Reducing data complexity

Table of contents

```
library(tidyverse)
library(ade4)
library(tidymodels)
library(latex2exp)
library(imager)
library(LearnPCA)
```

1 Principal component analysis

This document is a reproduction using the `tidyverse` and `tidymodels` of the `LearnPCA`[1] package vignettes and adding or deleting information.

1.1 Sources

- Statquest:
  - StatQuest: Principal Component Analysis (PCA), Step-by-Step:
    - https://youtu.be/FgakZw6K1QQ?feature=shared
  - Principal Components Analysis in R: Step-by-Step Example:
    - https://www.statology.org/principal-components-analysis-in-r/
  - PCA in tidyverse framework:
    - https://cmdlinetips.com/2022/12/pca-with-tidyverse/
  - Making sense of principal component analysis, eigenvectors & eigenvalues:
    - https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues/140579
  - Relationship between SVD and PCA. How to use SVD to perform PCA?:
    - https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca
  - PCA and UMAP with tidymodels and #TidyTuesday cocktail recipes:
    - https://juliasilge.com/blog/cocktail-recipes-umap/

[1] https://CRAN.R-project.org/package=LearnPCA

  - `LearnPCA`
    * https://CRAN.R-project.org/package=LearnPCA

1.2 Data for PCA

- Samples organized by rows
- Variables organized in columns, which are measure for each sample

1.3 Purpose of PCA

- Data reduction
  - Identify variables that are not informative
  - Collapsing correlating variables

1.4 What I get from PCA

- Indication of how many principal components (PC) are needed to describe the data
  - Summary in scree plot (In `broom` we use matrix `d`)
- Scores
  - In `broom` matrix `u`
  - Relationships between samples
- Loadings
  - In `broom` matrix `v`
  - Contributions of the different variables

1.5 Data for the tutorial

```
data(tintoodiel)
TO <- tintoodiel$tab |>
  as_tibble(rownames = "sites")
FeCu <- TO |>
  select(Fe2O3, Cu) |>
```

```
  slice(28:43)
FeCu
```

```
# A tibble: 16 x 2
   Fe2O3     Cu
   <dbl>  <dbl>
 1  9.5  594
 2  7.35 328
 3 15.0    1.67
 4 18.5    2.99
 5  7.72 402
 6 13.4  342
 7 12.4    1.05
 8 11.8  399
 9 20.6    3.67
10 19.4    2.25
11 25.6    2.02
12 32.4  787
13 12    238
14 11.0  249
15 16.6    1.28
16 12.7  238
```

1.5.1 Raw data visualization

```
FeCu |>
  ggplot(aes(x = Fe2O3, y = Cu)) +
  geom_point() +
  labs(x = TeX(r'($Fe_2O_3$ (percent))'),
       y = TeX(r'($Cu$ (ppm))'))
```

