Statistical inference for (Python) Data Analysis.

An introduction

Piotr Milanowski

daftcode

Statistical inference?

Wait, why?

- Quantify a level of trust for values you obtain
- Compare values
- Infer validity of provided data

Buzz phrases

for this talk

- Probability
- Distribution
- Random variable
- Significance
- Hypothesis testing
- Statistic

Part 1
Preparation

Building Python statistical stack

Neccesary modules:

- Numpy
- Scipy

Helpful modules:

- Pandas
- Matplotlib

NumPy

http://www.numpy.org

Numerical library optimized

for speed and memory efficiency

Many **useful and intuitive functionalities**, and methods

(especially for multidimensional arrays)

Python

```
>>> # Vector
>>> v = [1, 2, 3, 4]
>>> # scaling vector 2v
>>> v2 = [2*i for i in v]
>>> # Adding vectors v+v2
>>> v3 = [v[i]+v2[i] for i in range(len(v))]
>>> # Vector normalization
>>> mean = sum(v)/len(v)
>>> zero_mean = [(i - mean) for i in v]
>>> std = sum(i**2 for i in zero_mean)/len(v)
>>> normalized = [i/std for i in zero_mean]
```

Python + NumPy

```
>>> import numpy as np
>>> # Vector
>>> v = np.array([1, 2, 3, 4])
>>> # sacling vector 2v
>>> v2 = 2*v
>>> # Adding vectors v+v2
>>> v3 = v2 + v
>>> # Normalization
>>> normalized = v.mean()/v.std()
```

SciPy

http://www.scipy.org

A set of scientific libraries for signal analysis (scipy.signal), image analysis (scipy.ndimage), Fourier transform (scipy.fftpack), linear algebra (scipy.linalg), integration (scipy.integrate).....

Here: scipy.stats

)

