Mid RC for STAT4710J

by Youchen Qing 2024/4/1 Lecture 10-18

Visualization (Lecture 10-11)

GOAL:

1. To help your own understanding of your data/results.
2. To communicate results/conclusions to others.

Distribution

Probability distribution or probability density function is a function f_X associated to a discrete random variable X defined as

$f_X : \Omega \to \mathbb{R}$

with Ω a countable subset of $\mathbb{R}$ satisfying the properties that

(i) $f_X(x) \geq 0 \, \forall x \in \Omega$ and

(ii) $\sum_{x \in \Omega} f_X(x) = 1.$

For a continuous random variable X the probability distribution is defined as

$f_X : \mathbb{R} \to \mathbb{R}$

with the properties that

(i) $f_X(x) \geq 0 \, \forall x \in \mathbb{R}$ and

(ii) $\int_{x \in \mathbb{R}} f_X(x) \, dx = 1.$

Bar Plot

A **Bar plot** shows the relationship between a numeric and a categorical variable, with:

- Entities represented as bars.
- Values represented as the size of the bars.

In Seaborn

For plotting the number of occurrences in each category:

```
seaborn.countplot(data=None, x=None, y=None, hue=None)
```

For a general bar plot:

```
seaborn.barplot(data=None, x=None, y=None, hue=None, estimator='mean')
```

The hue argument in Seaborn is used for colour encoding.

Histogram

Vertical bar graph

- Each bar represents the proportion or number of data in a given range
- Categories are called bins

Skewness and tail

- Tail on the left—left skewed
- Tail on the right—right skewed

Mode

- Local or global maximum
- Number mostly depends on the density curve (KDE)

```
n = w × h × N
```

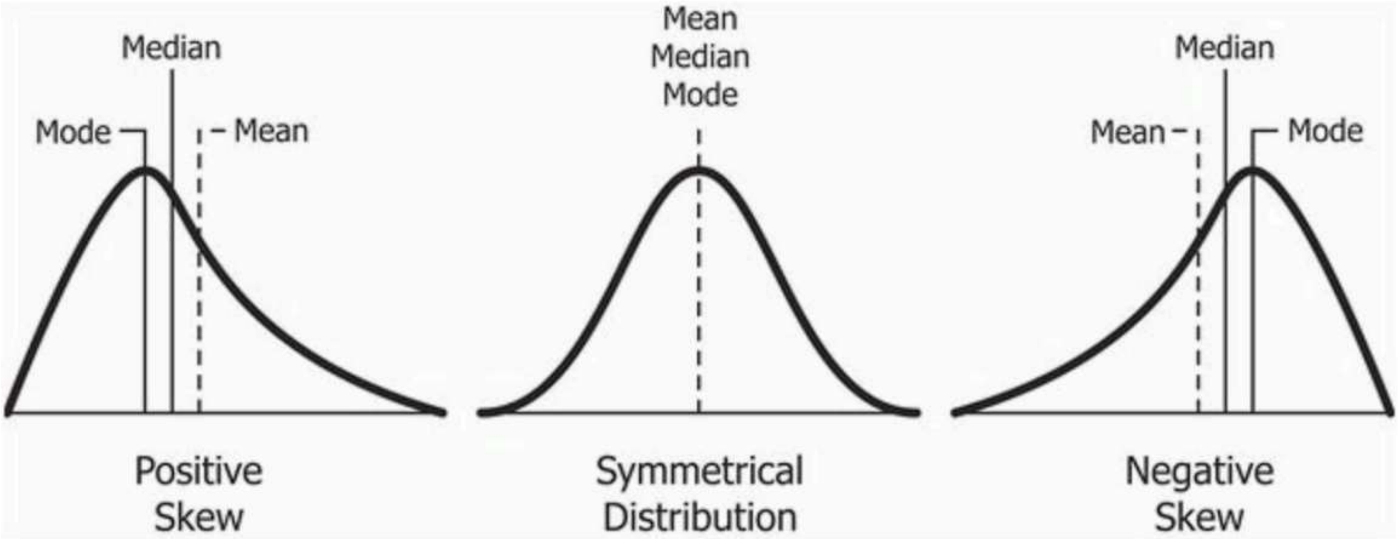
- `n` : number of samples in a bin
- `w` : bin width
- `h` : bar height
- `N` : total number of samples

