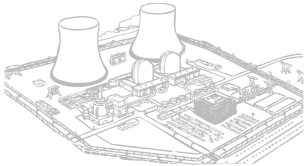
RISK ENGINEERING



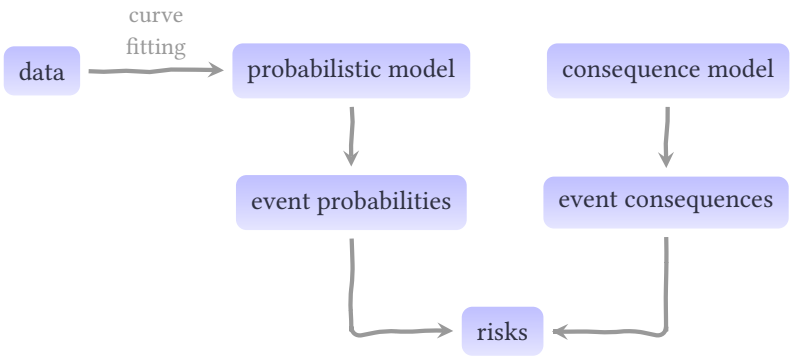
Analyzing data using Python

Eric Marsden

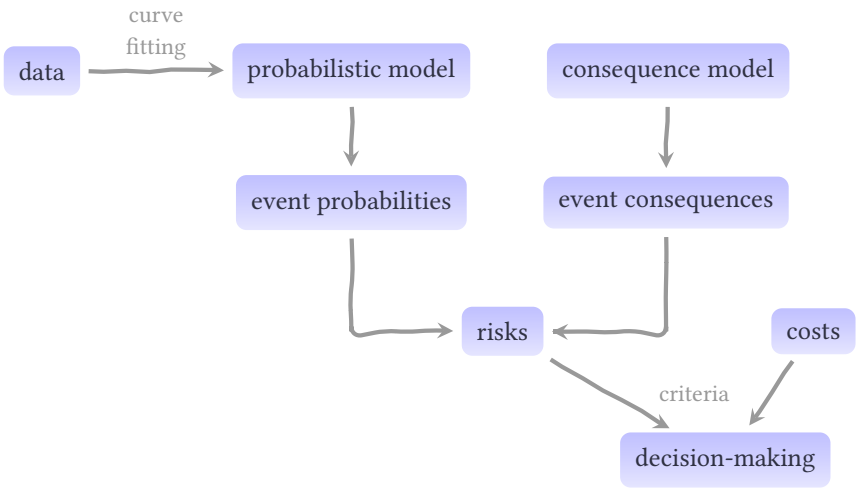
<eric.marsden@risk-engineering.org>

The purpose of computing is insight, not numbers.
– Richard Hamming

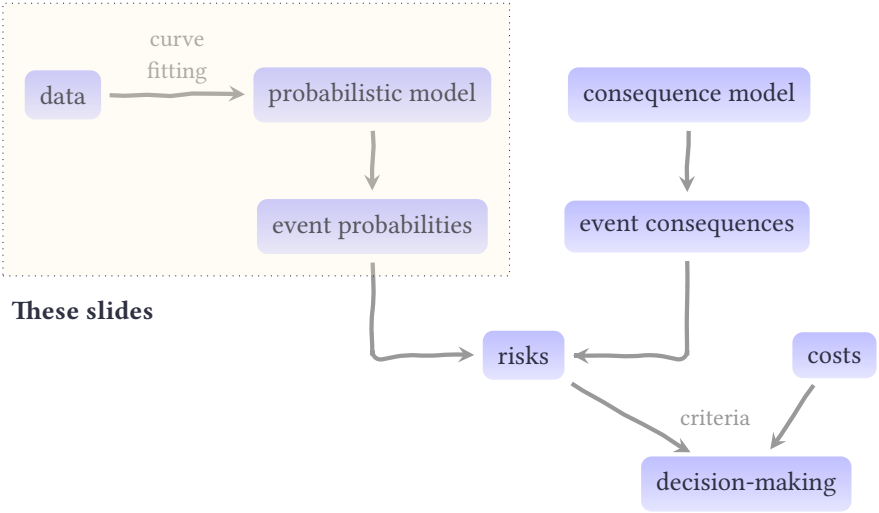
Where does this fit into risk engineering?



Where does this fit into risk engineering?



Where does this fit into risk engineering?

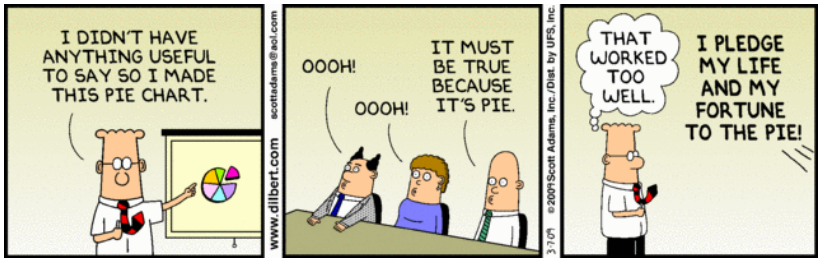


Descriptive statistics

Descriptive statistics

- **Descriptive statistics** allow you to summarize information about observations
  - organize and simplify data to help understand it
- **Inferential statistics** use observations (data from a sample) to make inferences about the total population
  - generalize from a sample to a population

