Post 01 - Centering and Re-scaling Data before Principal Component Analysis

Matthew Sit
October 31, 2017

(Import Statements)

```
library(readr)
```

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

```
# Download a data set containing height and weight measurements for 200 people.
dat_original <- read.csv('data/Davis.csv')

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

```
library(dplyr)
```

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

##
## Attaching package: 'dplyr'

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

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

Introduction

In class, we've been learning about using Principal Component Analysis (PCA) as a technique for summarizing the systematic variation within our data set and reducing its dimensionality. While PCA may be appropriate in many situations, it is important to be aware of how it works mathematically so that we can better understand how our result was computed and what it represents.

Therefore, the purpose of this post is to elaborate upon the necessity of setting the `center=TRUE` and the `scale.=TRUE` parameters in `prcomp()`, R's provided function for performing PCA.

Motivation

While R provides a nice built-in function for performing PCA, `prcomp()`, simply feeding a data set into it and using the result without consideration of the context is dangerous because it may lead to the reporting of non-representative results and unsound conclusions. Without an understanding of the data that is used and the assumptions that were made through PCA, it is possible that you can be mislead into believing in a useless or skewed result.

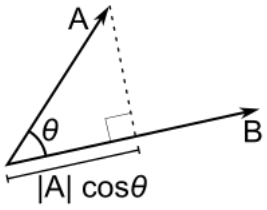
From lecture, we know that `center` and `scale.` should be set to `TRUE`. But why? What happens if we don't? Will this cause us to receive obscure results? In what way?

Background

In Principal Component Analysis, the idea is to reduce the dimensionality of the data. We do this by projecting the data onto a component, such that the direction of the component is chosen so that the variance of the points on the component are maximized. (when this happens, it is not possible to find another direction that we can project on to that will result in greater

variance among our projected data points. This must be true because if it weren't, then we haven't indeed maximized as we had specified.)

If we have the setup below, where *A* is a vector that represents a data point from our input matrix (recall that each data point is composed of several dimensions, for example, each point might have an *x* coordinate and a *y* coordinate), and where *B* is a vector pointing in the direction that we want to project upon.



The projection therefore is found by dropping a perpendicular from *A* down to *B*. This projection can be expressed as $A^\top B$. This make sense because in this expression, we are trying to express the data point *A* using only the bases provided by *B*, thus giving us a projection of *A* on to the subspace spanned by *B*'s bases.

So now if we stack up all of our input data points into a data matrix *X*, where each row represents a data point and each of the columns represents each of the dimensions corresponding to each data point, and if we express the vector specifying the direction we are choosing as *u*, then we can express the goal of PCA mathematically as:

$u = \text{argmax}_{\|u\|_2 = 1} \text{var}(Xu)$

Basically, this means that we want to maximize the variance of the data points *X* when they are projected on to a direction *u*. We enforce the constrain that *u* must be of length 1 (in other words, it is a unit vector). Once we have maximized it, we return the argument *u* that gave us this maximum value.

I now follow to prove what was suggested in the lecture slides--that the solution to this equation is dependent upon the eigenvalues of the matrix $X^\top X$.

Assuming that our input data is centered around zero (it has zero-mean) (recall that you can specify a parameter to the function in R to do this for you), we have...

$u = \text{argmax}_{\|u\|_2 = 1} E[(Xu)^2]$

Expectation is the same as taking average over our data points, which is the same as a summation over all our data points divided by the number of data points we have. We can perform these summations in parallel by vectorizing it into a matrix and taking advantage of how matrix multiplication works in parallel. (We can drop the "divide by n" term since it is a sacalar that applies to all cases and so it does not change which *u* it is that gives us the maximum value.)

$u = \text{argmax}_{\|u\|_2 = 1} \|Xu\|_2^2$

We can expand this to...

$u = \text{argmax}_{\|u\|_2 = 1} u^\top X^\top X u$

Using the spectral theorem (which is a special kind of matrix decomposition that applies for symmetric, square matrices), we can decompose $X^\top X$ into $V\Lambda V^\top$, where *V* are orthnormal eigenvectors and Λ is a diagonal matrix of the corresponding eigenvalues.

$u = \text{argmax}_{\|u\|_2 = 1} u^\top V \Lambda V^\top u$

From here, it is evident that the direction returned by principal component analysis is dependent upon the eigenvalues of the input matrix. Thus, different scalings of the columns of our input data matrix will affect its eigenvalues and will in turn affect the result of PCA. Therefore, before PCA is performed, we must standardize our data by zero-centering it as assumed in this derivation and by setting the variance to be 1 (which is called whitening). By whitening our data, we ensure that we receive the same results regardless of the scalings of the dimensions of our input points because we always standardize the scalings before performing PCA if we set `scale.=TRUE`.

Examples and Discussion

In the background section, I mentioned two important parameters to enable when using R's `prcomp()`. First, we must zero-mean our data (basically subtracting out the mean of our data set from all values so that the new mean of our data set is now 0). Second, we must whiten our data (scaling our data set so that the variance of our data set becomes 1). In the

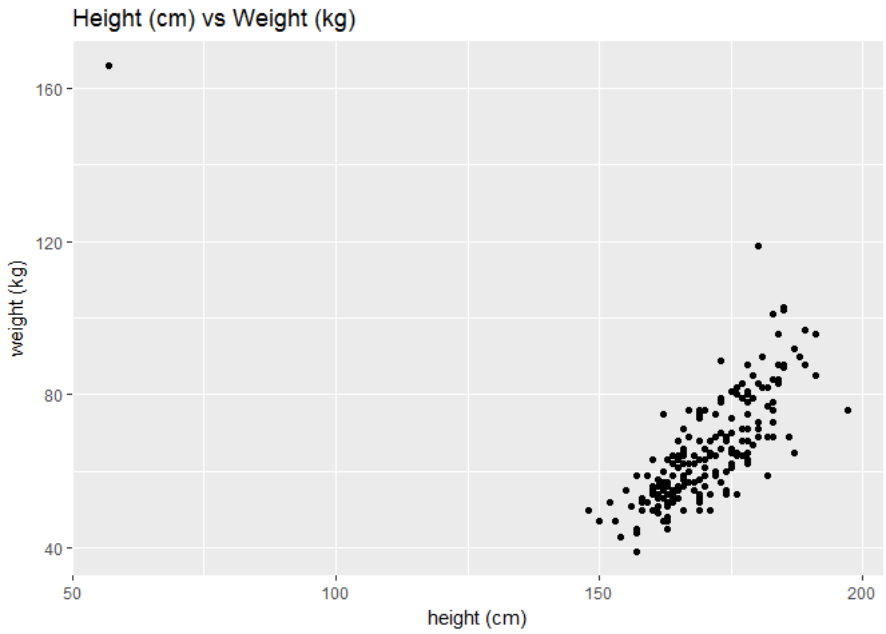
examples below, I demonstrate what happens when we do and do not adhere to these guidelines.

Our data set

To demonstrate, here I have a data set that has the heights (in centimeters) and weights (in kilograms) of 200 people.

