Vector Basics

Vector Norm

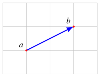
$\|x\| = \sqrt{x_1^2 + x_2^2 + ... + x_3^2}$

- length of the vector. **Geometric Interpretation**: Distance from the origin to the point presented by the vector x

Vector Distance

$\|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$

aka *Euclidean Distance*
Geometric Interpretation: Straight-line distance between the points represented by vectors



Vector Products

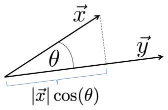
Element-wise (Hadamard Product)

$\vec{a} \cdot \vec{b} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \end{pmatrix}$

Inner Product (Dot Product)

$\vec{x} \cdot \vec{y} = \begin{pmatrix} x_1 & x_2 & ... & x_N \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = x_1 y_1 + x_2 y_2 + ... + x_N y_N$

Represents:
- magnitude of one vector projected onto another.
- Angle θ between two non-zero vectors
- $\vec{x} \cdot \vec{y} = |\vec{x}| \, |\vec{y}| \cos(\theta)$



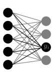
- **Several cases of θ for some vector a and b:**
  - $\theta = \frac{\pi}{2} = 90°$: a and b are orthogonal, i.e., $a \perp b$
  - $\theta = 0$: a and b are aligned. $a^T b = \|a\| \cdot \|b\|$
  - $\theta = \pi = 180°$: a and b are anti-aligned. $a^T b = -\|a\| \cdot \|b\|$
  - $\theta \in (0, \frac{\pi}{2})$: a and b make an acute angle. $a^T b > 0$.
  - $\theta \in (\frac{\pi}{2}, \pi)$: a and b make an obtuse angle. $a^T b < 0$.

Outer Product

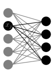
$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \cdot \begin{pmatrix} y_1 & y_2 & ... & y_M \end{pmatrix} = \begin{pmatrix} x_1 y_1 & x_1 y_2 & ... & x_1 y_M \\ x_2 y_1 & x_2 y_2 & ... & x_2 y_M \\ \vdots & \vdots & \ddots & \vdots \\ x_N y_1 & x_N y_2 & ... & x_N y_M \end{pmatrix}$$

Matrix Basics

Matrix × Vector (Inner Product / Outer Product)



$y_i = W_{i1} x_1 + W_{i2} x_2 + ... + W_{iN} x_N$



$\vec{y} = x_1 \overrightarrow{W}^{(1)} + x_2 \overrightarrow{W}^{(2)} + ... + x_N \overrightarrow{W}^{(N)}$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_M \end{pmatrix} = \begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1N} \\ W_{21} & W_{22} & \cdots & W_{2N} \\ \vdots & \vdots & & \vdots \\ W_{i1} & W_{i2} & \cdots & W_{iN} \\ \vdots & \vdots & \ddots & \vdots \\ W_{M1} & W_{M2} & \cdots & W_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$$

$$\begin{array}{cccc} (\ \overrightarrow{W}^{(1)} & \overrightarrow{W}^{(2)} & & \overrightarrow{W}^{(N)}\ ) \\ \downarrow & \downarrow & & \downarrow \end{array}$$

$$\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1N} \\ W_{21} & W_{22} & \cdots & W_{2N} \\ \vdots & \vdots & & \vdots \\ W_{M1} & W_{M2} & \cdots & W_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = x_1 \overrightarrow{W}^{(1)} + x_2 \overrightarrow{W}^{(2)} + \cdots + x_N \overrightarrow{W}^{(N)}$$

Matrix × Matrix (Inner Product / Outer Product)

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1P} \\ A_{21} & A_{22} & \cdots & A_{2P} \\ \vdots & \vdots & & \vdots \\ A_{i1} & A_{i2} & \cdots & A_{iP} \\ \vdots & \vdots & & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NP} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1j} & \cdots & B_{1M} \\ B_{21} & B_{22} & \cdots & B_{2j} & \cdots & B_{2M} \\ \vdots & \vdots & & \vdots & & \vdots \\ B_{P1} & B_{P2} & \cdots & B_{Pj} & \cdots & B_{PM} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1M} \\ C_{21} & C_{22} & \cdots & C_{2M} \\ \vdots & \vdots & C_{ij} & \vdots \\ C_{N1} & C_{N2} & \cdots & C_{NM} \end{pmatrix}$$

