LAB: Principal Component Analysis

2025-03-18

```
# We will use the following packages.
# If needed, install them : pak::pkg_install().
stopifnot(
  require("corrr"),
  require("magrittr"),
  require("lobstr"),
  require("ggforce"),
  require("gt"),
  require("glue"),
  require("skimr"),
  require("patchwork"),
  require("tidyverse"),
  require("ggfortify")
  # require("autoplotly")
)
```

```
old_theme <- theme_set(theme_minimal())

options(ggplot2.discrete.colour="viridis")
options(ggplot2.discrete.fill="viridis")
options(ggplot2.continuous.fill="viridis")
options(ggplot2.continuous.colour="viridis")
```
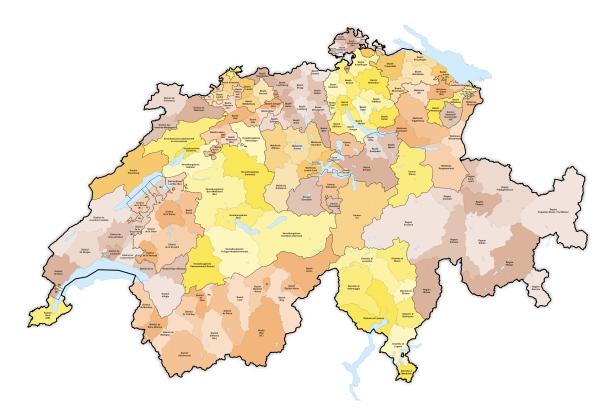

Swiss fertility data

Dataset `swiss` from `datasets::swiss` connect fertility and social, economic data within 47 French-speaking districts in Switzerland.

- `Fertility` : fertility index
- `Agriculture` : jobs in agricultural sector
- `Examination` : literacy index (military examination)
- `Education` : proportion of people with successful secondary education
- `Catholic` : proportion of Catholics
- `Infant.Mortality` : mortality quotient at age 0

Fertility index (`Fertility`) is considered as the *response variable*

1

The social and economic variables are *covariates* (*explanatory* variables).

See European Fertility Project for more on this dataset.

PCA (Principal Component Analysis) is concerned with covariates.

```
data("swiss")

swiss %>%
  glimpse(50)
```

```
Rows: 47
Columns: 6
$ Fertility        <dbl> 80.2, 83.1, 92.5, 85.8,~
$ Agriculture      <dbl> 17.0, 45.1, 39.7, 36.5,~
$ Examination      <int> 15, 6, 5, 12, 17, 9, 16~
$ Education        <int> 12, 9, 5, 7, 15, 7, 7, ~
$ Catholic         <dbl> 9.96, 84.84, 93.40, 33.~
$ Infant.Mortality <dbl> 22.2, 22.2, 20.2, 20.3,~
```

Have a look at the documentation of the dataset

Describe the dataset

> **ℹ Question**
>
> Compute summary for each variable

> 💡 **solution**
>
> It is enough to call `summary()` on each column of `swiss`. This can be done in a functional programming style using package `purrr`. The collections of summaries can be rearranged so as to build a dataframe that is fit for reporting.
>
> ```
> tt <- map_dfr(swiss, summary, .id = "var")
> ```
>
> ```
> tt %>%
>   knitr::kable()
> ```
>
> | var | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
> |---|---|---|---|---|---|---|
> | Fertility | 35.00 | 64.700 | 70.40 | 70.14255 | 78.450 | 92.5 |
> | Agriculture | 1.20 | 35.900 | 54.10 | 50.65957 | 67.650 | 89.7 |
> | Examination | 3.00 | 12.000 | 16.00 | 16.48936 | 22.000 | 37.0 |
> | Education | 1.00 | 6.000 | 8.00 | 10.97872 | 12.000 | 53.0 |
> | Catholic | 2.15 | 5.195 | 15.14 | 41.14383 | 93.125 | 100.0 |
> | Infant.Mortality | 10.80 | 18.150 | 20.00 | 19.94255 | 21.700 | 26.6 |
>
> Function `skim` from `skimr` delivers all univariate summaries in suitable form.
>
> ```
> foo <- swiss %>%
>   select(-Fertility) %>%
>   skim()
> ```
>
> ```
> foobar <- foo %>%
>   filter(skim_type=="numeric") %>%
>   rename(variable=skim_variable)  %>%
>     mutate(across(where(is.numeric), ~ round(.x, digits=1)))
> ```
>
> ```
> foobar %>%
>   knitr::kable()
> ```
>
> | skim_type | variable | n_missing | complete_rate | numeric.mean | numeric.sd | numeric.p0 | numeric.p25 | numeric.p50 | numeric.p75 | numeric.p100 | numeric.hist |
> |---|---|---|---|---|---|---|---|---|---|---|---|
> | numeric | Agriculture | 0 | 1 | 50.7 | 22.7 | 1.2 | 35.9 | 54.1 | 67.7 | 89.7 | |
> | numeric | Examination | 0 | 1 | 16.5 | 8.0 | 3.0 | 12.0 | 16.0 | 22.0 | 37.0 | |
> | numeric | Education | 0 | 1 | 11.0 | 9.6 | 1.0 | 6.0 | 8.0 | 12.0 | 53.0 | |
> | numeric | Catholic | 0 | 1 | 41.1 | 41.7 | 2.1 | 5.2 | 15.1 | 93.1 | 100.0 | |
> | numeric | Infant.Mortality | 0 | 1 | 19.9 | 2.9 | 10.8 | 18.1 | 20.0 | 21.7 | 26.6 | |

> ℹ **Question**
>
> Display graphic summary for each variable.

