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[1:nKnots, -c(1:nKnots)])
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)
```

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 = "")
```

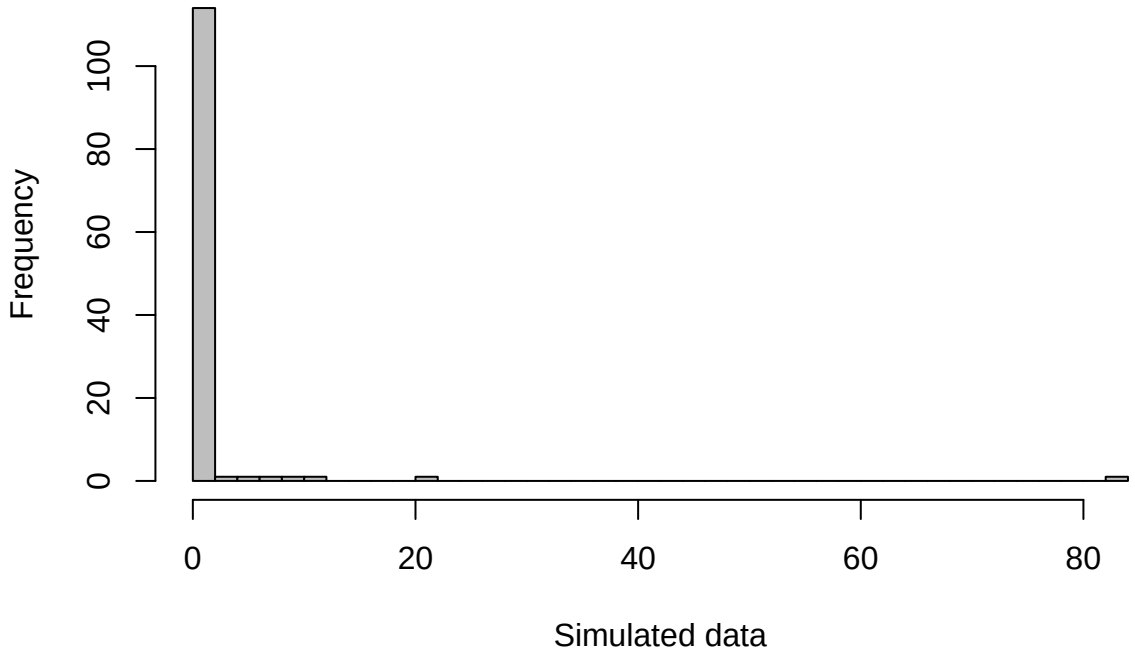

Figure 1: Simulated gamma data, using MVT random effects.

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.

```
jagsscript = cat("
model {
   # 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
   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) {
    for(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
```

