

SEN608 DATA ANALYTICS AND STATISTICS

Week 2: Exploratory Data Analysis (EDA)

Assoc. Prof. Dr. Onur Polat

Hacettepe University • Graduate School of Informatics

M.Sc. (Thesis) in Systems Engineering • Fall 2026, Lecture 2 / 14

What You Will Learn This Week

Reference: Bruce, P., Bruce, A. & Gedeck, P. (2026). AI-Assisted Statistics for Data Scientists, 3rd Ed., Chapter 1.

Estimates of Location

- Arithmetic mean and its limitations
- Trimmed mean (robust compromise)
- Median and breakdown point
- Weighted mean and weighted median

Estimates of Variability

- Standard deviation and variance
- Mean absolute deviation (MAD)
- Interquartile range (IQR)
- Percentiles and order statistics

Exploring Distributions

- Frequency tables and histograms
- Boxplots (Tukey's invention)
- Kernel density estimation (KDE)
- Skewness and kurtosis concepts

Multivariate Exploration

- Categorical data: mode, bar charts
- Correlation and scatterplots
- Hexbin and contour plots
- Violin plots and FacetGrid



PART I

Estimates of Location

Why Do We Need Location Estimates?

We want to summarize a variable with thousands of distinct values into a single number: a “typical value” — an estimate of where most of the data is concentrated.

Estimator	Formula	Sensitivity to Outliers	Best For
Arithmetic Mean	$\bar{x} = \sum x_i / n$	High	Symmetric distributions
Trimmed Mean	Drop top/bottom p%, average rest	Moderate	Slightly skewed data
Median	Middle value of sorted data	Low (breakdown = 50%)	Skewed distributions
Weighted Mean	$\bar{x}_w = \sum w_i x_i / \sum w_i$	High	Heterogeneous observation importance
Weighted Median	Midpoint of weighted sorted data	Low	Weighted + robust

The Arithmetic Mean

The most basic location estimate. The sum of all values divided by the number of observations.

Definition

$$\bar{x} = (1/n) \times \sum_{i=1}^n x_i$$

Example: {3, 5, 1, 2} $\rightarrow$ (3+5+1+2)/4 = 2.75

Critical Property

The mean uses ALL observations.

This provides statistical efficiency but makes it vulnerable to outliers.

If Bill Gates moves into your neighborhood, average income becomes meaningless.

```
import pandas as pd
from scipy.stats import trim_mean

state = pd.read_csv('state.csv')
print(f"Mean: {state['Population'].mean():.0f}") # 6,162,876
print(f"Median: {state['Population'].median():.0f}") # 4,436,370
```

Trimmed Mean — A Practical Compromise

The average of all values after dropping a fixed percentage of extreme values from each end.

Formula and Intuition

$$\bar{x}_t = \sum_{i=p+1}^{n-p} x_{(i)} / (n - 2p)$$

p: observations dropped from each end

- 10% trim: drop smallest/largest 10%, average the rest
- Same logic as Olympic diving: drop highest and lowest judge scores

Result: Robust to outliers while using more data than the median

State Population Example

Mean: 6,162,876
Trimmed (10%): 4,783,697
Trimmed (20%): 4,459,825
Median: 4,436,370

The mean is pulled upward by CA, TX, NY.
Trimmed mean resists this pull.

Key: Mean > Trimmed > Median
→ This ordering itself reveals right skew.

```
from scipy.stats import trim_mean
print(f"Trimmed (10%): {trim_mean(state['Population'], 0.1):,.0f}") # 4,783,697
print(f"Trimmed (20%): {trim_mean(state['Population'], 0.2):,.0f}") # 4,459,825
```

Median — The Robust Location Estimate

The middle value of sorted data. If n is even, the average of the two middle values.

Breakdown Point: 50%

The median's strongest property:
up to 50% of the data can change
without affecting the median.

This is the maximum possible

Outliers and Anomaly Detection

An outlier is a value very different from
the rest. It may be:

- A data error (unit confusion, sensor fault)
- A genuine observation (Bill Gates)
- Always worth investigating

Methodological note: If $\text{Mean} > \text{Trimmed Mean} > \text{Median}$, the distribution is right-skewed.

