October Post - Post 1

Jessica Cherny

Post 1: A Deeper Dive into the Subject of Principal Component Analysis

Introduction:

Principal Component Analysis! What in the world is it? Are you confused? So was I!

R-bloggers defines Principal Component Analysis as: "Principal Component Analysis (PCA) is routinely employed on a wide range of problems. From the detection of outliers to predictive modeling, PCA has the ability of projecting the observations described by p variables into few orthogonal components defined at where the data 'stretch' the most, rendering a simplified overview."

PCA is especially powerful in dealing with multicollinearity and variables that outnumber the samples. For example, if I was a sales executive at Nordstrom, and wanted to predict who will be the most likely buyer of a certain product and how to stock that product accordingly (let's say Steve Madden shoes), I have a lot of variables to consider. Here's a list of a couple of questions I would ask and gather variable data on for Nordstrom customers. What is the age of this buyer? What is their salary? What kind of job do they have? Is this person a mother, father, daughter, brother, etc? What occasion are they shopping for? What color shoes are they looking for? What is their price range? What is their shoe size? Do they do most of their shopping online or in store? Have they bought Steve Madden shoes before? What was their experience like (rating 1 to 10)?

We can think of dozens of more variables to discuss to further pinpoint which would be the best predictor of which customer profile will drive Steve Madden shoe sales the most. But what if we could linearly combine some of these features in order to reveal further insights and reduce feature complexity? This is where Principal Component Analysis comes in. I will discuss more below.

When we initially went over PCA's in lecture, I was confused by what exactly a PCA was. Even after going through the motions of Homework 3, I was not very sure what a principal component analysis really was. I knew it had to do with explaining variation but we did not do any deep dive into it during lecture and it was not tested on the midterm either. It seemed really complicated, but with cool potential applications. Since I thought it was the most complex topic we've covered in class so far, I decided to do further analysis into the topic. Here we go.

Background

Wikipedia defines a Principal Component Analysis as (through my paraphrasing): a statistical procedure that uses transformations to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

When the transformation is finished, the first principal component (pc1) has the largest possible variance (it accounts for as much of the data variability as possible), and every principal component after that one has the highest variance possible under the condition that it is orthogonal to the components before it. Each principal component has less and less variance. The vectors that result are an uncorrelated orthogonal basis set (linear algebra lingo that isn't really important). Source

With a large number of variables, (like our NBA dataset) the covariance matrix may be too large to study and interpret properly. There would be too many pairwise correlations between the variables to consider. Graphical display of data may also not be of particular help incase the data set is very large. To interpret the data in a more meaningful form, it is therefore necessary to **reduce** the number of variables to a few, interpretable linear combinations of the data. Each linear combination will correspond to a principal component. Source

The main idea is that with PCA, we try to reduce the dimensionality (condense information into variables) of a data set while retaining as much as possible of the variation present in the data.

PCs are obtained as linear combinations (weighted sums) of the original variables. (See picture below of weighted calculation). We're looking for PC's that have maximum variance and being mutually uncorrelated.

You can find principal components through diagonalization. To diagonlize a matrix, you just find its eigenvalue decomposition. We just use a prcomp() method in R to output the principal components for a given dataframe (more to come in the Code portion of this post).

Examples

We will look at some code examples using prcomp() on wine data and nba data. Find more examples below in the code section.

Let's just fill in a matrix and explore what prcomp() returns. I've commented ##Code:

```
# Generate scaled 4*5 matrix with random std normal samples
set.seed(101)
mat <- scale(matrix(rnorm(20), 4, 5))
dimnames(mat) <- list(paste("Sample", 1:4), paste("Var", 1:5))

# Perform PCA
myPCA <- prcomp(mat, scale. = F, center = F)

myPCA$sdev #the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix
```

```
## [1] 1.610925e+00 1.427213e+00 6.066157e-01 8.438656e-17
```

```
myPCA$rotation # the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors).
```

```
##               PC1         PC2          PC3         PC4
## Var 1 -0.4539226  0.3573552 -0.74669936  0.2764583
## Var 2 -0.4938492 -0.4228458  0.08865376  0.3935116
## Var 3  0.2739349 -0.6252428 -0.15610452  0.5154027
## Var 4  0.5613671 -0.1540599 -0.60312956 -0.1719370
## Var 5 -0.3998673 -0.5280490 -0.21551495 -0.6881258
```

```
myPCA$x # scores, the value of the rotated data (the centred (and scaled if requested) data multiplied by the rota
tion matrix) is returned.
```

```
##                 PC1         PC2          PC3           PC4
## Sample 1  1.3831715 -0.9315844 -0.63237079 -2.081668e-16
## Sample 2 -2.3289056 -0.2781218 -0.21187540 -2.220446e-16
## Sample 3  0.4497663 -0.8840925  0.81121686 -3.747003e-16
## Sample 4  0.4959677  2.0937987  0.03302933  7.771561e-16
```

```
pca_1 = myPCA$x[,1] #first PCA (the PCA wih the highest variability)
pca_1 #1st PCA
```

