Matrix algebra & R as a toy DSM laboratory

Distributional Semantic Models

Stefan Evert[1] & Alessandro Lenci[2]

[1]University of Osnabrück, Germany
[2]University of Pisa, Italy

ESSLLI'09 Bordeaux

The DSM data matrix

DSM data are given as a **term-term** or **term-context** matrix:

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	115	83	10	42	33	17
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27

- Most DSM parameters irrelevant for mathematical analysis (context type, terms vs. contexts, feature scaling, ...)
- Our example: targets (rows) are nouns, features (columns) are co-occurrences with verbs (V-Obj), raw counts from BNC

The DSM data matrix

DSM data are given as a **term-term** or **term-context** matrix:

$$\mathbf{M} = \begin{bmatrix} 51 & 20 & 84 & 0 & 3 & 0 \\ 52 & 58 & 4 & 4 & 6 & 26 \\ 115 & 83 & 10 & 42 & 33 & 17 \\ 59 & 39 & 23 & 4 & 0 & 0 \\ 98 & 14 & 6 & 2 & 1 & 0 \\ 12 & 17 & 3 & 2 & 9 & 27 \end{bmatrix}$$

- Mathematical notation: **matrix** $\mathbf{M}$ of real numbers
- Each row is a **feature vector** for one of the target terms, e.g.

$$\mathbf{v}_{\text{cat}} = \begin{bmatrix} 52 & 58 & 4 & 4 & 6 & 26 \end{bmatrix}$$

- n-dimensional **vector space** $\mathbb{R}^n \ni \mathbf{v} = (v_1, \ldots, v_n)$

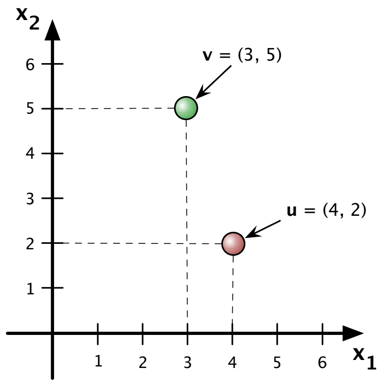
Why vector spaces?

- Vector spaces encode basic geometric intuitions
  - ☞ geometric interpretation of numerical feature lists
  - ☞ one reason why linear algebra is such a useful tool
- Interpretation of vectors $\mathbf{x}, \mathbf{y}, \ldots \in \mathbb{R}^n$ as **points** in n-dimensional Euclidean (= intuitive) space
  - ▸ $n = 2 \rightarrow$ Euclidean plane
  - ▸ $n = 3 \rightarrow$ three-dimensional Euclidean space
- Exploit geometric intuition for analysis of DSM data as group of points or arrows in Euclidean space
  - ▸ distance, length, direction, angle, dimension, ...
  - ▸ intuitive in $\mathbb{R}^2$ and $\mathbb{R}^3$
  - ▸ can be generalised to higher dimensions
  - ☞ I may refer to feature vectors for target terms as "data points"

The geometric interpretation of vectors

Vectors as points

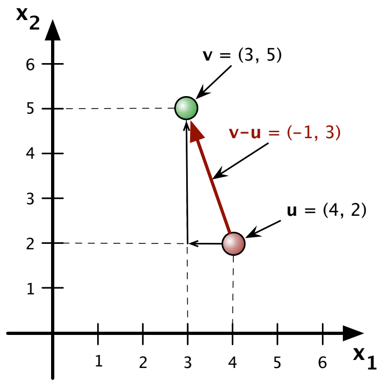
- Vectors like $\mathbf{u} = (4, 2)$ and $\mathbf{v} = (3, 5)$ can be understood as the coordinates of points in the Euclidean plane
- In this interpretation, vectors identify specific locations in the plane



The geometric interpretation of vectors

Vectors as arrows & vector addition

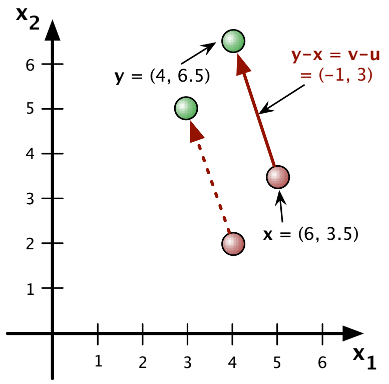
- Vectors can also be interpreted as "displacement arrows" between points
- Arrow from $\mathbf{u}$ to $\mathbf{v}$ is described by vector $(-1, 3)$
- Calculated as pointwise difference between components of $\mathbf{v}$ and $\mathbf{u}$: $\mathbf{v} - \mathbf{u} = (v_1 - u_1, v_2 - u_2)$
- General operation: **vector addition**



The geometric interpretation of vectors

Vectors as arrows

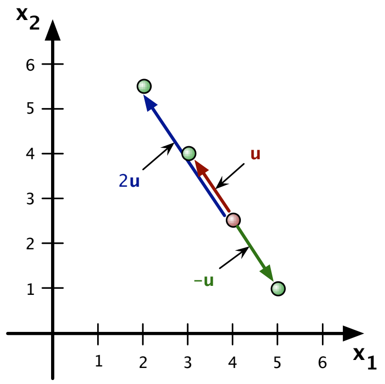
- Vectors as arrows are position-independent
- $\mathbf{y} - \mathbf{x} = \mathbf{v} - \mathbf{u}$ if the relative positions of $\mathbf{x}$ and $\mathbf{y}$ are the same as those of $\mathbf{u}$ and $\mathbf{v}$
- Regardless of their location in the plane



The geometric interpretation of vectors

Direction & scalar multiplication

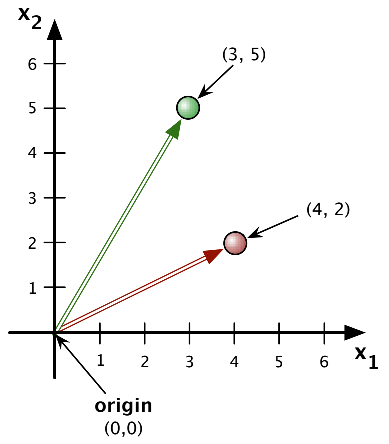
- Intuitively, arrows have a length and direction
- Arrows point in the same direction iff they are multiples of each other: **scalar multiplication** $\lambda\mathbf{u} = (\lambda u_1, \lambda u_2)$ with constant factor $\lambda \in \mathbb{R}$
- For $\lambda < 0$, arrows have opposite directions
- $-\mathbf{u} = (-1) \cdot \mathbf{u}$ is the inverse arrow of $\mathbf{u}$



