Statistics

PROF. JACQUES SAVOY
UNIVERSITY OF NEUCHATEL

Contents

Example

Location statistics

Procedure for a statistical test

T-test

Chi-square test

Application

National survey (in United States) about what Americans think and feel (gss.norc.org)

General Social Survey (GSS) from 1998, 2000, and 2002 (in our application).

Focusing only on the culture and arts (in United States)

Random sample of the US population (320M).

Data available in SPSS or STATA format.

Notation: Population

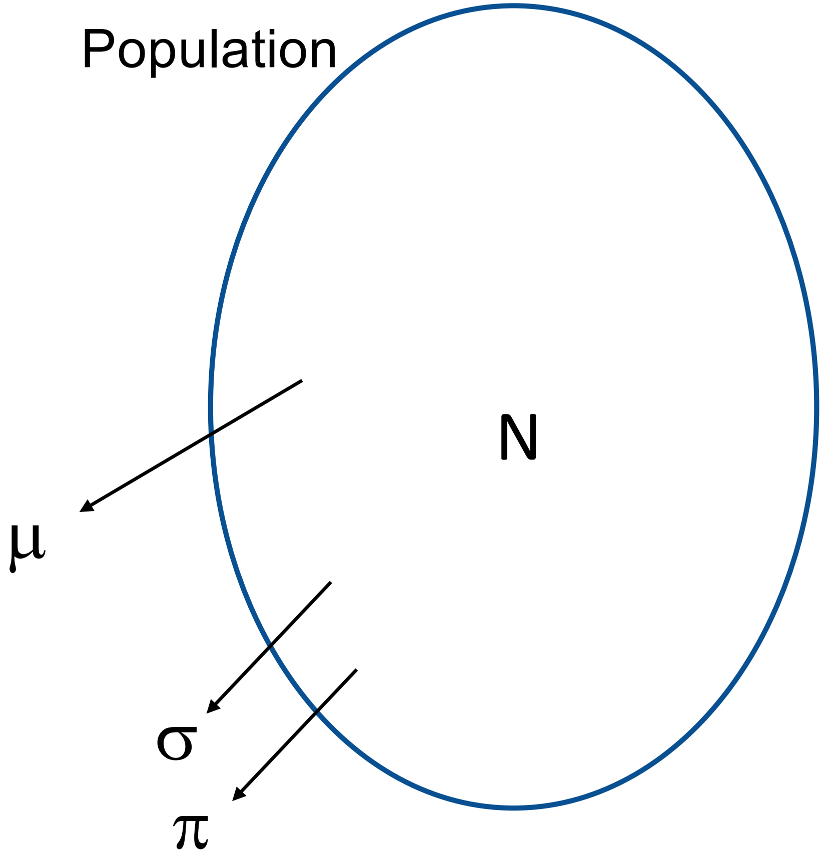
- Size of the population is denoted by N.
- Values of the variable(s) in a population are denoted by upper-case letters X, Y, Z, … (e.g., number of novels read during the last 12 months). (sometimes in lowercase).
- The values for the *n* individuals in the population by letters with subscripts X_1, X_2, …, X_k, …, X_N (e.g., 0, 3, 2, 1, 0, …).

  Be careful: X_k could denote a scalar or a vector of values.
- Population parameters are symbolized by Greek letters like μ, σ, π, …

  Recall that such values are *fixed* (and are not random variables).

  If you can ask all persons belonging to the defined population, you know exactly the number of novels read par each person.

Statistics Principle



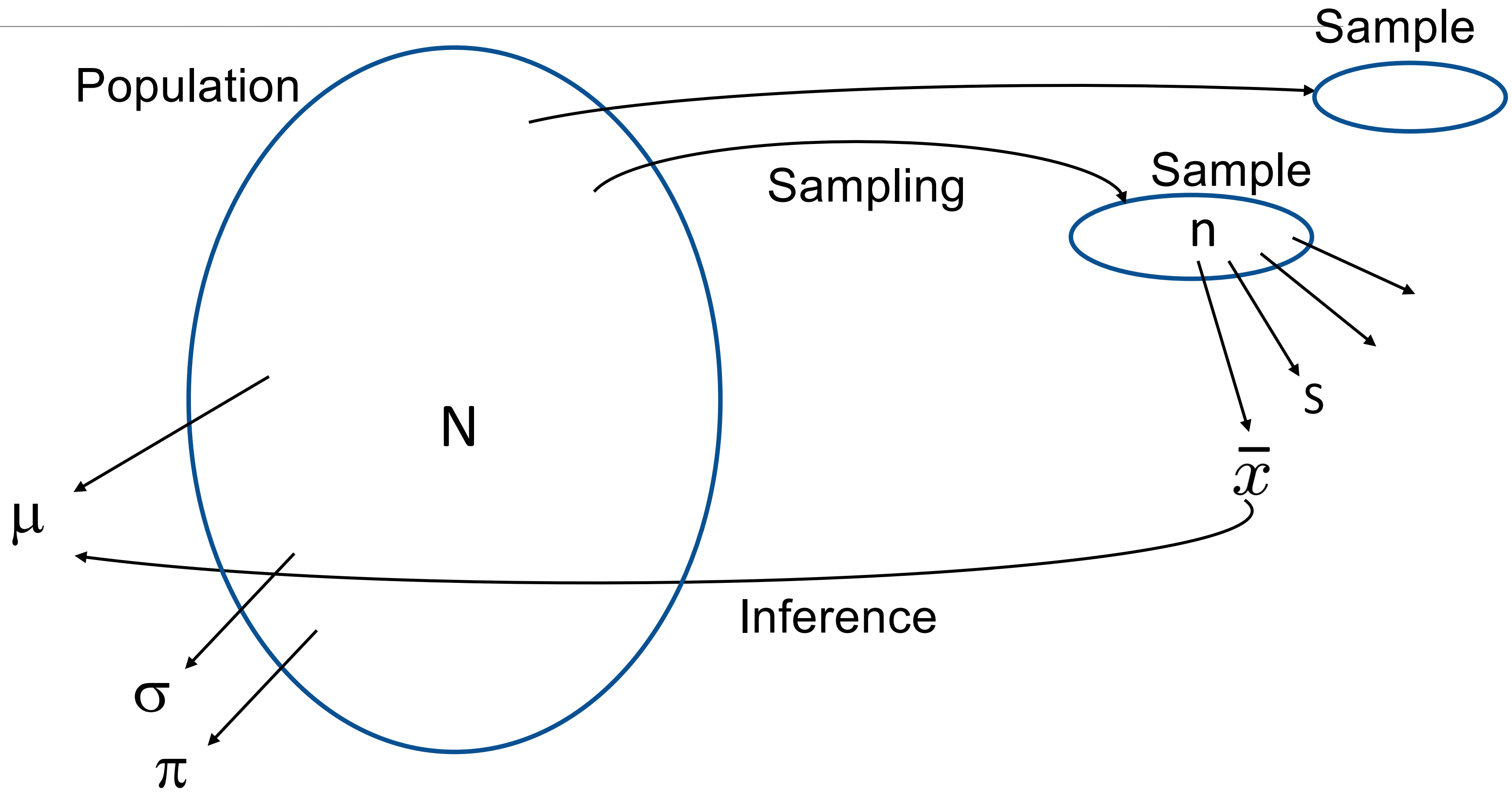
The problem:
We don't have the time, resources,
… to analyze the *entire* population!
320 millions of persons in US!

Notation: Sample

- Sampling procedure: select *randomly* a subset of the population (each citizen has the same chance to be selected). (Are we sure to respect this random aspect?)
- The sample size is denoted by *n.*
- Individual observations for a given variable are distinguished by subscripts x_1, x_2, …, x_i, …, x_n (and we used lowercase letters).
- Sample estimation of population parameters are represented by lower-case (roman) letters, such as s, p, or $\overline{x}$ with $\bar{x} = \hat{\mu}$

  These estimations may vary (if we take another sample or modify slightly the current one) and thus are subject to random variations (random variables).
- From them, one can infer the real (but unknow) values of the population.

Principle



Example

Analysis the mean of the population.

Population mean $$\mu = \frac{X_1 + X_2 + \cdots + X_N}{N} = \frac{\sum_{i=1}^{N} X_i}{N}$$

Sample mean $$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$$

If we change the sample, $\overline{x}$ will change but never μ.

A statistics = a function over the sample.

Plug-in principle: we use the same formula for the sample or for the population (recycling the idea).

Sometimes, the formula could differ between the one used for the population and for the sample.

Example

```
>>> import random, numpy; random.seed(1); numpy.random.seed(1);
```

Dataset GSS7214_R5.DTA is stored in compressed from as GSS7214_R5.DTA.gz

```
>>> import gzip
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> import math
>>> import statistics
>>> import collections
>>> import scipy.stats
```

Example

```
>>> with gzip.open('data/GSS7214_R5.DTA.gz', 'rb') as infile:
```

We restrict this (large) dataset to the variables of interest

```
...     columns = ['id', 'year', 'age', 'sex', 'race', 'reg16', 'degree',
...                'realrinc', 'readfict']
...     df = pd.read_stata(infile, columns=columns)
...
>>> len(df)
59599
>>> df.head()
   id  year age     sex   race            reg16          degree realrinc readfict
0   1  1972  23  female  white  middle atlantic        bachelor      NaN      NaN
1   2  1972  70    male  white  e. nor. central  lt high school      NaN      NaN
2   3  1972  48  female  white  e. nor. central     high school      NaN      NaN
3   4  1972  27  female  white          foreign        bachelor      NaN      NaN
4   5  1972  61  female  white  e. nor. central     high school      NaN      NaN
```

Example

```
>>> len(df)
    59599
```

Further limit dataset to the years we are interested in

```
>>> df = df.loc[df['year'].isin({1998, 2000, 2002})]
>>> len(df)
    8414
```

Limit dataset to exclude records from individuals who refused to report their income

```
>>> df = df.loc[df['realrinc'].notnull()]
>>> len(df)
    5447
```

