Chapter 2

Statistics

2.1 Quantities and Measures for Random Data

The most basic way to characterize a numerical data set is through one number- the mean (or median or mode).

- The **sample mean** for a discrete set of m numbers, $x_1, \ldots, x_m$ is given by:

  $$\bar{x} = \frac{1}{m} \sum_{k=1}^{m} x_k$$

- The **mean of a set of m vectors in $\mathbb{R}^n$**:

  Suppose we have m vectors in $\mathbb{R}^n$. We can similarly define the (sample) mean by replacing the scalar x_k with the k^{th} vector:

  $$\bar{\boldsymbol{x}} = \frac{1}{m} \sum_{k=1}^{m} \boldsymbol{x}^{(k)}$$

  The j^{th} element of the sample mean vector is just the sample mean of the (scalar) data in the j^{th} dimension of your collection of vectors.

- Note that we can also define the mean for a collection of $m \times n$ matrices, as well. For example, if I have a collection of k photos that are each $m \times n$ pixels, then I can compute the mean photo by summing the matrices together, then dividing by k.

- In fact, matrices have different properties that we can consider- A matrix can be thought of as a collection of column vectors, or as a collection of row vectors, or simply a collection of numbers.

  Similarly, we can compute a mean of over the columns (and getting a column), or the mean over all rows (and get a row), or we can compute the mean over all the numerical values of the matrix, which is called the **grand mean**.

- Computing the mean in Matlab, Python and R:

  - Matlab:

    * If $\boldsymbol{x}$ is a row or column vector (example):

