# 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("PCM20201212_TriangleModePrior&RiskCalculation              *** 2020/12/12 *** ")
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 triangular distribution as a prior distribution             ")
print("  see also https://en.wikipedia.org/wiki/Triangular_distribution              ")
print("  CMN-interval 'typical[fast ~ slow]' is interpreted ...                      ")
print("            as  triangle(fast=a, slow=b, 'typical'=mode=c)                    ")
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 = 6E4
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("-------------------------------------------------------------------------")
/**
 * @function seqOfThresholds - generates an array of thresholds between min and max
 * @property {number}  min - minimum = fastman's value
 * @poperty  {number}  max - maximum = slowman's value
 */
var seqOfThresholds = function(min, max) {
  var range = max - min
  var stepSize =  range/50
  var increment = function(x) {x * stepSize + min}
  mapN(increment, Math.floor(range/stepSize + 1))
}
//-------------------------------------------------------------------------------------
/**
 * @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   = seqOfThresholds(100, 200) // from typical value upto slowmans value
var tauCCrit   = seqOfThresholds( 70, 170) // from typical value upto slowmans value
var tauMCrit   = seqOfThresholds( 70, 100) // from typical value upto slowmans value
var tauSumCrit = seqOfThresholds(240, 470) // from typical value upto slowmans value
print("-------------------------------------------------------------------------")
/**
 * @object hyperParmTauX - parameters 'typical'=c, fast=a, and slow=b for prior Triangle
 *                       - 'typical', fast, and slow are taken from MHP
 *                       - 'typical' = mode = c
 * @property {number} 'typical' - value is the 'typical' value of the CMN-interval
 * @property {number}    a - value is the 'fast' parameter of Triangle(a, b, c)
 * @property {number}    b - value is the 'slow' parameter of Triangle(a, b, c)
 * @property {number}    c - value is the mode or c parameter of Triangle(a, b, c)
 */
var hyperParmTauP = {c:100.0, a:50.0, b:200.0}
print("hyperParmTauP = {c:" + hyperParmTauP.c + ", a:" + hyperParmTauP.a +
      ", b:" + hyperParmTauP.b + "}")
var hyperParmTauC = {c: 70.0, a:25.0, b:170.0}
print("hyperParmTauC = {c: " + hyperParmTauC.c + ", a:" + hyperParmTauC.a +
      ", b:" + hyperParmTauC.b + "}")
var hyperParmTauM = {c: 70.0, a:30.0, b:100.0}
print("hyperParmTauM = {c: " + hyperParmTauM.c + ", 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.l
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 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 displayDiffProbs = function(valsAtRisk, valsAtRiskText, probsPrior, probsPosterior) {
  var probDiffs = map2(function(priorPr, postPr) {
    postPr - priorPr  // change
  }, probsPrior, probsPosterior)
  map2(function(valAtRisk, probDiff) {
    if (probDiff < 0.05) {print(valsAtRiskText + " = " +  valAtRisk
                                + "; increase in risk probs = " + probDiff)}
    else {/* empty */ ;}}
      , valsAtRisk, probDiffs)
  viz.line(valsAtRisk, probDiffs, {xLabel: valsAtRiskText, yLabel: "Risk Excess"})
}
//=============================================================================================
/**
 * @description         - draws one sample from the Triangle(a, b, c)-distribution
 *                      - // https://en.wikipedia.org/wiki/Triangular_distribution
 * @function            - oneSampleOfTriangle
 * @param (number) fast - is the lower bound of the CMN-interval and a of Triangle(a, b, c)
 * @param (number) slow - is the upper bound of the CMN-interval and b of Triangle(a, b, c)
 * @param (number) mode - is the mode of the CMN-interval  and param c of Triangle(a, b, c)
 */
var oneSampleOfTriangle = function(a, b, c) {
  var u  = sample(Uniform({a:0, b:1}))
  var ba = b - a
  var bc = b - c
  var ca = c - a
  var Fc = ca / ba
  var x  = (0 < u) && (u < Fc) ?
