Comparing MVN and MVT spatial random effects

The objective of this is to simulate a model with multivariate-T spatial random effects, and evaluate whether that model does a better job than the conventional random effects model (Multivariate normal).

Spatial data simulation

We'll start and set the seed,

```
set.seed(3)
```

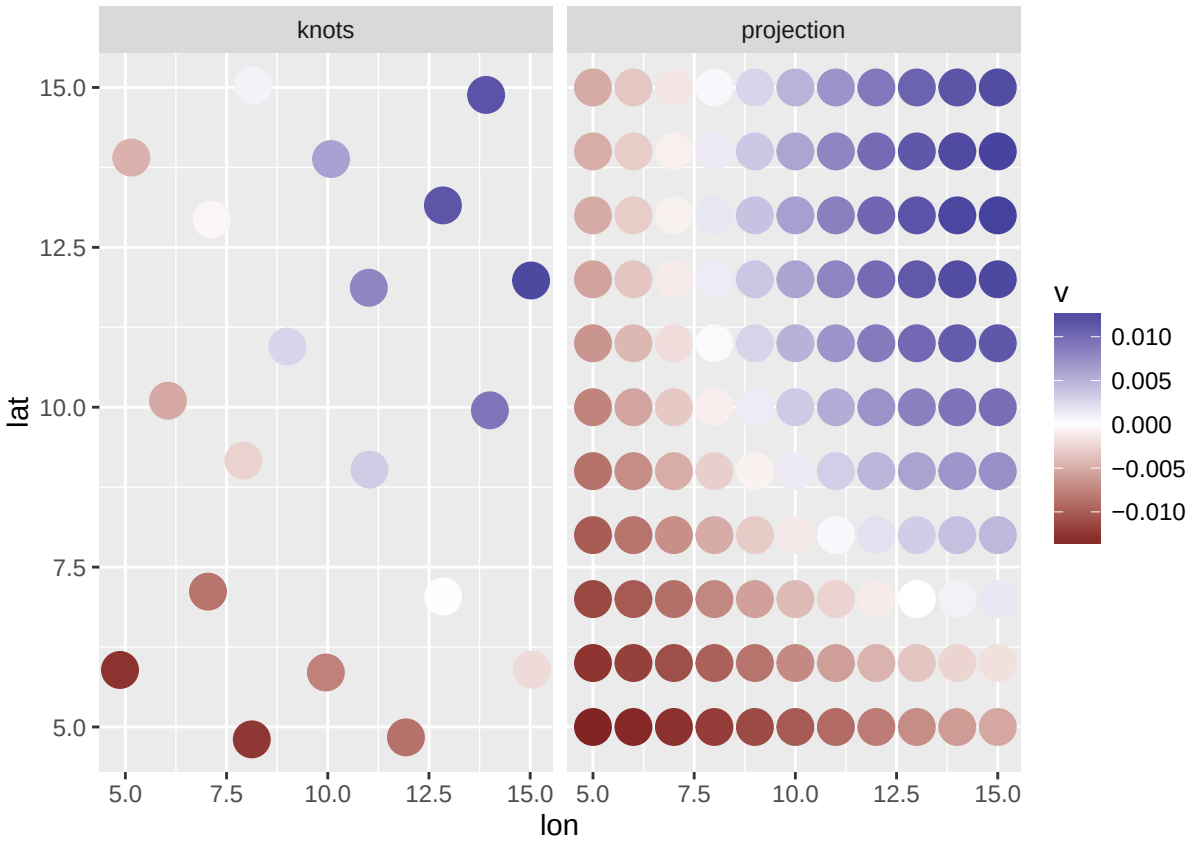
```
# Simulate data on grid
grid = as.matrix(expand.grid(lon = seq(5, 15, 1), lat = seq(5, 15, 1)))
nLocs = dim(grid)[1]

nKnots = 20  # Dimension of random effects
knots = jitter(pam(grid, nKnots)$medoids)
distKnots = as.matrix(dist(knots))
distKnotsSq = distKnots^2  # squared distances
# note: shape parameter scaled to distance matrix
gp_scale = 0.01
sigma.norm = 0.01
corKnots = exp(-gp_scale * distKnotsSq)
Sigma.normal = corKnots * sigma.norm * sigma.norm
invSigmaKnots.norm = solve(Sigma.normal)
# Calculate distance from knots to grid
distAll = as.matrix(dist(rbind(grid, knots)))^2
distKnots21Sq = t(distAll[-c(1:nLocs), -c((nLocs + 1):ncol(distAll))])
Sigma21.normal = exp(-gp_scale * distKnots21Sq) * sigma.norm * sigma.norm
# Generate vector of random effects
re.norm = rmvt(1, sigma = Sigma.normal, df = 2)
re.norm = re.norm - mean(re.norm)  # Scale

# Project random effects to locations of the data
proj.norm = t((Sigma21.normal %*% invSigmaKnots.norm) %*% t(re.norm))

diagKnots = diag(nKnots)
nPoints = length(proj.norm)
muZeros = rep(0, nKnots)
indices = seq(1, nPoints)
```

```
k <- data.frame(knots, v = 1 * t(re.norm), type = "knots")
p <- data.frame(grid, v = t(proj.norm), type = "projection")
d <- rbind(k, p)
ggplot(d, aes(lon, lat, colour = v)) + facet_wrap(~type) + geom_point(size = 6) +
    scale_colour_gradient2()
```



Simulating data with Gamma observation model

```
# Include observation error Use same gamma parameterization as JAGS
gamma.a = 0.03
gamma.b = gamma.a/exp(proj.norm)
# simulate observed data on grid
y.gamma = rgamma(length(proj.norm), shape = gamma.a, rate = gamma.b)

hist(y.gamma, 40, col = "grey", xlab = "Simulated data", main = "")
```

Simulating data with Poisson observation model

```
# Include observation error simulate observed data on grid
y.poisson = rpois(length(proj.norm), exp(proj.norm))

hist(y.poisson, 40, col = "grey", xlab = "Simulated data", main = "")
```

Comparing MVN and MVT random effects with Gamma model

Write the code for the gaussian random effects model with Gamma observation model.

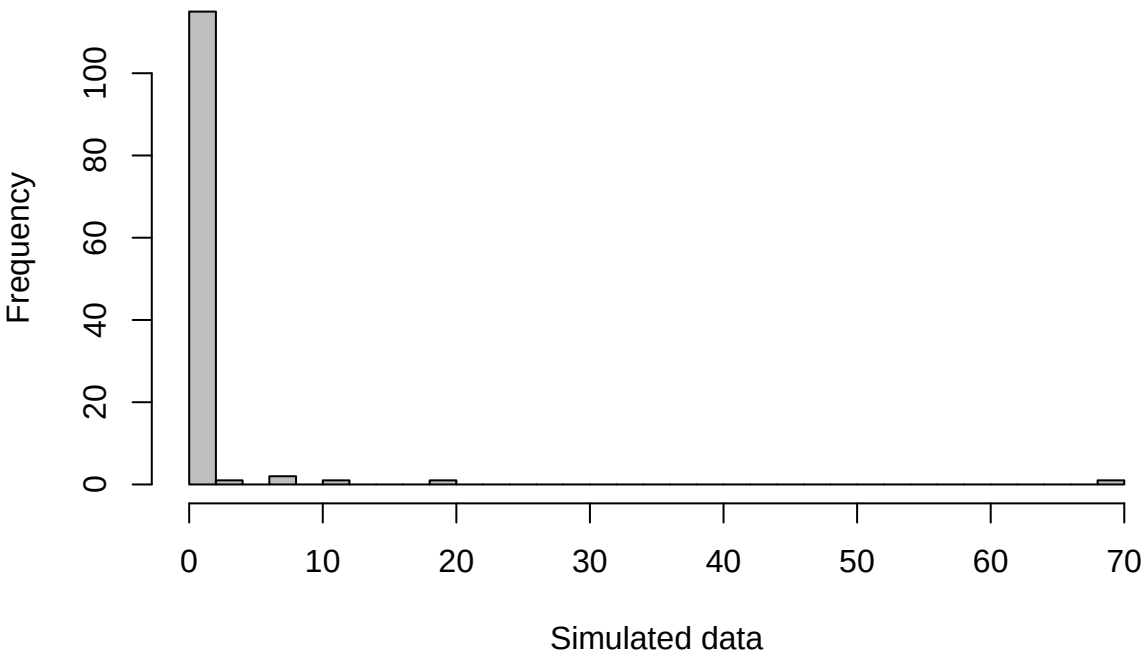


Figure 1: Simulated gamma data, using MVT random effects.

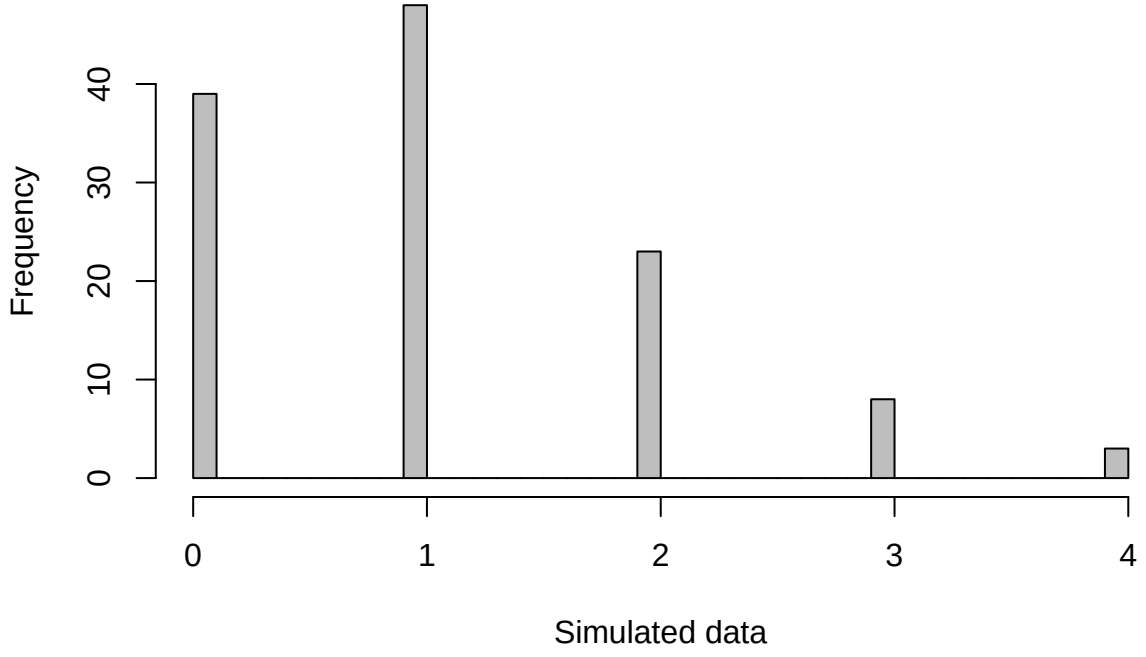


Figure 2: Simulated Poisson data, using MVT random effects.

```
jagsscript = cat("
model {\t
   # priors on parameters for gaussian process
   gp_scaleInv ~ dgamma(0.01,0.01); # shared btw normal/fat
   gp_scale <- 1/gp_scaleInv;
   gp_sigmaSqInv_norm ~ dgamma(0.01,0.01); # prior on normal var parameter
   gp_sigmaSq_norm <- 1/gp_sigmaSqInv_norm;
   gp_jitterSqInv ~ dgamma(0.01,0.01); # shared btw normal/fat \t
   gp_jitterSq <- 1/gp_jitterSqInv;

   # This builds the 2 cov matrices needed
   # SigmaKnots is the COV matrix btween knots
   # SigmaOffDiag is the COV matrix between new locations and knots, e.g. it's (100 x 10)
   for(i in 1:nKnots) {
   \tfor(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
  \t SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + (
  \t}
   }
   for(i in 1:nLocs) {
   \tfor(j in 1:nKnots) {
  \t SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
   \t}
   }\t
   invSigmaKnotsNorm <- inverse(SigmaKnotsNorm[,]); # inverse of matrix for projection and mvn distribu

   # Spatial random effects
   spatialEffectsKnotsNorm[1:nKnots,1] ~ dmnorm(muZeros, invSigmaKnotsNorm);
   spatialEffects[1:nLocs,1] <- (SigmaOffDiagNorm %*% invSigmaKnotsNorm) %*% spatialEffectsKnotsNorm[1:n

   # evaluate the likelihood
   gamma.a ~ dgamma(0.001,0.001); # basically CV
   for(i in 1:nPoints) {
       pred[i] <- exp(min(max(spatialEffects[indices[i], 1], -20), 20));
       y.gamma[i] ~ dgamma(gamma.a, gamma.a/pred[i]);
   }

}  ",
    file = "recover_rf_gaussianGamma.txt")
```

Run the model,

```
jags.data = list("nLocs", "nKnots", "y.gamma", "distKnotsSq", "distKnots21Sq",
    "diagKnots", "muZeros", "indices", "nPoints")
jags.params = c("gp_sigmaSq_norm", "spatialEffectsKnotsNorm", "gamma.a")
jagsmodel.gaussian = jags.parallel(jags.data, inits = NULL, parameters.to.save = jags.params,
    model.file = "recover_rf_gaussianGamma.txt", n.chains = 4, n.burnin = 20000,
    n.thin = 10, n.iter = 30000, DIC = TRUE)
```

Write the same model for the multivariate t distribution. Note that we can't use dmt() in the current version of JAGS, but can convert a multivariate normal distribution to multivariate Student's - t by hand.

```
jagsscript = cat("
model {\t
   # priors on parameters for gaussian process
   gp_scaleInv ~ dgamma(0.01,0.01); # shared btw normal/fat
   gp_scale <- 1/gp_scaleInv;
   gp_sigmaSqInv_norm ~ dgamma(0.01,0.01); # prior on normal var parameter
   gp_sigmaSq_norm <- 1/gp_sigmaSqInv_norm;
   gp_jitterSqInv ~ dgamma(0.01,0.01); # shared btw normal/fat \t
   gp_jitterSq <- 1/gp_jitterSqInv;

   # This builds the 2 cov matrices needed
   # SigmaKnots is the COV matrix btween knots
   # SigmaOffDiag is the COV matrix between new locations and knots, e.g. it's (100 x 10)
   for(i in 1:nKnots) {
   \tfor(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
  \t SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + d
  \t}
   }
   for(i in 1:nLocs) {
   \tfor(j in 1:nKnots) {
  \t SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
   \t}
   }\t
   invSigmaKnotsNorm <- inverse(SigmaKnotsNorm[,]); # inverse of matrix for projection and mvn distribu

   # Spatial random effects. MVT needs to be constructed manually, because of
   # well known problems with mvt() in JAGS. See discussions like this one:
   # http://sourceforge.net/p/mcmc-jags/discussion/610037/thread/491d9ccc/?limit=25
   spatialEffectsKnotsNorm.mvn[1:nKnots,1] ~ dmnorm(muZeros, invSigmaKnotsNorm);
   DF <- 2;
   scale.df ~ dchisq(DF);
   spatialEffectsKnotsNorm[1:nKnots,1] <- spatialEffectsKnotsNorm.mvn[1:nKnots,1] * sqrt(DF/scale.df);
   spatialEffects[1:nLocs,1] <- (SigmaOffDiagNorm %*% invSigmaKnotsNorm) %*% spatialEffectsKnotsNorm[1:n

   # evaluate the likelihood
   gamma.a ~ dgamma(0.001,0.001); # basically CV
   for(i in 1:nPoints) {
       pred[i] <- exp(min(max(spatialEffects[indices[i], 1], -20), 20));
       y.gamma[i] ~ dgamma(gamma.a, gamma.a/pred[i]);
   }

}  ",
    file = "recover_rf_mvtGamma.txt")
```

```
jags.data = list("nLocs", "nKnots", "y.gamma", "distKnotsSq", "distKnots21Sq",
    "diagKnots", "muZeros", "indices", "nPoints")
jags.params = c("gp_sigmaSq_norm", "spatialEffectsKnotsNorm", "gamma.a")
jagsmodel.mvt = jags.parallel(jags.data, inits = NULL, parameters.to.save = jags.params,
    model.file = "recover_rf_mvtGamma.txt", n.chains = 4, n.burnin = 20000,
    n.thin = 10, n.iter = 30000, DIC = TRUE)
```

Comparing MVN and MVT random effects with Poisson model

Write the code for the gaussian random effects model with Poisson observation model.

```
jagsscript = cat("
model {\t
   # priors on parameters for gaussian process
   gp_scaleInv ~ dgamma(0.01,0.01); # shared btw normal/fat
   gp_scale <- 1/gp_scaleInv;
   gp_sigmaSqInv_norm ~ dgamma(0.01,0.01); # prior on normal var parameter
   gp_sigmaSq_norm <- 1/gp_sigmaSqInv_norm;
   gp_jitterSqInv ~ dgamma(0.01,0.01); # shared btw normal/fat \t
   gp_jitterSq <- 1/gp_jitterSqInv;

   # This builds the 2 cov matrices needed
   # SigmaKnots is the COV matrix btween knots
   # SigmaOffDiag is the COV matrix between new locations and knots, e.g. it's (100 x 10)
   for(i in 1:nKnots) {
   \tfor(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
  \t SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + d
  \t}
   }
   for(i in 1:nLocs) {
   \tfor(j in 1:nKnots) {
  \t SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
   \t}
   }\t
   invSigmaKnotsNorm <- inverse(SigmaKnotsNorm[,]); # inverse of matrix for projection and mvn distribu

   # Spatial random effects
   spatialEffectsKnotsNorm[1:nKnots,1] ~ dmnorm(muZeros, invSigmaKnotsNorm);
   spatialEffects[1:nLocs,1] <- (SigmaOffDiagNorm %*% invSigmaKnotsNorm) %*% spatialEffectsKnotsNorm[1:n

   # evaluate the likelihood
   gamma.a ~ dgamma(0.001,0.001); # basically CV
   for(i in 1:nPoints) {
       y.poisson[i] ~ dpois(exp(min(max(spatialEffects[indices[i], 1], -20), 20)));
   }

}  ",
    file = "recover_rf_gaussianPoisson.txt")
```

Run the model,

```
jags.data = list("nLocs", "nKnots", "y.poisson", "distKnotsSq", "distKnots21Sq",
    "diagKnots", "muZeros", "indices", "nPoints")
jags.params = c("gp_sigmaSq_norm", "spatialEffectsKnotsNorm")
jagsmodel.gaussianPoisson = jags.parallel(jags.data, inits = NULL, parameters.to.save = jags.params,
    model.file = "recover_rf_gaussianPoisson.txt", n.chains = 4, n.burnin = 20000,
    n.thin = 10, n.iter = 30000, DIC = TRUE)
```

Write the same model for the multivariate t distribution.

```
jagsscript = cat("
model {\t
   # priors on parameters for gaussian process
   gp_scaleInv ~ dgamma(0.01,0.01); # shared btw normal/fat
   gp_scale <- 1/gp_scaleInv;
   gp_sigmaSqInv_norm ~ dgamma(0.01,0.01); # prior on normal var parameter
   gp_sigmaSq_norm <- 1/gp_sigmaSqInv_norm;
   gp_jitterSqInv ~ dgamma(0.01,0.01); # shared btw normal/fat \t
   gp_jitterSq <- 1/gp_jitterSqInv;

   # This builds the 2 cov matrices needed
   # SigmaKnots is the COV matrix btween knots
   # SigmaOffDiag is the COV matrix between new locations and knots, e.g. it's (100 x 10)
   for(i in 1:nKnots) {
   \tfor(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
  \t SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + d
  \t}
   }
   for(i in 1:nLocs) {
   \tfor(j in 1:nKnots) {
  \t SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
   \t}
   }\t
   invSigmaKnotsNorm <- inverse(SigmaKnotsNorm[,]); # inverse of matrix for projection and mvn distribu

   # Spatial random effects. MVT needs to be constructed manually, because of
   # well known problems with mvt() in JAGS. See discussions like this one:
   # http://sourceforge.net/p/mcmc-jags/discussion/610037/thread/491d9ccc/?limit=25
   spatialEffectsKnotsNorm.mvn[1:nKnots,1] ~ dmnorm(muZeros, invSigmaKnotsNorm);
   DF <- 2;
   scale.df ~ dchisq(DF);
   spatialEffectsKnotsNorm[1:nKnots,1] <- spatialEffectsKnotsNorm.mvn[1:nKnots,1] * sqrt(DF/scale.df);
   spatialEffects[1:nLocs,1] <- (SigmaOffDiagNorm %*% invSigmaKnotsNorm) %*% spatialEffectsKnotsNorm[1:n

   # evaluate the likelihood
   for(i in 1:nPoints) {
       pred[i] <- exp(min(max(spatialEffects[indices[i], 1], -20), 20));
       y.poisson[i] ~ dpois(exp(min(max(spatialEffects[indices[i], 1], -20), 20)));
   }

}  ",
    file = "recover_rf_mvtPoisson.txt")
```

```
jags.data = list("nLocs", "nKnots", "y.poisson", "distKnotsSq", "distKnots21Sq",
    "diagKnots", "muZeros", "indices", "nPoints")
jags.params = c("gp_sigmaSq_norm", "spatialEffectsKnotsNorm")
jagsmodel.mvtPoisson = jags.parallel(jags.data, inits = NULL, parameters.to.save = jags.params,
    model.file = "recover_rf_mvtPoisson.txt", n.chains = 4, n.burnin = 20000,
    n.thin = 10, n.iter = 30000, DIC = TRUE)
```

Comparing model results.

