Principal Component Analysis in R

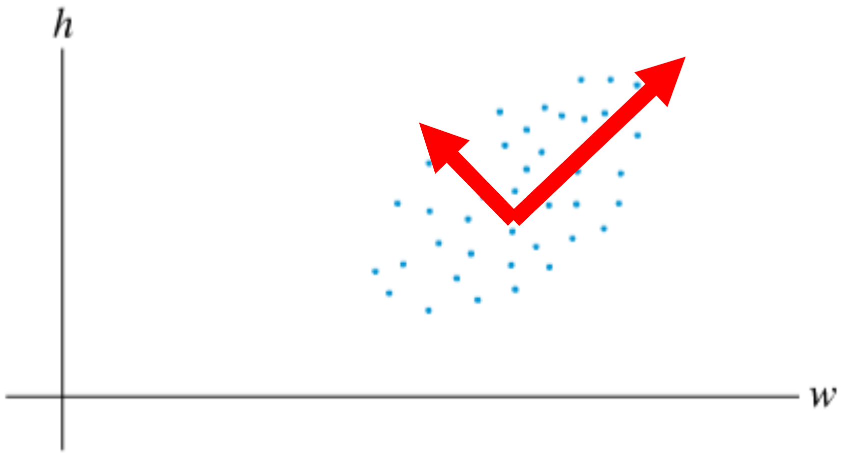
Anahita Zarei

PCA - Review

- Principal Component Analysis is a technique for exploratory data analysis.
- It’s specifically useful in cases where there are too many features.
- PCA assists in identifying which samples are similar to one another and which are different. This will tell you which variables make one group different from another.
- PCA directions are the directions where the data is most spread out and has the highest variance.
- Variables that correlate with one another, will all contribute strongly to the same principal component.

PCA - Review

- Eigenvectors of a cov/cor matrix are the new orthogonal directions and the corresponding eigenvalues show **how much variance** there is in the data in that direction.
- Reshaping a dataset based on PCA's does not change the data itself. We're just looking at it from a different angle, which should represent the data better.



Longley Data Set

```
> head(longley)
     GNP.deflator     GNP Unemployed Armed.Forces Population Year Employed
1947         83.0 234.289      235.6        159.0    107.608 1947   60.323
1948         88.5 259.426      232.5        145.6    108.632 1948   61.122
1949         88.2 258.054      368.2        161.6    109.773 1949   60.171
1950         89.5 284.599      335.1        165.0    110.929 1950   61.187
1951         96.2 328.975      209.9        309.9    112.075 1951   63.221
1952         98.1 346.999      193.2        359.4    113.270 1952   63.639
```

```
> d = longley[,-c(6,7)]
> head(d)
     GNP.deflator     GNP Unemployed Armed.Forces Population
1947         83.0 234.289      235.6        159.0    107.608
1948         88.5 259.426      232.5        145.6    108.632
1949         88.2 258.054      368.2        161.6    109.773
1950         89.5 284.599      335.1        165.0    110.929
1951         96.2 328.975      209.9        309.9    112.075
1952         98.1 346.999      193.2        359.4    113.270
```

Note that there's no categorical variable here.
If there were, you had to exclude them.

`GNP.deflator`

GNP implicit price deflator (*1954=100*)

`GNP`

Gross National Product.

`Unemployed`

number of unemployed.

`Armed.Forces`

number of people in the armed forces.

`Population`

'noninstitutionalized' population ≥ 14 years of age.

