Online Supplement

"Unpacking Habit With Bayesian Mixed Models: Dynamic Approach to Health Behaviors With Interchangeable Elements, Illustrated Through Multiple Sun Protection Behaviors"

Yuelin Li, Elizabeth Schofield, Jennifer L. Hay

Table of Contents:

Please feel free to contact the corresponding author if you have any questions.

1. R syntax code for fitting the Bayesian mixed model for sunscreen use

The R syntax code below takes the raw counts in sunscreen use in the Appendix (saved as a file called 'sunscn_rawCnt.csv') and turns them into a long format suitable for model fitting by Stan. Each row represents one assessment occasion so that one study id is repeated across several repeated assessments. Next, the fixed and random effect matrices are specified and entered into a data frame so that it can be analyzed by Stan. Note that the variable names in this data frame must be identical to variable names specified in the Stan syntax in the next section. The matrices may appear complicated at first, but they help make the Stan syntax easily scalable from the example of one single behavior to all four sun protection behaviors. That is, the Stan syntax in the next section can be easily adapted to all four behaviors, as well as testing specific model assumptions (e.g., separate random covariances in men and women).

```
# Raw data in the Appendix saved as a csv file called "sunscn_rawCnt.csv"
sunscn.dat <- read.csv(file = "sunscn_rawCnt.csv", skip = 4, header = TRUE)
sunscn.dat <- as.data.frame(sunscn.dat)

# Define a function to convert raw data table into long-format
expand.scn <- function ( data_row ) {
 sunny0.scn0 <- data_row["sunny0.scn0"]  # not sunny, no sunscreen use
 sunny0.scn1 <- data_row["sunny0.scn1"]  # not sunny, yes sunscreen use
 sunny1.scn0 <- data_row["sunny1.scn0"]  # sunny, no sunscreen
 sunny1.scn1 <- data_row["sunny1.scn1"]  # sunny, yes sunscreen
 # each row corresponds to outcome vector yy
 yy <- rep(c(0, 1, 0, 1), times = c(sunny0.scn0, sunny0.scn1,
                                   sunny1.scn0, sunny1.scn1))
 # stop and print error if yy vector mismatches data_row
 stopifnot ("Error in expanding y" = length(yy) ==
  sum(data_row[c("sunny0.scn0", "sunny0.scn1",
                "sunny1.scn0", "sunny1.scn1")]))
 # create variable sunny.hot
 sunny.hot <- rep(c(0, 0, 1, 1), times = c(sunny0.scn0, sunny0.scn1,
                                 sunny1.scn0, sunny1.scn1))
 # final data in long format ready
 res <- cbind(ID = data_row["ID"], male = data_row["male"],
              sunny.hot = sunny.hot, y = yy)
 invisible(data.frame(res))
}

library(data.table)
sunscn.df <- apply(sunscn.dat, MAR = 1, FUN = expand.scn)
sunscn.df <- data.table::rbindlist(sunscn.df)
# Write long data out as a backup
write.csv(sunscn.df, file = "sunscn4sas.csv", row.names = FALSE)

library("rstan")
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# specify fixed effect X and random effects Xran
X <- model.matrix(~ 1 + male*sunny.hot, sunscn.df)
print(names(X))   # how the variables are arranged in X
X <- unname(X)
Xran <- unname(model.matrix(~ 1 + sunny.hot, sunscn.df))
attr(X, "assign") <- NULL     # optional, NULL to avoid error
attr(Xran, "assign") <- NULL
# indicator variable for separate random effects for men and women
maleS <- sunscn.dat$male

# prepare input data for Stan, must match names in the Stan syntax
standat <- list(
    y = sunscn.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(sunscn.df$ID)), # needed b/c char(), not factor
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(sunscn.df$ID, unique(sunscn.df$ID)),
    # maleS has length 59, not full length of N
    maleS = maleS + 1   # female = 1; male = 2 to index Chol matrices
    )

sunscn_male.stan <- stan(file = "./stan/model1_male.stan",
     model_name = "rand_int", data = standat,
     seed = 271828, init = "random",
     control = list(adapt_delta = 0.95, max_treedepth = 10),
     thin = 10, iter = 25000, warmup = 5000, chains = 4,
     # refresh = 0 to suppress printing of iterations
     refresh = 0)

print(sunscn_male.stan, pars = c("beta", "sigma_u", "Sigma_u","L_u"),
      probs = c(0.025, 0.50, 0.975), digits_summary = 2)
```