We can switch to a predictive comparison eventually, this table is just to coarsely compare models with DIC. For some reason, real values aren't being spit out to table.

```
m = matrix(NA, 4, 3)
colnames(m) = c("Model", "DIC", "pD")
m[, 1] = c("mvn - Gamma", "mvt - Gamma", "mvn - Poisson", "mvt - Poisson")
m[, 2] = c(jagsmodel.gaussian$BUGSoutput$DIC, jagsmodel.mvt$BUGSoutput$DIC,
    jagsmodel.gaussianPoisson$DIC, jagsmodel.mvtPoisson$DIC)
m[, 3] = c(jagsmodel.gaussian$BUGSoutput$pD, jagsmodel.mvt$BUGSoutput$pD, jagsmodel.gaussianPoisson$pD,
    jagsmodel.mvtPoisson$pD)
# m[,2] = round(m[,2],2)
kable(m)
```

Model	DIC	pD
mvn - Gamma	-5758.82631763423	1.01791140500226
mvt - Gamma	-5758.90042780391	0.96820750301934
mvn - Poisson	1	1.01791140500226
mvt - Poisson	1	0.96820750301934

Model summaries can be printed (not shown) but the other comparison we can make is to the estimated random effects.

```
# print(jagsmodel.gaussian) print(jagsmodel.mvt)

par(mfrow = c(2, 2), mgp = c(2, 1, 0), mai = c(0.5, 0.5, 0.3, 0.05))
mvn.est = apply(jagsmodel.gaussian$BUGSoutput$sims.matrix[, -c(1:3)], 2, median)
mvt.est = apply(jagsmodel.mvt$BUGSoutput$sims.matrix[, -c(1:3)], 2, median)
plot(re.norm, mvn.est, xlab = "true REs @ knots", ylab = "estimates", main = "mvn - Gamma")
plot(re.norm, mvt.est, xlab = "true REs @ knots", ylab = "estimates", main = "mvt - Gamma")
plot(mvn.est, mvt.est, xlab = "MVN estimates @ knots", ylab = "MVT estimates @ knots")
abline(0, 1, col = "red")
```

```
# print(jagsmodel.gaussianPoisson) print(jagsmodel.mvtPoisson)
par(mfrow = c(2, 2), mgp = c(2, 1, 0), mai = c(0.5, 0.5, 0.3, 0.05))
mvn.est = apply(jagsmodel.gaussianPoisson$BUGSoutput$sims.matrix[, -c(1, 2)],
    2, median)
mvt.est = apply(jagsmodel.mvtPoisson$BUGSoutput$sims.matrix[, -c(1, 2)], 2,
    median)
plot(re.norm, mvn.est, xlab = "true REs @ knots", ylab = "estimates", main = "mvn - Poisson")
plot(re.norm, mvt.est, xlab = "true REs @ knots", ylab = "estimates", main = "mvt - Poisson")
plot(mvn.est, mvt.est, xlab = "MVN estimates @ knots", ylab = "MVT estimates @ knots")
abline(0, 1, col = "red")
```

Comparing MVN and MVT random effects with Poisson model (STAN)

