WebPPL is a feature-rich probabilistic programming language embedded in Javascript.

Check out some **demos** or try it yourself in the editor below.

```
print("================================================================================")
print("PCM20201110_GammaPrior&RiskCalculation                    *** 2020/11/11 *** ")
print("  see also Simple Reaction Time, Example 9, Card, Moran & Newell, 1983, p.66  ")
print("  see also https://www.humanbenchmark.com/tests/reactiontime/statistics       ")
print("  here we use the gamma distribution as a prior distribution                  ")
print("  see also https://en.wikipedia.org/wiki/Gamma_distribution                   ")
print("  CMN-interval 'typical[fast ~ slow]' is interpreted ...                      ")
print("  ... as gamma(shape, scale), where 'shape' and 'scale' have to be determined ")
print("================================================================================")
/**
 * @author - Claus Moebus   <claus.moebus@uol.de>
 */
//------------------------------------------------------------------------------------------
/**
 * @variable {number} startTime - used in method 'runtime' to compute runtime in sec and min
 */
var startTime = Date.now()
//------------------------------------------------------------------------------------------
print("Input parameter:")
/**
 * @variable {integer} nTrials - no of efficient samples (incl. burnout) in MCMC-sampling
 */
var nTrials = 5E4
print("nTrials = " + nTrials)
//------------------------------------------------------------------------------------------
/**
 * @variable {integer} nSigma - no of standard deviations between mean and 'slow', 'fast'
 *                              interval boundaries
 */
var nSigma = 3
print("nSigma = " + nSigma)
//------------------------------------------------------------------------------------------
/**
 * @variable {integer} myBurnPeriod - length of burnin period in MCMC process
 */
var myBurnPeriod = nTrials * 0.10
print("length of burn-in period = " + myBurnPeriod)
//------------------------------------------------------------------------------------------
/**
 * @variable {integer} myLag - only every myLag-th sample will be retained during MCMC
 */
var myLag = 10
print("length of lag = " + myLag)
//------------------------------------------------------------------------------------------
/**
 * @variable {array} data - author's reaction times in an experiment found here
```

```
 *                          https://www.humanbenchmark.com/tests/reactiontime/
 *                          visited March 2018
 */
var data =
    [458, 292, 228, 403, 271, 420, 350, 235, 260, 306]
print("response time data = [" + data + "]")
print("mean of data = " + listMean(data))
print("stdev of data = " + listStdev(data))
print("-------------------------------------------------------------------------")
/**
 * @variable {array} tauPCrit   - critical values-at-risk in msec for tauP
 * @variable {array} tauCCrit   - critical values-at-risk in msec for tauC
 * @variable {array} tauMCrit   - critical values-at-risk in msec for tauM
 * @variable {array} tauSumCrit - critical values-at-risk in msec for tauSum
 */
var tauPCrit   = [200, 175, 150, 125, 116]
print("critical values-at-risk in msec for tauP   = [" + tauPCrit + "]")
var tauCCrit   = [175, 150, 125, 100, 88]
print("critical values-at-risk in msec for tauC   = [" + tauCCrit + "]")
var tauMCrit   = [100, 90, 80, 70, 67]
print("critical values-at-risk in msec for tauM   = [" + tauMCrit + "]")
var tauSumCrit = [400, 375, 350, 325, 300, 275, 272]
print("critical values-at-risk in msec for tauSum = [" + tauSumCrit + "]")
print("-------------------------------------------------------------------------")
/**
 * @object hyperParmTauX - parameters shape=a and scale=b for prior Gamma derived from MHP
 * @property {number} a - value is the shape parameter of Gamma(a, b)
 * @property {number} b - value is the scale parameter of Gamma(a, b)
 */
var hyperParmTauP = {a:16.0, b:6.3}
print("hyperParmTauP = {a:" + hyperParmTauP.a + ",  b:" + hyperParmTauP.b + "}")
var hyperParmTauC = {a:8.4, b:8.3}
print("hyperParmTauC = {a:" + hyperParmTauC.a + ", b:" + hyperParmTauC.b + "}")
var hyperParmTauM = {a:36.0, b:1.9}
print("hyperParmTauM = {a:" + hyperParmTauM.a + ",  b:" + hyperParmTauM.b + "}")
print("-------------------------------------------------------------------------")
/**
 * @object hyperParmSigmaTauSum - shape=a and scale=b for variance of Gaussian Likelihood
 * @property {number} a - value is the shape parameter of Gamma(a, b)
 * @property {number} b - value is the scale parameter of Gamma(a, b)
 */
var hyperParmSigmaTauSum = {a:4.0, b:20.0}
print("hyperParmSigmaTauSum = {a:" + hyperParmSigmaTauSum.a + ", b:" + hyperParmSigmaTauSum.b
print("-------------------------------------------------------------------------")
//-------------------------------------------------------------------------------------
// function definitions
//-------------------------------------------------------------------------------------
/**
 * @function runtime - method to compute the runtime in seconds and minutes
 */
var runTime = function() {
  var stopTime = Date.now()
  var runSecs = (stopTime - startTime)/1000
  var runMins = runSecs/60
  print("runtime in seconds = " + runSecs)
  print("runtime in minutes = " + runMins)}
//-------------------------------------------------------------------------------------
/**
 * @description - descriptive statistics of a sample-generated gamma distribution
 * @function myTauXDistribution
 * @param {string} id - The identifier of the tauX distribution.
```

```
 * @param {distributionObject} tauXDistribution - tauX distribution (X = P, C, M, T)
 * @param {number} modeTauX - mode of tauX as a function of a and b
