Internship Report

Melissa Boutlendj

Supervisor: Anna Melnykova

June 2024

Contents

1 Introduction

This internship report represents my first serious step into academic research. The work presented here is the result of several weeks of study, practice, and exploration of statistical methods, with a particular focus on regression analysis. My main objective during this internship was not only to apply theoretical knowledge to real problems but also to gain a deeper understanding of how statistical tools can be implemented in practice.

Although this document summarizes the progress achieved so far, it also serves as the foundation for future research. The methods, results, and reflections presented here are meant to be continued and further developed in the next stages of my academic journey. I hope that this first experience will contribute to my growth as a researcher and provide a useful reference for the work that follows.

2 Simple Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable Y and one or more independent variables X. In the case of simple linear regression, we focus on a single explanatory variable. The goal is to find the parameters β_0 (intercept) and β_1 (slope) that define the linear relationship between X and Y.

The model is defined as:

$$Y = \beta_1 X + \beta_0 + \epsilon \tag{1}$$

where ϵ is the error term, representing the deviation of the observed values from the predicted values.

To estimate β_1 and β_0, we minimize the Mean Square Error (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \beta_1 X_i - \beta_0)^2 \tag{2}$$

Taking the partial derivatives of MSE with respect to β_1 and β_0 and setting them to zero, we obtain:

$$\frac{\partial \text{MSE}}{\partial \beta_1} = 0 \implies \sum_{i=1}^{n} X_i (Y_i - \beta_1 X_i - \beta_0) = 0 \tag{3}$$

$$\frac{\partial \text{MSE}}{\partial \beta_0} = 0 \implies \sum_{i=1}^{n} (Y_i - \beta_1 X_i - \beta_0) = 0 \tag{4}$$

Solving these equations simultaneously gives the estimators:

$$\hat{\beta}_1 = \frac{C_{xy}}{S_x^2} \tag{5}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \tag{6}$$

where C_{xy} is the empirical covariance between X and Y, and S_x^2 is the empirical variance of X.

2.1 Distributions of $\hat{\beta}_0$ and $\hat{\beta}_1$

Assuming normally distributed errors, the estimators $\hat{\beta}_1$ and $\hat{\beta}_0$ under a normally distributed $\epsilon \sim \mathcal{N}(0, \sigma^2)$ noise follow normal distributions:

$$\hat{\beta}_1 \sim \mathcal{N}\left(\beta_1, \frac{\sigma^2}{nS_x^2}\right)$$

$$\hat{\beta}_0 \sim \mathcal{N}\left(\beta_0, \frac{\sigma^2}{n}\left(1 + \frac{\bar{X}^2}{S_x^2}\right)\right)$$

where σ^2 is the variance of the errors. These estimators are unbiased:

$$\mathbb{E}[\hat{\beta}_1] = \beta_1$$

$$\mathbb{E}[\hat{\beta}_0] = \beta_0$$

To better understand the distribution of the estimated regression coefficients, we implemented an R function that calculates the parameters $\hat{\beta}_0$ and $\hat{\beta}_1$ given inputs Y and X (allowineg for multiple variables). The function computes the means of Y and X, the covariance between them, and the variance of X to use the formulas defined above.

We then ran the function 1000 times on random inputs, assuming the noise ϵ is normally distributed. By visualizing the histograms and density curves of the estimated coefficients, we can gain insights into their distribution. We considered three different values of noise variance (σ^2).

The histograms and density curves help us deduce the followineg:

- **Normality of Estimates:** The histograms of $\hat{\beta}_0$ and $\hat{\beta}_1$ are bell-shaped, indicating that the estimates are approximately normally distributed.

- **Bias of Estimates:** The means of the histograms are close to the true parameter values, suggesting that our estimators are unbiased.

- **Variance of Estimates:** The spread of the histograms reflects the variability of the estimates. Higher noise variance (σ^2) results in a wider spread, indicating less precise estimates.

- **Effect of Noise Variance:** As σ^2 increases, the histograms and density curves become broader and flatter, showineg that higher noise leads to greater variability in the estimates.

