LIVROS PARA ESTUDAR ARIMA

- Bíblia da coisa:

- BOX, George E.P.; JENKINS, G.M.; REINSEL, G.C.; Time series analysis: forecasting and control; *Wiley Series in Probability and Statistics*; John Wiley and Sons; 4ª edição; Hoboken; 2008

- Manual mais acessível e útil, com exemplos:

- PANKRATZ, Alan; Forecasting with univariate Box-Jenkins models: concepts and cases; *Wiley Series in Probability and Mathematical Statistics*; John Wiley and Sons; New York; 1983

DESCRIÇÃO DOS PROGRAMAS DE ARIMA

- PASTAS:

- **ARIMA_teste2**, inclui:
  1. **ARIMAteste2**: função principal, com acesso às restantes, que foi uma tentativa de implementação do ARIMA "à mão", seguindo o sugerido no manual de PANKRATZ, no capítulo 8 ("*Estimation*"), pág.192; desisti da ideia devido à necessidade de cálculo de um gradiente (derivada parcial) para cada coeficiente diferente, em cada caso de regressão ARIMA diferente;
  2. **marquardtTESTE**: função que tentava implementar o algoritmo de "Marquardt Compromise" (ou Levenberg-Marquardt), segundo o livro de PANKRATZ, a partir da pág.209; desisti disto, por causa do referido em cima;
  3. **ValuesResidues**: função que calcula f(x) de uma séries temporal, x, segundo os coeficientes já otimizados ao modelo ARIMA(p,d,q) previamente escolhido; após isto, a função calcula os resíduos entre os dados de treino, y, e os resultados obtidos, f(x). Contudo, não tenho a certeza se devolve os valores corretos;
- **Dataset Forecast**;
- **LMFnlsq**, inclui:
  1. Funções de otimização de coeficientes para equações não-lineares (como indicado para modelos ARIMA), segundo o algoritmo de Levenberg-Marquardt; parece ser poderosa, mas não consegui trabalhar com ela, porque pede a exposição clara da função a otimizar (como o ARIMA é autorregressivo, não consegui perceber a forma de fazer isto);
- **outros**, inclui:
  1. Outra função de otimização de coeficientes, por Levenberg-Marquardt; não consegui trabalhar com ela, pelas mesmas razões;
- **rstudio_cenas**, inclui:
  1. os programas de teste, para comparação dos resultados obtidos.

- PROGRAMAS "FINAIS"/"FUNCIONAIS":

1. **arima_MATLAB**:

   Implementação da técnica de ARIMA, utilizando as funções *built-in* do MATLAB adequadas.

2. **ARIMAforecast**:

Função que prevê valores futuros de determinado modelo ARIMA, previamente ajustado a dados de treino. *Inputs*: vetor B que inclui *ARcoefs* e *MAcoefs*, parâmetros *p*, *d* e *q*, dados de treino (*y*), resíduos do treino (*resid*), "desvio-padrão estimado dos resíduos (*random shocks*)" (*sigmahat*) e número de previsões (*numPrev*). *Outputs*: 1 vetor por cada uma das seguintes variáveis: previsões realizadas (*y_previsao*) e limites superior e inferior de 95% de confiança (*lowerConf95* e *upperConf95*).

Cada previsão depende de uma soma de 3 parcelas, que dizem respeito às 3 partes do modelo: AR, I e MA. Assim: **previsão = sumAR - sumI - sumMA**.

Isto segue a seguinte fórmula:

$$w_t = \phi_1 w_{t-1} + \cdots + \phi_p w_{t-p} + a_t - \theta_1 a_{t-1} - \cdots - \theta_q a_{t-q} \text{ , com}$$

$$w_t = (1-B)^d . z_t \text{ , sendo}$$

$$z_t = previsão\ a\ calcular, no\ instante\ "t",$$

$$w_t = polinómio\ dependente\ de\ "d", no\ instante\ "t",$$

$$a_t = resíduo\ ("random\ shock")\ no\ instante\ "t",$$

$$B = operador\ de\ atraso\ ("lag\ operator"),$$

$$B^d . z_t = z_{t-d} \text{ ,}$$

$$\phi_1 , \dots , \phi_p = coeficientes\ de\ autorregressão\ (do\ atraso\ 1, ao\ atraso\ "p"),$$

$$\theta_1 , \dots , \theta_q = coeficientes\ de\ média\ móvel\ (do\ atraso\ 1, ao\ atraso\ "q")$$

