Principal Component Analysis

Modern Data Mining

Contents

Contents

Objectives

Massive data is easily available to us. How can we efficiently extract important information from a large number of features or variables which will possess the following nice properties?

1) **Dimension reduction/noise reduction**: They are "close" to the original variables but only with a few newly formed variables.
2) **Grouping variables/subjects efficiently**: They will reveal insightful grouping structures.
3) **Visualization**: We can display high dimensional data.

Principal Component Analysis is a powerful method to extract low dimension variables. One may search among all linear combinations of the original variables and find a few of them to achieve the three goals above. Each newly formed variable is called a Principal Component. PCA is closely related to Singular Value Decomposition (SVD). Both PCA and Singular Value Decomposition are successfully applied in many fields such as face recognition, recommendation system, text mining, Gene array analyses among others. PCA is unsupervised learning. There will be no responses. It works well in clustering analyses. In addition, PCs can be used as input in supervised learning as well.

In this lecture, we analyze ASVAB tests (Armed Services Vocational Aptitude Battery) using PCA to see how different people are. In addition, PCA on test scores also reveals difference between males and females: while the total test scores are similar, females are strong in intelligence and males demonstrate better dexterity skills.

We will also apply SVD to build a recommender system based on the existing ratings over a large number of movies.

PCA: Principal Component Analysis

- Dimension reduction
  - capture the main features
  - reduce the noise hidden in the data
  - visualization of large dimension
- PC's interpretations
  - The best low dimension of linear approximation to the data (or closest to the data)
  - The direction of linear combination which has largest variance
  - We may take a small number of PCs as a set of input to other analyses

Outline

1. Case Study: ASVAB tests (Armed Services Vocational Aptitude Battery)
2. PCA
    - PC scores
    - PC loadings
    - PVE: determine the number of PCs
    - Biplot: display the data
    - `prcomp()`: PCA function
3. Appendices:
    - Appendix 1: formal definition of PCs
    - Appendix 2: PCA and Eigen decomposition of Correlation matrix
    - Appendix 3: PCs and SVD
    - Appendix 4: Missing values/recommender system
4. Data needed:

- `IQ.Full.csv`
- `MovieLens.csv`

5. R functions

- `prcomp()`: PCA function
- `svd()`: SVD function
- `factoextra` package: visualization of PCA results
- `softImpute` package: SVD with missing values

1 Case Study: How people differ in intelligence?

1.1 Background about ASVAB/AFQT

ASVAB (Armed Services Vocational Aptitude Battery) tests have been used as a screening test for those who want to join the army or other jobs. It helps to determine which army jobs are appropriate for applicants. No preparation is needed.

ASVAB has the following components:

- 10 tests: `Science`, `Arith` (Arithmetic reasoning), `Word` (Word knowledge), `Parag` (Paragraph comprehension), `Numer` (Numerical operation), `Coding` (Coding speed), `Auto` (Automotive and Shop information), `Math` (Math knowledge), `Mechanic` (Mechanic Comprehension) and `Elec` (Electronic information).
- AFQT (Armed Forces Qualifying Test) is a combination of Word, Parag, Math and Arith.
- Based on AFQT, one may qualify for service branch requirement: Air Force: 36, Army: 31, Marines: 32, Coast Guard: 40, Navy: 35 (out of 100 which is the max!)

Our goal:

- How can we summarize the set of tests and grab main information about each one's intelligence/abilities efficiently?
- How AFQT is formed?

NLSY79 study and data:

Data `IQ.Full.csv` is a subset of individuals from the 1979 National Longitudinal Study of Youth (NLSY79) survey who were re-interviewed in 2006. Information on family, personal demographic such as gender, race and education level, plus a set of ASVAB (Armed Services Vocational Aptitude Battery) test scores, taken

in 1981, are available. In addition, a set of self-evaluated self-esteem scores and income in 2005 are also included in this dataset. The data is very interesting on its own.

Note: One of the original study goals is to see how intelligence relates to one's future successes measured by income in 2005 and self-esteem scores.

Newer ASVAB data: Do you have any more recent data related to ASVAB?

(`NLSY79.csv` contains more information and it will be used later in the course.)

2 Data Prep and EDA

Get a quick look at the data

```
data.full <- read.csv("IQ.Full.csv")
dim(data.full) #str(data.full), summary(data.full)
```

```
## [1] 2584   32
```

```
names(data.full)
```

```
##  [1] "Subject"        "Imagazine"      "Inewspaper"     "Ilibrary"
##  [5] "MotherEd"       "FatherEd"       "FamilyIncome78" "Race"
##  [9] "Gender"         "Educ"           "Science"        "Arith"
## [13] "Word"           "Parag"          "Numer"          "Coding"
## [17] "Auto"           "Math"           "Mechanic"       "Elec"
## [21] "AFQT"           "Income2005"     "Esteem1"        "Esteem2"
## [25] "Esteem3"        "Esteem4"        "Esteem5"        "Esteem6"
## [29] "Esteem7"        "Esteem8"        "Esteem9"        "Esteem10"
```

There are 32 variables with 2,584 subjects/people. Many variables are coded as numeric but categorical by nature. For example, `Imagazine`, `Inewspaper` and `Ilibrary` have `yes`, `no` as values. There seems to be no missing values.

Our focus lies on analyzing `ASVAB` scores. We defer interesting EDA's.

3 AFQT tests: `Word`, `Math`, `Parag` and `Arith`

As one important summary of the `ASVAB` scores, AFQT scores combining with `Word`, `Math`, `Parag` and `Arith` are also reported for each test taker.

Question:

i) How best can we **capture the performance** using one or two scores based on the four tests?
ii) Can we separate people with good language skills or math skills?
iii) How is AFQT calculated? Is it merely the total scores of the four tests?

Note:

This is similar to the creation of SP500, a weighted index based on 500 stocks.

A subset: For simplicity we take a subset of 50 subjects and only include the `AFQT` and the four tests associated with it.

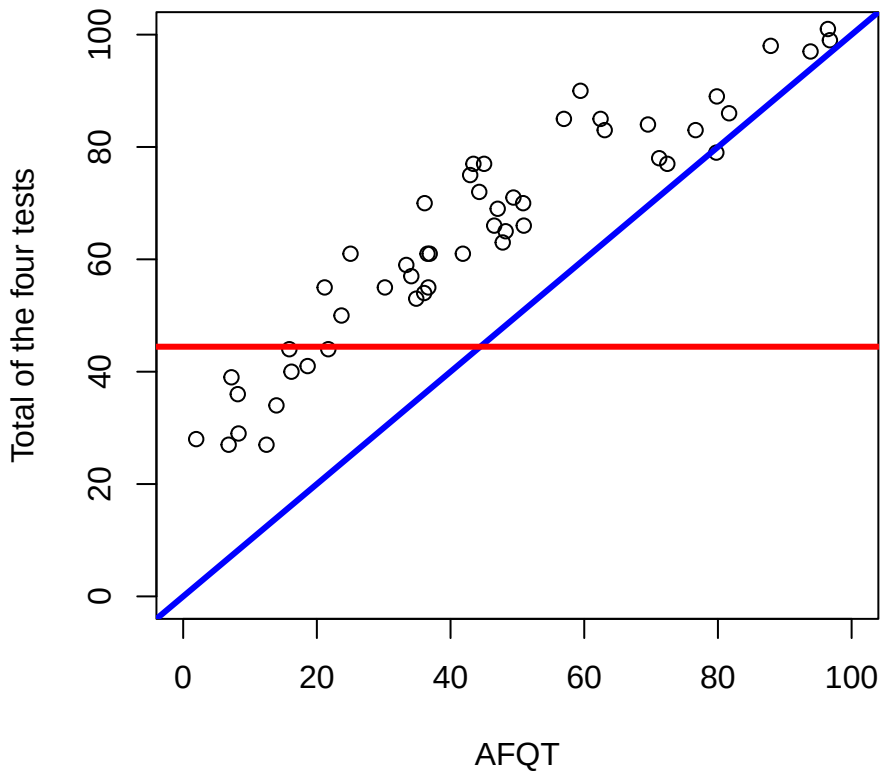
```
cols <- c("Subject", "Word", "Parag", "Math", "Arith", "AFQT", "Gender")
AFQT.full <- data.full[, cols]
# full data set of the AFQT scores, AFQT and Gender
set.seed(10) # make sure the same subset is generated each time
AFQT.sub <- AFQT.full[sample(nrow(data.full), 50, replace=FALSE), cols]
## dplyr way:
```

```r
# set.seed(10)
# AFQT.sub <- data.full %>% sample_n(50, replace = F) %>%
#   select(Subject,Word, Parag, Math, Arith, AFQT)
str(AFQT.sub) # hist(AFQT.sub$Word)
summary.AFQT <- skim(AFQT.sub)
names(summary.AFQT) # see skim's output
summary.AFQT %>% select(skim_variable, numeric.mean, numeric.sd, numeric.hist)
# only need mean, sd and hist
```

The four tests have different means and different standard deviations.

Is `AFQT` the sum of the four test scores? Not really!But they are highly correlated.

```r
plot(x = AFQT.sub$AFQT,
     y = rowSums(AFQT.sub %>% select(Word, Parag, Math, Arith)),
     xlab = "AFQT",
     ylab="Total of the four tests",
     xlim = c(0, 100),
    ylim = c(0, 100))
abline(a = 0, b =1, col = "blue", lwd =3) # a=intercept, b=slope
abline(h = mean(AFQT.sub$AFQT), col = "red", lwd =3) # a horizontal line of y= sample mean
```




```r
#cor(AFQT.sub$AFQT, rowSums(AFQT.sub %>% select(Word, Parag, Math, Arith)))
```

For simplicity we give names for each person in the subset.

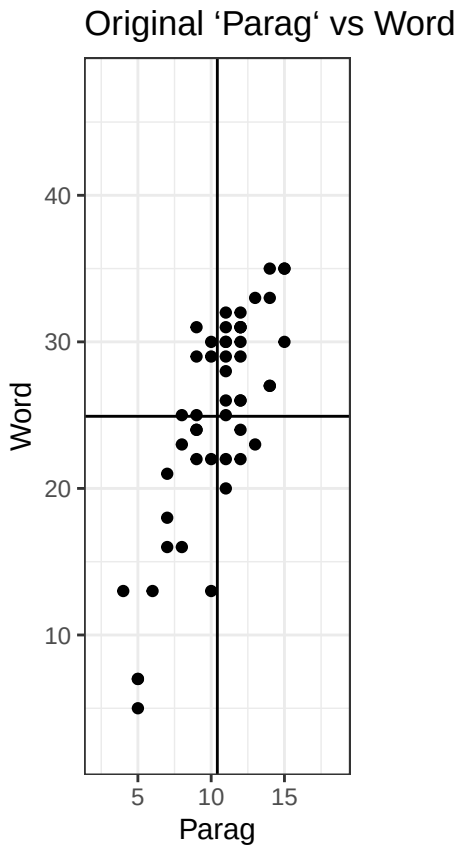
```
rownames(AFQT.sub)  # label for each person
rownames(AFQT.sub) <-  paste("p", seq(1:nrow(AFQT.sub)), sep="")
# reassign everyone's labels to be shorter.
rownames(AFQT.sub)
```

3.1 Motivations/Interpretations of PCA

3.1.1 PCA for only two tests

Let us focus on `Word` and `Parag` first by starting with a scatter plot of the original `Parag` and `Word` scores.

```
parag.word  <- as.data.frame(AFQT.sub[, c("Parag", "Word")])

mu_parag <- mean(parag.word$Parag, na.rm = TRUE)
sd_parag <- sd(parag.word$Parag,   na.rm = TRUE)

mu_word  <- mean(parag.word$Word,  na.rm = TRUE)
sd_word  <- sd(parag.word$Word,    na.rm = TRUE)

parag.word %>%
  ggplot(aes(x = Parag, y = Word)) +
  geom_point() +
  geom_vline(xintercept = mu_parag) +
  geom_hline(yintercept = mu_word) +
  theme_bw() +
  coord_fixed(
    ratio = 1,
    xlim = c(mu_parag - 3*sd_parag, mu_parag + 3*sd_parag),
    ylim = c(mu_word  - 3*sd_word,  mu_word  + 3*sd_word)
  ) +
  ggtitle("Original `Parag` vs Word")
```



How can we grab some main information about each person's language skills using one score or two scores based on the two tests `Word` and `Parag`?

We want to use one aggregated score or weighted sum of two scores with the following desirable properties:

1. The new score is weighted sum of `Word` and `Parag`, i.e., a linear combination of the two. We denote this by $Z_1 = \phi_{11}\text{Word} + \phi_{21}\text{Parag}$
2. The two scores `Word` and `Parag` should be "close" to the newly formed one score Z_1.

In other words, we are looking for a line, with a direction by (ϕ_{11}, ϕ_{21}), going through the cloud of the scatter plot of `Word` vs. `Parag` with minimum overall perpendicular distance. The projection of each point (Parag, Word), i.e., the weighted sum of the two scores Z_1 is called a Principal Component or a PC.

In the following sections, we give both informal and formal definitions of PCs and also show how to find the `weights` or `loadings`, (ϕ_{11}, ϕ_{21}) and the PC scores $Z_1 = \phi_{11}\text{Word} + \phi_{21}\text{Parag}$. We also define other PC's.

3.1.2 Geometric interpretations

The crux of PCA can be captured simply by a plot. Focus on the plots in this section. (Codes are hidden on propose. **YOU DON'T NEED TO KNOW THE CODES used to produce plots in this section!!!!**)

To demonstrate what are PCs and the geometric properties of PCAs, let us look at the scatter plots with centered `Word` and `Parag` scores. i.e., we subtract `Word` and `Parag` by its mean, respectively. We call this process centering the data. Positive centered score implies the raw score is above the mean and below the mean if it is negative.