Right skewed (positive skewness)

- Mode < Median < Mean

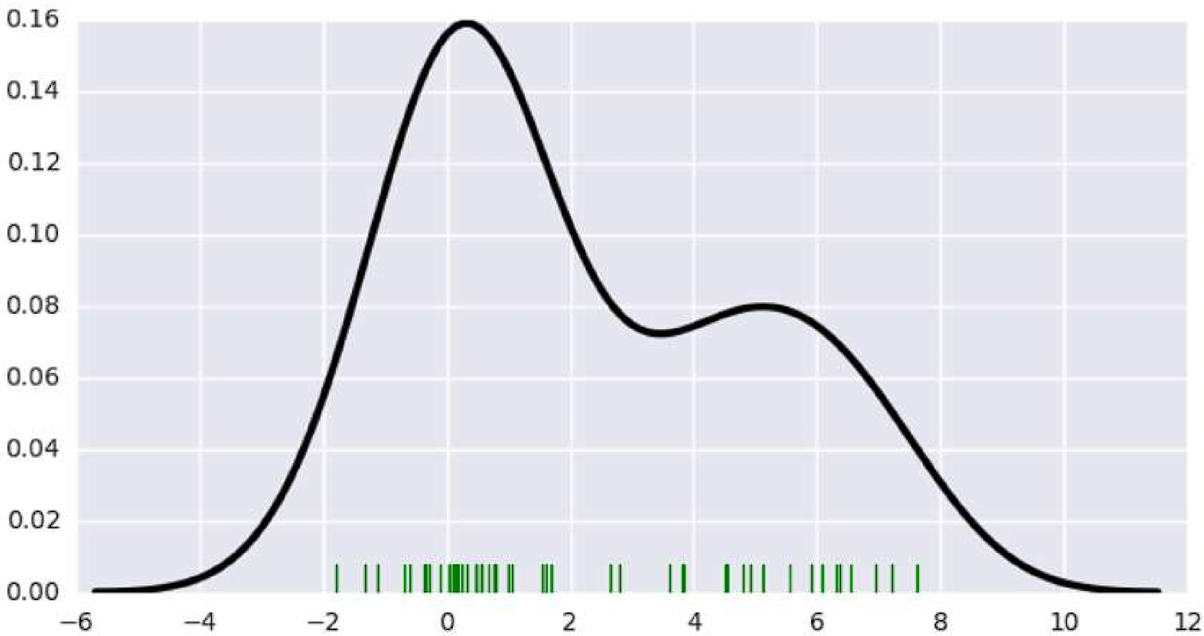
Left skewed (negative skewness)

- Mean < Median < Mode



Rug plot

- Displayed as marks along an axis
- Could be seen as 1-D scatter plot



KDE

A kernel density estimation (KDE) is used to estimate a probability density function (distribution).

Three steps

1. Place a kernel at each data point
2. Normalize kernels
   - Divide each kernel by the number of kernels in total
3. Sum kernels

Quartiles

- 25% of the data are no greater than the first quartile q_1
- 50% of the data are no greater than the second quartile q_2
- 75% of the data are no greater than the third quartile q_3

Interquartile range (IQR)

