Exploratory Data Analysis

Milan Vojnovic

ST445 Managing and Visualizing Data

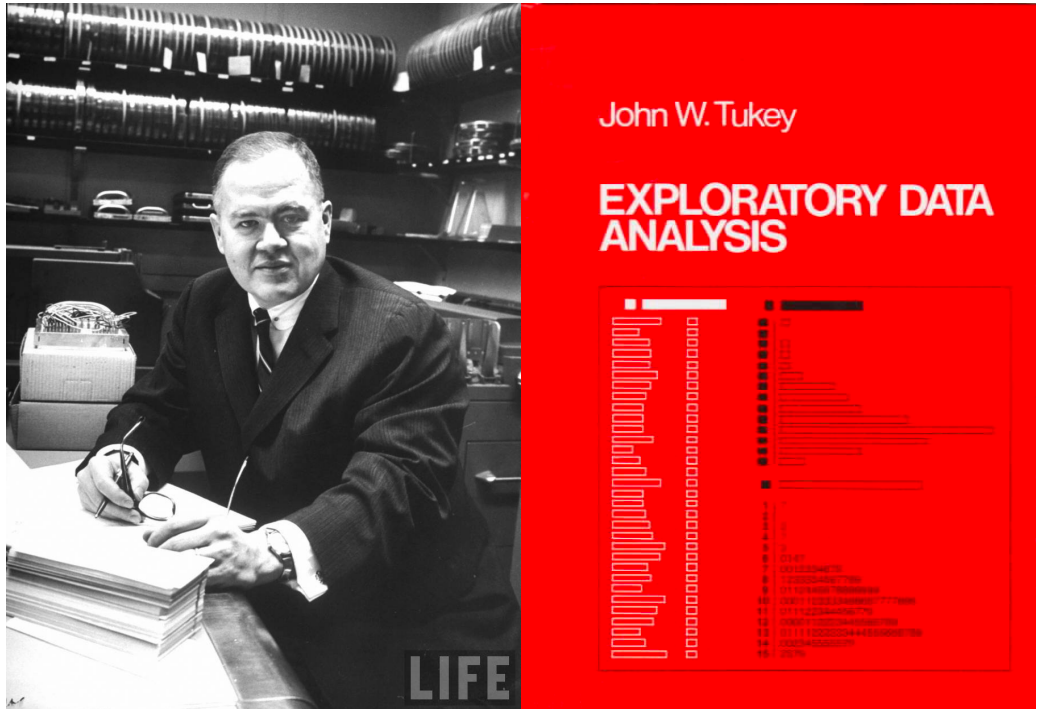
Exploratory Data Analysis (EDA)

- An approach to analyzing datasets to summarize their main characteristics, often using visual methods
- A statistical model may or may not be used
- It is about what data tell us beyond the formal modeling or hypothesis testing

What does EDA refer to?

"[P]rocedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data."

Tukey, 1961



Objectives of EDA

- Suggest hypothesis about the causes of observed phenomena
- Assess assumptions on which statistical inference will be based
- Support the selection of appropriate statistical tools and techniques
- Provide a basis for further data collection through surveys and experiments

The scope of this lecture

- Overview of standard statistical plots used in EDA
- Learning best data visualization practices
- Based on using Python software libraries Matplotlib and Seaborn
- Based on studying a concrete dataset from GitHub archive

Python plotting libraries

- Matplotlib: basic plotting library
  - https://matplotlib.org

- Seaborn: statistical data visualization
  - Python data visualization library based on matplotlib
  - High-level interface for statistical graphics
  - http://seaborn.pydata.org

Useful resources

- Matplotlib tutorial: https://matplotlib.org/users/pyplot_tutorial.html
- Seaborn tutorial: http://seaborn.pydata.org/tutorial

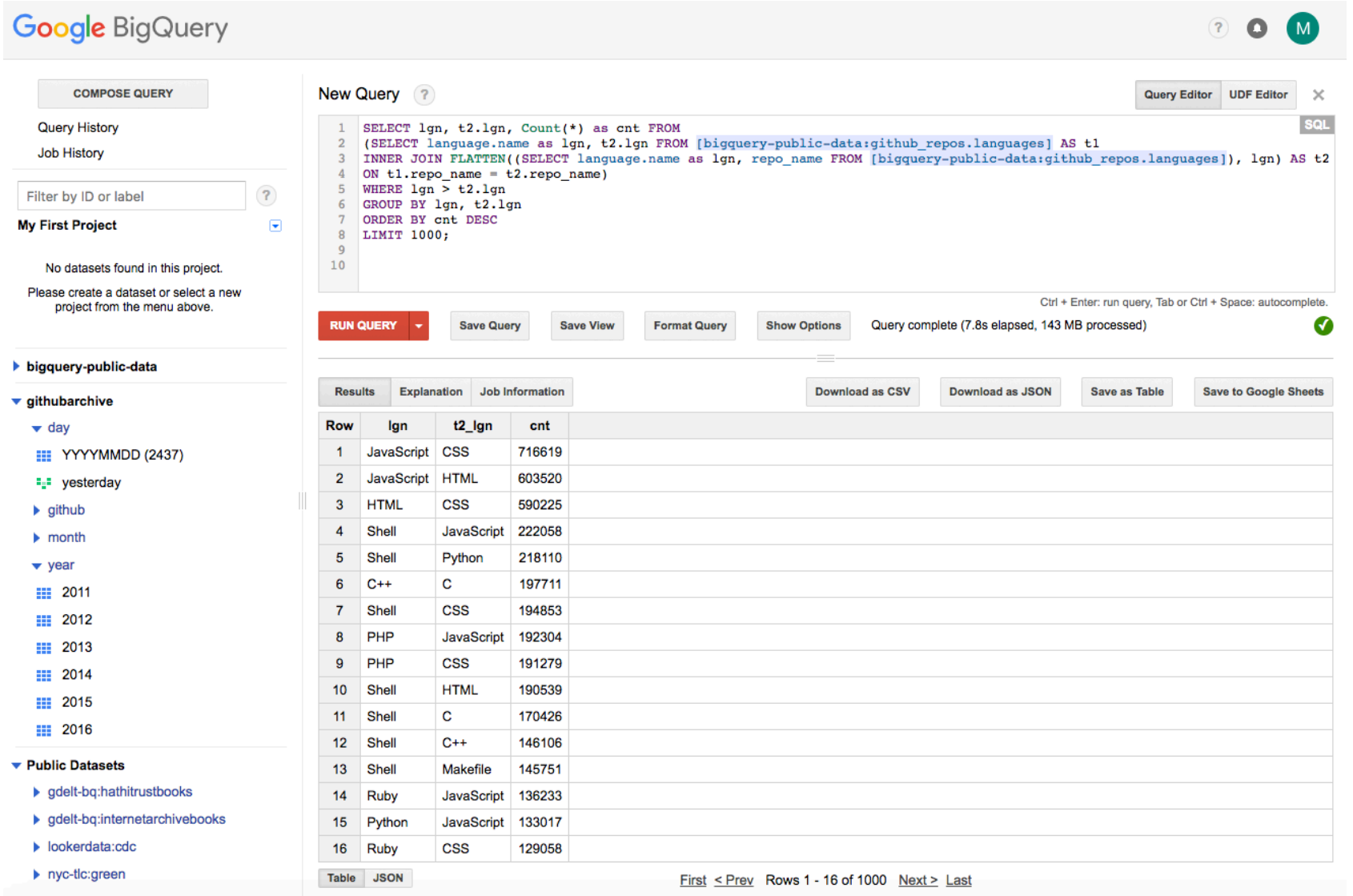
Running example: GitHub archive

- https://www.githubarchive.org
- Example: download for first Monday of March 2015 from from 15:00 to 16:00

```
curl http://data.githubarchive.org/2015-03-02-15.json.gz -o
2015-03-02-15.json.gz
```

Also available from Google BigQuery

https://cloud.google.com/bigquery

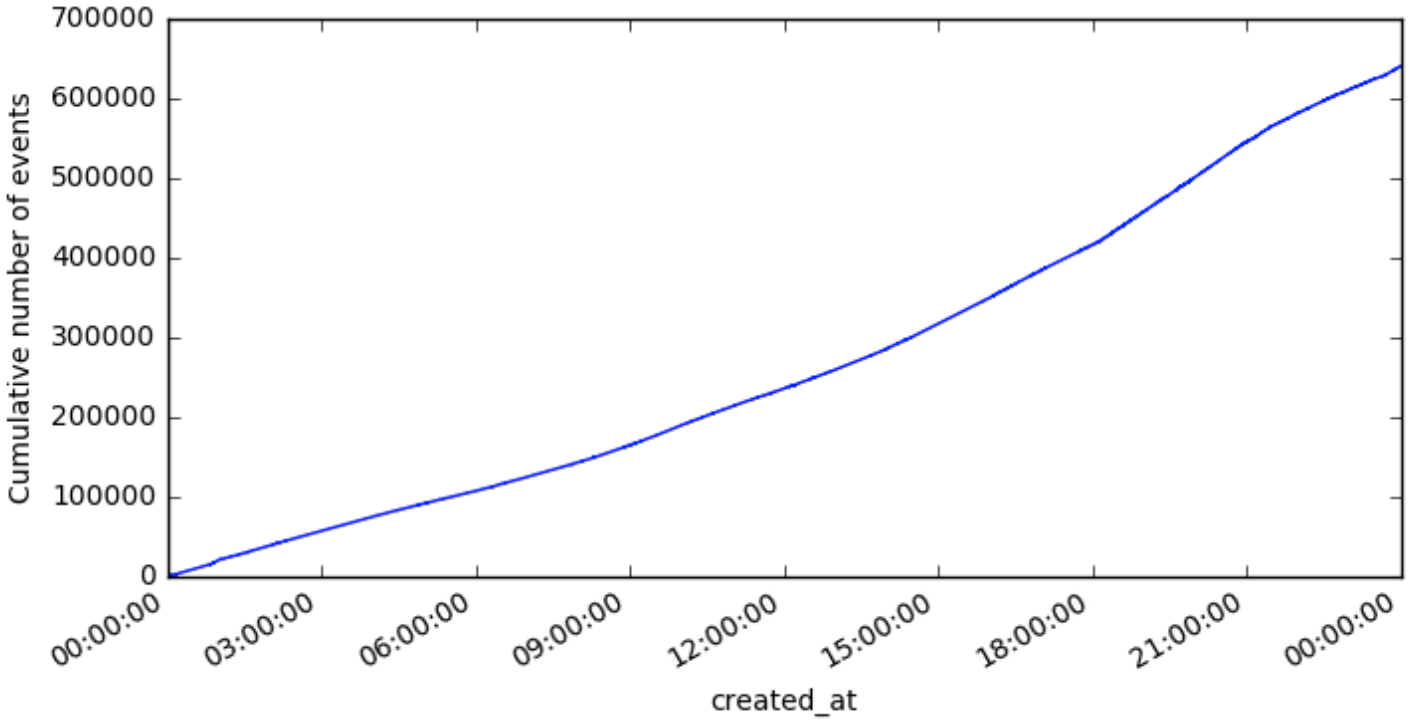


Dataframe

	created_at	id	actor.login	org.login	repo.name	type
0	2015-03-02 16:00:00	2617515383	instructure-gerrit	instructure	instructure/ruby-saml2	PushEvent
1	2015-03-02 16:00:00	2617515392	supby	NaN	supby/rtmonsystem	PushEvent
2	2015-03-02 16:00:00	2617515393	mikelohmann	DECK36	DECK36/drupal-fixtures	PushEvent
3	2015-03-02 16:00:01	2617515402	marqh	metarelate	metarelate/metOcean	IssuesEvent
4	2015-03-02 16:00:01	2617515404	edgarshurtado	udacity	udacity/create-your-own-adventure	PullRequestEvent

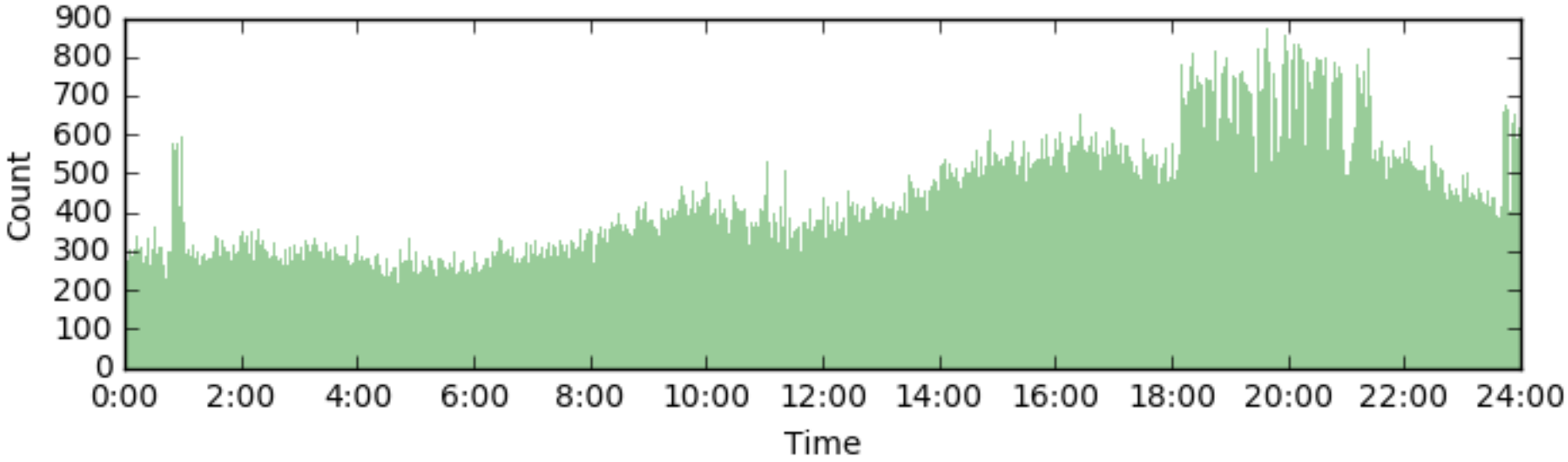
- This is the filtered dataset that we are going to explore

Cumulative events count



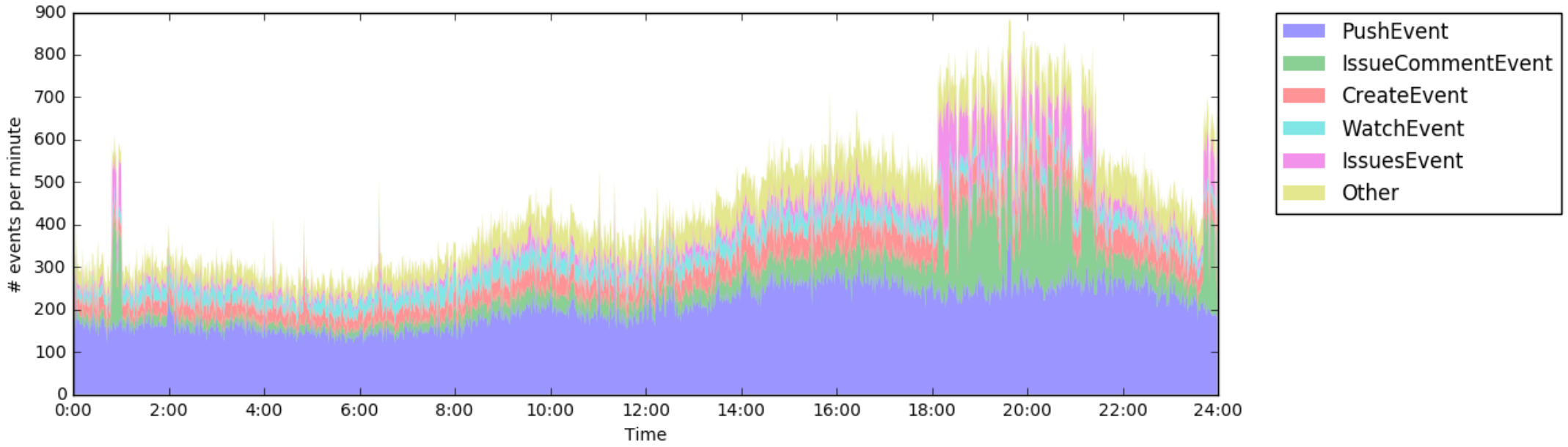
- Time series plot easily produced by using plotting methods of pandas `Series`
- The plot indicates some time of day effects
- Time is UTC time

Differenced time series



- The number of events over regular 1 minute time intervals
- Such a plot can be produced using pyplot's function `hist`

Time series conditional on event type

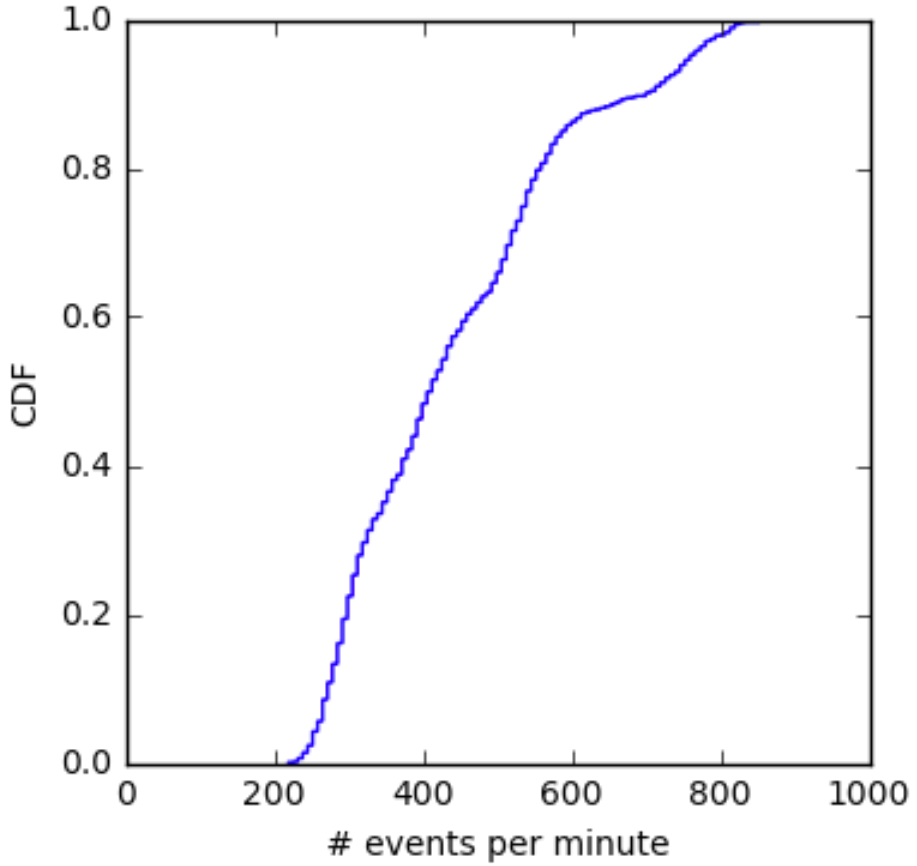


- Same plot as on previous slide but conditioned on the event type
- Such a plot can be produced with the pyplot's function `stackplot`
- We observe that certain event types have more pronounced frequency change