`Year`

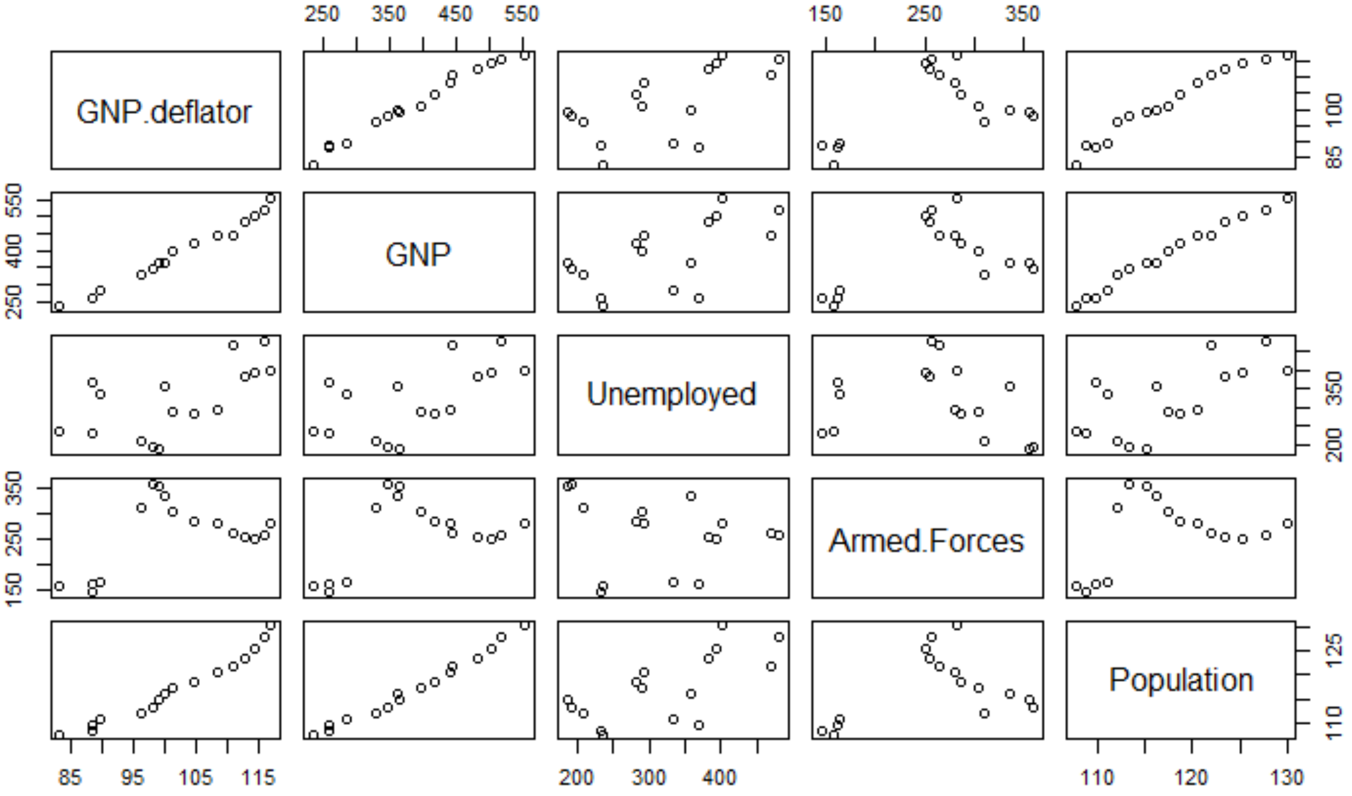
the year (time).

`Employed`

number of people employed.

Scatterplots

`> pairs(d)` : A matrix of scatterplots



Correlation Coefficient

```
library(GGally)
library(dplyr)
d %>%
  ggcorr(label = T)
```

prcomp

```
> library(stats)
> dPCA = prcomp(d, scale = TRUE)

> dPCA
Standard deviations:
[1] 1.89991292 1.08413093 0.44626826 0.12199281 0.03087773

Rotation:
                   PC1         PC2        PC3          PC4          PC5
GNP.deflator 0.5210129 -0.05808997  0.1889153  0.776958379  0.292946852
GNP          0.5199086 -0.05345522  0.3174971 -0.135947010 -0.779455948
Unemployed   0.3658062  0.59532321 -0.7100763  0.004614581 -0.086870665
Armed.Forces 0.2296424 -0.79831473 -0.5511572 -0.078584283 -0.002874243
Population   0.5212397  0.04529867  0.2356355 -0.609637027  0.546878225

> summary(dPCA)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5
Standard deviation     1.8999 1.0841 0.44627 0.12199 0.03088
Proportion of Variance 0.7219 0.2351 0.03983 0.00298 0.00019
Cumulative Proportion  0.7219 0.9570 0.99683 0.99981 1.00000
```

- Setting the “scale” to TRUE, normalizes the centered data such that they have unit variance.
- This is usually handy when the features have different units or their magnitude vary by orders of magnitude.
- Scaling the data corresponds to finding the eigenvalues/eigenvectors of the correlation matrix instead of the covariance matrix.

prcomp attributes

```
> attributes(dPCA)
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
> dPCA$sdev
[1] 1.89991292 1.08413093 0.44626826 0.12199281 0.03087773
```

Standard deviations in each principal component

Sqrt of Eigenvalues of correlation matrix

```
> dPCA$rotation
                   PC1         PC2        PC3          PC4          PC5
GNP.deflator 0.5210129 -0.05808997  0.1889153  0.776958379  0.292946852
GNP          0.5199086 -0.05345522  0.3174971 -0.135947010 -0.779455948
Unemployed   0.3658062  0.59532321 -0.7100763  0.004614581 -0.086870665
Armed.Forces 0.2296424 -0.79831473 -0.5511572 -0.078584283 -0.002874243
Population   0.5212397  0.04529867  0.2356355 -0.609637027  0.546878225
```

Eigenvectors of the correlation matrix (principal axes/principal direction)

prcomp attributes – cont.