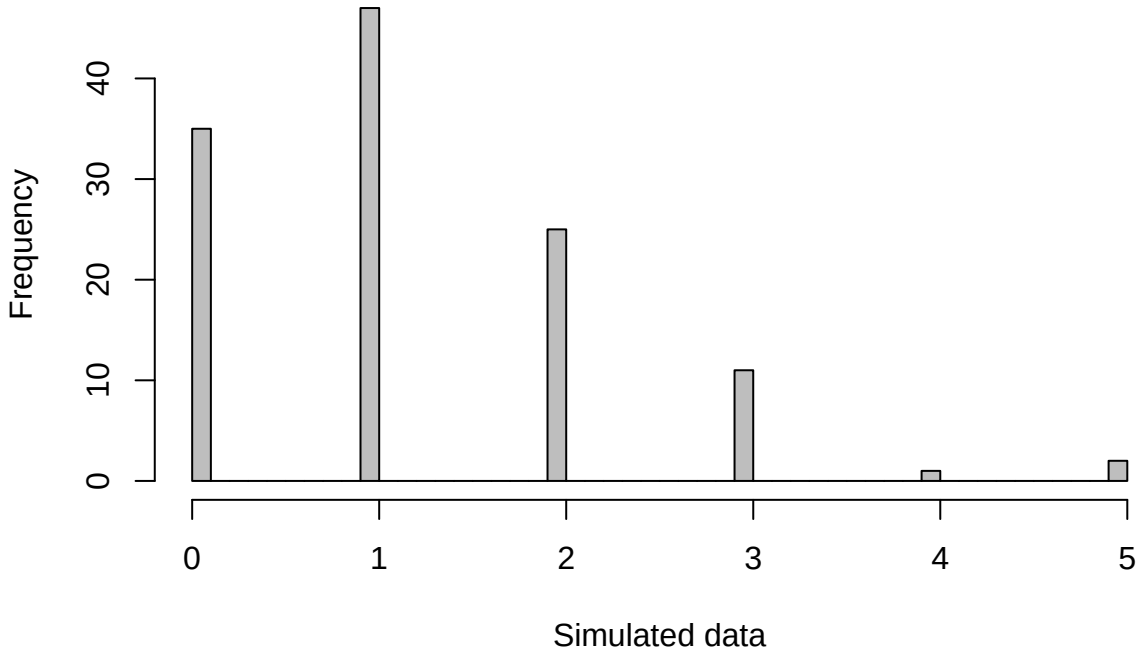

Figure 2: Simulated Poisson data, using MVT random effects.

```
     SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + d
    }
   }
   for(i in 1:nLocs) {
    for(j in 1:nKnots) {
     SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
    }
   }
   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 {
   # 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
   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) {
    for(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
     SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + d
    }
   }
   for(i in 1:nLocs) {
    for(j in 1:nKnots) {
     SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
    }
   }
   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 {
   # 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
   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) {
    for(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
     SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + d
    }
   }
   for(i in 1:nLocs) {
    for(j in 1:nKnots) {
     SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
    }
   }
   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 {
   # 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
   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) {
    for(j in 1:nKnots) {
     # this adds some jitter to the diagonal but not the off-diags
     SigmaKnotsNorm[i,j] <- (1-diagKnots[i,j]) * gp_sigmaSq_norm * exp(-gp_scale * distKnotsSq[i,j]) + (
    }
   }
   for(i in 1:nLocs) {
    for(j in 1:nKnots) {
     SigmaOffDiagNorm[i,j] <- gp_sigmaSq_norm * exp(-gp_scale * distKnots21Sq[i,j]);
    }
   }
   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")
```