 *                            mode = (a-1)*b for a >= 1
 * @returns {object} meanSigmaTauObject - object with mean and sigma of TauX
 * @property {number} meanTauX - mean of tauX (X = P, C, M, T) or tau
 * @property {number} sigmaTauX - standard deviation of tauX (X = P, C, M, T) or tau
 */
var myTauXDescription = function(id, tauXDistribution, modeTauX) {
  var myTauXDistribution = { // extraction of probs and support from WebPPL tauX distributior
    probs: map(function(eventTuple){ // object to compute mean and sigma of tauX
      Math.exp(tauXDistribution.score(eventTuple))}, tauXDistribution.support()),
    support: tauXDistribution.support()}
  print(id)
  // mode(tauX), mean(tauX), variance(tauX) and sigma(tauX)
  print("mode = " + modeTauX)
  var meanTauX = sum(map2(function(value, prob) {
    value*prob},myTauXDistribution.support, myTauXDistribution.probs))
  print("mean = " + meanTauX)
  var sigmaTauX = Math.sqrt(sum(map2(function(value, prob) {
    Math.pow((value-meanTauX), 2)*prob},
                                    myTauXDistribution.support,
                                    myTauXDistribution.probs)))
  print("sigma = " + sigmaTauX)
  var tauX_Intval = {fast:meanTauX - nSigma * sigmaTauX, mean:meanTauX,
                     slow:meanTauX + nSigma * sigmaTauX}
  return tauX_Intval}
//-----------------------------------------------------------------------------------------
/**
 * @description - cdf computes the cumulative density function P(X <= c)
 * @function cdf
 * @param {distributionObject} distrObject - must be generated by function 'Infer'
 * @param {real} c - function argument of cdf F(c) = P(X <= c)
 * @returns {real} - F(c) = P(X <= c)
 */
var cdf = function(distrObject, c) {
  var support = distrObject.support()
  var probs = map(function(xValue){
    Math.exp(distrObject.score(xValue))
  }, support)
  sum(map2(function(prob, xValue) {
    xValue <= c ? prob : 0
  }, probs, support))
}
//-------------------------------------------------------------
/**
 * @description - probsAtRisk computes the cumulative density function 1-F(c) = P(X > c)
 * @function probsAtRisk
 * @param {distributionObject} distrObject - must be generated by function 'Infer'
 * @param {real} valsAtRisk - function arguments of cdf 1-F(c) = P(X > c)
 * @returns {array} - F(c_i) = P(X <= c_i)  ; i = 1, ...
 */
var probsAtRisk = function (distrObject, valsAtRisk) {
  map(function(valAtRisk) {
    1.0 - cdf(distrObject, valAtRisk)
  }, valsAtRisk)
}
//-------------------------------------------------------------
/**
 * @description - prints a table of two column vectors:
 *               - values-at-risk and risk probabilities
 * @function printRiskProbs
```

```
 */
var printRiskProbs = function(valsAtRisk, valsAtRiskText, probs) {
  map2(function(valAtRisk, prob) {
    print(valsAtRiskText + " = " +  valAtRisk + ";  risk probability = " + prob)
  }, valsAtRisk, probs)
}
//-------------------------------------------------------------
/**
 * @description - prints a table of two column vectors:
 *               - values-at-risk and increase of risk probabilities
 * @function printDiffProbs
 */
var printDiffProbs = function(valsAtRisk, valsAtRiskText, probsPrior, probsPosterior) {
  var probDiffs = map2(function(priorPr, postPr) {
    postPr - priorPr  // change
  }, probsPrior, probsPosterior)
  map2(function(valAtRisk, probDiff) {
    print(valsAtRiskText + " = " +  valAtRisk + "; increase in risk probs = " + probDiff)
  }, valsAtRisk, probDiffs)
}
//=============================================================================================
/**
 * @function oneSampleOfPriors - takes o n e  sample from all priors tauP, tauC, tauM,
 *                             - tauSum = tauP + tauC + tauM, and sigmaTauSum
 * @returns {object} sampleOfPriors - o n e  priors-tuple
 * @returns {object} priorSigmaTauSum - o n e  sample from the Gamma distr
 *                                    - this is prior sigma for the Gaussian likelihood
 */
var oneSampleOfPriors = function () {
  var priorTauP = sample(Gamma({shape:hyperParmTauP.a, scale:hyperParmTauP.b}))
  var priorTauC = sample(Gamma({shape:hyperParmTauC.a, scale:hyperParmTauC.b}))
  var priorTauM = sample(Gamma({shape:hyperParmTauM.a, scale:hyperParmTauM.b}))
  var priorTauSum = priorTauP + priorTauC + priorTauM
  var priorSigmaTauSum =
      sample(Gamma({shape:hyperParmSigmaTauSum.a, scale:hyperParmSigmaTauSum.b}))
  return {priorTauP:priorTauP, priorTauC:priorTauC, priorTauM:priorTauM,
          priorTauSum:priorTauSum, priorSigmaTauSum:priorSigmaTauSum}
}
//---------------------------------------------------------------------------------------------
/**
 * @description - Infer generates an multivariate prior Gamma distribution for TauX
 * @variable {distribution} priorTauX - value is a WebPPL distribution object
 */
var priorTauX = Infer({model:oneSampleOfPriors, method: 'forward', samples: nTrials})
print('Univariate Priors TauX (X=P, C, M, Sum, SigmaTauSum) ~ Gamma(???, ???)')
viz.marginals(priorTauX)
print("-----------------------------------------------------------------------------")
print("model-generated "+ nSigma + "*sigma tau-interval: ")
var priorTauPIntval =
    myTauXDescription("priorTauP", marginalize(priorTauX,'priorTauP'), "unknown")
print("{fast:" + priorTauPIntval.fast + "  mean:" + priorTauPIntval.mean + "  slow:" + prior
var tauPProbsPrior = probsAtRisk(marginalize(priorTauX,'priorTauP'), tauPCrit)
printRiskProbs(tauPCrit, 'tauPCrit', tauPProbsPrior)
print("-----------------------------------------------------------------------------")
var priorTauCIntval =
    myTauXDescription("priorTauC", marginalize(priorTauX,'priorTauC'), "unknown")
print("{fast:" + priorTauCIntval.fast + "  mean:" + priorTauCIntval.mean + "  slow:" + prior