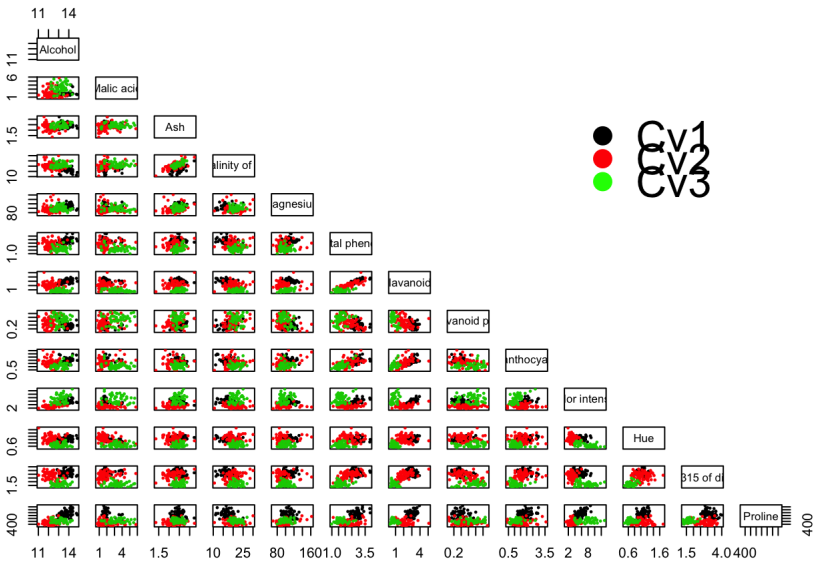
```
##   Sample 1   Sample 2   Sample 3   Sample 4
##  1.3831715 -2.3289056  0.4497663  0.4959677
```

Wine Data

For this part of the tutorial, I referenced Poisson Fish's blog article on Principal Component Analysis (link in references portion of this post). I retrieved some wine data from UC Irvine in the hopes of showing you another PCA example. This data consists of 13 physicochemical parameters measured in 178 wine samples from 3 different cultivars Italy. We want to investigate patterns in variable pairs, and then see what a PCA yields by graphing PCA 1 and PCA 2.

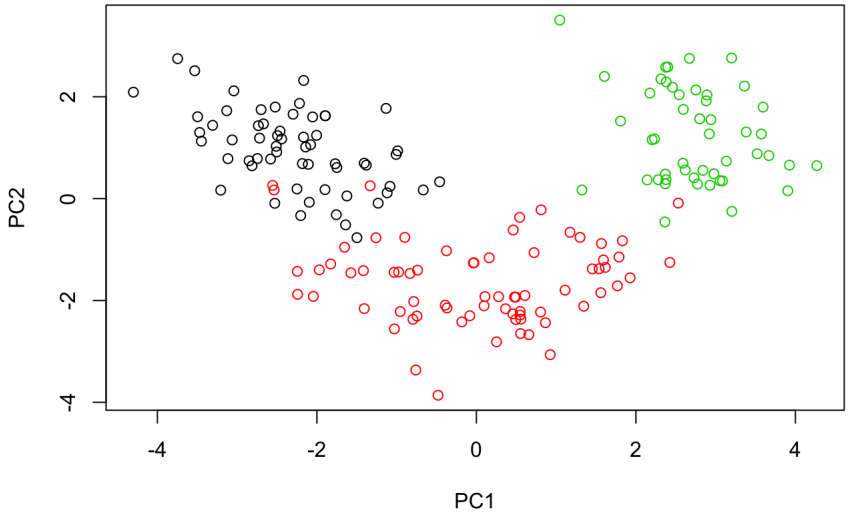
Here we go.

```
wine <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data", sep=",")

# Name the variables
colnames(wine) <- c("Cvs","Alcohol","Malic acid","Ash","Alcalinity of ash", "Magnesium", "Total phenols", "Flavano
ids", "Nonflavanoid phenols", "Proanthocyanins", "Color intensity", "Hue", "OD280/OD315 of diluted wines", "Prolin
e")

# The first column corresponds to the classes
wineClasses <- factor(wine$Cvs)

# Use pairs
pairs(wine[,-1], col = wineClasses, upper.panel = NULL, pch = 16, cex = 0.5)
legend("topright", bty = "n", legend = c("Cv1","Cv2","Cv3"), pch = 16, col = c("black","red","green"),xpd = T, cex
= 2, y.intersp = 0.5)
```



As you can see, we've color coordinated the 3 types of cultivars. It's quite difficult to distinguish meaningful relationships between all of these pairwise combinations in 13 variables. This can easily result in thousands of possible combinations. We don't want to do this manually. As a result, PCA is of great help. Let's continue.

```
winePCA <- prcomp(scale(wine[,-1]))
plot(winePCA$x[,1:2], col = wineClasses)
```