The mean and standard deviation for each feature

```
> dPCA$center
GNP.deflator          GNP   Unemployed Armed.Forces   Population
    101.6813     387.6984     319.3313     260.6687     117.4240
> dPCA$scale
GNP.deflator          GNP   Unemployed Armed.Forces   Population
   10.791553    99.394938    93.446425    69.591960     6.956102
```

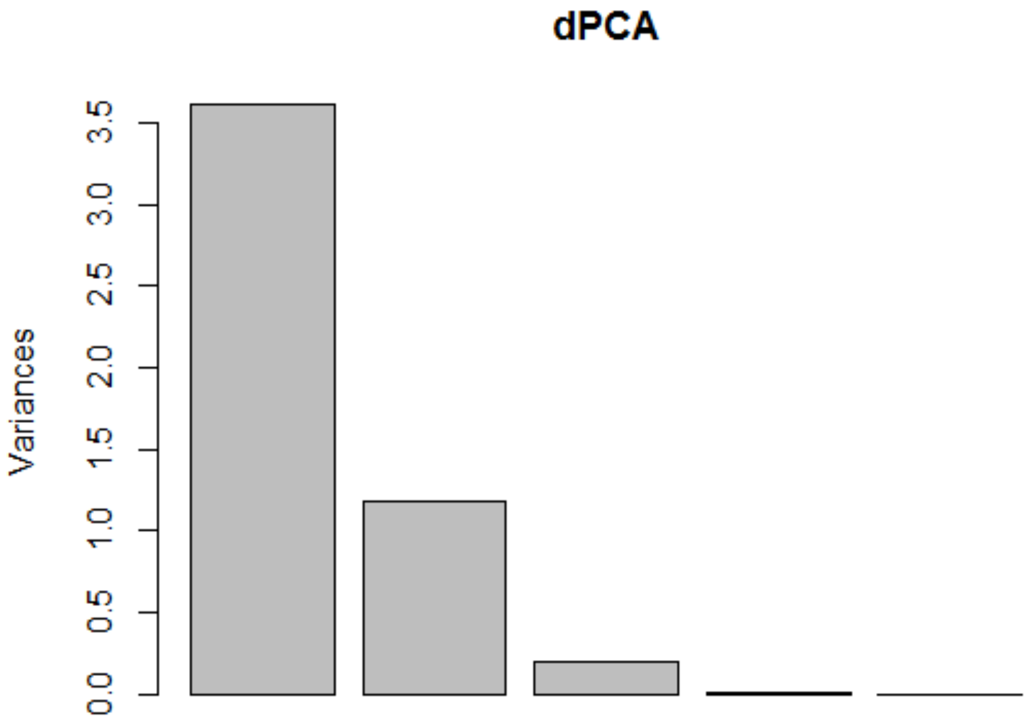
The new coordinates of the transformed data

```
> head(dPCA$x)
            PC1        PC2         PC3          PC4          PC5
1947 -3.1031746  0.7519905  0.29187307 -0.164215614  0.006237107
1948 -2.6857734  0.8495010  0.63281993  0.122620708  0.042355835
1949 -2.0379258  1.5402521 -0.49603161 -0.008466046  0.007863283
1950 -1.8680442  1.2766319 -0.12473100 -0.057963320 -0.043499782
1951 -1.2385402 -1.2356542 -0.02309218  0.093478781 -0.009116973
1952 -0.8650172 -1.9220172 -0.15690972  0.044169280  0.008545465
```

Scree Plots

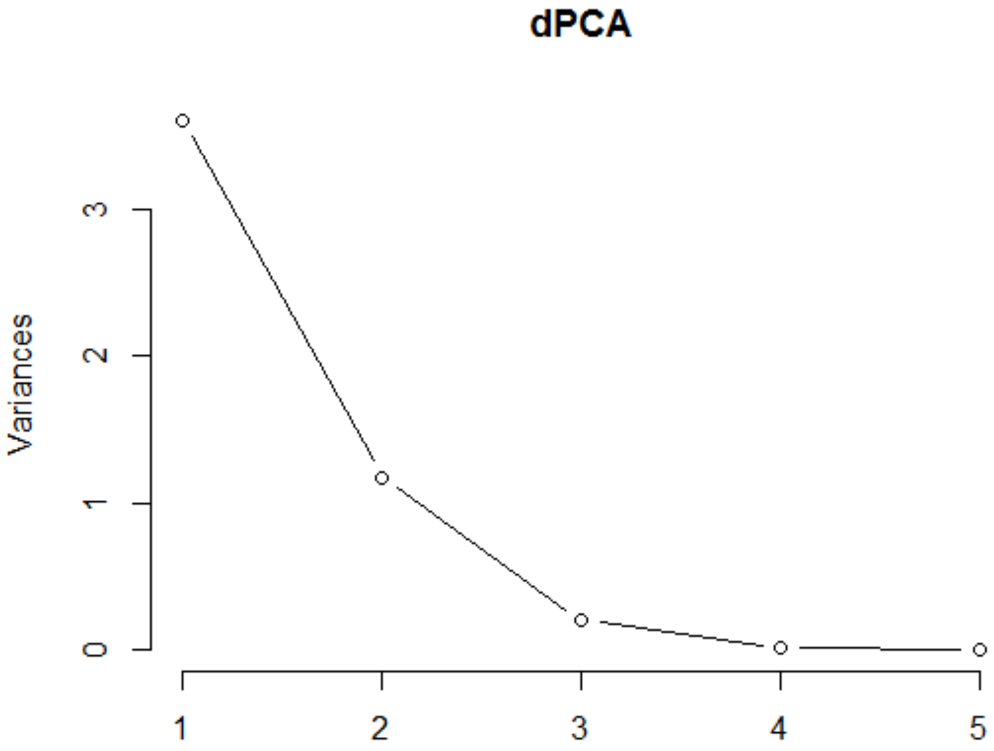
A scree plot displays the proportion of the total variation in a dataset that is explained by each of the components in a principle component analysis. It helps to identify how many of the components are needed to summaries the data.

```
> plot(dPCA)
> screeplot(dPCA)
```



Scree Plots

```
> screeplot(dPCA, type = "l")
```



Checking the values

```
> round(dPCA$sdev ^ 2, 2)
[1] 3.61 1.18 0.20 0.01 0.00
```

Above values match the variances on the plot as expected.

