Data Reduction: Principal Components and Factor Analysis

Principal Components

1. A way to reduce a large number of collinear variables to a few summary variables that contain "most of the information".
2. Can be used as a "transformation and variable selection" method for the X's in linear regression, logistic regression, etc.
3. Can be used as a way of "learning the structure" of the X's even if we won't use the results in a regression.
4. If we have several possible response variables (y's), can be used to "extract" a best summary response y^* from them.

We will illustrate 1, 2, 3 below...

- *Goal:*
  - Find the linear combination of X's with *the biggest variance*. This is the *First Principal Component (sometimes called "first factor" or F_1).*
  - Find the linear combination of X's *uncorrelated with* F_1 that has the *next biggest variance*. This is the *Second Principal Component (sometimes called "second factor" or F_2).*
  - Etc. etc.
- *Notes:*
  - The usual rule is not to work with the raw X's but rather to work with standardized variables: $z_1 = (x_1 - \overline{x}_1)/s_1$, $z_2 = (x_2 - \overline{x}_2)/s_2, \ldots$
  - We don't want to learn about the means of the X's, only the variability of the X's, so centering each X_j by $\overline{X}_j$ makes sense.
  - If we do not divide each X_j by sd_j, then F_1 will mostly be the original X with the largest variance, F_2, will mostly be the original X with the next largest variance, etc. These will give no insight about what groups of variables "go together", so we divide each x_j by its s_j as wll.
  - In cases where the X_k's are basically all measured on the same scale, dividing by $s_j = SD(X_k)$ is usually omitted.

Digression: A Derivation of Principal Components

Recall that the principal problem with collinearity in the regression problem

$$\underline{y} = X\beta + \epsilon \tag{1}$$

is a technical one: The least-squares estimate of β is

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

but if X has nearly collinear columns this is difficult to compute. Our goal is to transform X in some way, to make variable selection easier, so we can get rid of the collinearity problem.

To make the problem easier to think about let us imagine that X has a column of 1's, to fit an intercept, and let us write

$$X = [\ \mathbf{1}\ \ \mathbf{x}_1\ \ \cdots\ \ \mathbf{x}_k\ ]$$

where $\mathbf{1}$ is the intercept column and the columns $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are the independent variables of interest. We define

$$Z = [\ \mathbf{z}_1\ \ \cdots\ \ \mathbf{z}_k\ ] = [\ (\mathbf{x}_1 - \overline{x}_1)/s_1\ \ \cdots\ \ (\mathbf{x}_k - \overline{x}_k)/s_k\ ]$$

where $\overline{x}_j$ is the j^{th} column mean and s_j^2 is the sample variance of the j^{th} column. It is fairly easy to show that

$$\underline{y} = \mathbf{1}\beta_0 + Z\alpha + \epsilon \tag{2}$$

is the same regression problem as Equation (1), and

$$\begin{aligned} \hat{\alpha} &= (Z^T Z) Z^T \underline{y} \\ \text{Var}(\hat{\alpha}) &= \sigma_\epsilon^2 (Z^T Z)^{-1} \end{aligned}$$

There will be collinearity in X if and only if there is collinearity in Z, and Z has the pleasant property that

$$R_{XX} = \frac{1}{n-1} \cdot Z^T Z$$

is the matrix of sample correlations of the $\mathbf{x}_j$'s (with 1's down the main diagonal).

Note that R_{XX} is the first place we would look for collinearity among the X's: X's that are highly correlated will automatically be collinear.

We will construct an *orthogonal basis*

$$F = [\ \mathbf{F}_1\ \mathbf{F}_2\ \ \cdots\ \ \mathbf{F}_k\ ]$$

for the vector space spanned by the columns of

$$Z = [\ \mathbf{z}_1\ \mathbf{z}_2\ \ \cdots\ \ \mathbf{z}_k\ ]$$

There are many ways to proceed with this:

- Gram-Schmidt-like process
- Singular-value decomposition of Z
- Constrained optimization (leads to eigenvalues and eigenvectors of $R_{XX} = Z^T Z$)

We will do the constrained optimization.

We want to find

$$\begin{aligned}
\mathbf{F}_1 &= a_{11}\mathbf{z}_1 + a_{21}\mathbf{z}_2 + \cdots + a_{k1}\mathbf{z}_k \\
\mathbf{F}_2 &= a_{12}\mathbf{z}_1 + a_{22}\mathbf{z}_2 + \cdots + a_{k2}\mathbf{z}_k \\
\vdots &= \vdots \\
\mathbf{F}_k &= a_{1k}\mathbf{z}_1 + a_{2k}\mathbf{z}_2 + \cdots + a_{kk}\mathbf{z}_k
\end{aligned}$$

i.e.

$$\mathbf{F}_j = Z\underline{a}_j, \;\; \underline{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{kj} \end{bmatrix}$$

or

$$F = ZA, \;\; A = [\; \underline{a}_1 \; \underline{a}_2 \;\; \cdots \;\; \underline{a}_k \;]$$

with the following constraints:

- $\mathbf{F}_1$ should have maximum variance
- $\mathbf{F}_2$ should be uncorrelated (orthogonoal) to $\mathbf{F}_1$ and have as large a variance as possible
- $\mathbf{F}_3$ should be uncorrelated (orthogonal) to $\mathbf{F}_1$ and $\mathbf{F}_2$ and have as large a variance as possible
- Etc. Etc.

