Introduction à la régression

Lino Galiana
Version du 27 septembre 2026

La régression linéaire est la première modélisation statistique qu'on découvre dans un cursus quantitatif. Il s'agit en effet d'une méthode très intuitive et très riche. Le *Machine Learning* permet de l'appréhender d'une autre manière que l'économétrie. Avec `scikit` et `statsmodels`, on dispose de tous les outils pour satisfaire à la fois data scientists et économistes.

⚠️ **PDF généré automatiquement.** Ce document est produit à partir du site du cours et peut présenter des problèmes de mise en forme. Pour une version à jour et adaptée à la lecture, consultez la page du cours : pythonds.linogaliana.fr.

Table des matières

Le précédent chapitre visait à proposer un premier modèle pour comprendre les comtés où le parti Républicain l'emporte. La variable d'intérêt étant bimodale (victoire ou défaite), on était dans le cadre d'un modèle de classification.

Maintenant, sur les mêmes données, on va proposer un modèle de régression pour expliquer le score du parti Républicain. La variable est donc continue. Nous ignorerons le fait que ses bornes se trouvent entre 0 et 100 et donc qu'il faudrait, pour être rigoureux, transformer l'échelle afin d'avoir des données dans cet intervalle.

L'ensemble de la partie *machine learning* utilise le même jeu de données, présenté dans l'introduction de cette partie : les données de vote aux élections présidentielles américaines croisées à des variables sociodémographiques. Le code est disponible sur Github.

```
!pip install geopandas openpyxl plotnine plotly
```

```
import requests

url     =     'https://raw.githubusercontent.com/linogaliana/python-datascientist/main/content/
modelisation/get_data.py'
r = requests.get(url, allow_redirects=True)
open('getdata.py', 'wb').write(r.content)

import getdata
votes = getdata.create_votes_dataframes()
```

Ce chapitre va utiliser plusieurs *packages* de modélisation, les principaux étant `Scikit` et `Statsmodels`. Voici une suggestion d'import pour tous ces *packages*.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import sklearn.metrics
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

1 Principe général

Le principe général de la régression consiste à trouver une loi $h_\theta(X)$ telle que

$$h_\theta(X) = \mathbb{E}_\theta(Y|X)$$

Cette formalisation est extrêmement généraliste et ne se restreint d'ailleurs pas à la régression linéaire.

En économétrie, la régression offre une alternative aux méthodes de maximum de vraisemblance et aux méthodes des moments. La régression est un ensemble très vaste de méthodes, selon la famille de modèles (paramétriques, non paramétriques, etc.) et la structure de modèles.

1.1 La régression linéaire

C'est la manière la plus simple de représenter la loi $h_\theta(X)$ comme combinaison linéaire de variables X et de paramètres θ. Dans ce cas,

$$\mathbb{E}_\theta(Y|X) = X\beta$$

Cette relation est encore, sous cette formulation, théorique. Il convient de l'estimer à partir des données observées y. La méthode des moindres carrés consiste à minimiser l'erreur quadratique entre la prédiction et les données observées (ce qui explique qu'on puisse voir la régression comme un problème de *Machine Learning*). En toute généralité, la méthode des moindres carrés consiste à trouver l'ensemble de paramètres θ tel que

$$\theta = \arg\min_{\theta \in \Theta} \mathbb{E}\left[\left(y - h_\theta(X)\right)^2 \right]$$

Ce qui, dans le cadre de la régression linéaire, s'exprime de la manière suivante :

$$\beta = \arg\min \mathbb{E}\left[(y - X\beta)^2 \right]$$

Lorsqu'on amène le modèle théorique ($\mathbb{E}_\theta(Y|X) = X\beta$) aux données, on formalise le modèle de la manière suivante :

$$Y = X\beta + \epsilon$$

Avec une certaine distribution du bruit ϵ qui dépend des hypothèses faites. Par exemple, avec des $\epsilon \sim \mathcal{N}(0, \sigma^2)$ i.i.d., l'estimateur β obtenu est équivalent à celui du Maximum de Vraisemblance dont la théorie asymptotique nous assure l'absence de biais, la variance minimale (borne de Cramer-Rao).