      (a + Math.sqrt(u * ba * ca)) :
      (b - Math.sqrt((1 - u) * ba * bc))
  return x
}
//---------------------------------------------------------------------------------------------
```

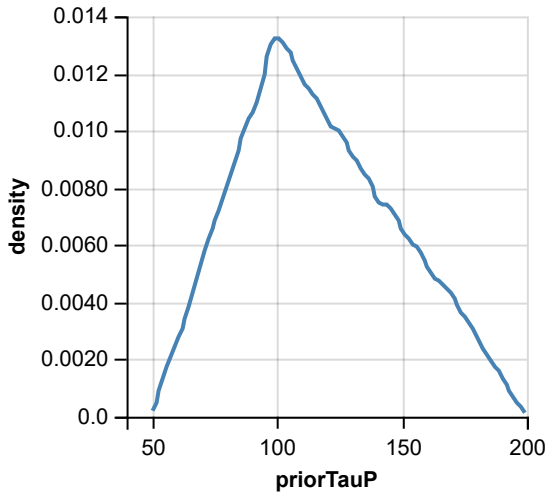
```
/**
 * @function oneSampleOfPriors      - takes  o n e  sample from all priors tauP, tauC, tauM,
 *                                  - tauSum = tauP + tauC + tauM, and sigmaTauSum
 * @returns {object} sampleOfPriors - o n e  priors-tuple
 * @returns {object} priorSigmaTauSum - o n e  sample from the Gamma distr
 *                                  - this is prior sigma for the Gaussian likelihood
 */
var oneSampleOfPriors = function () {
  var priorTauP = oneSampleOfTriangle(hyperParmTauP.a,hyperParmTauP.b,hyperParmTauP.c)
  var priorTauC = oneSampleOfTriangle(hyperParmTauC.a,hyperParmTauC.b,hyperParmTauC.c)
  var priorTauM = oneSampleOfTriangle(hyperParmTauM.a,hyperParmTauM.b,hyperParmTauM.c)
  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 univariate prior Gamma distribution for TauSum
 * @variable {distribution} priorTauSum - 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'), "unknow
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 = oneSampleOfTriangle(hyperParmTauP.a,hyperParmTauP.b,hyperParmTauP.c)
  var priorTauC = oneSampleOfTriangle(hyperParmTauC.a,hyperParmTauC.b,hyperParmTauC.c)
  var priorTauM = oneSampleOfTriangle(hyperParmTauM.a,hyperParmTauM.b,hyperParmTauM.c)
  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 distribution 'posteriorTauT'
 * @variable {distributionObject} posteriorTauT - multivariate 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("-------------------------------------------------------------")
displayDiffProbs(tauPCrit, 'tauPCrit', tauPProbsPrior, tauPProbsPosterior)
print("-------------------------------------------------------------------------------")
var postTauCIntval =
    myTauXDescription("postTauC", marginalize(posterior,'postTauC'), "unknown")
print("{fast:" + postTauCIntval.fast + "  mean:" + postTauCIntval.mean + "  slow:" + postTau(
var tauCProbsPosterior = probsAtRisk(marginalize(posterior,'postTauC'), tauCCrit)
printRiskProbs(tauCCrit, 'tauCCrit', tauCProbsPosterior)
print("-------------------------------------------------------------")
displayDiffProbs(tauCCrit, 'tauCCrit', tauCProbsPrior, tauCProbsPosterior)
print("-------------------------------------------------------------------------------")