NOTA: os resíduos associados a previsões são assumidos como nulos, assim como qualquer valor ou resíduo existente num atraso que não esteja representado nos dados de treino (valor demasiado antigo, para ser observado).

a. As previsões são feitas de forma iterativa, dado que cada nova previsão estará, obrigatoriamente, dependente do(s) valor(es) imediatamente anteriores. Assim, o vetor *yITERACAO* começa por só conter os dados de treino (do mais antigo, para o mais recente – ordem DECRESCENTE de *lags*/atrasos), mas, em cada iteração, vai tendo uma previsão adicionada ao seu final (o comprimento do vetor cresce uma unidade, a cada iteração);
b. Pela função *ARparcelas*, são devolvidas as variáveis: *sumAR* – soma resultante dos coeficientes correspondentes ao parâmetro *p* de autoregressão (*ARcoefs*) e que também depende do parâmetro *d* – e *Dcoefs* – vetor que inclui os coeficientes do polinómio introduzido no cálculo, quando *d* não é nulo (e que será importante, também, no cálculo da soma diretamente dependente do parâmetro *d*);
c. Segue-se o cálculo de *sumMA*, valor que depende do parâmetro *q* e dos coeficientes de média móvel (MAcoefs). Como referido, qualquer resíduo não associado aos dados de treino fornecidos será, automaticamente, zero. Para cada *lag* entre 1 e *q*, é multiplicado o valor *z* correspondente (entre z_{t-1} e z_{t-q}) pelo respetivo coeficiente θ (entre θ_1 e θ_q);
d. A seguir, calcula-se *sumI*, que provém do valor w_t. Assim, corresponde à soma $z_{t-1} + \cdots + z_{t-d}$, multiplicada, parcela a parcela, pelos elementos do vetor de coeficientes proveniente do caso notável $(1-B)^d . z_t$ à exceção do 1º elemento do polinómio, que dirá respeito a z_t, que queremos determinar;
e. Depois disto, resta calcular o valor previsto, segundo a soma de 3 parcelas indicada no início desta secção.

Ainda que o valor de *d* não deva ser superior a 1 ou 2 e o valor de *q* não pareça dar bons resultados fora dessa mesma gama, a função consegue lidar com qualquer valor para estes parâmetros.

3. **ARIMAteste_MATLAB**:

   Programa de interação que declara todas as variáveis necessárias à implementação do ARIMA. Chama a função principal, *ARIMAteste1*, com as variáveis pretendidas para esse teste, e recolhe os *outputs* resultantes, produzindo os gráficos respetivos.

4. **ARIMAteste1**:

   Função principal para a implementação do ARIMA(p,d,q). Recebe os parâmetros *p*, *d* e *q* necessários, o vetor de dados de treino do modelo (*y_antigo*) e o número de previsões pretendidas (*numPrev*). Devolve um vetor para cada um dos seguintes *outputs*: dados de treino (*y_past*), previsões feitas (*y_predict*), resíduos correspondentes ao treino (*resid*), limites de confiança de 95% das previsões (*lowerConf95* e *upperConf95*), coeficientes otimizados para o parâmetro *p* de autorregressão (*ARcoefs*) e coeficientes otimizados para o parâmetro *q* de média móvel (*MAcoefs*).

   a. Começa por retirar a média aos dados de treino (a média nula é condição necessária à otimização do modelo);
   b. Segue-se o cálculo da *partial autocorrelation function* e da *autocorrelation function*. Isto, agora, não é importante, mas poderia ser, dado que estes parâmetros são frequentemente usados, quer para a identificação do modelo a usar, quer como primeira estimativa dos coeficientes a otimizar;
   c. A seguir, pela função *arma_mle*, calculam-se os coeficientes do modelo (*ARcoefs* e *MAcoefs*), bem como os resíduos de treino (*resid*) e o "desvio-padrão estimado dos resíduos (*random shocks*)" (*sigmahat*). Os resíduos poderiam, alternativamente, ser calculados com a função *ValuesResidues*;
   d. Calculam-se, então, recorrendo à função *ARIMAforecast*, as previsões pretendidas (*y_predict*) e os respetivos intervalos de confiança (*lowerConf95* e *upperConf95*), segundo o modelo já estimado;
   e. A função termina com a soma da média dos dados de treino às variáveis de saída (uma vez que estas estavam adaptadas a uma média nula).