We see a clear separation among cultivar types. We should next see which features contribute the most to this separation and how much variance is each PC is responsible for using pcaMethods.

```
## Bioconductor version 3.5 (BiocInstaller 1.26.1), ?biocLite for help
```

```
## A newer version of Bioconductor is available for this version of R,
##   ?BiocUpgrade for help
```

```
## BioC_mirror: https://bioconductor.org
```

```
## Using Bioconductor 3.5 (BiocInstaller 1.26.1), R 3.4.1 (2017-06-30).
```

```
## Installing package(s) 'pcaMethods'
```

```
##
## The downloaded binary packages are in
##  /var/folders/_6/gk1_r0493c5dp140pkl23k0h0000gn/T//RtmpCcjm86/downloaded_packages
```

```
## Old packages: 'glue', 'lazyeval', 'lubridate'
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

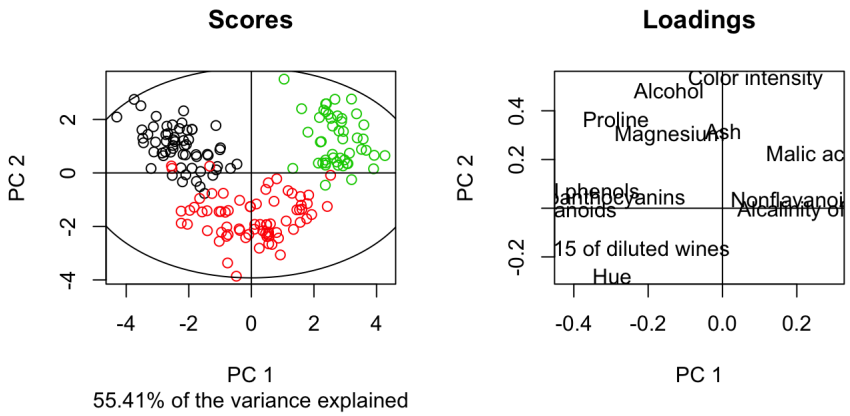
```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, cbind, colMeans,
##     colnames, colSums, do.call, duplicated, eval, evalq, Filter,
##     Find, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, Map, mapply, match, mget, order, paste, pmax,
##     pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce,
##     rowMeans, rownames, rowSums, sapply, setdiff, sort, table,
##     tapply, union, unique, unsplit, which, which.max, which.min
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
##
## Attaching package: 'pcaMethods'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
winePCAmethods <- pca(wine[,-1], scale = "uv", center = T, nPcs = 2, method = "svd")
slplot(winePCAmethods, scoresLoadings = c(T,T), scol = wineClasses)
```



From this plot, we can look at where the colored dots are placed and see which features they most strongly (or not) correlate to. From this plot we can conclude that:

- Wine from Cultivar 2 (marked in red) has a lighter color intensity, lower alcohol %, a greater OD ratio and hue, compared to the wine from Cultivar 1 and Cultivar 3. We can tell this because, for example, the red dots (in the scores chart) are farther away in distance from where the Color Intensity label is in the loadings chart.
- Wine from Cultivar 3 (marked in green) has a higher content of malic acid and non-flavanoid phenols, and a higher alkalinity of ash compared to the wine from Cultivar 1 (marked in black).

```
str(winePCAmethods)
```

```
## Formal class 'pcaRes' [package "pcaMethods"] with 19 slots
##   ..@ completeObs: num [1:178, 1:13] 14.2 13.2 13.2 14.4 13.2 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:13] "Alcohol" "Malic acid" "Ash" "Alcalinity of ash" ...
##   ..@ scores     : num [1:178, 1:2] -3.31 -2.2 -2.51 -3.75 -1.01 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "PC1" "PC2"
##   ..@ loadings   : num [1:13, 1:2] -0.14433 0.24519 0.00205 0.23932 -0.14199 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:13] "Alcohol" "Malic acid" "Ash" "Alcalinity of ash" ...
##   .. .. ..$ : chr [1:2] "PC1" "PC2"
##   ..@ R2cum      : Named num [1:2] 0.362 0.554
##   .. ..- attr(*, "names")= chr [1:2] "PC1" "PC2"
##   ..@ R2         : Named num [1:2] 0.362 0.192
##   .. ..- attr(*, "names")= chr [1:2] "PC1" "PC2"
##   ..@ cvstat     : NULL
##   ..@ sDev       : Named num [1:2] 2.17 1.58
##   .. ..- attr(*, "names")= chr [1:2] "PC1" "PC2"
##   ..@ nObs       : int 178
##   ..@ nVar       : int 13
##   ..@ centered   : logi TRUE
##   ..@ center     : Named num [1:13] 13 2.34 2.37 19.49 99.74 ...
##   .. ..- attr(*, "names")= chr [1:13] "Alcohol" "Malic acid" "Ash" "Alcalinity of ash" ...
##   ..@ subset     : NULL
##   ..@ scaled     : chr "uv"
##   ..@ scale      : Named num [1:13] 0.812 1.117 0.274 3.34 14.282 ...
##   .. ..- attr(*, "names")= chr [1:13] "Alcohol" "Malic acid" "Ash" "Alcalinity of ash" ...
##   ..@ varLimit   : num 1
##   ..@ nPcs       : int 2
##   ..@ method     : chr "svd"
##   ..@ missing    : logi [1:178, 1:13] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:13] "Alcohol" "Malic acid" "Ash" "Alcalinity of ash" ...
##   ..@ network    : NULL
```