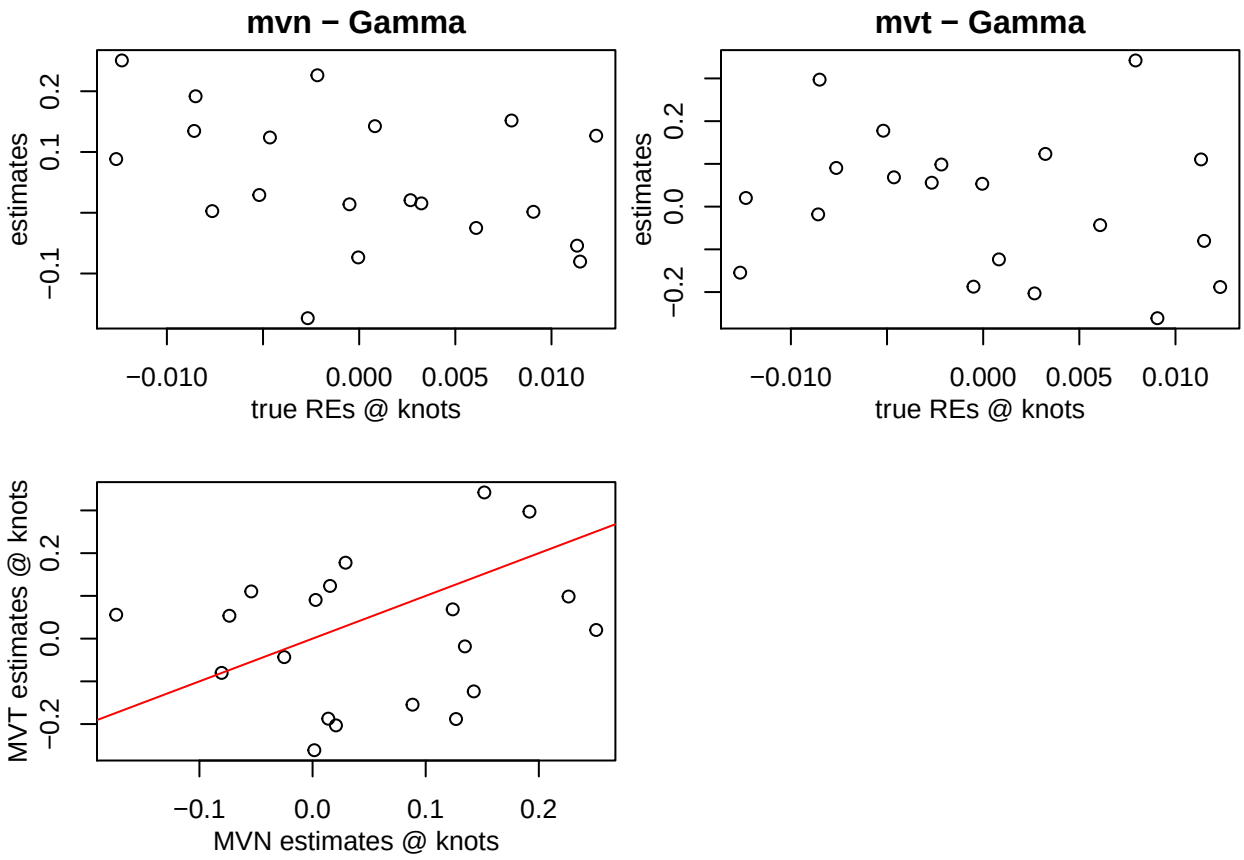


Figure 3: Estimates of random effects for simulated Gamma data

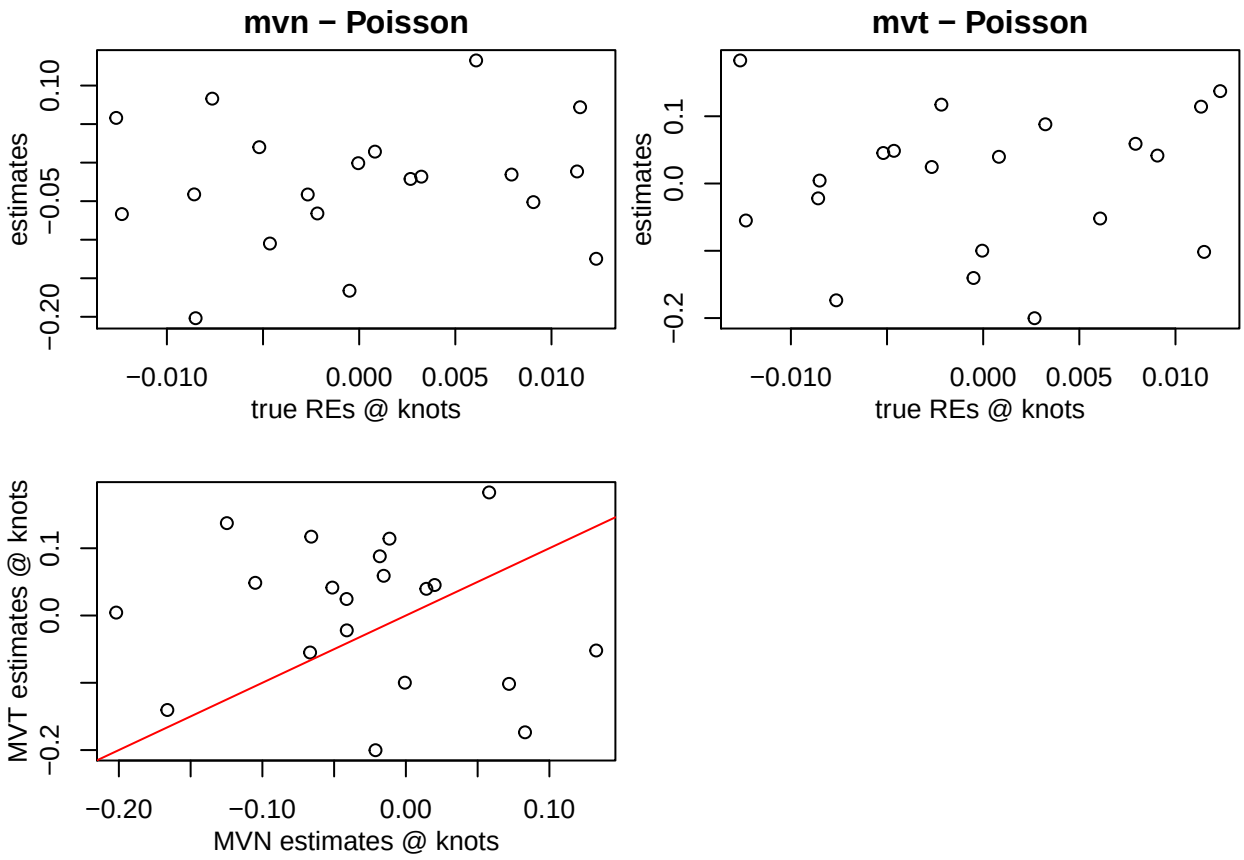


Figure 4: Estimates of random effects for simulated Poisson data

```r
# Named list of data
spatglm_data = list(nKnots = nKnots, nLocs = nLocs, N = nLocs, nT = 1, y = y.poisson,
    location = seq(1, nLocs), distKnotsSq = distKnotsSq, distKnots21Sq = distKnots21Sq)
# parameters to monitor
spatglm_pars = c("gp_scale", "spatialEffectsKnots", "gp_sigmaSq", "jitter_sq")

# fit the model
stanMod_gaussianPoisson = stan(file = "gaussianPoisson.stan", data = spatglm_data,
    verbose = TRUE, chains = 4, thin = 1, warmup = 500, iter = 1000, pars = spatglm_pars)
```

```
##
## TRANSLATING MODEL 'gaussianPoisson' FROM Stan CODE TO C++ CODE NOW.
## successful in parsing the Stan model 'gaussianPoisson'.
## OS: x86_64, darwin13.4.0; rstan: 2.9.0.3; Rcpp: 0.12.5; inline: 0.3.14
##  >> setting environment variables:
## PKG_LIBS =
## PKG_CPPFLAGS =   -isystem"/Users/eric.ward/Library/R/3.2/library/Rcpp/include/"  -isystem"/Users/eri
##  >> Program source :
##
##    1 :
##    2 : // includes from the plugin
##    3 :
##    4 :
##    5 : // user includes
##    6 : #define STAN__SERVICES__COMMAND_HPP// Code generated by Stan version 2.9
##    7 :
##    8 : #include <stan/model/model_header.hpp>
##    9 :
##   10 : namespace model53fb4d5344f6_gaussianPoisson_namespace {
##   11 :
##   12 : using std::istream;
##   13 : using std::string;
##   14 : using std::stringstream;
##   15 : using std::vector;
##   16 : using stan::io::dump;
##   17 : using stan::math::lgamma;
##   18 : using stan::model::prob_grad;
##   19 : using namespace stan::math;
##   20 :
##   21 : typedef Eigen::Matrix<double,Eigen::Dynamic,1> vector_d;
##   22 : typedef Eigen::Matrix<double,1,Eigen::Dynamic> row_vector_d;
##   23 : typedef Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic> matrix_d;
##   24 :
##   25 : static int current_statement_begin__;
##   26 : class model53fb4d5344f6_gaussianPoisson : public prob_grad {
##   27 : private:
##   28 :     int nKnots;
##   29 :     int nLocs;
##   30 :     int N;
##   31 :     int nT;
##   32 :     vector<int> y;
##   33 :     vector<int> location;
##   34 :     matrix_d distKnotsSq;
```

```
##   35 :     matrix_d distKnots21Sq;
##   36 : public:
##   37 :     model53fb4d5344f6_gaussianPoisson(stan::io::var_context& context__,
##   38 :         std::ostream* pstream__ = 0)
##   39 :         : prob_grad(0) {
##   40 :         current_statement_begin__ = -1;
##   41 :
##   42 :         static const char* function__ = "model53fb4d5344f6_gaussianPoisson_namespace::model53
##   43 :         (void) function__; // dummy call to supress warning
##   44 :         size_t pos__;
##   45 :         (void) pos__; // dummy call to supress warning
##   46 :         std::vector<int> vals_i__;
##   47 :         std::vector<double> vals_r__;
##   48 :         context__.validate_dims("data initialization", "nKnots", "int", context__.to_vec());
##   49 :         nKnots = int(0);
##   50 :         vals_i__ = context__.vals_i("nKnots");
##   51 :         pos__ = 0;
##   52 :         nKnots = vals_i__[pos__++];
##   53 :         context__.validate_dims("data initialization", "nLocs", "int", context__.to_vec());
##   54 :         nLocs = int(0);
##   55 :         vals_i__ = context__.vals_i("nLocs");
##   56 :         pos__ = 0;
##   57 :         nLocs = vals_i__[pos__++];
##   58 :         context__.validate_dims("data initialization", "N", "int", context__.to_vec());
##   59 :         N = int(0);
##   60 :         vals_i__ = context__.vals_i("N");
##   61 :         pos__ = 0;
##   62 :         N = vals_i__[pos__++];
##   63 :         context__.validate_dims("data initialization", "nT", "int", context__.to_vec());
##   64 :         nT = int(0);
##   65 :         vals_i__ = context__.vals_i("nT");
##   66 :         pos__ = 0;
##   67 :         nT = vals_i__[pos__++];
##   68 :         context__.validate_dims("data initialization", "y", "int", context__.to_vec(N));
##   69 :         validate_non_negative_index("y", "N", N);
##   70 :         y = std::vector<int>(N,int(0));
##   71 :         vals_i__ = context__.vals_i("y");
##   72 :         pos__ = 0;
##   73 :         size_t y_limit_0__ = N;
##   74 :         for (size_t i_0__ = 0; i_0__ < y_limit_0__; ++i_0__) {
##   75 :             y[i_0__] = vals_i__[pos__++];
##   76 :         }
##   77 :         context__.validate_dims("data initialization", "location", "int", context__.to_vec(N)
##   78 :         validate_non_negative_index("location", "N", N);
##   79 :         location = std::vector<int>(N,int(0));
##   80 :         vals_i__ = context__.vals_i("location");
##   81 :         pos__ = 0;
##   82 :         size_t location_limit_0__ = N;
##   83 :         for (size_t i_0__ = 0; i_0__ < location_limit_0__; ++i_0__) {
##   84 :             location[i_0__] = vals_i__[pos__++];
##   85 :         }
##   86 :         context__.validate_dims("data initialization", "distKnotsSq", "matrix_d", context__.to
##   87 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   88 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
```

```
##   89 :         distKnotsSq = matrix_d(nKnots,nKnots);
##   90 :         vals_r__ = context__.vals_r("distKnotsSq");
##   91 :         pos__ = 0;
##   92 :         size_t distKnotsSq_m_mat_lim__ = nKnots;
##   93 :         size_t distKnotsSq_n_mat_lim__ = nKnots;
##   94 :         for (size_t n_mat__ = 0; n_mat__ < distKnotsSq_n_mat_lim__; ++n_mat__) {
##   95 :             for (size_t m_mat__ = 0; m_mat__ < distKnotsSq_m_mat_lim__; ++m_mat__) {
##   96 :                 distKnotsSq(m_mat__,n_mat__) = vals_r__[pos__++];
##   97 :             }
##   98 :         }
##   99 :         context__.validate_dims("data initialization", "distKnots21Sq", "matrix_d", context__
##  100 :         validate_non_negative_index("distKnots21Sq", "nLocs", nLocs);
##  101 :         validate_non_negative_index("distKnots21Sq", "nKnots", nKnots);
##  102 :         distKnots21Sq = matrix_d(nLocs,nKnots);
##  103 :         vals_r__ = context__.vals_r("distKnots21Sq");
##  104 :         pos__ = 0;
##  105 :         size_t distKnots21Sq_m_mat_lim__ = nLocs;
##  106 :         size_t distKnots21Sq_n_mat_lim__ = nKnots;
##  107 :         for (size_t n_mat__ = 0; n_mat__ < distKnots21Sq_n_mat_lim__; ++n_mat__) {
##  108 :             for (size_t m_mat__ = 0; m_mat__ < distKnots21Sq_m_mat_lim__; ++m_mat__) {
##  109 :                 distKnots21Sq(m_mat__,n_mat__) = vals_r__[pos__++];
##  110 :             }
##  111 :         }
##  112 :
##  113 :         // validate data
##  114 :         check_greater_or_equal(function__,"nKnots",nKnots,1);
##  115 :         check_greater_or_equal(function__,"nLocs",nLocs,1);
##  116 :         check_greater_or_equal(function__,"N",N,1);
##  117 :         check_greater_or_equal(function__,"nT",nT,1);
##  118 :
##  119 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  120 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  121 :
##  122 :
##  123 :         // initialize transformed variables to avoid seg fault on val access
##  124 :
##  125 :         try {
##  126 :         } catch (const std::exception& e) {
##  127 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  128 :             // Next line prevents compiler griping about no return
##  129 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  130 :         }
##  131 :
##  132 :         // validate transformed data
##  133 :
##  134 :         // set parameter ranges
##  135 :         num_params_r__ = 0U;
##  136 :         param_ranges_i__.clear();
##  137 :         ++num_params_r__;
##  138 :         ++num_params_r__;
##  139 :         ++num_params_r__;
##  140 :         num_params_r__ += nKnots * nT;
##  141 :     }
##  142 :
```

```
##  143 :     ~model53fb4d5344f6_gaussianPoisson() { }
##  144 :
##  145 :
##  146 :     void transform_inits(const stan::io::var_context& context__,
##  147 :                          std::vector<int>& params_i__,
##  148 :                          std::vector<double>& params_r__,
##  149 :                          std::ostream* pstream__) const {
##  150 :         stan::io::writer<double> writer__(params_r__,params_i__);
##  151 :         size_t pos__;
##  152 :         (void) pos__; // dummy call to supress warning
##  153 :         std::vector<double> vals_r__;
##  154 :         std::vector<int> vals_i__;
##  155 :
##  156 :         if (!(context__.contains_r("gp_scale")))
##  157 :             throw std::runtime_error("variable gp_scale missing");
##  158 :         vals_r__ = context__.vals_r("gp_scale");
##  159 :         pos__ = 0U;
##  160 :         context__.validate_dims("initialization", "gp_scale", "double", context__.to_vec());
##  161 :         double gp_scale(0);
##  162 :         gp_scale = vals_r__[pos__++];
##  163 :         try {
##  164 :             writer__.scalar_lb_unconstrain(0,gp_scale);
##  165 :         } catch (const std::exception& e) {
##  166 :             throw std::runtime_error(std::string("Error transforming variable gp_scale: ") + e
##  167 :         }
##  168 :
##  169 :         if (!(context__.contains_r("gp_sigmaSq")))
##  170 :             throw std::runtime_error("variable gp_sigmaSq missing");
##  171 :         vals_r__ = context__.vals_r("gp_sigmaSq");
##  172 :         pos__ = 0U;
##  173 :         context__.validate_dims("initialization", "gp_sigmaSq", "double", context__.to_vec())
##  174 :         double gp_sigmaSq(0);
##  175 :         gp_sigmaSq = vals_r__[pos__++];
##  176 :         try {
##  177 :             writer__.scalar_lb_unconstrain(0,gp_sigmaSq);
##  178 :         } catch (const std::exception& e) {
##  179 :             throw std::runtime_error(std::string("Error transforming variable gp_sigmaSq: ") +
##  180 :         }
##  181 :
##  182 :         if (!(context__.contains_r("jitter_sq")))
##  183 :             throw std::runtime_error("variable jitter_sq missing");
##  184 :         vals_r__ = context__.vals_r("jitter_sq");
##  185 :         pos__ = 0U;
##  186 :         context__.validate_dims("initialization", "jitter_sq", "double", context__.to_vec());
##  187 :         double jitter_sq(0);
##  188 :         jitter_sq = vals_r__[pos__++];
##  189 :         try {
##  190 :             writer__.scalar_lb_unconstrain(0,jitter_sq);
##  191 :         } catch (const std::exception& e) {
##  192 :             throw std::runtime_error(std::string("Error transforming variable jitter_sq: ") +
##  193 :         }
##  194 :
##  195 :         if (!(context__.contains_r("spatialEffectsKnots")))
##  196 :             throw std::runtime_error("variable spatialEffectsKnots missing");
```

```
##  197 :         vals_r__ = context__.vals_r("spatialEffectsKnots");
##  198 :         pos__ = 0U;
##  199 :         context__.validate_dims("initialization", "spatialEffectsKnots", "vector_d", context_
##  200 :         std::vector<vector_d> spatialEffectsKnots(nT,vector_d(nKnots));
##  201 :         for (int j1__ = 0U; j1__ < nKnots; ++j1__)
##  202 :             for (int i0__ = 0U; i0__ < nT; ++i0__)
##  203 :                 spatialEffectsKnots[i0__](j1__) = vals_r__[pos__++];
##  204 :         for (int i0__ = 0U; i0__ < nT; ++i0__)
##  205 :             try {
##  206 :             writer__.vector_unconstrain(spatialEffectsKnots[i0__]);
##  207 :         } catch (const std::exception& e) {
##  208 :             throw std::runtime_error(std::string("Error transforming variable spatialEffectsKn
##  209 :         }
##  210 :
##  211 :         params_r__ = writer__.data_r();
##  212 :         params_i__ = writer__.data_i();
##  213 :     }
##  214 :
##  215 :     void transform_inits(const stan::io::var_context& context,
##  216 :                          Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  217 :                          std::ostream* pstream__) const {
##  218 :       std::vector<double> params_r_vec;
##  219 :       std::vector<int> params_i_vec;
##  220 :       transform_inits(context, params_i_vec, params_r_vec, pstream__);
##  221 :       params_r.resize(params_r_vec.size());
##  222 :       for (int i = 0; i < params_r.size(); ++i)
##  223 :         params_r(i) = params_r_vec[i];
##  224 :     }
##  225 :
##  226 :
##  227 :     template <bool propto__, bool jacobian__, typename T__>
##  228 :     T__ log_prob(vector<T__>& params_r__,
##  229 :                  vector<int>& params_i__,
##  230 :                  std::ostream* pstream__ = 0) const {
##  231 :
##  232 :         T__ DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  233 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  234 :
##  235 :         T__ lp__(0.0);
##  236 :         stan::math::accumulator<T__> lp_accum__;
##  237 :
##  238 :         // model parameters
##  239 :         stan::io::reader<T__> in__(params_r__,params_i__);
##  240 :
##  241 :         T__ gp_scale;
##  242 :         (void) gp_scale;   // dummy to suppress unused var warning
##  243 :         if (jacobian__)
##  244 :             gp_scale = in__.scalar_lb_constrain(0,lp__);
##  245 :         else
##  246 :             gp_scale = in__.scalar_lb_constrain(0);
##  247 :
##  248 :         T__ gp_sigmaSq;
##  249 :         (void) gp_sigmaSq;   // dummy to suppress unused var warning
##  250 :         if (jacobian__)
```

```
##  251 :             gp_sigmaSq = in__.scalar_lb_constrain(0,lp__);
##  252 :         else
##  253 :             gp_sigmaSq = in__.scalar_lb_constrain(0);
##  254 :
##  255 :         T__ jitter_sq;
##  256 :         (void) jitter_sq;   // dummy to suppress unused var warning
##  257 :         if (jacobian__)
##  258 :             jitter_sq = in__.scalar_lb_constrain(0,lp__);
##  259 :         else
##  260 :             jitter_sq = in__.scalar_lb_constrain(0);
##  261 :
##  262 :         vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffectsKnots;
##  263 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  264 :         spatialEffectsKnots.reserve(dim_spatialEffectsKnots_0__);
##  265 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  266 :             if (jacobian__)
##  267 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots,lp__));
##  268 :             else
##  269 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  270 :         }
##  271 :
##  272 :
##  273 :         // transformed parameters
##  274 :         Eigen::Matrix<T__,Eigen::Dynamic,1>  muZeros(nKnots);
##  275 :         (void) muZeros;  // dummy to suppress unused var warning
##  276 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  277 :
##  278 :         // initialize transformed variables to avoid seg fault on val access
##  279 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  280 :
##  281 :         try {
##  282 :             current_statement_begin__ = 23;
##  283 :             for (int i = 1; i <= nKnots; ++i) {
##  284 :                 current_statement_begin__ = 24;
##  285 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  286 :             }
##  287 :         } catch (const std::exception& e) {
##  288 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  289 :             // Next line prevents compiler griping about no return
##  290 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  291 :         }
##  292 :
##  293 :         // validate transformed parameters
##  294 :         for (int i0__ = 0; i0__ < nKnots; ++i0__) {
##  295 :             if (stan::math::is_uninitialized(muZeros(i0__))) {
##  296 :                 std::stringstream msg__;
##  297 :                 msg__ << "Undefined transformed parameter: muZeros" << '[' << i0__ << ']';
##  298 :                 throw std::runtime_error(msg__.str());
##  299 :             }
##  300 :         }
##  301 :
##  302 :         const char* function__ = "validate transformed params";
##  303 :         (void) function__; // dummy to suppress unused var warning
##  304 :
```

```
##  305 :         // model body
##  306 :         try {
##  307 :             {
##  308 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots(nKnots,nKnots);
##  309 :                 (void) SigmaKnots;  // dummy to suppress unused var warning
##  310 :                 stan::math::fill(SigmaKnots,DUMMY_VAR__);
##  311 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots_chol(nKnots,nKno
##  312 :                 (void) SigmaKnots_chol;  // dummy to suppress unused var warning
##  313 :                 stan::math::fill(SigmaKnots_chol,DUMMY_VAR__);
##  314 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaOffDiag(nLocs,nKnots);
##  315 :                 (void) SigmaOffDiag;  // dummy to suppress unused var warning
##  316 :                 stan::math::fill(SigmaOffDiag,DUMMY_VAR__);
##  317 :                 vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffects(nT, (Eigen::Matri
##  318 :                 stan::math::fill(spatialEffects,DUMMY_VAR__);
##  319 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  invSigmaKnots(nKnots,nKnots)
##  320 :                 (void) invSigmaKnots;  // dummy to suppress unused var warning
##  321 :                 stan::math::fill(invSigmaKnots,DUMMY_VAR__);
##  322 :                 stan::math::initialize(SigmaKnots, DUMMY_VAR__);
##  323 :                 stan::math::initialize(SigmaKnots_chol, DUMMY_VAR__);
##  324 :                 stan::math::initialize(SigmaOffDiag, DUMMY_VAR__);
##  325 :                 stan::math::initialize(spatialEffects, DUMMY_VAR__);
##  326 :                 stan::math::initialize(invSigmaKnots, DUMMY_VAR__);
##  327 :                 current_statement_begin__ = 35;
##  328 :                 stan::math::assign(SigmaKnots, multiply(gp_sigmaSq,exp(multiply(-(gp_scale),d
##  329 :                 current_statement_begin__ = 36;
##  330 :                 stan::math::assign(SigmaOffDiag, multiply(gp_sigmaSq,exp(multiply(-(gp_scale)
##  331 :                 current_statement_begin__ = 37;
##  332 :                 for (int i = 1; i <= nKnots; ++i) {
##  333 :                     current_statement_begin__ = 38;
##  334 :                     stan::math::assign(get_base1_lhs(SigmaKnots,i,i,"SigmaKnots",1), (jitter_s
##  335 :                     current_statement_begin__ = 39;
##  336 :                     stan::math::assign(get_base1_lhs(SigmaOffDiag,i,i,"SigmaOffDiag",1), (jit
##  337 :                 }
##  338 :                 current_statement_begin__ = 41;
##  339 :                 stan::math::assign(invSigmaKnots, inverse(SigmaKnots));
##  340 :                 current_statement_begin__ = 45;
##  341 :                 stan::math::assign(SigmaKnots_chol, cholesky_decompose(SigmaKnots));
##  342 :                 current_statement_begin__ = 48;
##  343 :                 lp_accum__.add(multi_normal_cholesky_log<propto__>(get_base1(spatialEffectsKn
##  344 :                 current_statement_begin__ = 50;
##  345 :                 stan::math::assign(get_base1_lhs(spatialEffects,1,"spatialEffects",1), multipl
##  346 :                 current_statement_begin__ = 53;
##  347 :                 lp_accum__.add(cauchy_log<propto__>(gp_scale, 0, 5));
##  348 :                 current_statement_begin__ = 54;
##  349 :                 lp_accum__.add(cauchy_log<propto__>(gp_sigmaSq, 0, 5));
##  350 :                 current_statement_begin__ = 55;
##  351 :                 lp_accum__.add(cauchy_log<propto__>(jitter_sq, 0, 5));
##  352 :                 current_statement_begin__ = 57;
##  353 :                 for (int n = 1; n <= N; ++n) {
##  354 :                     current_statement_begin__ = 58;
##  355 :                     lp_accum__.add(poisson_log<propto__>(get_base1(y,n,"y",1), exp(get_base1(g
##  356 :                 }
##  357 :             }
##  358 :         } catch (const std::exception& e) {
```

```
##  359 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  360 :             // Next line prevents compiler griping about no return
##  361 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  362 :         }
##  363 :
##  364 :         lp_accum__.add(lp__);
##  365 :         return lp_accum__.sum();
##  366 :
##  367 :     } // log_prob()
##  368 :
##  369 :     template <bool propto, bool jacobian, typename T_>
##  370 :     T_ log_prob(Eigen::Matrix<T_,Eigen::Dynamic,1>& params_r,
##  371 :                 std::ostream* pstream = 0) const {
##  372 :       std::vector<T_> vec_params_r;
##  373 :       vec_params_r.reserve(params_r.size());
##  374 :       for (int i = 0; i < params_r.size(); ++i)
##  375 :         vec_params_r.push_back(params_r(i));
##  376 :       std::vector<int> vec_params_i;
##  377 :       return log_prob<propto,jacobian,T_>(vec_params_r, vec_params_i, pstream);
##  378 :     }
##  379 :
##  380 :
##  381 :     void get_param_names(std::vector<std::string>& names__) const {
##  382 :         names__.resize(0);
##  383 :         names__.push_back("gp_scale");
##  384 :         names__.push_back("gp_sigmaSq");
##  385 :         names__.push_back("jitter_sq");
##  386 :         names__.push_back("spatialEffectsKnots");
##  387 :         names__.push_back("muZeros");
##  388 :     }
##  389 :
##  390 :
##  391 :     void get_dims(std::vector<std::vector<size_t> >& dimss__) const {
##  392 :         dimss__.resize(0);
##  393 :         std::vector<size_t> dims__;
##  394 :         dims__.resize(0);
##  395 :         dimss__.push_back(dims__);
##  396 :         dims__.resize(0);
##  397 :         dimss__.push_back(dims__);
##  398 :         dims__.resize(0);
##  399 :         dimss__.push_back(dims__);
##  400 :         dims__.resize(0);
##  401 :         dims__.push_back(nT);
##  402 :         dims__.push_back(nKnots);
##  403 :         dimss__.push_back(dims__);
##  404 :         dims__.resize(0);
##  405 :         dims__.push_back(nKnots);
##  406 :         dimss__.push_back(dims__);
##  407 :     }
##  408 :
##  409 :     template <typename RNG>
##  410 :     void write_array(RNG& base_rng__,
##  411 :                      std::vector<double>& params_r__,
##  412 :                      std::vector<int>& params_i__,
```

```