Descriptive statistics



Source: dilbert.com

Measures of central tendency

- Central tendency (the "middle" of your data) is measured either by the median or the mean
- The **median** is the point in the distribution where half the values are lower, and half higher
  - it's the 0.5 quantile
- The (arithmetic) **mean** (also called the *average* or the mathematical expectation) is the "center of mass" of the distribution
  - continuous case: $\mathbb{E}(X) = \int_a^b xf(x)dx$
  - discrete case: $\mathbb{E}(X) = \sum_i x_i P(x_i)$
- The **mode** is the element that occurs most frequently in your data

Illustration: fatigue life of aluminium sheeting

Measurements of fatigue life (thousands of cycles until rupture) of strips of 6061-T6 aluminium sheeting, subjected to loads of 21 000 PSI.

Data from Birnbaum and Saunders (1958).

```
> import numpy
> cycles = numpy.array([370, 1016, 1235, [...] 1560, 1792])
> cycles.mean()
1400.9108910891089
> numpy.mean(cycles)
1400.9108910891089
> numpy.median(cycles)
1416.0
```

Source: Birnbaum, Z. W. and Saunders, S. C. (1958), *A statistical model for life-length of materials*, Journal of the American Statistical Association, 53(281)

RISK ENGINEERING

Aside: sensitivity to outliers

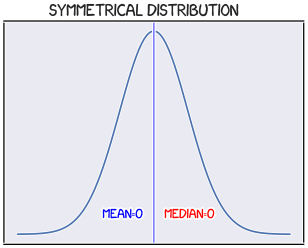
Note: the mean is quite sensitive to outliers, the median much less.

- the median is what's called a *robust* measure of central tendency

```
> import numpy
> weights = numpy.random.normal(80, 10, 1000)
> numpy.mean(weights)
79.83294314806949
> numpy.median(weights)
79.69717178759265
> numpy.percentile(weights, 50)
79.69717178759265  # 50th percentile = 0.5 quantile = median
> weights = numpy.append(weights, [10001, 101010])   # outliers
> numpy.mean(weights)
190.4630171138418    # <-- big change
> numpy.median(weights)
79.70768232050916    # <-- almost unchanged
```

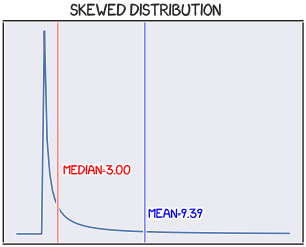
Measures of central tendency

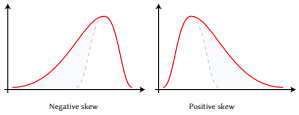
If the distribution of data is symmetrical, then the mean is equal to the median.



If the distribution is asymmetric (skewed), the mean is generally closer to the skew than the median.

Degree of asymmetry is measured by *skewness* (Python: `scipy.stats.skew()`)





Measures of variability

- **Variance** measures the dispersion (spread) of observations around the mean
  - $Var(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$
  - continuous case: $\sigma^2 = \int (x - \mu)^2 f(x) dx$ where $f(x)$ is the probability density function of X
  - discrete case: $\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2$
  - note: if observations are in metres, variance is measured in m^2
  - Python: `array.var()` or `numpy.var(array)`
- **Standard deviation** is the square root of the variance
  - it has the same units as the mean
  - Python: `array.std()` or `numpy.std(array)`

Exercise: Simple descriptive statistics

Task: Choose randomly 1000 integers from a uniform distribution between 100 and 200. Calculate the mean, min, max, variance and standard deviation of this sample.

```
> import numpy
> obs = numpy.random.randint(100, 201, 1000)
> obs.mean()
149.49199999999999
> obs.min()
100
> obs.max()
200
> obs.var()
823.99793599999998
> obs.std()
28.705364237368595
```

RISK ENGINEERING

Histograms: plots of variability

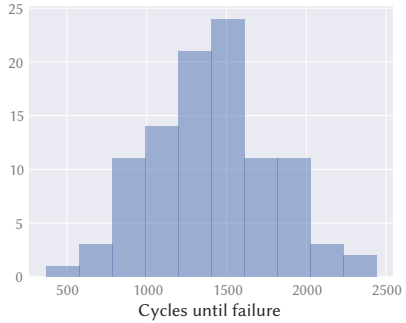
Histograms are a sort of bar graph that shows the distribution of data values. The vertical axis displays raw counts or proportions.