Pandas & Matplotlib

http://pandas.pydata.org

Great datastructures with helpful methods

http://matplotlib.org

Visualization library

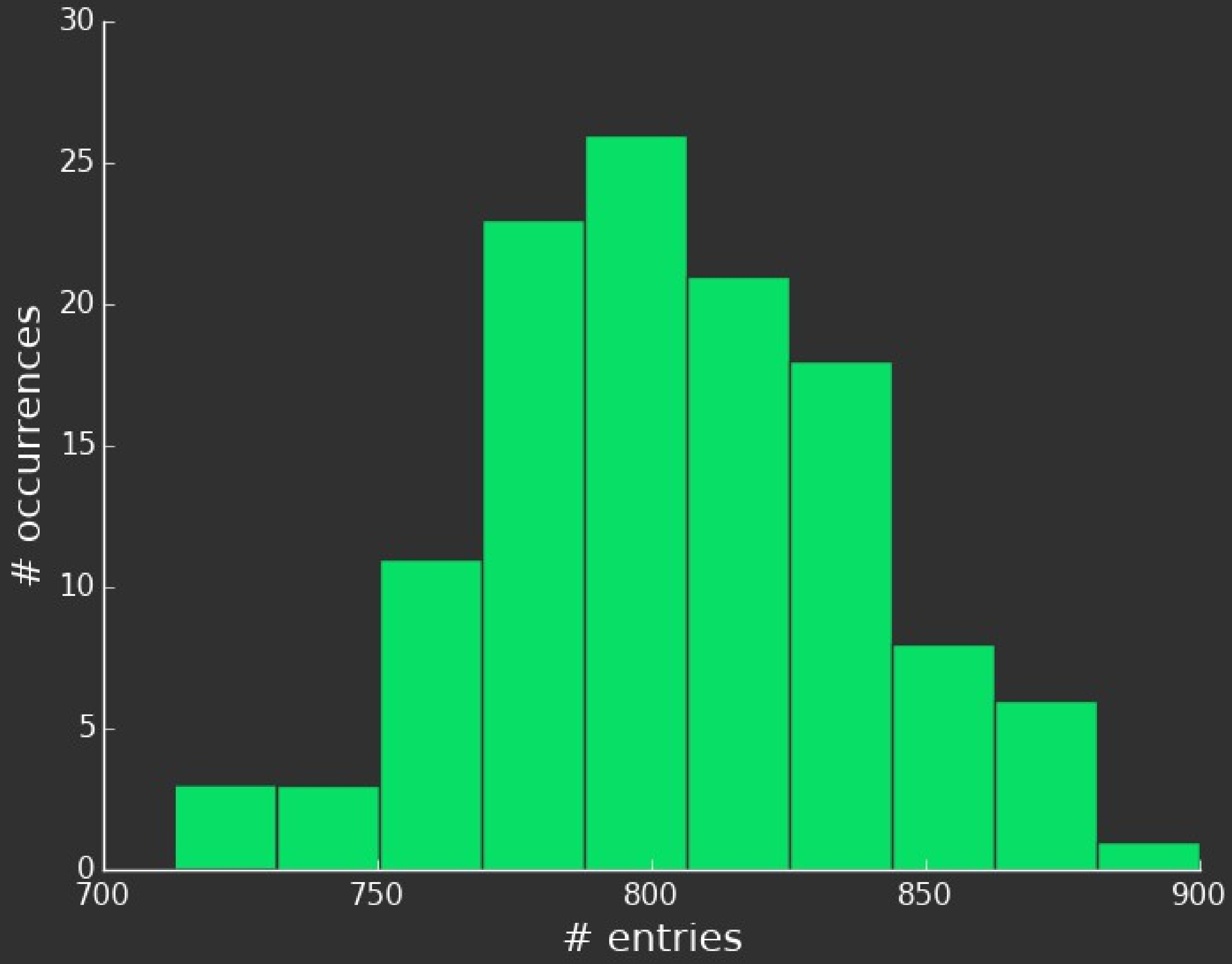
Part 2
Examples

Example 1
Anomaly detection

Data:
number of daily page entries from 3 months

Question:
should we be suspicious if for a given day we have 800, 850 and 900 entries?

```
>>> import numpy as np
>>> values = np.array([...]) # Get values from somewhere
>>> values array([796, 756, 827, 779, 787, 731, 847,
829, 794, 790, 831, 820, 822, 792, 767, 757, 777, 775,
799, 756, 849, 808, 800, 823, 798, 809, 783, 800, 848,
772, 853, 759, 745, 868, 713, 863, 819, 829, 799, 777,
785, 812, 877, 839, 783, 867, 875, 829, 818, 813, 763,
829, 812, 867, 790, 748, 822, 774, 900, 830, 781, 795,
825, 799, 795, 839, 840, 765, 781, 781, 797, 821, 852,
836, 811, 771, 800, 752, 776, 755, 798, 839, 821, 794,
846, 834, 825, 825, 830, 814, 839, 760, 786, 747, 803,
717, 801, 819, 789, 824, 835, 806, 858, 774, 848, 793,
826, 796, 798, 773, 779, 775, 779, 806, 768, 787, 788,
822, 843, 780])
>>> values.max(), values.min()
(900, 713)
```


30
25
20
15
10
5
0
700
750
800
850
900
# occurrences
# entries

Example 1
Anomaly detection

- **Assumption:** values are drawn from Poisson distribution
- What is the probability of obtaining 800, 850, 900 for Poisson distribution fitted to this data?
- What is threshold value?
- scipy.stats.poisson (and many other distributions)

```
>>> import scipy.stats as ss
>>> # Calculating distribution parameter
>>> mu = values.mean()
>>> # Check for 800
>>> 1 - ss.poisson.cdf(800, mu) # equal to
ss.poisson.sf(800, mu) 0.548801
>>> # Check for 900
>>> 1 - ss.poisson.cdf(900, mu)
0.00042
>>> # Check for 850
>>> 1 - ss.poisson.cdf(850, mu)
0.05205

>>> # Threshold for magical 5%
>>> ss.poisson.ppf(0.95, mu)
851
```