##  413 :                       std::vector<double>& vars__,
##  414 :                       bool include_tparams__ = true,
##  415 :                       bool include_gqs__ = true,
##  416 :                       std::ostream* pstream__ = 0) const {
##  417 :         vars__.resize(0);
##  418 :         stan::io::reader<double> in__(params_r__,params_i__);
##  419 :         static const char* function__ = "model53fb4d5344f6_gaussianPoisson_namespace::write_a
##  420 :         (void) function__; // dummy call to supress warning
##  421 :         // read-transform, write parameters
##  422 :         double gp_scale = in__.scalar_lb_constrain(0);
##  423 :         double gp_sigmaSq = in__.scalar_lb_constrain(0);
##  424 :         double jitter_sq = in__.scalar_lb_constrain(0);
##  425 :         vector<vector_d> spatialEffectsKnots;
##  426 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  427 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  428 :             spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  429 :         }
##  430 :         vars__.push_back(gp_scale);
##  431 :         vars__.push_back(gp_sigmaSq);
##  432 :         vars__.push_back(jitter_sq);
##  433 :         for (int k_1__ = 0; k_1__ < nKnots; ++k_1__) {
##  434 :             for (int k_0__ = 0; k_0__ < nT; ++k_0__) {
##  435 :                 vars__.push_back(spatialEffectsKnots[k_0__][k_1__]);
##  436 :             }
##  437 :         }
##  438 :
##  439 :         if (!include_tparams__) return;
##  440 :         // declare and define transformed parameters
##  441 :         double lp__ = 0.0;
##  442 :         (void) lp__; // dummy call to supress warning
##  443 :         stan::math::accumulator<double> lp_accum__;
##  444 :
##  445 :         vector_d muZeros(nKnots);
##  446 :         (void) muZeros;  // dummy to suppress unused var warning
##  447 :
##  448 :         try {
##  449 :             current_statement_begin__ = 23;
##  450 :             for (int i = 1; i <= nKnots; ++i) {
##  451 :                 current_statement_begin__ = 24;
##  452 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  453 :             }
##  454 :         } catch (const std::exception& e) {
##  455 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  456 :             // Next line prevents compiler griping about no return
##  457 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  458 :         }
##  459 :
##  460 :         // validate transformed parameters
##  461 :
##  462 :         // write transformed parameters
##  463 :         for (int k_0__ = 0; k_0__ < nKnots; ++k_0__) {
##  464 :             vars__.push_back(muZeros[k_0__]);
##  465 :         }
##  466 :
```

```
##  467 :         if (!include_gqs__) return;
##  468 :         // declare and define generated quantities
##  469 :
##  470 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  471 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  472 :
##  473 :
##  474 :         // initialize transformed variables to avoid seg fault on val access
##  475 :
##  476 :         try {
##  477 :         } catch (const std::exception& e) {
##  478 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  479 :             // Next line prevents compiler griping about no return
##  480 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  481 :         }
##  482 :
##  483 :         // validate generated quantities
##  484 :
##  485 :         // write generated quantities
##  486 :     }
##  487 :
##  488 :     template <typename RNG>
##  489 :     void write_array(RNG& base_rng,
##  490 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  491 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& vars,
##  492 :                      bool include_tparams = true,
##  493 :                      bool include_gqs = true,
##  494 :                      std::ostream* pstream = 0) const {
##  495 :       std::vector<double> params_r_vec(params_r.size());
##  496 :       for (int i = 0; i < params_r.size(); ++i)
##  497 :         params_r_vec[i] = params_r(i);
##  498 :       std::vector<double> vars_vec;
##  499 :       std::vector<int> params_i_vec;
##  500 :       write_array(base_rng,params_r_vec,params_i_vec,vars_vec,include_tparams,include_gqs,pst
##  501 :       vars.resize(vars_vec.size());
##  502 :       for (int i = 0; i < vars.size(); ++i)
##  503 :         vars(i) = vars_vec[i];
##  504 :     }
##  505 :
##  506 :     static std::string model_name() {
##  507 :         return "model53fb4d5344f6_gaussianPoisson";
##  508 :     }
##  509 :
##  510 :
##  511 :     void constrained_param_names(std::vector<std::string>& param_names__,
##  512 :                                  bool include_tparams__ = true,
##  513 :                                  bool include_gqs__ = true) const {
##  514 :         std::stringstream param_name_stream__;
##  515 :         param_name_stream__.str(std::string());
##  516 :         param_name_stream__ << "gp_scale";
##  517 :         param_names__.push_back(param_name_stream__.str());
##  518 :         param_name_stream__.str(std::string());
##  519 :         param_name_stream__ << "gp_sigmaSq";
##  520 :         param_names__.push_back(param_name_stream__.str());
```

```
##  521 :         param_name_stream__.str(std::string());
##  522 :         param_name_stream__ << "jitter_sq";
##  523 :         param_names__.push_back(param_name_stream__.str());
##  524 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  525 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  526 :                 param_name_stream__.str(std::string());
##  527 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  528 :                 param_names__.push_back(param_name_stream__.str());
##  529 :             }
##  530 :         }
##  531 :
##  532 :         if (!include_gqs__ && !include_tparams__) return;
##  533 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  534 :             param_name_stream__.str(std::string());
##  535 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  536 :             param_names__.push_back(param_name_stream__.str());
##  537 :         }
##  538 :
##  539 :         if (!include_gqs__) return;
##  540 :     }
##  541 :
##  542 :
##  543 :     void unconstrained_param_names(std::vector<std::string>& param_names__,
##  544 :                                    bool include_tparams__ = true,
##  545 :                                    bool include_gqs__ = true) const {
##  546 :         std::stringstream param_name_stream__;
##  547 :         param_name_stream__.str(std::string());
##  548 :         param_name_stream__ << "gp_scale";
##  549 :         param_names__.push_back(param_name_stream__.str());
##  550 :         param_name_stream__.str(std::string());
##  551 :         param_name_stream__ << "gp_sigmaSq";
##  552 :         param_names__.push_back(param_name_stream__.str());
##  553 :         param_name_stream__.str(std::string());
##  554 :         param_name_stream__ << "jitter_sq";
##  555 :         param_names__.push_back(param_name_stream__.str());
##  556 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  557 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  558 :                 param_name_stream__.str(std::string());
##  559 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  560 :                 param_names__.push_back(param_name_stream__.str());
##  561 :             }
##  562 :         }
##  563 :
##  564 :         if (!include_gqs__ && !include_tparams__) return;
##  565 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  566 :             param_name_stream__.str(std::string());
##  567 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  568 :             param_names__.push_back(param_name_stream__.str());
##  569 :         }
##  570 :
##  571 :         if (!include_gqs__) return;
##  572 :     }
##  573 :
##  574 : }; // model
```

```
##  575 :
##  576 : } // namespace
##  577 :
##  578 : typedef model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_gaussianPoisson stan_m
##  579 :
##  580 : #include <rstan/rstaninc.hpp>
##  581 : /**
##  582 :  * Define Rcpp Module to expose stan_fit's functions to R.
##  583 :  */
##  584 : RCPP_MODULE(stan_fit4model53fb4d5344f6_gaussianPoisson_mod){
##  585 :   Rcpp::class_<rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_
##  586 :                boost::random::ecuyer1988> >("stan_fit4model53fb4d5344f6_gaussianPoisson")
##  587 :     // .constructor<Rcpp::List>()
##  588 :     .constructor<SEXP, SEXP>()
##  589 :     // .constructor<SEXP, SEXP>()
##  590 :     .method("call_sampler",
##  591 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  592 :     .method("param_names",
##  593 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  594 :     .method("param_names_oi",
##  595 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  596 :     .method("param_fnames_oi",
##  597 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  598 :     .method("param_dims",
##  599 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  600 :     .method("param_dims_oi",
##  601 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  602 :     .method("update_param_oi",
##  603 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  604 :     .method("param_oi_tidx",
##  605 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  606 :     .method("grad_log_prob",
##  607 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  608 :     .method("log_prob",
##  609 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  610 :     .method("unconstrain_pars",
##  611 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  612 :     .method("constrain_pars",
##  613 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  614 :     .method("num_pars_unconstrained",
##  615 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  616 :     .method("unconstrained_param_names",
##  617 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  618 :     .method("constrained_param_names",
##  619 :             &rstan::stan_fit<model53fb4d5344f6_gaussianPoisson_namespace::model53fb4d5344f6_ga
##  620 :     ;
##  621 : }
##  622 :
##  623 : // declarations
##  624 : extern "C" {
##  625 : SEXP file53fb15d49a23( ) ;
##  626 : }
##  627 :
##  628 : // definition
```

```
##  629 :
##  630 : SEXP file53fb15d49a23(  ){
##  631 :  return Rcpp::wrap("gaussianPoisson");
##  632 : }
##  633 :
##  634 :
## Compilation argument:
##  /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB file53fb15d49a23.cpp 2> file53fb15d49a23.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'gaussianPoisson' NOW.
##
## COMPILING MODEL 'gaussianPoisson' NOW.
##
## STARTING SAMPLER FOR MODEL 'gaussianPoisson' NOW.
```

```
# Named list of data
spatglm_data = list(nKnots = nKnots, nLocs = nLocs, N = nLocs, nT = 1, y = y.poisson,
    location = seq(1, nLocs), distKnotsSq = distKnotsSq, distKnots21Sq = distKnots21Sq)
# parameters to monitor
spatglm_pars = c("gp_scale", "spatialEffectsKnots", "gp_sigmaSq", "jitter_sq")

# fit the model
stanMod_mvtPoisson = stan(file = "mvtPoisson.stan", data = spatglm_data, verbose = TRUE,
    chains = 4, thin = 1, warmup = 500, iter = 1000, pars = spatglm_pars)
```

```
##
## TRANSLATING MODEL 'mvtPoisson' FROM Stan CODE TO C++ CODE NOW.
## successful in parsing the Stan model 'mvtPoisson'.
## OS: x86_64, darwin13.4.0; rstan: 2.9.0.3; Rcpp: 0.12.5; inline: 0.3.14
##  >> setting environment variables:
## PKG_LIBS =
## PKG_CPPFLAGS =   -isystem"/Users/eric.ward/Library/R/3.2/library/Rcpp/include/"  -isystem"/Users/eri
##  >> Program source :
##
##    1 :
##    2 : // includes from the plugin
##    3 :
##    4 :
##    5 : // user includes
##    6 : #define STAN__SERVICES__COMMAND_HPP// Code generated by Stan version 2.9
##    7 :
##    8 : #include <stan/model/model_header.hpp>
##    9 :
##   10 : namespace model53fb443225b6_mvtPoisson_namespace {
##   11 :
##   12 : using std::istream;
##   13 : using std::string;
##   14 : using std::stringstream;
##   15 : using std::vector;
##   16 : using stan::io::dump;
##   17 : using stan::math::lgamma;
##   18 : using stan::model::prob_grad;
##   19 : using namespace stan::math;
##   20 :
```

```
##   21 : typedef Eigen::Matrix<double,Eigen::Dynamic,1> vector_d;
##   22 : typedef Eigen::Matrix<double,1,Eigen::Dynamic> row_vector_d;
##   23 : typedef Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic> matrix_d;
##   24 :
##   25 : static int current_statement_begin__;
##   26 : class model53fb443225b6_mvtPoisson : public prob_grad {
##   27 : private:
##   28 :     int nKnots;
##   29 :     int nLocs;
##   30 :     int N;
##   31 :     int nT;
##   32 :     vector<int> y;
##   33 :     vector<int> location;
##   34 :     matrix_d distKnotsSq;
##   35 :     matrix_d distKnots21Sq;
##   36 : public:
##   37 :     model53fb443225b6_mvtPoisson(stan::io::var_context& context__,
##   38 :         std::ostream* pstream__ = 0)
##   39 :         : prob_grad(0) {
##   40 :         current_statement_begin__ = -1;
##   41 :
##   42 :         static const char* function__ = "model53fb443225b6_mvtPoisson_namespace::model53fb443
##   43 :         (void) function__; // dummy call to supress warning
##   44 :         size_t pos__;
##   45 :         (void) pos__; // dummy call to supress warning
##   46 :         std::vector<int> vals_i__;
##   47 :         std::vector<double> vals_r__;
##   48 :         context__.validate_dims("data initialization", "nKnots", "int", context__.to_vec());
##   49 :         nKnots = int(0);
##   50 :         vals_i__ = context__.vals_i("nKnots");
##   51 :         pos__ = 0;
##   52 :         nKnots = vals_i__[pos__++];
##   53 :         context__.validate_dims("data initialization", "nLocs", "int", context__.to_vec());
##   54 :         nLocs = int(0);
##   55 :         vals_i__ = context__.vals_i("nLocs");
##   56 :         pos__ = 0;
##   57 :         nLocs = vals_i__[pos__++];
##   58 :         context__.validate_dims("data initialization", "N", "int", context__.to_vec());
##   59 :         N = int(0);
##   60 :         vals_i__ = context__.vals_i("N");
##   61 :         pos__ = 0;
##   62 :         N = vals_i__[pos__++];
##   63 :         context__.validate_dims("data initialization", "nT", "int", context__.to_vec());
##   64 :         nT = int(0);
##   65 :         vals_i__ = context__.vals_i("nT");
##   66 :         pos__ = 0;
##   67 :         nT = vals_i__[pos__++];
##   68 :         context__.validate_dims("data initialization", "y", "int", context__.to_vec(N));
##   69 :         validate_non_negative_index("y", "N", N);
##   70 :         y = std::vector<int>(N,int(0));
##   71 :         vals_i__ = context__.vals_i("y");
##   72 :         pos__ = 0;
##   73 :         size_t y_limit_0__ = N;
##   74 :         for (size_t i_0__ = 0; i_0__ < y_limit_0__; ++i_0__) {
```

```
##   75 :             y[i_0__] = vals_i__[pos__++];
##   76 :         }
##   77 :         context__.validate_dims("data initialization", "location", "int", context__.to_vec(N))
##   78 :         validate_non_negative_index("location", "N", N);
##   79 :         location = std::vector<int>(N,int(0));
##   80 :         vals_i__ = context__.vals_i("location");
##   81 :         pos__ = 0;
##   82 :         size_t location_limit_0__ = N;
##   83 :         for (size_t i_0__ = 0; i_0__ < location_limit_0__; ++i_0__) {
##   84 :             location[i_0__] = vals_i__[pos__++];
##   85 :         }
##   86 :         context__.validate_dims("data initialization", "distKnotsSq", "matrix_d", context__.to
##   87 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   88 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   89 :         distKnotsSq = matrix_d(nKnots,nKnots);
##   90 :         vals_r__ = context__.vals_r("distKnotsSq");
##   91 :         pos__ = 0;
##   92 :         size_t distKnotsSq_m_mat_lim__ = nKnots;
##   93 :         size_t distKnotsSq_n_mat_lim__ = nKnots;
##   94 :         for (size_t n_mat__ = 0; n_mat__ < distKnotsSq_n_mat_lim__; ++n_mat__) {
##   95 :             for (size_t m_mat__ = 0; m_mat__ < distKnotsSq_m_mat_lim__; ++m_mat__) {
##   96 :                 distKnotsSq(m_mat__,n_mat__) = vals_r__[pos__++];
##   97 :             }
##   98 :         }
##   99 :         context__.validate_dims("data initialization", "distKnots21Sq", "matrix_d", context__
##  100 :         validate_non_negative_index("distKnots21Sq", "nLocs", nLocs);
##  101 :         validate_non_negative_index("distKnots21Sq", "nKnots", nKnots);
##  102 :         distKnots21Sq = matrix_d(nLocs,nKnots);
##  103 :         vals_r__ = context__.vals_r("distKnots21Sq");
##  104 :         pos__ = 0;
##  105 :         size_t distKnots21Sq_m_mat_lim__ = nLocs;
##  106 :         size_t distKnots21Sq_n_mat_lim__ = nKnots;
##  107 :         for (size_t n_mat__ = 0; n_mat__ < distKnots21Sq_n_mat_lim__; ++n_mat__) {
##  108 :             for (size_t m_mat__ = 0; m_mat__ < distKnots21Sq_m_mat_lim__; ++m_mat__) {
##  109 :                 distKnots21Sq(m_mat__,n_mat__) = vals_r__[pos__++];
##  110 :             }
##  111 :         }
##  112 :
##  113 :         // validate data
##  114 :         check_greater_or_equal(function__,"nKnots",nKnots,1);
##  115 :         check_greater_or_equal(function__,"nLocs",nLocs,1);
##  116 :         check_greater_or_equal(function__,"N",N,1);
##  117 :         check_greater_or_equal(function__,"nT",nT,1);
##  118 :
##  119 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  120 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  121 :
##  122 :
##  123 :         // initialize transformed variables to avoid seg fault on val access
##  124 :
##  125 :         try {
##  126 :         } catch (const std::exception& e) {
##  127 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  128 :             // Next line prevents compiler griping about no return
```

```
##  129 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  130 :         }
##  131 :
##  132 :         // validate transformed data
##  133 :
##  134 :         // set parameter ranges
##  135 :         num_params_r__ = 0U;
##  136 :         param_ranges_i__.clear();
##  137 :         ++num_params_r__;
##  138 :         ++num_params_r__;
##  139 :         ++num_params_r__;
##  140 :         ++num_params_r__;
##  141 :         num_params_r__ += nKnots * nT;
##  142 :     }
##  143 :
##  144 :     ~model53fb443225b6_mvtPoisson() { }
##  145 :
##  146 :
##  147 :     void transform_inits(const stan::io::var_context& context__,
##  148 :                          std::vector<int>& params_i__,
##  149 :                          std::vector<double>& params_r__,
##  150 :                          std::ostream* pstream__) const {
##  151 :         stan::io::writer<double> writer__(params_r__,params_i__);
##  152 :         size_t pos__;
##  153 :         (void) pos__; // dummy call to supress warning
##  154 :         std::vector<double> vals_r__;
##  155 :         std::vector<int> vals_i__;
##  156 :
##  157 :         if (!(context__.contains_r("gp_scale")))
##  158 :             throw std::runtime_error("variable gp_scale missing");
##  159 :         vals_r__ = context__.vals_r("gp_scale");
##  160 :         pos__ = 0U;
##  161 :         context__.validate_dims("initialization", "gp_scale", "double", context__.to_vec());
##  162 :         double gp_scale(0);
##  163 :         gp_scale = vals_r__[pos__++];
##  164 :         try {
##  165 :             writer__.scalar_lb_unconstrain(0,gp_scale);
##  166 :         } catch (const std::exception& e) {
##  167 :             throw std::runtime_error(std::string("Error transforming variable gp_scale: ") + e
##  168 :         }
##  169 :
##  170 :         if (!(context__.contains_r("gp_sigmaSq")))
##  171 :             throw std::runtime_error("variable gp_sigmaSq missing");
##  172 :         vals_r__ = context__.vals_r("gp_sigmaSq");
##  173 :         pos__ = 0U;
##  174 :         context__.validate_dims("initialization", "gp_sigmaSq", "double", context__.to_vec())
##  175 :         double gp_sigmaSq(0);
##  176 :         gp_sigmaSq = vals_r__[pos__++];
##  177 :         try {
##  178 :             writer__.scalar_lb_unconstrain(0,gp_sigmaSq);
##  179 :         } catch (const std::exception& e) {
##  180 :             throw std::runtime_error(std::string("Error transforming variable gp_sigmaSq: ") +
##  181 :         }
##  182 :
```

```
##  183 :         if (!(context__.contains_r("jitter_sq")))
##  184 :             throw std::runtime_error("variable jitter_sq missing");
##  185 :         vals_r__ = context__.vals_r("jitter_sq");
##  186 :         pos__ = 0U;
##  187 :         context__.validate_dims("initialization", "jitter_sq", "double", context__.to_vec());
##  188 :         double jitter_sq(0);
##  189 :         jitter_sq = vals_r__[pos__++];
##  190 :         try {
##  191 :             writer__.scalar_lb_unconstrain(0,jitter_sq);
##  192 :         } catch (const std::exception& e) {
##  193 :             throw std::runtime_error(std::string("Error transforming variable jitter_sq: ") +
##  194 :         }
##  195 :
##  196 :         if (!(context__.contains_r("scaledf")))
##  197 :             throw std::runtime_error("variable scaledf missing");
##  198 :         vals_r__ = context__.vals_r("scaledf");
##  199 :         pos__ = 0U;
##  200 :         context__.validate_dims("initialization", "scaledf", "double", context__.to_vec());
##  201 :         double scaledf(0);
##  202 :         scaledf = vals_r__[pos__++];
##  203 :         try {
##  204 :             writer__.scalar_lb_unconstrain(0,scaledf);
##  205 :         } catch (const std::exception& e) {
##  206 :             throw std::runtime_error(std::string("Error transforming variable scaledf: ") + e
##  207 :         }
##  208 :
##  209 :         if (!(context__.contains_r("spatialEffectsKnots")))
##  210 :             throw std::runtime_error("variable spatialEffectsKnots missing");
##  211 :         vals_r__ = context__.vals_r("spatialEffectsKnots");
##  212 :         pos__ = 0U;
##  213 :         context__.validate_dims("initialization", "spatialEffectsKnots", "vector_d", context_
##  214 :         std::vector<vector_d> spatialEffectsKnots(nT,vector_d(nKnots));
##  215 :         for (int j1__ = 0U; j1__ < nKnots; ++j1__)
##  216 :             for (int i0__ = 0U; i0__ < nT; ++i0__)
##  217 :                 spatialEffectsKnots[i0__](j1__) = vals_r__[pos__++];
##  218 :         for (int i0__ = 0U; i0__ < nT; ++i0__)
##  219 :             try {
##  220 :             writer__.vector_unconstrain(spatialEffectsKnots[i0__]);
##  221 :         } catch (const std::exception& e) {
##  222 :             throw std::runtime_error(std::string("Error transforming variable spatialEffectsKn
##  223 :         }
##  224 :
##  225 :         params_r__ = writer__.data_r();
##  226 :         params_i__ = writer__.data_i();
##  227 :     }
##  228 :
##  229 :     void transform_inits(const stan::io::var_context& context,
##  230 :                          Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  231 :                          std::ostream* pstream__) const {
##  232 :       std::vector<double> params_r_vec;
##  233 :       std::vector<int> params_i_vec;
##  234 :       transform_inits(context, params_i_vec, params_r_vec, pstream__);
##  235 :       params_r.resize(params_r_vec.size());
##  236 :       for (int i = 0; i < params_r.size(); ++i)
```