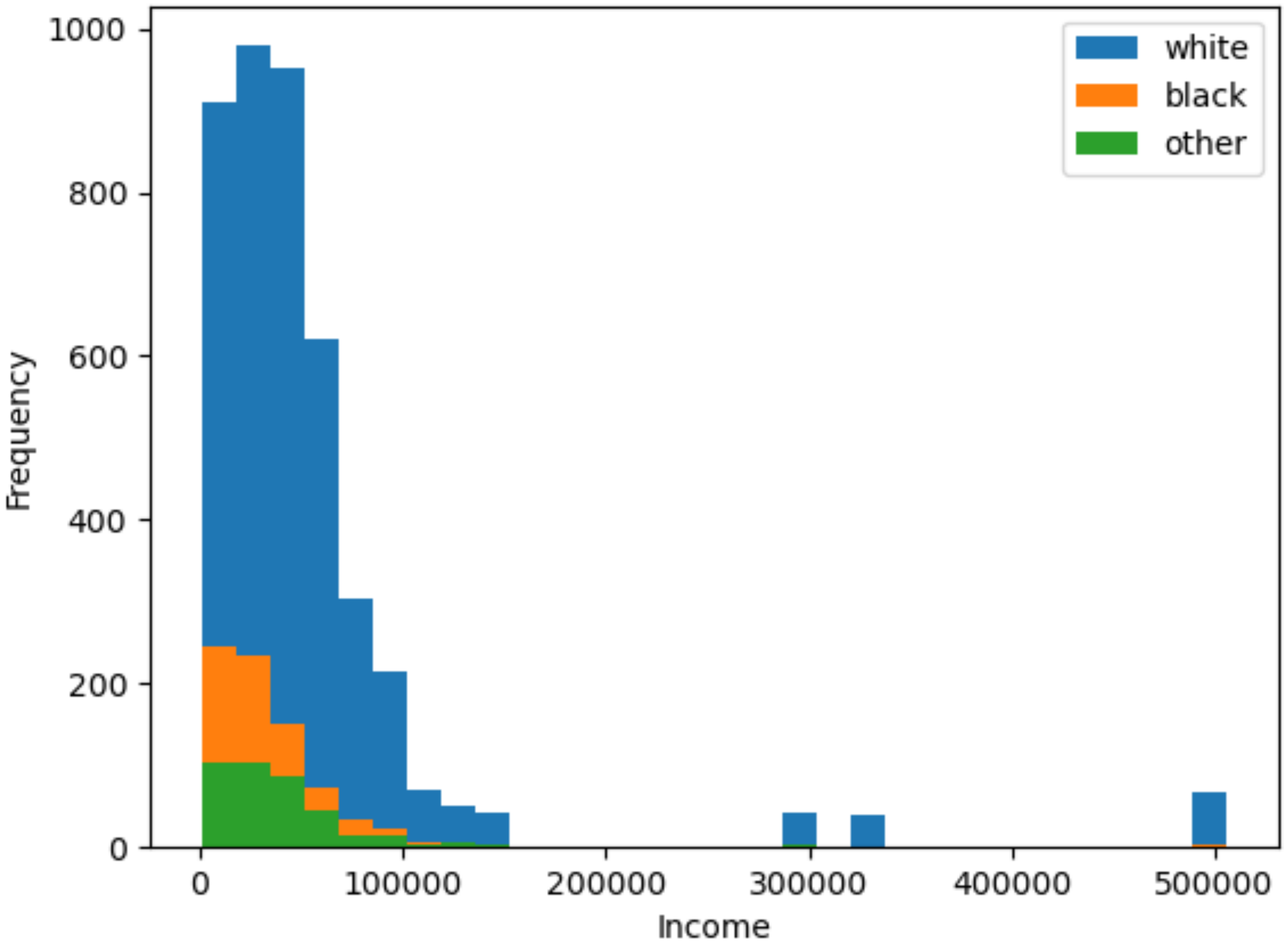
Example

```
# inflation measured via US Consumer Price Index (CPI), source:  http://www.dlt.ri.gov/lmi/pdf/cpi.pdf
>>> cpi2015_vs_1986 = 236.7 / 109.6
>>> assert df['realrinc'].astype(float).median() < 24000   # verification check
>>> df['realrinc2015'] = cpi2015_vs_1986 * df['realrinc'].astype(float)
>>> df.groupby('race')['realrinc2015'].plot(kind='hist', bins=30)
  race
  white    AxesSubplot(0.125,0.11;0.775x0.77)
  black    AxesSubplot(0.125,0.11;0.775x0.77)
  other    AxesSubplot(0.125,0.11;0.775x0.77)
  Name: realrinc2015, dtype: object
>>> plt.xlabel('Income')
  Text(0.5, 0, 'Income')
>>> plt.legend();
  <matplotlib.legend.Legend object at 0x7fa55805b9a0>
>>> plt.show()
```

Example

Annual household income in constant 2015 US dollars.

A figure to show all possible values with their frequencies

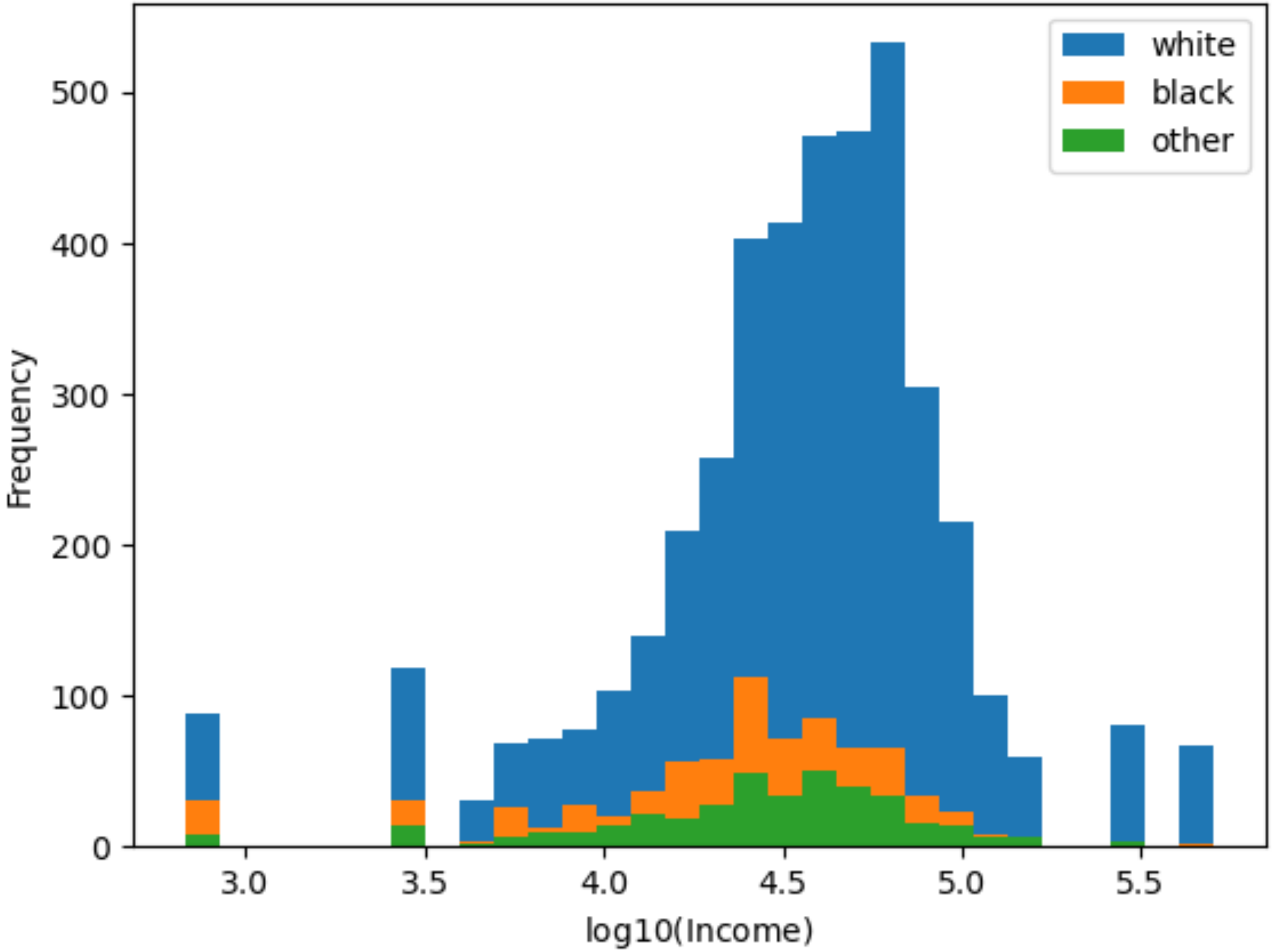


Example

```
>>> df['realrinc2015_log10'] = np.log10(df['realrinc2015'])
>>> df.groupby('race')['realrinc2015_log10'].plot(kind='hist', bins=30)
  race
  white    AxesSubplot(0.125,0.11;0.775x0.77)
  black    AxesSubplot(0.125,0.11;0.775x0.77)
  other    AxesSubplot(0.125,0.11;0.775x0.77)
  Name: realrinc2015_log10, dtype: object
>>> plt.xlabel(r'$\log10(\mathrm{Income})$')
    Text(0.5, 0, '$\\log10(\\mathrm{Income})$')
>>> plt.legend();
   <matplotlib.legend.Legend object at 0x7fa5687500d0>
>>> plt.show()
```

Example

Annual household income in constant 2015 US dollars (data converted to a logarithmic scale).



Contents

Example

Location statistics

Procedure for a statistical test

T-test

Chi-square test

Location Statistics

What can represent the typical income value?

How can we represent the variability of these values?

The (arithmetic) mean?

```
>>> print(df['realrinc2015'].mean())
   51296.74902490707
```

The ratio between the max and the min.

```
>>> print(df['realrinc2015'].max() / df['realrinc2015'].min())
   749.1342599999999
```

And now we know that the max income represents 749 times the lowest.

Median

Another location statistics.

The sample median is the [(n+1)/2]th observation when the values are sorted by values.

For n odd, the median is the value of the $median = x_{[\frac{n+1}{2}]}$

For n even, the median = all values located between $x_{[\frac{n}{2}]}\ and\ x_{[\frac{n+2}{2}]}$

Robust statistics (less sensitive to noise, extreme values).

Median

In our application:

The median:

```
>>> print(df['realrinc2015'].median())
   37160.92814781022
```

A value smaller than the mean.