Note: The larger the dataset used, the better the density curve fits our data, according to the Law of Large Numbers

Below are the results for the different values of σ^2 and for estimated coefficient : $\hat{\beta}_0$= -39.04714 and $\hat{\beta}_1$= 4.814762

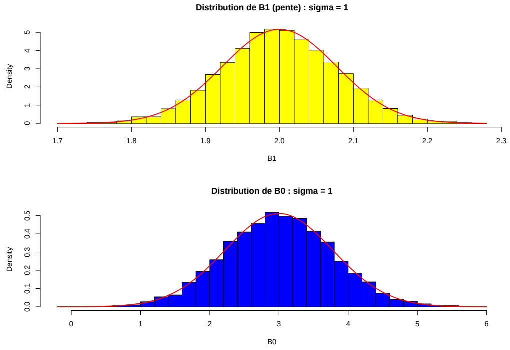

Figure 1: Distribution de β_1 and β_0: $\sigma = 1$

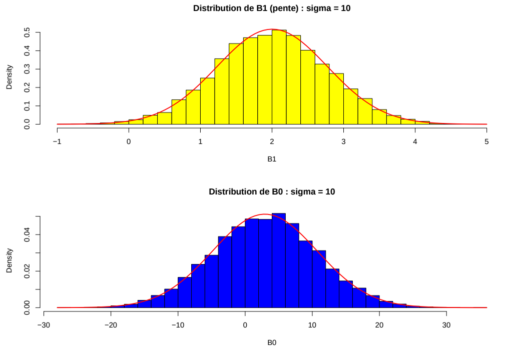

Figure 2: Distribution de β_1 and β_0: $\sigma = 10$

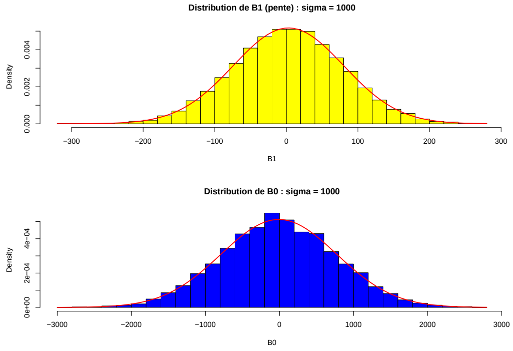

Figure 3: Distribution de β_1 and β_0: $\sigma = 1000$

2.2 Residual Analysis

Residuals, denoted as e_i, are the differences between the observed values of the dependent variable Y_i and the predicted values $\hat{Y}_i$. The residual for each observation i is given by:

$$e_i = Y_i - \hat{Y}_i.$$

Here, Y_i represents the actual value of the dependent variable for the i-th observation, and $\hat{Y}_i$ represents the predicted value of the dependent variable based on the regression model.

Residuals provide a measure of the discrepancy between the observed data and the model's predictions. They are crucial for assessing the goodness of fit of the model, identifying outliers, and checking for violations of model assumptions. In simple terms, residuals indicate how well the regression model captures the variation in the data. Here is the example we implemented to demonstrate simple linear regression: the relationship between speed and braking distance. the code can be found here (A.1)

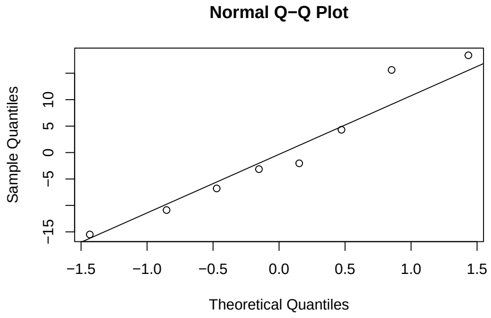

Figure 4: QQ plot for residuals

The Shapiro-Wilk normality test is a statistical test used to assess whether a given sample of data comes from a normally distributed population. After executing this R code we get those results : checking the normal distribution of residuals is crucial for the validation of the results of the model. The test rejects the hypothesis of normality when the p-value is less than or equal to 0.05. According to the results below, a p-value of 0.5736 is big enough to say that our data is normally distributed ei : the normality hypothesis has a big probability to be true

```

> # Shapiro-Wilk test for normality
> shapiro_test <- shapiro.test(resi)
> print("Shapiro-Wilk Test:")
[1] "Shapiro-Wilk Test:"
> print(shapiro_test)

 Shapiro-Wilk normality test

data:  resi
W = 0.93615, p-value = 0.5736
```

2.3 Student test

Before introducing the t-test, we compute the **t-statistics** for each regression coefficient. The t-statistic measures how many standard errors the estimated coefficient is away from the hypothesized value (usually zero). It is calculated as the ratio of the estimated coefficient minus its assumed value to its standard error:

$$t_i = \frac{\hat{\beta}_i - \beta_{i,0}}{\mathrm{SE}(\hat{\beta}_i)}$$

The **t-test**, also called the Student t-test or t-distribution, is a statistical test used to assess whether the means of two groups differ significantly or whether a single coefficient differs from zero. Performing this test allows us to determine if observed differences are statistically significant or if they could have arisen by chance. A small p-value (typically ≤ 0.05) indicates that the null hypothesis can be rejected, suggesting a meaningful difference, while a large p-value indicates insufficient evidence to reject the null hypothesis.

