UNIT IV

STATISTICAL THINKING

Distributions – Representing and plotting histograms, Outliers, Summarizing distributions, Variance, Reporting results; Probability mass function – Plotting PMFs, Other visualizations, The class size paradox, Data frame indexing; Cumulative distribution functions - Limits of PMFs, Representing CDFs, Percentile based statistics, Random numbers, Comparing percentile ranks; Modeling distributions - Exponential distribution, Normal distribution, Lognormal distribution.

4.1 Representing Histograms:

The Hist constructor can take a sequence, dictionary, pandas Series, or another Hist. we can instantiate a Hist object using the following command:

```
>>> import thinkstats2
>>> hist = thinkstats2.Hist([1, 2, 2, 3, 5])
>>> hist Hist({1: 1, 2: 2, 3: 1, 5: 1})
```

Hist objects provide Freq, which takes a value and returns its frequency:

```
>>> hist.Freq(2)2
```

The bracket operator does the same thing:

```
>>> hist[2]2
```

If you look up a value that has never appeared, the frequency is 0:

```
>>> hist.Freq(4)0
```

Values returns an unsorted list of the values in the Hist:

```
>>> hist.Values()[1, 5, 3, 2]
```

To loop through the values in order, we can use the built-in function sorted: for val in sorted(hist.Values()):

```
print(val, hist.Freq(val))
```

Or we can use Items to iterate through value-frequency pairs:

```
for val, freq in hist.Items():
print(val, freq)
```

4.1.1 Plotting Histogram:

A module called thinkplot.py that provides functions for plotting Hists and other objects definedin thinkstats2.py. It is based on pyplot,which is part of the matplotlib package.

To plot hist with thinkplot:

```
>>> import thinkplot
>>> thinkplot.Hist(hist)
>>> thinkplot.Show(xlabel='value', ylabel='frequency')
```

4.1.2 Outliers

Considering of histograms is easy to identify the most common values and the shape of the distribution, but rare values are not always visible.

Outliers are extreme values that might be errors in measurement and recording, or might be accurate reports of rare events.

Hist provides methods Largest and Smallest, which take an integer n and return the n largest or smallest values from the histogram:

```
for weeks, freq in hist.Smallest(10):
print(weeks, freq)
```

In the list of pregnancy lengths for live births, the 10 lowest values are [0, 4, 9, 13, 17, 18, 19, 20,21, 22]. Values below 10 weeks are certainly errors; the most likely explanation is that the outcomewas not coded correctly. Values higher than 30 weeks are probably legitimate. Between 10 and 30weeks, it is hard to be sure; some values are probably errors, but some represent premature babies.On the other end of the range, the highest values are:

weeks	count
43	148
44	46
45	10
46	1

47	1
48	7
50	2

Most doctors recommend induced labor if a pregnancy exceeds 42 weeks, so some of the longer values are surprising. In particular, 50 weeks seems medically unlikely.

The best way to handle outliers depends on “domain knowledge”; that is, information about wherethe data come from and what they mean. And it depends on what analysis we are planning to perform.

In this example, the motivating question is whether first babies tend to be early (or late), they are usually interested in full-term pregnancies, so for this analysis will focus on pregnancies longer than 27 weeks.

First Babies:

Now compare the distribution of pregnancy lengths for first babies and others. Divide the DataFrame of live births using birthord, and computed their histograms:

firsts = live[live.birthord == 1]others = live[live.birthord != 1]

first_hist = thinkstats2.Hist(firsts.prglngth) other_hist = thinkstats2.Hist(others.prglngth) Then plot their histograms on the same axis:

width = 0.45 thinkplot.PrePlot(2) thinkplot.Hist(first_hist,align='right',width=width) thinkplot.Hist(other_hist, align='left', width=width)thinkplot.Show(xlabel='weeks', ylabel='frequency')

thinkplot.PrePlot takes the number of histograms planning to plot;it uses this information to choosean appropriate collection of colors.

thinkplot.Hist normally uses align=’center so that each bar is centered over its value.

For this figure, use align=’right’ and align=’left’ to place corresponding bars on either side of the value. Withwidth=0.45, the total widthofthetwobarsis0.9, leaving some space between each pair.

Finally, adjust the axis to show only data between27and46weeks.Figure 4.1shows the corresponding result.

