Module 6, part I: Principal Component Analysis

BIOS 526

Reading

- Slides with additional examples: http://biostat.jhsph.edu/~jleek/teaching/2011/754/lecture13.pdf
- Chapter 1, Jolliffe, I. "Principal Component Analysis, Second Edition."
- Classic textbook: Chapter 8 in Mardia, K. and J. Kent and J. Bibby. *Multivariate Analysis*.
- A paper I really like, although it is more advanced than we cover in this course: Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61(3), 611-622.

Concepts

- Dimension Reduction
- EVD, SVD, and PCs
- Principal component regression for correlated variables.
- Regression for $p > n$.

Acknowlegments

- https://www.cs.toronto.edu/~urtasun/courses/CSC411/tutorial8.pdf
- https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/tut7_handout.pdf

PCA Introduction

Big goal: find meaning from high dimensional data.

In other words, find a low dimensional representation.

Suppose we have N measurements on each of p variables $\mathbf{X}_j,\ j = 1, \ldots, p$.

Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_p) \in \mathbb{R}^{N \times p}$.

There are several equivalent approaches to principal components:

- Generate a small set of uncorrelated variables:

  capturing most of the information in the original set.
- Approximate $\mathbf{X}$ by the best rank-q matrix $\hat{\mathbf{X}}_{(q)}$. This is the usual motivation for the Singular Value Decomposition (SVD).
- Approximate the original set of N points in $\mathbb{R}^p$ by a sequence of best linear approximations to the data.

PCA in Practice

Applications include

- Dimension reduction
- Factor analysis
- Data compression and reconstruction
- Data visualization
- A way to address multicollinearity via PCA regression
- A way to address $p > N$

Key concept: when your variables are correlated, there exists a lower dimensional representation capturing most of the information.

PCA in two dimensions

$\mathbf{Z}_1 = \mathbf{X}\mathbf{u}_1$ is the projection of the data onto the longest direction, and has the largest variance among projections.

It is the first principal component.

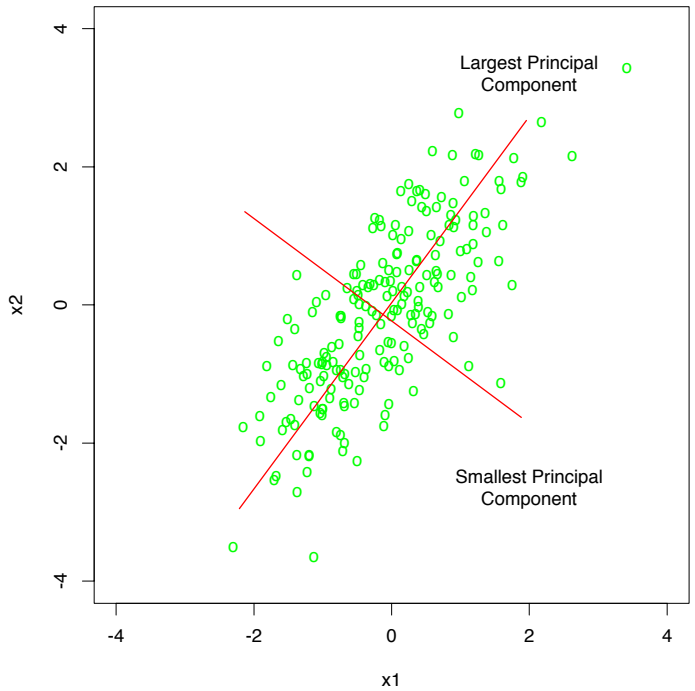


Figure: From Rob Tibshirani Lecture Notes, http://statweb.stanford.edu/~tibs/stat315a/Supplements/.

PCA in three dimensions

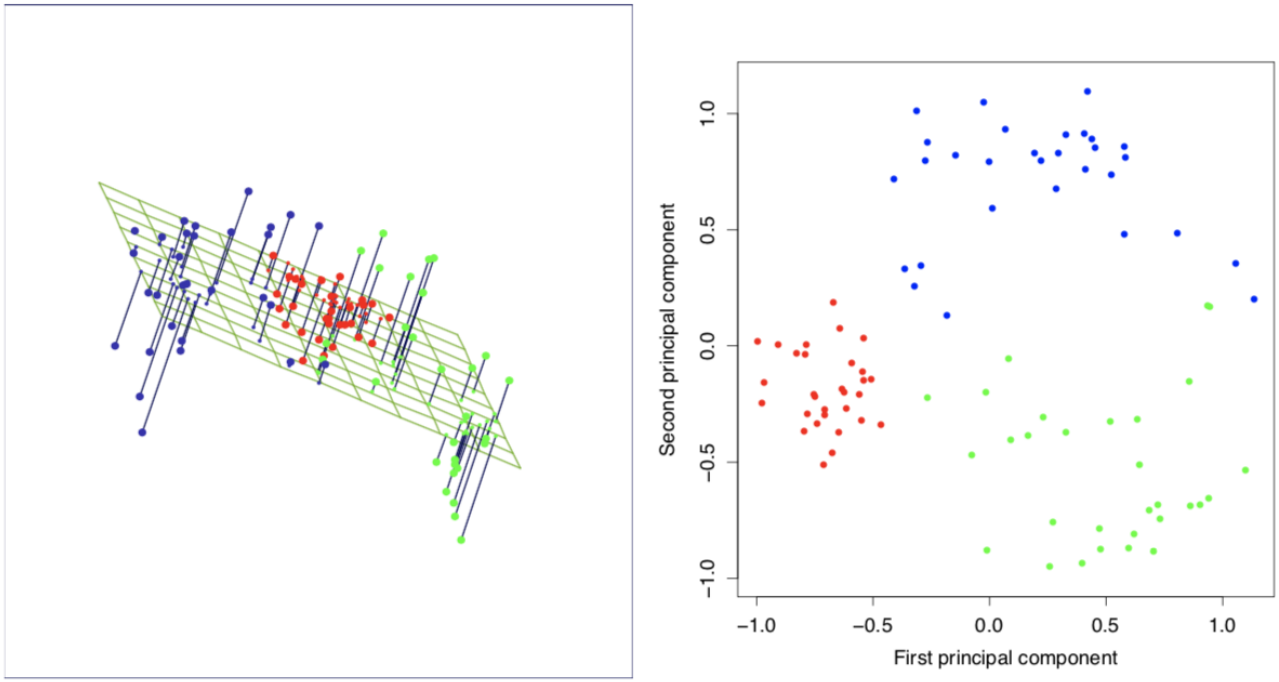


FIGURE 14.21. *The best rank-two linear approximation to the half-sphere data. The right panel shows the projected points with coordinates given by* $\mathbf{U}_2\mathbf{D}_2$*, the first two principal components of the data.*

Figure: Figure 14.21 from ESL II. The left displays the projection from $\mathbb{R}^3$ to $\mathbb{R}^2$. The right shows the points projected on the principal axes $\mathbf{u}_1$ and $\mathbf{u}_2$, which are the subject scores (z_{i1}, z_{i2}). Note: notation differs; ignore notation in screenshot image.