Since, for all j,

$$\overline{\mathbf{F}}_j = \frac{1}{n}\mathbf{1}^T\mathbf{F}_j = \frac{1}{n}\mathbf{1}^T Z\underline{a}_j = 0$$

it is easy to verify that, for all j,

$$\text{Var}(\mathbf{F}_j) = ||\mathbf{F}_j||^2 = \frac{1}{n-1}\mathbf{F}_j^T\mathbf{F}_j = \frac{1}{n-1}\underline{a}_j^T Z^T Z\underline{a}_j = \underline{a}_j^T R_{XX}\underline{a}_j \qquad (3)$$

so, a trivial way to maximize each $\text{Var}(\mathbf{F}_j)$ is to multiply each $\underline{a}_j$ by an arbitrarily large constant c (this jacks up $\text{Var}(\mathbf{F}_j)$ by a factor c^2. To avoid this, we add the constraint

$$||\underline{a}_j||^2 = \underline{a}_j^T a_j = 1 \; \forall j \qquad (4)$$

Similarly, it is easy to verify that

$$\text{Cov}(\mathbf{F}_i, \mathbf{F}_j) = 0 \;\; \textit{iff} \;\; \underline{a}_i^T \underline{a}_j = 0 \tag{5}$$

so, to make the components $\mathbf{F}_1, \mathbf{F}_2, \ldots$ uncorrelated, we should make the $\underline{a}_j$'s orthogonal to one another,

$$\underline{a}_i^T \underline{a}_j = 0 \;\; \forall i \neq j \tag{6}$$

This line of reasoning leads to the following system of Lagrange Multiplier problems to solve:

$$\left.\begin{array}{rcl} C_1 & = & \underline{a}_1^T R_{XX} \underline{a}_1 - \lambda_1(\underline{a}_1^T a_1 - 1) \\ C_2 & = & \underline{a}_2^T R_{XX} \underline{a}_2 - \lambda_2(\underline{a}_2^T a_2 - 1) - \mu_1(\underline{a}_2^T \underline{a}_1) \\ C_3 & = & \underline{a}_2^T R_{XX} \underline{a}_2 - \lambda_2(\underline{a}_2^T a_2 - 1) - \mu_2(\underline{a}_3^T \underline{a}_2) - \mu_1(\underline{a}_3^T \underline{a}_1) \\ \vdots & = & \vdots \\ C_k & = & \underline{a}_k^T R_{XX} \underline{a}_k - \lambda_k(\underline{a}_k^T a_2 - 1) - \displaystyle\sum_{j=1}^{k-1} \mu_j (\underline{a}_k^T a_j) \end{array}\right\} \tag{7}$$

For example, the partial derivatives of C_1 (after much tedium) turn out to be

$$\begin{array}{rcl} \dfrac{\partial C_1}{\partial \underline{a}_1} & = & 2R_{XX}\underline{a}_1 - 2\lambda \underline{a}_1 \\ \dfrac{\partial C_1}{\partial \lambda_1} & = & -(\underline{a}_1^T \underline{a}_1 - 1) \end{array}$$

Setting these equal to zero and cancelling out constants, we obtain

$$\begin{array}{rcl} (R_{XX} - \lambda_1 I_{k\times k})\underline{a}_1 & = & \underline{0} \\ \underline{a}_1^T \underline{a}_1 & = & 1 \end{array}$$

In other words, λ_1 and $\underline{a}$ are an eigenvalue and the corresponding unit eigenvector, for the correlation matrix R_{XX}! Clearly,

$$\text{Var}(\mathbf{F}_1) = \underline{a}_1^T R_{XX} \underline{a}_1 = \underline{a}_1^T \lambda_1 \underline{a}_1 = \lambda_1 \underline{a}_1^T \underline{a}_1 = \lambda_1$$

is maximized by taking λ_1 to be the *largest* eigenvalue of R_{XX}, and $\underline{a}_1$ to be a corresponding unit eigenvector.

Similarly, the entire system of contstrained objective functions (7) is maximized by

$$\begin{array}{ll} \lambda_1 \;\geq\; \lambda_2 \;\geq\; \cdots \;\geq\; \lambda_k : & \text{the eigenvalues of } R_{XX} = \frac{1}{n-1} Z^T Z\text{; and} \\ \underline{a}_1 \;\;,\;\;\; \underline{a}_2 \;,\;\;\;\; \ldots \;,\; \underline{a}_k : & \text{the corresponding unit eigenvectors.} \end{array}$$

We know immediately that

$$A = [\ \underline{a}_1\ \underline{a}_2\ \cdots\ \underline{a}_k\ ]$$

has orthonormal columns (of length k), i.e. $A^T A = AA^T = I_{k\times k}$, and we also know that $R_{XX}\underline{a}_j = \lambda_j \underline{a}_j,\ \ j = 1, \ldots k$, so that

$$A^T R_{XX} A = \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{bmatrix}$$

Therefore

$$\text{Var}(F) = \frac{1}{n-1} F^T F = \frac{1}{n-1} A^T Z^T Z A = A^T R_{XX} A = \Lambda$$

so that $\text{Var}(\mathbf{F}_j) = \lambda_j$, and the F_j's are uncorrelated.

Finally we can can solve

$$F = ZA$$

by post-multiplying by $L = A^T$ (and using $A^T A = I$ again) to obtain

$$Z = FA^T = FL$$

Summary:

- *The "Model" [really, a transformation]*: If X has K columns, we obtain $J \leq K$ principal components as follows:

  $$\mathbf{z}_j = L_{1j}\mathbf{F}_1 + L_{2j}\mathbf{F}_2 + \cdots + L_{kj}\mathbf{F}_k$$

  where $F = [F_{ij}]$ is the matrix of "component scores" and $L = [L_{ij}]$ is the matrix of "loadings".

- We can recover the original X's as follows:

  $$X_{ij} = (L_{1j}F_{i1} + L_{2j}F_{i2} + \ldots + L_{Jj}F_{iJ}) \cdot s_j + \overline{X}_j$$

- Interpretations and uses:
  - If $J = K$, the matrices X (raw data) and F (principal components) are just functions of each other.
  - If $J < K$, the matrix F (principal components) contains less information than X (hopefully, the "essential" information in X).
  - Often the principal components have "nice" interpretations, that can help us understand the structure of the data.
  - In a regression problem, we would try to work the with principal components (columns of the F matrix), $\mathbf{F}_1$, $\mathbf{F}_2$, $\mathbf{F}_3$, etc., instead of the original X's.