$IQR = q_3 - q_1$

Box plot

Fences

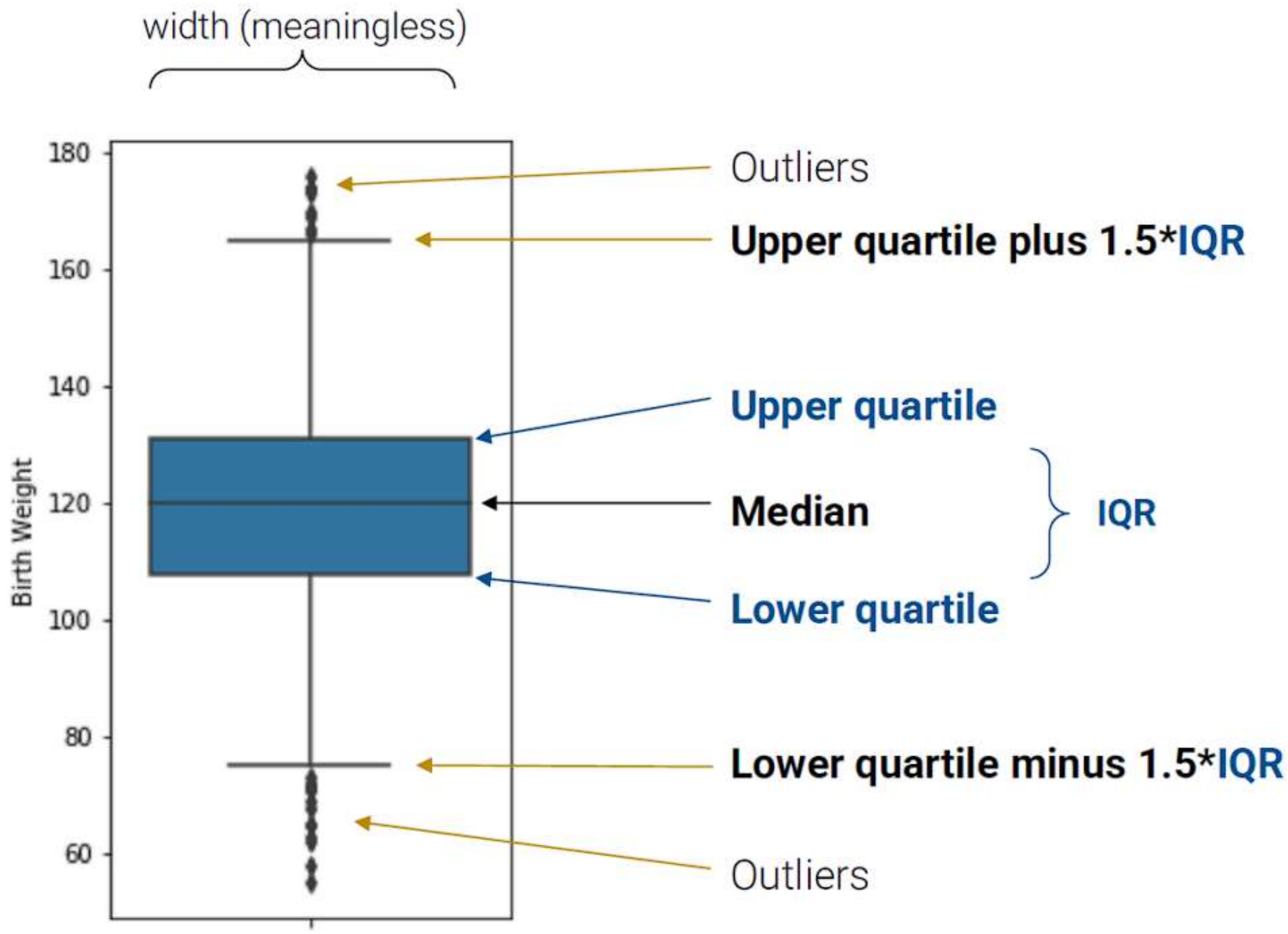
- Inner fences
  - $f_1 = q_1 - 1.5 \times IQR$
  - $f_3 = q_3 + 1.5 \times IQR$
- Outer fences
  - $F_1 = q_1 - 3 \times IQR$
  - $F_3 = q_3 + 3 \times IQR$

Adjacent values (where the line extending to the left and right of the box ends)

- $a_1 = \min\{x_k : x_k \geq f_1\}$
- $a_3 = \max\{x_k : x_k \leq f_3\}$

Outliers

- Near outliers: outside the inner fences but inside the outer fences
- Far outliers: outside the outer fences