var tauCProbsPrior = probsAtRisk(marginalize(priorTauX,'priorTauC'), tauCCrit)
printRiskProbs(tauCCrit, 'tauCCrit', tauCProbsPrior)
print("-----------------------------------------------------------------------------")
var priorTauMIntval =
```

```
    myTauXDescription("priorTauM", marginalize(priorTauX,'priorTauM'), "unknown")
print("{fast:" + priorTauMIntval.fast + "  mean:" + priorTauMIntval.mean + "  slow:" + prior
var tauMProbsPrior = probsAtRisk(marginalize(priorTauX,'priorTauM'), tauMCrit)
printRiskProbs(tauMCrit, 'tauMCrit', tauMProbsPrior)
print("-------------------------------------------------------------------------")
var priorTauSumIntval =
    myTauXDescription("priorTauSum", marginalize(priorTauX,'priorTauSum'), "unknown")
print("{fast:" + priorTauSumIntval.fast + "  mean:" + priorTauSumIntval.mean + "  slow:" + pr
var tauSumProbsPrior = probsAtRisk(marginalize(priorTauX,'priorTauSum'), tauSumCrit)
printRiskProbs(tauSumCrit, 'tauSumCrit', tauSumProbsPrior)
print("-------------------------------------------------------------------------")
var priorSigmaTauSum_Intval =
    myTauXDescription("priorSigmaTauSum", marginalize(priorTauX,'priorSigmaTauSum'), "unknown
print("model-generated "+ nSigma + "*sigma tau-interval: ")
print("{fast:" + priorSigmaTauSum_Intval.fast + "  mean:" + priorSigmaTauSum_Intval.mean + "
print("=========================================================================")
/**
 * @function oneSampleOfModel - takes  o n e  sample from the priors
 * @returns {object} posteriorTauSum - returns  o n e  sample of posterior TauSum-tuple
 */
var oneSampleOfModel = function() {
  /**
   * @variable {number} PriorTauSum - a sample from Gamma TauSum-distribution
   */
  var priorTauP = sample(Gamma({shape:hyperParmTauP.a, scale:hyperParmTauP.b}))
  var priorTauC = sample(Gamma({shape:hyperParmTauC.a, scale:hyperParmTauC.b}))
  var priorTauM = sample(Gamma({shape:hyperParmTauM.a, scale:hyperParmTauM.b}))
  var priorTauSum = priorTauP + priorTauC + priorTauM
  /**
   * @variable {number} priorSigmaTauSum - a sample from SigmaTauSum Gamma distribution
   */
  var priorSigmaTauSum =
      sample(Gamma({shape:hyperParmSigmaTauSum.a, scale:hyperParmSigmaTauSum.b}))
  //
  map(function(datum) {
      observe(Gaussian({mu:priorTauSum, sigma:priorSigmaTauSum}),datum)
  }, data)
  return {postTauP: priorTauP, postTauC: priorTauC,  postTauM: priorTauM,
          postTauSum:priorTauSum, postSigmaTauSum:priorSigmaTauSum}
}
//-------------------------------------------------------------------------------
/**
 * @description - Infer generates the posterior distributions 'posteriorTauX'
 * @variable {distributionObject} posteriorTauX - univariate posterior distribution
 */
print('Univariate Posteriors TauX (X=P, C, M, Sum) Gamma(???, ???) and SigmaTauSum Gamma(???,
var posterior = Infer({model:oneSampleOfModel, method:'MCMC', samples: nTrials,
                           burn:myBurnPeriod, lag:myLag})
viz.marginals(posterior)
print("-------------------------------------------------------------------------")
print("model-generated "+ nSigma + "*sigma tau-interval: ")
var postTauPIntval =
    myTauXDescription("postTauP", marginalize(posterior,'postTauP'), "unknown")
print("{fast:" + postTauPIntval.fast + "  mean:" + postTauPIntval.mean + "  slow:" + postTaul
var tauPProbsPosterior = probsAtRisk(marginalize(posterior,'postTauP'), tauPCrit)
printRiskProbs(tauPCrit, 'tauPCrit', tauPProbsPosterior)
print("-----------------------------------------------------------")
printDiffProbs(tauPCrit, 'tauPCrit', tauPProbsPrior, tauPProbsPosterior)
print("-------------------------------------------------------------------------")
var postTauCIntval =
    myTauXDescription("postTauC", marginalize(posterior,'postTauC'), "unknown")
```

```
print("{fast:" + postTauCIntval.fast + "  mean:" + postTauCIntval.mean + "  slow:" + postTauC
var tauCProbsPosterior = probsAtRisk(marginalize(posterior,'postTauC'), tauCCrit)
printRiskProbs(tauCCrit, 'tauCCrit', tauCProbsPosterior)
print("-------------------------------------------------------------")
printDiffProbs(tauCCrit, 'tauCCrit', tauCProbsPrior, tauCProbsPosterior)
print("-----------------------------------------------------------------------------")
var postTauMIntval =
    myTauXDescription("postTauM", marginalize(posterior,'postTauM'), "unknown")
print("{fast:" + postTauMIntval.fast + "  mean:" + postTauMIntval.mean + "  slow:" + postTauM
var tauMProbsPosterior = probsAtRisk(marginalize(posterior,'postTauM'), tauMCrit)
printRiskProbs(tauMCrit, 'tauMCrit', tauMProbsPosterior)
print("-------------------------------------------------------------")
printDiffProbs(tauMCrit, 'tauMCrit', tauMProbsPrior, tauMProbsPosterior)
print("-----------------------------------------------------------------------------")
var postTauSumIntval =
    myTauXDescription("postTauSum", marginalize(posterior,'postTauSum'), "unknown")
print("{fast:" + postTauSumIntval.fast + "  mean:" + postTauSumIntval.mean + "  slow:" + post
var tauSumProbsPosterior = probsAtRisk(marginalize(posterior,'postTauSum'), tauSumCrit)
printRiskProbs(tauSumCrit, 'tauSumCrit', tauSumProbsPosterior)
print("-------------------------------------------------------------")
printDiffProbs(tauSumCrit, 'tauSumCrit', tauSumProbsPrior, tauSumProbsPosterior)
print("-----------------------------------------------------------------------------")
var postSigmaTauSumIntval =
    myTauXDescription("postSigmaTauSum", marginalize(posterior,'postSigmaTauSum'), "unknown"
print("{fast:" + postSigmaTauSumIntval.fast + "  mean:" + postSigmaTauSumIntval.mean + "  sl
print("=============================================================================")
runTime()
print("=============================================================================")
```

