ALGORITMO EM LINGUAGEM PYTHON PARA REPORTAR INTERVALOS DE CREDIBILIDADE NOS LAUDOS DE CÁLCULO DE VELOCIDADE POR VÍDEO

A PYTHON ALGORITHM FOR REPORTING CREDIBILITY INTERVALS IN VIDEO-BASED SPEED ESTIMATION FORENSIC REPORTS

IMPLEMENTACIÓN DE UN ALGORITMO EN PYTHON PARA REPORTAR INTERVALOS DE CREDIBILIDAD EN INFORMES PERICIALES DE ESTIMACIÓN DE VELOCIDAD A PARTIR DE VÍDEO

Tadeu Junior Gross[1], Fábio Hideki Maruyama[2], Alexandre Vinícius Festa[3]

1. Perito Oficial Criminal na POLITEC-MT. Doutor (EESC-USP), Mestre (POLI-USP) e Bacharel (UEL-PR) em Engenharia Elétrica. Lotado na Gerência de Perícias em Áudio e Vídeo, atua principalmente em exames de fotogrametria e verificação de edição.

Currículo Lattes: http://lattes.cnpq.br/8108119439454104

2. Perito Oficial Criminal na POLITEC-MT. Engenheiro Civil e Bacharel em Direito pela UFMT. Lotado na Gerência de Perícias em Crimes de Trânsito, atua em reconstrução forense de acidentes.

Currículo Lattes: http://lattes.cnpq.br/2041678414967649

3. Perito Oficial Criminal na POLITEC-MT. Mestre (EESC-USP) e Bacharel (UFMT) em Engenharia Elétrica. Lotado na Gerência de Perícias de Computação, atua em perícias de extração, preservação e interpretação de dados digitais.

Currículo Lattes: http://lattes.cnpq.br/3440801295921170

Ano 07 | Vol. 12
Março de 2026

Abstract

This study presents the implementation, in Python, of a methodology described in the literature for calculating credibility intervals associated with speed estimation from traffic accident videos. Due to the frequent misinterpretation of confidence intervals as probability statements, especially by the media and end-users of forensic reports, this work advocates reporting the inherent uncertainty in video-based speed estimates as a legitimate probability interval. A systematic literature review identified a probabilistic programming approach specifically designed for this purpose. The proposed implementation uses a free, open-source programming language widely adopted in scientific computing. As its main contribution, this study provides the forensic science community with a reliable computational tool that facilitates the application of bayesian credibility intervals in forensic speed estimation analyses. Validation was performed by direct comparison with the results from the reference study, using the same experimental dataset. The tests revealed nearly identical numerical and graphical results, along with computational performance suitable for practical forensic use. These findings support the feasibility and applicability of the Python-based tool as an auxiliary resource in traffic accident investigations.

Keywords: Traffic Accidents. Video Analysis. Speed Estimation. Photogrammetry. Bayesian Statistics.

Resumo

Neste artigo, implementa-se em Python uma metodologia descrita na literatura para calcular intervalos de credibilidade associados à medição de velocidade em vídeos de acidentes de trânsito. Dada a recorrente interpretação equivocada de intervalos de confiança como de probabilidade, especialmente pela imprensa e usuários finais dos laudos periciais, parece mais apropriado que a incerteza inerente à velocidade estimada por vídeo seja reportada como um intervalo de probabilidade legítimo. Em uma busca sistemática na literatura, foi encontrada uma abordagem de programação probabilística desenvolvida justamente para este fim. O presente artigo trata da implementação dessa abordagem em uma linguagem de programação gratuita e de código aberto e reconhecidamente apropriada para computação científica. A ideia é disponibilizar à comunidade de cientistas forenses uma ferramenta computacional confiável que facilite o uso de intervalos de credibilidade bayesianos nas perícias de cálculo de velocidade por vídeo. Adotou-se como estratégia de validação do código a comparação direta com os resultados do estudo norteador da implementação, sendo utilizados nos testes os mesmos dados experimentais desta referência. Essa comparação revelou achados numéricos e gráficos praticamente idênticos. Também foi observada uma latência computacional satisfatória para o uso prático. Enfim, o código em Python delineado neste trabalho parece viável como ferramenta pericial.

Palavras-chave: Trânsito. Vídeo. Velocidade. Fotogrametria. Estatística Bayesiana.

Resumen

Este artículo presenta la implementación, en Python, de una metodología descrita en la literatura para calcular intervalos de credibilidad asociados a la estimación de velocidad en vídeos de accidentes de tráfico. La frecuente confusión entre intervalos de confianza y declaraciones de probabilidad, especialmente entre los medios de comunicación y usuarios finales de informes periciales, motiva la adopción de un enfoque bayesiano que permita reportar la incertidumbre como un intervalo de probabilidad legítimo. Tras una revisión sistemática de la literatura, se identificó un método de programación probabilística diseñado específicamente para este fin. La implementación se realizó en un lenguaje de programación gratuito, de código abierto y ampliamente reconocido por su aplicación en computación científica. Como contribución principal, se ofrece a la comunidad forense una herramienta computacional robusta y fiable que facilita la utilización de intervalos de credibilidad bayesianos en análisis de velocidad basados en vídeo. La validación del código se llevó a cabo mediante comparación directa con los resultados del estudio de referencia, empleando el mismo conjunto de datos experimentales. Los resultados mostraron equivalencia numérica y gráfica, además de un tiempo de procesamiento adecuado para su uso práctico en el contexto pericial.

Palabras clave: Accidentes de Tráfico. Análisis de Vídeo. Estimación de Velocidade. Fotogrametria. Estadística Bayesiana.

1. Introdução

Hoje em dia, sistemas capazes de registrar vídeo estão por toda parte, sendo cada vez mais raros crimes que de algum modo não tenham sido, ao menos parcialmente, capturados por alguma câmera (XIAO et al, 2019). Dentre os crimes mais frequentemente perenizados na forma de vídeo, estão os de trânsito (EPSTEIN e WESTLAKE, 2019). Nas perícias de "acidente" de tráfego por vídeo realizadas por Peritos Oficiais Criminais, normalmente é necessário mensurar uma ou mais velocidades de forma precisa e acurada para que seja possível esclarecer ao Requisitante (Presidente de Inquérito Policial Militar, Delegado, Promotor ou Juiz) a provável causa determinante do evento (KIM et al, 2018). Além desta precípua importância técnico-jurídica, quando um condutor suspeito de dar causa a um "acidente" com vítima fatal trafega a uma velocidade aparentemente acima da regulamentada para a via, a sua magnitude real passa a ser uma indagação da sociedade em geral, de modo que não encontrar um valor plausível e com margem de erro minimamente estreita provoca uma percepção generalizada de injustiça social e ineficácia do Estado (BORGES, 2018).

Nos laudos periciais de vídeo, em regra, a incerteza associada à velocidade estimada é apresentada em termos de um intervalo de confiança (HOOGEBOOM e ALBERINK, 2010). Embora essa abordagem frequentista seja aceita e amplamente recomendada, o conceito estatístico subjacente não é dos mais intuitivos e, muitas vezes, o nível de confiança adotado acaba sendo interpretado equivocadamente como a probabilidade da velocidade real estar contida no intervalo reportado[1] - esta interpretação simples e direta não seria inapropriada se, ao invés de um intervalo de confiança frequentista, fosse reportado um intervalo de credibilidade bayesiano (MIEREMET et al, 2018). Assim sendo, parece adequado passar a empregar este tipo de intervalo nos laudos periciais de cálculo de velocidade por vídeo. Um método numérico para obtenção de intervalos de credibilidade de velocidade foi proposto e validado por Mieremet et al (2018). Entretanto, eles não disponibilizaram o script implementado, o que dificulta sobremaneira a utilização prática da metodologia pelos Institutos de Criminalística.