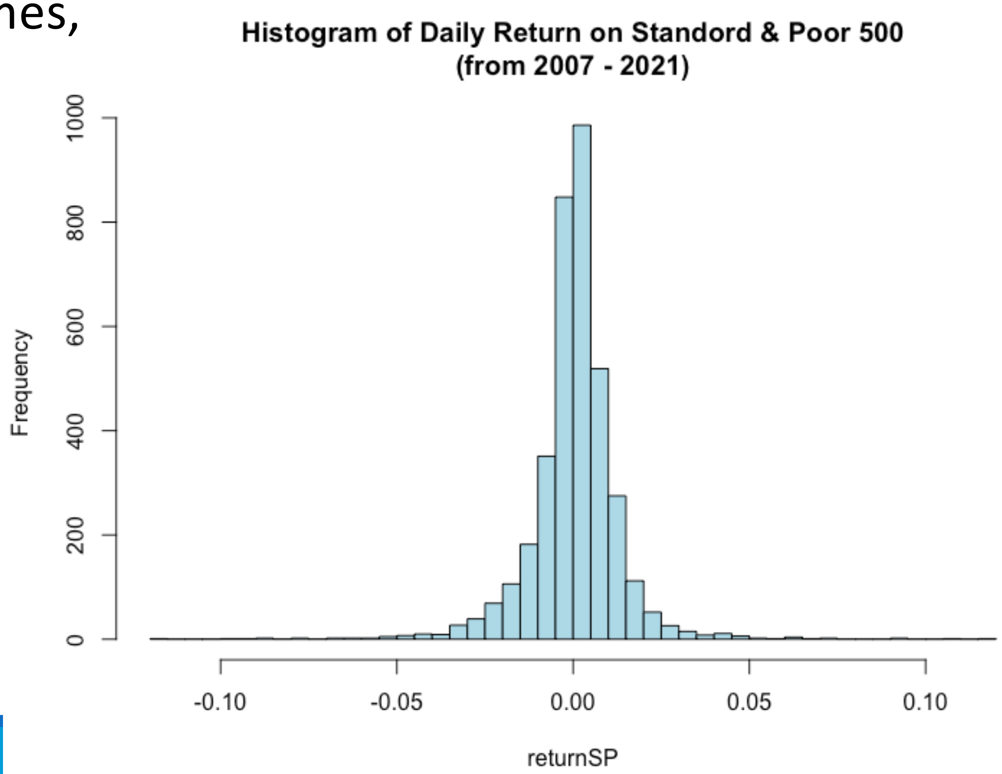
Mode

Most frequently occurring value (thus maybe more typical).

When all values occur the same number of times, we usually say that there is no unique mode.

When two values occur the same number of times and more than any other values, the distribution is said to be *bimodal*.

With continuous values, could be more problematic from time to time.



Mode

The value occurring the most often

The mode:

```
>>> print(df['realrinc2015'].mode())
   46674.821168
   dtype: float64
```

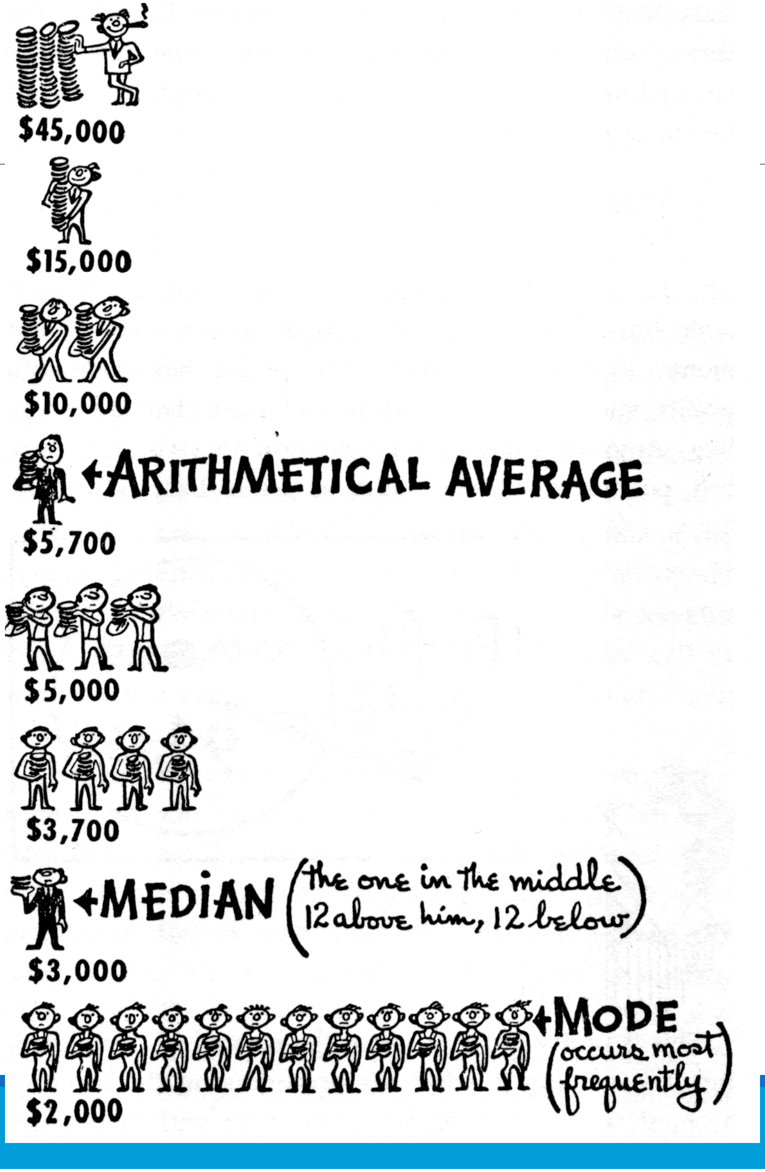
A value smaller than the mean.

Mean, Median or Mode

The difference …

but depends also of the data type.

For example, the mean for a nominal variable does not make sense!


$45,000
$15,000
$10,000
←ARITHMETICAL AVERAGE
$5,700
$5,000
$3,700
←MEDIAN (the one in the middle 12 above him, 12 below)
$3,000
←MODE (occurs most frequently)
$2,000

Mean or Median?

Other differences

```
>>> ex_true = [11159, 13392, 31620, 40919, 53856, 60809, 118484, 14361]

>>> print(np.mean(ex_true))
   43075.0

>>> print(np.median(ex_true))
   36269.5

>>> ex_corrupted = [11159, 13392, 31620, 40919, 53856, 60809,118484,1436180]

>>> print(np.mean(ex_corrupted))
   220802.375      #  effect on the mean of extreme values!

>>> print(np.median(ex_corrupted))
   47387.5  #  the median is less affected
```

For all Data Types?

```
>>> readfict_sample = df.loc[df['readfict'].notnull()].sample(8)['readfict']

>>> readfict_sample = readfict_sample.replace(['no', 'yes'], [0, 1])
>>> readfict_sample
37731    0
42612    1
37158    1
35957    1
41602    1
42544    1
35858    0
36985    1
Name: readfict, dtype: int64

>>> print("Mean:", readfict_sample.mean())
Mean: 0.75

>>> print("Median:", readfict_sample.median())
Median: 1.0
```

No sense!

Data Types

1. Numerical quantities (e.g., income, age, etc.).

All mathematical operators. The mean makes sense.

2. Ordinal quantities (e.g., small, medium, large, huge).

Impose order on values, *equal, >, <*

The mean makes no sense. Present the proportion of each category.

3. Nominal quantities (categorical, enumerated, "discrete") (e.g., red, green, blue)

Only *equality* tests can be performed.

The mean makes no sense. Present the proportion of each category.

Other Means

Instead of using the arithmetic mean with all observations, we may assume that each sample contains errors (mainly manual errors). We can thus reduce the sample size of 10% by removing the 5% lowest and 5% highest values.

In such case, you select the trimmed mean.

This is a robust statistics (like the median).

Example

We have the following game. You have a fortune of 100.

With a probability of 50%, you increase your estate of 30%.
With a probability of 50%, you decrease your estate of 25%.

Do you play this game?

Expectation is: $0.5 \cdot 30 - 0.5 \cdot 25 = 2.5$

The first two plays…

Play 1	Play 2
130	169
130	97.5
75	97.5
75	56.25

Example

But in long term… 200 plays with 100 players… (with Excel)

Statistics	
Mean	1546.1
Median	4.6
# ≥ 100	25
# < 100	75
Max	52792.8
Min	0.0

In this game, in long term we observe more losers than winners.

But the mean is larger than 100.

Explain …

Can be discover some reasons why an income could be higher than another?

Can be observe a smaller/larger variability of the income according to …

```
 >>> df_bachelor = df[df['degree'] == 'bachelor']
```

When observed=True instructs pandas to ignore categories without any observations

```
 >>> df_bachelor.groupby(['year', 'degree'],
observed=True)['realrinc2015'].agg(['size', 'mean', 'median'])

                   size          mean          median

 year degree

 1998 bachelor   363    63805.508302    48359.364964
 2000 bachelor   344    58819.407571    46674.821168
 2002 bachelor   307    85469.227956    50673.992929
```

Sample Variance

It is important to have an idea of the underlying variability of an observed phenomenon.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

$$s^2 = \frac{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot (n-1)}$$

$$s^2 = \frac{\sum_{i=1}^{n} x_i^2 - n \cdot \bar{x}^2}{(n-1)}$$

But s^2 is usually not "directly" visible.

Sample Standard Deviation

With the standard deviation, we have the *same unit* than the observed data.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

But two possible formulas. The first one is an unbiased estimate of the true standard deviation.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

Sample Standard Deviation

But with large sample, the difference is very small.

```
>>> print(f"statistics.stdev: {statistics.stdev(sim):.1f}\n"
...       f"         sim.std: {sim.std():.1f}\n"
...       f"          np.std: {np.std(sim):.1f}\n"
...       f"  np.std(ddof=1): {np.std(sim, ddof=1):.1f}")
statistics.stdev: 23239.9
         sim.std: 23239.9
          np.std: 23237.8
  np.std(ddof=1): 23239.9
```

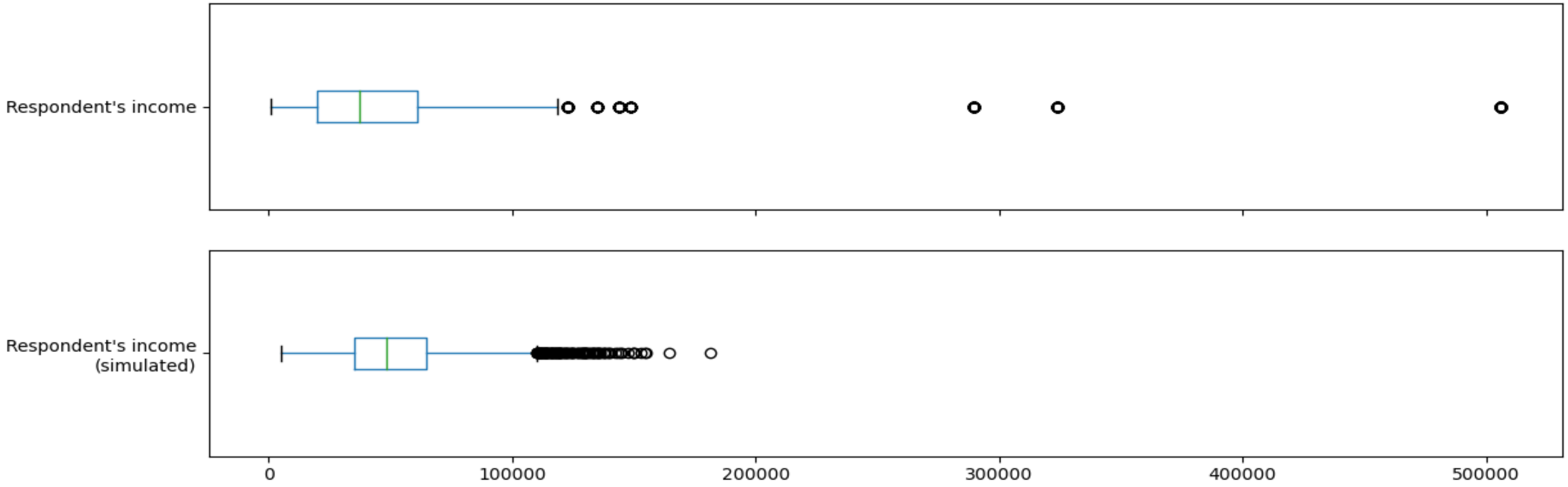
Compute with NumPy

All in One Figure: Box Plot

```
>>> fig, axes = plt.subplots(2, sharex=True)
>>> df['realrinc2015'].plot(kind='box', vert=False, ax=axes[0])
     <AxesSubplot:>
>>> axes[0].set_yticklabels(["          Respondent's income"])
     [Text(0, 1, "          Respondent's income")]
>>> sim.plot(kind='box', vert=False, ax=axes[1])
     <AxesSubplot:>
>>> axes[1].set_yticklabels(["          Respondent's
income\n(simulated)"]);
     [Text(0, 1, "          Respondent's income\n(simulated)")]
>>> plt.show()
```

Box Plot: All in One

Box plots of observed and simulated values for household income in constant 2015 US $. The interquartile is provided by the box. 25% of the values are lower than the lowest limit while 75% of them are lower than the upper limit of the box.



