Scientific Data Analysis

Cross-sectional data; non-parametric testing

Week 2

Dr. Rick Quax
Assistant Professor
Computational Science Lab

Course overview

Cross-sectional 1

Cross-sectional 2

Time-series

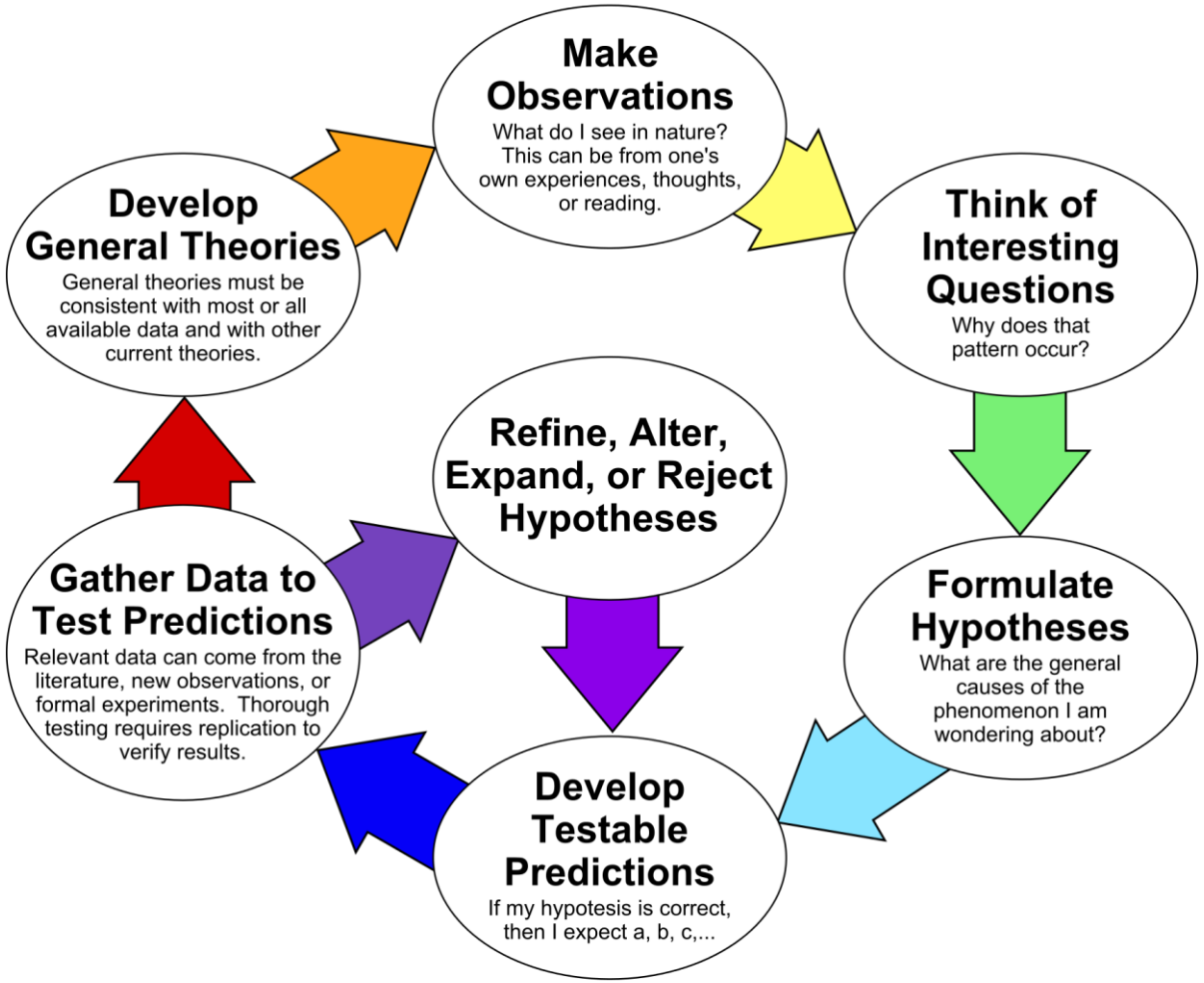
Text analysis

Final project

First things first

- How did Assignment 1 go?

Scientific process



Research yourself is useful!

Problem 2

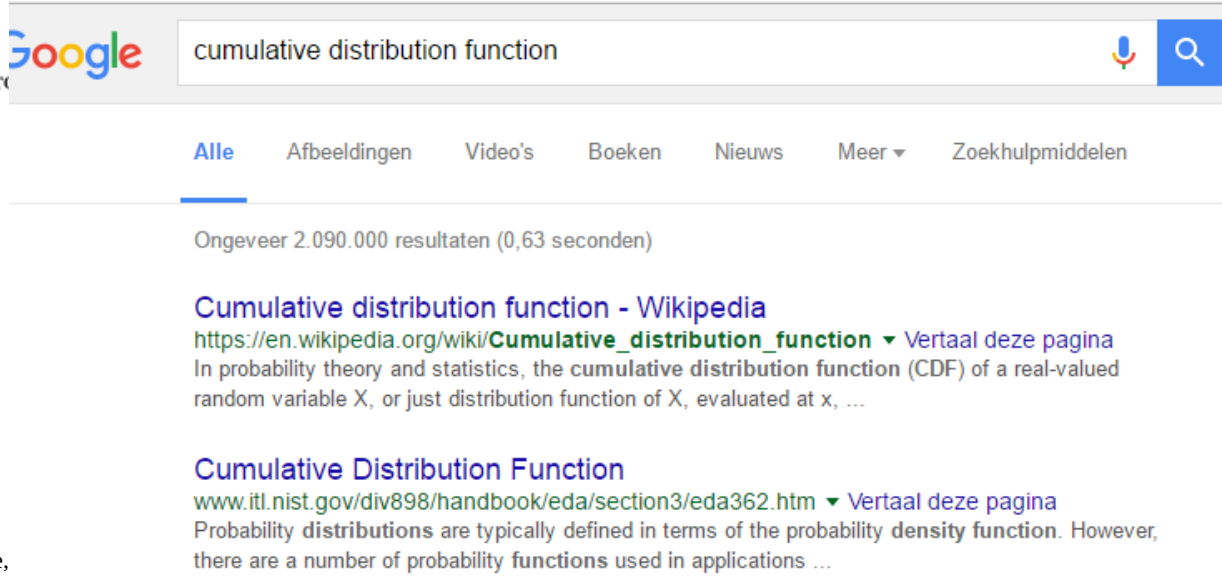
Now we have a uniformly distributed RNG, but we need more. All kinds of theoretical distributions are found in real data, including Normal, Log-normal, Pareto, etc. We need a way to generate random samples from any given distribution. In this assignment you will make an RNG for a Normal distribution.

1. A normally distributed random variable X is denoted $X \sim \mathcal{N}(\mu, \sigma^2)$. Please specify the formula for probability density function (PDF) $f(x)dx = \Pr(x < X < x + dx)$ in terms of μ and σ. Also please specify its cumulative density function (CDF) $\Pr(X \leq x)$.