To build a histogram:

1. Subdivide the observations into several equal classes or intervals (called “bins”)
2. Count the number of observations in each interval
3. Plot the number of observations in each interval

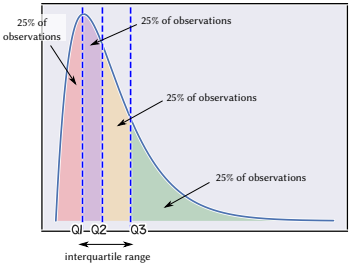
Note: the width of the bins is important to obtain a “reasonable” histogram, but is subjective.

```
import matplotlib.pyplot as plt
# our Birnbaum and Sanders failure data
plt.hist(cycles)
plt.xlabel("Cycles until failure")
```



Quartiles

- A *quartile* is the value that marks one of the divisions that breaks a dataset into four equal parts
- The first quartile, at the 25^{th} percentile, divides the first ¼ of cases from the latter ¾
- The second quartile, the median, divides the dataset in half
- The third quartile, the 75^{th} percentile, divides the first ¾ of cases from the latter ¼
- The *interquartile range* (IQR) is the distance between the first and third quartiles
  - 25^{th} percentile and the 75^{th} percentile

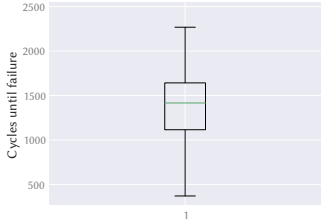


Box and whisker plot

A “box and whisker” plot or boxplot shows the spread of the data

- the median (horizontal line)
- lower and upper quartiles Q1 and Q3 (the box)
- upper whisker: last datum < Q3 + 1.5×IQR
- the lower whisker: first datum > Q1 - 1.5×IQR
- any data beyond the whiskers are typically called *outliers*

```
import matplotlib.pyplot as plt

plt.boxplot(cycles)
plt.xlabel("Cycles until failure")
```

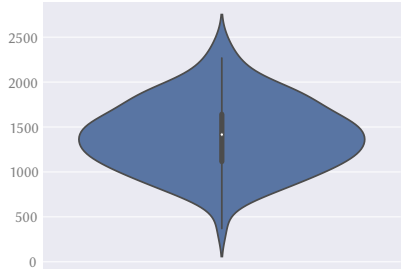


Note that some people plot whiskers differently, to represent the 5th and 95th percentiles for example, or even the min and max values...

Violin plot

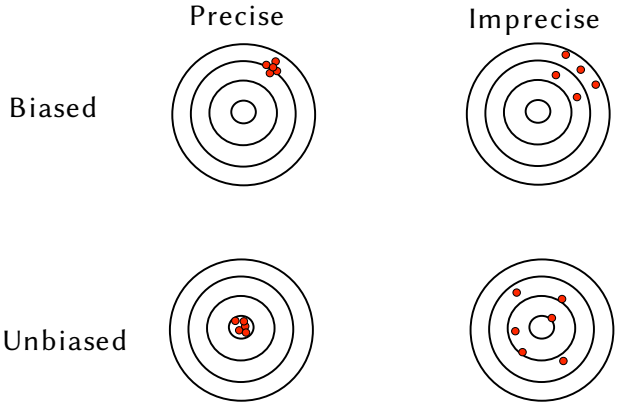
Adds a kernel density estimation to a boxplot

```
import seaborn as sns

sns.violinplot(cycles, orient="v")
plt.xlabel("Cycles until failure")
```



RISK ENGINEERING

Bias and precision



A good estimator should be unbiased, precise and consistent (converge as sample size increases).

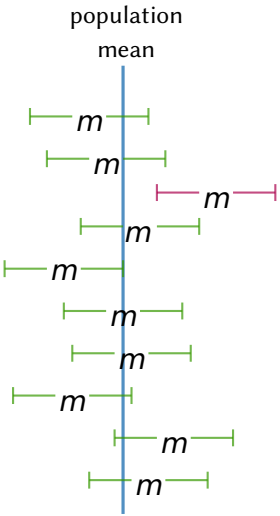
Estimating values

- In engineering, providing a point estimate is not enough: we also need to know the associated **uncertainty**
  - especially for risk engineering!
- One option is to report the **standard error**
  - $\frac{\hat{\sigma}}{\sqrt{n}}$, where $\hat{\sigma}$ is the sample standard deviation (an estimator for the population standard deviation) and n is the size of the sample
  - difficult to interpret without making assumptions about the distribution of the error (often assumed to be normal)
- Alternatively, we might report a **confidence interval**

Confidence intervals

- A two-sided confidence interval is an interval $[L, U]$ such that C% of the time, the parameter of interest will be included in that interval
  - most commonly, 95% confidence intervals are used
- Confidence intervals are used to describe the uncertainty in a point estimate
  - a wider confidence interval means greater uncertainty

Interpreting confidence intervals

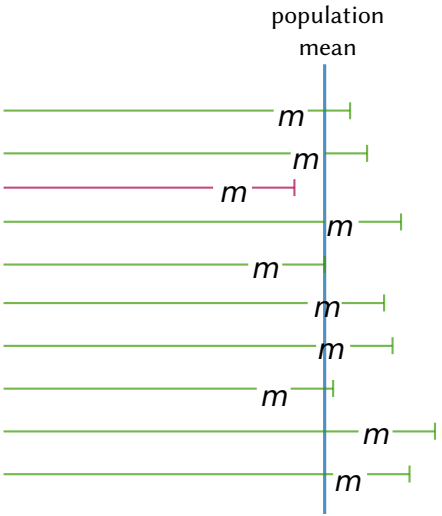


A 90% confidence interval means that 10% of the time, the parameter of interest will not be included in that interval.

Here, for a two-sided confidence interval.

RISK ENGINEERING

Interpreting confidence intervals



A 90% confidence interval means that 10% of the time, the parameter of interest will not be included in that interval.

Here, for a one-sided confidence interval.