We define the same linear regression model as in Section 2. Using the Student t-test, we want to determine if the estimated coefficients $\hat{\beta}_1$ and $\hat{\beta}_0$ are significantly different from zero.

Hypotheses: For each regression coefficient β_i, the null hypothesis (H_0) and alternative hypothesis (H_1) are:

$$H_0 : \beta_i = 0$$
$$H_1 : \beta_i \neq 0$$

We obtained the t-statistics and p-values necessary to test the null hypothesis of the coefficients. If the calculated t statistic exceeds a specific critical value, we reject the null hypothesis.

We implemented a function in R called `StudentTest`. This function calculates the t-statistics and p-values for the coefficients under the Student's t-distribution (see code reference: A.3).

Function explanation:

- The function first calculates the estimated coefficients $\hat{\beta}_1$ and $\hat{\beta}_0$.
- **t-statistics:** the ratio of the difference between an estimated coefficient and its hypothesized value to its standard error.
  - $\hat{\sigma}$ is the standard deviation of the residuals.
  - S_{xx} is the sum of squares of the deviations of X from its mean:

$$S_{xx} = \sum (X_i - \bar{X})^2$$

The standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ are calculated as follows:

$$SE_{\hat{\beta}_1} = \frac{\hat{\sigma}}{\sqrt{S_{xx}}},$$
$$SE_{\hat{\beta}_0} = \hat{\sigma}\sqrt{\frac{1}{n} + \frac{\bar{X}^2}{S_{xx}}}.$$

To assess the significance of the estimated regression coefficients, we calculate the **t-statistics**. Under the null hypothesis $H_0 : \beta_j = 0$, the test statistic follows a Student's t-distribution with $(n-k)$ degrees of freedom, where n is the number of observations and k is the number of estimated coefficients in the model:

$$t_{\hat{\beta}_1} = \frac{\hat{\beta}_1}{SE_{\hat{\beta}_1}} \sim t(n-k),$$
$$t_{\hat{\beta}_0} = \frac{\hat{\beta}_0}{SE_{\hat{\beta}_0}} \sim t(n-k).$$

Note: If the number of observations n is less than the number of parameters k, the model may be at risk of degeneracy.

p-values: The p-values are computed from the t-distribution to determine the significance of the coefficients. A low p-value (typically < 0.05) indicates that the coefficient is significantly different from zero, providing evidence against the null hypothesis.

They are calculated as follows:

$$p_{\hat{\beta}_1} = 2 \cdot \mathrm{pt}(-|t_{\hat{\beta}_1}|, \mathrm{df} = n-2), \quad p_{\hat{\beta}_0} = 2 \cdot \mathrm{pt}(-|t_{\hat{\beta}_0}|, \mathrm{df} = n-2).$$

2.3.1 Implementation

For the same data sample, we applied the `StudentTest` function (see Appendix A.1) and obtained the following results:

```
test <- StudentTest(speed, distance)
```

Estimated coefficients:

```
B0: -39.04714
B1: 4.814762
```

Thus, the estimated regression equation of the braking distance as a function of vehicle speed is:

$$\widehat{distance} = 4.814762 \cdot speed - 39.04714 \tag{7}$$

Corresponding p-values:

```
p_B0: 0.008504877
p_B1: 2.058236e-05
```

Interpretation:

- The intercept $\hat{\beta}_0 = -39.04714$ represents the estimated braking distance when the vehicle speed is zero. While negative distances are not physically meaningful, this value results from the extrapolation of the regression line and mainly adjusts the model to the observed data. Its p-value ($0.0085 < 0.05$) shows that the intercept is significantly different from zero.

- The slope $\hat{\beta}_1 = 4.814762$ indicates that, on average, the braking distance increases by about 4.81 units for every additional unit of speed. Its p-value (2.06×10^{-5}) is extremely small, confirming that speed has a highly significant effect on braking distance.

- Since both p-values are well below 0.05, we reject the null hypotheses $H_0 : \beta_0 = 0$ and $H_0 : \beta_1 = 0$. This means that both coefficients contribute significantly to the model.

In conclusion, the regression model demonstrates a strong linear relationship between vehicle speed and braking distance, confirming that the braking distance increases significantly with higher speed. This validates the efficiency and relevance of the model.

3 Multivariate Regression Analysis

3.0.1 Finding the matrix of coefficients

After performing regression in one dimension, we extended our analysis to multiple dimensions. Let $\boldsymbol{X}$ denote the design matrix of predictors, $\boldsymbol{Y}$ the vector of the target variable, and $\boldsymbol{\beta}$ the vector of coefficients to be estimated. The multivariate regression model is defined as:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{8}$$

where the error terms are assumed to follow a normal distribution:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I_n).$$

Expanded form: For an observation i with d predictors $(x_{i1}, x_{i2}, \dots, x_{id})$, the model can be written as:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_d x_{id} + \epsilon_i. \tag{9}$$

Thus, in multiple dimensions, the regression equation expresses the dependent variable as a linear combination of several independent variables, each weighted by its corresponding coefficient.