2. Stan syntax code for modeling sunscreen use

   The Stan syntax below is adapted from the Sorensen et al. (2016) paper, which contains detailed explanations line by line. We used the "`target +=`" syntax in Stan because it is required by the `bridgesampling` package in R to calculate the Bayes factor. Details on Bayes factor calculations are summarized in the next section.

```
// Stan syntax file for fitting separate random effects in men and women
// Adapted from Sorensen et al. (2016), listing 10, available at:
// http://www.ling.uni-potsdam.de/~vasishth/pdfs/SorensenHohensteinVasishth2016.pdf
data {
  int<lower=0> N;                  // n trials
  int<lower=1> P;                  // n fixefs
  int<lower=0> J;                  // n subjects
  int<lower=1> n_u;                // n subj ranefs
  int<lower=1,upper=J> subj[N];    // subject indicator
  int<lower=1,upper=2> maleS[J];   // if jth subject is male
  row_vector[P] X[N];              // fixed effects design matrix
  row_vector[n_u] Z_u[N];          // subj ranef design matrix
  int<lower=0,upper=1> y[N];       // dependent variable
}
parameters {
  vector[P] beta;                    // fixed effects coefs
  cholesky_factor_corr[n_u] L_u[2]; // Chol factor of subj ranef cor matrix
  vector<lower=0>[n_u] sigma_u[2];   // subj ranef std
  // declare a 2 by J array of 2-dimensional column vectors
  // see https://mc-stan.org/docs/2_18/
  // reference-manual/array-data-types-section.html
  vector[n_u] z_u[2,J];              // subj ranef, zu[1,] for women
}
transformed parameters {
  vector[n_u] u[J];                  // subj individual ranefs
  matrix[n_u,n_u] Sigma_u[2];        // subj ranef cov matrix
  for (k in 1:2) {                   // Cholesky for men and women
    Sigma_u[k] = diag_pre_multiply(sigma_u[k], L_u[k]);
  }
  for(j in 1:J)     {
    u[j] = Sigma_u[maleS[j]] * z_u[maleS[j], j]; // LKJ for ranef
    }
}
model {
  // priors
  target += lkj_corr_cholesky_lpdf(L_u[1] | 3.0);
  target += lkj_corr_cholesky_lpdf(L_u[2] | 3.0);

  for (j in 1:J) {
    // For each slice of z_u, generate a normally distributed variable
    target += normal_lpdf( z_u[1, j] | 0, 1 );
    target += normal_lpdf( z_u[2, j] | 0, 1 );
    }

  target += cauchy_lpdf(beta[1] | 0, 10);     // intercept prior
  // http://www.stat.columbia.edu/~gelman/research/published/priors11.pdf

  for (j in 2:P) {
    target += cauchy_lpdf(beta[j] | 0, 2.5); // prior all slopes
    }

  // target += notation to get Bayes factor with library(bridgesampling)
  for (i in 1:N)  {
   target += bernoulli_logit_lpmf(y[i] | X[i] * beta + Z_u[i] * u[subj[i]]);
  }
}
// generated quantities {
// OPTIONAL: skip to make Stan run faster.
// These can be used for additional model diag, see, e.g.,
// http://mc-stan.org/loo/articles/loo2-with-rstan.html
//  vector[N] log_lik;
//  vector[N] y_logit;
//  matrix[n_u,n_u] corr_f;       // corrleation for females
//  matrix[n_u,n_u] corr_m;
//  corr_f = tcrossprod(L_u[1]);  // square of Cholesky factor is corr
//  corr_m = tcrossprod(L_u[2]);
//for (i in 1:N)  {
//  y_logit[i] = X[i] * beta + Z_u[i] * u[subj[i]];
//  log_lik[i] = bernoulli_logit_lpmf(y | y_logit[i]);
//  }
//}
// make sure the Stan syntax file ends with a blank line
```