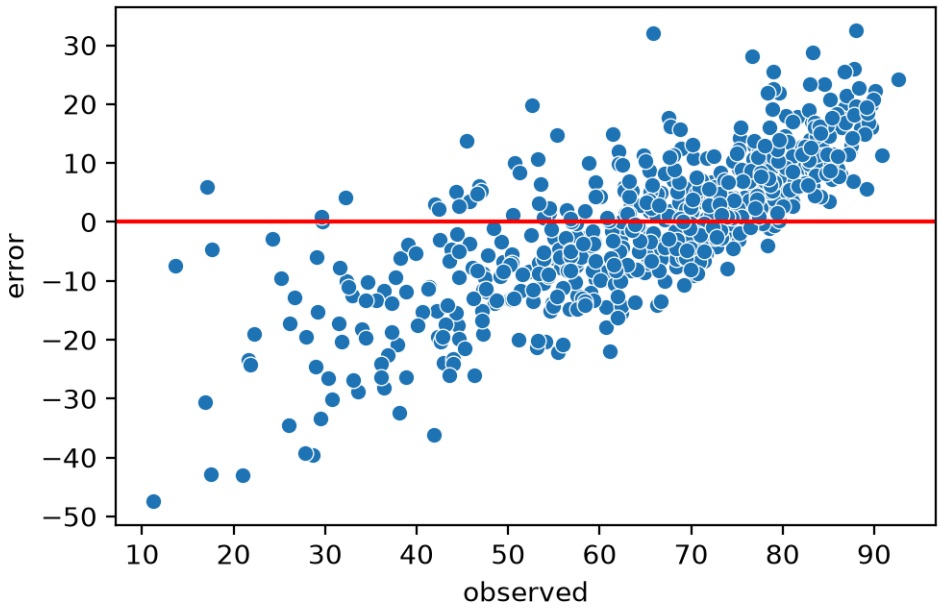
1.1.a Application

Toujours sous le patronage des héritiers de A. Siegfried [1], notre objectif, dans ce chapitre, est d'expliquer et prédire le score des Républicains à partir de quelques variables socioéconomiques. Contrairement au chapitre précédent, où on se focalisait sur un résultat binaire (victoire/défaite des Républicains), cette fois on va chercher à modéliser directement le score des Républicains.

Le prochain exercice vise à illustrer la manière d'effectuer une régression linéaire avec `scikit`. Dans ce domaine, `statsmodels` est nettement plus complet, ce que montrera l'exercice suivant. L'intérêt principal de faire des régressions avec `scikit` est de pouvoir comparer les résultats d'une régression linéaire avec d'autres modèles de régression dans une perspective de sélection du meilleur modèle prédictif.

> Exercice 1a : Régression linéaire avec scikit
>
> 1. A partir de quelques variables, par exemple, *"Unemployment_rate_2019", "Median_Household_Income_2021", "Percent of adults with less than a high school diploma, 2018-22", « Percent of adults with a bachelor's degree or higher, 2018-22 »*, expliquer la variable `per_gop` à l'aide d'un échantillon d'entraînement `X_train` constitué au préalable.
>
> ⚠️ Utiliser la variable `Median_Household_Income_2021` en `log` sinon son échelle risque d'écraser tout effet.
>
> 2. Afficher les valeurs des coefficients, constante comprise
> 3. Evaluer la pertinence du modèle avec le R^2 et la qualité du fit avec le MSE.
> 4. Représenter un nuage de points des valeurs observées et des erreurs de prédiction. Observez-vous un problème de spécification ?

À la question 4, on peut voir que la répartition des erreurs n'est clairement pas aléatoire en fonction de X.



Le modèle souffre donc d'un problème de spécification, il faudra par la suite faire un travail sur les variables sélectionnées. Avant cela, on peut refaire cet exercice avec le *package* `statsmodels`.