No presente artigo, baseando-se na metodologia apresentada em (MIEREMET et al, 2018), implementa-se um algoritmo em Python - linguagem amplamente utilizada em computação científica (PÉREZ et al, 2011) - para calcular intervalos de credibilidade de velocidade. A validação do script é realizada através de testes com os mesmos dados experimentais utilizados por Mieremet et al (2018). O propósito deste trabalho é disponibilizar aos Peritos Oficiais Criminais uma ferramenta prática de obtenção de intervalos de credibilidade para velocidades calculadas por vídeos de CFTV (Circuito Fechado de Televisão).

Este trabalho está organizado conforme segue. Na Seção 2, descreve-se de forma sucinta a estratégia rotineiramente adotada no meio forense para auferir uma única medida de velocidade por vídeo de CFTV. Na Seção 3, primeiramente se discorre a respeito dos dados de entrada, disponibilizando-se o trecho de código dedicado à sua leitura. Em seguida, passa-se à codificação da técnica de Monte Carlo por Cadeia de Markov utilizada por Mieremet et al (2018) para associar intervalos de credibilidade (ou probabilidade) às velocidades estimadas através de vídeos de CFTV. Na Seção 4, disponibilizam-se os resultados gráficos e numéricos obtidos com o código implementado. Na Seção 5, esses achados são discutidos de forma sistemática e pormenorizada, destacando-se a robustez da saída (i.e., a invariância do intervalo de credibilidade frente aos diferentes setups de simulação possíveis) e a latência de processamento reduzida bem como a notável similaridade com os resultados do estudo norteador da implementação. Por fim, na Seção 6, é encerrado o artigo dando-se ênfase às suas principais implicações.

2. Medição de velocidade por vídeo

Na obtenção de uma única medida de velocidade por vídeo de CFTV, primeiramente são escolhidos dois frames contendo o veículo de interesse em momentos distintos. Em seguida, combinando-os de forma a produzir uma nova imagem com esse veículo em posições distintas, mede-se o deslocamento via alguma técnica de fotogrametria, a exemplo da desenvolvida em (WONG et al, 2014). Posteriormente, a diferença entre os índices dos dois frames selecionados é dividida pela taxa de quadros do vídeo para se estimar o intervalo de tempo decorrido – caso essa taxa não seja constante, pode-se recorrer à diferença de "pkt_pts_time" (packet presentation time), conforme explicado em (EPSTEIN e WESTLAKE, 2019). Por fim, obtém-se uma medida de velocidade pela razão entre o deslocamento e o intervalo de tempo encontrados.

Na prática, as imagens de CFTV apresentam várias imperfeições, sendo a distorção de lente do tipo barril uma das mais comuns na casuística forense (HOOGEBOOM et al, 2015). Desta forma, na maioria dos casos reais, antes de efetivamente aplicar o procedimento do parágrafo anterior, é necessário submeter os frames de interesse a um processo de idealização (isto é, de mitigação das distorções identificadas). No "ImageJ" (SCHNEIDER et al, 2012), pode-se empregar o plugin "Undistorter", que é baseado no "modelo de divisão" de Bukhari e Dailey (2013) e idealiza as imagens a partir de encurvamentos fictícios indicados pelo usuário com linhas segmentadas[2].

3. Implementação do algoritmo

Dada a presença de inúmeras fontes de erro, a exemplo da distorção de lente e do borrão de movimento [3], é esperado que haja algum desvio entre a velocidade real v_{inc} (o subscrito "inc" é uma contração para incidente) e a sua estimativa m_{inc} obtida pelo procedimento resumido na Seção 2. Uma vez calculado m_{inc},

[1] *Caso o leitor tenha interesse na interpretação correta do intervalo de confiança frequentista, recomenda-se o conteúdo do endereço eletrônico https://fernandafperes.com.br/blog/intervalo-de-confianca/. Acesso em 03 de janeiro de 2024.*
[2] *https://www.youtube.com/watch?v=IKBvqocLHeo. Acesso em 27 de maio de 2023.*

é desejável ponderar a respeito dos valores mais prováveis da velocidade questionada v_{inc}. Esta é a finalidade da implementação em Python delineada nas subseções que seguem.

3.1 Dados de entrada: descrição e leitura

O comportamento do erro pode ser investigado via imagens de referência[4]. No enquadramento da câmera de CFTV que registrou o acidente, com um veículo preferencialmente similar, repete-se algumas vezes o percurso do veículo questionado. Em cada repetição, imprime-se ao veículo uma velocidade distinta cujo valor real se sabe com suficiente acurácia. No vídeo assim produzido, estima-se a velocidade do veículo em cada repetição via a mesma metodologia aplicada no vídeo questionado para calcular m_{inc} (vide a Seção 2). Esse processo resultará em duas coleções pareadas de velocidades, uma de verdadeiras (conhecidas), designada por $v = \{v_1, v_2, ..., v_N\}$, e uma de estimadas, representada por $m = \{m_1, m_2, ..., m_N\}$. Um número N de repetições entre 15 e 25 normalmente é satisfatório (MIEREMET et al, 2018).

Na POLITEC/MT - Perícia Oficial e Identificação Técnica do Estado do Mato Grosso - emprega-se o aparato da Figura 1 para registrar a velocidade v_i em cada repetição i. A distância entre os dois feixes paralelos de luz colimada (representados pelas linhas brancas) é pré-estabelecida e os instantes de bloqueio deles pelo veículo são identificados por fotoresistores (ou Light Dependent Resistors). Detalhes da instrumentação eletrônica envolvida não fazem parte do escopo do presente trabalho.

Em resumo, a estimativa m_{inc} e as coleções pareadas m e v, bem como a probabilidade p de v_{inc} pertencer ao intervalo de credibilidade[5], são os dados de entrada. A parte inicial do script, conforme mostrado na Tabela 1, trata da leitura (carregamento) desses dados, bem como da importação de todas as bibliotecas e funções utilizadas.

3.2 Geração de amostras simuladas

Figura 1 – Aparato empregado na POLITEC/MT para registrar a velocidade de referência v_i. Fonte: Tomada fotográfica realizada pelo Perito 3.

[3] Na aquisição de uma imagem, a luz incide no sensor da câmera por um intervalo denominado tempo de exposição. Um elemento de cena que seja estático, ou que se movimenta muito pouco durante esse tempo, ficará com os contornos bem definidos. Já um que venha a realizar um movimento abrupto, ocupará, de forma sucessiva, posições espaciais ligeiramente distintas ao longo dessa exposição, vindo, portanto, a sensibilizar uma área do sensor de estado sólido bem maior do que deveria e, por consequência, a ficar espalhado (ou seja, borrado) na imagem resultante. Eventualmente, a aparência borrada pode dificultar a inteligibilidade do elemento não-estático, tanto que é comum se empregar o termo degradação por borrão de movimento para se referir a esse fenômeno em que os contornos de um objeto ficam difusos por conta do seu dinamismo. Do ponto de vista de processamento de sinais, o borrão de movimento envolve algum colapso de informações de alta frequência do elemento dinâmico (LIN et al, 2008, p. 1334).

