Statistika Nejpopulárnějších písniček Spotify

- **Best Songs on Spotify from 1996-2023:**
  Tabulka s nasledujícími sloupci:
  title, artist, top genre, year, bpm, energy, danceability, dB, liveness, valence, duration, acousticness, speechiness, popularity

> Definice některých pojmů v angličtině:

> Data byla získana z webu kaggle.

Popis dat:

- **Danceability**: Describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.
- **Valence**: Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
- **Energy**: Represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale.
- **bpm (tempo)**: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece, and derives directly from the average beat duration.
- **Speechiness**: This detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
- **Liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
- **Acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.

In [6]:
```
import sys
#!{sys.executable} -m pip install numpy
#!{sys.executable} -m pip install pandas
#!{sys.executable} -m pip install plotly
#!{sys.executable} -m pip install matplotlib
#!{sys.executable} -m pip install scipy
#!{sys.executable} -m pip install scikit-learn
#!{sys.executable} -m pip install pingouin

#import pingouin
import numpy
#import plotly
import matplotlib.pyplot as plt
```

```
import pandas as pd
import os
import scipy.stats as stats
import statsmodels.api as sm
#from sklearn.linear_model import LinearRegression
```

Načtení dat

In [8]:
```
table = pd.read_csv('Songs.csv', delimiter=';')
table.head()
#Pokud chcete zobrazit celou tabulku, odkomentujte řádek níže
#pingouin.print_table(table)
```

Out[8]:

	title	artist	top genre	year	bpm	energy	danceability	dB	liveness	val
0	Flowers	Miley Cyrus	pop	2023	118	68	71	-4	3	
1	Cupid - Twin Ver.	FIFTY FIFTY	k-pop girl group	2023	120	59	78	-8	35	
2	BESO	ROSALÍA	pop	2023	95	64	77	-7	17	
3	Boy's a liar Pt. 2	PinkPantheress	bronx drill	2023	133	81	70	-8	25	
4	Creepin' (with The Weeknd & 21 Savage)	Metro Boomin	rap	2022	98	62	72	-6	8	

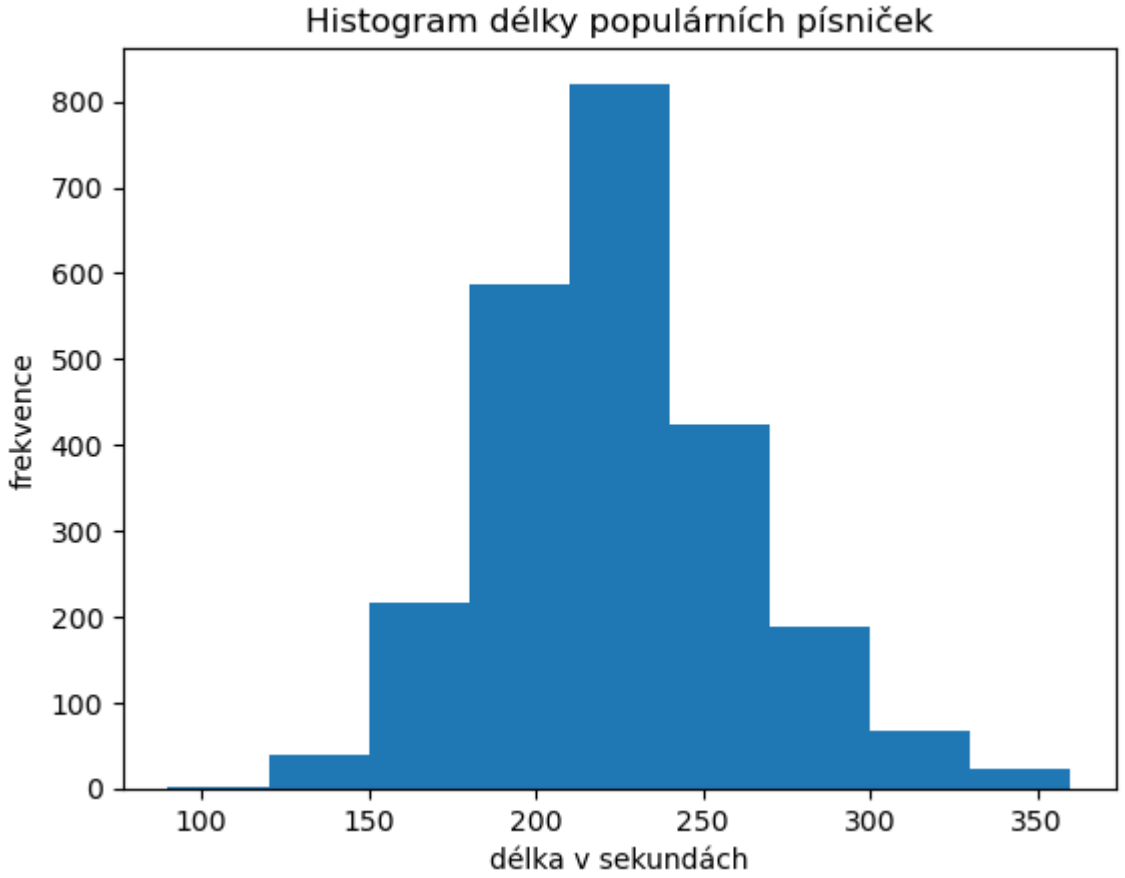
1. Délka písniček a normální rozdělení

Budeme zkoumat, zda je délka populárních písniček dobře modelována normálním rozdělením.
K tomu budeme používat chi-kvadrát test.