```
##  237 :         params_r(i) = params_r_vec[i];
##  238 :     }
##  239 :
##  240 :
##  241 :     template <bool propto__, bool jacobian__, typename T__>
##  242 :     T__ log_prob(vector<T__>& params_r__,
##  243 :                  vector<int>& params_i__,
##  244 :                  std::ostream* pstream__ = 0) const {
##  245 :
##  246 :         T__ DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  247 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  248 :
##  249 :         T__ lp__(0.0);
##  250 :         stan::math::accumulator<T__> lp_accum__;
##  251 :
##  252 :         // model parameters
##  253 :         stan::io::reader<T__> in__(params_r__,params_i__);
##  254 :
##  255 :         T__ gp_scale;
##  256 :         (void) gp_scale;   // dummy to suppress unused var warning
##  257 :         if (jacobian__)
##  258 :             gp_scale = in__.scalar_lb_constrain(0,lp__);
##  259 :         else
##  260 :             gp_scale = in__.scalar_lb_constrain(0);
##  261 :
##  262 :         T__ gp_sigmaSq;
##  263 :         (void) gp_sigmaSq;   // dummy to suppress unused var warning
##  264 :         if (jacobian__)
##  265 :             gp_sigmaSq = in__.scalar_lb_constrain(0,lp__);
##  266 :         else
##  267 :             gp_sigmaSq = in__.scalar_lb_constrain(0);
##  268 :
##  269 :         T__ jitter_sq;
##  270 :         (void) jitter_sq;   // dummy to suppress unused var warning
##  271 :         if (jacobian__)
##  272 :             jitter_sq = in__.scalar_lb_constrain(0,lp__);
##  273 :         else
##  274 :             jitter_sq = in__.scalar_lb_constrain(0);
##  275 :
##  276 :         T__ scaledf;
##  277 :         (void) scaledf;   // dummy to suppress unused var warning
##  278 :         if (jacobian__)
##  279 :             scaledf = in__.scalar_lb_constrain(0,lp__);
##  280 :         else
##  281 :             scaledf = in__.scalar_lb_constrain(0);
##  282 :
##  283 :         vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffectsKnots;
##  284 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  285 :         spatialEffectsKnots.reserve(dim_spatialEffectsKnots_0__);
##  286 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  287 :             if (jacobian__)
##  288 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots,lp__));
##  289 :             else
##  290 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
```

```
##  291 :         }
##  292 :
##  293 :
##  294 :         // transformed parameters
##  295 :         Eigen::Matrix<T__,Eigen::Dynamic,1>  muZeros(nKnots);
##  296 :         (void) muZeros;  // dummy to suppress unused var warning
##  297 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  298 :
##  299 :         // initialize transformed variables to avoid seg fault on val access
##  300 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  301 :
##  302 :         try {
##  303 :             current_statement_begin__ = 24;
##  304 :             for (int i = 1; i <= nKnots; ++i) {
##  305 :                 current_statement_begin__ = 25;
##  306 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  307 :             }
##  308 :         } catch (const std::exception& e) {
##  309 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  310 :             // Next line prevents compiler griping about no return
##  311 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  312 :         }
##  313 :
##  314 :         // validate transformed parameters
##  315 :         for (int i0__ = 0; i0__ < nKnots; ++i0__) {
##  316 :             if (stan::math::is_uninitialized(muZeros(i0__))) {
##  317 :                 std::stringstream msg__;
##  318 :                 msg__ << "Undefined transformed parameter: muZeros" << '[' << i0__ << ']';
##  319 :                 throw std::runtime_error(msg__.str());
##  320 :             }
##  321 :         }
##  322 :
##  323 :         const char* function__ = "validate transformed params";
##  324 :         (void) function__; // dummy to suppress unused var warning
##  325 :
##  326 :         // model body
##  327 :         try {
##  328 :             {
##  329 :                 T__ DF;
##  330 :                 (void) DF;  // dummy to suppress unused var warning
##  331 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots(nKnots,nKnots);
##  332 :                 (void) SigmaKnots;  // dummy to suppress unused var warning
##  333 :                 stan::math::fill(SigmaKnots,DUMMY_VAR__);
##  334 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots_chol(nKnots,nKno
##  335 :                 (void) SigmaKnots_chol;  // dummy to suppress unused var warning
##  336 :                 stan::math::fill(SigmaKnots_chol,DUMMY_VAR__);
##  337 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaOffDiag(nLocs,nKnots);
##  338 :                 (void) SigmaOffDiag;  // dummy to suppress unused var warning
##  339 :                 stan::math::fill(SigmaOffDiag,DUMMY_VAR__);
##  340 :                 vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffects(nT, (Eigen::Matri
##  341 :                 stan::math::fill(spatialEffects,DUMMY_VAR__);
##  342 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  invSigmaKnots(nKnots,nKnots
##  343 :                 (void) invSigmaKnots;  // dummy to suppress unused var warning
##  344 :                 stan::math::fill(invSigmaKnots,DUMMY_VAR__);
```

```
##  345 :                 stan::math::initialize(DF, DUMMY_VAR__);
##  346 :                 stan::math::initialize(SigmaKnots, DUMMY_VAR__);
##  347 :                 stan::math::initialize(SigmaKnots_chol, DUMMY_VAR__);
##  348 :                 stan::math::initialize(SigmaOffDiag, DUMMY_VAR__);
##  349 :                 stan::math::initialize(spatialEffects, DUMMY_VAR__);
##  350 :                 stan::math::initialize(invSigmaKnots, DUMMY_VAR__);
##  351 :                 current_statement_begin__ = 37;
##  352 :                 stan::math::assign(SigmaKnots, multiply(gp_sigmaSq,exp(multiply(-(gp_scale),di
##  353 :                 current_statement_begin__ = 38;
##  354 :                 stan::math::assign(SigmaOffDiag, multiply(gp_sigmaSq,exp(multiply(-(gp_scale)
##  355 :                 current_statement_begin__ = 39;
##  356 :                 for (int i = 1; i <= nKnots; ++i) {
##  357 :                     current_statement_begin__ = 40;
##  358 :                     stan::math::assign(get_base1_lhs(SigmaKnots,i,i,"SigmaKnots",1), (jitter_s
##  359 :                     current_statement_begin__ = 41;
##  360 :                     stan::math::assign(get_base1_lhs(SigmaOffDiag,i,i,"SigmaOffDiag",1), (jit
##  361 :                 }
##  362 :                 current_statement_begin__ = 43;
##  363 :                 stan::math::assign(invSigmaKnots, inverse(SigmaKnots));
##  364 :                 current_statement_begin__ = 47;
##  365 :                 stan::math::assign(SigmaKnots_chol, cholesky_decompose(SigmaKnots));
##  366 :                 current_statement_begin__ = 50;
##  367 :                 lp_accum__.add(multi_normal_cholesky_log<propto__>(get_base1(spatialEffectsKno
##  368 :                 current_statement_begin__ = 52;
##  369 :                 stan::math::assign(DF, 2);
##  370 :                 current_statement_begin__ = 53;
##  371 :                 lp_accum__.add(chi_square_log<propto__>(scaledf, DF));
##  372 :                 current_statement_begin__ = 54;
##  373 :                 stan::math::assign(get_base1_lhs(spatialEffects,1,"spatialEffects",1), multipl
##  374 :                 current_statement_begin__ = 57;
##  375 :                 lp_accum__.add(cauchy_log<propto__>(gp_scale, 0, 5));
##  376 :                 current_statement_begin__ = 58;
##  377 :                 lp_accum__.add(cauchy_log<propto__>(gp_sigmaSq, 0, 5));
##  378 :                 current_statement_begin__ = 59;
##  379 :                 lp_accum__.add(cauchy_log<propto__>(jitter_sq, 0, 5));
##  380 :                 current_statement_begin__ = 61;
##  381 :                 for (int n = 1; n <= N; ++n) {
##  382 :                     current_statement_begin__ = 62;
##  383 :                     lp_accum__.add(poisson_log<propto__>(get_base1(y,n,"y",1), exp(get_base1(g
##  384 :                 }
##  385 :             }
##  386 :         } catch (const std::exception& e) {
##  387 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  388 :             // Next line prevents compiler griping about no return
##  389 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  390 :         }
##  391 :
##  392 :         lp_accum__.add(lp__);
##  393 :         return lp_accum__.sum();
##  394 :
##  395 :     } // log_prob()
##  396 :
##  397 :     template <bool propto, bool jacobian, typename T_>
##  398 :     T_ log_prob(Eigen::Matrix<T_,Eigen::Dynamic,1>& params_r,
```

```
##  399 :                std::ostream* pstream = 0) const {
##  400 :       std::vector<T_> vec_params_r;
##  401 :       vec_params_r.reserve(params_r.size());
##  402 :       for (int i = 0; i < params_r.size(); ++i)
##  403 :         vec_params_r.push_back(params_r(i));
##  404 :       std::vector<int> vec_params_i;
##  405 :       return log_prob<propto,jacobian,T_>(vec_params_r, vec_params_i, pstream);
##  406 :     }
##  407 :
##  408 :
##  409 :     void get_param_names(std::vector<std::string>& names__) const {
##  410 :         names__.resize(0);
##  411 :         names__.push_back("gp_scale");
##  412 :         names__.push_back("gp_sigmaSq");
##  413 :         names__.push_back("jitter_sq");
##  414 :         names__.push_back("scaledf");
##  415 :         names__.push_back("spatialEffectsKnots");
##  416 :         names__.push_back("muZeros");
##  417 :     }
##  418 :
##  419 :
##  420 :     void get_dims(std::vector<std::vector<size_t> >& dimss__) const {
##  421 :         dimss__.resize(0);
##  422 :         std::vector<size_t> dims__;
##  423 :         dims__.resize(0);
##  424 :         dimss__.push_back(dims__);
##  425 :         dims__.resize(0);
##  426 :         dimss__.push_back(dims__);
##  427 :         dims__.resize(0);
##  428 :         dimss__.push_back(dims__);
##  429 :         dims__.resize(0);
##  430 :         dimss__.push_back(dims__);
##  431 :         dims__.resize(0);
##  432 :         dims__.push_back(nT);
##  433 :         dims__.push_back(nKnots);
##  434 :         dimss__.push_back(dims__);
##  435 :         dims__.resize(0);
##  436 :         dims__.push_back(nKnots);
##  437 :         dimss__.push_back(dims__);
##  438 :     }
##  439 :
##  440 :     template <typename RNG>
##  441 :     void write_array(RNG& base_rng__,
##  442 :                      std::vector<double>& params_r__,
##  443 :                      std::vector<int>& params_i__,
##  444 :                      std::vector<double>& vars__,
##  445 :                      bool include_tparams__ = true,
##  446 :                      bool include_gqs__ = true,
##  447 :                      std::ostream* pstream__ = 0) const {
##  448 :         vars__.resize(0);
##  449 :         stan::io::reader<double> in__(params_r__,params_i__);
##  450 :         static const char* function__ = "model53fb443225b6_mvtPoisson_namespace::write_array"
##  451 :         (void) function__; // dummy call to supress warning
##  452 :         // read-transform, write parameters
```

```
##  453 :         double gp_scale = in__.scalar_lb_constrain(0);
##  454 :         double gp_sigmaSq = in__.scalar_lb_constrain(0);
##  455 :         double jitter_sq = in__.scalar_lb_constrain(0);
##  456 :         double scaledf = in__.scalar_lb_constrain(0);
##  457 :         vector<vector_d> spatialEffectsKnots;
##  458 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  459 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  460 :             spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  461 :         }
##  462 :         vars__.push_back(gp_scale);
##  463 :         vars__.push_back(gp_sigmaSq);
##  464 :         vars__.push_back(jitter_sq);
##  465 :         vars__.push_back(scaledf);
##  466 :         for (int k_1__ = 0; k_1__ < nKnots; ++k_1__) {
##  467 :             for (int k_0__ = 0; k_0__ < nT; ++k_0__) {
##  468 :                 vars__.push_back(spatialEffectsKnots[k_0__][k_1__]);
##  469 :             }
##  470 :         }
##  471 :
##  472 :         if (!include_tparams__) return;
##  473 :         // declare and define transformed parameters
##  474 :         double lp__ = 0.0;
##  475 :         (void) lp__; // dummy call to supress warning
##  476 :         stan::math::accumulator<double> lp_accum__;
##  477 :
##  478 :         vector_d muZeros(nKnots);
##  479 :         (void) muZeros;  // dummy to suppress unused var warning
##  480 :
##  481 :         try {
##  482 :             current_statement_begin__ = 24;
##  483 :             for (int i = 1; i <= nKnots; ++i) {
##  484 :                 current_statement_begin__ = 25;
##  485 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  486 :             }
##  487 :         } catch (const std::exception& e) {
##  488 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  489 :             // Next line prevents compiler griping about no return
##  490 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  491 :         }
##  492 :
##  493 :         // validate transformed parameters
##  494 :
##  495 :         // write transformed parameters
##  496 :         for (int k_0__ = 0; k_0__ < nKnots; ++k_0__) {
##  497 :             vars__.push_back(muZeros[k_0__]);
##  498 :         }
##  499 :
##  500 :         if (!include_gqs__) return;
##  501 :         // declare and define generated quantities
##  502 :
##  503 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  504 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  505 :
##  506 :
```

```
##  507 :         // initialize transformed variables to avoid seg fault on val access
##  508 :
##  509 :         try {
##  510 :         } catch (const std::exception& e) {
##  511 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  512 :             // Next line prevents compiler griping about no return
##  513 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  514 :         }
##  515 :
##  516 :         // validate generated quantities
##  517 :
##  518 :         // write generated quantities
##  519 :     }
##  520 :
##  521 :     template <typename RNG>
##  522 :     void write_array(RNG& base_rng,
##  523 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  524 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& vars,
##  525 :                      bool include_tparams = true,
##  526 :                      bool include_gqs = true,
##  527 :                      std::ostream* pstream = 0) const {
##  528 :       std::vector<double> params_r_vec(params_r.size());
##  529 :       for (int i = 0; i < params_r.size(); ++i)
##  530 :         params_r_vec[i] = params_r(i);
##  531 :       std::vector<double> vars_vec;
##  532 :       std::vector<int> params_i_vec;
##  533 :       write_array(base_rng,params_r_vec,params_i_vec,vars_vec,include_tparams,include_gqs,pst
##  534 :       vars.resize(vars_vec.size());
##  535 :       for (int i = 0; i < vars.size(); ++i)
##  536 :         vars(i) = vars_vec[i];
##  537 :     }
##  538 :
##  539 :     static std::string model_name() {
##  540 :         return "model53fb443225b6_mvtPoisson";
##  541 :     }
##  542 :
##  543 :
##  544 :     void constrained_param_names(std::vector<std::string>& param_names__,
##  545 :                                  bool include_tparams__ = true,
##  546 :                                  bool include_gqs__ = true) const {
##  547 :         std::stringstream param_name_stream__;
##  548 :         param_name_stream__.str(std::string());
##  549 :         param_name_stream__ << "gp_scale";
##  550 :         param_names__.push_back(param_name_stream__.str());
##  551 :         param_name_stream__.str(std::string());
##  552 :         param_name_stream__ << "gp_sigmaSq";
##  553 :         param_names__.push_back(param_name_stream__.str());
##  554 :         param_name_stream__.str(std::string());
##  555 :         param_name_stream__ << "jitter_sq";
##  556 :         param_names__.push_back(param_name_stream__.str());
##  557 :         param_name_stream__.str(std::string());
##  558 :         param_name_stream__ << "scaledf";
##  559 :         param_names__.push_back(param_name_stream__.str());
##  560 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
```

```
##  561 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  562 :                 param_name_stream__.str(std::string());
##  563 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  564 :                 param_names__.push_back(param_name_stream__.str());
##  565 :             }
##  566 :         }
##  567 :
##  568 :         if (!include_gqs__ && !include_tparams__) return;
##  569 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  570 :             param_name_stream__.str(std::string());
##  571 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  572 :             param_names__.push_back(param_name_stream__.str());
##  573 :         }
##  574 :
##  575 :         if (!include_gqs__) return;
##  576 :     }
##  577 :
##  578 :
##  579 :     void unconstrained_param_names(std::vector<std::string>& param_names__,
##  580 :                                    bool include_tparams__ = true,
##  581 :                                    bool include_gqs__ = true) const {
##  582 :         std::stringstream param_name_stream__;
##  583 :         param_name_stream__.str(std::string());
##  584 :         param_name_stream__ << "gp_scale";
##  585 :         param_names__.push_back(param_name_stream__.str());
##  586 :         param_name_stream__.str(std::string());
##  587 :         param_name_stream__ << "gp_sigmaSq";
##  588 :         param_names__.push_back(param_name_stream__.str());
##  589 :         param_name_stream__.str(std::string());
##  590 :         param_name_stream__ << "jitter_sq";
##  591 :         param_names__.push_back(param_name_stream__.str());
##  592 :         param_name_stream__.str(std::string());
##  593 :         param_name_stream__ << "scaledf";
##  594 :         param_names__.push_back(param_name_stream__.str());
##  595 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  596 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  597 :                 param_name_stream__.str(std::string());
##  598 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  599 :                 param_names__.push_back(param_name_stream__.str());
##  600 :             }
##  601 :         }
##  602 :
##  603 :         if (!include_gqs__ && !include_tparams__) return;
##  604 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  605 :             param_name_stream__.str(std::string());
##  606 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  607 :             param_names__.push_back(param_name_stream__.str());
##  608 :         }
##  609 :
##  610 :         if (!include_gqs__) return;
##  611 :     }
##  612 :
##  613 : }; // model
##  614 :
```

```
##  615 : } // namespace
##  616 :
##  617 : typedef model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPoisson stan_model;
##  618 :
##  619 : #include <rstan/rstaninc.hpp>
##  620 : /**
##  621 :  * Define Rcpp Module to expose stan_fit's functions to R.
##  622 :  */
##  623 : RCPP_MODULE(stan_fit4model53fb443225b6_mvtPoisson_mod){
##  624 :   Rcpp::class_<rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtP
##  625 :                boost::random::ecuyer1988> >("stan_fit4model53fb443225b6_mvtPoisson")
##  626 :     // .constructor<Rcpp::List>()
##  627 :     .constructor<SEXP, SEXP>()
##  628 :     // .constructor<SEXP, SEXP>()
##  629 :     .method("call_sampler",
##  630 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  631 :     .method("param_names",
##  632 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  633 :     .method("param_names_oi",
##  634 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  635 :     .method("param_fnames_oi",
##  636 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  637 :     .method("param_dims",
##  638 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  639 :     .method("param_dims_oi",
##  640 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  641 :     .method("update_param_oi",
##  642 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  643 :     .method("param_oi_tidx",
##  644 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  645 :     .method("grad_log_prob",
##  646 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  647 :     .method("log_prob",
##  648 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  649 :     .method("unconstrain_pars",
##  650 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  651 :     .method("constrain_pars",
##  652 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  653 :     .method("num_pars_unconstrained",
##  654 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  655 :     .method("unconstrained_param_names",
##  656 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  657 :     .method("constrained_param_names",
##  658 :             &rstan::stan_fit<model53fb443225b6_mvtPoisson_namespace::model53fb443225b6_mvtPois
##  659 :     ;
##  660 : }
##  661 :
##  662 : // declarations
##  663 : extern "C" {
##  664 : SEXP file53fb3849f2b4( ) ;
##  665 : }
##  666 :
##  667 : // definition
##  668 :
```

```
##  669 : SEXP file53fb3849f2b4(  ){
##  670 :  return Rcpp::wrap("mvtPoisson");
##  671 : }
##  672 :
##  673 :
## Compilation argument:
##  /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB file53fb3849f2b4.cpp 2> file53fb3849f2b4.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'mvtPoisson' NOW.
##
## COMPILING MODEL 'mvtPoisson' NOW.
##
## STARTING SAMPLER FOR MODEL 'mvtPoisson' NOW.
```