Example 2
Trend analysis

Data

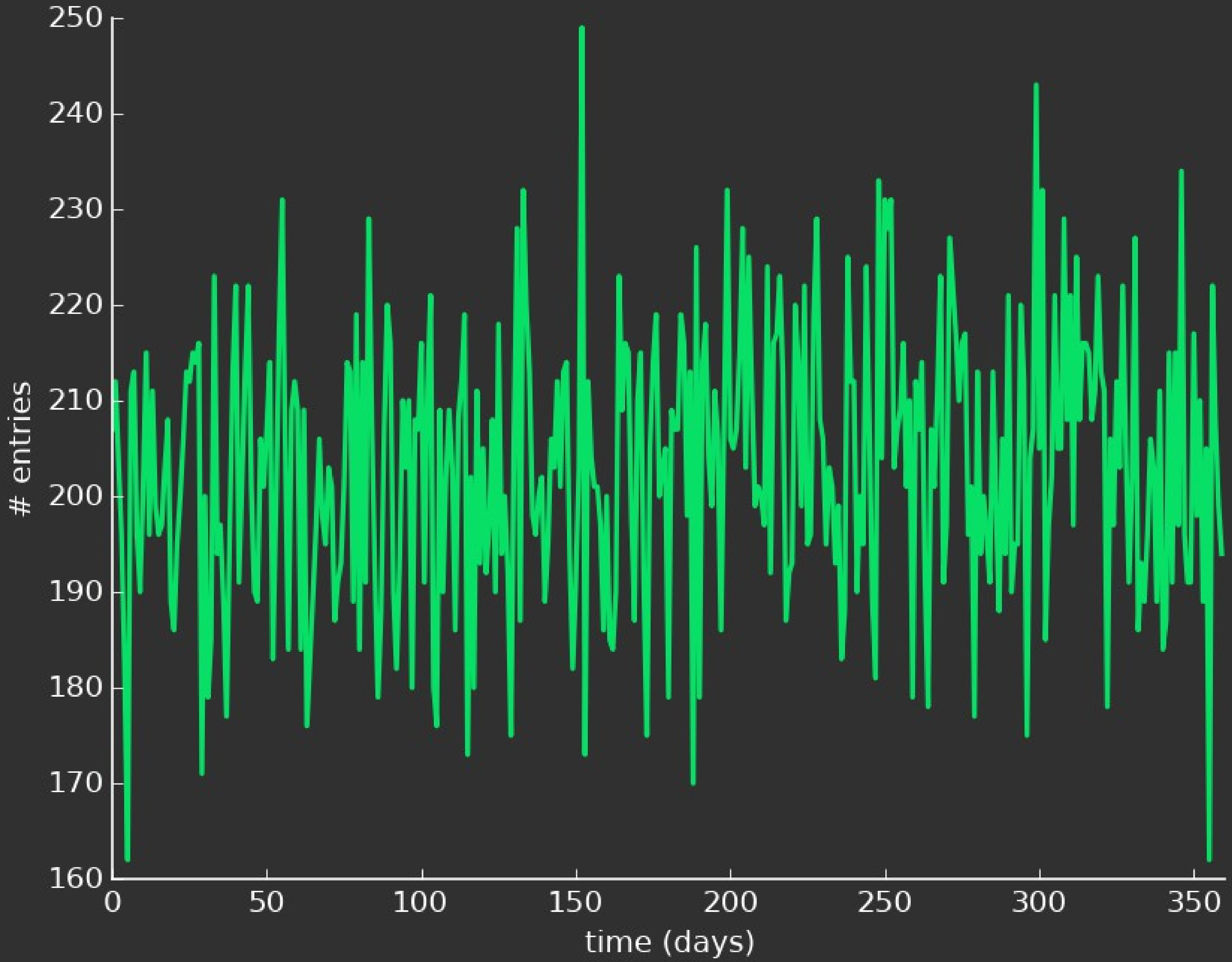
Yearly number of entries

Question

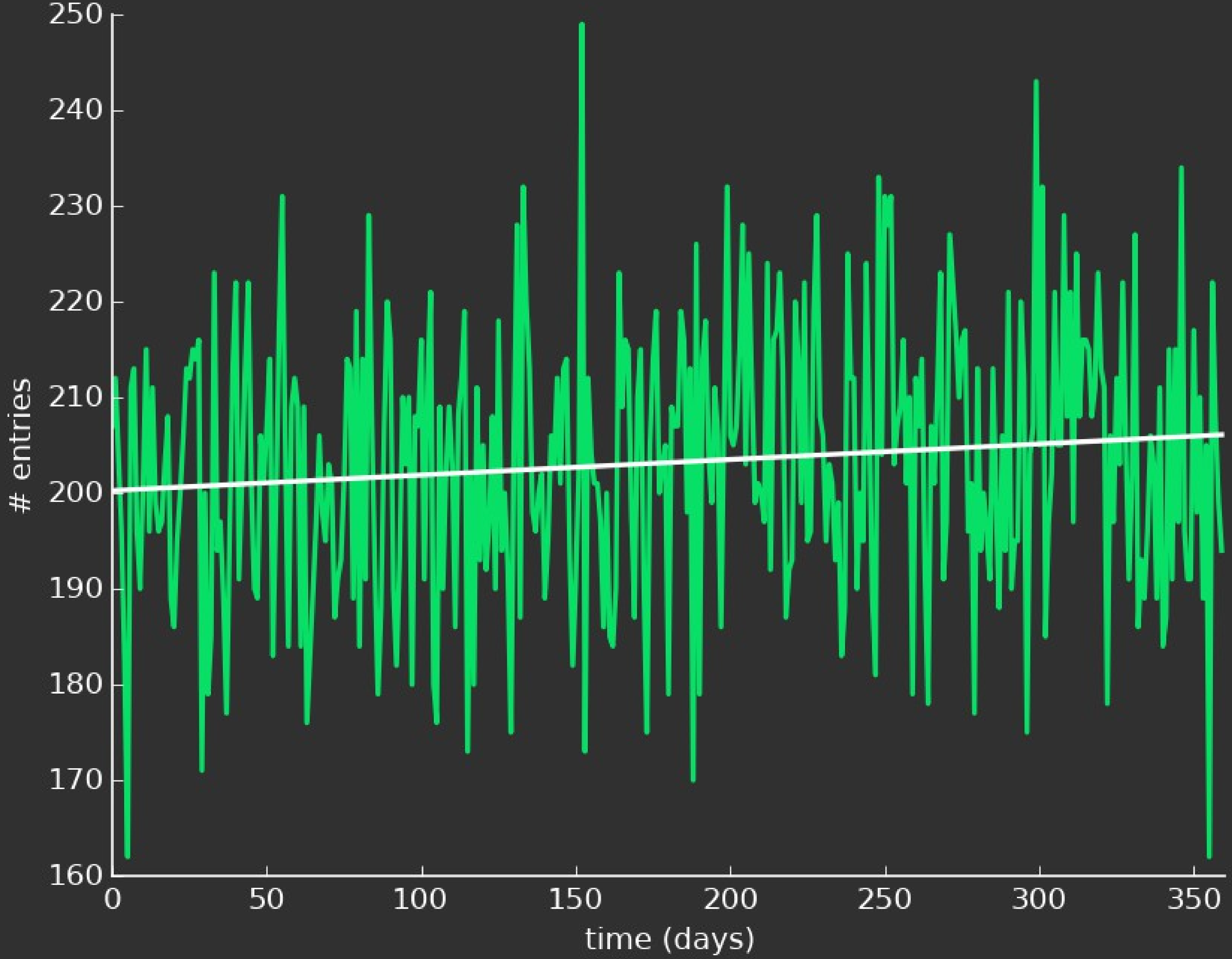
What is the trend of given data?

