CS170A -- HW#1 -- assignment and solution form -- Octave

Your name: Xuemin He

Your UID: 204468663

Please upload only this notebook to CCLE by the deadline.

Policy for late submission of solutions: We will use Paul Eggert's Late Policy: N days late $\Leftrightarrow$ 2^N points deducted} The number of days late is $N = 0$ for the first 24 hrs, $N = 1$ for the next 24 hrs, etc., and if you submit an assignment H hours late, $2^{\lfloor H/24 \rfloor}$ points are deducted.

NOTE: In this assignment we provide pseudocode to get you started.

In later assignments we will not do this.

Problem 1: SVD k-th order approximations (30 points)

If A is a matrix that has SVD $A = U\,S\,V'$, the **rank-k approximation of** A keeping only the first k columns of the SVD.

Specifically, given a $n \times p$ matrix A with SVD $A = U\,S\,V'$, then if $k \le n$ and $k \le p$, the rank-k approximation of A is

$$A^{(k)} \;\; = \;\; U\,S^{(k)}\,V'$$

where $S^{(k)}$ is the result of setting all diagonal elements to zero after the first k entries $(1 \le k \le p)$. If $U^{(k)}$ and $V^{(k)}$ are like U and V but with all columns zero after the first k, then

$$A^{(k)} \;\; = \;\; U\,S^{(k)}\,V' \;\; = \;\; U^{(k)}\,S^{(k)}\,V^{(k)'}.$$

In class, we saw a demo of the attached Matlab script `imagesvdgui.m` --- and the effectiveness of this approximation in retaining information about an image.

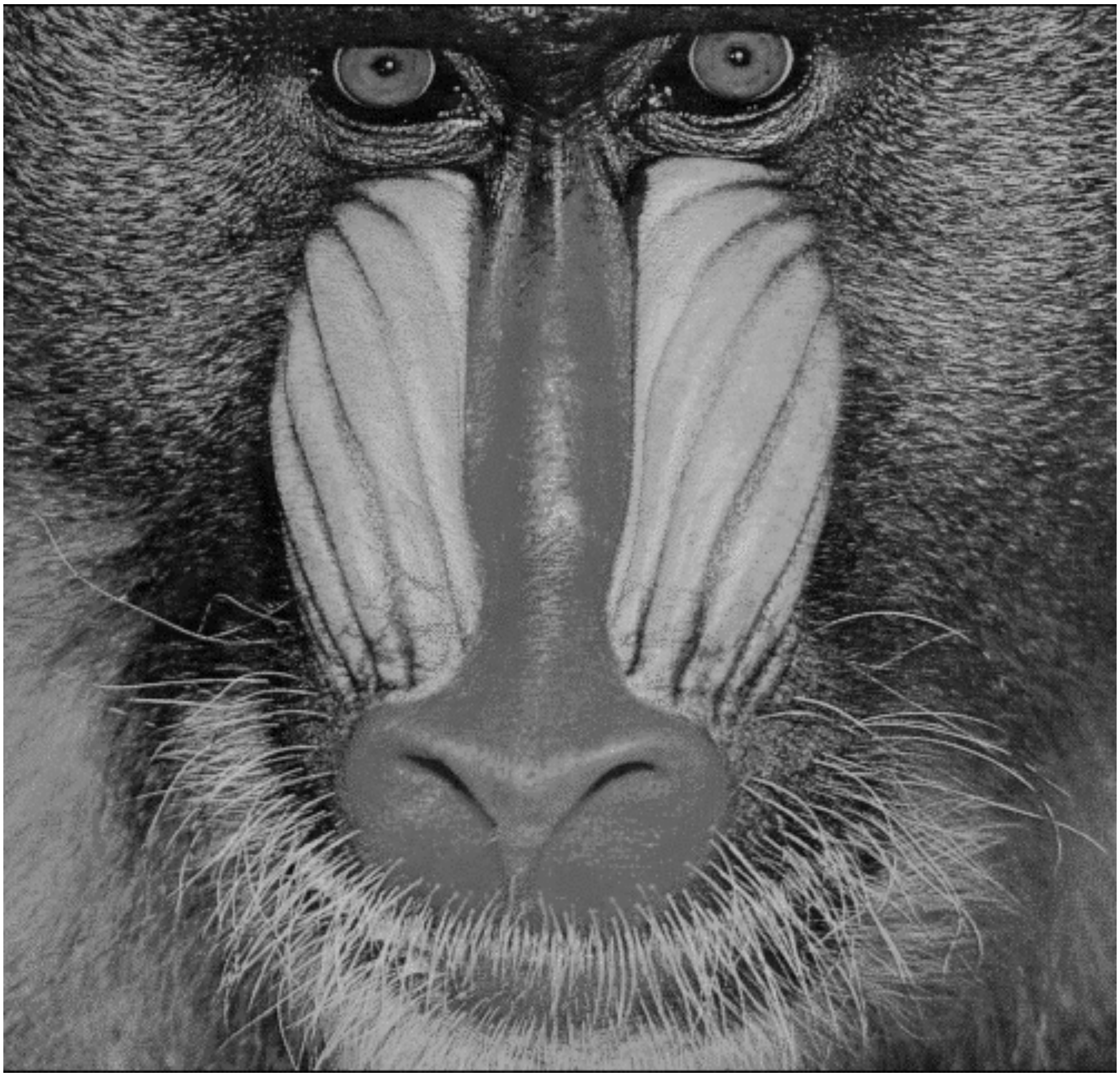
The goal of this problem is to implement this approximation for black-and-white (grayscale) images.

In [2]:

```
load mandrill
Mandrill = ind2rgb(X, map);

A = mean( Mandrill, 3 );  # grayscale image -- size 480 x 500.
size(A)

imwrite(A, 'GrayMandrill.bmp')  % Write the Mandrill to a bitmap image file

% The matrix A now contains the Mandrill image (in grayscale)
```

```
ans =

   480   500
```

Display the bitmap image file using an HTML img tag:



<img src="GrayMandrill.bmp">

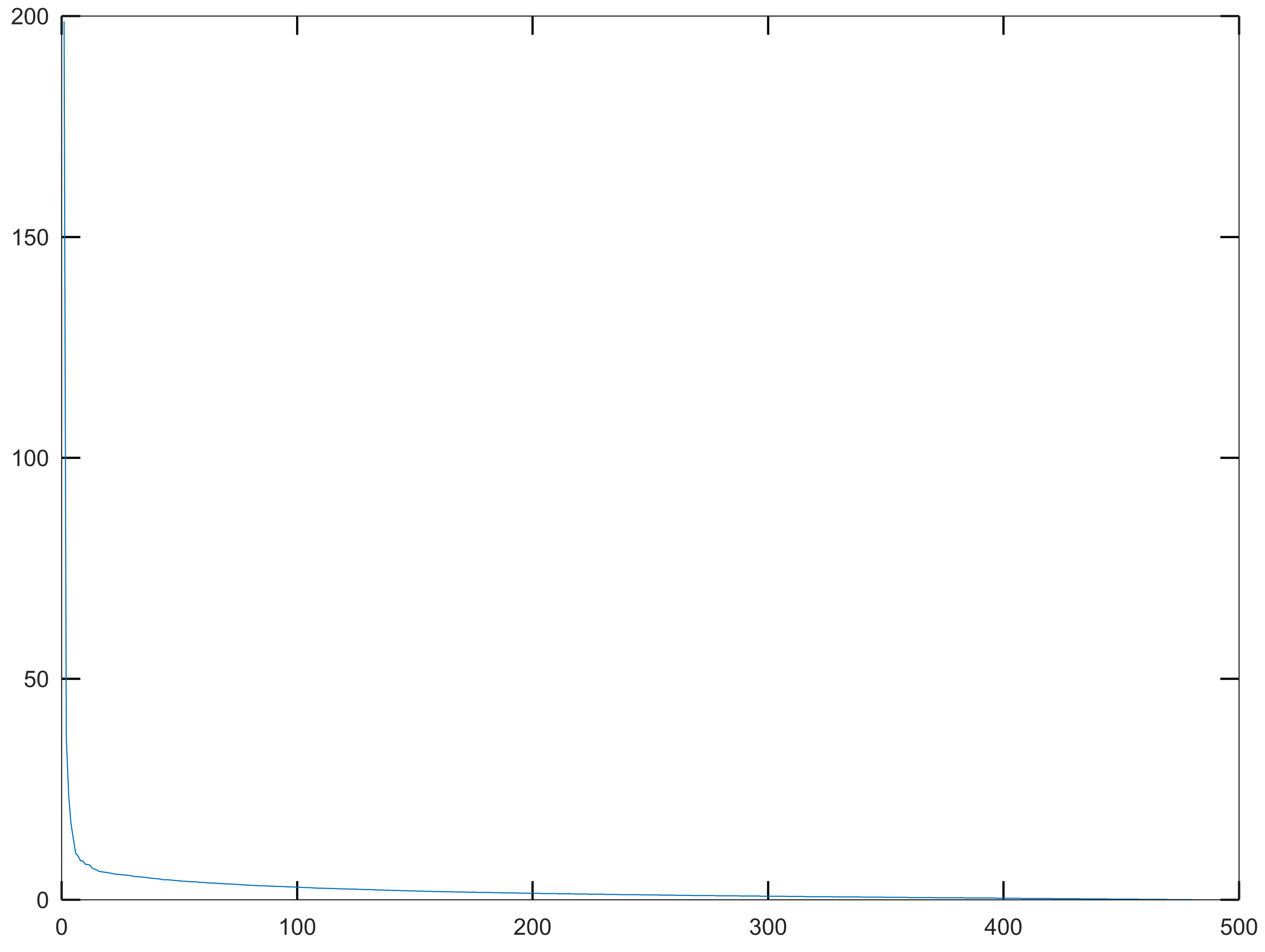
1.(a): Plot Singular Values of the Rank-k Approximation of an Image

As in HW0, construct a grayscale version of the Mandrill image, and take one of the 3 color planes as a 500x480 matrix. This is our `black and white' image A. You are to analyze the rank-k approximation of the image.

Compute the SVD of A, and plot the singular values σ_1 , σ_2 , ...

In [3]:

```
[U S V] = svd(A);         % U, S, V are now the SVD of A

norm( A - U * S * V' );  %  A should match the product of U, S, V'

sigmas = diag(S);
plot(sigmas)
```



1.(b): Optimal Rank-k Approximation of an Image

Find the value of k that minimizes $|| A - A^{(k)} ||_F^2 + k$.

In [4]:

```
[n p] = size(A);
maximum_possible_k = min(n,p);

orig = A;
[U S V] = svds(A,1);
oldResult = norm(orig - U*S*V', "fro")^2 + 1;

for k=2:maximum_possible_k
    [U S V] = svds(orig,k);
    A = U*S*V';
    result = norm(orig-A, "fro")^2 + k;
   if (result - oldResult < 0)
       break;
    endif
    oldResult = result;
end

k
```

```
k =  2
```

1.(c): The Rank-k Approximation is a Good Approximation

In the chapter on the SVD, the course reader presents a derivation for $A \; - \; A^{(k)}$:

$$\begin{aligned} A \; - \; A^{(k)} &= U \; S \; V' \; - \; U^{(k)} \; S^{(k)} \; V^{(k)'} \\ &= U \; S \; V' \; - \; U \; S^{(k)} \; V' \\ &= U \; (S \; - \; S^{(k)}) \; V' \end{aligned}$$

Prove the following:

$$|| \, A \; - \; A^{(k)} \; ||_F^2 \quad = \quad \sum_{i>k} \sigma_i^2 .$$

Proof (Enter your Proof here)

Because $A \ - \ A^{(k)} \ = \ U \ (S \ - \ S^{(k)}) \ V'$,

$S = diag([\sigma_1, \sigma_2, \ldots, \sigma_k, \ldots, \sigma_n]), \ S^{(k)} = diag([\sigma_1, \sigma_2, \ldots, \sigma_k])$,

$S - S^{(k)} \ = \ diag([\sigma_{k+1}, \sigma_{k+2}, \ldots, \sigma_n])$.

As a result, $|| \ A \ - \ A^{(k)} \ ||_F^2$

$= \ || \ U \ (S \ - \ S^{(k)}) \ V' \ ||_F^2$

$= \ trace(\ (U \ (S \ - \ S^{(k)}) \ V')' \ (U \ (S \ - \ S^{(k)}) \ V') \ )$

$= \ trace(V \ (S \ - \ S^{(k)})' \ U' \ U \ (S \ - \ S^{(k)}) \ V')$

$= \ trace(V \ (S \ - \ S^{(k)})' \ (S \ - \ S^{(k)}) \ V')$

$= \ trace(V \ (\sigma_{k+1}^2 + \sigma_{k+2}^2 + \ldots + \sigma_n^2) \ V')$

$= \ (\sum_{i>k} \sigma_i^2 \ )trace(V' \ V)$

$= \ \sum_{i>k} \sigma_i^2$ since V is unitary.

Therefore

$$|| \ A \ - \ A^{(k)} \ ||_F^2 \ = \ \sum_{i>k} \sigma_i^2 .$$

Problem 2: Baseball Visualization (40 points)

For this dataset you are given a matrix of statistics for Baseball players. You are to perform two kinds of analysis on this matrix.

Read in the Baseball Statistics

Statistics of top players after the last regular season game, obtained from MLB.com, October 2016.

In [5]:

```
%%% Stats = csvread('Baseball_Players_Stats_2016.csv', 1, 0);  # skip the header
(= row 0)
%%% Names = csvread('Baseball_Players_Names_2016.csv', 1, 0);

Baseball_Players_2016   %% execute Baseball_Players_2016.m  to load in the data
needed here
```

```
 added to session magics.
```

In [6]:

```
StatNames{1:3}

size(StatNames)

StatNames{:}
```

```
ans = Rank
ans = G
ans = AB
ans =

    1   17

ans = Rank
ans = G
ans = AB
ans = R
ans = H
ans = 2B
ans = 3B
ans = HR
ans = RBI
ans = BB
ans = SO
ans = SB
ans = CS
ans = AVG
ans = OBP
ans = SLG
ans = OPS
```

In [7]:

```
size(Stats)
Stats(1:3, :)
```

```
ans =

   146    17

ans =

 Columns 1 through 6:

     1.00000   146.00000   552.00000   104.00000   192.00000    32.0
0000
     2.00000   142.00000   531.00000    88.00000   184.00000    47.0
0000
     3.00000   161.00000   640.00000   108.00000   216.00000    42.0
0000

 Columns 7 through 12:

     8.00000    11.00000    66.00000    66.00000    80.00000    11.0
0000
     5.00000    25.00000   104.00000    35.00000    57.00000     5.0
0000
     5.00000    24.00000    96.00000    60.00000    70.00000    30.0
0000

 Columns 13 through 17:

     7.00000     0.34800     0.41600     0.49500     0.91100
     3.00000     0.34700     0.39000     0.59500     0.98500
    10.00000     0.33800     0.39600     0.53100     0.92800
```

In [8]:

```
size(PlayerNames)
PlayerNames{1:3}
```

```
ans =

   146     1

ans =   LeMahieu D
ans =   Murphy D
ans =   Altuve J
```

Compute a "scaled" version of the Stats matrix

We scale each column of values $\mathbf{x}$ in Stats to be $\mathbf{z} = (\mathbf{x} - \mu)/\sigma$ in ScaledStats, where μ is the mean of the $\mathbf{x}$ values, and σ is their standard deviation.