3. R syntax code for Bayes factor calculation and other model diagnostics

```
library("rstan")
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# Full model
X <- model.matrix(~ 1 + male*sunny.hot, sunscn.df)
print(names(X))
X <- unname(X)
Xran <- unname(model.matrix(~ 1 + sunny.hot, sunscn.df))
attr(X, "assign") <- NULL
attr(Xran, "assign") <- NULL

standat <- list(
    y = sunscn.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(sunscn.df$ID)), # needed b/c char(), not factor
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    maleS = maleS,
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(sunscn.df$ID, unique(sunscn.df$ID))
    )

sunscnFull.stan <- stan(file = "./stan/model1_male.stan",
     model_name = "rand_int", data = standat,
     seed = 271828, init = "random",
     control = list(adapt_delta = 0.95),
     # refresh = 0 to suppress printing of iterations
     refresh = 0, thin = 10, iter = 25000, warmup = 5000, chains = 4)

print(sunscnFull.stan, pars = c("beta", "sigma_u"),
      probs = c(0.025, 0.50, 0.975), digits_summary = 3)
###
# sunny.hot random slopes dropped
###
X <- model.matrix(~ 1 + male * sunny.hot, sunscn.df)
print(names(X))
X <- unname(X)
Xran <- unname(model.matrix(~ 1, sunscn.df))
attr(X, "assign") <- NULL
attr(Xran, "assign") <- NULL

standat <- list(
    y = sunscn.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(sunscn.df$ID)),
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    maleS = maleS,
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(sunscn.df$ID, unique(sunscn.df$ID))
    )

sunscn.dropSunny.stan <- stan(file = "./stan/model1_male.stan",
     model_name = "rand_int", data = standat,
     init = "random",
     control = list(adapt_delta = 0.95),
     # refresh = 0 to suppress printing of iterations
     refresh = 0, thin = 10, iter = 25000, warmup = 5000, chains = 4)

###
# random intercepts dropped
###
X <- model.matrix(~ 1 + male * sunny.hot, sunscn.df)
print(names(X))
X <- unname(X)
Xran <- unname(model.matrix(~ -1 + sunny.hot, sunscn.df))
attr(X, "assign") <- NULL
attr(Xran, "assign") <- NULL

standat <- list(
    y = sunscn.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(sunscn.df$ID)),
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    maleS = maleS,
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(sunscn.df$ID, unique(sunscn.df$ID))
    )

sunscn.noIntcpt <- stan(file = "./stan/model1_male.stan",
     model_name = "no_rand_int", data = standat,
     init = "random",
     control = list(adapt_delta = 0.95),
     # refresh = 0 to suppress printing of iterations
     refresh = 0, thin = 10, iter = 25000, warmup = 5000, chains = 4)

###
# fixed male:sunny.hot dropped
###
X <- model.matrix(~ 1 + male + sunny.hot, sunscn.df)
# print(names(X))
X <- unname(X)
Xran <- unname(model.matrix(~ 1 + sunny.hot, sunscn.df))
attr(X, "assign") <- NULL
attr(Xran, "assign") <- NULL

standat <- list(
    y = sunscn.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(sunscn.df$ID)),
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    maleS = maleS,
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(sunscn.df$ID, unique(sunscn.df$ID))
    )

sunscn.male.stan <- stan(file = "./stan/model1_male.stan",
     model_name = "rand_int", data = standat,
     seed = 271828, init = "random",
     control = list(adapt_delta = 0.95),
     # refresh = 0 to suppress printing of iterations
     refresh = 0, thin = 10, iter = 25000, warmup = 5000, chains = 4)

###
# men and women share a common covariance matrix
###
X <- model.matrix(~ 1 + male*sunny.hot, sunscn.df)
print(names(X))
X <- unname(X)
Xran <- unname(model.matrix(~ 1 + sunny.hot, sunscn.df))
attr(X, "assign") <- NULL
attr(Xran, "assign") <- NULL
standat <- list(
    y = sunscn.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(sunscn.df$ID)),
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    maleS = maleS,
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(sunscn.df$ID, unique(sunscn.df$ID))
    )

sunscn1Cov.stan <- stan(file = "./stan/model1.stan",
model_name = "rand_int", data = standat,
     seed = 271828, init = "random",
     control = list(adapt_delta = 0.95),
     # refresh = 0 to suppress printing of iterations
     refresh = 0, thin = 10, iter = 25000, warmup = 5000, chains = 4)

library("bridgesampling")
set.seed(23)

bridge.Full <- bridge_sampler(sunscnFull.stan)
bridge.dropSunny <- bridge_sampler(sunscn.dropSunny.stan)
bridge.noIntcpt <- bridge_sampler(sunscn.noIntcpt)
bridge.male <- bridge_sampler(sunscn.male.stan)
bridge.1Cov <- bridge_sampler(sunscn1Cov.stan)
##
# Like a Type-3 model comparison
##
bf(bridge.Full, bridge.dropSunny) # dropping sunny.hot random slopes?
bf(bridge.Full, bridge.noIntcpt)  # dropping random intercepts?
bf(bridge.Full, bridge.male)      # dropping male:sunny.hot fixed effect?
bf(bridge.Full, bridge.1Cov)      # men and women share identical coviance?
#
# Approximate percentage error of the marginal likelihood.  (also
# available is the relative mean-squared error of the marginal likelihood
# estimate and an approximate coefficient of variation).
#
error_measures(bridge.Full)$percentage
error_measures(bridge.dropSunny)$percentage
```

