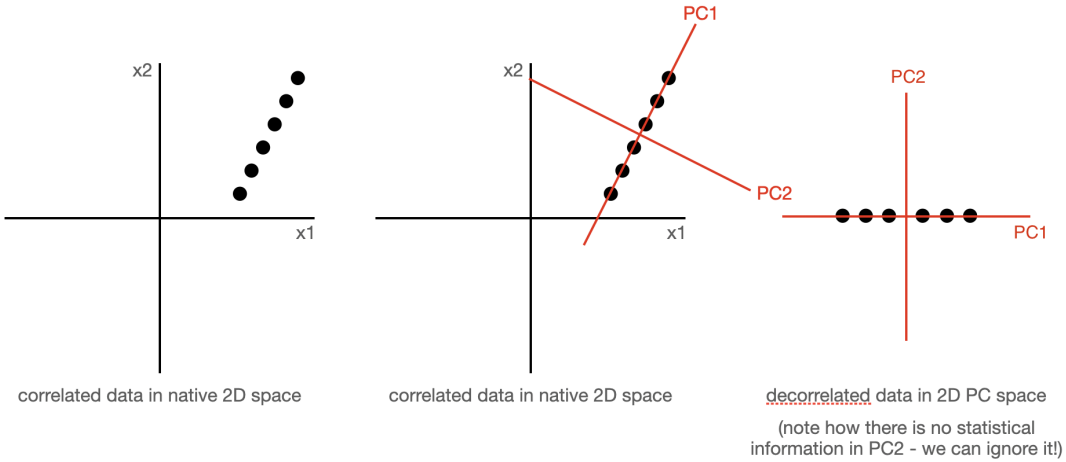
Principal Components Analysis

36-600

Fall 2021

Principal Components Analysis

Let's build up intuition for PCA by starting off with a picture:



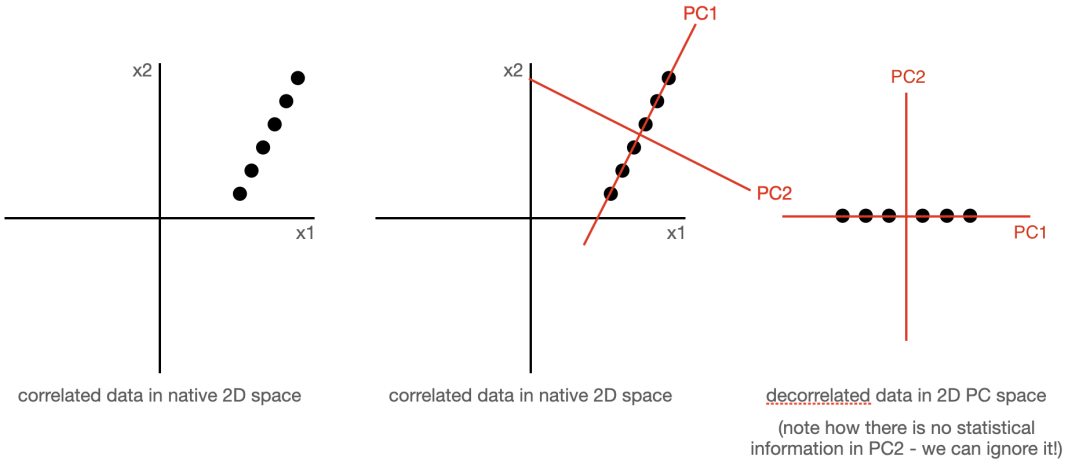
At left, we visualize the contents of a data frame with six rows (six data points) and two columns, x_1 and x_2. These data lay perfectly along a line.

What does the PCA algorithm do? It utilizes algorithms of linear algebra to move the coordinate system origin to the centroid of the point "cloud," then rotates the axes such that "PC 1" lays along the direction of greatest variation (i.e., along the points), and "PC 2" lays orthogonal to to "PC 1." If there were an x_3 axis coming out of the screen, then "PC 3" would be orthogonal to axes 1 and 2. Etc.

In the end, PCA is the linear transformation of a coordinate system...

Principal Components Analysis

...but, there is actually more to PCA then just the transformation.



At right, we move into "PC space," where PC1 has become the x axis, etc.

Note how the data all lay along the PC1 axis. In PC space, we can see that the true space of the data is one-dimensional: a second dimension adds no statistical information at all. We can visualize and interpret the data using, e.g., a histogram of PC1 values and ignore PC2 entirely.