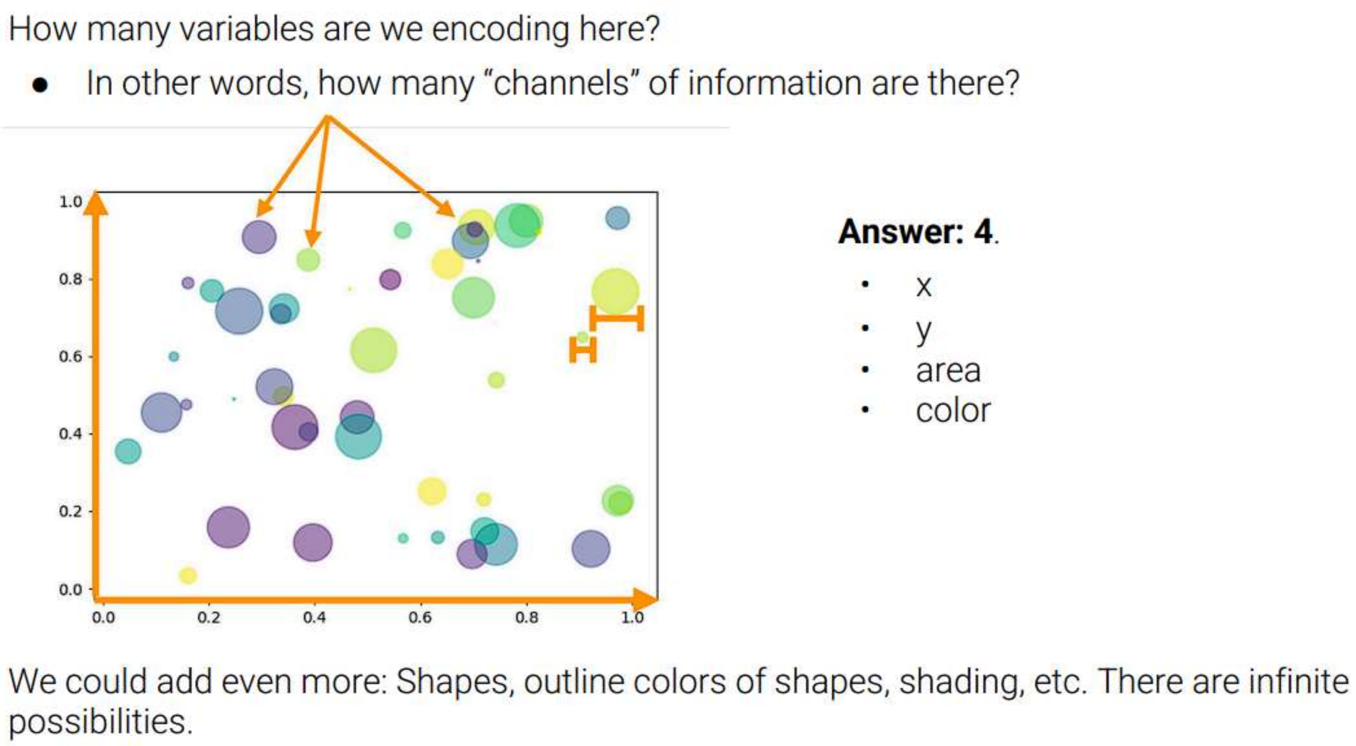


Choosing methods of visualization accordingly

- Comparing quantitative distributions:
  - overlaid histograms and density curves (unclearness vs. completeness)
  - side by side box/violin plots
- Relationships between quantitative variables:
  - Previously (histogram: number/frequency of value)
  - scatter plot: relationship between pairs of numeric variables
  - scatter plot suffers overplotting
    - add some random noise to x variable
    - adjust the transparency (alpha)
  - hex plot:
    - why use hexagons instead of squares?
    - marginal distribution are shown as histograms
  - contour plot:
    - 2 dimensional version of density curves
    - marginal distribution: density curve
- Interactive data visualization (for the web)