The relationship between location estimators provides indirect information about distributional shape.

(one extreme value is enough)

data defines “normal,” and deviations
from it are the signal.

Weighted Mean and Weighted Median

Used when some observations are intrinsically more important or more reliable than others.

Two Motivations

1. Variable reliability: some sensors are less accurate → downweight them
2. Representativeness correction: if the sample doesn't reflect the population, upweight underrepresented groups to correct the bias

Systems eng. analogy: weighting component failure rates by usage intensity

Murder Rate Example

Unweighted: Wyoming (580K) and California (37M) count equally.

Weighted by population: estimates the murder rate a randomly chosen US person would experience.

Weighted mean: 4.45

Weighted median: 4.40

```
import wquantiles
weighted_mean = np.average(state['Murder.Rate'], weights=state['Population'])
weighted_median = wquantiles.median(state['Murder.Rate'], weights=state['Population'])
print(f"Weighted mean: {weighted_mean:.2f}") # 4.45
print(f"Weighted median: {weighted_median:.2f}") # 4.40
```

Key Ideas — Location Estimates

- The basic location metric is the mean, but it can be sensitive to extreme values (outliers).
- Other metrics (median, trimmed mean) are less sensitive to outliers and unusual distributions — hence more robust.
- Which estimator is “correct” depends on the research question — there is no single right answer.

Estimate vs. Metric

Statisticians prefer “estimate” — emphasizing the gap between what we compute from data and the true population parameter. Data scientists prefer “metric” — focusing on concrete business objectives.

Statisticians estimate; data scientists measure.

— Bruce, Bruce & Gedeck (2026), p. 14



PART II

Estimates of Variability

Why Measure Variability?

“At the heart of statistics lies variability: measuring it, reducing it, distinguishing random from real variability, identifying the various sources, and making decisions in the presence of it.”

— Bruce, Bruce & Gedeck (2026)

Metric	Definition	Breakdown Point	Python
Std Deviation (s)	$s = \sqrt{\sum(x_i - \bar{x})^2 / (n-1)}$	0% — one outlier suffices	<code>df['col'].std()</code>
Variance (s ²)	$s^2 = \sum(x_i - \bar{x})^2 / (n-1)$	0%	<code>df['col'].var()</code>
Mean Abs Dev	$\sum x_i - \bar{x} / n$	~0%	<code>np.mean(np.abs(x - x.mean()))</code>
MAD (from median)	$\text{Median}(x_i - m)$	50% — maximally robust	<code>robust.scale.mad(x)</code>
IQR	$Q_3 - Q_1$	25%	<code>df.quantile(.75) - df.quantile(.25)</code>
Range	$\max - \min$	0%	<code>df.max() - df.min()</code>

Standard Deviation and Variance

Why Squared Deviations?

The sum of deviations ($x_i - \bar{x}$) is always zero — negatives cancel positives.

Two solutions:

1. Absolute values → MAD
2. Squaring → Variance

Squaring is mathematically convenient (differentiable, additive). This is why statistical theory is built on variance.

Std deviation = $\sqrt{\text{Variance}}$
(returns to the original scale)

n vs. n-1: Degrees of Freedom

Sample variance uses $n-1$ in the denominator. Why?

Computing the sample mean ($\bar{x}$) imposes one constraint: the last observation is determined by the others. This reduces the degrees of freedom to $n-1$.

$n \rightarrow$ biased estimate (underestimates)

$n-1 \rightarrow$ unbiased estimate

In practice: for large n the difference is negligible. But know the principle.

Python: `df.std()` uses $n-1$ (`ddof=1`)
`df.std(ddof=0)` uses n

MAD — The Robust Variability Estimate

$\text{MAD} = \text{Median}(|x_1 - m|, |x_2 - m|, \dots, |x_n - m|)$ where $m = \text{Median}(x)$

Why MAD?

- Like the median, unaffected by outliers (breakdown point = 50%)
- Far more robust than std deviation
- No normality assumption required

Scale factor:

$\text{MAD} \times 1.4826 \approx \text{Std Deviation}$

(under normality)

State Population Comparison

Std Deviation: 6,848,235

IQR: 4,847,308

MAD: 3,849,870

Std is nearly 2× the MAD!