Example: Back to the highway data...

In doing a principal components analysis there are two important things to look at "first":

- A "scree plot", that shows how much variance in the X's is accounted for by each principal component (analogous to R^2 in regression).

  Typically in a scree plot we see the component variances λ_j vs component number j, and also the percentage of total variation explained by the first j factors:

$$\frac{\sum_{\ell=1}^{j} \lambda_\ell}{\sum_{\ell=1}^{k} \lambda_\ell} = \frac{\sum_{\ell=1}^{j} \lambda_\ell}{k-1}$$

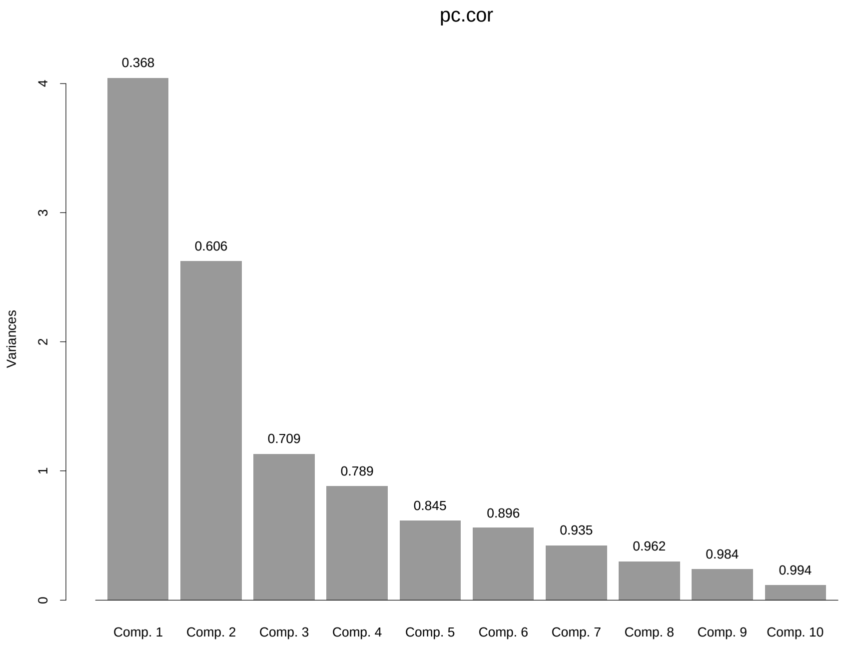
- A listing of the loadings matrix $L = [L_{ij}]$, to see which X's are related to which F's. Since

$$\left[\, \mathbf{z}_1 \; \mathbf{z}_2 \; \cdots \; \mathbf{z}_k \,\right] = \left[\, \mathbf{F}_1 \; \mathbf{F}_2 \; \cdots \mathbf{F}_k \,\right] \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1k} \\ L_{21} & L_{22} & \cdots & L_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ L_{k1} & L_{k2} & \cdots & L_{kk} \end{bmatrix}$$

  it follows that L_{ij} = *degree of influence of factor i on variable j.* When printing the factor loadings it is useful to "blank out" the small ones so that we can see relationships more clearly.

First we consider the scree plot:

```
> high _ read.table("highway.dat",header=T)
> X _ high[,-1]
> y _ high[,1]

> pc.cor _ princomp(X,cor=T) # cor=T uses Z's.  It is possible to
>                            # use X's (cor=F) but we will not...
> screeplot(pc.cor)
```



We interpret the scree plot by looking for either

- An "elbow" that suggests that more principal components don't help much. This suggests we should keep F_1 (component 1) and F_2 (component 2), and F_3 (component 3) which is right at the elbow; or
- A point at which most (say, 90%) of the variance in the original data has been explained. This suggests we should keep F_1, F_2, F_3 and F_4.