RISK ENGINEERING

Illustration: fatigue life of aluminium sheeting

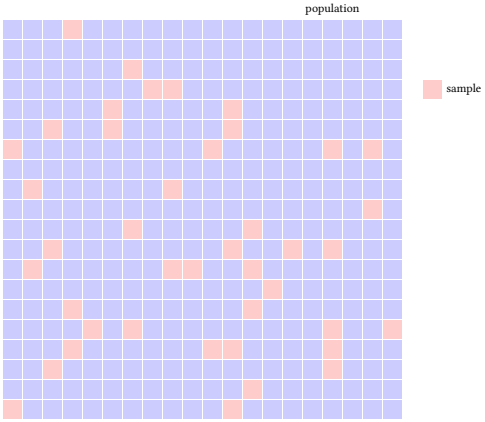
Confidence intervals can be displayed graphically on a barplot, as “error lines”.

Note however that this graphical presentation is ambiguous, because some authors represent the standard deviation on error bars. The caption should always state what the error bars represent.

```
import seaborn as sns

sns.barplot(cycles, ci=95, capsize=0.1)
plt.xlabel("Cycles until failure (95% CI)")
```

Data from Birnbaum and Saunders (1958)

Statistical inference



Statistical inference means deducing information about a population by examining only a subset of the population (the sample).

We use a **sample statistic** to estimate a **population parameter**.

RISK ENGINEERING

How to determine the confidence interval?

- If you make assumptions about the distribution of your data (eg. “the observations are normally distributed”), you can calculate the confidence interval for your estimation of the parameter of interest (eg. the mean) using analytical quantile measures

- If you don’t want to make too many assumptions, a technique called the **bootstrap** can help

- General idea:
  - I want to determine how much uncertainty is generated by the fact that I only have a limited sample of my full population
  - If I had a “full” sample, I could extract a large number of limited samples and examine the amount of variability in those samples (how much does my parameter of interest vary across these samples?)
  - I only have a limited sample, but I can look at a large number of limited samples from my own limited sample, by **sampling with replacement**

Bootstrap methods

- "Bootstrapping" means **resampling your data with replacement**
- Instead of fitting your model to the original X and y, you fit your model to resampled versions of X and y many times
- Provides n slightly different models from which we create a confidence interval
- **Hypothesis**: observations are the result of a model plus noise

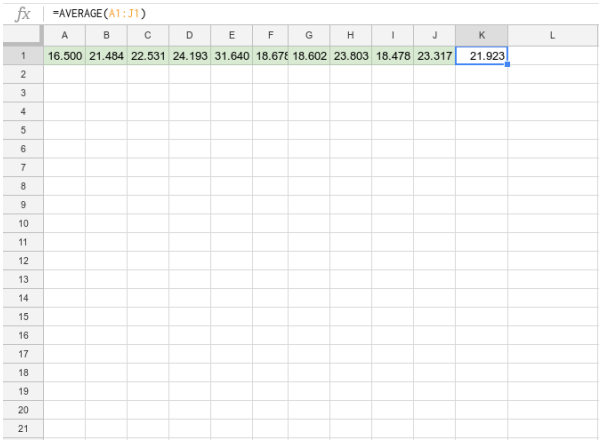
Bootstrapping confidence intervals in Python

1. Take a large number of samples of the same size as our original dataset, by sampling with replacement
2. For each sample, calculate the parameter of interest (eg. the mean)
3. Calculate the relevant percentile from the distribution of the parameter of interest

```
def bootstrap_confidence_intervals(data, estimator, percentiles, runs=1000):
    replicates = numpy.empty(runs)
    for i in range(runs):
        replicates[i] = estimator(numpy.random.choice(data, len(data), replace=True))
    est = numpy.mean(replicates)
    ci = numpy.percentile(numpy.sort(replicates), percentiles)
    return (est, ci)
```

RISK ENGINEERING

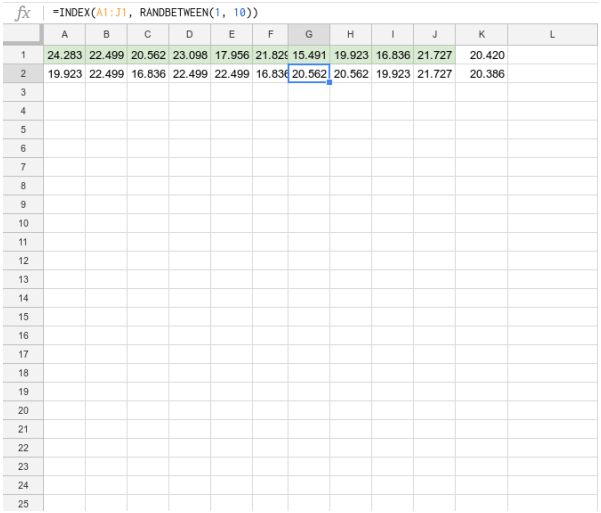
Illustration with Excel



Import your data into the first row of a spreadsheet (here we have 10 observations).

Calculate the sample mean (last column) using function AVERAGE.