So in the end, PCA is also a dimension-reduction tool.

Why Would We Do PCA?

There are two principal (pun not intended) reasons to use PCA:

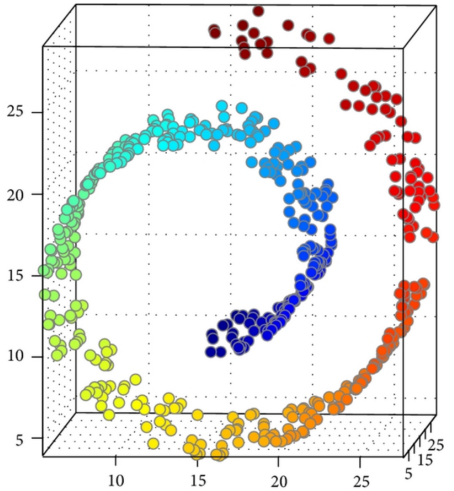
- It can assist data visualization. If the predictor data lay within subspaces of the native space (e.g., they lay along lines or on planes), they may be much more easily visualized: determine PCs, pick the first 1 or 2 (or 3...`scatterplot3d` can help here), and visualize the data in the (reduced) PC space. Color code your data by, e.g., the values of the response variable. You might be able to tell a data story.

- It can help statistical inference. For instance, if you want to do linear regression and your data are afflicted with multicollinearity, you can run the regression in PC space, either with all PCs or just the most-informative subset. Why? PCA "decorrelates" the data: there is no linear association between the data along PC1 and PC2, etc. Multicollinearity is mitigated! The issue in inference then shifts to interpretation, since any PC is a linear combination of the original predictor variables...how exactly do you explain the results?

Note: PCA is not appropriate for use on categorical data! There are off-the-beaten-track PCA-like methods out there for "mixed" data (quantitative and categorical) but we will not cover them here.

PCA: It's a Linear Algorithm

The main limitation of PCA is that it is a *linear* algorithm: it projects data to hyperplanes.



These data lay on a curving two-dimensional strip embedded in a three-dimensional space. Projecting these data to a two-dimensional plane or a one-dimensional line would effective randomize the statistical information they contain (i.e., the projection would not preserve the color ordering, which might represent the value of, say, a response variable).

Here, for visualization and/or learning, we'd use *nonlinear* techniques like diffusion map or local linear embedding, or self-organizing maps or tSNE.

PCA: It's Not Factor Analysis

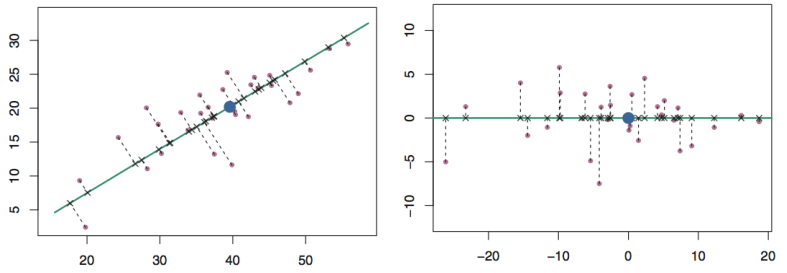
PCA is one of a family of related algorithms commonly dubbed *factor analyses*, and, for instance, some use the phrases "PCA" and "factor analysis" interchangeably. Note that:

1. The PCA algorithm is a **deterministic algorithm**. The algorithm does not take into account that the data to which it is applied are random variables; it just ingests the data values as they are and produces an answer, full stop.

2. *Exploratory factor analysis* is a variant of PCA which *does* attempt to take randomness into account. The idea is to map the p observed variables to $k < p$ factors; "exploratory" means we do not know *a priori* how that mapping may occur. Contrast this with...

3. *Confirmatory factor analysis*, which basically adds on a hypothesis-testing layer for confirming that links actually exist between the k factors uncovered by EFA and the p original variables.

PCA: Algorithm



The "score" or coordinate of the i^{th} observation along PC j is

$$Z_{ij} = \sum_{k=1}^{p} X_{ik}\phi_{kj}$$

where k represents the k^{th} variable or feature (i.e., the k^{th} column of your [predictor] data frame). The algorithm determines the rotation (or loading) matrix ϕ, which is normalized such that $\sum_{k=1}^{p} \phi_{kj}^2 = 1$. Note that j ranges from 1 to p.