> Exercice 1b : Régression linéaire avec statsmodels
>
> Cet exercice vise à illustrer la manière d'effectuer une régression linéaire avec `statsmodels` qui offre des fonctionnalités plus proches de celles de `R`, et moins orientées *Machine Learning*.
>
> L'objectif est toujours d'expliquer le score des Républicains en fonction de quelques variables.
>
> 1. A partir de quelques variables, par exemple, *"Unemployment_rate_2019", "Median_Household_Income_2021", "Percent of adults with less than a high school diploma, 2018-22", « Percent of adults with a bachelor's degree or higher, 2018-22 »*, expliquer la variable

`per_gop`. ⚠️ utiliser la variable `Median_Household_Income_2021` en `log` sinon son échelle risque d'écraser tout effet.
2. Afficher un tableau de régression.
3. Evaluer la pertinence du modèle avec le R^2.
4. Utiliser l'API `formula` pour régresser le score des républicains en fonction de la variable `Unemployment_rate_2021`, de `Unemployment_rate_2019` au carré et du log de `Median_Household_Income_2021`.

```
R2:  0.43016652968844415
```

Tip

Pour sortir une belle table pour un rapport sous $\LaTeX$, il est possible d'utiliser la méthode `Summary.as_latex`. Pour un rapport HTML, on utilisera `Summary.as_html`

Note

Les utilisateurs de `R` retrouveront des éléments très familiers avec `statsmodels`, notamment la possibilité d'utiliser une formule pour définir une régression. La philosophie de `statsmodels` est similaire à celle qui a influé sur la construction des packages `stats` et `MASS` de `R`: offrir une librairie généraliste, proposant une large gamme de modèles.

Néanmoins, `statsmodels` bénéficie de sa jeunesse par rapport aux packages `R`. Depuis les années 1990, les packages `R` visant à proposer des fonctionalités manquantes dans `stats` et `MASS` se sont multipliés alors que `statsmodels`, enfant des années 2010, n'a eu qu'à proposer un cadre général (les *generalized estimating equations*) pour englober ces modèles.

1.2 La régression logistique

Nous avons appliqué notre régression linéaire sur une variable d'*outcome* continue. Comment faire avec une distribution binaire ?
Dans ce cas, $\mathbb{E}_\theta(Y|X) = \mathbb{P}_\theta(Y = 1|X)$.
La régression logistique peut être vue comme un modèle linéaire en probabilité :

$$\text{logit}\Big(\mathbb{E}_\theta(Y|X)\Big) = \text{logit}\Big(\mathbb{P}_\theta(Y = 1|X)\Big) = X\beta$$

La fonction logit est $]0, 1[\rightarrow \mathbb{R} : p \mapsto \log(\frac{p}{1-p})$.

Elle permet ainsi de transformer une probabilité dans $\mathbb{R}$. Sa fonction réciproque est la sigmoïde $(\frac{1}{1+e^{-x}})$, objet central du *Deep Learning*.

Il convient de noter que les probabilités ne sont pas observées, c'est l'*outcome* binaire (0/1) qui l'est. Cela amène à voir la régression logistique de deux manières différentes :

- En économétrie, on s'intéresse au modèle latent qui détermine le choix de l'outcome. Par exemple, si on observe les choix de participer ou non au marché du travail, on va modéliser les facteurs déterminant ce choix ;
- En *Machine Learning*, le modèle latent n'est nécessaire que pour classifier dans la bonne catégorie les observations.

L'estimation des paramètres β peut se faire par maximum de vraisemblance ou par régression, les deux solutions sont équivalentes sous certaines hypothèses.

> **Note**
>
> Par défaut, `scikit` applique une régularisation pour pénaliser les modèles peu parcimonieux (comportement différent de celui de `statsmodels`). Ce comportement par défaut est à garder à l'esprit si l'objectif n'est pas de faire de la prédiction.

> **Exercice 2a : Régression logistique avec scikit**
>
> Avec `scikit`, en utilisant échantillons d'apprentissage et d'estimation :
>
> 1. Evaluer l'effet des variables déjà utilisées sur la probabilité des Républicains de gagner. Affichez la valeur des coefficients.
> 2. Déduire une matrice de confusion et une mesure de qualité du modèle.
> 3. Supprimer la régularisation grâce au paramètre `penalty`. Quel effet sur les paramètres estimés ?

> **Exercice 2b : Régression logistique avec statmodels**
>
> En utilisant échantillons d'apprentissage et d'estimation :
>
> 1. Evaluer l'effet des variables déjà utilisées sur la probabilité des Républicains de gagner.
> 2. Faire un test de ratio de vraisemblance concernant l'inclusion de la variable de (log)-revenu.

La p-value du test de maximum de ratio de vraisemblance étant proche de 1, cela signifie que la variable log revenu ajoute, presque à coup sûr, de l'information au modèle.