> 💡 **solution**
>
> We have to pick some graphical summary of the data. Boxplots and violine plots could be used if we look for concision.
> We use histograms to get more details about each column.
> Not that covariates have different meanings: `Agriculture`, `Catholic`, `Examination`, and `Education` are percentages with values between 0 and 100.
> We have no details about the standardized fertility index `Fertility`
> `Infant.Mortality` is also a rate:
>> Infant mortality is the death of an infant before his or her first birthday. The infant mortality rate is the number of infant deaths for every 1,000 live births. In addition to giving us key information about maternal and infant health, the infant mortality rate is an important marker of the overall health of a society.
>
> see Center for Desease Control
> We reuse the function we have already developped during previous sessions.
>
> ```
> make_biotifoul(swiss, .f = is.numeric)
> ```
>
>
>
> Histograms reveal that our covariates have very different distributions.
> Religious affiliation (`Catholic`) tells us that there two types of districts, which is reminiscent of the old principle *Cujus regio, ejus religio*, see Old Swiss Confederacy.
> `Agriculture` shows that in most districts, agriculture was still a very important activity.
> `Education` reveals that in all but a few districts, most children did not receive secondary education. `Examination` shows that some districts lag behind the bulk of districts. Even less exhibit a superior performance.
> The two demographic variables `Fertility` and `Infant.Mortality` look roughly unimodal with a few extreme districts.

Investigate pairwise correlations

> ℹ️ **Question**
>
> - Compute, display and comment the sample correlation matrix
> - Display jointplots for each pair of variables

> 💡 **solution**
>
> Package `corrr`, functions `correlate` and `rplot` provide a convenient tool.
> Note that `corrr::rplot()` creates a graphical object of class `ggplot`. We can endow
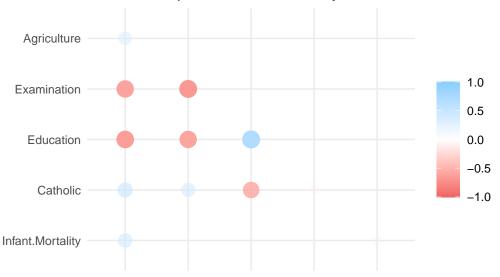> it with more layers.
>
> ```
> swiss |>
>     corrr::correlate(use="pairwise.complete.obs",method="pearson", quiet=T) |>
>   corrr::shave() |>
>   corrr::rplot() +
>   labs(title="Correlation plot for Swiss Fertility data") +
>   theme_minimal()
> ```
>
> ## Correlation plot for Swiss Fertility data
>
>
>
> The high positive linear correlation between `Education` and `Examination` is mod-
> erately surprising. The negative correlation between the proportion of people in-
> volved in `Agriculture` and `Education` and `Examination`is also not too surprising.
> Secondary schooling required pupils from rural areas to move to cities.
> A more intriguing observation concerns the pairs `Catholic` and `Examination` (neg-
> ative correlation) and `Catholic` and `Education` (little correlation).
> The response variable `Fertility` looks negatively correlated with `Examination` an
> `Education`. These correlations are worth being further explored. In Demography,
> the decline of Fertility is often associated with the the rise of women education.
> Note that `Examination` is about males, and that `Education` does not give details
> about the way women complete primary education.

Singular Value Decomposition (SVD)

> ℹ️ **Question**
>
> - Project the `swiss` dataset on the covariates (all columns but `Fertility`)
> - Center the projected data using matrix manipulation
> - Center the projected data using `dplyr` verbs
> - Compare the results with the output of `scale()` with various optional argu-
>   ments
> - Call the centered matrix `Y`

> 💡 **solution**
>
> Hand-made centering of the dataframe emphasises the fact that centering is a linear
> operation. As a matter of fact, it consists in projecting the data frame on the linear
> space orthogonal to the constant vector.
>
> ```r
> X <- select(swiss, -Fertility) |>
>     as.matrix()
>
> n <- nrow(X)
> ones <-  matrix(1, nrow = n, ncol=1)
>
> Y <-  X - (1/n)* (ones  %*%  t(ones) %*% X)
> ```
>
> We can also perform centering using `dplyr` verbs. This can be viewed as computing
> a window function over a trivial partition.
>
> ```r
> swiss |>
>   select(-Fertility) |>
>   mutate(across(everything(), \(x) x-mean(x)))
> ```

> 💡 Anyway, function `scale(X, scale=F)` from base `R` does the job.

> ℹ️ **Question**
>
> Check that the ouput of `svd(Y)` actually defines a Singular Value Decomposition.