[4] Imagem de referência é o registro que possui uma grandeza física cujo valor considera-se conhecido. Esse valor é a medida de referência (FILHO, 2022).

[5] Essa probabilidade é o análogo bayesiano do nível de confiança da estatística frequentista. Normalmente, estipula-se um valor de pelo menos 90% (MIEREMET et al, 2018).

Leitura dos dados e importação das bibliotecas e funções

```
import numpy as np
from scipy.stats import norm
from random import random

m_inc = 77.9

m = np.array([37.3,51.2,57.3,67.4,74.9,84.4,97.6,113.2,51.0,62.6,85.4,83.6,109.3,42.1,52.2,60.8,81.0,
              87.3,98.9])
v = np.array([39.3,50.6,60.1,70.5,77.4,88.8,98.8,115.7,53.2,64.5,87.0,84.8,110.8,43.6,53.8,61.8,83.1,
              88.9,101.5])
p = 0.95
```

Tabela 1 – Parte inicial da implementação em linguagem Python. Fonte: Esse trecho de código é de autoria própria. Já os dados são de Mieremet et al (2018).

via MCMC

Um modelo plausível para a velocidade estimada, segundo Mieremet et al (2018), é

$$m = a + b.v + e \quad (1)$$

onde a é o erro sistemático, b é o erro (adimensional) dependente da velocidade real v e e é o erro de medição - esses erros são considerados normalmente distribuídos (i.e., gaussianos), com e tendo média nula (para não introduzir outro erro sistemático) e desvio padrão 10^c (para garantir valores estritamente positivos)[6]. Nesse contexto, dado um estado qualquer - aqui representado pelo trio $(a,\ b,\ c)$ de parâmetros - gera-se uma amostra da velocidade questionada pela fórmula

$$v_{\text{inc}} = \frac{\text{m}_{\text{inc}} - a - N\left(0,\ 10^{c}\right)}{b} \quad (2)$$

A abordagem MCMC (Markov Chain Monte Carlo) é um processo iterativo em que os parâmetros do modelo evoluem progressivamente em direção à estacionariedade em termos de distribuição conjunta (SCHOOT et al, 2021). A cada iteração, um novo estado é visitado e, por conseguinte, uma nova amostra é gerada por meio da fórmula acima - as múltiplas amostras assim obtidas, após o descarte das geradas durante o transitório inicial (período de burn-in), definem a distribuição da velocidade questionada (MIEREMET et al, 2018). Essa abordagem consiste nos seis passos detalhados a seguir.

3.2.1 Primeiro passo: escolha do estado inicial

A fim de se ter uma ideia preliminar da relação entre m e v, pode-se recorrer a uma regressão linear simples (MIEREMET et al, 2018). O intercepto e a inclinação da reta de predição podem ser obtidos pela operação matricial (BOYD e VANDENBERGHE, 2018)[7]

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \mathbf{V}^{\dagger}\mathbf{m}, \quad (3)$$

$$\text{onde } \mathbf{m} = \begin{bmatrix} m_1 & m_2 & \cdots & m_N \end{bmatrix}^{\mathsf{T}},\ \mathbf{V} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_N \end{bmatrix}^{\mathsf{T}} \text{ e } \mathbf{V}^{\dagger} = \left(\mathbf{V}^{\mathsf{T}}\mathbf{V}\right)^{-1}\mathbf{V}^{\mathsf{T}}.$$

Os coeficientes $\hat{a}$ e $\hat{b}$ minimizam o erro quadrático médio (PENNSTATE, 2018)

$$\text{MSE} = \frac{\mathbf{e}^{\mathsf{T}}\mathbf{e}}{N-2} \quad (4)$$

onde $\mathbf{e}$ é o vetor-coluna dos resíduos de predição, que é dado pela diferença (BOYD e VANDENBERGHE, 2018; PENNSTATE, 2018)

$$\mathbf{e} = \mathbf{m} - \mathbf{V}\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} \quad (5)$$

Uma vez que o MSE pode ser interpretado como uma primeira estimativa da variância de e (MIEREMET et al, 2018; PENNSTATE,

[6] *Na estatística bayesiana, os parâmetros do modelo também são variáveis aleatórias (SCHOOT et al, 2021).*
[7] $\mathbf{V}^{\dagger}$ *é a pseudoinversa (ou inversa de Moore-Penrose) da matriz V (BOYD e VANDENBERGHE, 2018).*

2018), uma aproximação para o parâmetro c da Equação (2) é

$$\hat{c} = 0.5 \log_{10}(\mathrm{MSE}) \quad (6)$$

Na condição de estacionariedade, presume-se que seja bem mais provável encontrar o estado (a, b, c) nos arredores de $(\hat{a}, \hat{b}, \hat{c})$ do que em outras regiões do espaço de parâmetros. Por conta disso, escolheu-se $(\hat{a}, \hat{b}, \hat{c})$ como o estado inicial canônico.

A Tabela 2 apresenta o primeiro passo implementado em linguagem Python.

Inicialização dos parâmetros do modelo

```
N = len(v)
M = np.matrix(m).T
V = np.matrix([np.ones(N).tolist(),v]).T
a_chap, b_chap = np.linalg.pinv(V)*M                  # equação (3)
e = M – V*np.matrix([[a_chap[0,0]],[b_chap[0,0]]])    # equação (5)
MSE = e.T*e/(N-2.0)                                   # equação (4)
c_chap = 0.5*np.log10(MSE)                            # equação (6)
s0 = np.array([a_chap[0,0],b_chap[0,0],c_chap[0,0]])  # estado inicial escolhido
```

Tabela 2 – Implementação do primeiro passo em linguagem Python. Fonte: Autoria própria.

3.2.2 Segundo passo: cálculo da "probabilidade" a priori do estado

A especificação da priori dos parâmetros é fundamental em modelagem bayesiana e envolve sugerir a forma da distribuição (beta, gaussiana, uniforme etc.) bem como os valores numéricos dos seus hiperparâmetros[8]. Trata-se de escolher a distribuição para o estado (a, b, c) sem levar em conta os dados levantados. Essa escolha baseada exclusivamente no conhecimento prévio é denominada "elicitação da priori" (SCHOOT et al, 2021).

No presente trabalho, adotou-se a distribuição a priori sugerida por Mieremet et al (2018), que é dada por

$$f_{prior}(a, b, c) = g(a|\mu_a, \sigma_a).g(b|\mu_b, \sigma_b).g(c|\mu_c, \sigma_c) \quad (7)$$

com g sendo a função densidade de probabilidade gaussiana de média μ e desvio padrão σ (a densidade conjunta é o produtório das marginais porque os parâmetros do modelo são considerados independentes).

Em relação aos hiperparâmetros (média e desvio padrão dos parâmetros do modelo), foram atribuídos os valores numéricos

$$(\mu_a, \mu_b, \mu_c) = (0{,}1, 0{,}3) \text{ e } (\sigma_a, \sigma_b, \sigma_c) = (10, 0{,}1, 1)' \quad (8)$$

uma vez que, a priori, parece razoável esperar um erro sistemático de 0 km/h com um esvio padrão de 10 km/h, um erro dependente da velocidade em torno da unidade e um erro de medição com um desvio padrão tipicamente (confiança de 68%) entre $10^{0.3 - 1}$ = 0.2 km/h e $10^{0.3 + 1}$ = 20 km/h (MIEREMET et al, 2018).