To calculate the parameters of the regression, we first bind the matrix $\boldsymbol{X}$ (the explanatory variables) to add a column of ones that will later hold the value of the intercept. Using the formula extracted from "The matrix cook book":

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

We also need the formula to calculate the mathematical expectation and the variance

$$\begin{aligned}
\mathbb{E}[\hat{\beta}] &= \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}[\mathbf{Y}] \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta = \beta
\end{aligned}$$

$$\begin{aligned}
Var[\hat{\beta}] &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Var[\mathbf{Y}] \left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\right)^T \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\left(\mathbf{X}^T\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\right)^T\right) \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\left(\mathbf{X}^T\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\right)^T\right) \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\left(\mathbf{X}^T\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\right)\right) \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}
\end{aligned}$$

st : $(\mathbf{X}^T\mathbf{X})^{-1}$ is the inverse matrix,
$\mathbf{X}^T$ is the transpose of $\mathbf{X}$,
$\mathbf{X}^T\mathbf{Y}$ represents the matrix multiplication.

Regarding the residuals: After binding, we calculated the mean square errors using the followineg formula:

$$\mathbf{e} = \mathbf{Y} - \mathbf{X}\boldsymbol{\beta}$$

3.0.2 Student test in multivariate analysis :

Just like we did in simple linear regression, The `StudentTest` function allows for the statistical testing of regression coefficients in multiple linear regression. We define t-statistics of the previous model defined in equation (9)

By analyzing the coefficient estimates, standard errors, t-statistics, and p-values, we can determine the significance and impact of each explanatory variable on the dependent variable. The function demonstrates that even with randomly generated data, it is possible to identify significant relationships between variables, provided that the data is appropriately modeled and analyzed. To perform the Student's test in multiple dimensions, we utilized matrix calculations.

The Student test function and the Implementation example are here A.4 The function first calculates the coefficients then

Implementation example :

We consider a simple matrix $\boldsymbol{X}$ of dimension 10 * 100 which mean 10 explanatory variables and 100 observation, and a predicted variable0 then we crea key metrics from the regression analysis can be interpreted as follows:

- **Residual Standard Error**: 0.9899 on 89 degrees of freedom. Interpretation: The Residual Standard Error (RSE) provides a measure of the typical distance that the observed values fall from the regression line. An RSE of 0.9899 indicates that, on average, the actual values deviate from the predicted values by approximately 0.99 units. This relatively small error suggests that the model fits the data well.

- **Multiple R-squared**: 0.9933 Interpretation: The Multiple R-squared value represents the proportion of variance in the dependent variable that is explained by the independent variables. An R-squared value of 0.9933 means that 99.33 percent of the variance in the response variable Y is explained by the model. This indicates a very high level of explanatory power, suggesting that the model fits the data exceptionally well.

- **Adjusted R-squared**: 0.9926 Interpretation: The Adjusted R-squared adjusts the R-squared value for the number of predictors in the model, providing a more accurate measure of the model's explanatory power. An Adjusted R-squared value of 0.9926, which is very close to the Multiple R-squared value, indicates that the model remains highly explanatory even when accounting for the number of predictors.

- **F-statistic**: 1323 on 10 and 89 degrees of freedom, with a p-value $< 2.2e\text{-}16$ Interpretation: The F-statistic tests the overall significance of the regression model. An F-statistic of 1323, with corresponding degrees of freedom, indicates that the model is statistically significant. The very high value of the F-statistic suggests that the independent variables, as a group, significantly predict the dependent variable.

- **P-Value** : $< 2.2e\text{-}16$
  The p-value associated with the F-statistic indicates the probability that the observed F-statistic would occur if the null hypothesis (that all regression coefficients are equal to zero) were true. A p-value less than 2.2e-16 (which is extremely small) indicates that there is a very low probability that the observed F-statistic is due to chance. This means that the model is highly significant

This indicates a highly significant model with a good fit, explaining a large proportion of the variance in the response variable.

4 Logistic Regression

So far we know how to implement linear regression to find the best fitted model that predict a value of a dependent variable Y using explanatory variables X.But what if our Y can take only two values zero or one, true or false.Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given data set of independent variables.

