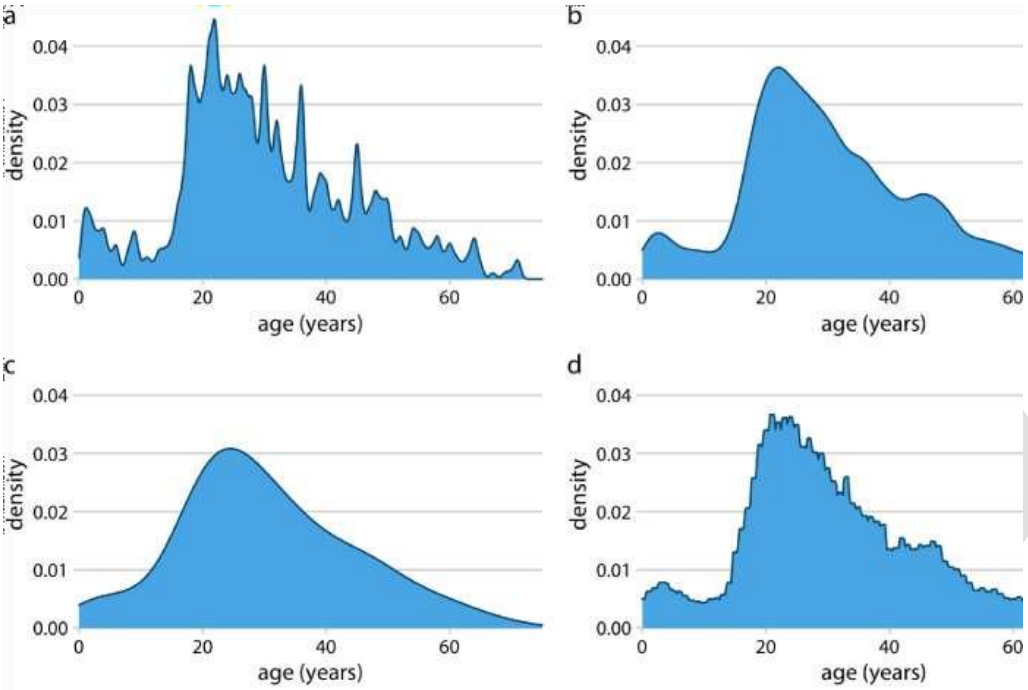


Figure 7.3: Kernel density estimate of the age distribution of passengers on the Titanic. The height of the curve is scaled such that the area under the curve equals one. The density estimate was performed with a Gaussian kernel and a bandwidth of2.

Just as is the case with histograms, the exact visual appearance of a density plot depends on the kernel and bandwidth choices (Figure 7.4). The bandwidth parameter behaves similarly to the bin width in histograms. If the bandwidth is too small, then the density estimate can become overly peaky and visually busy and the main trends in the data may be obscured. On the other hand, if the bandwidth is too large, then smaller features in the distribution of the data may disappear. In addition, the choice of the kernel affects the shape of the density curve. For example, a Gaussian kernel will have a tendency to produce density estimates that look Gaussian-like, with smooth features and tails. By contrast, a rectangular kernel can generate the appearance of steps in the density curve (Figure 7.4d). In general, the more data points there are in the data set, the less the choice of the kernel matters. Therefore, density plots tend to be quite reliable and informative for large data sets but can be misleading for data sets of only a few points.

Figure 7.4: Kernel density estimates depend on the chosen kernel and bandwidth. Here, the same age distribution of Titanic passengers is shown for four different combinations of these parameters: (a) Gaussian kernel, bandwidth = 0.5; (b) Gaussian kernel, bandwidth = 2; (c) Gaussian kernel, bandwidth = 5; (d) Rectangularkernel, bandwidth = 2.



Density curves are usually scaled such that the area under the curve equals one. This convention can make the *y* axis scale confusing, because it depends on the units of the *x* axis. For example, in the case of the age distribution, the data range onthe *x* axis goes from 0 to approximately 75. Therefore, we expect the mean height ofthe density curve to be 1/75 = 0.013. Indeed, when looking at the age density curves(e.g., Figure 7.4), we see that the *y* values range from 0 to approximately 0.04, with an average of somewhere close to 0.01.

Kernel density estimates have one pitfall that we need to be aware of: They have a tendency to produce the appearance of data where none exists, in particular in the tails. As a consequence, careless use of density estimates can easily lead to figures that make nonsensical statements. For example, if we don't pay attention, we might generate a visualization of an age distribution that includes negative ages (Figure 7.5).

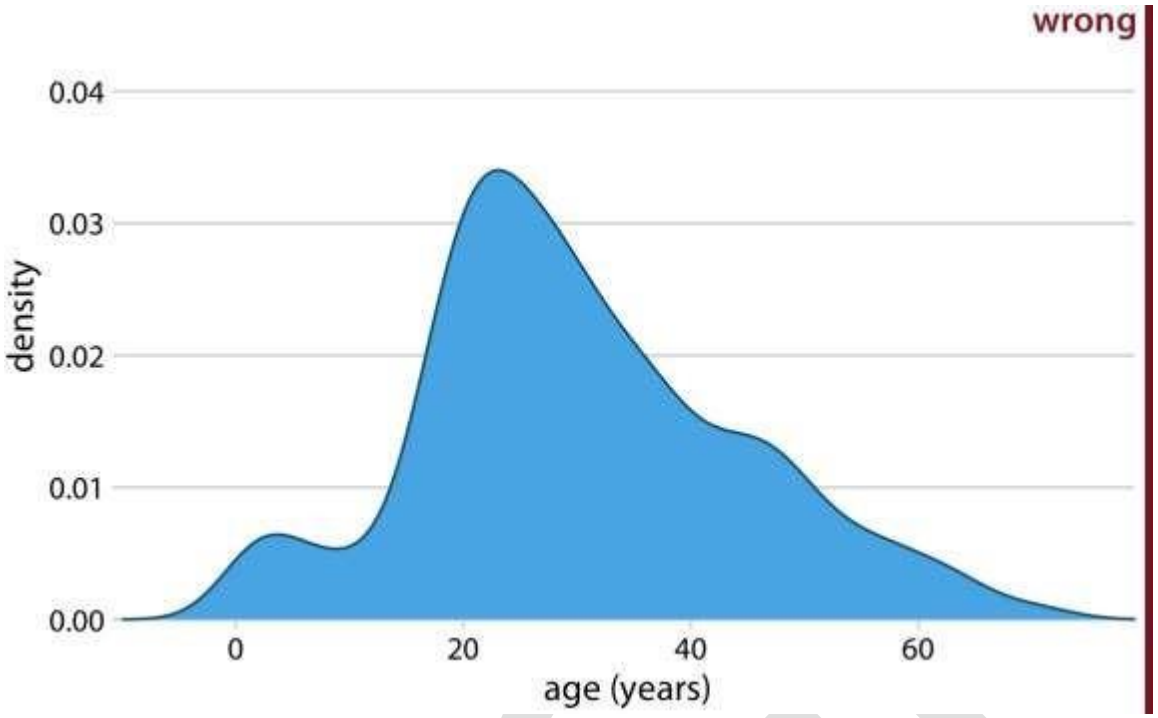


Figure 7.5: Kernel density estimates can extend the tails of the distribution into areas where no data exist and no data are even possible. Here, the density estimate has been allowed to extend into the negative age range. This is clearly nonsensical and should be avoided.

Always verify that your density estimate does not predict the existence of nonsensical data values.

So should you use a histogram or a density plot to visualize a distribution? Heated discussions can be had on this topic. Some people are vehemently against densityplots and believe that they are arbitrary and misleading. Others realize that histograms can be just as arbitrary and misleading. I think the choice is largely a matter of taste, but sometimes one or the other option may more accurately reflect the specific features of interest in the data at hand. There is also the possibility of using neither and instead choosing empirical cumulative density functions or q-q plots (Chapter 8). Finally, I believe that density estimates have an inherent advantage over histograms as soon as we want to visualize more than one distribution at a time (see next section).

Visualizing multiple distributions at the sametime

In many scenarios we have multiple distributions we would like to visualize simultaneously. For example, let's say we'd like to see how the ages of Titanic passengers are distributed between men and women. Were men and women passengers generally of the same age, or was there an age difference between the genders? One commonly employed visualization strategy in this case is a stacked

histogram, where we draw the histogram bars for women on top of the bars for men, in a different color (Figure 7.6).

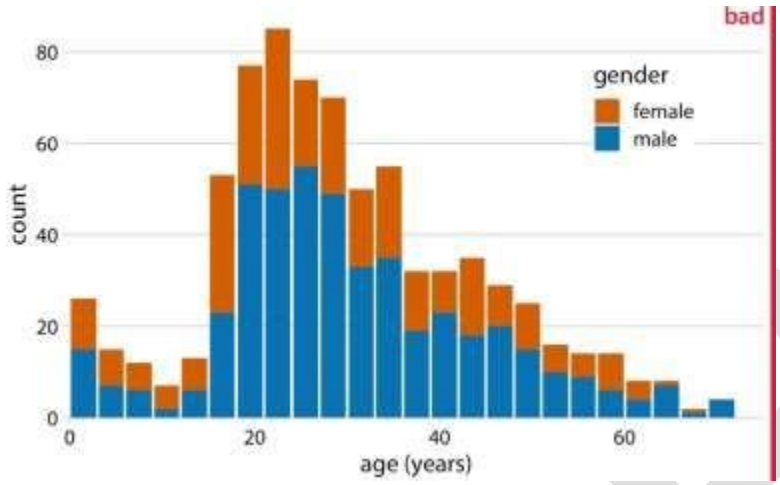


Figure 7.6: Histogram of the ages of Titanic passengers stratified by gender. This figure has been labeled as "bad" because stacked histograms are easily confused with overlapping histograms (see also Figure 7.7). In addition, the heights of the bars representing female passengers cannot easily be compared to each other.

In my opinion, this type of visualization should be avoided. There are two key problems here: First, from just looking at the figure, it is never entirely clear where exactly the bars begin. Do they start where the color changes or are they meant to start at zero? In other words, are there about 25 females of age 18–20 or are there almost 80? (The former is the case.) Second, the bar heights for the female counts cannot be directly compared to each other, because the bars all start at a different height. For example, the men were on average older than the women, and this fact is not at all visible in Figure 7.6.

We could try to address these problems by having all bars start at zero and making the bars partially transparent (Figure 7.7).

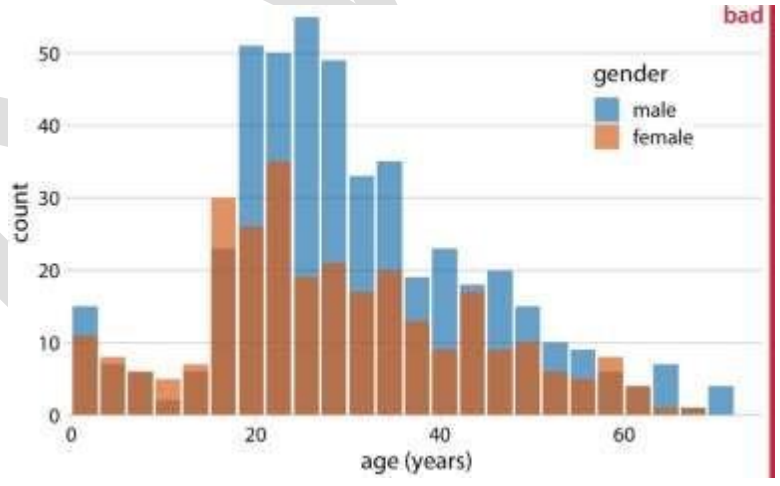


Figure 7.7: Age distributions of male and female Titanic passengers, shown as two overlapping histograms. This figure has been labeled as "bad" because there is no clear visual indication that all blue bars start at a count of 0.

However, this approach generates new problems. Now it appears that there are actually three different groups, not just two, and we're still not entirely sure where each bar starts and ends. Overlapping histograms don't work well because a semitransparent bar drawn on top of another tends to not look like a semi- transparent bar but instead like a bar drawn in a different color.

Overlapping density plots don't typically have the problem that overlapping histograms have, because the continuous density lines help the eye keep the distributions separate. However, for this particular dataset, the age distributions for male and female passengers are nearly identical up to around age 17 and then diverge, so that the resulting visualization is still not ideal (Figure 7.8).

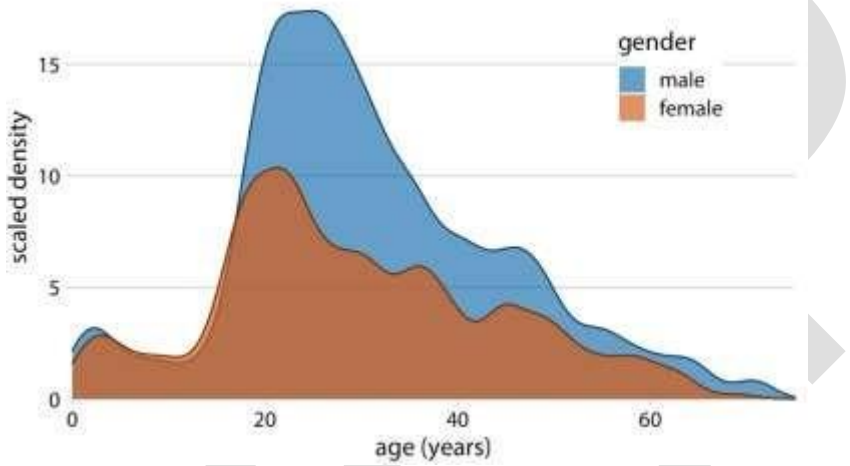


Figure 7.8: Density estimates of the ages of male and female Titanic passengers. To highlight that there were more male than female passengers, the density curves were scaled such that the area under each curve corresponds to the total number ofmale and female passengers with known age (468 and 288, respectively).

A solution that works well for this dataset is to show the age distributions of male and female passengers separately, each as a proportion of the overall age distribution (Figure 7.9). This visualization shows intuitively and clearly that there were many fewer women than men in the 20–50-year age range on the Titanic.