var postTauMIntval =
    myTauXDescription("postTauM", marginalize(posterior,'postTauM'), "unknown")
print("{fast:" + postTauMIntval.fast + "  mean:" + postTauMIntval.mean + "  slow:" + postTaul
var tauMProbsPosterior = probsAtRisk(marginalize(posterior,'postTauM'), tauMCrit)
printRiskProbs(tauMCrit, 'tauMCrit', tauMProbsPosterior)
print("-------------------------------------------------------------")
displayDiffProbs(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("-------------------------------------------------------------")
displayDiffProbs(tauSumCrit, 'tauSumCrit', tauSumProbsPrior, tauSumProbsPosterior)
```

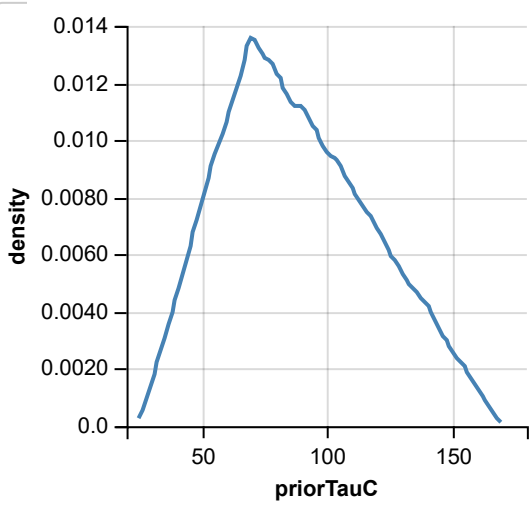
```
print("-------------------------------------------------------------------------")
var postSigmaTauSumIntval =
    myTauXDescription("postSigmaTauSum", marginalize(posterior,'postSigmaTauSum'), "unknown")
print("{fast:" + postSigmaTauSumIntval.fast + "  mean:" + postSigmaTauSumIntval.mean + "  slo
print("=========================================================================")
runTime()
print("=========================================================================")
```

run

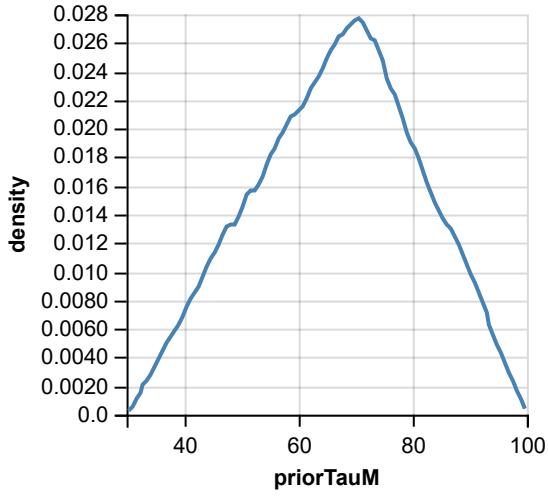
```
=========================================================================
PCM20201212_TriangleModePrior&RiskCalculation             *** 2020/12/12 ***
  see also Simple Reaction Time, Example 9, Card, Moran & Newell, 1983, p.66
  see also https://www.humanbenchmark.com/tests/reactiontime/statistics
  here we use the triangular distribution as a prior distribution
  see also https://en.wikipedia.org/wiki/Triangular_distribution
  CMN-interval 'typical[fast ~ slow]' is interpreted ...
            as  triangle(fast=a, slow=b, 'typical'=mode=c)
=========================================================================
Input parameter:
nTrials = 60000
nSigma = 3
length of burn-in period = 6000
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
-------------------------------------------------------------------------
-------------------------------------------------------------------------
hyperParmTauP = {c:100, a:50, b:200}
hyperParmTauC = {c: 70, a:25, b:170}
hyperParmTauM = {c: 70, a:30, b:100}
-------------------------------------------------------------------------
hyperParmSigmaTauSum = {a:4, b:20}
-------------------------------------------------------------------------
Univariate Priors TauX (X=P, C, M, Sum, SigmaTauSum) ~ Gamma(???, ???)