Is this trend caused by model properies or by some noise?

```
>>> import numpy as np.
>>> data = np.array([…]) #data read from somwhere
>>> a, b, left_c1, right_ci
>>> def theilsen(y, x=None):
...     N = len(y)
...     a = []
...     if x is None:
...         x = np.arange(len(y))
...     for i in range(1, N):
...         for j in range(i):
...             if x[i] != x[j]:
...                 a.append((y[i] – y[j])/(x[i] - x[j]))
...     return np.median(a)
>>> a = theilsen(data)
>>> b = np.median(data) – a*np.median(np.range(len(data)))
>>> print(a, b)
0.01624 200.08366
```



# entries
time (days)
250
240
230
220
210
200
190
180
170
160
0
50
100
150
200
250
300
350


250
240
230
220
210
200
190
180
170
160
# entries
0
50
100
150
200
250
300
350
time (days)

Example 2
Trend analysis

- Is this significant?
- Calculate 95% confidence intervals for the slope. Is it all positive/negative

```
>>> # Repeate slope calculation for resampled data
>>> N = len(data)
>>> no_reps = 1000
>>> slopes = np.zeros(no_reps)
>>> x = np.arange(N)
>>> for i in range(no_reps):
...     idx = np.random.randint(0, N, size=N)
...     new_x = x[idx]
...     new_y = data[idx]
...     slopes[i] = theilsen(new_data)
>>> left_ci, right_ci = np.percentile(slopes, [2.5, 97.5])
>>> print(left_ci, right_ci)

0.0050666094707 0.0267216613314

>>> # Or use buildin functionality
>>> a, b, left_ci, right_ci = ss.theilslopes(data)
>>> print(left_ci, right_ci)
0.0 0.03048780487804878
```