Nulová hypotéza: data jsou modelována normální rozdělením.
Alternativní hypotéza: data nejsou modelována normálním rozdělením.

Nejprve se podíváme na hostogram dat:

In [9]:
```
duration = table['duration'] #duration in seconds
fig, main_ax =plt.subplots()
main_ax.set_title('Histogram délky populárních písniček')
main_ax.set_xlabel('délka v sekundách')
main_ax.set_ylabel('frekvence')
plt.hist(duration, bins=[90, 120, 150, 180, 210, 240, 270, 300, 330, 360]
plt.show()
```



In [12]:
```
stats.normaltest(duration)
```

Out[12]:
```
NormaltestResult(statistic=1142.5234957008604, pvalue=8.020015103348854e
-249)
```

In [13]:
```
stats.normaltest?
```

Histogram celkem vypadá dobře. Takže by se ve skutečnosti mohlo jednat o normální rozdělení.

Teď spočteme průměr, rozptyl a směrodatnou odchylku:

In [1...
```
duration_mean = duration.mean()
duration_var = duration.var()
duration_std = numpy.std(duration)
print(f"Průměr = {duration_mean}; Rozptyl = {duration_var}; Směrodatná od
```

```
Průměr = 225.946750524109; Rozptyl = 1868.547918337484; Směrodatná odchylk
a = 43.21764061226389
```

Vytvoříme intervaly pro délky a spočteme, počet písniček v každém intervalu. Dále pro každý interval spočteme z-skóre. Z-skóre spočteme jako podíl, kde v čitateli je rozdíl horní hranice intervalu a průměru, a ve jmenovateli je směrodatná odchylka:

Intervaly: <120, <180, <240, <300, <360, >=360

In [1...
```
bounds = [120, 180, 240, 300, 360]
number_bounds = [0, 0, 0, 0, 0, 0]
number_bounds[0] = sum(duration < bounds[0])
number_bounds[len(bounds)] = sum(duration >= bounds[len(bounds)-1])
```

```
for i in range(1, len(bounds)):
    number_bounds[i] = sum(duration < bounds[i]) - sum(duration < bounds[
print("Počet písniček v každém intervalu: ", number_bounds)

z_score = [0, 0, 0, 0, 0, 0]
for i in range(0, len(bounds)):
    z_score[i] = (bounds[i]- duration_mean)/duration_std
print("Z-skóre pro každý interval: ", z_score)
```

```
Počet písniček v každém intervalu:  [2, 255, 1408, 614, 90, 16]
Z-skóre pro každý interval:  [-2.4514700252758463, -1.0631480542015215, 0.
32517391687280345, 1.7134958879471283, 3.101817859021453, 0]
```

Ze z-skóre spočítáme plochu pod křívkou normální distribuce N(0, 1):

In [1...
```
area_under = [0, 0, 0, 0, 0, 0]
for i in range(len(z_score)):
    area_under[i] = stats.norm.cdf(z_score[i])
area_under[len(area_under)-1] = 1
print("Plocha pod křívkou každého intervalu: ", area_under)
```

```
Plocha pod křívkou každého intervalu:  [0.007113701783713703, 0.1438574075
5201204, 0.6274752758867737, 0.9566893253931492, 0.9990383186982759, 1]
```

Dále spočítáme plochu v rámci jednotlivých intervalů:

In [1...
```
area_inside = [0, 0, 0, 0, 0, 0]
area_inside[0] = round(area_under[0], 8)

for i in range(1, len(area_under)):
    x = area_under[i] - area_under[i-1]
    area_inside[i] = round(x, 8)
print("Plocha uvnitř každého intervalu: ", area_inside)
```

```
Plocha uvnitř každého intervalu:  [0.0071137, 0.13674371, 0.48361787, 0.32
921405, 0.04234899, 0.00096168]
```

Plochu v rámci intervalů využijeme jako očekávané pravděpodobnosti pro chi-kvadrát test. Jenom pro chi-kvadrát test musíme ještě provést normalizaci dat tak, aby se součty očekavaných dat *(ta pravděpodobnost(`area_inside`) krát celkový počet písníček (`sum_elements`))* a pozorovaných dat (*`number_bounds`*) rovnaly.