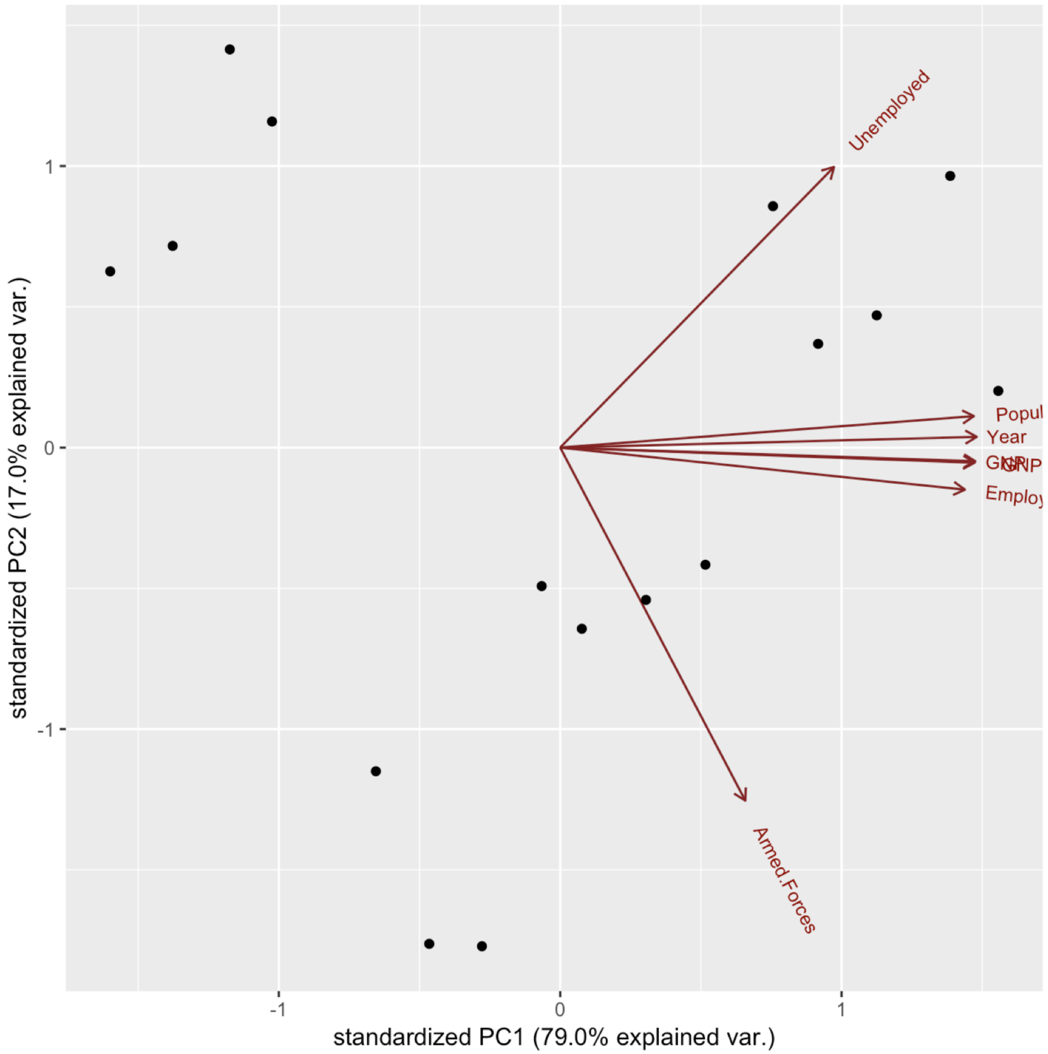
Calculating the Coordinates of New Points

```
> predict(dPCA, newdata = head(d))
            PC1        PC2         PC3          PC4          PC5
1947 -3.1031746  0.7519905  0.29187307 -0.164215614  0.006237107
1948 -2.6857734  0.8495010  0.63281993  0.122620708  0.042355835
1949 -2.0379258  1.5402521 -0.49603161 -0.008466046  0.007863283
1950 -1.8680442  1.2766319 -0.12473100 -0.057963320 -0.043499782
1951 -1.2385402 -1.2356542 -0.02309218  0.093478781 -0.009116973
1952 -0.8650172 -1.9220172 -0.15690972  0.044169280  0.008545465
```

```
> head(dPCA$x)
            PC1        PC2         PC3          PC4          PC5
1947 -3.1031746  0.7519905  0.29187307 -0.164215614  0.006237107
1948 -2.6857734  0.8495010  0.63281993  0.122620708  0.042355835
1949 -2.0379258  1.5402521 -0.49603161 -0.008466046  0.007863283
1950 -1.8680442  1.2766319 -0.12473100 -0.057963320 -0.043499782
1951 -1.2385402 -1.2356542 -0.02309218  0.093478781 -0.009116973
1952 -0.8650172 -1.9220172 -0.15690972  0.044169280  0.008545465
```