This confirms extreme values (CA, TX, NY) inflate the std deviation.

Conclusion: For skewed data, MAD and IQR provide more reliable variability estimates than std deviation.

```
from statsmodels import robust
print(f"Std: {state['Population'].std():.0f}") # 6,848,235
print(f"IQR: {state['Population'].quantile(0.75) - state['Population'].quantile(0.25):.0f}") # 4,847,308
print(f"MAD: {robust.scale.mad(state['Population']):.0f}") # 3,849,870
```

Percentiles and Order Statistics

The Pth percentile is the value below which P% of the data falls. Median = 50th percentile.

Percentile	Also Known As	Murder Rate	Interpretation
5th	Lower tail	1.60	5% of states below this value
25th	Q_1 (1st quartile)	2.42	Lower quarter boundary
50th	Median	4.00	Exact midpoint
75th	Q_3 (3rd quartile)	5.55	Upper quarter boundary
95th	Upper tail	6.51	95% of states below this value

```
state[['Murder.Rate']].quantile([0.05, 0.25, 0.50, 0.75, 0.95])
```



PART III

Exploring the Data Distribution

Boxplot — Tukey's Invention (1977)

Anatomy of a Boxplot

Upper whisker — Farthest point within

$$Q_3 + 1.5 \times IQR$$

—— — Q_3 (75th percentile)



—— — | — Median (50th percentile)



—— — Q_1 (25th percentile)

Lower whisker — Farthest point within

$$Q_1 - 1.5 \times IQR$$

○ ○ ○ Beyond whiskers:
potential outliers

$IQR = Q_3 - Q_1$ (box height)

What It Reveals

Five pieces of information at a glance:

1. Center (median line)
2. Spread (box = IQR)
3. Skewness (median position in box)
4. Tail length (whiskers)

```
ax = (state['Population'] / 1_000_000
      ).plot.box(figsize=(3, 4))
ax.set_ylabel('Population (millions)')
plt.title('State Population')
plt.show()
```

Frequency Tables and Histograms

Histogram Rules

1. Empty bins are included
2. Bin widths are equal
3. Number of bins is user-selected
4. Bars are contiguous (unlike bar charts)

Histogram ≠ Bar Chart

- Histogram: continuous variable

```
ax = (state['Population'] / 1_000_000).plot.hist(figsize=(4, 4))
ax.set_xlabel('Population (millions)')
plt.title('State Population Histogram')
plt.tight_layout()
plt.show()
```

Choosing the Number of Bins

Too few bins → structure is hidden

Too many bins → noise is captured

Rules of thumb:

- Sturges: $k = 1 + 3.322 \times \log_{10}(n)$
- Freedman-Diaconis: $h = 2 \times \text{IQR} / n^{1/3}$
- Trial and error: start with 15–50

```
plt.hist(x, bins='auto') # automatic
```

Density Plots and Kernel Density Estimation

A density plot is a smoothed version of the histogram, computed via Kernel Density Estimation (KDE).

How KDE Works

A kernel function (typically Gaussian) is placed at each data point. The sum of all kernels produces a smooth curve.

Bandwidth selection:

- Narrow bandwidth → captures noise
- Wide bandwidth → hides structure

Total area under the curve = 1
(normalized as a probability density).

Histogram vs. KDE

Histogram:

- ✓ Intuitive, easy to interpret
- ✗ Discrete, depends on bin choice

KDE:

- ✓ Continuous, smooth
- ✓ Distribution shape is clearer
- ✗ Depends on bandwidth choice
- ✗ Can mislead with small samples

Best practice: overlay both

```
ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0, 12], bins=range(1, 12))
state['Murder.Rate'].plot.density(ax=ax)
ax.set_xlabel('Murder Rate (per 100,000)')
```

Statistical Moments — Skewness and Kurtosis

Moment	Measures	Normal Dist.	Interpretation
1st: Location	Central tendency	$\mu = 0$ (standardized)	Where data clusters
2nd: Variability	Dispersion	$\sigma^2 = 1$ (standardized)	How spread out data is
3rd: Skewness	Asymmetry	$= 0$ (symmetric)	> 0 right-skewed, < 0 left-skewed
4th: Kurtosis	Tail weight	$= 3$ (mesokurtic)	> 3 heavy tails (leptokurtic)

