Reminder

Multiple Linear Regression

(Chapters 12-13 in Montgomery, Runger)

12-1: Multiple Linear Regression Model

12-1.1 Introduction

- Many applications of regression analysis involve situations in which there are more than one regressor variable X_k used to predict Y.

- A regression model then is called a **multiple regression model**.

Multiple Linear Regression Model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots \beta_k x_k + \varepsilon$$

One can also use powers and products of other variables or even non-linear functions like $\exp(x_i)$ or $\log(x_i)$ instead of $x_3, \ldots x_k$.

Example: the general two-variable quadratic regression has 6 constants:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1)^2 + \beta_4 (x_2)^2 + \beta_5 (x_1 x_2) + \varepsilon$$

12-1: Multiple Linear Regression Model

12-1.3 Matrix Approach to Multiple Linear Regression

Suppose the model relating the regressors to the response is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \varepsilon_i \qquad i = 1, 2, \ldots, n$$

In matrix notation this model can be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad (12\text{-}6)$$

12-1: Multiple Linear Regression Model

12-1.3 Matrix Approach to Multiple Linear Regression

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} \text{ and } \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

12-1.3 Matrix Approach to Multiple Linear Regression

We wish to find the vector $\hat{\beta}$ that minimizes the sum of squares of error terms:

$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon'\varepsilon = (y - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)$$

$$0 = \frac{\partial L}{2\partial \beta} = -X'(\mathbf{y} - \mathbf{X}\beta) = -X'y + (X'X)\boldsymbol{\beta}$$

The resulting least squares estimate is

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \qquad (12\text{-}7)$$

Analog of $\frac{1}{Var(X)}$ Analog of $Cov(X,Y)$

Multiple Linear Regression Model

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\,\mathbf{X}'\mathbf{y}$$

H is an idempotent matrix

$$\hat{y} = X\hat{\beta} = X(X'X)^{-1}X'y,$$

$$\hat{y} = Hy, \quad \text{and} \quad e = (I - H)y.$$

$H = H'$; $H^2 = X(X'X)^{-1}X'X(X'X)^{-1}X' = X(X'X)^{-1}X' = H$

I

Vectors $\hat{y}$ & e are orthogonal since

$\hat{y}' \cdot e = y'H(I-H)y = 0$ since

$$H(I - H) = H - H^2 = H - H = 0.$$

12-1: Multiple Linear Regression Models

12-1.4 Properties of the Least Squares Estimators

Unbiased estimators:

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}) &= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}] \\ &= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})] \\ &= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\boldsymbol{\beta} + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\epsilon}] \\ &= \boldsymbol{\beta} \end{aligned}$$

Covariance Matrix of Estimators:

$$\mathbf{C} = (\mathbf{X}'\mathbf{X})^{-1} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

12-1: Multiple Linear Regression Models

12-1.4 Properties of the Least Squares Estimators

Individual variances and covariances:

$$V(\hat{\beta}_j) = \sigma^2 C_{jj}, \qquad j = 0, 1, 2$$

$$\text{cov}(\hat{\beta}_i, \hat{\beta}_j) = \sigma^2 C_{ij}, \qquad i \neq j$$

In general,

$$\text{cov}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1} = \sigma^2 \mathbf{C}$$

12-1: Multiple Linear Regression Models

Estimating error variance σ_ε^2

An unbiased estimator of error variance σ_ε^2 is

$$\hat{\sigma}_\varepsilon^2 = \frac{\sum_{i=1}^{n} e_i^2}{n-p} = \frac{SS_E}{n-p} \qquad (12\text{-}16)$$

Here p=k+1 for k-variable multiple linear regression

R^2 and Adjusted R^2

The **coefficient of multiple determination R^2**

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T}$$

The **adjusted R^2** is

$$R^2_{\text{adj}} = 1 - \frac{SS_E/(n-p)}{SS_T/(n-1)} \approx 1 - \frac{\hat{\sigma}_\varepsilon^2}{\sigma_y^2} \qquad (12\text{-}23)$$