In [1...
```
sum_elements = sum(number_bounds)

for i in range(len(area_inside)):
    area_inside[i] = area_inside[i] * sum_elements

stats.chisquare(number_bounds, f_exp=area_inside)
```

Out[1...
```
Power_divergenceResult(statistic=205.3276643003221, pvalue=2.05818307457
8902e-42)
```

Závěr:

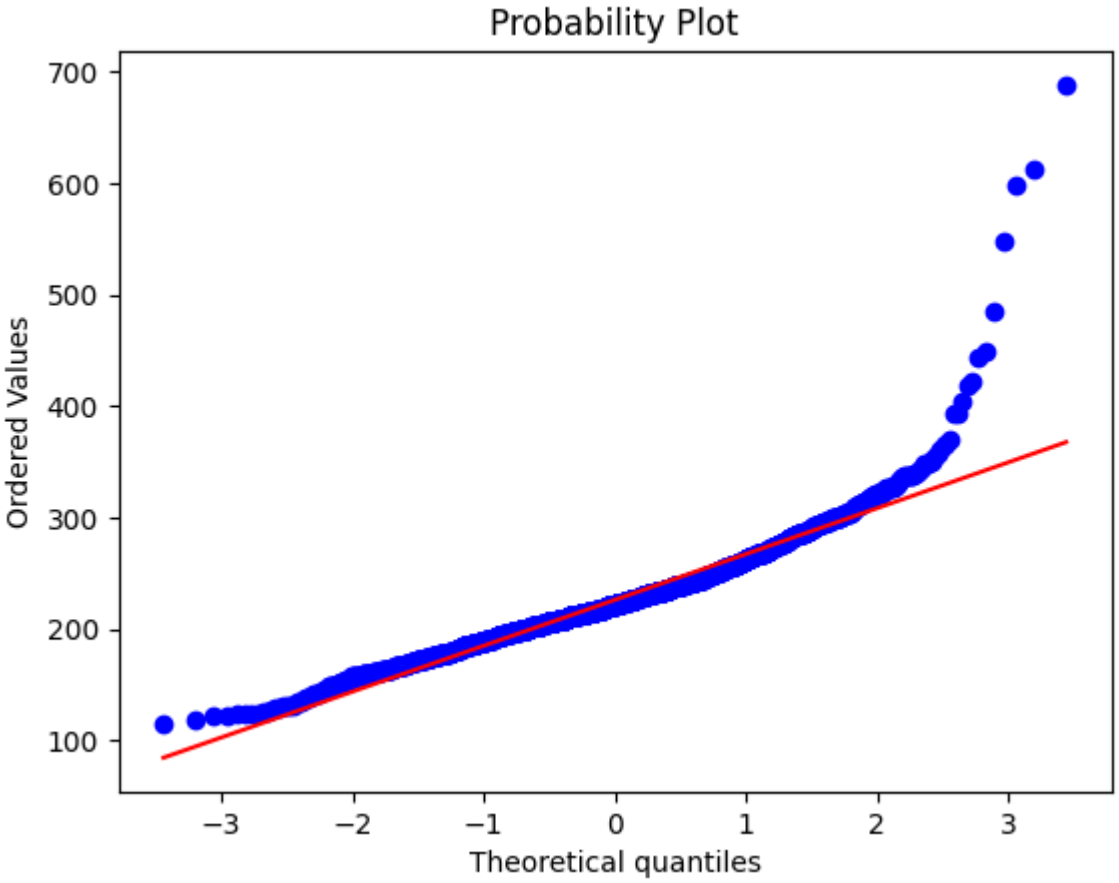
P-hodnota vyšla mnohem menší než 0,05, takže nulovou hypotézu musíme zamitnout a platí alternativní hypotéza: data nejsou modelovaná normálním rozdělením.

Alternativní způsoby, jak bychom mohli

postupovat.

- **Ještě je možné podívat se na Q-Q plot (quantile - quantile plot)**

In [1...
```
stats.probplot(duration, dist="norm", plot=plt)
plt.show()
```



Závěr:

Ačkoliv většina bodů je blízko přímce, je dost bodů, které leží daleko od přímky.

- **Shapiro-Wilkův test**

In [1...
```
res = stats.shapiro(duration)
print(res)
```

```
ShapiroResult(statistic=0.906105101108551, pvalue=5.1749929866740065e-36)
```