> 💡 **solution**
>
> `svd(Y)` is a list with 3 elements (`u`,`d`,`v`).
>
> $$Y = U \times D \times V^\top$$
>
> ```r
> svd_Y <-  svd(Y)
>
> svd_Y %$%                                                        ①
>   (Y - u %*% diag(d) %*% t(v)) %>%
>   norm(type = "F")                                               ②
>
> norm(
>   diag(1, ncol(Y)) -
>   (svd_Y %$% (t(v) %*% v)),
>   'F'
> )                                                                ③
> ```
>
> ① Exposing pipe from `magrittr`
> ② Checking the factorization
> ```
> [1] 1.04354e-13
> [1] 1.137847e-15
> ```

> 💡 Note that we used the exposing pipe `%$%` from `magrittr` to unpack `svd_Y` which is
> a list with class `svd` and members named `u`, `d` and `v`.
> We could have used `with(,)` from base `R`.

> **ℹ Question**
>
> Relate the SVD of Y and the eigen decomposition of $Y^\top \times Y$

> **💡 solution**
>
> The matrix $1/n Y^\top \times Y$ is the covariance matrix of the covariates.
> The spectral decomposition of the symmetric Semi Definite Positive (SDP) matrix $1/n Y^\top \times Y$ is related with the SVD factorization of Y.
> The spectral/eigen decomposition of $Y^\top \times Y$ can be obtained using `eigen()`.
> The eigenspaces of $Y^\top \times Y$ are the right eigenspaces of Y.
>
> ```
> (t(eigen(t(Y) %*% Y )$vectors) %*% svd_Y$v ) %>%
>   round(digits=2)
> ```
>
> ```
>      [,1] [,2] [,3] [,4] [,5]
> [1,]    1    0    0    0    0
> [2,]    0   -1    0    0    0
> [3,]    0    0    1    0    0
> [4,]    0    0    0    1    0
> [5,]    0    0    0    0    1
> ```
> The eigenvalues of $Y^\top \times Y$ are the squared singular values of Y
>
> ```
> eigen(t(Y) %*% Y )$values - (svd_Y$d)^2
> ```
>
> ```
> [1] -1.309672e-10 -1.455192e-11  8.640200e-12  8.526513e-12 -3.410605e-13
> ```

> **🔥** Here, the eigenvectors of $Y^\top \times Y$ coincide with the right singular vectors of Y corresponding to non-zero singular values. Up to sign changes, it is always true when the non-zero singular values are pairwise distinct.

Perform PCA on covariates

> **ℹ Question**
>
> Pairwise analysis did not provide us with a clear and simple picture of the French-speaking districts.
> PCA (Principal Component Analysis) aims at exploring the variations of multivariate datasets around their mean (center of inertia). In the sequel, we will perform PCA on the matrix of centered covariates, with and without standardizing the centered columns.
> Base R offers `prcomp()`. Call `prcomp()` on the centered covariates
> Note that R also offers `princomp`

We first call `prcomp()` with the default arguments for centering and scaling, that is, we center columns and do not attempt to standardize columns. Name the output `pco`.

What is the result made of?

> **solution**
>
> ```r
> pco <- swiss |>
>   select(-Fertility) |>
>   scale(scale = F) |>
>   prcomp(scale. = F)
> ```
>
> `pco` is a list with 5 members. It as a class attribute `prcomp`. It is an object of class `prcomp` (function `prcomp()` acts as a constructor for class `pco` just as `lm()` acts as a constructor for class `lm`). Class pco is an S3 class
>
> ```r
> rlang::is_list(pco)
> ```
>
> ```
> [1] TRUE
> ```
>
> ```r
> attributes(pco)
> ```
>
> ```
> $names
> [1] "sdev"     "rotation" "center"   "scale"     "x"
>
> $class
> [1] "prcomp"
> ```
>
> ```r
> sloop::s3_class(pco)
> ```
>
> ```
> [1] "prcomp"
> ```

> **Question**
>
> Check that `prcomp()` is indeed a wrapper for `svd()`.

💡 **solution**

We first check that the matrix can be recovered from the product of the components
of the `prcomp` object.

```
(Y - pco$x %*% t(pco$rotation )) %>%
  round(digits = 2)  %>%
  head()
```

	Agriculture	Examination	Education	Catholic	Infant.Mortality
Courtelary	0	0	0	0	0
Delemont	0	0	0	0	0
Franches-Mnt	0	0	0	0	0
Moutier	0	0	0	0	0
Neuveville	0	0	0	0	0
Porrentruy	0	0	0	0	0

We now check that the `rotation` component is indeed made of the right singular
vectors (the V factor)

```
(svd_Y$v %*% t(pco$rotation )) %>%
  round(2)
```

	Agriculture	Examination	Education	Catholic	Infant.Mortality
[1,]	1	0	0	0	0
[2,]	0	1	0	0	0
[3,]	0	0	1	0	0
[4,]	0	0	0	1	0
[5,]	0	0	0	0	1

The column vectors of component x are pairwise orthogonal.

```
(t(pco$x) %*% pco$x) %>%
  round(2)
```