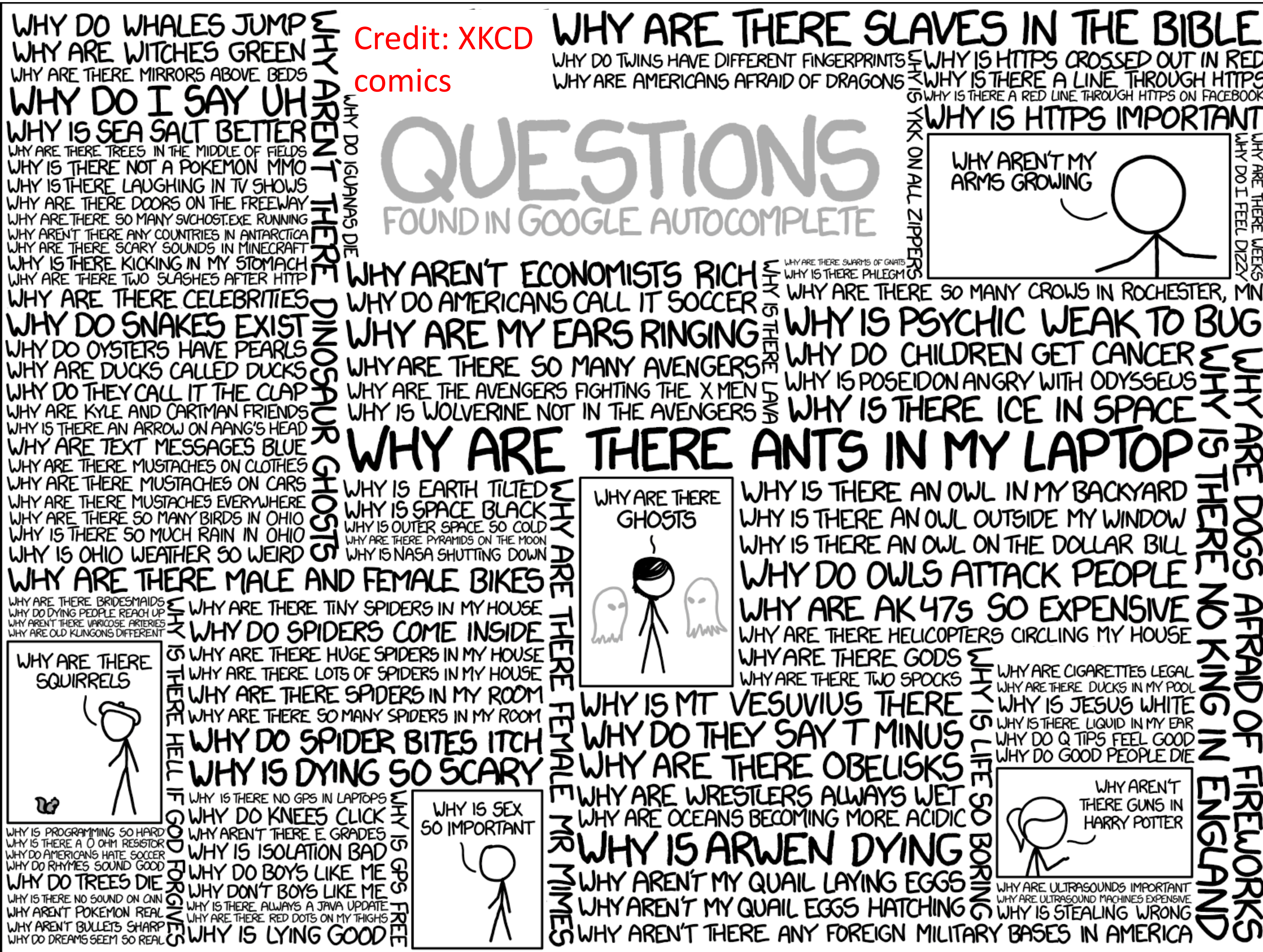
- The adjusted R^2 statistic penalizes adding terms to the MLR model.
- It can help guard against overfitting (including regressors that are not really useful)

How to know where to stop adding variables?

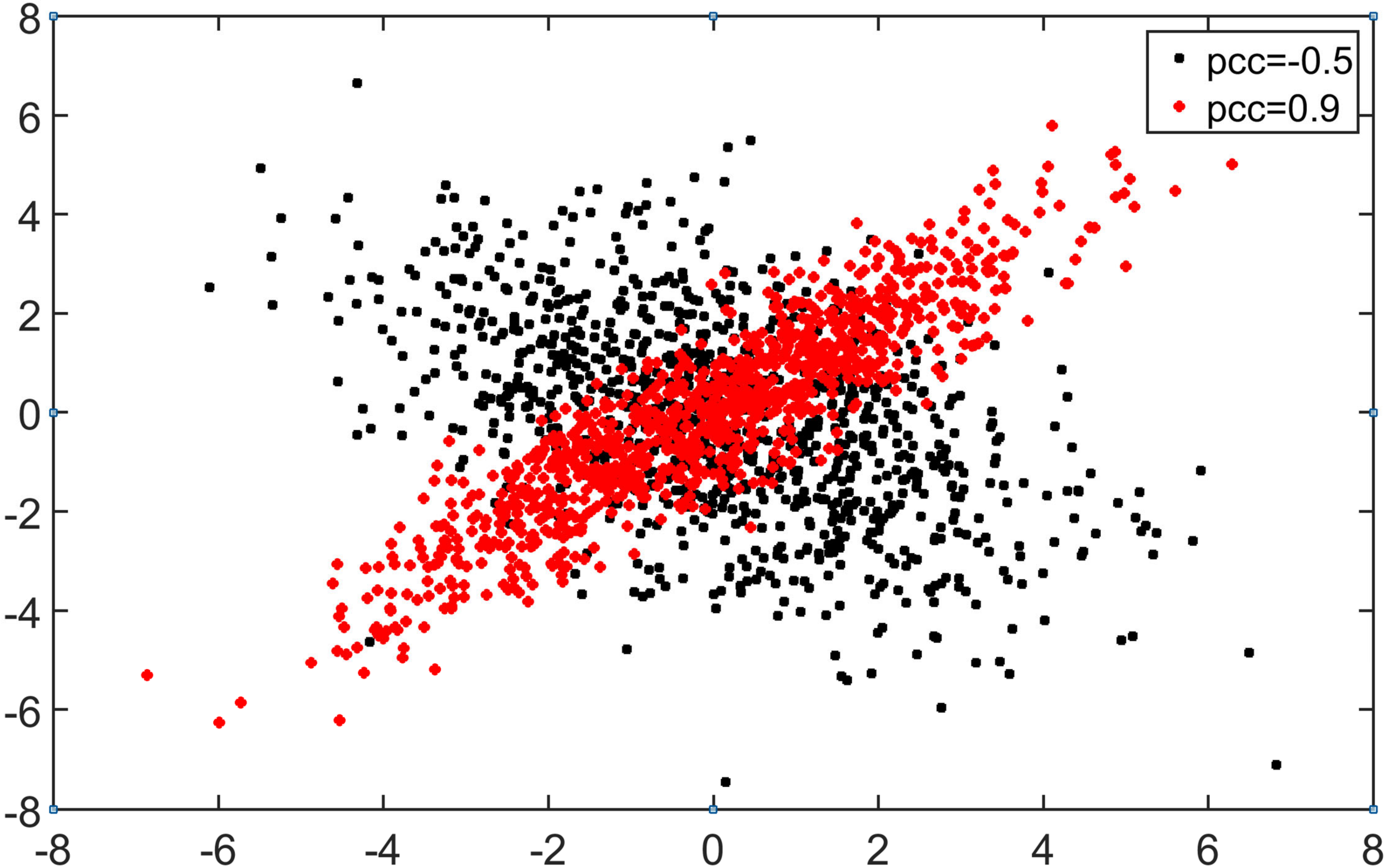
- Adding new variables x_i to MLR
  watch the adjusted R^2