The geometric interpretation of vectors

Linking points and arrows

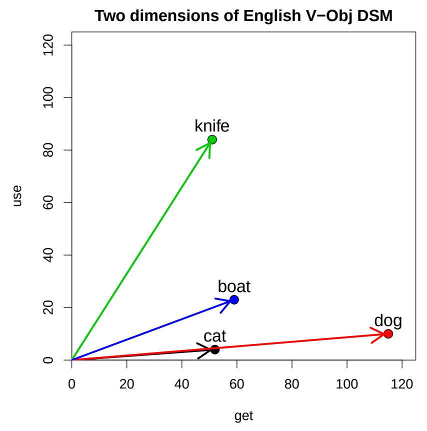
- Points in the plane can be identified by displacement arrows from fixed reference point
- A natural reference point is the **origin** $\mathbf{0} = (0,0)$
- These arrows are given by the same vectors as the point coordinates



Geometric interpretation of DSM data matrix

Reduce DSM matrix to two dimensions for visualisation:

	get	use
knife	51	84
cat	52	4
dog	115	10
boat	59	23
cup	98	6
pig	12	3



The *n*-dimensional Euclidean space

- The mathematical basis for matrix algebra is the theory of vector spaces, also known as **linear algebra**
- Before we focue on the analsis of DSM matrices, we will look at some fundamental definitions and results of linear algebra

- Definition: the *n*-dimensional **real Euclidean vector space** $\mathbb{R}^n$ is the set of all real-valued vectors $\mathbf{x} = (x_1, \dots, x_n)$ of length n, with the following operations:
  - **vector addition**: $\mathbf{u} + \mathbf{v} := (u_1 + v_1, \dots, u_n + v_n)$
  - **scalar multiplication**: $\lambda\mathbf{u} := (\lambda u_1, \dots, \lambda u_n)$ for $\lambda \in \mathbb{R}$

The *n*-dimensional Euclidean space

- Important properties of the addition and s-multiplication operations in $\mathbb{R}^n$
  1. $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$
  2. $\mathbf{u} + \mathbf{0} = \mathbf{0} + \mathbf{u} = \mathbf{u}$
  3. $\forall\mathbf{u}\ \exists(-\mathbf{u}):\ \mathbf{u} + (-\mathbf{u}) = (-\mathbf{u}) + \mathbf{u} = \mathbf{0}$
  4. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
  5. $(\lambda + \mu)\mathbf{u} = \lambda\mathbf{u} + \mu\mathbf{u}$
  6. $(\lambda\mu)\mathbf{u} = \lambda(\mu\mathbf{u})$
  7. $1 \cdot \mathbf{u} = \mathbf{u}$
  8. $\lambda(\mathbf{u} + \mathbf{v}) = \lambda\mathbf{u} + \lambda\mathbf{v}$

  for any $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and $\lambda, \mu \in \mathbb{R}$

The axioms of a general vector space

- Abstract **vector space** over the real numbers $\mathbb{R}$ $=$ set V of vectors $\mathbf{u} \in V$ with operations
  - $\mathbf{u} + \mathbf{v} \in V$ for $\mathbf{u}, \mathbf{v} \in V$ (**addition**)
  - $\lambda\mathbf{u} \in V$ for $\lambda \in \mathbb{R}$, $\mathbf{u} \in V$ (**scalar multiplication**)
- Addition and s-multiplication must satisfy the **axioms**
  1. $(\mathbf{u}+\mathbf{v})+\mathbf{w} = \mathbf{u}+(\mathbf{v}+\mathbf{w})$
  2. $\mathbf{u}+\mathbf{0} = \mathbf{0}+\mathbf{u} = \mathbf{u}$
  3. $\forall\mathbf{u}\ \exists\mathbf{u}' :\ \mathbf{u}+\mathbf{u}' = \mathbf{u}'+\mathbf{u} = \mathbf{0}$
  4. $\mathbf{u}+\mathbf{v} = \mathbf{v}+\mathbf{u}$
  5. $(\lambda+\mu)\mathbf{u} = \lambda\mathbf{u}+\mu\mathbf{u}$
  6. $(\lambda\mu)\mathbf{u} = \lambda(\mu\mathbf{u})$
  7. $1\cdot\mathbf{u} = \mathbf{u}$
  8. $\lambda(\mathbf{u}+\mathbf{v}) = \lambda\mathbf{u}+\lambda\mathbf{v}$

  for any $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and $\lambda, \mu \in \mathbb{R}$
- $\mathbf{0}$ is the unique **neutral element** of V, and the unique **inverse** $\mathbf{u}'$ of $\mathbf{u}$ is often written as $-\mathbf{u}$

Further properties of vector spaces

- Further properties of vector spaces:
  - $0\cdot\mathbf{u} = \mathbf{0}$
  - $\lambda\mathbf{0} = \mathbf{0}$
  - $\lambda\mathbf{u} = \mathbf{0} \Rightarrow \lambda = 0 \lor \mathbf{u} = \mathbf{0}$
  - $(-\lambda)\mathbf{u} = \lambda(-\mathbf{u}) = -(\lambda\mathbf{u}) =: -\lambda\mathbf{u}$
- It is easy to show these properties for $\mathbb{R}^n$, but they also follow directly from the general axioms for all vector spaces
- A non-trivial example: vector space $\mathcal{C}[a,b]$ of continuous real functions $f : x \mapsto f(x)$ over the interval $[a,b]$
  - vector addition: $\forall f, g \in \mathcal{C}[a,b]$, we define $f+g$ by $(f+g)(x) := f(x)+g(x)$
  - s-multiplication: $\forall\lambda \in \mathbb{R}$ and $\forall f \in \mathcal{C}[a,b]$, we define λf by $(\lambda f)(x) := \lambda\cdot f(x)$

☞ One can show that $\mathcal{C}[a,b]$ satisfies the vector space axioms

Linear combinations & dimensionality

- **Linear combination** of vectors $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(n)}$:

$$\lambda_1\mathbf{u}^{(1)} + \lambda_2\mathbf{u}^{(2)} + \cdots + \lambda_n\mathbf{u}^{(n)}$$

  for any coefficients $\lambda_1, \dots, \lambda_n \in \mathbb{R}$
  - intuition: all vectors that can be constructed from $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(n)}$ using the basic vector operations
- $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(n)}$ are **linearly independent** iff

$$\lambda_1\mathbf{u}^{(1)} + \lambda_2\mathbf{u}^{(2)} + \cdots + \lambda_n\mathbf{u}^{(n)} = \mathbf{0}$$

  implies $\lambda_1 = \lambda_2 = \cdots = \lambda_n = 0$
- Otherwise, they are **linearly dependent**
  - equivalent: one $\mathbf{u}^{(i)}$ is a linear combination of the other vectors