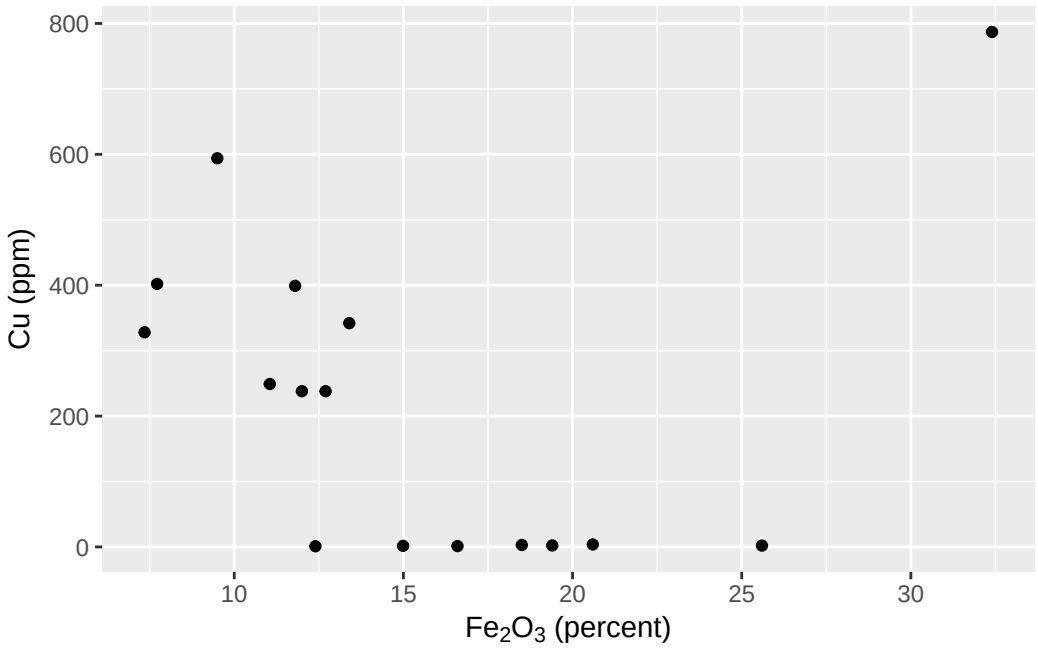


Figure 1: The relationship between the raw data values in `FeCu`

1.5.2 Centering the data

```
FeCu_centered <- FeCu |>
  mutate(across(Fe2O3:Cu, .fns = ~ scale(x = .x,
                                         center = TRUE,
                                         scale = FALSE)[,1]))
FeCu_centered
```

```
# A tibble: 16 x 2
    Fe2O3     Cu
    <dbl>  <dbl>
 1 -5.88   370.
 2 -8.03   104.
 3 -0.386 -223.
 4  3.12  -222.
 5 -7.66   178.
 6 -1.98   118.
 7 -2.98  -223.
 8 -3.58   175.
 9  5.22  -221.
```

```
10  4.02  -222.
11 10.2   -222.
12 17.0    563.
13 -3.38    13.5
14 -4.33    24.5
15  1.22  -223.
16 -2.68    13.5
```

1.5.3 Scaling the data

```
FeCu_centered_scaled <- FeCu_centered |>
  mutate(across(Fe2O3:Cu, .fns = ~ scale(x = .x,
                                         center = FALSE,
                                         scale = TRUE)[,1]))
FeCu_centered_scaled
```

```
# A tibble: 16 x 2
     Fe2O3      Cu
     <dbl>   <dbl>
 1 -0.879   1.52
 2 -1.20    0.426
 3 -0.0577 -0.917
 4  0.468  -0.911
 5 -1.15    0.730
 6 -0.296   0.483
 7 -0.445  -0.919
 8 -0.535   0.718
 9  0.782  -0.908
10  0.602  -0.914
11  1.53   -0.915
12  2.55    2.31
13 -0.505   0.0555
14 -0.647   0.101
15  0.183  -0.918
16 -0.400   0.0555
```

1.5.4 Data reduction

1.5.4.1 Using `prcomp`

```r
pca_FeCu <- FeCu_centered_scaled |>
  prcomp(center = FALSE, scale. = FALSE)
str(pca_FeCu)
```

```
List of 5
 $ sdev    : num [1:2] 1.007 0.992
 $ rotation: num [1:2, 1:2] 0.707 -0.707 -0.707 -0.707
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "Fe2O3" "Cu"
  .. ..$ : chr [1:2] "PC1" "PC2"
 $ center  : logi FALSE
 $ scale   : logi FALSE
 $ x       : num [1:16, 1:2] -1.697 -1.15 0.607 0.975 -1.327 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "PC1" "PC2"
 - attr(*, "class")= chr "prcomp"
```

If you compare Figure 1 to Figure 2, it looks broadly similar, but the points are rotated and the scales are different.

```r
scores_FeCu <- pca_FeCu |>
  tidy(matrix = "u") |>
  pivot_wider(id_cols = row,
              names_from = PC,
              values_from = value,
              names_glue = "pc_{PC}")

scores_FeCu |>
  ggplot(aes(x = pc_1, y = pc_2)) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2")
```