- C_{ij} is directly computed (similar to mat×vec inner product)
- $C_{ij} = \sum_{k=1}^{P} A_{ik} B_{kj}$: `for(Arow) for(Bcol) Cij = Arow * Bcol`

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1P} \\ A_{21} & A_{22} & \cdots & A_{2P} \\ \vdots & \vdots & & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NP} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1M} \\ B_{21} & B_{22} & \cdots & B_{2M} \\ \vdots & \vdots & & \vdots \\ B_{P1} & B_{P2} & \cdots & B_{PM} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1M} \\ C_{21} & C_{22} & \cdots & C_{2M} \\ \vdots & \vdots & & \vdots \\ C_{N1} & C_{N2} & \cdots & C_{NM} \end{pmatrix}$$

- A temporary $N \times M$ matrix is computed at each iteration
- C (final mat) $= \sum_{k=1}^{P} A^{(k)} B^{k^T}$: `for(A col) for(B row) C_tmp`

Special Matrices

Square Matrix

- Each column represents the new **standard basis vector**
- $\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$ becomes 1st column, $\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$ becomes 2nd column ...

Orthogonal Matrix

$$AA^T = \begin{bmatrix} | & | & & | \\ \vec{a}_1 & \vec{a}_2 & ... & \vec{a}_n \\ | & | & & | \end{bmatrix} \cdot \begin{bmatrix} — & \vec{a}_1^T & — \\ — & \vec{a}_2^T & — \\ & \vdots & \\ — & \vec{a}_n^T & — \end{bmatrix} = \begin{bmatrix} \vec{a}_1 \cdot \vec{a}_1 & \vec{a}_1 \cdot \vec{a}_2 & ... & \vec{a}_1 \cdot \vec{a}_n \\ \vec{a}_2 \cdot \vec{a}_1 & \vec{a}_2 \cdot \vec{a}_2 & ... & \vec{a}_2 \cdot \vec{a}_n \\ \vdots & \vdots & \ddots & \vdots \\ \vec{a}_n \cdot \vec{a}_n & \vec{a}_n \cdot \vec{a}_2 & ... & \vec{a}_n \cdot \vec{a}_n \end{bmatrix}$$

$$A^{-1} = A^T \qquad AA^{-1} = AA^T \qquad AA^T = I$$

$\vec{a}_i \cdot \vec{a}_i = 1$
$\vec{a}_i \cdot \vec{a}_j = 0$ for $i \neq j$
So $\vec{a}_i, \vec{a}_j$ at right angle

$\therefore$ new basis vectors are perpendicular to each other. i.e. **rotation**.

Symetric Matrix

- Definition: $A^T = A$ (symmetric along the diagonal)
- Eigenvectors are orthogonal of each other

Determinant

- Definition: factor by which a matrix scales the 2D plane, a 3D volume, etc.
- Negative: squash space into 0 then expand again (flip of 2D plane in 3D)
- Zero: reduce dimensions (2D space → 1D line / 0D dot)

2×2 Matrix

$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ $\det(A) = ad - bc$

if $\det(A) \neq 0 \Rightarrow$
$A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

3×3 Matrix

$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$

$\det(A) = a \cdot \det\begin{pmatrix} e & f \\ h & i \end{pmatrix} - b \cdot \det\begin{pmatrix} d & f \\ g & i \end{pmatrix} + c \cdot \det\begin{pmatrix} d & e \\ g & h \end{pmatrix}$

General Formulas

$\det(A^{-1}) = \frac{1}{\det(A)} = \det(A)^{-1}$
$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij})$ (it's recursive)

- a_{ij} is row i col j of A, and A_{ij} is the submatrix after removing row i col j