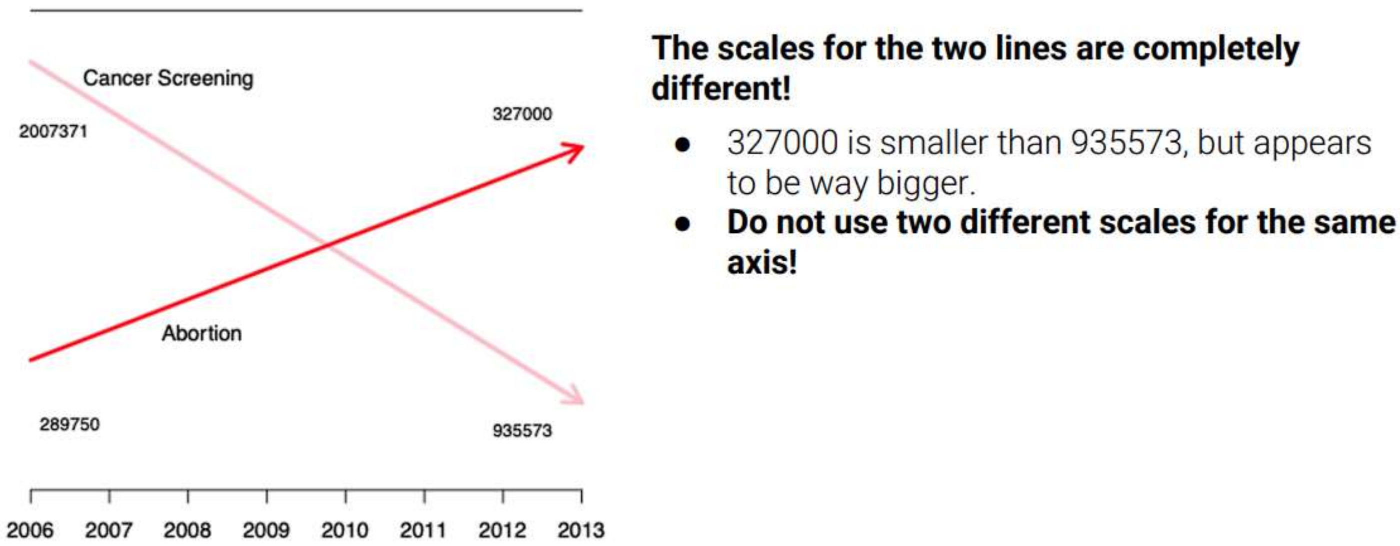
Visualization Theory

- **Information channels**



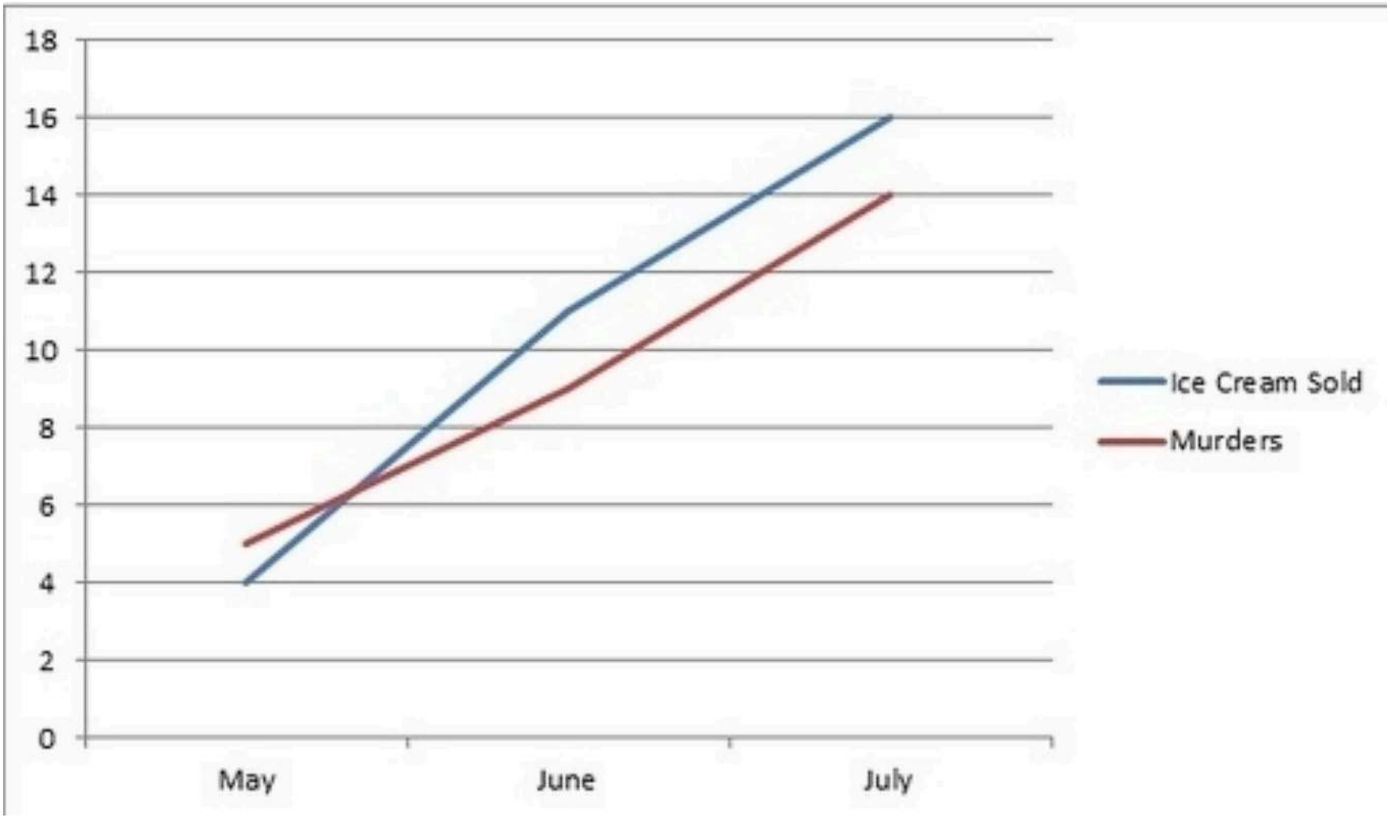
- **Axis**

Keep axis scales consistent (try to choose axis limits to fill the visualization)



- **Color**
- Perceptually uniform colormaps have the property that if the data goes from 0.1 to 0.2, the perceptual change is the same as when the data goes from 0.8 to 0.9.
- Qualitative:
  - Choose a qualitative scheme that makes it easy to distinguish between categories.
  - One category isn't "higher" or "lower" than another.
- Quantitative: Choose a color scheme that implies magnitude
- **Markings**
  - Lengths are easy to distinguish; angles are hard