Linear combinations & dimensionality

- Largest $n \in \mathbb{N}$ for which there is a set of n linearly independent vectors $\mathbf{u}^{(i)} \in V$ is called the **dimension** of V: $\dim V = n$
- It can be shown that $\dim \mathbb{R}^n = n$
- If there is no maximal number of linearly independent vectors, the vector space is **infinite-dimensional** ($\dim V = \infty$)
- An example is $\dim \mathcal{C}[a,b] = \infty$ (easy to show)
- Every finite-dimensional vector space V is **isomorphic** to the Euclidean space $\mathbb{R}^n$ (with $n = \dim V$)

  ☞ We will be able to prove this in a little while

Basis & coordinates

- A set of vectors $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n)} \in V$ is called a **basis** of V iff every $\mathbf{u} \in V$ can be written as a linear combination
$$\mathbf{u} = x_1\mathbf{b}^{(1)} + x_2\mathbf{b}^{(2)} + \dots + x_n\mathbf{b}^{(n)}$$
with unique coefficients $x_1, \dots, x_n$
- Number of vectors in a basis = dim V
- For every n-dimensional vector space V, a set of n vectors $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n)} \in V$ is a basis iff they are linearly independent
  - ☞ Can you think of a proof?

Basis & coordinates

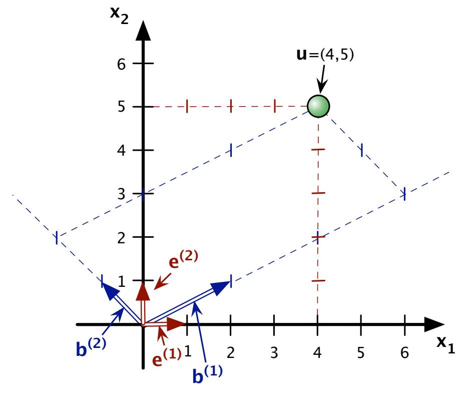
- The unique coefficients $x_1, \dots, x_n$ are called the **coordinates** of $\mathbf{u}$ wrt. the basis $B := \left(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n)}\right)$:
$$\mathbf{u} \equiv_B \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} =: \mathbf{x}$$
- $\mathbf{x} \in \mathbb{R}^n$ is the **coordinate vector** of $\mathbf{u} \in V$ wrt. B
  - ☞ V is isomorphic to $\mathbb{R}^n$ by virtue of this correspondence
- We can think of the rows (or columns) of a DSM matrix $\mathbf{M}$ as coordinates in an abstract vector space
  - ▸ coordinate transformations play an important role for DSMs

Basis & coordinates

- The components $(u_1, u_2, \dots, u_n)$ of a number vector $\mathbf{u} \in \mathbb{R}^n$ correspond to its **natural coordinates**
$$\mathbf{u} = (u_1, u_2, \dots, u_n) \equiv_E \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$
according to the **standard basis** $\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(n)}$ of $\mathbb{R}^n$:
$$\mathbf{e}^{(1)} = (1, 0, \dots, 0)$$
$$\mathbf{e}^{(2)} = (0, 1, \dots, 0)$$
$$\vdots$$
$$\mathbf{e}^{(n)} = (0, 0, \dots, 1)$$

Basis & coordinates

- $\mathbf{u} = (4, 5) \in \mathbb{R}^2$
- Basis B of $\mathbb{R}^2$:
$$\mathbf{b}^{(1)} = (2, 1)$$
$$\mathbf{b}^{(2)} = (-1, 1)$$
- $\mathbf{u} \equiv_B \begin{bmatrix} 3 \\ 2 \end{bmatrix}$
- Standard basis:
$$\mathbf{e}^{(1)} = (1, 0)$$
$$\mathbf{e}^{(2)} = (0, 1)$$
- $\mathbf{u} \equiv_E \begin{bmatrix} 4 \\ 5 \end{bmatrix}$



Linear subspaces

- The set of all linear combinations of vectors $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)} \in V$ is called the **span**

$$\text{sp}\left(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)}\right) := \left\{\lambda_1 \mathbf{b}^{(1)} + \dots + \lambda_k \mathbf{b}^{(k)} \mid \lambda_i \in \mathbb{R}\right\}$$

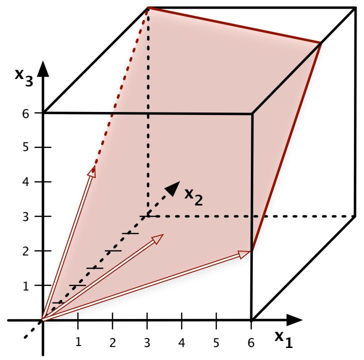
- $\text{sp}\left(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)}\right)$ forms a **linear subspace** of V
  - a linear subspace is a subset of V that is closed under vector addition and scalar multiplication
- $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)}$ form a basis of $\text{sp}\left(\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)}\right)$ iff they are linearly independent

☞ Can you prove that every linear subspace of $\mathbb{R}^n$ has a basis?

- The **rank** of vectors $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)}$ is the dimension of their span, corresponding to the largest number of linearly independent vectors among them

Linear combinations & linear subspace

- Example: linear subspace $U \subseteq \mathbb{R}^3$ spanned by vectors $\mathbf{b}^{(1)} = (6,0,2)$, $\mathbf{b}^{(2)} = (0,3,3)$ and $\mathbf{b}^{(3)} = (3,1,2)$
  - $\dim U = 2$ (because $\mathbf{b}^{(2)} = 3\mathbf{b}^{(3)} - \frac{3}{2}\mathbf{b}^{(1)}$)



Matrix as list of vectors

- Vector $\mathbf{u} \in \mathbb{R}^n$ = list of real numbers (coordinates)
- List of k vectors = rectangular array of real numbers, called a $n \times k$ **matrix** (or $k \times n$ row matrix)
- Example: vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$

$$\mathbf{u} \equiv \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix}, \quad \mathbf{v} \equiv \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

form the columns of a matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} \vdots & \vdots \\ \mathbf{u} & \mathbf{v} \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 0 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

Matrix = list of vectors

- $\text{rank}\,(\mathbf{A})$ = rank of the list of column vectors
- Column matrices are a convention in linear algebra
- But DSM matrix often has row vectors for the target terms

- Row rank and column rank of a matrix A are always the same (this is not trivial!)