Eigenvector & Eigenvalues

Eigenvalue

$$A\vec{v} = \lambda\vec{v}$$
$$A\vec{v} = (\lambda I)\vec{v}$$
$$(A - \lambda I)\vec{v} = \vec{0}$$

$$A - \lambda I = \begin{bmatrix} a-\lambda & b & c \\ d & e-\lambda & f \\ g & h & i-\lambda \end{bmatrix}$$

λ: eigenvalue
$\vec{v}$: eigenvector
$\because (M - \lambda I)$ map vectors $\vec{v}$ to $\vec{0}$, $\therefore$ $\det(M - \lambda I) = 0$

So for 2×2 matrix: $\det\left(\begin{bmatrix} a-\lambda & b \\ c & d-\lambda \end{bmatrix}\right) = 0, (a - \lambda)(d - \lambda) - bc = 0$

Eigenvector

After solving eigenvalue λ, substitute λ back to the matrix $(M - \lambda I)$

$$(M - \lambda I)\vec{v} = \vec{0}$$
$$\begin{pmatrix} a-\lambda & b \\ c & d-\lambda \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Solve $\begin{cases} (a - \lambda)x + by = 0 \\ cx + (d - \lambda)y = 0 \end{cases}$

Eigen-Decomposition

1. A has two eigenvectors $\vec{v}_1, \vec{v}_2$

$$A\vec{v}_1 = \lambda_1 \vec{v}_1$$
$$A\vec{v}_2 = \lambda_2 \vec{v}_2$$

2. Combine $\vec{v}_1 \vec{v}_2$ to matrix $\mathbf{U} = [\vec{u}_1 \ \vec{u}_2]$

$$A \begin{bmatrix} | & | \\ \vec{u}_1 & \vec{u}_2 \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \vec{u}_1 & \vec{u}_2 \\ | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

3. $\mathbf{AU} = \mathbf{U\Lambda} \Rightarrow \mathbf{A} = \mathbf{U\Lambda U^{-1}}$

4. For systems of lin. equation $Ax = b$ substitute $A = \mathbf{U\Lambda U^{-1}}$

$$Ax = b$$
$$\mathbf{U\Lambda U^{-1}}x = b$$
$$\mathbf{U^{-1}}(\mathbf{U\Lambda U^{-1}})x = \mathbf{U^{-1}}b$$
$$\mathbf{\Lambda U^{-1}}x = \mathbf{U^{-1}}b$$

5. Let $y = \mathbf{U}^{-1}x$ and solve $\mathbf{\Lambda}y = \mathbf{U}^{-1}b \rightarrow$

$$\begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \lambda_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \lambda_1 y_1 \\ \lambda_2 y_2 \\ \vdots \\ \lambda_n y_n \end{bmatrix} = \mathbf{U}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

6. After finding y solve $x = \mathbf{U}y$

Singular Value Decomposition (SVD)

$$A_{m \times n} = \mathbf{U\Sigma V}^T = \begin{bmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & ... & \vec{u}_m \\ | & | & & | \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & 0 & ... & 0 \\ 0 & \sigma_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \sigma_r \\ 0 & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_n^T & — \end{bmatrix}$$

- Find eigenvectors of AA^T: $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_m\}$
- Find eigenvectors of $A^T A$: $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_n\}$
- Find shared eigenvalues between AA^T and $A^T A$: $\{\lambda_1, \lambda_2, ..., \lambda_r\}$
  - Where $r = \min(n, m)$
  - Then we get $\{\sigma_1, \sigma_2, ..., \sigma_r\} = \{\sqrt{\lambda_1}, \sqrt{\lambda_2}, ..., \sqrt{\lambda_r}\}$

Interpretations

- Larger σ_i implies a more **important** feature $\vec{u}_i$.
- In PCA, such eigenvector $\vec{u}_i$ is the direction of greatest variance.