	PC1	PC2	PC3	PC4	PC5
PC1	86484.49	0.00	0.00	0.00	0.00
PC2	0.00	21127.44	0.00	0.00	0.00
PC3	0.00	0.00	2706.14	0.00	0.00
PC4	0.00	0.00	0.00	639.22	0.00
PC5	0.00	0.00	0.00	0.00	348.01

The `x` component of the `prcomp` object is the product of the U and D factors.

```
norm(pco$x - svd_Y$u %*% diag(svd_Y$d),type="F")
```

```
[1] 1.962527e-13
```

```
norm(as.matrix(pco$rotation) -svd_Y$v, type="F")
```

```
[1] 2.04934e-15
```

The connection between pcosdev$ and $svd_Y d$ is somewhat less transparent.

```
sum(abs(apply(pco$x, 2, sd) - pco$sdev))
```

```
[1] 1.865175e-14
```

The components of `pco$sdev` are the standard deviations of the columns of $U \times D$.

```
pco$sdev - svd_Y$d/sqrt(nrow(Y)-1)
```

```
[1] -7.105427e-15  0.000000e+00  8.881784e-16 -1.332268e-15 -1.332268e-15
```

> **ℹ Question**
>
> Check that rows and columns of component `rotation` of the result of `prcomp()` have unit norm.

> **💡 solution**
>
> Check that rows and columns of matrix `rotation` have unit norm.
>
> ```r
> apply(pco$rotation, 2, \(x) norm(x, "2"))
> ```
>
> ```
> PC1 PC2 PC3 PC4 PC5
>   1   1   1   1   1
> ```
>
> ```r
> apply(pco$rotation, 1, \(x) norm(x, "2"))
> ```
>
> ```
>      Agriculture       Examination         Education          Catholic
>                1                 1                 1                 1
> Infant.Mortality
>                1
> ```

> **ℹ Question**
>
> Check Orthogonality of V (component `rotation` of the `prcomp` object)

> **💡 solution**
>
> ```r
> # checking that pco$rotation is an orthogonal matrix
> t(pco$rotation) %*% pco$rotation
> ```
>
> ```
>               PC1           PC2           PC3           PC4           PC5
> PC1  1.000000e+00 -1.003429e-16  8.239937e-18 -1.097213e-16  6.938894e-18
> PC2 -1.003429e-16  1.000000e+00  1.181780e-16  5.074066e-17  4.857226e-17
> PC3  8.239937e-18  1.181780e-16  1.000000e+00  1.717376e-16 -6.938894e-17
> PC4 -1.097213e-16  5.074066e-17  1.717376e-16  1.000000e+00 -1.804112e-16
> PC5  6.938894e-18  4.857226e-17 -6.938894e-17 -1.804112e-16  1.000000e+00
> ```
>
> ```r
> pco$rotation %*% t(pco$rotation)
> ```
>
> ```
>                   Agriculture   Examination     Education      Catholic
> Agriculture      1.000000e+00  3.642919e-17 -1.153591e-16  1.689187e-16
> Examination      3.642919e-17  1.000000e+00 -8.630249e-17  2.244298e-17
> Education       -1.153591e-16 -8.630249e-17  1.000000e+00 -1.127570e-16
> Catholic         1.689187e-16  2.244298e-17 -1.127570e-16  1.000000e+00
> Infant.Mortality 2.081668e-17 -1.734723e-16  8.326673e-17 -2.081668e-17
>                  Infant.Mortality
> Agriculture          2.081668e-17
> Examination         -1.734723e-16
> Education            8.326673e-17
> Catholic            -2.081668e-17
> Infant.Mortality     1.000000e+00
> ```

> **ℹ Question**
>
> Make a scatterplot from the first two columns of the x component of the `prcomp` object.

💡 **solution**

Objects of class `prcomp` can be handled by generic functions like `plot()` or better
`autoplot()`. Namely, method `prcomp` for generic S3 function `autoplot()` from
`ggplot2` delivers one of classical SVD plots.

```
res <- autoplot(pco) +
  coord_fixed() +
  theme_minimal()

ts <- theme_set(theme_minimal())

res
```


`autoplot(pco)` is a scatterplot for the dataframe defined by matrix $U \times D$ projected
on its first two principal components (first two columns).

As `autoplot(pco)` is an instance of class `ggplot`, it can be annotated, decorated as
any other `ggplot` object.

```
(
  res + aes(color=Catholic) + theme_minimal()
) +
(
  res + aes(color=Education) + theme_minimal()
) +
  patchwork::plot_annotation(
    subtitle = "Scatterplot on the first two principal components (no column scaling)'
    title= "Share of catholics can almost be determined from the sign of the first PC'
    caption = "Swiss  Fertility data from R datasets"
  )
```

Share of catholics can almost be determined from the fir

Scatterplot on the first two principal components (no column scaling)

ℹ **Question**

Define a graphical pipeline for the *screeplot*.
Hint: use function `tidy()` from `broom`, to get the data in the right form from an instance of `prcomp`.