Distribution plots

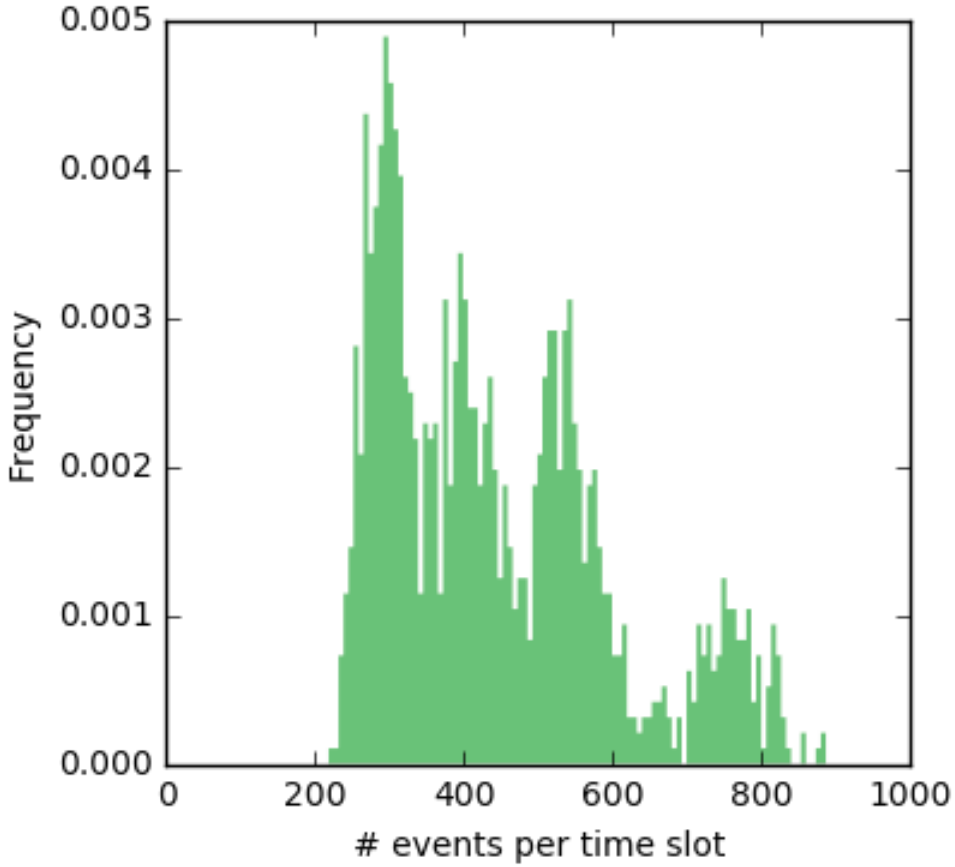
- Histograms
- Empirical cumulative distribution functions
- Data distribution summaries: boxplots and violinplots

Empirical cumulative distribution function



- Can be produced by either pyplot's function `hist` or statsmodel.api's `distribution.ECDF`

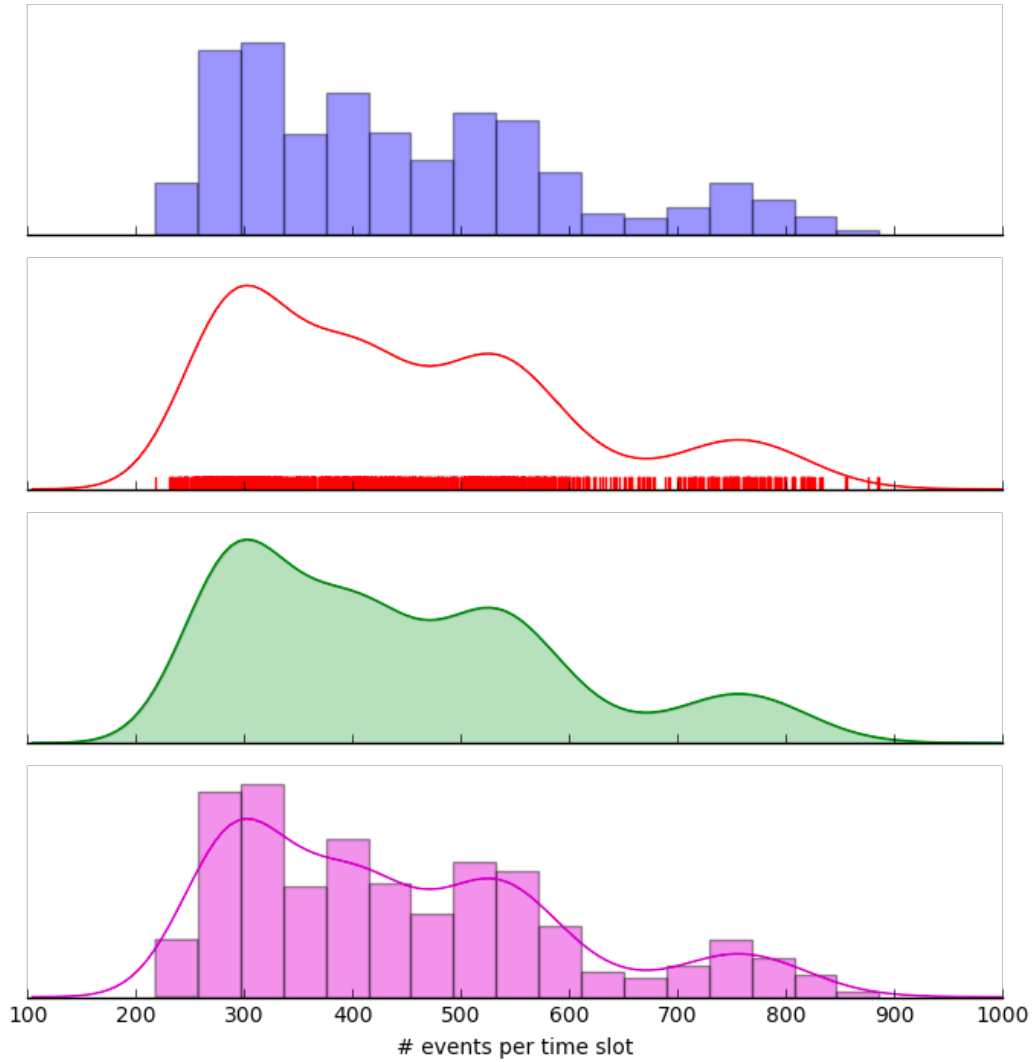
Histogram



- Also produced by pyplot's function `hist`
- The histogram is for equidistant bins with the number of bins equal to 100
- Bins can be user defined (allowing for uneven bin widths)

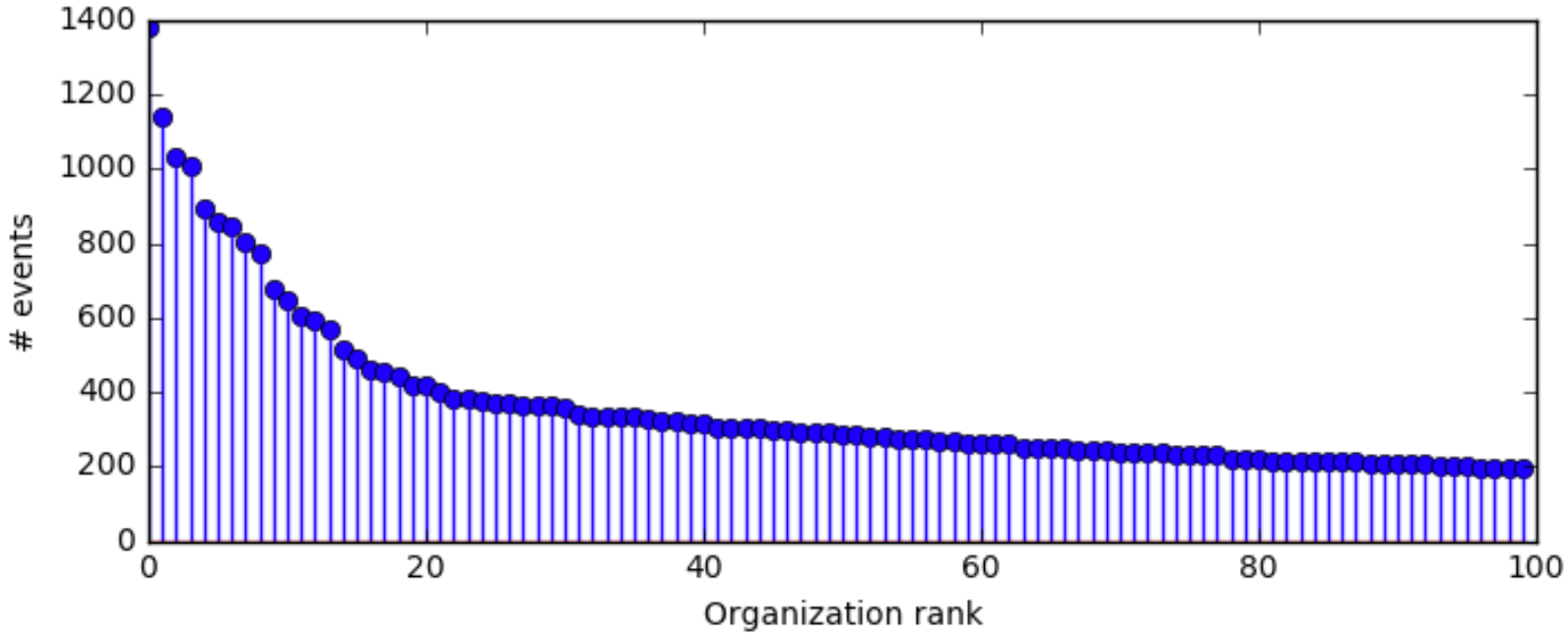
Histogram (cont'd)

- Smoothed histograms can be produced by using kernel density estimation
- The plots on this slide are produced by Seaborn's function `distplot` with suitable choice of parameters



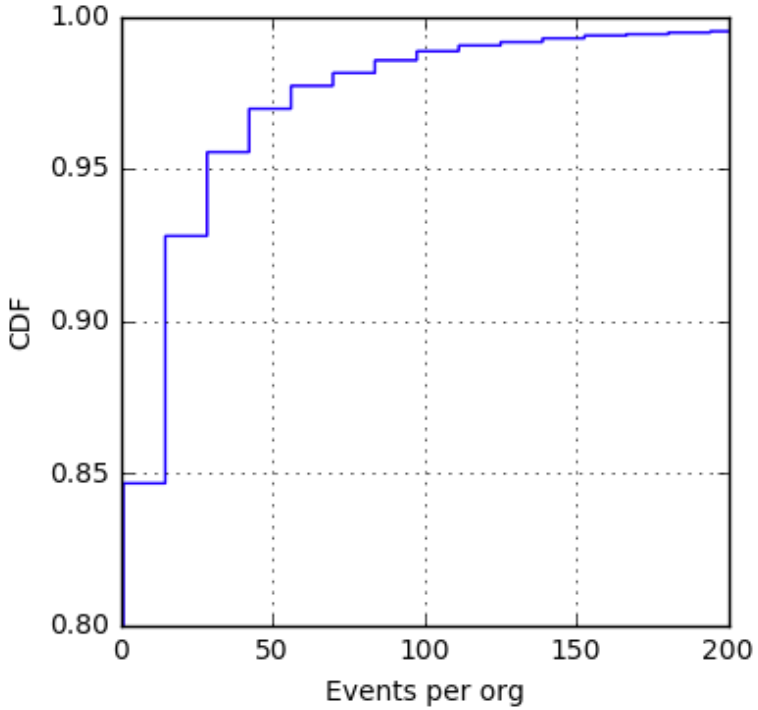
Events per organization

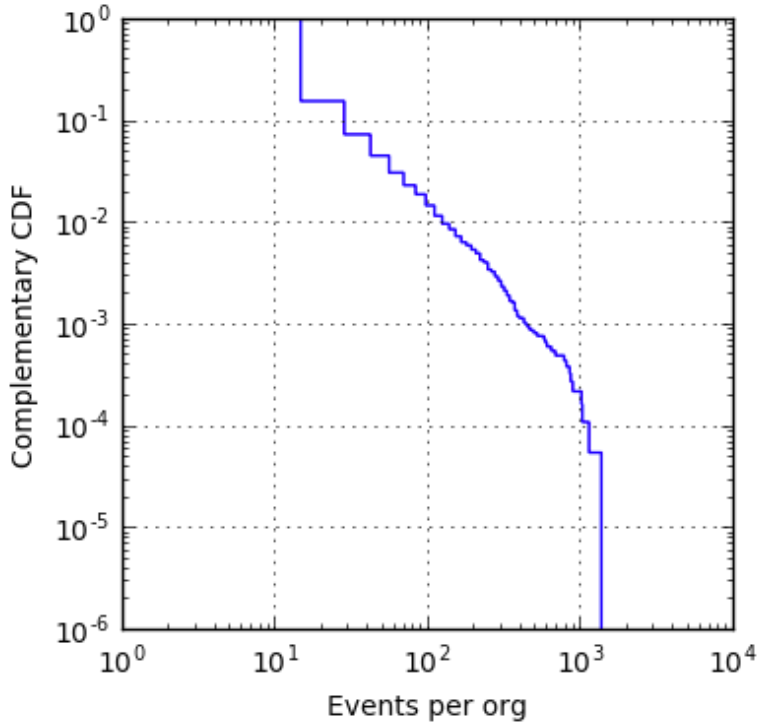
```
org.login
apache           1381
mozilla          1143
Door43           1034
sakai-mirror     1010
owncloud          897
docker            857
angular           846
facebook          802
edx               776
google            680
Name: org.login, dtype: int64
```



- Use of pyplot's function `stem`

Cumulative distribution functions

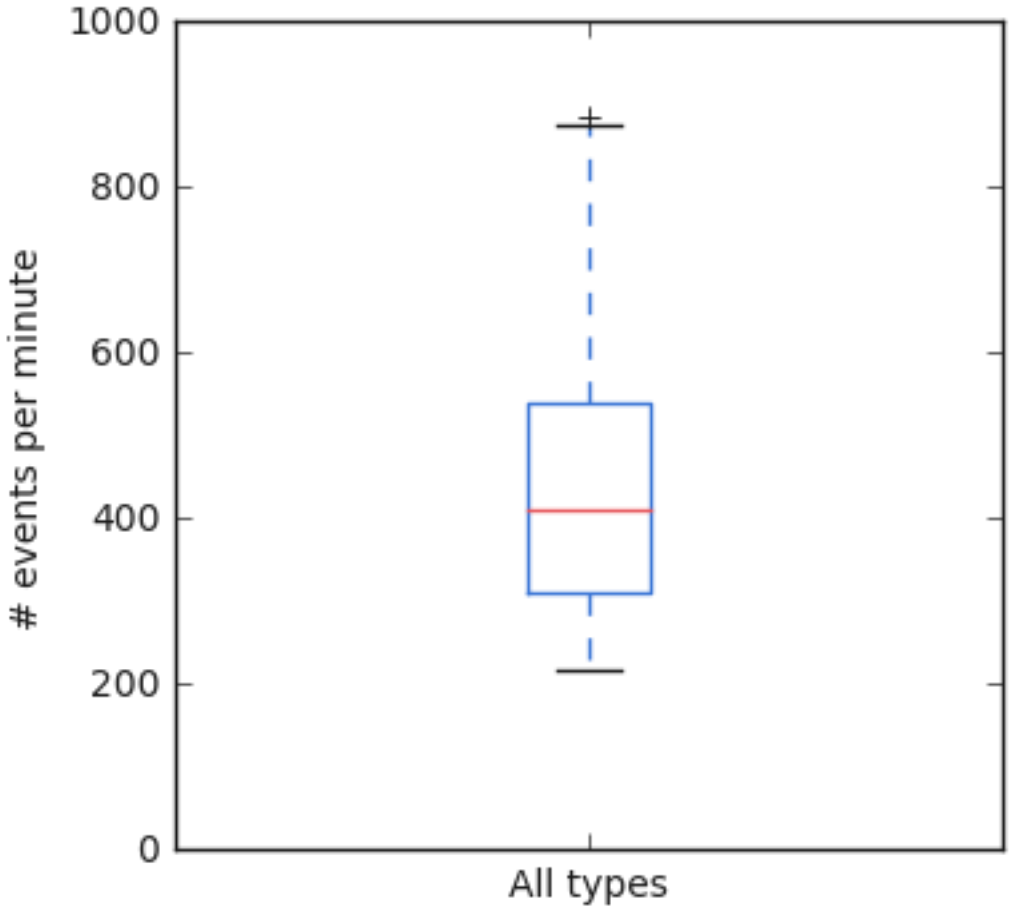




- Use appropriate scales for x and y axes
- Log-log scale to visualize a power-law distribution $\mathrm{P}[X > x] \propto 1/x^{p}$
  - Many distributions in nature exhibit a power-law (e.g. social networks)
- Q. What scale would you use to visualize a distribution with an exponential tail?