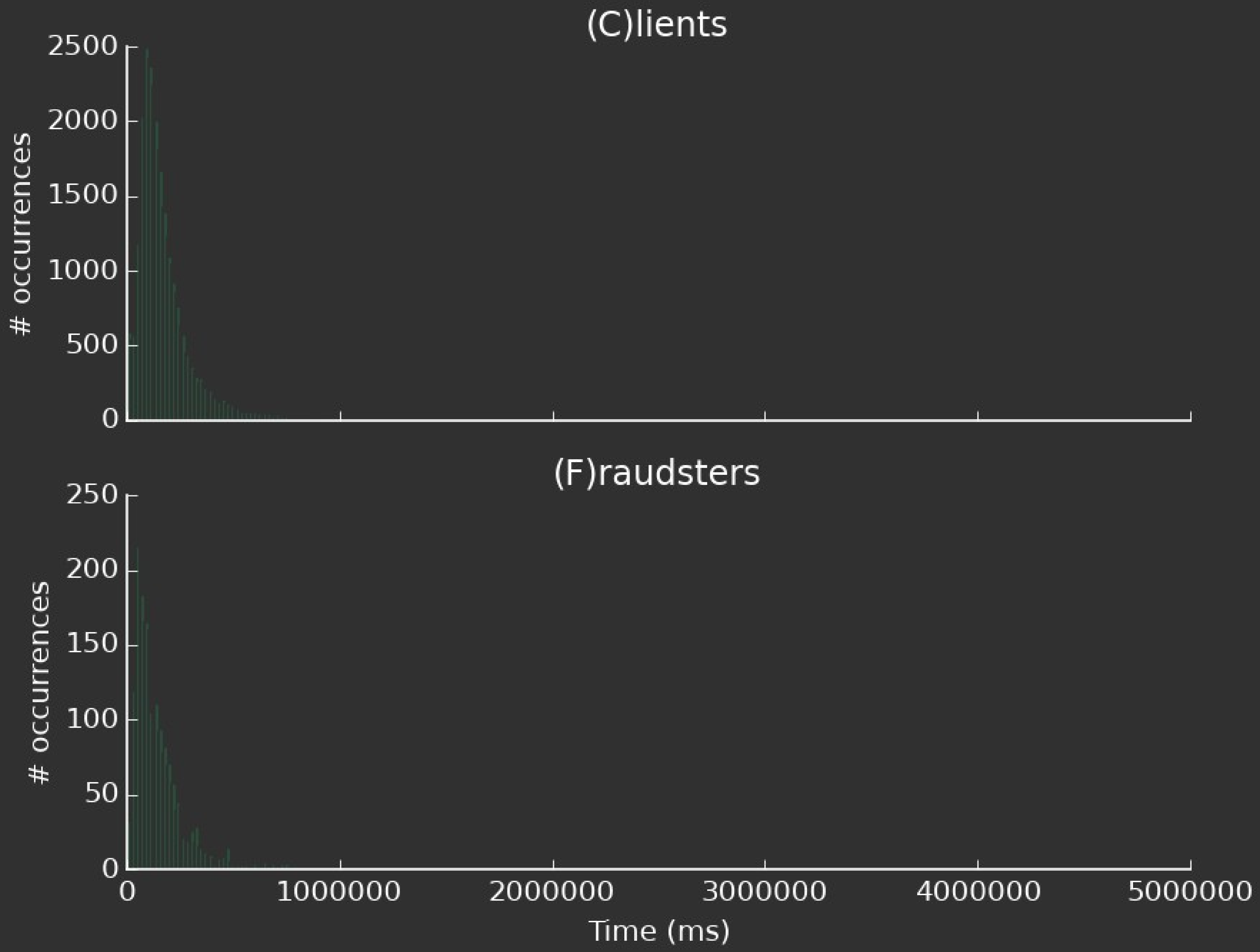
Example 3
Comparing distributions

Data

two sets of time spent on time – one set for fraud data (F), and second for non-fraud data (C)

Question

is there a (significant) difference in those two distributions?


(C)lients
# occurrences
2500
2000
1500
1000
500
0
(F)raudsters
# occurrences
250
200
150
100
50
0
0
1000000