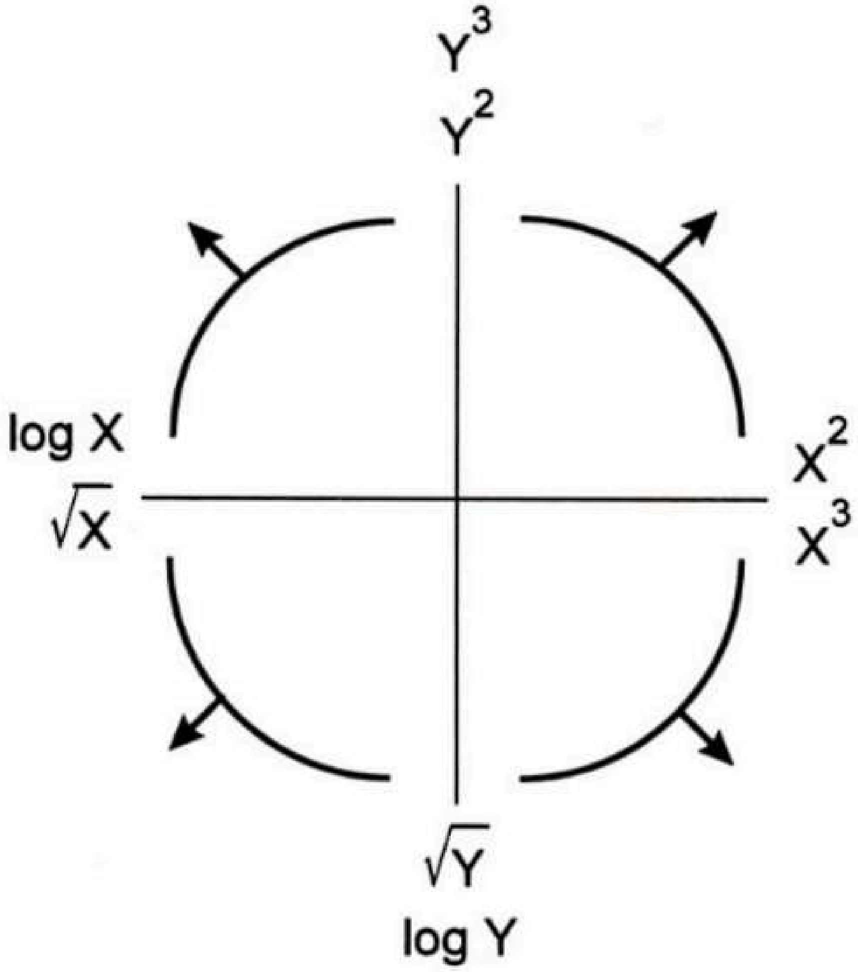
    - Avoid jiggling the baselines
- **Give informative titles, axis, labels, legends**
- **The plots make sense**



Tukey's Bulging Rule

Idea: Transform the values on the axes to get a (almost) linear relation.

General strategy:



The diagram illustrates how different transformations applied to the X and Y axes can linearize different types of relationships between variables. This strategy is used to identify the right transformation for achieving linearity in regression analysis.

Modeling

Loosely speaking, a (machine learning) model is a function f defined as

$f : A \to B$

with A the feature space and B the target space.

Feature space: Variables you use to train the model
Target space: Variables you want to study

In machine learning, all we do is to find a f that makes most sense.

Feature Engineering

- Select features from the feature space
- Create new features that are not originally in the feature space
- Enhance model performance

Encoding

Ways to encode categorical variables

- Label Encoding
- One-Hot Encoding
- Hash Encoding
- Target Encoding

Ways to encode words

- Bag-of-words encoding
  - Based on measuring the similarity between vectors
- TF-IDF
  - Take into account the importance of words

Choose the strategy that makes most sense.

Loss function and objective function

A loss function *L* is defined as

$L : T \to \mathbb{R}$

with *T* the tuple space of (y, ŷ).

- y: ground truth value
- ŷ: predicted value

An objective function is either a loss function or its opposite. We want to minimise (or maximise) the objective function to get the optimal model.

Simple linear regression (SLR)

- Model: $\hat{y} = a + bx$
- Loss function: $L(y, \hat{y}) = (y - \hat{y})^2$
- Objective function: $R(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, \hat{y}^{(i)})$
- Optimal solution:
  - $\hat{b} = r \frac{\sigma_y}{\sigma_x}$
  - $\hat{a} = \bar{y} - \hat{b}\bar{x}$

r is the correlation,
$r = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x^{(i)} - \bar{x}}{\sigma_x} \right) \left(\frac{y^{(i)} - \bar{y}}{\sigma_y} \right)$

Properties

- Passes $(\bar{x}, \bar{y})$

- Residuals sum up to 0

Constant model

- Model: $\hat{y} = \theta$
- Loss function: mean squared error (MSE) or mean absolute error (MAE)
- Optimal solution:
  - $\hat{\theta} = \text{mean}(y)$ for MSE
  - $\hat{\theta} = \text{median}(y)$ for MAE

Pros and cons

- With MSE:
  - Pros: differentiable; Easy to find the optimum
  - Cons: sensitive to outliers
- With MAE:
  - Pros: robust to outliers
  - Cons: piece-wise function; hard to find the optimum

General linear regression

Linear least squares

- Model: $\hat{y} = \sum_{j=1}^{m} \theta_j x_j$ (assuming $x_{[1]} = 1$, which is the bias term)
- Loss function: MSE
- Optimal solution: $\hat{\theta} = (X^T X)^{-1} X^T Y$

Properties:

- Residuals sum up to 0 (only when there is a bias term)
- $\hat{\theta}$ is unique $\iff$ X is column full rank

Keep the sample space larger than the feature space.