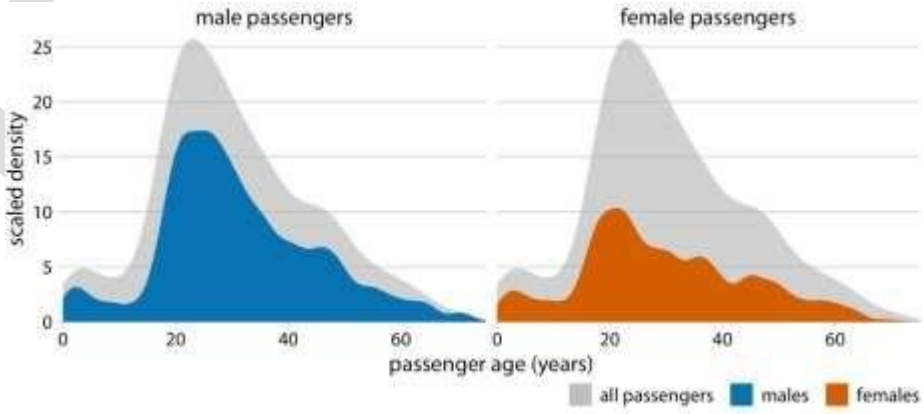


Figure 7.9: Age distributions of male and female Titanic passengers, shown as proportion of the passenger total. The colored areas show the density estimates of

the ages of male and female passengers, respectively, and the gray areas show the overall passenger age distribution.

Finally, when we want to visualize exactly two distributions, we can also make two separate histograms, rotate them by 90 degrees, and have the bars in one histogram point into the opposite direction of the other. This trick is commonly employed when visualizing age distributions, and the resulting plot is usually called an *age pyramid* (Figure 7.10).

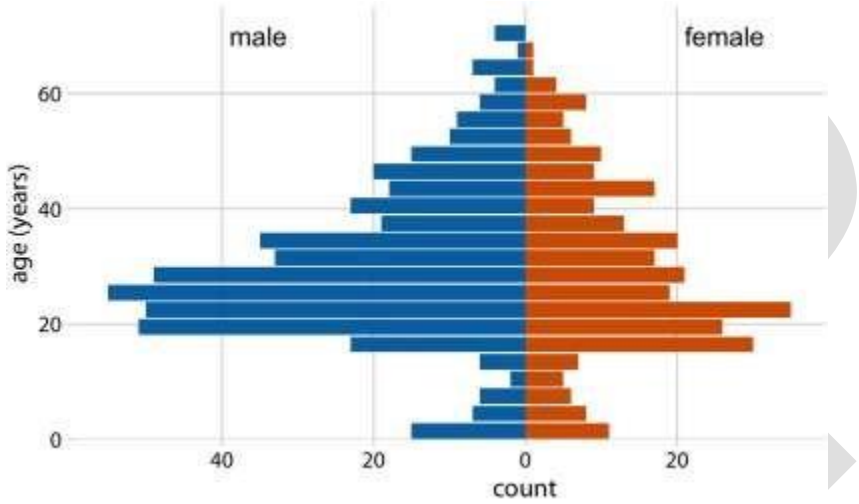


Figure 7.10: The age distributions of male and female Titanic passengers visualized as an age pyramid.

Importantly, this trick does not work when there are more than two distributions we want to visualize at the same time. For multiple distributions, histograms tend to become highly confusing, whereas density plots work well as long as the distributions are somewhat distinct and contiguous. For example, to visualize the distribution of butterfat percentage among cows from four different cattle breeds, density plots are fine (Figure 7.11).

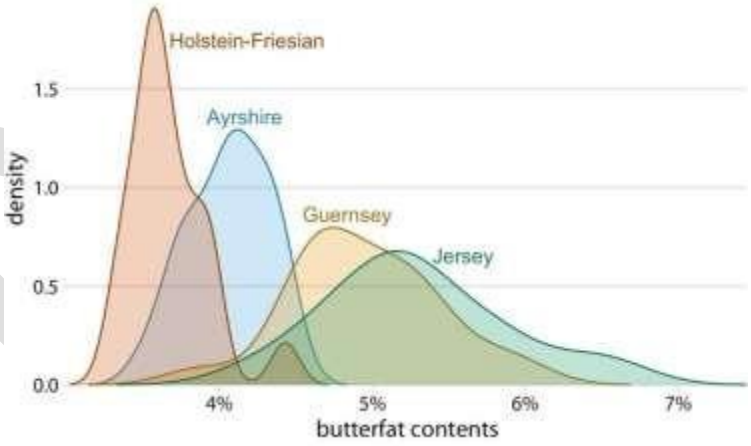


Figure 7.11: Density estimates of the butterfat percentage in the milk of four cattle breeds. Data Source: Canadian Record of Performance for Purebred Dairy Cattle

Visualizing proportions

A case for pie charts

From 1961 to 1983, the German parliament (called the *Bundestag*) was composed of members of three different parties, CDU/CSU, SPD, and FDP. During most of this time, CDU/CSU and SPD had approximately comparable numbers of seats, while the FDP typically held only a small fraction of seats. For example, in the 8th Bundestag, from 1976–1980, the CDU/CSU held 243 seats, SPD 214, and FDP 39, for a total of 496. Such parliamentary data is most commonly visualized as a pie chart (Figure 10.1).

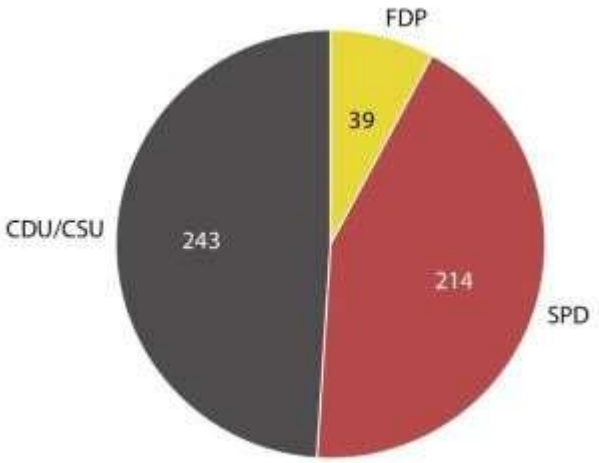


Figure 10.1: Party composition of the 8th German Bundestag, 1976–1980, visualized as a pie chart. This visualization shows clearly that the ruling coalition of SPD and FDP had a small majority over the opposition CDU/CSU.

A pie chart breaks a circle into slices such that the area of each slice is proportional to the fraction of the total it represents. The same procedure can be performed on a rectangle, and the result is a stacked bar chart (Figure 10.2). Depending on whether we slice the bar vertically or horizontally, we obtain vertically stacked bars (Figure 10.2a) or horizontally stacked bars (Figure 10.2b).

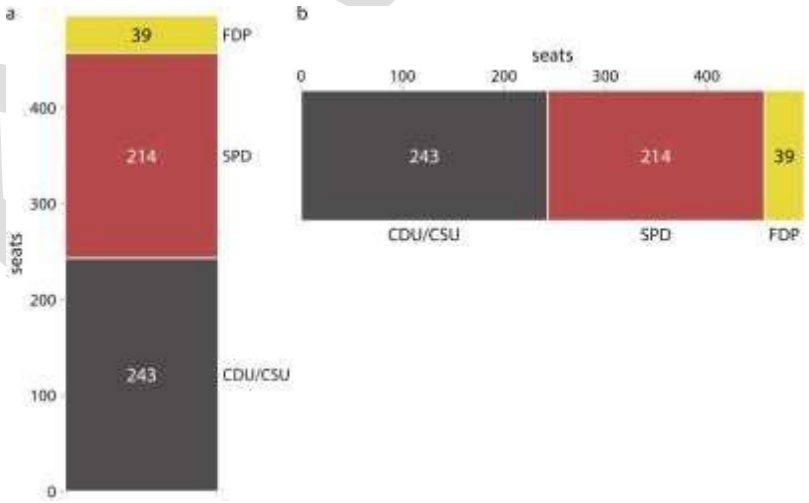


Figure 10.2: Party composition of the 8th German Bundestag, 1976–1980, visualizedas stacked bars. (a) Bars stacked vertically. (b) Bars stacked horizontally. It is not immediately obvious that SPD and FDP jointly had more seats than CDU/CSU.

We can also take the bars from Figure 10.2a and place them side-by-side rather than stacking them on top of each other. This visualization makes it easier to perform a direct comparison of the three groups, though it obscures other aspects ofthe data (Figure 10.3). Most importantly, in a side-by-side bar plot the relationship ofeach bar to the total is not visually obvious.

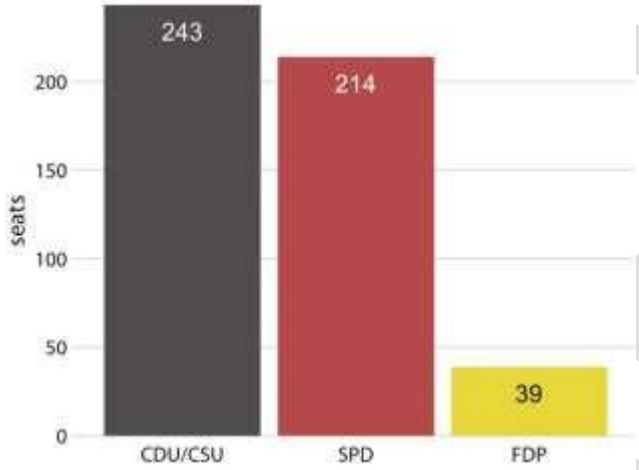


Figure 10.3: Party composition of the 8th German Bundestag, 1976–1980, visualizedas side-by-side bars. As in Figure 10.2, it is not immediately obvious that SPD and FDP jointly had more seats than CDU/CSU.

Many authors categorically reject pie charts and argue in favor of side-by-side or stacked bars. Others defend the use of pie charts in some applications. My own opinion is that none of these visualizations is consistently superior over any other. Depending on the features of the dataset and the specific story you want to tell, you may want to favor one or the other approach. In the case of the 8th German Bundestag, I think that a pie chart is the best option. It shows clearly that the ruling coalition of SPD and FDP jointly had a small majority over the CDU/CSU (Figure 10.1). This fact is not visually obvious in any of the other plots (Figures 10.2 and 10.3).

In general, pie charts work well when the goal is to emphasize simple fractions, suchas one-half, one-third, or one-quarter. They also work well when we have very smalldatasets. A single pie chart, as in Figure 10.1, looks just fine, but a single column of stacked bars, as in Figure 10.2a, looks awkward. Stacked bars, on the other hand, can work for side-by-side comparisons of multiple conditions or in a time series, and side-by-side bars are preferred when we want to directly compare the individual fractions to each other. A summary of the various pros and cons of pie charts, stacked bars, and side-by-side bars is provided in Table 10.1.

Table 10.1: Pros and cons of common approaches to visualizing proportions: pie charts, stacked bars, and side-by-side bars.

	Pie chart	Stacked bars	Side-by-side bars
Clearly visualizes the data as proportions of a whole	✓	✓	✕
Allows easy visual comparison of the relative proportions	✕	✕	✓
Visually emphasizes simple fractions, such as 1/2, 1/3, 1/4	✓	✕	✕
Looks visually appealing even for very small datasets	✓	✕	✓
Works well when the whole is broken into many pieces	✕	✕	✓
Works well for the visualization of many sets of proportions or time series of proportions	✕	✓	✕

A case for side-by-side bars

I will now demonstrate a case where pie charts fail. This example is modeled after a critique of pie charts originally posted on Wikipedia (Wikipedia 2007). Consider the hypothetical scenario of five companies, A, B, C, D, and E, who all have roughly comparable market share of approximately 20%. Our hypothetical dataset lists the market share of each company for three consecutive years. When we visualize this dataset with pie charts, it is difficult to see what exactly is going on (Figure 10.4). It appears that the market share of company A is growing and the one of company E isshrinking, but beyond this one observation we can't tell what's going on. In particular,it is unclear how exactly the market shares of the different companies compare within each year.

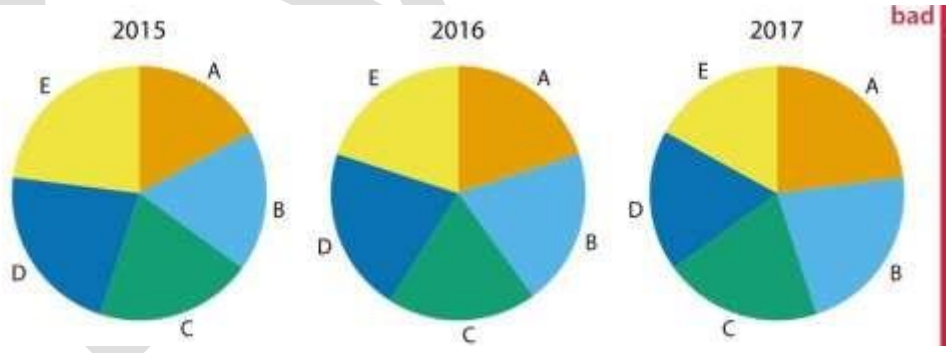


Figure 10.4: Market share of five hypothetical companies, A–E, for the years 2015–2017, visualized as pie charts. This visualization has two major problems: 1.
A comparison of relative market share within years is nearly impossible. 2.Changes in market share across years are difficult to see.

The picture becomes a little clearer when we switch to stacked bars (Figure 10.5).Now the trends of a growing market share for company A and a shrinking market

share for company E are clearly visible. However, the relative market shares of the five companies within each year are still hard to compare. And it is difficult to compare the market shares of companies B, C, and D across years, because the bars are shifted relative to each other across years. This is a general problem of stacked-bar plots, and the main reason why I normally do not recommend this typeof visualization.

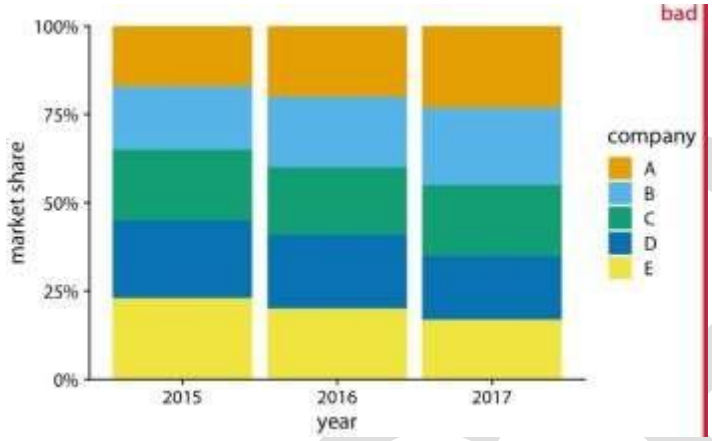


Figure 10.5: Market share of five hypothetical companies for the years 2015–2017, visualized as stacked bars. This visualization has two major problems: 1. A comparison of relative market shares within years is difficult. 2. Changes in market share across years are difficult to see for the middle companies B, C, and D, because the location of the bars changes across years.

