# Licence 3 informatique – Université de Lille – Facultés des Sciences et Technologies

## RAPPORT DE STAGE

## TITRE : Réalisation d'une bibliothèque de visualisation de données (DataChart) avec Roassal et DataFrame sur Pharo

Diallo Ibrahima Sambegou

Du 11 Avril 2022 au 29 juillet 2022

**Tuteur de Stage :** M. Miton Mamani

**Tuteur Universitaire :** M. Clément Quinton

**Établissement :** Université de Lille – Cité Scientifique 59650 Villeneuve -d'Ascq

**Entreprise :** INRIA – 40 Av. Halley 59650 Villeneuve-d'Ascq

**Équipe :** RMoD

# REMERCIEMENTS

J’aimerais tout d’abord commencer à remercier tout le personnel de l’université, qui durant toutes ces années nous ont accompagné et instruit.

J’envoie aussi mes remerciements à M. Stéphane DUCASSE, directeur de recherche et responsable de l’équipe de recherche RMoD , pour l’opportunité de stage qu’il m’a offert.

Je tiens à remercier aussi M. Milton Torres mon tuteur de stage, et ingénieur dans l’équipe RMoD. Avec qui j’ai travaillé durant ce stage, qui a été bien veillant avec moi et qui m’a épaulé tout au long de mon travail.
Sans oublier bien sûr les autres ingénieurs de l’équipe comme M. Oleksandr Zaitsev qui m’a d’ailleurs offert une formation sur coursera , Mme Agouf, M. Jordan Montano et bien d’autres.

Je remercie aussi M. Clément Quinton mon tuteur universitaire qui a accepté de m’accompagner

Mes remerciements vont aussi à tous les stagiaires présents avec moi, en commençant par Aboubacar , Abir, Antoine, Romain, Gaylord …, qui ont été d’une aide, dans le sens où, on a partagé des moments précieux et on a appris beaucoup de choses ensemble.

Je me remercie aussi pour l’effort que j’ai fourni pour chercher ce stage et pour le travail que j’ai réalisé.

# RÉSUMÉ

## Bibliothèque de visualisation de données sur Pharo DataChart

Dans le cadre de la validation de ma licence informatique, j'ai effectué un stage d'une durée de 4 mois au sein d'INRIA à Villeneuve-d'Ascq dans l'équipe RMoD.

Cette équipe a créé un langage orienté objet du nom de pharo, avec lequel elle travaille.
Nous avons sur Pharo beaucoup de bibliothèques. Mais pour l'instant, il n'existe aucune qui consiste à une visualisation compréhensive de données informatiques. Accompagné de mon tuteur de stage, mon travail consiste à la réalisation d'une bibliothèque de visualisation de données.

La visualisation de donnée ou encore appelé DataViz pour les anglophones, est toutes représentations graphiques de données statistiques, qui permettent une meilleure compréhension de ces données par un grand public. Pour commencer, j'avais en ma disposition deux bibliothèques, le premier nommé s, une bibliothèque permettant de faire des graphes, schémas, formes, dessins etc. Et le second nommé DataFrame, une bibliothèque qui a pour but de manipuler des données.

L'idée est d'utiliser ces deux bibliothèques pour pouvoir définir mes différents visuels. Et pour cela, je me suis référé à la bibliothèque Matplotlib de python, car elle est complète et est l'une des plus utilisées dans ce domaine.

La première partie de mon travail consistait à extraire ces données de mon dataframe pour les mettre dans un autre élément. La seconde partie, s'occupe essentiellement de faire mes graphes en fonction de données extraites ou introduites. En fin, la troisième partie, ajoute tous les petits détails nécessaires à la compréhension du visuel.

Une fois terminé, j'avais donc toute la base de mon projet, on peut donc rajouter autant de formes de graphes différents qu'on veut.

À ce stade du stage, je n'ai pas encore fini tous les visuels et autres arrangements que je dois faire. Mais durant les deux mois qui suivent, je compte bien arriver au bout de mes objectifs.
Ce stage aura été pour moi une grande découverte, non seulement dans le domaine informatique, mais aussi dans le monde professionnel.

# INTERSHIP REPPORT RESUME

## Data vizualisation library on Pharo DataChart

As part of the validation of my computer license, I did a 4-month internship at INRIA in Villeneuve-d'Ascq in the RMoD team.

This team has created an object-oriented language called pharo with which they work.
We have many libraries on Pharo. But for the moment there is no one that consists of a comprehensive visualization of computer data.

Accompanied by my internship tutor, my work consists of creating a data visualization library. Data visualization or also called DataViz for English speakers, it is all graphical representations of statistical data, which allows us a better understanding of these data by a public.

To start I had at my disposal two libraries, the first named Roassal, a library allowing to make graphs, diagrams, forms, drawings etc .., and the second named DataFrame, a library which allows us to store data.

The idea is now to use these two libraries to be able to define my different visuals. And for that, I referred to the Matplotlib library of python, because it is complete and is one of the most used in dataviz domain.

So I first made a part which allows me to extract this data from my dataframe, to put it in another variable, a second part which essentially deals with making my graphs according to extracted or introduced data, then a third part which adds all the little details necessary for understanding the visual. Once finished, I had the entire basis of my project, so we could add as many different graph shapes as we want.

At this stage of the internship, I have not yet finished all the visuals and other arrangements that I have to do. But during the next two months, I intend to reach the end of my objectives.
This internship will have been a great discovery for me in the professional world.

# DPP : Recherche de Stage

J'ai choisi ce DPP, car je le trouvais plus intéressant et consistant que les autres, et parce que j'avais des problèmes dans mes différentes démarches de recherche de stage. Donc, pour pallier ce défaut, j'ai suivi trois ateliers de formations :

- Le premier s'est déroulé le 24 janvier 2022, sur le thème 'Réussir son entretien'. Dans cet atelier, on a mis en exergue tout ce qui ne faut surtout pas faire dans un entretien, les bonnes techniques pour convaincre le recruteur et les différentes questions pièges. Et surtout comment répondre à une question dont on ne connaît pas la réponse sans paraître débile.
- Le deuxième s'est déroulé le 1er février 2022, sur le thème 'Trouver son stage'. Dans cet atelier, on nous a appris comment trouver un stage, notamment sur différentes techniques de recherches. Comment faire pour que ces recherches soient fructueuses. Comment préparer son CV et sa lettre de motivation. Les choses à ne pas mettre dedans. On s'est fait corriger nos CVs.
- Le dernier s'est déroulé le 22 février 2022, sur le thème 'Candidater à une formation'. Cette dernière n'était pas sur la recherche de stage, mais sur la recherche d'un Master. J'ai trouvé cela nécessaire, dans le sens où ça va m'aider dans ma future recherche de Master.

**Obtention de mon stage** :

M. Stéphane Ducasse à proposer des stages aux étudiants qui suivaient l'option MÉTA dont je faisais partie.

Parmi, il y avait un sujet sur la visualisation de données, vu que les données sont un domaine qui m'intéresse, alors j'ai postulé.

Deux semaine après, j'ai reçu un test technique que j'ai passé. Puis une semaine plus tard, j'ai été convoqué à INRIA, j'ai rencontré mon futur tuteur de stage. Et juste quelques jours après, j'ai signé ma convention de stage.

# Sommaire

**Table des matières**

# INTRODUCTION

L'informatique est un domaine qui m'a toujours intéressé. Étant petit, je me posais beaucoup de questions, du genre comment avec une simple application dans un téléphone, on peut contacter une personne à l'autre bout de la planète. Ou comment les personnages sont représentés dans un jeu vidéo. Et beaucoup d'autres questions. Donc, pour moi, le chemin était tout tracé vers l'informatique. Ce chemin a été long, Mais aujourd'hui je suis étudiant en dernière année de licence informatique.

C'est dans cette continuité, que je devais faire un stage de fin de licence , qui s'est déroulé à INRIA à Villeneuve-d'Ascq au sein de l'équipe RMoD. Au sein de cette équipe, composée essentiellement d'ingénieurs, de doctorants et de chercheurs, mon travail consistait à la création d'une bibliothèque de visualisation de données.

Pour cela, j'ai été mis en contact avec un ingénieur de l 'équipe, M. Milton, qui a déjà commencé à travailler sur la base du projet et qui est d'ailleurs mon tuteur de stage. Les deux premières semaines étaient principalement dédie a la compréhension du code déjà présent, que je devais utiliser pour la réalisation de mon projet. Tout le projet a été réalisé sur l'environnement et le langage pharo, développé par l'équipe RMoD.

Pour commencer, ce rapport est divisé en trois principales parties. Dans un premier temps, je vais vous parler de l'entreprise, son histoire et son domaine d'activité. Puis du langage pharo et son évolution . Mais aussi celles de l'équipe avec laquelle je collabore.

Dans un deuxième temps, je vais vous présenter en détails le travail que j'ai réalisé, les points sur lesquels j'ai eu des difficultés, comment je les ai surmonté, ainsi que frameworks sur lesquels je me suis référé et pourquoi .