Explanations

Can we find factor(s) to explain the income differences?

And the variability among people having the same level of education

To measure the variability, one can apply the standard deviation or the `mad()` function corresponding to the interquartile value (IQR):

$$IQR = \frac{1}{n} \cdot \sum_{i=1}^{n} |x_i - \bar{x}|$$

Variability

```
>>> df.groupby('degree')['realrinc2015'].mad().round()
 degree                                    # mad() = mean absolute deviation
 lt high school     19551.0    # Smallest variability for the first category
 high school        23568.0
 junior college     33776.0
 bachelor           45055.0
 graduate           77014.0     # Largest variability for the last category
 Name: realrinc2015, dtype: float64

>>> df_bachelor_or_more = df[df['degree'].isin(['bachelor', 'graduate'])]
```

Variability

```
 >>> df_bachelor_or_more.groupby(['degree', 'readfict'], observed=True)
['realrinc2015'].mean().round()

degree     readfict
bachelor   yes             71251.0
           no             139918.0    # don't read fiction, larger income
graduate   yes            113125.0
           no             153961.0
Name: realrinc2015, dtype: float64
```

Variations in Categorical Values

From different groups (to test some functions)

```
>>> group1 = ['high school', 'high school', 'high school', 'high school',
              'high school', 'high school', 'bachelor', 'bachelor']

>>> group2 = ['lt high school', 'lt high school', 'lt high school',
'lt high school', 'high school', 'junior college', 'bachelor', 'graduate']

>>> group3 = ['lt high school', 'lt high school', 'high school',
'high school', 'junior college', 'junior college', 'bachelor', 'graduate']
```

Variations in Categorical Values

Calculate the number of unique values in each group

```
>>> print([len(set(group)) for group in [group1, group2, group3]])
    [2, 5, 5]
```

Calculate the ratio of observed categories to total observations

```
>>> print([len(set(group)) / len(group) for group in [group1, group2, group3]])
     [0.25, 0.625, 0.625]
```

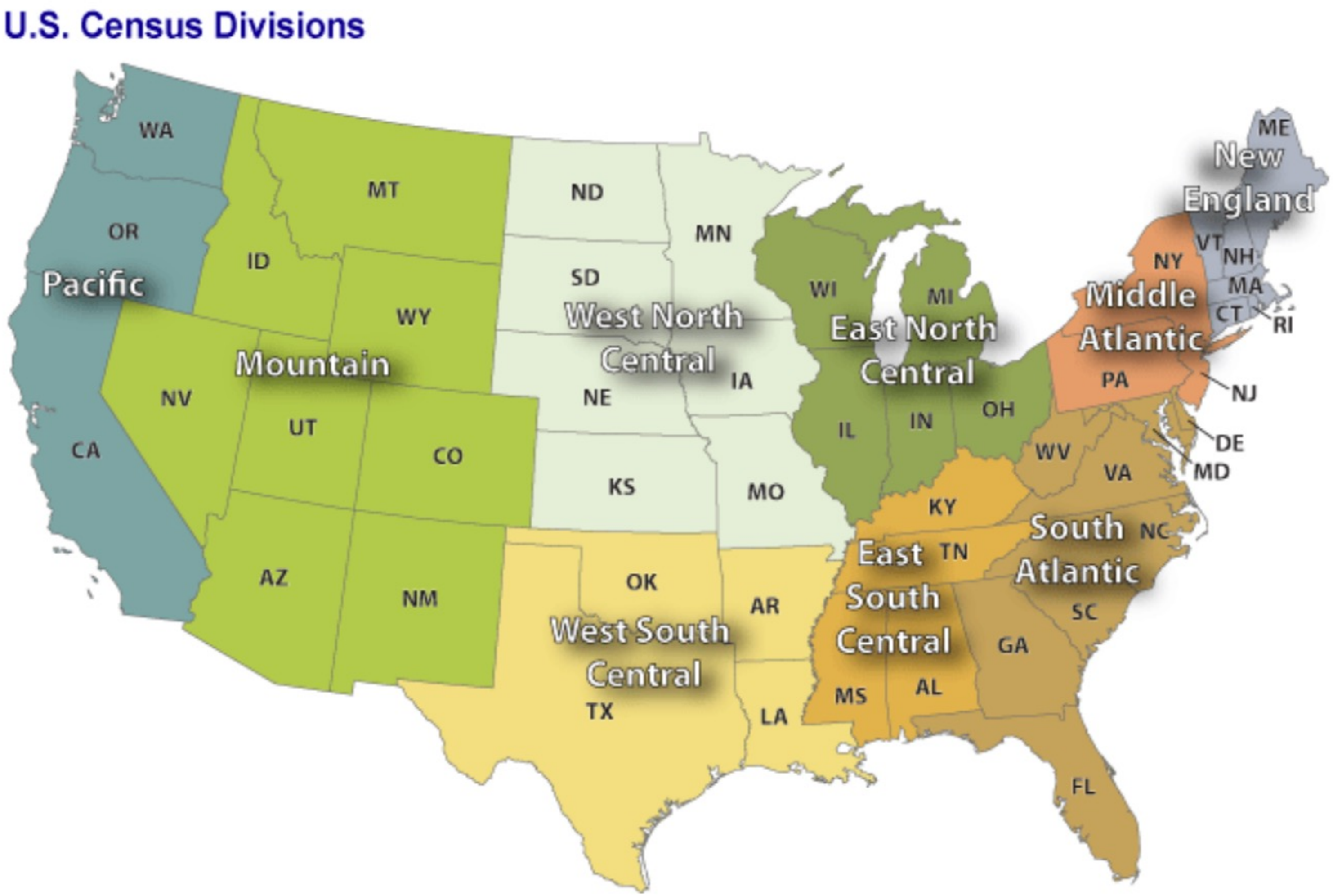
Variations in Categorical Values

The regions inside the US

```
>>> regions_oi = sorted(['pacific', 'e. sou. central', 'new england'])

>>> df_regions = df.loc[df['reg16'].isin(regions_oi)].copy()

>>> df_regions['reg16'] = df_regions['reg16'].cat.remove_unused_categories()

>>> df_regions.groupby('reg16')['degree'].value_counts(normalize=True).
round(1).to_frame()
```

Variations in Categorical Values

The regions inside the US according to the Census Bureau.



Variations in Categorical Values

```
                               degree
reg16           degree
new england     high school        0.5
                bachelor           0.3
                graduate           0.1
                junior college     0.1
                lt high school     0.1
e. sou. central high school        0.6
                lt high school     0.1
                bachelor           0.1
                junior college     0.1
                graduate           0.1
pacific         high school        0.5
                bachelor           0.2
                junior college     0.1
                graduate           0.1
                lt high school     0.1
```

Variations in Categorical Values

Compute the entropy as a way to measure the predictability of a given category (according to the distribution inside each possible value).

Information theory

Measure information in *bit*

1 bit reduces by 50% the uncertainty

0 bit = non uncertainty

Given a distribution, the info required to predict an event is its *entropy.*

Entropy gives the information required in bits (can involve fractions of bits!)

How to compute the entropy (predictability, uncertainty, impurity)?

Variations in Categorical Values

Compute the entropy as a way to measure the predictability of a given category (according to the distribution inside each possible value).

For a category having *k* possible values, the entropy is:

$$H = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i$$

The larger the most uncertain (unpredictable) is the distribution inside the *k* classes.

a.k.o. weighted mean

Example with two outcomes:

◦ Entropy(0.5, 0.5) = - 0.5·log$_2$(0.5) - 0.5·log$_2$(0.5) = - 0.5·(-1) - 0.5·(-1) = 1

◦ Entropy(0.8, 0.2) = - 0.8·log$_2$(0.8) - 0.2·log$_2$(0.2) = 0.72

Variations in Categorical Values

$$H = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i$$

The larger the most uncertain is the distribution of the observations inside the *k* classes.