```
winePCAmethods@R2
```

```
##     PC1     PC2
## 0.36199 0.19207
```

We can see that PC1 accounts for 36.2% of the variance in the data, and PC2 accounts for 19.2% of the variance in the data. The plot above says that both PC1 and PC2 account for about 54% of the variance in the data.

I've also included my PCA analysis from Homework 3, for more reference.

```
library(readr)    # importing data
library(dplyr)    # data wrangling
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:Biobase':
##
##     combine
```

```
## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

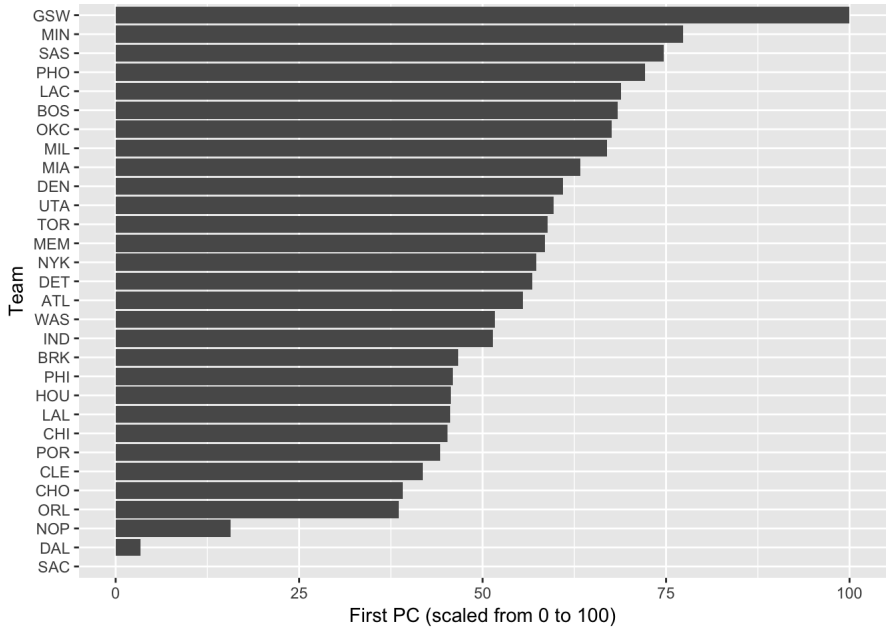
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)  # graphics

#importing the data
dat_roster =read.csv('/Users/jessicacherny/stat133/stat133-hws-fall17/post01/data/nba2017-roster.csv', stringsAsFa
ctors = FALSE)

dat_stats = read.csv('/Users/jessicacherny/stat133/stat133-hws-fall17/post01/data/nba2017-stats.csv', stringsAsFac
tors = FALSE)

#mutating the data
dat_stats <- mutate(dat_stats, missed_fg = field_goals_atts - field_goals_made,
                    missed_ft = points1_atts - points1_made,
                    points = 3*points3_made + 2*points2_made + points1_made,
                    rebounds = def_rebounds + off_rebounds,
                    efficiency = (points + rebounds + steals + blocks - missed_fg -
                                    missed_ft - turnovers) / games_played,
                    points3 = points3_made,
                    points2 = points2_made,
                    points1 = points1_made
                    )

#merging two tables together
merged_tbl = merge(dat_stats, dat_roster)

merged_tbl = merged_tbl %>% select(team, experience, salary, points3, points2, points1, points,
                      off_rebounds, def_rebounds, assists, steals, blocks, turnovers,
                      fouls, efficiency)

merged_tbl$experience[merged_tbl$experience == "R"] <- "0"
merged_tbl$experience <-as.integer(merged_tbl$experience)

#creating a teams table
teams = arrange(
  summarise(
    group_by(merged_tbl, team),
    experience = sum(experience),
    salary_new = sum(salary/ 1000000),
    points3 = sum(points3),
    points2 = sum(points2),
    free_throws = sum(points1),
    points = sum(points),
    off_rebounds = sum(off_rebounds),
    def_rebounds = sum(def_rebounds),
    assists = sum(assists),
    steals = sum(steals),
    blocks = sum(blocks),
    turnovers = sum(turnovers),
    fouls = sum(fouls),
    efficiency = sum(efficiency),
    avg_salary = mean(salary)/1000000)
)