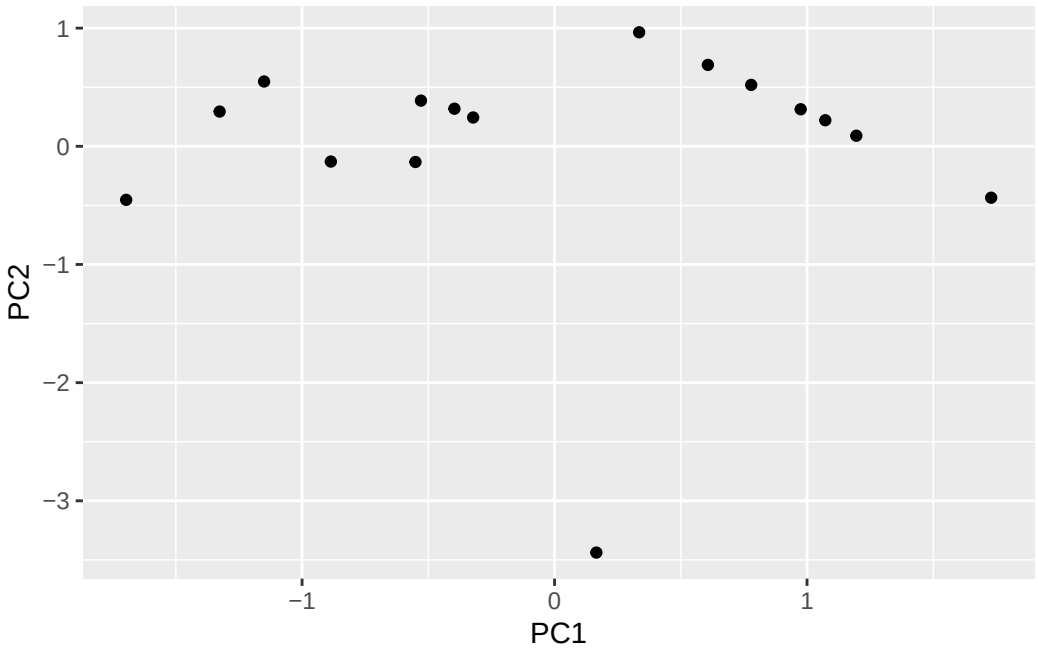


Figure 2: Scores

1.5.4.2 Using all the data

```
pca_TO <- prcomp(select(TO, -sites),
                 retx = TRUE,
                 center = TRUE, scale. = TRUE)
str(pca_TO)
```

```
List of 5
 $ sdev    : num [1:16] 2.1 1.77 1.35 1.17 1.06 ...
 $ rotation: num [1:16, 1:16] -0.4496 0.3152 -0.0889 0.2175 0.1645 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "SiO2" "Al2O3" "CaO" "MgO" ...
  .. ..$ : chr [1:16] "PC1" "PC2" "PC3" "PC4" ...
 $ center  : Named num [1:16] 53.57 12.1 1.7 1.45 2.21 ...
  ..- attr(*, "names")= chr [1:16] "SiO2" "Al2O3" "CaO" "MgO" ...
 $ scale   : Named num [1:16] 14.821 3.425 2.262 0.472 0.798 ...
  ..- attr(*, "names")= chr [1:16] "SiO2" "Al2O3" "CaO" "MgO" ...
 $ x       : num [1:52, 1:16] -2.75 1.122 -0.755 -0.336 2.647 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:16] "PC1" "PC2" "PC3" "PC4" ...
```

```
 - attr(*, "class")= chr "prcomp"
```

A similar plot of the raw data is not possible, because it is not two-dimensional: there are 16 dimensions corresponding to the 16 variables. Figure 3 shows the first two principal component scores:

```
scores_TO <- pca_TO |>
  tidy(matrix = "u") |>
  pivot_wider(id_cols = row,
              names_from = PC,
              values_from = value,
              names_glue = "pc_{PC}")

scores_TO |>
  ggplot(aes(x = pc_1, y = pc_2)) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2")
```