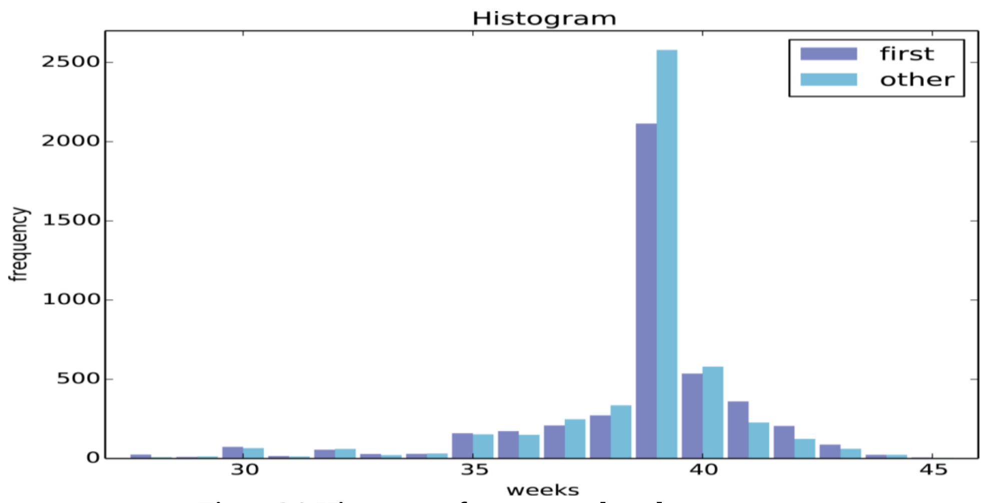


Figure4.1-Histogram of pregnancy lengths.

Histograms are useful because they make the most frequent values immediately apparent. But they are not the best choice for comparing two distributions. In this example, there are fewer "first babies" than" others," so some of the apparent differences in the histograms are due to sample sizes.

4.2 Summarizing Distributions:

A histogram is a complete description of the distribution of a sample; that is, given a histogram,could reconstruct the values in the sample(although not their order).

If the details of the distribution are important, it might be necessary to present a histogram. Butoften want to summarize the distribution with a few descriptive statistics.

Some of the characteristics we might want to report are:

Central tendency

Do the values tend to cluster around a particular point?

Modes

Is there more than one cluster?

spread

How much variability is there in the values?

Tails

How quickly do the probabilities drop off as we move away from the modes?

outliers

Are there extreme values far from the modes?

Statistics designed to answer these questions are called **summary statistics**. By far the most common summary statistic is the **mean**, which is meant to describe the **central tendency** of the distribution.

The words *mean* and *average* are sometimes used interchangeably, but we make this distinction:If we have a sample of nvalues, x_i, the mean, $\bar{x}$, is the sum of the values divided by the number of value

$$\bar{x} = \frac{1}{n}\sum_i x_i$$

- The *mean* of a sample is the summary statistic computed with the previous formula.
- An *average is* one of several summary statistics we might choose to describe a centraltendency. Sometimes the mean is a good description of a set of values. For example, apples are all pretty much the same size(atleast the ones sold in supermarkets).So if I buy 6 apples and the total weightis 3 pounds, it would be area son able summary to say they are about a half pound each.

But pumpkins are more diverse. Suppose I grow several varieties in my garden, and one day I harvest three decorative pumpkins that are 1 pound each, two pie pumpkins that are 3pounds each, and one Atlantic Giant pumpkin that weighs 591pounds.Themeanof this sample is 100 pounds, but if I told you “The average pumpkin in my garden is 100pounds,”that would be misleading. In this example, there is no meaningful average because there is no typical pumpkin.

Variance

If there is no single number that summarizes pumpkin weights, we can do a little better with numbers: mean and **variance**.

Variance is a summary statistic intended to describe the variability or **spread** of a distribution.The variance of a set of values is

$$S^2 = \frac{1}{n}\sum_i (x_i - \bar{x})^2$$

The term x_i-$\bar{x}$ is called the “deviation from the mean,” so variance is the mean squared deviation. The square root of variance, *S*, is the **standard deviation**.

Pandas data structures provides methods to compute mean, variance and standard deviation:mean = live.prglngth.mean()var =
live.prglngth.var()std = live.prglngth.std()

For all live births, the mean pregnancy lengthis38.6weeks,the standard deviation is2.7weeks,which means we should expect deviations of2-3weeks to be common.

Variance of pregnancy length is 7.3, which is hard to interpret, especially since the units are weeks,or" square weeks." Variance is useful in some calculations, but it is not a good summary statistic.

4.2.1 Reporting Results:

There are several ways to describe the difference in pregnancy length (if there is one) between first babies and others. How should report these results?

The answer depends on question. A scientist might be interested in any (real) effect, no matter how small. A doctor might only care about effects that are **clinically significant**; that is, differences that affect treatment decisions. A pregnant woman might be interested in results thatare relevant to her, like the probability of delivering early or late.