OLS with $\mathbf{X} \in \mathbb{R}^2$

PCA is an unsupervised learning problem.

For example: we have data in different categories and covariates; PCA finds "clusters" without knowing the categories. (Wine Example in R Code.)

In contrast, OLS is a type of supervised learning problem, since we are trying to predict $\mathbf{Y}$. Contrast the picture below with the previous slide:

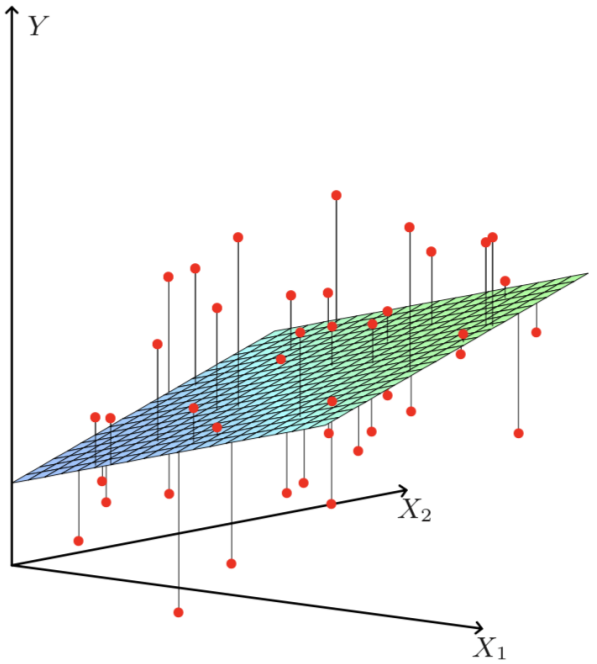


Figure: Figure 3.1 in ESL II.

Optimization Problem

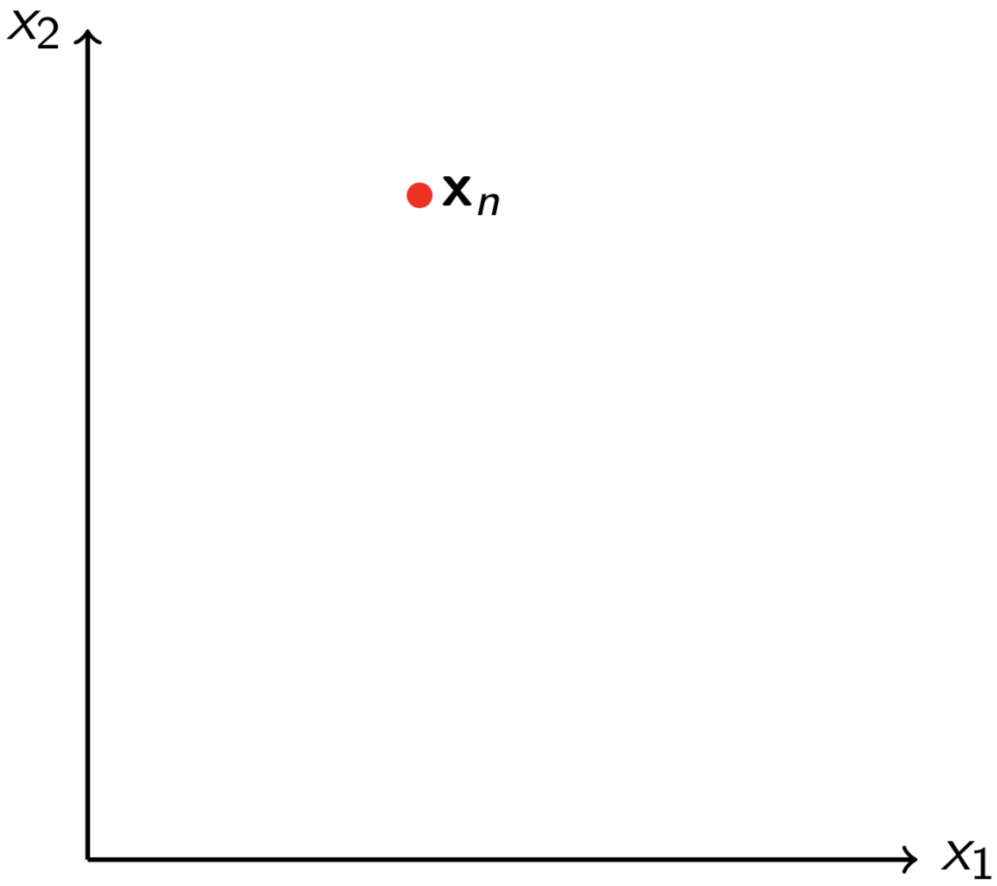
In PCA, we will find a sequence of directions that successively maximize the variance explained in the data, given the previous directions.

Let's first find the rank-1 projection of the data that captures the most variance:

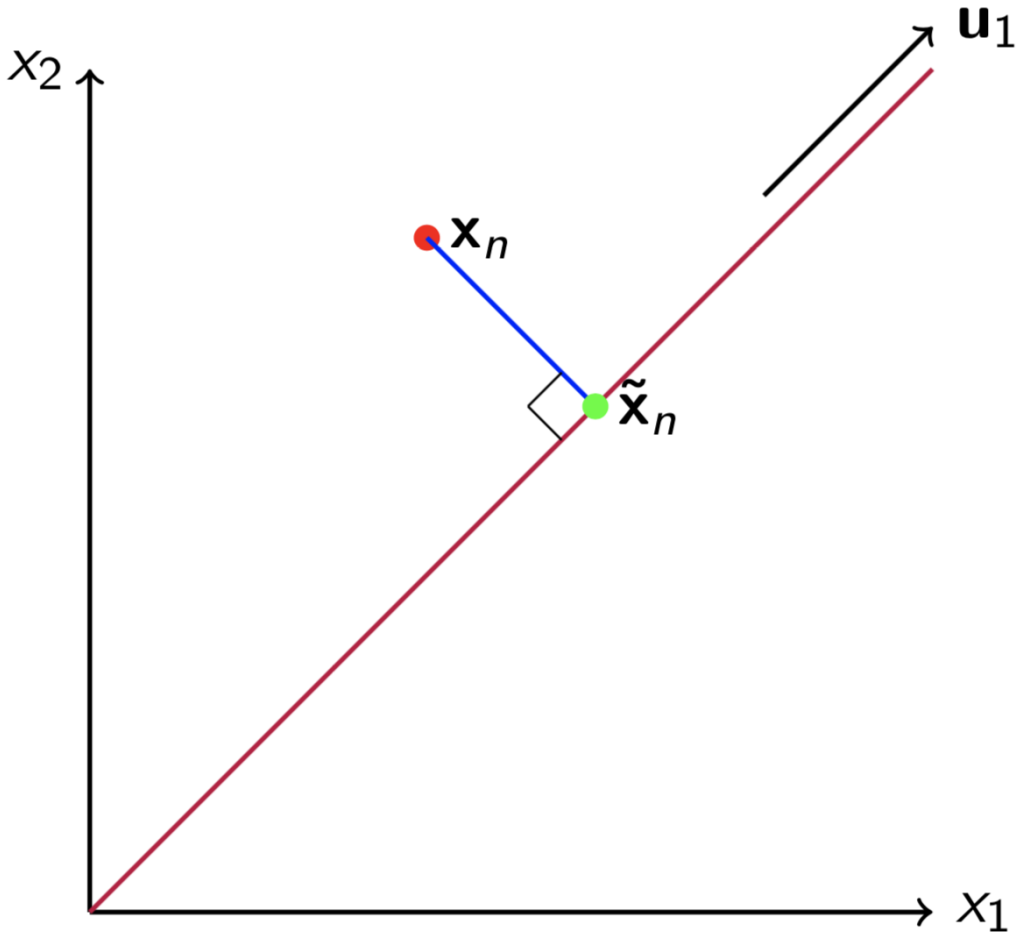
Note the constraint is necessary for this to be finite. In words, we are finding the direction that maximizes variance.

Note how PCA treats data symmetrically, which contrasts with OLS which for $\mathbf{x}_i \in \mathbb{R}^2$ tries to predict x_{i2}.

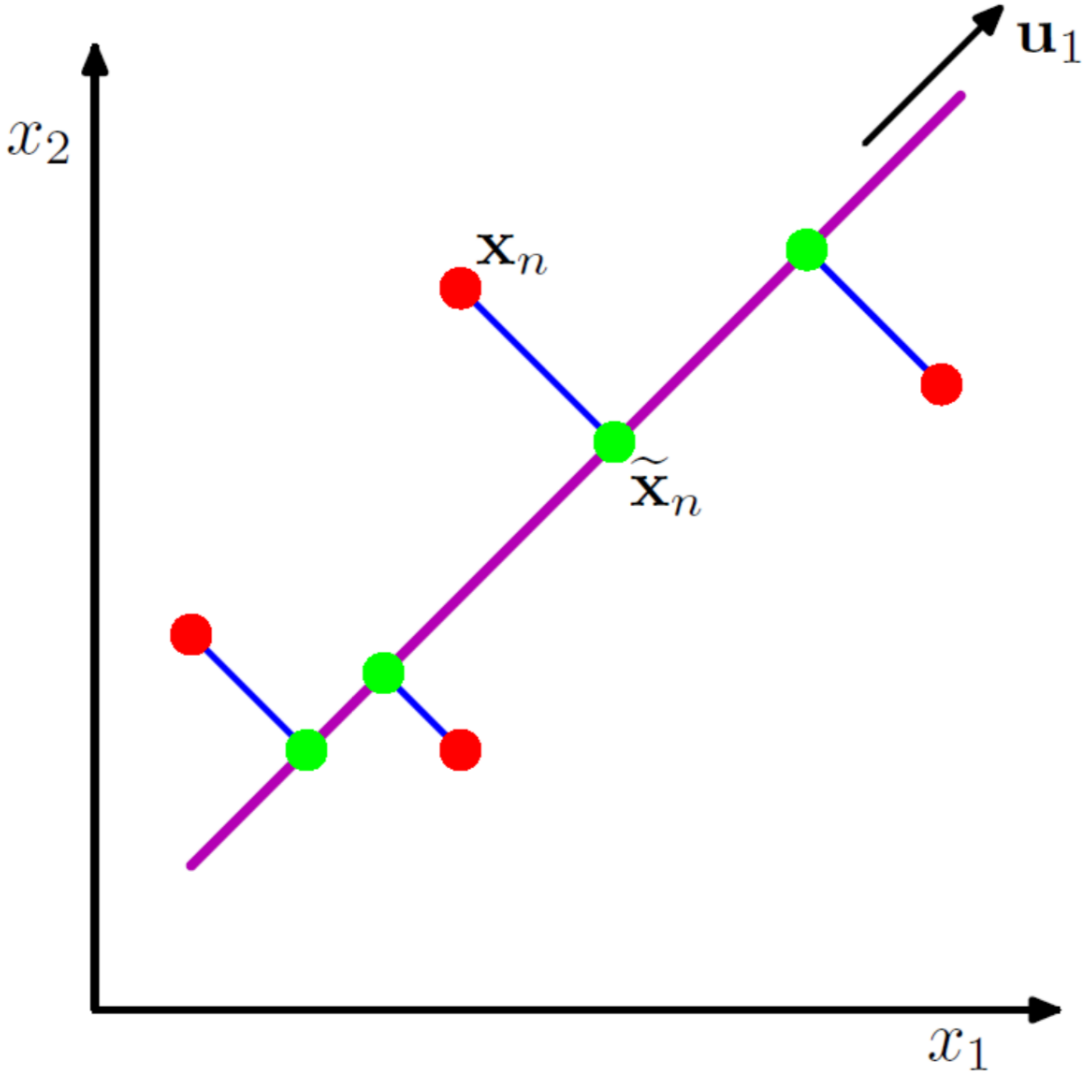
Projecting a data point



Projecting a data point



Projecting a data point



PC 1

$$\underset{\mathbf{u}_1 \in \mathbb{R}^p : \mathbf{u}_1' \mathbf{u}_1 = 1}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{u}_1'(\mathbf{x}_i - \bar{\mathbf{x}}))^2$$

Let $\mathbf{S} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})'$. The previous equation can be written as

We can solve this objective function using the method of Lagrange Multipliers.