For this hypothetical data set, side-by-side bars are the best choice (Figure 10.6). This visualization highlights that both companies A and B have increased their market share from 2015 to 2017 while both companies D and E have reduced theirs. It also shows that market shares increase sequentially from company A to E in 2015and similarly decrease in 2017.

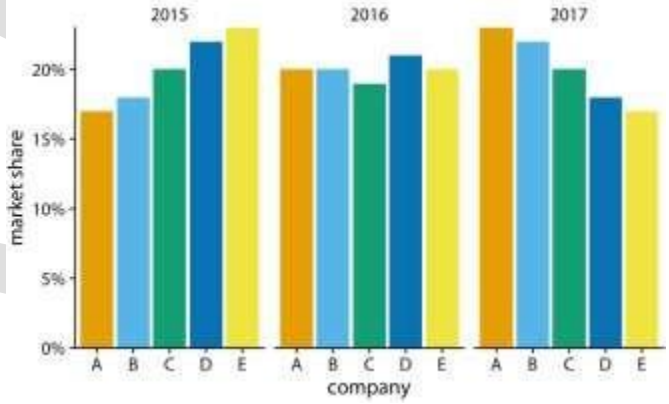


Figure 10.6: Market share of five hypothetical companies for the years 2015–2017,visualized as side-by-side bars.

A case for stacked bars and stackeddensities

In Section 10.2, I wrote that I don't normally recommend sequences of stacked bars, because the location of the internal bars shifts along the sequence.
However, the problem of shifting internal bars disappears if there are only two bars ineach stack, and in those cases the resulting visualization can be quite clear. As an example, consider the proportion of women in a country's national parliament. We will specifically look at the African country Rwanda, which as of 2016 tops the list of countries with the highest proportion of female parliament members. Rwanda has had a majority female parliament since 2008, and since 2013 nearly two-thirds of its members of parliament are female. To visualize how the proportion of women in the Rwandan parliament has changed over time, we can draw a sequence of stacked bar graphs (Figure 10.7). This figure provides an immediate visual representation of the changing proportions over time. To help the reader see exactly when the majorityturned female, I have added a dashed horizontal line at 50%. Without this line, it would be near impossible to determine whether from 2003 to 2007 the majority was male or female. I have not added similar lines at 25% and 75%, to avoid making the figure too cluttered.

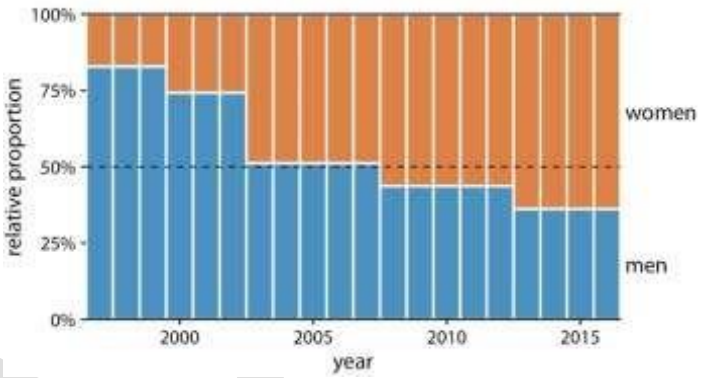


Figure 10.7: Change in the gender composition of the Rwandan parliament overtime, 1997 to 2016. Data source: Inter-Parliamentary Union (IPU), ipu.org.

If we want to visualize how proportions change in response to a continuous variable, we can switch from stacked bars to stacked densities. Stacked densities can be thought of as the limiting case of infinitely many infinitely small stacked bars arranged side-by-side. The densities in stacked-density plots are typically obtained from kernel density estimation, as described in Chapter 7, and I refer you to that chapter for a general discussion of the strengths and weaknesses of this method.

To give an example where stacked densities may be appropriate, consider the healthstatus of people as a function of age. Age can be considered a continuous variable, and visualizing the data in this way works reasonably well (Figure 10.8). Even though we have four health categories here, and I'm generally not a fan of stacking multiple conditions, as discussed above, I think in this case the figure is acceptable. We can see clearly that overall health declines as people age, and we can also see that despite this trend, over half of the population remain in good or excellent health until very old age.

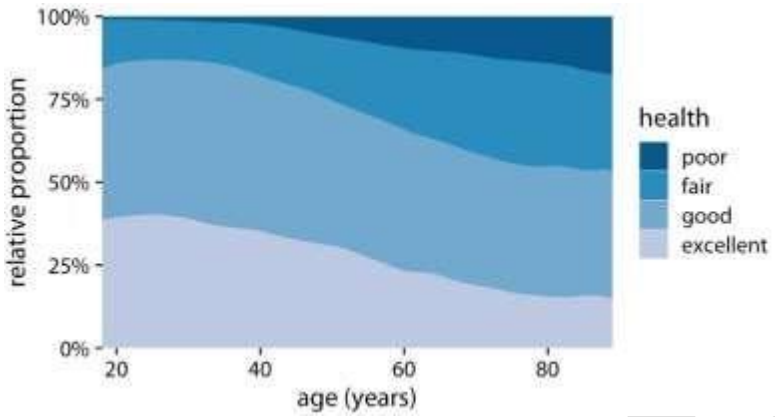


Figure 10.8: Health status by age, as reported by the general social survey (GSS).

Nevertheless, this figure has a major limitation: By visualizing the proportions of the four health conditions as percent of the total, the figure obscures that there are manymore young people than old people in the dataset. Thus, even though the *percentage* of people reporting to be in good health remains approximately unchanged across ages spanning seven decades, the *absolute number* of people in good health declines as the total number of people at a given age declines. I will present a potential solution to this problem in the next section.

Visualizing proportions separately as partsof the total

Side-by-side bars have the problem that they don't clearly visualize the size of the individual parts relative to the whole and stacked bars have the problem that the different bars cannot be compared easily because they have different baselines. We can resolve these two issues by making a separate plot for each part and in each plot showing the respective part relative to the whole. For the health dataset of Figure 10.8, this procedure results in Figure 10.9. The overall age distribution in the dataset is shown as the shaded gray areas, and the age distributions for each healthstatus are shown in blue. This figure highlights that in absolute terms, the number people with excellent or good health declines past ages 30–40, while the number of people with fair health remains approximately constant across all ages.

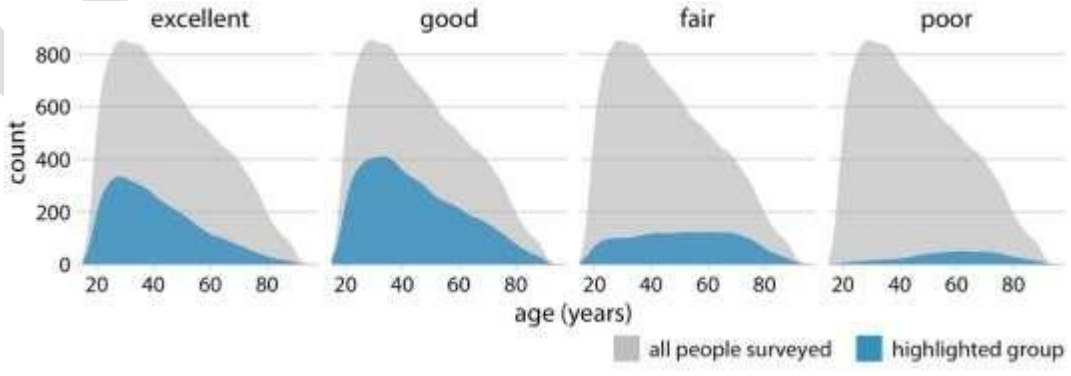


Figure 10.9: Health status by age, shown as proportion of the total number of people in the survey. The colored areas show the density estimates of the ages of people

with the respective health status and the gray areas show the overall agedistribution.

To provide a second example, let's consider a different variable from the same survey: marital status. Marital status changes much more drastically with age than does healthstatus, and a stacked densities plot of marital status vs age is not very illuminating (Figure 10.10).

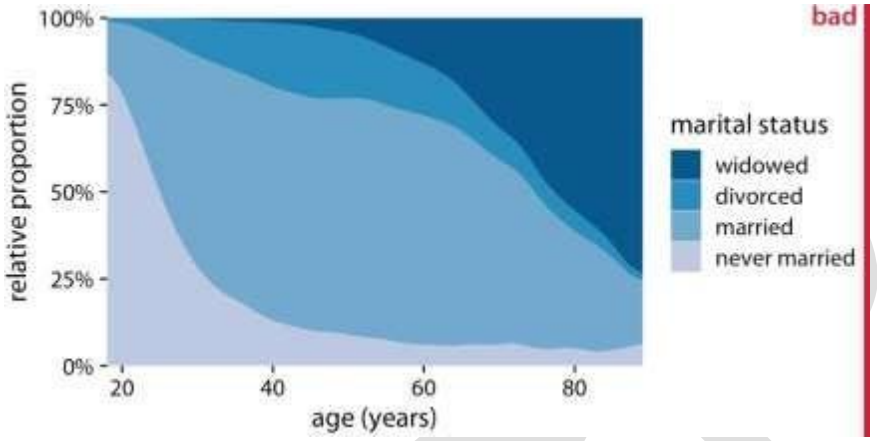


Figure 10.10: Marital status by age, as reported by the general social survey (GSS). To simplify the figure, I have removed a small number of cases that report as separated. I have labeled this figure as "bad" because the frequency of people who have never been married or are widowed changes so drastically with age that the age distributions of married and divorced people are highly distorted and difficult to interpret.

The same dataset visualized as partial densities is much clearer (Figure 10.11). In particular, we see that the proportion of married people peaks around the late 30s, the proportion of divorced people peaks around the early 40s, and the proportion ofwidowed people peaks around the mid 70s.

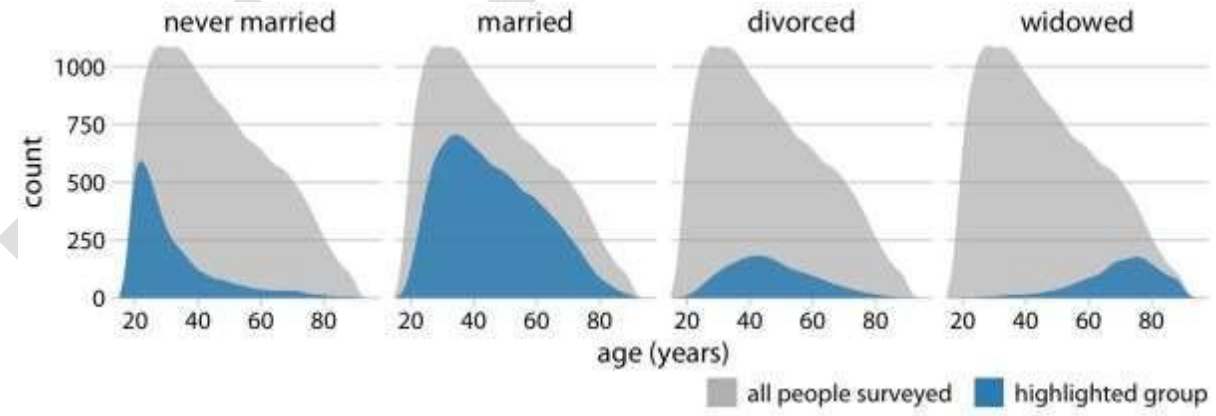


Figure 10.11: Marital status by age, shown as proportion of the total number of people in the survey. The colored areas show the density estimates of the ages ofpeople with the respective marital status, and the gray areas show the overall agedistribution.

However, one downside of Figure 10.11 is that this representation doesn't make it easy to determine relative proportions at any given point in time. For example, if we

wanted to know at what age more than 50% of all people surveyed are married, we could not easily tell from Figure 10.11. To answer this question, we can instead usethe same type of display but show relative proportions instead of absolute counts along the *y* axis (Figure 10.12). Now we see that married people are in the majority starting in their late 20s, and widowed people are in the majority starting in their mid70s.

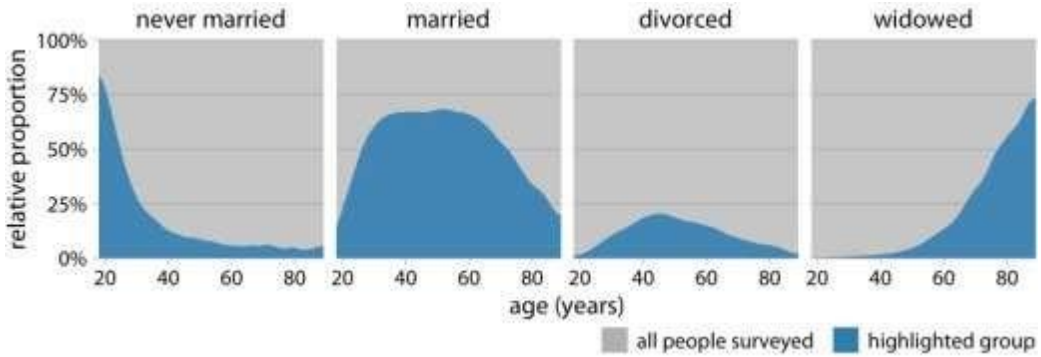


Figure 10.12: Marital status by age, shown as proportion of the total number of people in the survey. The areas colored in blue show the percent of people at the given age with the respective status, and the areas colored in gray show the percentof people with all other marital statuses

Visualizing associations among two or more quantitative variables

Many datasets contain two or more quantitative variables, and we may be interested in how these variables relate to each other. For example, we may have a dataset of quantiative measurements of different animals, such as the animals' height, weight, length, and daily energy demands. To plot the relationship of just two such variables,
e.g. the height and weight, we will normally use a scatter plot. If we want to show more than two variables at once, we may opt for a bubble chart, a scatter plot matrix, or a correlogram. Finally, for very high-dimensional datasets, it may be useful to perform dimension reduction, for example in the form of principal components analysis.

Scatter plots

I will demonstrate the basic scatter plot and several variations thereof using a dataset of measurements performed on 123 blue jay birds. The dataset contains information such as the head length (measured from the tip of the bill to the back ofthe head), the skull size (head length minus bill length), and the body mass of each bird. We expect that there are relationships between these variables. For example,birds with longer bills would be expected to have larger skull sizes, and birds with higher body mass should have larger bills and skulls than birds with lower body mass.