4. Stan syntax code for flat priors (e.g., $\mathcal{N}(0, 10^5)$ for fixed effect slopes)

   Flat priors can be used by modifying only a portion of the Stan syntax (see highlighted Stan syntax code in section 2 of this Supplement). The original 'weakly informative' priors are replaced below by vague but proper priors, e.g., a Cauchy (0, 1000) instead of the weakly informative Cauchy(0, 10) for the intercept and N(0, 10^5) instead of the original N(0, 2.5).

```
  target += cauchy_lpdf(beta[1] | 0, 1000);     // intercept prior

  for (j in 2:P) {
    target += cauchy_lpdf(beta[j] | 0, 100000); // prior all slopes
    }
```

5. R and Stan syntax code for modeling all four sun protection behaviors (sunscreen, hat, shade, and protective clothing)

```
# sunscreen behavior is the default reference level
ivr_4long.df$beh <- relevel(ivr_4long.df$beh, ref = "sunscn")

X <- model.matrix(~ 1 + male * beh * sunny.hot, ivr_4long.df)
tb.names <- colnames(X)   # names of fixed effects
print(tb.names)           # can be used to label beta[]
X <- unname(X)            # names removed for stan() later

Xran <- model.matrix(~ 1 + beh*sunny.hot, ivr_4long.df)
tr.names <- colnames(Xran) # names of random effects
tr.names
Xran <- unname(Xran)
attr(X, "assign") <- NULL
attr(Xran, "assign") <- NULL

# browser()
maleS <- with(ivr_4long.df, tapply(male, ID, function(x) {return(x[1])}))
maleS <- maleS + 1

library("rstan")
options(mc.cores = parallel::detectCores())
standat <- list(
    y = ivr_4long.df$y,
    N = nrow(X),
    P = ncol(X),
    J = length(unique(ivr_4long.df$ID)), # needed b/c char(), not factor
    X = X,
    Z_u = Xran,
    n_u = ncol(Xran),
    maleS = maleS,
    # subj indicator 1, 2, 3, ..., NOT '2001', '2002', etc.
    subj = match(ivr_4long.df$ID, unique(ivr_4long.df$ID))
)

# Define a function to generate initialization values
initfun <- function(chain_id = 1)  {
  n_u <- ncol(Xran)
  P <- ncol(X)
  nsubj <- length(unique(ivr_4long.df$ID))
  # Easier to derive the cholesky factor from a correlation matrix
  LCorr <- array(NA, dim = c(2, n_u, n_u))
  Corr <- matrix(runif(1, min = 0.01, max = 0.35), n_u, n_u)
  diag(Corr) <- 1.0
  LCorr[1, ,] <- t(chol(Corr))
  # 2nd slice of lower cholesky factor
  Corr <- matrix(runif(1, min = 0.01, max = 0.35), n_u, n_u)
  diag(Corr) <- 1.0
  LCorr[2, ,] <- t(chol(Corr))

  inits <- list(
        beta = as.vector( rnorm(P, mean = 0, sd = 1) ),
     sigma_u = matrix( runif(2*n_u, min = 1, max = 20), nrow = 2, ncol = n_u),
         z_u = array( runif(2*nsubj*n_u, min=1, max=20), dim=c(2,nsubj,n_u)),
         L_u = LCorr )
  return(inits)
  }
n_chains <- 4
init_ll <- lapply(1:n_chains, function(id) initfun(chain_id = id))

all_beh_male.stan <- stan(file = "./stan/model1_male.stan",
model_name = "all_beh_male", data = standat,
     seed = 23,
     chains = n_chains,
     init = init_ll,
     control = list(adapt_delta = 0.95),
     # refresh = 0 to suppress printing of iterations
     refresh = 0, thin = 10, warmup = 2000, iter = 12000)
}

print(tb.names)
print(all_beh_male.stan, pars = "beta", probs = c(0.025, 0.50, 0.975),
   digits_summary = 2)
```