4.3 Probability mass function

Another way to represent a distribution is a probability mass function (PMF), which maps fromeach value to its probability.

*A probability is a frequency expressed as a fraction of the sample size, n. To get from frequenciesto probabilities, we divide through by n, which is called normalization***.**

Given a Hist, we can make a dictionary that maps from each value to its probability:n = hist.Total()

```
d ={}
for x, freq in hist.Items():
d[x] = freq / n
```

Or use Pmf class provided by think stats2. Like Hist, the Pmf constructor can take a list, pandasSeries, dictionary, Hist, or another Pmf object.

Here's an example with a simple list:

```
>>> import thinkstats2
>>> pmf = thinkstats2.Pmf([1, 2, 2, 3, 5])
>>> pmf
Pmf({1: 0.2, 2: 0.4, 3: 0.2, 5: 0.2})
```

ThePmfisnormalizedsototalprobabilityis1.

Pmf and Hist objects are similar in many ways; in fact, they inherit many of their methods from a common parent class. For example, the methods Values and Items work the same way for both.The biggest difference is that a Hist maps from values to integer counters; a Pmf maps from values to floating-point probabilities.

To lookup the probability associated with a value, use Prob:

```
>>> pmf.Prob(2)0.4
```

The bracket operator is equivalent:

```
>>> pmf[2]0.4
```

We can modify an existing Pmf by incrementing the probability associated with a value:

```
>>> pmf.Incr(2, 0.2)
>>>pmf.Prob(2)0.6
```

Or we can multiply a probability by a factor:

```
>>> pmf.Mult(2, 0.5)
>>> pmf.Prob(2)0.3
```

If we modify a Pmf, the result may not be normalized; that is, the probabilities may no longeradd up to 1. To check, you can call Total, which returns the sum of the probabilities:

```
>>> pmf.Total()0.9
```

To re normalize, call Normalize:

```
>>> pmf.Normalize()
>>> pmf.Total()1.0
```

Pmf objects provide a Copy method so we can make and modify a copy without affecting the original.

My notation in this section might seem inconsistent, but there is a system: we use Pmf for the name of the class, pmf for an instance of the class, and PMF for the mathematical concept of a probability mass function.

4.4 Plotting PMFs:

Thinkplot provides two ways to plotPmfs:

To plot a Pmf as a bar graph, we can use think plot. Hist. Bar graphs are most useful lif the number of values in the Pmf is small.

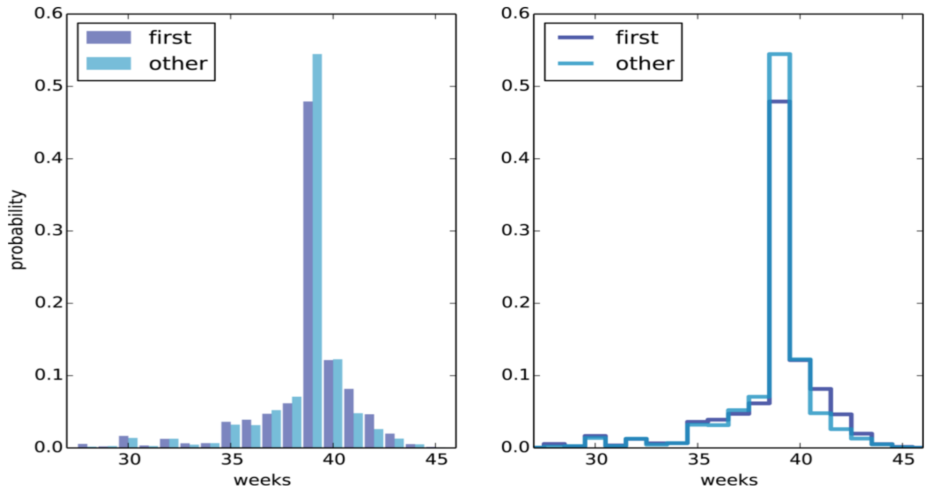
Toplot a Pmf as a step function, we can use thinkplot. Pmf. This option is most useful if there area large number of values and the Pmf is smooth. This functional so works with Hist objects.

Here’s the code that generates Figure 4.2: thinkplot.PrePlot(2,cols=2)thinkplot.Hist(first_pmf, align='right', width=width)thinkplot.Hist(other_pmf, align='left', width=width)thinkplot.Config(xlabel='weeks', ylabel='probability',axis=[27,46,0,0.6]
)
thinkplot.PrePlot(2)thinkplot.SubPlot(2)thinkplot.P mfs([first_pmf, other_pmf])thinkplot.Show(xlabel='weeks',
axis=[27, 46, 0, 0.6])

By plotting the PMF instead of the histogram, we can compare the two distributions withoutbeing misled by the difference in sample size

Figure 4.2 shows PMF so fpregnancy length for first babies and others using bargraphs(left) and stepfunctions(right).