Boxplots

- A compact summary of a distribution
- Plot produced by pyplot's function `boxplot`



History of boxplots

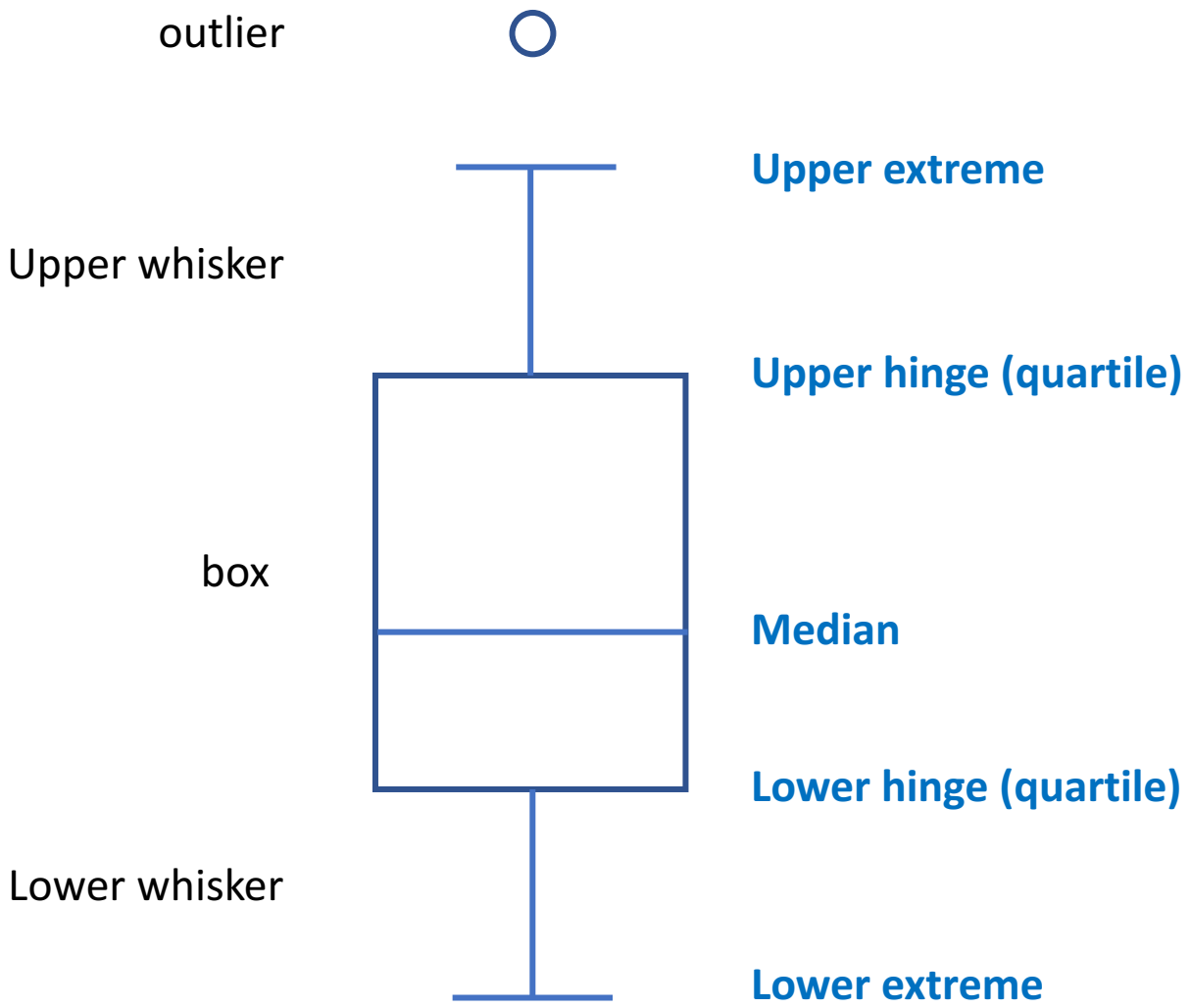
1977 Tukey, first proposal in his book

1978 Notched boxplot, proposed by McGill, Larsen and Tukey

1983 Boxplot stripped down to its essentials by Tufte
(no box shown, only lower and upper positions)

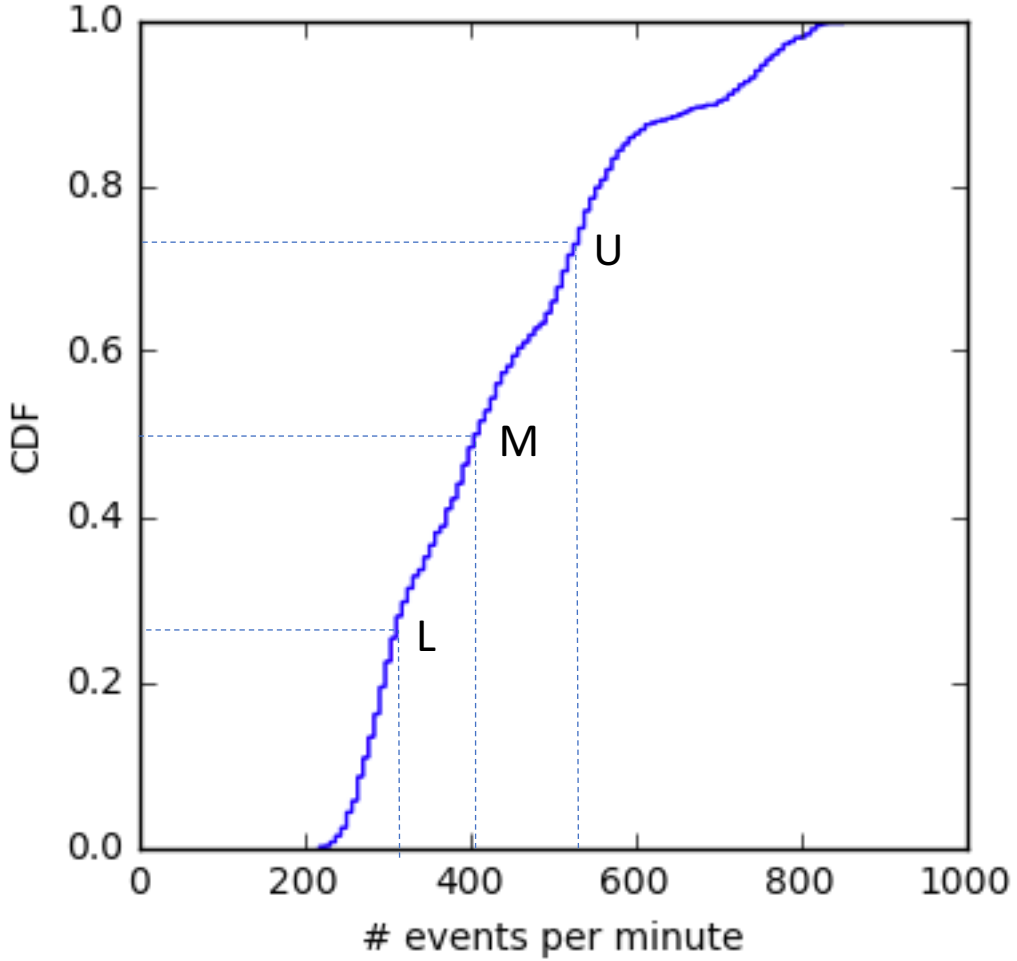
1988 Boxplot with sides used to portray density information
proposed by Benjamini

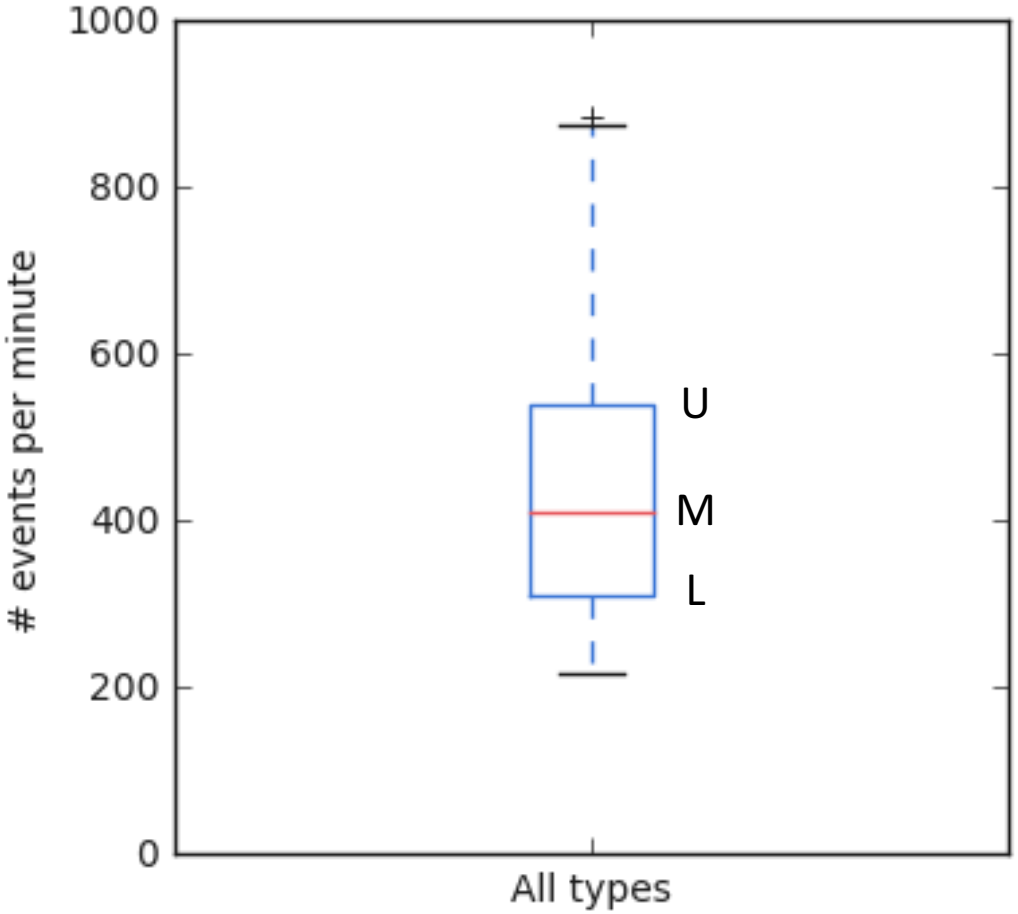
Boxplot: five number data summary



- A vertical line is extended from the middle of the top of the box to the largest observation within a step from the top
- The step is defined as 1.5x inter-quartile range
- A similarly defined line extends from the bottom of the box to the smallest observation within a step from the bottom
- Observations that are farther from the box than these lines are individually displayed (usually default setting)

Boxplot recap

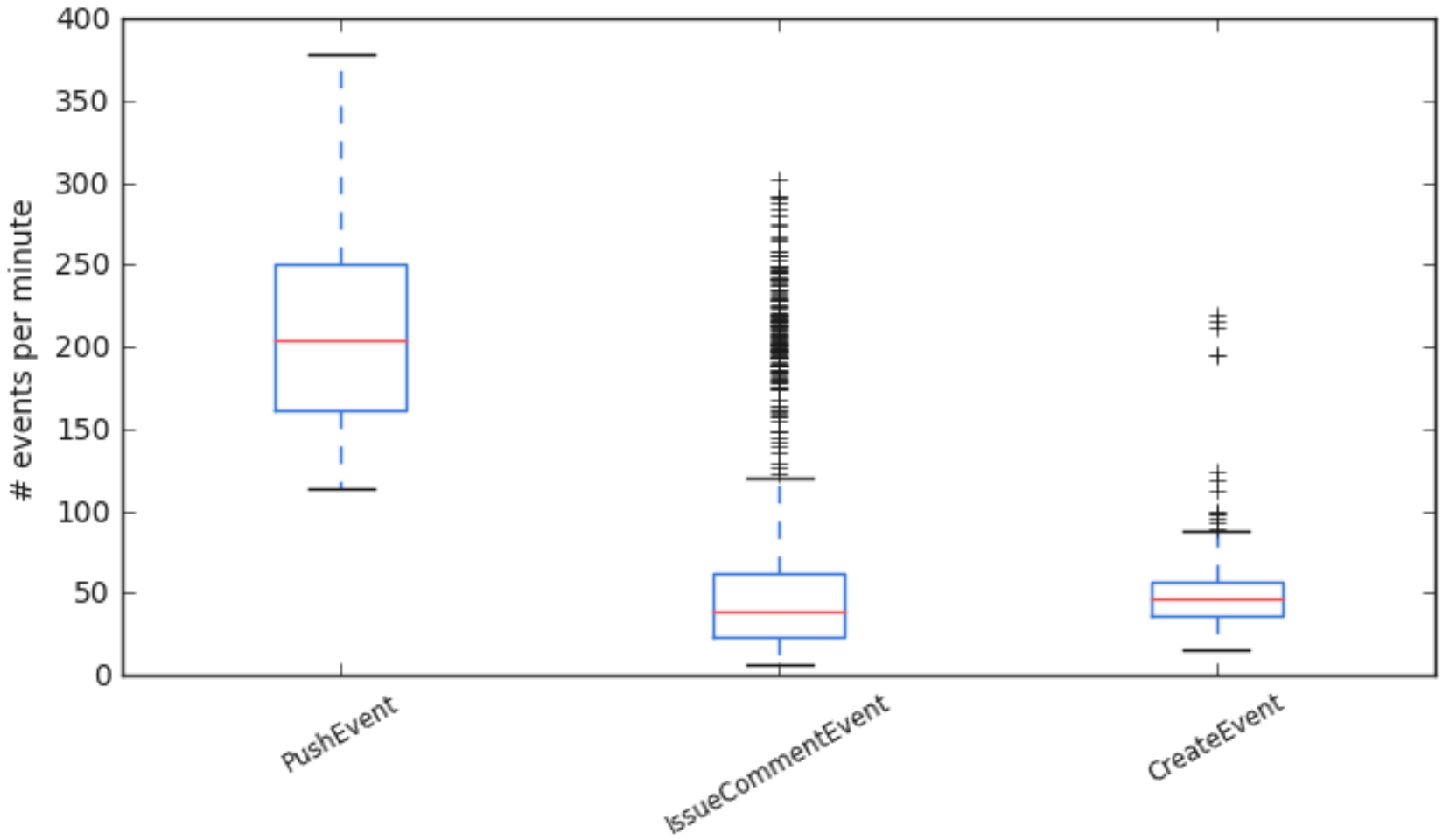




A use case: glance at distribution properties

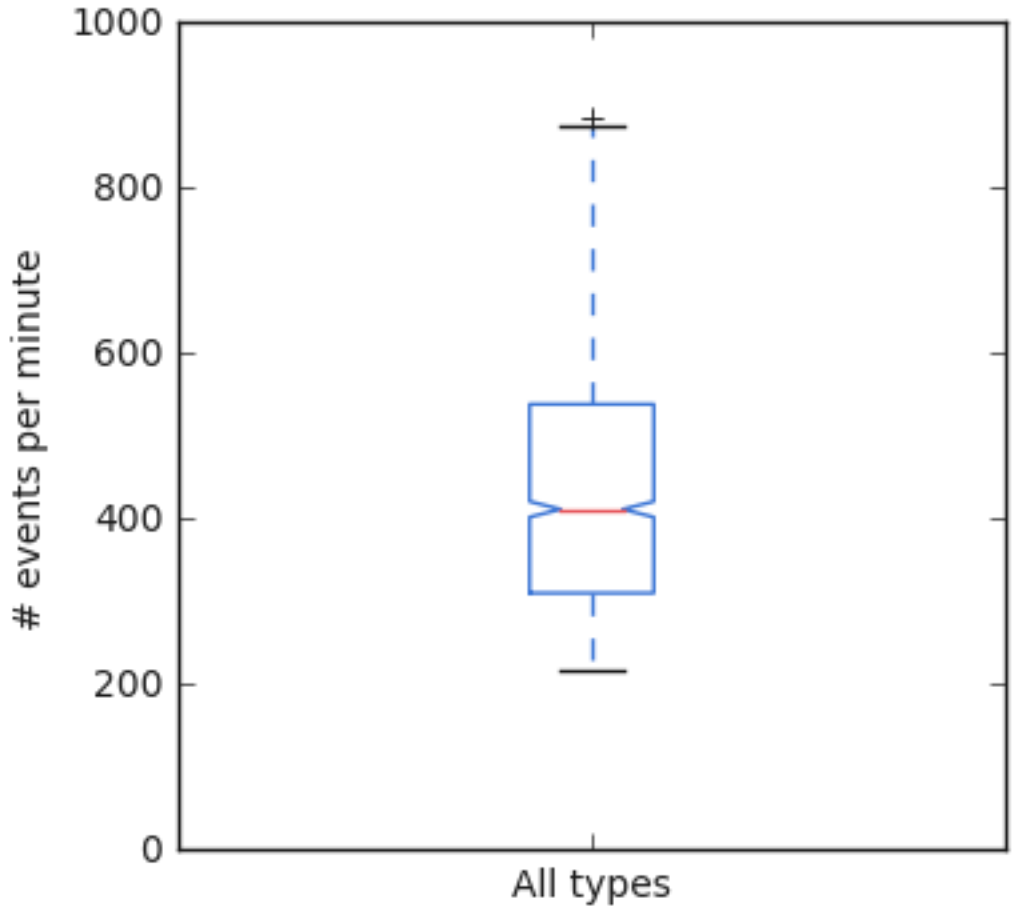
Location	Displayed by the cut line at the median and and the middle of the box
Spread	Length of the box and the distance between the ends of the whiskers and the range
Skewness	Deviation of the median line from the center of the box relative to the length of the box; by the the length of the upper whisker relative to the length of the lower one and by the number of individual observations displayed on each side
Longtailedness	Distance between the ends of the whiskers relative to the length of the box as well as by the number of observations specially marked