To explore these relationships, I begin with a plot of head length against body mass(Figure 12.1). In this plot, head length is shown along the *y* axis, body mass along the *x* axis, and each bird is represented by one dot. (Note the terminology: We say

that we plot the variable shown along the *y* axis against the variable shown along the*x* axis.) The dots form a dispersed cloud (hence the term *scatter plot*), yet undoubtedly there is a trend for birds with higher body mass to have longer heads.
The bird with the longest head falls close to the maximum body mass observed, andthe bird with the shortest head falls close to the minimum body mass observed.

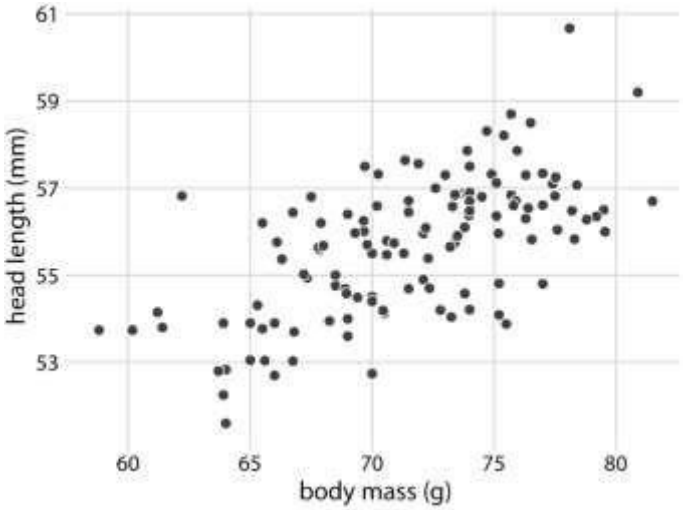


Figure 12.1: Head length (measured from the tip of the bill to the back of the head, inmm) versus body mass (in gram), for 123 blue jays. Each dot corresponds to one bird. There is a moderate tendency for heavier birds to have longer heads. Data source: Keith Tarvin, Oberlin College

The blue jay dataset contains both male and female birds, and we may want to knowwhether the overall relationship between head length and body mass holds up separately for each sex. To address this question, we can color the points in the scatter plot by the sex of the bird (Figure 12.2). This figure reveals that the overall trend in head length and body mass is at least in part driven by the sex of the birds. At the same body mass, females tend to have shorter heads than males. At the same time, females tend to be lighter than males on average.

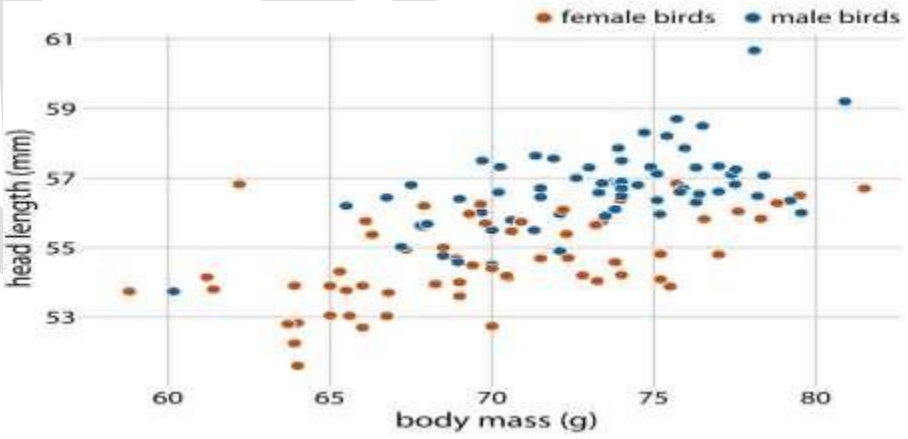


Figure 12.2: Head length versus body mass for 123 blue jays. The birds' sex is indicated by color. At the same body mass, male birds tend to have longer heads (and specifically, longer bills) than female birds. Data source: Keith Tarvin, OberlinCollege

Because the head length is defined as the distance from the tip of the bill to the back of the head, a larger head length could imply a longer bill, a larger skull, or both. We can disentangle bill length and skull size by looking at another variable in the dataset, the skull size, which is similar to the head length but excludes the bill. As weare already using the *x* position for body mass, the *y* position for head length, and the dot color for bird sex, we need another aesthetic to which we can map skull size. One option is to use the size of the dots, resulting in a visualization called a *bubble chart* (Figure 12.3).

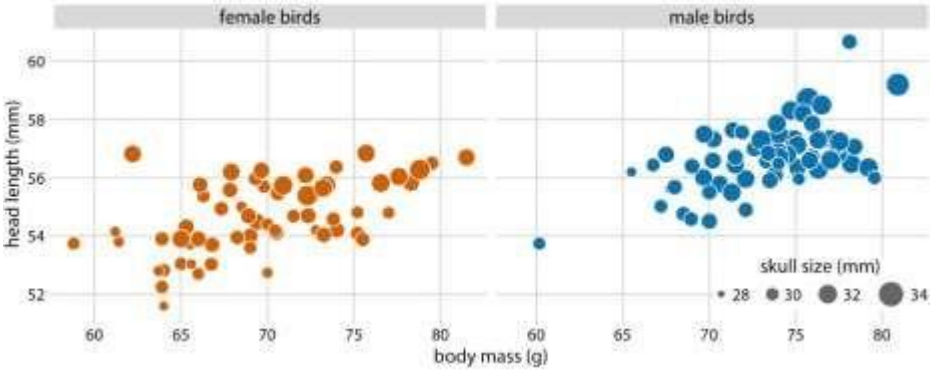


Figure 12.3: Head length versus body mass for 123 blue jays. The birds' sex is indicated by color, and the birds' skull size by symbol size. Head-length measurements include the length of the bill while skull-size measurements do not. Head length and skull size tend to be correlated, but there are some birds with unusually long or short bills given their skull size. Data source: Keith Tarvin, OberlinCollege

Bubble charts have the disadvantage that they show the same types of variables, quantitative variables, with two different types of scales, position and size. This makes it difficult to visually ascertain the strengths of associations between the various variables. Moreover, differences between data values encoded as bubble size are harder to perceive than differences between data values encoded as position. Because even the largest bubbles need to be somewhat small compared to the total figure size, the size differences between even the largest and the smallest bubbles are necessarily small. Consequently, smaller differences in data values will correspond to very small size differences that can be virtually impossible to see. In Figure 12.3, I used a size mapping that visually amplified the difference between the smallest skulls (around 28mm) and the largest skulls (around 34mm), and yet it is difficult to determine what the relationship is between skull size and either body mass or head length.

As an alternative to a bubble chart, it may be preferable to show an all-against-all matrix of scatter plots, where each individual plot shows two data dimensions (Figure 12.4). This figure shows clearly that the relationship between skull size and body mass is comparable for female and male birds except that the female birds tend to be somewhat smaller. However, the same is not true for the relationship between head length and body mass. There is a clear separation by sex. Male birds tend to have longer bills than female birds, all else equal.

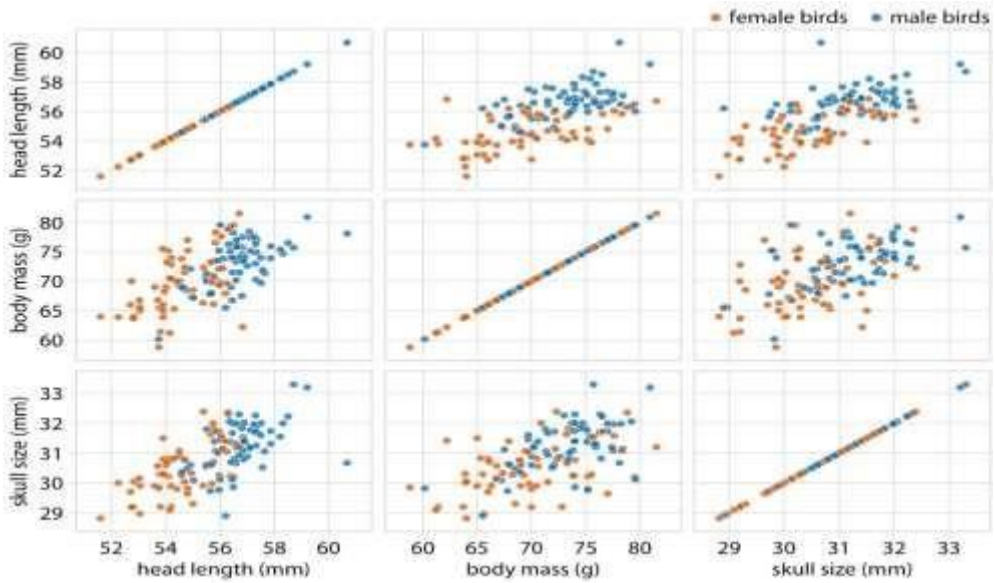


Figure 12.4: All-against-all scatter plot matrix of head length, body mass, and skullsize, for 123 blue jays. This figure shows the exact same data as Figure 12.2.
However, because we are better at judging position than symbol size, correlations between skull size and the other two variables are easier to perceive in the pairwise scatter plots than in Figure 12.2. Data source: Keith Tarvin, Oberlin College

Correlograms

When we have more than three to four quantiative variables, all-against-all scatter plot matrices quickly become unwieldy. In this case, it is more useful to quantify the amount of association between pairs of variables and visualize this quantity rather than the raw data. One common way to do this is to calculate *correlation coefficients*. The correlation coefficient r is a number between -1 and 1 that measures to what extent two variables covary. A value of $r = 0$ means there is no association whatsoever, and a value of either 1 or -1 indicates a perfect association. The sign of the correlation coefficient indicates whether the variables are *correlated* (larger values in one variable coincide with larger values in the other) or *anticorrelated* (larger values in one variable coincide with smaller values in the other). To provide visual examples of what different correlation strengths look like, in Figure 12.5 I show randomly generated sets of points that differ widely in the degree to which the x and y values are correlated.

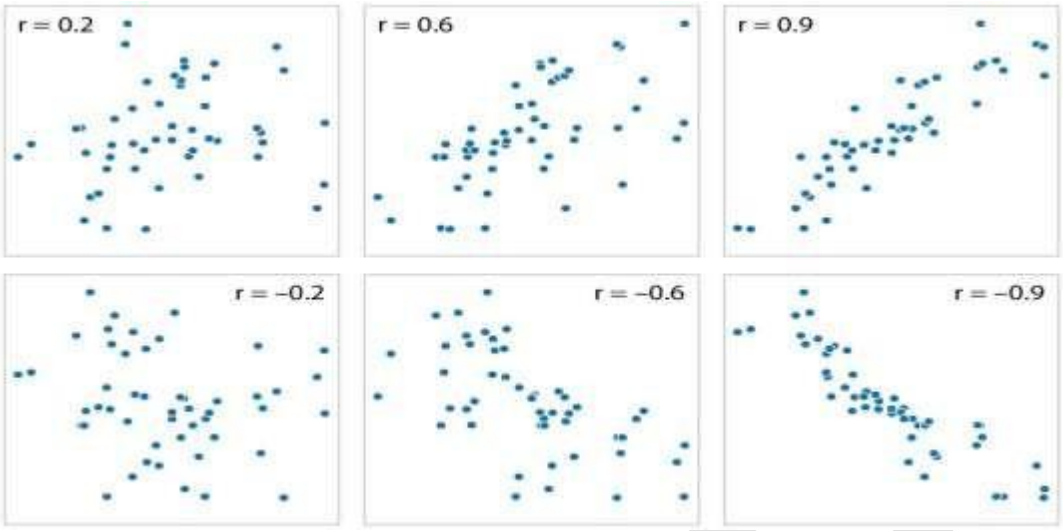


Figure 12.5: Examples of correlations of different magnitude and direction, with associated correlation coefficient *r*. In both rows, from left to right correlations go from weak to strong. In the top row the correlations are positive (larger values for one quantity are associated with larger values for the other) and in the bottom row they are negative (larger values for one quantity are associated with smaller values for the other). In all six panels, the sets of *x* and *y* values are identical, but the pairings between individual *x* and *y* values have been reshuffled to generate the specified correlation coefficients. The correlation coefficient is defined as

$r=\sum i(xi-\bar{}x)(yi-\bar{}y)\sqrt{\sum i(xi-\bar{}x)2}\sqrt{\sum i(yi-\bar{}y)2}$,$r=\sum i(xi-x\bar{})(yi-y\bar{})\sum i(xi-x\bar{})2\sum i$ ($yi-y\bar{})2$,where xixi and yiyi are two sets of observations and $\bar{}xx\bar{}$ and $\bar{}yy\bar{}$ are the corresponding sample means. We can make a number of observations from this formula. First, the formula is symmetric in xixi and yiyi, so the correlation of *x* with *y* is the same as the correlation of *y* with *x*. Second, the individual values xixiand yiyi only enter the formula in the context of differences to the respective sample mean, so if we shift an entire dataset by a constant amount,

e.g. we replace xixi with $x'i=xi+Cxi'=xi+C$ for some constant CC, the correlation coefficient remains unchanged. Third, the correlation coefficient also remains unchanged if we rescale the data, $x'i=Cxixi'=Cxi$, since the constant CC will appear both in the numerator and the denominator of the formula and hence can becancelled.

Visualizations of correlation coefficients are called *correlograms*. To illustrate the use of a correlogram, we will consider a data set of over 200 glass fragments obtained during forensic work. For each glass fragment, we have measurements about its composition, expressed as the percent in weight of various mineral oxides. There are seven different oxides for which we have measurements, yielding a total of 6 + 5 + 4 + 3 + 2 + 1 = 21 pairwise correlations. We can display these 21 correlations atonce as a matrix of colored tiles, where each tile represents one correlation coefficient (Figure 12.6). This correlogram allows us to quickly grasp trends in the data, such as that magnesium is negatively correlated with nearly all other oxides,and that aluminum and barium have a strong positive correlation.

