Análise estatística univariada

Packages modelstats e algumas funcionalidades do pandas e do scikit-learn

Sumário

- Análise estatística univariada
  - principais testes estatísticos
- Filtragem de datasets
  - Por métricas não supervisionadas
  - Por métricas supervisionadas
- Packages: *pandas*, *statsmodels*, *scikit-learn*

Testes estatísticos

Packages para análise estatística

- ***statsmodels*** - módulo Python que contém classes e funções que permitem a estimação de modelos estatísticos, realizar testes estatísticos e exploração de tratamento de dados.
  - Dependências:
    - pandas
    - patsy
  - Documentação: http://www.statsmodels.org/stable/index.html
- Módulo scipy.stats inclui diversas funções para análise estatística e alguns testes
- Algumas funções úteis para análise estatística podem ser corridas diretamente do *pandas*

Normalidade dos dados

- Importância de saber se os dados seguem uma distribuição normal para algumas análises

- Para verificar visualmente se os dados seguem a distribuição normal, além dos histogramas, podemos usar a função :

```
qqplot(data, dist=stats.norm, distargs=(),a=0, loc=0, scale=1,
fit=False,line=False, ax=None)
```

presente no package **statsmodels**

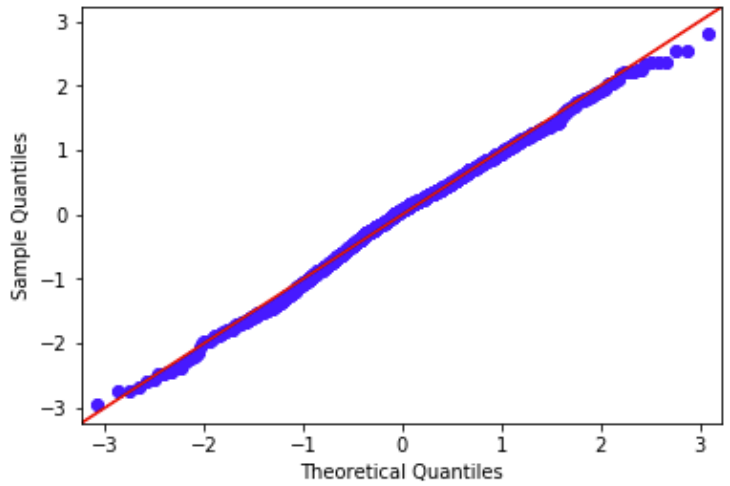
QQ plots - exemplo

- Gráfico Q-Q é um gráfico de probabilidades, usado comparar duas distribuições de probabilidade, traçando seus quantis uns contra os outros.

```
import numpy as np
import statsmodels.api as sm

test = np.random.normal(0,1, 1000)

sm.qqplot(test, line='45')
```



- Se as duas distribuições são semelhantes, os pontos no gráfico Q-Q vai repousar na linha y = x, aproximadamente

https://www.youtube.com/watch?v=X9_ISJ0YpGw

Testes estatísticos

- Baseados em intervalos de confiança
- Principal resultado do teste é o ***p-value***
- Valores pequenos permitem rejeitar a H0 com um grau de confiança de (1-p)
- Se H0 não for rejeitada o teste é inconclusivo
- Tipo de teste:
  - paramétricos : assume que os dados seguem a distribuição normal
  - não-paramétricos: não impõem nenhuma restrição à distribuição dos dados

Ver alguns de vários tipos de testes em Python:
https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/

Teste à normalidade

- Teste de Shapiro:
  - H0: dados seguem a distribuição normal
  - Função:
    - `scipy.stats.shapiro(x, a=None, reta=False)`

```
from scipy import stats

x = np.random.normal(0,1,1000)
w, p_value = stats.shapiro(x)
p_value
```

H0 not rejected

```
from scipy import stats

x = np.random.random(1000)
w, p_value = stats.shapiro(x)
p_value
```

H0 rejected

Teste à média (t-test)

- H0: o valor médio entre os valores de duas variáveis é igual
- Função em Python:
  - `scipy.stats.ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate'`
  - `statsmodels.stats.ttest_ind(x1, x2, alternative='two-sided', usevar='pooled', weights=(None, None), value=0)`

```
diasChuva16= [15, 10, 13, 7, 9, 8, 21, 9, 14, 8, 17, 15]
diasChuva17 = [15, 14, 12, 8, 14, 7, 16, 10, 15, 12, 16, 15]

stats.ttest_ind(diasChuva16, diasChuva17)

Out:
Ttest_indResult(statistic=-0.43409311309339899,
pvalue=0.66844831736662758)
```

Outros testes estatísticos

- Mann-Whitney U test
  - teste não paramétrico – usa a mediana
  - função: `scipy.stats.mannwhitneyu(x, y, use_continuity=True, alternative=None)[source]`
- Wilcoxon test
  - teste não paramétrico (dados emparelhados) – usa a mediana
  - função: `scipy.stats.wilcoxon(x, y=None, zero_method='wilcox', correction=False)`
- Chi-square test
  - testa se a amostra pertence a uma dada população
  - aplicado a dados discretos;
  - função: `scipy.stats.chisquare(f_obs, f_exp=None, ddof=0, axis=0)`
- Kolmogorov-Smirnov test
  - verifica se os dados seguem a distribuição teórica
  - testa se duas amostras provêm da mesma distribuição
  - função: `scipy.stats.kstest(rvs, cdf, args=(), N=20, alternative='two-sided', mode='approx')`