A use case: comparing parallel batches of data



Notched boxplot

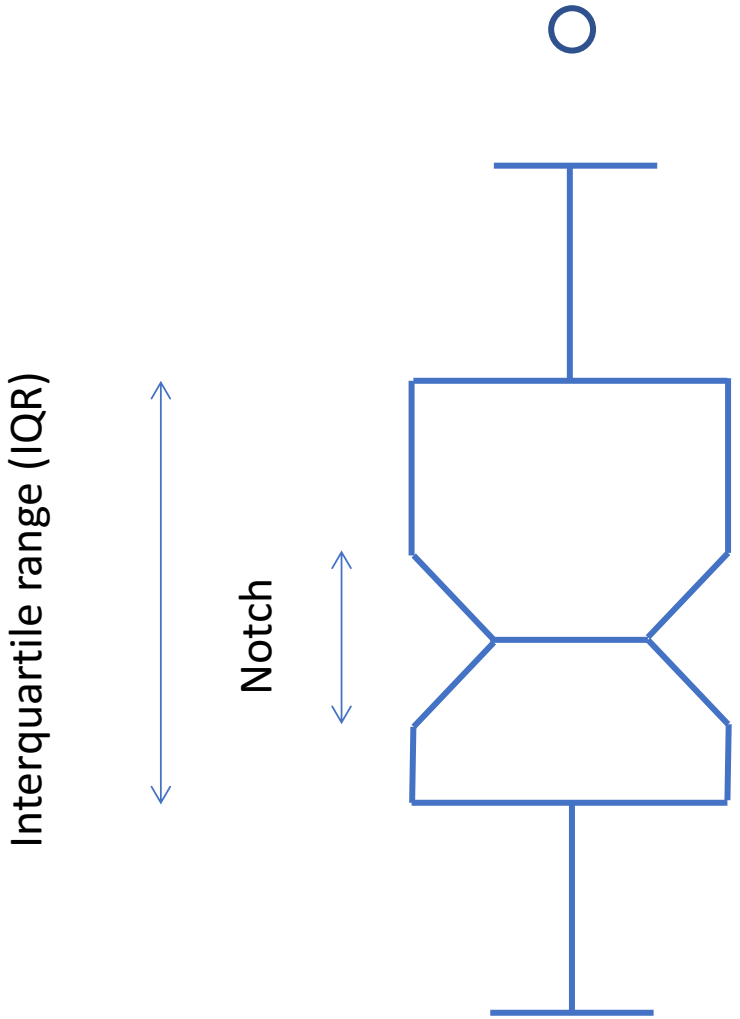
- Same boxplot as before but with a notch



Notched boxplot

- Notch interval end points:

$$\text{median} \pm 1.57\ \text{IQR}/\sqrt{n}$$



Notched boxplot

- Regular boxplot supplemented with a confidence interval for the median
- Shown as a pair of wedges taken out of the sides of the box
- Constructed in a way that when two notches of different boxplots do not overlap their medians are significantly different
- The formula for the confidence interval is a constant times the interquartile range divided by the square root of the batch size

Confidence interval of median estimate

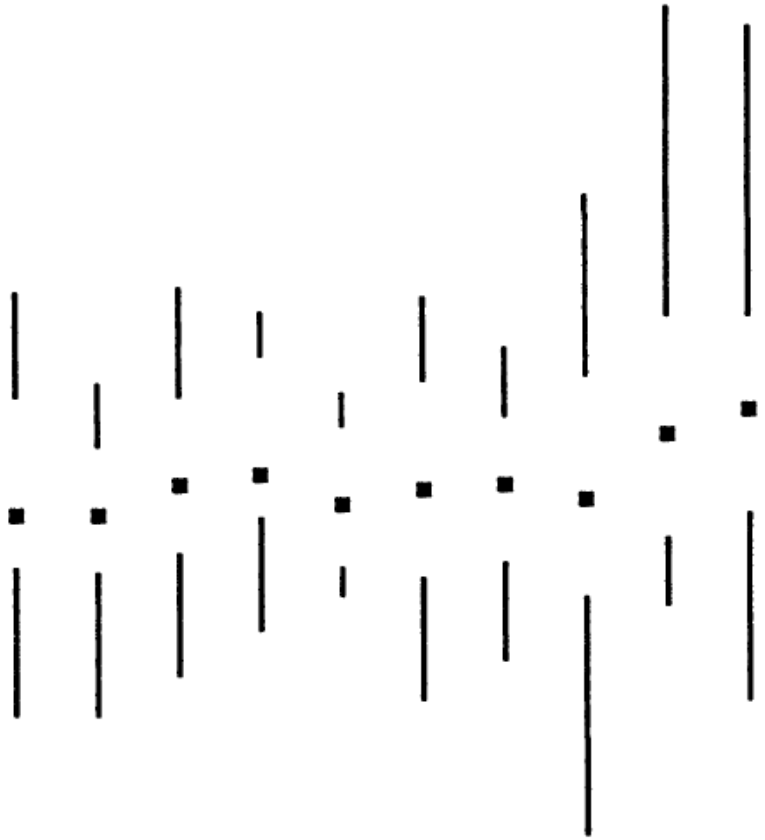
- Computed based on a Gaussian approximation
- q-quantile x_q: $\Phi\left(\frac{x_q-\mu}{\sigma}\right) = q$, where Φ is CDF of a normal random variable

  $\Rightarrow x_q = \left(\mu + \Phi^{-1}(q)\right)\sigma$
- $\mathrm{IQR} = x_{3/4} - x_{1/4} = \left(\Phi^{-1}\left(\frac{3}{4}\right) - \Phi^{-1}\left(\frac{1}{4}\right)\right)\sigma \approx 1.35\,\sigma$
- $\mathrm{CI}_\alpha = \pm\frac{\mathrm{tinv}(\alpha,n-1)\hat{\sigma}}{\sqrt{n}}$

  $\mathrm{tinv}(\alpha, n-1) \approx 1.65$ for $\alpha = 0.95$ and $n \to \infty$

$$\mathrm{CI}_\alpha \approx \pm 1.22\ \mathrm{IQR}/\sqrt{n}$$

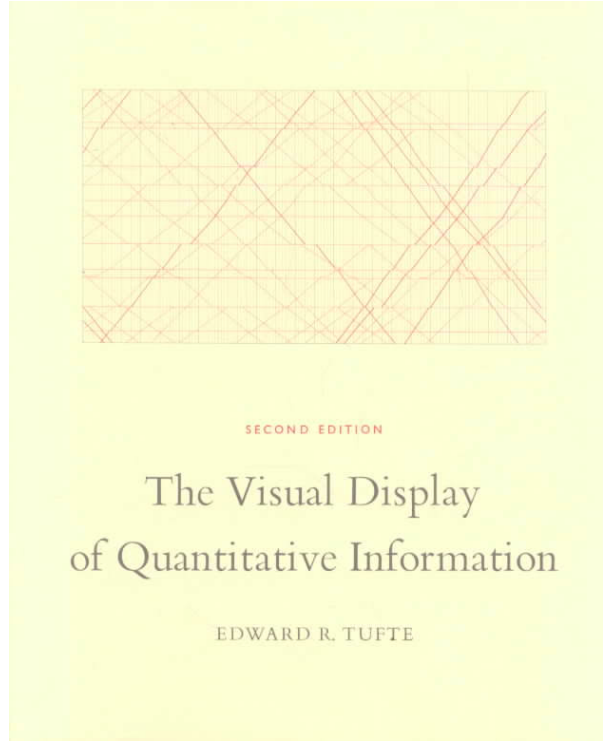
Tufte’s version of boxplots



- Source: Tufte

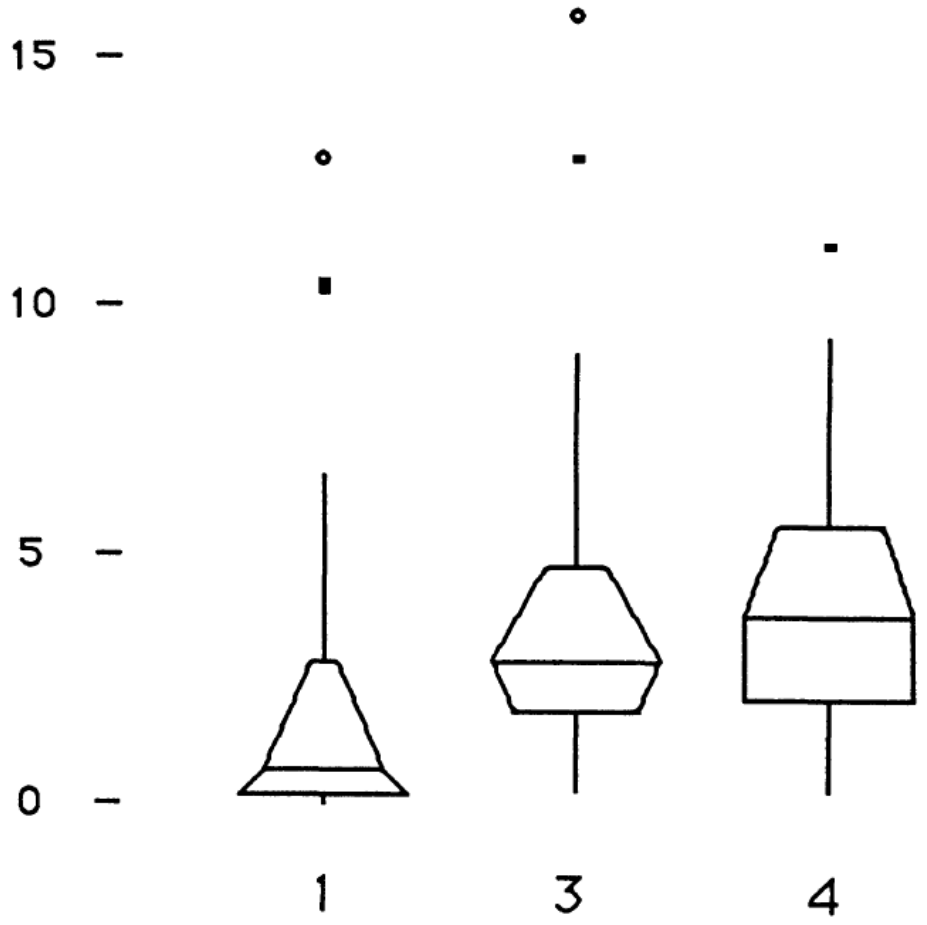
Historical remarks

- Edward Tufte
- American statistician and professor emeritus of political science, statistics, and computer science at Yale University
- A pioneer of data visualization



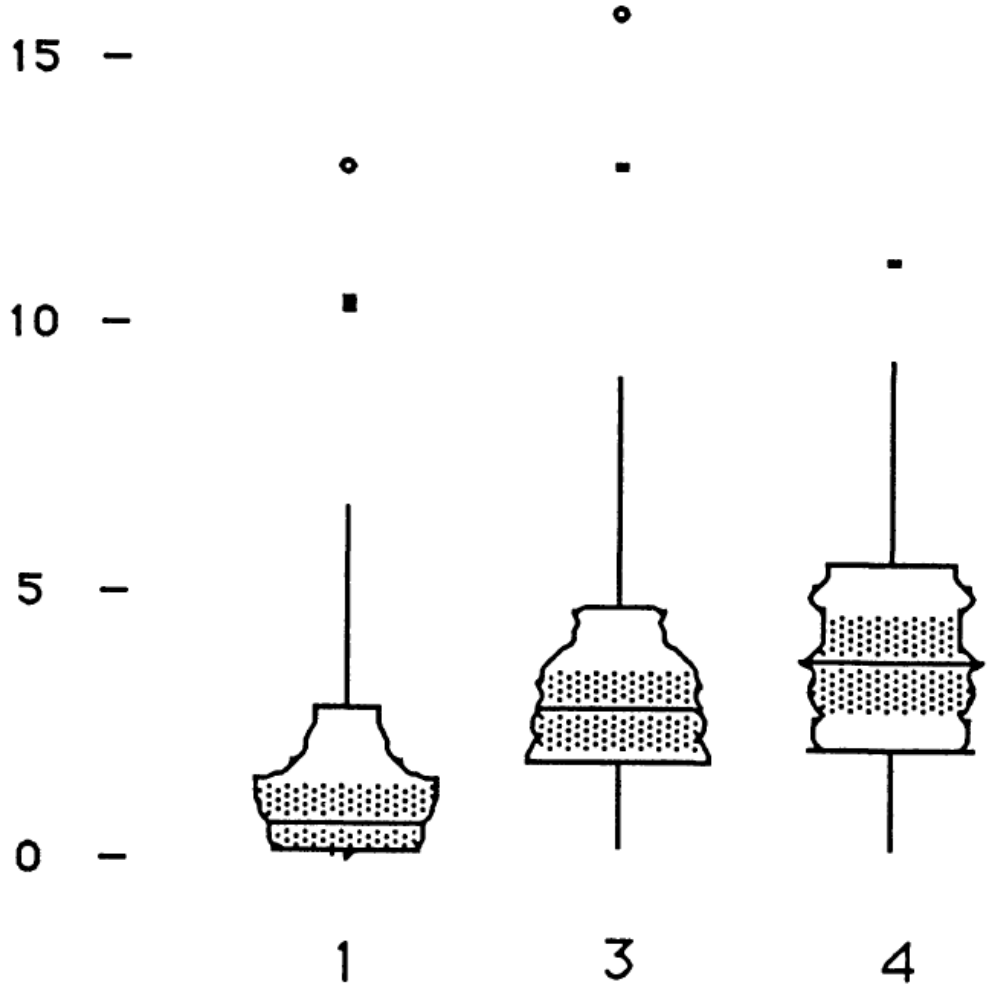
Histplot

- Draw the top, bottom and median line of the box with width at these respective points proportional to the estimated density
- Connect them with straight lines as in a frequency polygon
- Benjamini (1988)



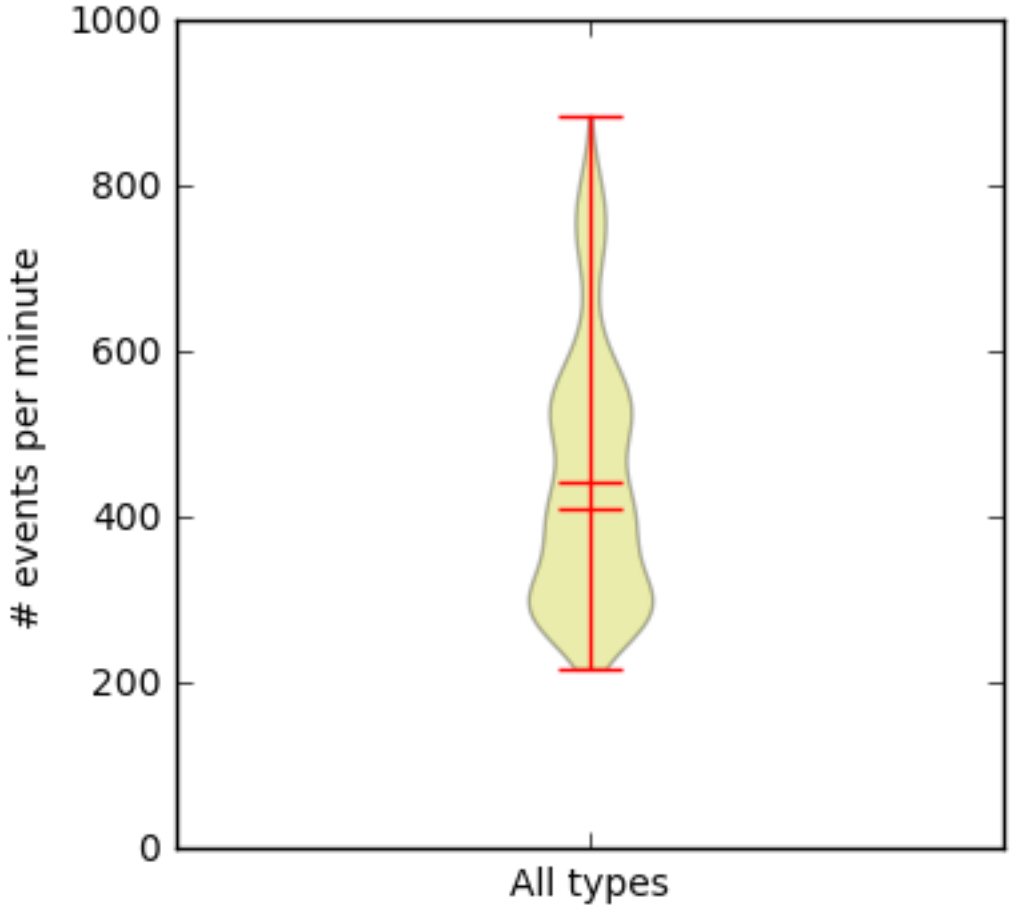
Vase Plots

- A boxplot with the width of the box at each point proportional to the estimated density
- The gray bars indicate confidence interval for the median value
- Benjamini (1988)