Enfin, je vous ferais un bilan technique et humain de mon stage, c'est à dire les points sur lesquels je me suis amélioré, ou encore les technos que j'ai utilisé durant ce stage et aussi toute autre activité culturelle et scientifique à laquelle j'ai pu assister. Mais surtout, je vous mettrais en exergue en quoi ma formation m'a été utile dans ce milieu.

# 1- CONTEXTE

## 1.1 INRIA

### 1.1.1 Histoire

**INRIA** est l'acronyme de l'institut national de recherche en informatique et en automatique, donc c'est un établissement public français financé donc par l'état.

Travaillant dans l'informatique et dans les mathématiques, elle est sous la double tutelle du ministère de l'enseignement supérieur, de la Recherche et de l'Innovation, et de ministère de l'Économie et des Finances.

Elle a été créée au début de l'année 1967, dans le cadre du **plan calcul**. Le plan calcul consiste à regroupé les plus hauts potentiels du pays, pour assurer une autonomie dans les techniques de l'information et développer l'informatique européen.

Elle s'est fait connaître à son début sous le non de **IRIA** (Institut de Recherche en Informatique et Automatique), et avait pour but principal la mise en place des systèmes de gestion de basses de données pour les universitaires, administratifs, et autres.

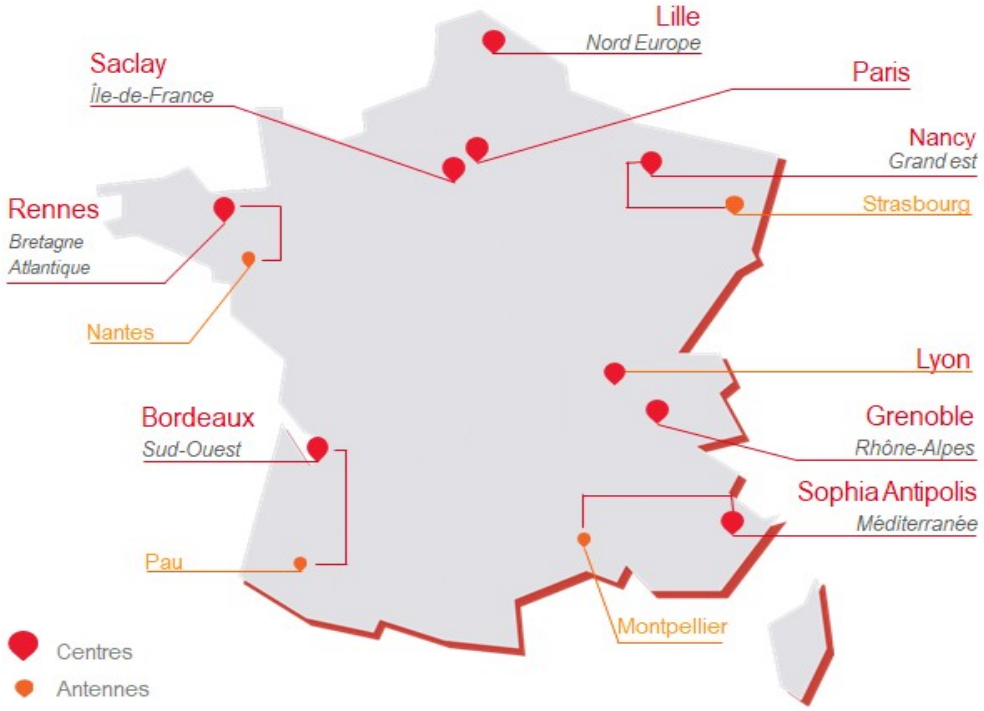
Puis est devenus INRIA en janvier 1980. En 1985, sa mission principale était d'entreprendre des recherches fondamentales et appliquées, de réaliser des développements des systèmes expérimentaux, d'assurer le transfert et la diffusion des connaissances.

Mais en 2018 les choses commencent à changer à l'interne. En effet, en juin 2018 avec son nouveau directeur Bruno Sportisse, INRIA se concentre sur la création des start-ups et s'associer plus avec les entreprises privées dans le but de les aidées à se développer. Ce qui ne manque pas à faire réagir certains chercheurs, en disant qu'il ne faut pas confondre la recherche et l'innovation.

### 1.1.2 Centres de recherches

**INRIA** est présent aujourd'hui dans 9 métropoles françaises. Son premier centre a été créé en 1967 à Rocquencourt.

**Différents Sites de INRIA en France**




Sur cette carte nous avons une idée précise des différents centres de recherche d'INRIA à travers le pays. Moi je fais mon stage dans celui de Lille nord Europe.

### 1.1.3 Partenariats et Relations

**INRIA** et **Microsoft Research** ont inauguré en 2011 un laboratoire de recherche qui se situe à Saclay. Ils sont aussi partenaire d'Hospices civils de Lyon, et fondateur de LIAMA (Laboratoire sino-français de recherche en informatique , automatique et mathématiques). Ils sont aussi en partenariat avec pleins d'universités californiennes comme Stanford, Berkeley et bien d'autres. INRIA participe aussi à l'espace européen de recherche à travers le consortium ERCIM, dont il est membre fondateur.

### 1.1.4 Projets et implications

**INRIA** a œuvré à la réalisation à des nombreux projets scientifiques et techniques, ils ont aussi créé de nombreux langages informatiques comme Coq, OCaml, Pharo. Et aussi contribuer à la réalisation des frameworks utiles comme Scikit-Learn une bibliothèque Python destinée à l'apprentissage automatique, pour ne citer que ceci.

## 1.2 L'Équipe RMoD

### 1.2.1 Présentation

RMoD c'est une équipe de recherche au sein d'INRIA dont le responsable est M. Stéphane DUCASSE. Elle est spécialisé dans  l'analyse et la construction de langage pour l'évolution des applications orientées objets. Elle a créé le langage Pharo, un langage purement objet basé sur le langage Smalltalk.

### 1.2.1 Membres

L'équipe est constitué de doctorants, chacun travaillant sur un sujet bien précis, d'ingénieurs qui s'occupe essentiellement de l'évolution des projets, de chargés de recherches et bien d'autres.

**Liste des Membres de l'Équipe RMoD**

Permanents

- Professeur
  - Anne Etien
- Directeur de recherche
  - Stéphane Ducasse (Responsable)
- Maîtres de conférences
  - Nicolas Anquetil (hdr)
  - Vincent Aranega
- Chargés de recherche
  - Steven Costiou
  - Marcus Denker
- Ingénieurs
  - Christophe Demarey
  - Guillermo Andres Polito
  - Pablo Adrian Tesone

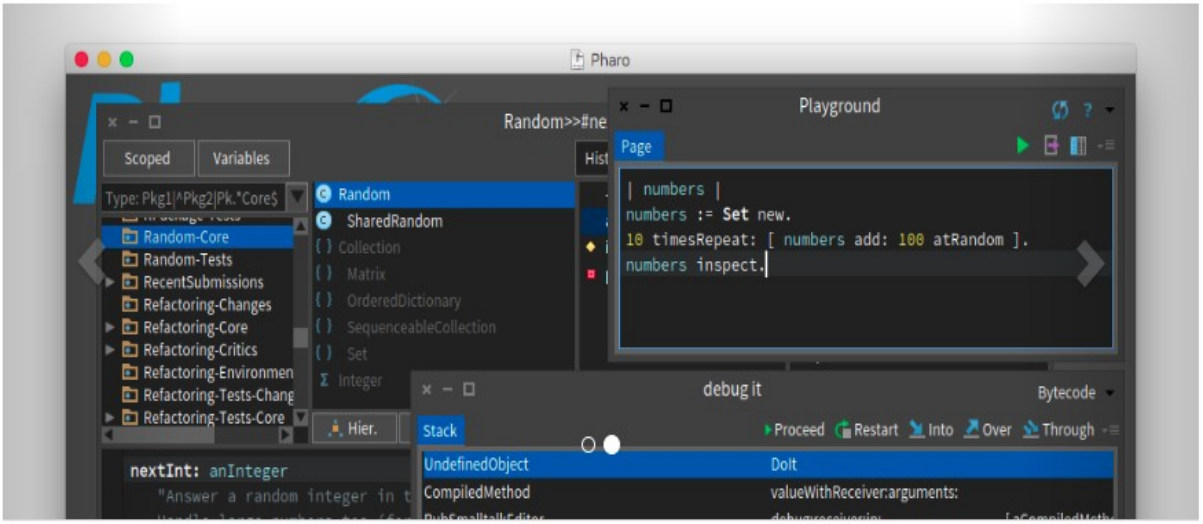
Non permanents

- Doctorants
  - Nour jihene Agouf
  - Santiago Bragagnolo
  - Aless Hosry
  - Pierre Misse-Chanabier
  - Théo Rogliano
  - Maximilian Willembrinck
  - Oleksandr Zaitsev
- Ingénieurs
  - Soufyane Labsari
  - Clotilde Toullec

## 1.3 Pharo

Pharo est un langage informatique largement inspiré de Smalltalk, il est basé sur une machine virtuelle, écrite en large partie en Pharo lui-même. Ce qui fait de lui un langage multi-plateforme. Crée en 2008 par M. Ducasse et Denker. Il est sous licence MIT c'est-à-dire open source, les principales caractéristiques de ce langage sont que tout est objet dans le sens de la programmation orienté objet, toutes les méthodes sont publiques et les attributs sont protégés. le système d'héritage est simple, c'est-à-dire une classe ne peut hériter que d'une seule autre. Il a un système appelé trait, un trait, est un ensemble de méthodes sous forme d'une classe, qui peut être utiliser dans n'importe quelle classe, disons, c'est une sorte de propriété.