Matrices and linear equation systems

- Matrices are a versatile instrument and a convenient way to express linear operations on sets of numbers
- E.g. **coefficient matrix** of a **linear system** of equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n &= b_k \end{aligned}$$

➥ $$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

Matrix algebra

- Concise notation of linear equation system by appropriate definition of **matrix-vector multiplication**

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n &= b_k \end{aligned}$$

➥ $$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

➥ $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$

Matrix algebra

- The set of all real-valued $k \times n$ matrices forms a $(k \cdot n)$-dimensional vector space over $\mathbb{R}$:
  - $\mathbf{A} + \mathbf{B}$ is defined by element-wise addition
  - $\lambda\mathbf{A}$ is defined by element-wise s-multiplication
  - these operations satisfy all vector space axioms
- Additional operation: **matrix multiplication**
  - two equation systems: $\mathbf{z} = \mathbf{B} \cdot \mathbf{y}$ and $\mathbf{y} = \mathbf{C} \cdot \mathbf{x}$
  - by inserting the expressions for $\mathbf{y}$ into the first system, we can express $\mathbf{z}$ directly in terms of $\mathbf{x}$ (and use this e.g. to solve the equations for $\mathbf{x}$)
  - the result is a linear equation system $\mathbf{z} = \mathbf{A} \cdot \mathbf{x}$
  - ☞ define matrix multiplication such that $\mathbf{A} = \mathbf{B} \cdot \mathbf{C}$

Matrix multiplication

$$\begin{bmatrix} a_{ij} & \quad \end{bmatrix} = \begin{bmatrix} b_{i1} & \cdots & b_{in} \end{bmatrix} \cdot \begin{bmatrix} c_{1j} & \quad \\ \vdots & \\ \vdots & \\ c_{nj} & \end{bmatrix}$$

$$\underset{(k \times m)}{\mathbf{A}} = \underset{(k \times n)}{\mathbf{B}} \cdot \underset{(n \times m)}{\mathbf{C}}$$

- **B** and **C** must be **conformable**

☞ $\mathbf{A} \cdot \mathbf{x}$ corresponds to matrix multiplication of **A** with a single-column matrix (containing the vector **x**)
  - convention: vector = column matrix

Matrix multiplication

- **Algebra** = vector space + multiplication operation with the following properties:
  - $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} =: \mathbf{ABC}$
  - $\mathbf{A}(\mathbf{B}+\mathbf{B}') = \mathbf{AB}+\mathbf{AB}'$
  - $(\mathbf{A}+\mathbf{A}')\mathbf{B} = \mathbf{AB}+\mathbf{A}'\mathbf{B}$
  - $(\lambda\mathbf{A})\mathbf{B} = \mathbf{A}(\lambda\mathbf{B}) = \lambda(\mathbf{AB}) =: \lambda\mathbf{AB}$
  - $\mathbf{A}\cdot\mathbf{0} = \mathbf{0}, \quad \mathbf{0}\cdot\mathbf{B} = \mathbf{0}$
  - $\mathbf{A}\cdot\mathbf{I} = \mathbf{A}, \quad \mathbf{I}\cdot\mathbf{B} = \mathbf{B}$

  where **A**, **B** and **C** are conformable matrices
- **0** is a **zero matrix** of arbitrary dimensions
- **I** is a square **identity matrix** of arbitrary dimensions:

$$\mathbf{I} := \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

Transposition

- The **transpose** $\mathbf{A}^T$ of a matrix **A** swaps rows and columns:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}^T = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$$

- Properties of the transpose:
  - $(\mathbf{A}+\mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
  - $(\lambda\mathbf{A})^T = \lambda(\mathbf{A}^T) =: \lambda\mathbf{A}^T$
  - $(\mathbf{A}\cdot\mathbf{B})^T = \mathbf{B}^T\cdot\mathbf{A}^T$ [note the different order of **A** and **B**!]
  - $\text{rank}\,(\mathbf{A}^T) = \text{rank}\,(\mathbf{A})$
  - $\mathbf{I}^T = \mathbf{I}$
- **A** is called **symmetric** iff $\mathbf{A}^T = \mathbf{A}$
  - symmetric matrices have many special properties that will become important later (e.g. eigenvalues)

Vectors and matrices

- A coordinate vector $\mathbf{x} \in \mathbb{R}^n$ can be identified with a $n \times 1$ matrix (i.e. a single-column matrix):

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^T$$

- Multiplication of a matrix **A** containing the vectors $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(k)}$ with a vector of coefficients $\lambda_1, \ldots, \lambda_k$ yields a linear combination of $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(k)}$:

$$\mathbf{A}\cdot\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_k \end{bmatrix} = \lambda_1\mathbf{a}^{(1)} + \cdots + \lambda_k\mathbf{a}^{(k)}$$

R as a toy DSM laboratory

- Matrix algebra is a powerful and convenient tool in numerical mathematics ➜ implement DSM with matrix operations
- Specialised (and highly optimised) libraries are available for various programming languages (C, C++, Perl, Python, . . .)
- Some numerical programming environments are even based entirely on matrix algebra (Matlab, Octave, NumPy/Sage)
- Statistical software packages like **R** also support matrices

- **R** as a **DSM laboratory** for toy models
  `http://www.r-project.org/`
- Integrates efficient matrix operations with statistical analysis, clustering, machine learning, visualisation, . . .

Matrix algebra with R

Vectors in R:

- `u1 <- c(3, 0, 2)`
- `u2 <- c(0, 2, 2)`
- `v <- 1:6`
- `print(v)`

```
[1] 1 2 3 4 5 6
```

Defining matrices:

- `A <- matrix(v, nrow=3)`
- `print(A)`

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Matrix algebra in R

Matrix of column vectors:

- `B <- cbind(u1, u2)`
- `print(B)`

```
     u1 u2
[1,]  3  0
[2,]  0  2
[3,]  2  2
```

Matrix of row vectors:

- `C <- rbind(u1, u2)`
- `print(C)`

```
   [,1] [,2] [,3]
u1    3    0    2
u2    0    2    2
```

Matrix algebra in R

Matrix multiplication:

- `A %*% C`

```
     [,1] [,2] [,3]
[1,]    3    8   10
[2,]    6   10   14
[3,]    9   12   18
```

- NB: `*` does *not* perform matrix multiplication

Also for multiplication of matrix with vector:

- `C %*% c(1,1,0)`

```
   [,1]
u1    3
u2    2
```

☞ result of multiplication is a column vector (i.e. plain vectors are interpreted as column vectors in matrix operations)