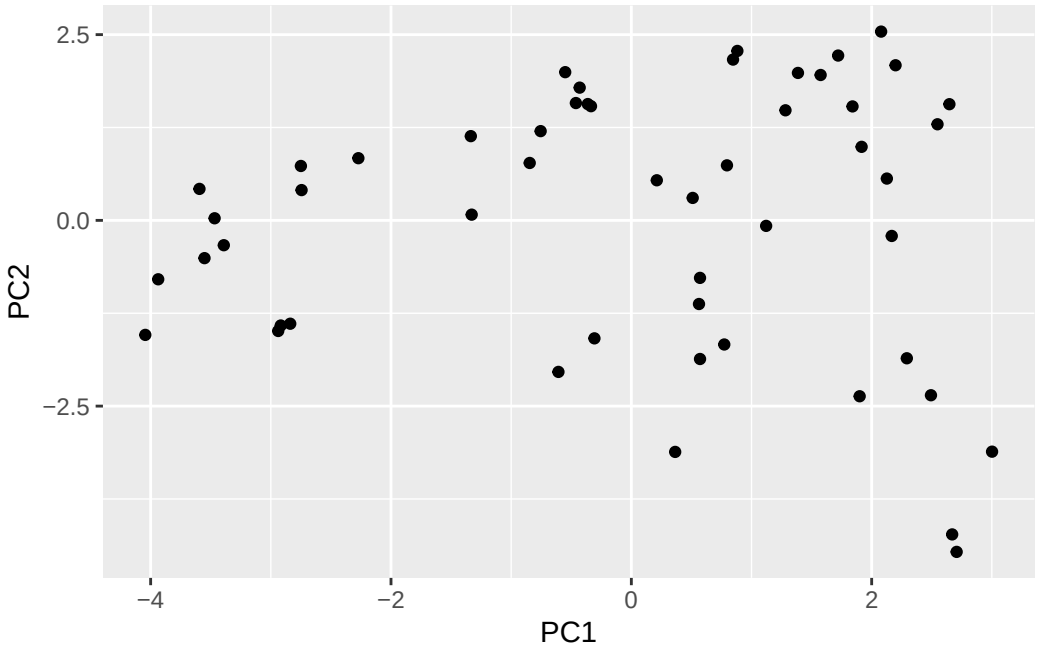


Figure 3: Score plot using all the data

1.5.4.3 What Else is in the PCA Results?

`prcomp`

- `pca_TO$sdev`: standard deviations of the principal components
- `pca_TO$rotation`: loadings
- `pca_TO$x`: scores
- `pca_TO$center`: values used for centering
- `pca_TO$scale`: values used for scaling

`broom::tidy`

- `tidy(pca_TO, matrix = "u")`: scores
- `tidy(pca_TO, matrix = "v")`: loadings
- `tidy(pca_TO, matrix = "d")`: standard deviations of the principal components

1.5.4.4 Scree plot

To obtain the eingevalues you need to extract the `pca_TO$sdev` values and square them to generate Figure 4.

```
eing_TO <- pca_TO |>
  tidy(matrix = "d")

eing_TO |>
  mutate(var = std.dev^2) |>
  ggplot(aes(x = PC, y = var)) +
  geom_col(color = "black") +
  scale_x_continuous(breaks = eing_TO$PC) +
  labs(x = "Principal component",
       y = "Variance")
```