Figure 4.2. PMF of pregnancy lengths for first babies and others, using bar graphs and step functions



4.4.1 Other Visualizations

Patterns and relationships. Once have an idea what is going on,a good nextstep is to design a visualization that makes the patterns we have identified as clear as possible.

weeks = range(35, 46)diffs = []

for week in weeks:

p1 = first_pmf.Prob(week) p2 = other_pmf.Prob(week)

diff = 100 * (p1 - p2)diffs.append(diff)thinkplot.Bar(weeks, diffs)

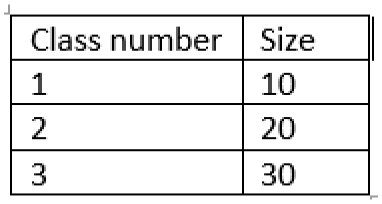
In this code, weeks is the range of weeks; diffs is the difference between the two PMFs in percentagepoints. Figure 4.2 shows the result as a bar chart. This figure makes the pattern clearer: first babiesare less likely to be born in week 39, and somewhat more likely to be born in weeks 41 and 42. use the same dataset to identify an apparent difference and then chose a visualization that makesthe difference apparent.

4.5 THE CLASS SIZE PARADOX

Class size paradox tells us that students tend to experience a greater number of classmates than thetrue average class size because most of a large number of those experiences are in the classes withmost students. Confusing?

Let’s examine this paradox with an example below.

I will calculate the Actual class size and Observed class size:



Class number	Size
1	10
2	20
3	30

If you calculate mean class size of the above dataset, it is going to be (10 + 20 + 30)/3 = 20

Actual class size = 20

- But, if you survey each student about their class size, a student in class 1 is going to say I have 9 other classmates in the class and student in class 2 is going to say that he/she has 19other classmates and so on.

Let’s take their responses into consideration and calculate the mean. This Mean is also called as Observed mean.

Observed mean = (10*10)+ (20 * 20) + (30 * 30)/(10+20+30)Observed mean = (100 + 400 + 900)/50

Observed Mean = 23.33

- I feel the observed mean calculation is like any weighted average calculation that we do inmost of the statistical exercises.

You can also calculate the observed mean using PMF for the above dataset:PMF = [(10: 10/60), (20: 20/60), (30: 30/60)]

PMF = [(10: 0.167), (20: 0.333), (30: 0.500)]

So the mean of the class size is: 10*0.167+20*0.333+30*0.500 = 23.33

Observed mean = 23.33

As you can see, the observed mean is higher than the actual mean. This is what class size paradox teaches us.

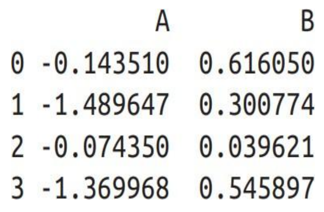
4.6 DATA FRAME INDEXING

In“DataFrames”wereadapandasDataFrameandusedittoselectandmodifydatacolumns.lookatrowsel ection.To,create a NumPyarray of random numbers and use it to initialize a DataFrame:

```
>>> import numpy as np
>>> import pandas
>>> array = np.random.randn(4, 2)
>>> df = pandas.DataFrame(array)
>>> df
```

By default, the rows and columns are numbered starting at zero, but we can provide column names:

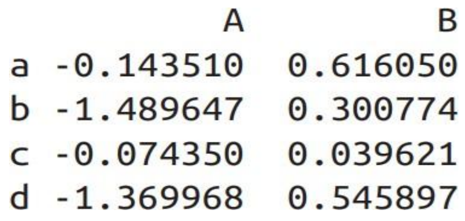
```
>>> columns = ['A', 'B']
>>> df = pandas.DataFrame(array, columns=columns)
>>> df
```



```
          A         B
0 -0.143510  0.616050
1 -1.489647  0.300774
2 -0.074350  0.039621
3 -1.369968  0.545897
```

We can also provide row names. The set of row names is called the **index**; the row names themselves are called **labels**.

```
>>> index = ['a', 'b', 'c', 'd']
>>> df = pandas.DataFrame(array, columns=columns, index=index)
>>> df
```



```
          A         B
a -0.143510  0.616050
b -1.489647  0.300774
c -0.074350  0.039621
d -1.369968  0.545897
```

4.7 CUMULATIVE DISTRIBUTION FUNCTIONS

4.7.1 The Limits of PMFs:

- PMFs work well if the number of values is small. But as the number of values increases,the probability associated with each value gets smaller and the effect of random noise Increases.
- The problems can be mitigated by binning the data; that is, dividing the range of valuesinto non-overlapping intervals and counting the number of values in each bin.