Matrix algebra in R

Transpose of matrix:

- `t(A)`

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Transposition of vectors:

- `t(u1)` (row vector)

```
     [,1] [,2] [,3]
[1,]    3    0    2
```

- `t(t(u1))` (explicit column vector)

```
     [,1]
[1,]    3
[2,]    0
[3,]    2
```

Matrix algebra in R

Rank of a matrix:

- `qr(A)$rank`
  ```
  2
  ```
- `la.rank <- function (A) qr(A)$rank`
- `la.rank(A)`

Column rank = row rank:

- `la.rank(A) == la.rank(t(A))`
  ```
  [1] TRUE
  ```

$\mathbf{A}^T \cdot \mathbf{A}$ is symmetric (can you prove this?):

- `t(A) %*% A`

Linear maps

- A **linear map** is a **homomorphism** between two vector spaces V and W, i.e. a function $f : V \to W$ that is compatible with addition and s-multiplication:
  1. $f(\mathbf{u} + \mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$
  2. $f(\lambda \mathbf{u}) = \lambda \cdot f(\mathbf{u})$
- Obviously, f is uniquely determined by the images $f\big(\mathbf{b}^{(1)}\big), \dots, f\big(\mathbf{b}^{(n)}\big)$ of any basis $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n)}$ of V
- Using natural coordinates, a linear map $f : \mathbb{R}^n \to \mathbb{R}^k$ can therefore be described by the vectors

$$f\big(\mathbf{e}^{(1)}\big) \equiv_E \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{k1} \end{bmatrix}, \ \dots\ , \ f\big(\mathbf{e}^{(n)}\big) \equiv_E \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{kn} \end{bmatrix}$$

Matrix representation of a linear map

- For a vector $\mathbf{u} = x_1\mathbf{e}^{(1)} + \dots + x_n\mathbf{e}^{(n)} \in \mathbb{R}^n$, we have

$$\begin{aligned} \mathbf{v} = f(\mathbf{u}) &= f\big(x_1\mathbf{e}^{(1)} + \dots + x_n\mathbf{e}^{(n)}\big) \\ &= x_1 \cdot f\big(\mathbf{e}^{(1)}\big) + \dots + x_n \cdot f\big(\mathbf{e}^{(n)}\big) \end{aligned}$$

  and hence the natural coordinate vector $\mathbf{y}$ of $\mathbf{v}$ is given by

$$y_j = x_1 \cdot a_{j1} + x_2 \cdot a_{j2} + \dots + x_n \cdot a_{jn}$$

- This corresponds to matrix multiplication

$$\begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

➥ $\mathbf{v} = f(\mathbf{u}) \iff \mathbf{y} = \mathbf{A} \cdot \mathbf{x}$

Image & kernel

- The **image** of a linear map $f : \mathbb{R}^n \to \mathbb{R}^k$ is the subspace of all values $\mathbf{v} \in \mathbb{R}^k$ that $f(\mathbf{u})$ can assume for $\mathbf{u} \in \mathbb{R}^n$:

$$\operatorname{Im}(f) := \operatorname{sp}\Big(f\big(\mathbf{e}^{(1)}\big), \dots, f\big(\mathbf{e}^{(n)}\big)\Big)$$

- The **rank** of f is defined by $\operatorname{rank}(f) := \dim\big(\operatorname{Im}(f)\big)$
- $\operatorname{rank}(f) = \operatorname{rank}(\mathbf{A})$ for the matrix representation $\mathbf{A}$
- f is **surjective** (onto) iff $\operatorname{Im}(f) = \mathbb{R}^k$, i.e. $\operatorname{rank}(f) = k$
- The **kernel** of f is the subspace of all $\mathbf{x} \in \mathbb{R}^n$ that are mapped to $\mathbf{0} \in \mathbb{R}^k$:

$$\operatorname{Ker}(f) := \{\mathbf{x} \in \mathbb{R}^n \,|\, f(\mathbf{x}) = \mathbf{0}\}$$

Rank & composition

- We have $\dim\big(\mathrm{Im}\,(f)\big) + \dim\big(\mathrm{Ker}\,(f)\big) = n$
- f is **injective** iff every $\mathbf{v} \in \mathrm{Im}\,(f)$ has a unique preimage $\mathbf{v} = f(\mathbf{u})$, i.e. iff $\mathrm{Ker}\,(f) = \{\mathbf{0}\}$ or $\mathrm{rank}\,(f) = n$
- The **composition** of linear maps corresponds to matrix multiplication:
  - $f : \mathbb{R}^n \to \mathbb{R}^k$ given by a $k \times n$ matrix $\mathbf{A}$
  - $g : \mathbb{R}^k \to \mathbb{R}^m$ given by a $m \times k$ matrix $\mathbf{B}$
  - recall that $(g \circ f)(\mathbf{u}) := g(f(\mathbf{u}))$
  - ➥ the composition $g \circ f : \mathbb{R}^n \to \mathbb{R}^m$ is given by the matrix product $\mathbf{B} \cdot \mathbf{A}$

The inverse matrix

- A linear map $f : \mathbb{R}^n \to \mathbb{R}^n$ is called an **endomorphism**
  - can be represented by a square matrix $\mathbf{A}$
- f surjective $\iff$ $\mathrm{rank}\,(f) = n$ $\iff$ f injective
- $\mathrm{rank}\,(f) = \mathrm{rank}\,\big(f\big(\mathbf{e}^{(1)}\big), \dots, f\big(\mathbf{e}^{(n)}\big)\big) = n$ $\iff \mathrm{rank}\,(\mathbf{A}) = n \iff \det \mathbf{A} \neq 0$
- ➥ f **bijective** (one-to-one) $\iff$ $\det \mathbf{A} \neq 0$
- If f is bijective, there exists an inverse function $f^{-1} : \mathbb{R}^n \to \mathbb{R}^n$, which is also a linear map and satisfies $f^{-1}(f(\mathbf{u})) = \mathbf{u}$ and $f(f^{-1}(\mathbf{v})) = \mathbf{v}$
- f^{-1} is given by the **inverse matrix** $\mathbf{A}^{-1}$ of $\mathbf{A}$, which must satisfy $\mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I}$

Linear equation systems

- Recall that a linear system of equations can be written in compact matrix notation:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