**Interface PHARO**




## 1.4 Le Projet

L'équipe RMoD travaille avec beaucoup d'entreprises privées, dont chacune à un problème spécifique, donc le but, c'est de réaliser des bibliothèques plus en plus diversifiées, des bibliothèques de gestions de tests à des bibliothèques apprentissage automatique. Mais ils n'ont forcément pas le temps à consacrer au développement de toutes ces nouvelles bibliothèques, car il y a beaucoup d'autres projets qu'ils doivent finir. D'où ma présence dans cette équipe. Mon travail, consiste à réaliser une bibliothèque de visualisation de données. Je ne suis pas le seul stagiaire dans l'équipe, il y en a d'autres, et chacun doit faire évoluer une bibliothèque ou en réaliser une.

# 2 – CONTRIBUTION

## 2.1 Formation

À mon arrivé, j'ai été informé du fonctionnement de l'équipe, notamment les rapports hebdomadaires qu'il fallait écrire sous forme de message et à envoyé à toute l'équipe. Ce petit rapport comporte l'ensemble des activités effectué de la semaine d'avant et celles planifiées la semaine .

Avant de débuter le projet, je devais obligatoirement lire un livre ''Pharo With Style'', ce livre écrit par M. Stéphane qui explique comment bien coder en pharo, quelles sont les règles à respecter, notamment sur le nommage des objets, des méthodes et surtout les principes de bases de tout langage orienté objet. Et aussi, je devais comprendre les deux bibliothèques, que je devais utilisés pour mon projet, à savoir Roassal ( Bibliothèque permettant de faire des graphes, schémas, formes, dessins etc .. ) et DataFrame (Bibliothèque qui a pour but de stocker des données ) .

Pour cela M. Stéphane Ducasse a mis en ma disposition un livre de 300 pages, ''AGILE VISUALIZATION WITH PHARO''. Ce livre est en quelque sorte un condensé de tout ce que je devais savoir sur Roassal, notamment sur le package Roassal-Chart , qui est la partie qui m'intéresse, car c'est particulièrement celle que je vais utiliser dans la suite pour faire mes classes de graphes. Cela m'a pris en tout deux semaines.

Je devais aussi étudier la bibliothèque Matplotlib de Python, une bibliothèque de visualisation de données très célèbre, il fallait donc que j'apprenne les bibliothèques pandas et numpy, car c'est d'elles qu'on se sert pour faire les visuels en fonction des données. Pandas et Numpy sont deux bibliothèques python qui permettent de stocké des données, son équivalent sur Pharo est la bibliothèque DataFrame. Après tout, j'avais toutes les informations nécessaires pour commencer le projet en question.

## 2.2 Roassal

Roassal est composé essentiellement de 5 packages sur lesquels tout le reste est basé.
Parmi on a :

- **Shapes :** ce package regroupe toutes les formes les figures qu'on peut avoir (Ligne, Box , Triangle, Cercle etc.. ). Voilà en quelque sorte les figures que je vais utiliser en faire mes différents plots.

- **Colors : c**omme son nom l'indique, ce dernier s'occupe essentiellement de la création des différentes palettes de couleur, qu'on utilisera pour colorer nos plots.
- **Layouts :** il a pour but de gérer les dispositions de nos figures. Disposition à gauche , à droite , en haut , en bas etc …
- **Interaction :** Il s'occupe de l'ensemble des interactions qu'on peut avoir sur les figures, par exemple on peut faire de sorte qu'un clique sur une figure puisse afficher quelque chose. Ou encore qu'on puisse déplacer une figure et pleins d'autres interactions qu'on peut avoir.
- **Animation : c**e package ne nous intéresse pas tellement dans le cadre de notre projet, mais il peut être utile dans le cas où on voudrait ajouter des animations sur les plots.
- **Chart :** ce dernier n'est pas un package mais plus tôt l'objet qui sert de support à tout ce qu'on va faire. C'est ce dernier qui nous permet d'afficher les figures sur l'écran et plein d'autre détails, comme gérer les animations.

## 2.3 DataFrame

DataFrame est une bibliothèque de manipulation et stockage de données sur Pharo. Dans ce projet, elle va essentiellement servir à stocker les données et aussi de support d'extraction.

On peut introduire les données de deux façons différentes. La première est de le faire manuellement, c'est-à-dire tu écris toi-même avec les méthodes faites pour.

**Exemple d'introduction manuelle**

```
df := DataFrame
      withCollumns:
         { { 'john'. 'mary'. 'peter'. 'jeff'. 'bill'.
'lisa'. 'jose' }.
         #(23 78 22 19 45 33 20).
         { 'M'. 'F'. 'M'. 'M'. 'M'. 'F'. 'M' }.
         { 'california'. 'dc'. 'california'. 'dc'.
'california'. 'texas'. 'texas' }.
         #(2 0 0 3 2 1 4).
         #(5 1 0 5 2 2 3) }
      columnNames: #(name age gender state num_children
num_pets)
```

la méthode **withcolumns** définis les différentes colonnes et **columnNames** définis les noms de ces colonnes, pour nommer les lignes on utilise le méthode **rowNames**, mais cette dernière n'est pas très utile, car les lignes sont numérotées automatiquement. On voit le résultat ci-dessous.

**Résultat**

DataFrame | Raw | Breakpoints | Meta

| | name | age | gender | state | num_children | num_pets |
|---|---|---|---|---|---|---|
| 1 | john | 23 | M | california | 2 | 5 |
| 2 | mary | 78 | F | dc | 0 | 1 |
| 3 | peter | 22 | M | california | 0 | 0 |
| 4 | jeff | 19 | M | dc | 3 | 5 |
| 5 | bill | 45 | M | california | 2 | 2 |
| 6 | lisa | 33 | F | texas | 1 | 2 |
| 7 | jose | 20 | M | texas | 4 | 3 |

La deuxième manière d'introduire les données, qui est d'ailleurs la manière la plus utile. Et cela ce fait à partir d'un fichier csv ou texte

**Exemple d'introduction de données à partir d'un fichier csv**

```
file := '/home/users/etudiant/Téléchargements/iris.csv'
asFileReference.
data := DataFrame readFromCsv: file.
```

Comme vous voyez sur l'image ci-dessus, on importe le fichier à partir d'un chemin relatif, et après, on le passe à notre DataFrame. Voir le résultat ci-dessus

**Résultat**

DataFrame | Raw | Breakpoints | Meta

| # | | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|---|---|
| 47 | 47 | 5.1 | 3.8 | 1.6 | 0.2 | setosa |
| 48 | 48 | 4.6 | 3.2 | 1.4 | 0.2 | setosa |
| 49 | 49 | 5.3 | 3.7 | 1.5 | 0.2 | setosa |
| 50 | 50 | 5.0 | 3.3 | 1.4 | 0.2 | setosa |
| 51 | 51 | 7.0 | 3.2 | 4.7 | 1.4 | versicolor |
| 52 | 52 | 6.4 | 3.2 | 4.5 | 1.5 | versicolor |
| 53 | 53 | 6.9 | 3.1 | 4.9 | 1.5 | versicolor |
| 54 | 54 | 5.5 | 2.3 | 4.0 | 1.3 | versicolor |
| 55 | 55 | 6.5 | 2.8 | 4.6 | 1.5 | versicolor |
| 56 | 56 | 5.7 | 2.8 | 4.5 | 1.3 | versicolor |
| 57 | 57 | 6.3 | 3.3 | 4.7 | 1.6 | versicolor |
| 58 | 58 | 4.9 | 2.4 | 3.3 | 1.0 | versicolor |

## 2.4 Cahier de charge

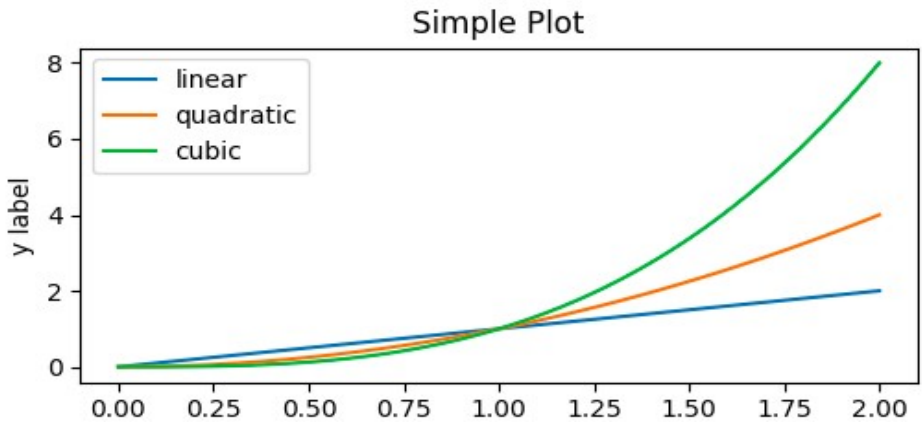
La question qu'on se pose quand on commence ce genre de projet est : Par où commencer ?
Pour répondre à cette question, on va se poser une autre, d'autres personnes ont déjà travaillé sur un projet ressemblant au nôtre ?. Et la réponse est oui, donc on va aller voir comment ce dernier a été réalisé, pour cela j'ai choisis l'une plus utilisée, Matplotlib de python. Ce qui est bien avec cette bibliothèque, ce qu'elle correspond exactement à celle qu'on veut implémenter sur pharo. Donc voyons comment elle fonctionne.