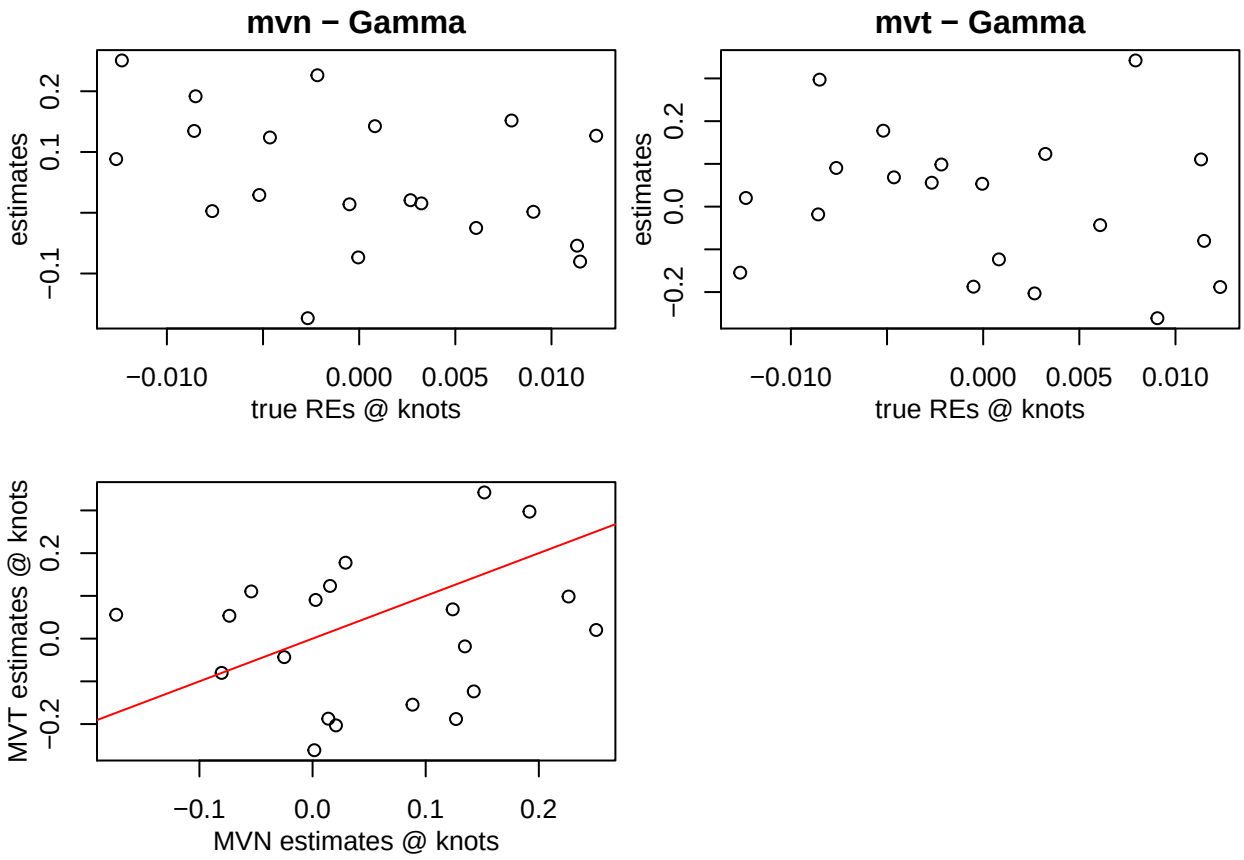

Figure 3: Estimates of random effects for simulated Gamma data

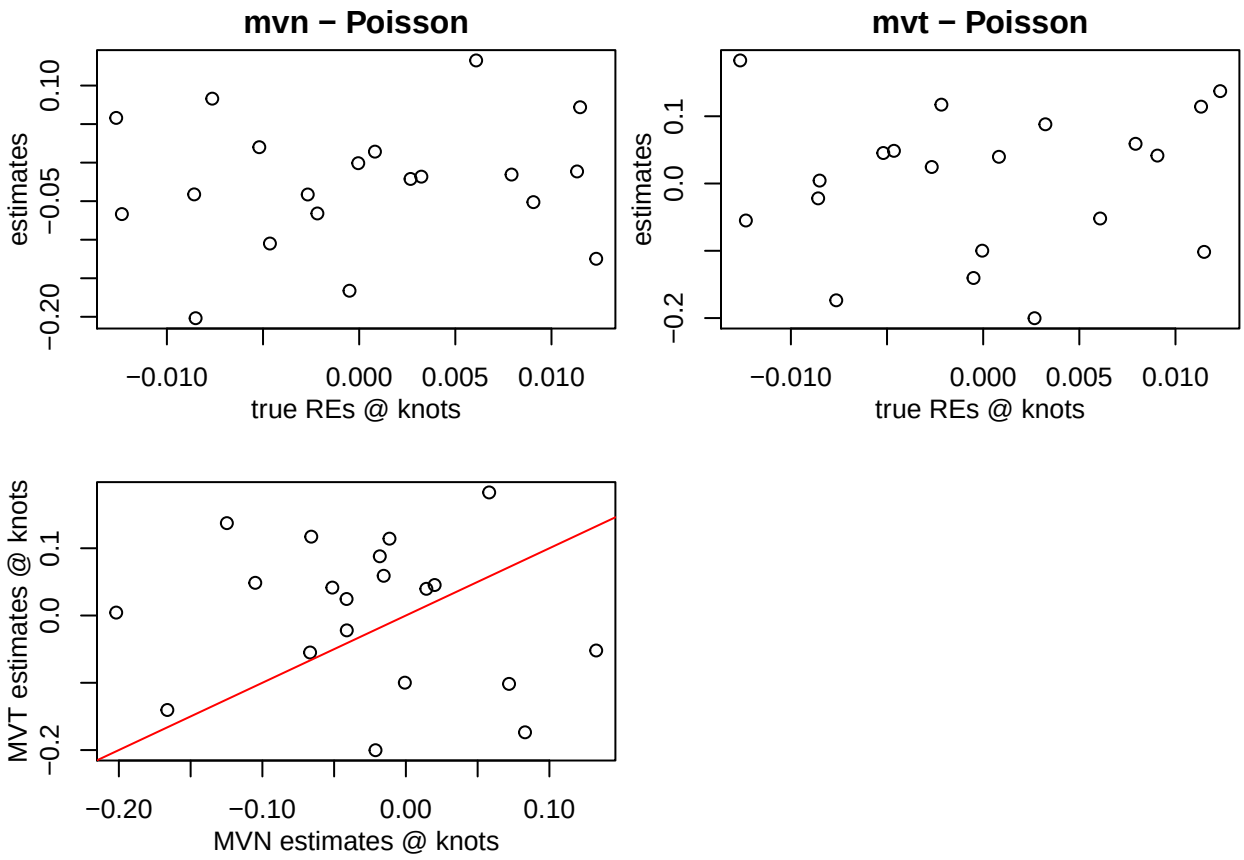

Figure 4: Estimates of random effects for simulated Poisson data