In the following R-chunk, we first center the two scores then make a scatter plot.

```
parag.word.centered <- AFQT.sub %>% select(Word, Parag) %>%
  mutate(word_centered = `Word` - mean(Word),
```

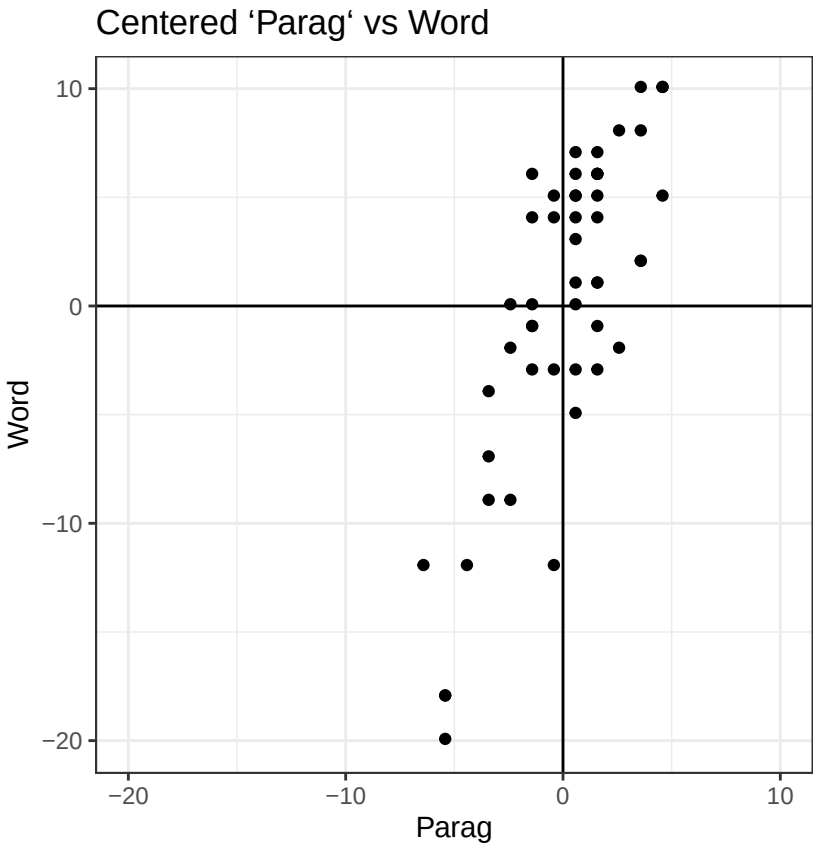
```
         parag_centered = `Parag` - mean(Parag))
# Making word and parag by centering each score.

# or use scale() to center the data
parag.word.centered <- scale(AFQT.sub[, c("Parag", "Word")],
                                          center = T, scale = F)
parag.word.centered <- as.data.frame(parag.word.centered)
# make centered data as a data frame
```

Notice the original and centered data only differ by the mean values while keep the same standard deviations.

```
round(sapply(parag.word.centered,mean), 3) # col mean with 3 decimals
sapply(AFQT.sub %>% select(Word, Parag), mean) # col mean
sapply(parag.word.centered,sd) #col sd
sapply(AFQT.sub %>% select( Parag, Word), sd) # col sd
```

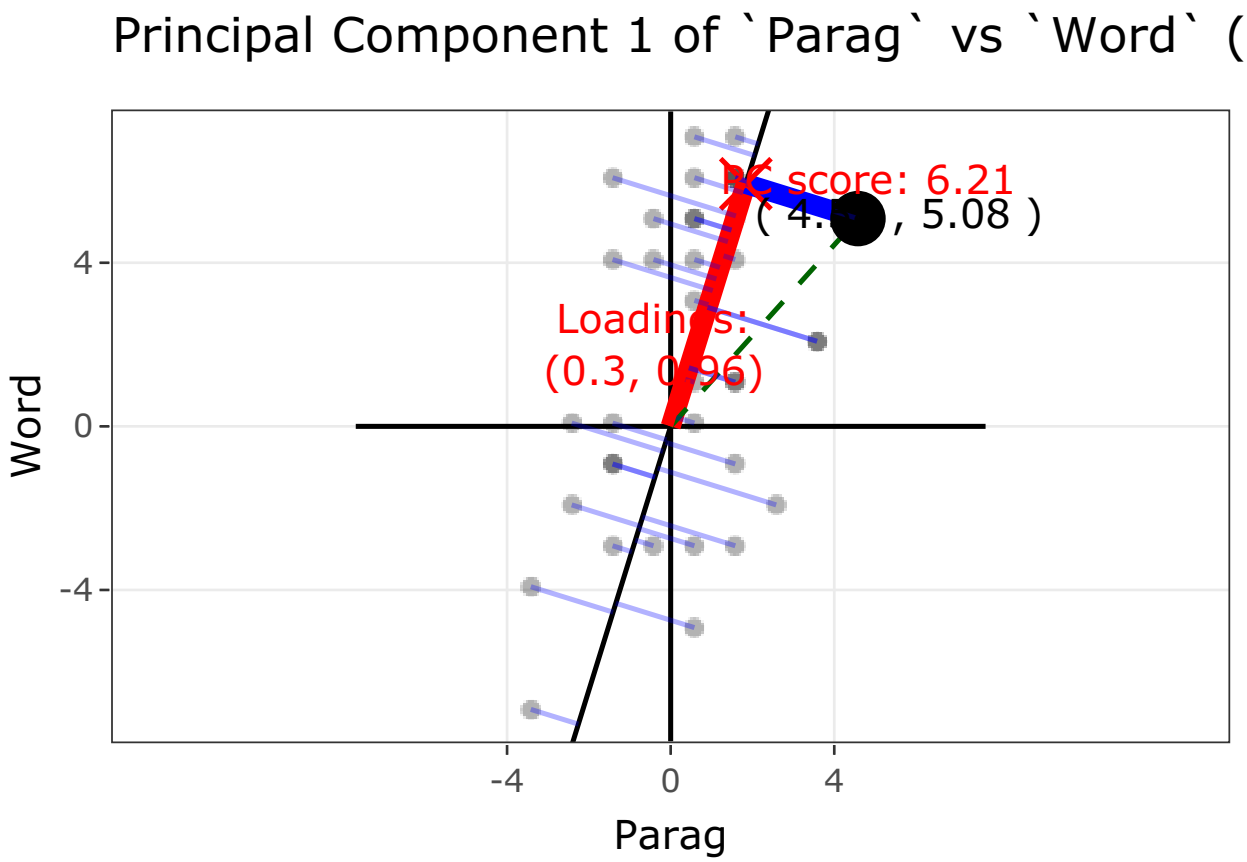
Look at the scatter plot of centered `Word` and `Parag`:

```
parag.word.centered %>%
  ggplot(aes(x = Parag, y = Word)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  theme_bw() +
  coord_fixed(ratio = 1, xlim = c(-20, 10), ylim = c(-20, 10)) +
  ggtitle("Centered `Parag` vs Word")
```



The following scatter plot of `Word` vs. `Parag` illustrates what is a desirable linear line we are looking for: **the total squared perpendicular distance from each point to the line should be minimized.**

No need to go through the chunks below. Focus on the plot.



PC1, first principal component: the linear combination of the two scores which minimizes the total squared perpendicular distance. That line is described by (`procom` function is used to produce all relevant quantities.)

- PC1 loadings: $(0.3, 0.96)$ which describes the direction of the line.
- PC1 scores: the projection score $0.3 \times$ **Parag_centered** $+ 0.96 \times$ **Word_centered**.

As an example, for the person with `Parag_centered` $= 4.58$ and `Word_centered` $= 5.08$, the PC score is $.3 \times 4.58 + .96 \times 5.08 = 6.25$ (It should be 6.21 as marked in the plot above due to rounding error.)

How much information lost using PC1?

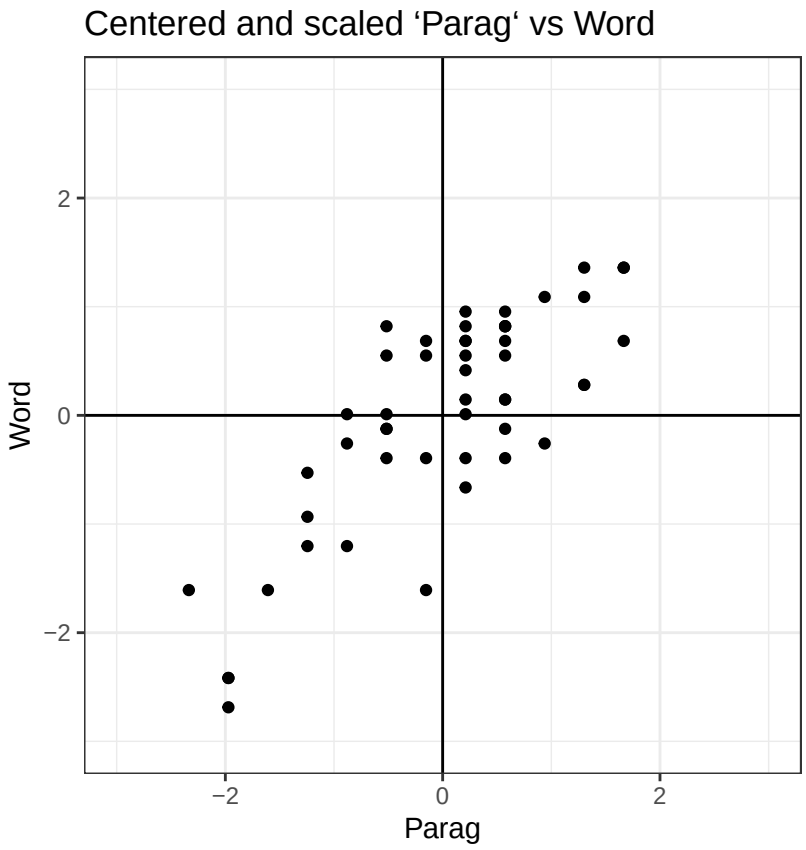
Instead of using `Word` and `Parag` we only use PC1. We will lose on average mean sum of squared distances.

3.1.3 Two defintions or interpretations of PCA

The above PC scores may have one problem: the two tests have different spread or standard deviation. Often we may want to find PCs among totally different variables with different units. In this case, it is a good idea to center and scale the data, by subtracting the mean and dividing the standard deviation for each test first, before performing PCA. This can be done using `scale()`.

```
parag.word.scaled.centered  <- as.data.frame(scale(AFQT.sub[, c("Parag", "Word")],
                                                 center = T, scale = T))
parag.word.scaled.centered %>%
  ggplot(aes(x = Parag, y = Word)) +
  geom_point() +
```

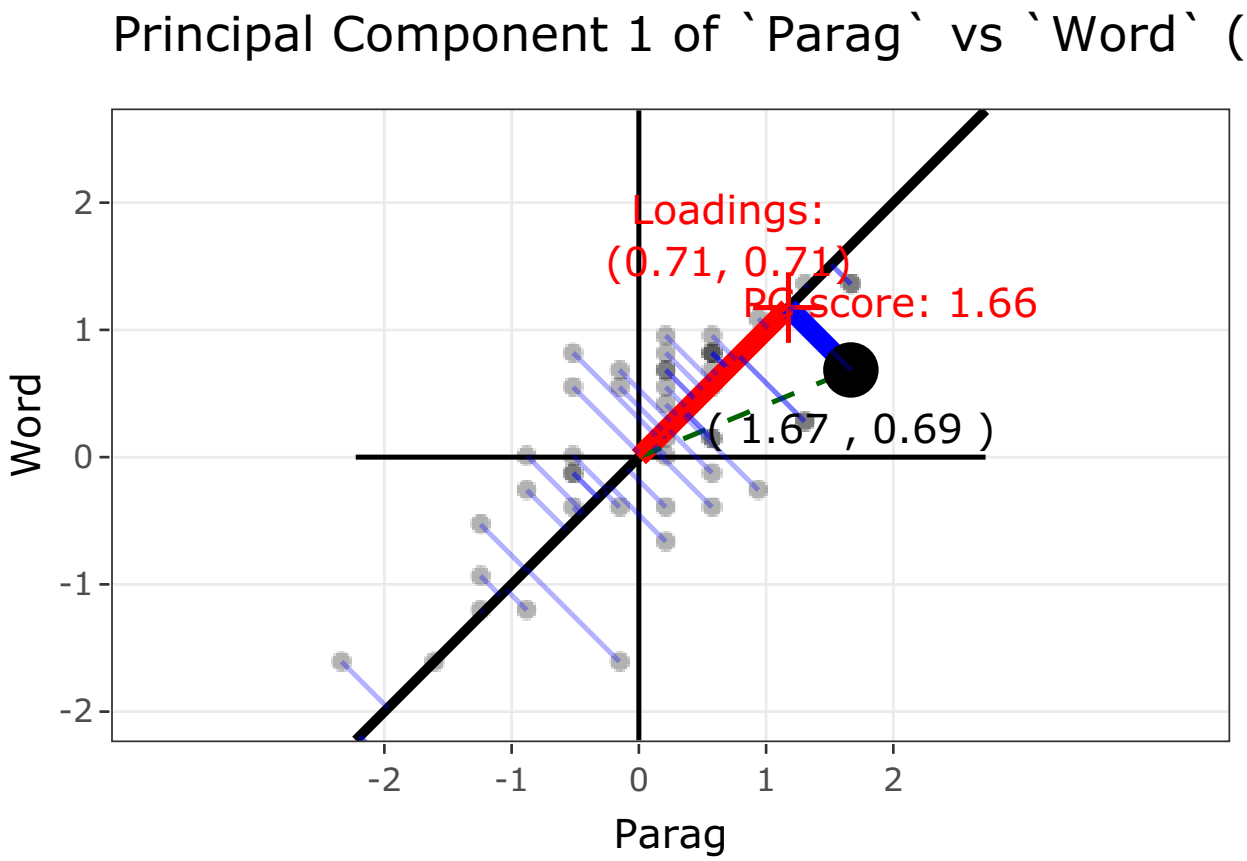
```
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  theme_bw() +
  coord_fixed(ratio = 1, xlim = c(-3, 3), ylim = c(-3, 3)) +
  ggtitle("Centered and scaled `Parag` vs Word")
```



The following plot illustrates the relationship among three metrics:

- Total sum of squares
- Sum of squared errors
- Variance of PC1

Skip the codes but concentrating on the plot please:



In the above plot we want to demonstrate the following beautiful geometric interpretation of PCA.

Fact 1: A line which minimizes the total squared distance must go through the origin (or sample means)

Fact 2: By the Pythagorean theorem, for any point:

$$\text{PC score}^2 + \text{Perpendicular distance}^2 = \text{Distance to origin}^2$$

Adding all the terms for each point, we have the following striking relationship:

$$\frac{1}{n-1}\text{Sum}(\text{PC score}^2) + \frac{1}{n-1}\text{Sum}(\text{Perpendicular distance}^2) = \frac{1}{n-1}\text{Sum}(\text{Distance to origin}^2)$$

Notice:

1. $\text{Sum}(\text{Distance to origin}^2)$ never changes.
2. $\frac{1}{n-1}\text{Sum}(\text{PC score}^2) = Var(\text{PC scores})$
3. $\frac{1}{n-1}\text{Sum}(\text{Perpendicular distance}^2) = \text{Mean squred errors}$

Hence, maximizing the variance of PC scores is equivalent to minimizing the mean squared error (perpendicular distances). Note that minimizing the mean *perpendicular* distances here is different from simple linear regression that minimizes the *vertical* distances between the linear line and points. Now we are ready to reveal two equivalent definitions of PCs:

Definition 1: The linear combination which minimizes the total squared perpendicular distance

Definition 2: The linear combination with maximum variance or with largest spread or formally, we are looking for a pair of weights ϕ_{11} and ϕ_{21} such that

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2$$

such that

$$\max_{\phi_{11}^2+\phi_{21}^2=1} \text{Var}(Z_1 = \phi_{11}X_1 + \phi_{21}X_2)$$

3.1.4 An animation to find the optimal weights (loadings)

The following chuck illustrates when the linear combination of different weights or the line has a different slope. The relationship between the three sums changes exactly as we have shown above.

As sum of squared errors increases, variance of the line decreases while both sums never change. Of course when the line minimizes the sum of squared errors it also maximizes the variance of the PC scores which gives us the First Principal Component!

We used `shiny` to make this illustration. When execute the following chunk, **a separate window will pop out.** By changing the slope of the line, you will see how the projection of points changes and how the squared distance and variance change. Compare with the red PC component line. (Remember to kill the graph once you are done; otherwise the following chunk will be running all the time.)

3.1.5 Other Principal Components

Once we find the leading principal component, we can look for the second linear combination of the `Word` and `Parag` such that the line goes through the data points with minimum squared distance or largest variance but with one constrain - the line needs to be perpendicular to the first principal component. We call that line, second Principal component.