- Binning can be useful, but it is tricky to get the size of the bins right. If they are big enoughto smooth out noise, they might also smooth out useful information.
- It is hard to tell which features are meaningful. Also, it is hard to see overall patterns insome cases.

4.7.2 Percentiles:

Example:

- If we consider a standardized test, we probably got the results in the form of a raw score and a percentile rank. In this context, the percentile rank is the fraction of peoplewho scored lower than you (or the same). So if we are "in the 90th percentile," we didas well as or better than 90% of the people who took the exam.

  Computation of the percentile rank value in the sequence scores: def PercentileRank(scores, your_score):

  count = 0

  for score in scores:

  if score <= your_score:

  count += 1

  percentile_rank = 100.0 * count / len(scores)return percentile_rank
- As an example, if the scores in the sequence were 55, 66, 77, 88 and 99, and we will get thevalue 88, then the percentile rank would be 100 * 4 / 5 which is 80.
- For a given a percentile rank and we want to find the corresponding value, one option is tosort the values and search for the one we want:

def Percentile(scores, percentile_rank):scores.sort()

for score in scores:

if PercentileRank(scores, score) >= percentile_rank:return score

- The result of this calculation is a percentile. For example, the 50th percentile is the valuewith percentile rank 50. In the distribution of exam scores, the 50th percentile is 77.

- To summarize, Percentile Rank takes a value and computes its percentile rank in a set ofvalues; Percentile takes a percentile rank and computes the corresponding value.

4.8 CDFs:

- The CDF is the function that maps from value to its percentile rank.
- The CDF is a function of x, where x is any value that might appear in the distribution.
- To evaluate CDF(x) for a particular value of x, we compute the fraction of values in thedistribution less than or equal to x.

  Let us consider the following Example : a function that takes a sequence, t, and a value, x:

  ```
  def EvalCdf(t, x):count = 0.0
  for value in t:
  if value <= x:
  count += 1
  prob = count / len(t)return prob
  ```

- This function is almost identical to PercentileRank, except that the result is probabilty inthe range 0–1 rather than a percentile rank in the range 0–100.

  As an example, suppose we collect a sample with the values [1, 2, 2, 3, 5]. Here aresome values from its CDF:

  CDF (0) = 0

  CDF (1) = 0.2

  CDF (2) = 0.6

  CDF (3) = 0.8

  CDF (4) = 0.8

  CDF (5) = 1

- We can evaluate the CDF for any value of x, not just values that appear in the sample.If x is less than the smallest value in the sample, CDF(x) is 0. If x is greater than the largest value, CDF(x) is 1.

Representing CDFs:

- Prob(x)

  Given a value x, computes the probability p = CDF(x). The bracket operator isequivalent to Prob.

- Value(p)

  Given a probability p, computes the corresponding value, x; that is, the inverseCDF of p.

➔ The Cdf constructor can take as an argument a list of values, a pandas Series, a Hist,Pmf, or another Cdf.

➔ The following code makes a Cdf for the distribution of pregnancylengths in the NSFG:

```
live, firsts, others = first.MakeFrames()
cdf = thinkstats2.Cdf(live.prglngth, label='prglngth')
```

❖ thinkplot provides a function named Cdf that plots Cdfs as lines:thinkplot.Cdf(cdf)

```
thinkplot.Show(xlabel='weeks', ylabel='CDF')
```

Percentile based statistics:

Introduction to Percentiles

Percentiles are statistical measures that divide a dataset into specific segments, indicating the relative position of a particular value within the distribution. They help us understand how data points are distributed in relation to each other and provide valuable insights into the dataset's overall characteristics. In data science, percentiles are commonly used for various purposes such as analyzing data distributions, outlier detection, and understanding data variability. Calculating Percentiles

To calculate percentiles, follow these steps:

Step 1: Arrange Data in Ascending Order Sort the dataset in ascending order. This ensures that thedata points are organized for percentile calculation.

Step 2: Identify the Desired Percentile Determine which percentile you want to calculate. Common percentiles include the median (50th percentile), quartiles (25th and 75th percentiles), and various other percentiles like the 90th or 95th percentile.

Step 3: Calculate the Position Use the formula position = (percentile / 100) * (n + 1) to find the position of the desired percentile within the dataset, where n is the total number of data points.

Step 4: Find the Data Value If the position is a whole number, the data value at that position is the desired percentile. If the position is not a whole number, calculate the weighted average of the data values at the floor and ceiling positions.