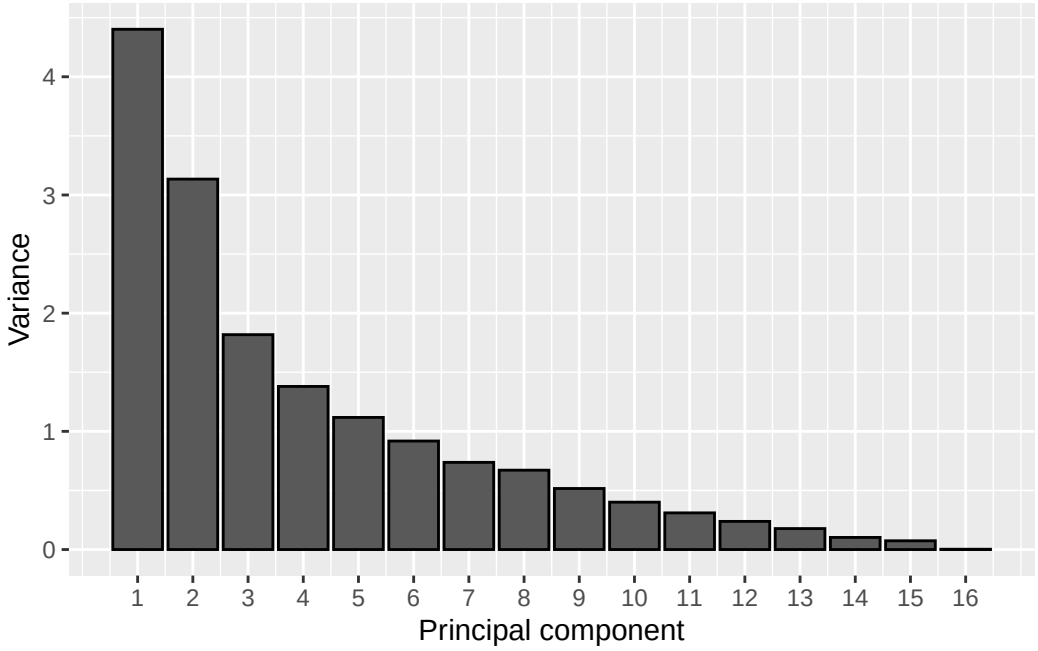


Figure 4: Scree plot

1.5.4.5 Loading plot

To generate the plot you choose the first principal component and arrange it as is shown in Figure 5.

```
loading_TO_pc_1 <- pca_TO |>
  tidy(matrix = "v") |>
  filter(PC == 1) |>
  arrange(desc(value)) |>
  mutate(column = fct_reorder(column, .x = value) |>
           fct_rev())

loading_TO_pc_1 |>
  ggplot(aes(x = column, y = value)) +
  geom_col(color = "black") +
  labs(x = NULL,
       y = NULL) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

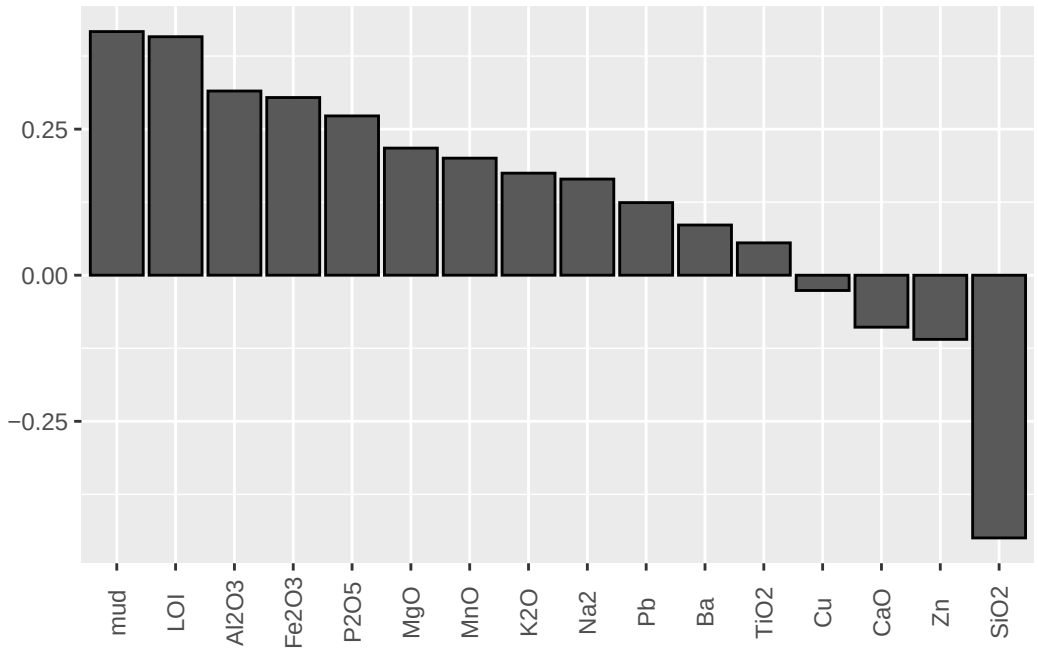


Figure 5: Plot of the loadings on PC1

1.6 Undoing the Scaling

1.6.1 Perfect reconstruction

```
scores <- pca_TO |>
  tidy(matrix = "u") |>
  pivot_wider(id_cols = row,
              names_from = PC,
              values_from = value,
              names_glue = "pc_{PC}") |>
  select(-row)

loading <- pca_TO |>
  tidy(matrix = "v") |>
  pivot_wider(id_cols = column,
              names_from = PC,
              values_from = value,
              names_glue = "pc_{PC}") |>
  select(-column)
```

```r
TOhat <- (as.matrix(scores) %*% t(as.matrix(loading))) |>
  scale(center = FALSE, scale = 1 / pca_TO$scale) |>
  scale(center = -pca_TO$center, scale = FALSE) |>
  as_tibble(.name_repair = "unique") |>
  set_names(colnames(TO)[-1]) |>
  add_column(sites = TO$sites, .before = "SiO2")

mean(as.matrix(select(TO, -sites)) - as.matrix(select(TOhat, -sites)))
```

```
[1] -2.120299e-15
```

1.7 Partial reconstruction

1.7.1 Example with 3 components

```r
ncomp <- 3
### Calculate Xhat
Xhat <- pca_TO$x[, 1:ncomp] %*% t(pca_TO$rotation[, 1:ncomp])
### Undoing the scaling
Xhat <- scale(x = Xhat, center = FALSE, scale = 1/pca_TO$scale)
### Undoing the centering
Xhat <- scale(x = Xhat, center = -pca_TO$center, scale = FALSE)
### Calculating the original data
X <- TO |>
  select(-sites) |>
  as.matrix()
## Compare original data vs reconstruction
error <- X - Xhat
dim(X)
```

```
[1] 52 16
```

```r
### Root Mean Squared Deviation (RMSD)
rmsd <- sqrt(sum(error^2) / length(error))
rmsd
```

```
[1] 109.6484
```

1.7.2 Visualizing the Root Mean Squared Deviation (RMSD) by adding components