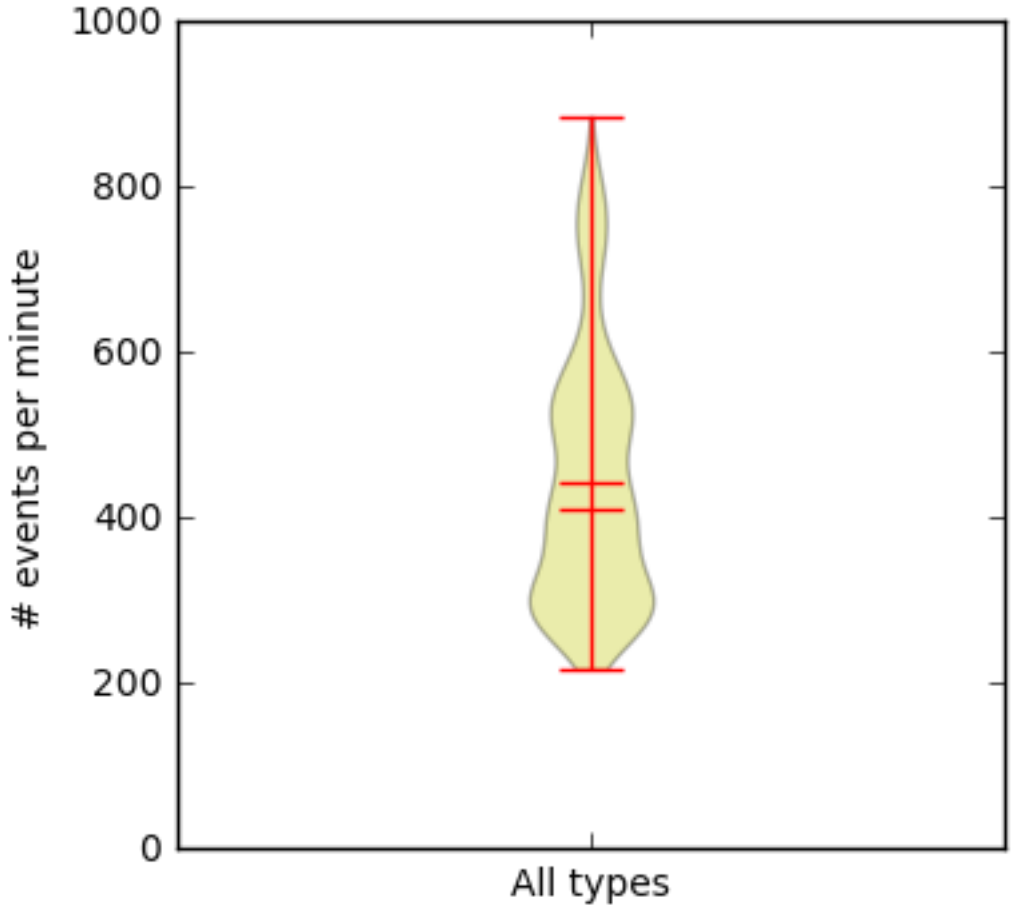


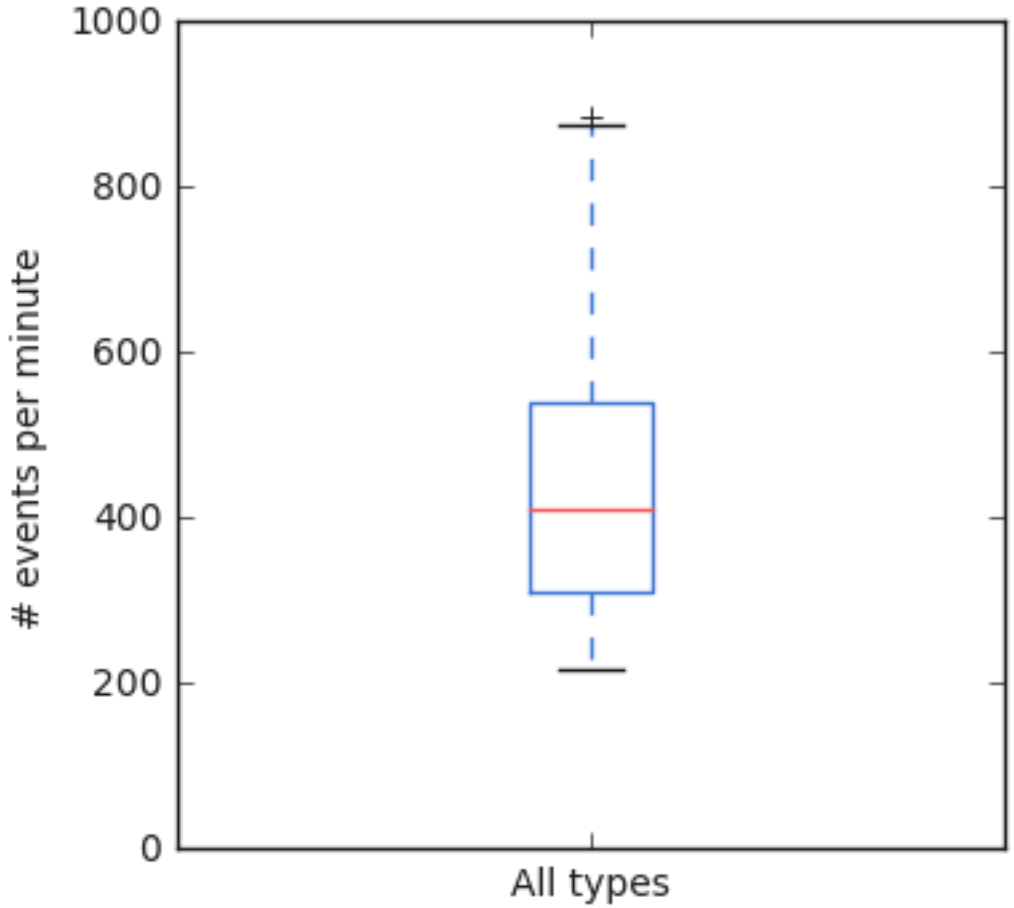
Violin plot

- Violin plots use kernel density estimation (KDE) to compute an estimation of the distribution
- The plot can be produced by using pyplot's function: `violinplot`
- Horizontal lines represent the minimum, the median, the mean and the maximum value, respectively

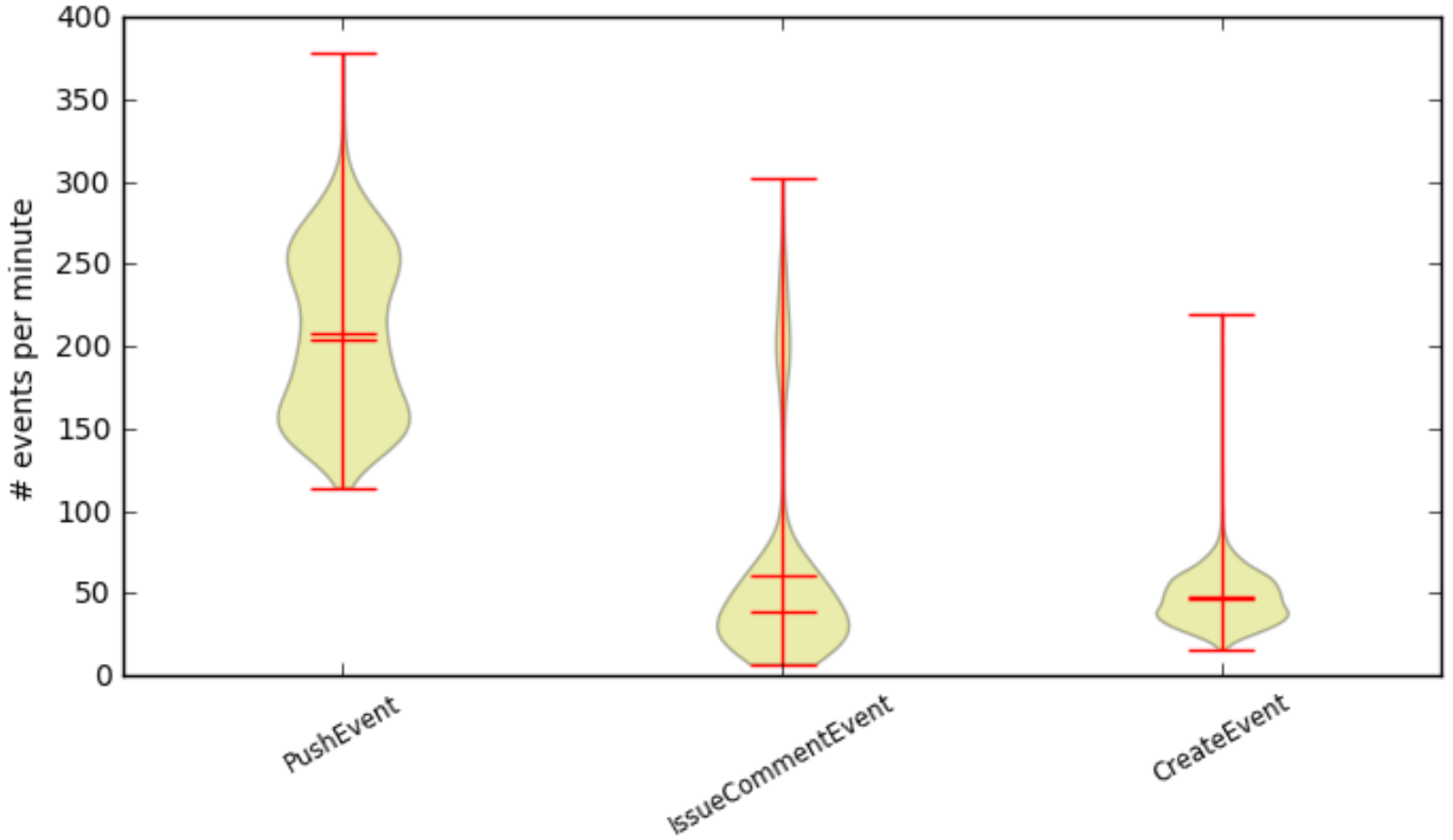


Comparison of violin plots with boxplots





Violin plots for parallel batches of data

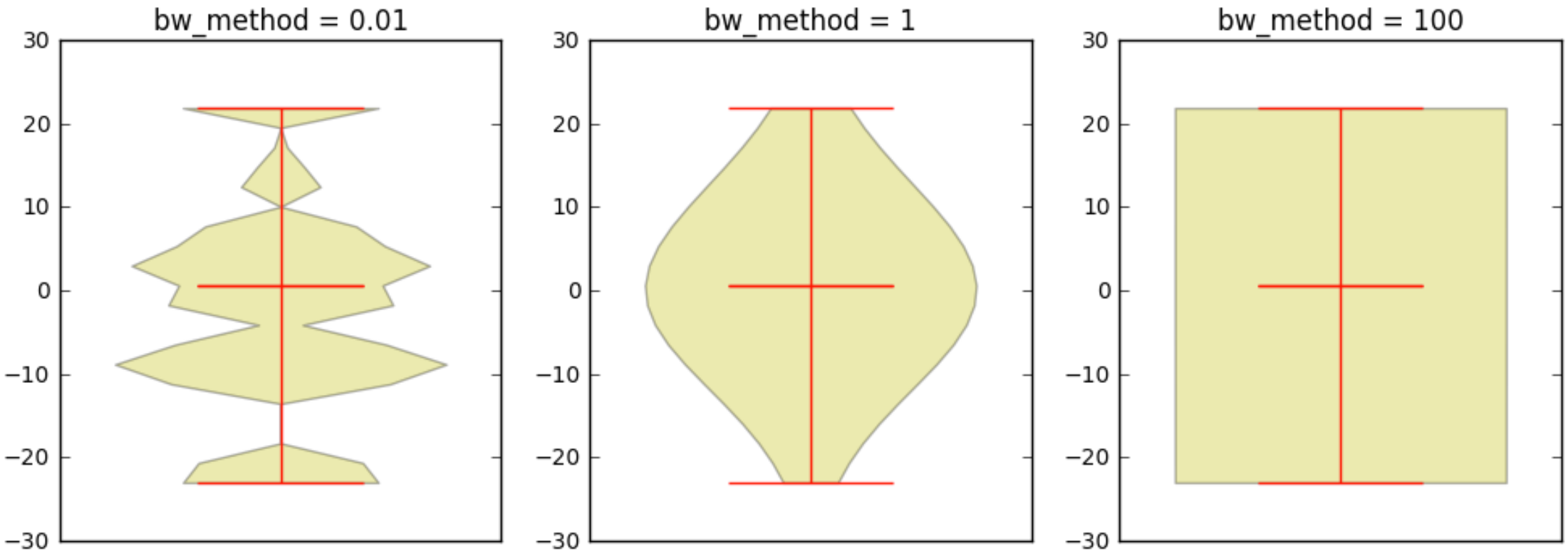


- Produced by pyplot's function `violinplot`

Kernel Density Estimation

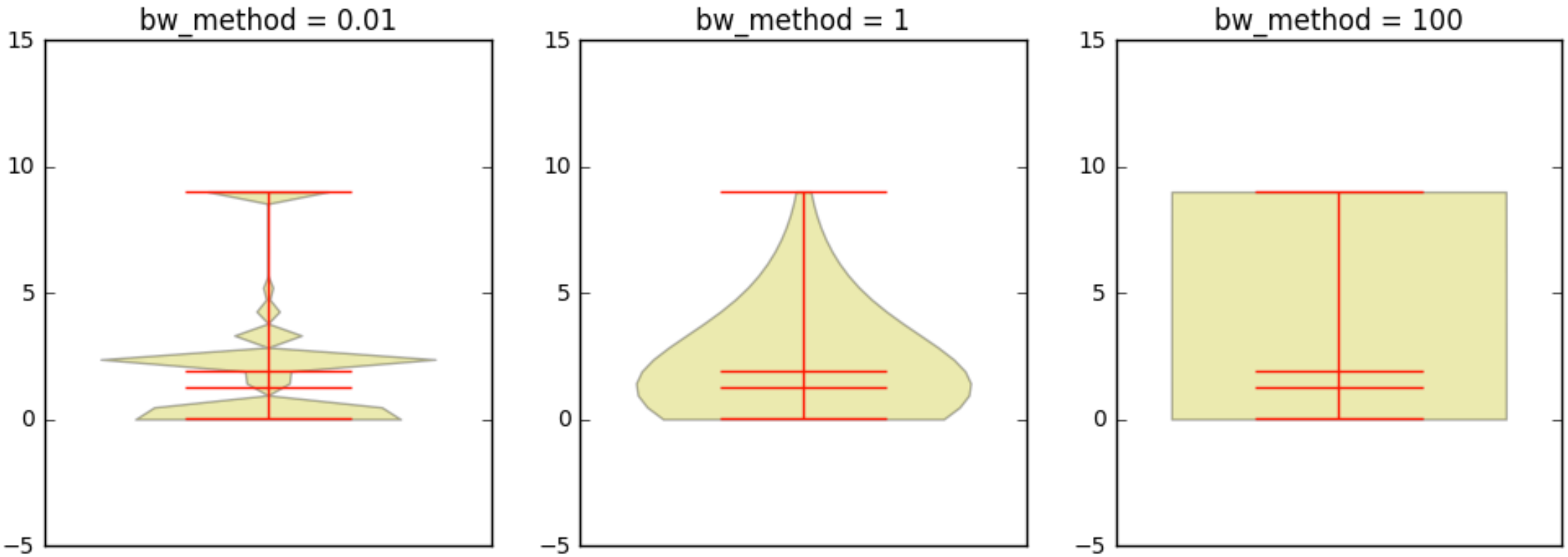
- Function f defined as: $f(x) = \frac{1}{mh}\sum_{i=1}^{m} w\left(\frac{x_i - x}{h}\right)$ where $w(x/h)/h$ is the kernel function and h is the window width
- Example kernel functions: cosine, Gaussian, or the boxcart function $w(x) = 1$ if $|x| \leq 1/2$ and $w(x) = 0$ otherwise
- For small values of h parameter, there is a spike at each sample
- For kernel functions with a bounded support (such as boxcart): for h larger than the range of sample points f is constant, thus standard boxplot is recovered for large enough value of the window width parameter
- There need not be one single best representation: different window widths can be used for different purposes