A Tabela 3 disponibiliza a função utilizada para calcular a priori em $s = (a, b, c)$.

[8] Os hiperparâmetros controlam a "informatividade" da distribuição. É através deles que se estabelece uma priori informativa, fracamente informativa ou difusa (SCHOOT et al, 2021). No caso de uma gaussiana, eles são representados pela média e o desvio padrão escolhidos, onde este último pode ser interpretado como o nível de incerteza quanto ao parâmetro estar perto de tal média. Estando previamente convencido de que o parâmetro gaussiano é bem retratado pelos arredores da média escolhida, emprega-se um desvio padrão relativamente baixo. Do contrário, isto é, dispondo de pouca informação em relação ao parâmetro gaussiano, adota-se um desvio padrão mais elevado.

Função para o cálculo de f_{prior}

```
def fprior(s):
    priormean = np.array([0.0, 1.0, 0.3])          # parte esquerda de (8)
    priorstd  = np.array([10.0, 0.1, 1.0])         # parte direita de (8)
    return np.prod(norm.pdf(s,priormean,priorstd)) # equação (7)
```

Tabela 3 – Função em Python para retornar o valor da priori no estado $s = (a, b, c)$. Fonte: Autoria própria.

3.2.3 Terceiro passo: cálculo da "probabilidade" a posteriori do estado

A estatística bayesiana é uma abordagem de análise de dados baseada no Teorema de Bayes, onde o conhecimento acerca dos parâmetros do modelo é atualizado com as informações presentes nos dados observados - a distribuição a priori, que encapsula o conhecimento prévio, é combinada com a função de verossimilhança dos dados, dando origem à distribuição a posteriori, a qual expressa o conhecimento atualizado sobre os parâmetros (SCHOOT et al, 2021). Matematicamente, escreve-se (GELMAN et al, 2013)[9]

$$f_{post}\left(parâmetros|dados\right) \propto L\left(dados|parâmetros\right).f_{prior}\left(parâmetros\right) \quad (9)$$

onde L é a mencionada função de verossimilhança dos dados.

No presente trabalho, deseja-se ponderar a respeito das diferentes fontes de erro a partir do confronto entre as velocidades estimadas (m) e verdadeiras (v). Em outros termos, é de interesse aqui verificar o que m e v dizem sobre as distribuições de a, b e c. Para tanto, é necessário explorar a posteriori $f_{post}(a, b,c|\mathrm{m,v})$, cuja forma é dada por (MIEREMET et al, 2018)[10]

$$f_{post}\left(a,b,c|m,v\right) \propto \left[\prod_{i=1}^{N} g\left(m_i|a+b.v_i,c\right)\right].f_{prior}\left(a,b,c\right), \quad (10)$$

onde g é a função densidade de probabilidade normal, com média $a+b.v_i$ e desvio padrão c.

A Tabela 4 disponibiliza a função utilizada para calcular a posteriori em $s = (a, b, c)$.

Função para o cálculo de f_{post}

```
def fpost(s):
    return np.prod(norm.pdf(m,s[0]+s[1]*v,10**s[2]))*fprior(s)  # expressão em (10)
```

Tabela 4 – Função em Python para retornar o valor da posteriori no estado $s = (a, b, c)$. Fonte: Autoria própria.

3.2.4 Quarto passo: proposição de um novo estado

Em termos simples, para obtenção das amostras de (a, b, c), deve-se "passear" pelo espaço paramétrico - isto é, ao longo do suporte da densidade em (10) - de tal forma que se visite mais vezes as regiões de maior probabilidade[11]*. Nesse "passeio guiado", estando em $(a, b, c)_{cur}$, cogita-se um próximo estado $(a, b, c)_{prop}$, o qual será efetivamente visitado somente se satisfizer um critério de aceite (vide a Seção 3.2.5). Uma maneira prática de sugerir um novo estado a partir do atual é através da adição de um pequeno valor aleatório em cada um dos parâmetros (MIEREMET et al, 2018), i.e.,*[12]

$$(a,b,c)_{prop} = (a,b,c)_{cur} + (n_a,n_b,n_c), \quad (11)$$

com $n_a = N(0,\tau_a)$, $n_b = N(0, \tau_b)$ e $n_c = N(0, \tau_c)$ onde τ_a, τ_b, e τ_c dimensionam o "passo" (i.e., o tamanho típico do "salto").

Na prática, em uma normal com desvio padrão τ, muito raramente ocorrem valores a mais de 3τ da média. Portanto, considerando a necessidade de amostrar o intervalo típico Δ de cada parâmetro em pelo menos K pontos, parece razoável estipular que τ não seja superior a $\Delta/(3K)$. De acordo com os valores especificados em (8), sabe-se, por experiência, que $\Delta_a = 10 - (-10) = 20$, $\Delta_b = 1.1 - 0.9 = 0.2$ e $\Delta_c = 1.3 - (-0.7)=2$. Assim, impondo $K = 20$, deduz-se que

$$(\tau_a, \tau_b, \tau_c) = \left(\frac{1}{3}, \frac{1}{300}, \frac{1}{30}\right) \quad (12)$$

é uma escolha apropriada. Essa estratégia para estabelecer (τ_a, τ_b, τ_c), inclusive os valores adotados nos cálculos, devem-se a Mieremet et al (2018).

[9,10] *Note-se que (9) e (10) não são equações. Trata-se de relações de proporcionalidade.*
[11] *https://www.youtube.com/watch?v=4l6TaYo9j_Y. Acesso em 25 de agosto de 2023.*
[12] *https://www.youtube.com/watch?v=oX2wlGSn4jY. Acesso em 28 de agosto de 2023.*

3.2.5 Quinto passo: escolha entre o estado atual e o proposto

O método MCMC utilizado por Mieremet et al (2018) para conduzir o "passeio" pelo domínio da posteriori em (10) foi o de Metropolis et al (1953). Neste algoritmo, o candidato a próximo estado (vide a Seção 3.2.4) é comparado ao atual por meio da razão (GELMAN et al, 2013; MIEREMET et al, 2018)[13]

$$r = \left[f_{post}^{prosp} \Big/ f_{post}^{cur} \right] \qquad (13)$$

onde o numerador e o denominador decorrem de (10) calculada em $(a, b, c)_{prop}$ e $(a, b, c)_{cur}$, respectivamente. Sublinhe-se que essa divisão cancela qualquer fator da posteriori que não seja função dos parâmetros, por isso este tipo de fator foi omitido em (10).