Min: All observations have the same value. H = 0.0 why?
Distribution: "good": 10 cases, "Bad": 0 cases

Max: All possible outcomes have the same number of observations. H = $\log_2(1/k)$

Example: $k = 2$ Max: $-\log_2(1/2) = 1$
$k = 8$ Max: $-\log_2(1/8) = 3$

Variations in Categorical Values

```
>>> for n, group in enumerate([group1, group2, group3], 1):
...     degree_counts = list(collections.Counter(group).values())
...     H = scipy.stats.entropy(degree_counts)
...     print(f'Group {n} entropy: {H:.1f}')
```

```
Group 1 entropy: 0.6   # more concentrate distribution

Group 2 entropy: 1.4

Group 3 entropy: 1.6
```

Variations in Regions

```
>>>df.groupby('reg16')['degree'].apply
                (lambda x: scipy.stats.entropy(x.value_counts()))
```

```
reg16
foreign              1.505782
new england          1.345351
middle atlantic      1.321904
e. nor. central      1.246287
w. nor. central      1.211067
south atlantic       1.261397
e. sou. central      1.196932
w. sou. central      1.290568
mountain             1.214591
pacific              1.283073
Name: degree, dtype: float64
```

Correlation

Compute the Kendall coefficient (τ) of correlation (-1 ≤ τ ≤ 1)

$$\tau = \frac{(number\ of\ concordant\ pairs) - (number\ of\ discordant\ pairs)}{\binom{n}{2}}$$

```
>>> df_subset[['age', 'realrinc2015_log10']].corr('kendall')
                         age  realrinc2015_log10
age                 1.000000            0.204298
realrinc2015_log10  0.204298            1.000000
>>> df_subset = df.loc[df['readfict'].notnull(), ['reg16', 'readfict']]
```

Contents

Example

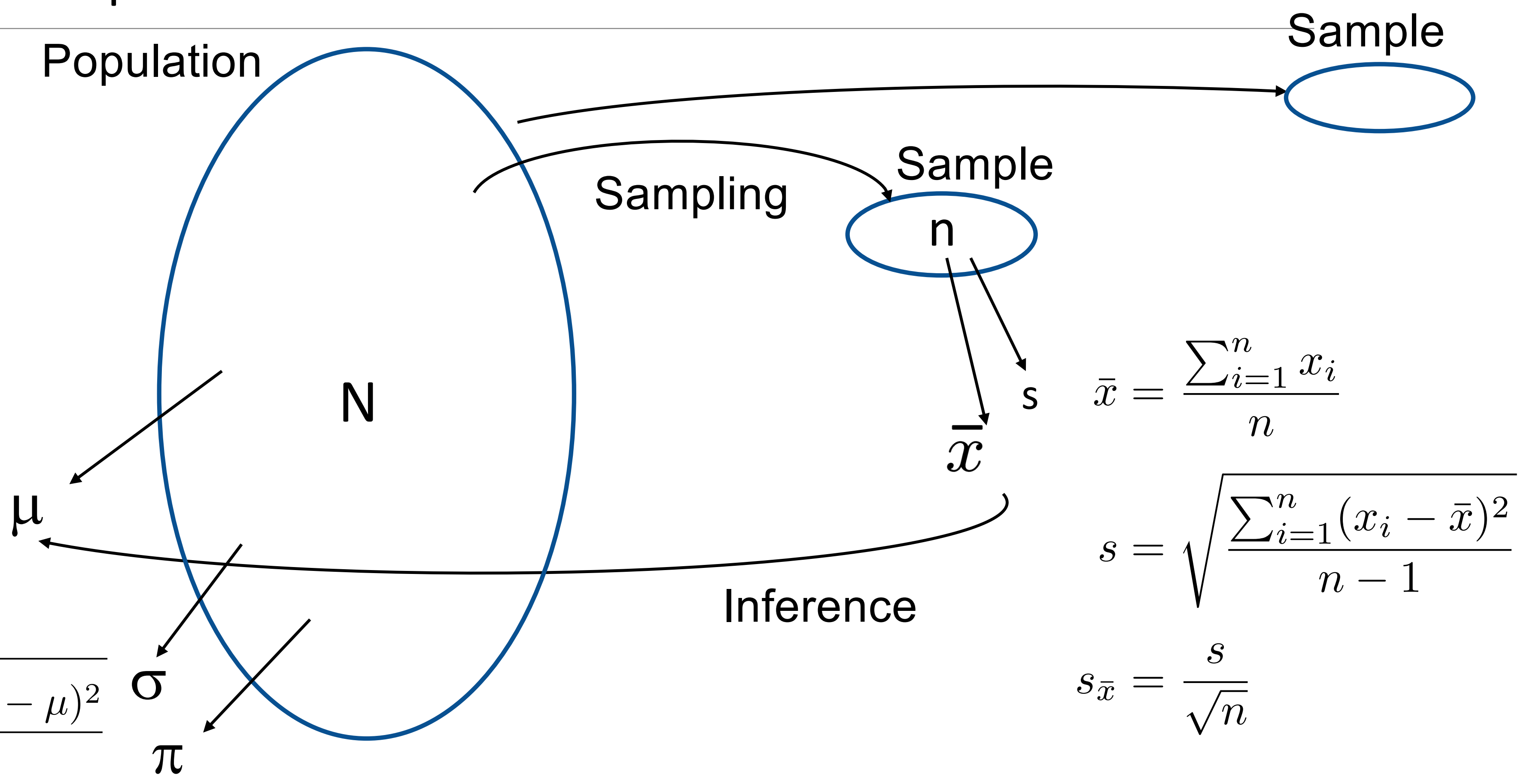
Location statistics

Procedure for a statistical test

T-test

Chi-square test

The Principle



$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}}$$

$$s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

Principle

We can propose an estimator for each unknown parameter of the population

For example the mean $\bar{x}$

but this estimator is a random variable

with a (known) standard deviation

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} = \frac{\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}}{\sqrt{n}}$$

Tyranny of the mean!

Because we have a clean formula for its standard deviation and simple statistical tests!

Test for the Mean

We can apply the *t*-test for the mean.

We will present two *t*-tests (with similar procedures).

With the first, we want to verify if the mean of the population (denoted μ) could be equal to a given value (usually denoted μ_0) (which is a fixed value, not a random one!).

Second, we will compare two means, each coming from a given sample and we want to verify if these two means could be equal (if it is possible that $\mu_0 = \mu_1$).

A similar procedure can be applied to verify if $\mu_0 > \mu_1$.

Establishment of Test Procedures

With a statistical test we want to verify aa (statistical) hypothesis.

A statistical hypothesis is a statement about a *statistical* population and usually is a statement about the values of one or more parameters of the population (μ, σ^2, π, ...)

Statistical tests separate *significant effects* from mere luck or *random chance*.

The same general idea is valid for all statistical tests.

Establishment of Test Procedures

Every statistical test specifies a null hypothesis H_0 against an alternative hypothesis denoted H_1 (only two possibilities).

The denomination “*null*” means usually “*nothing*”
(no difference or no effect). Usually we want to reject this null hypothesis.

The alternative hypothesis presumes the presence of a difference or an effect.

If you reject the null hypothesis, you must accept H_1. The world is represented by onyl these two disjoint hypotheses.

Establishment of Test Procedures

1. A null hypothesis, usually denoted by H_0.

2. An alternative hypothesis, usually denoted by H_1.

$$\begin{aligned} H_0 &: \mu = \mu_0 \\ H_1 &: \mu \neq \mu_0 \qquad (two - sided) \end{aligned}$$

In the alternative hypothesis, we consider *two* cases, μ greater or smaller than μ_0.

The world (mean of the population) is either equal to μ_0 or different.

Recall: a Greek letter = a constant

Establishment of Test Procedures

Another view of the world:

1. A null hypothesis, usually denoted by H_0.

2. An alternative hypothesis, usually denoted by H_1.

H_0: $\mu_1 = \mu_2$

H_1: $\mu_1 > \mu_2$

We do not consider $\mu_1 < \mu_2$ because the theory (practice / statement) specifies that is impossible in our context. We'll consider the possibility that $\mu_1 > \mu_2$.

Adding this information, the statistical test will be more effective.

Establishment of Test Procedures

All hypothesis tests have unavoidable - but quantifiable - risks of making the wrong conclusion.

In testing hypotheses, there are two types of errors which can be made:

Type I error

Type II error

Establishment of Test Procedures

Example: “This morning, if I think it’s going to rain, I take an umbrella. If not, I do not”.

At the end of the day, we make an evaluation of the decision (two possible types of error):

Decision	Hypothesis true	
	rainy day	sunny day
take an umbrella	good decision	error
do not take an umbrella	error	good decision

After the decision has been taken, I can encounter only a single error.

Establishment of Test Procedures

Type I error

The rejection of the hypothesis H_0 when it is true.

Practically, the type I error can be interpreted as the probability of deciding that a significant effect is present (reject H_0) when it isn't (H_0 is true).

Why?

The sample tends to demonstrate a significant effect but it is due to random variability. The sampling provides an extreme (but still possible) sample.

Establishment of Test Procedures

Type II error

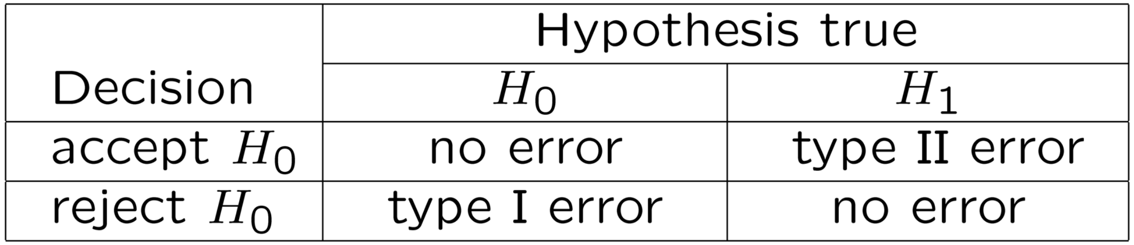
The acceptance of the hypothesis H_0 which is false (and so H_1 true).

Practically, the type II error can be interpreted as the probability of not detecting a significant effect (accept H_0) when one exists (H_0 is false).

Why?

The true effect (H_1) is too close to the H_0 effect.
The effect is too small to be detected.
The sample size is small to detect the difference.

Establishment of Test Procedures



	Hypothesis true	
Decision	H_0	H_1
accept H_0	no error	type II error
reject H_0	type I error	no error

When setting up an experiment to test a hypothesis it is desirable to minimize the probabilities of making these errors.

Establishment of Test Procedures

- The probability of making a type I error is denoted by α.
- The probability of making a type II error is denoted by β.

	Hypothesis true	
Decision	H_0	H_1
accept H_0	1-α	β
reject H_0	α	$1 - \beta$

It should also be noted that α (in percentage) is commonly referred to as the significance level (e.g. α = 5% or α = 1%).