priorTauP:
```



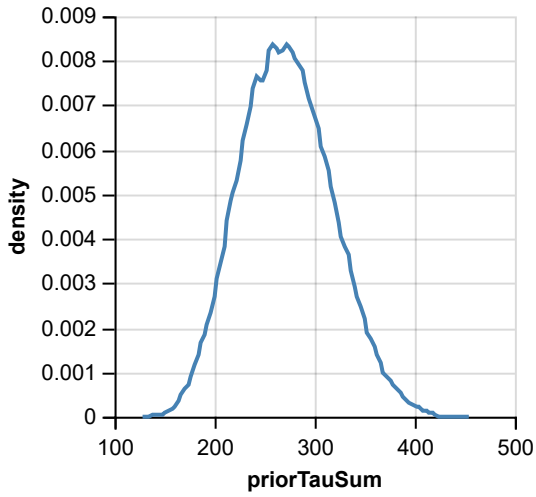

priorTauC:



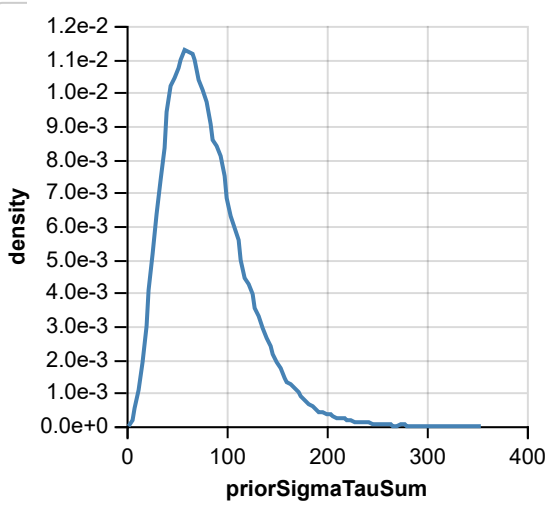

priorTauM:




priorTauSum:




priorSigmaTauSum:




---

model-generated 3*sigma tau-interval:

priorTauP

mode = unknown

mean = 116.69741323038144

sigma = 31.369305989517542

{fast:22.589495261828816 mean:116.69741323038144 slow:210.80533119893408}

---

priorTauC

mode = unknown

mean = 88.25270878644645

sigma = 30.252110728266686

{fast:-2.5036233983536107 mean:88.25270878644645 slow:179.00904097124652}

---

priorTauM

mode = unknown

mean = 66.73448775450258

sigma = 14.355816524021051

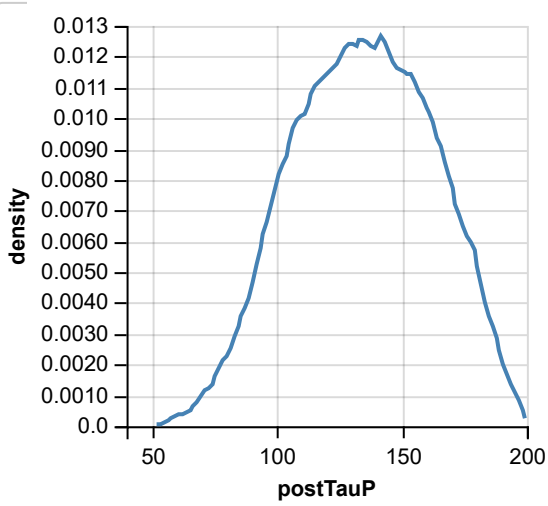
{fast:23.667038182439427 mean:66.73448775450258 slow:109.80193732656574}

---

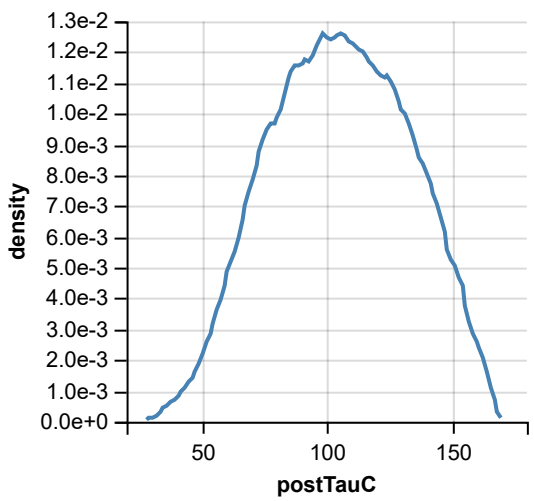
priorTauSum

mode = unknown

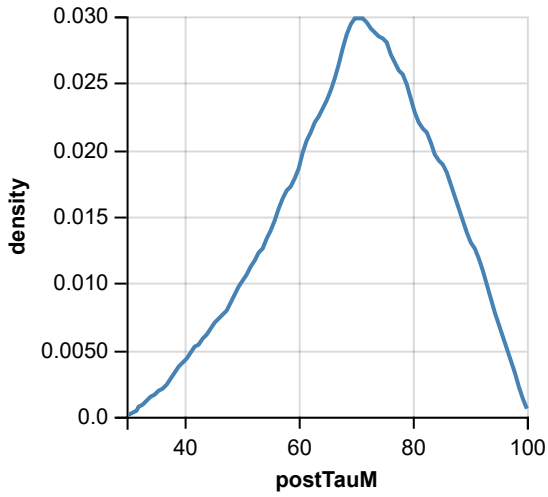
```
mean = 271.6846097713323
sigma = 45.893075153745926
{fast:134.00538431009454  mean:271.6846097713323  slow:409.36383523257007}
------------------------------------------------------------------------------
priorSigmaTauSum
mode = unknown
mean = 79.82583170579565
sigma = 40.1605487409151
model-generated 3*sigma tau-interval:
{fast:-40.65581451694965  mean:79.82583170579565  slow:200.30747792854095}
==============================================================================
Univariate Posteriors TauX (X=P, C, M, Sum) Gamma(???, ???) and SigmaTauSum Gamma
(???, ???)
postTauP:
```



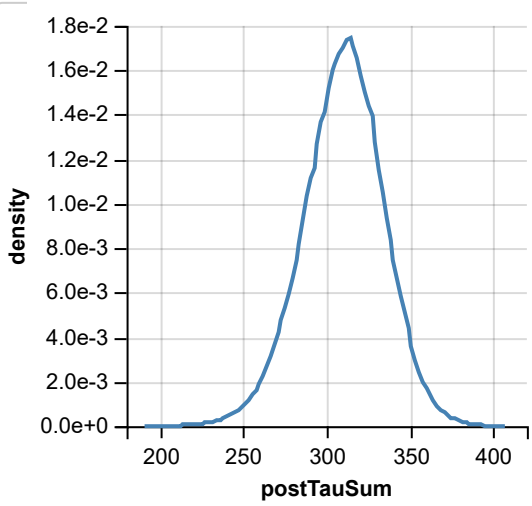

```
postTauC:
```



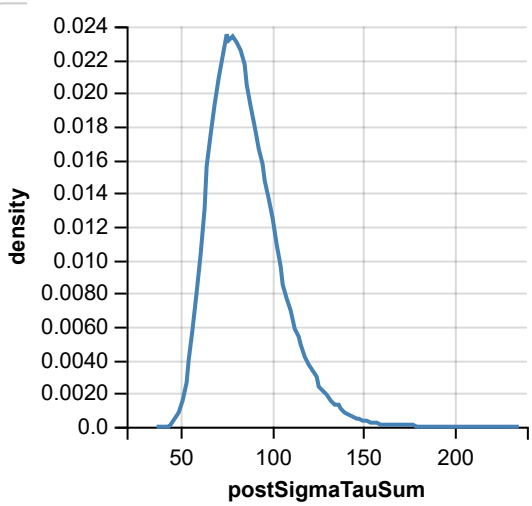

```
postTauM:
```




```
postTauSum:
```

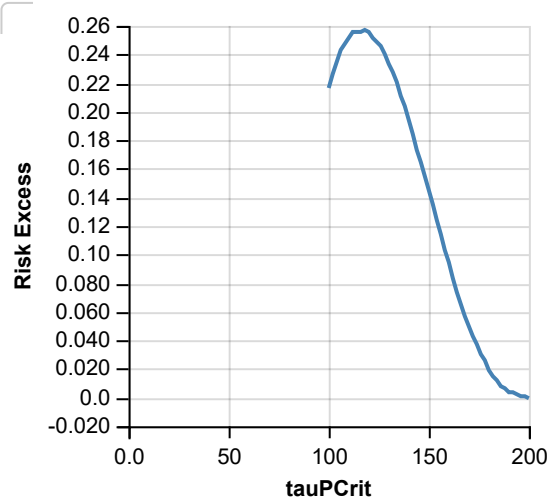



```
postSigmaTauSum:
```



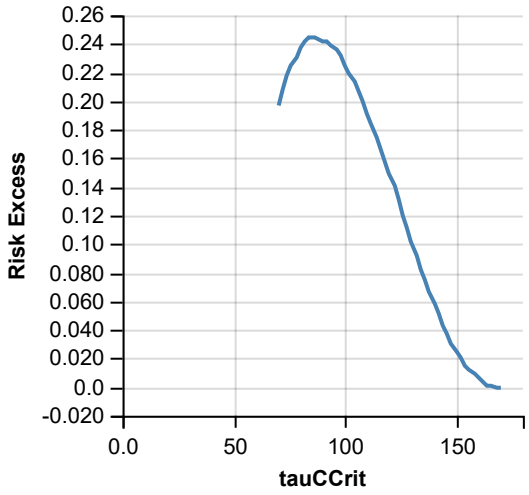

```
--------------------------------------------------------------------------------
model-generated 3*sigma tau-interval:
postTauP
mode = unknown
mean = 134.47664140029102
sigma = 27.949695322751094
{fast:50.62755543203774  mean:134.47664140029102  slow:218.32572736854428}
------------------------------------------------------------
tauPCrit = 172; increase in risk probs = 0.042799999999992955
```

```
tauPCrit = 174; increase in risk probs = 0.037216666666660236
tauPCrit = 176; increase in risk probs = 0.031033333333327584
tauPCrit = 178; increase in risk probs = 0.02636666666666132
tauPCrit = 180; increase in risk probs = 0.019749999999995382
tauPCrit = 182; increase in risk probs = 0.0150499999999959
tauPCrit = 184; increase in risk probs = 0.012849999999996031
tauPCrit = 186; increase in risk probs = 0.008549999999996505
tauPCrit = 188; increase in risk probs = 0.006633333333330049
tauPCrit = 190; increase in risk probs = 0.004449999999996956
tauPCrit = 192; increase in risk probs = 0.0031499999999969885
tauPCrit = 194; increase in risk probs = 0.002333333333330412
tauPCrit = 196; increase in risk probs = 0.0009999999999972253
tauPCrit = 198; increase in risk probs = 0.00041666666666395624
tauPCrit = 200; increase in risk probs = -2.6645352591003757e-15
```



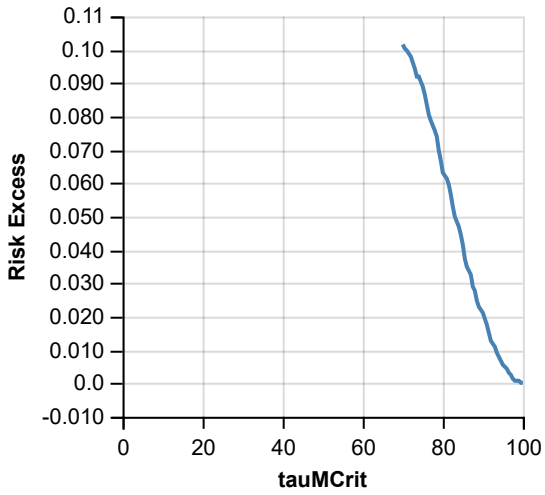

```
------------------------------------------------------------------------
postTauC
mode = unknown
mean = 105.09391516556296
sigma = 27.820525848789224
{fast:21.632337619195297  mean:105.09391516556296  slow:188.55549271193064}
------------------------------------------------------------
tauCCrit = 144; increase in risk probs = 0.04353333333332665
tauCCrit = 146; increase in risk probs = 0.03673333333332729
tauCCrit = 148; increase in risk probs = 0.030783333333327945
tauCCrit = 150; increase in risk probs = 0.024916666666661924
tauCCrit = 152; increase in risk probs = 0.02079999999999571