#creating a data frame to perform Principal Component Analysis on
pca_df = teams %>% select(points3, points2, free_throws, off_rebounds, def_rebounds, assists,
                          steals, blocks, turnovers, fouls)
pca = prcomp(pca_df, scale. = TRUE)
pca_1 = pca$x[,1]

prop = round(pca$sdev^2 / sum(pca$sdev^2), 4)
#Createa a data frame with the eigenvalues:
eigs = data.frame(eigenvalue = pca$sdev^2,
  prop_eig = prop,
  cumprop = cumsum(prop)
  )
eigs
```

```
##    eigenvalue prop_eig cumprop
## 1  4.69588631   0.4696  0.4696
## 2  1.70201009   0.1702  0.6398
## 3  0.97952464   0.0980  0.7378
## 4  0.77171938   0.0772  0.8150
## 5  0.53408824   0.0534  0.8684
## 6  0.47801622   0.0478  0.9162
## 7  0.38220374   0.0382  0.9544
## 8  0.26026243   0.0260  0.9804
## 9  0.13359274   0.0134  0.9938
## 10 0.06269622   0.0063  1.0001
```

```
#standardizing the data
s_1 = 100 * (pca_1 - min(pca_1)) / (max(pca_1) - min(pca_1))

team_pca = teams %>% select(team)
team_pca$s1 = s_1
team_pca = arrange(team_pca, s1)
team_pca$num = 1:nrow(teams)
```