We reformulate the constraint in the objective function such that the argmax of the constrained obj fun will be a stationary point:

PC 1

Finding the partial derivative and setting it equal to zero:

$$J(\mathbf{u}_1) = \mathbf{u}_1'\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1'\mathbf{u}_1)$$

$$\frac{\partial J}{\partial \mathbf{u}_1} =$$

Then we have

PC 1, PC 2

Hence, $\mathbf{u}_1$ is an eigenvector of $\mathbf{S}$, and its eigenvalue is λ_1.

Clearly, the objective function is maximized at λ_1 corresponding to the largest eigenvalue.

Assume $p > 2$. Then we next want to find $\mathbf{u}_2$ orthogonal to $\mathbf{u}_1$ capturing the most information:

$$\underset{\mathbf{u}_2 \in \mathbb{R}^p}{\operatorname{argmax}} \quad \mathbf{u}_2' \mathbf{S} \mathbf{u}_2$$

subject to

$$\mathbf{u}_2' \mathbf{u}_2 = 1$$
$$\mathbf{u}_2' \mathbf{u}_1 = 0.$$

PC 2

Again using the method of Lagrange Multipliers:

$$J(\mathbf{u}_2) = \mathbf{u}_2'\mathbf{S}\mathbf{u}_2 + \lambda_2(1 - \mathbf{u}_2'\mathbf{u}_2) + \beta\mathbf{u}_2'\mathbf{u}_1$$
$$\frac{\partial J}{\partial \mathbf{u}_2} = 2\mathbf{S}\mathbf{u}_2 - \lambda_2 2\mathbf{u}_2 + \beta\mathbf{u}_1 = 0.$$

Let's try to solve for β. We have

$$\begin{aligned} 2\mathbf{u}_1'\mathbf{S}\mathbf{u}_2 - \lambda_2 2\mathbf{u}_1'\mathbf{u}_2 + \beta\mathbf{u}_1'\mathbf{u}_1 = 0 \\ \implies 2\lambda_1\mathbf{u}_1'\mathbf{u}_2 - 0 + \beta = 0 \\ \implies \beta = 0. \end{aligned}$$

PC 2

So the Lagrangian equation simplifies to

$$J(\mathbf{u}_2) = \mathbf{u}_2'\mathbf{S}\mathbf{u}_2 + \lambda_2(1 - \mathbf{u}_2'\mathbf{u}_2)$$

And as before

$$\mathbf{S}\mathbf{u}_2 = \lambda_2\mathbf{u}_2.$$

Hence, λ_2 is the second largest eigenvalue and $\mathbf{u}_2$ its eigenvector.

Note in $\mathbb{R}^2$, $\mathbf{u}_2$ is just going to be the unique direction that is orthogonal to $\mathbf{u}_1$, which is the projection the minimizes the variance.

The Eigenvalue Decomposition

Any positive definite symmetric matrix can be decomposed using the eigenvalue decomposition. This is also called the spectral decomposition.

- $\mathbf{\Lambda}$ is a diagonal matrix with elements $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_p > 0$.
- The last eigenvalue is strictly greater than 0 when the matrix is positive definite ($\mathbf{x}'\mathbf{A}\mathbf{x} > 0 \; \forall \; \mathbf{x} \in \mathbb{R}^p$.
- Intuitively, positive definite matrices generalize the notion of a positive integer to matrices.
- $\mathbf{U}'\mathbf{U} = \mathbf{I}$, i.e., $\mathbf{U}$ is an orthogonal matrix (orthogonal rows, unit norm).

EVD

This decomposition provides us with the PC directions and their variances.

Let $\mathbf{X}_c$ be $\mathbf{X}$ with the rows centered (i.e., for each variable, subtract its mean).

Then $\mathbf{S} = \frac{1}{N}\mathbf{X}_c\mathbf{X}_c'$.

We can take the EVD of the sample covariance matrix, then project the data onto the principal subspace.

Using the decomposition $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$, let

$$\mathbf{U}_{(q)} = [\mathbf{u}_1, \ldots, \mathbf{u}_q]$$

be the first q PC directions.

Letting $\mathbf{U}_{(q)}$ be the first q PC directions, and $\mathbf{U}_{(-q)}$ be the $q+1$ to p directions containing the remaining variance:

Then $\mathbf{X}_c\mathbf{U}_{(q)}\mathbf{U}'_{(q)}$ is the rank-q projection of $\mathbf{X}$ with the highest variance among the class of all linear projections.

Define . These are the principal component *scores*, which is a subspace of $\mathbb{R}^N$.

For the ith subject, assuming $\mathbf{x}_i$ is centered, $\mathbf{x}_i \in \mathbb{R}^p$, and the kth component, $z_{ik} = \mathbf{u}'_k\mathbf{x}_i$.

In this projection, weights the variables. These weights are called the *loadings* and form a subspace of $\mathbb{R}^p$.

Think subject scores and variable loadings.

The terminology is confusing, particularly because different authors arrange the data differently, such that the data matrix may be $p \times n$ ...

PC Variances

Note that

Thus, the eigenvalues are the variances of the principal component scores. (Note some treatments constrain the scores to have unit variance in which case the loadings are related to the variance.)

SVD

The singular value decomposition (SVD) is closely related:

$$\mathbf{X}_c = \mathbf{V}\mathbf{D}\mathbf{U}',$$

where $\mathbf{V}$ are the left singular vectors (orthonormal), $\mathbf{D}$ the singular values, and $\mathbf{U}'$ the right singular vectors. Note

So the right singular vectors of the SVD equal the eigenvectors from the covariance decomposition!

When $p \gg n$, we don't need to calculate the $p \times p$ covariance matrix. This is an important computational consideration for vbdata.

SVD and PCs

We have

$$\mathbf{Z} = \mathbf{X}_c \mathbf{U}_{(q)}$$

and

$$\mathbf{X}_c = (\mathbf{V}\mathbf{D})\mathbf{U}'$$

and it follows that $\mathbf{Z} = \mathbf{V}_{(q)}\mathbf{D}_{(q)}$.

In words, the first q principal component scores are the first q left singular vectors scaled by their singular values.

Maximize variance and minimize reconstruction

We can also think of PCA as minimizing residual error. For notational simplicity, let $\mathbf{x}_i$ be centered. For $q = 1$:

It turns out the solution is the same as we previously derived!