OLS derivation

Optimisation problem

minimize over θ:
$\min_\theta ||Y - X\theta||_2$

The loss function is
$L(\theta) = ||Y - X\theta||_2$
$= (Y - X\theta)^T (Y - X\theta)$
$= Y^T Y - Y^T X\theta - \theta^T X^T Y + \theta^T X^T X$

Set the gradient of L to 0
$\frac{\partial L(\theta)}{\partial \theta} = \frac{\partial (Y^T Y - Y^T X\theta - \theta^T X^T Y + \theta^T X^T X)}{\partial \theta}$
$= -2X^T Y + 2X^T X\theta = 0$

Hence,
$\hat{\theta} = (X^T X)^{-1} X^T Y$

You can refer to the slides for linear operations

Logistic Regression

Optimization problem

The goal is to find the best parameters θ that minimize the logistic regression cost function, which is the binary cross-entropy loss for a dataset with m examples:

$\min_\theta J(\theta)$

The loss function $J(\theta)$ for logistic regression is defined as:

$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$

where $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ is the hypothesis function for logistic regression.

The gradient of the loss function with respect to θ is:

$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T (h_\theta(X) - Y)$

Where:

- X is the matrix of input features,
- Y is the vector of observed outputs (0 or 1),
- $h_\theta(X)$ is the vector of predicted probabilities.

The parameters θ are updated iteratively using an optimization algorithm such as gradient descent:

$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$

where α is the learning rate.

Properties:

- Output is a probability that the given input point belongs to the class labeled as "1".
- Logistic regression models a binary outcome.

Cross Validation Summary

When selecting between models, we want to pick the one that we believe would generalize best on unseen data. Generalization is estimated with a "cross validation score". When selecting between models, keep the model with the best **score**.

Two techniques to compute a "cross validation score":

- The Holdout Method: Break data into a separate **training set** and **validation set**.
  - Use **training set** to fit parameters (thetas) for the model.
  - Use **validation set** to score the model.
  - Also called "Simple Cross Validation" in some sources.
- K-Fold Cross Validation: Break data into K contiguous non-overlapping "folds".
  - Perform K rounds of Simple Cross Validation, except:
    - Each fold gets to be the **validation set** exactly once.
    - The final **score** of a model is the average validation score across the K trials.

SQL

```
SELECT [DISTINCT] <column expression list>
FROM <table>
[WHERE <predicate>]
[GROUP BY <column list>]
[HAVING <predicate>]
[ORDER BY <column list>]
[LIMIT <number of rows>]
[OFFSET <number of rows>];
```

Note: Column Expressions may include aggregation functions (MAX, MIN, etc) and DISTINCT.

SQL Joins

INNER JOIN

An `INNER JOIN` retrieves records that have matching values in both tables. It returns rows when there is at least one match in both tables, excluding rows with no match.

FULL OUTER JOIN

A `FULL OUTER JOIN` returns all records when there is a match in the left, right, or both tables. It combines the results of both `LEFT` and `RIGHT OUTER JOIN`.

CROSS JOIN

A `CROSS JOIN` produces the Cartesian product of two tables, combining each row from the first table with each row from the second table.

LEFT OUTER JOIN (or LEFT JOIN)

A `LEFT OUTER JOIN` returns all records from the left table and the matched records from the right table. If there is no match, the result from the right side will be `NULL`.

RIGHT OUTER JOIN (or RIGHT JOIN)

A `RIGHT OUTER JOIN` returns all records from the right table and the matched records from the left table. If there is no match, the result from the left side will be `NULL`.

Reference

Mid_RC_Part2 Yuxuan Zheng
Mid_RC_note Sizhe Zhou
lecture slides 2023 summer