Now we look at the loadings to see if there are nice relationships between the X's and the F's. This often helps us interpret the F's.

```
> print(loadings(pc.cor),cutoff=.1) # blanks all loadings below +/- 0.1
     Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5 Comp. 6 Comp. 7 Comp. 8 Comp. 9
 Len  0.114   0.417  -0.372   0.181   0.497           0.562  -0.172   0.198
 Adt -0.446  -0.141  -0.170   0.223  -0.125                   0.108
 Trk          0.447  -0.191   0.414  -0.111  -0.553  -0.460   0.166  -0.120
  Sp -0.275   0.386   0.249  -0.310   0.140  -0.136   0.127   0.535
  lw         -0.134   0.769   0.421   0.254  -0.240   0.198
 shl -0.358   0.147          -0.606          -0.288          -0.172   0.102
 itg -0.415  -0.117  -0.169   0.286  -0.283   0.133   0.166   0.310   0.440
sigs         -0.466  -0.210           0.658  -0.154  -0.370   0.270   0.251
acpt  0.188  -0.405  -0.167          -0.277  -0.685   0.407
 lan -0.417  -0.150  -0.196           0.209           0.165          -0.790
type  0.441                                   0.132   0.226   0.666  -0.217
```

```
     Comp. 10 Comp. 11
 Len
 Adt  0.266    0.767
 Trk  0.137
  Sp -0.493    0.168
  lw  0.198
 shl  0.578   -0.152
 itg          -0.537
sigs
acpt -0.213
 lan          -0.248
type  0.483

> print(loadings(pc.cor),cutoff=.2) # blanks all loadings below +/- 0.2
     Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5 Comp. 6 Comp. 7 Comp. 8 Comp. 9
 Len          0.417  -0.372           0.497           0.562
 Adt -0.446                   0.223
 Trk          0.447           0.414          -0.553  -0.460
  Sp -0.275   0.386   0.249  -0.310                           0.535
  lw                  0.769   0.421   0.254  -0.240
 shl -0.358                  -0.606          -0.288
 itg -0.415                   0.286  -0.283                   0.310   0.440
sigs         -0.466  -0.210           0.658          -0.370   0.270   0.251
acpt         -0.405                  -0.277  -0.685   0.407
 lan -0.417                           0.209                          -0.790
type  0.441                                           0.226   0.666  -0.217

     Comp. 10 Comp. 11
 Len
 Adt  0.266    0.767
 Trk
  Sp -0.493
  lw
 shl  0.578
 itg          -0.537
sigs
acpt -0.213
 lan          -0.248
type  0.483
```

```
> print(loadings(pc.cor),cutoff=.3) # # blanks all loadings below +/- 0.3
     Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5 Comp. 6 Comp. 7 Comp. 8 Comp. 9
 Len          0.417  -0.372           0.497           0.562
 Adt -0.446
 Trk          0.447           0.414          -0.553  -0.460
  Sp          0.386          -0.310                          0.535
  lw                  0.769   0.421
 shl -0.358                  -0.606
 itg -0.415                                                  0.310   0.440
sigs         -0.466                   0.658          -0.370
acpt         -0.405                          -0.685   0.407
 lan -0.417                                                         -0.790
type  0.441                                                  0.666

     Comp. 10 Comp. 11
 Len
 Adt           0.767
 Trk
  Sp -0.493
  lw
 shl  0.578
 itg          -0.537
sigs
acpt
 lan
type  0.483
```

From this we can see that

- Component 1 is mostly related to:

```
Adt  - average daily traffic count in thousands (estimated)
shl  - width in feet of outer shoulder on the roadway
itg  - number of freeway-type interchanges per mile in the segment
lan  - total number of lanes of traffic in both directions
type -  1 if Federal aid interstate highway,
        2 if principal arterial interstate highway,
        3 if minor arterial interstate highway,
        4 otherwise
```

  and seems to be related to whether the highway is a *major road*.

- Component 2 is mostly related to:

```
Len  - length of segment in miles
Trk  - truck volume as a percent of total volume
Sp   - speed limit (in 1973, before the 55 mph limits)
sigs - number of signalized interchanges per mile in the segment
acpt - number of access points per mile in the segment
```

  and shares little in common with component 1! This component seems to be related to aspects of the highway that *slow down* traffic; it is also the same set of variables that BIC variable selection picked up.

- Component 3 is mostly related to:

```
Len  - length of segment in miles
lw   - lane width in feet
```

  Perhaps this component is related to whether this is an *open road* where visibility is good.

- Component 4 is mostly related to:

```
Trk  - truck volume as a percent of total volume
Sp   - speed limit (in 1973, before the 55 mph limits)
lw   - lane width in feet
shl  - width in feet of outer shoulder on the roadway
```

  This component may be related to *crowdedness* on the highway.

Let's give these components interpretable names, and re-do the regression analysis:

```
> dim(pc.cor$scores)  # this is where F1, F2, ...  are kept
[1] 39 11
> major.road _   - pc.cor$scores[,1]
> slow.downs _   - pc.cor$scores[,2]
> open.road  _   - pc.cor$scores[,3]
> crowdedness _    pc.cor$scores[,4]

> attach(high)
> summary(pclm _ lm(Rate ~ major.road + slow.downs +
+ open.road + crowdedness))

Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept)   3.9333   0.1751    22.4637   0.0000
```

```
 major.road  -0.2711   0.0871    -3.1136   0.0037
 slow.downs   0.9628   0.1081     8.9078   0.0000
  open.road   0.2500   0.1645     1.5198   0.1378
crowdedness  -0.0058   0.1864    -0.0311   0.9754

Residual standard error: 1.093 on 34 degrees of freedom
Multiple R-Squared: 0.7288
F-statistic: 22.84 on 4 and 34 degrees of freedom,
     the p-value is 3.116e-09

Correlation of Coefficients:
            (Intercept) major.road slow.downs open.road
 major.road  0
 slow.downs  0           0
  open.road  0           0          0
crowdedness  0           0          0          0
```

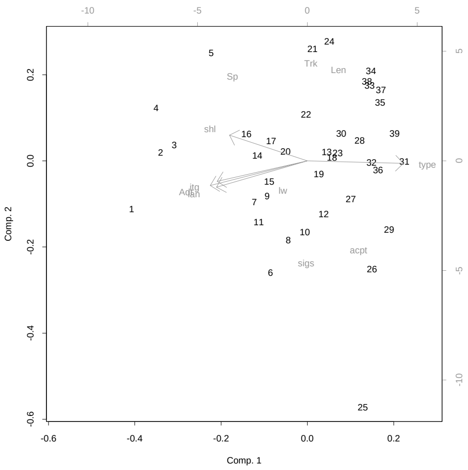
Compared with our first regression analysis of highway accident rate, using all the variables,

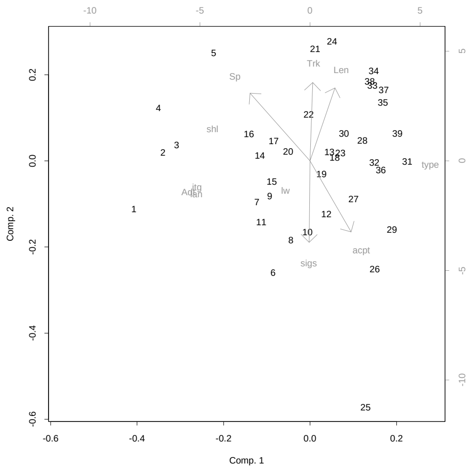
- The estimated β's have 0 correlation, because the components are themselves perfectly uncorrelated—no collinearity at all!
- It looks like only the first two components, major.road and slow.downs, were needed after all.
- We haven't lost much explanatory power from the original regression analysis! R^2 dropped only from 75% to 73%, and you can verifty that the usual sets residual diagnostic plots for the two regressions are very similar.