Or mathematically we are looking for another linear transformation Z_2 of $X_1 =$ `Word`, $X_2 =$ `Parag` to have the max variance of Z_2 and Z_1 is orthogonal to Z_2.

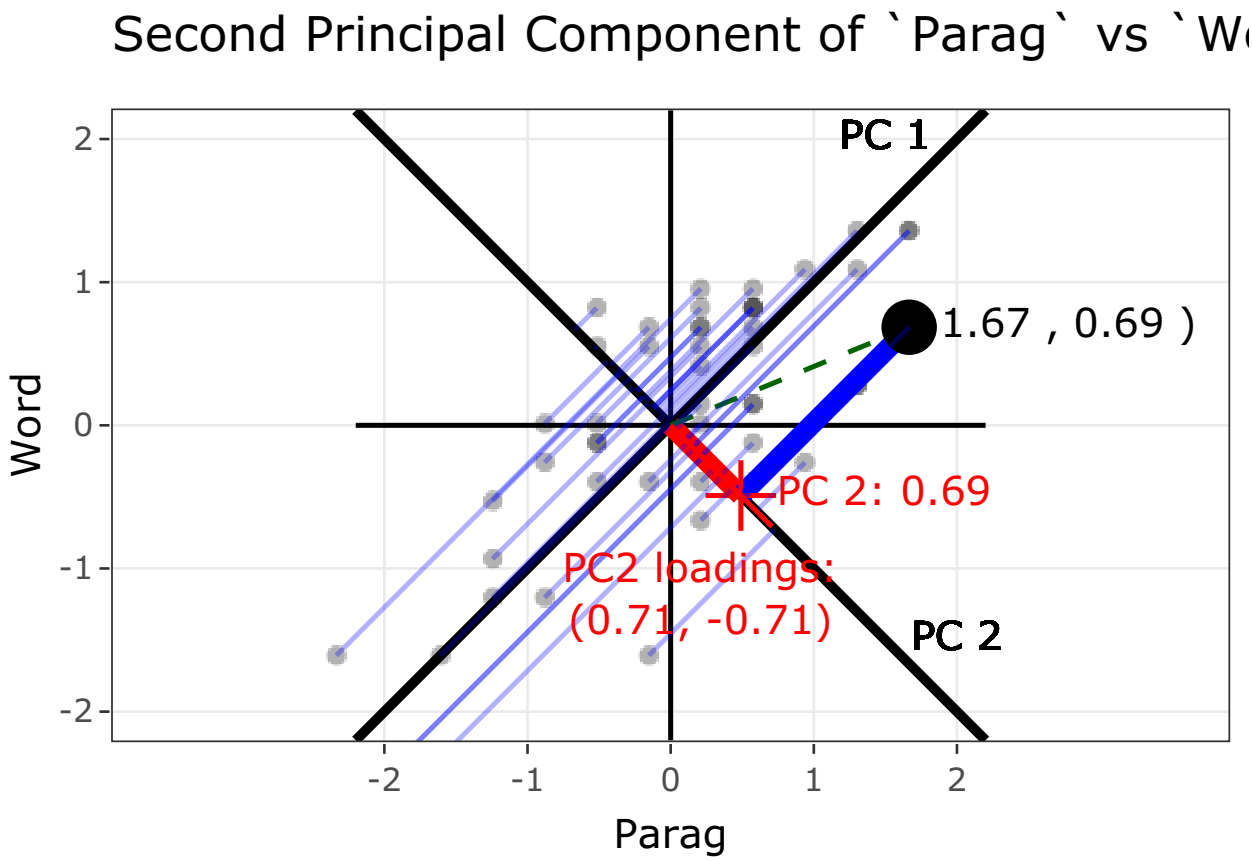
This is same as finding (ϕ_{12}, ϕ_{22}) such that $Z_2 = \phi_{12}X_1 + \phi_{22}X_2$ and

$$\max_{\phi_{12}^2+\phi_{22}^2=1} \text{Var}(Z_2)$$

and two sets of loadings are perpendicular: or (ϕ_{11}, ϕ_{21}) and (ϕ_{12}, ϕ_{22}) are orthogonal.

Remark: By the definition we know the variance of PC1 is larger than that of PC2.

The following plot shows how to find the second principal component.



If we only use PC1, we will lose some information contained in two scores. The error is described by the sum of squared error or the variance of PC1.

If we use both PC1 and PC2 we do not lose any information at all. In other words, the variance of PC1 plus variance of PC2 is exactly variance of `Word` + variance of `Parag`!

Remark: We can have maximum two PCs when there are only two variables.

4 Principal Component of AFQT tests

AFQT contains four tests. Our goal is to use less number of variables to capture the information contained in 4 scores. Analogous to two variables, we can define PC's for 4 variables. The leading PC's would be a linear combination of 4 scores which maximize the variance of the linear combination. We have postpone the formal definitions to the appendices.

There will be no more than 4 PCs since there are only 4 variables. Each PC will be controlled by the loadings or the weights to each variable. All 4 sets of loadings are orthogonal with unit 1. All 4 PCs are also orthogonal or uncorrelated with decreasing variances.

We hope using a few principal components to capture the structure among all variables. Often the clustering information may appear in PC coordinates. If we are lucky enough we may discover clear interpretations of each PC in terms of the original scores. In general we may lose interpretation from each score.

Question:

1) What does each PC score mean?
2) How many PCs should be used?
3) What interesting facts can be revealed by PCs?

4.1 Find PCs and Loadings

`prcomp()` is the main function used to give us all the loadings and PCs and variances of each PC. You will find simple, beautiful mathematics how PCA is done through eigen decomposition and SVD (Singular Value Decomposition) in Appendix.

To conduct PCA:

Step I: To find sensible PCs, it is recommended to

- center each variable by subtracting its mean
- scale each variable by dividing its sd (rather complex on this issue)
- `prcomp()` has an option to scale or we could use `scale()` explicitly

Step II: Run `prcomp()`

- Output all the loadings: one set for each PC
- Obtain all PC scores
- Report the variances for each PC and for each original variable

We next perform PCA for four variables `Word`, `Parag`, `Math`, `Arith`.

```
data.AFQT <- AFQT.sub %>% select(Word, Parag, Math, Arith)
# summary(data.AFQT) # no sd's
# skim(data.AFQT) # provides mean, sd and more
# skim(data.AFQT) %>% select(skim_variable, numeric.mean, numeric.sd)
pc.4 <- prcomp(data.AFQT, scale=TRUE, center=T)  # by default, center=True but scale=FALSE!!!
names(pc.4) #check output
# rotation: loadings pc.4$rotat
# x: PC scores
# sdev: standard dev of the PC scores (pc.4$sdev)
# pc.4$center # sample mean of the original x's
# pc.4$scale   # sd of the original x's
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

`prcomp()` outputs the following:

- `$rotation`: loadings
- `$x`: PC scores
- `$sdev`: standard deviations of the four PC's
- `$center`: means of the four tests
- `$scale`: standard deviatoins of the four tests

Loadings

Each loadings give us a set of four numbers which determines the direction of each **line**. Let us take a look at the leading PC's loadings:

```
pc.4.loading <- pc.4$rotation   #pc.4$x (pc.4$sd)^2
knitr::kable(pc.4.loading)
```

	PC1	PC2	PC3	PC4
Word	0.509	0.442	-0.365	-0.642
Parag	0.500	0.546	0.308	0.598
Math	0.496	-0.483	0.652	-0.310
Arith	0.494	-0.523	-0.589	0.368

Remark:

- Loadings are unique up to sign. For example PC1 loading can be $(.51, .50, .5, .5)$ or $(-.51, -.50, -.5, -.5)$. Why so???
- The magnitude of loadings tells us how much each variable contributes to the PCs.

PCs

We can get PCs by taking the linear combination of loadings and variables as:

$$\begin{aligned}\texttt{PC1} &= 0.509 \times \texttt{Word\_centered\_scaled} + 0.5 \times \texttt{Parag\_centered\_scaled} \\ &\quad + 0.496 \times \texttt{Math\_centered\_scaled} + 0.494 \times \texttt{Arith\_centered\_scaled} \\ \texttt{PC2} &= (0.442) \times \texttt{Word\_centered\_scaled} + (0.546) \times \texttt{Parag\_centered\_scaled} \\ &\quad + -0.483 \times \texttt{Math\_centered\_scaled} + -0.523 \times \texttt{Arith\_centered\_scaled}\end{aligned}$$

We will continue to get PC3 and PC4.

All the PCs are computed. Each person will have four PC scores. Let us take PCs for the first 5 people

```
knitr::kable(pc.4$x[1:5, ])
```

	PC1	PC2	PC3	PC4
p1	-0.920	0.044	0.923	-0.611
p2	0.611	-1.171	-0.585	-0.004
p3	-0.312	0.697	0.100	-0.683
p4	3.069	-0.049	0.314	0.105
p5	2.818	-0.440	0.219	0.112

Interpretations of loadings and PCs

Loadings determine contribution of each variable to the PCs. Loadings are also proportional to the correlations between each PC and variable.

```
pc.4.loading
```

```
##          PC1    PC2    PC3    PC4
## Word   0.509  0.442 -0.365 -0.642
## Parag  0.500  0.546  0.308  0.598
## Math   0.496 -0.483  0.652 -0.310
## Arith  0.494 -0.523 -0.589  0.368
```

PC1: since the four loadings are approximately the same around .5 so PC1 is proportional to the total of the four scores. In other words:

$$\begin{aligned}\texttt{PC1} = .5 \times (&\texttt{Word\_centered\_scaled} + \texttt{Parag\_centered\_scaled} \\ &+ \texttt{Math\_centered\_scaled} + \texttt{Arith\_centered\_scaled})\end{aligned}$$

Higher PC1 $\implies$ Higher weighted total score.

PC2:

$$\begin{aligned}\texttt{PC2} = .5 \times (&\texttt{Word\_centered\_scaled} + \texttt{Parag\_centered\_scaled}) \\ &- .5 \times (\texttt{Math\_centered\_scaled} + \texttt{Arith\_centered\_scaled})\end{aligned}$$

- Approximately proportional to the difference between to sum of `Math`/`Arith` and sum of `Word` and `Parag`
- If total scores are comparable, higher PC2 implies strong math talent while lower PC2 implies superior language ability

Combine centered and scaled `Word`, `Parag`, `Math`, `Arith` with `PC1`, `PC2`, `PC3`, `PC4`. We list a few people's scores and PCs. Can you calculate the PCs from the `Word`, `Parag`, `Math`, `Arith` using the loadings?

```
AFQT.PC.Scores <- cbind(scale(data.AFQT, scale = TRUE), pc.4$x)
   arrange(as.data.frame(AFQT.PC.Scores), desc(PC1)) %>%
     head()
```

```
##      Word Parag Math Arith  PC1     PC2    PC3     PC4
## p30 1.360 1.302 1.89  1.67 3.11 -0.4783  0.153 -0.0642
## p4  1.360 1.666 1.72  1.40 3.07 -0.0492  0.314  0.1052
## p5  1.090 1.302 1.72  1.54 2.82 -0.4399  0.219  0.1121
## p21 0.955 0.575 2.07  1.81 2.70 -1.2121  0.109 -0.2426
## p49 0.281 1.302 1.72  1.26 2.27 -0.6519  0.678  0.5290
## p44 0.820 0.575 1.19  1.54 2.06 -0.7020 -0.251  0.0136
```

Loadings and correlations between PC and each scores:

Loadings account for weights of each variable in the PC. They are in fact proportional to the correlation between PC to each score with `sd(PC)` as a factor. In other words,

$$\texttt{Corr(PC1, data.AFQT.scale)} = \texttt{sd(PC1)} \times \texttt{PC1\_loadings}$$

```
cor(pc.4$x[, 1], scale(data.AFQT, scale = TRUE))[1,]
```

```
##  Word Parag  Math Arith
## 0.891 0.874 0.868 0.865
```

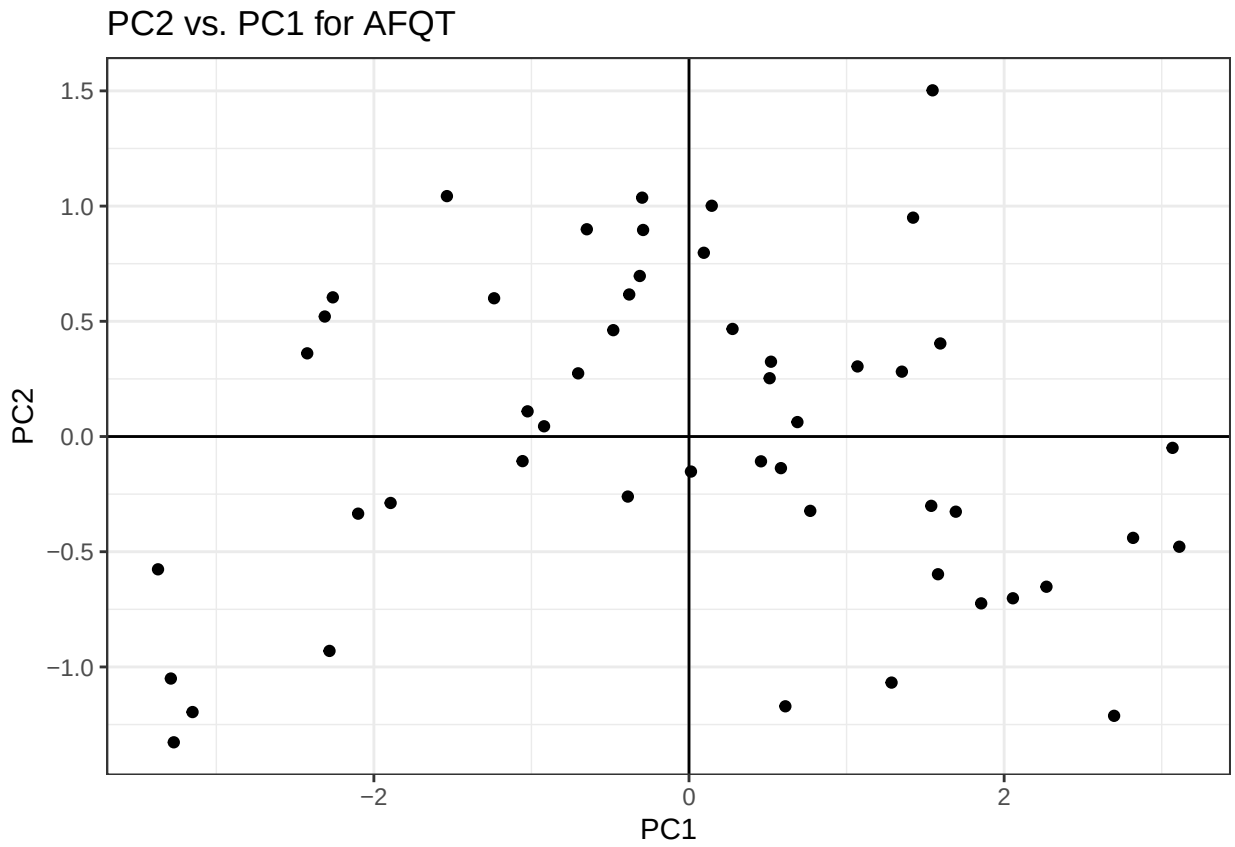
```
sd(pc.4$x[, 1]) * pc.4$rot[,1]
```

```
##  Word Parag  Math Arith
## 0.891 0.874 0.868 0.865
```

4.2 Scatter plot of PCs

Often a scatter plot of PCs may reveal interesting information. For example, we know PC1 and PC2 have clear interpretaion, by plotting PC2 vs. PC1, we can locate people with different strength.