Univariable Derivative

$\frac{\partial y}{\partial x} = \lim_{x \to \infty} \frac{f(x+\Delta x) - f(x)}{\Delta x}$ Simplify first, then take limits.

$$\frac{\partial}{\partial x}(f(x) \pm g(x)) = \frac{\partial f(x)}{\partial x} \pm \frac{\partial g(x)}{\partial x} \qquad \frac{\partial x^p}{\partial x} = px^{p-1}$$

$$\frac{\partial}{\partial x}(f(x)g(x)) = f'(x)g(x) + f(x)g'(x) \qquad \frac{\partial a^x}{\partial x} = a^x \ln(a)$$

$$\frac{\partial}{\partial x}\left(\frac{f(x)}{g(x)}\right) = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2} \qquad \frac{\partial \log_b x}{\partial x} = \frac{1}{x \ln(b)}$$

$$\frac{\partial}{\partial x}(f(g(x))) = f'(g(x)) \cdot g'(x) \qquad \frac{\partial \ln(x)}{\partial x} = \frac{1}{x}$$

Partial Derivative

vector → scalar

Let $y = f(\vec{x}), \vec{x} \in \mathbb{R}^N$ ($\mathbb{R}^N \to \mathbb{R}$ many-to-one function)

1. Compute $\left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right\}$
2. For each term, view all other variables as **constants**
   - e.g., when $\frac{\partial f}{\partial x_1}$, treat $x_2 ... x_n$ as constants
3. $\nabla_x f = \frac{\partial f}{\partial x}$

vector → vector (Jacobian Matrix)

- Definition: A matrix of all first-order derivatives of a vector-valued function
- Shows gradient of a **many-to-many function** at a specific point

Let $\vec{y} = f(\vec{x}), \vec{y} \in \mathbb{R}^M, \vec{x} \in \mathbb{R}^N$ ($\mathbb{R}^N \to \mathbb{R}^M$ many-to-many function)

$$\mathbf{J} = \nabla_x f = \frac{\partial f(x)}{\partial x} = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} & \cdots & \frac{\partial f(x)}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{m(x)}}{\partial x_1} & \cdots & \frac{\partial f_{m(x)}}{\partial x_n} \end{pmatrix}$$

Numerator Layout

$\frac{\partial \vec{y}}{\partial \vec{x}} \in \mathbb{R}^{M \times N}$ outputs × inputs $\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdot & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$

Denominator Layout

$\frac{\partial \vec{y}}{\partial \vec{x}} \in \mathbb{R}^{N \times M}$ inputs × outputs $\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$

Matrix Gradient

$\mathbf{A} \in \mathbb{R}^{m \times n}$ w.r.t $\mathbf{B} \in \mathbb{R}^{p \times q}$; dimensional tensor $J = \frac{\partial \mathbf{A}}{\partial \mathbf{B}} \in \mathbb{R}^{(m \times n) \times (p \times q)}$

$$\frac{\partial}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^\top = \left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)^\top \tag{5.99}$$
$$\frac{\partial}{\partial \boldsymbol{X}} \mathrm{tr}(\boldsymbol{f}(\boldsymbol{X})) = \mathrm{tr}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{5.100}$$
$$\frac{\partial}{\partial \boldsymbol{X}} \det(\boldsymbol{f}(\boldsymbol{X})) = \det(\boldsymbol{f}(\boldsymbol{X}))\mathrm{tr}\left(\boldsymbol{f}(\boldsymbol{X})^{-1} \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{5.101}$$
$$\frac{\partial}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^{-1} = -\boldsymbol{f}(\boldsymbol{X})^{-1} \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^{-1} \tag{5.102}$$
$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}^{-1} \boldsymbol{b}}{\partial \boldsymbol{X}} = -(\boldsymbol{X}^{-1})^\top \boldsymbol{a} \boldsymbol{b}^\top (\boldsymbol{X}^{-1})^\top \tag{5.103}$$
$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{5.104}$$
$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{5.105}$$
$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X} \boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a} \boldsymbol{b}^\top \tag{5.106}$$
$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^\top (\boldsymbol{B} + \boldsymbol{B}^\top) \tag{5.107}$$
$$\frac{\partial}{\partial \boldsymbol{s}} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = -2(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W} \boldsymbol{A} \quad \text{for symmetric } \boldsymbol{W} \tag{5.108}$$