In Octave/Matlab, the function `mean()` computes column means, and `std()` computes standard deviations. The function `zscore()` computes both, and uses them to "scale" each column in this way.

This scaling is also called **normalization** and **standardization**. The **z-scores** $\mathbf{z} = (\mathbf{x} - \mu)/\sigma$ are also called the standardized or normalized values for $\mathbf{x}$.

In [9]:

```
ScaledStats  =  zscore(Stats);   %  z  =  (x-mu)/sigma

mean(ScaledStats)  %  the means of each column after normalization should be 0
std(ScaledStats)   %  the standard deviations of each column after normalization
should be 1
```

```
ans =

 Columns 1 through 6:

  -3.1938e-17   5.0797e-16  -1.1140e-15  -1.6805e-16   2.6159e-16
2.2509e-16

 Columns 7 through 12:

  -4.5626e-17  -4.6766e-17   2.3573e-16  -1.5632e-16  -1.3079e-16  -
3.5740e-17

 Columns 13 through 17:

  -2.4714e-17   4.5930e-15   6.1853e-15  -1.4676e-15   7.8856e-15

ans =

 Columns 1 through 8:

   1.00000   1.00000   1.00000   1.00000   1.00000   1.00000   1.000
00   1.00000

 Columns 9 through 16:

   1.00000   1.00000   1.00000   1.00000   1.00000   1.00000   1.000
00   1.00000

 Column 17:

   1.00000
```

2 (a): Random Projections

A fundamental problem in data science is that it is impossible to visualize a dataset that has many features. Given an $n \times p$ dataset (matrix) A in which the number of features p is large, there is no obvious way to plot the data.

Dimensionality reduction algorithms have been developed that attempt to find datasets that have lower values of p but approximate A in some way. Although there are sophisticated algorithms, a competitive approach is to compute a ***random projection*** of A into a few dimensions. When the projection is into 2 or 3 dimensions, the result can be visualized.

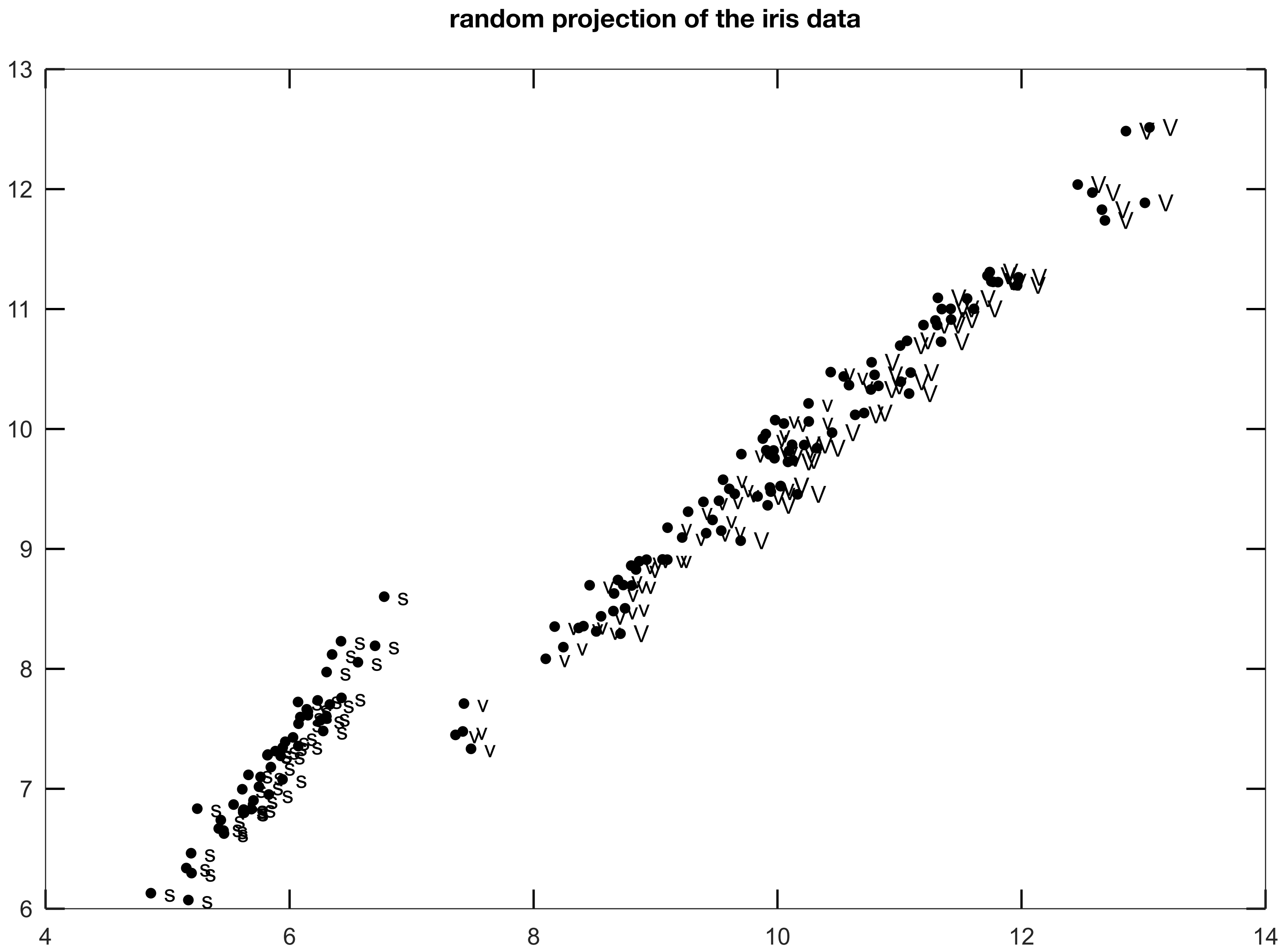
A ***random*** k***-D projection*** of a $n \times p$ dataset (matrix) A is the result $(A\,P)$ of multiplying A on the right by a $p \times k$ matrix P of random values.

The result is a $n \times k$ matrix, assigning each row in A a new pair of values (x, y), and these can be interpreted as positions in a 2D plot.