Tout d'abord prenons le plus basique le LinePlot.

**Exemples : LinePlot Matplotlib**

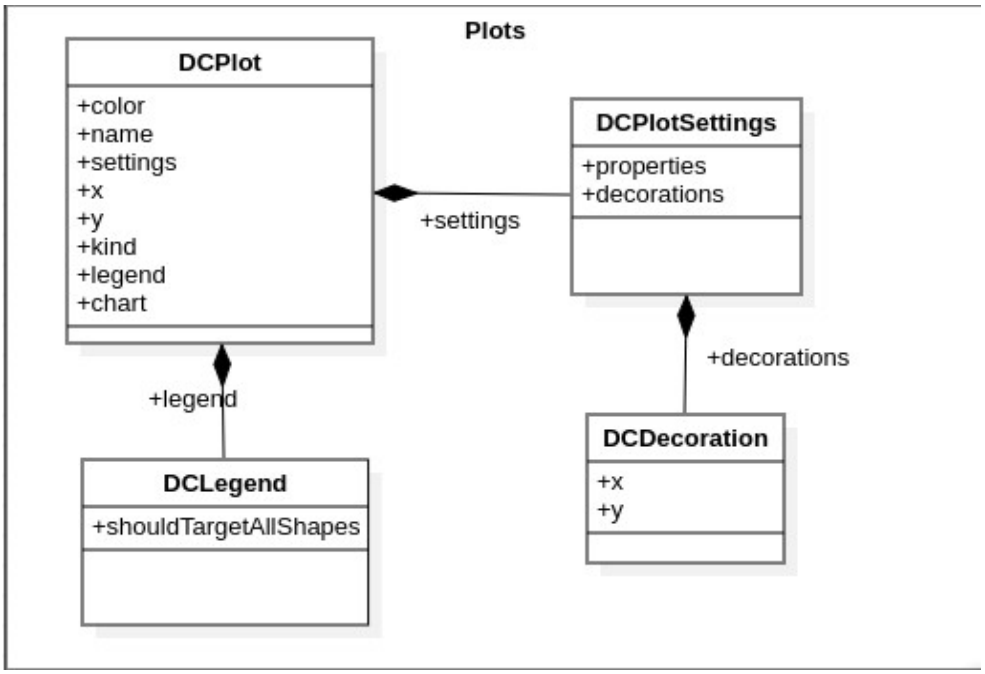
```
x = np.linspace(0, 2, 100)  # Sample data.

# Note that even in the OO-style, we use `.pyplot.figure` to create the Figure.
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend();  # Add a legend.
```




On remarque que dans la configuration de ces différents plots, ils ont des choses en commun, notamment chaque plot peut avoir sa couleur, ses données sur les axes, son label, et son type. Par contre le titre, les labels, la légende sont plus tôt des caractéristiques du chart. Donc l'idée est de créer des objets, qui vont avoir chacun un seul objectif. Mais tout en regroupant tout ce qui est point commun entre eux dans une classe. Et en découpant correctement le code.

## Squelette du projet




Comme indiqué sur ce petit diagramme, nous avons quatre classes, qui sont les bases de tout ce qu'on va définir dans le futur. J'ai utilisé essentiellement le principe de composition, on regroupe tout dans la classe DCPlot représentant nos différents plots. Mais chaque attribut est géré par une autre classe. Tout ce qu'on a faire, c'est de mettre nos paramètre, avec les méthodes dédiées, et chaque classe fabrique son objet et le remet dans le chart de la classe en question. Voyez la classe DCPlot comme un centre de contrôle qui distribue les tâches et attend les résultats. Dans la classe DCPlotSettings, nous avons deux attributs **properties** et **decorations**, tous deux des dictionnaires.

Le premier contient toutes nos propriétés comme les différents labels, le titre ou encore le background, et le second contient les petites décorations qu'on aimerait ajouter à nos visuels. Ces décorations ont pour but principale de permettre à l'utilisateur d'ajouter des petites précisions à son chart. Comme par exemple un point sur le chart , sur lequel il aimerait bien accentuer.

Et ces décorations sont à leur tour gérée par une autre classe DCDecoration, se trouvant dans un autre paquet et qui est la classe mère de toutes les décorations. Puis on a la légende qui est fabriquée par la classe DCLegend. Le reste des attributs comme kind, chart, color, name  sont des objets Roassal, donc on ne s'en occupe pas trop, car Roassal sait bien comment le faire.

Maintenant, la base est définie, le second problème est de savoir comment introduire les données, car ces plots ont une signification. Et avant toute visualisation, il est indispensable de se poser des questions suivantes :

- *Quels types de données vont être représentés ?*
- *Dans quels buts ?*
- *À qui les visualisations vont être représentées ?*

Autrement dit qu'est-ce qu'on veut étudier et en fonction ,on choisit notre visuel. Par exemple, on utilise un LinePlot pour étudier ou observer l'évolution d'un produit sur un marché, en fonction d'un paramètre qui peut être le temps ou le lieu. Par contre si on veut observer la distribution de nos données ou leur répartition, on utilise un HistogrammPlot, ou un PiePlot, donc ce sont deux visualisations complètement différentes, d'où la nécessité de bien les séparer. En résumé, les visuels doivent refléter un objectif précis. Mais dans ce rapport, je ne vais pas me concentrer sur la signification des résultats, je vais plus tôt vous montrer les solutions que j'ai mises en place pour avoir les différents visuels, tout en respectant bien les propriétés de chaque plot.

J'ouvre ici une petite parenthèse. Avant d'arrivé à cette conception, j'avais tout d'abord essayé de mettre mes différents visuels directement dans la bibliothèque DataFrame, comme c'est cette dernière qui contient les données. Mais ce n'était pas la meilleure façon de faire, car il est fort probable que cette bibliothèque évolue ou change complètement. Dans ce cas, on aurait perdu tout ce qui avais fais. La deuxième idée étaits de rajouter les visuels directement dans Roassal, mais Roassal contient déjà beaucoup trop d'objets, en rajouter n'est pas une bonne idée. Alors j'ai décidé de créer un nouveau projet à part que j'ai nommé DataChart. C'est de là vient le début nominal DC de mes différents objets. En effet, c'est une façon très spéciale de nommer les objets sur pharo, une façon de dire que cet objet appartient au package DataChart, c'est comme l'objet RSChart qui est de la bibliothèque Roassal. Ceci fait, j'ai donc crée une première classe qui regroupe tout, mais je me suis rendu compte que cela allait être lourd pour ne seule classe, par ce que plus j'avance, plus j'aurais des choses à ajouter. Et voilà comment je suis arrivé à cette conception décrite ci-dessus.

Ce projet étant essentiellement de la visualisation de données, je vais donc mettre un résultat pour chaque partie que je vais traiter, pas plus, sinon le rapport risque de contenir que des images, car il y a beaucoup de plots.

Maintenant, parlons de ces plots.

## 2.5 Plots

Avant d'introduire directement nos différents plots. Nous allons decrire cette classe mère DCPlot. DCPlot en lui est une sorte de classe abstraite en Java qui définit tout ce qui est commun entre les différentes classes. La seule chose qui les diffère, c'est le type de plot, qui est référencé par l'attribut kind. Donc, au moment de l'initialisation, chaque plot doit générer son propre type.

```
initialize
    super initialize.
    settings := DCPlotSettings new.
    chart := RSChart new.
    kind := self defaultKind.
    legend := DCLegend new.
```

Comme vous le constatez, on a un message **defaultKind**. Voyons comment il est défini.

```
defaultKind
    ^ self subclassResponsibility
```

Il retourne ce qu'on appelle une responsabilité aux sous-classes. C'est une manière de faire spécifique à pharo. Et cela veut dire tout simplement que chaque classe qui hérite de la classe mère doit obligatoirement l'implémenter. À titre d'exemple pour le LinePlot, il retourne l'objet RSLinePlot qui représente une ligne.

```
defaultKind
    ^ RSLinePlot new
```

On a aussi l'objet RSChart, ce dernier contient le Canvas comme expliqué en haut, il est indispensable dans le sens ou il sert de support à nos différents objets qu'on veut visualiser, et aussi, l'objet DCLegend représente juste la légende, dont on va parler dans la suite.

## 2.5 .1 Extraction de données

Maintenant, qu'on a l'objet DCPlot, comment on va extraire des données du DataFrame et l'introduire dans DCPlot. Après essai de plusieurs maquettes de modèles, j'ai gardé deux modèles que je trouve plus utiles et essentiels.