The main differences between the two are :

- Linear regression is used to predict the continuous dependent variable using a given set of independent variables while Logistic Regression is used to predict the categorical dependent variable using a given set of independent variables.
- Linear Regression is used for solving Regression problem. Logistic regression is used for solving Classification problems.
- In linear regression, we find the best fit line, by which we can easily predict the output. In Logistic Regression, we find the S-curve by which we can classify the samples.
- In Linear regression, it is required that relationship between dependent variable and independent variable must be linear. In Logistic regression, it is not required to have the linear relationship between the dependent and independent variable.

4.1 Estimation of the parameters

To implement our very first logistic regression model I used a dataset regarding the survival of Titanic passengers. some description of the data set

```
     PassengerId Survived Pclass
                            Name    Sex      Age SibSp Parch
1             1        0      3
     Braund, Mr. Owen Harris   male 22.00000     1     0
2             2        1      1       Cumings, Mrs. John Bradley
    (Florence Briggs Thayer) female 38.00000     1     0
3             3        1      3
      Heikkinen, Miss. Laina female 26.00000     0     0
4             4        1      1              Futrelle, Mrs.
    Jacques Heath (Lily May Peel) female 35.00000     1
```

To do this, I divided the data into two parts: one for training and the other for testing. I then used the glm function to perform the logistic regression and calculate the parameters.To calculate the prediction I used the sigmoid function.We can adjust the function according to the number of independent variables.

$$F(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{10}$$

I then chose three variable to train my model (Age, train,Fare) , to keep things simple I did not use qualitative variable like sex. to make the model more efficient I replaced the missing values in the age column by the mean age of passengers. I calculated the probability that a passenger survives on the training and the testing data and I consider that the personnel is alive if the result >= 0.5.

4.2 Validation of the result

To verify whether the model was efficient, I tried to calculate the MSE of output of my model, but I did not get good estimation, so I calculated the F1 score.

4.2.1 F1 score

F1 score is a machine learning evaluation metric that combines precision and recall scores

$$F1 \text{ score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

where

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

5 Principal Component Analysis

5.1 Singular Value Decomposition (SVD)

Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are two fundamental techniques in linear algebra and data analysis. They play a crucial role in reducing the dimensionality of data and extracting essential information.

SVD is a factorization method that decomposes a matrix into three other matrices: U, Σ, and $\mathbf{V}^\mathsf{T}$. Here is a simplified breakdown:

Matrix $A \in \mathbb{R}^{m \times n}$ is decomposed into $U \in \mathbb{R}^{m \times m}, \Sigma \in \mathbb{R}^{m \times n}$, and $\mathbf{V} \in \mathbb{R}^{n \times n}$ U contains orthogonal columns that represent the left singular vectors. Σ is a diagonal matrix containing the singular values. $\mathbf{V}^\mathsf{T}$ contains orthogonal rows representing the right singular vectors. SVD is used to approximate a matrix with lower rank, retaining the most significant singular values. To implement this important concept step by step we consider an R function that takes a matrix and calculates the singular value by calculating the eigen values first then the eigen vectors. The aim is to find the tree matrices that we will multiply the corresponding code here. A.6

Before diving into how to implement PCA we should discuss some important notions like data normalization and the covariance matrix

5.1.1 Matrix normalization

Considering a matrix of quantitative explanatory variables, the value of each of them is different (in the `cars` data set it's normal that the weight of a vehicle is much greater than its fuel consumption). Normalization consists of subtracting the mean of each column and dividing by its standard deviation, it is a crucial step before calculation SVD or PCA values because it guarantees the adjustment of the used variables though the efficiency of results. The code for normalization is available here A.6.1.

5.1.2 Covariance matrix :

The variance-covariance matrix is a square matrix with diagonal elements that represent the variance and the non-diagonal components that express covariance. The covariance of a variable can take any real value-positive, negative, or zero. A positive covariance suggests that the two variables have a positive relationship, whereas a negative covariance indicates that they do not. If two elements do not vary together, they have a zero covariance. code here A.6.2

5.1.3 Principal Component Analysis :

PCA is a statistical procedure that aims to transform data into a new coordinate system where the axes are the principal components. These components are orthogonal and capture the maximum variance in the data. To perform the PCA, we will go through three essential steps :

- Normalize the data to have zero mean and unit variance.
- Compute the covariance matrix of the standardized data.
- Compute the eigenvectors and eigenvalues of the covariance matrix (Eigenvalues represent the total amount of variance that can be explained by a given principal component. They can be positive or negative in theory, but in practice they explain variance, which is always positive.)
- Sort eigenvalues in descending order and choose the top-k eigenvalues to form principal components.(we chose the highest eigenvalues because we want to find the variables with bigger variance).
- Project the original data onto the principal components to create a lower-dimensional representation, (using some R libraries we can draw the scree plot of the PCA)

5.1.4 The relationship between SVD et PCA :

While PCA and SVD aren't directly comparable, it's important to highlight the relationship between them. PCA often employs SVD as a mathematical tool to achieve its goals. PCA is the multiplication of the matrix U obtained from the SVD of the normalized initial matrix, which gives us the PCA matrix. This matrix groups the most important variables in the first columns.