Note that since we are "mixing axes" when doing PCA, it is generally best to standardize (or scale) the data frame before applying the algorithm.

PCA: Algorithm (Deeper Detail)

PCA proceeds by factoring the (scaled) data frame via *singular value decomposition*, or *SVD*:

$$X = UDV^T$$

where X is the scaled data frame, U and V are eigenvector matrices, and D is the diagonal matrix of eigenvalues. (V^T means "the transpose of V.") The PC coordinates are then $Z = XV$, meaning that ϕ from the last slide is the matrix of eigenvalues V.

Note that PCA will really only work if n, the number of rows in your data frame, is greater than p, the number of variables.

PCA: Choosing the Number of Dimensions to Retain

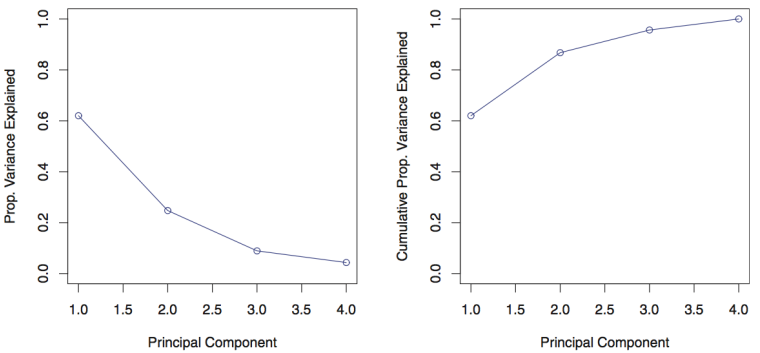
As usual, there is no unique method for doing this.

The ideal would be the following: choose $M < p$ such that

$$x_{ij} = \sum_{m=1}^{p} z_{im}\phi_{jm} \approx \sum_{m=1}^{M} z_{im}\phi_{jm} \, .$$

In other words, we don't lose much ability to reconstruct the input data X by dropping the last $p - M$ principal components, which we assume represent random variation in the data (i.e., noise). (In the example given at the beginning of these notes, we would be able to reconstruct the original data perfectly, even if we dropped PC2.)

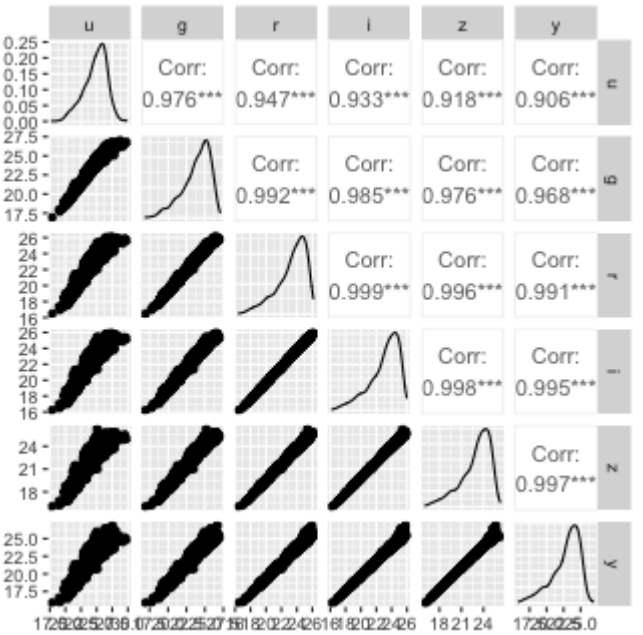
One convention is to sum up the amount of variance explained in the first M PCs, and adopt the smallest value of M such that 90% or 95% or 99%, etc., of the overall variance is "explained." Another is to look for an "elbow" in the left plot (the "scree plot").



PCA: Example

Below we load in examples of galaxy magnitudes in different wavelength bands, from u (for ultraviolet) to z and y in the near-infrared. Because a galaxy that is bright (has low magnitude) in one band tends to be bright in all bands, we see that the magnitude data are obviously correlated.

```
suppressMessages(library(GGally))
ggpairs(pred,progress=FALSE)
```



PCA: Example

```
pca.out <- prcomp(pred,scale=TRUE)
v <- pca.out$sdev^2
round(cumsum(v/sum(v)),3)
```

```
## [1] 0.977 0.998 0.999 1.000 1.000 1.000
```

What we observe is that the first principal component explains 97.7% of the variance in the dataset: the data can safely be transformed from a six-dimensional space to a one-dimensional one with minimal loss of statistical information.