Another tool for displaying the component scores and factor loadings is the *biplot*:

- Each observation i as a point $(F_{i1}/\sqrt{\lambda_1}, F_{i2}/\sqrt{\lambda_2})$;
- Each variable X_j as a point (L_{j1}, L_{j2})

The biplot uses the component scores and factor loadings to relate observations and variables to one another, on the same plot:





Principal Components Regression

The *principal components* of Z are the columns of the matrix

$$F = ZA$$

Since $AA^T = I$, Equation (2) can be written

$$\begin{aligned} \underline{y} &= \mathbf{1}\beta_0 + Z\alpha + \epsilon \\ &= \mathbf{1}\beta_0 + (ZA)(A^T\alpha) + \epsilon \\ &= \mathbf{1}\beta_0 + F\gamma + \epsilon \end{aligned} \tag{8}$$

(where $\gamma = A^T\alpha$), and this is equivalent to our original Equation (1).

The least-squares estimates of γ are $\hat{\gamma} = (F^TF)^{-1}F^Ty$, and as usual

$$\text{Var}(\hat{\gamma}) = \sigma_\epsilon^2(F^TF)^{-1} = \sigma_\epsilon^2(A^TZ^TZA)^{-1} = \frac{\sigma_\epsilon^2}{n-1}\Lambda^{-1}$$

is a diagonal matrix, and Var$(\hat{\gamma}_k) = \frac{1}{n-1}\sigma^2\lambda_k^{-1}$, so that

- the $\hat{\gamma}$'s are uncorrelated; and
- the data tells us the least about the γ_k for the smallest eigenvalue λ_k.

This suggests that we can "improve" the model by dropping columns of F corresponding to high Var$(\hat{\gamma}_k)$ i.e. low λ_j. If the original model in Equation (1) was the correct model for the data, then the estimates $\hat{\beta}$, $\hat{\alpha}$ and $\hat{\gamma}$ in these problems should all be unbiased.

To see what the effect of dropping columns of F is, consider minimizing the sum of mean square errors (SMSE) in the coefficients, instead of in the predicted values,

$$
\begin{aligned}
SMSE &= \sum_{j=1}^{k} E[(\hat{\gamma}_j - \gamma_j)^2] \\
&= \sum_{j=1}^{k} \{\text{Var}(\hat{\gamma}_j) + [bias(\hat{\gamma}_j)]^2\} \\
&= \frac{\sigma_\epsilon^2}{n-1} \sum_{j=1}^{k} \lambda_j^{-1} + \sum_{j=1}^{k} \{E[\hat{\gamma}_i - \gamma_j]\}^2 \qquad (9)
\end{aligned}
$$

where $\hat{\gamma}_j$ are the least-squares estimates. When the full set of least-squares estimates are used, $\hat{\gamma}_j$ will be unbiased and the "bias" term on the right will be zero, so

$$
SMSE_{OLS} = \frac{\sigma_\epsilon^2}{n-1} \sum_{j=1}^{k} \lambda_j^{-1}
$$

Because the columns of W are orthogonal, when we drop columns we do not change the least-squares estimates of the remaining coefficients. Dropping the last m columns for example we obtain

$$
SMSE_{PC(k-m)} = \frac{\sigma_\epsilon^2}{n-1} \sum_{j=1}^{k-m} \lambda_j^{-1} + \sum_{j=k-m+1}^{k} \gamma_j^2
$$

This is a classic case of the *bias/variance tradeoff* in statistics: by dropping only a few terms we can reduce SMSE by paying for a large drop in variance with a small increase in bias; clearly we can minimize $SMSE_{PC(k-m)}$ over m if desired.

For the highway data, we can estmate both SMSE's:

```
> names(pc.cor)
 [1] "sdev"         "loadings"     "correlations" "scores"       "center"
 [6] "scale"        "n.obs"        "call"         "factor.sdev"  "coef"
```

```
> summary(lm(y ~ pc.cor$scores))

Coefficients:
                        Value Std. Error  t value Pr(>|t|)
          (Intercept)   3.9333   0.1876    20.9661   0.0000
 pc.cor$scoresComp. 1   0.2711   0.0933     2.9060   0.0072
 pc.cor$scoresComp. 2  -0.9628   0.1158    -8.3139   0.0000
 pc.cor$scoresComp. 3  -0.2500   0.1762    -1.4185   0.1675
 pc.cor$scoresComp. 4  -0.0058   0.1997    -0.0290   0.9771
```

```
 pc.cor$scoresComp. 5  -0.3396   0.2394    -1.4188   0.1674
 pc.cor$scoresComp. 6  -0.1522   0.2508    -0.6068   0.5490
 pc.cor$scoresComp. 7   0.0724   0.2882     0.2510   0.8037
 pc.cor$scoresComp. 8   0.1167   0.3430     0.3402   0.7364
 pc.cor$scoresComp. 9   0.0296   0.3838     0.0772   0.9391
pc.cor$scoresComp. 10  -0.0931   0.5475    -0.1700   0.8663
pc.cor$scoresComp. 11   0.1133   0.7480     0.1515   0.8807

Residual standard error: 1.172 on 27 degrees of freedom
Multiple R-Squared: 0.7527
F-statistic: 7.472 on 11 and 27 degrees of freedom,
the p-value is 1.051e-05

> dim(high)
[1] 39 12
> n _ 39

> print(SMSE.OLS _ (1.172)^2 * sum(pc.cor$factor.sdev^(-2))/(n-1) )
[1] 1.458874
> coefs _ lm(y ~ pc.cor$scores)$coef
> coefs[-c(1:3)]
 pc.cor$scoresComp. 3 pc.cor$scoresComp. 4 pc.cor$scoresComp. 5
           -0.2499623         -0.005792799            -0.339593

 pc.cor$scoresComp. 6 pc.cor$scoresComp. 7 pc.cor$scoresComp. 8
           -0.1521965           0.07235734             0.116677

 pc.cor$scoresComp. 9 pc.cor$scoresComp. 10 pc.cor$scoresComp. 11
           0.02961616          -0.09308246             0.1133036
> print( SMSE.PC2 _ (1.172)^2 * sum(pc.cor$factor.sdev[1:2]^(-2))/(n-1) +
+ sum(coefs[-c(1:3)]^2) )
[1] 0.2649427
```