Tendo em mãos o valor da razão em (13), procede-se da seguinte forma (GELMAN et al, 2013; MIEREMET et al, 2018): [14]

- *Um r maior do que a unidade significa que o estado proposto explica melhor os dados do que o atual. Neste caso, ele é aceito de imediato e passa a ser o novo estado atual.*
- *Um r menor do que a unidade significa que o candidato não explica os dados tão bem quanto o estado atual. Entretanto, isso não pode implicar por si só sua rejeição, afinal de contas não se trata de um processo de maximização. Nesta situação, o que se faz é aprovar o candidato com uma probabilidade r - um número aleatório entre 0 e 1 é gerado de uma distribuição uniforme e, caso ele seja inferior a r, o estado proposto é aceito e torna-se o novo estado atual; do contrário, o estado não é alterado.*

Detalhes teóricos acerca da convergência e estacionariedade do método Metropolis-Hastings[15] são discutidos por Hill e Spall (2019). Aspectos computacionais e teóricos relacionados a cadeias de Markov para amostragem da posteriori são apresentados de forma aprofundada e didática nos capítulos 11 e 12 de (GELMAN et al, 2013) – esta é uma das principais referências em Computação Bayesiana e é recorrentemente apontada na literatura da área.

3.2.6 Sexto passo: obtenção de uma amostra da velocidade questionada

Por último, do novo estado[16], gera-se uma amostra de v_{inc}. Isto é, os novos valores de a, b e c são substituídos em (2) para a subsequente obtenção de um valor simulado v_{inc} da velocidade questionada – vide a Tabela 5. Feito isto, armazenam-se os valores a, b, c e v_{inc} da iteração vigente e, em seguida, executa-se o processo novamente a partir da proposição de um próximo estado (quarto passo).

Função para gerar um valor simulado de v_{inc}

```
def vinc(s):
    return (m_inc-s[0]-np.random.normal(0.0,10**s[2]))/s[1]   # equação (2)
```

Tabela 5 – Função em Python que retorna uma amostra de v_{inc} para o estado $s = (a, b, c)$. Fonte: Autoria própria.

O processo do quarto ao sexto passo é reiterado até que se alcance uma quantidade predefinida de amostras da velocidade questionada – vide a Tabela 6. Essas amostras nada mais são do que realizações da distribuição real de v_{inc} e, portanto, podem ser empregadas na inferência do intervalo de credibilidade da velocidade questionada (vide a Seção 3.3).
No código apresentado na Tabela 6, atente-se que:

- *np.random.multivariate_normal([0.0, 0.0, 0.0], np.diag(tau**2), Niter) gera todos os Niter saltos aleatórios do tipo $n = (n_a, n_b, n_c)$;*
- *A cada iteração, anexa-se um estado (o proposto ou o atual) à lista S;*
- *S[-1] retorna o último estado anexado à lista dinâmica S de estados.*

Reiteração do quarto ao sexto passo por Niter vezes

```
S = []                                  # declara S como uma lista vazia
v_inc = []                              # declara v_inc como uma lista vazia
S.append(s0)                            # anexa o estado inicial s0 à lista S - vide a Tabela 2
v_inc.append(vinc(s0))                  # anexa a amostra vinc(s0) à lista v_inc - vide a Tabela 5
den = fpost(s0)                         # valor da posteriori para o estado inicial s0 - vide a Tabela 4
Nburn = 10**4                           # período de burn-in
Niter = 10**5 + Nburn                   # número de iterações
tau = np.array([1.0, 0.01, 0.1])/3.0    # tamanho típico do salto - vide a equação (12)

for n in np.random.multivariate_normal([0.0, 0.0, 0.0], np.diag(tau**2), Niter):
    sprop = S[-1] + n                   # equação (11)
    num = fpost(sprop)                  # valor da posteriori para o estado candidato - vide a Tabela 4
    r = num/den                         # equação (13)
    if random() <= min(r,1):            # vide a nota de rodapé 14
        S.append(sprop)                 # se for aceito, o candidato é anexado ao final da lista S
        den = num                       # evita o cálculo do denominador de r na próxima iteração
    else: S.append(S[-1])               # se o candidato não for aceito, anexa-se o estado atual em S
    v_inc.append(vinc(S[-1]))           # acrescenta ao final de v_inc uma amostra de velocidade
                                        # para o estado recém anexado à lista S - vide a Tabela 5
v_inc_steady = v_inc[Nburn:Niter]       # amostras de velocidade após o burn-in
```

Tabela 6 – Reiteração do quarto ao sexto passo implementada em linguagem Python. Fonte: Autoria própria.

[13] https://youtu.be/Mv58A8m70q0?si=xsN4adPGScvkKcYb. Acesso em 31 de agosto de 2023.
[14] A implementação deste procedimento é relativamente simples. Dado um ponto $u \sim Uniforme\ (0, 1)$, faz-se $s_{t+1} = C$ se $u \leq \min(r, 1)$, senão $s_{t+1} = s_t$, onde C é o candidato, s_t o estado atual e s_{t+1} o próximo estado (HILL e SPALL, 2019).

3.3 Obtenção do intervalo de credibilidade da velocidade questionada

Na última linha da Tabela 6, descartam-se os valores produzidos no transitório inicial a fim de que sejam consideradas no cálculo do intervalo de credibilidade somente as L amostras de velocidade geradas sob condição de estacionariedade dos parâmetros de estado a, b e c. O intervalo de credibilidade p% é simplesmente a porção (central) do histograma (simétrico) que contém p% dessas L amostras[17]. A Tabela 7 apresenta a função empregada na obtenção desse intervalo. Já a Tabela 8 mostra o fechamento global do código implementado em Python.

Função para calcular o intervalo de credibilidade de v_{inc}

```
def Credible_Interval(p, v_inc_steady):
    L = len(v_inc_steady)              # número de amostras após o burn-in
    TL = int(0.5*(1.0-p)*L)            # número de amostras para descartar em cada lado
    vss = np.sort(v_inc_steady)        # amostras de velocidade em ordem crescente
    return [vss[TL], vss[L-TL]]        # intervalo que contém p% das L amostras de velocidade
```

Tabela 7 – Função em Python para o cálculo do intervalo de credibilidade de vinc. Fonte: Autoria própria.

Visualização do intervalo de credibilidade e da mediana de v_{inc}

```
v_inc_median = np.median(v_inc_steady)      # estimativa pontual: mediana das amostras
CI = Credible_Interval(p, v_inc_steady)     # intervalo de credibilidade – vide a Tabela 7
print("v_inc = " + "%.1f" % v_inc_median +  # apresenta os resultados na tela do computador
" km/h (mediana); CI_" + "%.0f" % (100*p) + "% = [" + "%.1f" % CI[0] + ";", "%.1f" % CI[1] + "] km/h")
```

Tabela 8 – Parte final do código implementado em linguagem Python. Fonte: Autoria própria.

3.4 Algoritmo implementado em linguagem Python: versão completa

A Tabela 9 consolida os códigos parciais disponibilizados da Tabela 1 até a 8. Ao interessado em reportar a velocidade questionada através de um intervalo de credibilidade bayesiano, basta executar o script dessa tabela. Antes é preciso somente trocar os valores grifados de vermelho pelos próprios dados.