Comparing MVN and MVT random effects with Gamma model (STAN)

```
# Named list of data
spatglm_data = list(nKnots = nKnots, nLocs = nLocs, N = nLocs, nT = 1, y = y.gamma,
    location = seq(1, nLocs), distKnotsSq = distKnotsSq, distKnots21Sq = distKnots21Sq)
# parameters to monitor
spatglm_pars = c("gp_scale", "spatialEffectsKnots", "gp_sigmaSq", "jitter_sq")

# fit the model
stanMod_gaussianGamma = stan(file = "gaussianGamma.stan", data = spatglm_data,
    verbose = TRUE, chains = 4, thin = 1, warmup = 500, iter = 1000, pars = spatglm_pars)
```

```
##
## TRANSLATING MODEL 'gaussianGamma' FROM Stan CODE TO C++ CODE NOW.
## successful in parsing the Stan model 'gaussianGamma'.
## OS: x86_64, darwin13.4.0; rstan: 2.9.0.3; Rcpp: 0.12.5; inline: 0.3.14
##  >> setting environment variables:
## PKG_LIBS =
## PKG_CPPFLAGS =   -isystem"/Users/eric.ward/Library/R/3.2/library/Rcpp/include/"  -isystem"/Users/eri
##  >> Program source :
##
##    1 :
##    2 : // includes from the plugin
##    3 :
##    4 :
##    5 : // user includes
##    6 : #define STAN__SERVICES__COMMAND_HPP// Code generated by Stan version 2.9
##    7 :
##    8 : #include <stan/model/model_header.hpp>
##    9 :
##   10 : namespace model53fb6879b757_gaussianGamma_namespace {
##   11 :
##   12 : using std::istream;
##   13 : using std::string;
##   14 : using std::stringstream;
##   15 : using std::vector;
##   16 : using stan::io::dump;
##   17 : using stan::math::lgamma;
```

```
##   18 : using stan::model::prob_grad;
##   19 : using namespace stan::math;
##   20 :
##   21 : typedef Eigen::Matrix<double,Eigen::Dynamic,1> vector_d;
##   22 : typedef Eigen::Matrix<double,1,Eigen::Dynamic> row_vector_d;
##   23 : typedef Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic> matrix_d;
##   24 :
##   25 : static int current_statement_begin__;
##   26 : class model53fb6879b757_gaussianGamma : public prob_grad {
##   27 : private:
##   28 :     int nKnots;
##   29 :     int nLocs;
##   30 :     int N;
##   31 :     int nT;
##   32 :     vector<double> y;
##   33 :     vector<int> location;
##   34 :     matrix_d distKnotsSq;
##   35 :     matrix_d distKnots21Sq;
##   36 : public:
##   37 :     model53fb6879b757_gaussianGamma(stan::io::var_context& context__,
##   38 :         std::ostream* pstream__ = 0)
##   39 :         : prob_grad(0) {
##   40 :         current_statement_begin__ = -1;
##   41 :
##   42 :         static const char* function__ = "model53fb6879b757_gaussianGamma_namespace::model53fb
##   43 :         (void) function__; // dummy call to supress warning
##   44 :         size_t pos__;
##   45 :         (void) pos__; // dummy call to supress warning
##   46 :         std::vector<int> vals_i__;
##   47 :         std::vector<double> vals_r__;
##   48 :         context__.validate_dims("data initialization", "nKnots", "int", context__.to_vec());
##   49 :         nKnots = int(0);
##   50 :         vals_i__ = context__.vals_i("nKnots");
##   51 :         pos__ = 0;
##   52 :         nKnots = vals_i__[pos__++];
##   53 :         context__.validate_dims("data initialization", "nLocs", "int", context__.to_vec());
##   54 :         nLocs = int(0);
##   55 :         vals_i__ = context__.vals_i("nLocs");
##   56 :         pos__ = 0;
##   57 :         nLocs = vals_i__[pos__++];
##   58 :         context__.validate_dims("data initialization", "N", "int", context__.to_vec());
##   59 :         N = int(0);
##   60 :         vals_i__ = context__.vals_i("N");
##   61 :         pos__ = 0;
##   62 :         N = vals_i__[pos__++];
##   63 :         context__.validate_dims("data initialization", "nT", "int", context__.to_vec());
##   64 :         nT = int(0);
##   65 :         vals_i__ = context__.vals_i("nT");
##   66 :         pos__ = 0;
##   67 :         nT = vals_i__[pos__++];
##   68 :         context__.validate_dims("data initialization", "y", "double", context__.to_vec(N));
##   69 :         validate_non_negative_index("y", "N", N);
##   70 :         y = std::vector<double>(N,double(0));
##   71 :         vals_r__ = context__.vals_r("y");
```

```
##   72 :         pos__ = 0;
##   73 :         size_t y_limit_0__ = N;
##   74 :         for (size_t i_0__ = 0; i_0__ < y_limit_0__; ++i_0__) {
##   75 :             y[i_0__] = vals_r__[pos__++];
##   76 :         }
##   77 :         context__.validate_dims("data initialization", "location", "int", context__.to_vec(N)
##   78 :         validate_non_negative_index("location", "N", N);
##   79 :         location = std::vector<int>(N,int(0));
##   80 :         vals_i__ = context__.vals_i("location");
##   81 :         pos__ = 0;
##   82 :         size_t location_limit_0__ = N;
##   83 :         for (size_t i_0__ = 0; i_0__ < location_limit_0__; ++i_0__) {
##   84 :             location[i_0__] = vals_i__[pos__++];
##   85 :         }
##   86 :         context__.validate_dims("data initialization", "distKnotsSq", "matrix_d", context__.t
##   87 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   88 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   89 :         distKnotsSq = matrix_d(nKnots,nKnots);
##   90 :         vals_r__ = context__.vals_r("distKnotsSq");
##   91 :         pos__ = 0;
##   92 :         size_t distKnotsSq_m_mat_lim__ = nKnots;
##   93 :         size_t distKnotsSq_n_mat_lim__ = nKnots;
##   94 :         for (size_t n_mat__ = 0; n_mat__ < distKnotsSq_n_mat_lim__; ++n_mat__) {
##   95 :             for (size_t m_mat__ = 0; m_mat__ < distKnotsSq_m_mat_lim__; ++m_mat__) {
##   96 :                 distKnotsSq(m_mat__,n_mat__) = vals_r__[pos__++];
##   97 :             }
##   98 :         }
##   99 :         context__.validate_dims("data initialization", "distKnots21Sq", "matrix_d", context__
##  100 :         validate_non_negative_index("distKnots21Sq", "nLocs", nLocs);
##  101 :         validate_non_negative_index("distKnots21Sq", "nKnots", nKnots);
##  102 :         distKnots21Sq = matrix_d(nLocs,nKnots);
##  103 :         vals_r__ = context__.vals_r("distKnots21Sq");
##  104 :         pos__ = 0;
##  105 :         size_t distKnots21Sq_m_mat_lim__ = nLocs;
##  106 :         size_t distKnots21Sq_n_mat_lim__ = nKnots;
##  107 :         for (size_t n_mat__ = 0; n_mat__ < distKnots21Sq_n_mat_lim__; ++n_mat__) {
##  108 :             for (size_t m_mat__ = 0; m_mat__ < distKnots21Sq_m_mat_lim__; ++m_mat__) {
##  109 :                 distKnots21Sq(m_mat__,n_mat__) = vals_r__[pos__++];
##  110 :             }
##  111 :         }
##  112 :
##  113 :         // validate data
##  114 :         check_greater_or_equal(function__,"nKnots",nKnots,1);
##  115 :         check_greater_or_equal(function__,"nLocs",nLocs,1);
##  116 :         check_greater_or_equal(function__,"N",N,1);
##  117 :         check_greater_or_equal(function__,"nT",nT,1);
##  118 :
##  119 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  120 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  121 :
##  122 :
##  123 :         // initialize transformed variables to avoid seg fault on val access
##  124 :
##  125 :         try {
```

```
##  126 :         } catch (const std::exception& e) {
##  127 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  128 :             // Next line prevents compiler griping about no return
##  129 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  130 :         }
##  131 :
##  132 :         // validate transformed data
##  133 :
##  134 :         // set parameter ranges
##  135 :         num_params_r__ = 0U;
##  136 :         param_ranges_i__.clear();
##  137 :         ++num_params_r__;
##  138 :         ++num_params_r__;
##  139 :         ++num_params_r__;
##  140 :         ++num_params_r__;
##  141 :         num_params_r__ += nKnots * nT;
##  142 :     }
##  143 :
##  144 :     ~model53fb6879b757_gaussianGamma() { }
##  145 :
##  146 :
##  147 :     void transform_inits(const stan::io::var_context& context__,
##  148 :                          std::vector<int>& params_i__,
##  149 :                          std::vector<double>& params_r__,
##  150 :                          std::ostream* pstream__) const {
##  151 :         stan::io::writer<double> writer__(params_r__,params_i__);
##  152 :         size_t pos__;
##  153 :         (void) pos__; // dummy call to supress warning
##  154 :         std::vector<double> vals_r__;
##  155 :         std::vector<int> vals_i__;
##  156 :
##  157 :         if (!(context__.contains_r("gp_scale")))
##  158 :             throw std::runtime_error("variable gp_scale missing");
##  159 :         vals_r__ = context__.vals_r("gp_scale");
##  160 :         pos__ = 0U;
##  161 :         context__.validate_dims("initialization", "gp_scale", "double", context__.to_vec());
##  162 :         double gp_scale(0);
##  163 :         gp_scale = vals_r__[pos__++];
##  164 :         try {
##  165 :             writer__.scalar_lb_unconstrain(0,gp_scale);
##  166 :         } catch (const std::exception& e) {
##  167 :             throw std::runtime_error(std::string("Error transforming variable gp_scale: ") + e
##  168 :         }
##  169 :
##  170 :         if (!(context__.contains_r("gp_sigmaSq")))
##  171 :             throw std::runtime_error("variable gp_sigmaSq missing");
##  172 :         vals_r__ = context__.vals_r("gp_sigmaSq");
##  173 :         pos__ = 0U;
##  174 :         context__.validate_dims("initialization", "gp_sigmaSq", "double", context__.to_vec())
##  175 :         double gp_sigmaSq(0);
##  176 :         gp_sigmaSq = vals_r__[pos__++];
##  177 :         try {
##  178 :             writer__.scalar_lb_unconstrain(0,gp_sigmaSq);
##  179 :         } catch (const std::exception& e) {
```

```
##  180 :             throw std::runtime_error(std::string("Error transforming variable gp_sigmaSq: ") +
##  181 :         }
##  182 :
##  183 :         if (!(context__.contains_r("jitter_sq")))
##  184 :             throw std::runtime_error("variable jitter_sq missing");
##  185 :         vals_r__ = context__.vals_r("jitter_sq");
##  186 :         pos__ = 0U;
##  187 :         context__.validate_dims("initialization", "jitter_sq", "double", context__.to_vec());
##  188 :         double jitter_sq(0);
##  189 :         jitter_sq = vals_r__[pos__++];
##  190 :         try {
##  191 :             writer__.scalar_lb_unconstrain(0,jitter_sq);
##  192 :         } catch (const std::exception& e) {
##  193 :             throw std::runtime_error(std::string("Error transforming variable jitter_sq: ") +
##  194 :         }
##  195 :
##  196 :         if (!(context__.contains_r("gammaA")))
##  197 :             throw std::runtime_error("variable gammaA missing");
##  198 :         vals_r__ = context__.vals_r("gammaA");
##  199 :         pos__ = 0U;
##  200 :         context__.validate_dims("initialization", "gammaA", "double", context__.to_vec());
##  201 :         double gammaA(0);
##  202 :         gammaA = vals_r__[pos__++];
##  203 :         try {
##  204 :             writer__.scalar_lb_unconstrain(0,gammaA);
##  205 :         } catch (const std::exception& e) {
##  206 :             throw std::runtime_error(std::string("Error transforming variable gammaA: ") + e.w
##  207 :         }
##  208 :
##  209 :         if (!(context__.contains_r("spatialEffectsKnots")))
##  210 :             throw std::runtime_error("variable spatialEffectsKnots missing");
##  211 :         vals_r__ = context__.vals_r("spatialEffectsKnots");
##  212 :         pos__ = 0U;
##  213 :         context__.validate_dims("initialization", "spatialEffectsKnots", "vector_d", context_
##  214 :         std::vector<vector_d> spatialEffectsKnots(nT,vector_d(nKnots));
##  215 :         for (int j1__ = 0U; j1__ < nKnots; ++j1__)
##  216 :             for (int i0__ = 0U; i0__ < nT; ++i0__)
##  217 :                 spatialEffectsKnots[i0__](j1__) = vals_r__[pos__++];
##  218 :         for (int i0__ = 0U; i0__ < nT; ++i0__)
##  219 :             try {
##  220 :             writer__.vector_unconstrain(spatialEffectsKnots[i0__]);
##  221 :         } catch (const std::exception& e) {
##  222 :             throw std::runtime_error(std::string("Error transforming variable spatialEffectsKn
##  223 :         }
##  224 :
##  225 :         params_r__ = writer__.data_r();
##  226 :         params_i__ = writer__.data_i();
##  227 :     }
##  228 :
##  229 :     void transform_inits(const stan::io::var_context& context,
##  230 :                          Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  231 :                          std::ostream* pstream__) const {
##  232 :       std::vector<double> params_r_vec;
##  233 :       std::vector<int> params_i_vec;
```

```
##  234 :       transform_inits(context, params_i_vec, params_r_vec, pstream__);
##  235 :       params_r.resize(params_r_vec.size());
##  236 :       for (int i = 0; i < params_r.size(); ++i)
##  237 :         params_r(i) = params_r_vec[i];
##  238 :     }
##  239 :
##  240 :
##  241 :     template <bool propto__, bool jacobian__, typename T__>
##  242 :     T__ log_prob(vector<T__>& params_r__,
##  243 :                  vector<int>& params_i__,
##  244 :                  std::ostream* pstream__ = 0) const {
##  245 :
##  246 :         T__ DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  247 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  248 :
##  249 :         T__ lp__(0.0);
##  250 :         stan::math::accumulator<T__> lp_accum__;
##  251 :
##  252 :         // model parameters
##  253 :         stan::io::reader<T__> in__(params_r__,params_i__);
##  254 :
##  255 :         T__ gp_scale;
##  256 :         (void) gp_scale;   // dummy to suppress unused var warning
##  257 :         if (jacobian__)
##  258 :             gp_scale = in__.scalar_lb_constrain(0,lp__);
##  259 :         else
##  260 :             gp_scale = in__.scalar_lb_constrain(0);
##  261 :
##  262 :         T__ gp_sigmaSq;
##  263 :         (void) gp_sigmaSq;   // dummy to suppress unused var warning
##  264 :         if (jacobian__)
##  265 :             gp_sigmaSq = in__.scalar_lb_constrain(0,lp__);
##  266 :         else
##  267 :             gp_sigmaSq = in__.scalar_lb_constrain(0);
##  268 :
##  269 :         T__ jitter_sq;
##  270 :         (void) jitter_sq;   // dummy to suppress unused var warning
##  271 :         if (jacobian__)
##  272 :             jitter_sq = in__.scalar_lb_constrain(0,lp__);
##  273 :         else
##  274 :             jitter_sq = in__.scalar_lb_constrain(0);
##  275 :
##  276 :         T__ gammaA;
##  277 :         (void) gammaA;   // dummy to suppress unused var warning
##  278 :         if (jacobian__)
##  279 :             gammaA = in__.scalar_lb_constrain(0,lp__);
##  280 :         else
##  281 :             gammaA = in__.scalar_lb_constrain(0);
##  282 :
##  283 :         vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffectsKnots;
##  284 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  285 :         spatialEffectsKnots.reserve(dim_spatialEffectsKnots_0__);
##  286 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  287 :             if (jacobian__)
```

```
##  288 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots,lp__));
##  289 :             else
##  290 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  291 :         }
##  292 :
##  293 :
##  294 :         // transformed parameters
##  295 :         Eigen::Matrix<T__,Eigen::Dynamic,1>  muZeros(nKnots);
##  296 :         (void) muZeros;  // dummy to suppress unused var warning
##  297 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  298 :
##  299 :         // initialize transformed variables to avoid seg fault on val access
##  300 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  301 :
##  302 :         try {
##  303 :             current_statement_begin__ = 24;
##  304 :             for (int i = 1; i <= nKnots; ++i) {
##  305 :                 current_statement_begin__ = 25;
##  306 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  307 :             }
##  308 :         } catch (const std::exception& e) {
##  309 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  310 :             // Next line prevents compiler griping about no return
##  311 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  312 :         }
##  313 :
##  314 :         // validate transformed parameters
##  315 :         for (int i0__ = 0; i0__ < nKnots; ++i0__) {
##  316 :             if (stan::math::is_uninitialized(muZeros(i0__))) {
##  317 :                 std::stringstream msg__;
##  318 :                 msg__ << "Undefined transformed parameter: muZeros" << '[' << i0__ << ']';
##  319 :                 throw std::runtime_error(msg__.str());
##  320 :             }
##  321 :         }
##  322 :
##  323 :         const char* function__ = "validate transformed params";
##  324 :         (void) function__; // dummy to suppress unused var warning
##  325 :
##  326 :         // model body
##  327 :         try {
##  328 :             {
##  329 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots(nKnots,nKnots);
##  330 :                 (void) SigmaKnots;  // dummy to suppress unused var warning
##  331 :                 stan::math::fill(SigmaKnots,DUMMY_VAR__);
##  332 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots_chol(nKnots,nKno
##  333 :                 (void) SigmaKnots_chol;  // dummy to suppress unused var warning
##  334 :                 stan::math::fill(SigmaKnots_chol,DUMMY_VAR__);
##  335 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaOffDiag(nLocs,nKnots);
##  336 :                 (void) SigmaOffDiag;  // dummy to suppress unused var warning
##  337 :                 stan::math::fill(SigmaOffDiag,DUMMY_VAR__);
##  338 :                 vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffects(nT, (Eigen::Matri
##  339 :                 stan::math::fill(spatialEffects,DUMMY_VAR__);
##  340 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  invSigmaKnots(nKnots,nKnots
##  341 :                 (void) invSigmaKnots;  // dummy to suppress unused var warning
```

```
##  342 :                 stan::math::fill(invSigmaKnots,DUMMY_VAR__);
##  343 :                 stan::math::initialize(SigmaKnots, DUMMY_VAR__);
##  344 :                 stan::math::initialize(SigmaKnots_chol, DUMMY_VAR__);
##  345 :                 stan::math::initialize(SigmaOffDiag, DUMMY_VAR__);
##  346 :                 stan::math::initialize(spatialEffects, DUMMY_VAR__);
##  347 :                 stan::math::initialize(invSigmaKnots, DUMMY_VAR__);
##  348 :                 current_statement_begin__ = 36;
##  349 :                 stan::math::assign(SigmaKnots, multiply(gp_sigmaSq,exp(multiply(-(gp_scale),di
##  350 :                 current_statement_begin__ = 37;
##  351 :                 stan::math::assign(SigmaOffDiag, multiply(gp_sigmaSq,exp(multiply(-(gp_scale)
##  352 :                 current_statement_begin__ = 38;
##  353 :                 for (int i = 1; i <= nKnots; ++i) {
##  354 :                     current_statement_begin__ = 39;
##  355 :                     stan::math::assign(get_base1_lhs(SigmaKnots,i,i,"SigmaKnots",1), (jitter_s
##  356 :                     current_statement_begin__ = 40;
##  357 :                     stan::math::assign(get_base1_lhs(SigmaOffDiag,i,i,"SigmaOffDiag",1), (jit
##  358 :                 }
##  359 :                 current_statement_begin__ = 42;
##  360 :                 stan::math::assign(invSigmaKnots, inverse(SigmaKnots));
##  361 :                 current_statement_begin__ = 46;
##  362 :                 stan::math::assign(SigmaKnots_chol, cholesky_decompose(SigmaKnots));
##  363 :                 current_statement_begin__ = 49;
##  364 :                 lp_accum__.add(multi_normal_cholesky_log<propto__>(get_base1(spatialEffectsKno
##  365 :                 current_statement_begin__ = 51;
##  366 :                 stan::math::assign(get_base1_lhs(spatialEffects,1,"spatialEffects",1), multipl
##  367 :                 current_statement_begin__ = 54;
##  368 :                 lp_accum__.add(cauchy_log<propto__>(gp_scale, 0, 5));
##  369 :                 current_statement_begin__ = 55;
##  370 :                 lp_accum__.add(cauchy_log<propto__>(gp_sigmaSq, 0, 5));
##  371 :                 current_statement_begin__ = 56;
##  372 :                 lp_accum__.add(cauchy_log<propto__>(jitter_sq, 0, 5));
##  373 :                 current_statement_begin__ = 57;
##  374 :                 lp_accum__.add(cauchy_log<propto__>(gammaA, 0, 5));
##  375 :                 current_statement_begin__ = 59;
##  376 :                 for (int n = 1; n <= N; ++n) {
##  377 :                     current_statement_begin__ = 60;
##  378 :                     lp_accum__.add(gamma_log<propto__>(get_base1(y,n,"y",1), gammaA, (gammaA /
##  379 :                 }
##  380 :             }
##  381 :         } catch (const std::exception& e) {
##  382 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  383 :             // Next line prevents compiler griping about no return
##  384 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  385 :         }
##  386 :
##  387 :         lp_accum__.add(lp__);
##  388 :         return lp_accum__.sum();
##  389 :
##  390 :     } // log_prob()
##  391 :
##  392 :     template <bool propto, bool jacobian, typename T_>
##  393 :     T_ log_prob(Eigen::Matrix<T_,Eigen::Dynamic,1>& params_r,
##  394 :                 std::ostream* pstream = 0) const {
##  395 :       std::vector<T_> vec_params_r;
```

```
##  396 :       vec_params_r.reserve(params_r.size());
##  397 :       for (int i = 0; i < params_r.size(); ++i)
##  398 :         vec_params_r.push_back(params_r(i));
##  399 :       std::vector<int> vec_params_i;
##  400 :       return log_prob<propto,jacobian,T_>(vec_params_r, vec_params_i, pstream);
##  401 :     }
##  402 :
##  403 :
##  404 :     void get_param_names(std::vector<std::string>& names__) const {
##  405 :         names__.resize(0);
##  406 :         names__.push_back("gp_scale");
##  407 :         names__.push_back("gp_sigmaSq");
##  408 :         names__.push_back("jitter_sq");
##  409 :         names__.push_back("gammaA");
##  410 :         names__.push_back("spatialEffectsKnots");
##  411 :         names__.push_back("muZeros");
##  412 :     }
##  413 :
##  414 :
##  415 :     void get_dims(std::vector<std::vector<size_t> >& dimss__) const {
##  416 :         dimss__.resize(0);
##  417 :         std::vector<size_t> dims__;
##  418 :         dims__.resize(0);
##  419 :         dimss__.push_back(dims__);
##  420 :         dims__.resize(0);
##  421 :         dimss__.push_back(dims__);
##  422 :         dims__.resize(0);
##  423 :         dimss__.push_back(dims__);
##  424 :         dims__.resize(0);
##  425 :         dimss__.push_back(dims__);
##  426 :         dims__.resize(0);
##  427 :         dims__.push_back(nT);
##  428 :         dims__.push_back(nKnots);
##  429 :         dimss__.push_back(dims__);
##  430 :         dims__.resize(0);
##  431 :         dims__.push_back(nKnots);
##  432 :         dimss__.push_back(dims__);
##  433 :     }
##  434 :
##  435 :     template <typename RNG>
##  436 :     void write_array(RNG& base_rng__,
##  437 :                      std::vector<double>& params_r__,
##  438 :                      std::vector<int>& params_i__,
##  439 :                      std::vector<double>& vars__,
##  440 :                      bool include_tparams__ = true,
##  441 :                      bool include_gqs__ = true,
##  442 :                      std::ostream* pstream__ = 0) const {
##  443 :         vars__.resize(0);
##  444 :         stan::io::reader<double> in__(params_r__,params_i__);
##  445 :         static const char* function__ = "model53fb6879b757_gaussianGamma_namespace::write_arra
##  446 :         (void) function__; // dummy call to supress warning
##  447 :         // read-transform, write parameters
##  448 :         double gp_scale = in__.scalar_lb_constrain(0);
##  449 :         double gp_sigmaSq = in__.scalar_lb_constrain(0);
```