- Once the adjusted R^2
  no longer increases = stop.
  Now you did the best you can.

Credit: XKCD comics

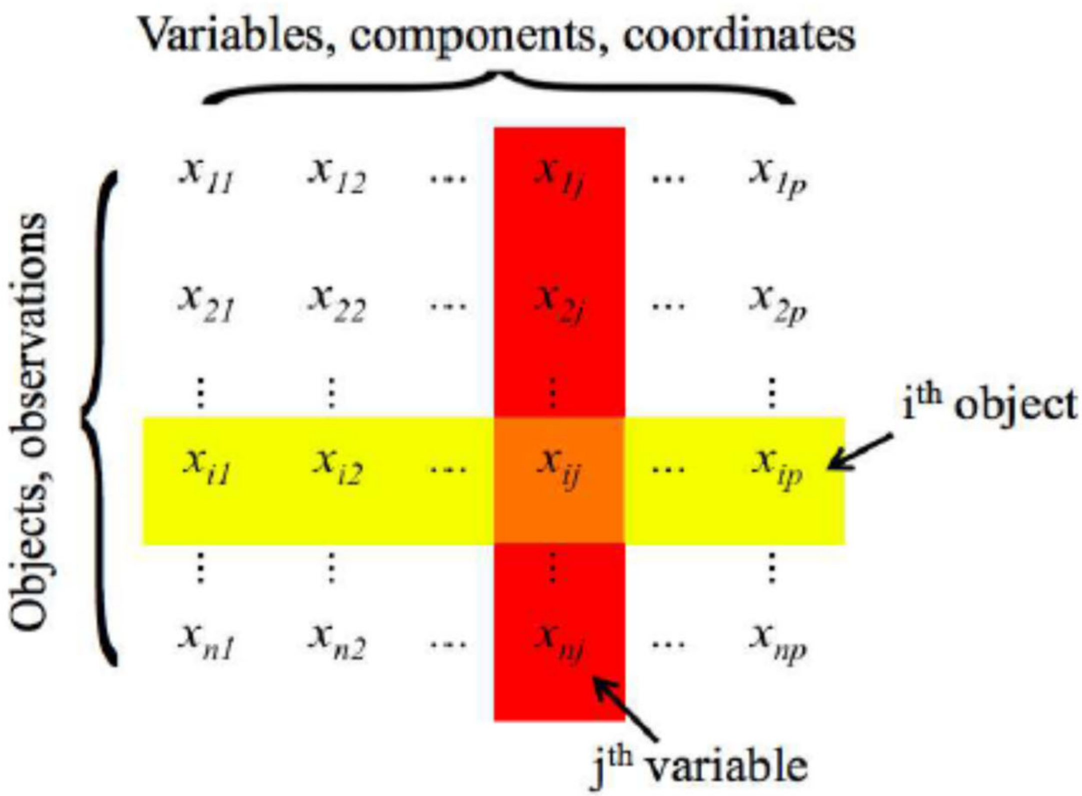


Principal Component Analysis


pcc=-0.5
pcc=0.9
8
6
4
2
0
-2
-4
-6
-8
-8
-6
-4
-2
0
2
4
6
8

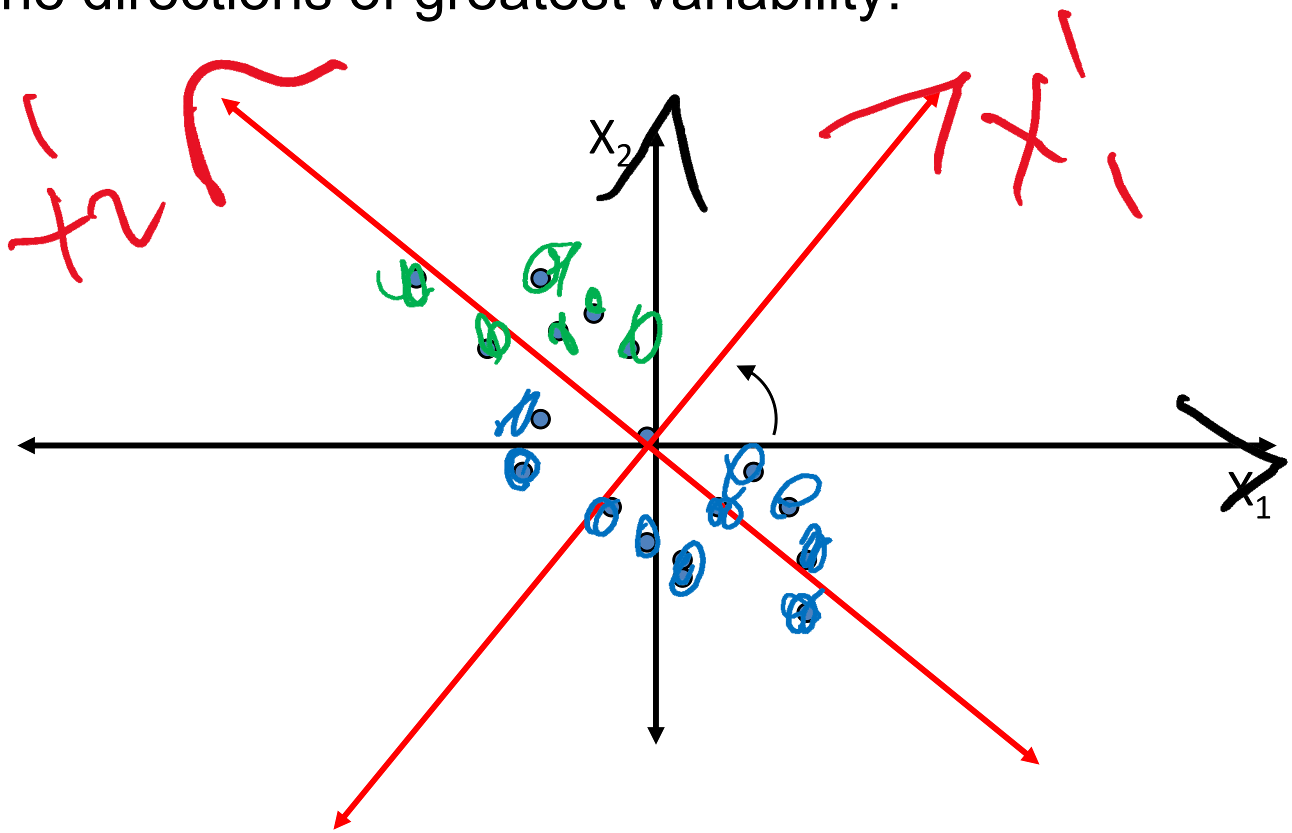
Multivariable statistics and **Principal Component Analysis (PCA)**

- A table of n observations in which p variables were measured



Trick: Rotate Coordinate Axes

Suppose we have a population measured on p random variables $X_1, \ldots, X_p$. Note that these random variables represent the p-axes of the Cartesian coordinate system in which the population resides. Our goal is to develop a new set of p axes (linear combinations of the original p axes) in the directions of greatest variability:



This is accomplished by rotating the axes.

Adapted from slides by Prof. S. Narasimhan, "Computer Vision" course at CMU

Applications of PCA

- Uses:
  - Data Visualization
  - Dimensional Reduction
  - Data Classification

- Examples:
  - How many unique "sub-sets" are in the sample?
  - How are they similar / different?
  - What are the underlying factors that most influence the samples?
  - Which measurements are best to differentiate between samples?
  - How to best present what is "interesting"?
  - Which "sub-set" does this new sample rightfully belong?

Adapted from slides by Prof. S. Narasimhan, "Computer Vision" course at CMU

PCA: *General*

From p original variables: $x_1, x_2, \ldots, x_p$:

Produce k new variables: $x'_1, x'_2, \ldots, x'_p$:

$$x'_1 = v_{11}x_1 + v_{12}x_2 + \ldots + v_{1p}x_p$$

$$x'_2 = v_{21}x_1 + v_{22}x_2 + \ldots + v_{2p}x_p$$

...

$$x'_p = v_{p1}x_1 + v_{p2}x_2 + \ldots + v_{pp}x_p$$

such that:

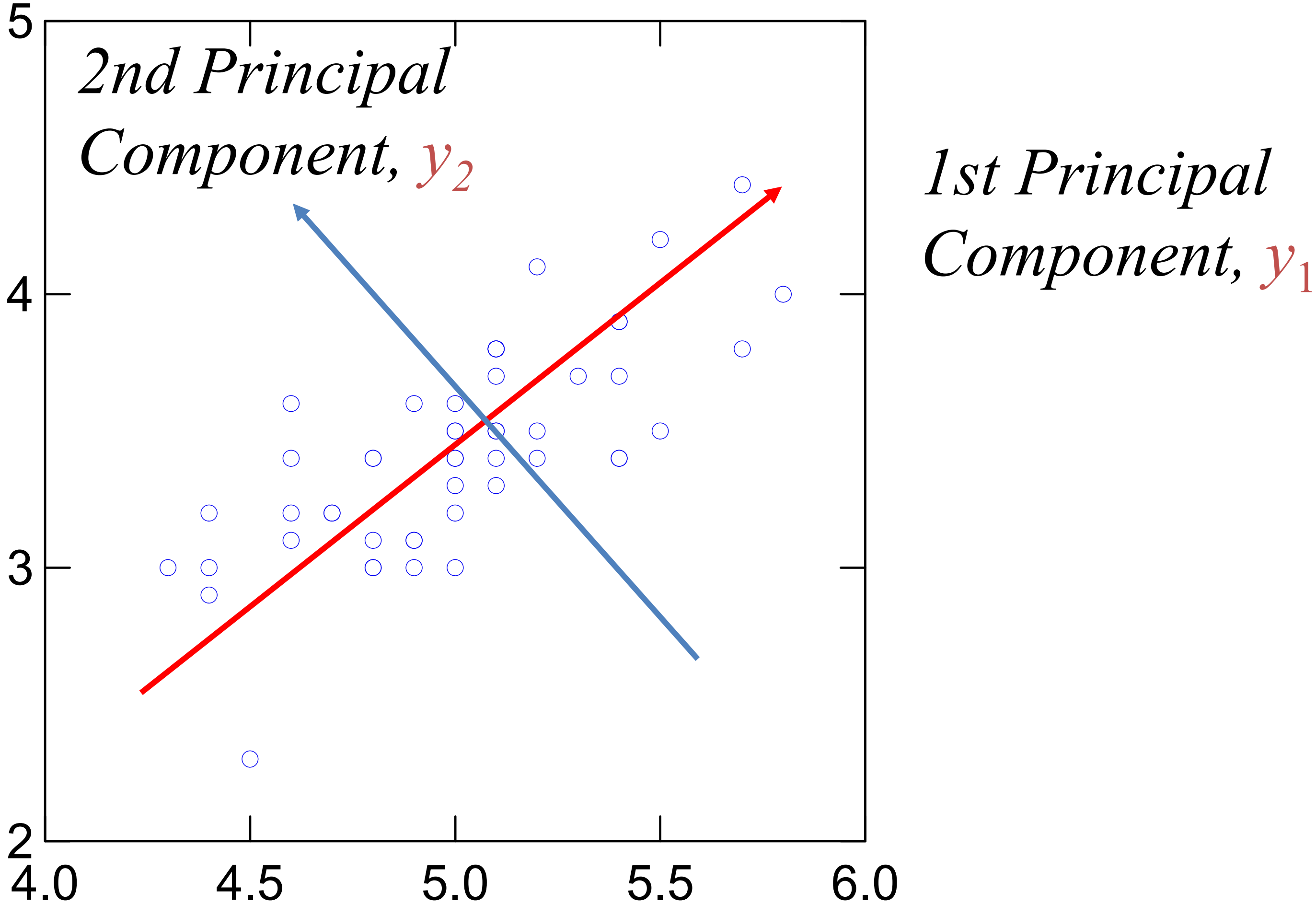
x'_i's are uncorrelated (orthogonal)

x'_1 explains as much as possible of original variance in data set

x'_2 explains as much as possible of remaining variance

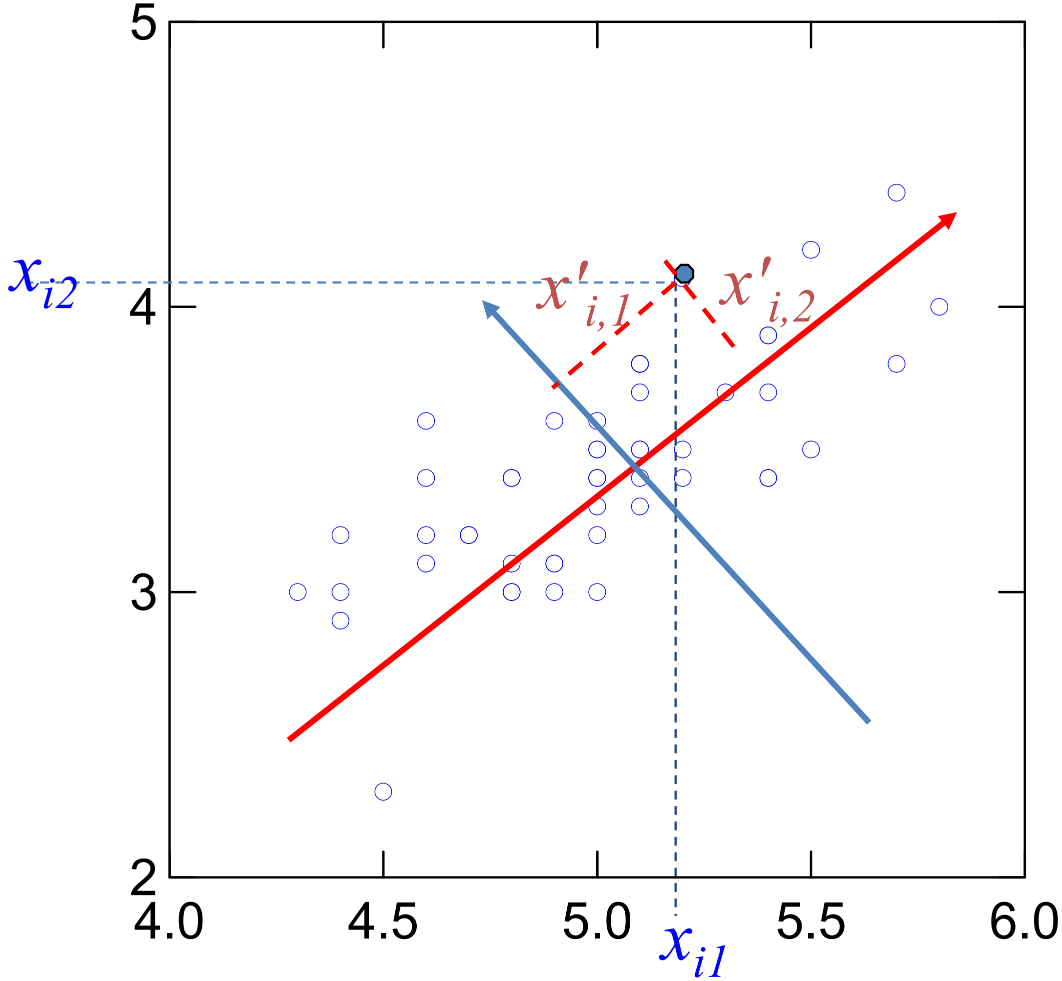
etc.