This formulation is the motivation for PCA as a data compression technique.

Equivalently, for rank q decomposition, we minimize the variance on the $p - q$ remaining directions, $\mathbf{X}\mathbf{U}_{(-q)}\mathbf{U}'_{(-q)}$

What about statistics? PCA and factor analysis

The discussion so far has not really involved statistics.

We can formulate a population PCA model, as in probabilistic PCA (PPCA) by Tipping and Bishop (1999).

Let $\mathbf{z}_i$ be latent variables (random vector), where $\mathbf{z}_i \in \mathbb{R}^q$ for $q < p$. Let $\mathbf{W} \in \mathbb{R}^{p \times q}$ be a mixing matrix (fixed). Let $\boldsymbol{\epsilon}_i$ be measurement error, with $\boldsymbol{\epsilon}_i \perp\!\!\!\perp \mathbf{z}_i$. We assume isotropic noise.

This is a factor analysis model.

We need additional assumptions for identifiability, which we won't address here.

This model allows us to gain insight into the covariance structure of $\mathbf{X}$.

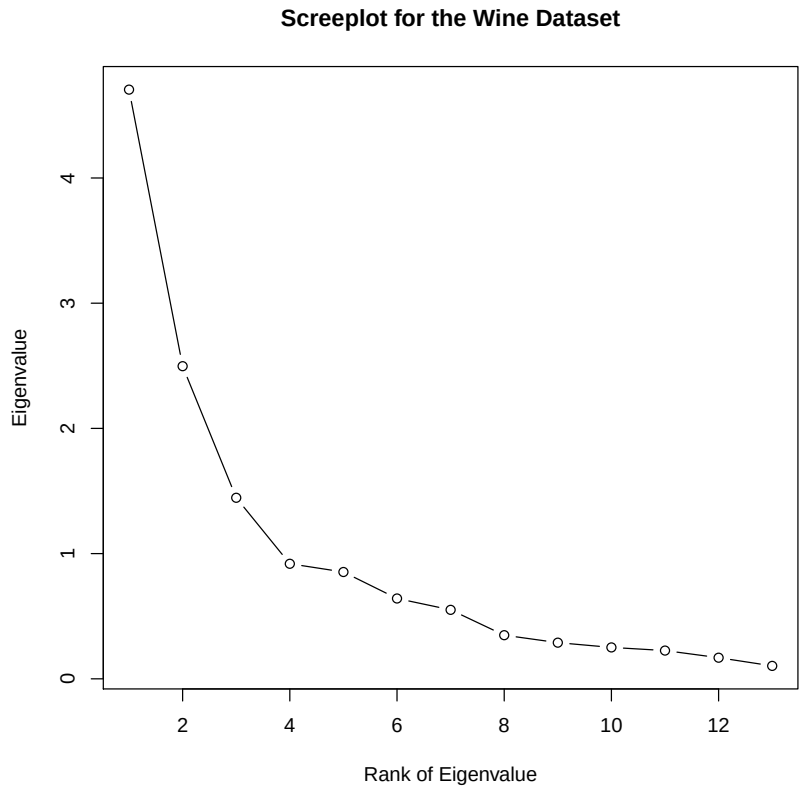
Let $\mathbf{W} = \mathbf{U}$ from the EVD, and assume that the columns of $\mathbf{Z}$ are orthogonal.

Let $\mathbf{\Gamma} = \mathrm{diag}(\gamma_1, \ldots, \gamma_q)$ and $\mathbf{\Gamma}_+$ be the $p \times p$ matrix composed of $\mathbf{\Gamma}$ padded with zeros.

From this representation, we see that the largest eigenvalues / eigenvectors are associated with the latent factors because their variance is $\gamma_k + \sigma^2$ for $k \leq q$, whereas the pure-noise directions include σ^2 only.

Scree plot

This motivates a scree plot:



We typically look for an "elbow," very roughly where the eigenvalues go from exponential decay to a roughly linear trend, where a linear trend results in an isotropic noise model from the *sorted* sample estimates of the variance of the noise directions.

Scree

PCA In Practice

- The singular value decomposition is sensitive to scaling.
- We usually center and standardize the variables in a dataset $\mathbf{X}$ prior to conducting PCA.
- Otherwise, PCA is sensitive to scaling.
  - E.g., if you measured one variable in millimeters and another in kilometers, the first variable would dominate the decomposition because the numbers are much larger and hence has larger variance.
- In other words, we decompose the correlation matrix.

Wine data set

Here we perform dimension reduction on a data set with 178 wine samples.

13 variables measuring physicochemical properties:

- alcohol, malic acid, ash, alkalinity, magnesium, phenols, flavanoids, non-flavinoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, proline

We also have a variable "cultivar," or cultivated variety. You can think of it as group labels.

We will pretend we do not have this variable, and see whether or not we can visualize patterns in the multivariate data from the 13 variables in a lower dimensional space.

Many variables are highly correlated – PCA can find a smaller number of orthogonal variables that capture most of the variance.