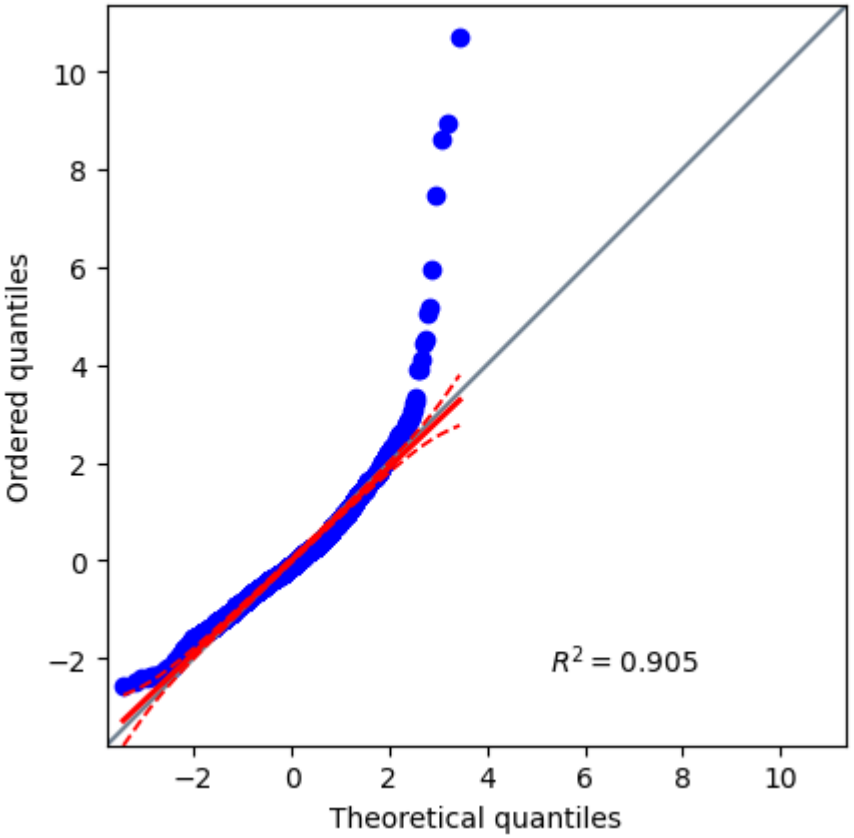
Závěr:

P-hodnota je zase menší než 0,05. Proto nulová hypotéza neplatí.

- **Ten samý qqplot, jenom z knihovny pingouin**

In [1...
```
print(pingouin.normality(duration))
ax = pingouin.qqplot(duration, dist='norm')
```

```
                 W          pval  normal
duration  0.906105  5.174993e-36   False
```



Závěr:

P-hodnota je znovu menší než 0,05. A ještě k tomu máme proměnnou `normal`, která říká, zda je rozdělení normální.

2. Závislost mezi popularitou a speechiness písniček

Zde budeme zkoumat, zda existuje vztah mezi popularitou písniček a jejich speechiness. Očekáváme, že čím větší je popularita, tím menší je speechiness. Provedeme lineární regresi.

Nulová hypotéza: mezi popularitou a speechiness není lineární vztah.
Alternativní hypotéza: mezi popularitou a speechiness existuje lineární vztah.

Na začátku načteme a přeformujeme data:

In [15]:
```
popularity = table['popularity']
speechiness = table['speechiness']

x = popularity.values.reshape(-1, 1)
y = speechiness.values.reshape(-1, 1)
```

In [46]:
```
A = table.query("`speechiness` > 10")
B = table.query("`speechiness` <= 10")

a = A['popularity'].values
b = B['popularity'].values
```

In [47]:
```
stats.ttest_ind(a,b)
```

Out[47]:
```
Ttest_indResult(statistic=-3.5719915638650224, pvalue=0.0003612981272167
1543)
```