> **Tip**
>
> La statistique du test est :
>
> $$LR = -2\log\left(\frac{\mathcal{L}_\theta}{\mathcal{L}_{\theta_0}}\right) = -2(\ell_\theta - \ell_{\theta_0})$$

2 Pour aller plus loin

Ce chapitre n'évoque les enjeux de la régression que de manière très introductive. Pour compléter ceci, il est recommandé d'aller plus loin en fonction de vos centres d'intérêt et de vos besoins.

Dans le domaine du *machine learning*, les principales voies d'approfondissement sont les suivantes:

- Les modèles de régression alternatifs comme les forêts aléatoires.
- Les méthodes de *boosting* et *bagging* pour découvrir la manière dont plusieurs modèles peuvent être entraînés de manière conjointe et leur prédiction sélectionnée selon un principe démocratique pour converger vers une meilleure décision qu'un modèle simple.
- Les enjeux liés à l'explicabilité des modèles, un champ de recherche très actif, pour mieux comprendre les critères de décision des modèles.

Dans le domaine de l'économétrie, les principales voies d'approfondissement sont les suivantes:

- Les modèles linéaires généralisés pour découvrir la régression avec des hypothèses plus générales que celles que nous avons posées jusqu'à présent ;

- Les tests d'hypothèses pour aller plus loin sur ces questions que notre test de ratio de vraisemblance.

Références

Informations additionnelles

i Environnement Python

Ce site a été construit automatiquement par le biais d'une action `Github` utilisant le logiciel de publication reproductible `Quarto` (version 1.10.18).

L'environnement utilisé pour obtenir les résultats est reproductible par le biais d'uv. Le fichier `pyproject.toml` utilisé pour construire cet environnement est disponible sur le dépôt `linogaliana/python-datascientist`

```
[project]
name = "python-datascientist"
version = "0.1.0"
description = "Source code for Lino Galiana's Python for data science course"
readme = "README.md"
requires-python = "≥3.13,<3.14"
dependencies = [
    "altair≥6.0.0",
    "cartiflette",
    "contextily==1.6.2",
    "duckdb≥0.10.1",
    "folium≥0.19.6",
    "gdal==3.11.4",
    "graphviz==0.20.3",
    "great-tables≥0.12.0",
    "gt-extras≥0.0.8",
    "ipykernel≥6.29.5",
    "jupyter≥1.1.1",
    "jupyter-cache≥1.0.0",
    "kaleido≥0.2.1",
    "langchain-community≥0.3.27",
    "loguru==0.7.3",
    "markdown≥3.8",
    "nbclient≥0.10.0",
    "nbformat≥5.10.4",
    "nltk≥3.9.1",
    "pandas≥3.0",
    "pip≥25.1.1",
    "plotly≥6.1.2",
    "plotnine≥0.15",
    "polars≥1.8.2",
    "pyarrow≥17.0.0",
    "pynsee≥0.1.8",
    "python-dotenv≥1.0.1",
    "python-frontmatter≥1.1.0",
    "pywaffle≥1.1.1",
    "requests≥2.32.3",
    "scikit-image≥0.24.0",
    "scikit-learn≥1.8.0",
```

```
    "scipy>=1.13.0",
    "seaborn>=0.13.2",
    "selenium<4.39.0",
    "spacy>=3.8.4",
    "webdriver-manager>=4.0.2",
    "wordcloud==1.9.3",
]

[tool.uv.sources]
cartiflette = { git = "https://github.com/inseefrlab/cartiflette" }
gdal = [
  { index = "gdal-wheels", marker = "sys_platform == 'linux'" },
  { index = "geospatial_wheels", marker = "sys_platform == 'win32'" },
]

[[tool.uv.index]]
name = "geospatial_wheels"
url = "https://nathanjmcdougall.github.io/geospatial-wheels-index/"
explicit = true

[[tool.uv.index]]
name = "gdal-wheels"
url = "https://gitlab.com/api/v4/projects/61637378/packages/pypi/simple"
explicit = true

[dependency-groups]
dev = [
    "nb-clean>=4.0.1",
]
```

Pour utiliser exactement le même environnement (version de `Python` et *packages*), se reporter à la documentation d'`uv`.

Bibliographie

[1] A. Siegfried, *Tableau politique de la France de l'ouest sous la troisième république: 102 cartes et croquis, 1 carte hors texte*. A. Colin, 1913.