Análisis estadístico de datos

Trabajo práctico: Ajuste de datos

Nombre: Alvaro Concha

Fecha: 30 de noviembre de 2021

Ajuste de la eficiencia

1. Considerar una variable aleatoria k que sigue una distribución binomial con un número de pruebas de Bernoulli $n = 30$ y probabilidad de éxito o eficiencia p. Calcular la función de verosimilitud $L(p)$, el estimador de máxima verosimilitud $\hat{p}$, la verosimilitud máxima $L_{\max}$, el cociente de verosimilitudes $\lambda(p)$ y la función de costo $J_1(p)$ para los casos:

- k =0
- 0 <k<n
- k = n

1. Graficar la función de costo $J_1(p)$ para los tres casos anteriores.

2. En un experimento se midieron los 20 datos de la tabla 3 que contiene la variable independiente x (columna 1), el número de pruebas de Bernoulli n (columna 2) y el número de éxitos k (columna 3). Los datos se proveen además en el archivo `eficiencia.dat`. Graficar los datos usando el eje horizontal para la variable independiente x y el eje vertical para el estimador de la eficiencia calculado a partir de k y n.

x	n	k
0.050	30	0
0.100	30	0
0.150	30	0
0.200	30	3
0.250	30	3
0.300	30	2
0.350	30	8
0.400	30	5
0.450	30	4
0.500	30	11
0.550	30	18
0.600	30	15
0.650	30	19
0.700	30	20
0.750	30	26

x	n	k
0.800	30	24
0.850	30	26
0.900	30	29
0.950	30	30
1.000	30	30

1. El modelo a ajustar es la función sigmoide,

$$s(x; a, b) = \frac{1}{1 + \exp[(x - a)/b]},$$

con a y b los parámetros del ajuste. En base al modelo $s(x)$ y a la función de costo $J_1(p)$ escribir el código para la función de costo del ajuste $J(\theta)$ con $\theta = (a, b)$.

2. Minimizar el costo del ajuste $J(\theta)$. Calcular los estimadores de máxima verosimilitud y los errores de los parámetros a y b. Calcular la correlación entre los estimadores de a y b.

3. Calcular el χ^2 del ajuste y calcular su pvalor.

4. Graficar el ajuste junto a los datos.

5. Calcular y graficar la banda de error del ajuste considerando que

$$\frac{\partial s}{\partial a} = s(1 - s)\left(\frac{-1}{b}\right)$$
$$\frac{\partial s}{\partial b} = s(1 - s)\left(\frac{a}{b^2}\right).$$

Nota: Entregar informe del TP en formato PDF + código + figuras

In [1]:
```
"""Instalar librerias necesarias"""
%pip install numpy scipy seaborn
```

```
Requirement already satisfied: numpy in /home/alvaro/.local/lib/python3.8/sit
e-packages (1.17.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-package
s (1.5.3)
Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packa
ges (0.11.0)
Requirement already satisfied: pandas>=0.23 in /home/alvaro/.local/lib/python
3.8/site-packages (from seaborn) (1.1.3)
Requirement already satisfied: matplotlib>=2.2 in /home/alvaro/.local/lib/pyt
hon3.8/site-packages (from seaborn) (3.4.3)
Requirement already satisfied: pytz>=2017.2 in /usr/lib/python3/dist-packages
(from pandas>=0.23->seaborn) (2019.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/lib/python3/dis
t-packages (from pandas>=0.23->seaborn) (2.7.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/
dist-packages (from matplotlib>=2.2->seaborn) (1.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/d
ist-packages (from matplotlib>=2.2->seaborn) (2.4.7)
```

```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-
packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in /home/alvaro/.local/lib/pytho
n3.8/site-packages (from matplotlib>=2.2->seaborn) (8.3.2)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from cy
cler>=0.10->matplotlib>=2.2->seaborn) (1.14.0)
Note: you may need to restart the kernel to use updated packages.
```

Solución

Sea la variable aleatoria discreta

$$K \sim \text{Binomial}(n, p),$$

con $n = 30$ fijo, $0 \leq p \leq 1$ y función de masa de probabilidad

$$f(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k},$$

Luego, en el caso $0 < k < n$, la likelihood es

$$\begin{aligned} L(p) &= f(k; n, p) \\ &= \binom{n}{k} p^k (1-p)^{n-k}, \end{aligned}$$

y el estimador de maximum likelihood (MLE) es

$$\begin{aligned} \hat{p} &= \underset{p}{\text{argmax}} L(p) \\ &= \underset{p}{\text{argmax}} \ln L(p), \end{aligned}$$

donde aplicar el logaritmo no modifica la ubicación del máximo, por ser una función monótona creciente, y resulta la log-likelihood

$$\ln L(p) = \underbrace{\ln \binom{n}{k}}_{\text{constante en } p} + k \ln p + (n-k) \ln (1-p).$$