Adapted from slides by Prof. S. Narasimhan, “Computer Vision” course at CMU



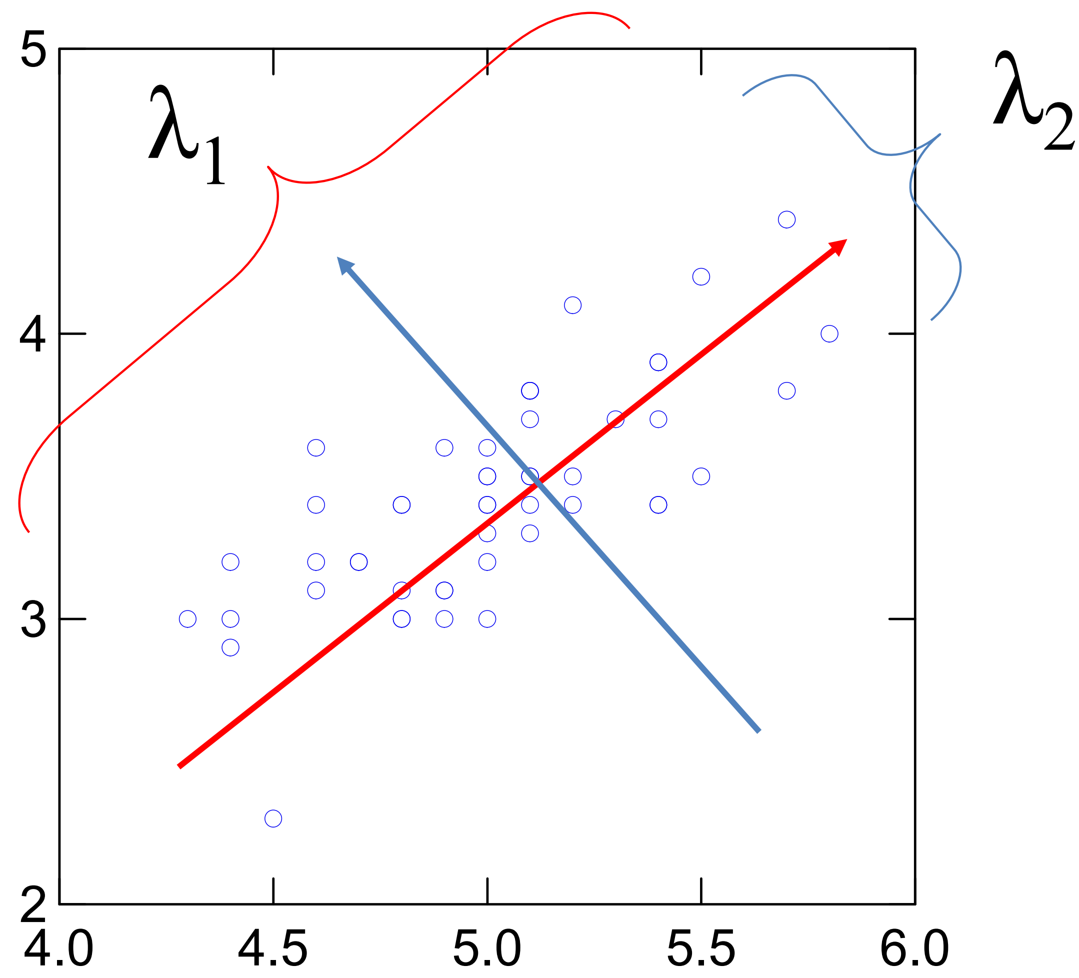
Adapted from slides by Prof. S. Narasimhan, “Computer Vision” course at CMU