In [48]:
```
a.mean()
```

Out[48]:
```
69.2273381294964
```

In [49]:
```
b.mean()
```

Out[49]:
```
71.06094674556213
```

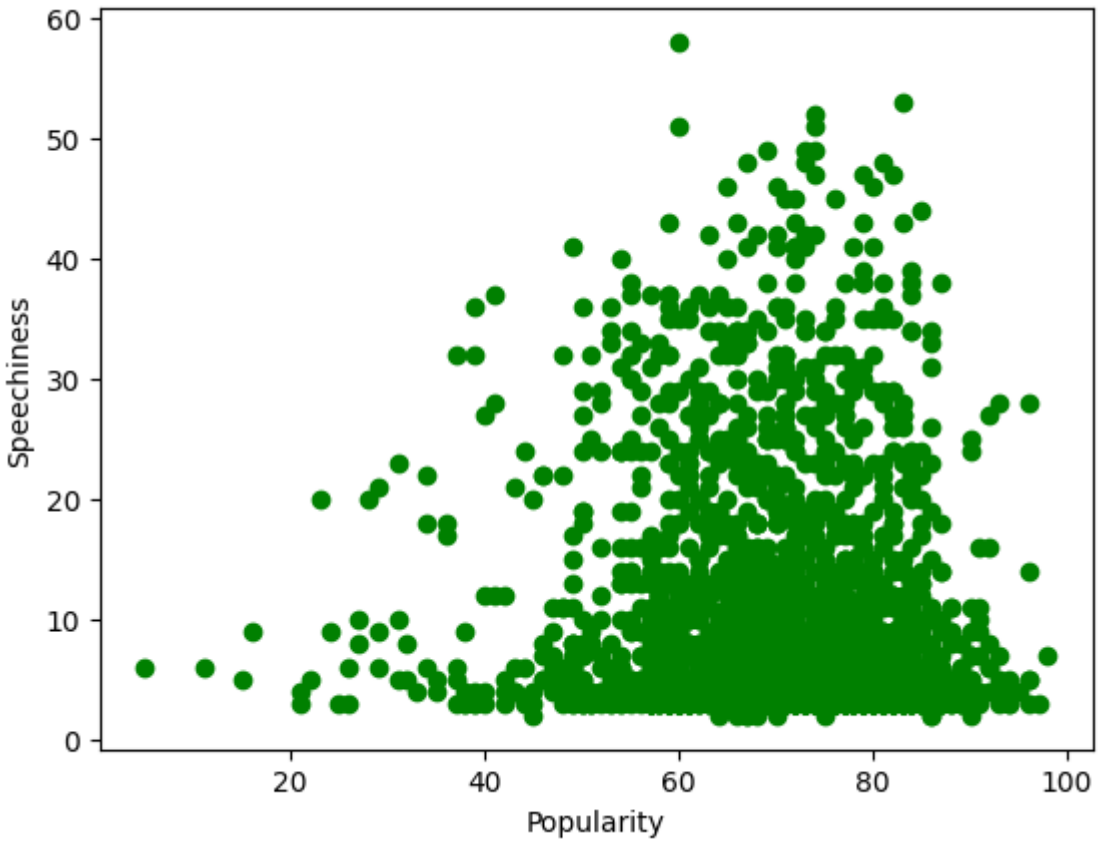
In [ ]:
```

```

Podíváme se, jak data vypadají.

In [1...
```
fig, ax = plt.subplots()
ax.set_ylabel('Speechiness')
ax.set_xlabel('Popularity')

plt.scatter(x, y, color="green")
plt.show()
```



Poté spočítáme průměr **speechiness** a průměr **popularity**.

In [1...
```
speechiness_mean = speechiness.mean()
popularity_mean = popularity.mean()
print(f'Průměr speechiness = {speechiness_mean}, \nPrůměr popularity = {p
```

```
Průměr speechiness = 10.112368972746332,
Průměr popularity = 70.5266247379455
```

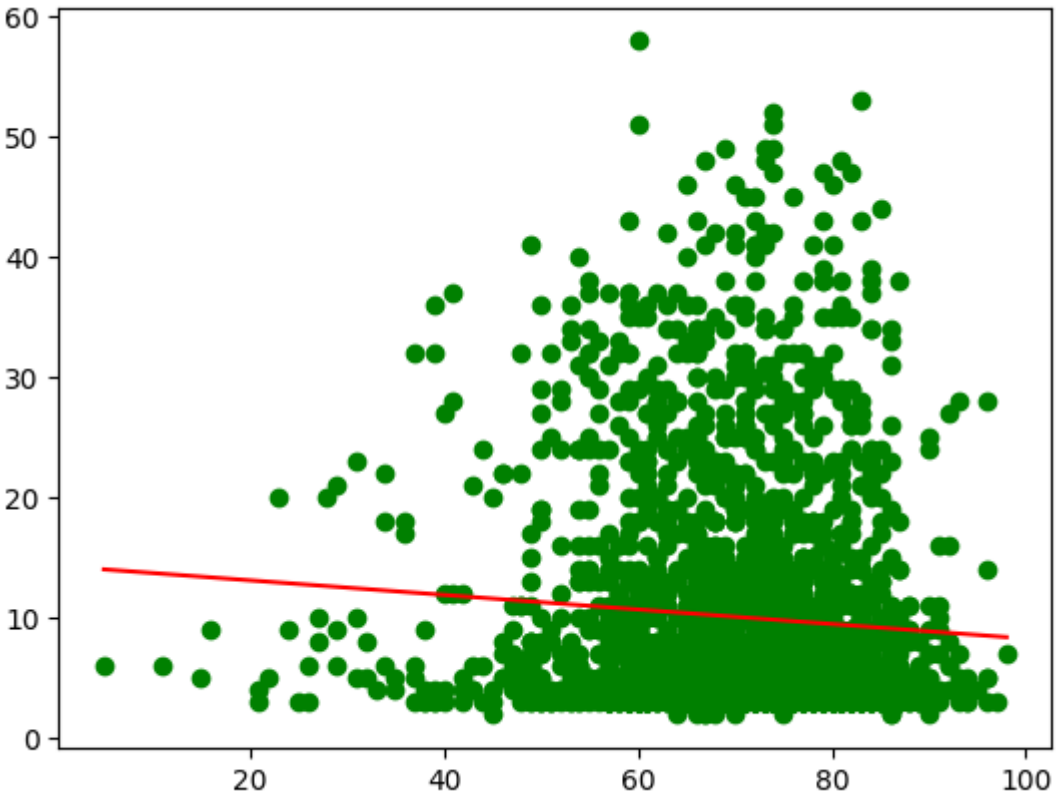
Pro přímku potřebujeme a a b. Spočíteme je podle informce z poslední přednašky:

In [1...
```
sum_popularity = 0
a = 0
for i in range(len(speechiness)):
    a += (popularity[i] - popularity_mean)*(speechiness[i] - speechiness_
    sum_popularity  += (popularity[i]-popularity_mean)**2
a = a/sum_popularity
b = speechiness_mean - a*popularity_mean
print("The linear model is: Y = {:.5} + {:.5}*X".format(b, a))
```

```
The linear model is: Y = 14.385 + -0.060586*X
```

Teď vyneseme přímku do grafu:

In [2...
```
plt.scatter(x, y, color="green")
plt.plot(x, a*x+b, c='red')
plt.show()
```



Pro to, abychom zjistili p-hodnotu, musíme dostat výsledky regrese následujícím způsobem: uděláme **popularitu** konstantou a provedeme OLS regresi. Částečně informace byla získána zde.