   [3 points]

2. Plot the CDF[4] on the range of $\mu - 2\sigma, \mu + 2\sigma$ for an arbitrary choice of values for μ and σ. (Always label your axes and provide a concise but explanatory caption.) Use specifically this figure to indicate and explain (i) which numbers should have the highest probability and (ii) which numbers should have lower probability of being sampled by a corresponding normally distributed RNG.

3. For an RNG to be normally distributed we need the pr



Important: picking the H_0

- Null hypothesis H_0: usually represents *independence / no difference / no effect*
- Let's try:
  1. Does taking aspirin every day reduce the chance of having a heart attack?
  2. A machine should produce devices of 50 grams. Take a random sample of 10 devices. Does the machine work properly?
  3. Player A is better at tennis than player B.

Statistical testing in Python (two-sided, parametric)

In [14]:
```
import scipy.stats as st
```

In [23]:
```
# let us take a random sample (from a standard normal distribution)
s = np.random.randn()
s
```

Out[23]:
```
-1.2395671263567687
```

In [24]:
```
# H_0: "s is drawn from the standard normal distribution". Let's test at 95% confidence level.
# for a two-sided test, what is the 95% confidence interval? So 2.5% of all possible values
# must be on the left of this interval, and the same is true for the right side
# note: for shorthand I use st.norm.* here, but you may also program your own function with the PDF and CDF from
# https://www.wikiwand.com/en/Normal_distribution
print st.norm.ppf(0.025)  # 'percentile point function', can also be found by fitting a such that st.norm.cdf(a) == 0.025
print st.norm.ppf(0.975)
```

```
-1.95996398454
1.95996398454
```

In [25]:
```
# we can now see that we cannot reject H_0 (which is a good thing), because s is within the confidence interval
```

Statistical testing in Python (two-sided, parametric)

In [14]:
```
import scipy.stats as st
```

In [23]:
```
# let us take a random sample (from a standard normal distribution)
s = np.random.randn()
s
```

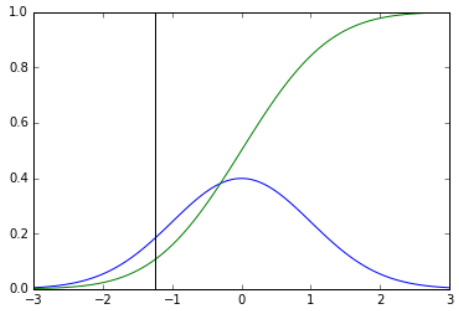
Out[23]:
```
-1.2395671263567687
```

In [24]:
```
# H_0: "s is drawn from the standard normal distribution". Let's test at 95% confidence level.
# for a two-sided test, what is the 95% confidence interval? So 2.5% of all possible values
# must be on the left of this interval, and the same is true for the right side
# note: for shorthand I use st.norm.* here, but you may also program your own function with the PDF and CDF from
# https://www.wikiwand.com/en/Normal_distribution
print st.norm.ppf(0.025)  # 'percentile point function', can also be found by fitting a such that st.norm.cdf(a) == 0.025
print st.norm.ppf(0.975)
```

```
-1.95996398454
1.95996398454
```

In [25]:
```
# we can now see that we cannot reject H_0 (which is a good thing), because s is within the confidence interval
```

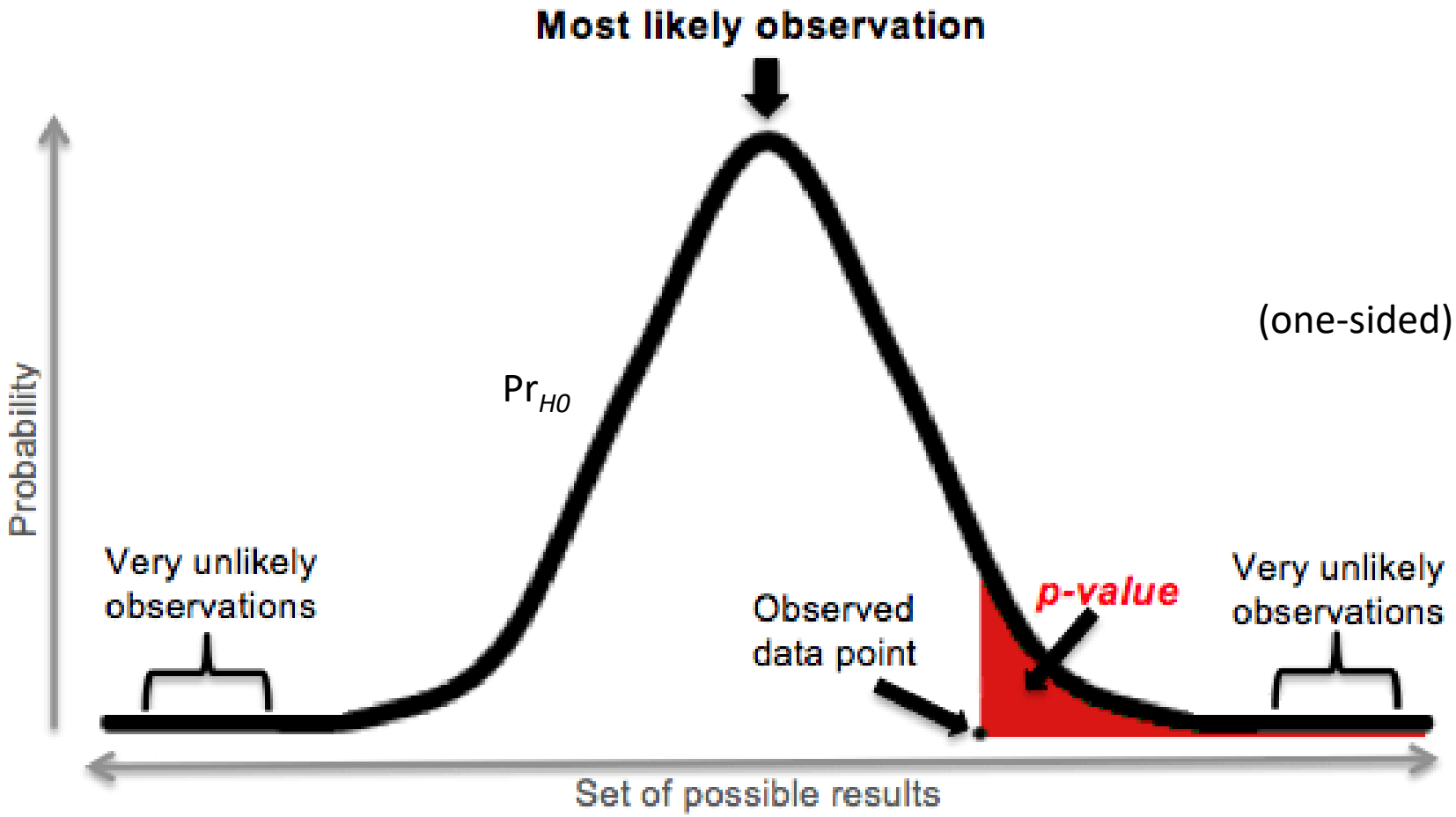
In [26]:
```
# let us show this in another, equivalent way: what is the probability of s 'or more extreme'?
xs = np.linspace(-3, 3, 100)
ys = [st.norm.pdf(x) for x in xs]
ys2 = st.norm.cdf(xs)
plt.plot(xs, ys)  # pdf
plt.plot(xs, ys2)  # cdf
plt.plot([s, s], [plt.ylim()[0], plt.ylim()[1]], '-k')
plt.show()
```