Illustration with Excel



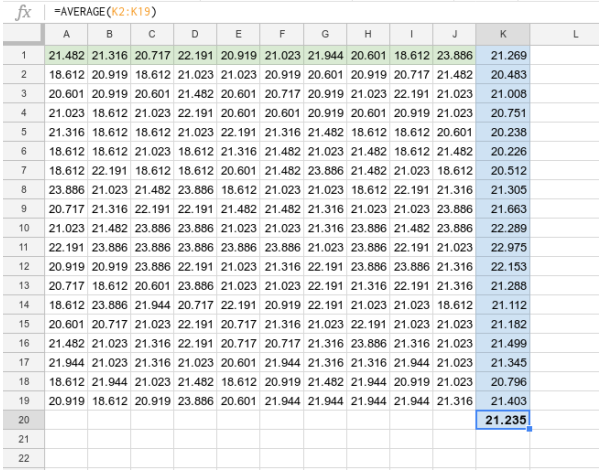
	A	B	C	D	E	F	G	H	I	J	K	L
1	24.283	22.499	20.562	23.098	17.956	21.826	15.491	19.923	16.836	21.727	20.420	
2	19.923	22.499	16.836	22.499	22.499	16.836	20.562	20.562	19.923	21.727	20.386	
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												

The first replicate, placed in the row under the sample, is obtained by resampling with replacement from the original sample.

Each cell contains the formula `INDEX(A1:J1, RANDBETWEEN(1, 10))`, meaning “choose a random element from the first row”.

Calculate the mean of the first replicate (last column) using function `AVERAGE`.

Illustration with Excel



	A	B	C	D	E	F	G	H	I	J	K	L
1	21.482	21.316	20.717	22.191	20.919	21.023	21.944	20.601	18.612	23.886	21.269	
2	18.612	20.919	18.612	21.023	21.023	20.919	20.601	20.919	20.717	21.482	20.483	
3	20.601	20.919	20.601	21.482	20.601	20.717	20.919	21.023	22.191	21.023	21.008	
4	21.023	18.612	21.023	22.191	20.601	20.601	20.919	20.601	20.919	21.023	20.751	
5	21.316	18.612	18.612	21.023	22.191	21.316	21.482	18.612	18.612	20.601	20.238	
6	18.612	18.612	21.023	18.612	21.316	21.482	21.023	21.482	18.612	21.482	20.226	
7	18.612	22.191	18.612	18.612	20.601	21.482	23.886	21.482	21.023	18.612	20.512	
8	23.886	21.023	21.482	23.886	18.612	21.023	21.023	18.612	22.191	21.316	21.305	
9	20.717	21.316	22.191	22.191	21.482	21.482	21.316	21.023	21.023	23.886	21.663	
10	21.023	21.482	23.886	23.886	21.023	21.023	21.316	23.886	21.482	23.886	22.289	
11	22.191	23.886	23.886	23.886	23.886	23.886	21.023	23.886	22.191	21.023	22.975	
12	20.919	20.919	23.886	22.191	21.023	21.316	22.191	23.886	23.886	21.316	22.153	
13	20.717	18.612	20.601	23.886	21.023	21.023	22.191	21.316	22.191	21.316	21.288	
14	18.612	23.886	21.944	20.717	22.191	20.919	22.191	21.023	21.023	18.612	21.112	
15	20.601	20.717	21.023	22.191	20.717	21.316	21.023	22.191	21.023	21.023	21.182	
16	21.482	21.023	21.316	22.191	20.717	20.717	21.316	23.886	21.316	21.023	21.499	
17	21.944	21.023	21.316	21.023	20.601	21.944	21.316	21.316	21.944	21.023	21.345	
18	18.612	21.944	21.023	21.482	18.612	20.919	21.482	21.944	20.919	21.023	20.796	
19	20.919	18.612	20.919	23.886	20.601	21.944	21.944	21.944	21.944	21.316	21.403	
20											**21.235**	
21												
22												

Generate a large number of replicates (here 18), in successive rows of the spreadsheet.

The mean of all the replicate means (here in bold) is the **bootstrapped mean**. Other estimations such as confidence intervals can be obtained from the blue column of replicate means.

Illustration: bootstrapped mean of Birnbaum data

Let's calculate the bootstrapped mean and associated 95% confidence interval for the Birnbaum and Saunders (1958) fatigue data.

```
> import numpy
> cycles = numpy.array([370, 1016, 1235, [...] 1560, 1792])
> cycles.mean()
1400.9108910891089
> est, ci = bootstrap_confidence_intervals(cycles, numpy.mean, [5, 95])
> print("Bootstrapped mean and CI: {:.1f} [{:.1f}, {:.1f}]".format(est, ci[0], ci[1]))
Bootstrapped mean and CI95: 1403.7 [1332.8, 1478.9]
```

Fitting models

Fitting a distribution to observations

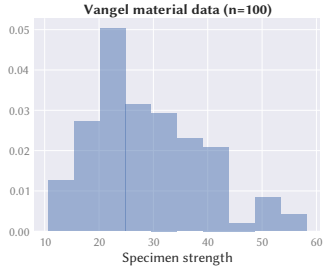
- The probability distributions defined in `scipy.stats` have a `fit()` method to find the parameters that provide a "best" fit
  - more precisely, the maximum likelihood of being the best fit
- For example
  - `scipy.stats.norm.fit(data)`
  - `scipy.stats.lognorm.fit(data)`
  - `scipy.stats.expon.fit(data)`
  - `scipy.stats.pareto.fit(data)`
- They return different parameters depending on the distribution, with some common parameters
  - `loc` for the mean
  - `scale` for the standard deviation