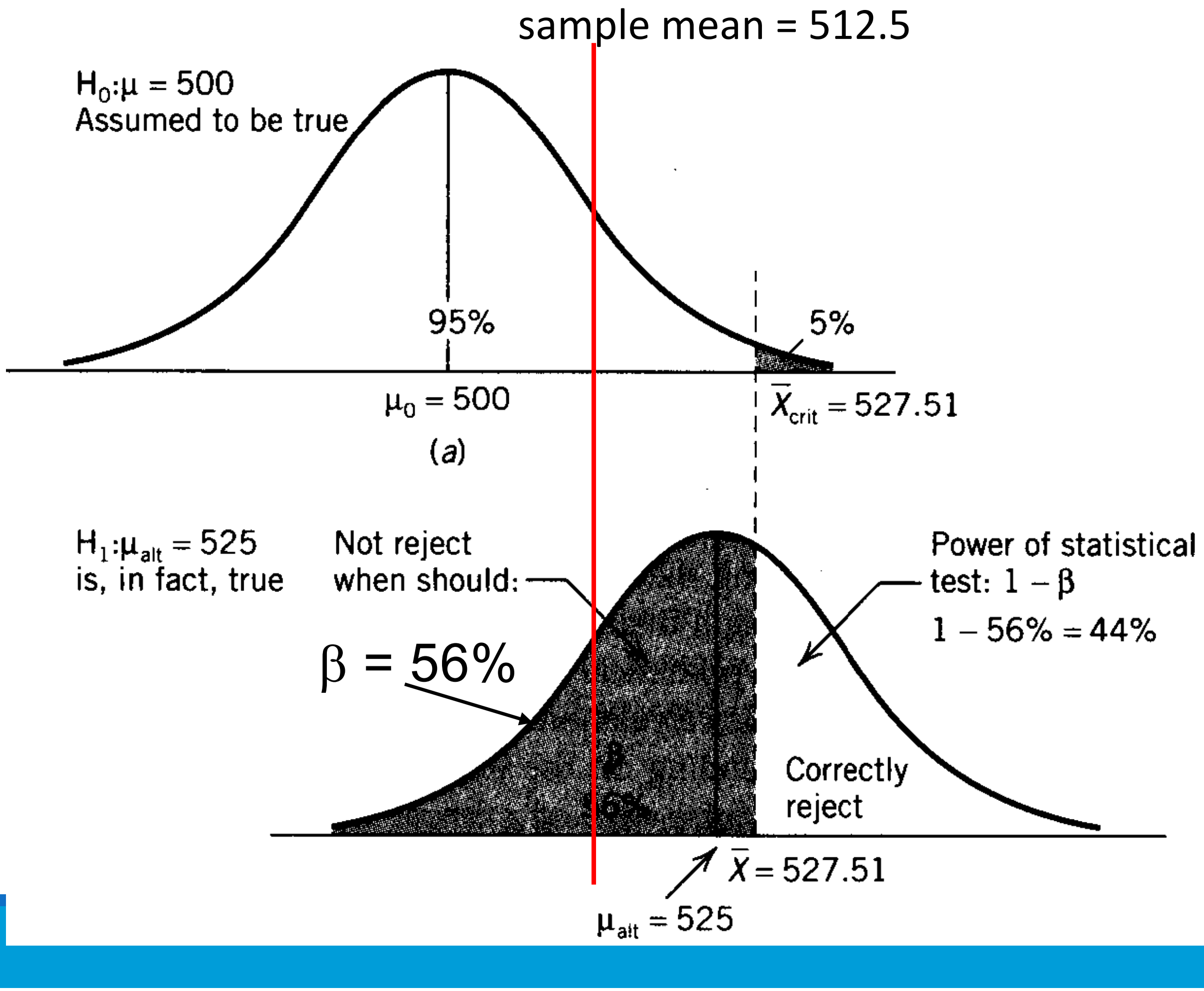
An important consideration in discussing the probabilities of type II errors is the "*degree of falseness*" of a false hypothesis.

Establishment of Test Procedures

We want to verify if the mean of the population is equal 500. H_0: μ = 500

Selecting a sample, we can obtain different estimation of this mean. $\overline{x}$ = 512.5

The real population μ could be equal to 525.



Establishment of Test Procedures

Incidentally, the rejection region is frequently referred to as the critical region.

Using the notation, it is seen that:

α = Prob [reject H_0 | H_0 is true]

β = Prob [accept H_0 | H_1 is true]

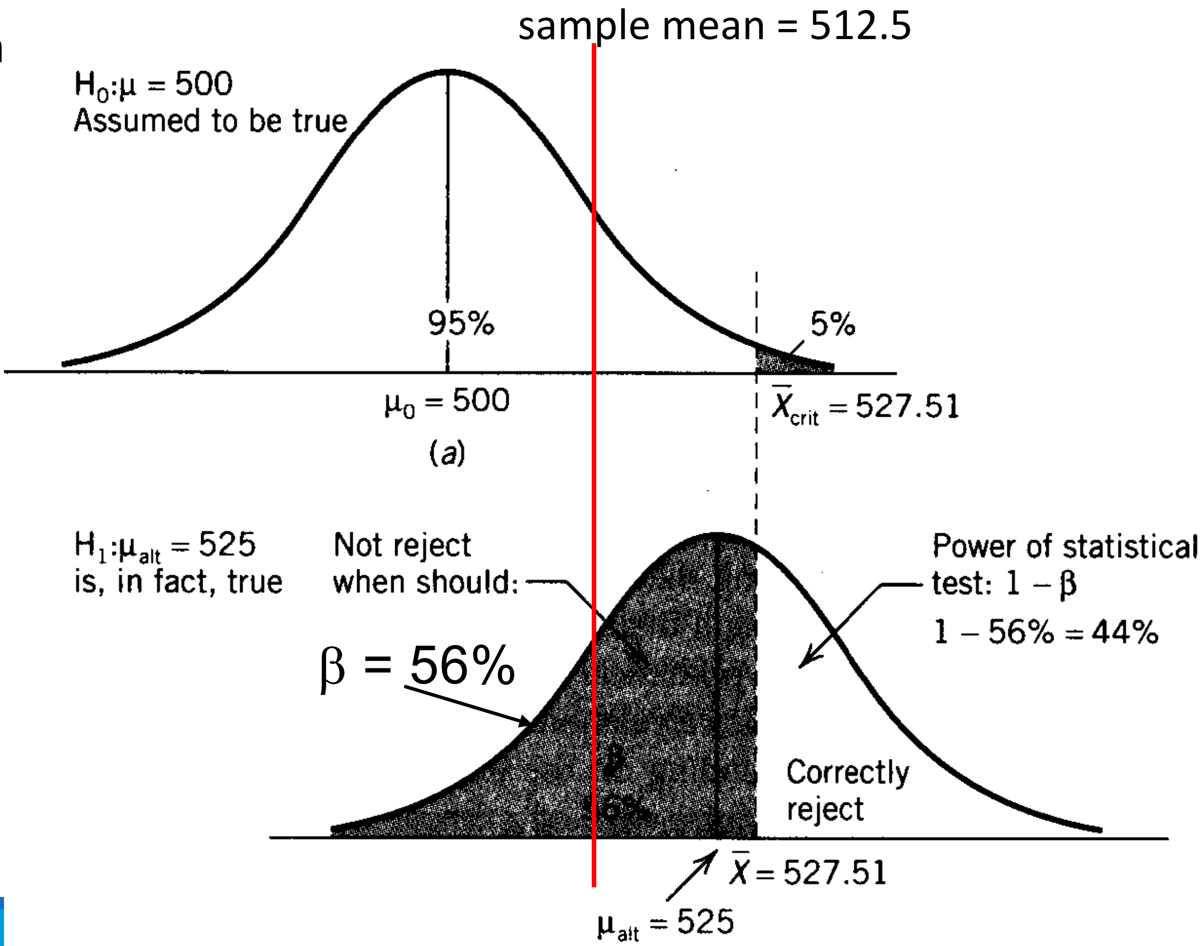
1 - β = power of the test

Establishment of Test Procedures

We want to verify if the mean of the population is equal 500. Thus H_0: μ = 500

Selecting a sample we can obtain different estimation of this mean. $\bar{x}$ = 512.5

The real population μ could be equal to 525.



Establishment of Test Procedures

Size of the sample

It is possible to *reduce* the probability of type I error and yet to control the probability of a type II error by using *large enough samples*.

The variance of the distribution of a sample mean is σ^2/n so that it decreases as the sample size increases.

Establishment of Test Procedures

Test procedure

Step 1
Formulation of the null and alternative hypotheses. Use the theory to formulate these hypothesis.

Step 2
Decide on the *significance level* α to be use in conducting the test. It is the probability of type I error we are willing to accept. The level 0.05 is conventionally used (other choice 0.01).

Establishment of Test Procedures

Step 3

Select the probability distribution.

Step 4

Calculate a test statistic

Step 5

Testing the hypothesis and take a decision.

Contents

Example

Location statistics

Procedure for a statistical test

T-test

Chi-square test

t-test

Verify the quality of the random number generator.

We generate 100 random values from 1 to 6 (dice).

```
>>> import pandas as pd
>>> import numpy as np
>>> from scipy import stats
```

# fix a seed for being able to reproduce the same random sampling

```
 >>> np.random.seed(123759)
>>> valuesA = np.random.randint(0, 6, size=100)
```

# sample of values from 1 to 6, each with the same probabilty
# the mean must be 3.5

```
>>> valuesA[0:5]
    array([0, 3, 3, 2, 2])
```

t-test

The different steps

Step 1: H_0 $\mu = 3.5$ vs. H_1 $\mu \neq 3.5$

Step 2: $\alpha = 1\%$ or 0.01

Step 3: the sample mean $\overline{x}$ follows a Normal distribution
assuming i.i.d (independent and identical distributed)

Step 4: the computer will compute a statistic t_c (a Student distribution in this case).
This statistic follow a Student distribution with *n*-1 degree of freedom.

Step 5: Testing the hypothesis; compare the *p*-value with α.

t-test

Does the generated values correspond to a population with a mean of 3.5?

# The sample mean is:

```
>>> print('mean:', np.mean(valuesA),'range:', min(valuesA),':', max(valuesA))
   mean: 2.9 range: 0 : 5
```

# With the sample, we apply the *t*-test with the population mean = 3.5 (H_0)

```
>>> result = stats.ttest_1samp(valuesA, 3.5)
```

```
>>> result.statistic
  -3.51172
```