In [10]:

```
% plotting 2D values
Iris = csvread('iris.csv', 1,0);   % skip over the header line
A = Iris(:, 1:4);   % just the measurement columns
[n p] = size(A);
P = rand(p,2);
disp('random projection weights:')
disp(P)
XY = A * P;
plot(XY(:,1), XY(:,2), 'b.')
title('random projection of the iris data')
species = {'  s', '  v', '  V'}
text(XY(:,1), XY(:,2), species(Iris(:,5)), 'fontsize', 10 )
```

```
random projection weights:
   0.80574   0.92153
   0.20434   0.62176
   0.64155   0.20403
   0.80413   0.76801
species =
{
  [1,1] =   s
  [1,2] =   v
  [1,3] =   V
}
```



Problem: write a function `random_projection(A,k)` that, given an input matrix A of size $n \times p$ and an integer $k > 0$, produces a random k-D projection.

Please use uniform random values in the matrix P.

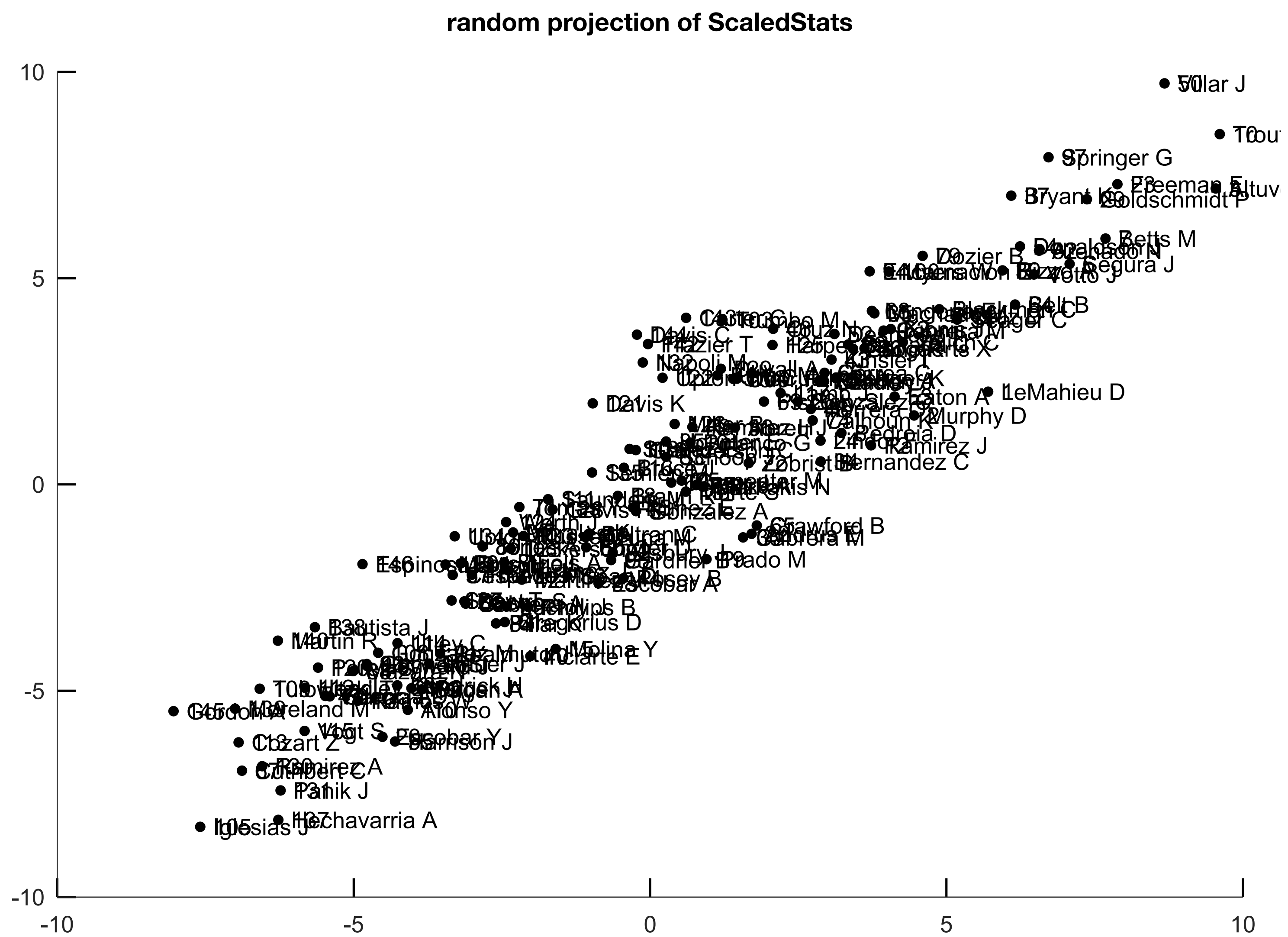
Then: plot the result of 3 random 2D projections of the data.

In each plot, identify the ***greatest outlier*** -- the player with (x, y) values that have the largest total $x + y$. Print the row in the dataset whose projection is this outlier.

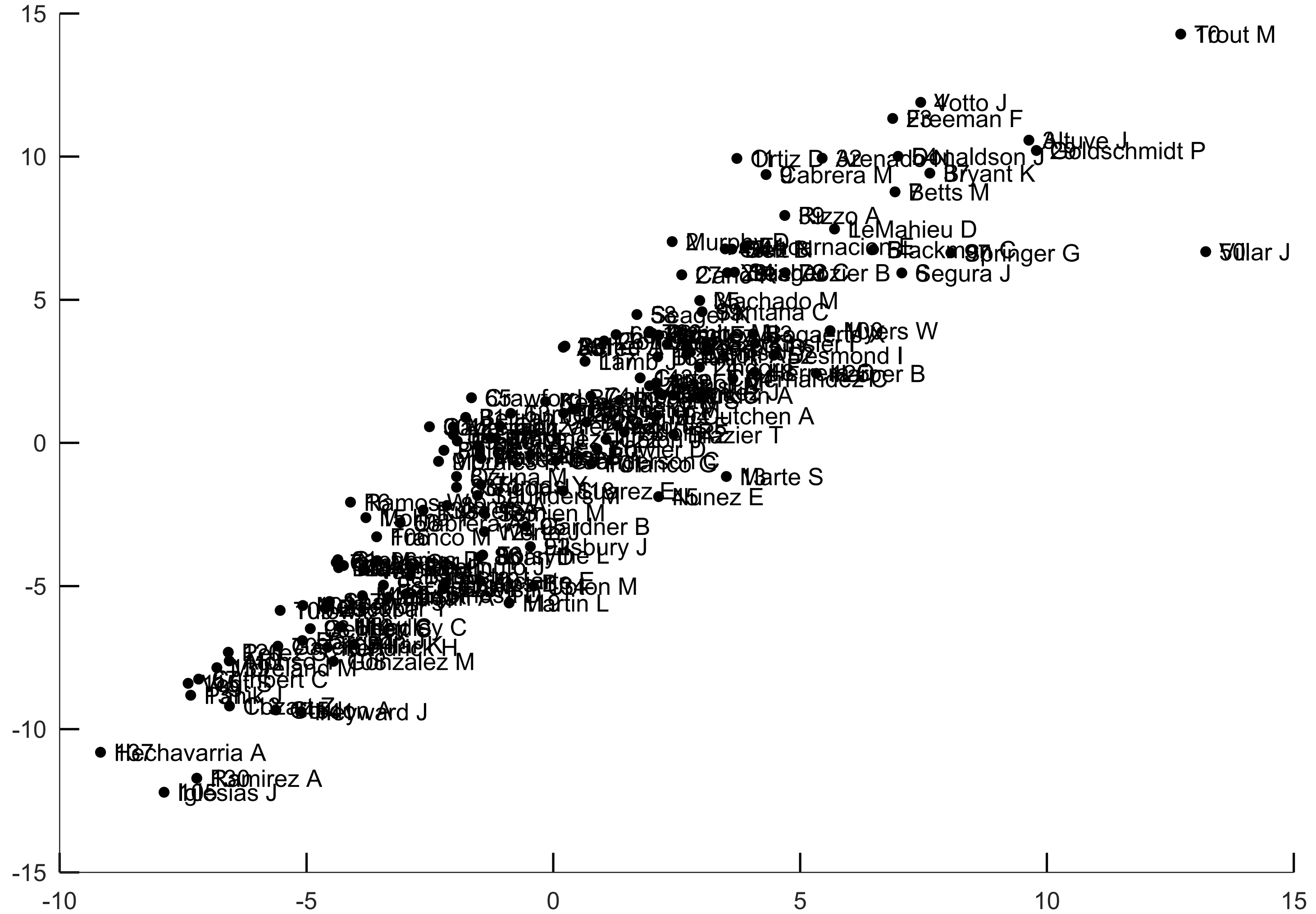
In [11]:

```
function XY = random_projection(A,k)
    [n p] = size(A);
    p = rand(p,k);
    XY = A * p;
endfunction

for i = 1:3
    XY = random_projection( ScaledStats, 2 );
    figure();
    hold on
    plot(XY(:,1),XY(:,2),'b.')
    title('random projection of ScaledStats')
    text(XY(:,1),XY(:,2), PlayerNames, 'fontsize', 10 )
    text(XY(:,1),XY(:,2), PlayerRanks, 'fontsize', 10 )
    XplusY = sum(XY,2);
    greatest_outlier = max(XplusY)
end
```