Code:

```
import numpy as np

data = [18, 22, 25, 27, 38, 33, 37, 41, 45, 50]

# Step 1: Sort data

sorted_data  =  np.sort(data)

# Step 2: Calculate positions

percentile_25 = 25

position_25 = (percentile_25 / 100) *(len(sorted_data) + 1)

percentile_75 = 75

position_75 = (percentile_75 / 100) *(len(sorted_data) + 1)

# Step 3 and 4: Calculate percentiles

if position_25.is_integer():
  percentile_value_25 = sorted_data[int(position_25)-1]

else:
  floor_position_25= int(np.floor(position_25))

  ceil_position_25= int(np.ceil(position_25))
  percentile_value_25 = sorted_data[floor_position_25] + (position_25 -
floor_position_25) * (sorted_data[ceil_position_25] -
sorted_data[floor_position_25])


if position_75.is_integer():

  percentile_value_75= sorted_data[int (position_75) - 1]

else:
  floor_position_75= int(np.floor(position_75))
  ceil_position_75= int(np.ceil(position_75))
  percentile_value_75 = sorted_data[floor_position_75] + (position_75 -
floor_position_75) * (sorted_data[ceil_position_75] -
sorted_data[floor_position_75])

print(f"25th percentile: {percentile_value_25}")
print(f"75th percentile: {percentile_value_75}")
```

output:

```
25th percentile: 26.5
75th percentile: 46.25
```

In this example, we calculate the 25th and 75th percentiles of the dataset using Python's NumPy library for sorting and mathematical functions.

Percentiles are crucial statistical measures in data science for understanding data distributions and making informed decisions. By calculating percentiles, you gain insights into the relative positionof data points within a dataset, helping you analyze the spread and variability of the data. Read more on Sarthaks.com -

Q: What is a percentile in statistics?

A: A percentile is a measure used in statistics to indicate a particular position within a dataset. It represents the value below which a given percentage of observations fall. For example, the 25th percentile (also known as the first quartile) is the value below which 25% of the data points lie.

Q: How do you calculate the kth percentile?

A: To calculate the kth percentile, follow these steps: Arrange the data in ascending order. Calculate the desired position using the formula: Position = (k / 100) * (N + 1), where N is the total number of data points. If the position is an integer, the kth percentile is the value at that position. If the position is not an integer, the kth percentile is the average of the values at the floor(position) and ceil(position) positions.

Q: How do you calculate percentiles in Python?

A: You can calculate percentiles in Python using the numpy library.Here's an example:

```
import numpy as np data = [12, 15, 17, 20, 23, 25, 27, 30, 32, 35]
percentile_value = 25
# For the 25th percentile
result = np. percentile (data, percentile_value)
print (f"The {percentile_value}th percentile is: {result}")
```

output:

```
The 25th percentile is: 17.75
```

Q: How do you find the median using percentiles?

A: The median is the 50th percentile. You can find it using the same numpy library: import numpyas np

```
data = [12, 15, 17, 20, 23, 25, 27, 30, 32, 35]
median = np. percentile (data, 50) print (f"The median is: {median}")
```

output:

```
The median is: 24.0
```

Q: What is the interquartile range (IQR)?

A: The interquartile range is a measure of statistical dispersion, based on the difference between the third quartile (Q3, 75th percentile) and the first quartile (Q1, 25th percentile). It gives an ideaof how spread out the middle 50% of the data is.

Q: How do you calculate the IQR in Python?

A: You can calculate the interquartile range using the numpy library:

```
import numpy as np data = [12, 15, 17, 20, 23, 25, 27, 30, 32, 35]q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)iqr = q3 - q1
print(f"The interquartile range (IQR) is: {iqr}")
```

output:

```
The interquartile range (IQR) is: 11.5
```

Important Interview Questions and Answers on Data Science - Statistics Percentiles

Q: What is a percentile?

A percentile is a measure used in statistics to describe the relative standing of a particular value within a dataset. It indicates the percentage of values that are less than or equal to the given value.

Q: How do you calculate the nth percentile?

To calculate the nth percentile, you can follow these steps:

1. Arrange the data in ascending order.
2. Compute the index (position) of the desired percentile using the formula: index = (percentile / 100) * (N + 1), where N is the total number of data points.
3. If the index is an integer, the nth percentile is the value at that index. If the index is not an integer, take the weighted average of the values at the integer part of the index and the next higher index.

Q: What is the median? How is it related to the 50th percentile?

The median is the value that separates a dataset into two equal halves when the data is ordered. It'salso the 50th percentile, as it's the value below which 50% of the data falls.