What do we do now? Let's print the column(s) for the PCs that we retain:

```
round(pca.out$rotation[,1],3)
```

```
##     u     g     r     i     z     y
## 0.396 0.411 0.413 0.412 0.410 0.408
```

What we would do is *inference*: if we retain only this PC, what are the variables that map most strongly to it? (Sign doesn't matter: we just want to know which variables are associated with the largest numbers for each PC.) Here, we see that all the original variables contribute almost equally to the first PC.

We'd conclude that we can view the data as lying along a one-dimensional line that exists in the native six-dimensional space, and that the orientation of the line is such that it spans all six of the original dimensions, or that all six of the original variables appear to contain useful statistical information. Full stop.

PCA: Example

Before we leave: what information is in that second PC we ignored, but could have kept?

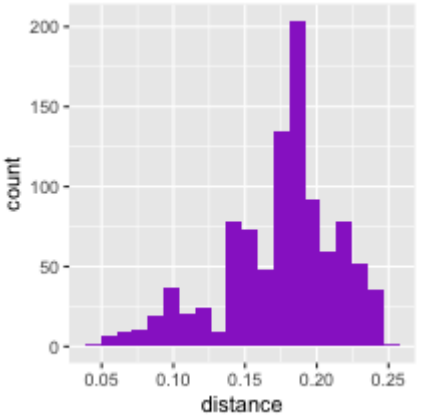
```
round(pca.out$rotation[,1:2],3)
```

```
##     PC1    PC2
## u 0.396  0.793
## g 0.411  0.247
## r 0.413 -0.089
## i 0.412 -0.205
## z 0.410 -0.320
## y 0.408 -0.398
```

We see that the second PC primarily maps to u-band magnitude...basically, the PC2 vector, orthogonal to the PC1 vector, points largely along the u-band axis, meaning there is some variability along that axis that PC1 did not pick up. So PC2 is primarily about the u band. (To build intuition, go back to the `ggpairs` plot, and you'll see that the u-band data tends to be "fuzzier" than the data in the other bands. So the PC2 direction makes sense.) If you decided to retain the second PC, the above information is all the client needs.

PC Regression

Let's assume we retain the first two PCs, and that we have a response variable, which in this case is a measure of the distance of a galaxy from the Sun.



(We see that the response variable data are not perfectly normally distributed...but is probably close enough. Also, a reminder that what needs to be normal is $Y|\mathbf{x}$, not Y itself, so perhaps we are OK. Anyway, as analysts often do, we will proceed with linear regression anyway!)

PC Regression

```
pc1 <- pca.out$x[,1]
pc2 <- pca.out$x[,2]
summary(lm(resp~pc1+pc2))
```

```
##
## Call:
## lm(formula = resp ~ pc1 + pc2)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.108585 -0.020810  0.002557  0.025889  0.091891
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.1745508  0.0011581  150.72  < 2e-16 ***
## pc1         0.0056901  0.0004787   11.89  < 2e-16 ***
## pc2         0.0235364  0.0032110    7.33 4.77e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03651 on 991 degrees of freedom
## Multiple R-squared:  0.1644,    Adjusted R-squared:  0.1628
## F-statistic: 97.52 on 2 and 991 DF,  p-value: < 2.2e-16
```

A reminder: why did we do this again? Because the original data had much multicollinearity.

The issue, as stated before, is inference: yes, we now know that if we change PC2 by 1 unit, the distance will change by 0.024...but how does that change of 1 unit map back to the original variables?

Any statistical analysis done in the original space of data can be done in PC space, but interpretation always involves the original variables...so you will always have to qualitatively describe the mapping between retained PCs and the original variables.

A Parting Note

PCA is **not** a variable selection tool (like, e.g., `bestglm()`). For instance, if you have a dataset with eight predictor variables, did PCA, discovered you only needed to retain two PCs, and saw that only four predictor variables were represented in those two PCs…you would *not* select those predictor variables and go on to do, e.g., linear regression using only those four predictor variables.

To recap: you should use PCA to select (and perhaps further use) PCs, but not to select a subset of the original variables. Use, e.g., `bestglm()`, for that.