```
as.data.frame(pc.4$x) %>%
  ggplot(aes(x=PC1, y=PC2)) +
  geom_point()+
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  ggtitle("PC2 vs. PC1 for AFQT") +
  theme_bw()
```



People on the far right are high in total scores. The first quadrant contains people with strong math skills while one in the forth quadrant good in language skills.

4.3 Properties of PCs and Loadings

All loadings are perpendicular and with unit 1

```
round(t(pc.4$rotation) %*% pc.4$rotation) # to check the loadings are unit 1
```

```
##     PC1 PC2 PC3 PC4
## PC1   1   0   0   0
## PC2   0   1   0   0
## PC3   0   0   1   0
## PC4   0   0   0   1
```

```
# or
colSums((pc.4$rotation)^2)
```

```
## PC1 PC2 PC3 PC4
##   1   1   1   1
```

`var(PC1)` > `var(PC2)` > `...` and they add up to be 4. Why so?

```
round((pc.4$sdev)^2, 2)  # Var(PC1), Var(PC2),...#sum((pc.4$sdev)^2)
```

```
## [1] 3.06 0.49 0.27 0.18
```

```
# knitr::kable(summary(pc.4)$importance)
```

Notice var(PC1) is much larger than the rest of the variances. PC1 captures large amount of variability in the data.

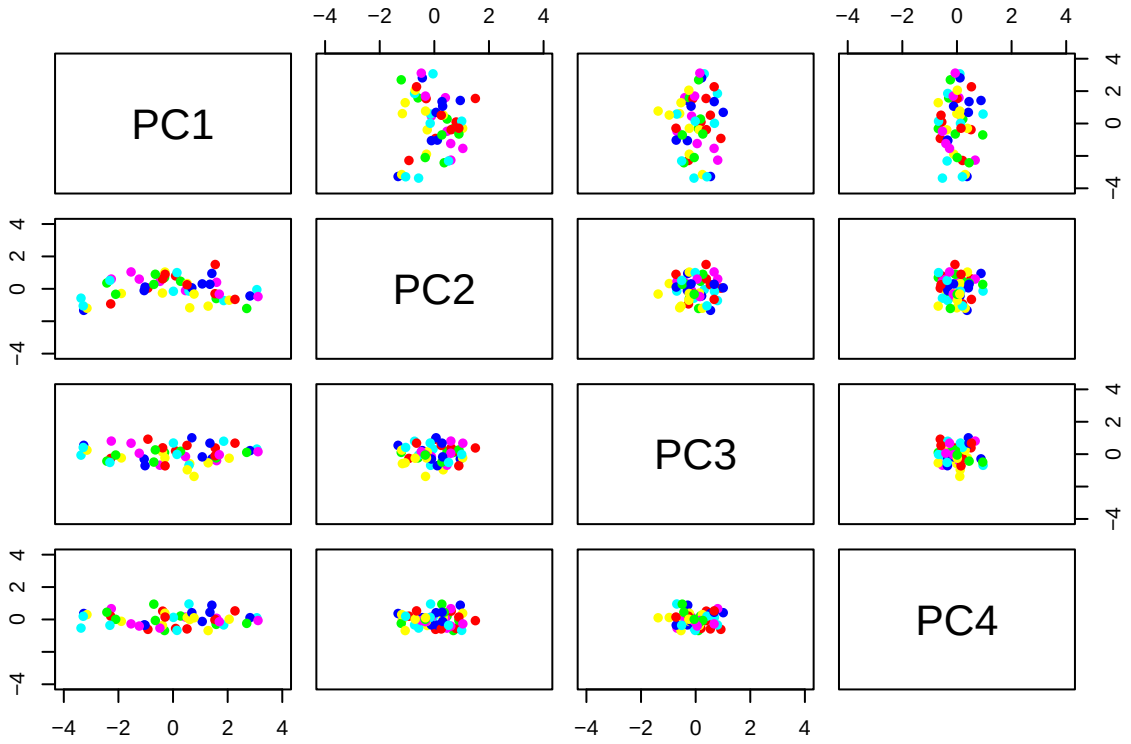
All 4 PC scores are uncorrelated

```
round(cov(pc.4$x), 4)
```

```
##       PC1   PC2   PC3  PC4
## PC1 3.06 0.000 0.000 0.00
## PC2 0.00 0.488 0.000 0.00
## PC3 0.00 0.000 0.271 0.00
## PC4 0.00 0.000 0.000 0.18
```

From the following pair-wise plots we see the variability of each PC in a decreasing scale.

```
pairs(pc.4$x, xlim=c(-4, 4), ylim=c(-4, 4), col=rainbow(6), pch=16)
```



4.4 Proportion of variance explained (PVE)

One goal of principal component is to find as few as many PCs which have as large variances as possible. How many PCs are informative? We introduce the measurement of proportion of variance explained (PVE) as

$$\text{PVE} = \text{Var(PC)} \,/\, \text{Total Variances}$$

We can calculate the PVE or get proportion of variance from the output

```r
summary(pc.4)$importance  #notice it is from summary()  names(summary(pc.4))
```

```
##                           PC1   PC2    PC3    PC4
## Standard deviation     1.750 0.699 0.5208 0.4238
## Proportion of Variance 0.765 0.122 0.0678 0.0449
## Cumulative Proportion  0.765 0.887 0.9551 1.0000
```

```r
names(pc.4)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

The summary reports standard deviations, PVE and cumulative proportions.

For example, the leading principal component explains 0.765 of the total variance.

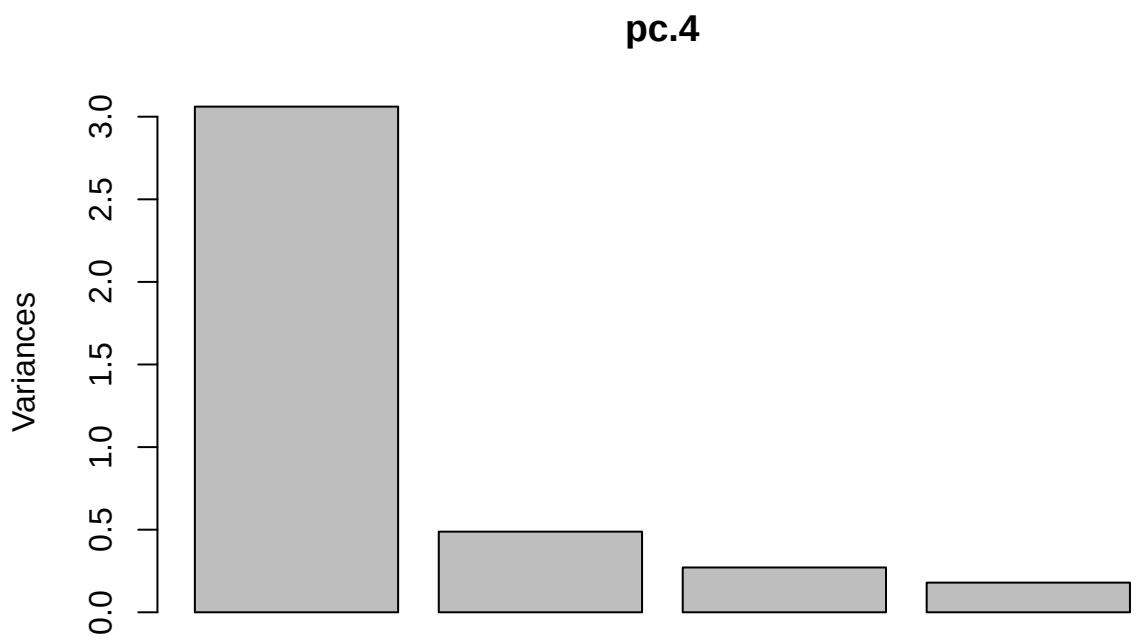
We also see clearly that variance of PC1 is larger than that of PC2, etc.

Scree Plots

A scree plot of PVE or Cumulative PVE can help us to see how much variance is captured by each PC.

Scree plot of variances:

```r
plot(pc.4) # variances of each pc
```

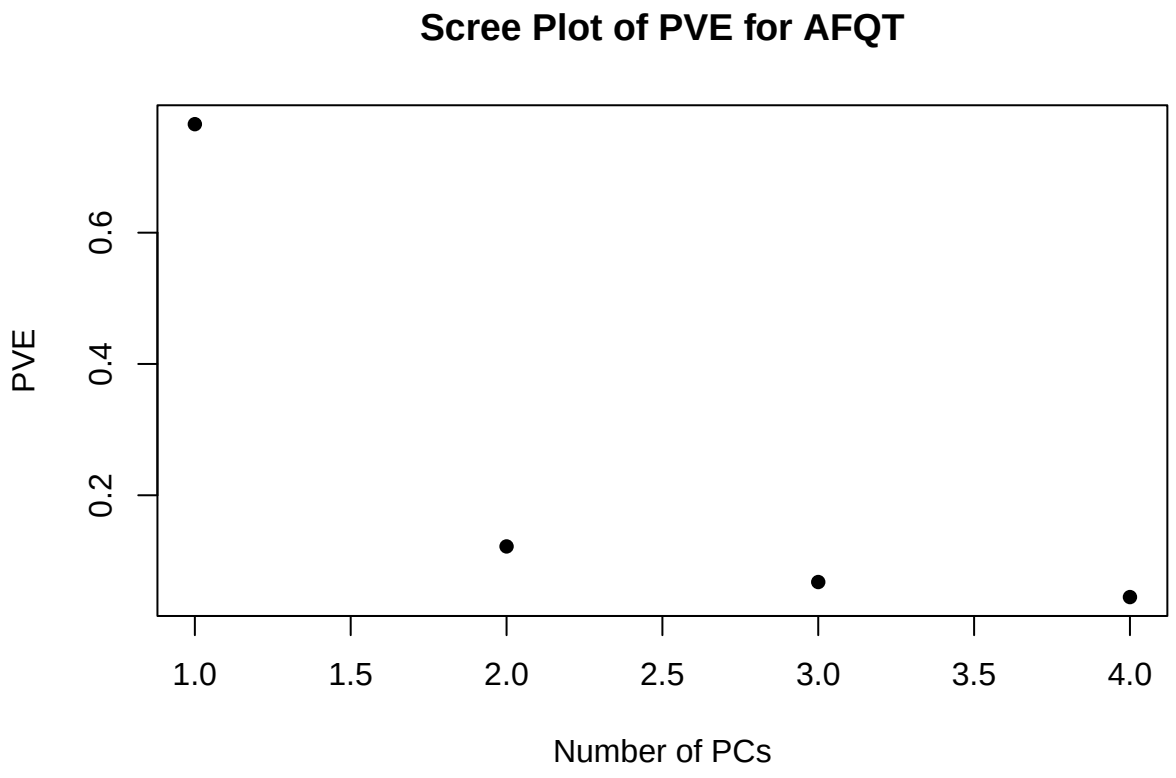


How many PCs to use?

We may look at the scree plot of PVEs and apply elbow rules: take the number of PCs when there is a sharp drop in the scree plot.

Here is the scree plot of PVEs.

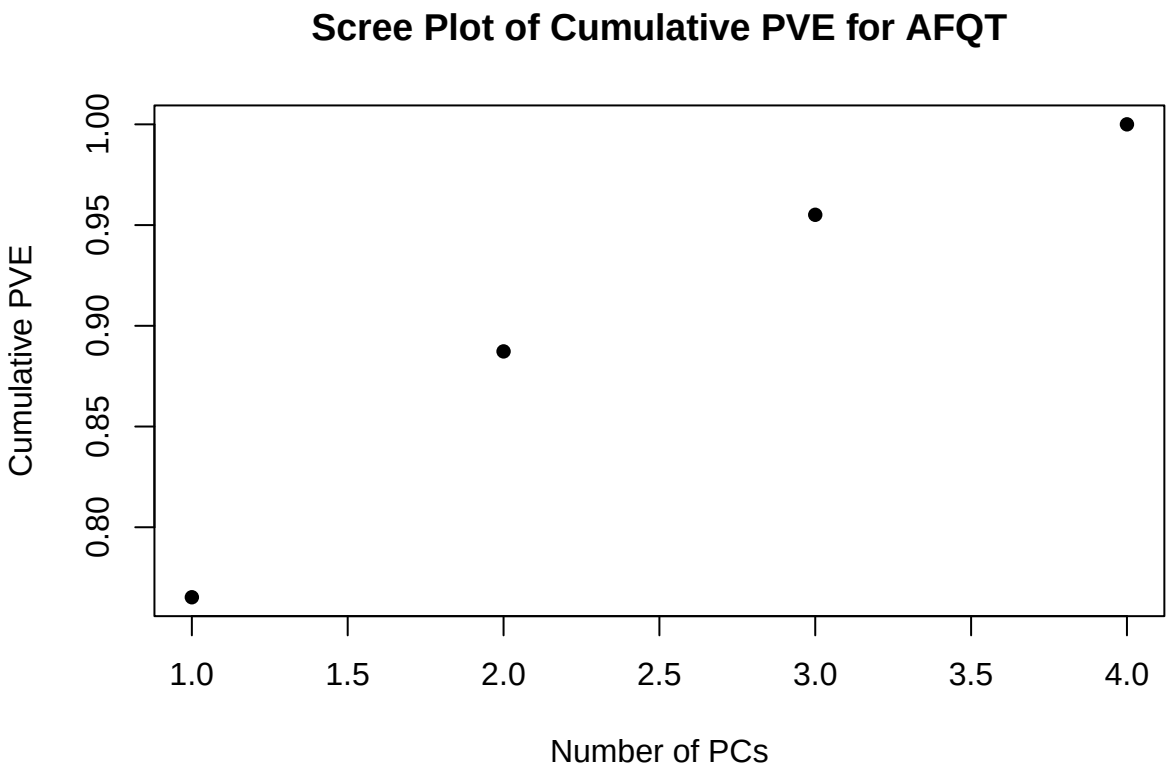
```r
plot(summary(pc.4)$importance[2, ],  # PVE
     ylab="PVE",
     xlab="Number of PCs",
     pch = 16,
     main="Scree Plot of PVE for AFQT")
```



It indicates that two leading PCs should be enough for certain purposes.

Lastly we may look at the cumulative variance explained by each PC.