💡 **solution**

The screeplot is a bar plot where each bar corresponds to a singular value. The bar height is proportional to the square of the corresponding singular value.
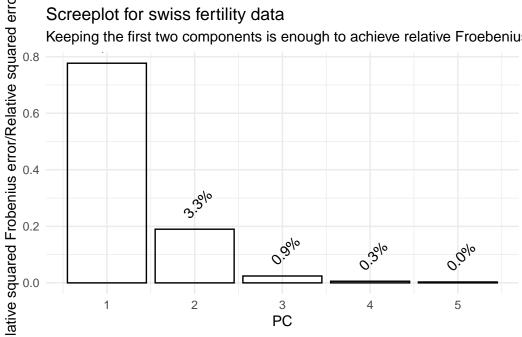
```
p_screeplot <- . %>%                                                    ①
  broom::tidy(matrix="pcs") %>% {
  ggplot(.) +
  aes(x=PC, y=percent, label=pct_format(1.-cumulative)) +              ③
  geom_text(angle=45, vjust=-1, hjust=-.1) +
  geom_col(fill=NA, colour="black") +                                  ②
  theme_minimal()
  }
```

① Define a pipeline for building a screeplot
② Mind the braces on the right side of the first pipe
③ `1- cumulative` tell the reader about the relative Frobenious error achieved by keeping the first components of the SVD expansion.

```
pco %>%
  p_screeplot() +
  ylab('Relative squared Frobenius error/Relative squared error') +
  labs(
    title="Screeplot for swiss fertility data",
    subtitle="Keeping the first two components is enough to achieve relative Froebeniu
  theme_minimal()
```


Screeplot for swiss fertility data
Keeping the first two components is enough to achieve relative Froebenius
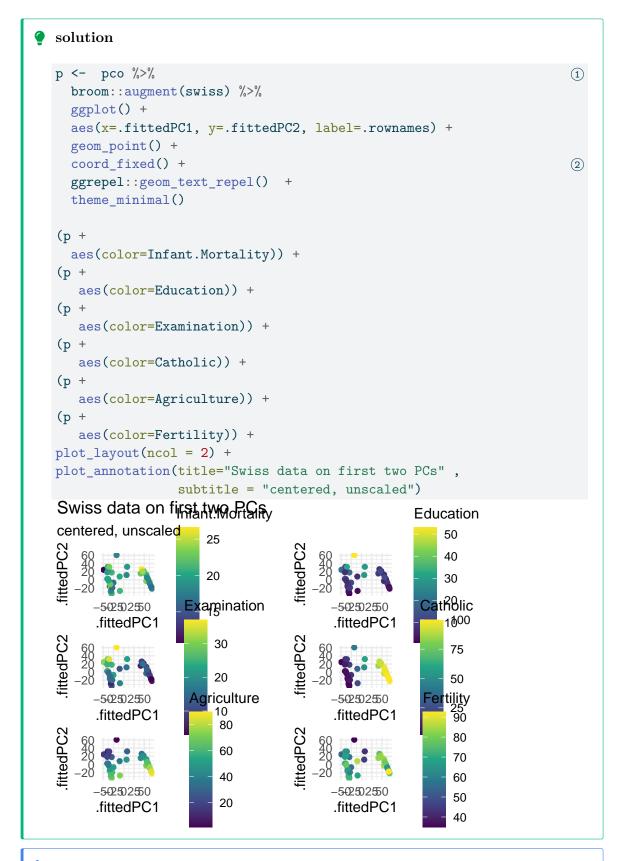
The screeplot is a visualization of the Eckart-Young-Mirsky Theorem. It tells us about the relative errors incurent when approximating the data matrix (with centered columns) by the low rank approximations defined by the truncated SVDs.

> **ℹ Question**
>
> Define a function that replicates `autoplot.prcomp()`
> Project the dataset on the first two principal components (perform dimension reduction) and build a scatterplot. Colour the points according to the value of original covariates.
> Hint: use generic function `augment` from `broom`.

💡 **solution**

```
p <-  pco %>%                                                    ①
  broom::augment(swiss) %>%
  ggplot() +
  aes(x=.fittedPC1, y=.fittedPC2, label=.rownames) +
  geom_point() +
  coord_fixed() +                                               ②
  ggrepel::geom_text_repel()  +
  theme_minimal()

(p +
  aes(color=Infant.Mortality)) +
(p +
  aes(color=Education)) +
(p +
  aes(color=Examination)) +
(p +
  aes(color=Catholic)) +
(p +
  aes(color=Agriculture)) +
(p +
  aes(color=Fertility)) +
plot_layout(ncol = 2) +
plot_annotation(title="Swiss data on first two PCs" ,
                subtitle = "centered, unscaled")
```


Swiss data on first two PCs
centered, unscaled

ℹ **Question**

Apply `broom::tidy()` with optional argument `matrix="v"` or `matrix="loadings"`
to the `prcomp` object.
Comment.