Buscamos puntos críticos de $\ln L(p)$, por lo que igualamos su derivada a cero

$$\begin{aligned} 0 &= d_p \ln L(p) \big|_{p=\hat{p}} \\ &= \frac{k}{\hat{p}} - \frac{n-k}{1-\hat{p}} \\ \implies \hat{p} &= \frac{k}{n}, \end{aligned}$$

y $\hat{p}$ es donde ocurre el máximo, pues la derivada segunda es negativa

$$\begin{aligned} d_p^2 \ln L(p)|_{p=\hat{p}} &= -\frac{k}{\hat{p}^2} - \frac{n-k}{(1-\hat{p})^2} \\ &< 0. \end{aligned}$$

Entonces, la likelihood máxima es

$$\begin{aligned} L_{\max} &= \max_p L(p) \\ &= L(\hat{p}) \\ &= \binom{n}{k} \hat{p}^k (1-\hat{p})^{n-k}, \end{aligned}$$

el cociente de likelihood resulta

$$\begin{aligned} \lambda(p) &= \frac{L(p)}{L_{\max}} \\ &= \left(\frac{p}{\hat{p}}\right)^k \left(\frac{1-p}{1-\hat{p}}\right)^{n-k}, \end{aligned}$$

y la función de costo es

$$\begin{aligned} J_1(p) &= -2 \ln \lambda(p) \\ &= 2n \left[\hat{p} \ln\left(\frac{\hat{p}}{p}\right) + (1-\hat{p}) \ln\left(\frac{1-\hat{p}}{1-p}\right) \right]. \end{aligned}$$

Por su parte, en el caso $k = 0$, la likelihood es

$$L(p) = \underbrace{(1-p)^n}_{n \text{ fracasos}},$$

con estimador de maximum likelihood

$$\hat{p} = 0,$$

donde

$$L_{\max} = 1,$$

y el cociente de likelihood resulta

$$\lambda(p) = (1-p)^n,$$

con funcion de costo

$$J_1(p) = 2n \ln\left(\frac{1}{1-p}\right).$$

Mientras que, para el caso $k = n$, la likelihood es

$$L(p) = \underbrace{p^n}_{n \text{ éxitos}},$$

con estimador de maximum likelihood

$$\hat{p} = 1,$$

donde

$$L_{\max} = 1,$$

y el cociente de likelihood resulta

$$\lambda(p) = p^n,$$

con funcion de costo

$$J_1(p) = 2n \ln\left(\frac{1}{p}\right).$$

Entonces, resulta que la función de costo binomial depende de p y $\hat{p}$, y denotamos esto como $J_1(p; \hat{p})$.

En resumen

	Casos		
Resultados	$k = 0$	$0 < k < n$	$k = n$
$L(p)$	$(1-p)^n$	$\binom{n}{k} p^k (1-p)^{n-k}$	p^n
$\hat{p}$	0	$\frac{k}{n}$	1
$L_{\max}$	1	$\binom{n}{k} \hat{p}^k (1-\hat{p})^{n-k}$	1
$\lambda(p)$	$(1-p)^n$	$\left(\frac{p}{\hat{p}}\right)^k \left(\frac{1-p}{1-\hat{p}}\right)^{n-k}$	p^n
$J_1(p; \hat{p})$	$2n \ln\left(\frac{1}{1-p}\right)$	$2n \left[\hat{p} \ln\left(\frac{\hat{p}}{p}\right) + (1-\hat{p}) \ln\left(\frac{1-\hat{p}}{1-p}\right)\right]$	$2n \ln\left(\frac{1}{p}\right)$

A partir de estos resultados, podemos expresar la función de costo del ajuste como

$$J(\theta) = \sum_{i=1}^{N_{\text{obs}}} J_1(\mu_i(\theta); \hat{p}_i),$$

siendo $\mu_i(\theta)$ nuestro modelo para el parámetro de la distribución binomial p_i, de la forma

$$\begin{aligned} \mu_i(\theta) &= s(x_i; a, b) \\ &= \frac{1}{1 + \exp[(x_i - a)/b]}, \end{aligned}$$

donde $\theta = (a, b)$ son los parámetros del ajuste y tenemos a disposición el conjunto de N_{obs} pares de datos $(x_i, \hat{p}_i)$, con x_i variable independiente y p_i variable dependiente calculada como el estimador de maximum likelihood

$$\hat{p}_i = \frac{k_i}{n}.$$

Finalmente, se obutiveron los siguientes valores para los estimadores de los parámetros del ajuste, usando el método de maximum likelihood

$$\hat{a} = 0.571 \pm 0.015,$$

y

$$\hat{b} = -0.121 \pm 0.009,$$

con una correlación

$$\rho = -0.04,$$

un chi-cuadrado

$$\begin{aligned} \chi^2 &= \min J(\theta) \\ &= J(\hat{\theta}) \\ &= 24.01, \end{aligned}$$

y un p-value

$$\begin{aligned} p\text{-value} &= (1 - F_{N_{\text{dof}}}(\chi^2)) \times 100\% \\ &= 15.5\%, \end{aligned}$$

donde $F_{N_{\text{dof}}}$ es una densidad acumulada chi-cuadrado con grados de libertad

$$\begin{aligned} N_{\text{dof}} &= N_{\text{obs}} - \#\theta \\ &= 18. \end{aligned}$$

Dado que el p-value es de 15.5% no podemos rechazar la hipótesis nula, y, bajo esta evidencia, aceptamos el modelo de ajuste propuesto.