```
rmsd_tbl <- 1:ncol(X) |>
  map(.f = ~ XtoPCAtoXhat(X, .x, sd)) |>
  map(.f = ~ .x - X) |>
  map(.f = ~ sqrt(sum(.x^2)/length(.x))) |>
  enframe(name = "pc", value = "rmsd") |>
  unnest(cols = rmsd)

rmsd_tbl |>
  ggplot(aes(x = pc, y = rmsd)) +
  geom_point(shape = 21, color = "black",
             fill = "#E31A1C", size = 3) +
  geom_line() +
  scale_x_continuous(breaks = 1:16) +
  scale_y_continuous(breaks = seq.int(from = 0,
                                      to = max(rmsd_tbl$rmsd) |> ceiling(),
                                      by = 20)) +
  labs(x = "Principal Component",
       y = "Root Mean Squared Deviation (RMSD)")
```

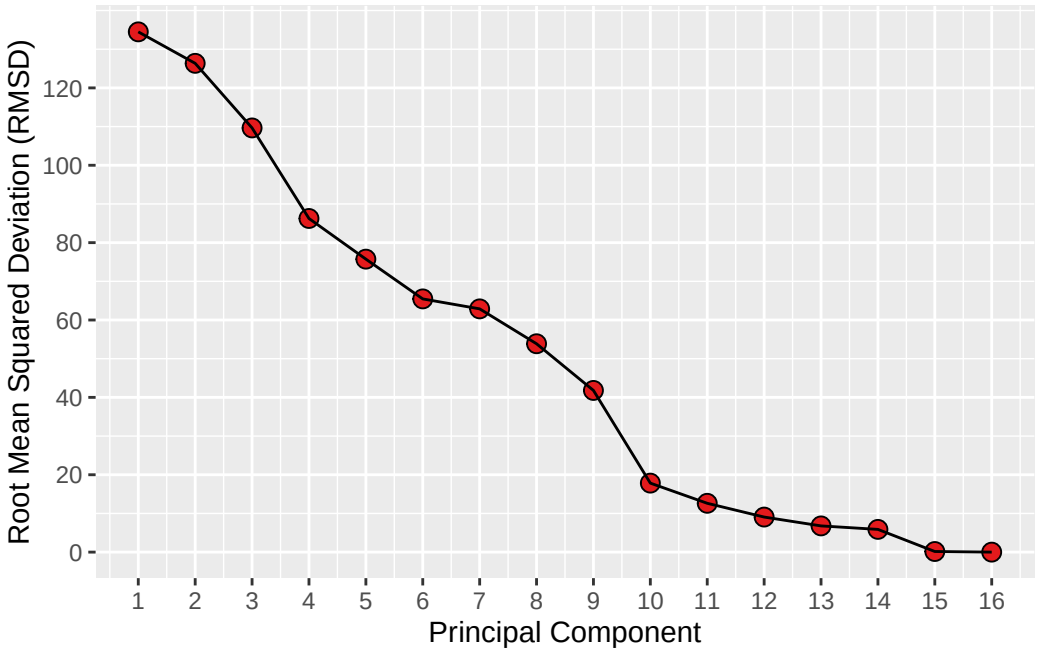


Figure 6: Reduction of error as the number of components included in the reconstruction increases

1.8 Image reconstruction

- Reconstructing Images Using PCA
  - https://www.r-bloggers.com/2019/10/reconstructing-images-using-pca/
  - https://kieranhealy.org/blog/archives/2019/10/27/reconstructing-images-using-pca/

1.8.1 Sources for images

- The USC-SIPI Image Database:
  - `https://sipi.usc.edu/database/ > Miscellaneous`

1.8.2 Import image

```
boat_gray <- load.image(file = "images/008_boat_gray_512_x_512.tiff")
dim(boat_gray)
```

```
[1] 512 512   1   1
```

```
str(boat_gray)
```

```
 'cimg' num [1:512, 1:512, 1, 1] 0.498 0.482 0.49 0.471 0.494 ...
```

```
class(boat_gray)
```

```
[1] "cimg"         "imager_array" "numeric"
```

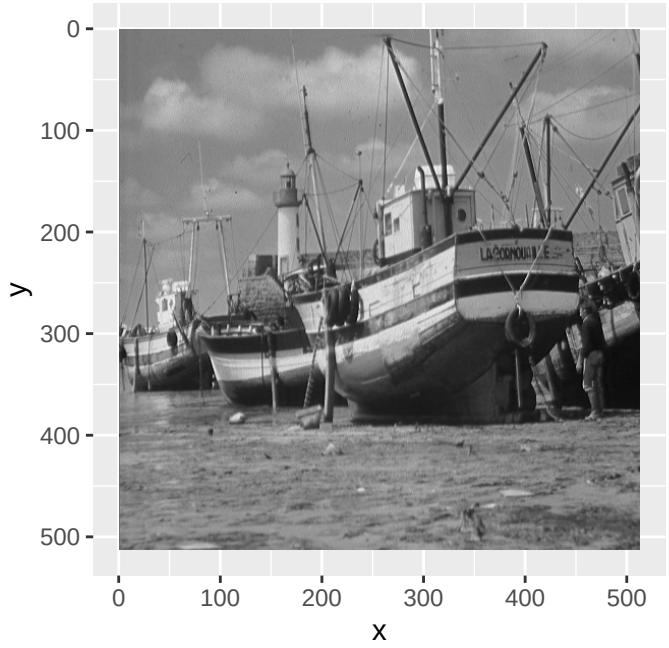
1.8.3 Express image as data