tauCCrit = 154; increase in risk probs = 0.01503333333332968
tauCCrit = 156; increase in risk probs = 0.01158333333333006
tauCCrit = 158; increase in risk probs = 0.008716666666663597
tauCCrit = 160; increase in risk probs = 0.006433333333330515
tauCCrit = 162; increase in risk probs = 0.0031833333333308733
tauCCrit = 164; increase in risk probs = 0.0015833333333310495
tauCCrit = 166; increase in risk probs = 0.00031666666666441134
tauCCrit = 168; increase in risk probs = 0.00006666666666443888
```

```
tauCCrit = 170; increase in risk probs = -2.220446049250313e-15
```



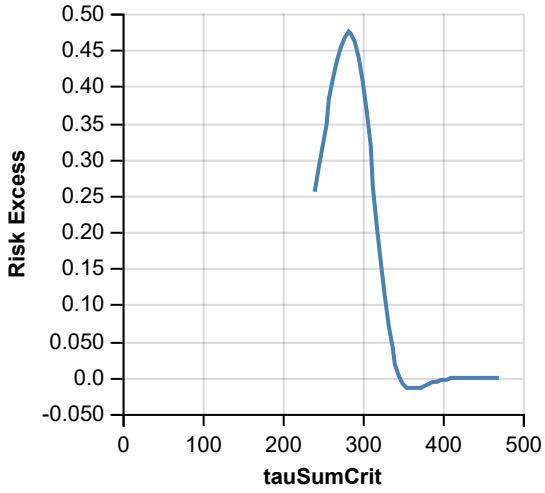

```
------------------------------------------------------------------------------
postTauM
mode = unknown
mean = 70.32273725361163
sigma = 13.62182402910084
{fast:29.457265166309114  mean:70.32273725361163  slow:111.18820934091416}
------------------------------------------------------------
tauMCrit = 83.8; increase in risk probs = 0.047099999999994036
tauMCrit = 84.4; increase in risk probs = 0.04434999999999423
tauMCrit = 85; increase in risk probs = 0.04141666666666122
tauMCrit = 85.6; increase in risk probs = 0.03759999999999508
tauMCrit = 86.2; increase in risk probs = 0.034883333333328714
tauMCrit = 86.8; increase in risk probs = 0.03258333333332886
tauMCrit = 87.4; increase in risk probs = 0.029233333333329226
tauMCrit = 88; increase in risk probs = 0.027616666666662737
tauMCrit = 88.6; increase in risk probs = 0.02496666666666303
tauMCrit = 89.2; increase in risk probs = 0.023099999999996568
tauMCrit = 89.8; increase in risk probs = 0.021099999999996788
tauMCrit = 90.4; increase in risk probs = 0.019149999999997003
tauMCrit = 91; increase in risk probs = 0.01731666666666387
tauMCrit = 91.6; increase in risk probs = 0.01506666666666412
tauMCrit = 92.2; increase in risk probs = 0.012783333333331037