Machine Learning

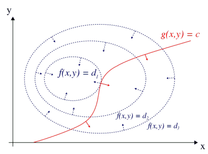
Binary CE: $L = \hat{y}\log(y) + (1 - \hat{y})\log(1 - y)$; $\frac{\partial L}{\partial y} = \frac{\hat{y}}{y} + \frac{1-\hat{y}}{y-1}$
Sigmoid Function: $y = \frac{1}{1+\exp(-z)}$; $\frac{\partial y}{\partial z} = \frac{\exp(-z)}{(\exp(-z)+1)^2}$
Linear Combination: $z = \sum_{i=0} w_i x_i$

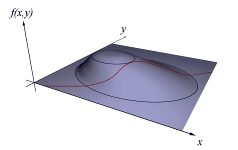
- $\frac{\partial z}{\partial w_i} = x_i$
- $\frac{\partial z}{\partial x_i} = w_i$

Lagrange Multiplier

Definition:
- maximize$_{x,y}$ $f(x, y)$
- subject to $g(x, y) = k$

As shown below, $f(x, y)$ forms a plane in 3D.
- Max f: scan top to bottom for first tangency of f, g
- Min f: scan bottom to top for first tangency of f, g





1. When f, g tangent, their gradient vectors are parallel, while differ differ by scalar λ (a.k.a lagrange multiplier).

$$\nabla f(x, y) = \lambda \nabla g(x, y)$$
$$L(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - k)$$

2. Set $L(x, y, \lambda) = 0$, to find its minimum point
3. Perform partial differentiation: $\left\{ \frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}, \frac{\partial L}{\partial \lambda} \right\} = \{0, 0, 0\}$
4. Finally, solve equation for x, y, λ

R Programming Language

R Basics

R Workspace

- Objects held in RAM
- Save workspace image: `save.image()`
- `ls(), rm(), getwd(), setwd()`
- Access general help with `help.start()`
- Get help on a function with `help(function)` or `?function`
- List functions containing a string: `apropos('string')`
- See an example of a function: `example(function)`

R Commands

- Assign using `=` or `<-`
- **Naming rules:**
  - Contain or start with dot(.); contain underscore;
  - Contain but NOT start with number
  - Case sensitive
- Comment lines start with `#`
- Check shadow naming / masking built-in function `conflicts()`

Datasets and Packages

- List datasets: `data()`
- Help on dataset: `help('dataset')`
- Install packages: `install.packages('package')`

Data Input & Manipulation

Data Types

Vectors
- numeric: `c(1,2,3)`
- character: `c('a','b','c')`
- logical: `c(T,F,T)`
- long integer `c(1L, 2L, 3L)`

Matrix
- `matrix(vec, nrow=r, ncol=c)`
- `byrow=T/F`, vec filled by row or col
- `dimnames=list(rowname, colname)` optional label for row, col

Arrays Like matrix, but can >2 dimensions
List Ordered collection of various objects (can be any types)

- `l[2]` return sub-list containing that element; `l[[2]]` return element

Data Frames

- `data.frame(col1,col2,..)`
- Access Methods: `df[1:2], df[c('col1','col2')], df$col1`
- Assign column names: `names(df) <- c('col1', 'col2', col3')`
- `attach(df)` - can directly access df variable `x` without need for `df$x`

Data Input

- CSV: `read.csv('/path/to/file.csv', header=T/F, sep=',')`
- Export delimited file: `write.table(data, 'file.txt', sep='\t')`

Viewing Data

- `ls()` list objects in the working environment
- `names(data)` list variables (e.g. col names of dataframe) in data
- `str(data)` list structure of data
- `dim(data)` dimensions of data
- `class(data)` list data types of an object
- `length(data)` number of elements or components in the object
- `head(data, n=2)` return the first 2 rows of data
- `tail(data, n=1)` return the last rows of data

Missing Data

- `x <- c(1,2,NA,3)`
- `mean(x)` returns NA, `mean(x, na.rm=T)` returns 2
- `complete.cases()` returns logical vector of rows without NA
- `df[!complete.cases(df)]` return complete rows
- `df <- na.omit(df)` create new dataset without the missing data