run ▼

X

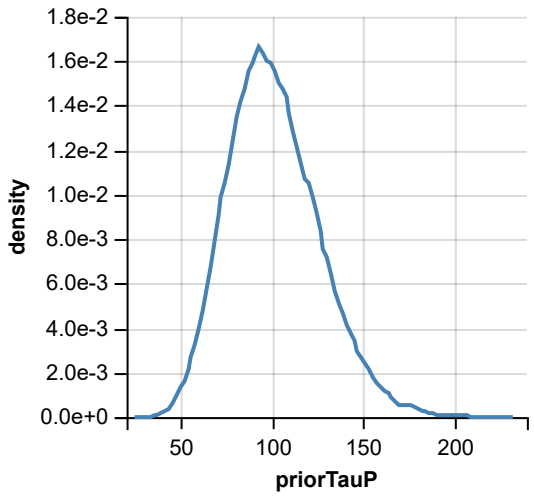
```
PCM20201110_GammaPrior&RiskCalculation                    *** 2020/11/11 ***
  see also Simple Reaction Time, Example 9, Card, Moran & Newell, 1983, p.66
  see also https://www.humanbenchmark.com/tests/reactiontime/statistics
  here we use the gamma distribution as a prior distribution
  see also https://en.wikipedia.org/wiki/Gamma_distribution
  CMN-interval 'typical[fast ~ slow]' is interpreted ...
  ... as gamma(shape, scale), where 'shape' and 'scale' have to be determined
=============================================================================
Input parameter:
nTrials = 50000
nSigma = 3
length of burn-in period = 5000
length of lag = 10
response time data = [458,292,228,403,271,420,350,235,260,306]
mean of data = 322.3
stdev of data = 77.10389095240265
-----------------------------------------------------------------------------
critical values-at-risk in msec for tauP   = [200,175,150,125,116]
critical values-at-risk in msec for tauC   = [175,150,125,100,88]
critical values-at-risk in msec for tauM   = [100,90,80,70,67]
```

```
critical values-at-risk in msec for tauSum = [400,375,350,325,300,275,272]
```

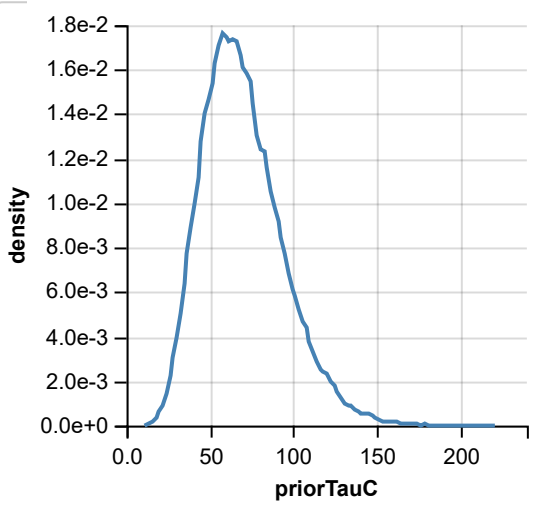
```
hyperParmTauP = {a:16,  b:6.3}
hyperParmTauC = {a:8.4, b:8.3}
hyperParmTauM = {a:36,  b:1.9}
```

```
hyperParmSigmaTauSum = {a:4, b:20}
```