Comentarios

Valores iniciales de los estimadores de los parámetros en el proceso de optimización

Por un lado, para elegir un valor inicial del estimador del parámetro a, utilizamos que el punto central x_c de la sigmoide

$$\begin{aligned} s(x_c; a, b) &= \frac{1}{1 + \exp[(x_c - a)/b]} \\ &= 0.5, \end{aligned}$$

es igual a

$$x_c = a,$$

y, observando la gráfica de los datos, este punto sería

$$x_c \approx 0.5,$$

por lo que inicializamos el estimador en

$$a = 0.5.$$

Por otro lado, para elegir un valor inicial del estimador del parámetro b, utilizamos la pendiente de la sigmoide en el punto central x_c,

$$\begin{aligned} d_x s(x;a,b)|_{x=x_c} &= -\underbrace{[s(x_c;a,b)]^2}_{0.25}\underbrace{\exp[(x_c-a)/b]}_{1}\frac{1}{b} \\ &= -0.25\frac{1}{b}, \end{aligned}$$

y, nuevamente, de la gráfica de los datos, vemos que la pendiente sería

$$d_x s(x;a,b)|_{x=x_c} \approx 1,$$

por lo que inicializamos el estimador en

$$b = -0.25.$$

Relación de la función de costo con la divergencia de Kullback-Leibler

La función de costo del ajuste de los parámetros es una suma de divergencias de Kullback-Leibler [Wikipedia - Kullback-Leibler divergence, Etz A - Technical Notes on Kullback-Leibler Divergence].

Esto se debe a que la función de costo binomial $J_1(p;\hat{p})$ es proporcional a la divergencia de Kullback-Leibler $D_{KL}(P_{\text{datos}}||P_{\text{modelo}})$ de las probabilidades de éxito y de fracaso de los datos P_{datos} y del modelo P_{modelo}

$$\begin{aligned} \frac{J_1(p;\hat{p})}{2n} &= \begin{cases} \ln\left(\frac{1}{1-p}\right) & \text{si } \hat{p}=0 \\ \left[\hat{p}\ln\left(\frac{\hat{p}}{p}\right)+(1-\hat{p})\ln\left(\frac{1-\hat{p}}{1-p}\right)\right] & \text{si } 0<\hat{p}<1 \\ \ln\left(\frac{1}{p}\right) & \text{si } \hat{p}=1 \end{cases} \\ &= D_{KL}(P_{\text{datos}}||P_{\text{modelo}}), \end{aligned}$$

donde tenemos las probabilidades

$$P_{\text{datos}} = \begin{cases} \hat{p} & \text{éxito} \\ 1-\hat{p} & \text{fracaso} \end{cases},$$

para el estimador de maximum likelihood $\hat{p}$ calculado a partir de los datos como

$$\hat{p} = \frac{k}{n},$$

con n fijo, y

$$P_{\text{modelo}} = \begin{cases} p & \text{éxito} \\ 1-p & \text{fracaso} \end{cases},$$

para el parámetro p del modelo, dado por la sigmoide de parámetros a y b

$$\begin{aligned} p &= s(x; a, b) \\ &= \frac{1}{1 + \exp[(x-a)/b]}, \end{aligned}$$

donde x es la variable independiente.

En clase vimos que para ajuste de datos gaussianos con método de mínimos cuadrados, maximizar la likelihood equivalía a minimizar el error cuadrático (caso homocedástico) o la distancia de Mahalanobis (caso heterocedástico) de los datos.

En este ejercicio, vemos que maximizar la likelihood, usando datos con distribución binomial, equivale a minimizar la divergencia de Kullback-Leibler.

De manera intuitiva, la divergencia de Kullback-Leibler es una medida de qué tan diferente es una distribución de probabilidad, respecto de otra.

Por lo que tiene sentido que emerja como función de costo en un problema de ajuste de datos y de optimización.

In [2]:

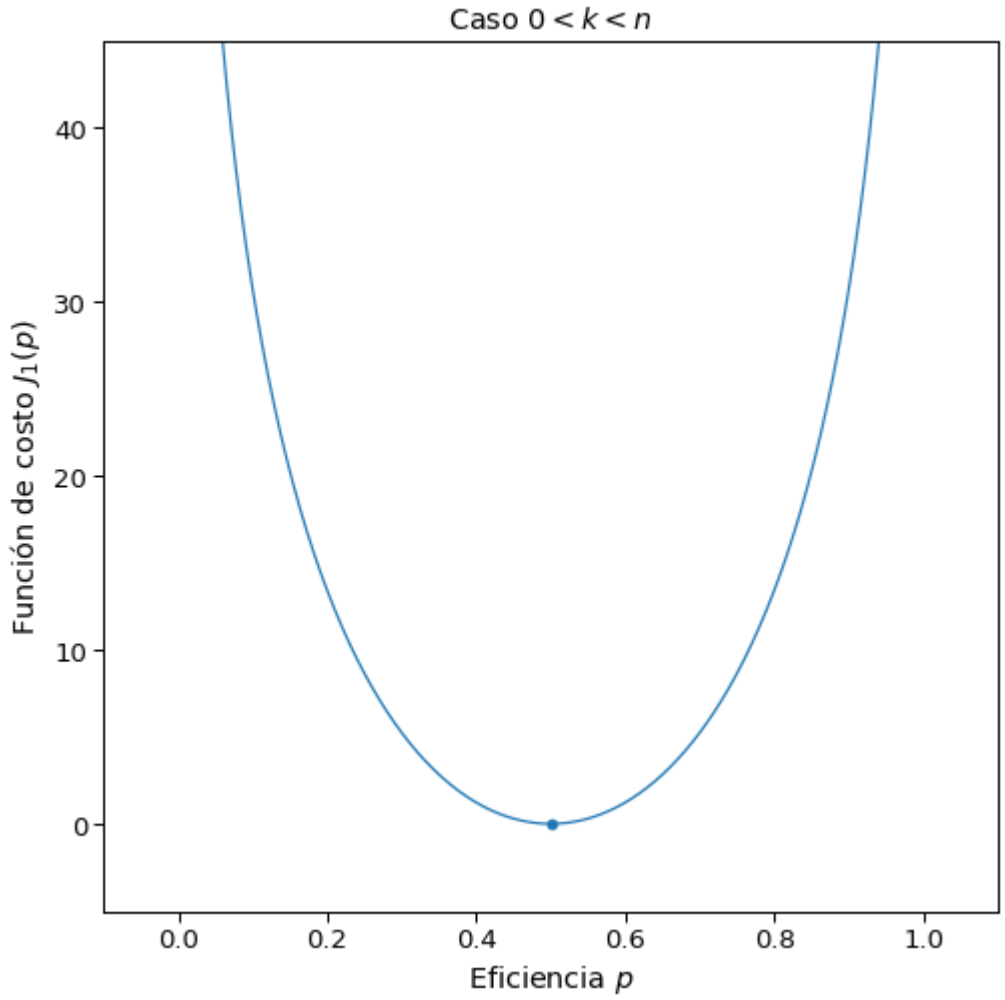
```
"""Codigo Punto 2"""
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ipywidgets as widgets
from IPython.display import display


def plot_punto2(n, k, save=False):
    """Grafica Punto 2."""
    p_lims = (1e-12, 1.0 - 1e-12)
    ps = np.linspace(*p_lims, 1000)
    p_MLE = k / n

    if k == 0:
        title = r"Caso $k=0$"
        def J_1(n, _, ps):
            return 2.0 * n * np.log(1.0 / (1.0 - ps))
    elif k == n:
        title = r"Caso $k=n$"
        def J_1(n, _, ps):
            return 2.0 * n * np.log(1.0 / ps)
    else:
        title = r"Caso $0<k<n$"
        def J_1(n, p_MLE, ps):
            return 2.0 * n * (
                p_MLE * np.log(p_MLE / ps) +
                (1 - p_MLE) * np.log((1 - p_MLE) / (1 - ps))
            )

    plt.figure(figsize=(8, 8))
    sns.set_context("paper", font_scale=1.5)
```

```
    plt.plot(ps, J_1(n, p_MLE, ps))
    plt.scatter(p_MLE, J_1(n, p_MLE, p_MLE))
    plt.xlabel(r"Eficiencia $p$")
    plt.ylabel(r"Función de costo $J_1(p)$")
    plt.title(title)
    plt.xlim(-0.1, 1.1)
    plt.ylim(-5.0, 45.0)
    if save:
        plt.savefig(save, bbox_inches="tight")
        plt.close()
    else:
        plt.show()
        plt.close()


def punto2():
    """Punto 2."""
    n = widgets.IntSlider(description="n", value=30, min=1, max=100, step=1)
    k = widgets.IntSlider(description="k", value=15, min=0, max=30, step=1)

    def update_k(*args):
        k.max = n.value
    k.observe(update_k, "value")

    def update_n(*args):
        n.min = k.value + 1
    n.observe(update_n, "value")

    parameters = {
        "n": n,
        "k": k,
    }
    out = widgets.interactive_output(plot_punto2, parameters)
    out.observe(lambda _: display(out), "value")
    title = widgets.Label(
        "Seleccionar parámetros",
        layout=widgets.Layout(display="flex", justify_content="center"),
    )
    sliders = [title, *parameters.values(), out]
    display(
        widgets.VBox(
            sliders,
            layout=widgets.Layout(
                width="100%",
                display="flex",
                align_items="center",
            ),
        )
    )


if __name__ == "__main__":
    punto2()
    n = 30
    plot_punto2(n=n, k=0, save="tp7_punto2_k=0.pdf")
    plot_punto2(n=n, k=15, save="tp7_punto2_k=15.pdf")
    plot_punto2(n=n, k=n, save="tp7_punto2_k=n.pdf")
```