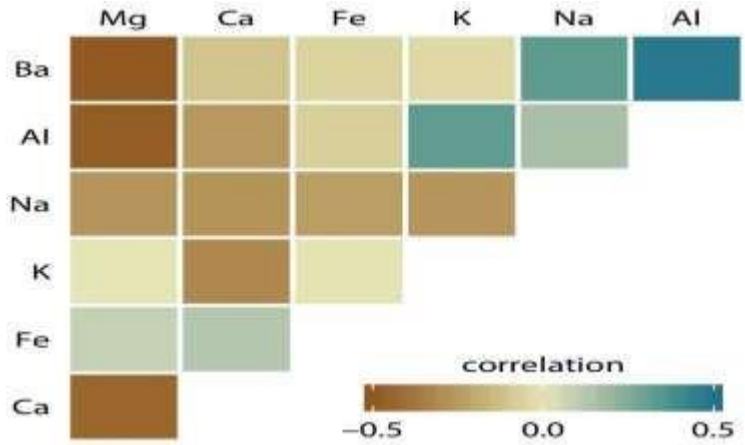


Figure 12.6: Correlations in mineral content for 214 samples of glass fragments obtained during forensic work. The dataset contains seven variables measuring the amounts of magnesium (Mg), calcium (Ca), iron (Fe), potassium (K), sodium (Na), aluminum (Al), and barium (Ba) found in each glass fragment. The colored tiles represents the correlations between pairs of these variables. Data source: B. German

One weakness of the correlogram of Figure 12.6 is that low correlations, i.e. correlations with absolute value near zero, are not as visually suppressed as they should be. For example, magnesium (Mg) and potassium (K) are not at all correlated but Figure 12.6 doesn't immediately show this. To overcome this limitation, we can display the correlations as colored circles and scale the circle size with the absolute value of the correlation coefficient (Figure 12.6). In this way, low correlations are suppressed and high correlations stand out better.

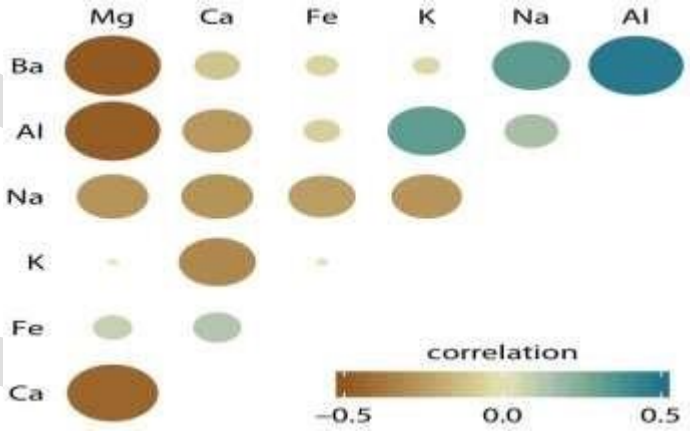


Figure 12.7: Correlations in mineral content for forensic glass samples. The color scale is identical to Figure 12.6. However, now the magnitude of each correlation is also encoded in the size of the colored circles. This choice visually deemphasizes cases with correlations near zero. Data source: B. German

All correlograms have one important drawback: They are fairly abstract. While they show us important patterns in the data, they also hide the underlying data points and may cause us to draw incorrect conclusions. It is always better to visualize the raw data rather than abstract, derived quantities that have been calculated from it.

Fortunately, we can frequently find a middle ground between showing important patterns and showing the raw data by applying techniques of dimension reduction.

Dimension reduction

Dimension reduction relies on the key insight that most high-dimensional datasets consist of multiple correlated variables that convey overlapping information. Such datasets can be reduced to a smaller number of key dimensions without loss of much critical information. As a simple, intuitive example, consider a dataset of multiple physical traits of people, including quantities such as each person's height and weight, the lengths of the arms and legs, the circumferences of waist, hip, and chest, etc. We can understand immediately that all these quantities will relate first and foremost to the overall size of each person. All else being equal, a larger personwill be taller, weigh more, have longer arms and legs, and larger waist, hip, and chest circumferences. The next important dimension is going to be the person's sex. Male and female measurements are substantially different for persons of comparable size. For example, a woman will tend to have higher hip circumference than a man, all else being equal.

There are many techniques for dimension reduction. I will discuss only one technique here, the most widely used one, called *principal components analysis* (PCA). PCA introduces a new set of variables (called principal components, PCs) by linear combination of the original variables in the data, standardized to zero mean and unit variance (see Figure 12.8 for a toy example in two dimensions). The PCs are chosen such that they are uncorrelated, and they are ordered such that the first component captures the largest possible amount of variation in the data, and subsequent components capture increasingly less. Usually, key features in the data can be seen from only the first two or three PCs.

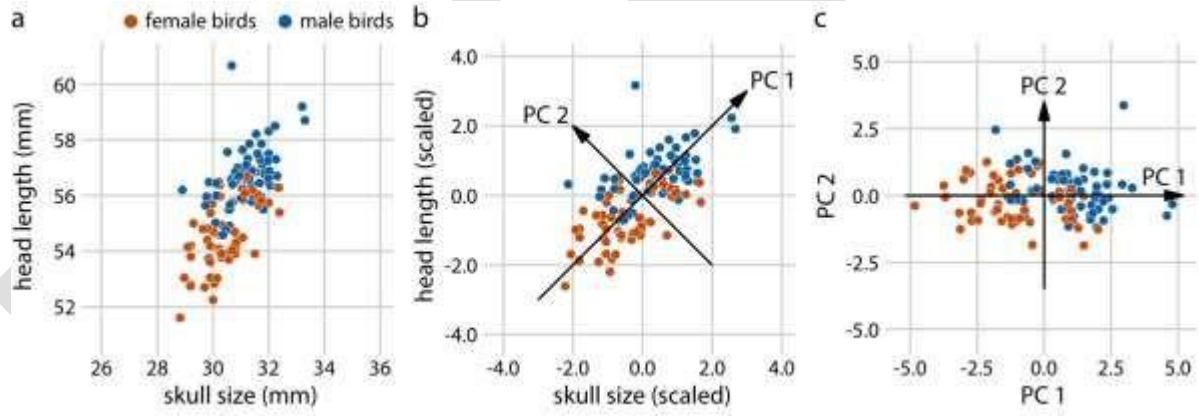


Figure 12.8: Example principal components (PC) analysis in two dimensions. (a) Theoriginal data. As example data, I am using the head-length and skull-size measurements from the blue jays dataset. Female and male birds are distinguished by color, but this distinction has no effect on the PC analysis. (b) As the first step in PCA, we scale the original data values to zero mean and unit variance. We then we define new variables (the principal components, PCs) along the directions of maximum variation in the data. (c) Finally, we project the data into the new coordinates. Mathematically, this projection is equivalent to a rotation of the data

points around the origin. In the 2D example shown here, the data points are rotatedclockwise by 45 degrees.

When we perform PCA, we are generally interested in two pieces of information: (i)the composition of the PCs and (ii) the location of the individual data points in the principal components space. Let's look at these two pieces in a PC analysis of the forensic glass dataset.

First, we look at the component composition (Figure 12.9). Here, we only consider the first two components, PC 1 and PC 2. Because the PCs are linear combinationsof the original variables (after standardization), we can represent the original variables as arrows indicating to what extent they contribute to the PCs. Here, we see that barium and sodium contribute primarily to PC 1 and not to PC 2, calcium and potassium contribute primarily to PC 2 and not to PC 1, and the other variables contribute in varying amounts to both components (Figure 12.9). The arrows are of varying lengths because there are more than two PCs. For example the arrow for iron is particularly short because it contributes primarily to higherorder PCs (not shown).

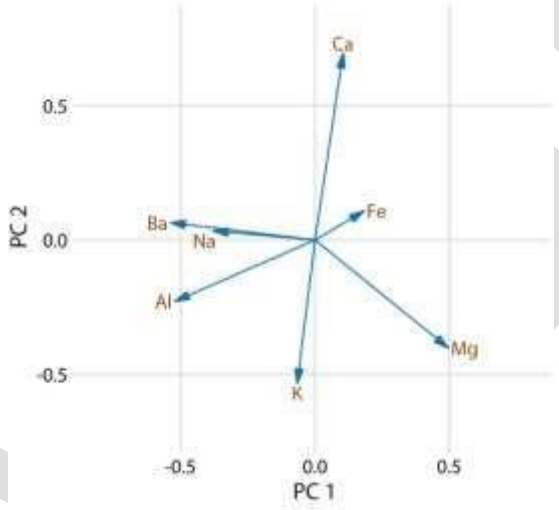


Figure 12.9: Composition of the first two components in a principal components analysis (PCA) of the forensic glass dataset. Component one (PC 1) measures primarily the amount of aluminum, barium, sodium, and magnesium contents in a glass fragment, whereas component two (PC 2) measures primarily the amount ofcalcium and potassium content, and to some extent the amount of aluminum and magnesium.

Next, we project the original data into the principal components space (Figure 12.10). We see a clear clustering of distinct types of glass fragments in this plot. Fragments from both headlamps and windows fall into clearly delineated regions in the PC plot, with few outliers. Fragments from tableware and from containers are a little more spread out, but nevertheless clearly distinct from both headlamp and window fragments. By comparing Figure 12.10 with Figure 12.9, we can conclude that window samples tend to have higher than average magnesium content and lower than average barium, aluminum, and sodium content, whereas the opposite istrue for headlamp samples.

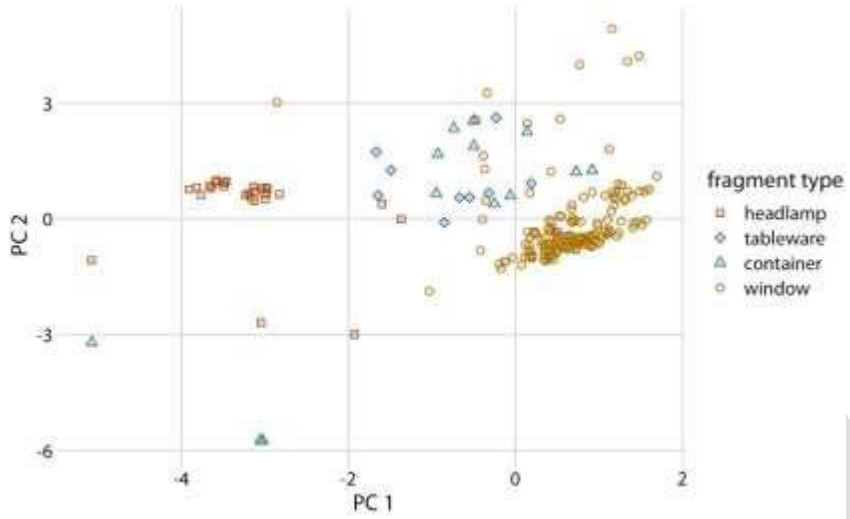


Figure 12.10: Composition of individual glass fragments visualized in the principal components space defined in Figure 12.9. We see that the different types of glass samples cluster at characteristic values of PC 1 and 2. In particular, headlamps are characterized by a negative PC 1 value whereas windows tend to have a positive PC1 value. Tableware and containers have PC 1 values close to zero and tend to have positive PC 2 values. However, there are a few exceptions where container fragments have both a negative PC 1 value and a negative PC 2 value. These are fragments whose composition drastically differs from all other fragments analyzed.

Paired data

A special case of multivariate quantitative data is paired data: Data where there are two or more measurements of the same quantity under slightly different conditions. Examples include two comparable measurements on each subject (e.g., the length of the right and the left arm of a person), repeat measurements on the same subject at different time points (e.g., a person's weight at two different times during the year),or measurements on two closely related subjects (e.g., the heights of two identical twins). For paired data, it is reasonable to assume that the two measurements belonging to a pair are more similar to each other than to the measurements belonging to other pairs. Two twins will be approximately of the same height but will differ in height from other twins. Therefore, for paired data, we need to choose visualizations that highlight any differences between the paired measurements.

An excellent choice in this case is a simple scatter plot on top of a diagonal line marking $x = y$. In such a plot, if the only difference between the two measurements ofeach pair is random noise, then all points in the sample will be scattered symmetrically around this line. Any systematic differences between the paired measurements, by contrast, will be visible in a systematic shift of the data points up or down relative to the diagonal. As an example, consider the carbon dioxide (CO_2) emissions per person, measured for 166 countries both in 1970 and in 2010 (Figure 12.11). This example highlights two common features of paired data. First, most points are relatively close to the diagonal line. Even though CO_2 emissions vary over nearly four orders of magnitude among countries, they are fairly consistent within each country over a 40-year time span. Second, the points are systematically shifted

upwards relative to the diagonal line. The majority of countries has seen an increasein CO_2 emissions over the 40 years considered.

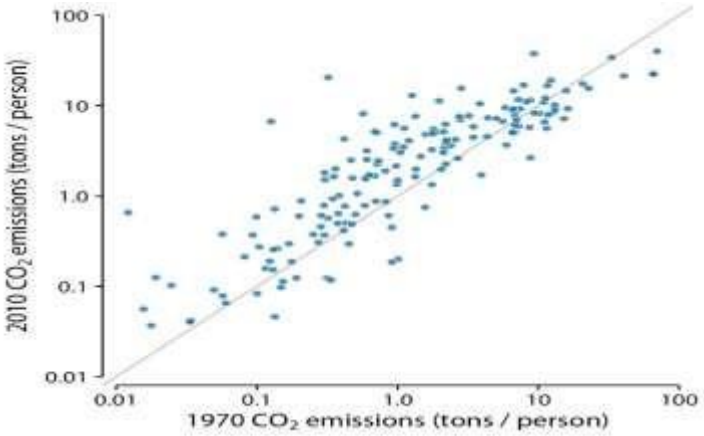


Figure 12.11: Carbon dioxide (CO_2) emissions per person in 1970 and 2010, for 166 countries. Each dot represents one country. The diagonal line represents identical CO_2 emissions in 1970 and 2010. The points are systematically shifted upwards relative to the diagonal line: In the majority of countries, emissions were higher in 2010 than in 1970. Data source: Carbon Dioxide Information Analysis Center

Scatter plots such as Figure 12.11 work well when we have a large number of data points and/or are interested in a systematic deviation of the entire data set from the null expectation. By contrast, if we have only a small number of observations and are primarily interested in the identity of each individual case, a *slopegraph* may be a better choice. In a slopegraph, we draw individual measurements as dots arranged into two columns and indicate pairings by connecting the paired dots with a line. The slope of each line highlights the magnitude and direction of change. Figure 12.12 uses this approach to show the ten countries with the largest difference in CO_2 emissions per person from 2000 to 2010.