- Obviously, $\mathbf{A}$ describes a linear map $f : \mathbb{R}^n \to \mathbb{R}^k$, and the linear system of equations can be written $f(\mathbf{x}) = \mathbf{b}$
- This linear system can be solved iff $\mathbf{b} \in \mathrm{Im}\,(f)$, i.e. iff $\mathbf{b}$ is a linear combination of the column vectors of $\mathbf{A}$
- The solution is given by the coefficients $x_1, \dots, x_n$ of this linear combination

Linear equation systems

- The linear system has a solution for arbitrary $\mathbf{b} \in \mathbb{R}^k$ iff f is surjective, i.e. iff $\mathrm{rank}\,(\mathbf{A}) = k$
- Solutions of the linear system are unique iff f is injective, i.e. iff $\mathrm{rank}\,(\mathbf{A}) = n$ (the column vectors are linearly independent)
- If $k = n$ (i.e. $\mathbf{A}$ is a square matrix), the linear map f is an endomorphism. Consequently, the linear system has a unique solution for arbitrary $\mathbf{b}$ iff $\det \mathbf{A} \neq 0$
- In this case, the solution can be computed with the inverse function f^{-1} or the inverse matrix $\mathbf{A}^{-1}$:

$$\mathbf{x} = f^{-1}(\mathbf{b}) = \mathbf{A}^{-1} \cdot \mathbf{b}$$

☞ practically, $\mathbf{A}^{-1}$ is often determined by solving the corresponding linear system of equations

Linear equation systems

Solving equation systems in R:

- `A <- rbind(c(1,3), c(2,-1))`
- `b <- c(5,3)`
- `la.rank(A)` (test that $\mathbf{A}$ is invertible)
- `A.inv <- solve(A)` (inverse matrix $\mathbf{A}^{-1}$)
- `print(round(A.inv, digits=3))`

```
      [,1]   [,2]
[1,] 0.143  0.429
[2,] 0.286 -0.143
```

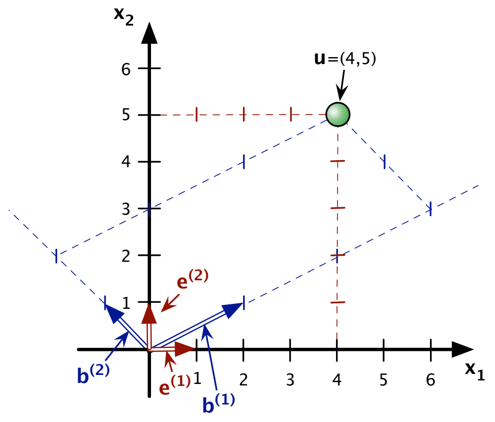
- `A.inv %*% b`

```
     [,1]
[1,]    2
[2,]    1
```

- `solve(A, b)` (recommended: calculate $\mathbf{A}^{-1} \cdot \mathbf{b}$ directly)

Coordinate transformations

- We want to **transform** between coordinates with respect to a basis $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n)}$ and standard coordinates in $\mathbb{R}^n$



Coordinate transformations

- The basis can be represented by a matrix $\mathbf{B}$ whose columns are the standard coordinates of $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(n)}$
- Given a vector $\mathbf{u} \in \mathbb{R}^n$ with standard coordinates $\mathbf{u} \equiv_E \mathbf{x}$ and $\mathbf{B}$-coordinates $\mathbf{u} \equiv_B \mathbf{y}$, we have

$$\mathbf{u} = y_1 \mathbf{b}^{(1)} + \dots + y_n \mathbf{b}^{(n)}$$

- In standard coordinates, this equation corresponds to matrix multiplication:

$$\mathbf{x} = \mathbf{B} \cdot \mathbf{y}$$

➥ Matrix $\mathbf{B}$ transforms B-coordinates into standard coordinates

Coordinate transformations

- To transform from standard coordinates into B-coordinates, i.e. from $\mathbf{x}$ to $\mathbf{y}$, we must solve the linear system $\mathbf{x} = \mathbf{B}\mathbf{y}$
- Since the $\mathbf{b}^{(i)}$ are linearly independent, $\mathbf{B}$ is regular and the inverse $\mathbf{B}^{-1}$ exists, so that

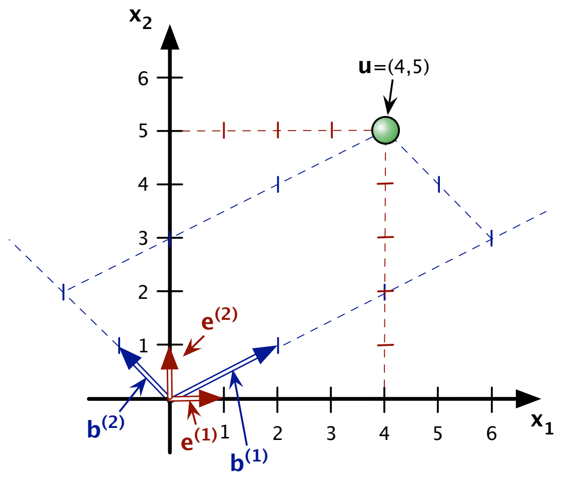
$$\mathbf{y} = \mathbf{B}^{-1}\mathbf{x}$$

➥ The inverse matrix $\mathbf{B}^{-1}$ transforms from standard coordinates into B-coordinates

- Recall that $\mathbf{B}\mathbf{B}^{-1} = \mathbf{B}^{-1}\mathbf{B} = \mathbf{I}$ (transform back & forth)
- Transformation from B-coordinates ($\mathbf{u} \equiv_B \mathbf{y}$) into arbitrary C-coordinates ($\mathbf{u} \equiv_C \mathbf{z}$):

$$\mathbf{z} = \mathbf{C}^{-1}\mathbf{B}\mathbf{y}$$

Coordinate transformations: an example



Coordinate transformations: an example

- Basis $\mathbf{b}^{(1)} = (2,1)$, $\mathbf{b}^{(2)} = (-1,1)$ with matrix representation

$$\mathbf{B} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{B}^{-1} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

- Vector $\mathbf{u} = (4,5)$ with standard and $\mathbf{B}$-coordinates

$$\mathbf{u} \equiv_E \begin{bmatrix} 4 \\ 5 \end{bmatrix}, \quad \mathbf{u} \equiv_C \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

- Check that these equalities hold:

$$\begin{bmatrix} 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

- Now perform the calculations in R!