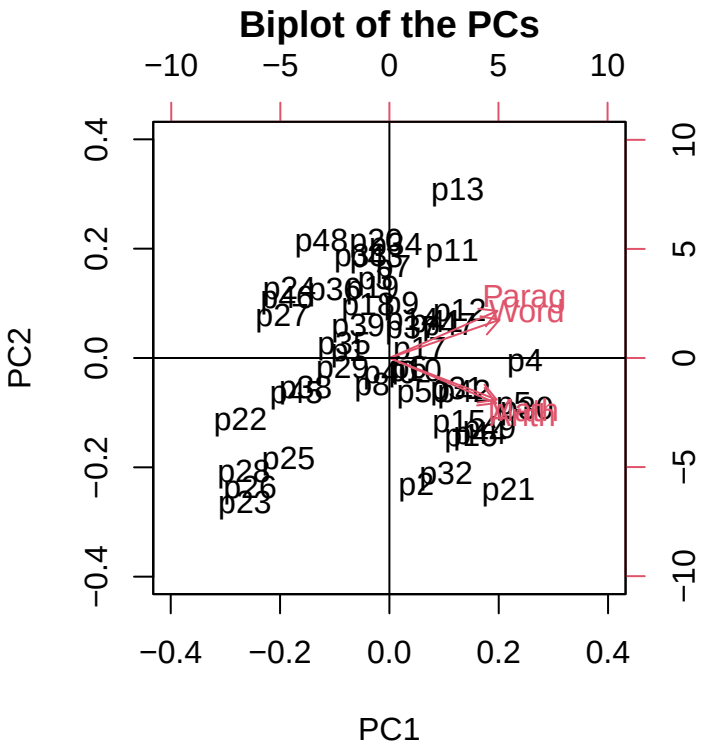
```r
plot(summary(pc.4)$importance[3, ], pch=16,
     ylab="Cumulative PVE",
     xlab="Number of PCs",
     main="Scree Plot of Cumulative PVE for AFQT")
```



4.5 Biplot

Visualize the PC scores together with the loadings of the original variables. They also reveal correlation structures among all variables.

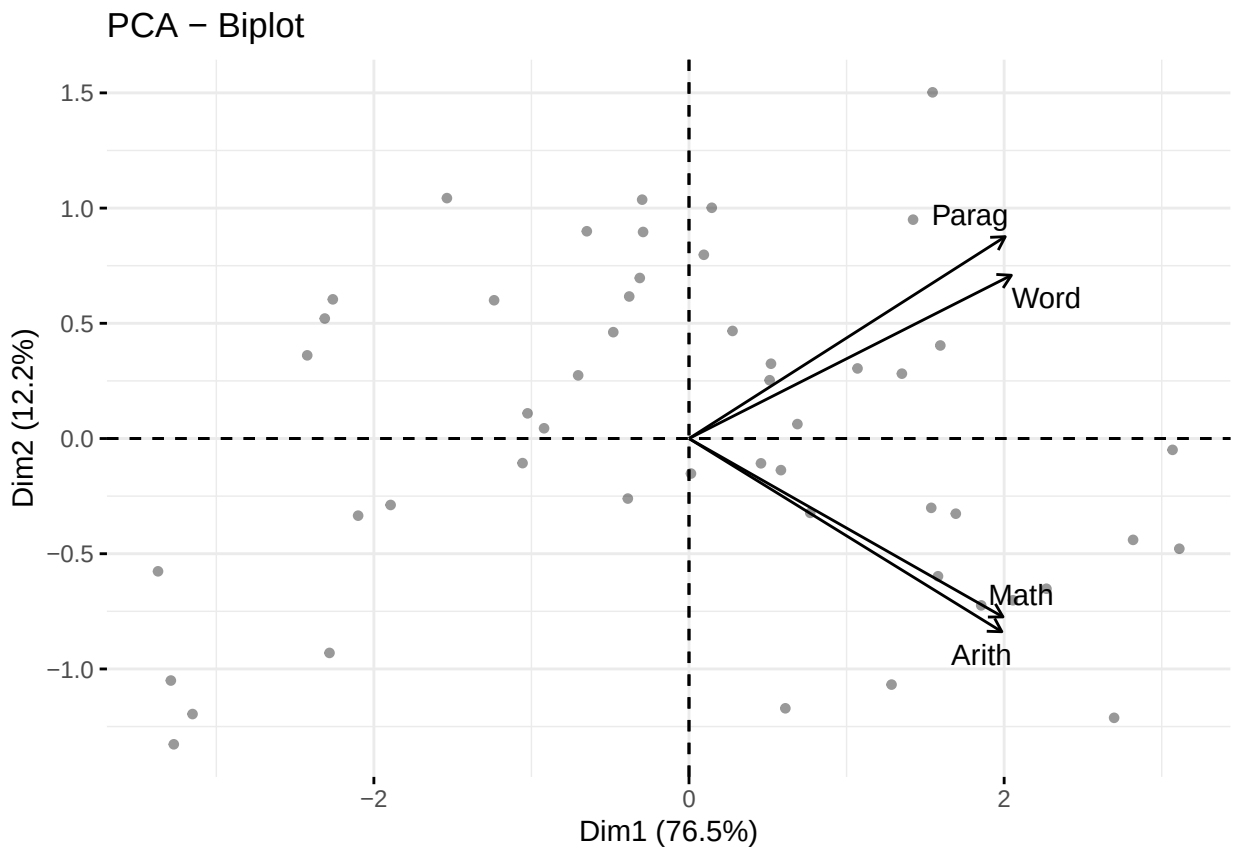
```
lim <- c(-.4, .4)
biplot(pc.4,        # choices=c(1,3),
       xlim=lim,
       ylim=lim,
       main="Biplot of the PCs")
abline(v=0, h=0)
```




```
# x-axis: PC1 (prop)
# y-axis: PC2
# top x-axis: prop to loadings for PC1
# right y-axis: prop to loadings for PC2
# using argument of choices = c(1,3), we can explore scatter plots of other PCs
```

Better biplot using factoextra package (factor analysis visualization not a categorical variable)

```
fviz_pca_biplot(
  pc.4,
  label = "var",
  repel = TRUE,
  pointsize = 1.2,
  alpha.ind = 0.4,
  col.var = "black"
)
```



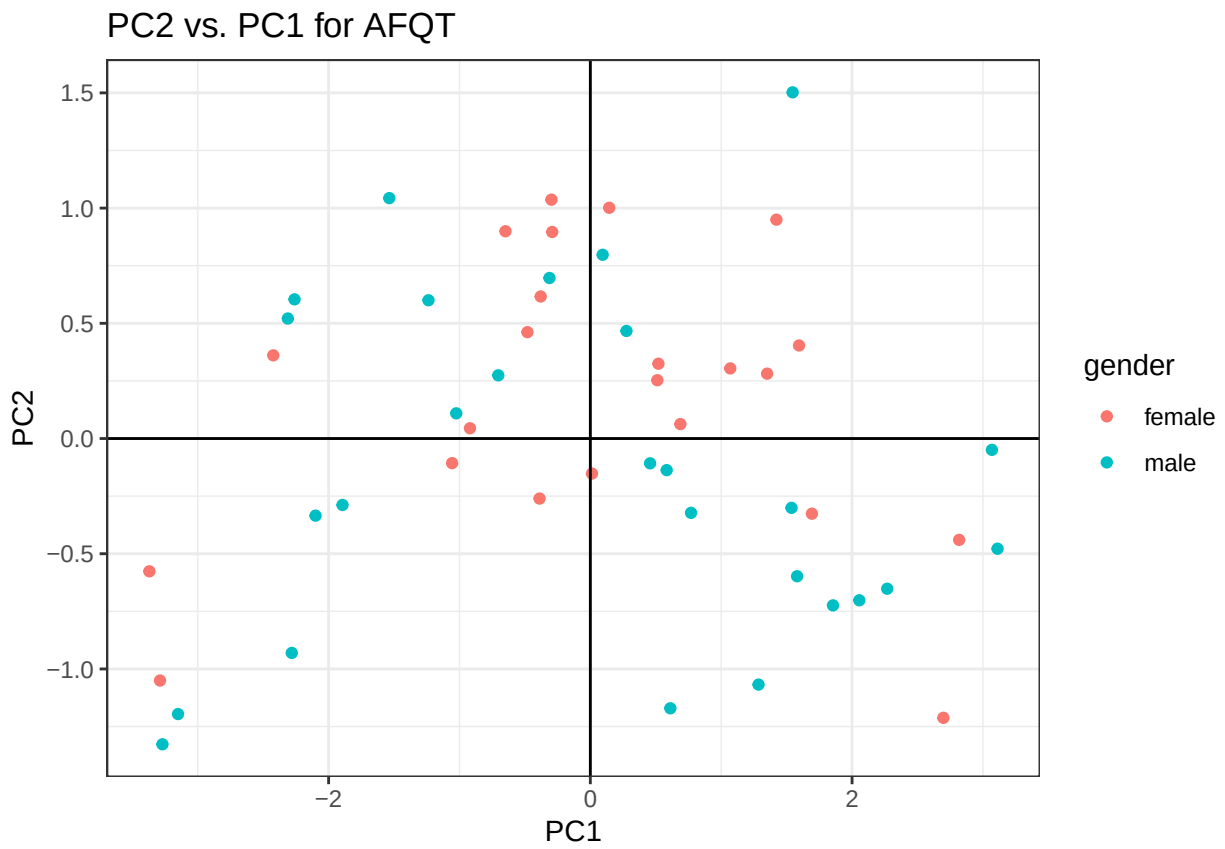
The biplot indicates

- PC1 loadings are similar in magnitudes and with same signs
- PC2 captures difference between total of `Math`, `Arith` and total of `Word` and `Parag`
- `Math` and `Arith` are highly correlated, and so are `Word` and `Parag`!

4.6 Gender effects?

Are there systematic differences among men and women? By various plots of PCs we try to see any possible patterns.

```
as.data.frame(pc.4$x) %>%
  mutate(gender = AFQT.sub$Gender) %>%
  ggplot(aes(x=PC1, y=PC2)) +     # Try other PCs vs. PC1, any patterns?
  geom_point(aes(color=gender))+
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  ggtitle("PC2 vs. PC1 for AFQT") +
  theme_bw()
```

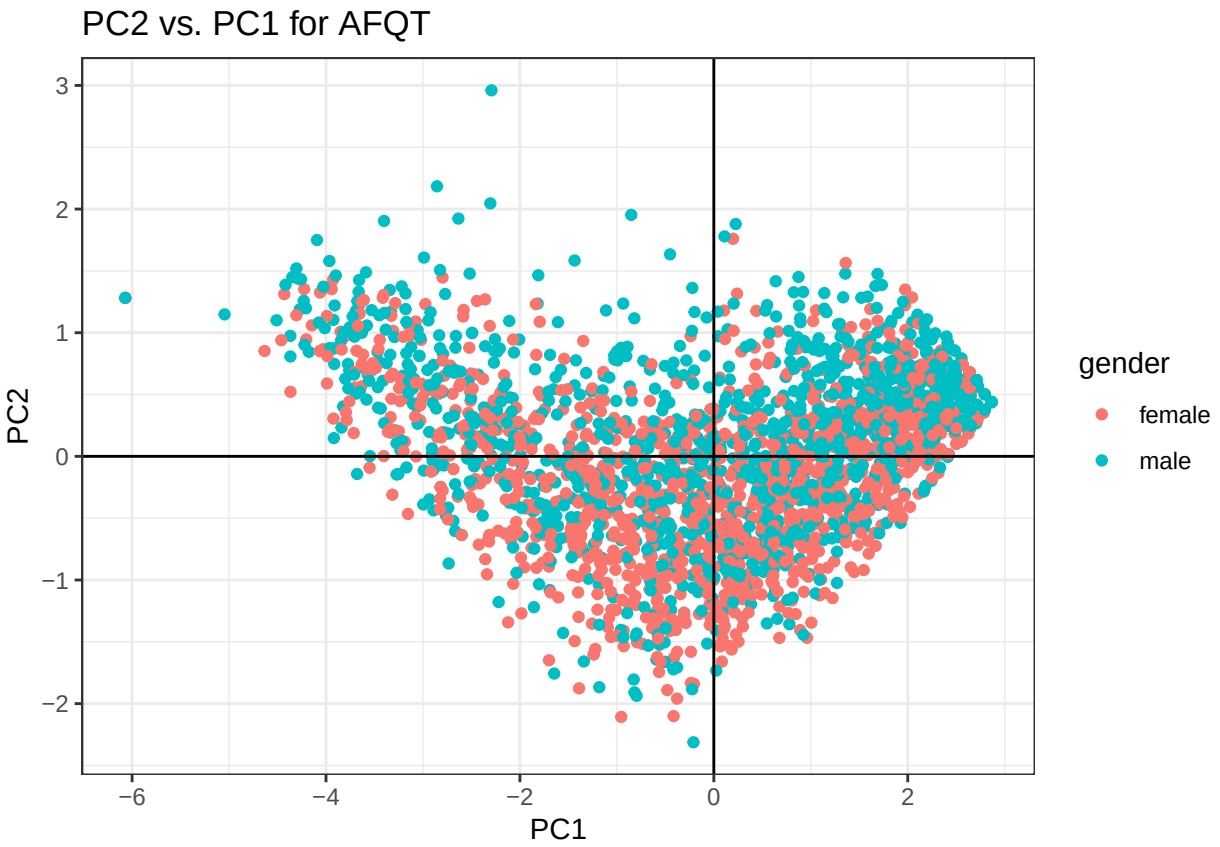


We couldn't tell any relationship between Gender and AFQT.

Lastly AFQT PCs and Gender:

So far the PCA is done for a subset of 50 subjects. Finally we bring all subjects and run PCA of over `AFQT` and `Gender`

```
pc.4.all <- AFQT.full %>% select(Word, Parag, Arith, Math) %>%
     prcomp(scale=TRUE)   #pc.4.all$rotation # check the signs
#pc.4.all$rotation
as.data.frame(pc.4.all$x) %>%
  mutate(gender = AFQT.full$Gender) %>%
  ggplot(aes(x=PC1, y=PC2)) +
  geom_point(aes(color=gender))+
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  ggtitle("PC2 vs. PC1 for AFQT") +
  theme_bw()
```



Questions: - What are the PC1 and PC2 loadings? - What are the interpretations of PC1 and PC2? - Do you see systematic difference between men's and women's AFQT scores? - How do you summarize the performance based on the above PCs plot?

4.7 Summary

1. To capture the main features of AFQT four scores we could use two newly formed PC scores:
   - PC1: Total scores (weighted)
   - PC2: Difference between `Math+Arith` and `Word+Parag`
2. **What is AFQT score** reported? Might it be the PC1 of the four tests? Or is it similar to the total scores?

```
cor(AFQT.full$AFQT, pc.4.all$x[, 1]) #plot(AFQT.full$AFQT, pc.4.all$x[, 1])
```

```
## [1] 0.966
```

```
total_no_weight <- AFQT.full %>%   # create total scores
    mutate(total = Word+Parag+Math+Arith) %>%
    select(total)
cor(AFQT.full$AFQT, total_no_weight) # data.frame(AFQT.full$AFQT, total_no_weight)
```

```
##      total
## [1,] 0.964
```

Final questions to ask: what happens if we run PCA without scaling?

PCA on AFQT without scaling:

```
pc.4.no.scale <- prcomp(AFQT.full %>% select(-Subject, -AFQT, -Gender), scale = FALSE)
pc.4.no.scale
```

What do you think?

5 PCA of SVABS

SVABS contains 10 test scores. How can we use a few summary scores to capture some main features hidden in the 10 scores? How can we tell who is good in certain areas? Are there systematic difference between men and women in the SVABS tests?

We will explore how well PCA can answer all the questions raised.