```
boat_gray_long <- boat_gray |>
  as.data.frame() |>
  as_tibble()
boat_gray_long
```

```
# A tibble: 262,144 x 3
       x     y value
   <int> <int> <dbl>
 1     1     1 0.498
 2     2     1 0.482
 3     3     1 0.490
 4     4     1 0.471
 5     5     1 0.494
 6     6     1 0.482
 7     7     1 0.498
 8     8     1 0.502
 9     9     1 0.490
10    10     1 0.506
# i 262,134 more rows
```

1.8.4 Visualize image

```
boat_gray_long |>
  ggplot(aes(x = x, y = y)) +
  geom_raster(aes(fill = value)) +
  # We need this part because imager uses
  # the following coordinate system
  ## Top-left origin (0, 0)
  ## Positive x-coordinates increase rightwards
  ## Positive y-coordinates increase downwards
  # However ggplot used the following coordinate
  # system
  ## Bottom-left origin (0, 0)
  ## Positive x-coordinates increase rightwards
  ## Positive y-coordinates increase upwards
  scale_y_reverse() +
  scale_fill_gradient(low = "black", high = "white") +
  theme(legend.position = "none",
        aspect.ratio = 1)
```



1.8.5 Reconstruction using components

1.8.5.1 Prepare data

```
boat_gray_wider <- boat_gray_long |>
  pivot_wider(id_cols = x,
              names_from = y,
              values_from = value)
boat_gray_wider
```

```
# A tibble: 512 x 513
      x   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`
  <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     1 0.498 0.502 0.502 0.486 0.494 0.490 0.498 0.482 0.494 0.486 0.478 0.494
2     2 0.482 0.494 0.486 0.498 0.490 0.498 0.498 0.529 0.502 0.502 0.494 0.498
3     3 0.490 0.502 0.502 0.502 0.502 0.494 0.494 0.471 0.486 0.498 0.502 0.490
4     4 0.471 0.478 0.494 0.506 0.494 0.494 0.486 0.502 0.502 0.486 0.494 0.478
5     5 0.494 0.490 0.498 0.475 0.494 0.502 0.471 0.475 0.490 0.498 0.482 0.490
6     6 0.482 0.490 0.471 0.502 0.490 0.502 0.498 0.482 0.482 0.475 0.498 0.475
7     7 0.498 0.478 0.502 0.506 0.498 0.502 0.502 0.494 0.502 0.502 0.486 0.498
8     8 0.502 0.506 0.506 0.502 0.502 0.494 0.494 0.494 0.510 0.510 0.506 0.502
9     9 0.490 0.498 0.502 0.506 0.514 0.510 0.502 0.502 0.502 0.518 0.514 0.514
```

