Supplementary Material 2

Article Title: Effects of meditation practice on body balance training

Supplementary methods 2: Code for the analysis of the outcome

Monopodal center of mass fluctuations

Algorithm 1 shows the Stan language code for the Bayesian linear model we used to esteem the improvement (percentage change after training vs before training) of center of mass fluctuations in the control and meditation group while standing on one foot on a flat steady surface.

Balance on fit-ball

Algorithm 2 shows the Stan language code for the Bayesian Cox Proportional Hazard model that we used to esteem the β coefficient of the Cox model ($H(t;\boldsymbol{Z}) = H_0(t)e^{\boldsymbol{Z}'\boldsymbol{\beta}}$, see Methods in the main article for details). From β, we then calculated the esteem of the fall rate proportion (hazard rate ratio, $100 \cdot e^{\beta}$), that is the fall rate of the meditation group as percentage of the fall rate of the control group reported in the article results.

Algorithm 1 Stan programming language code for the Bayesian linear model

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
transformed data {
  // Standardize the data:
  vector[N] zy;
  real y_m;
  real y_sd;
  y_m = mean(y);
  y_sd = sd(y);
  zy = (y - y_m) / y_sd;
}
parameters {
  real z_alpha;
  real z_delta;
  real<lower=0> z_sigma_ctrls;
  real<lower=0> z_sigma_meds;
  real<lower=0> nu_minus_one_ctrls;
  real<lower=0> nu_minus_one_meds;
}
transformed parameters {
  real<lower=0> nu_ctrls;
  real<lower=0> nu_meds;
  nu_ctrls = nu_minus_one_ctrls+1;
  nu_meds = nu_minus_one_meds+1;
  real z_delta_sigma;
  z_delta_sigma = z_sigma_meds - z_sigma_ctrls;
  real delta_nu;
  delta_nu = nu_meds - nu_ctrls;
}
model {
  z_alpha ~ normal(0, 5);
  z_delta  ~ normal(0, 5);
  z_sigma_ctrls ~ uniform(1.0E-3, 1.0E+3);
  z_sigma_meds ~ uniform(1.0E-3, 1.0E+3);
  nu_minus_one_ctrls ~ exponential(1/29.0);
  nu_minus_one_meds ~ exponential(1/29.0);
  zy ~ student_t(nu_ctrls + delta_nu * x, z_alpha + z_delta * x, z_sigma_ctrls + z_delta_sigma * x);
}
generated quantities {
  real mu_ctrls;
  real mu_meds;
  real diff_CM;
  real<lower=0> sigma_ctrls;
  real<lower=0> sigma_meds;
  // Transoforms back to original scale
  mu_ctrls = (z_alpha * y_sd) + y_m;
  mu_meds = ((z_alpha + z_delta) * y_sd) + y_m;
  sigma_ctrls = z_sigma_ctrls * y_sd;
  sigma_meds = (z_sigma_meds) * y_sd;
  diff_CM = z_delta * y_sd;
}
```

Algorithm 2 Stan programming language code for the Bayesian Cox Proportional Hazard model

```
data {
  int<lower=0> K;          // num covariates
  int<lower=0> N;          // num uncensored obs
  vector[N] t;             // event time (non-strict decreasing)
  matrix[N, K] x;          // covariates for uncensored obs
  int N_c;                 // num censored obs
  real<lower=t[N]> t_c;    // censoring time
  matrix[N_c, K] x_c;      // covariates for censored obs
}
parameters {
vector[K] beta;          // slopes (no intercept)
}
model {
  beta ~ normal(0, 2);
  vector[N] log_theta = x * beta;
  vector[N_c] log_theta_c = x_c * beta;
  real log_denom = log_sum_exp(log_theta_c);
  for (n in 1:N) {
    log_denom = log_sum_exp(log_denom, log_theta[n]);
    target += log_theta[n] - log_denom;   // log likelihood
  }
}
```