Implementação em Python para estimar intervalo de credibilidade de velocidade

```
import numpy as np
from scipy.stats import norm
from random import random
m_inc = 77.9
m = np.array([37.3,51.2,57.3,67.4,74.9,84.4,97.6,113.2,51.0,62.6,85.4,83.6,109.3,42.1,52.2,60.8,81.0,87.3,98.9])
v = np.array([39.3,50.6,60.1,70.5,77.4,88.8,98.8,115.7,53.2,64.5,87.0,84.8,110.8,43.6,53.8,61.8,83.1,88.9,101.5])
p = 0.95
N = len(v)
M = np.matrix(m).T
V = np.matrix([np.ones(N).tolist(),v]).T
a_chap, b_chap = np.linalg.pinv(V)*M
e = M - V*np.matrix([[a_chap[0,0]],[b_chap[0,0]]])
MSE = e.T*e/(N-2.0)
c_chap = 0.5*np.log10(MSE)
s0 = np.array([a_chap[0,0],b_chap[0,0],c_chap[0,0]])
def fprior(s):
    priormean = np.array([0.0, 1.0, 0.3])
    priorstd  = np.array([10.0, 0.1, 1.0])
    return np.prod(norm.pdf(s,priormean,priorstd))
def fpost(s):
    return np.prod(norm.pdf(m,s[0]+s[1]*v,10**s[2]))*fprior(s)
def vinc(s):
    return (m_inc-s[0]-np.random.normal(0.0,10**s[2]))/s[1]
S = []
v_inc = []
S.append(s0)
v_inc.append(vinc(s0))
den = fpost(s0)
Nburn = 10**4
Niter = 10**5 + Nburn
tau = np.array([1.0, 0.01, 0.1])/3.0
for n in np.random.multivariate_normal([0.0, 0.0, 0.0], np.diag(tau**2), Niter):
    sprop = S[-1] + n
    num = fpost(sprop)
    r = num/den
    if random() <= min(r,1):
        S.append(sprop)
        den = num
    else: S.append(S[-1])
    v_inc.append(vinc(S[-1]))
v_inc_steady = v_inc[Nburn:Niter]
def Credible_Interval(p, v_inc_steady):
    L = len(v_inc_steady)
    TL = int(0.5*(1.0-p)*L)
    vss = np.sort(v_inc_steady)
    return [vss[TL], vss[L-TL]]
v_inc_median = np.median(v_inc_steady)
CI = Credible_Interval(p, v_inc_steady)
print("v_inc = " + "%.1f" % v_inc_median + " km/h (mediana); CI_" + "%.0f" % (100*p) + "% = [" + "%.1f" % CI[0] + ";",
"%.1f" % CI[1] + "] km/h")
```

Tabela 9 – Versão completa do script desenvolvido em linguagem Python. Fonte: Autoria própria.

[15] *O método de amostragem introduzido por Metropolis et al (1953) foi generalizado por Hastings (1970). Desde então, ele é visto como um caso particular do algoritmo Metropolis-Hastings (GELMAN et al, 2013).*
[16] *Atente-se que o novo estado efetivo não necessariamente é diferente do anterior. Vide a Seção 3.2.5.*
[17] *https://courses.cs.washington.edu/courses/cse312/20su (Lecture 22). Acesso em 03 de janeiro de 2024.*

4. Resultados

Neste trabalho, a estratégia para validar o script desenvolvido consiste em verificar se ele produz resultados similares aos divulgados por Mieremet et al (2018). O trecho de código específico para gerar os gráficos mostrados nesta seção é um tanto rotineiro e foi omitido na Tabela 9 a fim de não dificultar a percepção da essência do algoritmo.[18] Seguem os principais resultados obtidos.

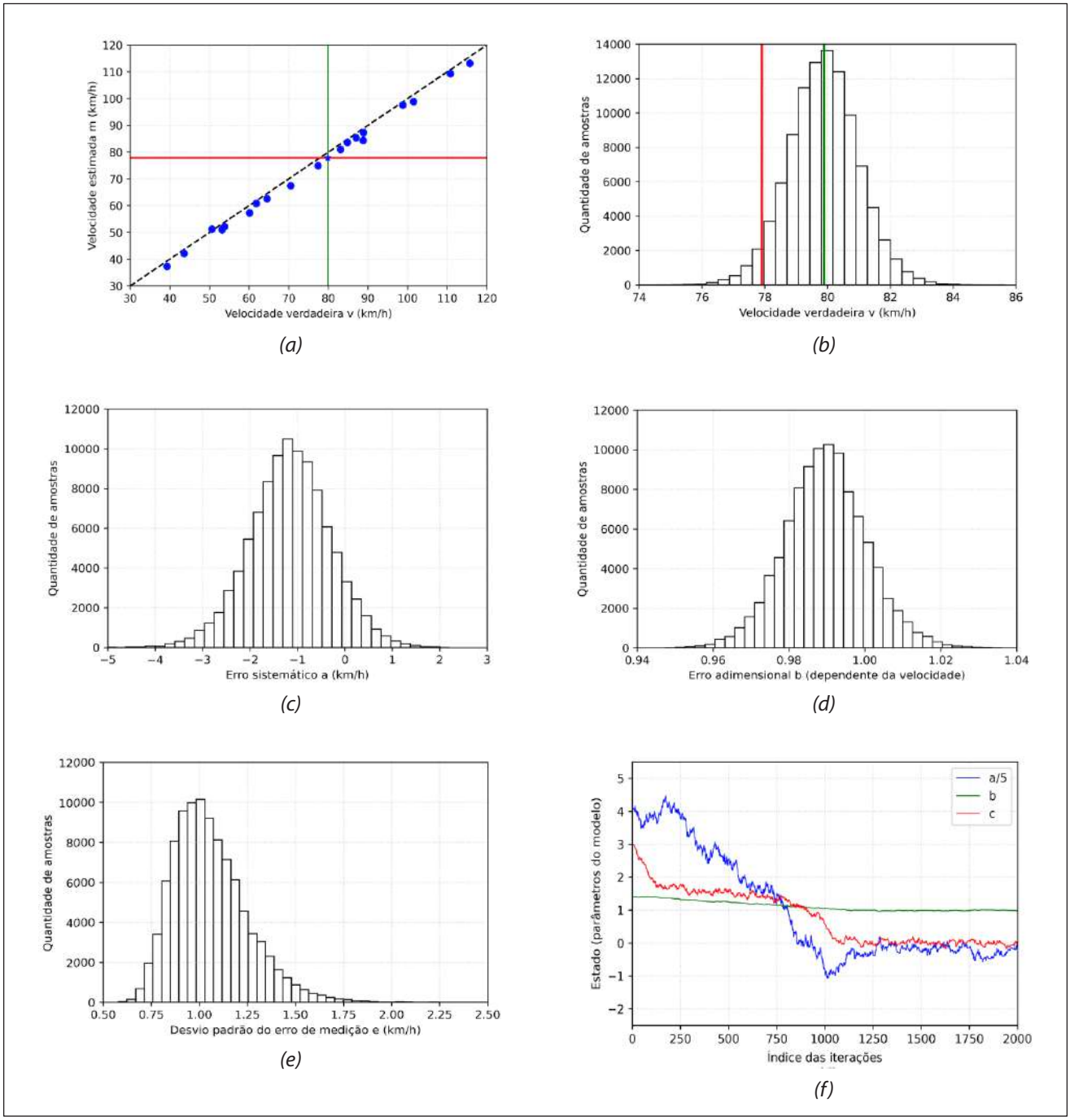

Figura 2 – (a) Diagrama de dispersão mostrando a correspondência entre as velocidades estimadas e as verdadeiras. Os círculos azuis representam as medidas de referência. A linha horizontal vermelha é a velocidade estimada para o incidente sob investigação. Já a vertical verde é a mediana das amostras geradas pelo método MCMC. O tracejado diagonal denota o caso ideal em que a velocidade estimada e a verdadeira são exatamente iguais. (b) Histograma para a velocidade verdadeira. A linha verde é a mediana das amostras e a vermelha é a estimativa não-corrigida da velocidade de interesse. (c) Histograma do erro sistemático. (d) Histograma do erro dependente da velocidade. (e) Histograma do desvio padrão do erro de medição. (f) Evolução dos parâmetros do modelo a partir do estado (20, 1.4, 3). Nota-se que, depois de 1100 iterações, o estado parece já ter convergido para o regime estacionário. Fonte: Autoria própria.