Seleccionar parámetros

n 30

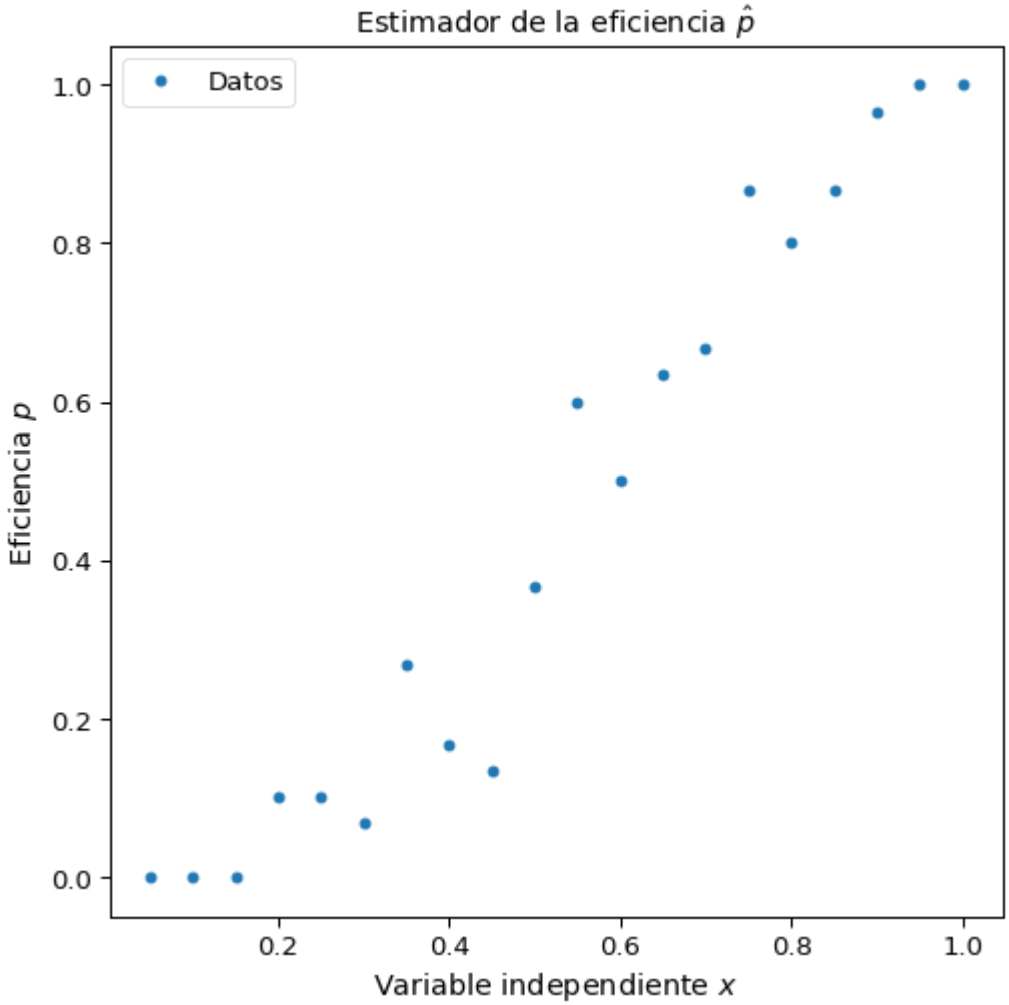
k 15



In [3]:

```
"""Codigo Punto 3"""
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


def punto3():
    """Punto 3."""
    xs, ns, ks = np.loadtxt("eficiencia.dat").T
    ps = ks / ns

    plt.figure(figsize=(8, 8))
    sns.set_context("paper", font_scale=1.5)
    plt.plot(xs, ps, "o", label="Datos")
    plt.xlabel(r"Variable independiente $x$")
    plt.ylabel(r"Eficiencia $p$")
    plt.title(r"Estimador de la eficiencia $\hat{p}$")
    plt.legend()
    plt.savefig("tp7_punto3.pdf", bbox_inches="tight")
    plt.show()
    plt.close()
```

```
if __name__ == "__main__":
    punto3()
```



In [4]:

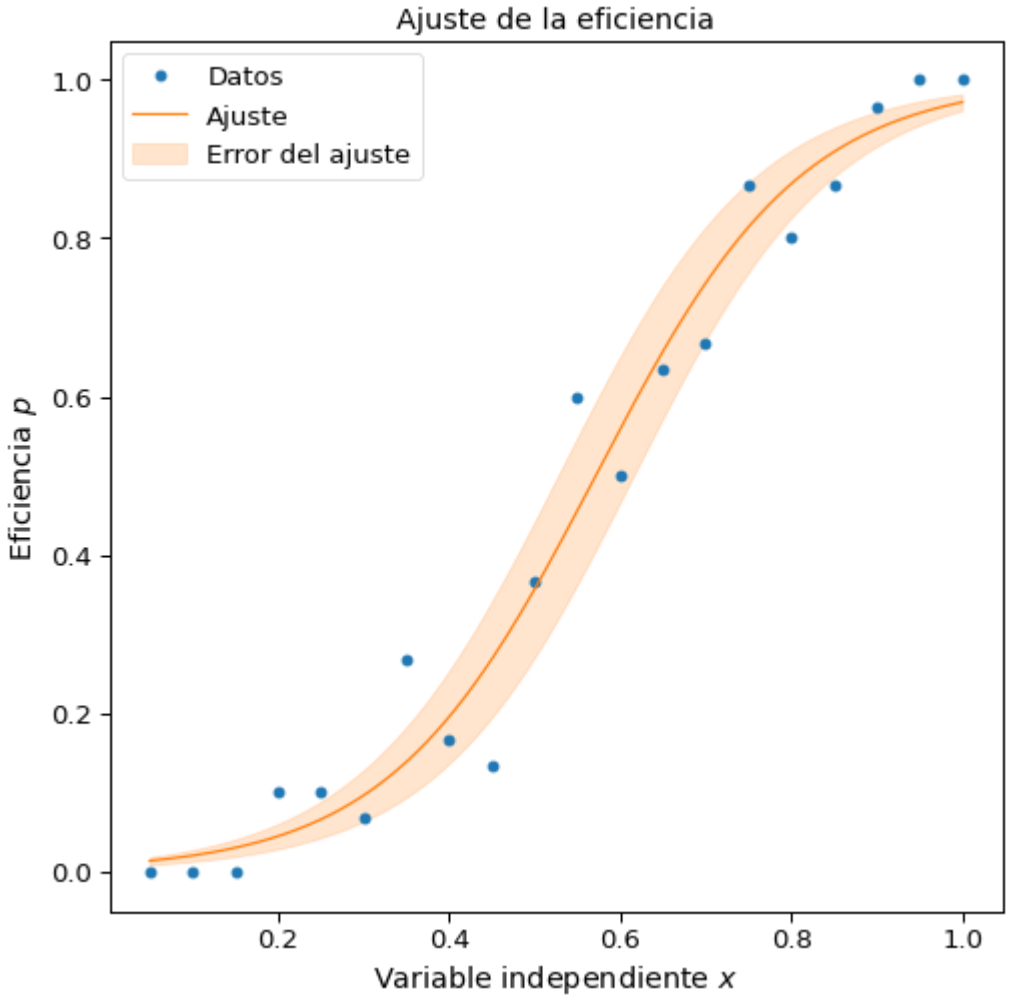
```
"""Codigo Punto 8"""
import numpy as np
from scipy.stats import chi2
from scipy.optimize import minimize
import matplotlib.pyplot as plt
import seaborn as sns


def J_1(n, p, k):
    """Funcion de costo binomial."""
    if k == 0:
        return 2.0 * n * np.log(1.0 / (1.0 - p))
    elif k == n:
        return 2.0 * n * np.log(1.0 / p)
    else:
        p_MLE = k / n
        return 2.0 * n * (
            p_MLE * np.log(p_MLE / p) +
            (1 - p_MLE) * np.log((1 - p_MLE) / (1 - p))
        )


def modelo(theta, x):
    """Funcion sigmoide."""
```