In [29]:
```
# the probability of s 'or more extreme' is the cumulative probability on the left side of -|s| as well as
# the cumulative probability of |s| on the right side:
print st.norm.cdf(-abs(s)) + (1-st.norm.cdf(abs(s)))
# since this probability is 21.5%, which is not equal or lower than 5%, we cannot reject H_0; in other words, s
# is not 'absurd' enough to be observed if H_0 would be true.
```

```
0.215135546071
```

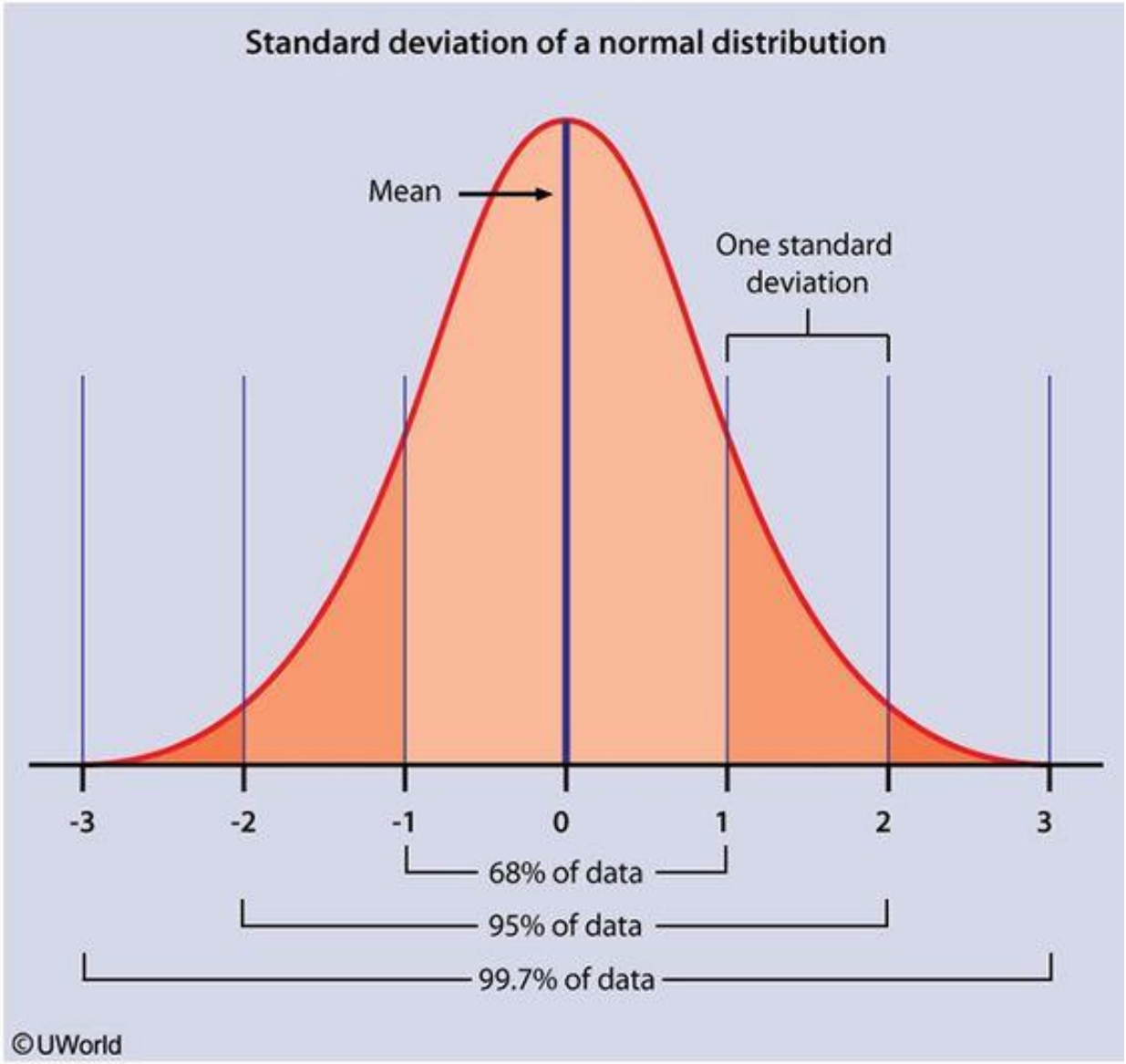
p-value



(one-sided)

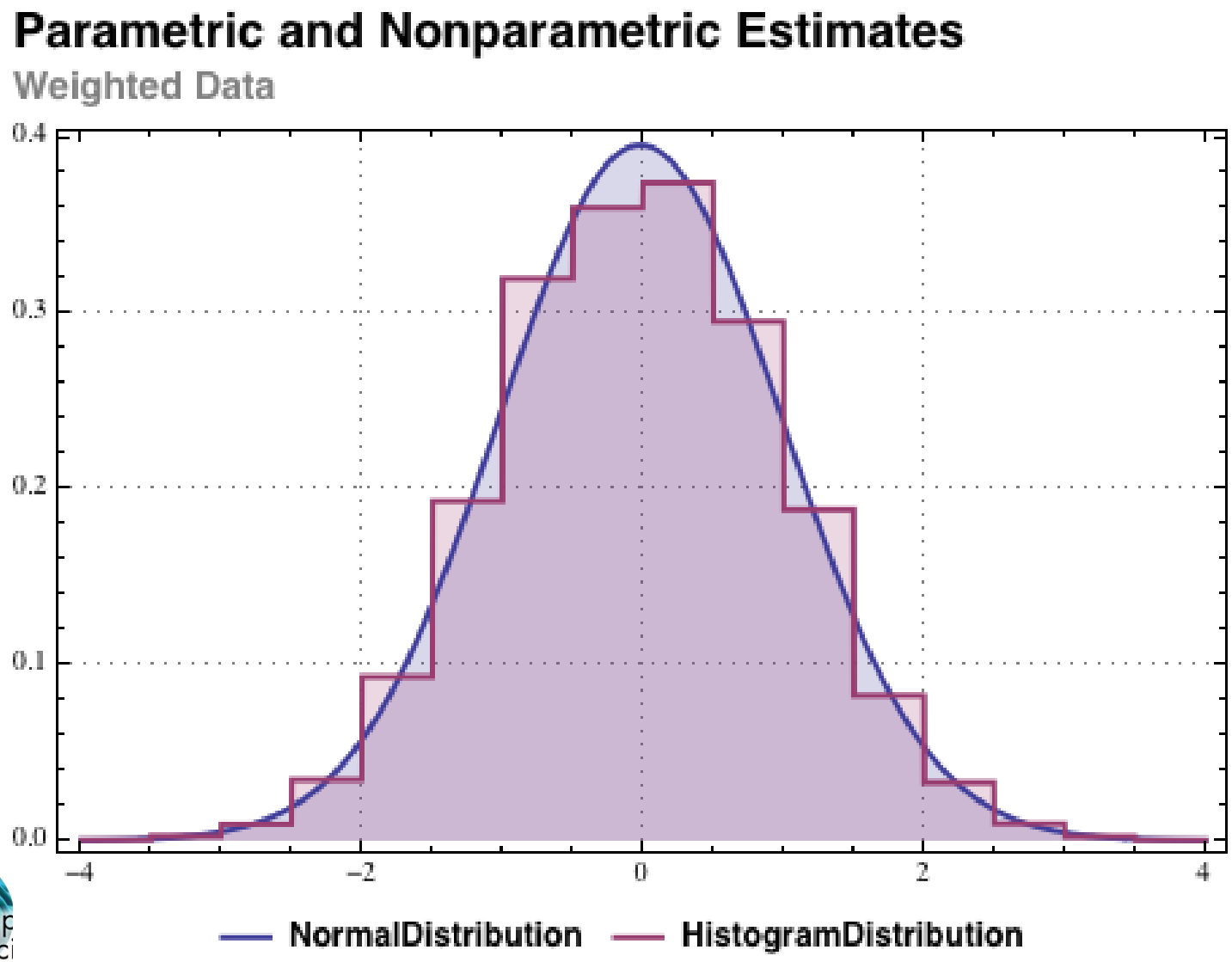
A *p-value* (shaded red area) is the probability of an observed (or more extreme) result arising by chance