Manipulating Data

- `c(obj1, obj2, ...)` combine objects into a vector
- `cbind(obj1, obj2, ...)` nrow don't change, ncol++
- `rbind(obj1, obj2, ...)` ncol don't change, nrow++

Operators & Control Structure

- addition: `+`
- subtraction: `-`
- multiply: `*`
- division: `/`
- exponential: `^ / **`
- modulus: `x %% y`
- int division: `%/%`
- test if T: `isTRUE()`
- comparators: `<,<=,>,>=,==,!=,==`
- notj: `!x`
- `|,&` element-wise logical operator
- `||,&&` single element logical operator
- NOT bitwise operator
- `if(cond) 1 else 2`
- `for(var in seq) 1`
- `while(cond) 1`
- `switch(expr, default_case, case1=result1 ...)`
- `ifelse(cond, yes, no)` similar to a?b:c

Numeric Functions

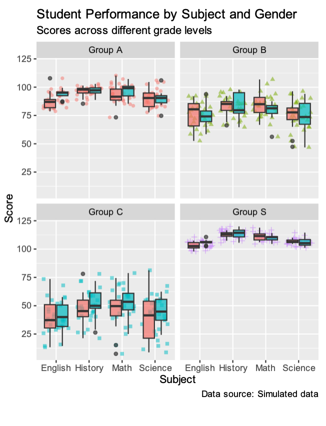
- absolute: `abs(x)`
- square root: `sqrt(x)`
- 3.4→4: `ceiling(x)`
- −3.4→−4: `floor(x)` round $-\infty$
- −3.4→−3: `trunc(x)` round to 0
- `round(3.475, digits=2)` 3.48
- `signif(3.475, digits=2)` 3.5
- `cos(),sin(),tan(),acos()..)`
- Natural log: `log(x)`
- Common log: `log10(x)`
- e^x: `exp(x)`
- `pretty(c(min, max), count)` evenly spaced numbers
- random variable: `rxxx(n, ...)`
- density function: `dxxx`
- cumulative: `pxxx` value → prob
- quantile: `qxxx` prob → value
- `xbinom(n,size,prob,lower.tai`
  - T=$P(X \leq x)$, F=$P(X > x)$
- `xnorm(_,m,sd)`
- `xpois(_,lambda)`
- `xunif(_,min,max)`

Plotting in R

- `plot(x)` plot scatterplot of x=index, y=x
- `plot(x, y)` plot scatterplot of x=x, y=y
  - `type`: p=point, l=lines, b=both
- `hist(x)` plot density histogram (distribution of x)
  - `breaks`: vector giving the breakpoints between histogram cells / function computing breakpoints / single value - number of cells / function computing num of cells

Plotting in `ggplot2`

```
ggplot(data, aes(x=subject, y=score, fill=gender)) +
  geom_point(
    aes(shape = class, color = class),
    position = "jitter",
    alpha = 0.5
  ) +
  geom_boxplot(alpha = 0.7) +
  facet_wrap(~class, ncol = 2) +
  labs(x = "Subject",
     y = "Score",
     color = "class",
     fill = "Gender",
     title = "Student Performance by Subject and Gender",
     subtitle = "Scores across different grade levels",
     caption = "Data source: Simulated data")
```




```
ggplot(student_data, aes(x = study_hours, y = score)) +
  geom_point(aes(color = class), alpha = 0.7) +
  geom_smooth(method = "lm") +
  labs(x = "Study Hours",
     y = "Score",
     color = "Class Group",
     title = "Score vs. Study Hours",
     caption = "Data source: Simulated data") +
  scale_color_manual(values = c(
    "Group C" = "blue",
    "Group B" = "purple",
    "Group A" = "orange",
    "Group S" = "red"
  ))
```