Wine data

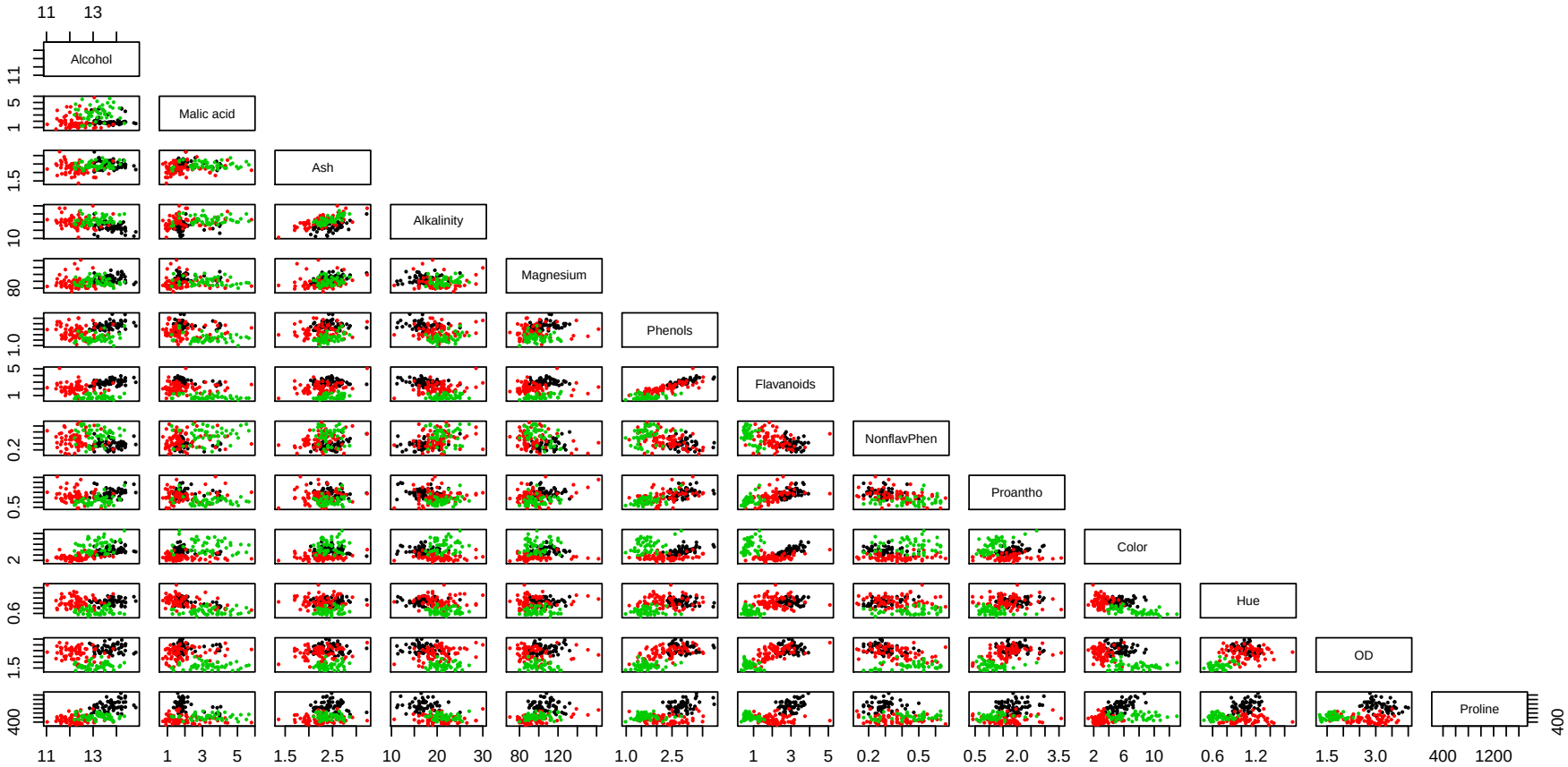
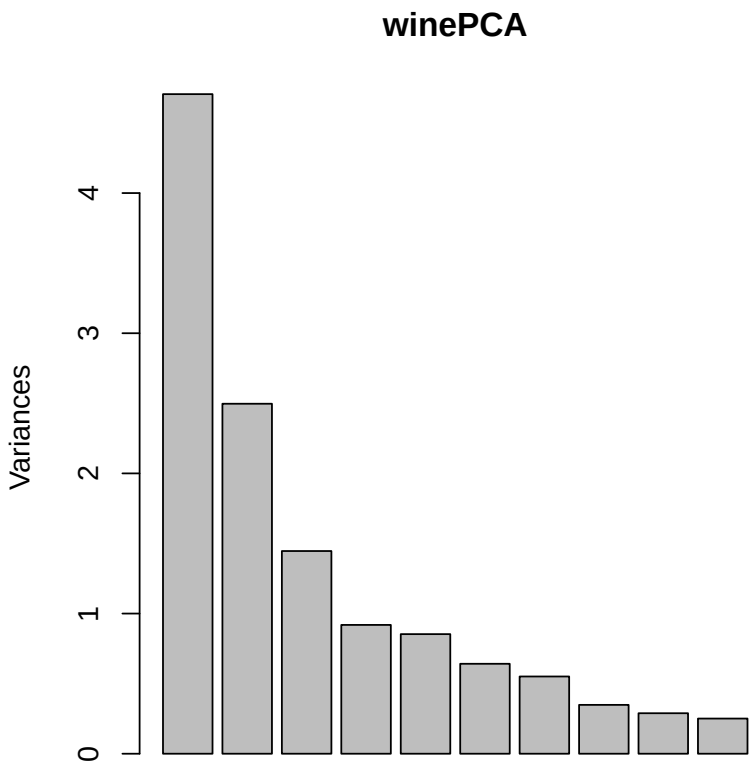


Figure: Scatter plots of the 13 variables in the wine dataset. Points are colored by cultivar.

Screeplot

How many components?

```
winePCA <- prcomp(wine[,-1],center = TRUE, scale. = TRUE)
screeplot(winePCA)
> # here, 3 would be a good option since ith corresponds to the "elbow"
> cumsum(winePCA$sdev^2)/sum(winePCA$sdev^2)
 [1] 0.3619885 0.5540634 0.6652997 0.7359900 0.8016229 0.8509812 0.8933680 0.92
[11] 0.9790655 0.9920479 1.0000000
```



prcomp

```
   > summary(winePCA)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     2.169 1.5802 1.2025 0.95863 0.92370 0.80103
Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936
Cumulative Proportion  0.362 0.5541 0.6653 0.73599 0.80162 0.85098
                           PC7     PC8     PC9   PC10    PC11
Standard deviation     0.74231 0.59034 0.53748 0.5009 0.47517
Proportion of Variance 0.04239 0.02681 0.02222 0.0193 0.01737
Cumulative Proportion  0.89337 0.92018 0.94240 0.9617 0.97907
                          PC12    PC13
Standard deviation     0.41082 0.32152
Proportion of Variance 0.01298 0.00795
Cumulative Proportion  0.99205 1.00000
```