```
10    10 0.506 0.502 0.514 0.522 0.498 0.506 0.514 0.522 0.518 0.522 0.525 0.514
# i 502 more rows
# i 500 more variables: `13` <dbl>, `14` <dbl>, `15` <dbl>, `16` <dbl>,
#   `17` <dbl>, `18` <dbl>, `19` <dbl>, `20` <dbl>, `21` <dbl>, `22` <dbl>,
#   `23` <dbl>, `24` <dbl>, `25` <dbl>, `26` <dbl>, `27` <dbl>, `28` <dbl>,
#   `29` <dbl>, `30` <dbl>, `31` <dbl>, `32` <dbl>, `33` <dbl>, `34` <dbl>,
#   `35` <dbl>, `36` <dbl>, `37` <dbl>, `38` <dbl>, `39` <dbl>, `40` <dbl>,
#   `41` <dbl>, `42` <dbl>, `43` <dbl>, `44` <dbl>, `45` <dbl>, `46` <dbl>, ...
```

1.8.5.2 Principal component analysis

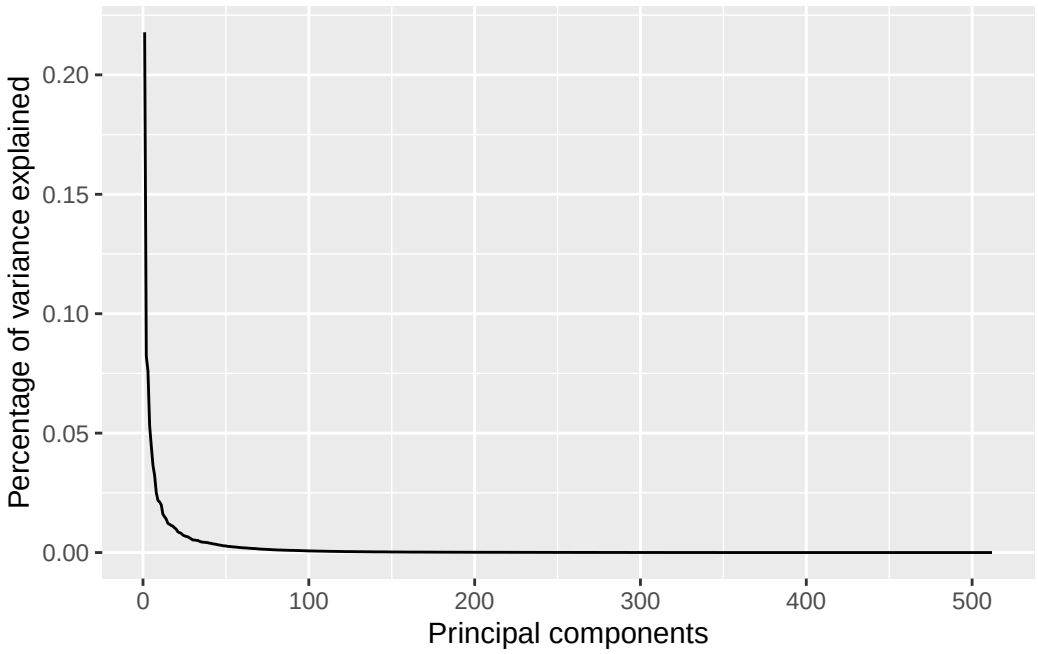
```
boat_gray_wider_pca <- boat_gray_wider |>
  select(-x) |>
  prcomp(retx = TRUE, center = TRUE, scale. = TRUE)
```

1.8.5.3 Variance explained

```
boat_gray_pca_variance <- boat_gray_wider_pca |>
  tidy(matrix = "d")
boat_gray_pca_variance
```

```
# A tibble: 512 x 4
      PC std.dev percent cumulative
   <dbl>   <dbl>   <dbl>      <dbl>
 1     1   10.6   0.218       0.218
 2     2    6.49  0.0823      0.300
 3     3    6.24  0.0760      0.376
 4     4    5.21  0.0531      0.429
 5     5    4.78  0.0446      0.474
 6     6    4.33  0.0366      0.510
 7     7    4.07  0.0323      0.543
 8     8    3.59  0.0252      0.568
 9     9    3.34  0.0218      0.590
10    10    3.29  0.0212      0.611
# i 502 more rows
```

```
boat_gray_pca_variance |>
  ggplot() +
  geom_line(aes(x = PC, y = percent)) +
  labs(x = "Principal components",
       y = "Percentage of variance explained")
```



1.8.5.4 Result of reconstruction using different number of components

```
n_pc <- c(1, 5, 10, 50, 100, 500)

pca_tidy_reconstruction <- tibble(pc = n_pc) |>
  mutate(x_hat = map(.x = n_pc,
                     .f = ~ boat_gray_wider_pca$x[, 1:.x] %*%
                           t(boat_gray_wider_pca$rotation[, 1:.x])),
         x_hat = map(.x = x_hat,
                     .f = ~ scale(x = .x,
                                  center = FALSE,
                                  scale = 1/boat_gray_wider_pca$scale)),
         x_hat = map(.x = x_hat,
                     .f = ~ scale(x = .x,
                                  center = -boat_gray_wider_pca$center,
                                  scale = FALSE)),
         x_hat = map(.x = x_hat,
                     .f = ~ as_tibble(.x)),
         x_hat = map(.x = x_hat,
                     .f = ~ bind_cols(select(boat_gray_wider, x), .x)),
         x_hat = map(.x = x_hat,
                     .f = ~ pivot_longer(.x,
                                         cols = -x,
                                         names_to = "y",
                                         values_to = "value",
                                         names_transform = list(y = as.integer)))) |>
```

```
  mutate(pc_label = str_glue("Number of components: {pc}"),
         .after = pc) |>
  mutate(pc_label = fct_reorder(.f = pc_label, .x = pc)) |>
  unnest(x_hat)

pca_tidy_reconstruction |>
  ggplot(aes(x = x, y = y)) +
  geom_raster(aes(fill = value)) +
  scale_y_reverse() +
  scale_fill_gradient(low = "black", high = "white") +
  facet_wrap(facets = vars(pc),
             nrow = 2, ncol = 3) +
  theme(legend.position = "none",
        aspect.ratio = 1)
```