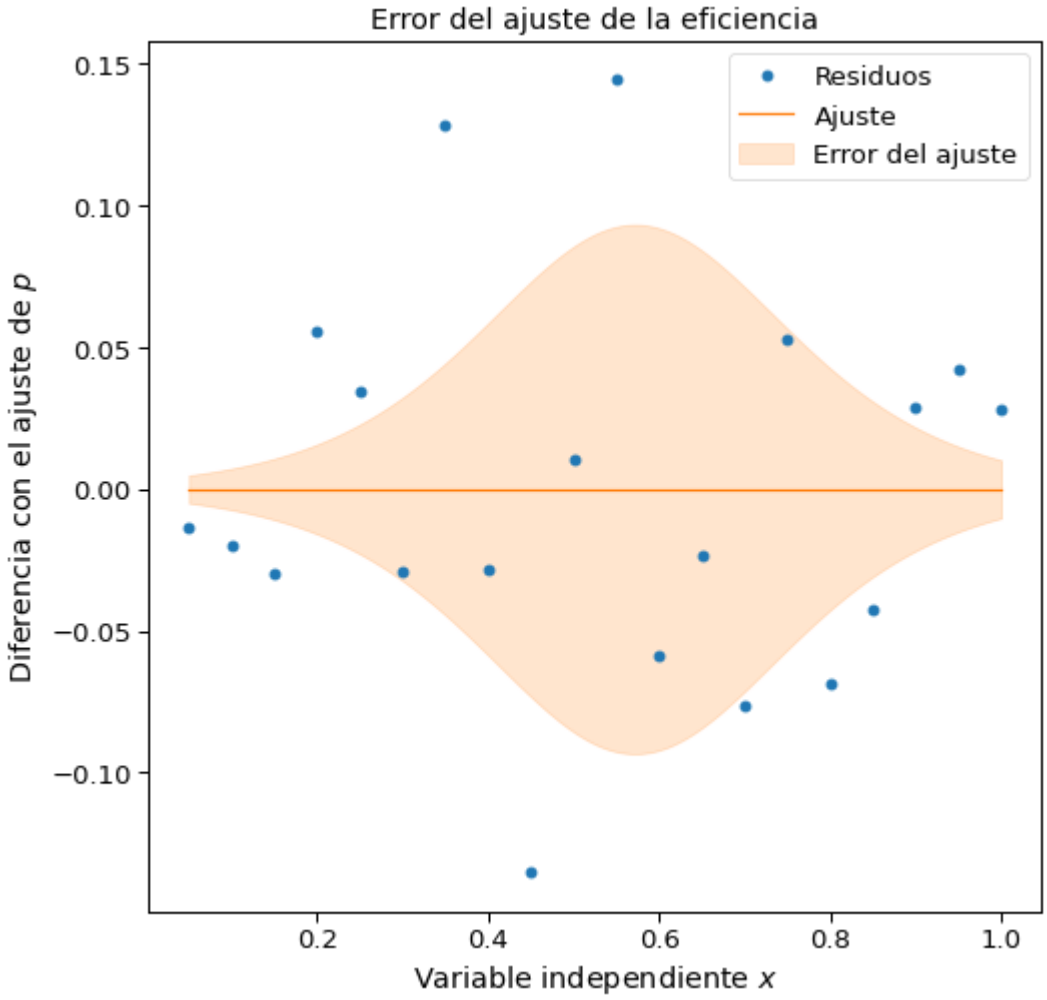
```python
    a, b = theta
    return 1.0 / (1.0 + np.exp((x - a) / b))


def funcion_costo_ajuste(theta, x, n, k):
    """Funcion de costo del ajuste."""
    mu = modelo(theta=theta, x=x)
    return np.vectorize(J_1)(n=n, p=mu, k=k).sum()


def punto8():
    """Grafica Punto 8."""
    xs, ns, ks = np.loadtxt("eficiencia.dat").T
    ps = ks / ns
    J = lambda theta: funcion_costo_ajuste(theta, x=xs, n=ns, k=ks)

    res = minimize(J, x0=(0.5, -0.25))
    print(res)

    x_est = np.linspace(xs.min(),xs.max(), 256)
    mu_est = modelo(theta=res.x, x=x_est)
    theta_est = res.x
    cov_est = 2*res.hess_inv
    error_est = np.sqrt( np.diagonal(cov_est) )
    rho_est = cov_est[0][1]/(error_est[0]*error_est[1])
    print(f"a = {theta_est[0]:.3f} +- {error_est[0]:.3f}")
    print(f"b = {theta_est[1]:.3f} +- {error_est[1]:.3f}")
    print(f"ρ = {rho_est:.2f}")

    chi2_min = res.fun
    ndof = len(xs) - len(theta_est)
    pvalor = chi2.sf(chi2_min, ndof)
    print(f"chi-cuadrado min = {chi2_min:.2f}")
    print(f"ndof = {ndof}")
    print(f"pvalor = {pvalor*100:.1f}%")

    grad_theta1 = mu_est * (1 - mu_est) * (-1.0 / theta_est[1])
    grad_theta2 = mu_est * (1 - mu_est) * (
        theta_est[0] / (theta_est[1] ** 2)
    )
    grad = np.column_stack([grad_theta1, grad_theta2])
    var_mu_est = np.einsum("ki,ij,kj->k", grad, cov_est, grad)
    sigma_mu_est = np.sqrt(var_mu_est)


    plt.figure(figsize=(8, 8))
    sns.set_context("paper", font_scale=1.5)
    plt.plot(xs, ps, "o", label="Datos")
    plt.plot(x_est, mu_est, label="Ajuste")
    plt.fill_between(
        x_est, mu_est-sigma_mu_est, mu_est+sigma_mu_est,
        color="tab:orange", alpha=0.2, label="Error del ajuste",
    )
    plt.xlabel(r"Variable independiente $x$")
    plt.ylabel(r"Eficiencia $p$")
    plt.title("Ajuste de la eficiencia")
    plt.legend()
    plt.savefig("tp7_punto8_ajuste.pdf", bbox_inches="tight")
    plt.show()
    plt.close()

    plt.figure(figsize=(8, 8))
```