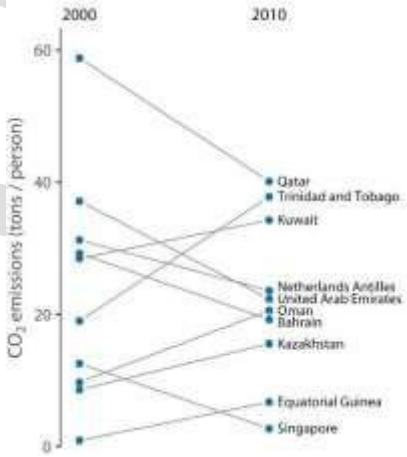


Figure 12.12: Carbon dioxide (CO_2) emissions per person in 2000 and 2010, for the ten countries with the largest difference between these two years. Data source: Carbon Dioxide Information Analysis Center

Slopegraphs have one important advantage over scatter plots: They can be used tocompare more than two measurements at a time. For example, we can modify

Figure 12.12 to show CO_2 emissions at three time points, here the years 2000, 2005,and 2010 (Figure 12.13). This choice highlights both countries with a large change inemissions over the entire decade as well as countries such as Qatar or Trinidad andTobago for which there is a large difference in the trend seen for the first five-year interval and the second one.

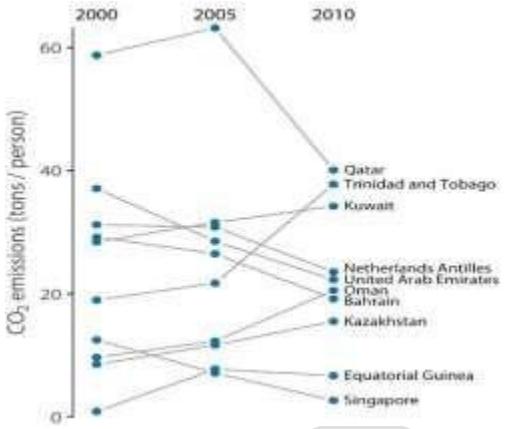


Figure 12.13: CO_2 emissions per person in 2000, 2005, and 2010, for the ten countries with the largest difference between the years 2000 and 2010. Data source: Carbon Dioxide Information Analysis Center **Visualizing time series**

Individual time series

As a first demonstration of a time series, we will consider the pattern of monthly preprint submissions in biology. Preprints are scientific articles that researchers postonline before formal peer review and publication in a scientific journal. The preprint server bioRxiv, which was founded in November 2013 specifically for researchers working in the biological sciences, has seen substantial growth in monthly submissions since. We can visualize this growth by making a form of scatter plot (Chapter 12) where we draw dots representing the number of submissions in each month (Figure 13.1).

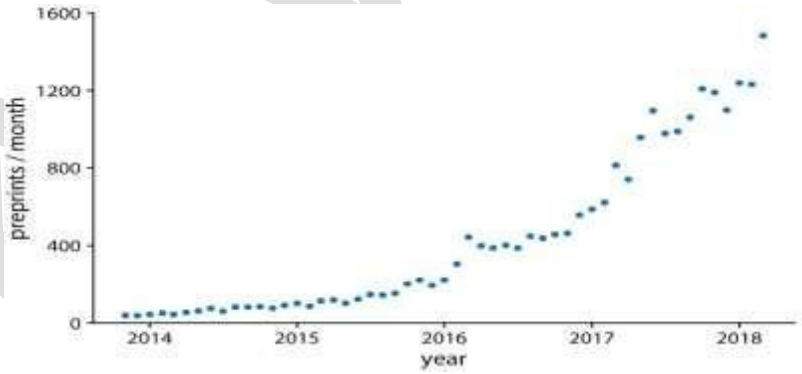


Figure 13.1: Monthly submissions to the preprint server bioRxiv, from its inception in November 2014 until April 2018. Each dot represents the number of submissions in one month. There has been a steady increase in submission volume throughout the entire 4.5-year period. Data source: Jordan Anaya, http://www.prepubmed.org/

There is an important difference however between Figure 13.1 and the scatter plots discussed in Chapter 12. In Figure 13.1, the dots are spaced evenly along the *x* axis, and there is a defined order among them. Each dot has exactly one left and one right neighbor (except the leftmost and rightmost points which have only one neighbor each). We can visually emphasize this order by connecting neighboring points with lines (Figure 13.2). Such a plot is called a *line graph*.

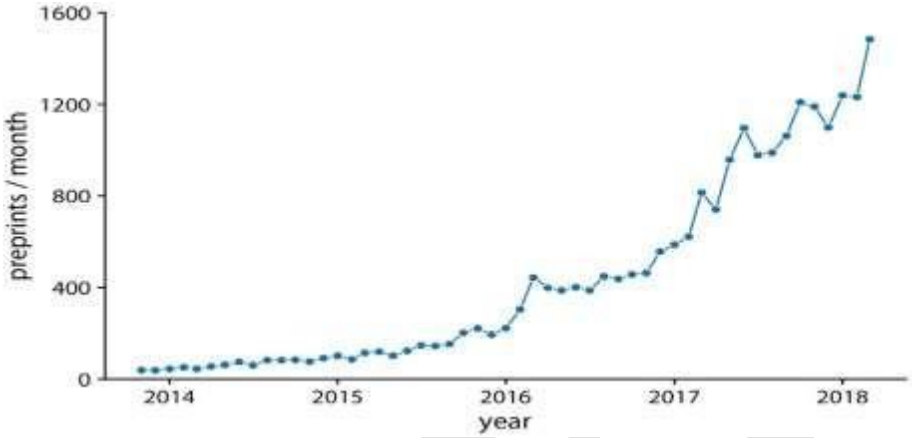


Figure 13.2: Monthly submissions to the preprint server bioRxiv, shown as dots connected by lines. The lines do not represent data but are only meant as a guide to the eye. By connecting the individual dots with lines, we emphasize that there is an order between the dots, each dot has exactly one neighbor that comes before and one that comes after. Data source: Jordan Anaya, http://www.prepubmed.org/

Some people object to drawing lines between points because the lines do not represent observed data. In particular, if there are only a few observations spaced far apart, had observations been made at intermediate times they would probably not have fallen exactly onto the lines shown. Thus, in a sense, the lines correspond to made-up data. Yet they may help with perception when the points are spaced far apart or are unevenly spaced. We can somewhat resolve this dilemma by pointing it out in the figure caption, for example by writing "lines are meant as a guide to the eye" (see caption of Figure 13.2).

Using lines to represent time series is generally accepted practice, however, and frequently the dots are omitted altogether (Figure 13.3). Without dots, the figure places more emphasis on the overall trend in the data and less on individual observations. A figure without dots is also visually less busy. In general, the denser the time series, the less important it is to show individual obserations with dots. For the preprint dataset shown here, I think omitting the dots is fine.

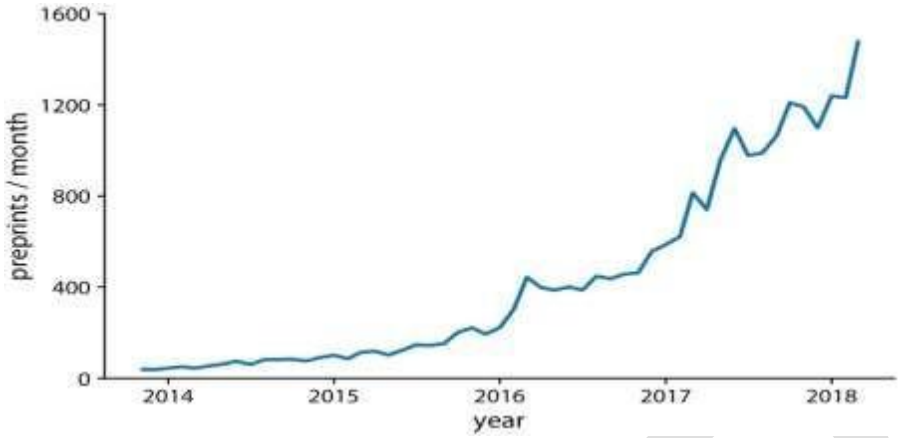


Figure 13.3: Monthly submissions to the preprint server bioRxiv, shown as a line graph without dots. Omitting the dots emphasizes the overall temporal trend while de-emphasizing individual observations at specific time points. It is particularly useful when the time points are spaced very densely. Data source: Jordan Anaya,http://www.prepubmed.org/

We can also fill the area under the curve with a solid color (Figure 13.4). This choice further emphasizes the overarching trend in the data, because it visually separates the area above the curve from the area below. However, this visualization is only valid if the *y* axis starts at zero, so that the height of the shaded area at each time point represents the data value at that time point.

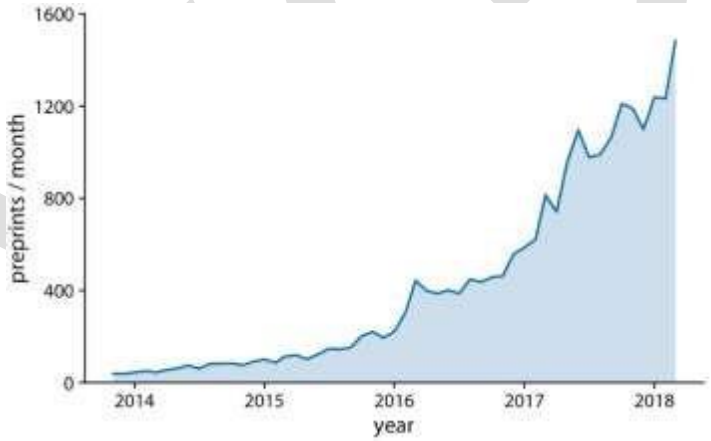


Figure 13.4: Monthly submissions to the preprint server bioRxiv, shown as a line graph with filled area underneath. By filling the area under the curve, we put even more emphasis on the overarching temporal trend than if we just draw a line (Figure 13.3). Data source: Jordan Anaya, http://www.prepubmed.org/

Visualizing trends

When making scatter plots (Chapter 12) or time series (Chapter 13), we are often more interested in the overarching trend of the data than in the specific detail of where each individual data point lies. By drawing the trend on top of or instead of theactual data points, usually in the form of a straight or curved line, we can create a

visualization that helps the reader immediately see key features of the data. There are two fundamental approaches to determining a trend: We can either smooth the data by some method, such as a moving average, or we can fit a curve with a defined functional form and then draw the fitted curve. Once we have identified a trend in a dataset, it may also be useful to look specifically at deviations from the trend or to separate the data into multiple components, including the underlying trend, any existing cyclical components, and episodic components or random noise.

Smoothing

Let us consider a time series of the Dow Jones Industrial Average (Dow Jones for short), a stock-market index representing the price of 30 large, publicly owned U.S.companies. Specifically, we will look at the year 2009, right after the 2008 crash (Figure 14.1). During the tail end of the crash, in the first three months of the year 2009, the market lost over 2400 points (~27%). Then it slowly recovered for the remainder of the year. How can we visualize these longer-term trends while de-emphasizing the less important short-term fluctuations?

Figure 14.1: Daily closing values of the Dow Jones Industrial Average for the year2009. Data source: Yahoo! Finance

In statistical terms, we are looking for a way to *smooth* the stock-market time series. The act of smoothing produces a function that captures key patterns in the data while removing irrelevant minor detail or noise. Financial analysts usually smooth stock-market data by calculating moving averages. To generate a moving average, we take a time window, say the first 20 days in the time series, calculate the average price over these 20 days, then move the time window by one day, so it now spans the 2nd to 21st day, calculate the average over these 20 days, move the time window again, and so on. The result is a new time series consisting of a sequence ofaveraged prices.

To plot this sequence of moving averages, we need to decide which specific time point to associate with the average for each time window. Financial analysts often plot each average at the end of its respective time window. This choice results in curves that lag the original data (Figure 14.2a), with more severe lags corresponding

to larger averaging time windows. Statisticians, on the other hand, plot the averageat the center of the time window, which results in a curve that overlays perfectly onthe original data (Figure 14.2b).

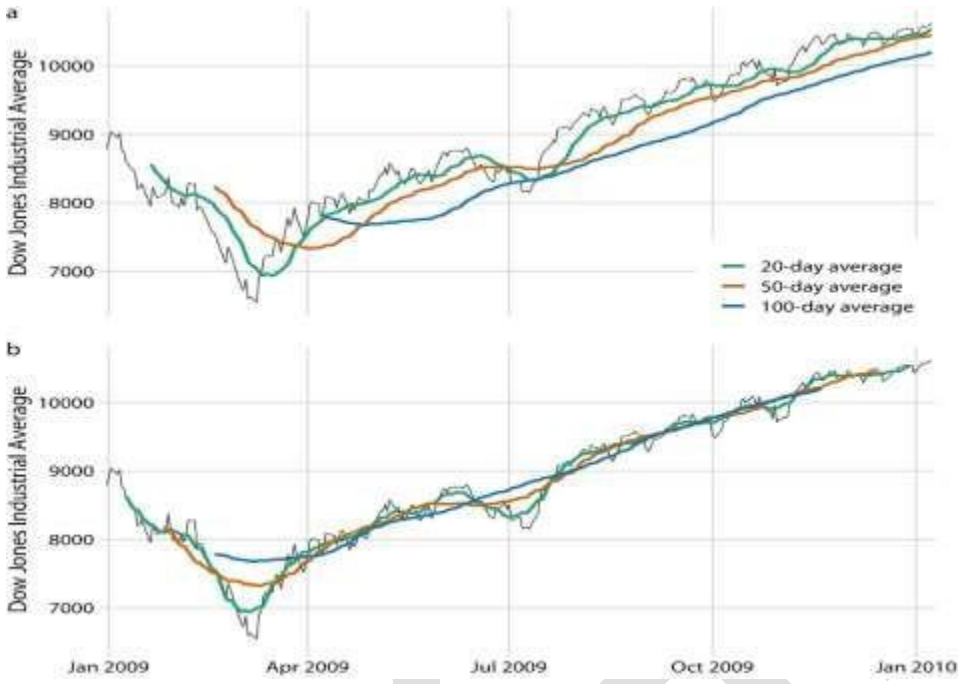


Figure 14.2: Daily closing values of the Dow Jones Industrial Average for the year 2009, shown together with their 20-day, 50-day, and 100-day moving averages. (a) The moving averages are plotted at the end of the moving time windows. (b) The moving averages are plotted in the center of the moving time windows. Data source:Yahoo! Finance

Regardless of whether we plot the smoothed time series with or without lag, we can see that the length of the time window over which we average sets the scale of the fluctuations that remain visible in the smoothed curve. The 20-day moving average only removes small, short-term spikes but otherwise follows the daily data closely. The 100-day moving average, on the other hand, removes even fairly substantial drops or spikes that play out over a time span of multiple weeks. For example, the massive drop to below 7000 points in the first quarter of 2009 is not visible in the 100-day moving average, which replaces it with a gentle curve that doesn't dip much below 8000 points (Figure 14.2). Similarly, the drop around July 2009 is completely invisible in the 100-day moving average.