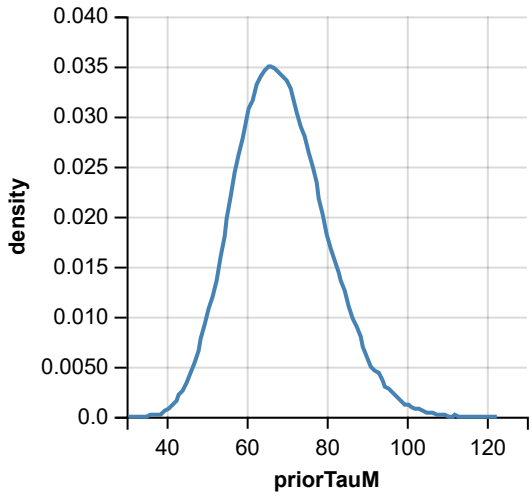
```
Univariate Priors TauX (X=P, C, M, Sum, SigmaTauSum) ~ Gamma(???, ???)
priorTauP:
```



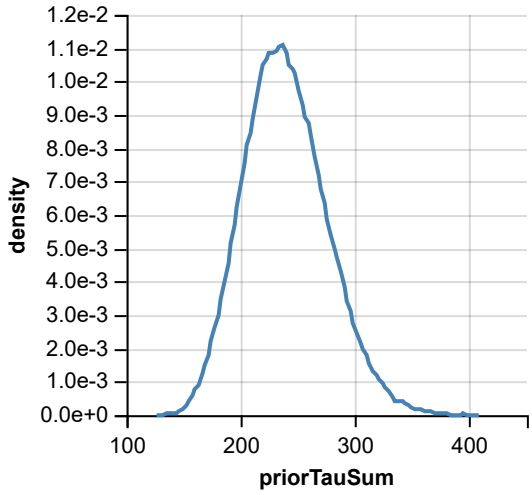

```
priorTauC:
```



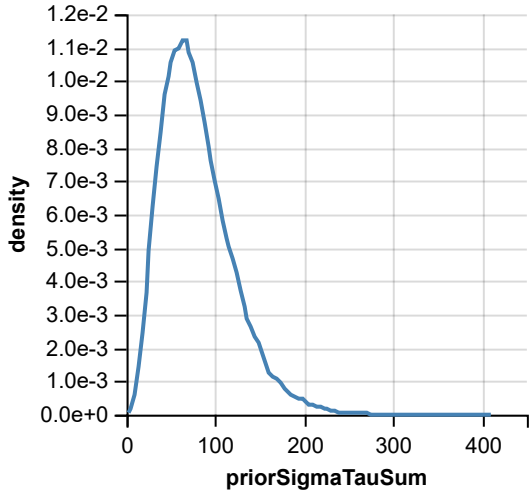

```
priorTauM:
```



priorTauSum:



priorSigmaTauSum:



---

model-generated 3*sigma tau-interval:

priorTauP

mode = unknown

mean = 101.02059184913485

sigma = 25.072119282875473

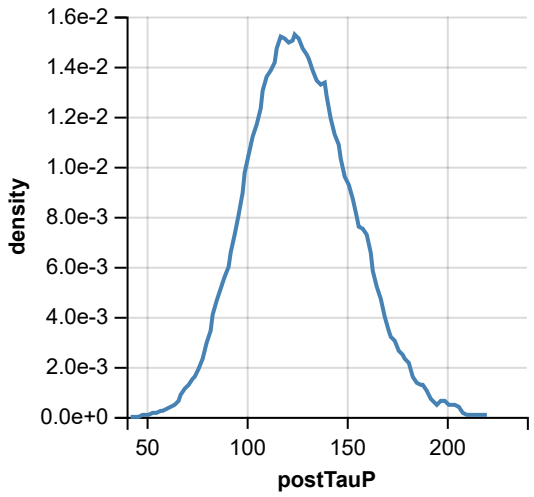
{fast:25.80423400050843 mean:101.02059184913485 slow:176.23694969776125}