the computed value of the statistic t_c

```
>>> result.pvalue
   0.00067
```

the resulting *p*-value

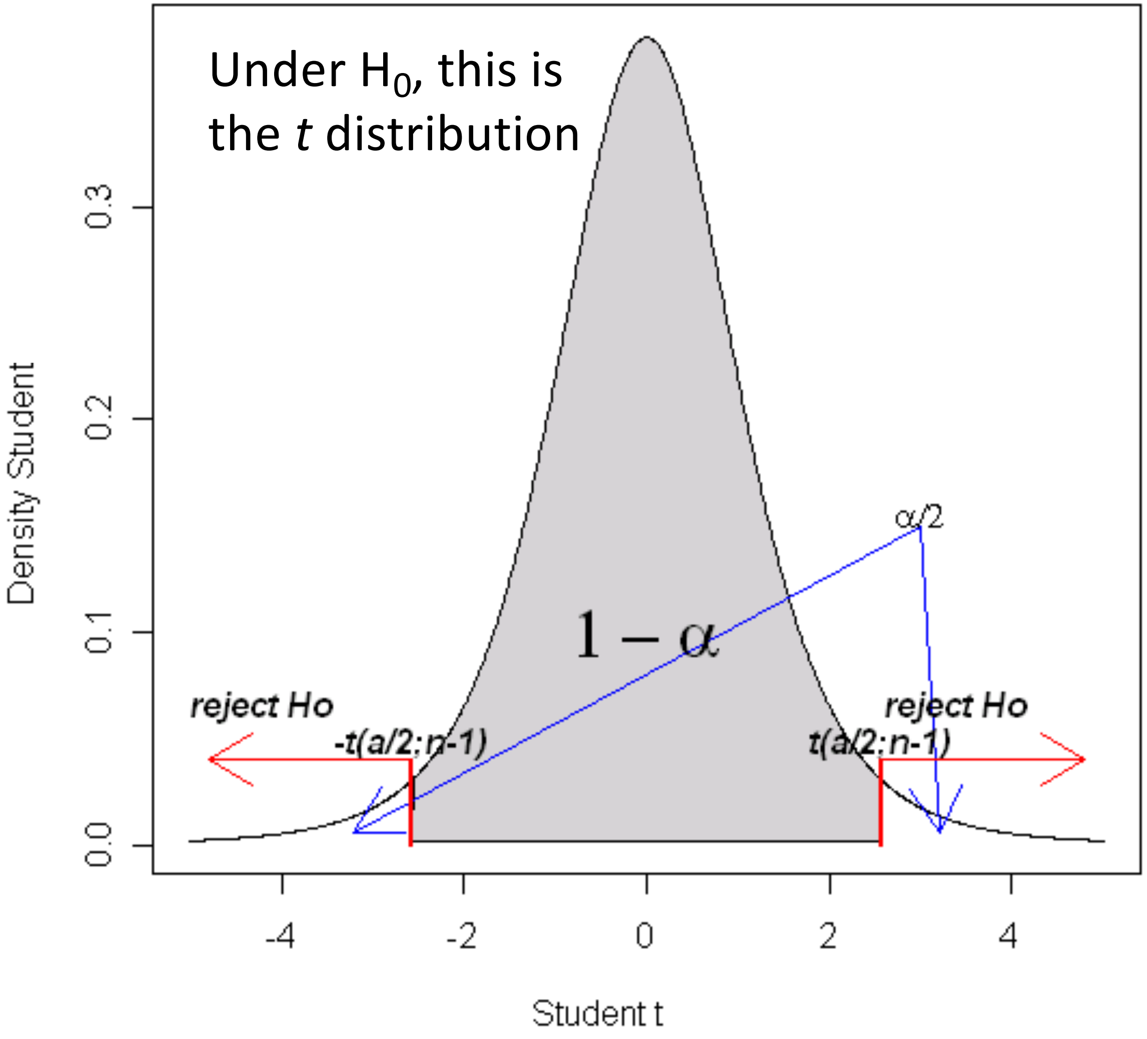
t-test

Two-sided test

H_0: $\mu = \mu_0$

H_1: $\mu \neq \mu_0$

Reject H_0 if the statistic t_c is either very large or very small.



Under H_0, this is the *t* distribution

t-test

Two-sided test

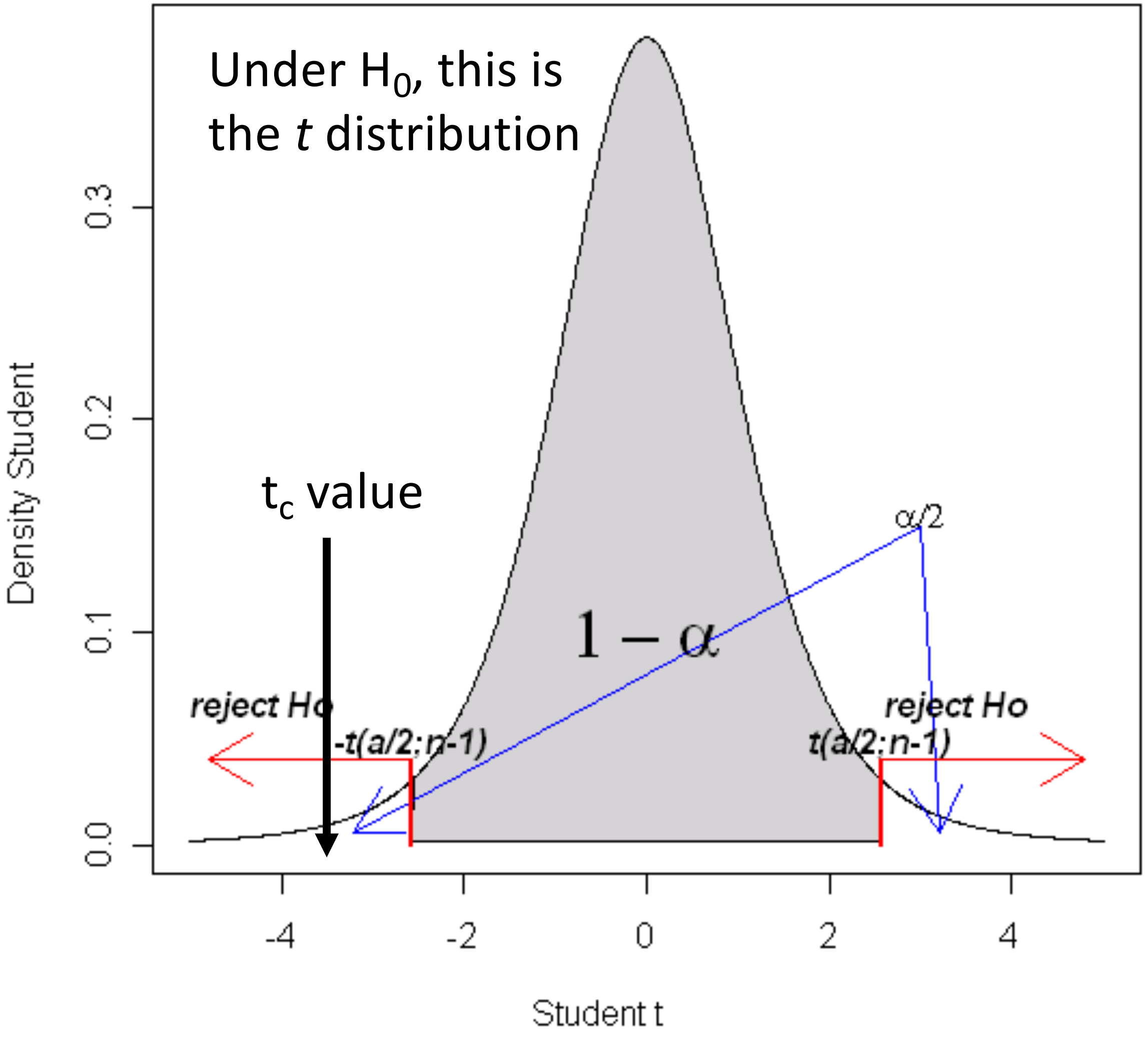
H_0: $\mu = \mu_0$

H_1: $\mu \neq \mu_0$

The statistic $t_c = -3.51$ (rather small)

The values in the sample do not support H_0 that the mean = 3.5.

p-value = 0.00067



Under H_0, this is the *t* distribution

t_c value

t-test

The *p*-value indicates the probability that having the values included in our sample, the hypothesis H_0 is true.

Therefore, a large *p*-value indicates that the sample supports clearly the null hypothesis.

A small p-value indicates that the observation (the sample) does not fully support H_0 is true.

But H_0 can *still* be true and we have the values observed in the sample… but this is supported with a small probability.

If the *p*-value < α, we reject H_0 (and therefore accept H_1).

Otherwise, we accept H_0 (not effect).

t-test

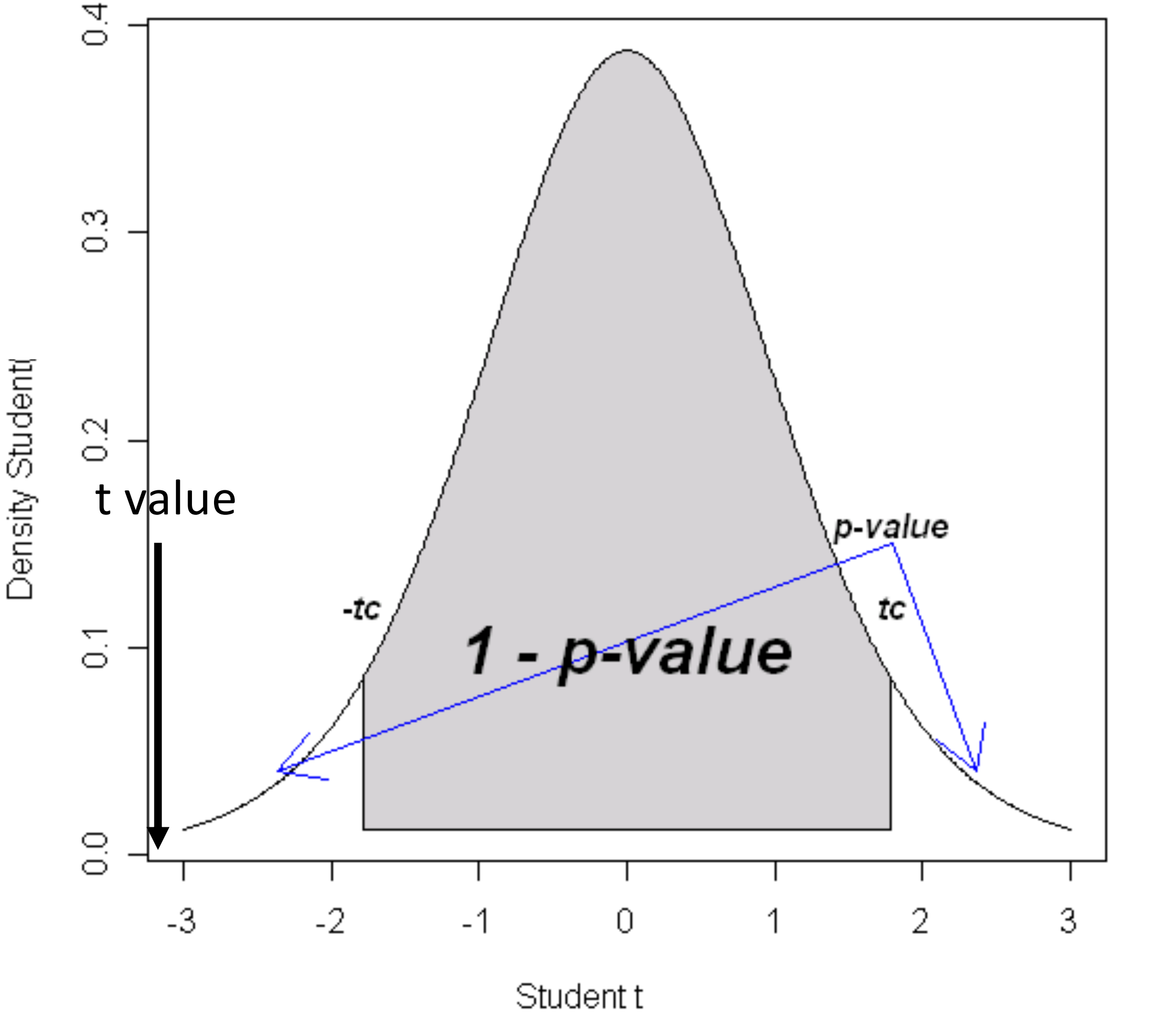
Two-sided test

H_0: $\mu = \mu_0$

H_1: $\mu \neq \mu_0$

The statistic $t_c = -3.51$

and the *p*-value = 0.00067



t-test

Generating a second sample

```
>>> valuesB = np.random.randint(0, 7, size=100)

>>> valuesB[0:10]
     array([1, 4, 3, 4, 4, 2, 5, 1, 6, 1])

>>> print('mean:',np.mean(valuesB),'range:',min(valuesB),':',max(valuesB))
    mean: 2.83 range: 0 : 6

>>> result = stats.ttest_1samp(valuesB, 3.5)

>>> result.statistic    # Is it possible that the population mean is 3.5?
    -3.7144

>>> result.pvalue       # the p-value is small and < 1%
    0.000337
```

t-test

Comparing the mean of the two distinct samples

```
>>> result = stats.ttest_ind(valuesA, valuesB, equal_var=True)

>>> result.statistic     # Is it possible that the population mean is 3.5?
    0.2817

>>> result.pvalue        # the p-value is larger than 1%
    0.7784
```

t-test: French Plays

With our French plays, three text genres (Comédie, Tragédie, and Tragi-Comédie)

Can we detect a significant length difference between a tragedy and a comedy?