In practice: Skewness and kurtosis are usually discovered visually through histograms and boxplots rather than reported as numeric values. The visual approach is more informative and less misleading.

```
print(f"Skewness: {log_ret.skew():.3f}")  
print(f"Kurtosis: {log_ret.kurtosis():.3f}") # excess kurtosis (Normal = 0)
```



PART IV

Categorical Data and Correlation

Exploring Binary and Categorical Data

Key Concepts

Mode: most frequent category.

Expected value: weighted average of categorical outcomes.

Proportions and percentages: the primary summary tool for categorical data.

Bar chart: compares frequencies across categories. Unlike histograms, bars are separated (discrete).

DFW Airport — Delay Causes

Carrier: 23.0%

ATC: 30.4%

Weather: 21.5%

Security: 0.1%

Inbound: 42.1%

Pareto principle: Inbound (42%) and ATC (30%) account for the majority.

Systems eng. analogy: Pareto analysis of failure modes — target the vital few.

```
dfw = pd.read_csv(f'{DATA_URL}/dfw_airline.csv')
print((100 * dfw / dfw.to_numpy().sum()).round(1))

ax = dfw.transpose().plot.bar(figsize=(5, 4), legend=False)
ax.set_xlabel('Cause of Delay'); ax.set_ylabel('Count')
```

Pearson Correlation Coefficient (r)

Measures the direction and strength of the linear relationship between two numeric variables.

Pearson r	Strength	Interpretation
$r = 1$	Perfect positive	As one increases, the other increases proportionally
$0.7 < r < 1$	Strong positive	Clear co-movement
$0.3 < r < 0.7$	Moderate	Trend visible but noisy
$-0.3 < r < 0.3$	Weak / none	Little evidence of linear relationship
$r = -1$	Perfect negative	As one increases, the other decreases proportionally

CRITICAL: Correlation $\neq$ Causation. Two variables can be strongly correlated due to a common confound, reverse causality, or pure coincidence. Causal inference requires experiments or causal inference methods.

Heatmaps and Scatterplots

Correlation Heatmap

Visual encoding of the full correlation matrix using color:

- Dark red: strong positive
- Dark blue: strong negative
- White/light: weak or none

Scatterplot Interpretation

Shows the shape, direction, and strength of the relationship — including non-linear relationships r cannot capture.

Also reveals:

- Clusters and subgroups

```
# Heatmap
sns.heatmap(etfs.corr(), vmin=-1, vmax=1,
            cmap=sns.diverging_palette(20, 220, as_cmap=True), square=True)

# Scatterplot
telecom.plot.scatter(x='T', y='VZ', figsize=(5, 5), alpha=0.5)
```

(e.g., parabolic patterns).



PART V

Multivariate Visualization

Hexagonal Binning (Hexbin)

With 432,000+ observations, standard scatterplots become unreadable (overplotting). Hexbin encodes density as color.

Why Hexbin?

- Prevents overplotting in large datasets
- Hexagonal cells share fewer edges than squares → smoother appearance
- Color intensity = observation density
- Easy to interpret: dark = crowded

King County Housing Data

kc_tax dataset: 432,000+ observations
SqFtTotLiving vs. TaxAssessedValue

Filtered to reasonable ranges:

SqFtTotLiving: 100–3500

TaxAssessedValue: < 750,000

This filtering is a key EDA step —
extreme values obscure the pattern.

```
kc_tax0 = kc_tax.loc[(kc_tax.TaxAssessedValue < 750_000) &  
                    (kc_tax.SqFtTotLiving > 100) &  
                    (kc_tax.SqFtTotLiving < 3500), :]  
ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',  
                        gridsize=30, sharex=False, figsize=(6, 4))
```

2D KDE Contours and Contingency Tables

2D Kernel Density Estimate

The two-dimensional extension of KDE.
Places a 2D Gaussian kernel at each point;
the sum yields a smooth density surface.

Contour lines = equal-density curves
(like altitude lines on a topographic map).