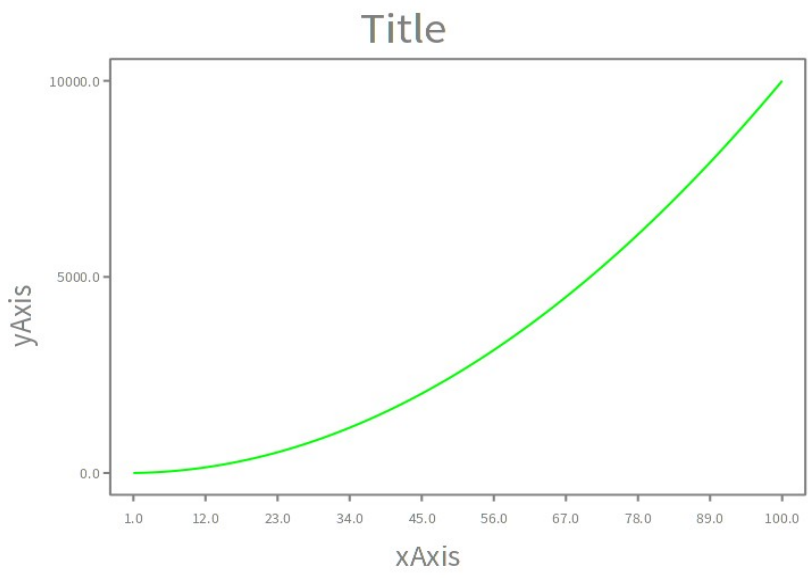
### 2.5.1.1 Premier Modèle

Cette façon de faire permet à l'utilisateur de générer ou d'extraire lui-même les données avant de les introduire. Sur l'exemple ci-dessus, j'ai généré des entiers allant de 1 à 100 sur l'axes des x, et j'ai mis le carré de ces nombres sur l'axe des y.  Je vais donc avoir le visuel d'une fonction carrée $y = x^2$. On va  potentiellement voir une petite courbe linéaire, vu qu'on a utilisé un LinePlot.

**Exemple 1 LinePlot**

```
exampleDCLine
   <script: 'self new exampleDCLine'>
   | x |
   x := 1 to: 100.
   ^ DCLinePlot new
      x: x;
      y: (x raisedTo: 2);
      color: Color green;
      build;
      show.
```

**Résultat Exemple 1 LinePlot**



#### 2.5.1.1 Deuxième Modèle

Ce modèle ressemble beaucoup à la façon dont pandas se sert de Matplotlib pour faire de la visualisation. Donc, en premier lieu, on prépare notre DataFrame, en introduisant les données. Puis après, on fais juste appel au plot dont on a besoin, pour faire la visualisation.

**Exemple 2 scatterPlot**

```
| file data x y xvalues yValues yvalues |
file := '/home/users/etudiant/Téléchargements/iris.csv' asFileReference.
data := DataFrame readFromCsv: file.
data scatterPlot
   xColumnName: 'Sepal.Length';
   yColumnName: 'Sepal.Width';
   build;
   show.
```

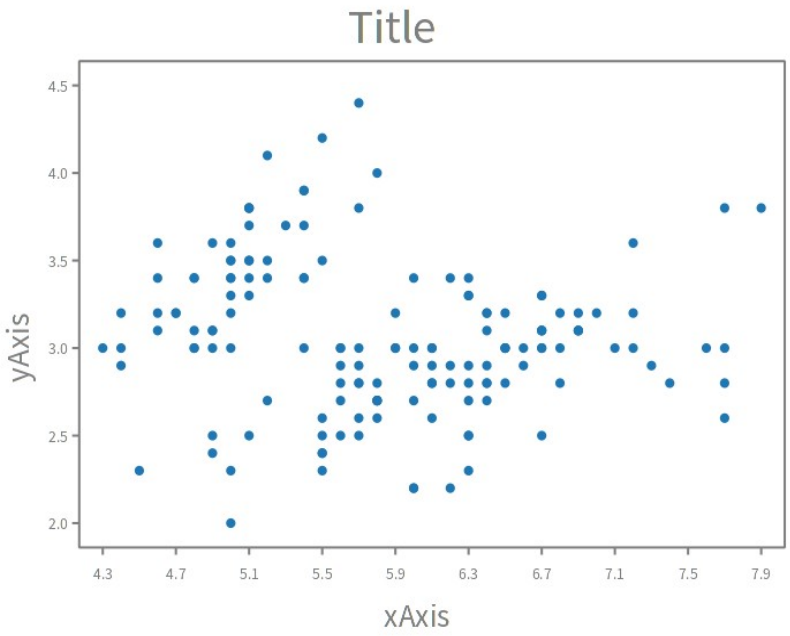
Dans cet exemple, nous avons directement envoyé le message scatterPlot au DataFrame, qui veut dire qu'on veut un plot de type ScatterPlot, et après nous avons donné le nom des colonnes qu'on veut visualiser. C'est une manière de faire ultra simple et instinctif. Et juste après, on a les message **build** qui fais en gros la construction de notre visuel, et **show** qui comme son nom l'indique montre tout simplement le visuel. En effet, j'ai gardé les mêmes messages pour les deux cas d'utilisation, sans pour autant faire une duplication de code. J'ai juste fais une extension, c'est-à-dire, j'ai utilisé la technique du dispatching, qui consiste à la fois à séparer la partie extraction à la partie construction. J'ai donc juste ajouté le message **scatterPlot** à l'objet DataFrame.

```
scatterPlot
   ^ DCScatterPlot new
      dataFrame: self;
      yourself
```

Comme vous le constatez sur l'image précédente, j'ai juste retourné au final un objet DCScatterPlot provenant de ma bibliothèque et j'ai dit à cet objet en question que son dataframe est le **self**, donc l'objet qui a reçu le message, c'est-à-dire le DataFrame dans lequel sont stockées

les données. Comme ça, rien ne change de l'autre côté, on a juste à ajouter une méthode qui extrait ces données.

**Résultat d'Exemple 2 scatterPlot**




J'ai mis par défaut pour chaque plot comme titre :'title', xlabel : 'xAxix', ylable : 'yAxix'. Maintenant, voyons comment on peut les ajouter au chart.

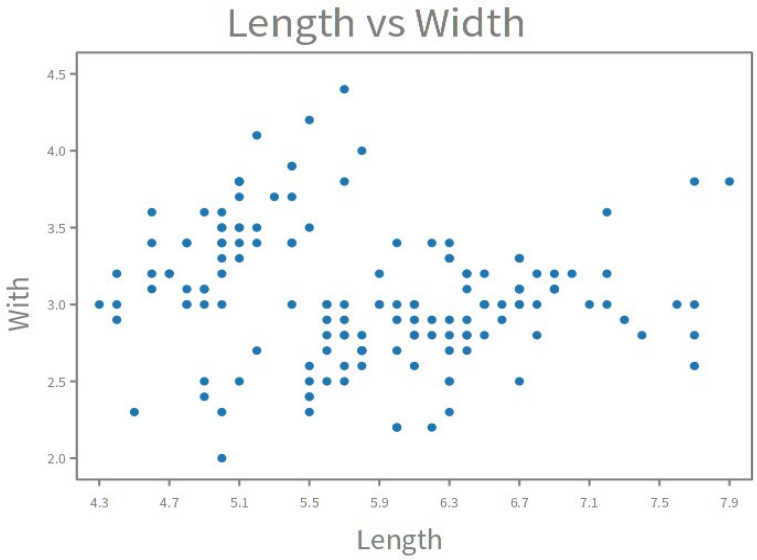
## 2.5.2 Ajout De Propriétés : Titre , xlabel, ylabel, background etc...

Reprenons les mêmes exemples et ajoutons y un titre, un label pour chaque axe.

**Exemple scatterPlot avec Les labels**

```
| file data x y xvalues yValues yvalues |
file := '/home/users/etudiant/Téléchargements/iris.csv' asFileReference.
data := DataFrame readFromCsv: file.
data scatterPlot
   xColumnName: 'Sepal.Length';
   yColumnName: 'Sepal.Width';
   title: 'Length vs Width';
   xlabel: 'Length';
   ylabel: 'With';
   build;
   show.
```

**Résultat  Exemple scatterPlot avec Les labels**




Comme expliquer au paravent, cette partie est gérée par la classe DCPlotSettings lors de la construction du visuel. Pour l'instant, je ne me suis pas préoccupé du background.