tauMCrit = 92.8; increase in risk probs = 0.011183333333331213
tauMCrit = 93.4; increase in risk probs = 0.009083333333331445
tauMCrit = 94; increase in risk probs = 0.007816666666664918
tauMCrit = 94.6; increase in risk probs = 0.006533333333331726
tauMCrit = 95.2; increase in risk probs = 0.005316666666665082
tauMCrit = 95.8; increase in risk probs = 0.004133333333331879
tauMCrit = 96.4; increase in risk probs = 0.003049999999998665
tauMCrit = 97; increase in risk probs = 0.002466666666665396
tauMCrit = 97.6; increase in risk probs = 0.0011833333333322038
tauMCrit = 98.2; increase in risk probs = 0.0006166666666655995
tauMCrit = 98.8; increase in risk probs = 0.0006166666666655995
```

```
tauMCrit = 99.4; increase in risk probs = 0.00008333333333232495
tauMCrit = 100; increase in risk probs = -9.992007221626409e-16
```




```
------------------------------------------------------------------------
postTauSum
mode = unknown
mean = 309.89329381946413
sigma = 24.21338842156764
{fast:237.25312855476122  mean:309.89329381946413  slow:382.533459084167}
------------------------------------------------------------
tauSumCrit = 336.6; increase in risk probs = 0.04188333333332872
tauSumCrit = 341.2; increase in risk probs = 0.018683333333331276
tauSumCrit = 345.8; increase in risk probs = 0.002149999999999763
tauSumCrit = 350.4; increase in risk probs = -0.008216666666665762
tauSumCrit = 355; increase in risk probs = -0.013399999999998524
tauSumCrit = 359.6; increase in risk probs = -0.0148333333333317
tauSumCrit = 364.2; increase in risk probs = -0.015583333333331617
tauSumCrit = 368.79999999999995; increase in risk probs = -0.014466666666665073
tauSumCrit = 373.4; increase in risk probs = -0.012949999999998574
tauSumCrit = 378; increase in risk probs = -0.010516666666665508