```
#plotting the standardized variation of nba teams
ggplot(team_pca, aes(x = reorder(team, num), y = s1)) +
  geom_bar(stat='identity') +
  coord_flip() + xlab("Team") + ylab("First PC (scaled from 0 to 100)")
```



This graph outputs all the normalized PCAs and shows that the Golden State Warriors team has a lot of variability, a lot of teams have the same amount of variability (middle of the pack), while the Dallas team doesn't have a lot of variability.

Discussion

Ultimately, Principal component analysis is used to highlight variation and illuminate strong patterns in a particular dataset. It's much more easy to visualize and explore data when the most important factors and relationships are boiled down and shown out of all the dozens of features that exist in a dataset.

References

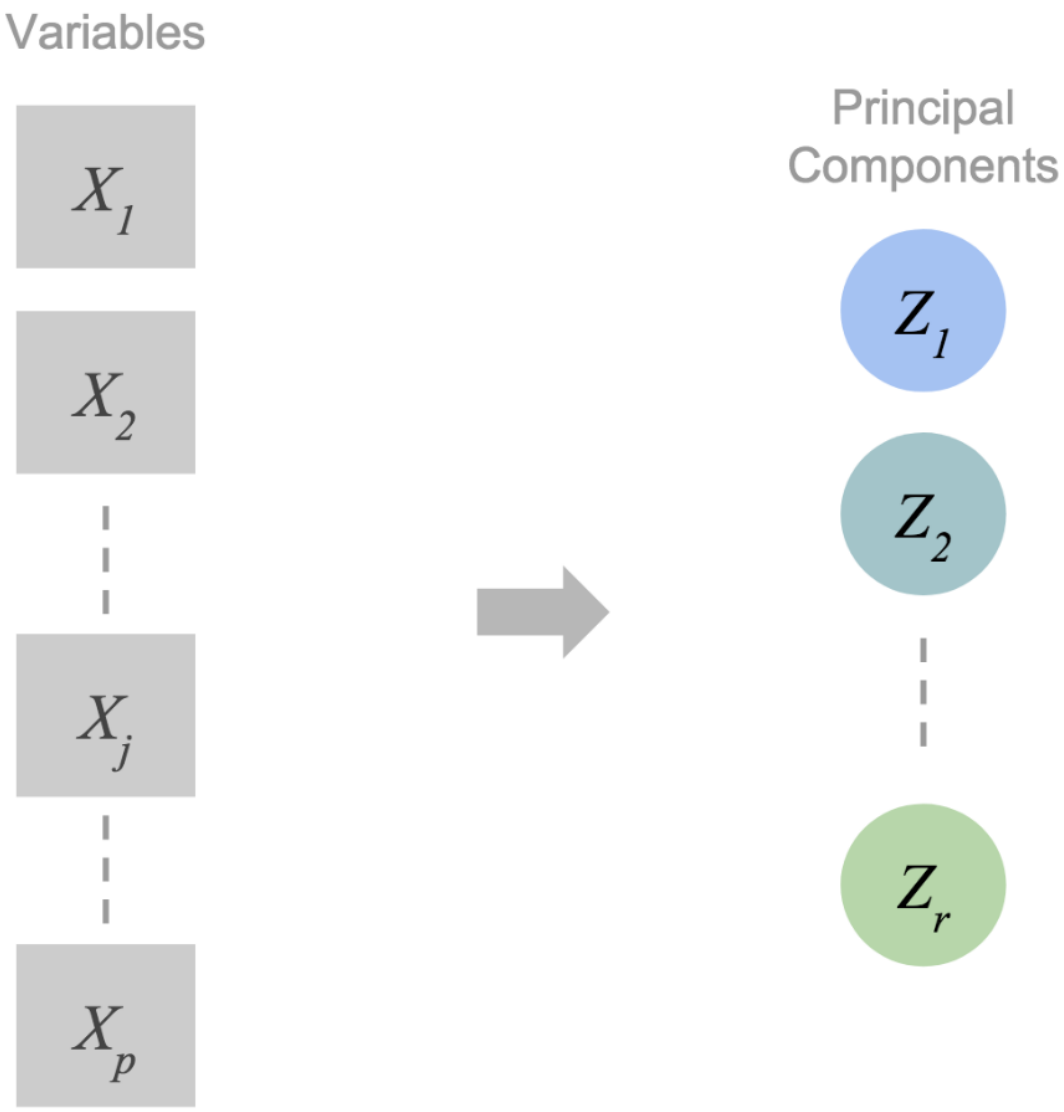
1. https://en.wikipedia.org/wiki/Principal_component_analysis
2. https://onlinecourses.science.psu.edu/stat505/node/49
3. http://setosa.io/ev/principal-component-analysis/