```
##  450 :         double jitter_sq = in__.scalar_lb_constrain(0);
##  451 :         double gammaA = in__.scalar_lb_constrain(0);
##  452 :         vector<vector_d> spatialEffectsKnots;
##  453 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  454 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  455 :             spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  456 :         }
##  457 :         vars__.push_back(gp_scale);
##  458 :         vars__.push_back(gp_sigmaSq);
##  459 :         vars__.push_back(jitter_sq);
##  460 :         vars__.push_back(gammaA);
##  461 :         for (int k_1__ = 0; k_1__ < nKnots; ++k_1__) {
##  462 :             for (int k_0__ = 0; k_0__ < nT; ++k_0__) {
##  463 :                 vars__.push_back(spatialEffectsKnots[k_0__][k_1__]);
##  464 :             }
##  465 :         }
##  466 :
##  467 :         if (!include_tparams__) return;
##  468 :         // declare and define transformed parameters
##  469 :         double lp__ = 0.0;
##  470 :         (void) lp__; // dummy call to supress warning
##  471 :         stan::math::accumulator<double> lp_accum__;
##  472 :
##  473 :         vector_d muZeros(nKnots);
##  474 :         (void) muZeros;  // dummy to suppress unused var warning
##  475 :
##  476 :         try {
##  477 :             current_statement_begin__ = 24;
##  478 :             for (int i = 1; i <= nKnots; ++i) {
##  479 :                 current_statement_begin__ = 25;
##  480 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  481 :             }
##  482 :         } catch (const std::exception& e) {
##  483 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  484 :             // Next line prevents compiler griping about no return
##  485 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  486 :         }
##  487 :
##  488 :         // validate transformed parameters
##  489 :
##  490 :         // write transformed parameters
##  491 :         for (int k_0__ = 0; k_0__ < nKnots; ++k_0__) {
##  492 :             vars__.push_back(muZeros[k_0__]);
##  493 :         }
##  494 :
##  495 :         if (!include_gqs__) return;
##  496 :         // declare and define generated quantities
##  497 :
##  498 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  499 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  500 :
##  501 :
##  502 :         // initialize transformed variables to avoid seg fault on val access
##  503 :
```

```
##  504 :         try {
##  505 :         } catch (const std::exception& e) {
##  506 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  507 :             // Next line prevents compiler griping about no return
##  508 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  509 :         }
##  510 :
##  511 :         // validate generated quantities
##  512 :
##  513 :         // write generated quantities
##  514 :     }
##  515 :
##  516 :     template <typename RNG>
##  517 :     void write_array(RNG& base_rng,
##  518 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  519 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& vars,
##  520 :                      bool include_tparams = true,
##  521 :                      bool include_gqs = true,
##  522 :                      std::ostream* pstream = 0) const {
##  523 :       std::vector<double> params_r_vec(params_r.size());
##  524 :       for (int i = 0; i < params_r.size(); ++i)
##  525 :         params_r_vec[i] = params_r(i);
##  526 :       std::vector<double> vars_vec;
##  527 :       std::vector<int> params_i_vec;
##  528 :       write_array(base_rng,params_r_vec,params_i_vec,vars_vec,include_tparams,include_gqs,pst
##  529 :       vars.resize(vars_vec.size());
##  530 :       for (int i = 0; i < vars.size(); ++i)
##  531 :         vars(i) = vars_vec[i];
##  532 :     }
##  533 :
##  534 :     static std::string model_name() {
##  535 :         return "model53fb6879b757_gaussianGamma";
##  536 :     }
##  537 :
##  538 :
##  539 :     void constrained_param_names(std::vector<std::string>& param_names__,
##  540 :                                  bool include_tparams__ = true,
##  541 :                                  bool include_gqs__ = true) const {
##  542 :         std::stringstream param_name_stream__;
##  543 :         param_name_stream__.str(std::string());
##  544 :         param_name_stream__ << "gp_scale";
##  545 :         param_names__.push_back(param_name_stream__.str());
##  546 :         param_name_stream__.str(std::string());
##  547 :         param_name_stream__ << "gp_sigmaSq";
##  548 :         param_names__.push_back(param_name_stream__.str());
##  549 :         param_name_stream__.str(std::string());
##  550 :         param_name_stream__ << "jitter_sq";
##  551 :         param_names__.push_back(param_name_stream__.str());
##  552 :         param_name_stream__.str(std::string());
##  553 :         param_name_stream__ << "gammaA";
##  554 :         param_names__.push_back(param_name_stream__.str());
##  555 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  556 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  557 :                 param_name_stream__.str(std::string());
```

```
##  558 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  559 :                 param_names__.push_back(param_name_stream__.str());
##  560 :             }
##  561 :         }
##  562 :
##  563 :         if (!include_gqs__ && !include_tparams__) return;
##  564 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  565 :             param_name_stream__.str(std::string());
##  566 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  567 :             param_names__.push_back(param_name_stream__.str());
##  568 :         }
##  569 :
##  570 :         if (!include_gqs__) return;
##  571 :     }
##  572 :
##  573 :
##  574 :     void unconstrained_param_names(std::vector<std::string>& param_names__,
##  575 :                                    bool include_tparams__ = true,
##  576 :                                    bool include_gqs__ = true) const {
##  577 :         std::stringstream param_name_stream__;
##  578 :         param_name_stream__.str(std::string());
##  579 :         param_name_stream__ << "gp_scale";
##  580 :         param_names__.push_back(param_name_stream__.str());
##  581 :         param_name_stream__.str(std::string());
##  582 :         param_name_stream__ << "gp_sigmaSq";
##  583 :         param_names__.push_back(param_name_stream__.str());
##  584 :         param_name_stream__.str(std::string());
##  585 :         param_name_stream__ << "jitter_sq";
##  586 :         param_names__.push_back(param_name_stream__.str());
##  587 :         param_name_stream__.str(std::string());
##  588 :         param_name_stream__ << "gammaA";
##  589 :         param_names__.push_back(param_name_stream__.str());
##  590 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  591 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  592 :                 param_name_stream__.str(std::string());
##  593 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  594 :                 param_names__.push_back(param_name_stream__.str());
##  595 :             }
##  596 :         }
##  597 :
##  598 :         if (!include_gqs__ && !include_tparams__) return;
##  599 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  600 :             param_name_stream__.str(std::string());
##  601 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  602 :             param_names__.push_back(param_name_stream__.str());
##  603 :         }
##  604 :
##  605 :         if (!include_gqs__) return;
##  606 :     }
##  607 :
##  608 : }; // model
##  609 :
##  610 : } // namespace
##  611 :
```

```
##  612 : typedef model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaussianGamma stan_model
##  613 :
##  614 : #include <rstan/rstaninc.hpp>
##  615 : /**
##  616 :  * Define Rcpp Module to expose stan_fit's functions to R.
##  617 :  */
##  618 : RCPP_MODULE(stan_fit4model53fb6879b757_gaussianGamma_mod){
##  619 :   Rcpp::class_<rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_ga
##  620 :                boost::random::ecuyer1988> >("stan_fit4model53fb6879b757_gaussianGamma")
##  621 :     // .constructor<Rcpp::List>()
##  622 :     .constructor<SEXP, SEXP>()
##  623 :     // .constructor<SEXP, SEXP>()
##  624 :     .method("call_sampler",
##  625 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  626 :     .method("param_names",
##  627 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  628 :     .method("param_names_oi",
##  629 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  630 :     .method("param_fnames_oi",
##  631 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  632 :     .method("param_dims",
##  633 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  634 :     .method("param_dims_oi",
##  635 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  636 :     .method("update_param_oi",
##  637 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  638 :     .method("param_oi_tidx",
##  639 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  640 :     .method("grad_log_prob",
##  641 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  642 :     .method("log_prob",
##  643 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  644 :     .method("unconstrain_pars",
##  645 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  646 :     .method("constrain_pars",
##  647 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  648 :     .method("num_pars_unconstrained",
##  649 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  650 :     .method("unconstrained_param_names",
##  651 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  652 :     .method("constrained_param_names",
##  653 :             &rstan::stan_fit<model53fb6879b757_gaussianGamma_namespace::model53fb6879b757_gaus
##  654 :     ;
##  655 : }
##  656 :
##  657 : // declarations
##  658 : extern "C" {
##  659 : SEXP file53fbef3e657( ) ;
##  660 : }
##  661 :
##  662 : // definition
##  663 :
##  664 : SEXP file53fbef3e657(  ){
##  665 :  return Rcpp::wrap("gaussianGamma");
```

```
##  666 : }
##  667 :
##  668 :
## Compilation argument:
##  /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB file53fbef3e657.cpp 2> file53fbef3e657.cp
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'gaussianGamma' NOW.
##
## COMPILING MODEL 'gaussianGamma' NOW.
##
## STARTING SAMPLER FOR MODEL 'gaussianGamma' NOW.
```

```
# Named list of data
spatglm_data = list(nKnots = nKnots, nLocs = nLocs, N = nLocs, nT = 1, y = y.gamma,
    location = seq(1, nLocs), distKnotsSq = distKnotsSq, distKnots21Sq = distKnots21Sq)
# parameters to monitor
spatglm_pars = c("gp_scale", "spatialEffectsKnots", "gp_sigmaSq", "jitter_sq",
    "scaledf")

# fit the model
stanMod_mvtGamma = stan(file = "mvtGamma.stan", data = spatglm_data, verbose = TRUE,
    chains = 4, thin = 1, warmup = 500, iter = 1000, pars = spatglm_pars)
```

```
##
## TRANSLATING MODEL 'mvtGamma' FROM Stan CODE TO C++ CODE NOW.
## successful in parsing the Stan model 'mvtGamma'.
## OS: x86_64, darwin13.4.0; rstan: 2.9.0.3; Rcpp: 0.12.5; inline: 0.3.14
##  >> setting environment variables:
## PKG_LIBS =
## PKG_CPPFLAGS =   -isystem"/Users/eric.ward/Library/R/3.2/library/Rcpp/include/"  -isystem"/Users/eri
##  >> Program source :
##
##    1 :
##    2 : // includes from the plugin
##    3 :
##    4 :
##    5 : // user includes
##    6 : #define STAN__SERVICES__COMMAND_HPP// Code generated by Stan version 2.9
##    7 :
##    8 : #include <stan/model/model_header.hpp>
##    9 :
##   10 : namespace model53fb547d1159_mvtGamma_namespace {
##   11 :
##   12 : using std::istream;
##   13 : using std::string;
##   14 : using std::stringstream;
##   15 : using std::vector;
##   16 : using stan::io::dump;
##   17 : using stan::math::lgamma;
##   18 : using stan::model::prob_grad;
##   19 : using namespace stan::math;
##   20 :
##   21 : typedef Eigen::Matrix<double,Eigen::Dynamic,1> vector_d;
##   22 : typedef Eigen::Matrix<double,1,Eigen::Dynamic> row_vector_d;
```

```
##   23 : typedef Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic> matrix_d;
##   24 :
##   25 : static int current_statement_begin__;
##   26 : class model53fb547d1159_mvtGamma : public prob_grad {
##   27 : private:
##   28 :     int nKnots;
##   29 :     int nLocs;
##   30 :     int N;
##   31 :     int nT;
##   32 :     vector<double> y;
##   33 :     vector<int> location;
##   34 :     matrix_d distKnotsSq;
##   35 :     matrix_d distKnots21Sq;
##   36 : public:
##   37 :     model53fb547d1159_mvtGamma(stan::io::var_context& context__,
##   38 :         std::ostream* pstream__ = 0)
##   39 :         : prob_grad(0) {
##   40 :         current_statement_begin__ = -1;
##   41 :
##   42 :         static const char* function__ = "model53fb547d1159_mvtGamma_namespace::model53fb547d1
##   43 :         (void) function__; // dummy call to supress warning
##   44 :         size_t pos__;
##   45 :         (void) pos__; // dummy call to supress warning
##   46 :         std::vector<int> vals_i__;
##   47 :         std::vector<double> vals_r__;
##   48 :         context__.validate_dims("data initialization", "nKnots", "int", context__.to_vec());
##   49 :         nKnots = int(0);
##   50 :         vals_i__ = context__.vals_i("nKnots");
##   51 :         pos__ = 0;
##   52 :         nKnots = vals_i__[pos__++];
##   53 :         context__.validate_dims("data initialization", "nLocs", "int", context__.to_vec());
##   54 :         nLocs = int(0);
##   55 :         vals_i__ = context__.vals_i("nLocs");
##   56 :         pos__ = 0;
##   57 :         nLocs = vals_i__[pos__++];
##   58 :         context__.validate_dims("data initialization", "N", "int", context__.to_vec());
##   59 :         N = int(0);
##   60 :         vals_i__ = context__.vals_i("N");
##   61 :         pos__ = 0;
##   62 :         N = vals_i__[pos__++];
##   63 :         context__.validate_dims("data initialization", "nT", "int", context__.to_vec());
##   64 :         nT = int(0);
##   65 :         vals_i__ = context__.vals_i("nT");
##   66 :         pos__ = 0;
##   67 :         nT = vals_i__[pos__++];
##   68 :         context__.validate_dims("data initialization", "y", "double", context__.to_vec(N));
##   69 :         validate_non_negative_index("y", "N", N);
##   70 :         y = std::vector<double>(N,double(0));
##   71 :         vals_r__ = context__.vals_r("y");
##   72 :         pos__ = 0;
##   73 :         size_t y_limit_0__ = N;
##   74 :         for (size_t i_0__ = 0; i_0__ < y_limit_0__; ++i_0__) {
##   75 :             y[i_0__] = vals_r__[pos__++];
##   76 :         }
```

```
##   77 :         context__.validate_dims("data initialization", "location", "int", context__.to_vec(N))
##   78 :         validate_non_negative_index("location", "N", N);
##   79 :         location = std::vector<int>(N,int(0));
##   80 :         vals_i__ = context__.vals_i("location");
##   81 :         pos__ = 0;
##   82 :         size_t location_limit_0__ = N;
##   83 :         for (size_t i_0__ = 0; i_0__ < location_limit_0__; ++i_0__) {
##   84 :             location[i_0__] = vals_i__[pos__++];
##   85 :         }
##   86 :         context__.validate_dims("data initialization", "distKnotsSq", "matrix_d", context__.to
##   87 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   88 :         validate_non_negative_index("distKnotsSq", "nKnots", nKnots);
##   89 :         distKnotsSq = matrix_d(nKnots,nKnots);
##   90 :         vals_r__ = context__.vals_r("distKnotsSq");
##   91 :         pos__ = 0;
##   92 :         size_t distKnotsSq_m_mat_lim__ = nKnots;
##   93 :         size_t distKnotsSq_n_mat_lim__ = nKnots;
##   94 :         for (size_t n_mat__ = 0; n_mat__ < distKnotsSq_n_mat_lim__; ++n_mat__) {
##   95 :             for (size_t m_mat__ = 0; m_mat__ < distKnotsSq_m_mat_lim__; ++m_mat__) {
##   96 :                 distKnotsSq(m_mat__,n_mat__) = vals_r__[pos__++];
##   97 :             }
##   98 :         }
##   99 :         context__.validate_dims("data initialization", "distKnots21Sq", "matrix_d", context__
##  100 :         validate_non_negative_index("distKnots21Sq", "nLocs", nLocs);
##  101 :         validate_non_negative_index("distKnots21Sq", "nKnots", nKnots);
##  102 :         distKnots21Sq = matrix_d(nLocs,nKnots);
##  103 :         vals_r__ = context__.vals_r("distKnots21Sq");
##  104 :         pos__ = 0;
##  105 :         size_t distKnots21Sq_m_mat_lim__ = nLocs;
##  106 :         size_t distKnots21Sq_n_mat_lim__ = nKnots;
##  107 :         for (size_t n_mat__ = 0; n_mat__ < distKnots21Sq_n_mat_lim__; ++n_mat__) {
##  108 :             for (size_t m_mat__ = 0; m_mat__ < distKnots21Sq_m_mat_lim__; ++m_mat__) {
##  109 :                 distKnots21Sq(m_mat__,n_mat__) = vals_r__[pos__++];
##  110 :             }
##  111 :         }
##  112 :
##  113 :         // validate data
##  114 :         check_greater_or_equal(function__,"nKnots",nKnots,1);
##  115 :         check_greater_or_equal(function__,"nLocs",nLocs,1);
##  116 :         check_greater_or_equal(function__,"N",N,1);
##  117 :         check_greater_or_equal(function__,"nT",nT,1);
##  118 :
##  119 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  120 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  121 :
##  122 :
##  123 :         // initialize transformed variables to avoid seg fault on val access
##  124 :
##  125 :         try {
##  126 :         } catch (const std::exception& e) {
##  127 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  128 :             // Next line prevents compiler griping about no return
##  129 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  130 :         }
```

```
##  131 :
##  132 :         // validate transformed data
##  133 :
##  134 :         // set parameter ranges
##  135 :         num_params_r__ = 0U;
##  136 :         param_ranges_i__.clear();
##  137 :         ++num_params_r__;
##  138 :         ++num_params_r__;
##  139 :         ++num_params_r__;
##  140 :         ++num_params_r__;
##  141 :         ++num_params_r__;
##  142 :         num_params_r__ += nKnots * nT;
##  143 :     }
##  144 :
##  145 :     ~model53fb547d1159_mvtGamma() { }
##  146 :
##  147 :
##  148 :     void transform_inits(const stan::io::var_context& context__,
##  149 :                          std::vector<int>& params_i__,
##  150 :                          std::vector<double>& params_r__,
##  151 :                          std::ostream* pstream__) const {
##  152 :         stan::io::writer<double> writer__(params_r__,params_i__);
##  153 :         size_t pos__;
##  154 :         (void) pos__; // dummy call to supress warning
##  155 :         std::vector<double> vals_r__;
##  156 :         std::vector<int> vals_i__;
##  157 :
##  158 :         if (!(context__.contains_r("gp_scale")))
##  159 :             throw std::runtime_error("variable gp_scale missing");
##  160 :         vals_r__ = context__.vals_r("gp_scale");
##  161 :         pos__ = 0U;
##  162 :         context__.validate_dims("initialization", "gp_scale", "double", context__.to_vec());
##  163 :         double gp_scale(0);
##  164 :         gp_scale = vals_r__[pos__++];
##  165 :         try {
##  166 :             writer__.scalar_lb_unconstrain(0,gp_scale);
##  167 :         } catch (const std::exception& e) {
##  168 :             throw std::runtime_error(std::string("Error transforming variable gp_scale: ") + e
##  169 :         }
##  170 :
##  171 :         if (!(context__.contains_r("gp_sigmaSq")))
##  172 :             throw std::runtime_error("variable gp_sigmaSq missing");
##  173 :         vals_r__ = context__.vals_r("gp_sigmaSq");
##  174 :         pos__ = 0U;
##  175 :         context__.validate_dims("initialization", "gp_sigmaSq", "double", context__.to_vec())
##  176 :         double gp_sigmaSq(0);
##  177 :         gp_sigmaSq = vals_r__[pos__++];
##  178 :         try {
##  179 :             writer__.scalar_lb_unconstrain(0,gp_sigmaSq);
##  180 :         } catch (const std::exception& e) {
##  181 :             throw std::runtime_error(std::string("Error transforming variable gp_sigmaSq: ") +
##  182 :         }
##  183 :
##  184 :         if (!(context__.contains_r("jitter_sq")))
```

```
##  185 :             throw std::runtime_error("variable jitter_sq missing");
##  186 :         vals_r__ = context__.vals_r("jitter_sq");
##  187 :         pos__ = 0U;
##  188 :         context__.validate_dims("initialization", "jitter_sq", "double", context__.to_vec());
##  189 :         double jitter_sq(0);
##  190 :         jitter_sq = vals_r__[pos__++];
##  191 :         try {
##  192 :             writer__.scalar_lb_unconstrain(0,jitter_sq);
##  193 :         } catch (const std::exception& e) {
##  194 :             throw std::runtime_error(std::string("Error transforming variable jitter_sq: ") +
##  195 :         }
##  196 :
##  197 :         if (!(context__.contains_r("scaledf")))
##  198 :             throw std::runtime_error("variable scaledf missing");
##  199 :         vals_r__ = context__.vals_r("scaledf");
##  200 :         pos__ = 0U;
##  201 :         context__.validate_dims("initialization", "scaledf", "double", context__.to_vec());
##  202 :         double scaledf(0);
##  203 :         scaledf = vals_r__[pos__++];
##  204 :         try {
##  205 :             writer__.scalar_lb_unconstrain(0,scaledf);
##  206 :         } catch (const std::exception& e) {
##  207 :             throw std::runtime_error(std::string("Error transforming variable scaledf: ") + e
##  208 :         }
##  209 :
##  210 :         if (!(context__.contains_r("gammaA")))
##  211 :             throw std::runtime_error("variable gammaA missing");
##  212 :         vals_r__ = context__.vals_r("gammaA");
##  213 :         pos__ = 0U;
##  214 :         context__.validate_dims("initialization", "gammaA", "double", context__.to_vec());
##  215 :         double gammaA(0);
##  216 :         gammaA = vals_r__[pos__++];
##  217 :         try {
##  218 :             writer__.scalar_lb_unconstrain(0,gammaA);
##  219 :         } catch (const std::exception& e) {
##  220 :             throw std::runtime_error(std::string("Error transforming variable gammaA: ") + e.w
##  221 :         }
##  222 :
##  223 :         if (!(context__.contains_r("spatialEffectsKnots")))
##  224 :             throw std::runtime_error("variable spatialEffectsKnots missing");
##  225 :         vals_r__ = context__.vals_r("spatialEffectsKnots");
##  226 :         pos__ = 0U;
##  227 :         context__.validate_dims("initialization", "spatialEffectsKnots", "vector_d", context_
##  228 :         std::vector<vector_d> spatialEffectsKnots(nT,vector_d(nKnots));
##  229 :         for (int j1__ = 0U; j1__ < nKnots; ++j1__)
##  230 :             for (int i0__ = 0U; i0__ < nT; ++i0__)
##  231 :                 spatialEffectsKnots[i0__](j1__) = vals_r__[pos__++];
##  232 :         for (int i0__ = 0U; i0__ < nT; ++i0__)
##  233 :             try {
##  234 :             writer__.vector_unconstrain(spatialEffectsKnots[i0__]);
##  235 :         } catch (const std::exception& e) {
##  236 :             throw std::runtime_error(std::string("Error transforming variable spatialEffectsKn
##  237 :         }
##  238 :
```

```
