U LISBOA UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR DE AGRONOMIA

ESTATÍSTICA 2024/25

Guia de conversão R ↔ Python

```
import os
       pandas as pd
       numpy as np
       statistics as stat
       matplotlib.pyplot as plt
       seaborn as sns
```

Categoria	R	Python
path	`setwd(path)`	`os.chdir(path)`
	`getwd()`	`os.getcwd()`
ficheiros	`read.csv(path_to_csv_file)`	`pd.read_csv(path_to_csv_file)`
	`write.csv(df, path_to_file, sep=",")`	`df.to_csv(path_to_file, sep=",")`
criar data frame df	`df <- data.frame(a = c(1, 2, 3, 4), b = c(5, 6, 7, 8))`	`df = pd.DataFrame({'a': [1,2,3,4], 'b': [5,6,7,8]})`
características da data frame df	`str(df)`	`df.dtypes / df.info()`
	`dim(df)`	`df.shape`
	`NROW(df)`	
	`NCOL(df)`	
ver os dados	`head(df)`	`df.head()  /  df.head(n)`
	`tail(df)`	`df.tail()  /  df.tail(n)`
	`summary(df)`	`df.describe()`
concatenar por linha	`rbind(df_1, ..., df_n)`	`pd.concat([df_1, ..., df_n], axis=0)`
concatenar por coluna	`cbind(df_1, ..., df_n)`	`pd.concat([df_1, ..., df_n], axis=1)`
ordenar as linhas da data frame df por odem crescente da coluna k	`df[order(df[,k], decreasing = FALSE), ]`	`df.sort_values(by=k, ascending=True)`
eliminar linhas repetidas	`unique(df)`	`df.drop_duplicates()`
eliminar linhas com NA	`na.omit(df)`	`df.dropna()`
valor lógico verdade	`TRUE  /  T`	`True`
valor lógico falso	`FALSE  /  F`	`False`
extrair a coluna "a" da data frame df	`df["a"]  é um objeto do tipo vector`	`df['a'] / df.a  é um objeto do tipo series`
selecionar linhas e colunas usando índices	`df[c(1,2), 2]`	`df.iloc[[0,1], 1]`
selecionar linhas e colunas usando condições e/ou nomes	`df[ df["a"] > 1,  "b"]`	`df.loc[ df.a > 1, 'b']`
Indicadores numéricos unidimensionais	`x <- vetor numérico`	`x = [ lista ]`
	`min(x)`	`min(x)`
	`max(x)`	`max(x)`
	`length(x)`	`len(x)`
	`mean(x)`	`stat.mean(x)`
	`var(x)`	`stat.variance(x)`
	`sd(x)`	`stat.stdev(x)`
	`median(x)`	`stat.median(x)`
	`quantiles(x)`	`stat.quantiles(x)`
		`stat.mode(x)`

Indicadores numéricos bidimensionais	`y <- vetor numérico`	`y = [ lista ]`
	`cov(x, y)`	`stat.covariance(x,y)`
	`cor(x, y)`	`stat.correlation(x,y)`
	`lm(y ~ x)`	`slope, intercept = stat.linear_regression(x, y)`

diagrama de barras	`height <- c(3, 12, 5, 18, 45)`
	`bars <- c('A', 'B', 'C', 'D', 'E')`
	`barplot(height, names.arg=bars)`
		`height = [3, 12, 5, 18, 45]`
		`bars = ('A', 'B', 'C', 'D', 'E')`
		`y_pos = range(len(bars)) # criar array 0,…,4`
		`# criar as barras:`
		`plt.bar(y_pos, height)`
		`# acrescentar nomes:`
		`plt.xticks(y_pos, bars)`
		`plt.show()  # mostrar o gráfico`

histograma com n classes de amplitude constante	`hist(y, breaks=n,`
	`col='lightblue',`
	`border='gray',`
	`freq=TRUE)`
		`plt.hist(y, bins=n,`
		`color='lightblue',`
		`edgecolor='gray')`
		`plt.show()`
		
		ou
		
		`y_df = pd.DataFrame({'y':y})`
		`style.use('ggplot')`
		`y_df.hist(y, bins=n)`
		`plt.show()`

histograma com classes de amplitude variável	`hist(y, breaks=c(…),`
	`col='lightblue',`
	`border='gray',`
	`freq=FALSE)`
		`plt.hist(y, bins=[...],`
		`color='lightblue',`
		`edgecolor='gray',`
		`density=True)`
		`plt.show()`
		
		ou
		
		`y_df = pd.DataFrame({'y':y})`
		`style.use('ggplot')`
		`y_df.hist(y, bins=[...],`
		`edgecolor='gray'`
		`density=True)`
		`plt.show()`

boxplot	`boxplot(y)`	`# com o matplotlib.pyplot`
		`plt.boxplot(y)`
		`plt.show()`
		
		`# com o pandas`
		`y_df=pd.DataFrame({'y': y})`
		`y_df.boxplot()`
		`plt.show()`
		
		`# com o seaborn`
		`sns.boxplot(data=y_df)`
		`plt.show()`

nuvem de pontos	`plot(x, y)`	`plt.scatter(x, y)`
sobrepôr a reta de regressão	`abline(lm(y ~ x))`	`plt.plot(x, intercept+slope*x)`