For large data: sample before computing
to manage computation time.

Contingency Tables (Crosstabs)

Shows joint distribution of two
categorical variables.

LendingClub: grade vs. repayment status

Finding: Default rate increases
monotonically from A (2.2%) to G (12.6%).

This validates the grading system as a
meaningful risk classifier — analogous to
testing whether a severity classification

```
# 2D KDE
ax = sns.kdeplot(data=kc_tax0.sample(10_000, random_state=42),
                 x='SqFtTotLiving', y='TaxAssessedValue')

# Contingency table
crosstab = lc_loans.pivot_table(index='grade', columns='status', aggfunc=len, margins=True)
```

Violin Plots and FacetGrid

Violin Plot

Shows the full density shape of each group — richer than a boxplot.

Reveals features boxplots hide:

- Bimodality
- Heavy tails
- Asymmetry within the IQR

Useful for ANOVA pre-diagnostics (Week 9):
are group distributions comparable?

FacetGrid — Conditional Visualization

Applies the same plot type to categorical subsets, enabling direct structural comparison across groups.

King County example by zip code:
slope, spread, and density center
all vary by neighborhood.

This heterogeneity is invisible in a
single aggregate plot.

```
# Violin plot
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay', inner='quartile')

# FacetGrid
g = sns.FacetGrid(kc_tax_zip, col='ZipCode', col_wrap=2, height=3.5)
g.map(hexbin, 'SqFtTotLiving', 'TaxAssessedValue', extent=[0, 3500, 0, 700_000])
```



PART VI

AI-Assisted Exploration (3rd Edition)

Interpreting Visualizations with AI

Multimodal AI tools (ChatGPT, Claude, Gemini) can help interpret complex statistical visualizations.

Step	Action	Example Prompt
1. Create	Build the chart in Python	—
2. Submit	Paste into an AI tool	"Interpret this figure in one paragraph"
3. Deepen	Ask follow-up questions	"What do the two clusters above the main band mean?"
4. Condition	Suggest stratification	"What if we facet by location?"
5. Summarize	Request report-ready text	"Summarize findings concisely"

CRITICAL: AI interpretations are a starting point, not the final analysis. Every output must be validated with domain knowledge. The hallucination risk applies to visuals too — AI may “see” patterns that don’t exist.

Creating Data Dictionaries with AI

Traditional Approach

Manual data dictionary creation:

1. Inspect each column individually
2. Determine data type
3. Identify unit and valid range
4. Write description
5. Document missing data encoding

This process can take hours for large datasets and is prone to errors.

AI-Assisted Approach (3rd Ed.)

Feed DataFrame structure to an LLM:

Example prompt:

"Create a data dictionary for this DataFrame: type, unit, range, description for each column."

Advantages:

- ✓ Fast drafts, standard format
- X May miss domain-specific nuances
- X Always requires expert validation

The 6-Step Exploratory Analysis Pipeline

#	Stage	Python Tools	Question Answered
1	Load Data	<code>pd.read_csv()</code> , <code>pd.read_excel()</code>	Where is the data and in what format?
2	Inspect	<code>shape</code> , <code>dtypes</code> , <code>head()</code> , <code>describe()</code>	What does the data look like?
3	Clean	<code>dropna()</code> , <code>fillna()</code> , <code>astype()</code>	Can we trust it?
4	Feature Engineering	new columns, <code>cut()</code> , <code>apply()</code>	Can we create more informative variables?
5	Summarize	<code>groupby()</code> , <code>agg()</code> , <code>crosstab()</code>	Are there group differences?
6	Visualize	<code>hist()</code> , <code>boxplot()</code> , <code>scatter()</code> , <code>heatmap()</code>	What are the distributions and relationships?