p-value test == CI test



Computatio Science

Non-parametric testing

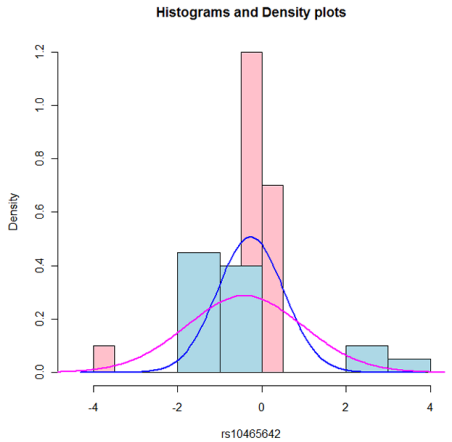


P and NP counterparts

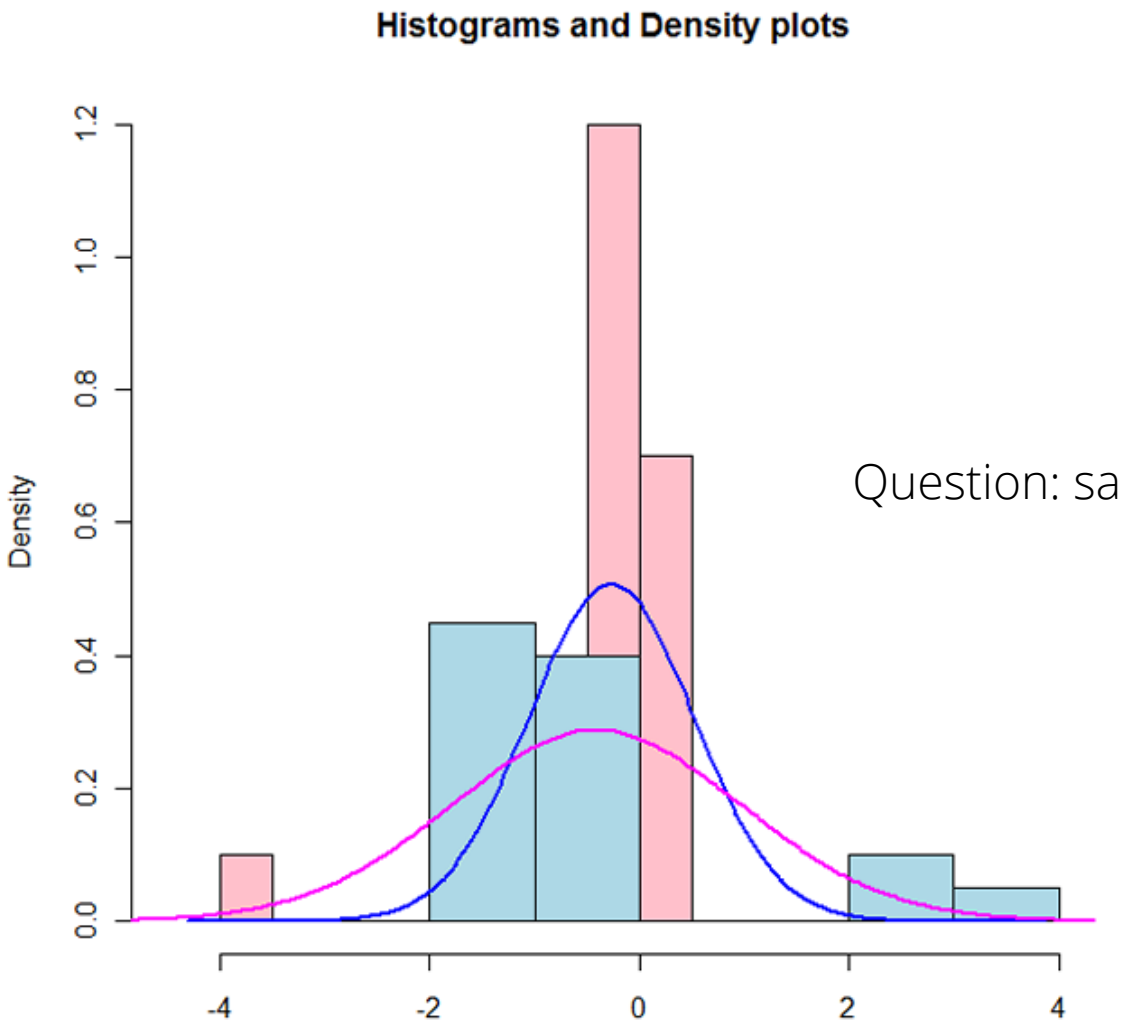
Parametric test	Non-Parametric equivalent
Paired t-test	Wilcoxon Rank sum Test
Unpaired t-test	Mann-Whitney U test
Pearson correlation	Spearman correlation
One way Analysis of variance	Kruskal Wallis Test

Reasons for NP

- Assumption of normal distribution is violated
  - Caveat: often can transform data to get 'close enough'
- Not willing to make assumption
  - E.g. very small dataset
  - Or no idea about the generating mechanism
- Easy!



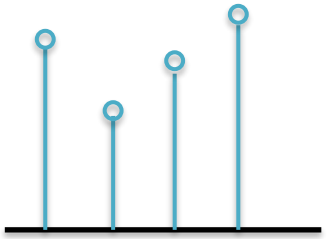
Example: normal distribution



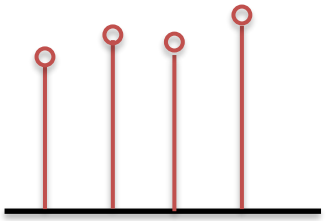
Question: same distribution?