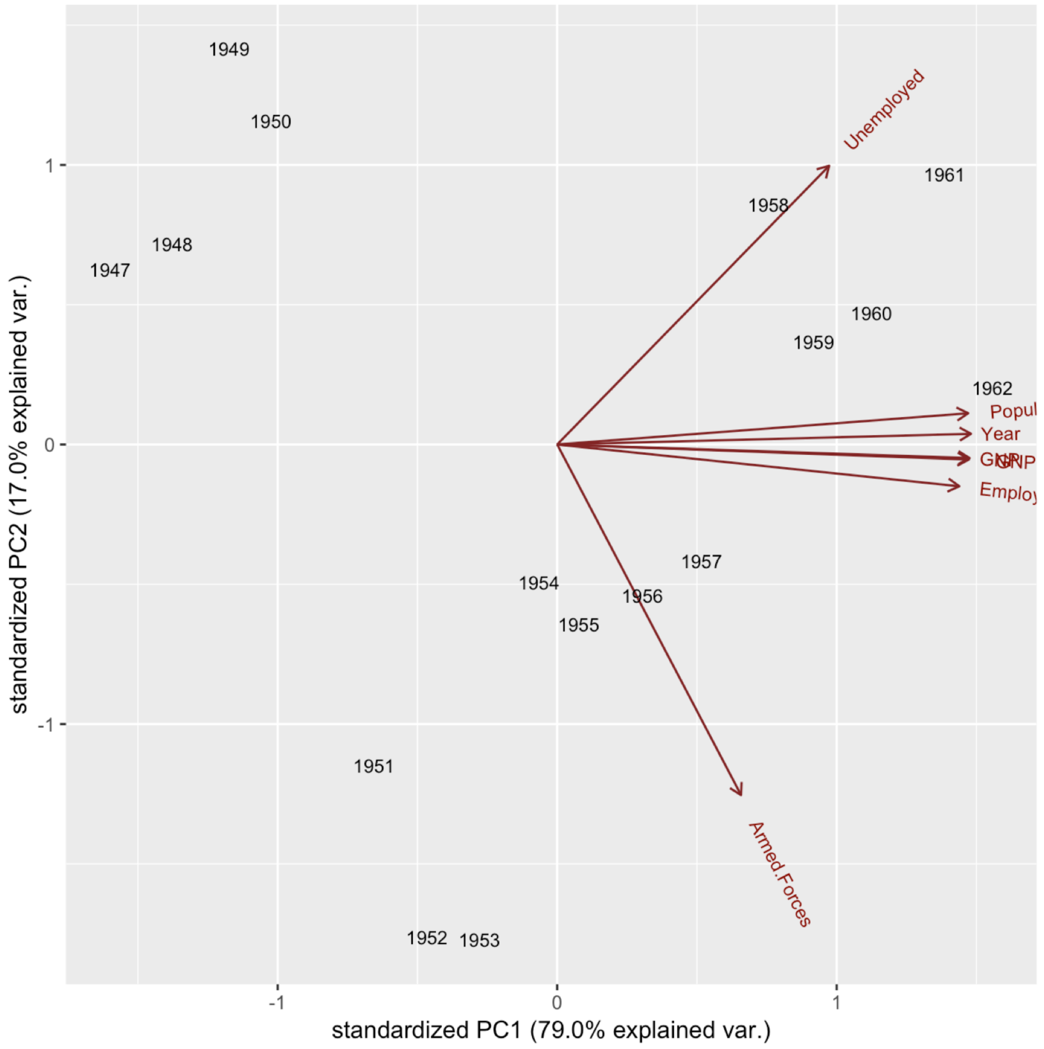
biplot

- A biplot is a plot that will allow you to visualize how the samples relate to one another in the PCA (similar and dissimilar) and will simultaneously reveal how each variable contributes to each principal component, hence the name biplot.
- Here, you see that the variables pop, year, GNP, and employed all contribute to PC1, with higher values in those variables moving the samples to the right on this plot.
- This lets you see how the data points relate to the axes, but it's not very informative without knowing which point corresponds to which sample




```
library(ggbiplot)
ggbiplot(dPCA)
```