RISK ENGINEERING

Example: material strength data

Let us examine material strength data collected by the US NIST. We plot a histogram to examine the distribution of the observations.

```
import pandas
import matplotlib.pyplot as plt
data = pandas.read_csv("VANGEL5.DAT", header=None)
vangel = data[0]
N = len(vangel)
plt.hist(vangel, density=True, alpha=0.5)
plt.title("Vangel material data (n={})".format(N))
plt.xlabel("Specimen strength")
```



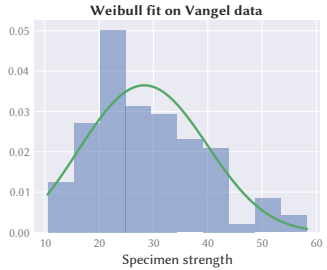
Data source: `VANGEL5.DAT` from NIST dataplot datasets

Example: material strength data

There are no obvious outliers in this dataset. The distribution is asymmetric (the right tail is longer than the left tail) and the literature suggests that material strength data can often be well modeled using a Weibull distribution, so we fit a Weibull distribution to the data, which we plot superimposed on the histogram.

The `weibull_min.fit()` function returns the parameters for a Weibull distribution that shows the best fit to the distribution (using a technique called *maximum likelihood estimation* that we will not describe here).

```
from scipy.stats import weibull_min
plt.hist(vangel, density=True, alpha=0.5)
shape, loc, scale = weibull_min.fit(vangel, floc=0)
x = numpy.linspace(vangel.min(), vangel.max(), 100)
plt.plot(x, weibull_min(shape, loc, scale).pdf(x))
plt.title("Weibull fit on Vangel data")
plt.xlabel("Specimen strength")
```

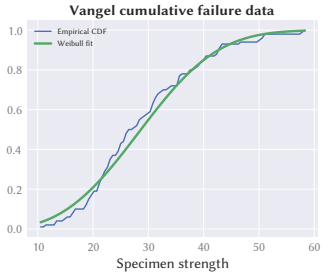


Data source: `VANGEL5.DAT` from NIST dataplot datasets

Example: material strength data

It may be more revealing to plot the **cumulative failure-intensity data**, the CDF of the dataset. We superimpose the empirical CDF generated directly from the observations, and the analytical CDF of the fitted Weibull distribution.

```
import statsmodels.distributions
ecdf = statsmodels.distributions.ECDF(vangel)
plt.plot(x, ecdf(x), label="Empirical CDF")
plt.plot(x, weibull_min(shape,loc,scale).cdf(x),\
  label="Weibull fit")
plt.title("Vangel cumulative failure intensity")
plt.xlabel("Specimen strength")
plt.legend()
```



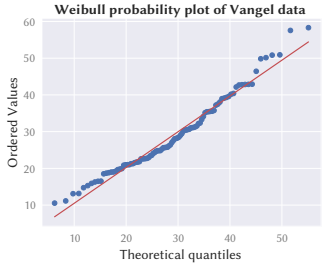
Example: material strength data

To assess whether the Weibull distribution is a good fit for this dataset, we examine a **probability plot** of the quantiles of the empirical dataset against the quantiles of the Weibull distribution. If the data comes from a Weibull distribution, the points will be close to a diagonal line.

Distributions generally differ the most in the tails, so the position of the points at the left and right extremes of the plot are the most important to examine.

In this case, the fit with a Weibull distribution is good.

```
from scipy.stats import probplot, weibull_min
probplot(vangel, \
         dist=weibull_min(shape,loc,scale),\
         plot=plt.figure().add_subplot(111))
plt.title("Weibull probability plot of Vangel data")
```



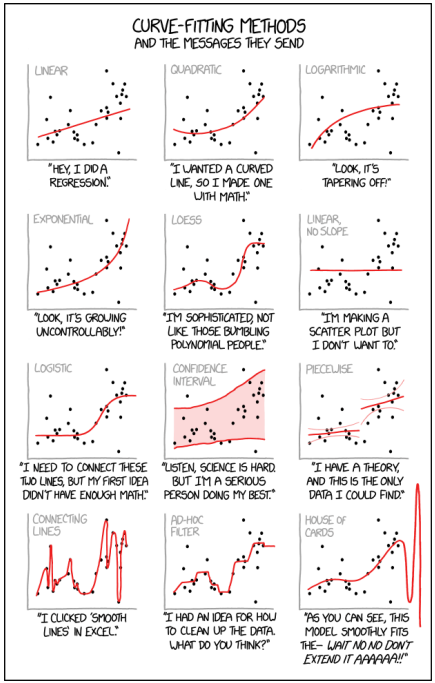
Example: material strength data

We can use the fitted distribution (the model for our observations) to estimate confidence intervals concerning the material strength.

For example, what is the minimal strength of this material, with a 99% confidence level? It's the 0.01 quantile, or the first percentile of our distribution.

```
> scipy.stats.weibull_min(shape, loc, scale).ppf(0.01)
7.039123866878374
```

RISK ENGINEERING



Source: xkcd.com/2048, CC BY-NC licence

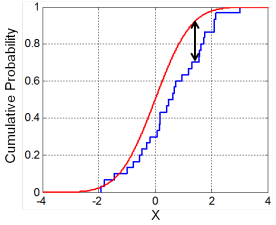
RISK ENGINEERING

Assessing goodness of fit

The Kolmogorov-Smirnov test provides a measure of goodness of fit.