Example: two sample tests

Dataset 1



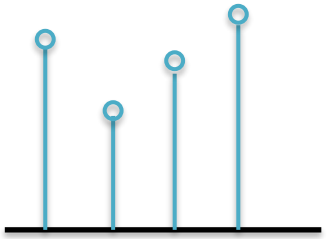
Dataset 2



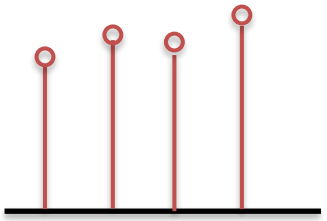
H_0: “the two population *means* are equal”

Example: two sample tests

Dataset 1

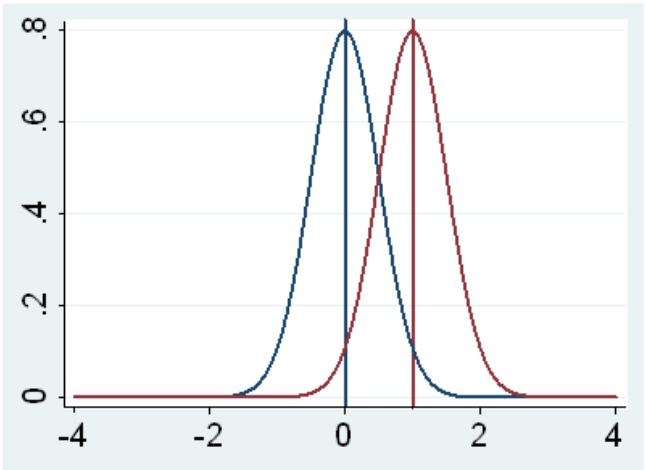


Dataset 2

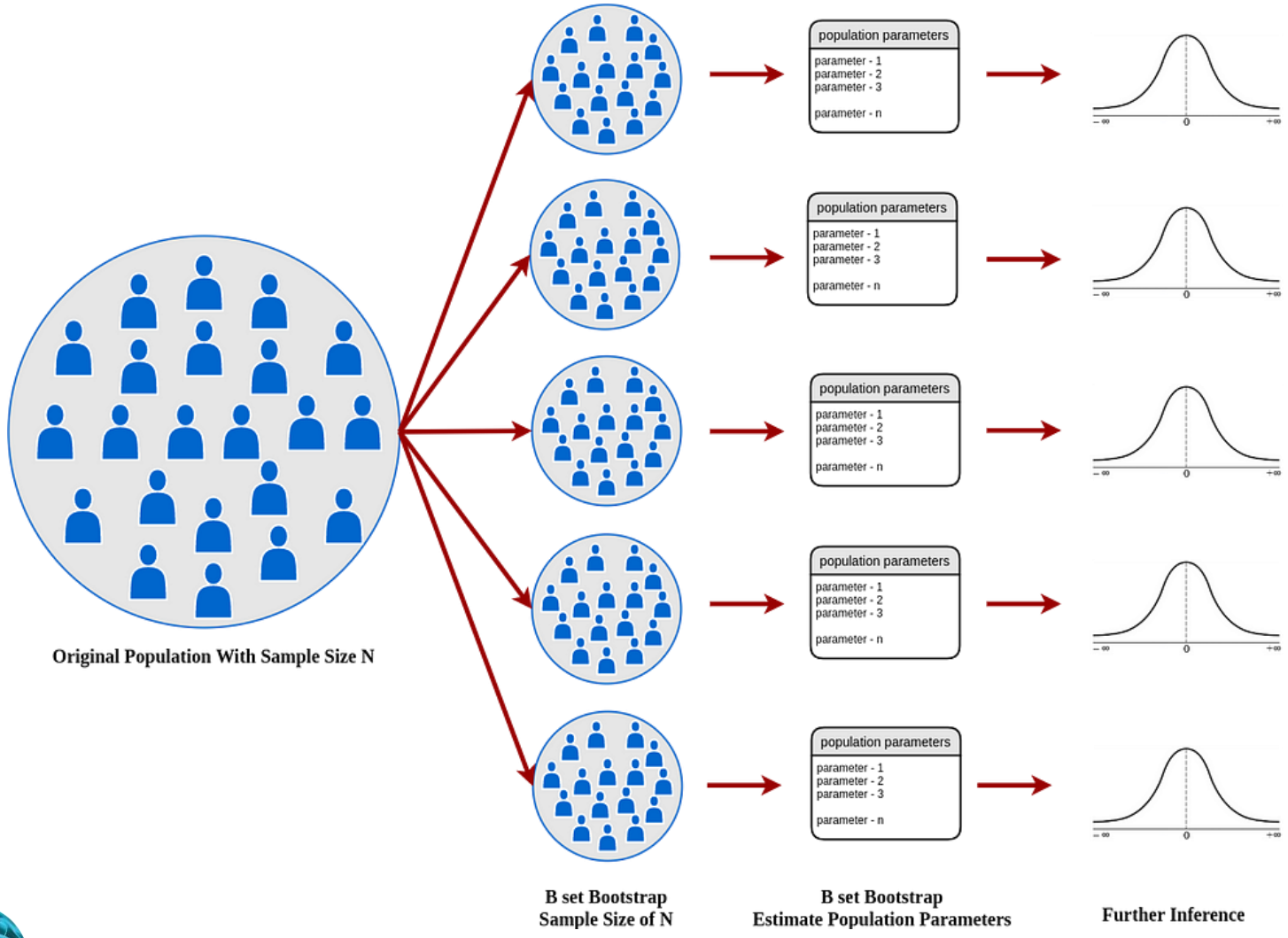


H_0: "the two population *means* are equal"