      ```
      x=[1,2,3,4,5];
      mean(x)
      ```

    * If X is an $m \times n$ matrix (example):

```
    X=[1,2,3; 4,5,6];
    mean(X)       %Returns a row vector as default
    m=mean(X,1);  %Returns a row vector 1 x n
    m=mean(X,2);  %Returns a column vector m x 1
    m=mean(mean(X)); % Returns the grand mean (a scalar)
    ```

  - Python: First, import *numpy*: `import numpy as np`
    - If $\boldsymbol{x}$ is a row or column vector- two methods are shown below:

      ```
      x=np.array([1,2,3,4,5])  #Example vector
      np.mean(x)  #Returns a scalar
      x.mean(0)   #Returns an array- a scalar or a vector
      ```

    - If X is an $m \times n$ array

      ```
      X=np.array([[1,2,3],[4,5,6]])
      X.mean(0)  # Output: array([2.5, 3.5, 4.5])
      X.mean(1)  # Output: array([2., 5.])
      X.mean()   # Output the grand mean: 3.5
      ```

      Alternatively, for the row, column means respectively:

      ```
      Xr=np.mean(X,axis=0)
      Xm=np.mean(X,axis=1)
      ```

  - R
    - If $\boldsymbol{x}$ is a row or column vector: then

      ```
      x<-c(1,2,3,4,5)
      np.mean(x)  #Returns a scalar
      x.mean(0)   #Returns an array- a scalar or a vector
      ```

    - If X is an $m \times n$ array

      ```
      x<-array(1:6,c(2,3))  #x is a 2 x 3 matrix
      colMeans(x)       #Returns: 1.5 3.5 5.5
      rowMeans(x)       #Returns: 3 4
      ```

      Alternatively,

      ```
      apply(x,1,mean)   #Returns: 3 4
      apply(x,2,mean)   #Returns: 1.5 3.5 5.5
      ```

  A note about language: Should "row mean" be the mean found by summing the rows together, and producing a row, or should "row mean" be the sum through the rows, producing a column? I will typically mean the former, but I see that R uses the latter (the command `colMeans` produces the mean down the columns and returns a row, for example).

  Just be sure you're consistent with whichever method you want to define.

- The *Median* is a number so that exactly half the data is above that number, and half the data is below that number. Although the median does not have to be unique, we follow the definitions below if we are given a finite sample:

  If there are an odd number of data points, the median is the middle point. If there is an even number of data points, then there are two numbers in the middle- the median is the average of these.

  The syntax for the median works in very much the same way as the mean.

- The *Mode* is the value taken the most number of times. In the case of ties, the data is multi-modal.

  We typically would not use the mode unless there is a special reason to do so.

2.1.1 Matlab note, Linear Algebra

We'll be subtracting a row vector from each row of a matrix, and similarly, we'll subtract a column vector from each column of a matrix. You'll note that, if A is a matrix, $\boldsymbol{r}$ is a row, and $\boldsymbol{c}$ is a column, then writing something like:

$$A - \boldsymbol{r} \qquad A - \boldsymbol{c}$$

would not be defined in linear algebra, and for older versions of Matlab, this was the case as well. This changed several years ago, so that "Matrix - Row" is assumed to be row subtraction for each row of the matrix, and "Matrix - Column" is assumed to be carried out column-wise on the matrix. We'll see this below.

2.1.2 Centering and Double Centering Data

Let matrix A be $m \times n$, which may be considered n points in $\mathbb{R}^m$ (this looks at the data column-wise) or m points in $\mathbb{R}^n$ (looking at the data row-wise). If we wish to look at A both ways, a double-centering may be appropriate.

The result of the double-centering will be that (in Matlab), we determine $\hat{A}$ so that

$$\texttt{mean}(\hat{A}, 1) = 0, \qquad \texttt{mean}(\hat{A}, 2) = 0$$

There are a couple of ways to do this. Here is onw algorithm, where the means are computed first.

Algorithm for Double Centering

- Given matrix A:
  - Compute the mean of the rows. Call this row $\boldsymbol{r}$.
  - Compute the mean of the columns. Call this column $\boldsymbol{c}$.
  - Compute the grand mean. Call this scalar g.
- Output the matrix: $A - \boldsymbol{r} - \boldsymbol{c} + g$.

Here is the implementation in Matlab, Python and R:

Matlab	Python	R
`A=[1,2,3;4,5,6];`	`A=np.array([[1,2,3],[4,5,6]])`	`A<-array(1:6,c(2,3))`
`r=mean(A,1);`	`r=A.mean(A,0)`	`r=apply(A,2,mean)`
`c=mean(A,2);`	`c=A.mean(A,1)`	`c=apply(A,1,mean)`
	`c=c[:,np.newaxis]`	
`g=mean(mean(A));`	`g=A.mean()`	`g=mean(apply(A,1,mean))`
`B=A-r-c+g`	`B=A-r-c+g`	`B1=sweep(A,2,r)`
		`B2=sweep(B1,1,c)`
		`B=B2+g`

As a final note, this technique is only suitable if it is reasonable that the $m \times n$ matrix may be data in either $\mathbb{R}^n$ or $\mathbb{R}^m$. For example, you probably would not double center a matrix that is 5000×2- Treat this as 5000 points in $\mathbb{R}^2$, so that the mean is in $\mathbb{R}^2$.