So we win the bias/variance tradeoff game by regressing on only the first two principal components (which we named, in our previous analysis, the `major.road` and `slow.downs` indices).

Principal Components as a Smoothing Technique

The principal components transformation is $F = ZA$, which we can express as a "model" for the Z's (or equivalently, for the X's) as

$$Z = FL$$

When we drop the last few columns of F (and the corresponding rows of L), we are throwing away information about the small variations from case to case within each column of Z (this is, again, because the last few λ_ℓ's are small, so these last few F_ℓ's can only be telling about small variations in the Z's).

We can use this fact to compute a "smoothed version" of

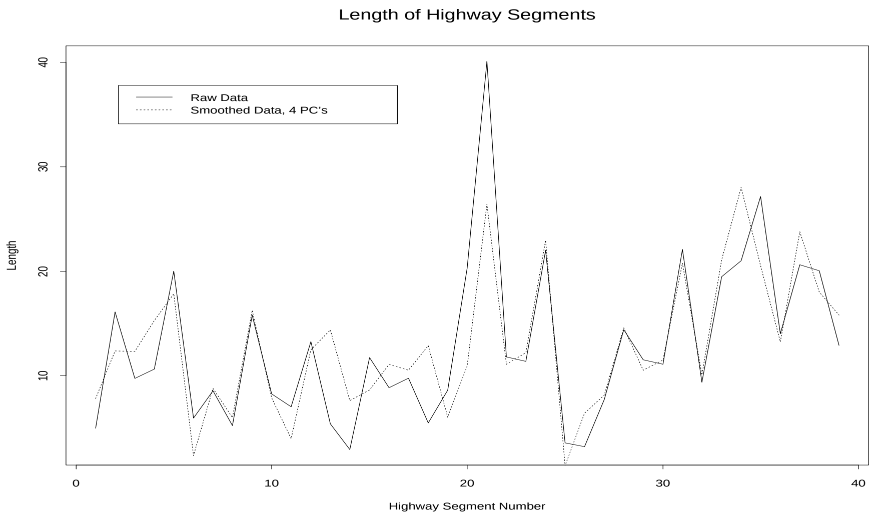
$$Z_{smooth} = F_{reduced} L_{reduced}$$

The columns of Z_{smooth} have less jitteryness, and often less extreme peaks and troughs, than the columns of Z itself. We can feed this back through to get a smoothed version of X, as follows:

$$\mathbf{x}_j = \mathbf{z}_j * s_j + \overline{x}_j = (L_{1j}\mathbf{F}_1 + L_{2j}\mathbf{F}_2 + \cdots + L_{Jj}\mathbf{F}_J) + \overline{x}_j$$

where $J < K$. For example, the plot below shows the effect of reconsructing the predictor variable `Len` from just four principal components, instead of all 11. It was created from a principal components analysis as follows:

```
> pc.cor.4 _ princomp(X,cor=T,scores=4)   # only retain first 4 factors
> xsmooth _  t(pc.cor.4$scores %*% pc.cor.4$loadings[1,] *
+ pc.cor.4$scale[1] + pc.cor.4$center[1])
> plot(X[,1],type="n",xlab="",ylab="")
> lines(X[,1],lty=1)
> lines(xsmooth,lty=2)
> legend(locator(),lty=1:2,legend=c("Raw Data","Smoothed Data, 4 PC's"))
> title("Length of Highway Segments",
+ xlab="Highway Segment Number",ylab="Length")
```



Factor Analysis

In the principal components representation

$$Z = FL,$$

when we drop columns of F (and the corresponding rows of L), we are not asserting that the difference between Z and $F_{reduced}L_{reduced}$ is random noise, but rather we are just throwing away some of the information about Z.

There is a related model (a true model, not just a representation!), called the *factor analysis model*, that asserts that

$$Z = F_{reduced}L_{reduced} + U$$

where U is a matrix of residual errors, analogous to $\underline{\epsilon}$ in linear regression. To save space and typing I will write the factor analysis model as just

$$Z_{n\times K} = F_{n\times J}L_{J\times K} + U_{n\times K}$$

where $J < K$, and usually I will drop the subscripts altogether.

The full set of assumptions for the factor analysis model are:

- $Z = FL + U$, where Z, F, L and U have the dimensions indicated above.
- The columns of F are uncorrelated, with unit variance;
- The columns of U are uncorrelated, with possibly different variances.

It is usually assumed that the data in F and U, and therefore Z, actually come from a multivariate normal distributions:

- $(F_{i1}, \ldots, F_{iJ})^T \overset{iid}{\sim} MVN(\underline{0}_{J\times 1}, I_{J\times J})$
- $(U_{i1}, \ldots, U_{iK})^T \overset{iid}{\sim} MVN(\underline{0}_{J\times 1}, Q_{K\times K})$, independent of the F's.

where I is the identity matrix and Q is a diagonal matrix with variances of U_j's written down its diagonal. Note that this means that also

- $(Z_{i1}, \ldots, Z_{iK})^T \overset{iid}{\sim} MVN(\underline{0}_{J\times 1}, L^TL + Q)$

and so the only things to "estimate" are the diagonal entries of the matrix Q, and the loadings in the matrix L.