Stan syntax code in file named "`model1_male.stan`". It is needed in the R syntax above.

```
// Stan syntax file for fitting separate random effects in men and women
// Adapted from Sorensen et al. (2016), listing 10, available at:
// http://www.ling.uni-potsdam.de/~vasishth/pdfs/SorensenHohensteinVasishth2016.
pdf
data {
  int<lower=0> N;                  // n trials
  int<lower=1> P;                  // n fixefs
  int<lower=0> J;                  // n subjects
  int<lower=1> n_u;                // n subj ranefs
  int<lower=1,upper=J> subj[N];    // subject indicator
  int<lower=1,upper=2> maleS[J];   // if jth subject is male
  row_vector[P] X[N];              // fixed effects design matrix
  row_vector[n_u] Z_u[N];          // subj ranef design matrix
  int<lower=0,upper=1> y[N];       // dependent variable
}
parameters {
  vector[P] beta;                    // fixed effects coefs
  cholesky_factor_corr[n_u] L_u[2]; // Chol factor of subj ranef cor matrix
  vector<lower=0>[n_u] sigma_u[2];  // subj ranef std
  // declare a 2 by J array of 2-dimensional column vectors
  // see https://mc-stan.org/docs/2_18/
  // reference-manual/array-data-types-section.html
  vector[n_u] z_u[2,J];              // subj ranef, zu[1,] for women
}
transformed parameters {
  vector[n_u] u[J];                  // subj individual ranefs
  matrix[n_u,n_u] Sigma_u[2];        // subj ranef cov matrix
  for (k in 1:2) {                   // Cholesky for men and women
    Sigma_u[k] = diag_pre_multiply(sigma_u[k], L_u[k]);
  }
  for(j in 1:J)     {
    u[j] = Sigma_u[maleS[j]] * z_u[maleS[j], j]; // LKJ for ranef
    }
}
model {
  // priors
  target += lkj_corr_cholesky_lpdf(L_u[1] | 3.0);
  target += lkj_corr_cholesky_lpdf(L_u[2] | 3.0);

  for (j in 1:J) {
    // For each slice of z_u, generate a normally distributed variable
    target += normal_lpdf( z_u[1, j] | 0, 1 );
    target += normal_lpdf( z_u[2, j] | 0, 1 );
    }

  target += cauchy_lpdf(beta[1] | 0, 10);    // intercept prior
  // http://www.stat.columbia.edu/~gelman/research/published/priors11.pdf

  for (j in 2:P) {
    target += cauchy_lpdf(beta[j] | 0, 2.5); // prior all slopes
    }

  // target += notation to get Bayes factor with library(bridgesampling)
  for (i in 1:N)  {
   target += bernoulli_logit_lpmf(y[i] | X[i] * beta + Z_u[i] * u[subj[i]]);
  }
}
generated quantities {
// OPTIONAL: skip to make Stan run faster.
// These can be used for additional model diag, see, e.g.,
// http://mc-stan.org/loo/articles/loo2-with-rstan.html
  vector[N] log_lik;
  vector[N] y_logit;
  matrix[n_u,n_u] corr_f;       // corrleation for females
  matrix[n_u,n_u] corr_m;
  corr_f = tcrossprod(L_u[1]);  // square of Cholesky factor is corr
  corr_m = tcrossprod(L_u[2]);
for (i in 1:N)  {
  y_logit[i] = X[i] * beta + Z_u[i] * u[subj[i]];
  log_lik[i] = bernoulli_logit_lpmf(y[i] | y_logit[i]);
  }
}
```