## 2.5.3 Changer L'étendue du chart

Il pourrait arriver dans un cas qu'on veuille changer la taille de nos différents axes, soit en rapetissant ou en augmentant, dans l'objectif de mieux visualiser les résultats, ou juste pour une question d'ergonomie. Pour cela, il suffit de le faire avec le message **extent,** auquel vous passez en paramètre un point. Un point c'est un type d'objet sur pharo représentant les cordonnées sur les deux axes x et y. Exemple le point ( 12@16) (un point ayant pour abscisse 12 et ordonné 16). Je vais donc vous montrer deux exemples, où je vais augmenter et diminuer, les axes de mon chart.  Il faut savoir que par défaut le chart à un extent de 200@300

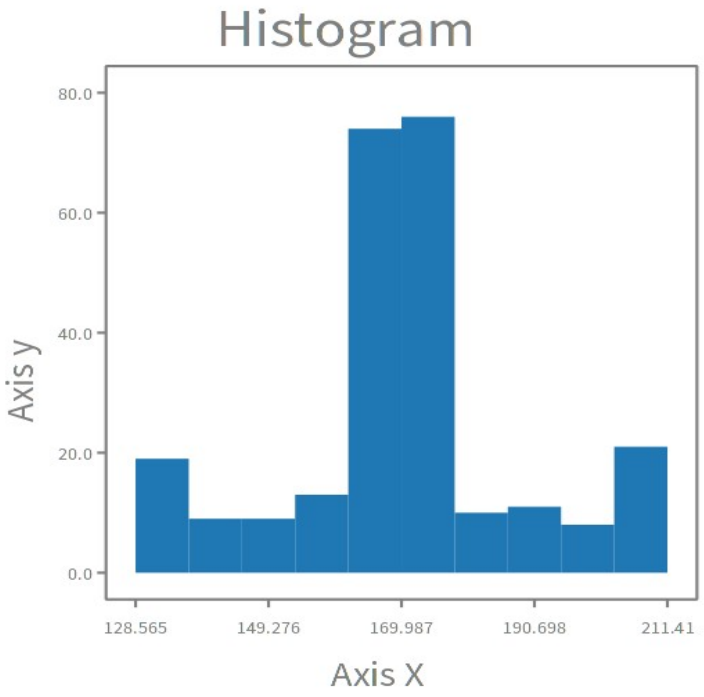
**Exemple diminution chart**

```
<script: 'self new exampleDCHistogram'>
| x |
x :=  self randomValues.
^ DCHistogramPlot new
   x: x;
   title: 'Histogram';
   xlabel: 'Axis X';
   ylabel: 'Axis y';
   extent:200@200;
   build;
   show
```

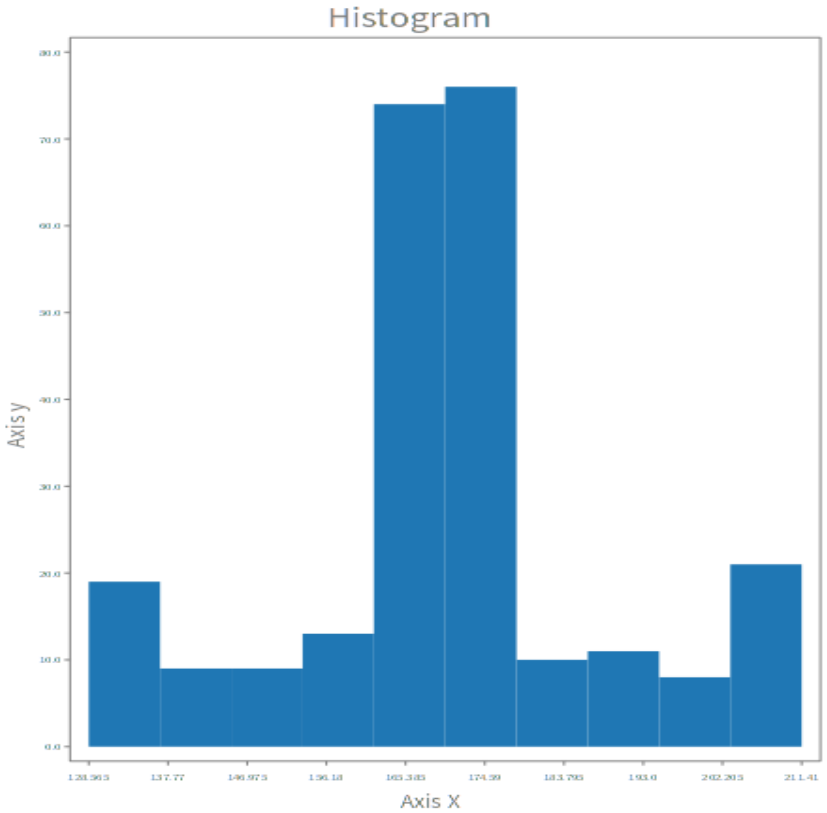
**Exemple augmentation chart**

```
<script: 'self new exampleDCHistogram'>
| x |
x :=  self randomValues.
^ DCHistogramPlot new
   x: x;
   title: 'Histogram';
   xlabel: 'Axis X';
   ylabel: 'Axis y';
   extent:400@500;
   build;
   show
```

**Résultat Exemple axes diminuer**




**Résultat Exemple axes augmenter**



## 2.5.4 Ajout de Légende

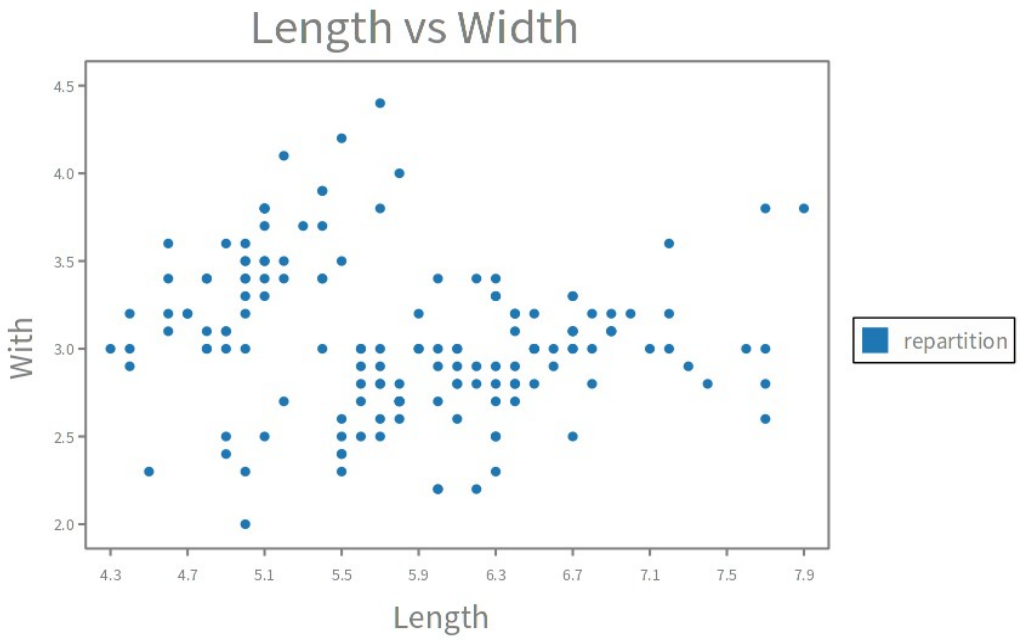
La Légende est très importante pour une bonne compréhension des visuels. Et pour son ajout, on a besoin de savoir le nom qu'on va donner au plot. Reprenons l 'exemple ci-dessus, et ajoutons une légende.

**Exemple 3 Avec Légende**

```
| file data x y xvalues yValues yvalues |
file := '/home/users/etudiant/Téléchargements/iris.csv' asFileReference.
data := DataFrame readFromCsv: file.
data scatterPlot
   xColumnName: 'Sepal.Length';
   yColumnName: 'Sepal.Width';
   title: 'Length vs Width';
   xlabel: 'Length';
   ylabel: 'With';
   name: 'repartition';
   withLegend;
   build;
   show.
```

Comme vous le constatez, avec une simple méthode **withLegend** précédée de la méthode **name** qui permet de donner un nom au plot, on peut ajouter la légende. Voyons le résultat

**Résultat Exemple 3 Avec Légende**



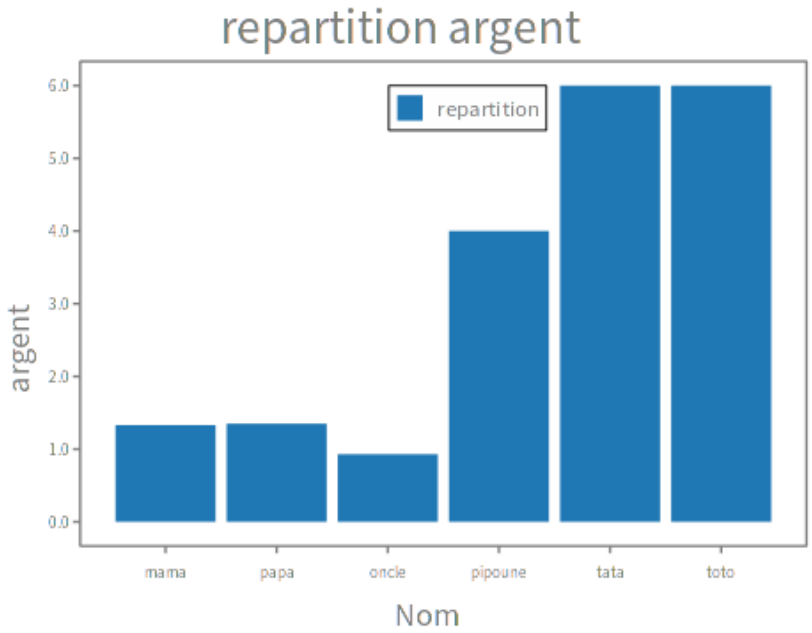
Par défaut, j'ai mis la légende à droite hors du chart, mais si on souhaite changer sa forme ou encore sa position, j'ai fais en sorte que ça soit possible. Voir l'exemple ci-dessus avec barPlot