The moving average is the most simplistic approach to smoothing, and it has some obvious limitations. First, it results in a smoothed curve that is shorter than the original curve (Figure 14.2). Parts are missing at either the beginning or the end or both. And the more the time series is smoothed (i.e., the larger the averaging window), the shorter the smoothed curve. Second, even with a large averaging window, a moving average is not necessarily that smooth. It may exhibit small bumps and wiggles even though larger-scale smoothing has been achieved (Figure 14.2). These wiggles are caused by individual data points that enter or exit the averaging window. Since all data points in the window are weighted equally,

individual data points at the window boundaries can have visible impact on theaverage.

Statisticians have developed numerous approaches to smoothing that alleviate the downsides of moving averages. These approaches are much more complex and computationally costly, but they are readily available in modern statistical computing environments. One widely used method is LOESS (locally estimated scatterplot smoothing, W. S. Cleveland (1979)), which fits low-degree polynomials to subsets ofthe data. Importantly, the points in the center of each subset are weighted more heavily than points at the boundaries, and this weighting scheme yields a much smoother result than we get from a weighted average (Figure 14.3). The LOESS curve shown here looks similar to the 100-day average, but this similarity should notbe overinterpreted. The smoothness of a LOESS curve can be tuned by adjusting a parameter, and different parameter choices would have produced LOESS curves looking more like the 20-day or 50-day average.

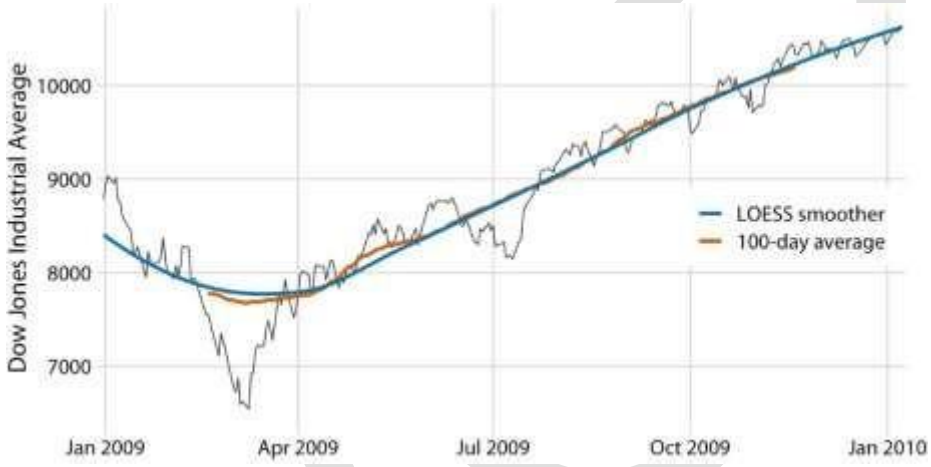


Figure 14.3: Comparison of LOESS fit to 100-day moving average for the Dow Jonesdata of Figure 14.2. The overall trend shown by the LOESS smooth is nearly identical to the 100-day moving average, but the LOESS curve is much smoother and it extends to the entire range of the data. Data source: Yahoo! Finance

Importantly, LOESS is not limited to time series. It can be applied to arbitrary scatter plots, as is apparent from its name, *locally estimated scatterplot smoothing*. For example, we can use LOESS to look for trends in the relationship between a car's fuel-tank capacity and its price (Figure 14.4). The LOESS line shows that tank capacity grows approximately linearly with price for cheap cars (below $20,000) but it levels off for more expensive cars. Above a price of approximately $20,000, buyinga more expensive car will not get you one with a larger fuel tank.

Figure 14.4: Fuel-tank capacity versus price of 93 cars released for the 1993 model year. Each dot corresponds to one car. The solid line represents a LOESS smooth ofthe data. We see that fuel-tank capacity increases approximately linearly with price, up to a price of approximately $20,000, and then it levels off. Data source: Robin H. Lock, St. Lawrence University

LOESS is a very popular smoothing approach because it tends to produce resultsthat look right to the human eye. However, it requires the fitting of many separateregression models. This makes it slow for large datasets, even on modern computing equipment.

As a faster alternative to LOESS, we can use spline models. A spline is a piecewise polynomial function that is highly flexible yet always looks smooth. When working with splines, we will encounter the term *knot*. The knots in a spline are the endpointsof the individual spline segments. If we fit a spline with k segments, we need to specify $k + 1$ knots. While spline fitting is computationally efficient, in particular if the number of knots is not too large, splines have their own downsides. Most importantly, there is a bewildering array of different types of splines, including cubic splines, B-splines, thin-plate splines, Gaussian process splines, and many others, and which one to pick may not be obvious. The specific choice of the type of spline and number of knots used can result in widely different smoothing functions for the same data (Figure 14.5).

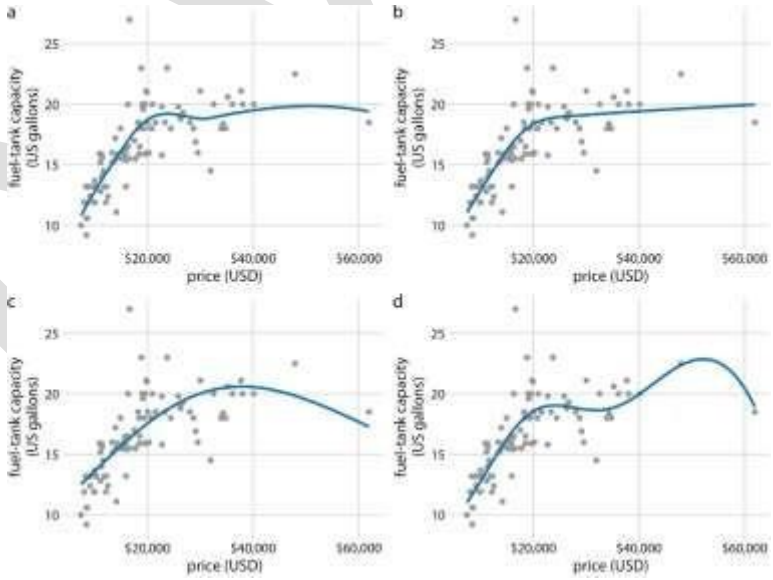


Figure 14.5: Different smoothing models display widely different behaviors, in particular near the boundaries of the data. (a) LOESS smoother, as in Figure 14.4.
(b) Cubic regression splines with 5 knots. (c) Thin-plate regression spline with 3 knots. (d) Gaussian process spline with 6 knots. Data source: Robin H. Lock, St.Lawrence University

Most data visualization software will provide smoothing features, likely implemented as either a type of local regression (such as LOESS) or a type of spline. The smoothing method may be referred to as a GAM, a generalized additive model, which is a superset of all these types of smoothers. It is important to be aware that the output of the smoothing feature is highly dependent on the specific GAM model that is fit. Unless you try out a number of different choices you may never realize to what extent the results you see depend on the specific default choices made by your statistical software.

Be careful when interpreting the results from a smoothing function. The same dataset can be smoothed in many different ways.

Showing trends with a defined functionalform

As we can see in Figure 14.5, the behavior of general-purpose smoothers can be somewhat unpredictable for any given dataset. These smoothers also do not provide parameter estimates that have a meaningful interpretation. Therefore, whenever possible, it is preferable to fit a curve with a specific functional form that is appropriate for the data and that uses parameters with clear meaning.

For the fuel-tank data, we need a curve that initially rises linearly but then levels off at a constant value. The function $y=A-B\exp(-mx)$ may fit that bill. Here, A, B, and m are the constants we adjust to fit the curve to the data. The function is approximately linear for small x, with $y\approx A-B+Bmx$, it approaches a constant value for large x, $y\approx A$, and it is strictly increasing for all values of x.
Figure 14.6 shows that this equation fits the data at least as well as any of the smoothers we considered previously (Figure 14.5).

Figure 14.6: Fuel-tank data represented with an explicit analytical model. The solid line corresponds to a least-squares fit of the formula $y=A-B\exp(-mx)$ to the data. Fitted parameters are $A=19.6$, $B=29.2$, $m=0.00015$. Data source: Robin H. Lock, St. Lawrence University

A functional form that is applicable in many different contexts is the simple straight line, $y=A+mx$. Approximately linear relationships between two variables are surprisingly common in real-world datasets. For example, in Chapter 12, I discussed the relationship between head length and body mass in blue jays. This relationship is approximately linear, for both female and male birds, and drawing linear trend lines on top of the points in a scatter plot helps the reader perceive the trends (Figure 14.7).

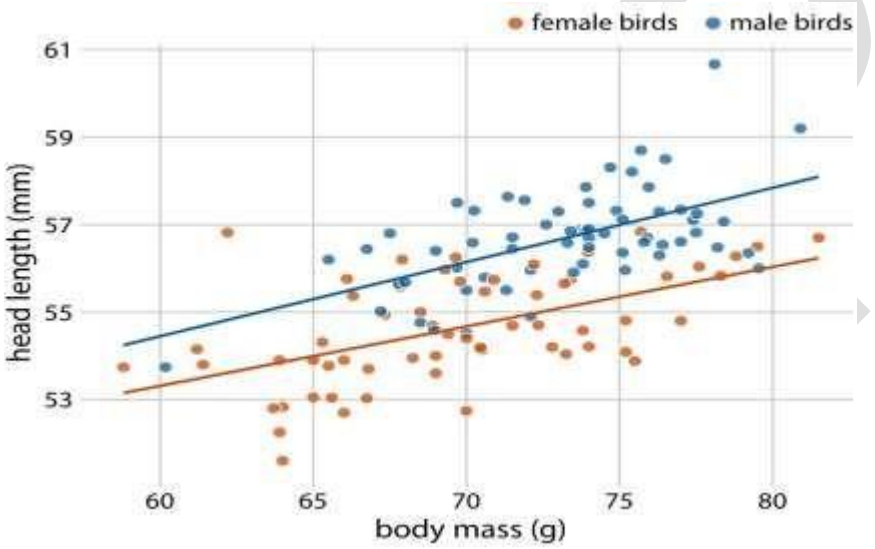


Figure 14.7: Head length versus body mass for 123 blue jays. The birds' sex is indicated by color. This figure is equivalent to Figure 12.2, except that now we have drawn linear trend lines on top of the individual data points. Data source: Keith Tarvin, Oberlin College

When the data display a non-linear relationship, we need to guess what an appropriate functional form might be. In this case, we can assess the accuracy of our guess by transforming the axes in such a way that a linear relationship emerges. To demonstrate this principle, let's return to the monthly submissions to the preprint server bioRxiv, discussed in Chapter 12. If the increase in submissions in each month is proportional to the number of submissions in the previous month, i.e., if submissions grow by a fixed percentage each month, then the resulting curve is exponential. This assumption seems to be met for the bioRxiv data, because a curve with exponential form, $y=A\exp(mx)$, fits the bioRxiv submission data well (Figure 14.8).

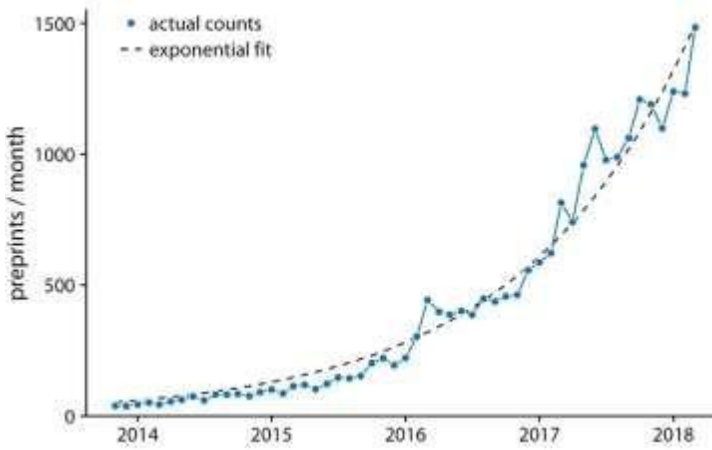


Figure 14.8: Monthly submissions to the preprint server bioRxiv. The solid blue line represents the actual monthly preprint counts and the dashed black line represents an exponential fit to the data, y=60exp[0.77(x−2014)]y=60exp [0.77(x−2014)]. Data source: Jordan Anaya, http://www.prepubmed.org/

If the original curve is exponential, y=Aexp(mx)y=Aexp (mx), then a logtransformation of the *y* values will turn it into a linear relationship, log(y)=log(A)+mxlog (y)=log(A)+mx. Therefore, plotting the data with log-transformed *y* values (or equivalently, with a logarithmic *y* axis) and looking for a linear relationship is a good way of determining whether a dataset exhibits exponential growth. For the bioRxiv submission numbers, we indeed obtaina linear relationship when using a logarithmic *y* axis (Figure 14.9).