[18] *No código da Tabela 9, as séries para a, b e c podem ser obtidas da lista S acrescentando-se depois do laço os comandos a = np.array([s[0] for s in S]), b = np.array([s[1] for s in S]) e c = np.array([s[2] for s in S]), respectivamente.*

Execução	Estimativa pontual	Intervalo de credibilidade 95%	Tempo de execução
1	79.9	77.7–82.1	9.38 segundos
2	79.9	77.7–82.1	9.40 segundos
3	79.9	77.7–82.1	9.35 segundos
4	79.9	77.7–82.1	9.55 segundos
5	79.9	77.6–82.1	9.37 segundos
6	79.9	77.7–82.1	9.50 segundos
7	79.9	77.7–82.1	9.30 segundos
8	79.9	77.6–82.1	9.47 segundos
9	79.9	77.6–82.1	9.34 segundos
10	79.9	77.7–82.1	9.56 segundos

Tabela 10 – Resultados numéricos para dez execuções do script da Tabela 9. Foi empregado um notebook da Apple, com sistema operacional macOS Sonoma 14.1.1, chip M2 (CPU e GPU de 8 e 10 núcleos, respectivamente), memória unificada de 8 GB e SSD de 256 GB. Fonte: Autoria própria.

Tipo de intervalo	90%	95%	97.5%	99%
Credibilidade	78.0–81.7	77.7–82.1	77.3–82.5	76.8–82.9
Confiança	78.0–81.6	77.6–82.0	77.3–82.4	76.8–82.8

Tabela 11 – Os intervalos de credibilidade resultam da execução do script da Tabela 9 para diferentes níveis de p. Já os intervalos de confiança foram calculados pela fórmula da incerteza expandida de T-Student disponível em (HOOGEBOOM et al, 2015). Fonte: Autoria própria.

Estado inicial	90%	95%	97.5%	99%
(-1.14, 1.0, 0.5)	78.1–81.7	77.6–82.1	77.3–82.5	76.8–82.9
(0, 1, 0)	78.0–81.7	77.7–82.1	77.3–82.5	76.8–83.0
(20, 1.4, 3)	78.0–81.7	77.7–82.1	77.3–82.5	76.8–83.0

Tabela 12 – Intervalos de credibilidade fornecidos pelo código da Tabela 9 considerando estados iniciais diferentes do canônico (vide a Seção 3.2.1). Fonte: Autoria própria.

Distribuição *a priori*	90%	95%	97.5%	99%
$(\mu_a, \mu_b, \mu_c) = (0, 1, 0.3)$ $(\sigma_a, \sigma_b, \sigma_c) = (100, 1, 2)$	78.0–81.7	77.6–82.1	77.3–82.5	76.8–82.9
$(\mu_a, \mu_b, \mu_c) = (0, 1, 0.3)$ $(\sigma_a, \sigma_b, \sigma_c) = (1, 0.01, 0.4)$	78.0–81.6	77.6–82.0	77.2–82.4	76.7–82.8
$(\mu_a, \mu_b, \mu_c) = (0, 1, 0.3)$ $(\sigma_a, \sigma_b, \sigma_c) = (0.1, 0.001, 0.1)$	74.6–81.5	73.8–82.2	73.2–82.8	72.4–83.6

Tabela 13 – Intervalos de credibilidade obtidos com o script da Tabela 9 para diferentes níveis de "informatividade" da priori gaussiana. Fonte: Autoria própria.

Passo típico	90%	95%	97.5%	99%
$\tau/8$	78.1–81.7	77.7–82.1	77.4–82.4	76.8–82.9
$\tau/2$	78.0–81.7	77.6–82.1	77.3–82.5	76.8–82.9
2τ	78.0–81.7	77.6–82.1	77.3–82.5	76.8–82.9
8τ	78.1–81.7	77.6–82.1	77.3–82.5	76.8–82.9

Tabela 14 – Intervalos de credibilidade gerados pelo script da Tabela 9 para diferentes passos típicos. Fonte: Autoria própria.

5. Discussão

No diagrama de dispersão da Figura 2.a, a preponderância de pontos abaixo da reta tracejada evidencia uma tendência de subestimação da velocidade real. Assim sendo, o resultado na Figura 2.b é consistente no sentido de que o algoritmo implementado gerou amostras para a velocidade real v_{inc} *predominantemente acima de sua estimativa primária* m_{inc} *(linha vermelha). Em relação às Figuras 2.c, 2.d e 2.e, o comportamento gaussiano do erro sistemático e do erro dependente da velocidade, bem como o perfil log-normal do desvio padrão do erro de medição, estão em plena concordância com os pressupostos do modelo bayesiano adotado - na Seção 3.2, vide descrição dos parâmetros da equação (1). Na Figura 2.f, visualizam-se os parâmetros do modelo convergindo para a estacionariedade mesmo o processo iterativo tendo iniciado em um estado relativamente distante do sugerido pela regressão linear simples da Seção 3.2.1, o que é um comportamento altamente desejável. Esses gráficos da Figura 2 são notoriamente semelhantes aos disponibilizados em (MIEREMET et al, 2018), sendo esta uma importante conformidade de validação do algoritmo proposto, uma vez que foram empregados os mesmos dados de entrada (vide a Seção 3.1).*

A saída do algoritmo implementado (vide a Seção 3.4) consiste em um intervalo de credibilidade bayesiano e um representante pontual[19] para a velocidade real/verdadeira (mediana das amostras). De forma geral, os resultados numéricos da Tabela 10 até a 14 evidenciam a significativa robustez dessa saída (flutuação desprezível). Na Tabela 10, em particular, é nítida a regularidade de comportamento da implementação, tanto em termos da própria saída quanto em relação à latência computacional, sendo que esta, em nenhuma das execuções, sequer atingiu o patamar de dez segundos - essa rapidez de processamento computacional parece satisfatória à utilização prática pelos Institutos de Criminalística. Na Tabela 11, as diferenças numéricas entre o intervalo de confiança (frequentista) e o de credibilidade (bayesiano) são inferiores a 0,1 km/h - como esses dois intervalos são conceitos fundamentalmente distintos, é de se esperar que exista alguma discrepância entre eles. A Tabela 12 sugere que a escolha exata do estado inicial tende a não influenciar significativamente a saída do algoritmo - a despeito disso, caso o "ponto" de partida no espaço paramétrico seja demasiado irrealista, o MCMC pode não convergir adequadamente (MIEREMET et al, 2018). Na Tabela 13, percebe-se que a "informatividade" da priori gaussiana praticamente não afeta a saída, havendo um alargamento no intervalo de credibilidade apenas quando ela é substancialmente exagerada (vide a terceira linha da tabela). Por fim, na Tabela 14, os intervalos fornecidos sofreram variação marginal para a ampla faixa de passos típicos testados – não obstante, uma vez que a magnitude desse passo típico controla a taxa de aceite/rejeição dos estados propostos e a literatura aponta como ideal uma aceitação de 23% para a abordagem MCMC, recomenda-se empregar um passo típico em torno de 2.τ (MIEREMET et al, 2018).