5.1.5 Implementation example of PCA :

here is the code of the R function that does the PCA following the previous steps.A.6.3 we used the wine quality data set to apply the PCA on it after finding the three matrices U, Σ and V, using SVD method, we then performed the dot product of the normalized data matrix with the U matrix (the two matrices should be from the same dimension).Finally, we obtained the scree plot that shows the variance explained by each component in PCA. It plots the eigenvalues, which are the measures of how much each component contributes to the total variance, against the component number.In the figure below we can notice that the first 4 component have a considerable variance. The scree plot criterion looks for the “elbow” in the curve and selects all components just before the line flattens out. (In the PCA literature, the plot is called a ‘Scree’ Plot because it often looks like a ‘scree’ slope, where rocks have fallen down and accumulated on the side of a mountain.) the “elbow” in this figure if I might say is component number 4.

Scree plot

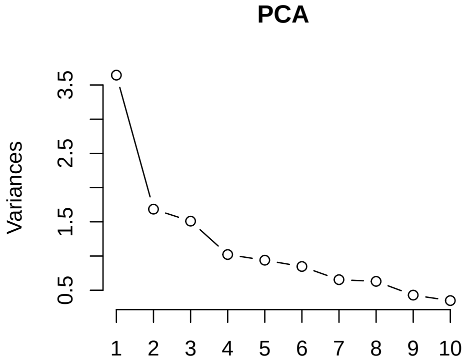

Figure 5: Scree plot of PCA applied on the wine quality data set

To figure out what are those important component :

```
    # Extract the loadings
> loadings <- PCA$rotation
>
> # Get the loadings for the first principal component (PC1)
> pc1_loadings <- loadings[, 1]
> pc1_loadings
         fixed.acidity     volatile.acidity          citric.acid
          residual.sugar             chlorides
          -0.13985253          -0.03717914          -0.15916346
            -0.42207636          -0.17374524
free.sulfur.dioxide total.sulfur.dioxide              density
                   pH              sulphates
          -0.31756473          -0.36912124          -0.47665042
             0.23553496          -0.01083701
              alcohol              quality
           0.41877810           0.22389145
```

We can affirm that fixed.acidity,volatile.acidity ,citric.acid ,residual.sugar. are the varaibles that effect more the quality of wine.

To understand the relationship between the diffrent variables we draw a Biplot using component 1 and 2. By selecting the two variables with the highest eigenvalues. In the figure 7 below, we observe how the data is grouped: wines that are more sugary and dense are on the left, while wines that are less dense and sugary are on the right. you find here the code that applies the PCA function to the wine quality data set A.6.3 some figures that show the result :

Biplot :

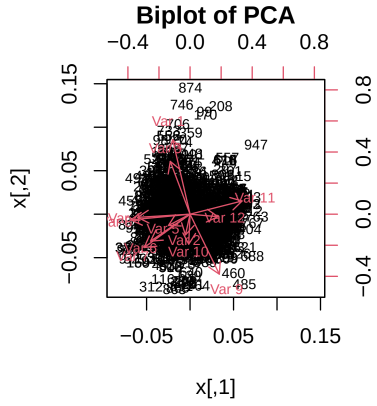

Figure 6: Biplot of PCA applied on the wine quality data set

- Why do we need it
- How to do it
- What is the Principal Component?

- How to understand if the model is good?

What to do with the practical part:

1. Start with taking some matrix of observations, which we will call X (for example, winee quality data, or `iris` data in R)

2. Compute the covariance matrix of the observations (we will call it Σ in what follows). Think about how can it be expressed in a matrix multiplication form

3. Using singular value decomposition (you can find a corresponding function in R, you can use it), write your covariance matrix as a product of three matrices which you will define. For that, you need to read about SVD a bit.

4. Look at the obtained matrixes. What are the eigenvectors of the Σ? What are the eigenvalues?

5. Do a change of basis for your initial matrix of observations, using the decomposition. Plot the first two dimensions of the resulting data. Do they well represent the dataset? How can we know it?

6. Normalize the data (so that each variable has zero mean and 1 variance) and perform the PCA on the new, normalized dataset. Plot the first two dimensions. Did something change?