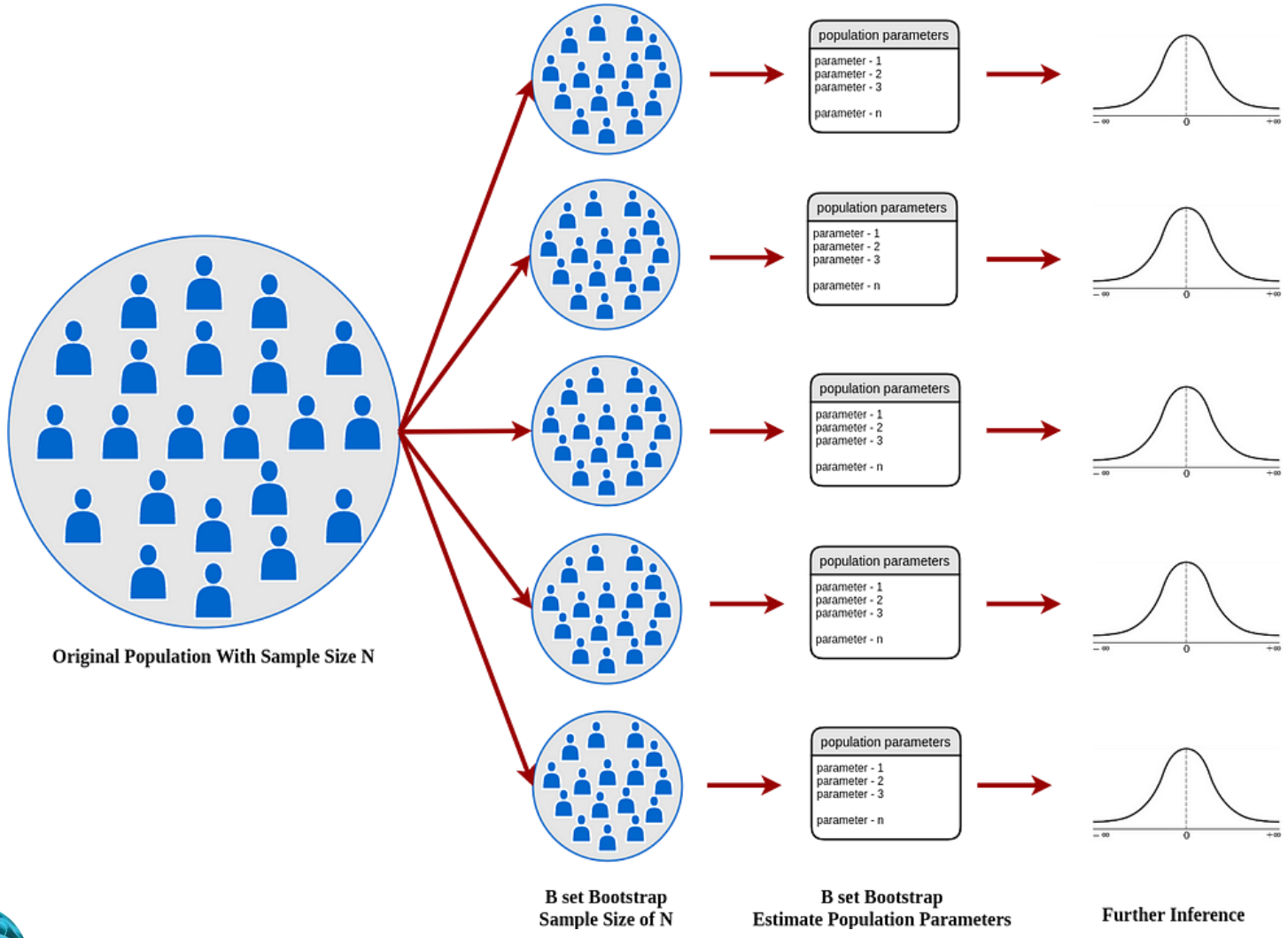
- Bootstrap each sample separately;
- Compute the new mean;
- Do this many time to build a histogram (distribution);
- Calculate the p-value of one sample's mean using the other sample's distribution.



Bootstrap == resample



Bootstrap ≈ simulation



Example: identical *distributions*

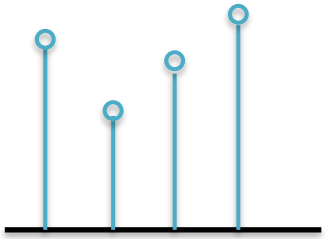
- Two datasets
  - H_0: “there is no difference in distribution.”

Example: identical *distributions*

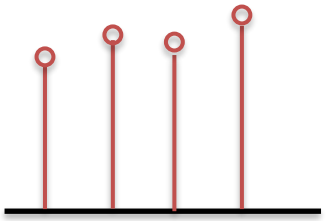
- Two datasets
  - H_0: "there is no difference in distribution."
  - scramble the two sets of values across each other to make a single 'population'
  - create *N* random 'surrogate' pairs of datasets
  - Compute the CI of a test statistic of choice
    - e.g., their difference of the two 'modes'
  - Test if the observed difference between the original data sets falls inside/outside the CI of the 'surrogate' mean differences
    - Equivalently, compute the *p-value* and compare it to the chosen confidence level (α)

Example: two sample tests

Dataset 1



Dataset 2

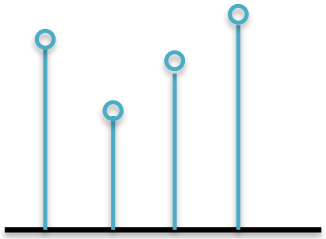


H_0: “the two population *means* are equal”

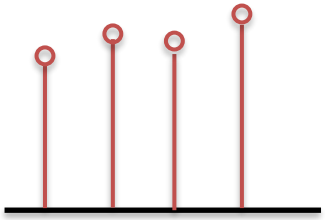
H_0: “the two population *distributions* are equal”

Example: two sample tests

Dataset 1



Dataset 2



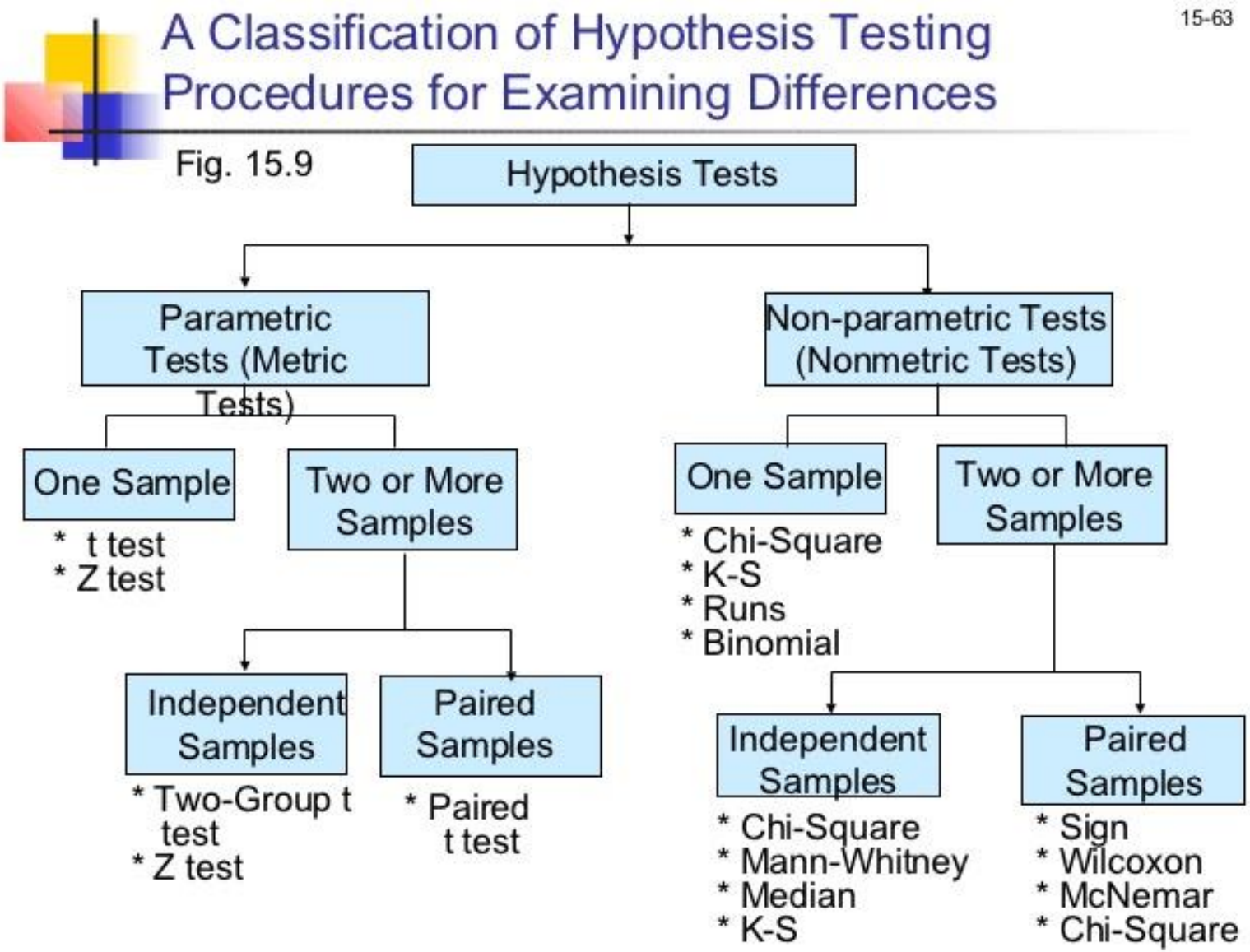
H_0: "the two population *means* are equal"