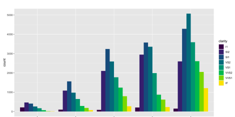

```
ggplot(student_data, aes(x = score, fill = class)) +
  geom_density(alpha = 0.3) +
  geom_histogram(
    aes(y = ..density..),
    binwidth = 5,
    alpha = 0.6,
    position = "identity"
  ) +
  labs(x = "Score", y = "Frequency", fill = "Class Group",
    title = "Distribution of Scores by Class Grouops",
    caption = "Data source: Simulated data") +
  theme_light() +
  scale_fill_manual(values = c(
    "Group C" = "red",
    "Group B" = "darkgreen",
    "Group A" = "blue",
    "Group S" = "purple")
  )
```

Coordinate Systems

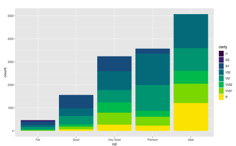
```
g <- ggplot(...)
g + coord_flip() // Switch to Cartesian coordinate system
g + coord_polar() // Switch to polar coordinate system
```

General Template

```
ggplot(data = DATA) +
  GEOM_FUNCTION(
    mapping = aes(MAPPINGS),
    stat = STAT,
    position = POSITION
  ) +
  COORDINATE_FUNCTION() +
  FACET_FUNCTION() +
  LABEL_FUNCTION()
```

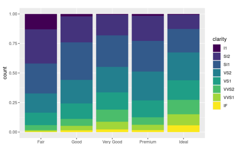


Position Arguments



- position: "identity"

- position: "dodge"



- position: "fill"

For reproducibility: `set.seed(1234)`

Sequence generation: `seq_len(length)` ≡ `seq(1, length)`

`seq_along(obj)` ≡ `seq(1, length(obj))`

Regresssion

Sum of Squared Error (**SSE**): $S(w,b) = \sum_i \epsilon^2 = \sum_i (y_i - wx_i - b)^2$

$$\frac{\partial S}{\partial w} = 2\sum_i (y_i - wx_i - b) * (-1) = 0, \qquad \overline{x} = \tfrac{1}{n}\sum_i x_i \text{ and } \overline{y} = \tfrac{1}{n}\sum_i y_i$$

$$s_{xx} = \tfrac{1}{n-1}\sum_i (x_i - \overline{x})^2$$

$$\frac{\partial S}{\partial b} = 2\sum_i (y_i - wx_i - b) * (-x_i) = 0 \qquad s_{xy} = \tfrac{1}{n-1}\sum_i (x_i - \overline{x})(y_i - \overline{y})$$

$$\hat{w} = \frac{s_{xy}}{s_{xx}} \text{ and } \hat{b} = \overline{y} - \hat{w} \cdot \overline{x}$$

R Linear Model `lm()`

Univariate

```
x <- c(0, 2, 3)
y <- c(1, 1, 4)
dataset <- data.frame(x, y)
model <- lm(y ~ x, data =
dataset)
coefficients(model)
predict(model,
data.frame(x=2))
```

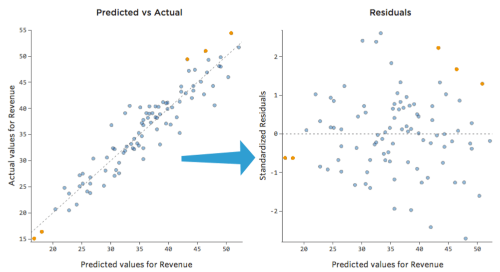
Multivariate

```
x1 <- c(1, 2, 3)
x2 <- c(4, 7, 8)
y <- c(9, 10, 11)
dataset <-
data.frame(x1,x2,y)
model <- lm(y ~ x1 + x2, data
= dataset)
predict(model,
data.frame(x1=2, x2=3))
```

- `summary()`: comprehensive summary including coefficients, standard error, **R-squared**, F-statistic, t-value, and p-value
- `attributes()`: reveal various components stored in model, such as coefficients, residuals, fitted values, and the formula used in the call
- `coefficients()`: directly extract model's coeff. (intercept & slope)
- `predict()`: make predictions based on fitted model.