7. Look into a documentation of `FactoMineR` package and perform PCA using the available functions. You can reproduce the example (with the corresponding plots and so on) from this page: `http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/`

A Code Listings

A.1 Shapiro-Wilk test for normality

```
speed <- seq(5,40,5)
distance <- c(3.42,5.96,31.14,41.76,74.54,94.52,133.78,169.16)

resi <- Residue(speed, distance)

# Visualize residuals using a QQ plot
qqnorm(resi)
qqline(resi)


# Shapiro-Wilk test for normality
shapiro_test <- shapiro.test(resi)
print("Shapiro-Wilk Test:")
print(shapiro_test)
```

A.2 Simple Linear regression

How the parameters are computed? (I think it's done ?) The code below explains the formulas for calculating p-values and t-statistics

```
CalculeParam <- function(x, y)
{
  x_mean = mean(x)
  y_mean = mean(y)

  cxy <- cov(x, y) #covariance entre x et y
  sxx <- var(x) # la variance de x

  B1 = cxy/sxx
  B0 = y_mean - B1 * x_mean

  coefficients <- c(B1,B0)

  return(coefficients)

}

StudentTest <- function(x, y) {
  # Get the number of observations
  n <- length(x)
```

```r

# Calculate the regression coefficients
coeffs <- CalculeParam(x, y)
B1 <- coeffs[1]
B0 <- coeffs[2]

# Calculer les residus
residuals <- Residue(x, y)

# Nombre d'observations
n <- length(y)

# Estimer l'erreur standard des residus
sigma_hat <- sqrt(sum(residuals^2) / (n - 2))

# Calculer les erreurs standard des coefficients
x_mean <- mean(x)
sxx <- sum((x - x_mean)^2)
se_B1 <- sigma_hat / sqrt(sxx)
se_B0 <- sigma_hat * sqrt(1/n + x_mean^2 / sxx)

# Calculer les t-statistiques
t_B1 <- B1 / se_B1
t_B0 <- B0 / se_B0

# Calculer les p-values
p_B1 <- 2 * pt(-abs(t_B1), df = n - 2)
p_B0 <- 2 * pt(-abs(t_B0), df = n - 2)

# Afficher les resultats
cat("Coefficient estimates:\n")
cat("B0:", B0, "\n")
cat("B1:", B1, "\n\n")

cat("Standard errors:\n")
cat("SE_B0:", se_B0, "\n")
cat("SE_B1:", se_B1, "\n\n")

cat("t-statistics:\n")
cat("t_B0:", t_B0, "\n")
cat("t_B1:", t_B1, "\n\n")

cat("p-values:\n")
cat("p_B0:", p_B0, "\n")
cat("p_B1:", p_B1, "\n")
```

```

  # Retourner les resultats sous forme de liste
  return(list(B0 = B0, B1 = B1, p_B0 = p_B0, p_B1 = p_B1))
}
```

A.3 Residuals calculation

```

speed <- seq(5,40,5)
distance <- c(3.42,5.96,31.14,41.76,74.54,94.52,133.78,169.16)

resi <- Residue(speed, distance)

# Visualize residuals using a QQ plot
qqnorm(resi)
qqline(resi)
```

A.4 Multivariate Linear Regression

```
CalculeParam <- function(X, Y) {
  # Ajouter une colonne de 1s pour le terme d'interception
  X <- cbind(1, X)

  # Calculer la matrice des coefficients de regression
  beta <- solve(t(X) %*% X) %*% t(X) %*% Y

  return(beta)
}


Residue <- function(x,y)
{
  B=CalculeParam(x,y)
  x <- cbind(1, x)

  e= y - x %*% B
  return (e^2)

}

StudentTest <- function(x, y) {
  #x <- cbind(1, x)  #je vais faire le bind dna la foncton calculParam
  n <- nrow(x)
  p <- ncol(x)


  coeffs <- CalculeParam(x, y)  # Calculer les coefficients de regression
  residuals <- Residue(x, y)  # Calculer les residus

  sigma_hat <- sqrt(sum(residuals^2) / (n - 2))

  #x_mean <- colMeans(x)

  # x^T x
  matrice_inv <- solve(t(x) %*% x)
  x1 = cbind(1,x)
  x_mean <- colMeans(x1)
  se <- numeric(p+1)
  for (i in 1:(p+1)) {
    if (i == 1) {
      # B0
      #se[i] <- (sigma_hat / n) * sqrt(1 + x_mean[i]^2 / sum((x1[, i] - x_mean[i])^2))
      se[i] <- (sigma_hat / n) * (1 + (x_mean[i]^2 / var(x[, i])))
    } else {
      sxx <- sum(((x1[, i] - x_mean[i])^2)/n)
      se[i] <- sigma_hat / sxx
    }
  }

  t_statistics <- coeffs / se
  p_values <- 2 * pt(-abs(t_statistics), df = n - 2)

  return(list(coeffs = coeffs, se = se, t_statistics = t_statistics, p_values = p_values))
}


StudentTest <- function(x, y) {
  n <- nrow(x)
  p <- ncol(x)
  x <- cbind(1, x)

  coeffs <- solve(t(x) %*% x) %*% t(x) %*% y
  residuals <- y - x %*% coeffs
```

```r
  sigma_hat <- sqrt(sum(residuals^2) / (n - p - 1))

  # Calculer la matrice (X'X)^-1
  XTX_inv <- solve(t(x) %*% x)

  se <- sigma_hat * sqrt(diag(XTX_inv))
  t_statistics <- coeffs / se
  p_values <- 2 * pt(-abs(t_statistics), df = n - p - 1)


  return(list(coeffs = coeffs, se = se, t_statistics = t_statistics, p_values = p_values))
}


d <- 10  # Nombre de variables explicatives
# Coefficients des variables explicatives
B <- c(4.8, -1.5, 2.7, -3, 0.5, 3.6, 2.5,2, 5,8.7)

n <- 100  # Nombre d'observations

X <- matrix(rnorm(n * d), ncol = d)
Y <- 3 + X %*% B + rnorm(n)
resi = StudentTest(X,Y)
resi
```