2000000
3000000
4000000
5000000
Time (ms)

```
>>> ok = np.array(ok) # non-fraud
>>> fraud = np.array(fraud)
>>> np.median(ok)
140261.0
>>> np.median(fraud)
109883.0
```

Unknown distributions:
nonparametric test

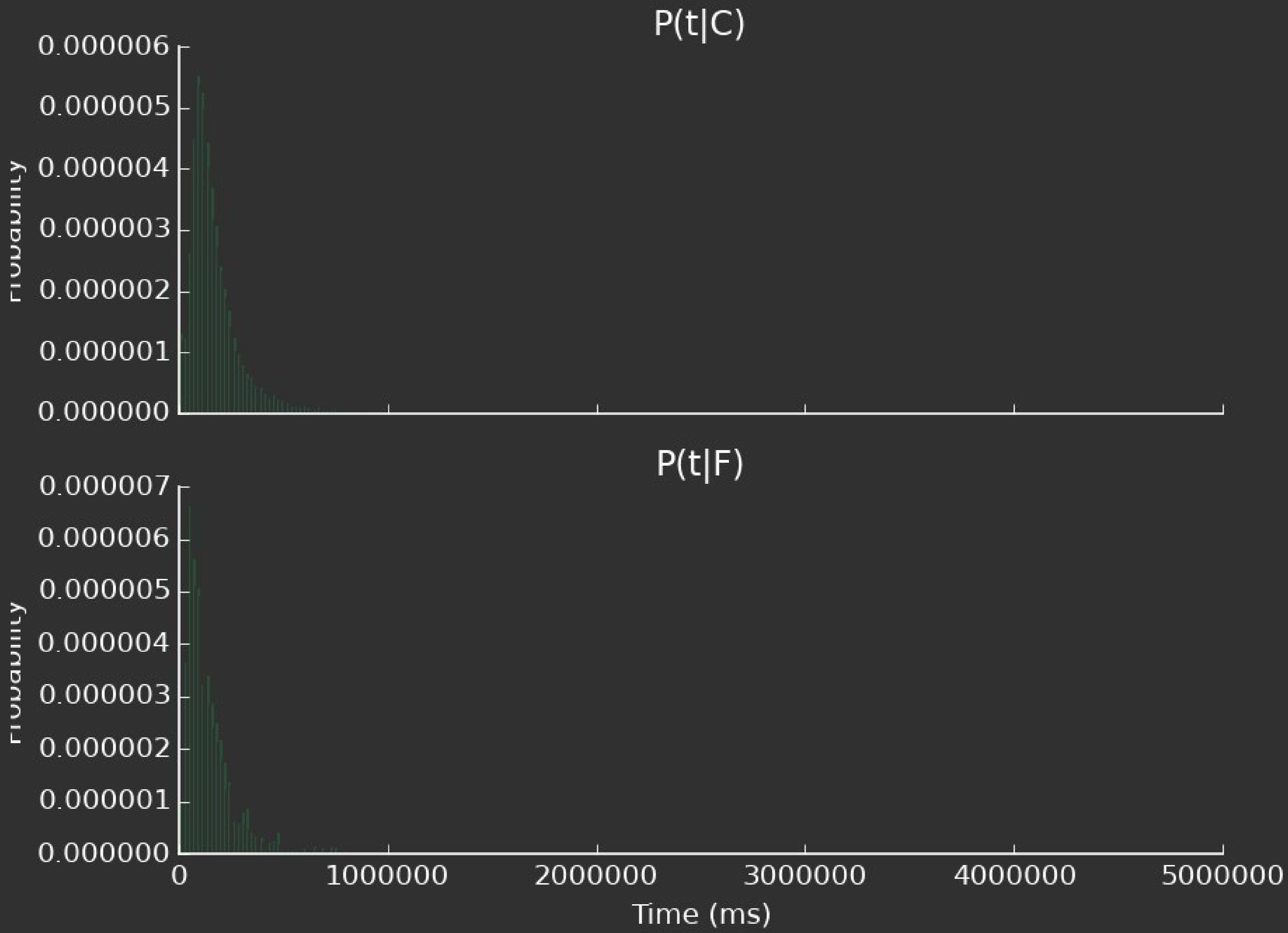
```
>>> ss.mannwhitneyu(ok, fraud)
MannwhitneyuResuls(statistic=54457079.5,
pvalue=1.05701588547616e-59)
```