Análise à variância (one-way)

- Usado para testar se a média em vários grupos de observações é a mesma
- Assume que os dados seguem a distribuição normal
- H0: a média de cada grupo é a mesma
- ANOVA:
  - `scipy.stats.f_oneway`
  - `statsmodels.stats.api.anova_lm`

Análise à variância/ regressão linear

```
iris = pd.read_csv('iris.csv')
iris=iris.iloc[:,1:]
iris.columns =["Sepal_Length","Sepal_Width","Petal_Length",
                    "Petal_Width","Species"]
g1 = iris[iris["Species"]=="setosa"]
g2 = iris[iris["Species"]=="virginica"]
g3 = iris[iris["Species"]=="versicolor"]
stats.f_oneway(g1.Petal_Length,g2.Petal_Length,g3.Petal_Length)

Out:
F_onewayResult(statistic=1180.161182252981, pvalue=2.8567766109615584e-
91))
```

```
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm

lm = ols("iris.Petal_Length~iris.Species", iris).fit()
anova_lm(lm)
```

Out[104]:

	df	sum_sq	mean_sq	F	PR(>F)
iris.Species	2.0	437.1028	218.551400	1180.161182	2.856777e-91
Residual	147.0	27.2226	0.185188	NaN	NaN

Post-hoc tests e testes múltiplos

- **Tukey's HSD test:** quando usado juntamente com a ANOVA permite identificar pares com médias significativamente diferentes.
- **MultiComparison -** package *statsmodels*: objeto com os dados para efetuar testes múltiplos (`tukeyhsd`)

```
import statsmodels.stats.multicomp as multi
x = np.random.choice(['A','B','C'], 50)
y = np.random.rand(50)
mcDate = multi.MultiComparison(y,x)
Results = mcDate.tukeyhsd()
print(Results)
```

```
Multiple Comparison of Means - Tukey HSD,FWER=0.05
=============================================
group1 group2 meandiff  lower  upper  reject
---------------------------------------------
  A      B    -0.0089  -0.2408 0.223   False
  A      C     0.0329  -0.2358 0.3016  False
  B      C     0.0419  -0.2344 0.3181  False
---------------------------------------------
```

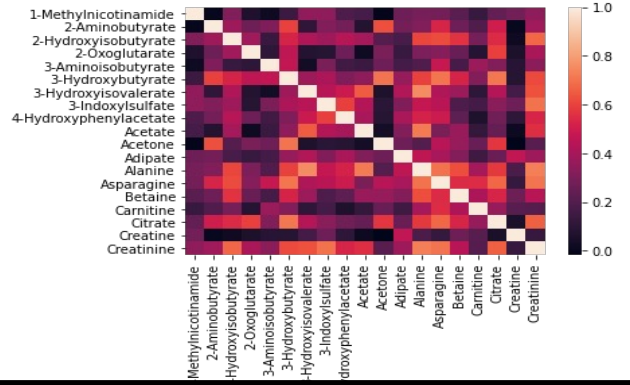
- Função `statsmodels.stats.multitest.multipletests` permite fazer a correção de p-values para testes múltiplos

Análise à variância

- Quando os dados não seguem a distribuição normal, devemos usar testes não paramétricos
- **Kruskal-Wallis test**: teste não paramétrico de análise à variância.
- Função:
  - `multiComparison.kruskal` (package statsmodels)
  - `scipy.stats.kruskal`

Correlações

- Podemos usar o módulo *stats* para calcular correlações:
  - **Pearson** (paramétrico) – assume relações lineares
  - **Spearman** (não paramétrico) – relacionado com ordenação dos valores (monotonia)
- Função para calcular a matriz de correlações entre cada uma das variáveis de um *DataFrame* em *pandas*: `df.corr()`
  - Retorna um DataFrame – matriz nº variáveis x nº variáveis
  - Permite indicar o método como argumento ("pearson" – omissão; "spearman")
- Visualização – usando heatmaps



Redução/ filtragem de variáveis

- Em muitos casos, há necessidade/ vantagens em fazer uma redução do nº de variáveis de um conjunto de dados
- Esta pode ser feita tendo em conta apenas o conteúdo das diversas variáveis do conjunto de dados e métricas calculadas – não supervisionado
  - Variabilidade dos dados (remover as variáveis que variam menos)
  - Remover variáveis muito correlacionadas
- Em alternativa, pode-se considerar outra variável externa (e.g. metadados) como referências para calcular as métricas a usar - supervisionado
  - Testes estatísticos univariados – e.g. ANOVA/ T-test/ testes não paramétricos, se a variável externa for nominal
  - Análise de regressão linear – se for contínua