This 6-step workflow is applied every week. The process is non-linear — EDA often sends you back to cleaning; modeling may reveal the need for additional data.

seaborn — Statistical Visualization Library

Function	Purpose	Usage
<code>sns.histplot()</code>	Histogram + optional KDE	<code>sns.histplot(data=df, x='col', kde=True)</code>
<code>sns.boxplot()</code>	Grouped boxplot	<code>sns.boxplot(data=df, x='cat', y='num')</code>
<code>sns.violinplot()</code>	Full distribution shape	<code>sns.violinplot(data=df, x='cat', y='num')</code>
<code>sns.scatterplot()</code>	Scatter with color/size	<code>sns.scatterplot(data=df, x='x', y='y', hue='cat')</code>
<code>sns.heatmap()</code>	Correlation heatmap	<code>sns.heatmap(df.corr(), annot=True)</code>
<code>sns.pairplot()</code>	All pairwise scatters	<code>sns.pairplot(df[['a', 'b', 'c']])</code>
<code>sns.kdeplot()</code>	1D or 2D density	<code>sns.kdeplot(data=df, x='a', y='b')</code>
<code>sns.FacetGrid()</code>	Conditional multi-panel	<code>g = sns.FacetGrid(df, col='cat')</code>

Hands-On: EDA Pipeline

Open the Week 2 Colab notebook and complete the following:

1. Compute mean, median, and trimmed mean (5% and 20%) for Murder.Rate.
Compare results and explain the differences.
2. For airline_stats, compute std, IQR, and MAD of pct_carrier_delay
per airline using groupby. Which airline has the most variable performance?
3. Create a histogram of SqFtTotLiving from kc_tax0 (bins=50).
Describe the shape: symmetric, skewed, unimodal, bimodal? Overlay KDE.
4. Select any two ETFs from the etfs dataset. Create a scatterplot,
compute Pearson correlation, and write a 2–3 sentence interpretation.
5. AI Exploration: Have AI interpret one of your visualizations.
Ask 2 follow-up questions. Critically evaluate the AI's output.

Assignment 2 — Exploratory Data Analysis

1. Load the tips dataset (seaborn). Compute ALL location and variability estimates for total_bill and tip. Report as a formatted table.
2. Create $\text{tip_ratio} = \text{tip} / \text{total_bill} \times 100$. Plot histogram (bins=30, with KDE). Describe the distribution shape.
3. Create grouped boxplots AND violin plots by day.
Which day has higher and more variable tipping?
4. Scatterplot: total_bill vs. tip. Compute Pearson AND Spearman correlations. Explain why they might differ.

Deadline: before Week 3 class • Submission: Moodle Assignment • Late penalty: -20% per day (max 3 days)

Discuss distributional differences between groups.

6. AI Task: Have AI interpret your violin plot. Ask 2 follow-ups.
Write one critical paragraph evaluating the AI's response.

Frequent Errors in EDA

#	Mistake	Solution
1	Deleting outliers without investigation	Always investigate why a value is extreme before removing
2	Relying on a single location estimate	Report mean, median, and trimmed mean together
3	Not optimizing histogram bin count	Try bins=15, 30, 50; use bins='auto'
4	Interpreting correlation as causation	Always caveat: correlation $\neq$ causation
5	Using scatterplots for large datasets	For 10K+ observations: hexbin or 2D KDE
6	Ignoring distributional shape	Plot histogram + boxplot for every numeric variable

What You Should Know After This Week

Conceptual

- ✓ Define mean, median, trimmed mean and their weighted variants
- ✓ Explain robustness differences: std deviation vs. MAD vs. IQR
- ✓ Distinguish what histograms, boxplots, and KDE each reveal
- ✓ Interpret correlation coefficient and recognize the causation fallacy
- ✓ Intuitively understand skewness and kurtosis from visual displays

Practical

- ✓ Compute all location and variability estimates in Python
- ✓ Create histograms, KDE, boxplots, violin plots, scatterplots, heatmaps
- ✓ Build frequency tables and crosstabs
- ✓ Use hexbin and FacetGrid for large or grouped datasets
- ✓ Critically use AI tools for visualization interpretation

Week 3: Inference Basics — Permutation Tests and p-values

Preparation

- Read Bruce et al., Chapter 2 (Data and Sampling Distributions) and Chapter 3 (Statistical Experiments)
- Complete Assignment 2 and submit via Moodle
- Think about this question: In an A/B test, how do you determine whether the observed difference is real or just due to chance? Next week we answer this with permutation tests.

Systems engineering connection: Permutation tests are distribution-free — they don't require normality assumptions that real-world engineering data often violates.