Factor Analysis

Signal Data Science

In this assignment, you'll be learning about factor analysis. Like PCA, factor analysis falls into the class of linear dimensionality reduction; however, both the technique and the philosophy of factor analysis are quite different.

Jn PCA, one tries to explain *all* of the variance in p-dimensional data with the p principal components. In contrast, in factor analysis, one chooses a *number of factors* $k < p$ and calculates those factors such that they explain as much of the *shared variance* in the data as they possibly can. In other words, we *posit* that the observed data linearly are generated via k unobserved or *latent* factors and then find the best possible factors given this model, each of which can be expressed as a linear combination of the observed variables. PCA re-expresses your data in a different fashion but the p principal components together capture 100% of the information of the original dataset; this cannot be said of the factors in factor analysis, which is a generative model which, again, focuses only on finding a lower-dimensional structure which captures as much of the *shared* variance as possible and *not* the *unique* variance in each individual variable (*e.g.*, noise).

Keep the following in mind while you work:

- Factor analysis is implemented in the R package `psych` as `fa()`.
- The outputs of factor analysis, accessed with `fa()`, are similar to the outputs of `prcomp()`. In particular, if we have `p = prcomp(...)` and `f = fa(...)`, then `f$loadings` is analogous to `p$rotation` and `f$scores` is analogous to `p$x`.
- You should be consistently using `corrplot()` to visualize the loadings of principal components / factors.
- The two dataset analysis questions are *intentionally* unstructured and open-ended. Feel free to spend time exploring the data from multiple angles, reading about related material, and in general just trying any of the tools you've learned about.

Write up your analysis in an R Markdown file, including your interpretation of any results.

PCA vs. factor analysis with simulated data

For reproducibility, place `set.seed(1)` at the top of your code.

Part 1: PCA

We'll explore the differences between PCA and factor analysis by seeing how they perform when our data are *really* generated from a set of unobserved, latent variables. In particular, we'll create three vectors `X`, `Y`, and `Z` from which we'll derive the rest of our data.

- Make a `factors` data frame with 100 observations of 3 normally distributed variables `X`, `Y`, and `Z`. Each variable should be drawn from the standard normal distribution. The "factors" here have nothing to do with factors in R, instead representing latent variables.

Next, we need a function to help us create `derived` variables. The simplest latent variable model is one where each observed variable is noisily generated from just a single latent variable, so we'll write a function that helps us create such observed variables, *i.e.*, "proxies" to the factors.

- Write a function `noisyProxies(feature, k, correlation)` that takes a vector `feature` and returns a data frame with `k` noisy proxies to the feature which are (1) correlated with the feature at the level of `correlation` and (2) differ from the feature by a normally distributed error term.[1] Your function should not include `feature` itself in the data frame.
- Make a dataframe `noisies` with 4 noisy proxies to `X` and 3 noisy proxies to `Y` with `correlation = 0.9`. Use `corrplot()` to plot the correlation matrix of `noisies`.
- Run PCA on the noisy proxies and plot the correlations between the principal components and the `noisies` data frame. Also, plot the correlations between the principal components and the `factors` data frame.

You should see that the first two principal components pick up on the factors `X` and `Y`, but only imperfectly.

Part 2: Orthogonal factor analysis

Factor analysis is an alternative to principal component analysis for dimensionality reduction. Here, one picks a natural number k and assumes that each of the variables V_i can be written as a sum of

[1] Refer back to *Linear Regression: Simulated Data* for information on how to do so. Essentially, you want to take the sum of `correlation*feature` and an error term proportional to $\sqrt{1 - (\text{correlation})^2}$.

$$V_i = a_1F_1 + a_2F_2 + \cdots + a_kF_k + \text{error}_i$$

where each a_j is a constant that depends on V_i. Goodness of fit is measured by taking the correlation matrix of the errors error_i and measuring how far it is from being the identity matrix with 1s down the diagonal (as usual) and 0s off of the diagonal.

Again, the key difference from PCA here is that the factors are supposed to explain as much of the correlations *between* the variables as possible, rather than as much *total* variance as possible: we don't try to pick up on the variables to the extent that they're not correlated with one another.

- Run factor analysis on `noisies` with `nfactors=2` and `rotate="varimax"`. Compare the correlations between the extracted factors and the true factors `X` and `Y`.

You should see that the extracted factors are closer to the true factors than the principal components.

Next, we'll explore the case where our observed variables are generated from *multiple* latent variables, specifically the case where each observed variable is a random linear combination of `X`, `Y`, and `Z` along with some noise.

- Generate 50 variables, each given by

  ```
  X*runif(1) + Y*runif(1) + Z*runif(1) + 0.5*error
  ```

  where `error` is normally distributed with standard deviation 1.

- Run principal component analysis on the 50 variables. Compute the correlations between the first 3 principal components and the variables `X`, `Y`, and `Z`.

- Now do factor analysis with `k=3` and `rotate="varimax"` and compute the correlations between the modeled factors and the true factors.

Part 3: Oblique factor analysis:

In the last factor analysis exercise, we were using a type of factor analysis which is analogous to PCA in assuming the factors to be orthogonal (perpendicular) by specifying `rotate="varimax"`. This was appropriate for our analysis because `X`, `Y`, and `Z` were all uncorrelated with each other. Here, we explore *oblique* factor analysis which allows for correlated factors by considering a new latent variable which *is* correlated with both `X` and `Y`.

- Set `W = 0.5*X + Y`.

- Examine the correlation between `W` and `Y`.

- Use `noisyProxies()` to generate 10 indicators associated with `X` and 4 noisy indicators associated with W, with correlations 0.8 in both cases.
- Plot the associated correlation matrix.
- Setting `nfactors=2` in both cases, compare the results of using `fa()` with `rotate="varimax"` and `rotate = "oblimin"` by looking at the correlations of the results with the noisy indicators and with the true factors.

"Varimax" rotation fails to correctly identify the factors because it tries to force the extracted factors to be orthogonal, whereas "oblimin" rotation allows for correlated factors and consequently correctly identifies the true factors.

Factor analysis on real datasets

Factor analysis can be most fruitfully applied to real datasets, where it has the potential to uncover the true underlying factors behind real-world phenomena.

Speed dating dataset

We'll first return to the aggregated speed dating dataset (`speeddating-aggregated.csv` in the `speed-dating` folder).

- Run factor analysis on the 17 activity variables. Try `nfactor = 1`, `2`, `3`, and `4`, both with `rotate="varimax"` and with `rotate="oblimin"`. Use `corrplot()` to visualize the factor loadings (which you can get with `$loadings`). Interpret the results.

Big Five personality data

We'll conclude our exploration of factor analysis by taking a look at personality data. Many different trait theories have been developed which claim that human psychological variation can be reduced to a number of different factors.

- Download the BIG5 dataset here (posted 5/18/2014).
- Compare the results of PCA on the 50 questions with factor analysis using `nfactor=5`. In particular, use `corrplot()` to compare the loadings of the first five principal components and the results of factor analysis.
- Create a logistic regression predictive model of gender in terms of the 50 questions, then in terms of the 5 factors derived from them. Compare the coefficients of the two models. You can use regular `glm()` because there are sufficently many data points compared to variables that the effect of regularization will be relatively minor.

- For each of the five traits, consider only its 10 corresponding questions. Can you replicate the subfactors of the Big 5 personality inventory traits given in The Items in the Big Five Aspects Scales?[2]

[2] See also the original paper Between Facets and Domains: 10 Aspects of the Big Five.