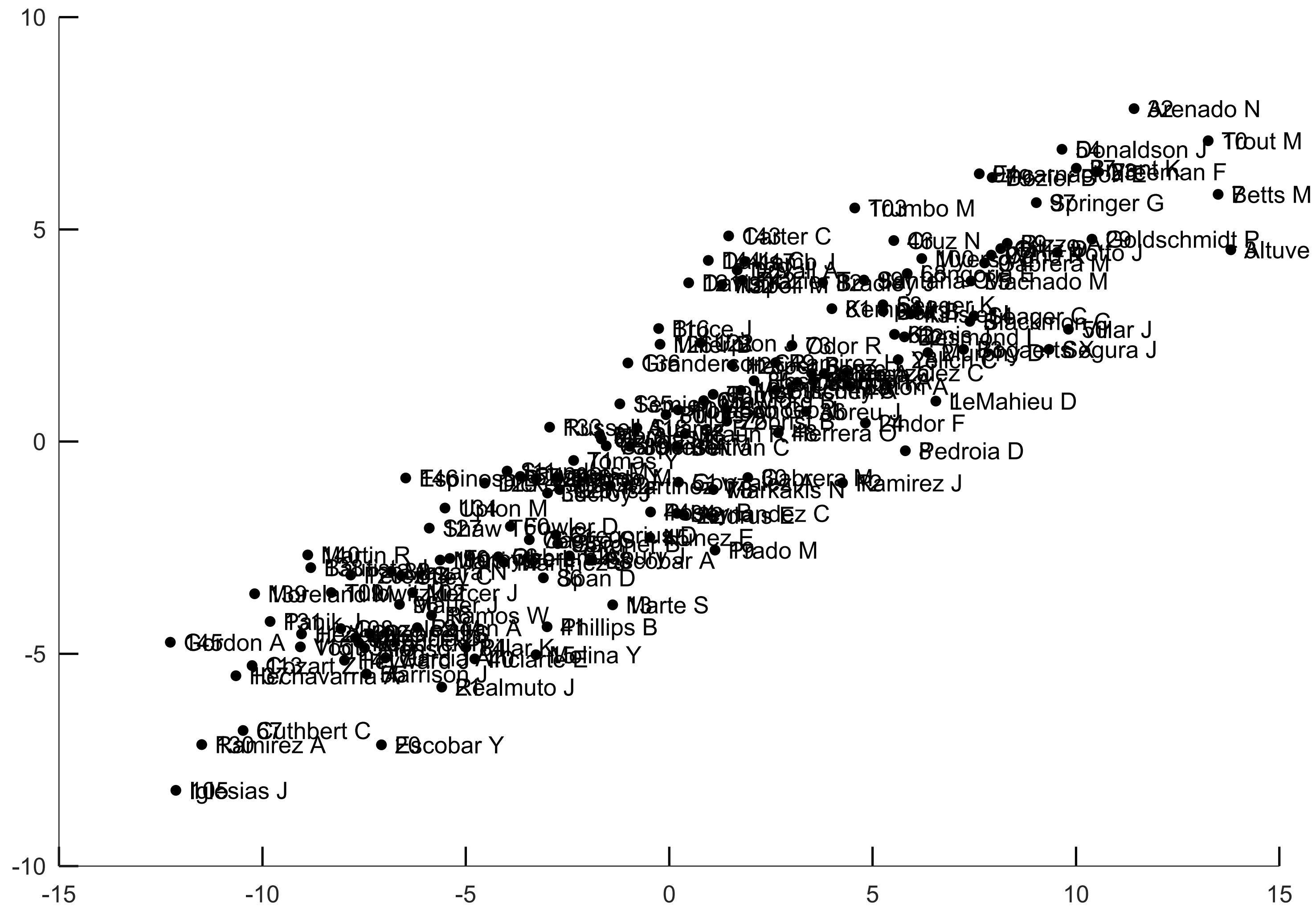
```
greatest_outlier =  20.342
greatest_outlier =  26.994
greatest_outlier =  18.405
```



random projection of ScaledStats



random projection of ScaledStats



2 (b): Latent Semantic Analysis

The course reader describes **Latent Semantic Indexing** for a matrix of values measuring association between X terms vs. Y terms.

The classic example is a "term/document matrix" for Keywords vs. Books, shown below. The code shown produces an LSI plot for the data.