Equalize sample sizes
(just to be sure)

```
>>> N = len(fraud)
>>> idx = np.arange(0, len(ok))
>>> np.random.shuffle(idx)
>>> ok_subsample = ok[idx[:N]]
>>> ss.mannwhitneyu(ok_subsample, fraud)
>>> MannwhitneyuResult(statistic=3548976.0,
pvalue=3.1818273295679098e-30)
```

Example 4
Naive Bayes

- **Question:** Can we classify fraud based on time spent on a page?
- Use Naive Bayes: $P(F|t) \sim P(t|F)P(F)$ $P(C|t) \sim P(t|C)P(C)$
- $P(t|F)$, $P(t|C)$ are sample distributions $P(C)$, $P(F)$



P(t|C)
P(t|F)
Probability
0.000000
0.000001
0.000002
0.000003
0.000004
0.000005
0.000006
0.000007
0
1000000
2000000
3000000
4000000
5000000
Time (ms)

```
>>>  # Retrieve probability based on histogram
>>>  def get_prob(value, bins, probs):
...       idx = np.searchsorted(bins)
...       return probs[idx]
>>>  N = len(ok) + len(fraud)
>>>  P_ok = len(ok)/N
>>>  P_fraud = len(fraud)/N
>>>  probs_ok, bins_ok = np.histogram(ok, bins=500, normed=True)
>>>  probs_fraud, bins_fraud = np.histogram(fraud, bins=bins_ok,
normed=True)

>>>  # Test!
>>>  test_times = np.linspace(0, 1e6, 100000)
>>>  detected_as_fraud = 0
>>>  detected_as_ok = 0
>>>  for el in test_times:
...       p_ok = get_prob(el, bins_ok, probs_ok)*P_ok
...       p_fr = get_prob(el, bins_fraud, probs_fraud)*P_fraud
...       if p_ok > p_fr:
...           detected_as_ok += 1
...       else:
...           detected_as_fraud += 1
>>> print(detected_as_ok, detected_as_fraud)
100000 0
```

Example 4
Naive Bayes

- NB doesn't seem to work that well
in this example
- Better results by just putting a threshold
- But still, several lines of code
and classifier ready!

Almost at the end. Just one more slide... and it's a summary!

Summary

- Statistical inference is used to compare and validate values
- It gives some quantification, but there still is a room for subjective decisions (p-values, priors)
- It is quite easy to do statistics in Python when you have proper tools