PCA Scores



Adapted from slides by Prof. S. Narasimhan, “Computer Vision” course at CMU

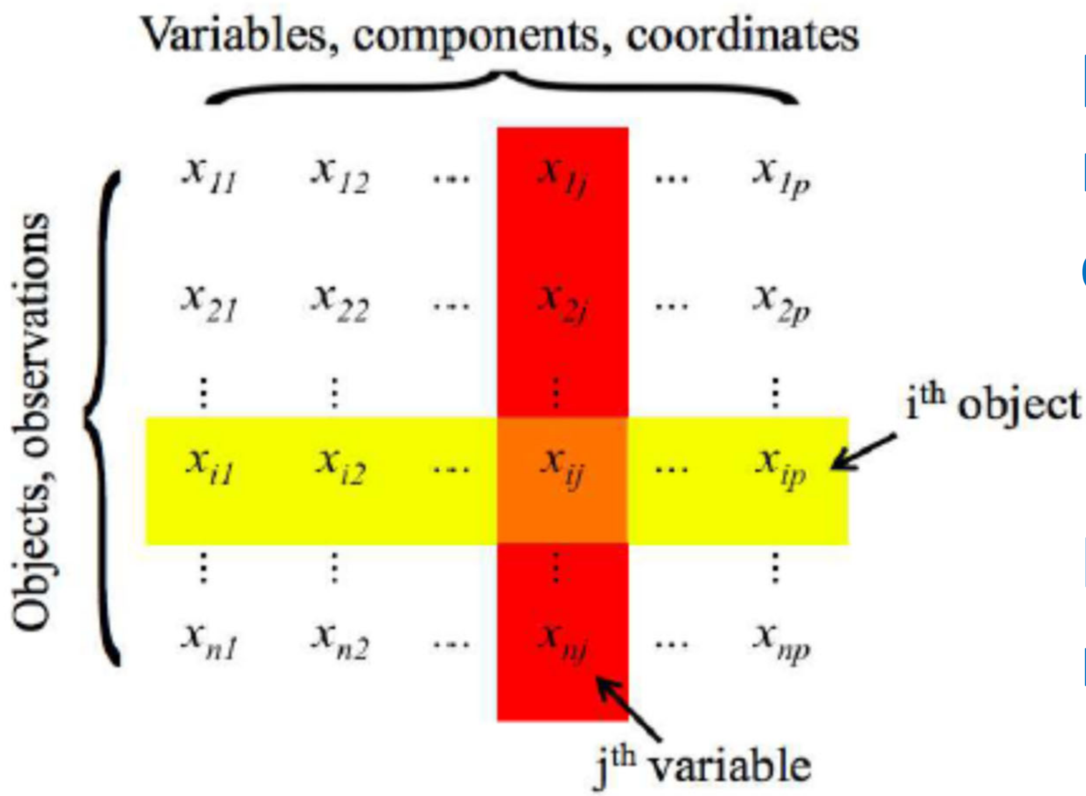
PCA Eigenvalues



Adapted from slides by Prof. S. Narasimhan, “Computer Vision” course at CMU

Multivariable statistics and **Principal Component Analysis (PCA)**

- A table of n observations in which p variables were measured



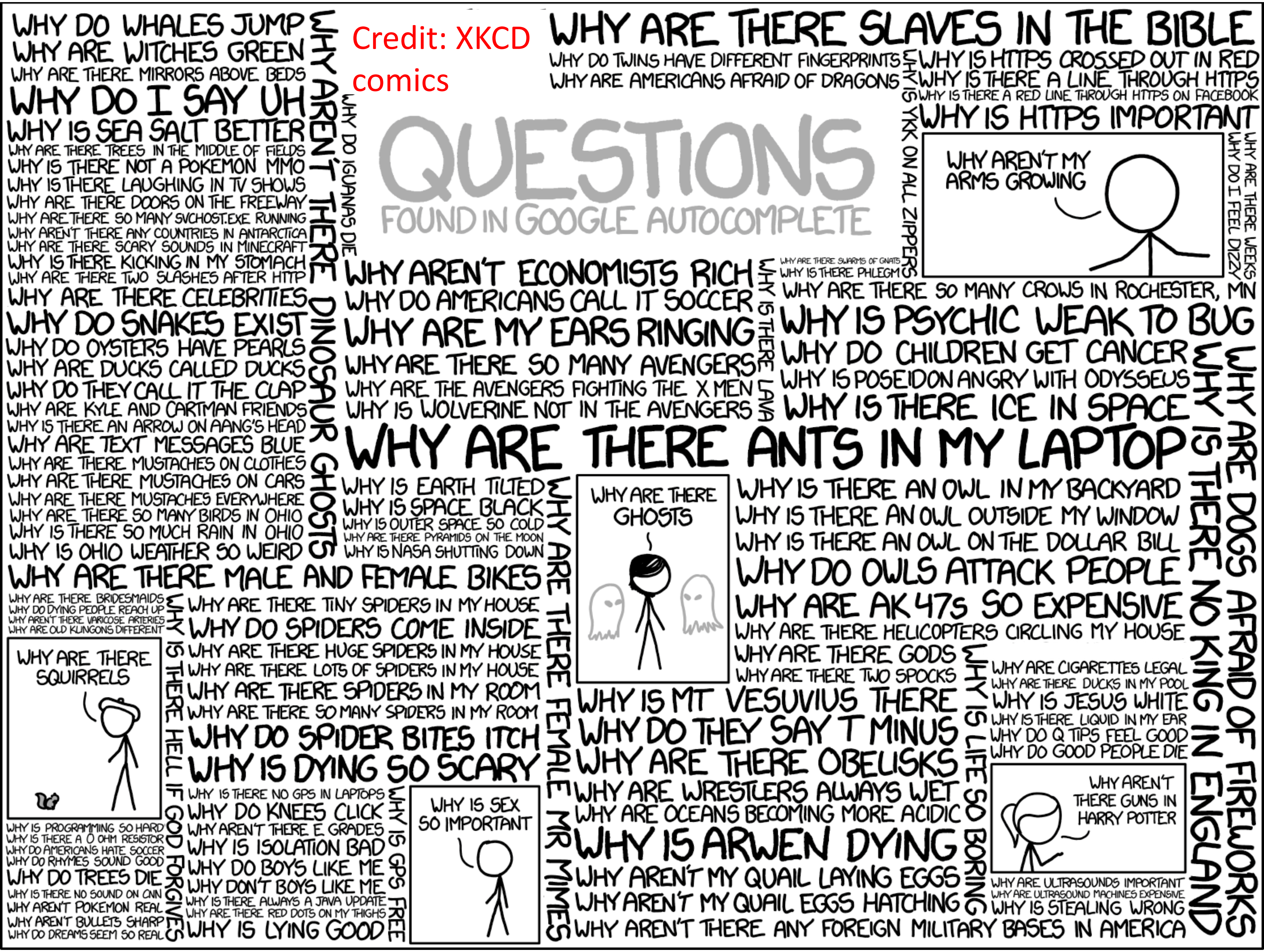
p x p symmetric matrix R of corr. coefficients