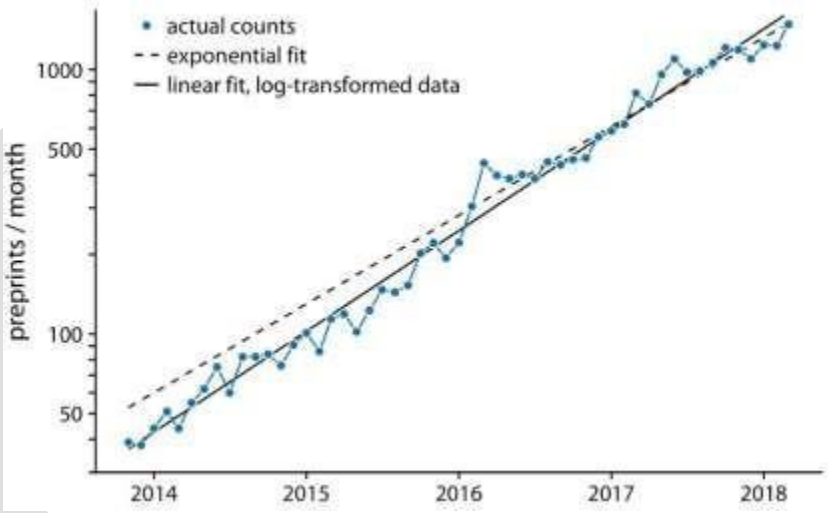


Figure 14.9: Monthly submissions to the preprint server bioRxiv, shown on a logscale. The solid blue line represents the actual monthly preprint counts, the dashed black line represents the exponential fit from Figure 14.8, and the solid black line represents a linear fit to log-transformed data, corresponding to y=43exp[0.88(x−2014)]y=43exp [0.88(x−2014)]. Data source: Jordan Anaya, http://www.prepubmed.org/

In Figure 14.9, in addition to the actual submission counts, I am also showing the exponential fit from Figure 14.8 and a linear fit to the log-transformed data. These two fits are similar but not identical. In particular, the slope of the dashed line seems

somewhat off. The line systematically falls above the individual data points for half the time series. This is a common problem with exponential fits: The square deviations from the data points to the fitted curve are so much larger for the largest data values than for the smallest data values that the deviations of the smallest data values contribute little to the overall sum squares that the fit minimizes. As a result, the fitted line systematically overshoots or undershoots the smallest data values. Forthis reason, I generally advise to avoid exponential fits and instead use linear fits on log-transformed data.

It is usually better to fit a straight line to transformed data than to fit a nonlinear curve to untransformed data.

A plot such as Figure 14.9 is commonly referred to as *log–linear*, since the *y* axis is logarithmic and the *x* axis is linear. Other plots we may encounter include *log– log*, where both the *y* and the *x* axis are logarithmic, or *linear–log*, where *y* is linear and *x*is logarithmic. In a log–log plot, power laws of the form $y \sim x_\alpha y \sim x\alpha$ appear as straight lines (see Figure 8.7 for an example), and in a linear–log plot, logarithmic relationships of the form $y \sim \log(x) y \sim \log(x)$ appear as a straight lines. Other functional forms can be turned into linear relationships with more specialized coordinate transformations, but these three (log–linear, log–log, linear–log) cover a wide range of real-world applications.

Detrending and time-series decomposition

For any time series with a prominent long-term trend, it may be useful to remove this trend to specifically highlight any notable deviations. This technique is called *detrending*, and I will demonstrate it here with house prices. In the U.S., the mortgage lender Freddie Mac publishes a monthly index, called the *Freddie Mac House Price Index*, that tracks the change in house prices over time. The index attempts to capture the state of the entire house market in a given region, such that an increase in the index by, for example, 10% can be interpreted as an average house price increase of 10% in the respective market. The index is arbitrarily set to a value of 100 in December 2000.

Over long periods of time, house prices tend to display consistent annual growth, approximately in line with inflation. However, overlaid on top of this trend are housing bubbles that lead to severe boom and bust cycles. Figure 14.10 shows the actual house price index and its long-term trend for four select U.S. states. We see that between 1980 and 2017, California underwent two bubbles, one in 1990 and one in the mid-2000s. During the same period, Nevada experienced only one bubble, in themid-2000s, and house prices in Texas and West Virginia closely followed their long- term trends the entire time. Because house prices tend to grow in percent increments, i.e., exponentially, I have chosen a logarithmic *y* axis in Figure 14.10.
The straight lines correspond to a 4.7% annual price increase in California and a2.8% annual price increase each in Nevada, Texas, and West Virginia.

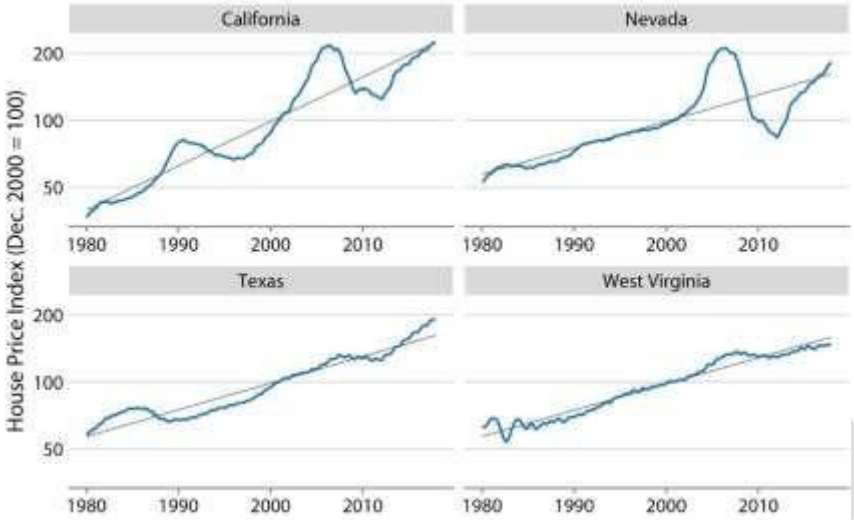


Figure 14.10: Freddie Mac House Price Index from 1980 through 2017, for four selected states (California, Nevada, Texas, and West Virginia). The House Price Index is a unitless number that tracks relative house prices in the chosen geographicregion over time. The index is scaled arbitrarily such that it equals 100 in December of the year 2000. The blue lines show the monthly fluctuations in the index and the straight gray lines show the long-term price trends in the respective states. Note thatthe *y* axes are logarithmic, so that the straight gray lines represent consistent exponential growth. Data source: Freddie Mac House Prices Index

We detrend housing prices by dividing the actual price index at each time point by the respective value in the long-term trend. Visually, this division will look like we are subtracting the gray lines from the blue lines in Figure 14.10, because a division of the untransformed values is equivalent to a subtraction of the logtransformed values. The resulting detrended house prices show the housing bubbles more clearly (Figure 14.11), as the detrending emphasizes the unexpected movements in a time series.

For example, in the original time series, the decline in home prices in California from1990 to about 1998 looks modest (Figure 14.10). However, during that same time period, on the basis of the longterm trend we would have expected prices to rise.

Relative to the expected rise the drop in prices was substantial, amounting to 25% atthe lowest point (Figure 14.11).

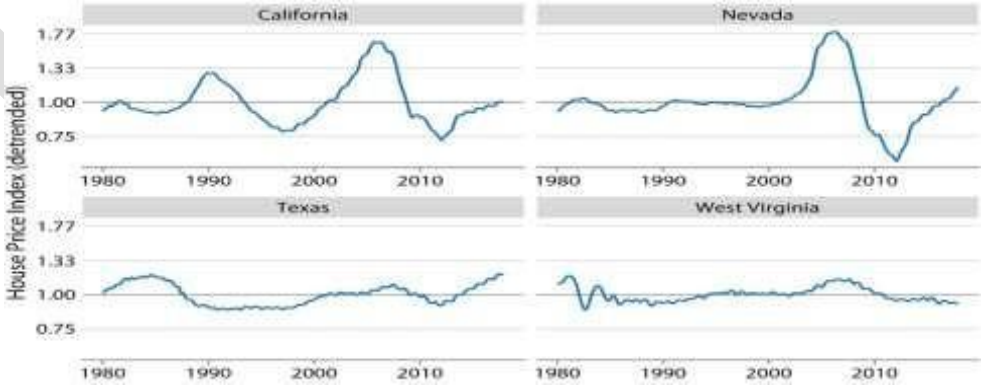


Figure 14.11: Detrended version of the Freddie Mac House Price Index shown in Figure 14.10. The detrended index was calculated by dividing the actual index (blue lines in Figure 14.10) by the expected value based on the long-term trend (straight

gray lines in Figure 14.10). This visualization shows that California experienced two housing bubbles, around 1990 and in the mid-2000s, identifiable from a rapid rise and subsequent decline in the actual housing prices relative to what would have been expected from the long-term trend. Similarly, Nevada experienced one housingbubble, in the mid-2000s, and neither Texas nor West Virginia experienced much of a bubble at all. Data source: Freddie Mac HousePrices Index

Beyond simple detrending, we can also separate a time series into multiple distinct components, such that their sum recovers the original time series. In general, in addition to a long-term trend, there are three distinct components that may shape a time series. First, there is random noise, which causes small, erratic movements up and down. This noise is visible in all the time series shown in this chapter, but maybethe most clearly in Figure 14.9. Second, there can be unique external events that leave their mark in the time series, such as the distinct housing bubbles seen in Figure 14.10. Third, there can be cyclical variations. For example, outside temperatures show daily cyclical variations. The highest temperatures are reached inthe early afternoon and the lowest temperatures in the early morning. Outside temperatures also show yearly cyclical variations. They tend to rise in the spring, reach their maximum in the summer, and then decline in fall and reach their minimum in the winter (Figure 3.2).

To demonstrate the concept of distinct time-series components, I will here decompose the Keeling curve, which shows changes in CO_2 abundance over time (Figure 14.12). CO_2 is measured in parts per million (ppm). We see a long-term increase in CO_2 abundance that is slightly faster than linear, from below 325 ppm inthe 1960s to above 400 in the second decade of the 21st century (Figure 14.12). CO_2 abundance also fluctuates anually, following a consistent upand-down pattern overlaid on top of the overall increase. The annual fluctuation are driven by plant growth in the northern hemisphere. Plants consume CO_2 during photosynthesis. Because most of the globe's land masses are located in the northern hemisphere, and plant growth is most active in the spring and summer, we see an annual global decline in atmospheric CO_2 that coincides with the summer months in the northern hemisphere.

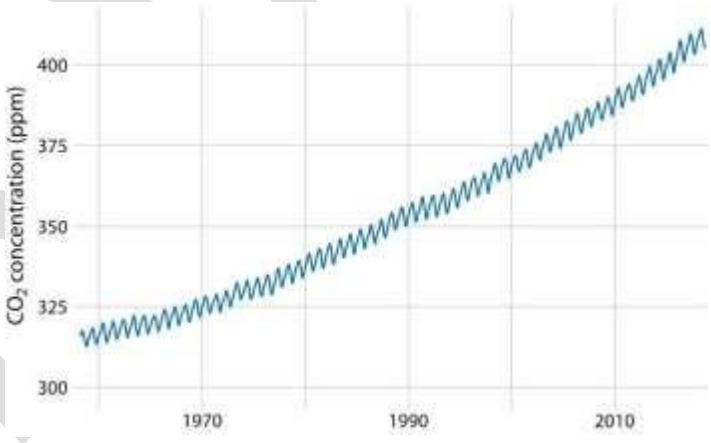


Figure 14.12: The Keeling curve. The Keeling curve shows the change of

CO_2 abundance in the atmosphere over time. Since 1958, CO_2 abundance has been continuously monitored at the Mauna Loa Observatory in Hawaii, initially under the direction of Charles Keeling. Shown here are monthly average CO_2 readings, expressed in parts per million (ppm). The CO_2 readings fluctuate anually with the seasons but show a consistent long-term trend of increase. Data source: Dr. Pieter Tans, NOAA/ESRL, and Dr. Ralph Keeling, Scripps Institution of Oceanography

We can decompose the Keeling curve into its long-term trend, seasonal fluctuations, and remainder (Figure 14.13). The specific method I am using here is called STL (Seasonal decomposition of Time series by LOESS, R. B. Cleveland et al. (1990)), but there are many other methods that achieve similar goals. The decomposition shows that over the last three decades, CO_2 abundance has increased by over 50 ppm. By comparison, seasonal fluctuations amount to less than 8 ppm (they never cause an increase or a decrease in more than 4 ppm relative to the long-term trend), and the remainder amounts to less than 1.6 ppm (Figure 14.13). The remainder is the difference between the actual readings and the sum of the long-term trend and the seasonal fluctuations, and here it corresponds to random noise in the monthly CO_2 readings. More generally, however, the remainder could also capture unique external events. For example, if a massive volcano erruption released substantial amounts of CO_2, such an event might be visible as a sudden spike in the remainder. Figure 14.13 shows that no such unique external events have had a major effect on the Keeling curve in recent decades.

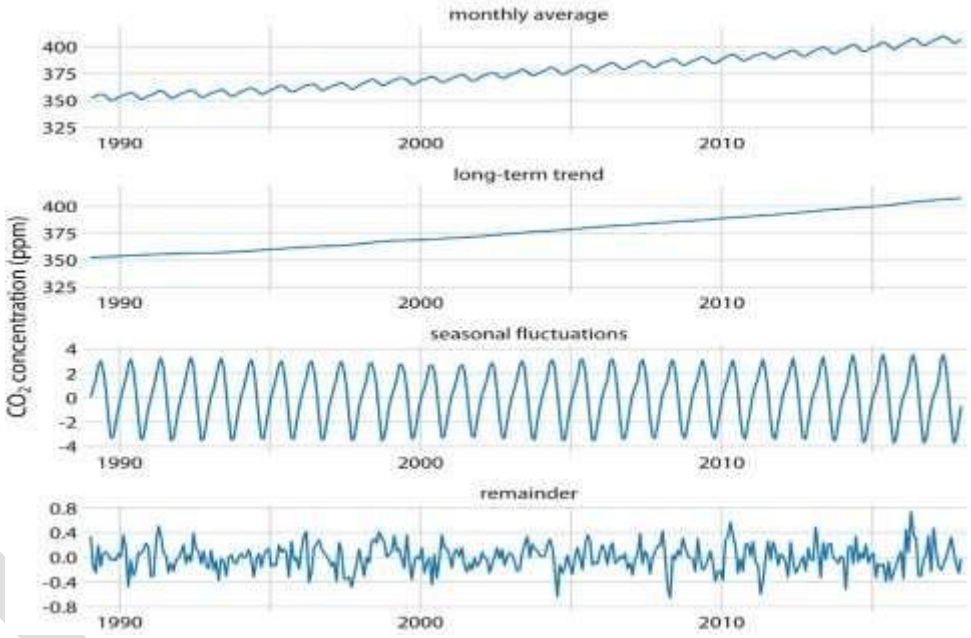


Figure 14.13: Time-series decomposition of the Keeling curve, showing the monthly average (as in Figure 14.12), the long-term trend, seasonal fluctuations, and the remainder. The remainder is the difference between the actual readings and the sum of the long-term trend and the seasonal fluctuations, and it represents random noise. I have zoomed into the most recent 30 years of data to more clearly show the shape of the annual fluctuations. Data source: Dr. Pieter Tans, NOAA/ESRL, and Dr. RalphKeeling, Scripps Institution of Oceanography