i

> **◉ solution**
>
> We can extract factor V from the SVD factorization using generic function `tidy` from package `broom`
>
> ```
> pco %>%
>   broom::tidy(matrix="v") %>%
>   sample_n(5) |>
>   gt::gt()
> ```
>
> | column | PC | value |
> |---|---|---|
> | Infant.Mortality | 5 | -0.99110977 |
> | Education | 3 | -0.80696178 |
> | Education | 1 | -0.05841770 |
> | Agriculture | 4 | -0.02652821 |
> | Education | 2 | 0.31078077 |
>
> The result is a tibble in *long form*. It is worth pivoting the dataframe into wide form. This gives back the rotation matrix.
>
> ```
> om <- pco %>%
>   broom::tidy(matrix="v") %>%
>   tidyr::pivot_wider(id_cols =column,
>             names_from = PC,
>             values_from = value) |>
>   select(-1) |>
>   as.matrix()
>
> norm((om %*% t(om))-diag(1,5), "F")
> ```
>
> ```
> [1] 8.196585e-16
> ```

i Question

> Build the third SVD plot, the so called *correlation circle*.

💡 **solution**

The correlation circle is built from the *loadings*, that is, from the `rotation` component of the `prcomp` object.

We define a preprocessing function to transform the rotation object into a proper `tibble` form.

```r
prep_co_circle <- function(pco) {
  r <- pco$rotation
  as_tibble(r) |>
    rename_with(.fn = \(x) gsub('PC', '', x), .cols=everything()) |>
    mutate(row_id=rownames(r))
}
```

The The next virtual graphical object will be our key tool to build the correlation circle.
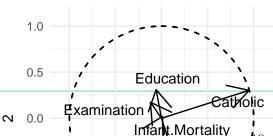
```r
co_circle_ppl <- (
    pco %>%
    prep_co_circle() %>%
    filter(F)
    ) %>%
  ggplot() +
  aes(x=`1`, y=`2`, label=row_id) +
  geom_segment(aes(xend=0, yend=0), arrow = grid::arrow(ends = "first")) +
  ggrepel::geom_text_repel() +
  coord_fixed() +                                                        ①
  xlim(c(-1.1, 1.1)) + ylim(c(-1.1, 1.1))  +
  ggforce::geom_circle(aes(x0=0, y0=0, r=1), linetype="dashed") +
  theme_minimal()
```

① important

```r
co_circle_ppl %+% (
  pco %>%
  prep_co_circle()
  )  +
  labs(title="Correlation circle",
          subtitle = "centered, unscaled",
          caption= "Swiss Fertility dataset") +
  theme_minimal()
```

> **i Question**
>
> Compute PCA after standardizing the columns, draw the correlation circle.

> **solution**
>
> ```r
> pco2 <- select(swiss, -Fertility) |>
>  prcomp(scale. = T)
>
> co_circle_ppl %+% (
>   pco2 %>%
>   prep_co_circle()
>   )  +
>   labs(
>     title="Correlation circle",
>     subtitle = "centered, scaled",
>     caption="Swiss fertility dataset") +
>   theme_minimal()
> ```
>
> ## Correlation circle
> ### centered, scaled
>
>
>
> Swiss fertility dataset
>
> Scaling columns seriously modify the correlation circle.

Compare standardized and non-standardized PCA

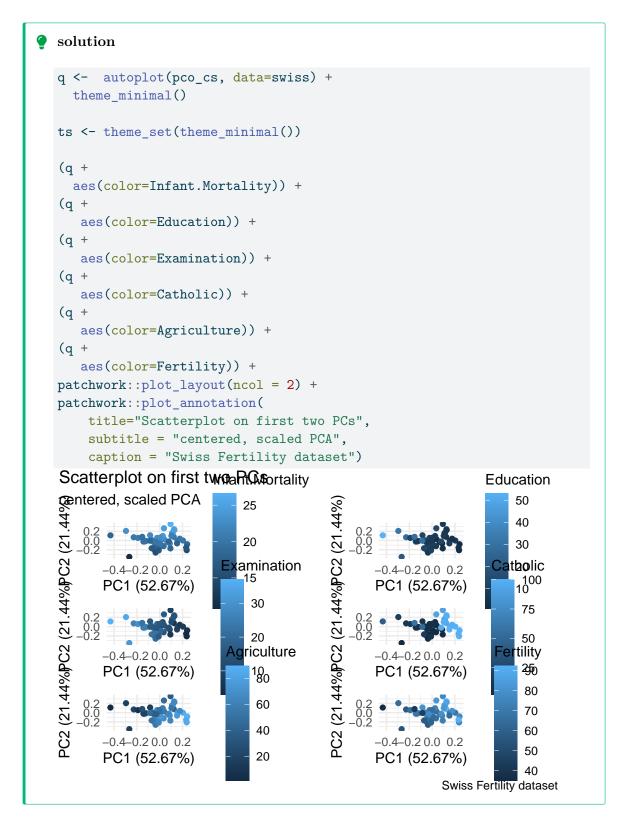> **i Question**
>
> Pay attention to the correlation circles.
>   1. How well are variables represented?
>   2. Which variables contribute to the first axis?