Playtime: toy DSM laboratory

- Goal: construct and analyse DSM entirely in **R**
- We will build the small noun-verb matrix from the introduction
- Data: verb-object co-occurrence tokens from British National Corpus (extracted with regexp query, both words lemmatised)

- Text table with 3,406,821 co-occurence tokens in file `bnc_vobj_filtered.txt.gz`

acquire	deficiency
affect	body
fight	infection
face	condition
serve	interest
put	back

Preliminaries

```
# This is a comment: do not type comment lines into R!
# You should be able to execute most commands by copy & paste
> (1:10)^2
[1]   1   4   9  16  25  36  49  64  81 100

# The > indicates the R command prompt; it is not part of the input!
# Output of an R command is shown in blue below the command

# Long commands may require continuation lines starting with +;
# you should enter such commands on a single line, if possible
> c(1,
+   2,
+   3)
[1] 1 2 3
```

Reading the co-occurrence tokens

```
# Load tabular data with read.table(); options save memory and ensure
# that strings are loaded correctly; gzfile() decompresses on the fly
> tokens <- read.table(gzfile("bnc_vobj_filtered.txt.gz"),
+                      colClasses="character", quote="",
+                      col.names=c("verb", "noun"))

# You must first ‘‘change working directory’’ to where you have saved the file;
# if you can’t, then replace filename by file.choose() above

# If you have problems with the compressed file, then decompress the disk file
# (some Web browsers may do this automatically) and load with
> tokens <- read.table("bnc_vobj_filtered.txt",
+                      colClasses="character", quote="",
+                      col.names=c("verb", "noun"))
```

Reading the co-occurrence tokens

```
# The variable tokens now holds co-occurrence tokens as a table
# (in R lingo, such tables are called data.frames)

# Size of the table (rows, columns) and first 6 rows
> dim(tokens)
[1] 3406821       2

> head(tokens, 6)
     verb       noun
1 acquire deficiency
2  affect       body
3   fight  infection
4    face  condition
5   serve   interest
6     put       back
```

Filtering selected verbs & nouns

```
# Example matrix for selected nouns and verbs
> selected.nouns <- c("knife","cat","dog","boat","cup","pig")
> selected.verbs <- c("get","see","use","hear","eat","kill")

# %in% operator tests whether value is contained in list;
# note the single & for logical ‘‘and’’ (vector operation)
> tokens <- subset(tokens, verb %in% selected.verbs &
+                          noun %in% selected.nouns)

# How many co-occurrence tokens are left?
> dim(tokens)
[1] 924   2
> head(tokens, 5)
      verb  noun
2813   get knife
6021   see   pig
6489   see   cat
24130  see   cat
26620  see  boat
```

Co-occurrence counts

```
# Contstruct matrix of co-occurrence counts (contingency table)
> M <- table(tokens$noun, tokens$verb)
> M
        eat get hear kill see use
  boat    0  59    4    0  39  23
  cat     6  52    4   26  58   4
  cup     1  98    2    0  14   6
  dog    33 115   42   17  83  10
  knife   3  51    0    0  20  84
  pig     9  12    2   27  17   3

# Use subscripts to extract row and column vectors
> M["cat", ]
 eat  get hear kill  see  use
   6   52    4   26   58    4
> M[, "use"]
 boat   cat   cup   dog knife   pig
   23     4     6    10    84     3
```

Marginal frequencies

```
# For the calculating association scores, we need the marginal frequencies
# of the nouns and verbs; for simplicity, we obtain them by summing over the
# rows and columns of the table (this is not mathematically correct!)
> f.nouns <- rowSums(M)
> f.verbs <- colSums(M)
> N <- sum(M)   # sample size (sum over all cells of the table)

> f.nouns
 boat   cat   cup   dog knife   pig
  125   150   121   300   158    70
> f.verbs
 eat  get hear kill  see  use
  52  387   54   70  231  130
> N
[1] 924
```

Expected and observed frequencies

Expected frequencies: $E_{ij} = \dfrac{f_i^{(\text{noun})} \cdot f_j^{(\text{verb})}}{N}$

can be calculated efficiently with **outer product** $\mathbf{f}^{(n)} \cdot (\mathbf{f}^{(v)})^T$:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 \\ x_2y_1 & x_2y_2 & x_2y_3 \end{bmatrix}$$

```
> E <- f.nouns %*% t(f.verbs) / N
> round(E, 1)
     eat  get hear kill  see  use
[1,] 7.0 52.4  7.3  9.5 31.2 17.6
[2,] 8.4 62.8  8.8 11.4 37.5 21.1
[3,] 6.8 50.7  7.1  9.2 30.2 17.0
...
# Observed frequencies are simply the entries of M
> O <- M
```

Feature scaling: log frequencies

```
# Because of Zipf's law, frequency distributions are highly skewed;
# DSM matrix M will be dominated by high-frequency entries

# Solution 1: transform into logarithmic frequencies
> M1 <- log10(M + 1)   # discounted (+1) to avoid log(0)
> round(M1, 2)
       eat  get hear kill  see  use
 boat  0.00 1.78 0.70 0.00 1.60 1.38
 cat   0.85 1.72 0.70 1.43 1.77 0.70
 cup   0.30 2.00 0.48 0.00 1.18 0.85
 dog   1.53 2.06 1.63 1.26 1.92 1.04
 knife 0.60 1.72 0.00 0.00 1.32 1.93
 pig   1.00 1.11 0.48 1.45 1.26 0.60
```

Feature scaling: association measures

Simple association measures can be expressed in terms of observed (O) and expected (E) frequencies, e.g. **t-score**:

$$t = \frac{O - E}{\sqrt{O}}$$

You can implement any of the equations in (Evert 2008)

```
> M2 <- (O - E) / sqrt(O + 1)   # discounted to avoid division by 0
> round(M2, 2)
         eat    get   hear   kill    see    use
 boat  -7.03   0.86  -1.48  -9.47   1.23   1.11
 cat   -0.92  -1.49  -2.13   2.82   2.67  -7.65
 cup   -4.11   4.76  -2.93  -9.17  -4.20  -4.17
 dog    2.76  -0.99   3.73  -1.35   0.87  -9.71
 knife -2.95  -2.10  -9.23 -11.97  -4.26   6.70
 pig    1.60  -4.80  -1.21   4.10  -0.12  -3.42
```

Feature scaling: sparse association measures

```
# ''Sparse'' association measures set all negative associations to 0;
# this can be done with ifelse(), a vectorised if statement
> M3 <- ifelse(O >= E, (O - E) / sqrt(O), 0)
> round(M3, 2)
       eat  get hear kill  see  use
boat  0.00 0.87 0.00 0.00 1.24 1.13
cat   0.00 0.00 0.00 2.87 2.69 0.00
cup   0.00 4.78 0.00 0.00 0.00 0.00
dog   2.81 0.00 3.78 0.00 0.88 0.00
knife 0.00 0.00 0.00 0.00 0.00 6.74
pig   1.69 0.00 0.00 4.18 0.00 0.00

# Pick your favourite scaling method here!
> M <- M2
```