PCA Scores

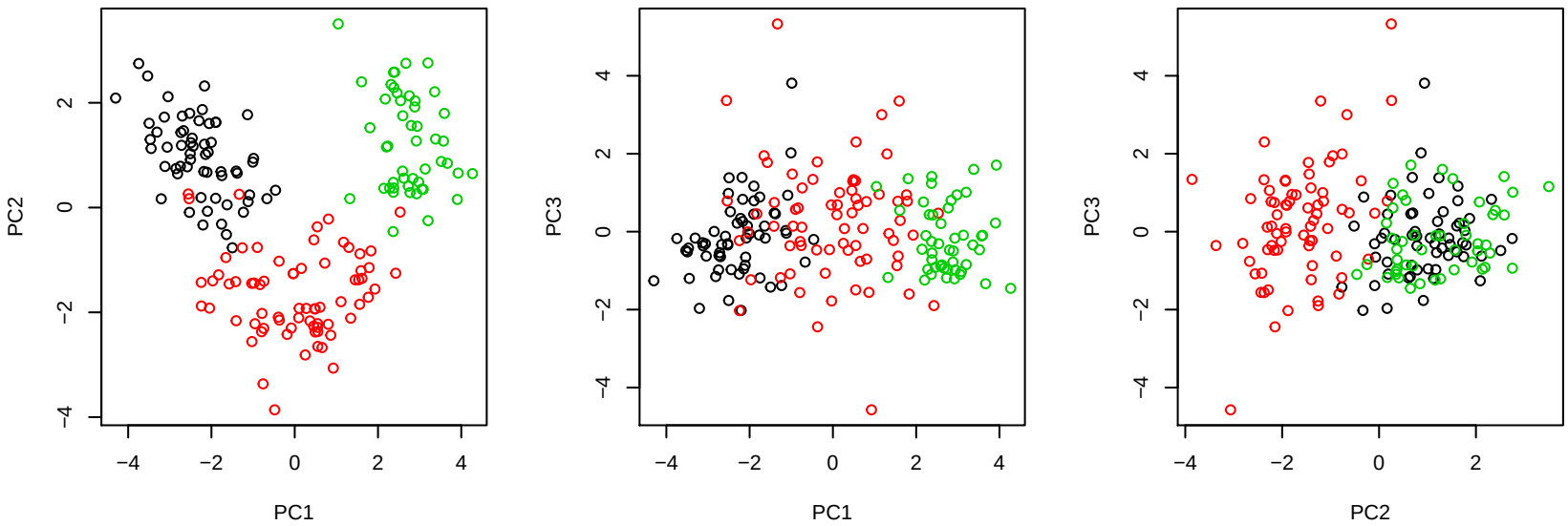
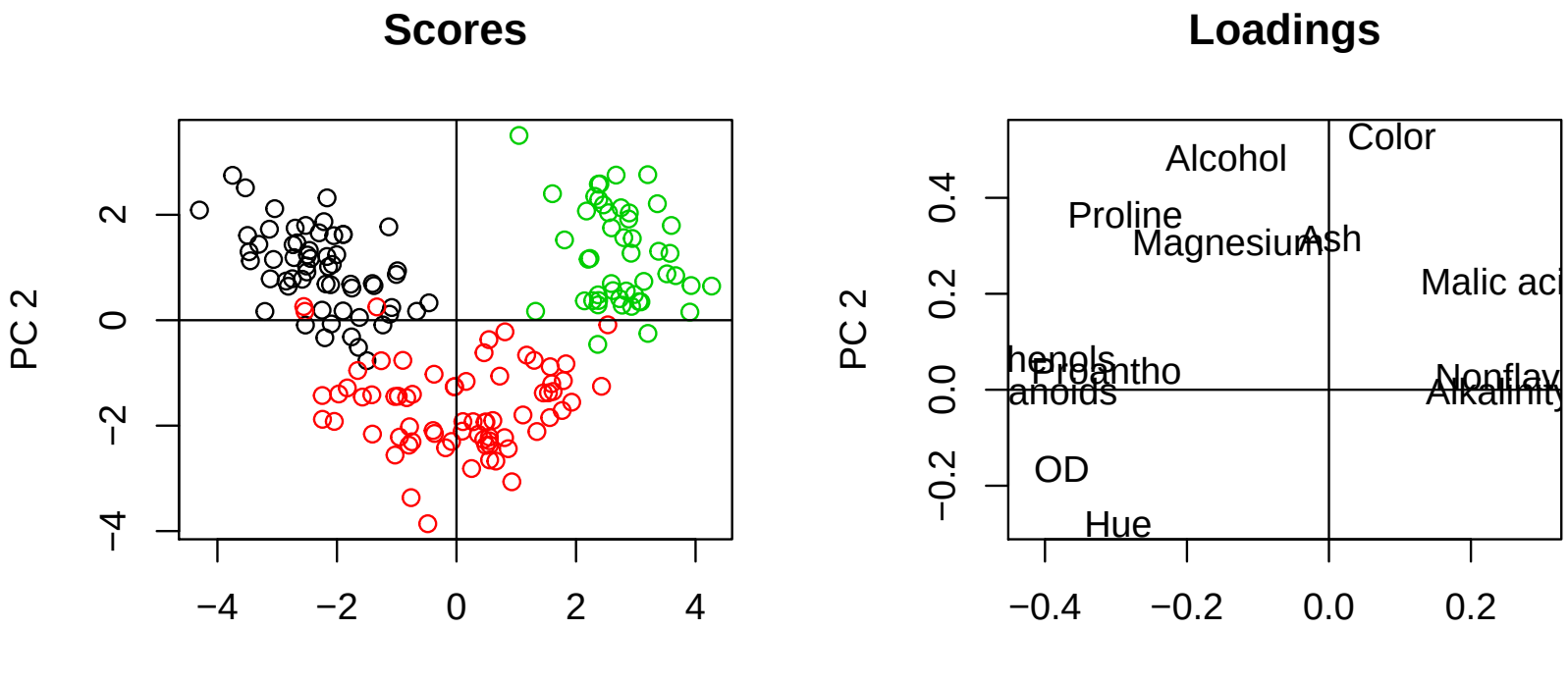


Figure: Wine data represented with 3 PCs. Each PC has 178 scores.

Another PCA package

```
  BiocManager::install("pcaMethods")
library(pcaMethods)
winePCAmethods <- pca(wine[,-1], scale = "uv", center = T,
nPcs = 3, method = "svd")
slplot(winePCAmethods, pcs=c(1,2),scoresLoadings = c(T,T),
scol = wineClasses,hotelling=FALSE)
```



PCA Regression

PCA regression involves estimating the first q principal components, and then using these components (subject scores) as predictors in linear regression:

By estimating $q < p$ coefficients, we reduce overfitting.

Moreover, the predictors are orthogonal, so we obtain more precise estimates of their standard errors than the original $\mathbf{x}_i$.

However, they are more difficult to interpret since each component is a linear combination of the other components:

Notes on PCA Regression

When $q < p$, PCA regression assumes that the directions that contain the most variance in $\mathbf{x}_i$ are most closely associated with the response y_i.