biplot

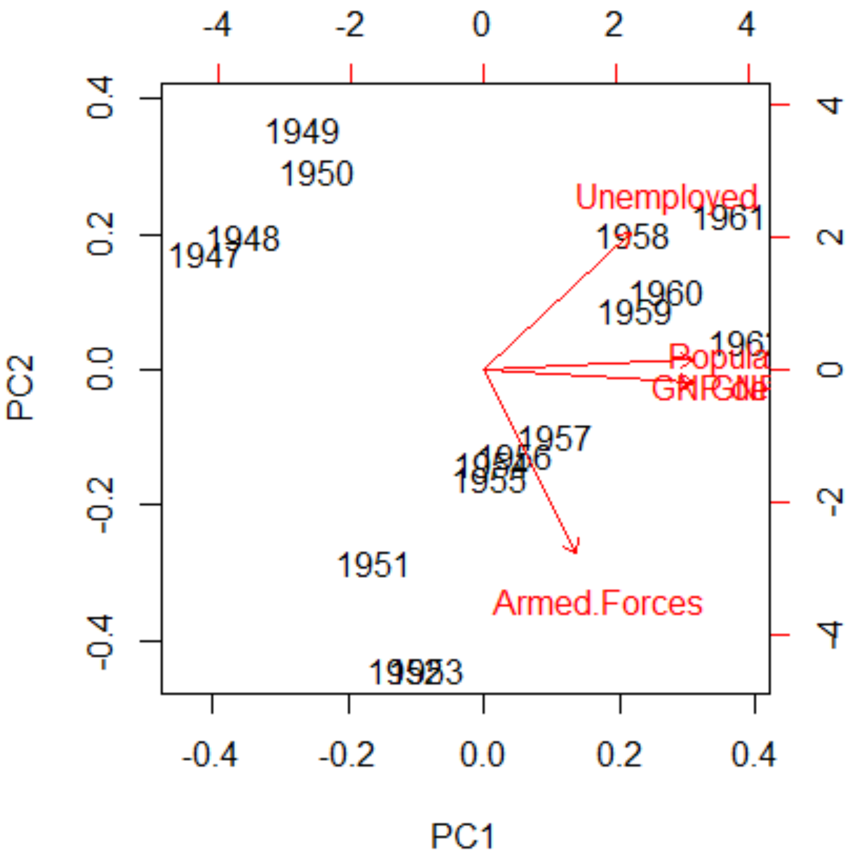


Provide the rownames as labels.

```
ggbiplot(dPCA, labels = rownames(d))
```

biplot

```
> biplot(dPCA)
```



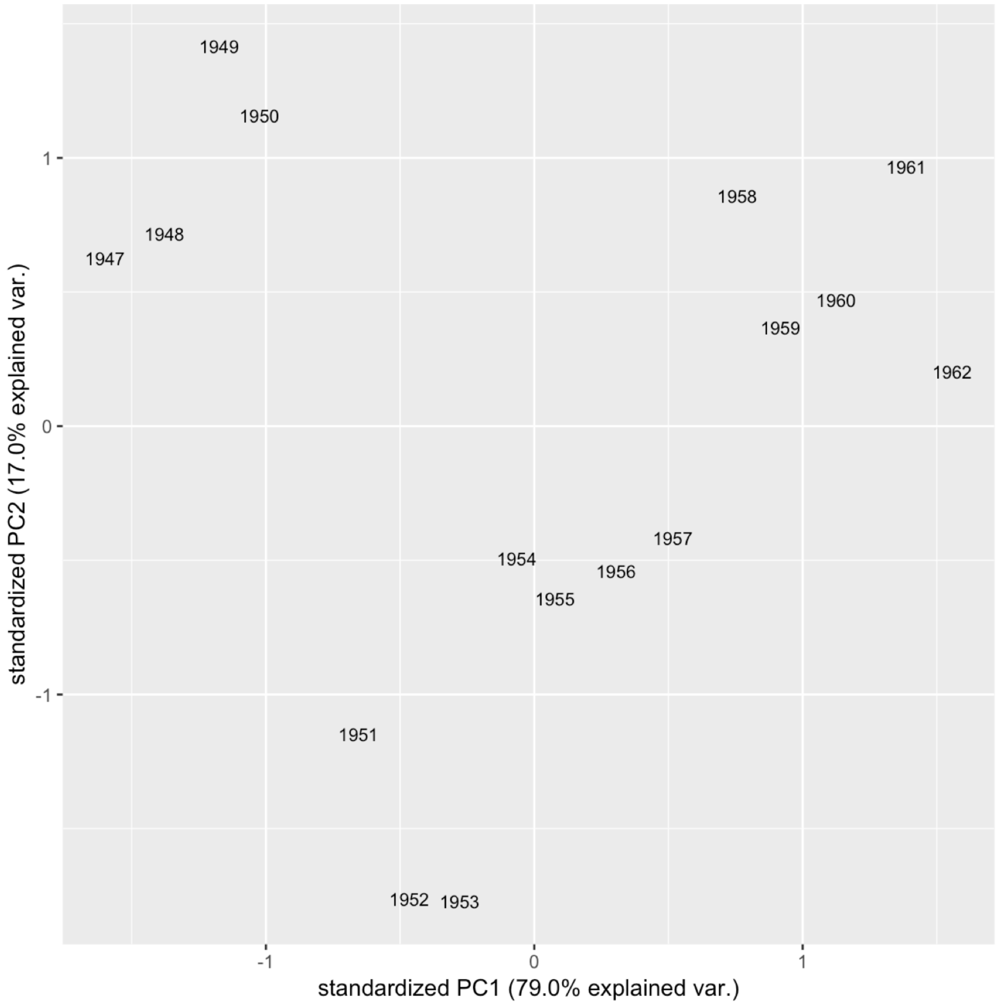

```
> dPCA
Standard deviations:
[1] 1.89991292 1.08413093 0.44626826 0.12199281 0.03087773

Rotation:
                   PC1         PC2        PC3          PC4          PC5
GNP.deflator 0.5210129 -0.05808997  0.1889153  0.776958379  0.292946852
GNP          0.5199086 -0.05345522  0.3174971 -0.135947010 -0.779455948
Unemployed   0.3658062  0.59532321 -0.7100763  0.004614581 -0.086870665
Armed.Forces 0.2296424 -0.79831473 -0.5511572 -0.078584283 -0.002874243
Population   0.5212397  0.04529867  0.2356355 -0.609637027  0.546878225
```