Q: How do you interpret the 25th and 75th percentiles, also known as the first and third quartiles? The 25th percentile (Q1) is the value below which 25% of the data falls. The 75th percentile (Q3)is the value below which 75% of the data falls. The interquartile range (IQR) is the difference between the third and first quartiles (IQR = Q3 - Q1), which provides a measure of the spread of the middle 50% of the data.

Q: How do you handle outliers when calculating percentiles?

Outliers can significantly affect percentile calculations. One approach to handle outliers is to use the "trimmed" dataset (removing the extreme values) to calculate percentiles. Another approach isto use robust estimators like the median, which is less sensitive to outliers compared to the mean.

Example Code: Calculating Percentiles in Pythonimport numpy as np

# Sample dataset

data = [12, 15, 18, 20, 22, 25, 28, 30, 35, 40]

 # Calculate the 25th and 75th percentiles (first and third quartiles)q1 = np.percentile(data, 25)

q3 = np.percentile(data, 75) print("Q1 (25th percentile):", q1)print("Q3 (75th percentile):", q3)**output:**

Q1 (25th percentile): 18.5Q3 (75th percentile): 29.5

In this example, the numpy library's percentile function is used to calculate the desired percentiles.

4.8 Random Numbers:

Example:

- If we choose a random sample from the population of live births and look up thepercentile rank of their birth weights we use CDF which is shown in 4.3
- Computation of CDF's of birth weights use the following codeweights = live.totalwgt_lb
  cdf = thinkstats2.Cdf(weights, label='totalwgt_lb')
- Then we generate a sample and compute the percentile rank of each value in the sample.sample = np.random.choice(weights, 100, replace=True)
  ranks = [cdf.PercentileRank(x) for x in sample]
- sample is a random sample of 100 birth weights, chosen with replacement; that is, thesame value could be chosen more than once. ranks is a list of percentile ranks.

● Finally we make and plot the Cdf of the percentile ranks. rank_cdf = thinkstats2.Cdf(ranks) thinkplot.Cdf(rank_cdf) thinkplot.Show(xlabel='percentile rank', ylabel='CDF')

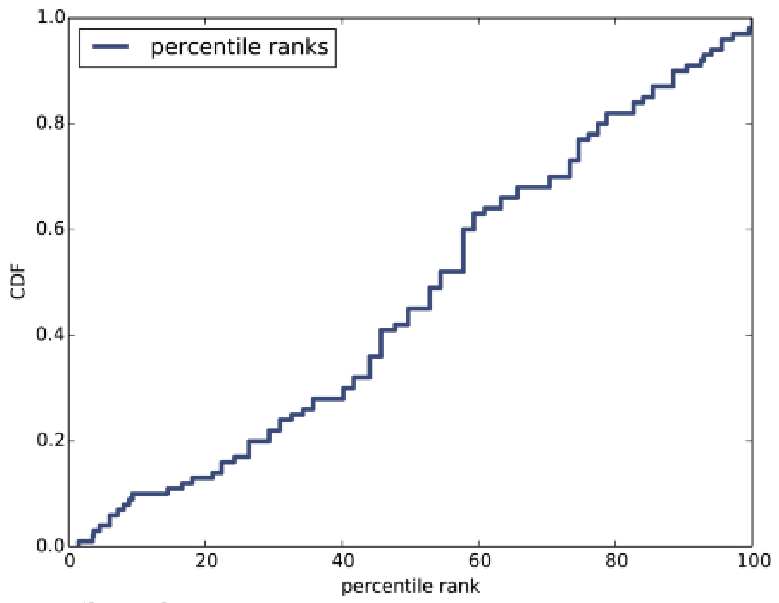


Fig 4.3: CDF of Percentile Rank

The CDF is approximately a straight line, which means that the distribution is uniform

4.9 Comparing Percentile Ranks

● Percentile ranks are useful for comparing measurements across different groups. **Example**: People who compete in foot races are usually grouped by age and gender. Tocompare people in different age groups, you can convert race times to percentile ranks.

● A few years ago Sam ran the James Joyce Ramble 10K in Dedham MA; Sam finished in 42:44, which was 97th in a field of 1633. sam beat or tied 1537 runners out of 1633, Sam percentile rank in the field was 94%.

● More generally, given position and field size, we can compute percentile rank: def PositionToPercentile(position, field_size):

beat = field_size - position + 1 percentile = 100.0 * beat / field_size

return percentile

- In sam age group, denoted M4049 for “males between 40 and 49 years of age”, sam camein 26th out of 256. So my percentile rank in my age group was 90%.