5.1 Leading PCs

Now bring all the subjects with all 10 test scores in the following code. We first list PC1 loadings in a decreasing order. Roughly speaking, the loadings are similar indicating that PC1 captures the total scores (scaled by the standard deviations for each test.)

```
pca.all <- prcomp(data.full[,  c(11:20)], scale=TRUE)   # all the tests
# loadings and with test names
pca.all.loading <- data.frame(tests=row.names(pca.all$rotation), pca.all$rotation)
pca.all.loading %>% select(tests, PC1, PC2) %>%
  arrange(-PC1)
```

```
##             tests   PC1     PC2
## Arith       Arith 0.354  0.0487
## Science   Science 0.352 -0.1419
## Word         Word 0.350  0.0615
## Math         Math 0.336  0.1426
## Parag       Parag 0.326  0.1897
## Elec         Elec 0.324 -0.3353
## Mechanic Mechanic 0.316 -0.3345
## Numer       Numer 0.275  0.4517
## Auto         Auto 0.272 -0.4735
## Coding     Coding 0.234  0.5146
```

We next look into the PC2 loadings.

```
# loadings and with test names
pca.all.loading <- data.frame(tests=row.names(pca.all$rotation), pca.all$rotation)
pca.all.loading %>% select(tests, PC1, PC2, PC3) %>%
  arrange(-PC2)
```

```
##             tests   PC1     PC2     PC3
## Coding     Coding 0.234  0.5146  0.5152
## Numer       Numer 0.275  0.4517  0.3276
## Parag       Parag 0.326  0.1897 -0.3911
## Math         Math 0.336  0.1426 -0.2232
## Word         Word 0.350  0.0615 -0.3429
## Arith       Arith 0.354  0.0487 -0.0919
## Science   Science 0.352 -0.1419 -0.2220
## Mechanic Mechanic 0.316 -0.3345  0.2522
## Elec         Elec 0.324 -0.3353  0.0860
## Auto         Auto 0.272 -0.4735  0.4220
```

It captures the difference between the total of `coding`, `Numer`, `Parag`, `Math` and the total of `Mechanic`, `Elec` and `Auto`.

PC1: Proportional to the total scores. **PC2:** Difference between intelligence(such as math/comprehensive understanding?) and dexterity

5.2 PVE

How much variations do leading PCs account for?

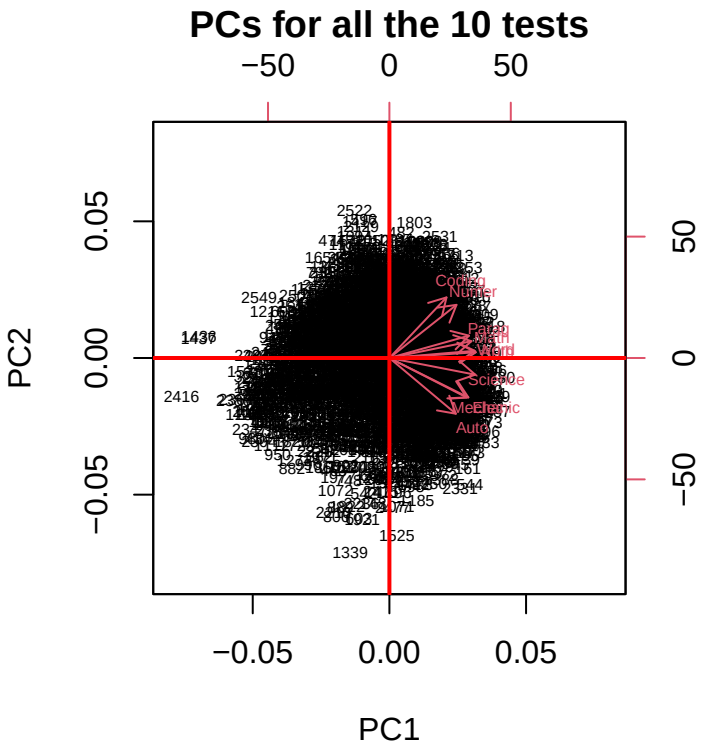
```
summary(pca.all)$importance
```

```
##                           PC1   PC2    PC3    PC4    PC5    PC6    PC7    PC8
## Standard deviation     2.472 1.193 0.7515 0.7059 0.5578 0.5425 0.4845 0.4758
## Proportion of Variance 0.611 0.142 0.0565 0.0498 0.0311 0.0294 0.0235 0.0226
## Cumulative Proportion  0.611 0.753 0.8097 0.8595 0.8906 0.9201 0.9435 0.9662
##                           PC9   PC10
## Standard deviation     0.4196 0.4027
## Proportion of Variance 0.0176 0.0162
## Cumulative Proportion  0.9838 1.0000
```

5.3 Biplot

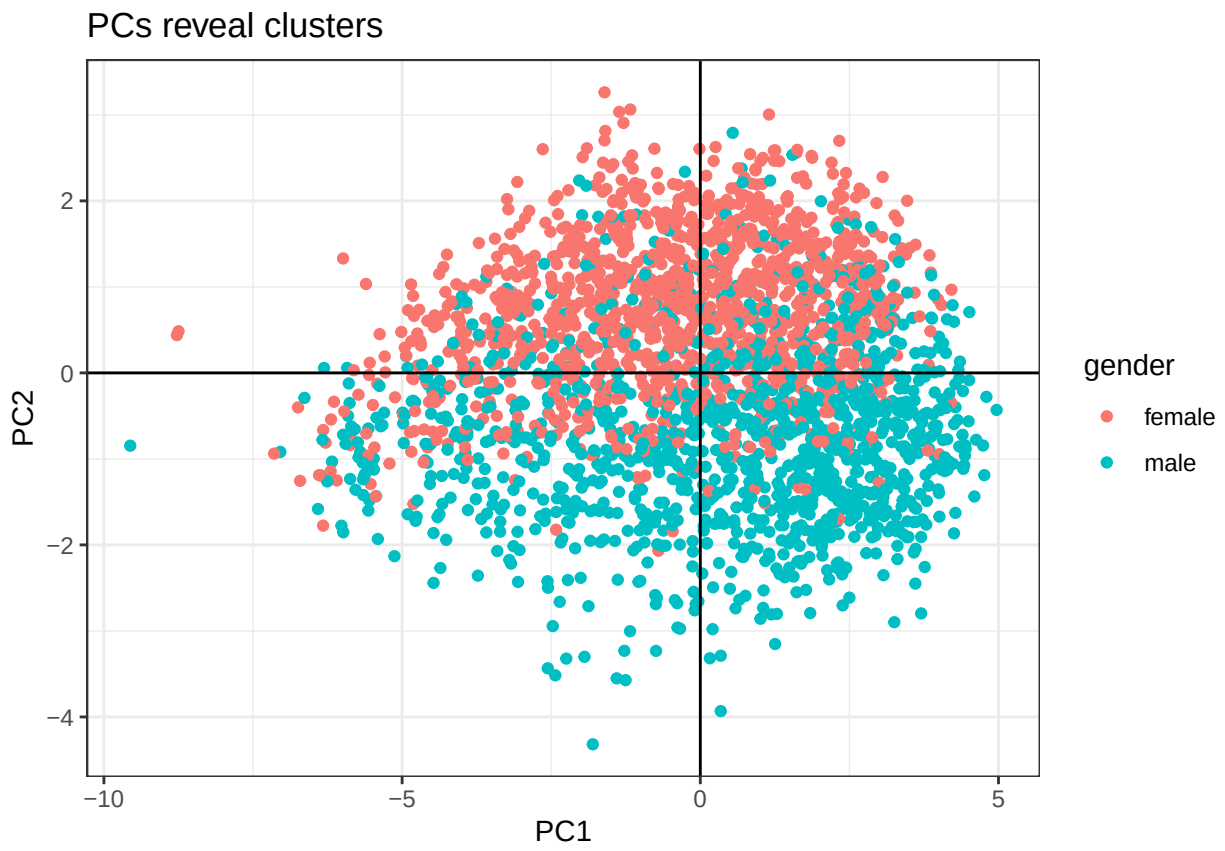
To visualize the loadings and the correlations among test scores, here is the biplot.

```
biplot(pca.all, cex=0.5, xlim=c(-.08, .08),
       ylim=c(-.08, .08),
       main="PCs for all the 10 tests")
abline(h=0, v=0, col="red", lwd=2)
```



5.4 How Gender plays the role here?

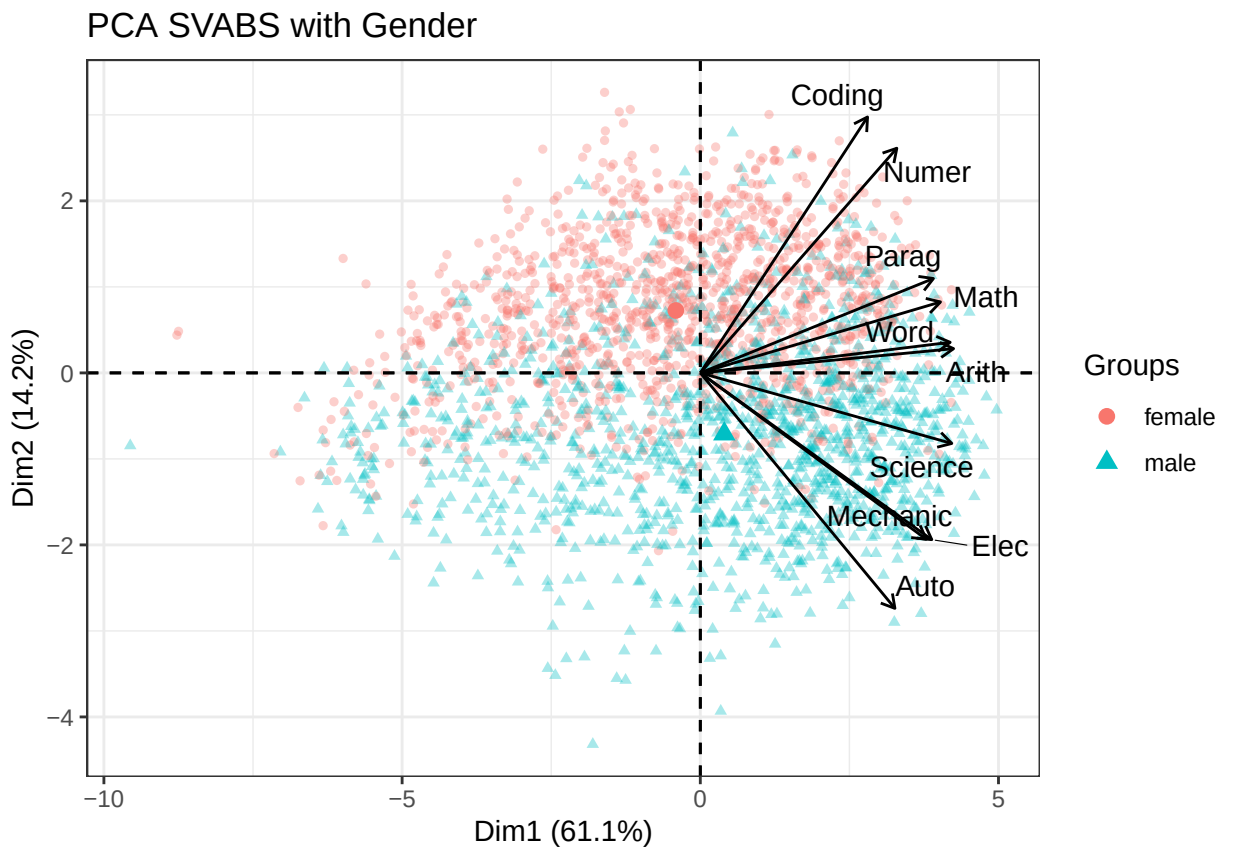
```
as.data.frame(pca.all$x) %>%
  mutate(gender = data.full$Gender) %>%
  ggplot(aes(x = PC1, y = PC2)) +  # pca.all$rotation  try PC3 vs. PC1, no gender effect
  geom_point(aes(color = gender)) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  ggtitle("PCs reveal clusters") +
  theme_bw()
```



WOW! Males and females are clearly separated by PC2! That implies females (red circles) are strong in intelligence and males are strong in dexterity! Does that agree with your intuition?

`factoextra` package also provides nice PCA biplot.

```
library(factoextra)
gender <- factor(data.full$Gender)

p <- fviz_pca_biplot(
  pca.all,
  habillage = gender,
  label = "var",
  repel = TRUE,
  pointsize = 1.3,
  alpha.ind = 0.35,
  col.var = "black"
)

p + theme_bw() + ggtitle("PCA SVABS with Gender")
```



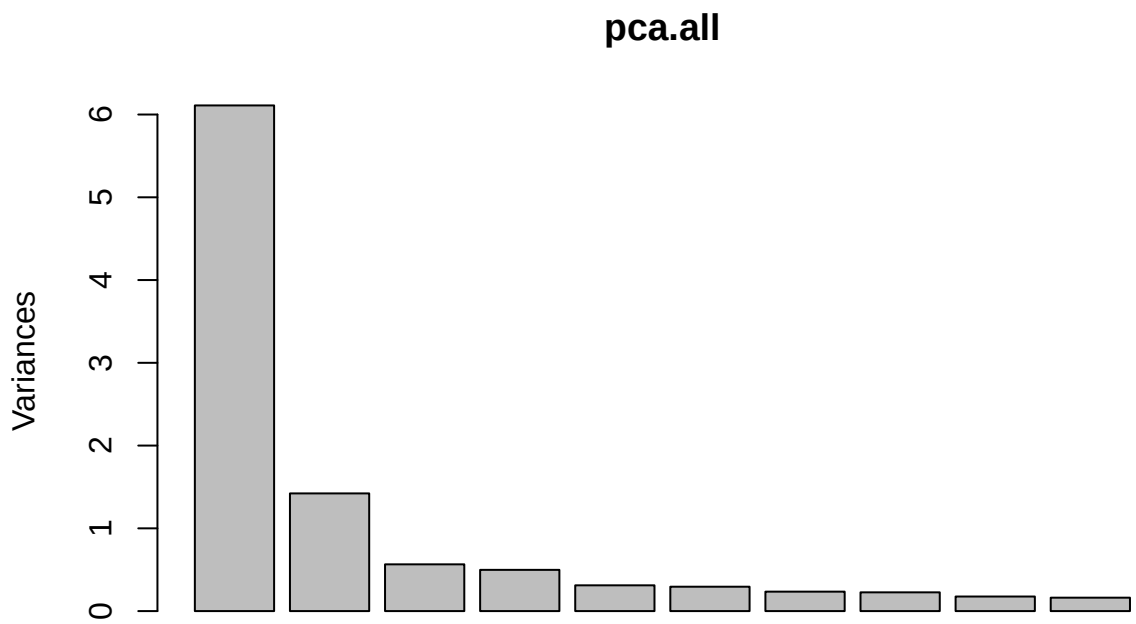
5.5 PVE

How much variability do PC1 and PC2 explain?

```
knitr::kable(summary(pca.all)$importance)
```