Gaussian distribution example

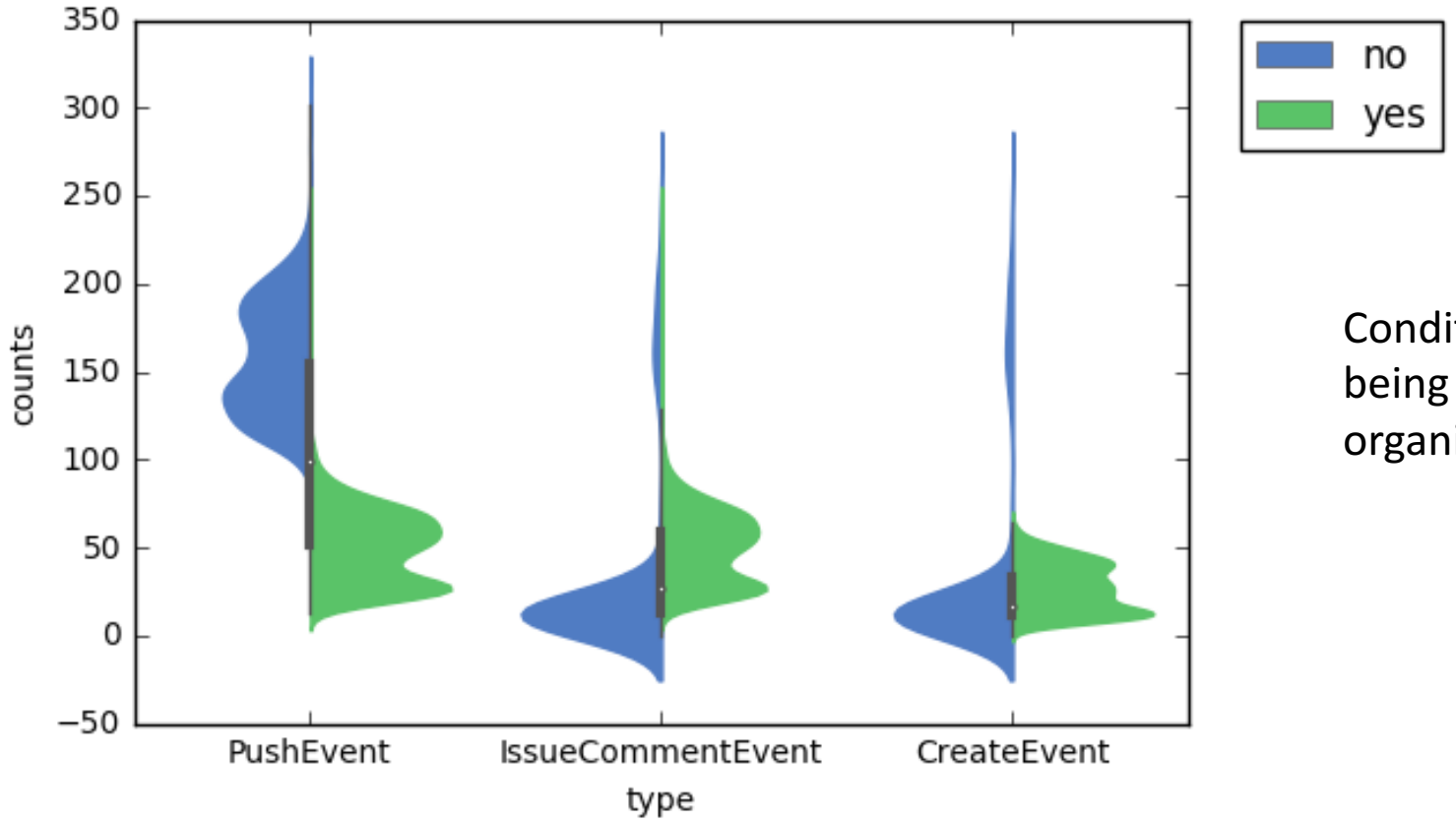


bw_method corresponds to the window width parameter h

Pareto distribution example



Violin plots with conditioning



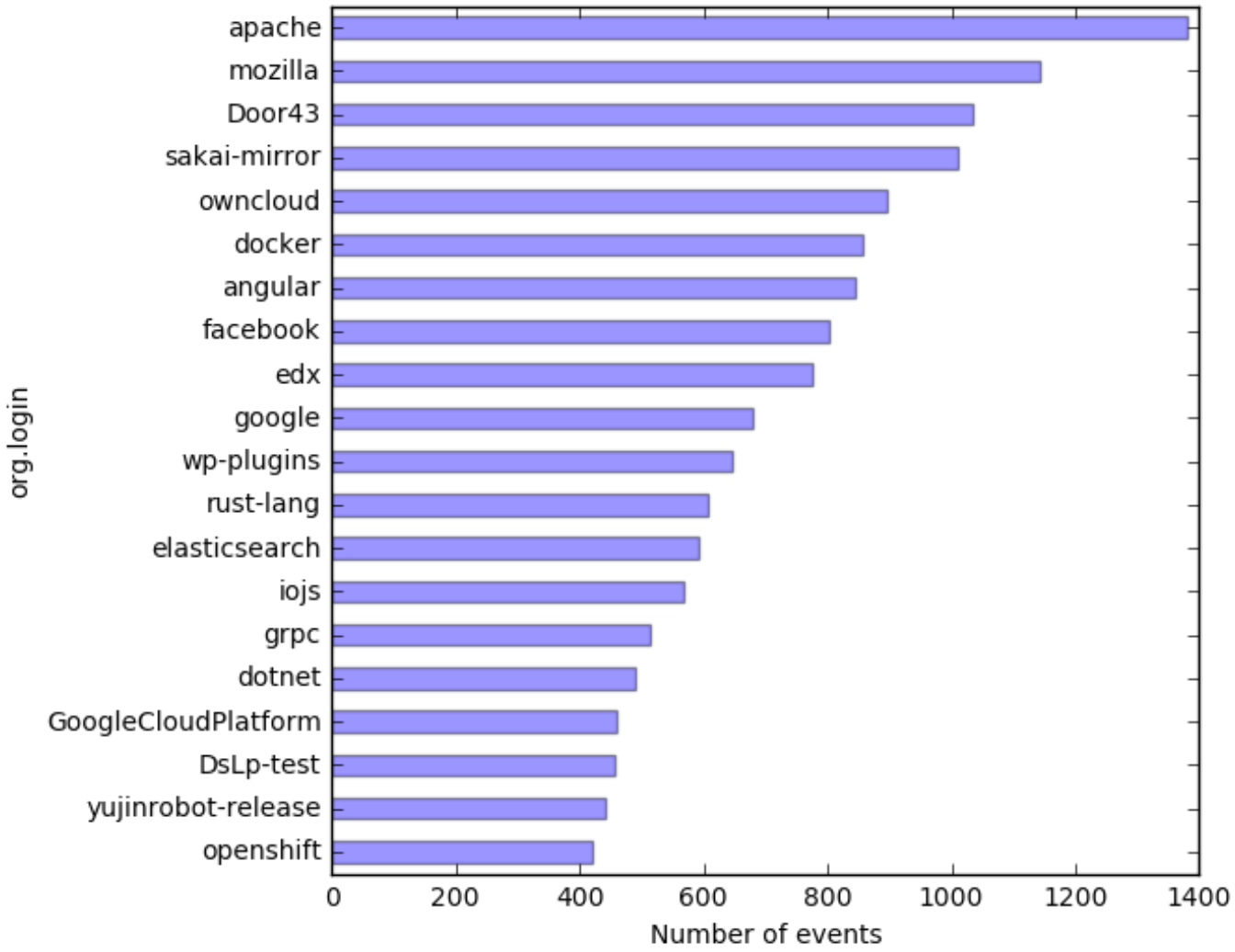
Conditioning on repository being associated with an organization or not

- Produced with Seaborn's function `violinplot`

Bar plots

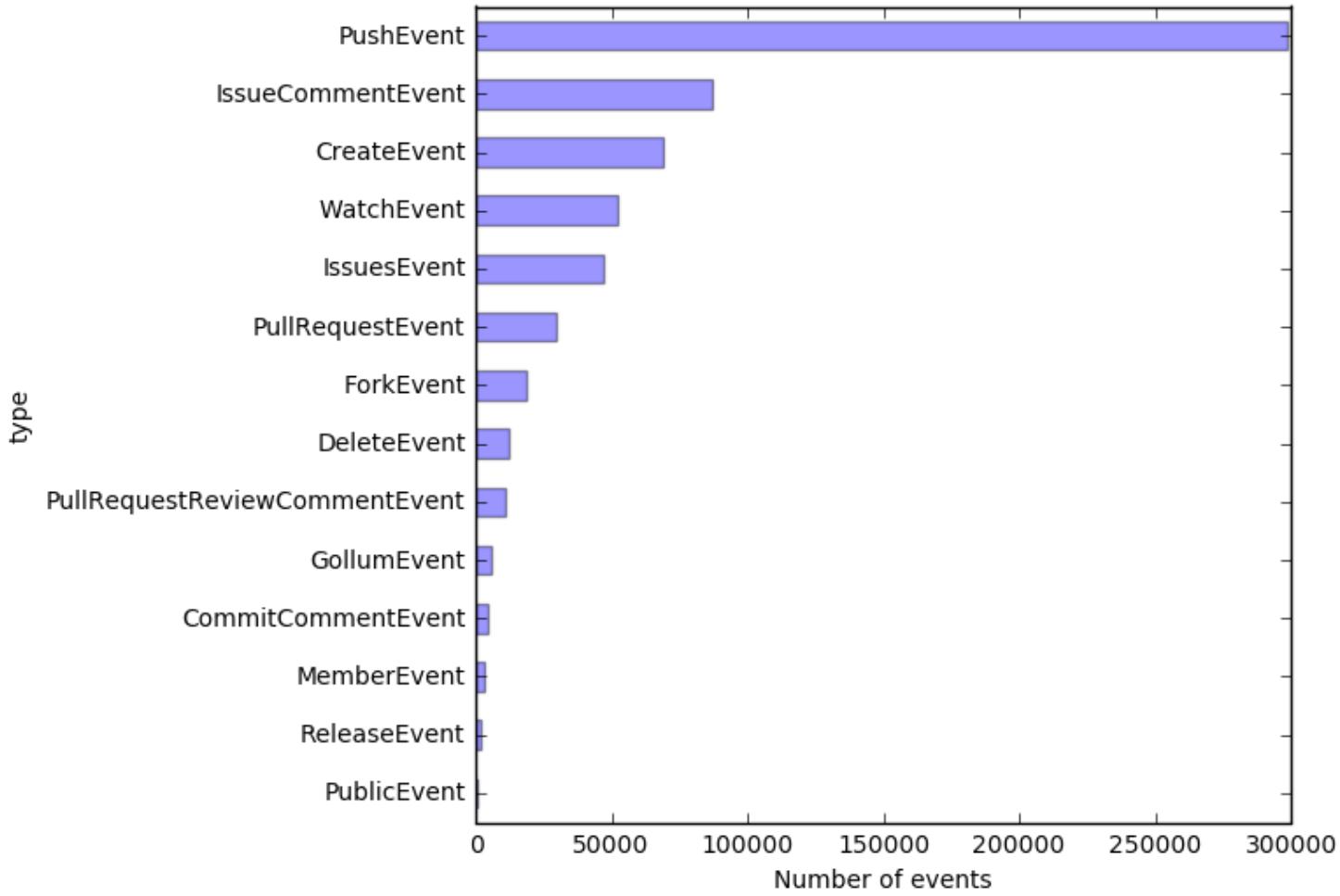
- Horizontal bar plot for the number of events per organization

- Plot produced by using pyplot's function `plot` for `kind=barh`

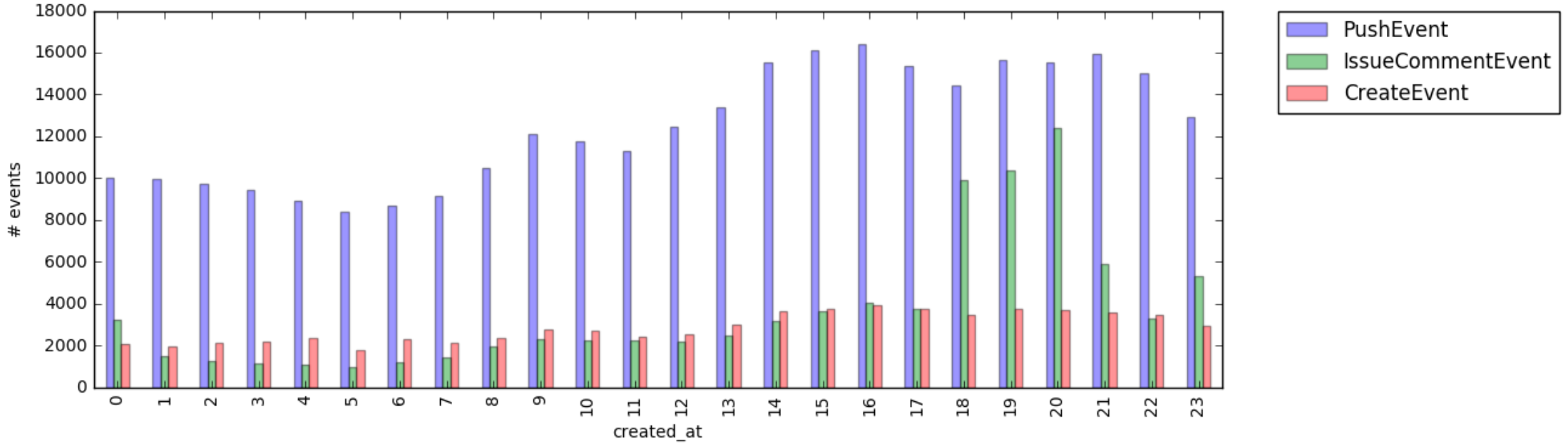


Bar plots (cont'd)

- Same but for the frequency counts of event types



Bar plots (cont'd)

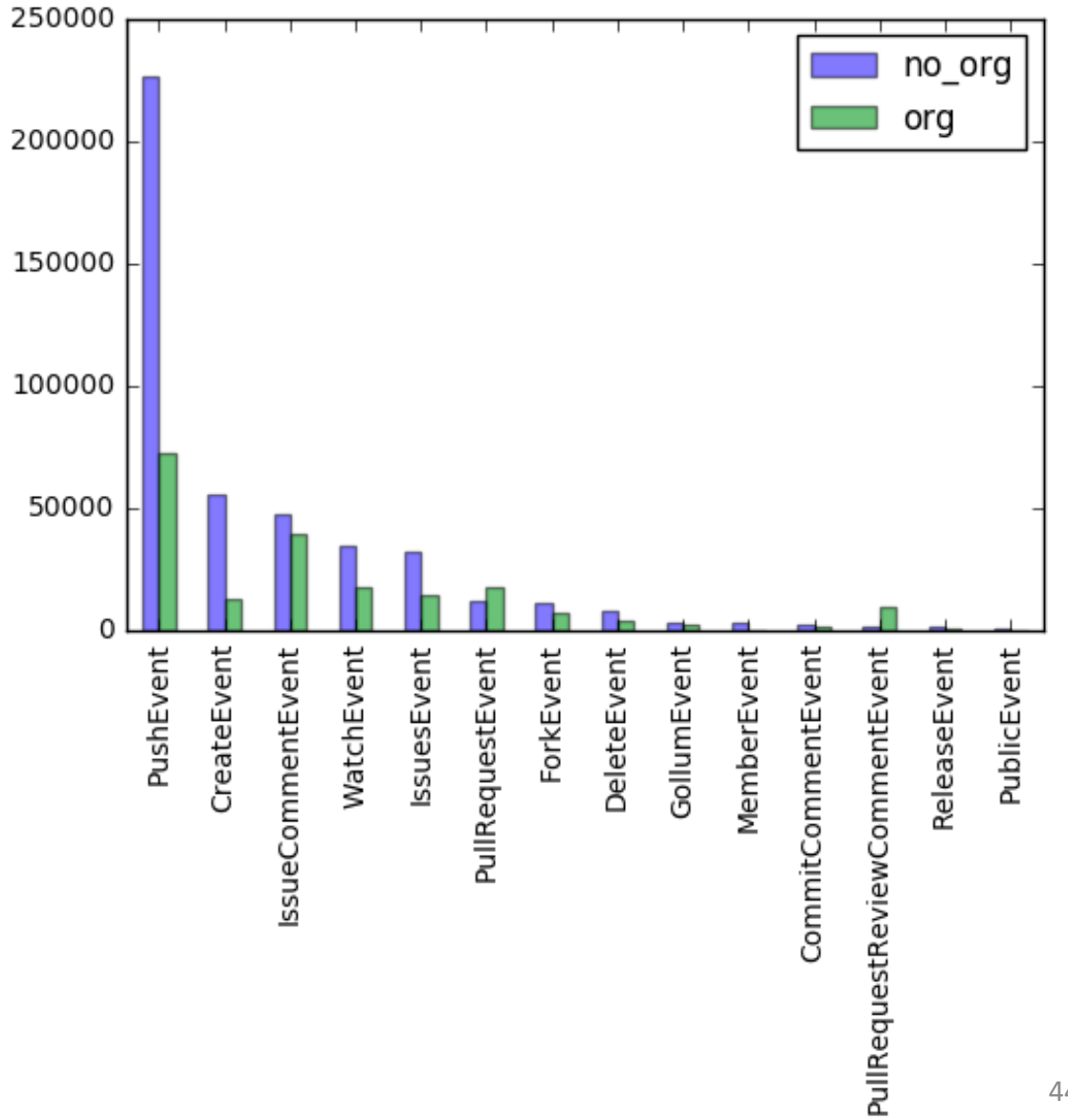


- Standard bar plot showing frequency of event types for each hour of day

Bar plots (cont'd)

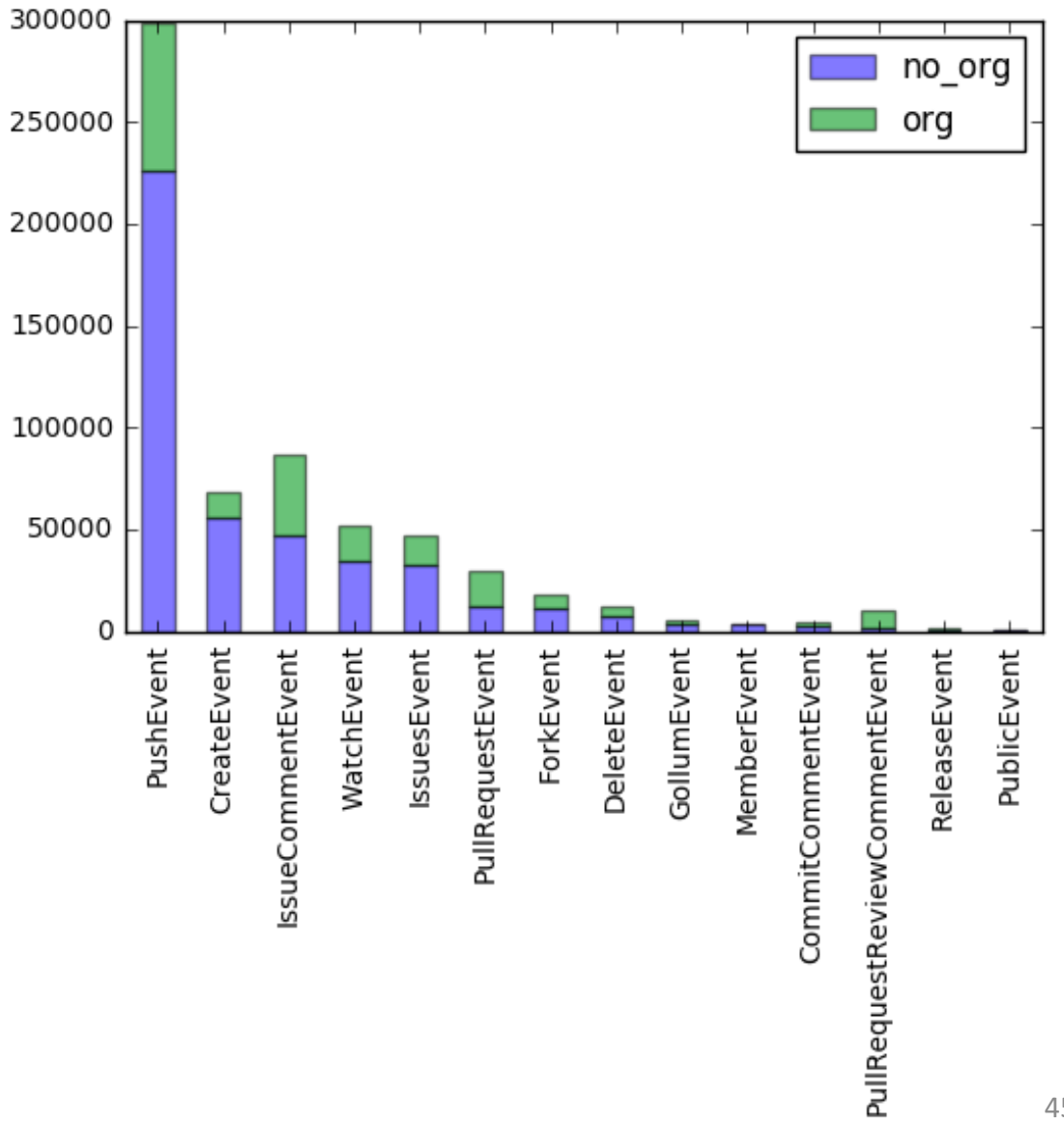
- Conditioning on whether or not an event is directed to an organizational repository

- Sorting of event types in decreasing frequency of events not directed to organizational repositories