Modeling distributions:

Exponential distribution:

The exponential distribution represents the time until a continuous, random event occurs. In the context of reliability engineering, this distribution is employed to model the lifespan of a device or system before it fails. This information aids in maintenance planning and ensuring uninterrupted operation.

The time intervals between successive earthquakes in a certain region can be accurately modeled by an exponential distribution. This is especially true when these events occur randomly over time,but the probability of them happening in a particular time frame is constant.

Normal distribution:

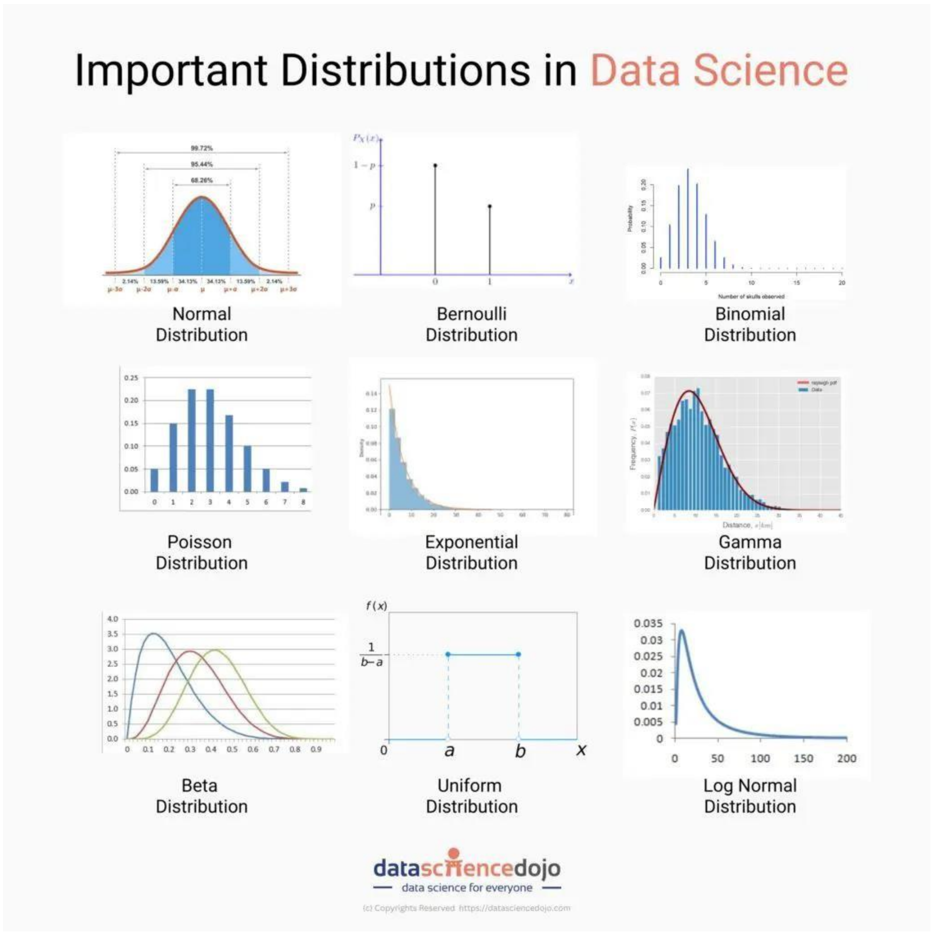
The normal distribution, characterized by its bell-shaped curve, is prevalent in various natural phenomena. For instance, IQ scores in a population tend to follow a normal distribution. This allows psychologists and educators to understand the distribution of intelligence levels and makeinformed decisions regarding education programs and interventions.

Heights of adult males in a given population often exhibit a normal distribution. In such a scenario,most men tend to cluster around the average height, with fewer individuals being exceptionally tall or short. This means that the majority fall within one standard deviation of the mean, while a smaller percentage deviates further from the average.

Lognormal distribution:

The log normal distribution describes a random variable whose logarithm is normally distributed.In finance, this distribution is applied to model the prices of financial assets, such as stocks. Understanding the log normal distribution is crucial for making informed investment decisions.

The distribution of wealth among individuals in an economy often follows a log-normal distribution. This means that when the logarithm of wealth is considered, the resulting values tendto cluster around a central point, reflecting the skewed nature of wealth distribution in many societies.


Important Distributions in Data Science
Normal Distribution
Bernoulli Distribution
Binomial Distribution
Poisson Distribution
Exponential Distribution
Gamma Distribution
Beta Distribution
Uniform Distribution
Log Normal Distribution
datasciencedojo
data science for everyone
(c) Copyrights Reserved https://datasciencedojo.com