4. https://github.com/ucb-stat133/stat133-fall-2017/blob/master/slides/15-principal-components1.pdf
5. https://www.utdallas.edu/~herve/abdi-awPCA2010.pdf
6. https://www.r-bloggers.com/principal-component-analysis-in-r/
7. https://plot.ly/ipython-notebooks/principal-component-analysis/
8. https://poissonisfish.wordpress.com/2017/01/23/principal-component-analysis-in-r/

Images:

Some diagrams and plots to better understand PCAs.

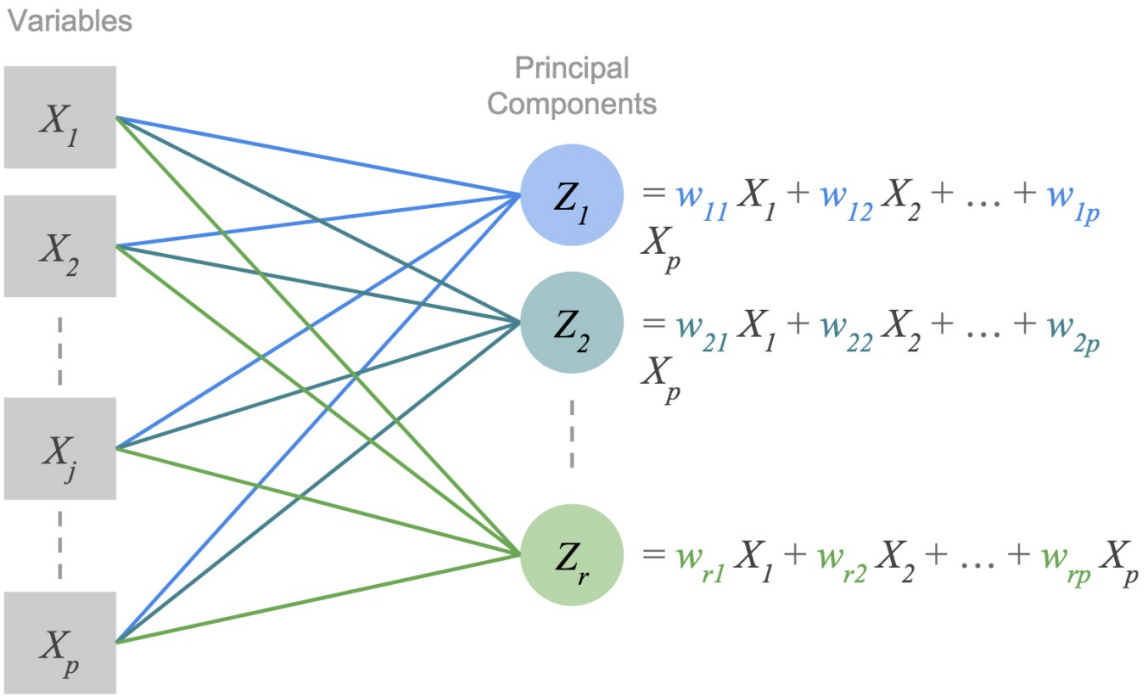
PCA graphic that explains how you boil down variables to a lesser amount of PCAs (condensing variables without losing a lot of information on variability)

Looking for PCs



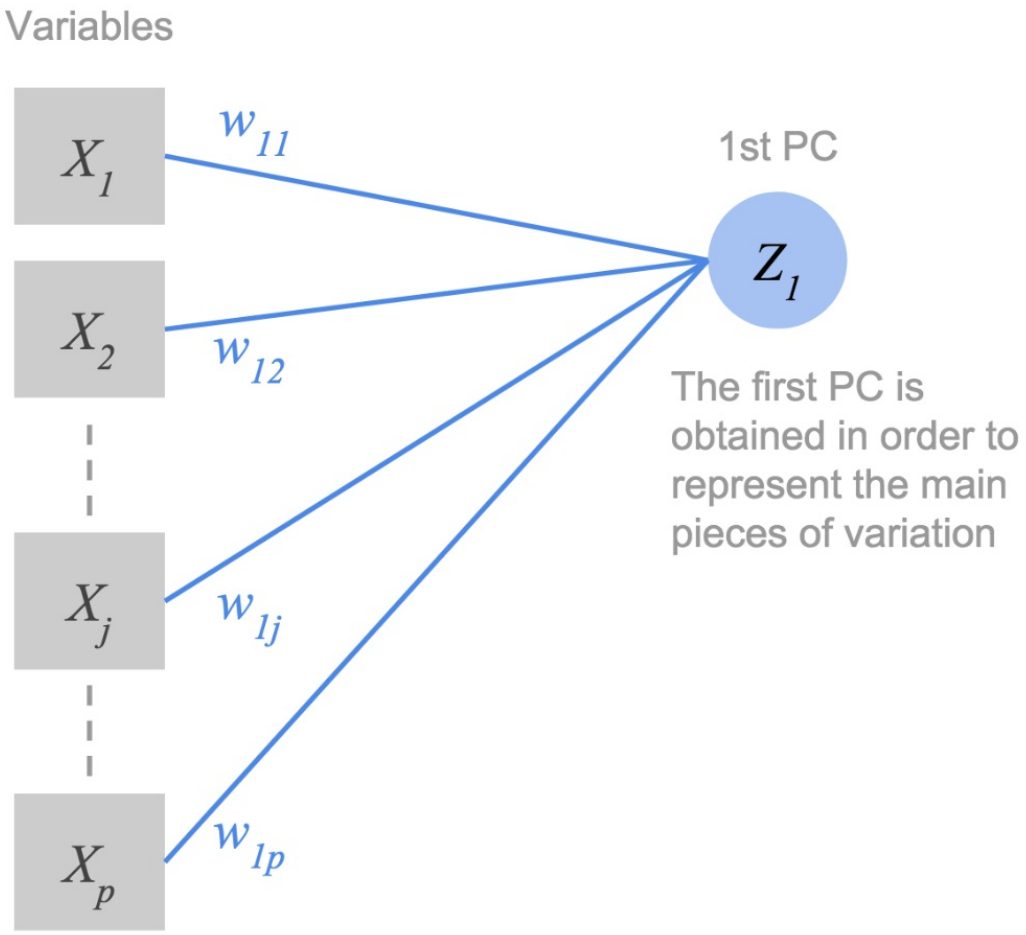
We start out with a lot of variables but want to boil down those variables into fewer variables without losing information.

PCs as linear combinations

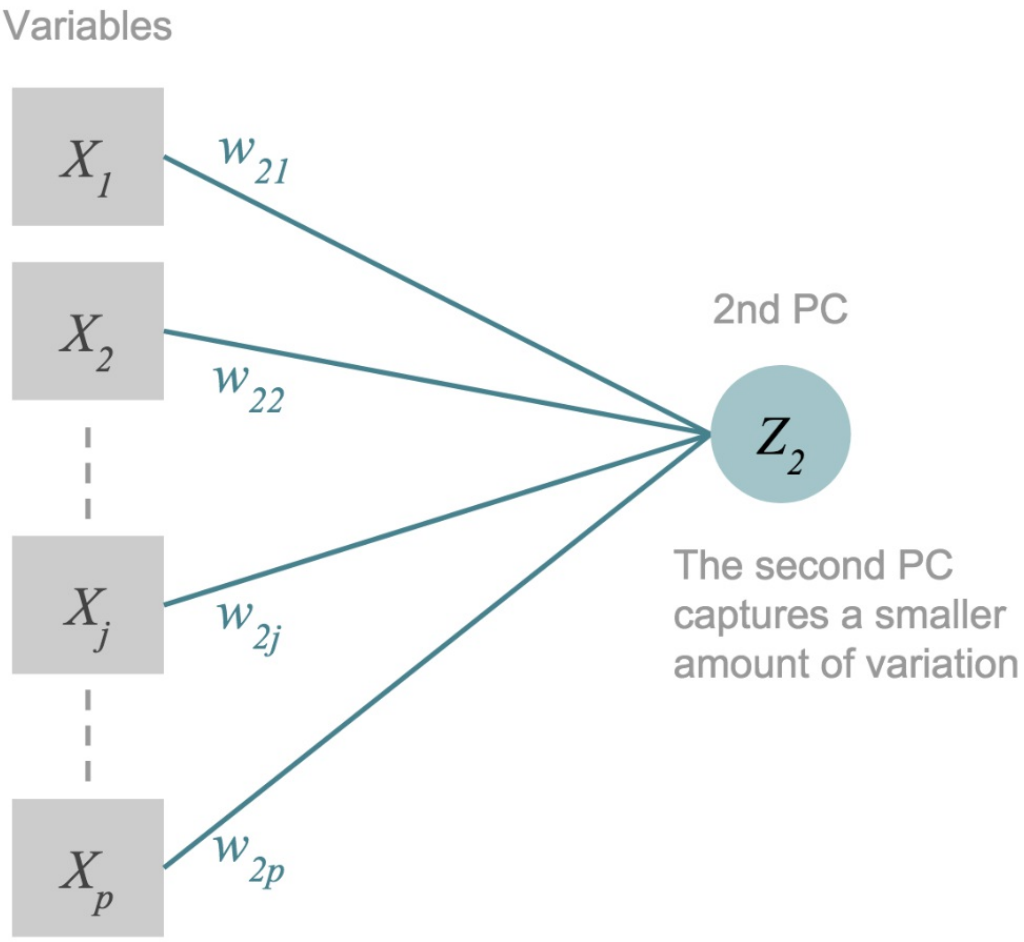


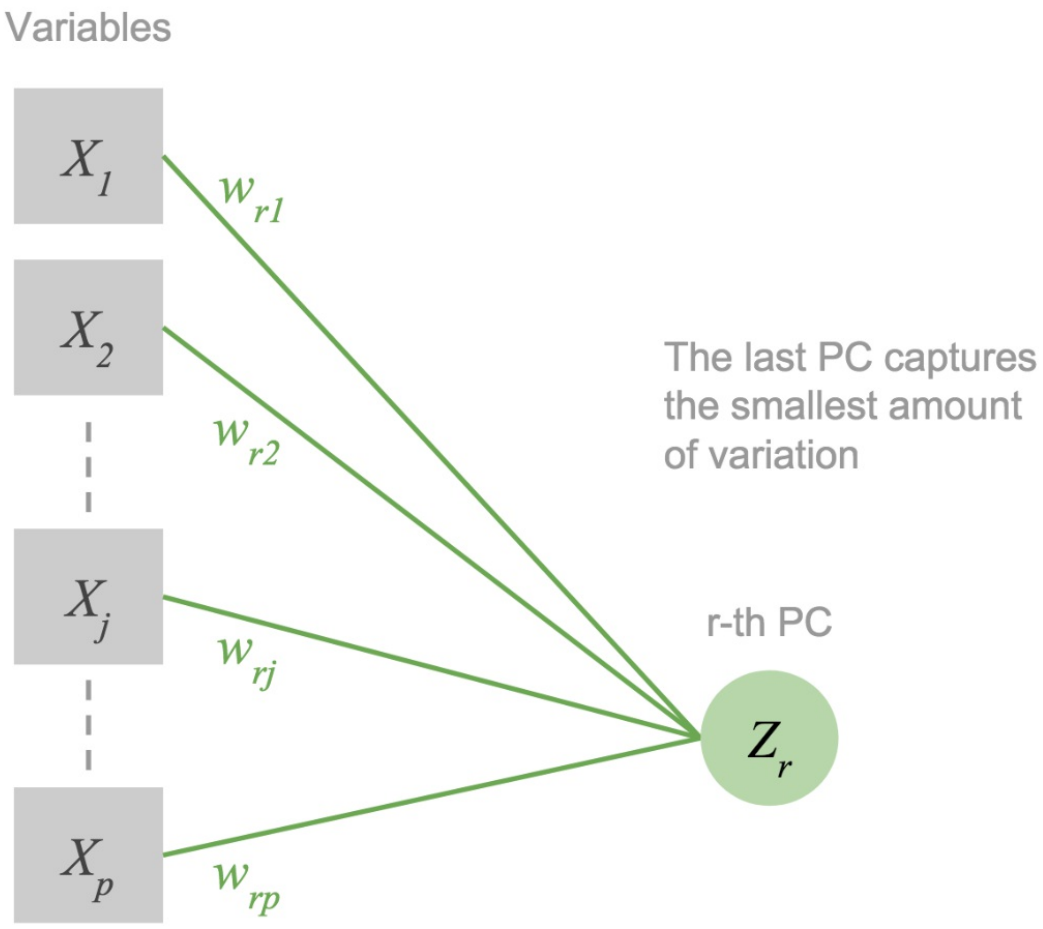
We make linear combinations of our original variables in order to make fewer, more telling variables to reveal deeper insights.

1st PC


Variables
X_1
X_2
X_j
X_p
w_{11}
w_{12}
w_{1j}
w_{1p}
1st PC
Z_1
The first PC is obtained in order to represent the main pieces of variation

2nd PC


Variables
X_1
X_2
X_j
X_p
w_{21}
w_{22}
w_{2j}
w_{2p}
2nd PC
Z_2
The second PC captures a smaller amount of variation



Do this for the “k” amount of new principal components

Conclusion

In essence, the enormous size of data in our contemporary world could lead to huge bottlenecks for performance of many statistical computing and machine learning algorithms. The main goal of a Principal component analysis is to highlight patterns in data, boil down those features that reveal patterns, and detect independent correlation between those features in a data set in order to reveal key insights. PCA saves the day by finding the directions of maximum variance in a high-dimensional data set and reduces it into a smaller dimensional subspace while retaining most of the information.