```
# Extract height and weight columns from the raw data set obtained from online.
dat <- dat_original %>%
  select(height, weight)

# Plot height (cm) vs weight (kg).
ggplot(dat, aes(x=height,y=weight)) +
  geom_point() +
  labs(title='Height (cm) vs Weight (kg)',x='height (cm)',y='weight (kg)')
```



Above is a plot so that we can observe the general shape of the scatter points representing our input data. We can see that there is a positive correlation in the data. Take note of the shape of the point cloud here because we will compare it to a slightly different version in the next plot.

The eigenvalues with `scale.=TRUE` are:

```
# Compute PCA with scaling.
pca_cm_kg <- prcomp(dat, scale.=TRUE)
pca_cm_kg$sdev^2
```

```
## [1] 1.1896496 0.8103504
```

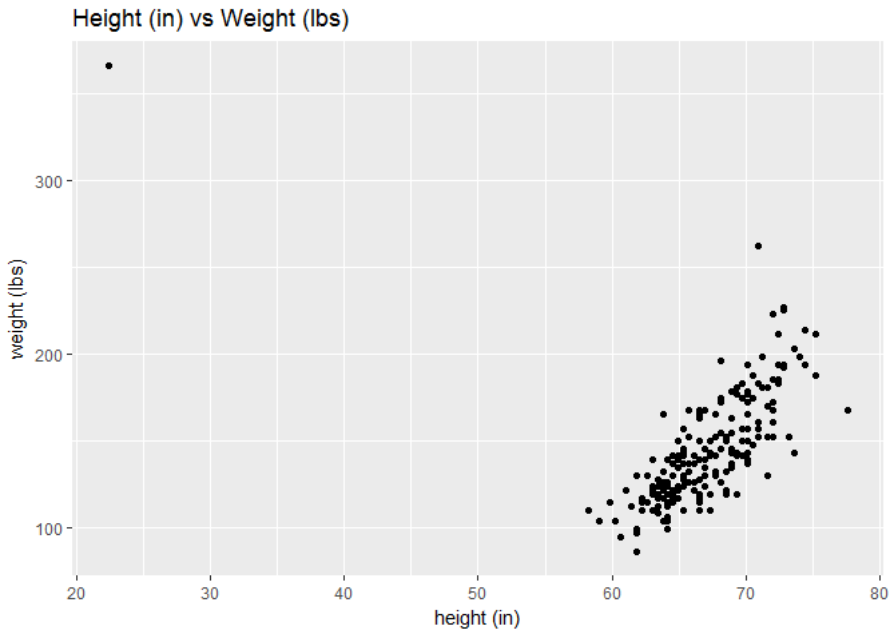
Scaling our data set

Now if we scale our inputs, for example, by converting the height from centimeters to inches and weight from kilograms to pounds...