tauSumCrit = 382.6; increase in risk probs = -0.008816666666665696
tauSumCrit = 387.2; increase in risk probs = -0.006816666666665916
tauSumCrit = 391.79999999999995; increase in risk probs = -0.005299999999999416
tauSumCrit = 396.4; increase in risk probs = -0.004016666666666224
tauSumCrit = 401; increase in risk probs = -0.002949999999999675
tauSumCrit = 405.6; increase in risk probs = -0.00188333333333126
tauSumCrit = 410.2; increase in risk probs = -0.001166666666665382
tauSumCrit = 414.79999999999995; increase in risk probs = -0.0006833333333332581
tauSumCrit = 419.4; increase in risk probs = -0.0003333333333332966
tauSumCrit = 424; increase in risk probs = -0.00018333333333331314
tauSumCrit = 428.6; increase in risk probs = -0.00016666666666666483
tauSumCrit = 433.2; increase in risk probs = -0.00006666666666665932
tauSumCrit = 437.79999999999995; increase in risk probs = -0.00003333333333332966
tauSumCrit = 442.4; increase in risk probs = -0.00001666666666666483
tauSumCrit = 447; increase in risk probs = -0.00001666666666666483
```

```
tauSumCrit = 451.6; increase in risk probs = -0.00001666666666666483
tauSumCrit = 456.2; increase in risk probs = 0
tauSumCrit = 460.79999999999995; increase in risk probs = 0
tauSumCrit = 465.4; increase in risk probs = 0
tauSumCrit = 470; increase in risk probs = 0
```




```
------------------------------------------------------------------------
postSigmaTauSum
mode = unknown
mean = 85.56243421239003
sigma = 18.957748415066664
{fast:28.689188967190034  mean:85.56243421239003  slow:142.43567945759003}
========================================================================
runtime in seconds = 8292.426
runtime in minutes = 138.2071
========================================================================
```

## 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