A.5 Logistic Regression

A.6 PCA

```r
CalculeSVD <- function(A) {
  ATA <- t(A) %*% A
  AAT <- A %*% t(A)

  ATA.e <- eigen(ATA)
  AAT.e <- eigen(AAT)
  v.mat <- ATA.e$vectors
  u.mat <- AAT.e$vectors[, 1:ncol(A)]

  # Singular values
  r <- sqrt(ATA.e$values)
  r.mat <- diag(r)
  svd.matrix <- u.mat %*% r.mat %*% t(v.mat)

  return(list(u = u.mat, s = r.mat, v = v.mat, reconstructed = svd.matrix))
}
```

A.6.1 Data normalization code :

```r
Normalize <- function(A)
{
  norA <- matrix(0, nrow = nrow(A), ncol = ncol(A))

  x_bar <- colMeans(A)
  sd_A <- apply(A, 2, sd)

  for (i in 1:ncol(A)) {
    norA[, i] <- (A[, i] - x_bar[i]) / sd_A[i]
  }

  return(norA)
}
```

```r
    CalculePCA <- function(A) {
  c <- Normalize(A)
  svd_matrix <- CalculeSVD(c)
  u <- svd_matrix$u
  s <-svd_matrix$s
  # dim(u) = 10   4 = dim(c)
  pca_matrix <- c*u
  eigenvalues <- diag(s)^2
  return(list(pca = pca_matrix, u = u, s = svd_matrix$s, v = svd_matrix$v, eigenv = eigenvalues))
}
```

A.6.2 PCA emplimentation code

```r
#fonction qui calcule la matrice de  covariance
CalculeCov <- function(A)
{
c <- Normalize(A)
cov_matrix <- matrix(0, nrow = ncol(c), ncol = ncol(c))

for (i in 1:nrow(c)) {
  centered_row <- c[i,]
  cov_matrix <- cov_matrix + centered_row %*% t(centered_row)
}

cov_matrix <- cov_matrix/ (nrow(c) - 1)

return(cov_matrix)
}
```

A.6.3 PCA en wine quality data set

```
train_full <- read.csv("/home/melissa/Desktop/wine/StageState/wineequality-white.csv", sep = ";")

train = train_full[1:1000,]
str(train)
res = CalculePCA(train)
biplot(res$u[, 1:2], res$v[, 1:2], cex = 0.7, main = "Biplot of PCA")
biplot(res$u[, 2:3], res$v[, 2:3], cex = 0.7, main = "Biplot of PCA")
plot(res$eigenv, type = "b")

# the same, but with prcomp
res_built = prcomp(train, scale = TRUE)
summary(res_built)
biplot(res_built, cex = 0.4, lwd = 2)
plot(res_built,type = "l")
```

References

[**1**] Petersen, Kaare Brandt, and Michael Syskind Pedersen. ”The matrix cookbook.” Technical University of Denmark 7.15 (2008): 510.

[**2**] Lecture notes on Linear Regression (by Cosma Rohilla Shalizi). `https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/13/lecture-13.pdf`

[**3**] Shalizi, Cosma. ”Advanced data analysis from an elementary point of view.” (2013).

[**4**] Lecture notes on ”Principes et Méthodes Statistiques” by Olivier Gaudoin `https://membres-ljk.imag.fr/Olivier.Gaudoin/PMS.pdf`

[**5**] Puntanen, Simo. ”Methods of multivariate analysis, by alvin c. rencher, william f. christensen.” International Statistical Review 81.2 (2013): 328-329.