> ### 💡 solution
>
> ```r
> pco_c <- swiss %>%
>   select(-Fertility) %>%
>   prcomp()
>
> pco_cs <- swiss %>%
>   select(-Fertility) %>%
>   prcomp(scale.=T, center=T)
> ```
>
> ```r
> (
>   co_circle_ppl %+%
>   prep_co_circle(pco_c)  +
>   labs(
>     subtitle = "centered, unscaled"
>   ) +
>   theme_minimal()
> ) +
> (
>   co_circle_ppl %+%
>   prep_co_circle(pco_cs) +
>   labs(
>     subtitle = "centered, scaled"
>   ) +
>   theme_minimal()
> ) +
>   patchwork::plot_annotation(
>         title="Swiss, correlation circle"
>   )
> ```
>
> ## Swiss, correlation circle
>
>

> ### ℹ Question
>
> Explain the contrast between the two correlation circles.

In the sequel we focus on standardized PCA.

> 💡 **solution**
>
> ```r
> q <-  autoplot(pco_cs, data=swiss) +
>   theme_minimal()
>
> ts <- theme_set(theme_minimal())
>
> (q +
>   aes(color=Infant.Mortality)) +
> (q +
>   aes(color=Education)) +
> (q +
>   aes(color=Examination)) +
> (q +
>   aes(color=Catholic)) +
> (q +
>   aes(color=Agriculture)) +
> (q +
>   aes(color=Fertility)) +
> patchwork::plot_layout(ncol = 2) +
> patchwork::plot_annotation(
>     title="Scatterplot on first two PCs",
>     subtitle = "centered, scaled PCA",
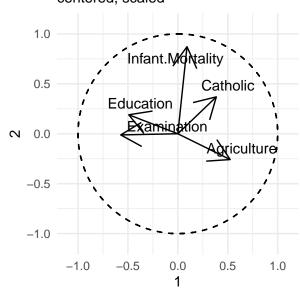>   caption = "Swiss Fertility dataset")
> ```
>
> 

Provide an interpretation of the first two principal axes

> ℹ **Question**
>
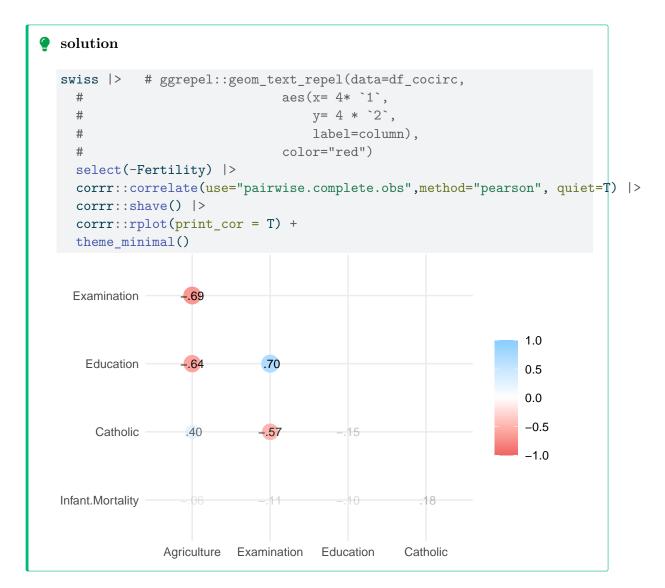> Which variables contribute to the two first principal axes?

> 💡 **solution**
>
> This comes from the correlation circle. We rely on function `prep_co_circle` and on the graphical pipeline `co_circle_ppl`.
>
> ```
> (
>   co_circle_ppl %+%
>     prep_co_circle(pco_cs) +
>     ggtitle("Swiss, correlation circle",
>             subtitle = "centered, scaled") +
>   theme_minimal()
> )
> ```
>
> ## Swiss, correlation circle
> ### centered, scaled
>
>

> ℹ️ **Question**
>
> Analyze the signs of correlations between variables and axes?

> 💡 **solution**
>
> ```
> swiss |>    # ggrepel::geom_text_repel(data=df_cocirc,
>   #                         aes(x= 4* `1`,
>   #                            y= 4 * `2`,
>   #                            label=column),
>   #                         color="red")
>   select(-Fertility) |>
>   corrr::correlate(use="pairwise.complete.obs",method="pearson", quiet=T) |>
>   corrr::shave() |>
>   corrr::rplot(print_cor = T) +
>   theme_minimal()
> ```
>
>

Add the `Fertility` variable

> ℹ️ **Question**
>
> Plot again the correlation circle using the same principal axes as before, but add the `Fertility` variable.
> How does `Fertility` relate with covariates? with principal axes?