We have more comedies than tragedies (a comedy is more popular, and easier to sell?)

```
>>> print (len(sizeComedie), len(sizeTragedie), len(sizeTragiComedie))
   310 150 38

>>> print('mean:', np.mean(sizeComedie), 'range:', min(sizeComedie), ':',
max(sizeComedie))
   mean: 9934.91 range: 944 : 28634

>>> print('mean:', np.mean(sizeTragedie), 'range:', min(sizeTragedie), ':',
max(sizeTragedie))
   mean: 14272.513 range: 1372 : 30065
```

The mean length is clearly different. But significantly?

t-test: French Plays

Two samples (of different size) for the two genres.

```
>>> result = stats.ttest_ind(sizeComedie, sizeTragedie, equal_var=True)

>>> result.statistic
   -9.253022799799291      # rather small  $t_c$ value

>>> result.pvalue
   8.385918237102514e-19  # p-value = 0!   Significantly different.
```

The mean length is clearly different. But significantly?

```
>>> result = stats.ttest_ind(sizeComedie, sizeTragedie, equal_var=False)

>>> result.statistic
    -10.670496974984113    # and a smaller $t_c$ value

>>> result.pvalue              # p-value = 0!   Significantly different.
    1.105984213125518e-23
```

Contents

Example

Location statistics

Procedure for a statistical test

T-test

Chi-square test (or χ^2 test)

Chi-square test

Back with our example with reading in the US

Is the distribution over the regions statistically similar?

```
>>> regions_oi = sorted(['new england', 'mountain', 'pacific', 'foreign'])
>>> df_regions = df.loc[df['reg16'].isin(regions_oi)].copy()
>>> df_regions['reg16'] = df_regions['reg16'].cat.remove_unused_categories()
>>> df_regions.groupby('reg16')['degree'].value_counts().to_frame()
reg16       degree
foreign     high school          134        mountain    high school         152
            bachelor              80                    bachelor             69
            graduate              63                    graduate             20
            lt high school        57                    lt high school       19
            junior college        32                    junior college       16
new england high school          120        pacific     high school         318
            bachelor              72                    bachelor            112
            graduate              34                    junior college       63
            junior college        19                    graduate             48
            lt high school        17                    lt high school       42
```

Chi-square test

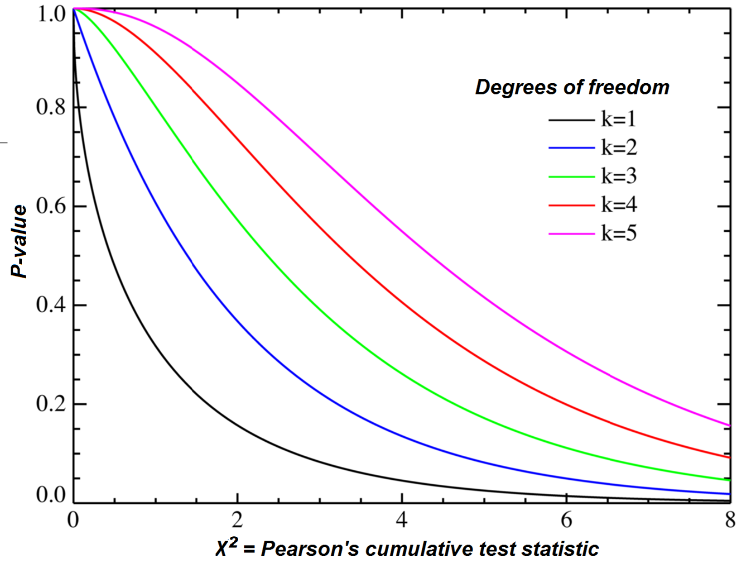
```
>>> chi, pval, dof, exp = stats.chi2_contingency(subjects)

>>> print('p-value is: ', pval)
     p-value is:  3.2772644250479287e-08

>>> significance = 0.05

>>> p = 1 - significance

>>> critical_value = stats.chi2.ppf(p, dof)

>>> print('chi=%.6f, critical value=%.6f\n' % (chi, critical_value))

     chi=59.110530, critical value=21.026070
```

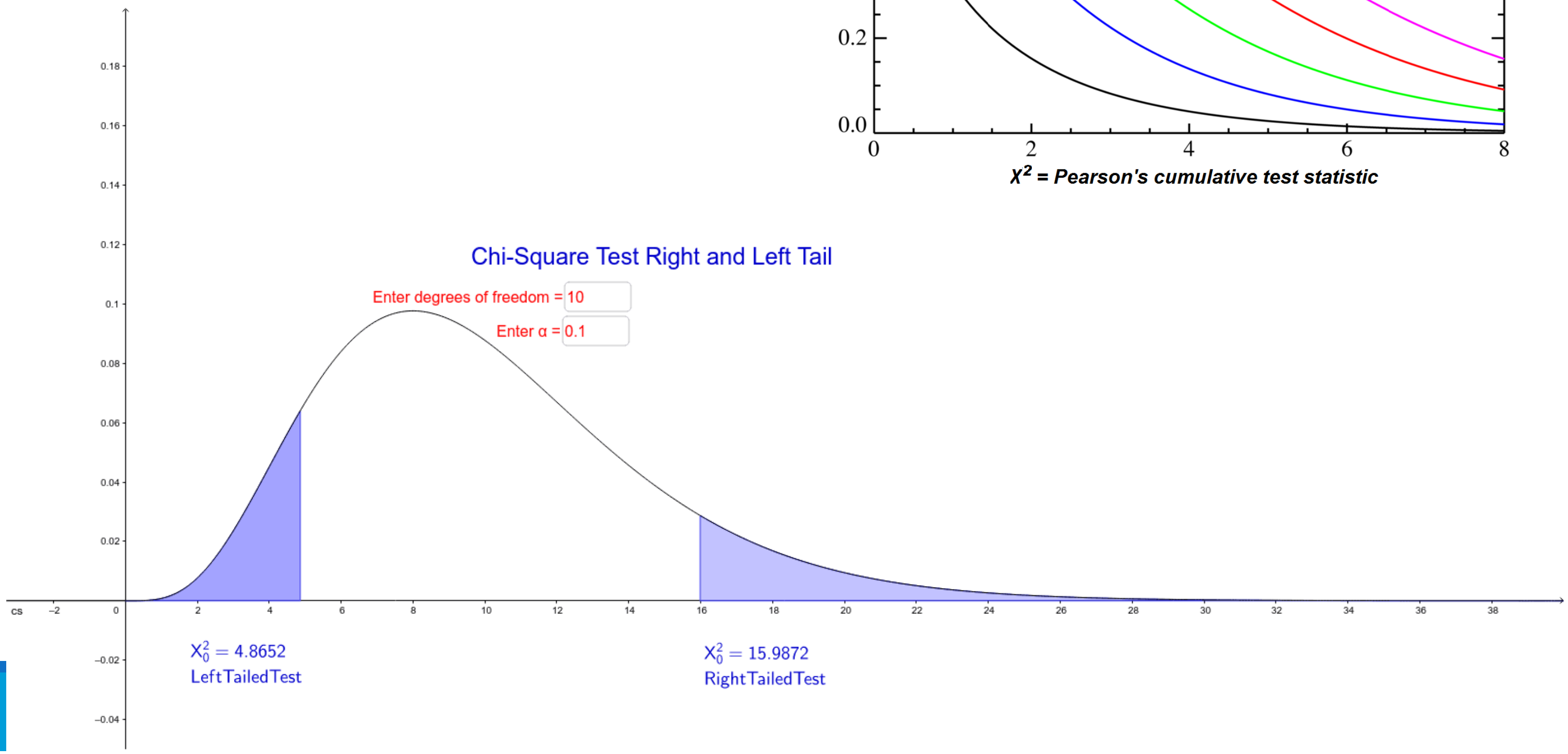
Chi-square test

The chi-square distribution.

Only positive values

Not symmetric.




Chi-Square Test Right and Left Tail

Enter degrees of freedom = 10

Enter α = 0.1

Chi-square test

Second example

```
>>> tshirts
              Black  White  Red  Blue
    Male         48     22   33    47
    Female       35     36   42    27

>>> results = stats.chi2_contingency(tshirts)

>>> results[0]    # x2 value
    11.56978992417547

>>> results[1]    # p-value
    0.0090120251137970З

>>> results[2]    # dof
    3

>>> myTable = results[3]   # expected values
```

Chi-square test

```
>>> myTable = results[3]   # expected values
>>> myTable
   array([[42.93103448, 30.        , 38.79310345, 38.27586207],
          [40.06896552, 28.        , 36.20689655, 35.72413793]])
>>> significance = 0.01
>>> p = 1- significance
>>> dof = results[2]
>>> critical_value = stats.chi.ppf(p, dof)
>>> critical_value
    3.3682141752187276
```

Conclusion

The data must speak for themselves.

Use both the visual effect and the precision of a numerical computation.

- Summarizing data by *numerical* measures (statistic)

  - Measures of central tendency

    mean, median, mode

  - Measures of variability

    variance, standard deviation

    quantiles : quartiles, deciles, centiles

  - View the distribution of the data

    (e.g. histogram, density plot, box-plot)

Conclusion

- Follow strictly the steps described in the first part (specify the a level before doing the test).
- The *t*-test is robust (even if the underling distribution is not really normally distributed).
- Using statistical tests is important.
- The tyranny of the sample mean (with the *t*-test over the median).
- The chi-square for independence is the second most important test.