In [2...
```
X2 = sm.add_constant(x)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
```

```
print("P-hodnota je ", est2.pvalues[1])
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                      y   R-squared:
0.005
Model:                            OLS   Adj. R-squared:
0.005
Method:                 Least Squares   F-statistic:                     1
2.65
Date:                Sat, 08 Jul 2023   Prob (F-statistic):           0.00
0383
Time:                        15:10:26   Log-Likelihood:                -87
51.9
No. Observations:                2385   AIC:                          1.751
e+04
Df Residuals:                    2383   BIC:                          1.752
e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const         14.3853      1.217     11.821      0.000      11.999      1
6.772
x1            -0.0606      0.017     -3.557      0.000      -0.094      -
0.027
==============================================================================
====
Omnibus:                      826.731   Durbin-Watson:
2.002
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             230
4.508
Skew:                           1.852   Prob(JB):
0.00
Kurtosis:                       6.077   Cond. No.
447.
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
P-hodnota je  0.0003826317979757927
```

Závěr:

Jelikož p-hodnota vyšla menší než 0,05, musíme nulovou hypotézu zamitnout a tedy platí, že lineární vztah mezi popularitou a speechiness existuje. A z grafu můžeme vidět, že populárnější jsou ty písničky, které mají nižší speechiness. Informace částečně byla získána zde.

Alternativní způsoby, jak bychom mohli

postupovat.

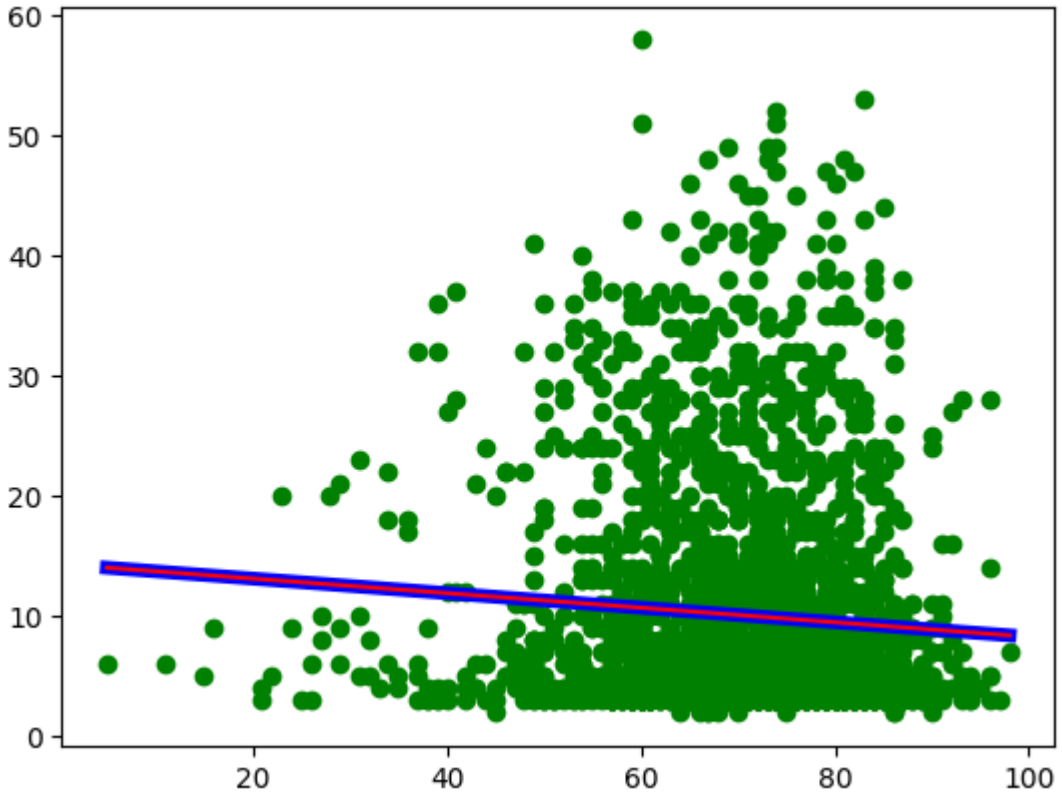
- **Můžeme ještě ověřit, jestli jsme správně spočitali přímku pomocí `LinearRegression` funkce z `sklearn.linear_model` knihovny.**
  Podíváme se, zda máme správně vypočty a nakreslíme dvě přímky: červená - náše vypočty, modrá - z knihovny.