Residuals

$y_i = \hat{w}x_i + \hat{b} + \epsilon_i$, where ϵ_i is the left over term (or error) between ground truth and prediction. Formally, ϵ_i is called **residuals**



- **Homoscedasticity**: ϵ_i has constant variance for all i:
  - $\mathrm{Var}(\epsilon_i | x_i) = \text{constant}^2$
- **Heteroscedasticity**: Intuitively, there is info inherent in the system not captured by the linear model
  - $\mathrm{Var}(\epsilon_i | x_i) = f(x_i)$; where $f(x_i)$ is the not captured info

Measuring Model Performance

- **Observed**: $\boldsymbol{y} = \{y_1, y_2, ..., y_n\}$
- **Predicted**: $\hat{\boldsymbol{y}} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$
- Total Sum of Squares (**TSS**): $\text{TSS} = \sum_i (y_i - \overline{y})^2$
- Explained Sum of Squares (**ESS**): $\text{ESS} = \sum_i (\hat{y}_i - \overline{y})^2$
- Residual Sum of Squares (**RSS**): $\text{RSS} = \sum_i (y_i - \hat{y}_i)^2$

TSS = ESS + RSS

- **RSS** is the "Unexplained Variability"
- **ESS** is the "Explained Variability"

$$\mathbf{R}^2 = \frac{\mathbf{ESS}}{\mathbf{TSS}} = 1 - \frac{\mathbf{RSS}}{\mathbf{TSS}}$$

Time Series Forcasting

ARIMA Model

- **Definition**: Autoregressive Integrated Moving Average
  - Integration of autoregressive (AR) and moving average (MA) models
  - **AR model** forecast: linear combination of past values of variables
  - **MA model** forecast: linear combination of past forecast errors

$$y_t = \underbrace{\delta}_{\text{Intercept}} + \underbrace{\{\phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p}\}}_{\text{AR}} + \underbrace{\{\theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}\}}_{\text{MA}} + \underbrace{\epsilon_t}_{\text{Error}}$$

$$\Longrightarrow \; y_t = \delta + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

```
install.packages('forecast')
library(forecast)
model <- auto.arima(AirPassengers)
plot(forecast(model, h=10*12))
```

More on SVD

[SVD 分解] 性质：$A^{[m\times n]} = U^{[m\times m]}\Sigma^{[m\times n]}(V^{\mathrm{T}})^{[n\times n]}$，$U^{\mathrm{T}}U = V^{\mathrm{T}}V = I$

1. 计算$(AA^{\mathrm{T}})^{[m\times m]}$，找到它的特征向量$\vec{u}$、它的特征值$\lambda$并开方 $\qquad \sigma := \sqrt{\lambda}$

2. 给σ降序排序得到序列σ^*，然后得到Σ： $\qquad \boldsymbol{\Sigma}^{[m\times n]} = \mathbf{diag}(\boldsymbol{\sigma}^*)$，**其他地方补0**

3. 对所有$\vec{u}$按照对应的σ^*排序，得到U，即： $\qquad \boldsymbol{U}^{[m\times m]} = \left[\frac{\vec{u}_1^*}{|\vec{u}_1^*|} \quad \frac{\vec{u}_2^*}{|\vec{u}_2^*|} \quad \cdots \quad \frac{\vec{u}_r^*}{|\vec{u}_r^*|}\right]$

4. 根据$A = U\Sigma V^{\mathrm{T}}$，得到$\boldsymbol{V}^{\mathrm{T}} = \boldsymbol{\Sigma}^{+}\boldsymbol{U}^{-1}\boldsymbol{A}$。$\Sigma^+$是左广义逆矩阵（$\Sigma^+\Sigma = I$）。例如，

$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \Longrightarrow \Sigma^+ = \Sigma^{-1} = \frac{1}{\Sigma} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 3 \\ 0 & 0 \end{bmatrix} \Longrightarrow \Sigma^+ = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \end{bmatrix} \text{ (pseudoinverse)}$$

去除$\sigma_k^* = 0$（甚至忽略较小的σ_k^*）对应的U、V的列，实现数据压缩，例如，

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \cdot 2 \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}^{\mathrm{T}}$$