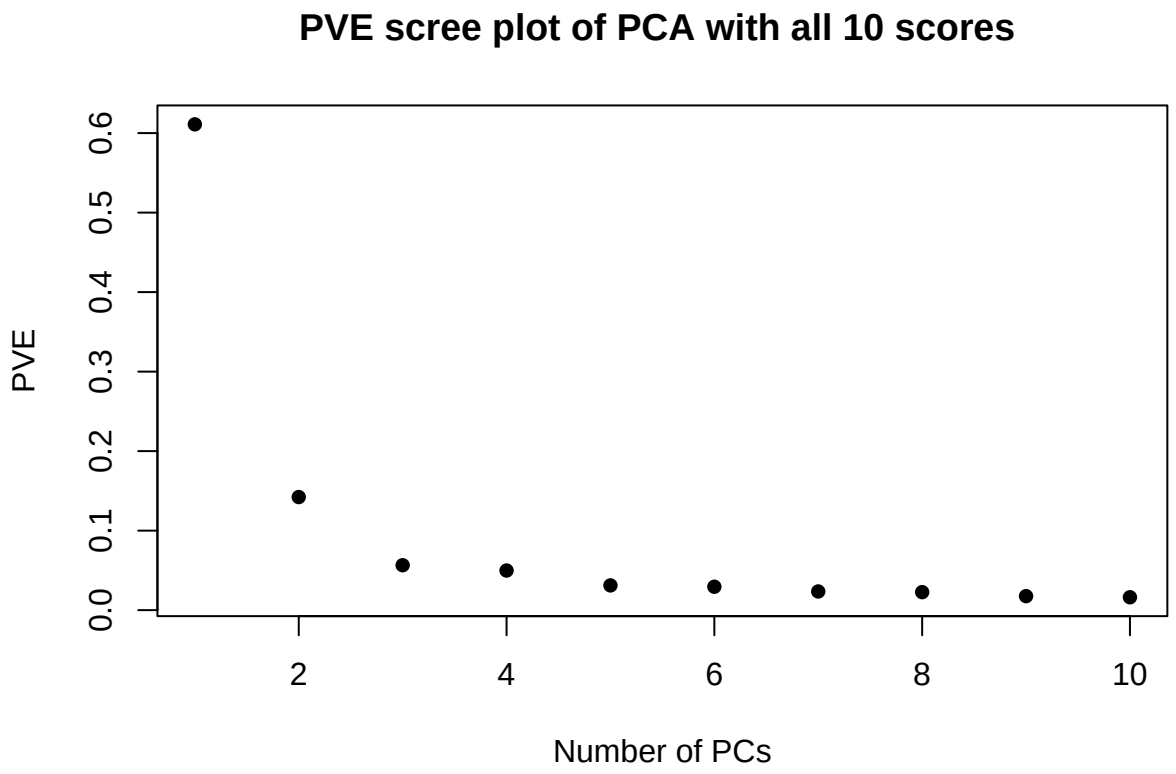
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Standard deviation	2.472	1.193	0.752	0.706	0.558	0.543	0.485	0.476	0.420	0.403
Proportion of Variance	0.611	0.142	0.056	0.050	0.031	0.029	0.023	0.023	0.018	0.016
Cumulative Proportion	0.611	0.753	0.810	0.860	0.891	0.920	0.944	0.966	0.984	1.000

```
plot(pca.all)
```



PVE plot

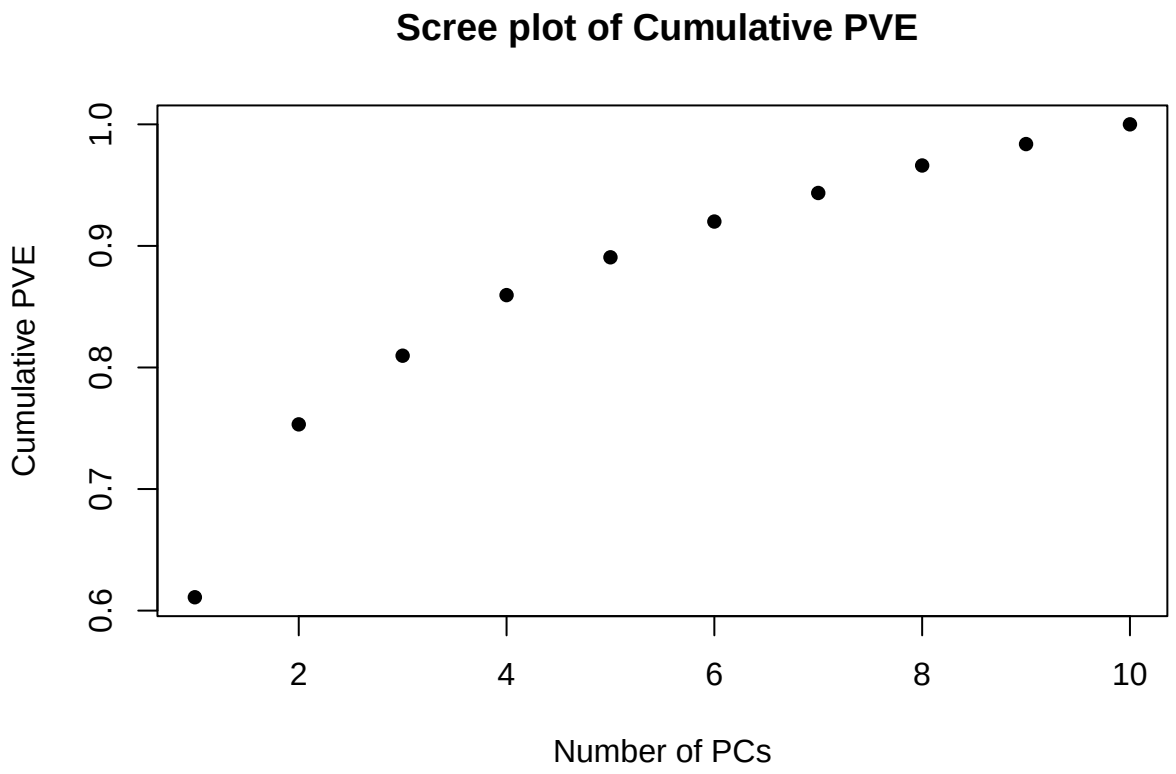
```
plot(summary(pca.all)$importance[2, ], pch=16,
     ylab="PVE",
     xlab="Number of PCs",
     main="PVE scree plot of PCA with all 10 scores ")
```



We see that PC1 accounts for 61% of the total variation in the 10 scores following by PC2 with 14%. With only two leading PCs we capture about 75% of the variance.

The scree plot of CPVE

```
plot(summary(pca.all)$importance[3, ], pch=16,
     ylab="Cumulative PVE",
     xlab="Number of PCs",
     main="Scree plot of Cumulative PVE ")
```



75% of the total variability are explained or captured by the two leading PCs.

To capture the main features of all the tests we could use two summary scores

- PC1: Total scores (weighted)
- PC2: Difference between intelligence(such as math/words?) and dexterity

6 Recap

Principal Component Analysis finds linear combinations of the variables that capture the most information contained in the full data. We may even find some striking relationships among variables through a few PCs. It is often useful to reveal group information or to identify clusters. All PCs are orthogonal which can be an advantageous property when we use them as predictors. The drawback is that we may lose the interpretation based on the original variables.

7 Appendix 1: PC definitions via maximizing the variance of linear combinations.

In this section we write formal definition of PCs with four `AFQT` tests.

7.1 First Principal Component

We are looking for a linear transformation Z_1 of $X_1 =$ `Word`, $X_2 =$ `Parag`, $X_3 =$ `Math`, and $X_4 =$ `Arith` to have the max variance.

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \phi_{31}X_3 + \phi_{41}X_4$$

such that

$$\max_{\phi_{11}^2+\phi_{21}^2+\phi_{31}^2+\phi_{41}^2=1} \text{Var}(Z_1)$$

7.2 Second Principal Component

Similarly, we are looking for another linear transformation Z_2 of $X_1 =$ `Word`, $X_2 =$ `Parag`, $X_3 =$ `Math`, and $X_4 =$ `Arith` to have the max variance and Z_1 is orthogonal to Z_2.

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + \phi_{32}X_3 + \phi_{42}X_4$$

such that

$$\max_{\phi_{12}^2+\phi_{22}^2+\phi_{32}^2+\phi_{42}^2=1} \text{Var}(Z_2)$$

By definitions we know two sets of loadings are perpendicular: or $(\phi_{11}, \phi_{21}, \phi_{31}, \phi_{41})$ and $(\phi_{12}, \phi_{22}, \phi_{32}, \phi_{42})$ are orthogonal.

7.3 More PC components

We keep going to obtain Z_3, and Z_4.

8 Appendix 2: PCA and Eigen decomposition Correlation Matrix

How to get all the loadings, PCs? There are elegant, simple mathematics behind it.

To find the PC loadings we want to maximize the variance of the linear combination of the variables. Let $X = (X_1, X_2, \ldots, X_p)$ be the design matrix. We simply list all values of first variable X_1 for all subjects, and with similar ways to list X_2 and so on. Notice that the design matrix is an $n \times p$ matrix.

It is easy to show that the PC loadings are nothing but eigenvectors/values of $corr(X_1, X_2, \ldots, X_p) = X^\top X/(n-1)$ (if centered and scaled) or $cov(X_1, X_2)$ (unscaled).

Let us use `data.AFQT` which has 50 people and 4 variables `Word`, `Parag`, `Math` and `Arith`.

Eigenvectors of `cor(data.AFQT)` give us the loadings with ordered PC1, PC2 and so on. Eigenvalues are the variances of PC1, PC2 and so on.

```
PC.eig <- eigen(cor(data.AFQT))
PC.eig$vectors   # Loadings
```

```
##       [,1]   [,2]   [,3]   [,4]
## [1,] 0.509  0.442  0.365  0.642
## [2,] 0.500  0.546 -0.308 -0.598
## [3,] 0.496 -0.483 -0.652  0.310
## [4,] 0.494 -0.523  0.589 -0.368
```

```
EigenVectors <- as.data.frame(PC.eig$vectors)
names(EigenVectors) <- c("Eigvec.1", "Eigvec.2", "Eigvec.3", "Eigvec.4")
# create eigen vectors
PC.eig$values    # Variances of each PCs
```

```
## [1] 3.061 0.488 0.271 0.180
```

Let us check against `prcomp()` output:

```r
# We use prcomp() here
PC <- prcomp(data.AFQT, scale=TRUE)
PC  # should be exactly same as PCs from eigen decomposition (up to the sign)
```

```
## Standard deviations (1, .., p=4):
## [1] 1.750 0.699 0.521 0.424
##
## Rotation (n x k) = (4 x 4):
##         PC1    PC2    PC3    PC4
## Word  0.509  0.442 -0.365 -0.642
## Parag 0.500  0.546  0.308  0.598
## Math  0.496 -0.483  0.652 -0.310
## Arith 0.494 -0.523 -0.589  0.368
```

```r
phi <- PC$rotation
phi
```

```
##         PC1    PC2    PC3    PC4
## Word  0.509  0.442 -0.365 -0.642
## Parag 0.500  0.546  0.308  0.598
## Math  0.496 -0.483  0.652 -0.310
## Arith 0.494 -0.523 -0.589  0.368
```

```r
cbind(EigenVectors, PC$rotation) # Putting the first PC together
# one from eigen-values???the other from prcomp().
# They should be exactly the same (to the sign) and they are the same.
```

Eigenvectors and PC rotations are the same up to a sign difference. Are you convinced that PC loadings are the same as eigenvectors of correlation matrix of variables?

9 Appendix 3: PCA and SVD

A matrix X can be decomposed by Singular Value Decomposition (SVD). PCs can be obtained through SVD. SVD is very useful in applications, e.g., matrix completion, recommendation systems. Assume that X is centered and scaled.

Fact: any matrix X can be decomposed as follows:

$$X_{n\times p} = U_{n\times p} D_{p\times p} V^\top_{n\times p}$$

Here U is column orthonormal and it is call left singular vector for each column. V is right singular vector of orthonormal matrix. D is a diagonal matrix with decreasing values $d_1 > d_2, ... > d_p$ and it is call singular values accordingly.

9.1 Properties of SVD

1. Matrix of rank 1 representation

Rewrite the matrix SVD to a sum of rank 1 matrices

$$X_{n\times p} = d_1 u_1 v_1^\top + d_2 u_2 v_2^\top + \dots d_p u_p v_p^\top$$

Here $u_1, \dots, u_p$ are columns of U and $v_1, \dots, v_p$ are columns of V with norm 1, i.e. $\|u_j\|_2 = 1$ for $j = 1, \dots, p$.

2. Since $d_1, \dots, d_p$ are decreasing, we may take top singular vectors to approximate the matrix X.
3. It is easy to see $v_1, \dots, v_p$ is the eigenvectors of $X^\top X$ and $d_1^2, \dots, d_p^2$ are corresponding eigenvalues. So the right singular vectors $v_1, \dots, v_p$ give us the loadings for PCs.

It is easy to prove by plugging in $X = UDV^\top$ and notice that U and V are orthonormal, i.e., $U^\top U = I$ and $V^\top V = I$. Immediately we get

$$\frac{X^\top X}{n-1} V = \frac{1}{n-1} V D U^\top U D V^\top V = V \frac{D^2}{n-1}$$

4. $XV = UD$ that means PC scores $= UD$.

How beautiful!

9.2 Compare PCA and SVD

Let us verify this using function `svd()`. We use `data.AFQT` again.

```
data.AFQT <- AFQT.sub %>% select(Word, Parag, Math, Arith)
data.AFQT.center.scale <- scale(data.AFQT, scale = TRUE)

pc.4 <- prcomp(data.AFQT.center.scale, scale = TRUE)
AFQT.svd <- svd(data.AFQT.center.scale)
names(AFQT.svd)
```

```
## [1] "d" "u" "v"
```

9.2.1 PC loadings

Right singular vectors $v_1, \ldots, v_p$ are PC loadings.

```
AFQT.svd$v
```

```
##       [,1]   [,2]   [,3]   [,4]
## [1,] 0.509  0.442 -0.365 -0.642
## [2,] 0.500  0.546  0.308  0.598
## [3,] 0.496 -0.483  0.652 -0.310
## [4,] 0.494 -0.523 -0.589  0.368
```

Compare with PC loadings

```
pc.4$rotation
```

```
##         PC1    PC2    PC3    PC4
## Word  0.509  0.442 -0.365 -0.642
## Parag 0.500  0.546  0.308  0.598
## Math  0.496 -0.483  0.652 -0.310
## Arith 0.494 -0.523 -0.589  0.368
```

9.2.2 PC scores

Let's take a look at PC1 first. $PC1 = d_1 u_1$

```
d1.u1 <- AFQT.svd$d[1] * AFQT.svd$u[, 1]
pc1 <- prcomp(data.AFQT.center.scale)$x[, 1]
cbind(d1.u1, pc1)[1:5, ]
```

```
##     d1.u1    pc1
## p1 -0.920 -0.920
## p2  0.611  0.611
## p3 -0.312 -0.312
## p4  3.069  3.069
## p5  2.818  2.818
```

We compare PC scores computed by UD and by `prcomp()`.

```
(AFQT.svd$u %*% diag(AFQT.svd$d) - pc.4$x)[1:5,]
```

```
##          PC1       PC2       PC3      PC4
## p1 -2.33e-15 -6.25e-16 -6.66e-16 2.22e-16
## p2 -3.33e-16 -6.66e-16  0.00e+00 8.85e-17
## p3  3.89e-16  4.44e-16  1.39e-16 2.22e-16
## p4  0.00e+00  2.84e-16  3.89e-16 1.28e-15
## p5  0.00e+00  0.00e+00  4.72e-16 1.14e-15
```

9.2.3 Variance of PC scores

Variance of PC scores are $d_i^2/(n-1)$.

```
var.pc <- (pc.4$sdev)^2
var.pc.svd <- AFQT.svd$d^2/(nrow(data.AFQT)-1)
cbind(var.pc, var.pc.svd)
```

```
##      var.pc var.pc.svd
## [1,]  3.061      3.061
## [2,]  0.488      0.488
## [3,]  0.271      0.271
## [4,]  0.180      0.180
```

10 Appendix 4: Missing values and recommender system

Often datasets have missing values, which can be a nuisance. Many data analysis functions will simply delete the rows with missing values. Other examples such as recommender systems where based on what are known we may come up with informative recommendations for the missing cells. For instance, we may form a matrix X of the movie ratings that n customers have given to the entire catalog of p movies. Most of the matrix will be missing, since no customer will have seen and rated more than a tiny fraction of the catalog.