$$r_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

PCA: Diagonalize matrix R

Principle Component Analysis (PCA)

- p x p symmetric matrix *R* of corr. coefficients $r_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$
- *R*=*n*$^{-1}$*Z'***Z* is a "square" of the matrix Z of standardized r.v.: $z_{\alpha i} = \frac{x_{\alpha i} - \mu_i}{\sigma_i}$ → all eigenvalues of R are non-negative
- Diagonal elements=1 → *tr(R)=p*
- Can be diagonalized:
  *R=V*D*V'* where *D* is the diagonal matrix
- d(1,1) –largest eig. value, d(p,p) – the smallest one
- The meaning of V(i,k) – contribution of the data type i to the k-th eigenvector
- tr(D)=p, the largest eigenvalue d(1,1) absorbs a fraction =d(1,1)/p of all correlations can be ~100%
- Scores: X'=Z*V: n x p matrix. Meaning of *X'*(α,*k*) – participation of the sample # α in the *k*-th eigenvector

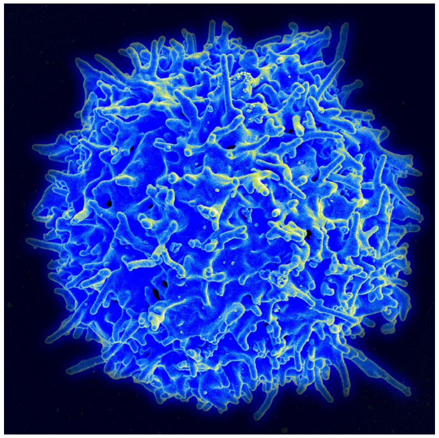


Credit: XKCD comics

Human T cell expression data

- The matrix contains 47 expression samples from Lukk et al, Nature Biotechnology 2010
- All samples are from T cells in different individuals
- Only the top 3000 genes with the largest variability were used
- The value is log2 of gene's expression level in a given sample as measured by the microarray technology

a T cell



A global map of human gene expression

Margus Lukk, Misha Kapushesky, Janne Nikkilä, Helen Parkinson, Angela Goncalves, Wolfgang Huber, Esko Ukkonen & Alvis Brazma

Affiliations | Corresponding author

Nature Biotechnology **28**, 322–324 (2010) | doi:10.1038/nbt0410-322

Although there is only one human genome sequence, different genes are expressed in many different cell types and tissues, as well as in different developmental stages or diseases. The structure of this 'expression space' is still largely unknown, as most transcriptomics experiments focus on sampling small regions. We have constructed a global gene expression map by integrating microarray data from 5,372 human samples representing 369 different cell and tissue types, disease states and cell lines. These have been compiled in an online resource (http://www.ebi.ac.uk/gxa/array/U133A) that allows the user to search for a gene of interest and

Matlab exercise on MLR

- Every group works with
  g0=2907; g1=1527; g2=2629; g3=2881;
  g4=1144; g5=1066;
- Compute Multiple Linear Regression (MLR): where
  y=exp_t (g0); x1= exp_t (g1); x2= exp_t (g2);
- How much better the MLR did compared to the Single Linear Regression (SLR)?
- Continue increasing the number of genes in x until R_adj starts to decrease

How I did it

```
• g0=2907; g1=1527; g2=2629; g3=2881;g4=1144; g5=1066;
• y=exp_t(g0,:)';
• %% first use one x to predict y
• x=exp_t(g1,:)';
• figure; plot(x,y,'ko')
• lm=fitlm(x,y)
• y_fit=lm.Fitted;
• hold on;
• plot(x,lm.Fitted,'r-');
• %% now use 2 x's to predict y
• x=[exp_t(g1,:)', exp_t(g2,:)'];
• lm2=fitlm(x,y)
• y_fit=lm2.Fitted;
• hold on; plot(x(:,1),y_fit,'gd');
• %% now use m x's to predict y
• corr_matrix=corr(exp_t');
• g0=2907;
• [u v]=sort(corr_matrix(g0,:),'descend');
• x=[exp_t(v(2:m+1),:)'];
• lm3=fitlm(x,y)
• y_fit=lm3.Fitted;
• plot(x(:,1),y_fit,'s');
```

Credit: XKCD comics