Resample each dataset from itself

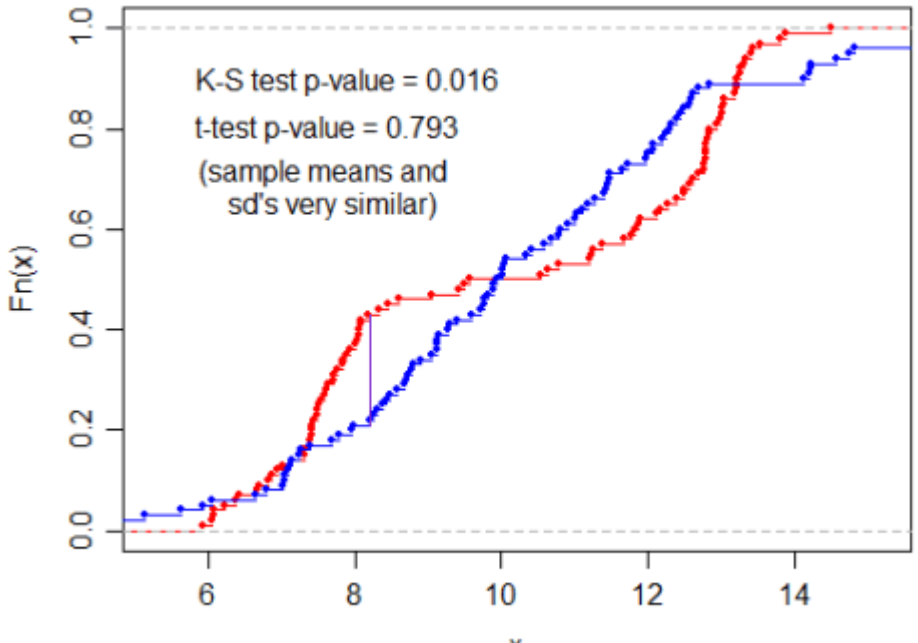
H_0: "the two population *distributions* are equal"

Resample each dataset from the combination of the two

Decision tree example

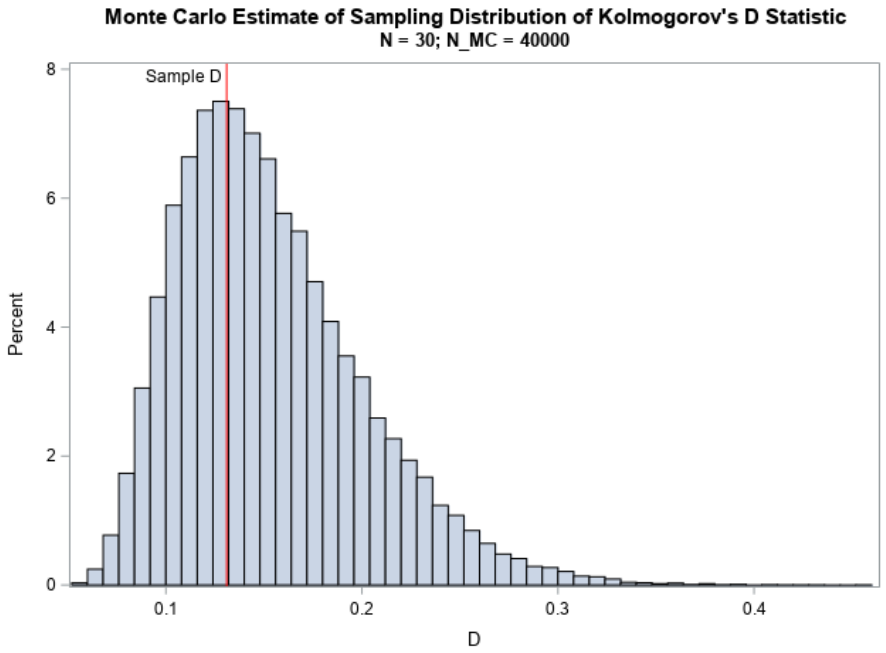


Normality testing: Kolmogorov-Smirnov test



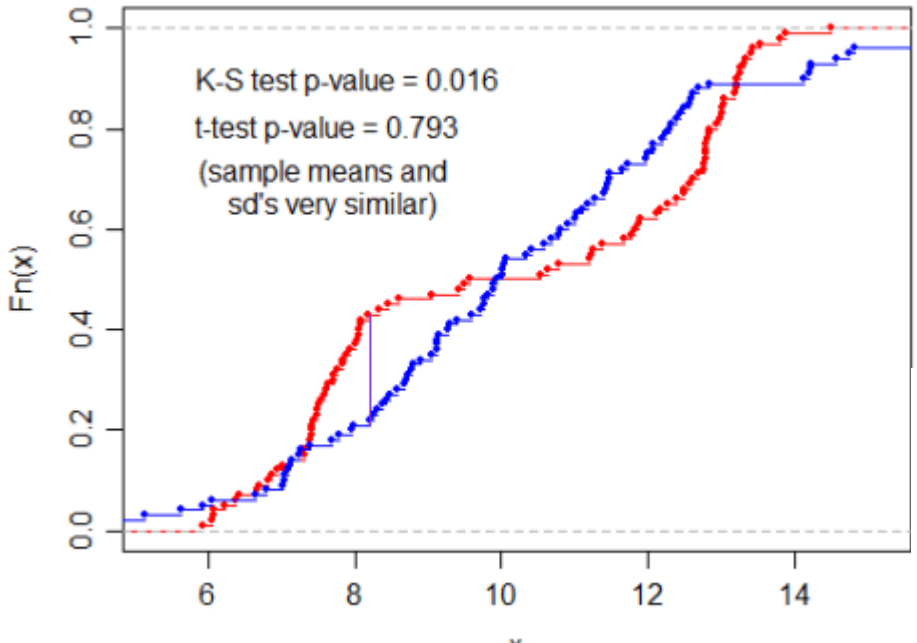
Source: https://towardsdatascience.com/kolmogorov-smirnov-test-84c92fb4158d