This may not be true, as a low variance direction could be associated with y_i.

Namely, the dimension reduction step does not see y_i, and hence this method may not be optimal.

However, it is often a very useful approximation.

Other methods like partial least squares (PLS) regression use y_i in the dimension reduction, but often do not lead to substantive improvements.

Note that when $p > n$, the usual OLS estimate is undefined. More on this in M6, part II.

For $p > n$, one approach is to choose some $q < n < p$, and then use PCR.

Baseball example

Based on example 6.7, Lab 3, James et al ISL.

MLB stats in 1986.
Salary: salary in 1987.

```
 > # this library contains datasets using in ISL:
> library(ISLR)
> # this library contains some short-cut functions for PCR:
> library(pls)
> Hitters = na.omit(Hitters)
> pcr.fit = pcr(Salary~., data=Hitters, scale=TRUE)
> summary(pcr.fit)
Data:  X dimension: 263 19
Y dimension: 263 1
Fit method: svdpc
Number of components considered: 19
TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X         38.31    60.16    70.84    79.03    84.29    88.63    92.26
Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69
        8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
X         94.96    96.28     97.26     97.98     98.65     99.15
Salary    46.75    46.86     47.76     47.82     47.85     48.10
        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
X          99.47     99.75     99.89     99.97     99.99    100.00
```

Cross validation in PC Regression

```
> validationplot(pcr.fit,estimate='all',legendpos='topright')
```

PC Regression, cont.

In this example, cross validation is not very helpful.

Minimized at 16, but not a clear minimum.

It is interesting to note that 1 component does fairly well.

Let's take a detailed look at what is happening in the regression.

```
  > # create a matrix that can be used by prcomp:
> hitters = model.matrix(Salary~.,data=Hitters)
> pca.hitters = prcomp(x=hitters[,-1],center = TRUE, scale. = TRUE)
> tempdata = data.frame('Salary'=Hitters$Salary,pca.hitters$x[,1:16])
> pcr.fit.v3 = lm(Salary~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8
+PC9+PC10+PC11+PC12+PC13+PC14+PC15+PC16,data=tempdata)
> vif(pcr.fit.v3)
 PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10 PC11 PC12 PC13 PC14 PC15 PC16
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
```

PCA Examples

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	535.926	19.678	27.235	< 2e-16	***
PC1	106.571	7.307	14.584	< 2e-16	***
PC2	21.645	9.678	2.236	0.026220	*
PC3	-24.341	13.836	-1.759	0.079787	.
PC4	-37.056	15.802	-2.345	0.019823	*
PC5	58.525	19.729	2.966	0.003309	**
PC6	-62.325	21.700	-2.872	0.004433	**
PC7	-24.686	23.746	-1.040	0.299562	
PC8	15.858	27.526	0.576	0.565054	
PC9	29.633	39.373	0.753	0.452398	
PC10	99.838	45.860	2.177	0.030432	*
PC11	-30.170	53.218	-0.567	0.571298	
PC12	-21.033	55.219	-0.381	0.703608	
PC13	-72.540	63.769	-1.138	0.256418	
PC14	-277.213	79.802	-3.474	0.000606	***
PC15	74.312	86.478	0.859	0.391001	
PC16	-423.532	117.811	-3.595	0.000392	***

PC Regression, loadings

PC1 is highly significant, so let's take a look to see what variables are driving PC1.

These are the PC loadings:

```
> pca.hitters$rotation[,1]
       AtBat         Hits        HmRun         Runs
 0.1982903511  0.1958612933  0.2043689229  0.1983370917
         RBI        Walks        Years       CAtBat
 0.2351738026  0.2089237517  0.2825754503  0.3304629263
       CHits       CHmRun        CRuns         CRBI
 0.3307416802  0.3189794925  0.3382078595  0.3403428387
      CWalks      LeagueN    DivisionW      PutOuts
 0.3168029362 -0.0544708722 -0.0257252900  0.0776971752
     Assists       Errors    NewLeagueN
-0.0008416413 -0.0078593695 -0.0419103083
```

We see that career nubers tend to have larger loadings than 1986 numbers.

Compare to OLS

```
> ols.fit = lm(Salary~.,data=Hitters)
> car::vif(ols.fit)
     AtBat       Hits      HmRun       Runs        RBI      Walks
 22.944366  30.281255   7.758668  15.246418  11.921715   4.148712
     Years     CAtBat      CHits     CHmRun      CRuns       CRBI
  9.313280 251.561160 502.954289  46.488462 162.520810 131.965858
    CWalks     League   Division    PutOuts    Assists     Errors
 19.744105   4.134115   1.075398   1.236317   2.709341   2.214543
 NewLeague
  4.099063
```

Compare to OLS, cont.

```
 Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  163.10359   90.77854   1.797 0.073622 .
AtBat         -1.97987    0.63398  -3.123 0.002008 **
Hits           7.50077    2.37753   3.155 0.001808 **
HmRun          4.33088    6.20145   0.698 0.485616
Runs          -2.37621    2.98076  -0.797 0.426122
RBI           -1.04496    2.60088  -0.402 0.688204
Walks          6.23129    1.82850   3.408 0.000766 ***
Years         -3.48905   12.41219  -0.281 0.778874
CAtBat        -0.17134    0.13524  -1.267 0.206380
CHits          0.13399    0.67455   0.199 0.842713
CHmRun        -0.17286    1.61724  -0.107 0.914967
CRuns          1.45430    0.75046   1.938 0.053795 .
CRBI           0.80771    0.69262   1.166 0.244691
CWalks        -0.81157    0.32808  -2.474 0.014057 *
LeagueN       62.59942   79.26140   0.790 0.430424
DivisionW   -116.84925   40.36695  -2.895 0.004141 **
PutOuts        0.28189    0.07744   3.640 0.000333 ***
Assists        0.37107    0.22120   1.678 0.094723 .
Errors        -3.36076    4.39163  -0.765 0.444857
NewLeagueN   -24.76233   79.00263  -0.313 0.754218
```

PC Regression versus OLS

Multicollinearity a huge issue in OLS. In particular, CAtBat, CHits, CHmRun, CRuns and CRBI have VIFs>40.

In OLS, these variables are not significant. Obviously we should not trust OLS due to VIFs.

In PC Regression, the first component is highly significant, and the largest loadings in absolute value are on CRBI, CRuns, CHits, CAtBats, CHmRun, and CWalks. This indicates these career numbers (up to 1986) are important drivers of salary in 1987.