It returns a distance D, the maximum distance between the CDFs of the two samples. The closer this number is to 0, the more likely it is that the two samples were drawn from the same distribution.

Python: `scipy.stats.kstest(obs, distribution)`



(The K-S test also returns a p-value, which describes the statistical significance of the D statistic. However, the p-value is not valid when testing against a model that has been fitted to the data, so we will ignore it here.)

Exercise

Problem

We have the following field data for time to failure of a pump (in hours): 3568 2599 3681 3100 2772 3272 3529 1770 2951 3024 3761 3671 2977 3110 2567 3341 2825 3921 2498 2447 3615 2601 2185 3324 2892 2942 3992 2924 3544 3359 2702 3658 3089 3272 2833 3090 1879 2151 2371 2936

What is the probability that the pump will fail after it has worked for at least 2000 hours? Provide a 95% confidence interval for your estimate.

Exercise

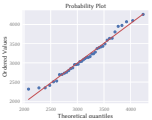
Solution (1/3)

```
import scipy.stats
import matplotlib.pyplot as plt

obs = [3568, 2599, 3681, 3100, 2772, 3272, 3529, 1770, 2951, \
  3024, 3761, 3671, 2977, 3110, 2567, 3341, 2825, 3921, 2498, \
  2447, 3615, 2601, 2185,  3324, 2892, 2942, 3992, 2924, 3544, \
  3359, 2702, 3658, 3089, 3272, 2833, 3090, 1879, 2151, 2371, 2936]

# start with some exploratory data analysis to look at the “shape”
# of the data
plt.hist(obs, density=True)
plt.title("Distribution of failure times (hours)")
```

Exercise

Solution (2/3)

```
# From the histogram, a normal distribution looks like a
# reasonable model. We fit a distribution and plot an
# overlay of the fitted distribution on the histogram,
# as well as a probability plot.
mu, sigma = scipy.stats.norm.fit(obs)
fitted = scipy.stats.norm(mu, sigma)
plt.hist(obs, density=True, alpha=0.5)
support = numpy.linspace(obs.min(), obs.max(), 100)
plt.plot(support, fitted.pdf(support), lw=3)
plt.title("Distribution of failure times (hours)")

fig = plt.figure().add_subplot(111)
scipy.stats.probplot(obs, dist=fitted, plot=fig)
```



Exercise

Solution (3/3)

```
def failure_prob(observations):
    mu, sigma = scipy.stats.norm.fit(observations)
    return scipy.stats.norm(mu, sigma).cdf(2000)

# Estimate confidence intervals using the bootstrap method. This is
# estimating the amount of uncertainty in our estimated failure probability
# that is caused by the limited number of observations.
est, ci = bootstrap_confidence_intervals(obs, failure_prob, [2.5, 97.5])
print("Estimate {:.5f}, CI95=[{:.5f}, {:.5f}]".format(est, ci[0], ci[1]))
```

Results are something like 0.01075, $CI_{95} = [0.00206, 0.02429]$.

RISK ENGINEERING

Illustration: fitting a distribution to wind speed data

Let us examine a histogram of wind speed data from TLS airport, in 2013.

```
data = pandas.read_csv("TLS-weather-data.csv")
wind = data["Mean Wind SpeedKm/h"]
plt.hist(wind, density=True, alpha=0.5)
plt.xlabel("Wind speed (km/h)")
plt.title("TLS 2013 wind speed data")
```

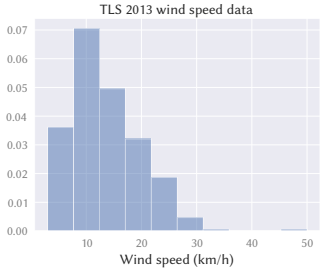
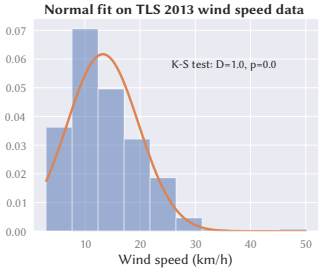


Illustration: fitting a distribution to wind speed data

We can attempt to fit a normal distribution to the data, and examine a probability plot. (Note that the data distribution looks skewed and the normal distribution is symmetric, so it's not really a very good choice).

```
shape, loc = scipy.stats.norm.fit(wind)
fitted = scipy.stats.norm(shape, loc)
plt.hist(wind, density=True, alpha=0.5)
x = numpy.linspace(wind.min(), wind.max(), 100)
plt.plot(x, fitted.pdf(x))
plt.title("Normal fit on TLS 2013 wind speed data")
plt.xlabel("Wind speed (km/h)")

scipy.stats.probplot(wind, dist=fitted,
  plot=plt.figure().add_subplot(111))
plt.title("Normal probability plot of wind speed")
```

Indeed, the probability plot shows quite a poor fit for the normal distribution, in particular in the tails of the distributions.