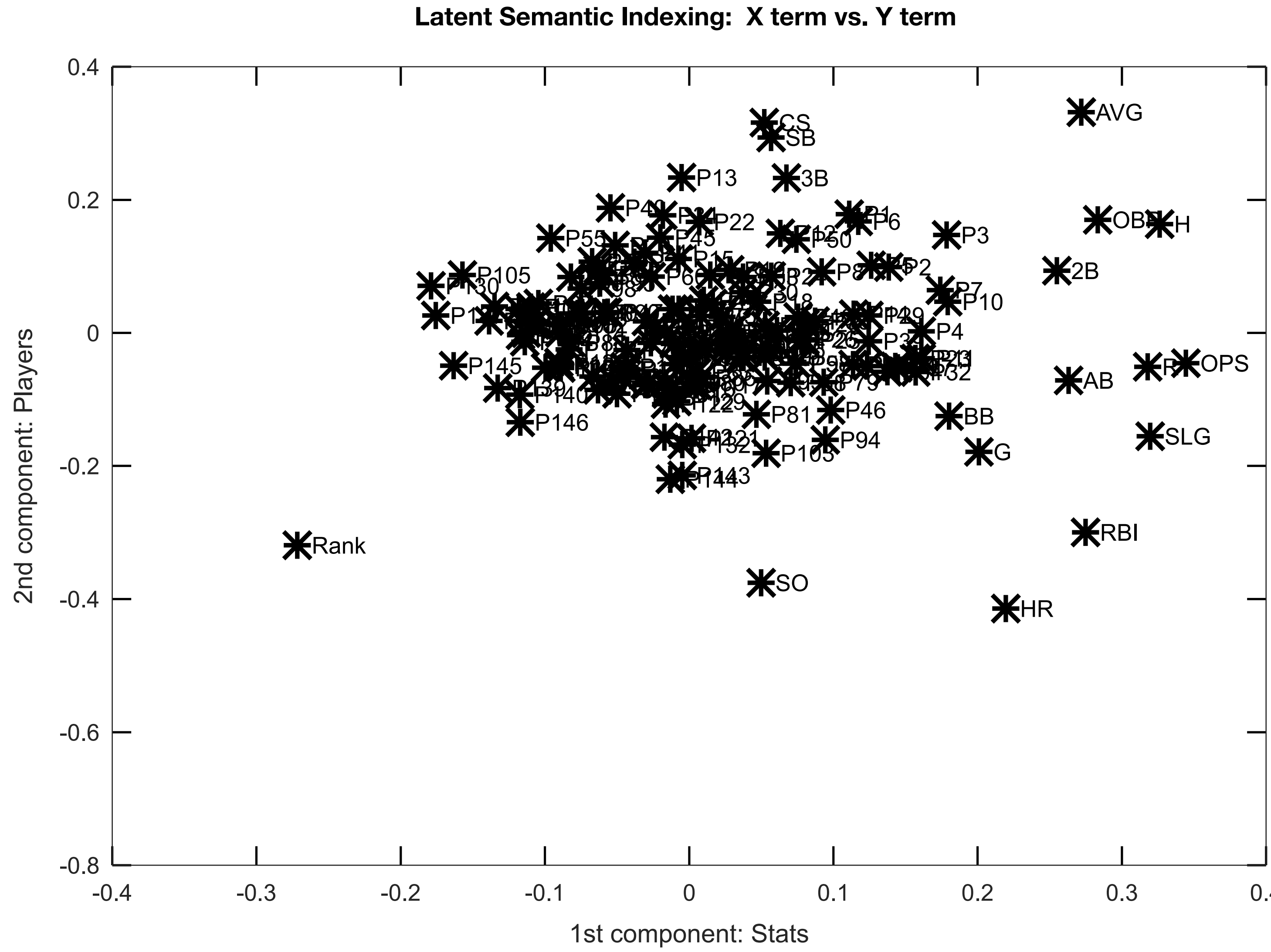
Your job is to produce an analogous LSI plot for the table of Baseball players.

Components are computed as in: Berry, M. W., Dumais, S. T., and O'Brien, G. W. (1995). "Using linear algebra for intelligent information retrieval." SIAM Review, 37(4), 1995, 573-595.

Some LSI references: [lsi.research.telcordia.com/lsi/LSIpapers.html (http://lsi.research.telcordia.com/lsi/LSIpapers.html)](http://lsi.research.telcordia.com/lsi/LSIpapers.html)

In [12]:

```
X = StatNames';
nX = numel(X);

Y = PlayerNames;
nY = numel(Y);

coOccurrence = ScaledStats';

[U S V] = svd(coOccurrence);

plot( diag(S), 'b' )

Xfactor = U(:,1:2);
Yfactor = V(:,1:2);

% plot the 2D projection of the data

text_offset = 0.01;
plot( Xfactor(:,1), Xfactor(:,2), 'r*' )
hold on
plot( Yfactor(:,1), Yfactor(:,2), 'b*' )

for i = (1:nX)
    text( Xfactor(i,1)+text_offset, Xfactor(i,2), X(i))
end

for i = (1:nY)
    text( Yfactor(i,1)+text_offset, Yfactor(i,2), sprintf('P%d',i))
end

title( 'Latent Semantic Indexing:  X term vs. Y term ' )
xlabel( '1st component: Stats' )
ylabel( '2nd component: Players' )

zoom on
hold off
```



In [ ]:

```

```

Problem 3: Global Warming again (30 points)

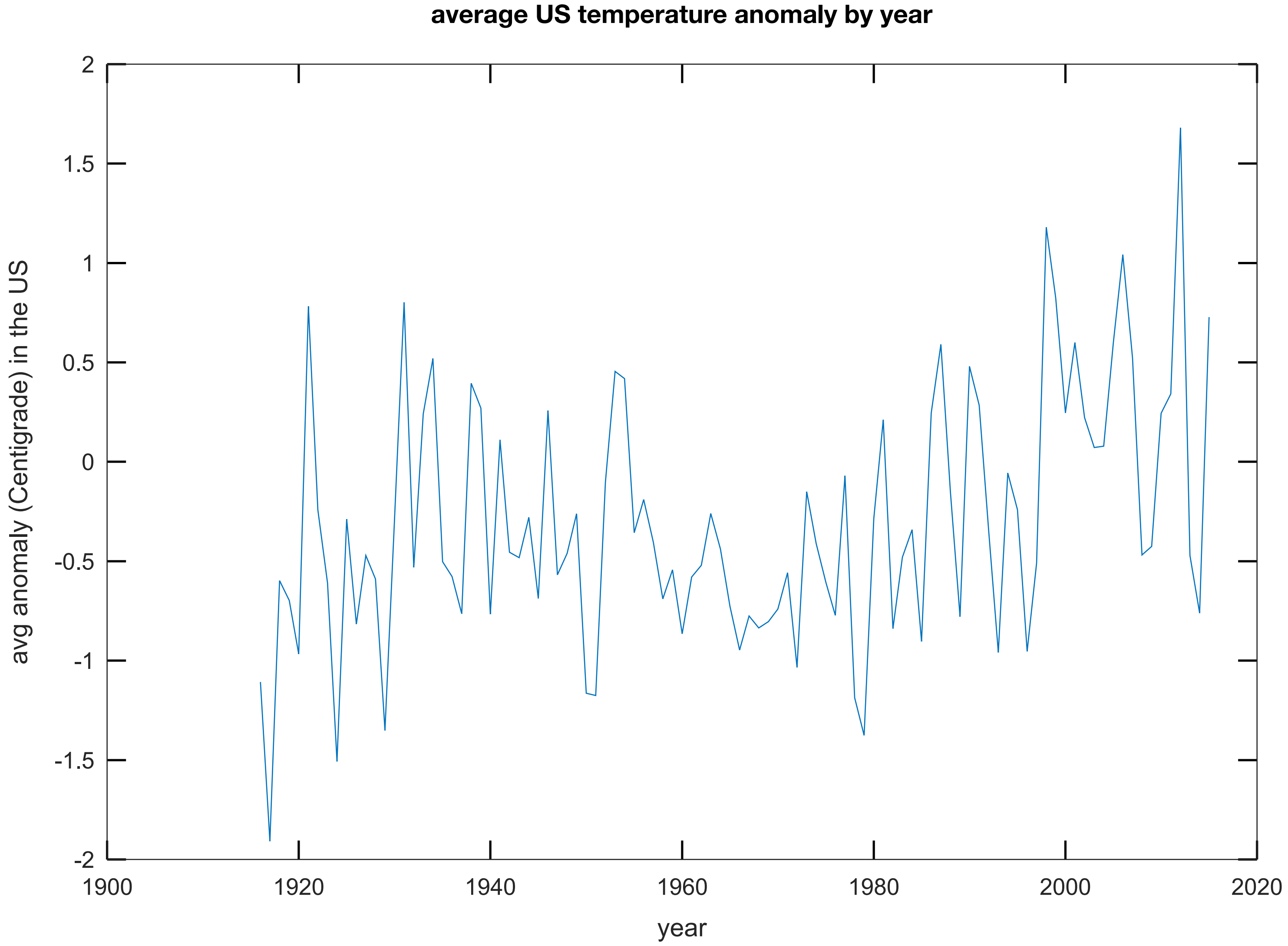
In HW0, you plotted the average (non-missing-value) temperature anomaly over the entire grid, for every year from 1916 to 2015.