```
# Extract height and weight from raw data set, coverting to (in) and (lbs) respectively.
dat <- dat_original %>%
  select(height, weight) %>%
  mutate(height=height*0.393700787, weight=weight*2.20462)
```

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

```
# Plot height (in) vs weight (lbs).
ggplot(dat, aes(x=height,y=weight)) +
  geom_point() +
  labs(title='Height (in) vs Weight (lbs)',x='height (in)',y='weight (lbs)')
```



From this graph, we notice that the scaling of the graph's axes have changed but that the shape of the scatter cloud is the same as before. This makes sense because the inherent information contained by this data is the same, it is just being expressed in a different scaling.

Using `scale.=TRUE` we see that the result of PCA is the same even though we've scaled our data differently now:

```
# Compute PCA with scaling.
pca_in_lbs <- prcomp(dat, scale.=TRUE)
pca_in_lbs$sdev^2
```

```
## [1] 1.1896496 0.8103504
```

Not scaling our data

However, we see that if we now perform PCA using the two versions of the data but without using `scale.=TRUE`, our eigenvalues differ, as suggested by our derivation.

```
# Extract height and weight columns from the raw data set obtained from online.
dat <- dat_original %>%
  select(height, weight)

# Compute PCA without scaling.
pca_cm_kg <- prcomp(dat, scale.=FALSE)
pca_cm_kg$sdev^2
```

```
## [1] 240.1712 131.8787
```

```
# Extract height and weight from raw data set, coverting to (in) and (lbs) respectively.
dat <- dat_original %>%
  select(height, weight) %>%
  mutate(height=height*0.393700787, weight=weight*2.20462)

# Compute PCA without scaling.
pca_in_lbs <- prcomp(dat, scale.=FALSE)
pca_in_lbs$sdev^2
```

```
## [1] 1108.2956   21.5298
```

This is not desired because as we saw from the scatterplots, we would want these results to be the same, regardless of the units used in our measurements/input data.

Zero-meaning vs. not zero-meaning

Finally, let us now use one of the versions of the data and see that the eigenvalues differ depending on whether we zero-mean our data or not before performing PCA. (To keep these comparisons consistent, we will scale for both examples below.)

```
# Extract height and weight columns from the raw data set obtained from online.
dat <- dat_original %>%
  select(height, weight)

# Compute PCA with centering.
pca_cm_kg <- prcomp(dat, scale.=TRUE, center=TRUE)
pca_cm_kg$sdev^2
```

```
## [1] 1.1896496 0.8103504
```

```
# Extract height and weight columns from the raw data set obtained from online.
dat <- dat_original %>%
  select(height, weight)

# Compute PCA without centering.
pca_cm_kg <- prcomp(dat, scale.=TRUE, center=FALSE)
pca_cm_kg$sdev^2
```

```
## [1] 1.97536594 0.02463406
```

We again observe that there is a difference. In general, we want to zero-mean our data so that we can better focus upon extracting out useful information from the range it covers, rather than going with a biased result due to its shift from the origin.

Conclusions

Principal component analysis can be extremeley useful in helping us analyze our data by showing us what the biggest differences are. It does this by identifying the linear combination of the input features that maximizes the variance.

When we use `prcomp()` in R to compute PCA, we should be sure to enable `center=TRUE` and `scale.=TRUE`. This way, our PCA result is not impacted by different shifts and different scalings of data that is inherently the same. We demonstrate what happens when we do not; we observe non-matching eigenvalues and we also have shown that this will change the unit vector projection direction that is chosen.

The final take-home message is that we should ensure that `center=TRUE` and `scale.=TRUE` when we call `prcomp()`. Note that `scale.` is by default set to `FALSE` in R (due to historical/legacy reasons from S).

References

- Principal Component Analysis (part I), Lecture 15 Slides, Stat 133 Fall 2017, Gaston Sanchez, UC Berkeley, https://github.com/ucb-stat133/stat133-fall-2017/blob/master/slides/15-principal-components1.pdf.
- PCA/CCA, Notes 9 and 10, CS 189 Fall 2017, UC Berkeley, http://www.eecs189.org/static/notes/n9.pdf and http://www.eecs189.org/static/notes/n10.pdf.
- Self-Reports of Height and Weight, R datasets, Davis, https://vincentarelbundock.github.io/Rdatasets/datasets.html.
- Dot Product, Wikipedia, https://en.wikipedia.org/wiki/Dot_product.
- Projection (linear algebra), Wikipedia, https://en.wikipedia.org/wiki/Projection_(linear_algebra)
- The Idea of Making the Data have a Zero-Mean, Stats Stack Exchange, https://stats.stackexchange.com/questions/104528/the-idea-of-making-the-data-have-a-zero-mean
- Normalizing Data, Herve Abdi, The University of Texas at Dallas, https://www.utdallas.edu/~herve/abdi-Normalizing2010-pretty.pdf
- Principal Components Analysis, R Documentation, https://stat.ethz.ch/R-manual/R-devel/library/stats/html/prcomp.html