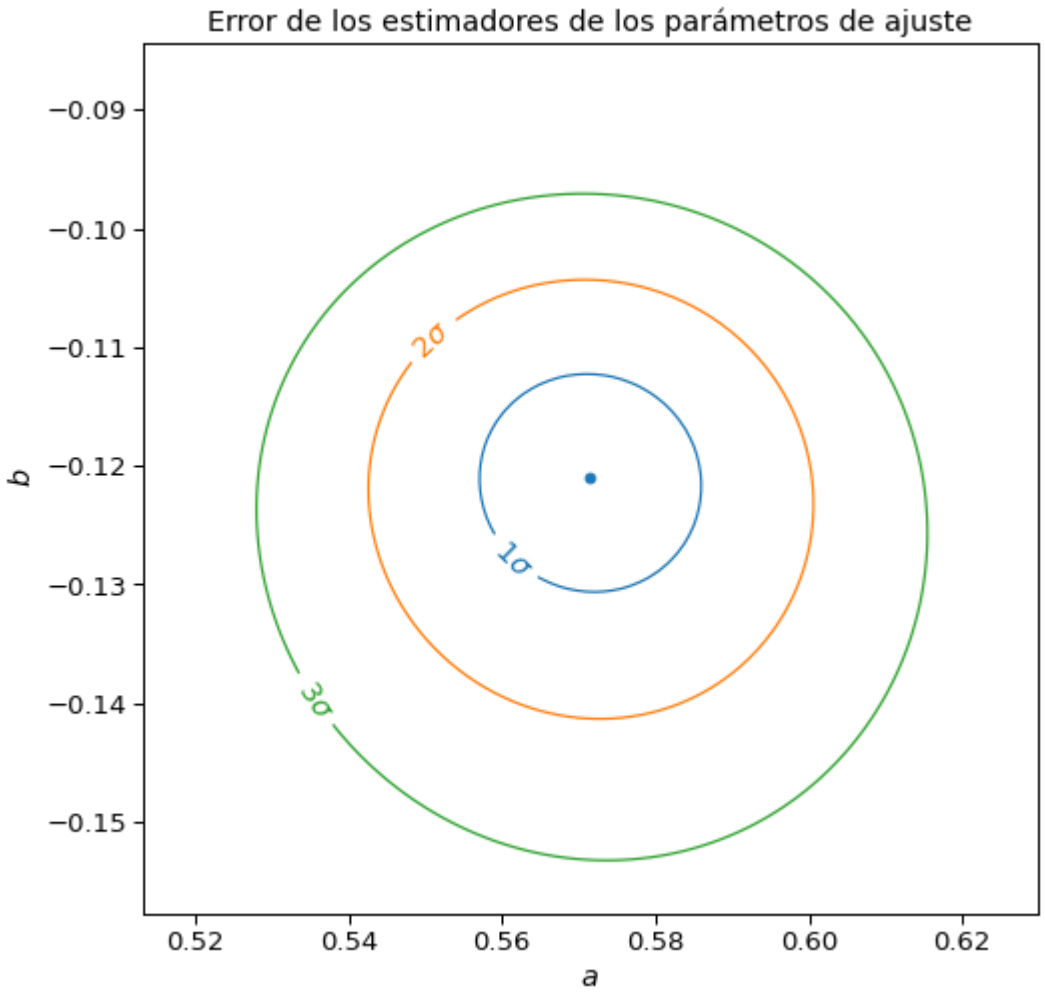
```
    sns.set_context("paper", font_scale=1.5)
    plt.plot(xs, ps - modelo(theta=res.x, x=xs), "o", label="Residuos")
    plt.plot(x_est, np.zeros_like(x_est), label="Ajuste")
    plt.fill_between(
        x_est, -sigma_mu_est, sigma_mu_est,
        color="tab:orange", alpha=0.2, label="Error del ajuste",
    )
    plt.xlabel(r"Variable independiente $x$")
    plt.ylabel(r"Diferencia con el ajuste de $p$")
    plt.title("Error del ajuste de la eficiencia")
    plt.legend()
    plt.savefig("tp7_punto8_residuos.pdf", bbox_inches="tight")
    plt.show()
    plt.close()


    theta1 = np.linspace(
        theta_est[0] - 4.0 * error_est[0],
        theta_est[0] + 4.0 * error_est[0],
        100,
    )
    theta2 = np.linspace(
        theta_est[1] - 4.0 * error_est[1],
        theta_est[1] + 4.0 * error_est[1],
        100
    )
    theta1, theta2 = np.meshgrid(theta1, theta2)
    z = np.vectorize(lambda a, b: J((a, b)))(theta1, theta2)

    plt.figure(figsize=(8, 8))
    sns.set_context("paper", font_scale=1.5)
    plt.plot(*theta_est, "o")
    levels = chi2_min + np.array([1, 4, 9])
    contour = plt.contour(
        theta1, theta2, z, levels,
        colors=["tab:blue", "tab:orange", "tab:green"]
    )
    fmt = {}
    level_labels = [r"$1\sigma$", r"$2\sigma$", r"$3\sigma$"]
    for l, s in zip(contour.levels, level_labels):
        fmt[l] = s
    plt.clabel(contour, fmt=fmt)
    plt.xlabel(r"$a$")
    plt.ylabel(r"$b$")
    plt.title("Error de los estimadores de los parámetros de ajuste")
    plt.savefig("tp7_punto8_costo.pdf", bbox_inches="tight")
    plt.show()
    plt.close()


if __name__ == "__main__":
    punto8()
```

```
      fun: 24.010555590050185
hess_inv: array([[ 1.06051928e-04, -2.43833494e-06],
       [-2.43833494e-06,  4.19878575e-05]])
      jac: array([ 5.72204590e-06, -7.15255737e-07])
  message: 'Optimization terminated successfully.'
     nfev: 45
      nit: 10
     njev: 15
   status: 0
```

```
  success: True
        x: array([ 0.57149282, -0.1209944 ])
a = 0.571 +- 0.015
b = -0.121 +- 0.009
ρ = -0.04
chi-cuadrado min = 24.01
ndof = 18
pvalor = 15.5%
```




Error del ajuste de la eficiencia
Residuos
Ajuste
Error del ajuste
0.15
0.10
0.05
0.00
−0.05
−0.10
Diferencia con el ajuste de p
0.2
0.4
0.6
0.8
1.0
Variable independiente x


Error de los estimadores de los parámetros de ajuste
−0.09
−0.10
−0.11
−0.12
−0.13
−0.14
−0.15
b
0.52
0.54
0.56
0.58
0.60
0.62
a
1σ
2σ
3σ