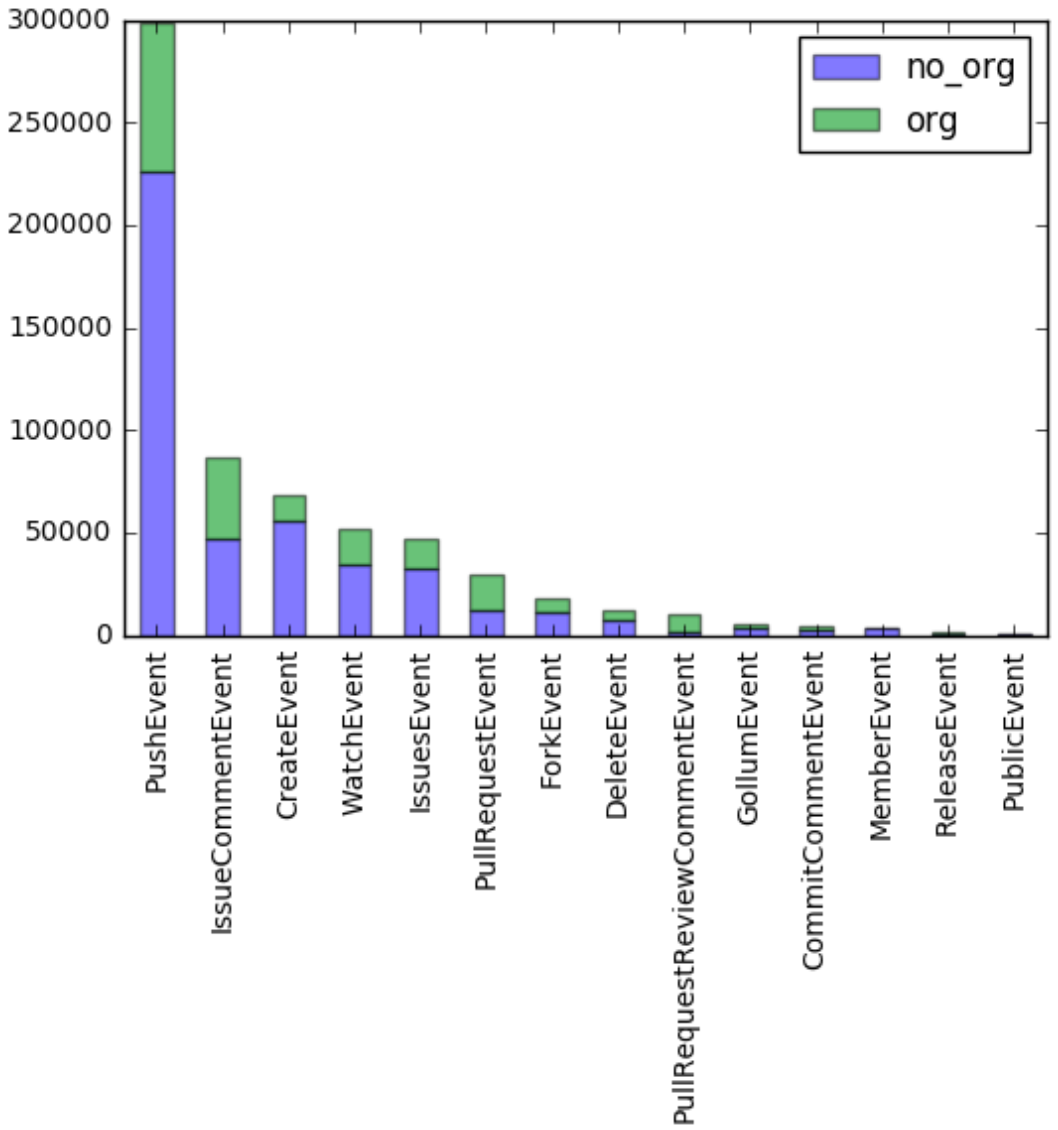
Stacked bar plot

- Same as in the previous slide but using stacked bar plot
- Using same pyplot's function `bar` but with `stacked=True`

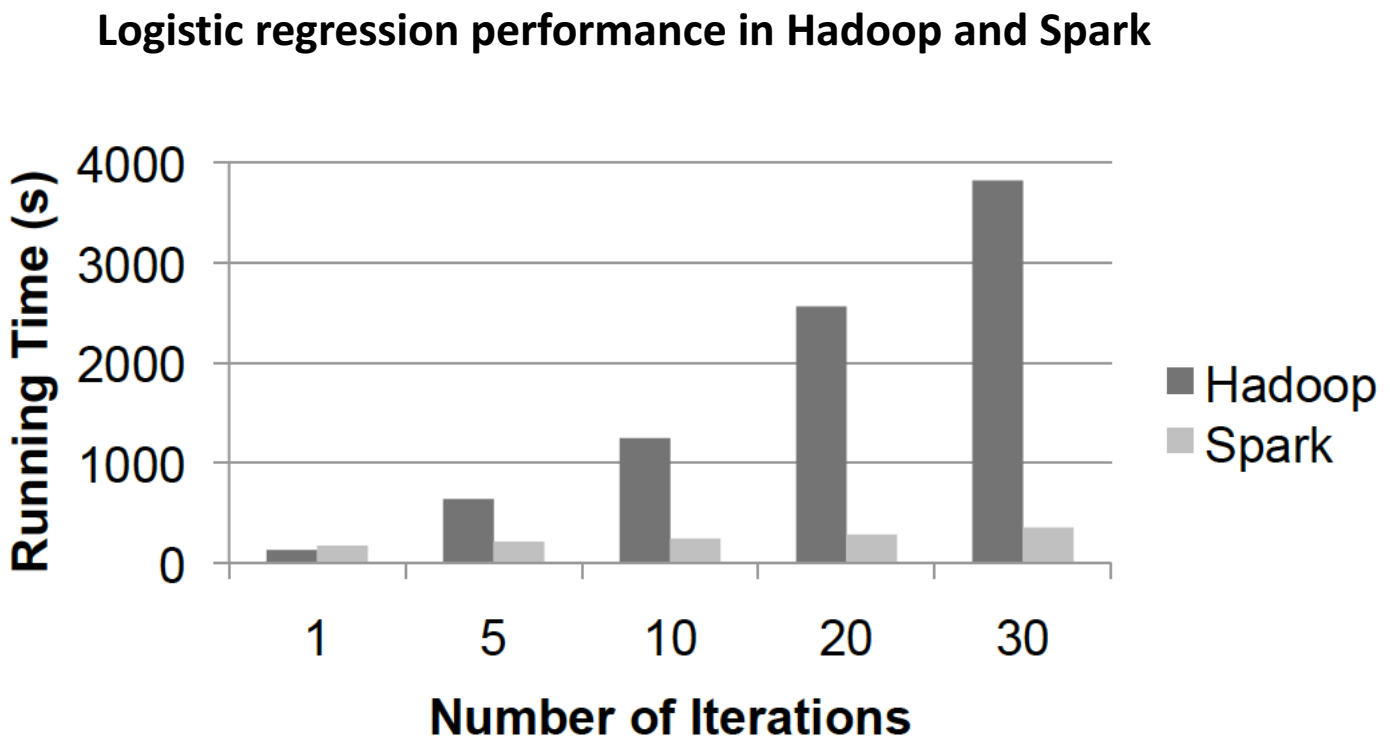


Stacked bar plot (cont'd)

- Same as in the previous slide but event types sorted in decreasing order of the total frequencies

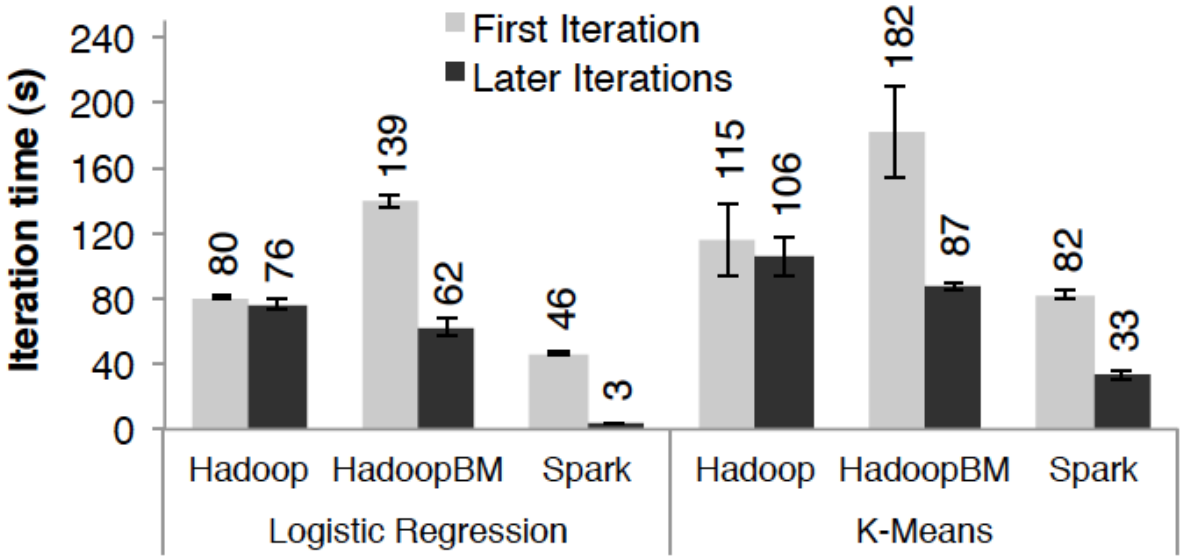


Use of bar plots in literature



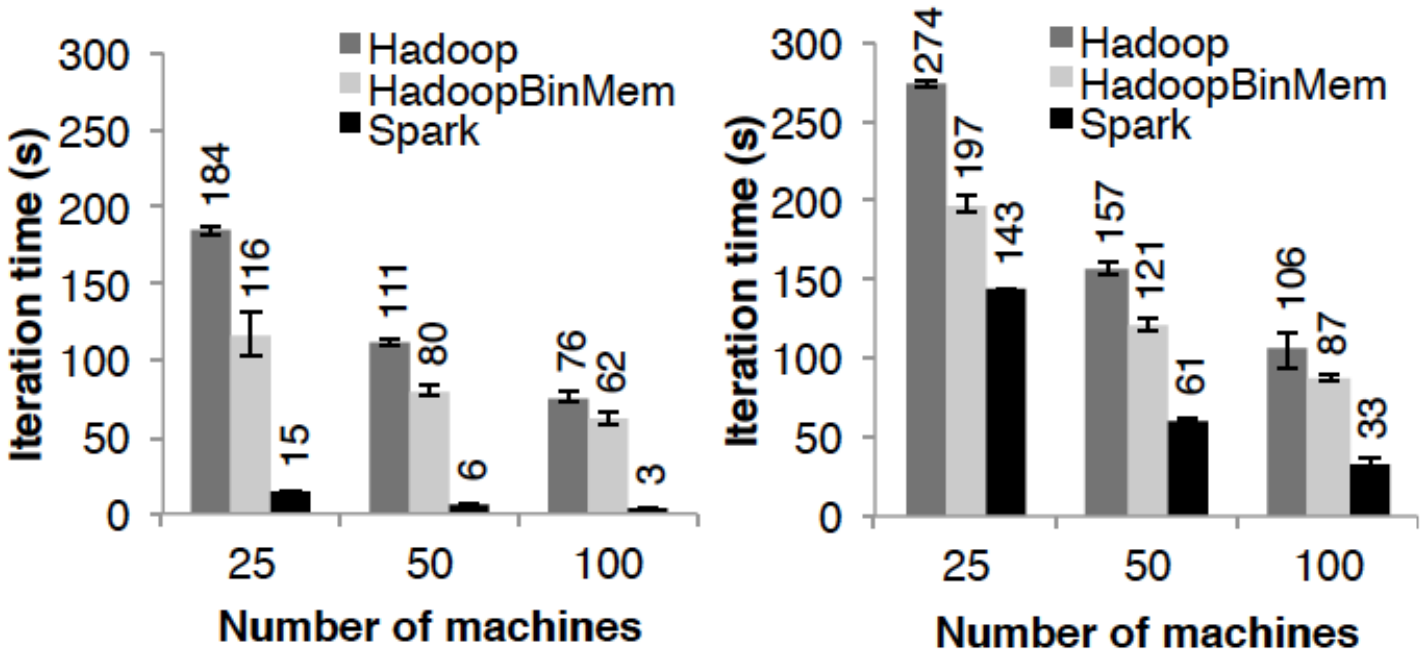
- Source: M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica, Spark: Cluster Computing With Working Sets, HotCloud 2010
- The only performance graph in the whole paper! Makes a strong point

Use of bar plots in literature (cont'd)



- Source: M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, I. Stoica, Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, NSDI 2012
- Summary of performance results in a compact real estate

Use of bar plots in literature (cont'd)



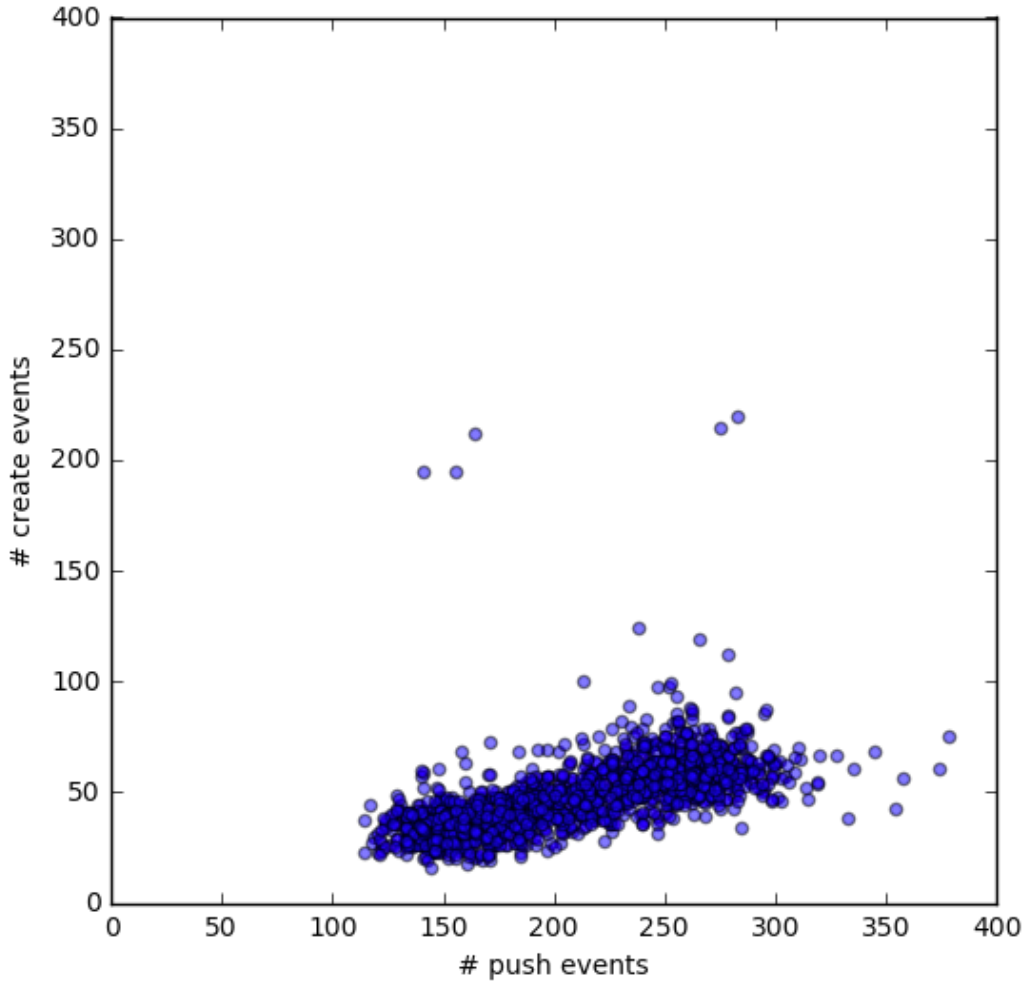
- Another summary of performance results from the same paper

Multivariate data visualization

- Scatter plots
- Scatter matrices

Scatter plot

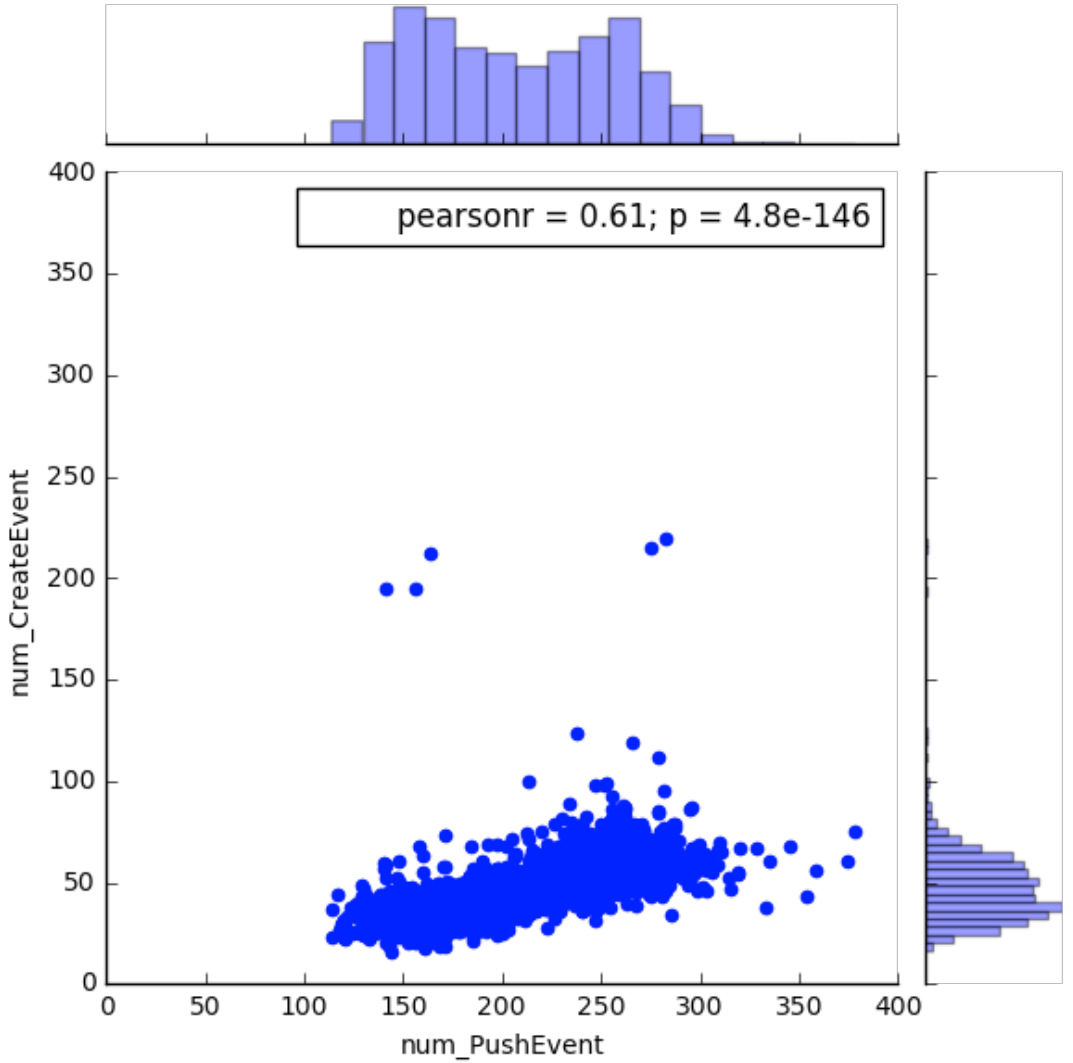
- Used for visualizations of the degree of correlations of bivariate data
- Plot produced by pyplot's function `scatter`