2.2 Variance and Standard Deviation

The number that is used to describe the spread of the data about its mean is the *variance*. As with the mean, we rarely know the underlying distribution, so again we'll focus on the sample variance.

Let $\{x_1, \ldots, x_m\}$ be m real numbers, and $\bar{x}$ its sample mean. Then the **sample variance** is:

$$s^2 = \frac{1}{m-1} \sum_{k=1}^{m} (x_k - \bar{x})^2$$

If we think of the data as a vector of length m, then this formula becomes:

$$s^2 = \frac{1}{m-1}\|\boldsymbol{x} - \bar{x}\|^2$$

The **standard deviation** is the square root of the variance, so the standard deviation is s.

Quick Example

Let's take some template data to look at what the variance (and standard deviation) measure: Consider the data:

$$-\frac{2}{n}, -\frac{1}{n}, 0, \frac{1}{n}, \frac{2}{n}$$

If n is large, our data is tightly packed together about the mean, 0. If n is small, the data are spread out. The variance and standard deviation of this sample is:

$$s^2 = \frac{1}{4}\left(\frac{4+1+0+1+4}{n^2}\right) = \frac{5}{2}\frac{1}{n^2}, \qquad s = \sqrt{\frac{5}{2}}\frac{1}{n}$$

and this is in agreement with our heuristic: If n is large, our data is tightly packed about the mean, and the standard deviation is small. If n is small, our data is loosely distributed about the mean, and the standard deviation is large. Another way to look at the standard deviation is in linear algebra terms: If the data is put into a vector of length m (call it $\boldsymbol{x}$), then the (sample) standard deviation can be computed as:

$$s = \frac{\|\boldsymbol{x} - \bar{x}\|}{\sqrt{m-1}}$$

2.2.1 Covariance and Correlation Coefficients

If we have two data sets, sometimes we would like to compare them to see how they relate to each other. In this case, it is important that the two data sets be ordered so that x_1 is being compared to y_1, then x_2 is compared to y_2, and so on.

Definition: Let $X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_n\}$ be two ordered data sets with means m_x, m_y respectively. Then the *sample covariance* of the data sets is given by:

$$\mathrm{Cov}(X,Y) = s^2_{xy} = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - m_x)(y_k - m_y)$$

There are exercises at the end of the chapter that will reinforce the notation and give you some methods for manipulating the covariance. In the meantime, it is easy to remember this formula if you think of the following:

If X and Y have mean zero, and we think of X and Y as vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, then the covariance is just the dot product between the vectors, divided by $n-1$:

$$\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{n-1}\boldsymbol{x}^T\boldsymbol{y}$$

We can then interpret what it means for X, Y to have a covariance of zero: $\boldsymbol{x}$ is "orthogonal" to $\boldsymbol{y}$. Continuing with this analogy, if we normalized by the size of $\boldsymbol{x}$ and the size of $\boldsymbol{y}$, we'd get the cosine of the angle between them. This is the definition of the correlation coefficient, and gives the relationship between the covariance and correlation coefficient:

Definition: The **correlation coefficient** between the data ordered in vector $\boldsymbol{x}$ and data in $\boldsymbol{y}$ is given by:

$$r_{xy} = \frac{s^2_{xy}}{s_x s_y} = \frac{\sum_{k=1}^{n}(x_k - m_x)(y_k - m_y)}{\sqrt{\sum_{k=1}^{n}(x_k - m_x)^2 \cdot \sum_{k=1}^{n}(y_k - m_y)^2}}$$

If the data in $\boldsymbol{x}$ and $\boldsymbol{y}$ have been mean subtracted, then the formula is reminiscent of something from linear algebra:

$$r_{xy} = \frac{\boldsymbol{x}^T\boldsymbol{y}}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|} = \cos(\theta)$$

This works out so nicely because we have a $\frac{1}{n-1}$ in both the numerator and denominator, so they cancel each other out.

We also see immediately that r_{xy} can only take on the real numbers between -1 and 1. Some interesting values of r_{xy}:

If r_{xy} is:	Then the data is:
1	Perfectly correlated ($\theta = 0$)
0	Uncorrelated ($\theta = \frac{\pi}{2}$)
-1	Perfectly (negatively) correlated ($\theta = \pi$)

One last comment before we leave this section: The covariance s_{xy}^2 and correlation coefficient r_{xy} only look for *linear* relationships between data sets!

For example, we know that $\sin(x)$ and $\cos(x)$, either as functions, or as data points sampled at equally spaced intervals, will be uncorrelated, but, because $\sin^2(x) + \cos^2(x) = 1$, we see that $\sin^2(x)$ and $\cos^2(x)$ are perfectly correlated.

This difference is the difference between the words “correlated” and “statistically independent”. Statistical independence (not defined here) and correlations are not the same thing. We will look at this difference closely in a later section.