Interpretation.

This says that the columns in Z are linear combinations of the unobservable (or "latent") factors (columns of F), plus random noise.

Such models get used in

- Education and psychology: For example, the colunms of Z might be standardized test scores (e.g. sections of the SAT, ACT, GRE, etc.) for a group of students, and the columns (latent factors) of F might be interpreted as measures of basic skills in mathematical and verbal reasoning.
- Settings in the sciences where observations (Z's or X's) are measurements, that include some measurement error (U's), of more fundamental quantities F's.

How do we estimate the model?

There are two common methods.

- *Principal factor analysis*:
  1. Begin with estimates of the diagonal entries q_{jj} of Q. These "error variances" are called "*uniquenesses*". One such estimate is
  $$q_{jj} = 1 - R_j^2,$$
  where R_j^2 is the R^2 from regressing Z_j on all the other Z's.
  2. Note that the covariance matrix for Z is $R_{XX} = L'L + Q$; so compute $R^* = R_{XX} - \hat{Q}$, (the Q matrix with the estimated q_{jj}'s).
  3. Do the principal components analysis
  $$R^* = A\Lambda A^T$$
  and define $L = \sqrt{\Lambda^{-1}}A^T$ ($\sqrt{\Lambda^{-1}}$ is easy to compute, since Λ is a diagonal matrix). It follows that
  $$R^* = L^T L$$
  4. Delete all rows of L after the first J rows, to obtain $L_{reduced}$. Under the model
  $$R_{XX} = L_{reduced}^T L_{reduced} + Q$$
  so a new estimate of the diagonal entries q_{jj} is
  $$q_{jj} = (j^{th} \text{ diag entry of } R_{XX}) - (j^{th} \text{ diag entry of } L_{reduced}^T L_{reduced}) = 1 - \sum_{k=1}^{J} L_{jk}^2$$
  Repeat from step 2 with the new $\hat{Q}$ matrix, and continue until no more changes occur.

- *Maximum likelihood:*

  For fixed J, the model

  $$(Z_{i1},\ldots,Z_{iK})^T \overset{iid}{\sim} MVN(\underline{0}_{J\times 1}, L^T L + Q)$$

  has a likelihood; and under suitable constraints we can differentiate with respect to entries L_{ij} and q_{jj} and obtain maximum likelihood estimates for this model.

- *Notes:*

  - These methods typically require that J be fixed in advance. If you are not sure how large J should be, pick a relatively large value of $J < K$, and look at a scree plot of the λ's in the fitted factor analysis model.
  - We mentioned that q_{jj} are called *uniquenesses* or *unique variances*. The complementary quantities,

    $$r^*_{jj} = 1 - q_{jj} = \sum_{k=1}^{J} L^2_{jk}$$

    are called the *communalities* or *common variances*. The closer to 1 the communalities, the better you like the factor analysis model. (Diagnostic plots of the U's, etc. would also be interesting to look at, to check normality, etc.)