10.1 Case study: recommender system to provide favorite movies

Let us look at the real dataset collected from MovieLens website. We use the small one which contains $100,836$ ratings to $p = 9,742$ movies by $n = 610$ users. The data contains variable `userId`, `movieId`, `rating` and `timestamp`. Each row is one rating, which is what we have termed a long form. **We hope to fill in all the missing cells so that we can recommend a user movies that he/she is very likely enjoy watching.**

10.1.1 A quick EDA

Read the data and examine the format of the variables. We format the time first into readable time scale.

```
movieLens_raw <- read_csv('MovieLens.csv')
```

```
## Rows: 100836 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): userId, movieId, rating, timestamp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
names(movieLens_raw)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp"
```

```
dim(movieLens_raw)
```

```
## [1] 100836      4
```

```
head(movieLens_raw, 2)  # notice timestmp is a real number
```

```
## # A tibble: 2 x 4
##   userId movieId rating timestamp
##    <dbl>   <dbl>  <dbl>     <dbl>
## 1      1       1      4 964982703
## 2      1       3      4 964981247
```

```
# convert timestamp into readable time using `lubridate::as_datetime()`
movieLens_raw <- movieLens_raw %>%
  mutate(time = as_datetime(timestamp))
head(movieLens_raw, 10)
```

```
## # A tibble: 10 x 5
##    userId movieId rating timestamp time
##     <dbl>   <dbl>  <dbl>     <dbl> <dttm>
##  1      1       1      4 964982703 2000-07-30 18:45:03
##  2      1       3      4 964981247 2000-07-30 18:20:47
##  3      1       6      4 964982224 2000-07-30 18:37:04
##  4      1      47      5 964983815 2000-07-30 19:03:35
##  5      1      50      5 964982931 2000-07-30 18:48:51
##  6      1      70      3 964982400 2000-07-30 18:40:00
##  7      1     101      5 964980868 2000-07-30 18:14:28
##  8      1     110      4 964982176 2000-07-30 18:36:16
##  9      1     151      5 964984041 2000-07-30 19:07:21
## 10      1     157      5 964984100 2000-07-30 19:08:20
```

How many unique users and movies?

```
# number of unique movies
length(unique(movieLens_raw$movieId))
```

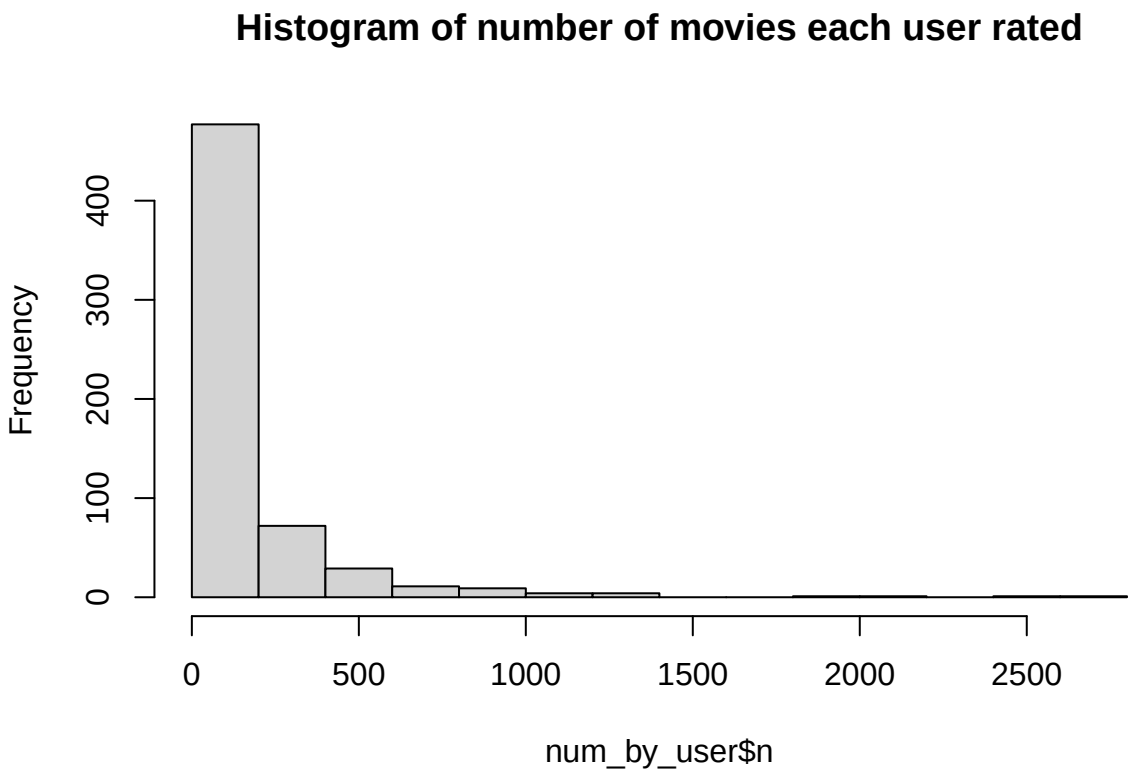
```
## [1] 9724
```

```
# number of unique users
length(unique(movieLens_raw$userId))
```

```
## [1] 610
```

How many movies each user rated? How many movies each user rated on average?

```
num_by_user <- movieLens_raw %>%
  group_by(userId) %>%
  summarize(n = n())

hist(num_by_user$n,
     main = "Histogram of number of movies each user rated")
```




```
# the user who rates the most movies
max(num_by_user$n)
```

```
## [1] 2698
```

```
# average number of rated movies by user
mean(num_by_user$n)
```

```
## [1] 165
```

```
# average proportion of rated movies by user
mean(num_by_user$n)/length(unique(movieLens_raw$movieId))
```

```
## [1] 0.017
```

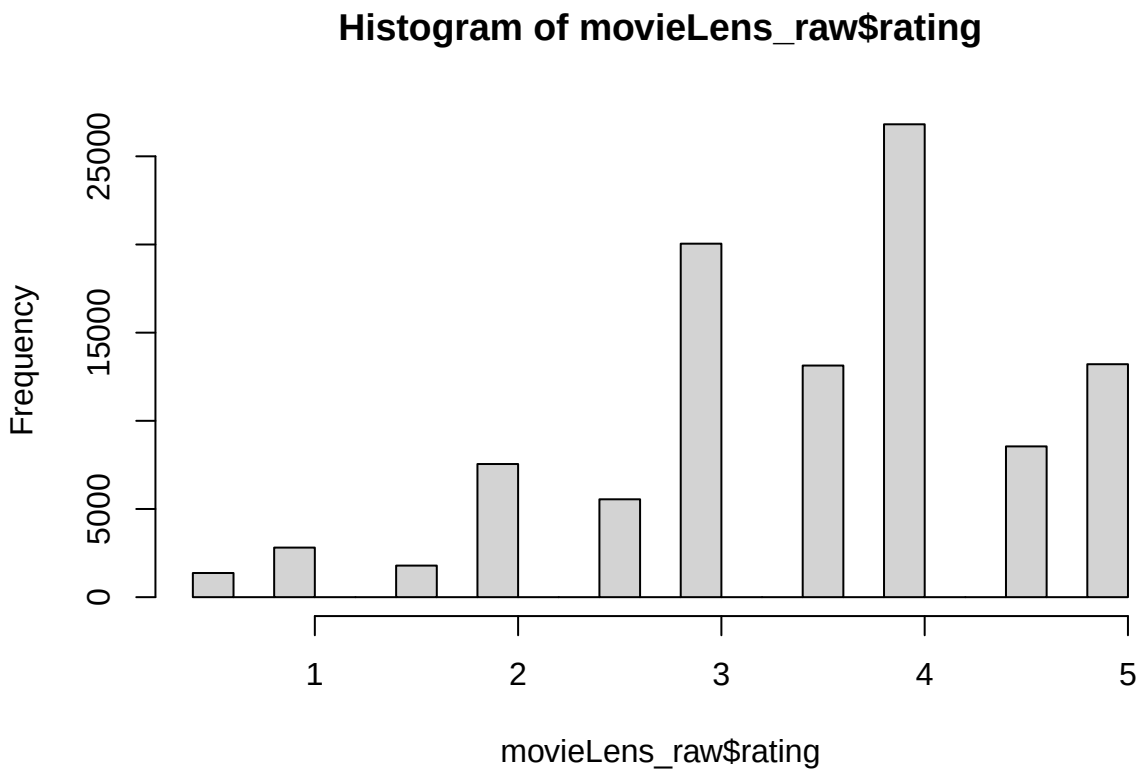
Let's first look at the histogram of ratings.

```
movieLens_raw %>%
  group_by(rating) %>%
  summarise(n = n())
```

```
## # A tibble: 10 x 2
##    rating     n
##     <dbl> <int>
##  1    0.5  1370
##  2    1    2811
##  3    1.5  1791
##  4    2    7551
##  5    2.5  5550
```

```
##  6    3   20047
##  7    3.5 13136
##  8    4   26818
##  9    4.5  8551
## 10    5   13211
```

```
hist(movieLens_raw$rating)
```



10.1.2 Data preparation

We need to first convert from long to wide format using `pivor_wider()` so that

- each row is the ratings of movies from one user
- each column is the ratings of one movie from users
- we should expect many entries are `NA`

We convert the data from long to wide so that it becomes a matrix (with `NA`s) to apply matrix completion algorithms (such as `softImpute()` we will use later).

```
movieLens <- movieLens_raw %>%
  # select(-timestamp, -time) %>%
  mutate(userId = paste0('user', userId)) %>%
  pivot_wider(id_cols = userId,  # each row
              names_from = movieId, names_prefix = 'movie',
              values_from = rating)

## pivot back to longer format
# movieLens %>%
```

```
#   pivot_longer(cols = starts_with("movie"),
#                names_to = "movieId",
#                values_to = "rating")

movieLens[1:5, 1:5]
```

```
## # A tibble: 5 x 5
##   userId movie1 movie3 movie6 movie47
##   <chr>   <dbl>  <dbl>  <dbl>   <dbl>
## 1 user1       4      4      4       5
## 2 user2      NA     NA     NA      NA
## 3 user3      NA     NA     NA      NA
## 4 user4      NA     NA     NA       2
## 5 user5       4     NA     NA      NA
```

```
sum(is.na(movieLens))
```

```
## [1] 5830804
```

Digital streaming services like Netflix and Amazon use data about the content that a customer has viewed in the past, as well as data from other customers, to suggest other content for the customer. If we can impute the missing values well, we will have an idea of what each customer will think of movies they have not yet seen and be able to suggest a movie that a particular customer might like. Principal components analysis would be an useful tool to impute the missing values.

10.2 Objective function

Given the data matrix $X \in \mathbb{R}^{n \times p}$, some of the observations x_{ij} are missing and $\mathcal{O}$ denotes the set of all observed pairs of indices (i, j), a subset of the possible $n \times p$ pairs. Given a pre-determined number M of components, our goal is to find low rank approximations $U \in \mathbb{R}^{n \times M}$ and $V \in \mathbb{R}^{M \times p}$ which minimize

$$\sum_{(i,j) \in \mathcal{O}} \left(x_{ij} - \sum_{m=1}^{M} d_m u_{im} v_{jm} \right)^2 .$$

In other words, we are trying to find the best approximation $\widehat{U} \in \mathbb{R}^{n \times M}$ and $\widehat{V} \in \mathbb{R}^{M \times p}$ based on observed entries. Once we solve this problem, we can estimate a missing observation x_{ij} using

$$\widehat{x}_{ij} = \sum_{m=1}^{M} d_m \widehat{u}_{im} \widehat{v}_{jm}$$

where $\widehat{u}_{im}$ and $\widehat{v}_{jm}$ are the (i, m) and (j, m) elements, respectively, of the best approximation $\widehat{U} \in \mathbb{R}^{n \times M}$ and $\widehat{V} \in \mathbb{R}^{M \times p}$.

10.3 Algorithm

It turns out that solving this problem exactly is difficult, unlike in the case of complete data: the vanilla PCA no longer applies. However, many researchers found that iteratively applying the vanilla PCA (or SVD) provides a good solution. The R package named `softImpute` yields a nice approximation, based on this simple idea.

10.4 Recommender system

As we have seen before, the recommender system aims to impute the missing values of rating. The key idea is that the set of movies which the ith customer has seen will overlap with those which other customers have

seen. Furthermore, some of those other customers will have similar movie preferences to the ith customer. Thus, we may use similar customers' movies ratings that the ith customer has not seen to predict whether the ith customer will like those movies.

We can use the same imputing algorithm to predict the ith customer's rating. `softImpute`. More concretely, the ith customer's rating would be

$$\widehat{x}_{ij} = \sum_{m=1}^{M} \widehat{u}_{im}\widehat{v}_{jm}$$

Here, $\widehat{u}_{im}$ represents the strength with which the ith user belongs to the mth clique, where a clique is a group of customers that enjoys movies of the mth genre. Not only that, $\widehat{v}_{jm}$ represents the strength with which the jth movie belongs to the mth genre.

We implement recommender system through `softImpute` package with the choice $M = 5$, the number of hidden components. The result is stored as an `svd` object named `fit` with components `u`, `d`, and `v`.

```
fit <- movieLens %>% select(-userId) %>%
  as.matrix.data.frame() %>%
  softImpute(type='als', rank.max=5)

str(fit) #hist(fit$d)
```

```
## List of 3
##  $ u: num [1:610, 1:5] -0.0654 -0.0264 -0.0177 -0.0495 -0.0229 ...
##  $ d: num [1:5] 3288 1362 1180 990 955
##  $ v: num [1:9724, 1:5] -0.027 -0.017 -0.027 -0.027 -0.0293 ...
##  - attr(*, "lambda")= num 0
##  - attr(*, "call")= language softImpute(x = ., rank.max = 5, type = "als")
```

The following code yields the predicted rating matrix.

```
movieLens_pred <- fit$u %*% diag(fit$d) %*% t(fit$v)  # may try complete()
colnames(movieLens_pred) <- colnames(movieLens)[-1]

movieLens_pred <- as_tibble(movieLens_pred) %>%
  add_column(userId = pull(movieLens, userId), .before = 'movie1')
movieLens_pred[1, 1:20] # movieLens[1, 1:20]
```

```
## # A tibble: 1 x 20
##   userId movie1 movie3 movie6 movie47 movie50 movie70 movie101 movie110 movie151
##   <chr>   <dbl>  <dbl>  <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 user1    4.49   3.93   4.45    4.30    5.26    3.04     4.84     4.49     4.58
## # i 10 more variables: movie157 <dbl>, movie163 <dbl>, movie216 <dbl>,
## #   movie223 <dbl>, movie231 <dbl>, movie235 <dbl>, movie260 <dbl>,
## #   movie296 <dbl>, movie316 <dbl>, movie333 <dbl>
```

```
#sum(is.na(movieLens_pred))   #sum(is.na(movieLens))
```