- Bootstrap an empirical CDF with equal number of points;
- Compute maximum vertical distance;
- Do this many times;
- Plot histogram:



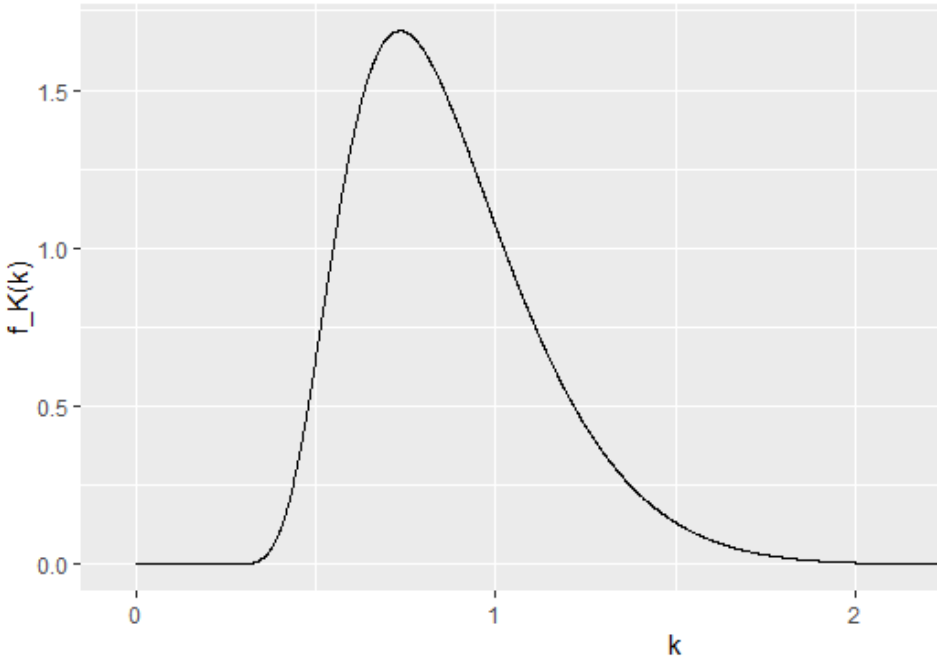
Source: https://blogs.sas.com/content/iml/2019/05/20/critical-values-kolmogorov-test.html

Normality testing: Kolmogorov-Smirnov test

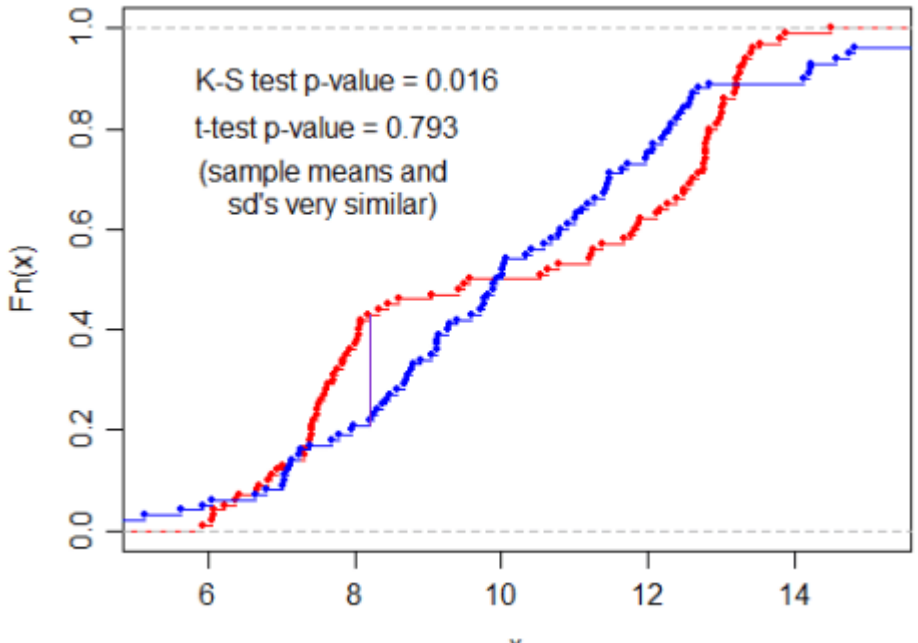


Source: https://towardsdatascience.com/kolmogorov-smirnov-test-84c92fb4158d

- Bootstrap an empirical CDF with equal number of points;
- Compute maximum vertical distance;
- Do this many times;
- Plot histogram:

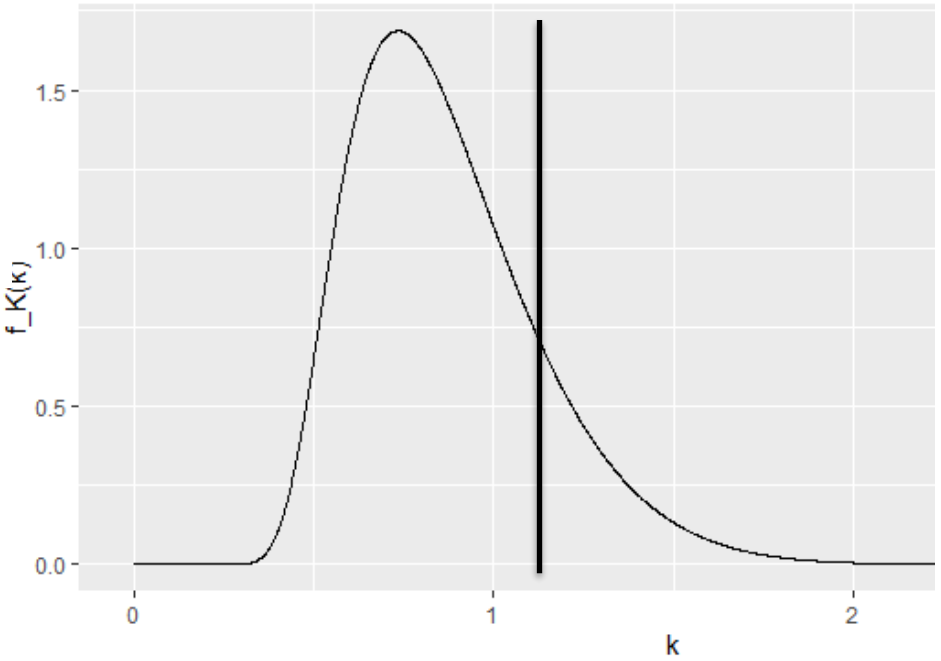


Normality testing: Kolmogorov-Smirnov test



Source: https://towardsdatascience.com/kolmogorov-smirnov-test-84c92fb4158d

- Compute p-value of test statistic (i.e., maximum vertical distance) of the actual data set given. (One-sided.)



Source: https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test

Example: rank test

- Design your own!
- Question (ranks data):
  - Is one group 'better' than the other?

Group 1		Group 2	
Raw	Rank	Raw	Rank
11.5	3	15.2	7
12.6	5	8.6	1
19.4	13	9.3	2
21.3	14	14.4	6
32.5	17	15.6	8
18.6	12	11.8	4
17.0	10	16.3	9
23.4	15	17.8	11
29.6	16		

Source