**Exemple 4 Modification de la Légende**

```
| x plot bars window |
x := #(1.33 1.35 0.93 4 6 6).
plot := DCBarPlot new
      x: #('mama' 'papa' 'oncle' 'pipoune' 'tata' 'toto');
      y: x;
      title: 'repartition argent';
      xlabel: ' Nom ';
      ylabel: 'argent ';
      name: 'repartition';
      legendDo: [ : legend|
         legend location inner top offset: 10];
      yourself.
plot build.
window := plot show.
window.c
```

Avec le message **legendDo** on peut changer tout ce qu'on veut sur la légende, la position avec l'attribut **location** suivit de **inner,** qui permet de le placer à l'intérieur de notre chart. On a aussi **offset** pour changer le décalage, on peut vraiment changer énormément de choses. La légende est gérée à part, par la classe DCLegend

**Résultat Exemple 4 Modification de la Légende**



## 2.5.6 Ajout De Décorations

Imaginons qu'on a envie de préciser un point important sur notre visuel avec un texte pour montre à quoi il sert. Dans ce cas, on utilise une décoration faite pour, donc en résumé les décorations servent de précisions, ce qui en fait une partie très importante du projet. Mais à ce stade du projet, je n'ai pu créer que deux.

### 2.5.6.1 Le Label Décoration

Le label décoration géré par la classe **DCLabelDecoration,** permet essentiellement de faire des ajouts de précisions textuelles explicatives sur un point précis du chart. Voir l'exemple ci-dessus.

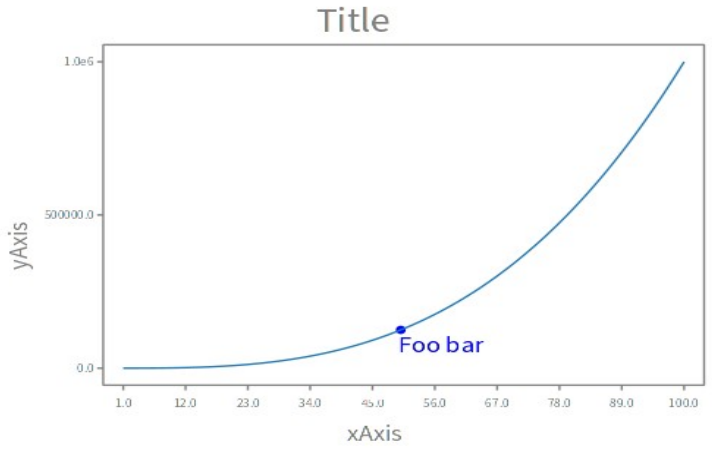
**Exemple 5 Ajout de décoration**

```
| x y decoration |
x := 1 to: 100.
y := x raisedTo: 3.
decoration := (DCLabelDecoration new
   text: 'Foo bar';
   x: 50;
   y: (50 raisedTo: 3);
 yourself ).

^ DCLinePlot new
   x: x;
   y: y;
   add: decoration;
   build;
   show.
```

Dans cet exemple à droite, j'ai créé une décoration nommée 'foo bar' et je l'ai placé aux cordonnées x=50 et y= $50^3$ , puis je l'ai ajouté à mon plot avec la méthode **add.** Comme vous le constatez, la décoration est créée à part et ajoutée tout simplement après. J'ai décider de faire comme ça, car c'est moins encombrant et plus esthétique.

**Résultat Exemple 4 Ajout de décoration**



### 2.5.6.2 Le vertical Line Décoration

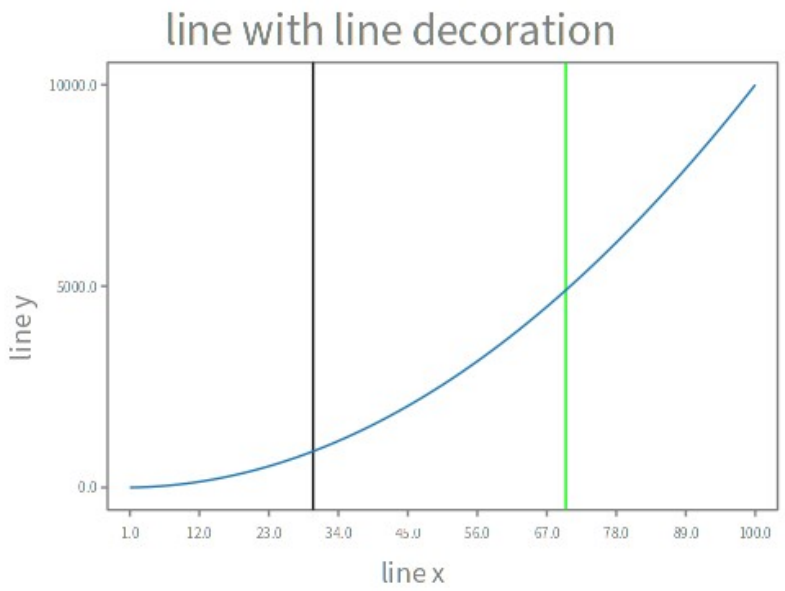
Ce dernier est géré par la classe **DCVerticalLineDecoration** et peut être utilisé faire beaucoup de chose, par exemple si on veut montrer la médiane dans un HistogrammPlot ou encore délimiter par deux lignes deux points de ton graphe. Bref, prenez cette décoration comme une ligne verticale qu'on peut ajouter à tout ce qui nous semble utile et persistant.

**Exemple Ajout vertical ligne décoration**

```
| x decoration1 decoration2 |
decoration1 := DCVerticalLineDecoration new x: 70;
      color: Color green;
      yourself.
decoration2 := DCVerticalLineDecoration new x:30;
      color: Color black;
      yourself.
x := 1 to: 100.
^ DCLinePlot new
   x: x;
   y: (x raisedTo: 2);
   title: 'line with line decoration';
   xlabel: 'line x';
   ylabel: 'line y';
   add: decoration1;
   add: decoration2;
   build;
   show
```

Dans cet exemple j'ajoute deux lignes verticales, la première à l'abscisse 70 avec une couleur verte et le seconde à l'abscisse 30 avec une couleur noire. Voir le résultat ci-dessus

**Résultat Exemple Ajout vertical ligne décoration**



#### 2.5.6.3 La suite sur Les décorations

Dans la suite, on aimerait avoir des décorations spécifiques pour chaque plots, par exemple un label au dessus des BarPlots qui pourrait indiquer leur valeur, ou encore des sortes de traits entre eux qui pourrais indiquer leur différence. Bref, on pourrait en créer autant qu'on veut.

Maintenant, je vais vous parler d'un type plots très important, que j'ai nommé les combinedPlots.

### 2.5.7 combinedPlots

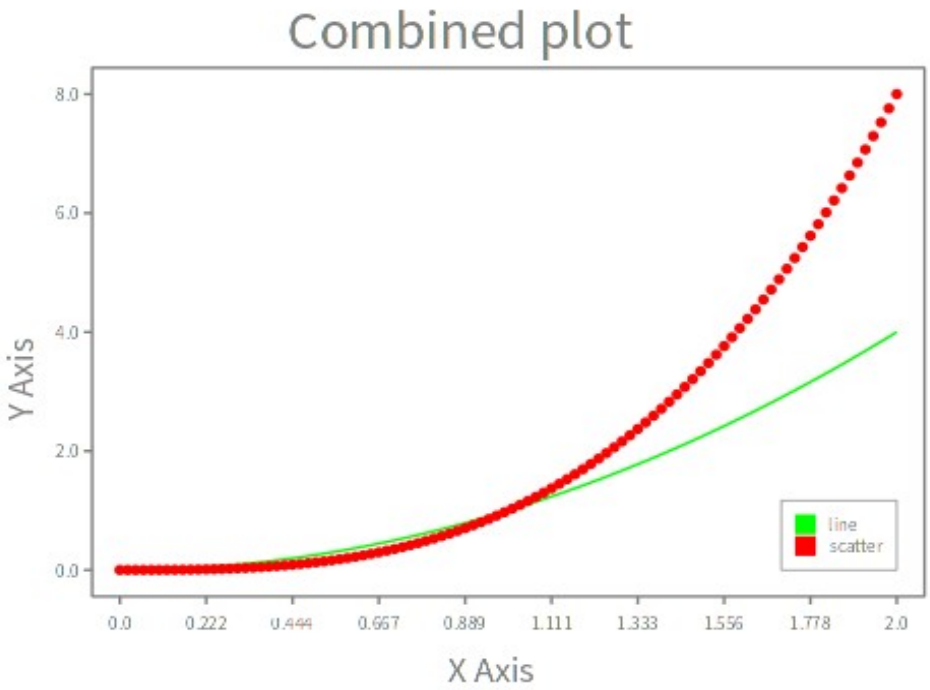
Jusqu'ici, on a vu la représentation d'un plot dans le chart. Les combinedPlots sont un peu là, pour permettre de mettre à la fois plusieurs plots dans un même chart qu'il soit différent ou pas.
Pour cela j'ai décidé de faire une conception un peu différente de celle de Matplotlib. Dans mon cas, on prépare tous mes plots qu'on veut visualiser et en suite, on les ajouter tous dans l'objet combinedPlots. Et ce dernier s'occupera de les réarranger. La classe combinedPlots hérite de la classe DCPlot, donc toutes les propriétés expliquées précédemment sont aussi applicable à ce dernier. C'est un composite de la classe DCPlot dans le sens où il peut contenir tous les objet du même type que lui. Voyons un exemple de la combinaison d'un scatterPlot , et  d'un LinePlot.