Removing the axes

You may see the clusters better if you remove the arrows.

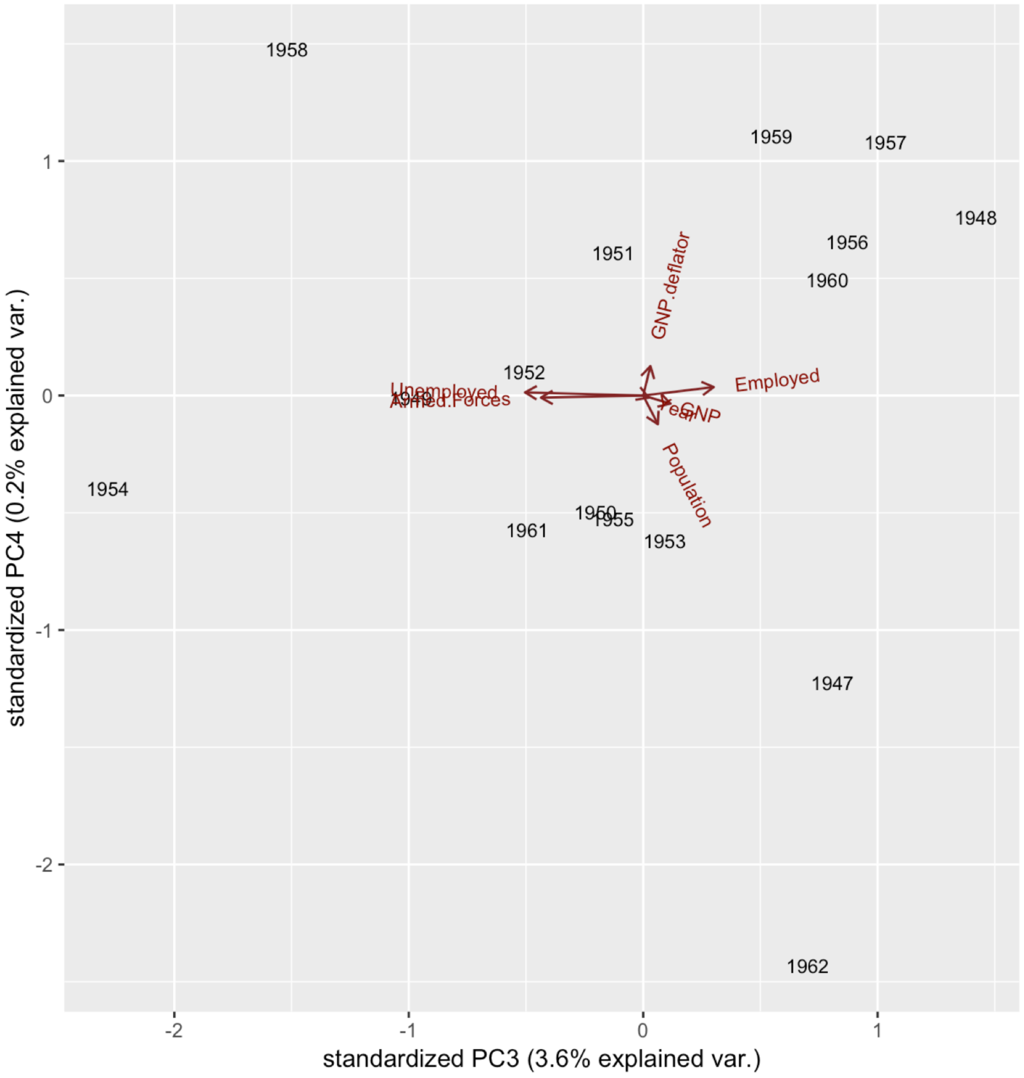
```
ggbiplot(dPCA, labels = rownames(d), var.axes = F)
```



Plotting for other PC’s

- You don't see clear clusters here, but it shouldn‘t be too surprising.
- PC3 and PC4 explain very small percentages of the total variation, so it would be surprising if you found that they were very informative and separated the groups or revealed apparent patterns.