##  239 :         params_r__ = writer__.data_r();
##  240 :         params_i__ = writer__.data_i();
##  241 :     }
##  242 :
##  243 :     void transform_inits(const stan::io::var_context& context,
##  244 :                          Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  245 :                          std::ostream* pstream__) const {
##  246 :       std::vector<double> params_r_vec;
##  247 :       std::vector<int> params_i_vec;
##  248 :       transform_inits(context, params_i_vec, params_r_vec, pstream__);
##  249 :       params_r.resize(params_r_vec.size());
##  250 :       for (int i = 0; i < params_r.size(); ++i)
##  251 :         params_r(i) = params_r_vec[i];
##  252 :     }
##  253 :
##  254 :
##  255 :     template <bool propto__, bool jacobian__, typename T__>
##  256 :     T__ log_prob(vector<T__>& params_r__,
##  257 :                  vector<int>& params_i__,
##  258 :                  std::ostream* pstream__ = 0) const {
##  259 :
##  260 :         T__ DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  261 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  262 :
##  263 :         T__ lp__(0.0);
##  264 :         stan::math::accumulator<T__> lp_accum__;
##  265 :
##  266 :         // model parameters
##  267 :         stan::io::reader<T__> in__(params_r__,params_i__);
##  268 :
##  269 :         T__ gp_scale;
##  270 :         (void) gp_scale;   // dummy to suppress unused var warning
##  271 :         if (jacobian__)
##  272 :             gp_scale = in__.scalar_lb_constrain(0,lp__);
##  273 :         else
##  274 :             gp_scale = in__.scalar_lb_constrain(0);
##  275 :
##  276 :         T__ gp_sigmaSq;
##  277 :         (void) gp_sigmaSq;   // dummy to suppress unused var warning
##  278 :         if (jacobian__)
##  279 :             gp_sigmaSq = in__.scalar_lb_constrain(0,lp__);
##  280 :         else
##  281 :             gp_sigmaSq = in__.scalar_lb_constrain(0);
##  282 :
##  283 :         T__ jitter_sq;
##  284 :         (void) jitter_sq;   // dummy to suppress unused var warning
##  285 :         if (jacobian__)
##  286 :             jitter_sq = in__.scalar_lb_constrain(0,lp__);
##  287 :         else
##  288 :             jitter_sq = in__.scalar_lb_constrain(0);
##  289 :
##  290 :         T__ scaledf;
##  291 :         (void) scaledf;   // dummy to suppress unused var warning
##  292 :         if (jacobian__)
```

```
##  293 :             scaledf = in__.scalar_lb_constrain(0,lp__);
##  294 :         else
##  295 :             scaledf = in__.scalar_lb_constrain(0);
##  296 :
##  297 :         T__ gammaA;
##  298 :         (void) gammaA;   // dummy to suppress unused var warning
##  299 :         if (jacobian__)
##  300 :             gammaA = in__.scalar_lb_constrain(0,lp__);
##  301 :         else
##  302 :             gammaA = in__.scalar_lb_constrain(0);
##  303 :
##  304 :         vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffectsKnots;
##  305 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  306 :         spatialEffectsKnots.reserve(dim_spatialEffectsKnots_0__);
##  307 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  308 :             if (jacobian__)
##  309 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots,lp__));
##  310 :             else
##  311 :                 spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  312 :         }
##  313 :
##  314 :
##  315 :         // transformed parameters
##  316 :         Eigen::Matrix<T__,Eigen::Dynamic,1>  muZeros(nKnots);
##  317 :         (void) muZeros;  // dummy to suppress unused var warning
##  318 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  319 :
##  320 :         // initialize transformed variables to avoid seg fault on val access
##  321 :         stan::math::fill(muZeros,DUMMY_VAR__);
##  322 :
##  323 :         try {
##  324 :             current_statement_begin__ = 25;
##  325 :             for (int i = 1; i <= nKnots; ++i) {
##  326 :                 current_statement_begin__ = 26;
##  327 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  328 :             }
##  329 :         } catch (const std::exception& e) {
##  330 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  331 :             // Next line prevents compiler griping about no return
##  332 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  333 :         }
##  334 :
##  335 :         // validate transformed parameters
##  336 :         for (int i0__ = 0; i0__ < nKnots; ++i0__) {
##  337 :             if (stan::math::is_uninitialized(muZeros(i0__))) {
##  338 :                 std::stringstream msg__;
##  339 :                 msg__ << "Undefined transformed parameter: muZeros" << '[' << i0__ << ']';
##  340 :                 throw std::runtime_error(msg__.str());
##  341 :             }
##  342 :         }
##  343 :
##  344 :         const char* function__ = "validate transformed params";
##  345 :         (void) function__; // dummy to suppress unused var warning
##  346 :
```

```
##  347 :         // model body
##  348 :         try {
##  349 :             {
##  350 :                 T__ DF;
##  351 :                 (void) DF;  // dummy to suppress unused var warning
##  352 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots(nKnots,nKnots);
##  353 :                 (void) SigmaKnots;  // dummy to suppress unused var warning
##  354 :                 stan::math::fill(SigmaKnots,DUMMY_VAR__);
##  355 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaKnots_chol(nKnots,nKno
##  356 :                 (void) SigmaKnots_chol;  // dummy to suppress unused var warning
##  357 :                 stan::math::fill(SigmaKnots_chol,DUMMY_VAR__);
##  358 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  SigmaOffDiag(nLocs,nKnots);
##  359 :                 (void) SigmaOffDiag;  // dummy to suppress unused var warning
##  360 :                 stan::math::fill(SigmaOffDiag,DUMMY_VAR__);
##  361 :                 vector<Eigen::Matrix<T__,Eigen::Dynamic,1> > spatialEffects(nT, (Eigen::Matri
##  362 :                 stan::math::fill(spatialEffects,DUMMY_VAR__);
##  363 :                 Eigen::Matrix<T__,Eigen::Dynamic,Eigen::Dynamic>  invSigmaKnots(nKnots,nKnots
##  364 :                 (void) invSigmaKnots;  // dummy to suppress unused var warning
##  365 :                 stan::math::fill(invSigmaKnots,DUMMY_VAR__);
##  366 :                 stan::math::initialize(DF, DUMMY_VAR__);
##  367 :                 stan::math::initialize(SigmaKnots, DUMMY_VAR__);
##  368 :                 stan::math::initialize(SigmaKnots_chol, DUMMY_VAR__);
##  369 :                 stan::math::initialize(SigmaOffDiag, DUMMY_VAR__);
##  370 :                 stan::math::initialize(spatialEffects, DUMMY_VAR__);
##  371 :                 stan::math::initialize(invSigmaKnots, DUMMY_VAR__);
##  372 :                 current_statement_begin__ = 38;
##  373 :                 stan::math::assign(SigmaKnots, multiply(gp_sigmaSq,exp(multiply(-(gp_scale),di
##  374 :                 current_statement_begin__ = 39;
##  375 :                 stan::math::assign(SigmaOffDiag, multiply(gp_sigmaSq,exp(multiply(-(gp_scale)
##  376 :                 current_statement_begin__ = 40;
##  377 :                 for (int i = 1; i <= nKnots; ++i) {
##  378 :                     current_statement_begin__ = 41;
##  379 :                     stan::math::assign(get_base1_lhs(SigmaKnots,i,i,"SigmaKnots",1), (jitter_s
##  380 :                     current_statement_begin__ = 42;
##  381 :                     stan::math::assign(get_base1_lhs(SigmaOffDiag,i,i,"SigmaOffDiag",1), (jit
##  382 :                 }
##  383 :                 current_statement_begin__ = 44;
##  384 :                 stan::math::assign(invSigmaKnots, inverse(SigmaKnots));
##  385 :                 current_statement_begin__ = 48;
##  386 :                 stan::math::assign(SigmaKnots_chol, cholesky_decompose(SigmaKnots));
##  387 :                 current_statement_begin__ = 51;
##  388 :                 lp_accum__.add(multi_normal_cholesky_log<propto__>(get_base1(spatialEffectsKno
##  389 :                 current_statement_begin__ = 53;
##  390 :                 stan::math::assign(DF, 2);
##  391 :                 current_statement_begin__ = 54;
##  392 :                 lp_accum__.add(chi_square_log<propto__>(scaledf, DF));
##  393 :                 current_statement_begin__ = 55;
##  394 :                 stan::math::assign(get_base1_lhs(spatialEffects,1,"spatialEffects",1), multipl
##  395 :                 current_statement_begin__ = 58;
##  396 :                 lp_accum__.add(cauchy_log<propto__>(gp_scale, 0, 5));
##  397 :                 current_statement_begin__ = 59;
##  398 :                 lp_accum__.add(cauchy_log<propto__>(gp_sigmaSq, 0, 5));
##  399 :                 current_statement_begin__ = 60;
##  400 :                 lp_accum__.add(cauchy_log<propto__>(jitter_sq, 0, 5));
```

```
##  401 :                 current_statement_begin__ = 61;
##  402 :                 lp_accum__.add(cauchy_log<propto__>(gammaA, 0, 5));
##  403 :                 current_statement_begin__ = 62;
##  404 :                 for (int n = 1; n <= N; ++n) {
##  405 :                     current_statement_begin__ = 63;
##  406 :                     lp_accum__.add(gamma_log<propto__>(get_base1(y,n,"y",1), gammaA, (gammaA /
##  407 :                 }
##  408 :             }
##  409 :         } catch (const std::exception& e) {
##  410 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  411 :             // Next line prevents compiler griping about no return
##  412 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  413 :         }
##  414 :
##  415 :         lp_accum__.add(lp__);
##  416 :         return lp_accum__.sum();
##  417 :
##  418 :     } // log_prob()
##  419 :
##  420 :     template <bool propto, bool jacobian, typename T_>
##  421 :     T_ log_prob(Eigen::Matrix<T_,Eigen::Dynamic,1>& params_r,
##  422 :                 std::ostream* pstream = 0) const {
##  423 :       std::vector<T_> vec_params_r;
##  424 :       vec_params_r.reserve(params_r.size());
##  425 :       for (int i = 0; i < params_r.size(); ++i)
##  426 :         vec_params_r.push_back(params_r(i));
##  427 :       std::vector<int> vec_params_i;
##  428 :       return log_prob<propto,jacobian,T_>(vec_params_r, vec_params_i, pstream);
##  429 :     }
##  430 :
##  431 :
##  432 :     void get_param_names(std::vector<std::string>& names__) const {
##  433 :         names__.resize(0);
##  434 :         names__.push_back("gp_scale");
##  435 :         names__.push_back("gp_sigmaSq");
##  436 :         names__.push_back("jitter_sq");
##  437 :         names__.push_back("scaledf");
##  438 :         names__.push_back("gammaA");
##  439 :         names__.push_back("spatialEffectsKnots");
##  440 :         names__.push_back("muZeros");
##  441 :     }
##  442 :
##  443 :
##  444 :     void get_dims(std::vector<std::vector<size_t> >& dimss__) const {
##  445 :         dimss__.resize(0);
##  446 :         std::vector<size_t> dims__;
##  447 :         dims__.resize(0);
##  448 :         dimss__.push_back(dims__);
##  449 :         dims__.resize(0);
##  450 :         dimss__.push_back(dims__);
##  451 :         dims__.resize(0);
##  452 :         dimss__.push_back(dims__);
##  453 :         dims__.resize(0);
##  454 :         dimss__.push_back(dims__);
```

```
##  455 :         dims__.resize(0);
##  456 :         dimss__.push_back(dims__);
##  457 :         dims__.resize(0);
##  458 :         dims__.push_back(nT);
##  459 :         dims__.push_back(nKnots);
##  460 :         dimss__.push_back(dims__);
##  461 :         dims__.resize(0);
##  462 :         dims__.push_back(nKnots);
##  463 :         dimss__.push_back(dims__);
##  464 :     }
##  465 :
##  466 :     template <typename RNG>
##  467 :     void write_array(RNG& base_rng__,
##  468 :                      std::vector<double>& params_r__,
##  469 :                      std::vector<int>& params_i__,
##  470 :                      std::vector<double>& vars__,
##  471 :                      bool include_tparams__ = true,
##  472 :                      bool include_gqs__ = true,
##  473 :                      std::ostream* pstream__ = 0) const {
##  474 :         vars__.resize(0);
##  475 :         stan::io::reader<double> in__(params_r__,params_i__);
##  476 :         static const char* function__ = "model53fb547d1159_mvtGamma_namespace::write_array";
##  477 :         (void) function__; // dummy call to supress warning
##  478 :         // read-transform, write parameters
##  479 :         double gp_scale = in__.scalar_lb_constrain(0);
##  480 :         double gp_sigmaSq = in__.scalar_lb_constrain(0);
##  481 :         double jitter_sq = in__.scalar_lb_constrain(0);
##  482 :         double scaledf = in__.scalar_lb_constrain(0);
##  483 :         double gammaA = in__.scalar_lb_constrain(0);
##  484 :         vector<vector_d> spatialEffectsKnots;
##  485 :         size_t dim_spatialEffectsKnots_0__ = nT;
##  486 :         for (size_t k_0__ = 0; k_0__ < dim_spatialEffectsKnots_0__; ++k_0__) {
##  487 :             spatialEffectsKnots.push_back(in__.vector_constrain(nKnots));
##  488 :         }
##  489 :         vars__.push_back(gp_scale);
##  490 :         vars__.push_back(gp_sigmaSq);
##  491 :         vars__.push_back(jitter_sq);
##  492 :         vars__.push_back(scaledf);
##  493 :         vars__.push_back(gammaA);
##  494 :         for (int k_1__ = 0; k_1__ < nKnots; ++k_1__) {
##  495 :             for (int k_0__ = 0; k_0__ < nT; ++k_0__) {
##  496 :                 vars__.push_back(spatialEffectsKnots[k_0__][k_1__]);
##  497 :             }
##  498 :         }
##  499 :
##  500 :         if (!include_tparams__) return;
##  501 :         // declare and define transformed parameters
##  502 :         double lp__ = 0.0;
##  503 :         (void) lp__; // dummy call to supress warning
##  504 :         stan::math::accumulator<double> lp_accum__;
##  505 :
##  506 :         vector_d muZeros(nKnots);
##  507 :         (void) muZeros;  // dummy to suppress unused var warning
##  508 :
```

```
##  509 :         try {
##  510 :             current_statement_begin__ = 25;
##  511 :             for (int i = 1; i <= nKnots; ++i) {
##  512 :                 current_statement_begin__ = 26;
##  513 :                 stan::math::assign(get_base1_lhs(muZeros,i,"muZeros",1), 0);
##  514 :             }
##  515 :         } catch (const std::exception& e) {
##  516 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  517 :             // Next line prevents compiler griping about no return
##  518 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  519 :         }
##  520 :
##  521 :         // validate transformed parameters
##  522 :
##  523 :         // write transformed parameters
##  524 :         for (int k_0__ = 0; k_0__ < nKnots; ++k_0__) {
##  525 :             vars__.push_back(muZeros[k_0__]);
##  526 :         }
##  527 :
##  528 :         if (!include_gqs__) return;
##  529 :         // declare and define generated quantities
##  530 :
##  531 :         double DUMMY_VAR__(std::numeric_limits<double>::quiet_NaN());
##  532 :         (void) DUMMY_VAR__;  // suppress unused var warning
##  533 :
##  534 :
##  535 :         // initialize transformed variables to avoid seg fault on val access
##  536 :
##  537 :         try {
##  538 :         } catch (const std::exception& e) {
##  539 :             stan::lang::rethrow_located(e,current_statement_begin__);
##  540 :             // Next line prevents compiler griping about no return
##  541 : throw std::runtime_error("*** IF YOU SEE THIS, PLEASE REPORT A BUG ***");
##  542 :         }
##  543 :
##  544 :         // validate generated quantities
##  545 :
##  546 :         // write generated quantities
##  547 :     }
##  548 :
##  549 :     template <typename RNG>
##  550 :     void write_array(RNG& base_rng,
##  551 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& params_r,
##  552 :                      Eigen::Matrix<double,Eigen::Dynamic,1>& vars,
##  553 :                      bool include_tparams = true,
##  554 :                      bool include_gqs = true,
##  555 :                      std::ostream* pstream = 0) const {
##  556 :       std::vector<double> params_r_vec(params_r.size());
##  557 :       for (int i = 0; i < params_r.size(); ++i)
##  558 :         params_r_vec[i] = params_r(i);
##  559 :       std::vector<double> vars_vec;
##  560 :       std::vector<int> params_i_vec;
##  561 :       write_array(base_rng,params_r_vec,params_i_vec,vars_vec,include_tparams,include_gqs,pst
##  562 :       vars.resize(vars_vec.size());
```

```
##  563 :       for (int i = 0; i < vars.size(); ++i)
##  564 :         vars(i) = vars_vec[i];
##  565 :     }
##  566 :
##  567 :     static std::string model_name() {
##  568 :         return "model53fb547d1159_mvtGamma";
##  569 :     }
##  570 :
##  571 :
##  572 :     void constrained_param_names(std::vector<std::string>& param_names__,
##  573 :                                  bool include_tparams__ = true,
##  574 :                                  bool include_gqs__ = true) const {
##  575 :         std::stringstream param_name_stream__;
##  576 :         param_name_stream__.str(std::string());
##  577 :         param_name_stream__ << "gp_scale";
##  578 :         param_names__.push_back(param_name_stream__.str());
##  579 :         param_name_stream__.str(std::string());
##  580 :         param_name_stream__ << "gp_sigmaSq";
##  581 :         param_names__.push_back(param_name_stream__.str());
##  582 :         param_name_stream__.str(std::string());
##  583 :         param_name_stream__ << "jitter_sq";
##  584 :         param_names__.push_back(param_name_stream__.str());
##  585 :         param_name_stream__.str(std::string());
##  586 :         param_name_stream__ << "scaledf";
##  587 :         param_names__.push_back(param_name_stream__.str());
##  588 :         param_name_stream__.str(std::string());
##  589 :         param_name_stream__ << "gammaA";
##  590 :         param_names__.push_back(param_name_stream__.str());
##  591 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  592 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  593 :                 param_name_stream__.str(std::string());
##  594 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  595 :                 param_names__.push_back(param_name_stream__.str());
##  596 :             }
##  597 :         }
##  598 :
##  599 :         if (!include_gqs__ && !include_tparams__) return;
##  600 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  601 :             param_name_stream__.str(std::string());
##  602 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  603 :             param_names__.push_back(param_name_stream__.str());
##  604 :         }
##  605 :
##  606 :         if (!include_gqs__) return;
##  607 :     }
##  608 :
##  609 :
##  610 :     void unconstrained_param_names(std::vector<std::string>& param_names__,
##  611 :                                    bool include_tparams__ = true,
##  612 :                                    bool include_gqs__ = true) const {
##  613 :         std::stringstream param_name_stream__;
##  614 :         param_name_stream__.str(std::string());
##  615 :         param_name_stream__ << "gp_scale";
##  616 :         param_names__.push_back(param_name_stream__.str());
```

```
##  617 :         param_name_stream__.str(std::string());
##  618 :         param_name_stream__ << "gp_sigmaSq";
##  619 :         param_names__.push_back(param_name_stream__.str());
##  620 :         param_name_stream__.str(std::string());
##  621 :         param_name_stream__ << "jitter_sq";
##  622 :         param_names__.push_back(param_name_stream__.str());
##  623 :         param_name_stream__.str(std::string());
##  624 :         param_name_stream__ << "scaledf";
##  625 :         param_names__.push_back(param_name_stream__.str());
##  626 :         param_name_stream__.str(std::string());
##  627 :         param_name_stream__ << "gammaA";
##  628 :         param_names__.push_back(param_name_stream__.str());
##  629 :         for (int k_1__ = 1; k_1__ <= nKnots; ++k_1__) {
##  630 :             for (int k_0__ = 1; k_0__ <= nT; ++k_0__) {
##  631 :                 param_name_stream__.str(std::string());
##  632 :                 param_name_stream__ << "spatialEffectsKnots" << '.' << k_0__ << '.' << k_1__;
##  633 :                 param_names__.push_back(param_name_stream__.str());
##  634 :             }
##  635 :         }
##  636 :
##  637 :         if (!include_gqs__ && !include_tparams__) return;
##  638 :         for (int k_0__ = 1; k_0__ <= nKnots; ++k_0__) {
##  639 :             param_name_stream__.str(std::string());
##  640 :             param_name_stream__ << "muZeros" << '.' << k_0__;
##  641 :             param_names__.push_back(param_name_stream__.str());
##  642 :         }
##  643 :
##  644 :         if (!include_gqs__) return;
##  645 :     }
##  646 :
##  647 : }; // model
##  648 :
##  649 : } // namespace
##  650 :
##  651 : typedef model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma stan_model;
##  652 :
##  653 : #include <rstan/rstaninc.hpp>
##  654 : /**
##  655 :  * Define Rcpp Module to expose stan_fit's functions to R.
##  656 :  */
##  657 : RCPP_MODULE(stan_fit4model53fb547d1159_mvtGamma_mod){
##  658 :   Rcpp::class_<rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamm
##  659 :                boost::random::ecuyer1988> >("stan_fit4model53fb547d1159_mvtGamma")
##  660 :     // .constructor<Rcpp::List>()
##  661 :     .constructor<SEXP, SEXP>()
##  662 :     // .constructor<SEXP, SEXP>()
##  663 :     .method("call_sampler",
##  664 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  665 :     .method("param_names",
##  666 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  667 :     .method("param_names_oi",
##  668 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  669 :     .method("param_fnames_oi",
##  670 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
```

```
##  671 :     .method("param_dims",
##  672 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  673 :     .method("param_dims_oi",
##  674 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  675 :     .method("update_param_oi",
##  676 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  677 :     .method("param_oi_tidx",
##  678 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  679 :     .method("grad_log_prob",
##  680 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  681 :     .method("log_prob",
##  682 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  683 :     .method("unconstrain_pars",
##  684 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  685 :     .method("constrain_pars",
##  686 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  687 :     .method("num_pars_unconstrained",
##  688 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  689 :     .method("unconstrained_param_names",
##  690 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  691 :     .method("constrained_param_names",
##  692 :             &rstan::stan_fit<model53fb547d1159_mvtGamma_namespace::model53fb547d1159_mvtGamma
##  693 :     ;
##  694 : }
##  695 :
##  696 : // declarations
##  697 : extern "C" {
##  698 : SEXP file53fb5efe1564( ) ;
##  699 : }
##  700 :
##  701 : // definition
##  702 :
##  703 : SEXP file53fb5efe1564(  ){
##  704 :  return Rcpp::wrap("mvtGamma");
##  705 : }
##  706 :
##  707 :
## Compilation argument:
##  /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB file53fb5efe1564.cpp 2> file53fb5efe1564.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'mvtGamma' NOW.
##
## COMPILING MODEL 'mvtGamma' NOW.
##
## STARTING SAMPLER FOR MODEL 'mvtGamma' NOW.
```