In this problem we want you to fit linear models through the data.

In [13]:

```
% set up everything as in HW#0:

GHCN = csvread('ghcn.csv');

%   The data was artificially shifted to [0, 4500];
%     its range should be [-2500, +2000]/100 = [-25,+20], in degrees Centigrade.
%     Since our focus here is on warming, we ignore temperatures below -5.
%   We omit the year and month in columns 1:2 before scaling:

GHCN_in_centigrade  = (GHCN(:,3:74) - 2500) / 100;

temperature_anomaly = reshape( GHCN_in_centigrade, [36, 12, 137, 72] );   % conv
ert to a 4D matrix, so we can use slices

missing_values = (temperature_anomaly == -25);
number_of_missing_values = sum(sum(sum(sum( missing_values ))));

maximum_anomaly_value = max(max(max(max( temperature_anomaly ))))
minimum_anomaly_value = min(min(min(min( temperature_anomaly .*  (~ missing_valu
es) ))))  %  '~' is 'not' in MATLAB

US_latitude  = 9:12
US_longitude = 15:20
my_years = 1916:2015;
my_slice = temperature_anomaly( US_latitude, :, my_years - 1880 + 1, US_longitud
e );

total_number_of_grid_squares = length(US_latitude) * length(US_longitude) * 12
N = total_number_of_grid_squares

average_US_anomaly_by_year = reshape( sum(sum(sum( my_slice, 4),2),1), [length(m
y_years) 1] ) / N;

plot( my_years, average_US_anomaly_by_year )
xlabel('year')
ylabel('avg anomaly (Centigrade) in the US')
title('average US temperature anomaly by year')
```

```
maximum_anomaly_value =  19.940
minimum_anomaly_value = -24.260
US_latitude =

    9   10   11   12

US_longitude =

   15   16   17   18   19   20

total_number_of_grid_squares =  288
N =  288
```



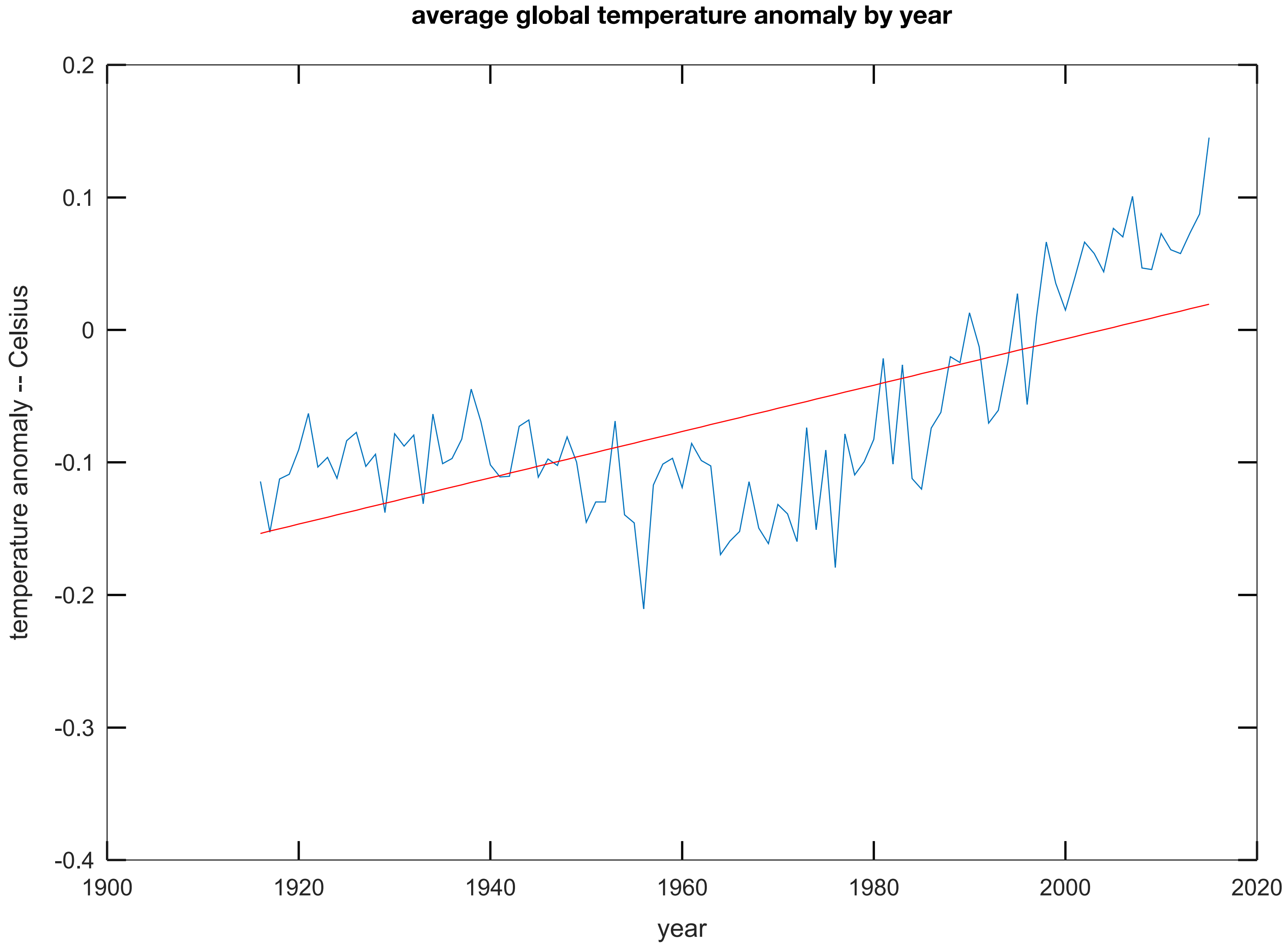
(a) Global Average Temperature Anomaly: Linear Model (Least Squares)

Problem: fit a line through the data, using Least Squares.

