Gait & Posture 41 (2015) 44–45

Contents lists available at ScienceDirect

Gait & Posture

journal homepage: www.elsevier.com/locate/gaitpost

Comments on "Ellipse area calculations and their applicability in posturography" (Schubert and Kirchner, vol.39, pages 518-522, 2014)

Marcos Duarte *

Biomedical Engineering, Federal University of ABC, Santo André, Brazil

ARTICLE INFO

Article history:
Received 22 July 2014
Received in revised form 1 August 2014
Accepted 11 August 2014

Keywords:
Balance
Posture
Posturography
Signal processing
Statistics

Dear Editor:

I would like to congratulate the authors for the article regarding the calculation of the so-called ellipse area [1]. As the authors indicated, the algorithms employed to calculate the area of the 95% prediction ellipse using the chi-square or the Rayleigh distribution are in fact only exact when the number of samples of the bivariate variable tends to infinite (when each univariate variable is assumed to have a normal distribution) [2]. As the authors also observed, for a typical data size in posturography, 30 s of data sampled at 100 Hz (3000 samples), this approximation is probably good enough, since the error is only 0.1%. The problem appears when the data size is much less: for 100 samples, the error is 2.5%, and for 10 samples, the error is 26%. These last two cases are unlikely scenarios in posturography, but possible for an unadvised user; and besides, the prediction ellipse area can be employed in any other data analysis. The authors described the calculation (see the supplementary data in [1]) but did not publish any algorithm to compute the exact 95% prediction ellipse area. They only made available the algorithm with the known approximation (i.e., they used the chi-square distribution and not the F distribution for the exact calculation). To fill this lacuna, at the end of this letter it is presented a computer program to calculate the exact 95% prediction ellipse area [2] for a Matlab-like environment software and the same algorithm implemented in the Python language, a free and open source software. The program input, the variable 'data', has '*n*' rows (the number of samples) and two columns for a bivariate data. In fact, this computer program is written to also calculate the hypervolume of a hyper-ellipsoid (with p dimensions) if 'data' has *p* columns. Briefly, the volume of the hyper-ellipsoid is calculated with the same equation for the volume of a *p*-dimensional ball (http://en.wikipedia.org/wiki/Volume_of_an_n-ball) with the radius replaced by the semi-axes of the hyper-ellipsoid. The variable 'hypervolume' contains the calculated ellipse area for 2-D data or the hypervolume for *p*-dimensional data.

The webpage 'Prediction ellipse and prediction ellipsoid' at the website https://github.com/demotu/BMC contains a detailed explanation about the prediction ellipse and a more complete code written in Python to compute and plot the results and other variables.

```
% Matlab code to calculate the hypervolume
  of the exact 95% prediction hyper-ellipsoid:
[n, p] = size(data);                           % 2-D array dimensions
covar = cov(data);                             % covariance matrix of data
[U, S, V] = svd(covar);                        % singular value decomposition
f95 = finv(.95,p,n − p)*(n − 1)*               % F 95 percent point function
  p*(n + 1)/n/(n − p);
saxes = sqrt(diag(S)*f95);                     % semi-axes lengths
hypervolume = pi^(p/2)/gamma
  (p/2 + 1)*prod(saxes)
```

* Tel.:+55 11 23206363.
E-mail address: duartexyz@gmail.com

http://dx.doi.org/10.1016/j.gaitpost.2014.08.008
0966-6362/© 2014 Elsevier B.V. All rights reserved.

```
# Python code to calculate the
  hypervolume of the exact 95%
  prediction hyper-ellipsoid:
import numpy as np                        # import Numpy package
from scipy.stats import f as F            # import F distribution
from scipy.special import gamma           # import Gamma function
n, p = np.asarray(data).shape             # 2-D array dimensions
cov = np.cov(data, rowvar = 0)            # covariance matrix of data
U, s, Vt = np.linalg.svd(cov)             # singular value decomposition
f95 = F.ppf(.95,p,n - p)*(n - 1)*        # F 95 percent point function
  p*(n + 1)/n/(n - p)
saxes = np.sqrt(s*f95)                    # semi-axes lengths
hypervolume = np.pi**(p/2)/
  gamma(p/2 + 1)*np.prod(saxes)
hypervolume
```

Conflicts of interest statement

There author declares no conflicts of interest.

References

[1] Schubert P, Kirchner M. Ellipse area calculations and their applicability in posturography. Gait Posture 2014;39:518–22.

[2] Chew V. Confidence, prediction, and tolerance regions for the multivariate normal distribution. J Am Stat Assoc 1966;61:605–17.