Em suma, a saída da implementação em linguagem Python apresentou estabilidade em relação às diversas opções de setup testadas. Além do mais, tanto graficamente quanto numericamente, os achados foram praticamente idênticos aos divulgados em (MIEREMET et al, 2018). Enfim, a implementação proposta (vide a Tabela 9) é uma ferramenta confiável para calcular intervalos de credibilidade bayesianos nas perícias de estimação de velocidade por vídeo.

6. Conclusão

Neste trabalho, implementou-se em Python uma abordagem MCMC da literatura para calcular intervalos de credibilidade na estimativa de velocidade por vídeo. Os testes revelaram significativa estabilidade da saída e latência computacional satisfatória para uso prático, bem como consistência com o estudo norteador da implementação. Esses achados consolidam o código desenvolvido como uma ferramenta confiável de análise da incerteza associada à velocidade estimada por vídeo. Ademais, na medida em que captura adequadamente a estrutura do erro a partir de imagens de referência, a implementação pode ser empregada para corrigir de forma automática a velocidade estimada para um valor pontual com maior chance de se tratar da velocidade verdadeira/real. Enfim, o script desenvolvido mostrou-se plenamente apto à utilização pelos Institutos de Criminalística, de modo que os Peritos Criminais podem lançar mão dele para reportar com segurança intervalos de credibilidade bayesianos nos laudos periciais de cálculo de velocidade por vídeo. A ferramenta disponibilizada neste artigo é limitada ao mensurando velocidade. Em trabalhos futuros, pretende-se estendê-la a outras medidas de interesse forense (e.g., altura de indivíduos em vídeos), uma vez que intervalos de credibilidade (estatística bayesiana) parecem menos susceptíveis a interpretação equivocada do que intervalos de confiança (estatística frequentista).

Agradecimento

Os autores agradecem a todos que colaboraram indiretamente com este trabalho.

[19] *A mediana das amostras nada mais é do que a versão corrigida da estimativa primária (MIEREMET et al, 2018) – na Figura 2, vide retas vermelhas e verdes. Para realizar essa correção, o algoritmo extrapola o erro subjacente à estimação confrontando as duas coleções pareadas de velocidades (vide a Seção 3.1)*

Referências

BORGES, Cíntia. Delegado: laudo técnico que beneficia médica "não tem valor". MidiaNews, Cuiabá, 07 de jun. de 2018. Disponível em: <www.midianews.com.br>. Acesso em: 17 de maio de 2023.

BOYD, Stephen; VANDENBERGHE, Lieven. Applied Linear Algebra: Vectors, Matrices, and Least Squares. Cambridge University Press, 2018.

BUKHARI, Faisal; DAILEY, Matthew. Automatic Radial Distortion Estimation from a Single Image. Journal of Mathematical Imaging and Vision, vol. 45 (1), p. 31 – 45, 2013.

EPSTEIN, Brandon; WESTLAKE, Bryce Garreth. Determination of Vehicle Speed from Recorded Video Using Reverse Projection Photogrammetry and File Metadata. Journal of Forensic Sciences, vol. 64, p. 1523-1529, 2019.

FILHO, José Rocha de Carvalho. Notas de Aula de Fotogrametria. IPOG, 2022.

GELMAN, A.; CARLIN, J.B.; STERN, H.S.; DUNSON, D.B.; VEHTARI, A.; RUBIN, D.B. Bayesian Data Analysis. Third edition, Chapman & Hall/CRC, 2013.

HASTINGS, W. K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. Biometrika, vol. 57 (1), p. 97-109, 1970.

HILL, S. D.; SPALL, J. C. Stationarity and Convergence of the Metropolis-Hastings Algorithm: Insights into Theoretical Aspects. IEEE Control Systems Magazine, vol. 39 (1), p. 56-67, 2019.

HOOGEBOOM, Bart; ALBERINK, Ivo. Measurement Uncertainty When Estimating the Velocity of an Allegedly Speeding Vehicle from Images. Journal of Forensic Sciences, vol. 55 (5), p. 1347-1351, 2010.

HOOGEBOOM, Bart; ALBERINK, Ivo; VRIJDAG, Derk. Photogrammetry in Digital Forensics. In: HO, Anthony T.S.; LI, Shujun. (org.). Handbook of Digital Forensics of Multimedia Data and Devices. Wiley-IEEE Press, 2015. p. 185-218.

KIM, Jong-Hyuk; OH, Won-Taek; CHOI, Ji-Hun; PARK, Jong-Chan. Reliability Verification of Vehicle Speed Estimate Method in Forensic Videos. Forensic Science International, vol. 287, p. 195-206, 2018.

LIN, Huei-Yung; LI, Kun-Jhih; CHANG, Chia-Hong. Vehicle Speed Detection from a Single Motion Blurred Image. Image and Vision Computing, vol. 26 (10), p. 1327-1337, 2008.

METROPOLIS, N.; ROSENBLUTH, A.W.; ROSENBLUTH, M.N.; TELLER, A.H.; TELLER, E. Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics, vol. 21 (6), p. 1087 – 1092, 1953.

MIEREMET, Arjan; ALBERINK, Ivo; HOOGEBOOM, Bart; VRIJDAG, Derk. Probability Intervals of Speed Estimations from Video Images: The Markov Chain Monte Carlo Approach. Forensic Science International, vol. 288, p. 29-35, 2018.

PENNSTATE. Eberly College of Science - STAT 462, c2018. What is the Common Error Variance? Disponível em: <https://online.stat.psu.edu/stat462/node/94/>. Acesso em: 23 de jun. de 2023.

PÉREZ, F.; GRANGER, B. E.; HUNTER, J. D. Python: An Ecosystem for Scientific Computing. Computing in Science & Engineering, vol. 13 (2), p. 13-21, 2011.

SCHNEIDER, C.A.; RASBAND, W.S.; ELICEIRI K.W. NIH Image to ImageJ: 25 Years of Image Analysis. Nature Methods, vol. 9 (7), p. 671-675, 2012.

SCHOOT, Rens van de et al. Bayesian statistics and modelling. Nature Reviews Methods Primers, vol. 1, p. 01-27, 2021.

WONG, T.W.; TAO, C.H.; CHENG, Y.K.; WONG, K.H.; TAM, C.N. Application of Cross-Ratio in Traffic Accident Reconstruction. Forensic Science International, vol. 235, p. 19-23, 2014.

XIAO, Jianyu; LI, Shancang; XU, Qingliang. Video-Based Evidence Analysis and Extraction in Digital Forensic Investigation. IEEE Access, vol. 7, p. 55432-55442, 2019.

COMO CITAR ESTE ARTIGO

GROSS, T. J.; MARUYAMA, F. H.; FESTA, A. V. A Python algorithm for reporting credibility intervals in video-based speed estimation forensic reports. Perícia Federal, v. 1, n. 54, p. 51–56, 2024. https://doi.org/10.29327/266815.1.56-2

GROSS, T. J. et al. (2026) https://doi.org/10.29327/266815.1.56-2