**Exemple combinedPlot**

```
| x line1 line2 line3 combined |
x := 0.0 to: 2 count: 100.
line1 := DCLinePlot new
   x: x;
   y: (x raisedTo: 2);
   name: 'line';
   color: 'green';
   yourself.
line2 := DCScatterPlot new
   x: x;
   y: (x raisedTo: 3);
   name: 'scatter';
   color: Color red;
   yourself.
combined := DCCombinePlot new withAll: { line1. line2.}.
^ combined
   title: 'Combined plot';
   xlabel: 'X Axis';
   ylabel: 'Y Axis';
   legendDo: [ : legend|
      legend location inner bottom right offset: -10];
   build;
   show.
```

Dans cet exemple comme expliquer ci-dessus, on a nos deux plots (scatterPlot et LinePlot), tous les deux ajoutés dans un combinedPlot avec le message **withAll**

**Résultat Exemple combinedPlot**




Mais comme vous vous doutez bien, il y a des types des plots qu'on ne peut pas juste s'amuser à ajouter, car ils ne répondent pas au même critère. Pour tous ces types de plots, j'ai créé des classes spécifiques et parmi nous avons clusterBarPlots et stackeBarPlots

- **clusterBarPlots** : ces types de plots nous permettent de visualiser plusieurs BarPlots les uns à coté des autres dans le but de comparer leur donnée ou bien d'autre choses
- **stackeBarPlots** : c'est la même idée que les clusterBarPlots mais à différence, les BarPlots sont les uns sur les autres.

Je vous mettrais un exemple de ces deux visuels dans l'annexe. Cela me permet d'économiser d'espace.

Et comme leur nom indique, ils ne sont composés que de BarPlots, autrement dit, on ne peut y ajouter d'autres types. Et pour l'instant je n'ai que ces types de combinedPlot spécifiques.
Dans la suite, j'aimerais aussi ajouter les **histogrammLinePlots (**une combinaison entre un ligne et un histogramme**)**, **scatterBarPlots** ( combinaison entre des scatterPlot et des BarPlots qui pourrais bien être utile) et il pourrait avoir plein d'autres combinaison des plots différents.

## 2.6 Suite du Stage

Pendant que j'écrivais ce rapport, je travaillais en parallèle sur mon projet, et mon stage devait finir dans 1 mois et demi, je vais donc ajouter certaines choses dont je n'aurais pas parlé dans ce rapport. Dans ce petit chapitre, je vais vous mettre en exergue tout ce que je souhaiterai ajouter. Tout d'abord à ce stade, j'ai pas encore ajouter les PiePlots, les kdePlots, les kiviatPlots, et plein d'autres. Les prochaines semaines, mon travail consistera à faire cela.

Je vais ensuite ajouter certaines décorations que je jugerais nécessaires au fur à mesure, je vais aussi travailler sur une manière simple de changer le font, la forme voir la couleur du chart, on pourrait avoir un chart sous d'un losange ou d'un cercle au lieu d'une boxe. Voir même le faire disparaître et juste laisser les plots. Après avoir fais suffisamment de Plots, je vais travailler sur leur disposition, c'est-à-dire que, je vais faire en sorte qu'on puisse avoir plusieurs charts côte à côte, un peu comme une sorte de tableau de chart. Bref, je pourrais vraiment ajouter beaucoup d'autres choses.

Si le temps le permet , je travaillerai aussi sur animations, en effet s'il est possible d'animer toutes figures que nous utilisons, cela pourrais être utile dans le cas on voudrait voir un avant et après de deux visualisations, pour mieux cerner l'évolution des données.

# 3- CONCLUSION

L'objet général de ce stage était de faire une bibliothèque complète de visualisation de données. J'ai réussi à faire une très bonne basse du projet, même si j'arrive pas à le finir à terme de ce stage, avec l'idée que j'ai introduit n'importe qui, avec un minimum d'informations pourrais continuer à le développer.

Durant ce stage, j'ai beaucoup appris, notamment sur la programmation orientée objet. En effet pharo étant un langage objet, j'ai appris beaucoup d'autres aspects de la programmation orienté, comme le dispatching, ou encore la notion des visiteurs.

La manière de structurer les méthodes dans un projet, en faisant de sorte que chaque méthode s'occupe d'une chose à la fois. J'ai pu aussi développer mes compétences en python, par le fait que j'ai du apprendre l'intégralité des outils de visualisation de données de ce langage, entre autre, Pandas, Numpy, Matplotlib et bien d'autre .

Je me suis exercé à la notion du pair Programming, avec mon tuteur de stage. J'ai aussi appris à prendre en main l'outil github, et TDD (Test Driven Developement). En effet, tout le projet s'est déroulé dans le respect de ce principe. De tout cela, la compétence que j'ai le plus développé est l'autonomie, car j'étais seul sur ce projet, juste accompagné de mon tuteur, qui valide ou pas tout que je faisais. J'ai aussi amélioré à l'oral, vu que mon tuteur est hispanique et ne parle pas français. Donc, toutes nos discutions se faisaient en anglais.

Avec tout ça, j'ai pu développer des visuels compréhensifs, capables d'être utilisé dans des réels projets. Il n'est pas encore complet, mais c'est n'est qu'une question de temps.

# 4 -BILAN

Ce stage a été pour moi un voyage dans le monde informatique. En effet j'ai pu assisté à des événements comme le **sprint** et le **kata,** qui consiste à s' associer avec un membre de l'équipe, pour résoudre en bug sur l'environnement de pharo ou problème. Cela te permet de te tester sur beaucoup d'aspects, notamment sur ta capacité à collaborer avec une personne, à gérer le stresse, car il y a un minuteur pour résoudre chaque bug ou problème.

Le plus important est surtout que tu vois comment les autres réfléchissent et résolvent des problèmes. Et tu peux prendre exemple sur eux, pour essayer de t'améliorer.

J'ai aussi eu la chance d'assister à des conférences sur des nouvelles technologies très enrichissantes, comme sur la robotique et l'intelligence artificielle.  Mais aussi sur l'écologie et l'économie circulaire.

Cependant, si j'ai quelque chose à déplorer, ce que parfois, il y avait beaucoup de choses à apprendre, qui au final certains ne m'ont pas été utile, dans le sens ou je n'en avais pas besoin pour faire mon projet. À titre d'exemple, Roassal utilise une bibliothèque du nom NSscale , qui permet de faire des calculs arithmétiques pour bien placé les point au bon endroit.

Donc, parfois, je passais des journées entières à chercher à savoir comment interagir avec des objets, par manque de documentations dans le code, et parfois, j'allais aussi à  l'aveugle en testant des choses au hasard. Parce que la personne qui a développé a omis des mettre des exemples d'utilisations. Une chose que j'en ai pas manquer de faire pour mon projet. Mais heureusement mon tuteur a été là pour moi.

A l'issue de ce stage, J'ai découvert beaucoup de métiers du milieu. Et je me suis fais l'idée de ce que je veux faire.

# 5 -BIBLIOGRAPHIE

https://github.com/sambegou122/DLPoster : Lien du dépôt github du projet de stage.

https://www.amazon.fr/Agile-Visualization-Pharo-Crafting-Interactive/dp/1484271602 : Lien du livre « **Agile vizualisation with Pharo**» de Pharo. Ce livre m'a permis de comprendre le bibliothèque Roassal.

https://github.com/bergel/AgileVisualizationAPressCode : Lien github des explication du livre « **Agile vizualisation with Pharo**»

https://github.com/akevalion/Roassal3 : Lien github du projet Roassal

https://github.com/PolyMathOrg/DataFrame : Lien github du projet DATAFRAME

https://www.coursera.org/lecture/python-data-analysis : Le lien vers le cours Pandas, Numpy, Matplotlib.

https://www-eni-training-com.ressources-electroniques.univ-lille.fr/portal/client/mediabook/home : deuxième lien vers la formation Numpy, Pandas, Matplotlib.

https://matplotlib.org/stable/plot_types/index.html : Lien vers mes références Matplotlib

https://pharo.org : Le site du langage Pharo

## GLOSSAIRE :

- - **Chart** : Ce mot est beaucoup utilisé dans mon rapport, il veut dire simplement graphique. C'est-a-dire l'ensemble du schéma.
- - **Plot** : Prenez ce mot dans mon rapport un peu comme un tracé.

# 6- ANNEXES

Dans cet annexe, je vais vous mettre quelques visuels que j'ai eu le temps de vous présenter dans mon rapport. Comme stackeBarPlots et clusterBarPlots