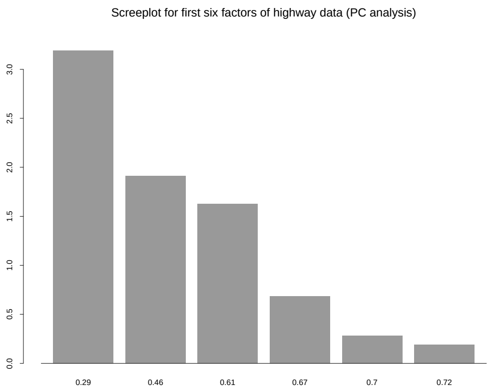
Example

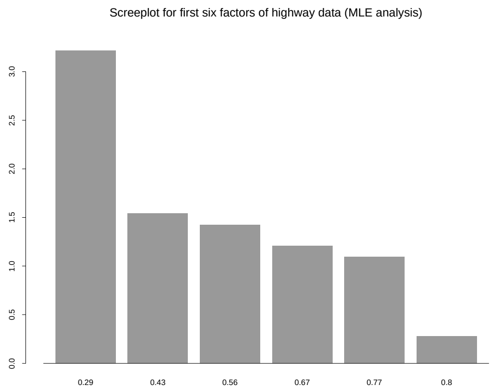
We will simply look at the factor analysis of the highway accident data again.

```
> fa.pc _ factanal(X,factors=6,method="principal")
> fa.ml _ factanal(X,factors=6,method="mle")
> summary(fa.pc)
Importance of factors:
                 Factor1   Factor2   Factor3    Factor4    Factor5    Factor6
   SS loadings 3.1927698 1.9121153 1.6264165 0.68544335 0.28001592 0.19064513
Proportion Var 0.2902518 0.1738287 0.1478560 0.06231303 0.02545599 0.01733138
Cumulative Var 0.2902518 0.4640805 0.6119365 0.67424954 0.69970554 0.71703691

The degrees of freedom for the model is 4.

Uniquenesses:
       Len        Adt       Trk        Sp        lw        shl        itg
 0.3892493 0.04806659 0.4205582 0.1882732 0.6000922 0.08180937 0.09006204

      sigs      acpt       lan      type
 0.4605226 0.4610779 0.1808086 0.1920741
```

```
Loadings:
     Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
 Len -0.184   0.580           0.419   0.228
 Adt  0.958           0.144
 Trk          0.712           0.176          -0.193
  Sp  0.176   0.495   0.692  -0.125           0.205
  lw                         -0.625
 shl  0.351           0.885
 itg  0.921                          -0.225
sigs  0.147  -0.649  -0.175           0.255
acpt -0.162  -0.593  -0.293   0.118          -0.238
 lan  0.803  -0.170   0.250           0.271
type -0.734  -0.117  -0.393   0.207  -0.173   0.170
> names(summary(fa.pc))
[1] "varex"        "uniquenesses" "dof"          "na.action"    "loadings"
[6] "cutoff"
> print(varex _ summary(fa.pc)$varex)
                  Factor1   Factor2   Factor3    Factor4    Factor5    Factor6
   SS loadings 3.1927698 1.9121153 1.6264165 0.68544335 0.28001592 0.19064513
Proportion Var 0.2902518 0.1738287 0.1478560 0.06231303 0.02545599 0.01733138
Cumulative Var 0.2902518 0.4640805 0.6119365 0.67424954 0.69970554 0.71703691

> # Here is a screeplot
> barplot(varex[1,],names=paste(round(varex[3,],2)))
> title("Screeplot for first six factors of highway data (PC analysis)")

> summary(fa.ml)
Importance of factors:
                  Factor1   Factor2   Factor3   Factor4    Factor5    Factor6
   SS loadings 3.2197268 1.5416271 1.4262569 1.2059143 1.09831735 0.28079665
Proportion Var 0.2927024 0.1401479 0.1296597 0.1096286 0.09984703 0.02552697
Cumulative Var 0.2927024 0.4328504 0.5625101 0.6721387 0.77198569 0.79751266

The degrees of freedom for the model is 4.

Uniquenesses:
       Len        Adt    Trk         Sp           lw    shl       itg      sigs
 0.5363893 0.02850704 4e-10 0.2031999 0.0001163469 4e-10 0.1251783 0.4873967

      acpt       lan      type
 0.4209439 0.1505476 0.2750925
```

```
Loadings:
     Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
 Len -0.223   0.391  -0.101   0.389  -0.271   0.162
 Adt  0.973           0.131
 Trk          0.283           0.954
  Sp  0.181   0.647   0.567   0.124
  lw                                  0.992
 shl  0.345   0.187   0.916
 itg  0.914                                  -0.169
sigs  0.149  -0.564          -0.279           0.290
acpt -0.164  -0.719  -0.115  -0.128
 lan  0.812           0.227  -0.104           0.356
type -0.722  -0.120  -0.402          -0.135

> # Here is a scree plot again:
> varex _ summary(fa.ml)$varex
> barplot(varex[1,],names=paste(round(varex[3,],2)))
> title("Screeplot for first six factors of highway data (MLE analysis)")
```





The principal factor analysis (on the left) suggests either two, or four or five, factors. The MLE factor analysis (on the right) seems to suggest two factors.

Let's try two factors…

```
> fa.pc _ factanal(X,factors=2,method="principal")
> fa.ml _ factanal(X,factors=2,method="mle")
```

Let's see what the factor structure looks like now…

```
> print(loadings(fa.pc),cutoff=.3)
     Factor1 Factor2
 Len          0.525
 Adt  0.934
 Trk          0.599
  Sp  0.435   0.689
  lw
 shl  0.613   0.313
 itg  0.820
sigs         -0.662
acpt         -0.633
 lan  0.837
type -0.852
> print(loadings(fa.ml),cutoff=.3)
     Factor1 Factor2
 Len          0.320
 Adt  0.994
 Trk          0.379
  Sp  0.336   0.885
  lw
 shl  0.517   0.551
 itg  0.901
sigs         -0.512
acpt         -0.572
 lan  0.827
type -0.780
```

It's nice to see that we again have fairly simple structure, with Factor 1 and Factor 2 influencing almost disjoint sets of variables. These factors have a lot in common with the first two principal components from our earlier principal components analysis:

- `major.road` mostly influenced `Adt, shl, itg, lan, type`
- `slow.down` mostly influenced `Len, Trk, Sp, sigs, acpt`

but there are some differences: `sp` and `shl` are now "shared" between the two factors, instead of being separated as before.

It may be disappointing that the two models didn't find "exactly" the same factors. That is a sad fact about factor analysis, principal components, and related methods: there are so many parameters, and so much freedom in estimating them, that seemingly small changes (like going from princ. comp. to factor analysis) can generate large, sometimes difficult to interpret, changes in parameter estimates.

These methods can be quite useful, especially if you have a large data set with many variables and you have to reduce it somehow, but *use with care, and lots of checking and self-doubt*.

Rotations (For Factor Analysis and For Principal Components)

Sometimes the factor loadings do not exhibit such a nice "simple" structure (in which non-zero (pr large) loadings in one factor imply zero (or small) loadings in all the others.

In that case, the factors can be "rotated" in J dimensional space, to try to achieve simpler structure in the loadings.

The relevant Splus function is `rotate`, for example,

```
> print(loadings(rotate(fa.ml,rotation="quartimax")))
     Factor1 Factor2
 Len -0.237   0.323
 Adt  0.992  -0.115
 Trk          0.380
  Sp  0.348   0.880
  lw  0.131
 shl  0.524   0.544
 itg  0.900
sigs         -0.514
acpt -0.236  -0.569
 lan  0.826
type -0.783  -0.183
```

(Note that this rotation did not simplify the structure!)

Some common methods of rotation are

- `varimax` rotation, which rotates F so that the sum of the variances of the *squares* of the entries in each column of F is maximized. By default `factanal` produces factors after `varimax` rotation. If you want to see the unrotated factors and loadings, specify `rotation="none"` as an argument to `factanal`.
- `quartimax` rotation, which rotates F so that the sum of the variances of the *fourth powers* of the entries in each column of F is maximized.

Consult `help(rotate)` (and the references listed there) for a myriad of other rotation options.