> 💡 **solution**
>
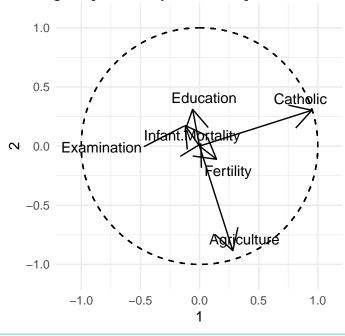> We use
> $$D^{-1} \times U^{\top} \times X = V^{\top}$$
> It is enough to multipliy the data matrix by $D^{-1} \times U^{\top}$ and to pipe the result into the coorelation circle graphical pipeline.
>
> ```r
> foo <- t(diag(svd_Y$d^(-1)) %*% t(svd_Y$u) %*% as.matrix(scale(swiss, scale=F)))
>
> co_circle_ppl  %+% (
>   as_tibble(foo) |>
>   rename_with(.fn = \(x) gsub('V', '', x), .cols=everything()) |>
>   mutate(row_id=rownames(foo))
> ) +
>   theme_minimal()
> ```
>
> ```
> Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
> `.name_repair` is omitted as of tibble 2.0.0.
> i Using compatibility `.name_repair`.
> ```
>
>

Biplot

> ℹ️ **Question**
>
> The last svd plot (biplot) consists of overlaying the scatter plot of component `x` of the `prcomp` object and the correlation circle.
> So the biplot is a graphical object built on two dataframes derived on components `x` and `rotation` of the `prcomp` objects.
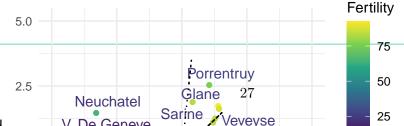> Design a graphical pipeline.

💡 **solution**

```r
pco <- swiss %>%
  select(-Fertility) %>%
  prcomp(scale.=T)

df_cocirc <- pco %>%
  broom::tidy(matrix="v") %>%
  tidyr::pivot_wider(id_cols =column,
              names_from = PC,
              values_from = value)

broom::augment(pco, data=swiss) %>%
  ggplot() +
  geom_point(aes(x=.fittedPC1,
              y=.fittedPC2,
              color=Fertility, label=.rownames)) +
  coord_fixed() +
  ggrepel::geom_text_repel(aes(x=.fittedPC1,
                          y=.fittedPC2,
                          color=Infant.Mortality,
                          label=.rownames)) +
  geom_segment(data=df_cocirc,
           mapping=aes(x= 4* `1`,
                    y= 4 * `2`,
                    linetype=factor(column),
                    label=column,
                    xend=0,
                    yend=0),
           arrow = grid::arrow(ends = "first",
                           unit(.1, "inches")
                           )) +
  scale_color_viridis_c() +
  xlim(c(-5,5)) +
  ylim(c(-5,5)) +
  theme_minimal()
```

```
Warning in geom_point(aes(x = .fittedPC1, y = .fittedPC2, color = Fertility, :
Ignoring unknown aesthetics: label
Warning in geom_segment(data = df_cocirc, mapping = aes(x = 4 * `1`, y = 4 * :
Ignoring unknown aesthetics: label
Warning: ggrepel: 37 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

> **ⓘ Question**
>
> `autoplot.prcomp()` has optional arguments. If set to `True`, logical argument `loadings` overlays the scatterplot defined by the principal components with the correlation circle.

> **ⓘ Question**
>
> `autoplot.prcomp()` has optional arguments. If set to `True`, logical argument `loadings` overlays the scatterplot defined by the principal components with the correlation circle.

Generics

`autoplot()` is an example of S3 generic function. Let us examine this function using `sloop`

> **i** Use `sloop::s3_dispatch()` to compare `autoplot(prcomp(swiss))` and `autoplot(lm(Fertility ~ ., swiss))`

> **solution**
>
> ```
> sloop::ftype(autoplot)
> ```
>
> ```
> [1] "S3"      "generic"
> ```
>
> ```
> sloop::s3_dispatch(autoplot(prcomp(swiss)))
> ```
>
> ```
> => autoplot.prcomp
>  * autoplot.default
> ```
>
> ```
> sloop::s3_dispatch(autoplot(lm(Fertility ~ ., swiss)))
> ```
>
> ```
> => autoplot.lm
>  * autoplot.default
> ```

> **i** Use `sloop::s3_getmethod()` to see the body of `autoplot.prcomp`

32

💡 **solution**

```
sloop::s3_get_method(autoplot.prcomp)
## function (object, data = NULL, scale = 1, x = 1, y = 2, variance_percentage = TRUE,
##     ...)
## {
##     plot.data <- ggplot2::fortify(object, data = data)
##     plot.data$rownames <- rownames(plot.data)
##     if (is_derived_from(object, "prcomp")) {
##         ve <- object$sdev^2/sum(object$sdev^2)
##         PC <- paste0("PC", c(x, y))
##         x.column <- PC[1]
##         y.column <- PC[2]
##         loadings.column <- "rotation"
##         lam <- object$sdev[c(x, y)]
##         lam <- lam * sqrt(nrow(plot.data))
##     }
##     else if (is_derived_from(object, "princomp")) {
##         ve <- object$sdev^2/sum(object$sdev^2)
##         PC <- paste0("Comp.", c(x, y))
##         x.column <- PC[1]
##         y.column <- PC[2]
##         loadings.column <- "loadings"
##         lam <- object$sdev[c(x, y)]
##         lam <- lam * sqrt(nrow(plot.data))
##     }
##     else if (is_derived_from(object, "factanal")) {
##         if (is.null(attr(object, "covariance"))) {
```

33

References

S3 classes

https://scholar.google.com/citations?user=xbCKOYMAAAAJ&hl=fr&oi=ao