Scatter plot (cont'd)

- Scatter plot with marginal distributions shown on the corresponding data axes

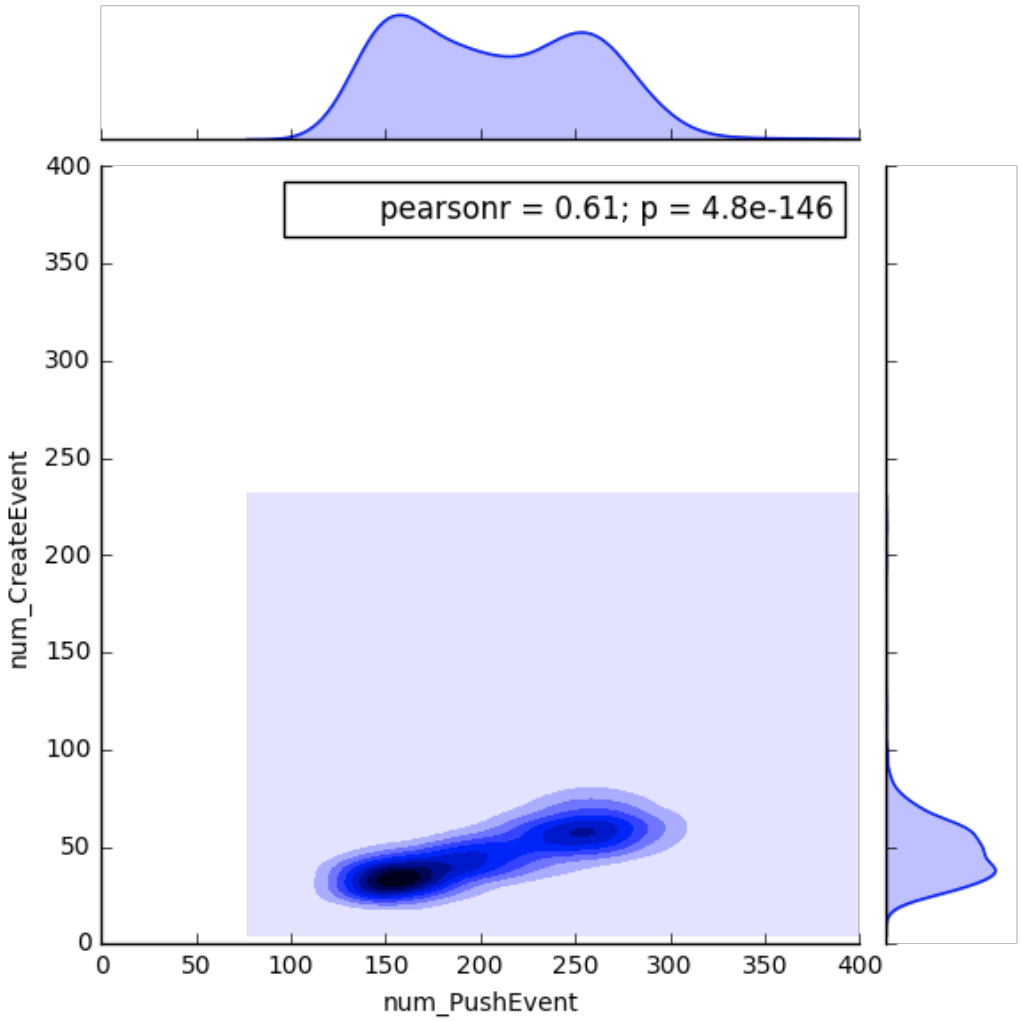
- Plot produced by Seaborn's `jointplot`



Scatter plot (cont'd)

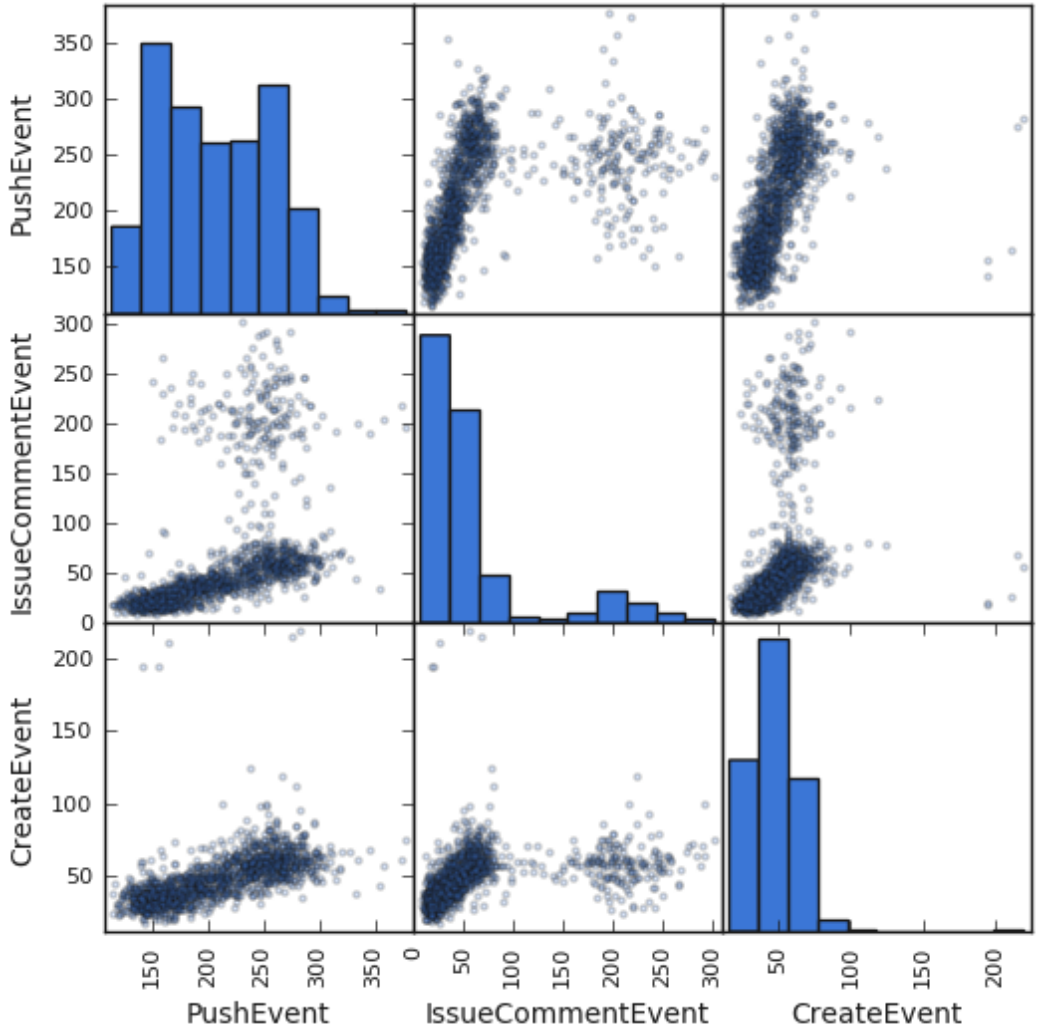
- Same as in the previous slide by with kernel density estimation

- Plot produced with Seaborn's `jointplot` with `kind=kde`



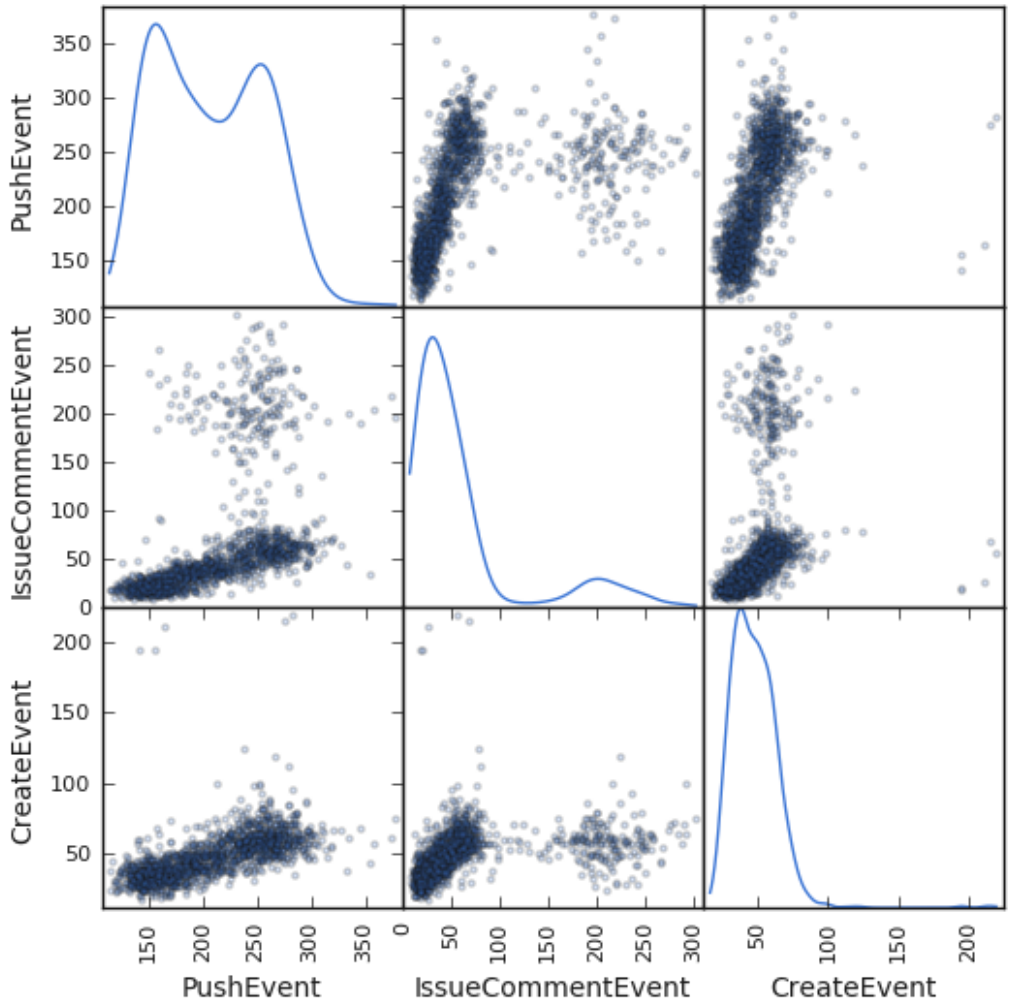
Scatter matrix

- Scatter plots for all possible pairs dimensions of input multivariate data
- Histograms shown on diagonal
- Effective means for quick exploratory data analysis
- Plot obtained by pyplot's function `scatter_matrix`



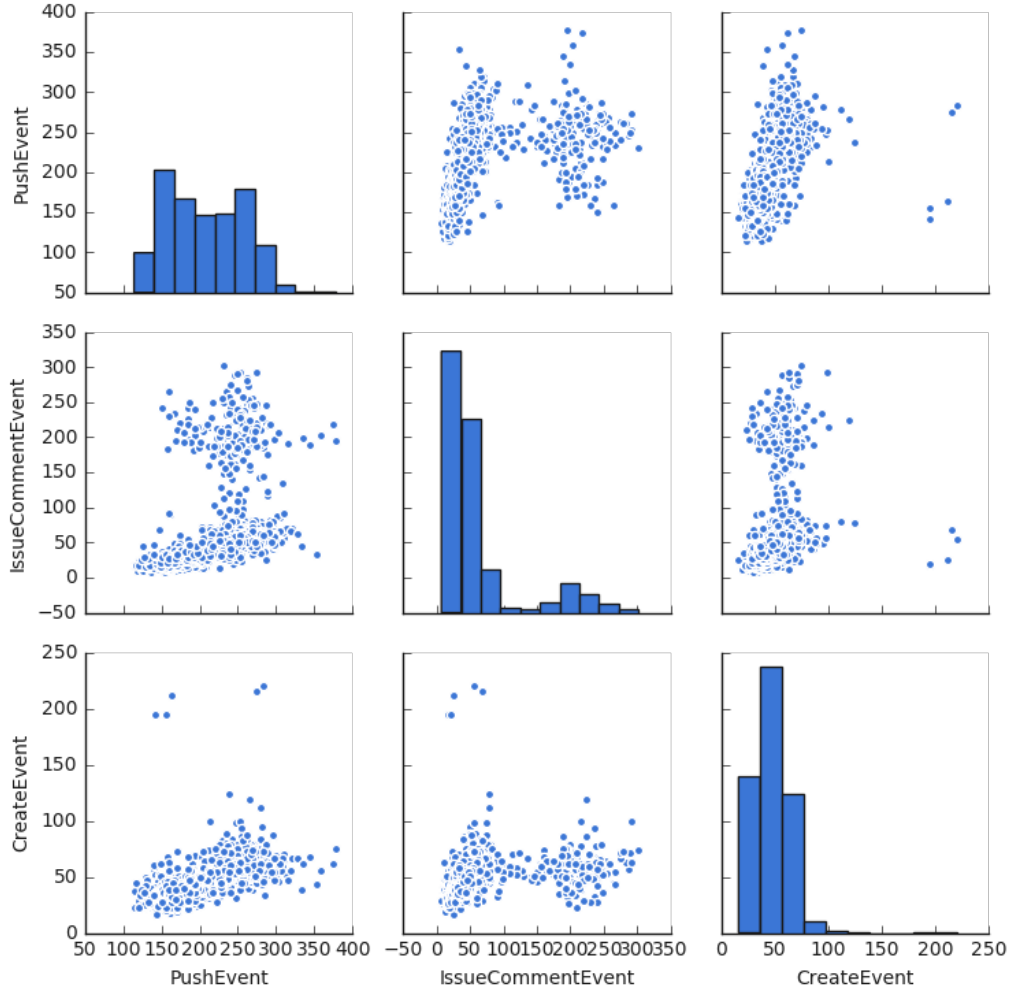
Scatter matrix (cont'd)

- Same as in the previous slide but with histograms replaced with density estimators



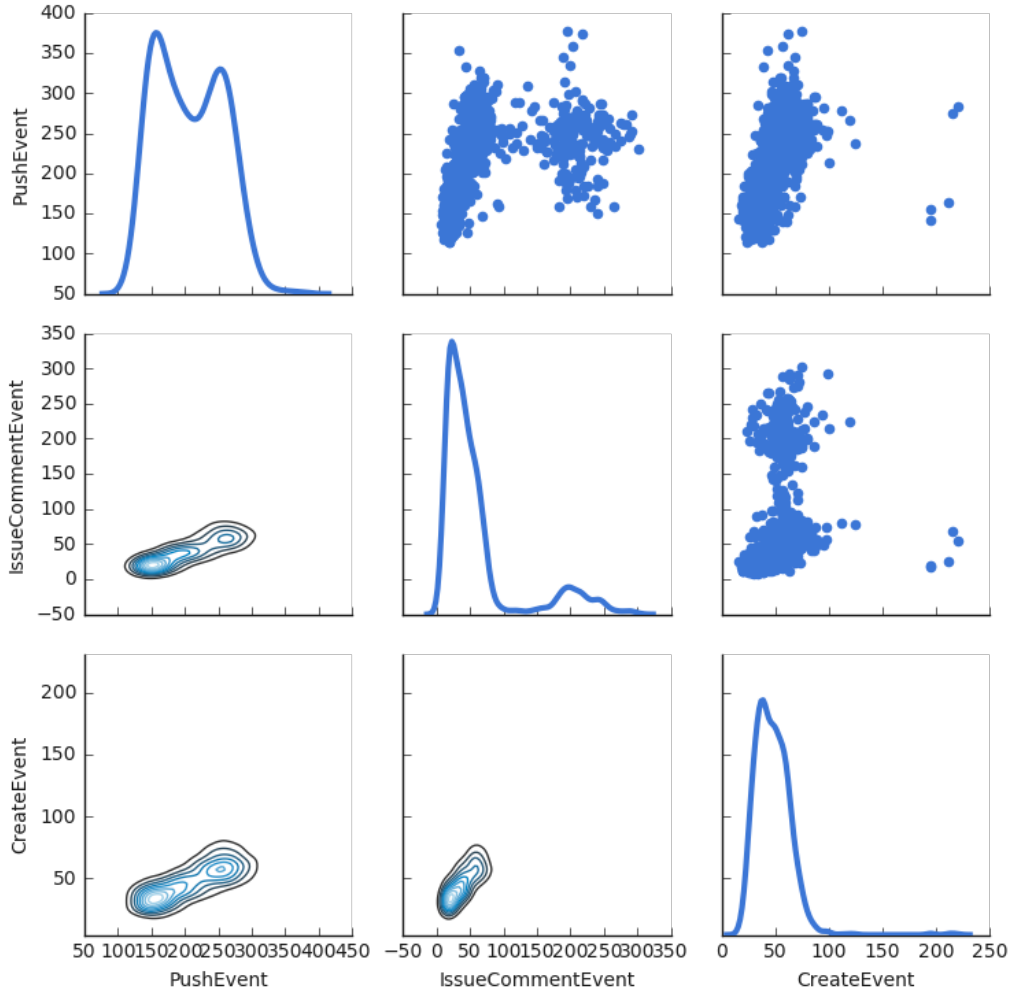
Scatter matrix (cont'd)

- Same thing but using Seaborn's `pairplot`



Scatter matrix (cont'd)

- Similar thing by combining scatter and kde plots using Seaborn's function `PairGrid`



References

- Y. Benjamini, Opening the Box of a Boxplot, The American Statistician, Vol 42, No 4, pp 257-262, 1988
- M. Krywinski and N. Altman, Nature Methods, Visualizing samples with box plots, Vol 11, No 2, Feb 2014
- R. Mcgill, J. W. Tukey and W. A. Larsen, Variations of Box Plots, The American Statistician, Vol 38, No 1, 1978
- H. Wickham and L. Stryjewski, 40 years of boxplots, had.o.nz, 2012
- Useful link: https://matplotlib.org/users/pyplot_tutorial.html
- Python Pandas Cookbook Lesson 3.2 (Descriptive Statistics and Visualizations) https://www.youtube.com/watch?v=E8OQAdQlljE