tauPCrit = 200; risk probability = 0.0006999999992814976

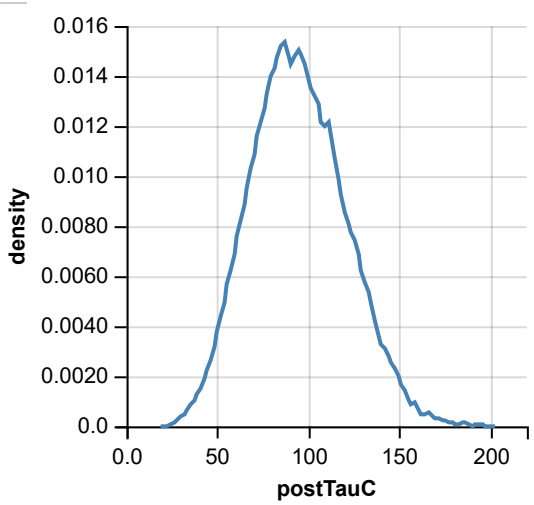
tauPCrit = 175; risk probability = 0.005939999999286738

```
tauPCrit = 150;  risk probability = 0.03675999999931756
tauPCrit = 125;  risk probability = 0.1674799999994483
tauPCrit = 116;  risk probability = 0.2587599999995396
------------------------------------------------------------------------
priorTauC
mode = unknown
mean = 69.63921434942415
sigma = 24.168909036128543
{fast:-2.8675127589614817  mean:69.63921434942415  slow:142.14594145780978}
tauCCrit = 175;  risk probability = 0.0006799999992814776
tauCCrit = 150;  risk probability = 0.004219999999285018
tauCCrit = 125;  risk probability = 0.0239999999993048
tauCCrit = 100;  risk probability = 0.11141999999939223
tauCCrit = 88;  risk probability = 0.20659999999948742
------------------------------------------------------------------------
priorTauM
mode = unknown
mean = 68.44847640951699
sigma = 11.414754914820241
{fast:34.20421166505626  mean:68.44847640951699  slow:102.6927411539777}
tauMCrit = 100;  risk probability = 0.006039999999286838
tauMCrit = 90;  risk probability = 0.03783999999931864
tauMCrit = 80;  risk probability = 0.1567999999994376
tauMCrit = 70;  risk probability = 0.4250799999997059
tauMCrit = 67;  risk probability = 0.5272799999998081
------------------------------------------------------------------------
priorTauSum
mode = unknown
mean = 239.10828260807529
sigma = 36.49722183786279
{fast:129.6166170944869  mean:239.10828260807529  slow:348.59994812166366}
tauSumCrit = 400;  risk probability = 0.00009999999928089753
tauSumCrit = 375;  risk probability = 0.0007599999992815576
tauSumCrit = 350;  risk probability = 0.003959999999284758
tauSumCrit = 325;  risk probability = 0.01587999999929668
tauSumCrit = 300;  risk probability = 0.05583999999933664
tauSumCrit = 275;  risk probability = 0.16151999999944233
tauSumCrit = 272;  risk probability = 0.18051999999946133
------------------------------------------------------------------------
priorSigmaTauSum
mode = unknown
mean = 80.02768056079135
sigma = 40.261658137514154
model-generated 3*sigma tau-interval:
{fast:-40.7572938517511  mean:80.02768056079135  slow:200.8126549733338}
```

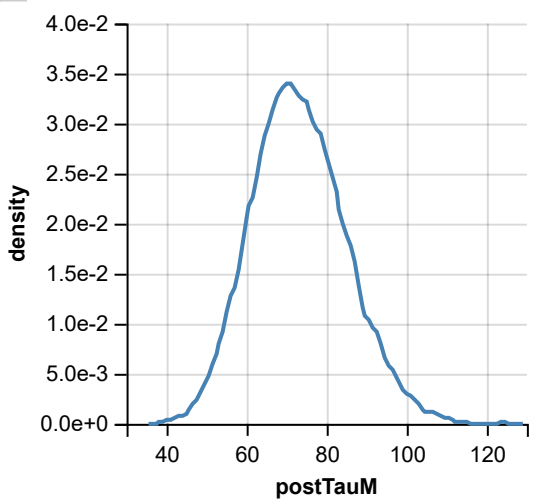
```
================================================================================

Univariate Posteriors TauX (X=P, C, M, Sum) Gamma(???, ???) and SigmaTauSum Gamma
(???, ???)

postTauP:
```



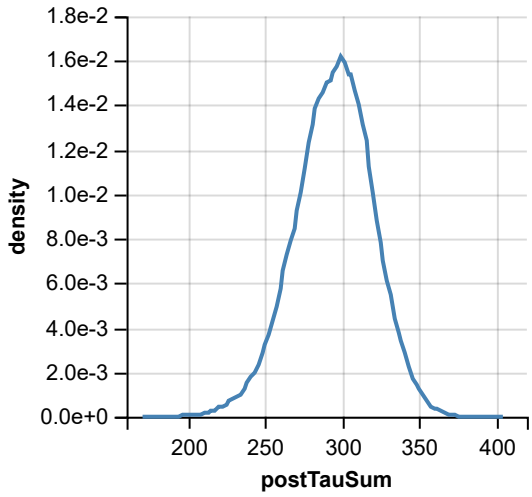

```
postTauC:
```



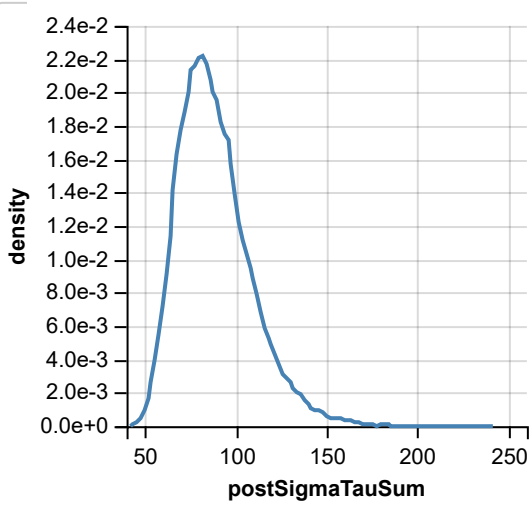

```
postTauM:
```




```
postTauSum:
```



postSigmaTauSum:



---

model-generated 3*sigma tau-interval:

postTauP

mode = unknown

mean = 126.91186649497132

sigma = 25.983574568079938

{fast:48.961142790731515 mean:126.91186649497132 slow:204.86259019921113}

tauPCrit = 200; risk probability = 0.003939999999624799

tauPCrit = 175; risk probability = 0.03863999999963219

tauPCrit = 150; risk probability = 0.19055999999968143

tauPCrit = 125; risk probability = 0.5104199999998723

tauPCrit = 116; risk probability = 0.6474999999999493

---

tauPCrit = 200; increase in risk probs = 0.0032400000003433016

tauPCrit = 175; increase in risk probs = 0.03270000000034545

tauPCrit = 150; increase in risk probs = 0.15380000000036387

tauPCrit = 125; increase in risk probs = 0.342940000000424

tauPCrit = 116; increase in risk probs = 0.38874000000040976

---

postTauC

mode = unknown

mean = 94.41345259535262

```
sigma = 26.053141783864998
{fast:16.254027243757633  mean:94.41345259535262  slow:172.5728779469476}
tauCCrit = 175;  risk probability = 0.0024199999996052934
tauCCrit = 150;  risk probability = 0.021019999999611794
tauCCrit = 125;  risk probability = 0.12741999999964582
tauCCrit = 100;  risk probability = 0.39965999999978474
tauCCrit = 88;  risk probability = 0.5749399999999003
------------------------------------------------------------
tauCCrit = 175; increase in risk probs = 0.0017400000003238159
tauCCrit = 150; increase in risk probs = 0.016800000000326776
tauCCrit = 125; increase in risk probs = 0.10342000000034102
tauCCrit = 100; increase in risk probs = 0.2882400000003925
tauCCrit = 88; increase in risk probs = 0.3683400000004129
------------------------------------------------------------------------
postTauM
mode = unknown
mean = 73.14663748213418
sigma = 11.832161904551604
{fast:37.65015176847937  mean:73.14663748213418  slow:108.643123195789}
tauMCrit = 100;  risk probability = 0.017559999999401277
tauMCrit = 90;  risk probability = 0.08437999999944701
tauMCrit = 80;  risk probability = 0.27077999999960467
tauMCrit = 70;  risk probability = 0.5846799999998891
tauMCrit = 67;  risk probability = 0.6842799999999689
------------------------------------------------------------
tauMCrit = 100; increase in risk probs = 0.011520000000114439
tauMCrit = 90; increase in risk probs = 0.04654000000012837
tauMCrit = 80; increase in risk probs = 0.11398000000016706
tauMCrit = 70; increase in risk probs = 0.15960000000018315
tauMCrit = 67; increase in risk probs = 0.1570000000001608
------------------------------------------------------------------------
postTauSum
mode = unknown
mean = 294.4719565724604
sigma = 25.394739115035033
{fast:218.2877392273553  mean:294.4719565724604  slow:370.65617391756547}
tauSumCrit = 400;  risk probability = 0.00003999999928472331
tauSumCrit = 375;  risk probability = 0.00029999999928498333
tauSumCrit = 350;  risk probability = 0.010339999999295024
tauSumCrit = 325;  risk probability = 0.107859999999393
tauSumCrit = 300;  risk probability = 0.42897999999971215
tauSumCrit = 275;  risk probability = 0.7869400000000164
tauSumCrit = 272;  risk probability = 0.8169800000000048
------------------------------------------------------------
tauSumCrit = 400; increase in risk probs = -0.00005999999999996174225
```

```
tauSumCrit = 375; increase in risk probs = -0.00045999999999657426
tauSumCrit = 350; increase in risk probs = 0.006380000000010266
tauSumCrit = 325; increase in risk probs = 0.09198000000009632
tauSumCrit = 300; increase in risk probs = 0.3731400000003755
tauSumCrit = 275; increase in risk probs = 0.6254200000005741
tauSumCrit = 272; increase in risk probs = 0.6364600000005435
------------------------------------------------------------------------
postSigmaTauSum
mode = unknown
mean = 88.59391173159861
sigma = 20.201878560370968
{fast:27.98827605048571  mean:88.59391173159861  slow:149.19954741271152}
========================================================================
runtime in seconds = 4693.912
runtime in minutes = 78.23186666666668
========================================================================
```

Features

- Runs on the command line with node.js (http://nodejs.org/) or in the browser (http://docs.webppl.org/en/master/development/workflow.html#browser-version).
- Supports modular and re-usable code using packages (http://docs.webppl.org/en/master/packages.html) built on top of the npm package system, and interoperates with existing Javascript packages in the npm ecosystem.
- Includes a large and expanding library of primitive distributions. (http://docs.webppl.org/en/master/distributions.html)
- Implements a variety of inference algorithms (http://docs.webppl.org/en/master/inference/index.html), including exact inference via enumeration, rejection sampling, Sequential Monte Carlo, Markov Chain Monte Carlo, Hamiltonian Monte Carlo, and inference-as-optimization (e.g. variational inference).
- Provides inference as a first-class operator in the language, allowing for nested inference (‘inference about inference’).
- Supports optimizable models with neural network components using adnn (https://www.npmjs.com/package/adnn).

Demos

Browser-based applications powered by WebPPL.

- Procedural vines with shape constraints (demos/vines/index.html)
- 3D procedural spaceships with shape constraints (http://dritchie.github.io/web-procmod/) (Note: the code in this demo is written in an older version of WebPPL)

Local install

Install WebPPL in two easy steps:

1. Install node.js (http://nodejs.org)
2. Run `npm install -g webppl`

Now, the `webppl` command is globally available.

To upgrade to the latest version, run `npm update -g webppl`.