In [2...
```
reg = LinearRegression().fit(x, y)
print("Lineární model: Y = {:.5} + {:.5}X".format(reg.intercept_[0], reg.

predictions = reg.predict(x)
plt.scatter(x, y, color="green")

plt.plot(
    x,
    predictions,
    c='blue',
    linewidth=5
)
plt.plot(x, a*x+b, c='red')
plt.show()
```

```
Lineární model: Y = 14.385 + -0.060586X
```



- **Ještě můžeme provést test lineární regrese pomocí knihovny `pingouin` .**

In [2...
```
res = pingouin.linear_regression(popularity, speechiness)
print(res)
```

```
        names       coef        se          T          pval        r2  \
0   Intercept  14.385303  1.216961  11.820674  2.255219e-31  0.005281  \
1  popularity  -0.060586  0.017034  -3.556862  3.826318e-04  0.005281

     adj_r2   CI[2.5%]  CI[97.5%]
0  0.004864  11.998891  16.771716
1  0.004864  -0.093988  -0.027184
```

3. Závislost mezi popularitou a danceability písniček

Zde budeme zkoumat, zda existuje vztah mezi popularitou písniček a jejich danceability. Očekáváme, že čím větší je popularita, tím větší je danceability. Provedeme lineární regresi.

Nulová hypotéza: mezi popularitou a danceability není lineární vztah.
Alternativní hypotéza: mezi popularitou a danceability existuje lineární vztah.

Zde nebudeme počítat rovnici přímky ručně, použijeme funkci `LinearRegression` z `sklearn.linear_model`:

Na začátku načteme a přeformujeme data:

In [2...
```
popularity = table['popularity']
dance = table['danceability']

x = popularity.values.reshape(-1, 1)
y = dance.values.reshape(-1, 1)
```

Vytvoříme model a najdeme rovnici přímky:

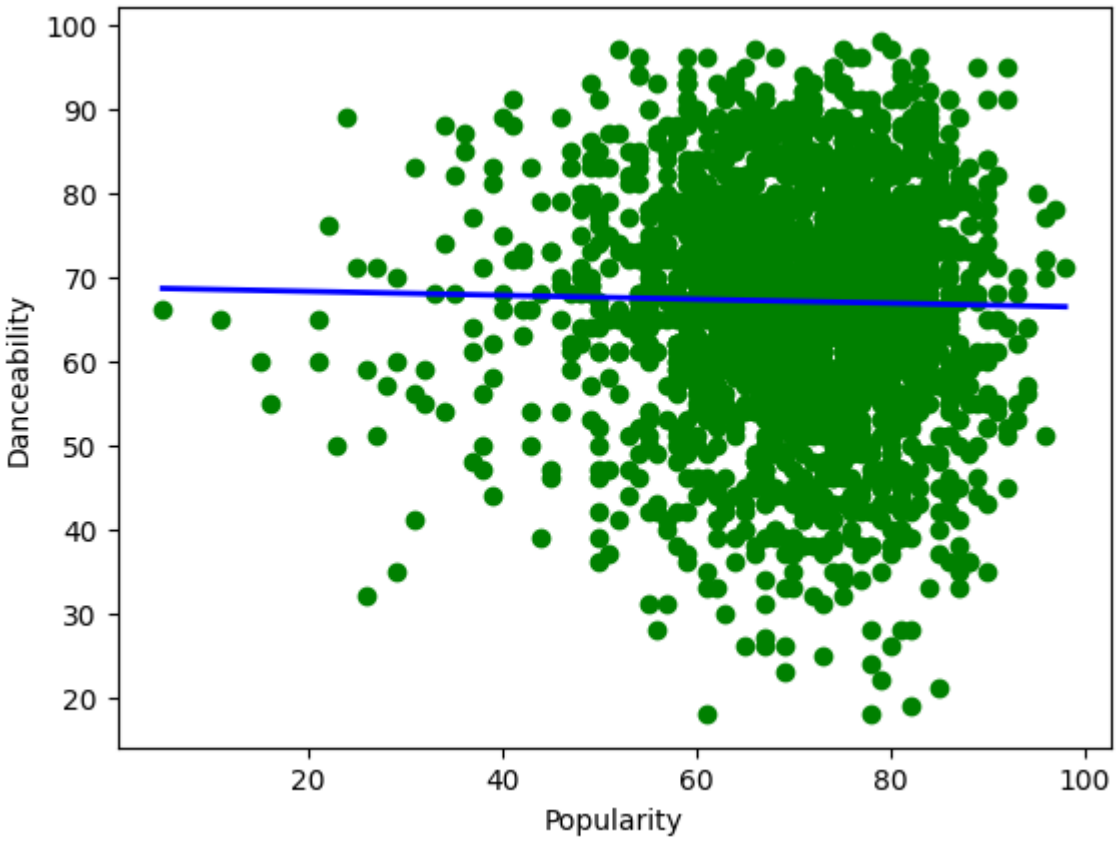
In [2...
```
reg = LinearRegression().fit(x, y)

fig, ax = plt.subplots()
ax.set_ylabel('Danceability')
ax.set_xlabel('Popularity')

plt.scatter(x, y, color="green")
print("Lineární model: Y = {:.5} + {:.5}X".format(reg.intercept_[0], reg.

predictions = reg.predict(x)
plt.plot(
    x,
    predictions,
    c='blue',
    linewidth=2
)
plt.show()
```

```
Lineární model: Y = 68.686 + -0.023415X
```



Teď najdeme p-hodnotu stejně jako v předchozím příkladě:

In [2...

```
X2 = sm.add_constant(x)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
print("P-hodnota je ", est2.pvalues[1])
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                      y   R-squared:
0.000
Model:                            OLS   Adj. R-squared:                -
0.000
Method:                 Least Squares   F-statistic:                    0.
9056
Date:                Sat, 08 Jul 2023   Prob (F-statistic):
0.341
Time:                        15:10:26   Log-Likelihood:                -96
29.0
No. Observations:                2385   AIC:                           1.926
e+04
Df Residuals:                    2383   BIC:                           1.927
e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const         68.6862      1.758     39.073      0.000      65.239      7
2.133
x1            -0.0234      0.025     -0.952      0.341      -0.072
0.025
==============================================================================
====
Omnibus:                       62.950   Durbin-Watson:
2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               6
7.503
Skew:                          -0.412   Prob(JB):                     2.20
e-15
Kurtosis:                       3.037   Cond. No.
447.
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
P-hodnota je  0.34137672571232536
```

Závěr:

Jelikož p-hodnota vyšla větší než 0,05, nulovou hypotézu numůžeme zamitnout. Z toho můžeme udělat závěr, že mezi danceability a popularitou není žádný lineatní vztah.

4. (bpm) Tempo hudby

Zde budeme zkoumat zda nějak liší střední hodnoty tempa písniček v prvním a ve druhém desetiletí 21. věku. K tomu použijeme dvouvýběrový test.

Nulová hypotéza: střední hodnoty dvou desetiletí se neliší.
Alternativní hypotéza: střední hodnoty se liší.

Na začátku připrávíme data:

In [2...
```
zeroes = []
tens = []
years = table['year']
bpms = table['bpm']
for i in range(len(years)):
    if years[i] > 2009:
        tens.append(bpms[i])
    else:
        if years[i] > 1999:
            zeroes.append(bpms[i])
```

Teď provedeme test pomocí funkce z knihovny `scipy.stats` :

In [2...
```
result = stats.ttest_ind(zeroes, tens, equal_var=False, alternative='two-
print(result)
```

```
Ttest_indResult(statistic=-0.926464309439658, pvalue=0.3543207988658451)
```

Závěr:

Jelkož p-hodnota vyšla větší než hladina významnosti testu 0,05, přijímáme nulovou hypotézu. Z toho můžeme udělat závěr, že tempo hudby je celkem stejné pro obě desetiletí.

Alternativní způsoby, jak bychom mohli postupovat.

- **Můžeme znovu použít knihovnu `pingouin`** . P-hodnotu budeme samozřejmě mít stejnou, ale knihovna ještě počítá intervalový odhad, pokud jej potřebujeme. Pomocí tohoto intervalu lze zjistir, že s pravděpodobností 95% se rozdíl střední hodnoty tempa 1. desetiletí a 2. se nachází v tomto intervalu (sloupec CI95%):

In [2...
```
res = pingouin.ttest(zeroes, tens)
print(res)
```

```
               T          dof alternative     p-val          CI95%    cohen
-d
T-test -0.926464  1926.210783   two-sided  0.354321  [-3.37, 1.21]  0.0395
89  \

         BF10     power
T-test  0.073  0.155471
```

5. Valence písniček

V posledním příkladu budeme zkoumat, zda se valence hudby na začátku 21. století nějak liší od valence hudby posledních let. Roky pro porovnání jsou: 2001-2005 a

2019-2023. Pro to zase použijeme dvouvýběrový test.

Nulová hypotéza: střední hodnoty se neliší.
Alternativní hypotéza: střední hodnoty se liší.

Na začátku připrávíme data:

In [2...
```
before = []
now = []
valence  = table['valence']

for i in range(len(years)):
    if 2000 < years[i] < 2006:
        before.append(valence[i])
    elif 2018 < years[i] < 2024:
        now.append(valence[i])
```

Teď provedeme testy pomocí funkce z knihovny `scipy.stats` a pomocí knihovny `pigouin` :

In [2...
```
result = stats.ttest_ind(before, now, equal_var=False, alternative='two-s
print(result)

res = pingouin.ttest(before, now)
print(res)
print(f'\nStřední hodnota valence písniček posledních let = {numpy.mean(n
```

```
Ttest_indResult(statistic=7.633226890107983, pvalue=5.987604487666038e-14)
               T         dof alternative         p-val         CI95%
T-test  7.633227  874.607766   two-sided  5.987604e-14  [8.51, 14.4]  \

         cohen-d       BF10  power
T-test  0.509788  9.332e+10    1.0

Střední hodnota valence písniček posledních let = 49.17761557177616,
Střední hodnota valence písniček na začátku 21. století = 60.6361788617886
15
```

Závěr:

P-hodnota je mnohem menší než je hladina významnosti testu 0,05. Proto nulovou hypotézu zamítáme a můžeme udělat závěr, že teď je hudba není tak pozitivní, jak byla na začátku století. A ještě stejně jako v předchozím příkladu máme intervalový odhad, kde s pravděpodobností 95% leží rozdíl středních hodnot valence hodby.