2.3 The Covariance Matrix

Suppose we have a collection of n columns, where each column represents data in one dimension. And suppose each column has p elements. The $p \times n$ matrix X can be thought of as either p points in $\mathbb{R}^n$ or n points in $\mathbb{R}^p$. Thinking of have p points in dimension i and p points in dimension j, we can compute the variance between those dimensions.

Continuing, we can compute the covariance between all pairings of the n dimensions resulting in an $n \times n$ matrix (note that the diagonal entries would be the covariance of a set of data with itself- which is the regular variance). Such a matrix is known as the covariance matrix.

In the formulas below, we'll assume that X has beende*mean-subtracted* (subtract the row representing the mean from all rows of X). The $(i,j)^{\text{th}}$ entry in the covariance matrix is then defined as the covariance between the i^{th} and j^{th} dimensions:

$$s_{ij}^2 = \frac{1}{p-1}\sum_{k=1}^{p} X(k,i) \cdot X(k,j)$$

Computing this for all i, j will result in an $n \times n$ symmetric matrix, C, for which $C_{ij} = s_{ij}^2$.

In the exercises, you'll show that an alternative way of computing the covariance matrix is by using what we'll refer to as its definition below.

Definition: Let X denote a matrix of data, so that, if X is $p \times n$, then we have p data points in $\mathbb{R}^n$. Furthermore, we assume that the data in X has been mean subtracted (so the mean in $\mathbb{R}^n$ is the zero vector). Then the $n \times n$ *covariance matrix* associated with X is given by:

$$C = \frac{1}{p-1}X^TX$$

In the language of your choice, it is straightforward to compute the covariance matrix- but be sure to keep in mind the dimensions.

Matlab	Python	R
`X=rand(10,3);`	`X=np.random.rand(10,3)`	`X<-matrix(runif(30),nrow=10)`
`C=cov(X);`	`C=np.cov(X.T,bias=False)`	`C<-cov(X)`

You might note that in Python, the default matrix arrangement is reversed, and the default number to divide by is n rather than $n-1$, unless you include the "bias" option.

2.4 Exercises

1. Compute the covariance between the following data sets:

x	-1.0	-0.7	-0.4	-0.1	0.2	0.5	0.8
y	-1.3	-0.7	-0.1	0.5	1.1	1.7	2.3

(2.1)

2. Let's explore some of the things mentioned in the text. Use a computer program to verify the following:
   (a) "If $\boldsymbol{t}$ is a vector of equally spaced points, the $\sin(t)$ and $\cos(t)$ (computer notation) will be uncorrelated". Show this by example using Matlab, Python or R.
   (b) Continuing, show that $\sin^2(t)$ and $\cos^2(t)$ are perfectly correlated (again, using Matlab, Python or R).
   (c) Take the vector $\boldsymbol{t}$, and let $\boldsymbol{y} = 2\boldsymbol{t} - 5$. Show that the vectors $\boldsymbol{t}$ and $\boldsymbol{y}$ have a correlation of 1.
   (d) Redo the previous experiment, but use any negative slope. What is the correlation coefficient?

3. Let $\boldsymbol{x}$ be a vector of data with mean $\bar{x}$, and let a, b be scalars.
   (a) Show, using the definition of the mean, that the mean of $a\boldsymbol{x}$ is $a\bar{x}$.
   (b) Similarly, find a formula for the mean of $a\boldsymbol{x} + b$ in terms of the mean of $\boldsymbol{x}$.

4. Let $\boldsymbol{x}$ be a vector of data with variance s_x^2, and let a, b be scalars.
   (a) Show, using the definition of variance, that the variance of $a\boldsymbol{x}$ is $a^2 s_x^2$. You might start with:

$$s_{(ax)}^2 = \frac{1}{m-1}\sum_{i=1}^{m}(ax_i - \overline{ax_i})^2$$

   (b) Similarly, find a formula for the variance of $ax + b$ in terms of the variance of $\boldsymbol{x}$.

The exercises below explore the notion that the correlation tries to find linear relationships between data.

5. Show that, for data in vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and a real scalar a,

$$\mathrm{Cov}(ax, y) = a\mathrm{Cov}(x, y) = \mathrm{Cov}(x, ay)$$

6. For a and b fixed scalars, and data in vector $\boldsymbol{x}$ find a formula for the $\mathrm{Cov}(x, ax + b)$ in terms of the variance of $\boldsymbol{x}$.

7. For a and b fixed scalars, and data in vector $\boldsymbol{x}$, let $\boldsymbol{y} = a\boldsymbol{x} + b$, find the correlation coefficient r_{xy}^2 and simplify as much as possible. What do you get?

8. Let X be a $p \times n$ matrix of data, where we n columns of p data points (you may assume each column has zero mean). Show that the $(i, j)^{\mathrm{th}}$ entry of $\dfrac{1}{p-1}X^T X$ is the covariance between the i^{th} and j^{th} columns of X. HINT: It might be convenient to write X in terms of its columns,

$$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$$

Also show that $\frac{1}{p-1}X^T X$ is a symmetric matrix.