6. Examples on how to test specific hypotheses (e.g., $H_1\colon \sigma_{1f} < \sigma_{1m}$)

The R code below calculates the empirical probability that the posterior distribution of women's standard deviation in sunscreen use on sunny and hot days is less than that of men's. Note that this is derived from the simulated Markov chain data and it assumes equal prior odds between two competing hypotheses.

```
sunscn_male.mat <- as.matrix(sunscn_male.stan)  # extract MCMC draws
###
# Pr(female_sigma_u < male_sigma_u)? First, in sunny & hot weather?
###
d.female <- sunscn_male.mat[, "sigma_u[1,2]"]   # row 1 women; col 2 sunny.hot
d.male <- sunscn_male.mat[, "sigma_u[2,2]"]
t.tab <- table(d.female < d.male)
prop.table(t.tab)                               # pr(sigma_f < sigma_m)
# what about in cool weather?
d.female <- sunscn_male.mat[, "sigma_u[1,1]"]
```

```
d.male <- sunscn_male.mat[, "sigma_u[2,1]"]
t.tab <- table(d.female < d.male)
prop.table(t.tab)
```

7. Examples on how to calculate conditional probabilities (e.g., $p(\text{hat}|\text{sunscreen})$)

   Supplement Table 1 shows the observed counts of sun protection behaviors reported by women in cool weather and sunny and hot weather. The conditional probability $p(\text{hat}|\text{sunscreen})$ equals $p(\text{hat, sunscreen})/p(\text{sunscreen})$ according to Bayes' rule. The numerator is the total number of times that both a hat and sunscreen are used and the denominator is the total number of times that sunscreen is used, summed over hat wearing, shade seeking and long sleeve clothing.

Supplement Table 1. Observed counts of sun protection behaviors reported by women.

		Sunny and hot weather = no							
		Sleeve = no				Sleeve = yes			
		Shade = no		Shade = yes		Shade = no		Shade = yes	
		Hat		Hat		Hat		Hat	
		no	yes	no	Yes	no	yes	no	yes
sunscn	no	49	5	16	3	54	5	16	8
	yes	16	2	7	1	24	13	9	6
		Sunny and hot weather = yes							
		Sleeve = no				Sleeve = yes			
		Shade = no		Shade = yes		Shade = no		Shade = yes	
		Hat		Hat		Hat		Hat	
		no	yes	no	yes	no	yes	no	yes
sunscn	no	51	10	56	14	22	3	56	23
	yes	61	26	85	56	49	23	28	36

For women on not sunny and hot days, the numerator $p(\text{hat, sunscreen})$ represents the joint probability of both hat and sunscreen. They are marked above as the cooccurrences using shaded table cells. The denominator $p(\text{sunscreen})$ represents the marginal probability of sunscreen use, the sum of all the counts in the row where sunscreen equals 'yes'. Thus, $p(\text{hat}|\text{sunscreen})$ equals (2+1+13+6)/(2+1+13+6+16+7+24+9) = 0.28.

The same steps can be used to calculate the conditional probability on sunny and hot days for women, which is (26+56+23+36)/(26+56+23+36+61+85+49+28) = 0.39. Table 4 in the main article is prepared by repeating the same process several times for men and for women. This process may appear tedious at first glance, but it follows a regular pattern so that it can be easily implemented in R syntax code to automate the calculation (available from the corresponding author upon request).

---- end of online supplementary materials ----