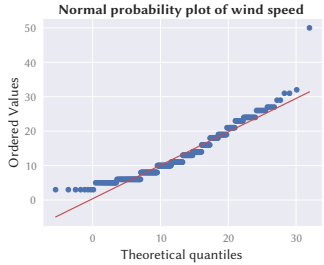
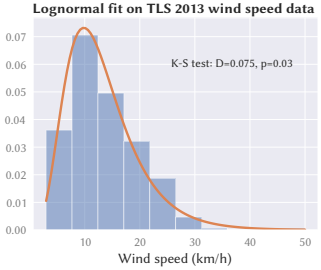


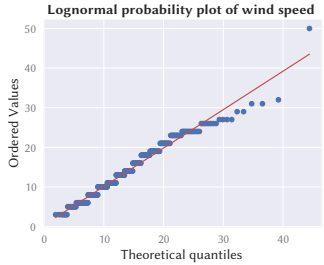
Illustration: fitting a distribution to wind speed data

We can attempt to fit a lognormal distribution to the data, and examine a probability plot.

```
shape, loc, scale = scipy.stats.lognorm.fit(wind)
fitted = scipy.stats.lognorm(shape, loc, scale)
plt.hist(wind, density=True, alpha=0.5)
x = numpy.linspace(wind.min(), wind.max(), 100)
plt.plot(x, fitted.pdf(x))
plt.title("Lognormal fit on TLS 2013 wind speed data")
plt.xlabel("Wind speed (km/h)")

scipy.stats.probplot(wind, dist=fitted,
  plot=plt.figure().add_subplot(111))
plt.title("Lognormal probability plot of wind speed")
```

The probability plot gives much better results here.





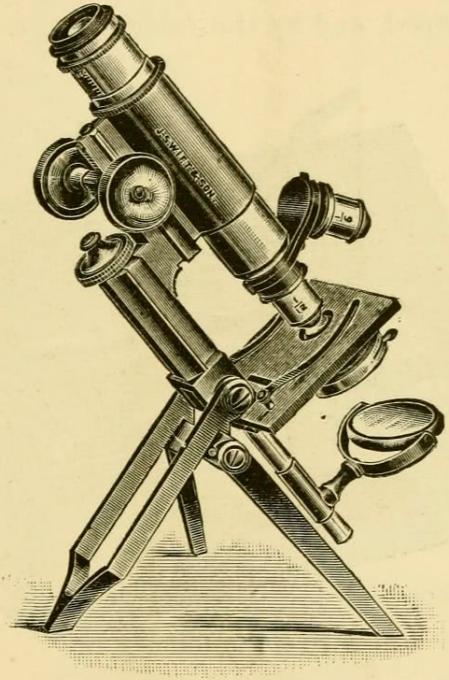
Exercise

- Download heat flow meter data collected by B. Zarr (NIST, 1990)

  → `https://www.itl.nist.gov/div898/handbook/eda/section4/eda4281.htm`

- Plot a histogram for the data
- Generate a normal probability plot to check whether the measurements fit a normal (Gaussian) distribution
- Fit a normal distribution to the data
- Calculate the sample mean and the estimated population mean using the bootstrap technique
- Calculate the standard deviation
- Estimate the 95% confidence interval for the population mean, using the bootstrap technique

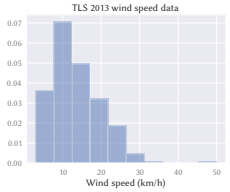
RISK ENGINEERING

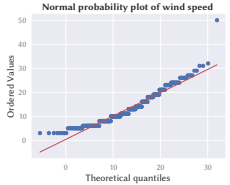


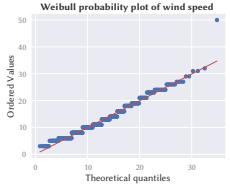
Analyzing data

Analyzing data: wind speed

- Import wind speed data for Toulouse airport
- Find the mean of the distribution
- Plot a histogram of the data
- Does the data seem to follow a normal distribution?
  - use a probability plot to check
- Check whether a Weibull distribution fits better
- Predict the highest wind speed expected in a 10-year interval



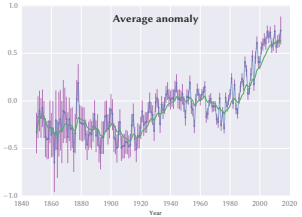




Data downloaded from `wunderground.com/history/monthly/fr/blagnac/LFBO`

Analyze HadCRUT4 data on global temperature change

- HadCRUT4 is a gridded dataset of global historical surface temperature anomalies relative to a 1961-1990 reference period
- Data available from `metoffice.gov.uk/hadobs/hadcrut4/`
- **Exercise**: import and plot the northern hemisphere ensemble median time series data, including uncertainty intervals



RISK ENGINEERING

Image credits

- Microscope on slide 41 adapted from `flic.kr/p/aeh1J5`, CC BY licence

For more free content on risk engineering, visit `risk-engineering.org`

RISK ENGINEERING

Feedback welcome!

Was some of the content unclear? Which parts were most useful to you? Your comments to `feedback@risk-engineering.org` (email) or `@LearnRiskEng` (Twitter) will help us to improve these materials. Thanks!

`@LearnRiskEng`

`fb.me/RiskEngineering`

For more free content on risk engineering, visit `risk-engineering.org`