5. **arma_mle**:

   Função que otimiza os coeficientes de determinado modelo ARMA(p,q). *Inputs*: parâmetros *p* e *q*, dados de treino (*y*) e opção binária de mostrar, ou não, a informação acerca da otimização feita (*info*). *Output*: variável *results*, que inclui os coeficientes otimizados (*ar* e *ma*), os resíduos de treino (*residuos*), o "desvio-padrão estimado dos resíduos (*random shocks*)" (*sigma*) e o parâmetro de *log-likelihood* (*loglik*).

   a. Faz as estimativas iniciais dos coeficientes do modelo, com a função interna *initialize_arma*;
   b. Utilizando a rotina *fminunc* do MATLAB, otimiza os parâmetros em causa, através da função interna *log_likelihood*.

   O parâmetro *p* (de autorregressão) assume maior preponderância neste algoritmo, sendo que ainda não encontrei, inclusivamente, nenhum caso em que *p=0* não resulte em erro do programa. Contudo, existem combinações de *inputs* que, apesar de resultarem em soluções de modelo divergentes, são tratadas pela função (esta divergência é notada nos gráficos resultantes, quer pela sua forma, quer pela deficiência dos intervalos de confiança).

   Regra geral, os melhores modelos não expressam valores de *q* e *d* maiores que 1.

6. **ARparcelas**:

   Função que calcula a soma de parcelas que depende dos coeficientes e atrasos de autorregressão (ligados a *p*), em interação com os valores de atraso dependentes de *d*. *Inputs*: dados de treino (*y*), vetor *B*, que inclui *ARcoefs* e *MAcoefs*, e parâmetros *p* e *d*. *Outputs*: *sumAR* e *Dcoefs* (explicação das variáveis no texto sobre *ARIMAforecast*).

a. Se *p=0*, tem-se *sumAR=0*;
b. Independentemente do valor de *p*, se *d* não é nulo, o vetor *Dcoefs* formar-se-á (*Dcoefs* também é utilizado no cálculo de *sumI*, que não depende de *d*);
c. No caso de *d=0*, o polinómio dependente de *d* não se forma, logo, *Dcoefs* assume um valor nulo. Assim, se *p* for diferente de zero, *sumAR* resultará, apenas, da soma dos valores correspondentes aos *p* atrasos a ter em conta;
d. Se nem *p*, nem *d*, forem nulos, *sumAR* resultará da soma dos parâmetros $w_{t-1} + \cdots + w_{t-p}$, multiplicados, caso a caso, pelos *ARcoefs* (como indicado nas fórmulas do texto de *ARIMAforecast*);
e. Com todas as previsões feitas, calculam-se os limites de confiança sobre cada valor, recorrendo a um vetor com os desvios associados a cada previsão (*desvioPadrao*), determinado com a função *desvioARIMA*.

7. **desvioARIMA**:

Função que calcula o desvio associado a cada previsão feita e devolve um vetor com esses valores. Tem, como *inputs*, o vetor *B*, que inclui *ARcoefs* e *MAcoefs*, e os valores *p*, *sigmahat* e *numPrev*.

Esta função segue os cálculos baseados nas págs.252-256 do livro de PANKRATZ. Contudo, a porção referente ao cálculo dos coeficientes Ψ não está bem explícita, pelo que generalizei para as fórmulas seguintes:

$$\psi_0 = 1 \,,$$

$$\psi_1 = \phi_1 - \sum_{j=1}^{q} \theta_j \,,$$

$$\psi_i = \sum_{k=1}^{p,i} \phi_k . \psi_{i-k} \text{ , com } i = 2, \ldots \text{ , } numPrev \,.$$