Documentation

To learn more about how to set up and use WebPPL, take a look at our documentation (http://docs.webppl.org) and the examples (https://github.com/probmods/webppl/tree/master/examples).

To learn more about how WebPPL works under the hood, check out our web book, The Design and Implementation of Probabilistic Programming Languages (http://dippl.org/).

For probabilistic modeling in general, our other web book, Probabilistic Models of Cognition (https://probmods.org), might be of interest.

License

The WebPPL code base is open source and freely available for commerical and non-commercial use under the MIT license (https://github.com/probmods/webppl/blob/master/LICENSE.md).

Contributions

We encourage you to contribute to WebPPL! Check out our guidelines for contributors (https://github.com/probmods/webppl/blob/master/CONTRIBUTING.md) and join the webppl-dev (https://groups.google.com/forum/#!forum/webppl-dev) mailing list.

Pronunciation

Say “web people”.

Citing

If you use WebPPL in academic projects and papers, please cite as:

> *N. D. Goodman and A. Stuhlmüller (electronic). The Design and Implementation of Probabilistic Programming Languages. Retrieved from `http://dippl.org`. [bibtex]*

Publications

If you publish a paper using/extending WebPPL, let us know (https://groups.google.com/forum/#!forum/webppl-dev) and we'll add it to this list:

> *D. Ritchie, P. Horsfall, and N. D. Goodman. Deep Amortized Inference for Probabilistic Programs (https://arxiv.org/abs/1610.05735). arXiv:1610.05735.*
>
> *L. Ouyang, M. H. Tessler, D. Ly, and N. D. Goodman. Practical optimal experiment design with probabilistic programs (https://arxiv.org/abs/1608.05046). arXiv:1608.05046.*
>
> *M. H. Tessler and N. D. Goodman. A Pragmatic Theory of Generic Language (https://arxiv.org/abs/1608.02926). arXiv:1608.02926.*

D. Ritchie, A. Thomas, P. Hanrahan, and N. D. Goodman. Neurally-Guided Procedural Models: Amortized Inference for Procedural Graphics Programs using Neural Networks (https://arxiv.org/abs/1603.06143). NIPS 2016.

D. Ritchie, A. Stuhlmüller, and N. D. Goodman. C3: Lightweight Incrementalized MCMC for Probabilistic Programs using Continuations and Callsite Caching (https://arxiv.org/abs/1509.02151). AISTATS 2016.

M. H. Tessler and N. D. Goodman. Communicating generalizations about events (http://stanford.edu/~mtessler/papers/Tessler2016-cogsci.pdf). Proceedings of the Thirty-Eighth Annual Conference of the Cognitive Science Society, 2016.

E. J. Yoon, M. H. Tessler, N. D. Goodman, and M. C. Frank. Talking with tact: Polite language as a balance between kindness and informativity (http://stanford.edu/~mtessler/papers/YoonTessler2016-cogsci.pdf). Proceedings of the Thirty-Eighth Annual Conference of the Cognitive Science Society, 2016.

C. Graf, J. Degen, R. X. D. Hawkins, and N. D. Goodman. Animal, dog, or dalmatian? Level of abstraction in nominal referring expressions (https://cocolab.stanford.edu/papers/GrafEtAl2016-Cogsci.pdf). Proceedings of the Thirty-Eighth Annual Conference of the Cognitive Science Society, 2016.

O. Evans, A. Stuhlmüller, and N. D. Goodman. Learning the Preferences of Ignorant, Inconsistent Agents (https://stuhlmueller.org/papers/preferences-aaai2016.pdf). AAAI 2016.

A. Stuhlmüller, R. X. D. Hawkins, N. Siddharth, and N. D. Goodman. Coarse-to-Fine Sequential Monte Carlo for Probabilistic Programs (https://arxiv.org/abs/1509.02962). arXiv:1509.02962.

O. Evans, A. Stuhlmüller, and N. D. Goodman. Learning the Preferences of Bounded Agents (https://stuhlmueller.org/papers/preferences-nipsworkshop2015.pdf). Workshop on Bounded Optimality, NIPS 2015.

R. X. D. Hawkins, A. Stuhlmüller, J. Degen, and N. D. Goodman. Why do you ask? Good questions provoke informative answers (https://stuhlmueller.org/papers/qa-cogsci2015.pdf). Proceedings of the Thirty-Seventh Annual Conference of the Cognitive Science Society, 2015.

G. Scontras and M. H. Tessler (electronic). Composition in Probabilistic Language Understanding (http://gscontras.github.io/ESSLLI-2016/). Retrieved from `http://gscontras.github.io/ESSLLI-2016` *.*

O. Evans, A. Stuhlmüller, J. Salvatier, and D. Filan (electronic). Modeling Agents with Probabilistic Programs (http://agentmodels.org). Retrieved from `http://agentmodels.org` *.*

N. D. Goodman and J. B. Tenenbaum (electronic). Probabilistic Models of Cognition (http://probmods.org). Retrieved from `http://probmods.org` *.*

N. D. Goodman and A. Stuhlmüller (electronic). The Design and Implementation of Probabilistic Programming Languages (http://dippl.org). Retrieved from `http://dippl.org` *.*

Acknowledgments

The WebPPL project is supported by grants from DARPA, under agreement number FA8750-14-2-0009, and the Office of Naval Research, grant number N00014-13-1-0788.

WebPPL is a Stanford CoCoLab (http://cocolab.stanford.edu/) project