In [14]:

```
GHCN = csvread('ghcn.csv');

GHCN_in_centigrade  = (GHCN(:,3:74) - 2500) / 100;

temperature_anomaly = reshape( GHCN_in_centigrade, [36, 12, 137, 72] );   % conv
ert to a 4D matrix, so we can use slices
size( temperature_anomaly )
number_of_all_GHCN_values = prod(size( temperature_anomaly ));


missing_values = (temperature_anomaly == -25);
number_of_missing_values = sum(sum(sum(sum( missing_values ))));

WORLD_latitude = 1:36;
WORLD_longitude = 1:72;

my_years = 1916:2015;
temperature_anomaly = temperature_anomaly .* (~ missing_values);
my_slice = temperature_anomaly( WORLD_latitude, :, my_years - 1880 + 1, WORLD_lo
ngitude );
total_number_of_grid_squares = length(WORLD_latitude) * length(WORLD_longitude)
* 12;
N = total_number_of_grid_squares;

average_WORLD_anomaly_by_year = reshape( sum(sum(sum( my_slice, 4),2),1), [lengt
h(my_years) 1] ) / N;


plot( my_years, average_WORLD_anomaly_by_year );hold
xlabel('year')
ylabel('temperature anomaly -- Celsius')
title('average global temperature anomaly by year')

%

A = [my_years' ones(size(my_years'),1)];
linear_model = A \ average_WORLD_anomaly_by_year;

linear_average_WORLD_anomaly_by_year = linear_model(1) * my_years' + linear_mode
l(2);

plot( my_years, linear_average_WORLD_anomaly_by_year, 'r')
```

```
ans =

    36    12   137    72
```



(b) Global Average Temperature Anomaly: Piecewise Linear Model (Least Squares)

Problem: fit a 2-segment piecewise linear model through the data, using Least Squares.

Specifically, find a pair of least squares models, one from 1916 up to year Y, and one from year Y+1 to 2015, such that the SSE (sum of squared errors) is minimized.

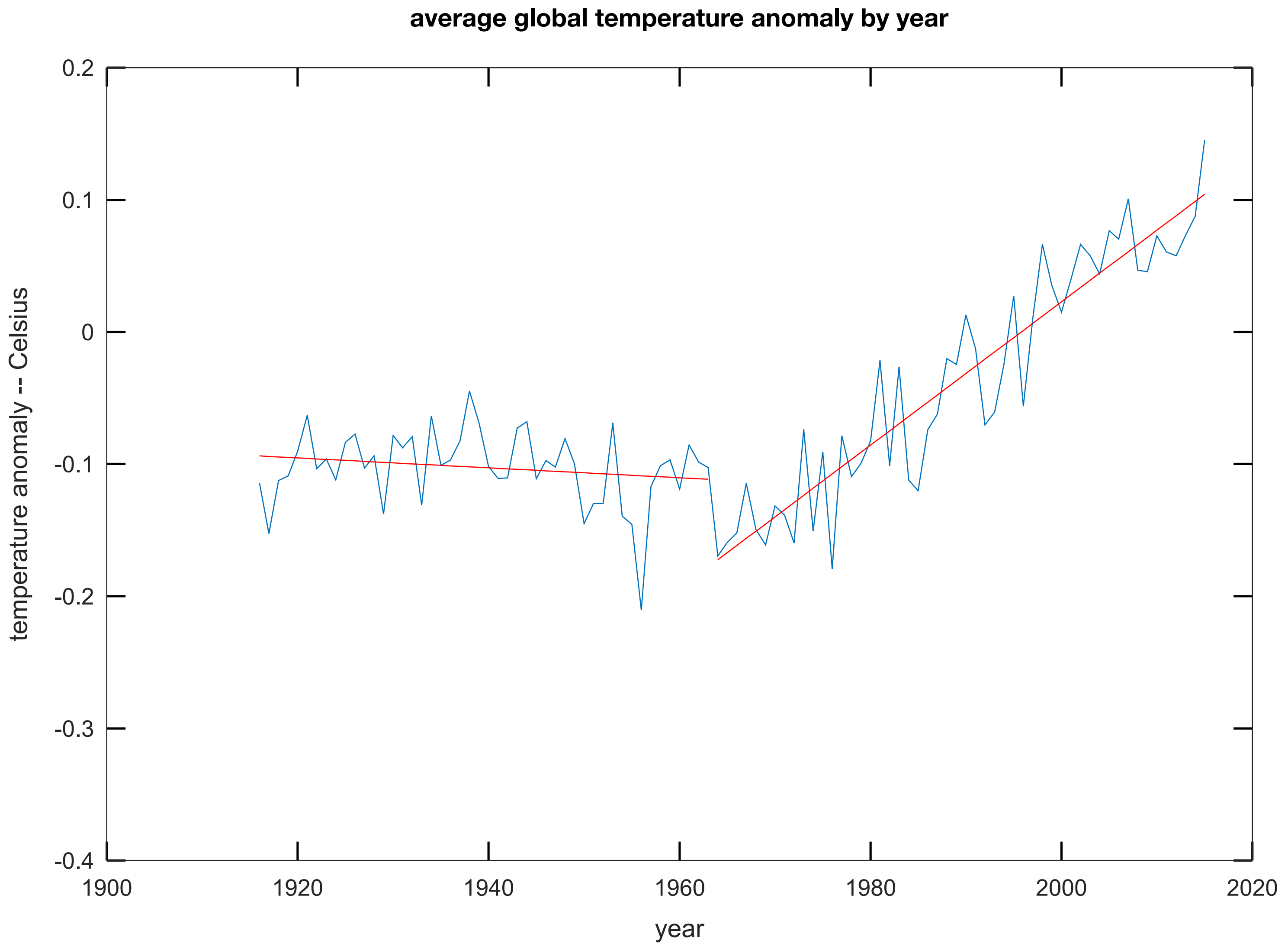
In [15]:

```
minimum_SSE = intmax;
correctY = 1916;

for Y = 1917:2013
    A1 = [(my_years(1,1:(Y-1915)))'];
    A1 = [A1, ones(size(A1), 1)];
    b1 = average_WORLD_anomaly_by_year(1:(Y-1915),1);
    A2 = [(my_years(1, (Y-1914):end))'];
    A2 = [A2, ones(size(A2), 1)];
    b2 = average_WORLD_anomaly_by_year((Y-1914):end,1);
   linear_model_up_to_Y = pinv(A1) * b1;
   linear_model_after_Y = pinv(A2) * b2;
   SSE_up_to_Y = norm( A1 * linear_model_up_to_Y - b1)^2;
   SSE_after_Y = norm( A2 * linear_model_after_Y - b2)^2;
   total_SSE = SSE_up_to_Y + SSE_after_Y; % SSE_up_to_Y + SSE_after_Y;
   if (total_SSE < minimum_SSE)
      minimum_SSE = total_SSE;
      correctY = Y;
   end
end

correctY
```

```
correctY =  1963
```

In [16]:

```
A1 = [(my_years(1,1:(correctY-1915)))'];
A1 = [A1, ones(size(A1),1)];
b1 = average_WORLD_anomaly_by_year(1:(correctY-1915),1);

A2 = [(my_years(1, (correctY-1914):end))' ones(numel((my_years(1, (correctY-1914
):end))'),1)];
A2 = [A2, ones(size(A2),1)];
b2 = average_WORLD_anomaly_by_year((correctY-1914):end,1);

linear_model_up_to_correctY = A1 \ b1;
linear_model_after_correctY = A2 \ b2;

%size(1916:correctY)
%size(A1 * linear_model_up_to_correctY)

%size(correctY:2015)
%size(A2 * linear_model_after_correctY)

plot( my_years, average_WORLD_anomaly_by_year );hold
plot(1916:correctY, A1 * linear_model_up_to_correctY,'r',(correctY+1):2015, A2 *
linear_model_after_correctY,'r')
xlabel('year')
ylabel('temperature anomaly -- Celsius')
title('average global temperature anomaly by year')
```



In [ ]:

```

```