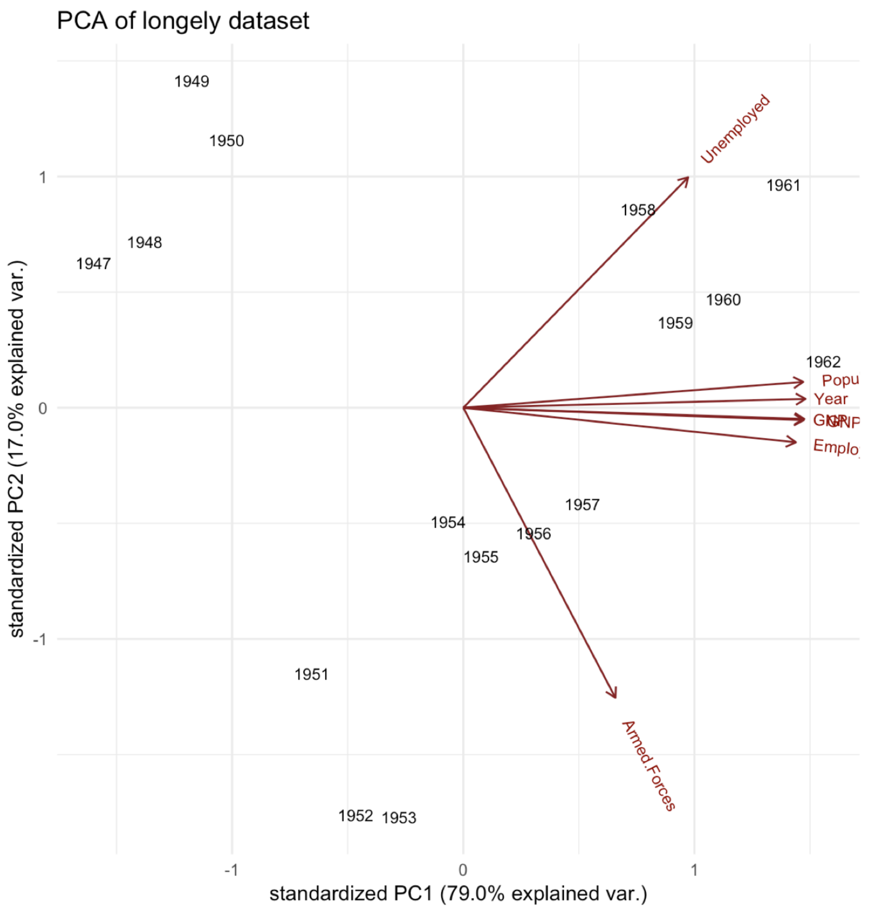
```
ggbiplot(dPCA, labels = rownames(d), choices = c(3,4))
```



More Plotting Options

- As ggbiplot is based on the ggplot function, therefore same set of graphical options are available to you to alter your biplots.
- Example:

```
ggbiplot(dPCA, labels = rownames(d)) +
  ggtitle("PCA of longely dataset")
```



Another Method for Finding PCA

- princomp can also be used to find PCA.
- Princomp uses eigenvalues of covariance/correlation matrix to find PC's.
- Prcomp uses singular value decomposition to find PC's.

```
> dPCA3 = princomp(d, cor = TRUE, scores = TRUE)

> summary(dPCA3)
Importance of components:
                          Comp.1    Comp.2     Comp.3      Comp.4       Comp.5
Standard deviation     1.8999129 1.0841309 0.44626826 0.121992810 0.0308777297
Proportion of Variance 0.7219338 0.2350680 0.03983107 0.002976449 0.0001906868
Cumulative Proportion  0.7219338 0.9570018 0.99683286 0.999809313 1.0000000000

> head(dPCA3$scores)
       Comp.1     Comp.2     Comp.3       Comp.4       Comp.5
1947 3.204945 -0.7766524  0.3014452 -0.169601157  0.006441657
1948 2.773855 -0.8773608  0.6535736  0.126642123  0.043744918
1949 2.104761 -1.5907655 -0.5122992 -0.008743694  0.008121164
1950 1.929308 -1.3184998 -0.1288216 -0.059864260 -0.044926382
1951 1.279159  1.2761782 -0.0238495  0.096544469 -0.009415969
1952 0.893386  1.9850508 -0.1620557  0.045617836  0.008825718
```

Princomp vs prcomp

prcomp() name	princomp() name	Description
sdev	sdev	the standard deviations of the principal components
rotation	loadings	the matrix of variable loadings (columns are eigenvectors)
center	center	the variable means (means that were substracted)
scale	scale	the variable standard deviations (the scalings applied to each variable)
x	scores	The coordinates of the individuals (observations) on the principal components.

Another Method for Finding PCA

```
> dPCA2 = preProcess(d, method=c("center", "scale", "pca"), thresh = 0.9)

> dPCA2
Created from 16 samples and 5 variables

Pre-processing:
  - centered (5)
  - ignored (0)
  - principal component signal extraction (5)
  - scaled (5)

PCA needed 2 components to capture 90 percent of the variance
```

```
> dPCA2 = preProcess(d, method=c("center", "scale", "pca"),
+                      thresh = 0.9, pcaComp = 3)

> dPCA2
Created from 16 samples and 5 variables

Pre-processing:
  - centered (5)
  - ignored (0)
  - principal component signal extraction (5)
  - scaled (5)

PCA used 3 components as specified
```