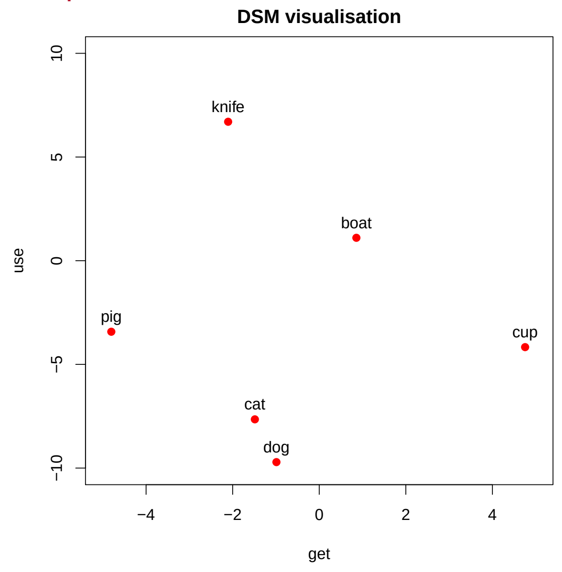
Visualisation: plot two selected dimensions

```
> M.2d <- M[, c("get", "use")]
> round(M.2d, 2)
        get   use
boat   0.86  1.11
cat   -1.49 -7.65
cup    4.76 -4.17
dog   -0.99 -9.71
knife -2.10  6.70
pig   -4.80 -3.42

# Two-column matrix automatically interpreted as x- and y-coordinates
> plot(M.2d, pch=20, col="red", main="DSM visualisation")

# Add labels: the text strings are the rownames of M
> text(M.2d, labels=rownames(M.2d), pos=3)
```

Visualisation: plot two selected dimensions



Norm & distance

Intuitive length of vector **x**: **Euclidean norm**

$$\mathbf{x} \mapsto \|\mathbf{x}\|_2 = \sqrt{(x_1)^2 + (x_2)^2 + \cdots + (x_n)^2}$$

Euclidean distance **metric**: $d_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$

☞ more about norms and distances on Thursday

```
# R function definitions look almost like mathematical definitions

euclid.norm <- function (x) sqrt(sum(x * x))

euclid.dist <- function (x, y) euclid.norm(x - y)
```

Normalisation to unit length

```
# Compute lengths (norms) of all row vectors
> row.norms <- apply(M, 1, euclid.norm)  # 1 = rows, 2 = columns
> round(row.norms, 2)
 boat   cat   cup   dog knife   pig
12.03  9.01 12.93 10.93 17.45  7.46

# Normalisation: divide each row by its norm; this a rescaling of the row
# ‘‘dimensions’’ and can be done by multiplication with a diagonal matrix
> scaling.matrix <- diag(1 / row.norms)
> round(scaling.matrix, 3)

> M.norm <- scaling.matrix %*% M
> round(M.norm, 2)
        eat   get  hear  kill   see   use
  [1,] -0.58  0.07 -0.12 -0.79  0.10  0.09
  [2,] -0.10 -0.17 -0.24  0.31  0.30 -0.85
  [3,] -0.32  0.37 -0.23 -0.71 -0.32 -0.32
  ...
```

Distances between row vectors

```
# Matrix multiplication has lost the row labels (copy from M)
> rownames(M.norm) <- rownames(M)

# To calculate distances of all terms e.g. from "dog", apply euclid.dist()
# function to rows, supplying the "dog" vector as fixed second argument
> v.dog <- M.norm["dog",]
> dist.dog <- apply(M.norm, 1, euclid.dist, y=v.dog)

# Now we can sort the vector of distances to find nearest neighbours
> sort(dist.dog)
     dog      cat      pig      cup     boat    knife
0.000000 0.839380 1.099067 1.298376 1.531342 1.725269
```

The distance matrix

```
# R has a built-in function to compute a full distance matrix
> distances <- dist(M.norm, method="euclidean")
> round(distances, 2)
      boat  cat  cup  dog knife
cat   1.56
cup   0.73 1.43
dog   1.53 0.84 1.30
knife 0.77 1.70 0.93 1.73
pig   1.80 0.80 1.74 1.10  1.69

# If you want to search nearest neighbours, convert triangular distance
# matrix to full symmetric matrix and extract distance vectors from rows
> dist.matrix <- as.matrix(distances)
> sort(dist.matrix["dog",])
     dog      cat      pig      cup     boat    knife
0.000000 0.839380 1.099067 1.298376 1.531342 1.725269
```

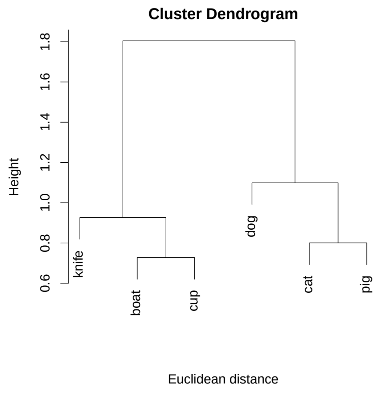
Clustering and semantic maps

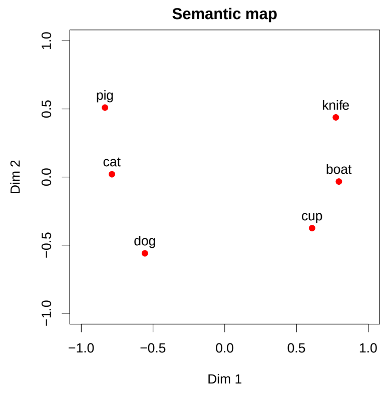
```
# Distance matrix is also the basis for a cluster analysis
> plot(hclust(distances))

# Visualisation as semantic map by projection into 2-dimensional space;
# uses non-linear multidimensional scaling (MDS)
> library(MASS)
> M.mds <- isoMDS(distances)$points
initial  value 2.611213
final  value 0.000000
converged

# Plot works in the same way as for the two selected dimensions above
> plot(M.mds, pch=20, col="red", main="Semantic map",
+             xlab="Dim 1", ylab="Dim 2")
> text(M.mds, labels=rownames(M.mds), pos=3)
```

Clustering and semantic maps


Cluster Dendrogram
Height
knife
boat
cup
dog
cat
pig
Euclidean distance


Semantic map
Dim 2
pig
cat
dog
knife
boat
cup
Dim 1