Math and simulations for Multivariate-T spatial knots

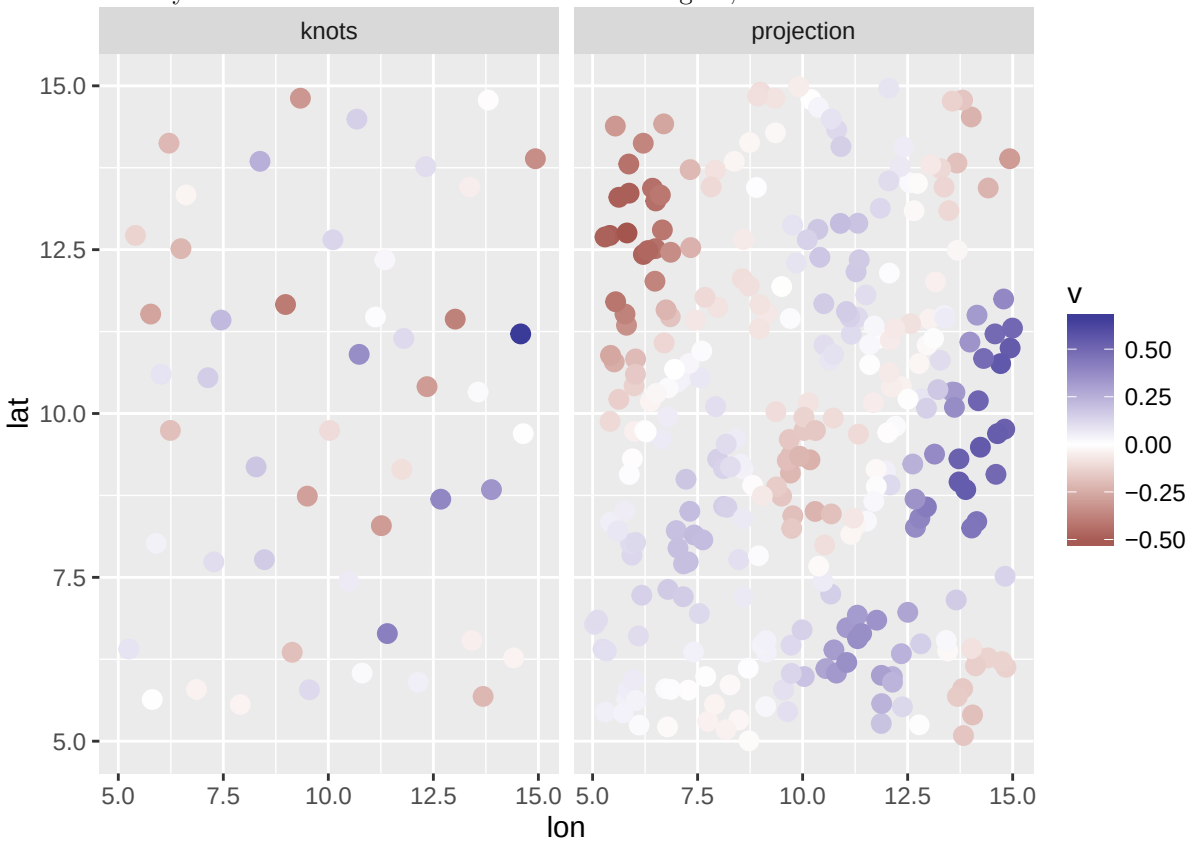
Simulating spatial data from multivariate t

We can simulate spatial data from knots with multivariate - t random effects with the function below.

```
nDataPoints = 300
grid = cbind("lon" = runif(nDataPoints, 5, 15),
  "lat" = runif(nDataPoints, 5, 15))
nLocs = dim(grid)[1]

simulateData = function(nKnots = 50, gp_scale = 0.3, sigma_t = 0.15, nDraws = 1000) {

  # cluster analysis to determine knot locations
  knots = jitter(pam(grid,nKnots)$medoids)
  distKnots = as.matrix(dist(knots))
  distKnotsSq = distKnots^2 # squared distances

  corKnots = exp(-gp_scale*distKnotsSq)
  sigmaKnots = diag(nKnots) * sigma_t * sigma_t# * corKnots *
  invSigmaKnots = solve(sigmaKnots)
  # 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 = exp(-gp_scale*distKnots21Sq) * sigma_t * sigma_t

  # Generate vector of random effects
  # each 'draw' here is hypothetical draw from posterior
  reKnots = rmvt(nDraws, sigma = sigmaKnots, df = 2)
  #reKnots = sweep(reKnots, 2, apply(reKnots, 2, mean))

  # Project random effects to locations of the data
  proj = t((Sigma21 %*% invSigmaKnots) %*% t(reKnots))
  return(list(knots = knots, reKnots = reKnots, proj = proj))
}
```

Checking projection

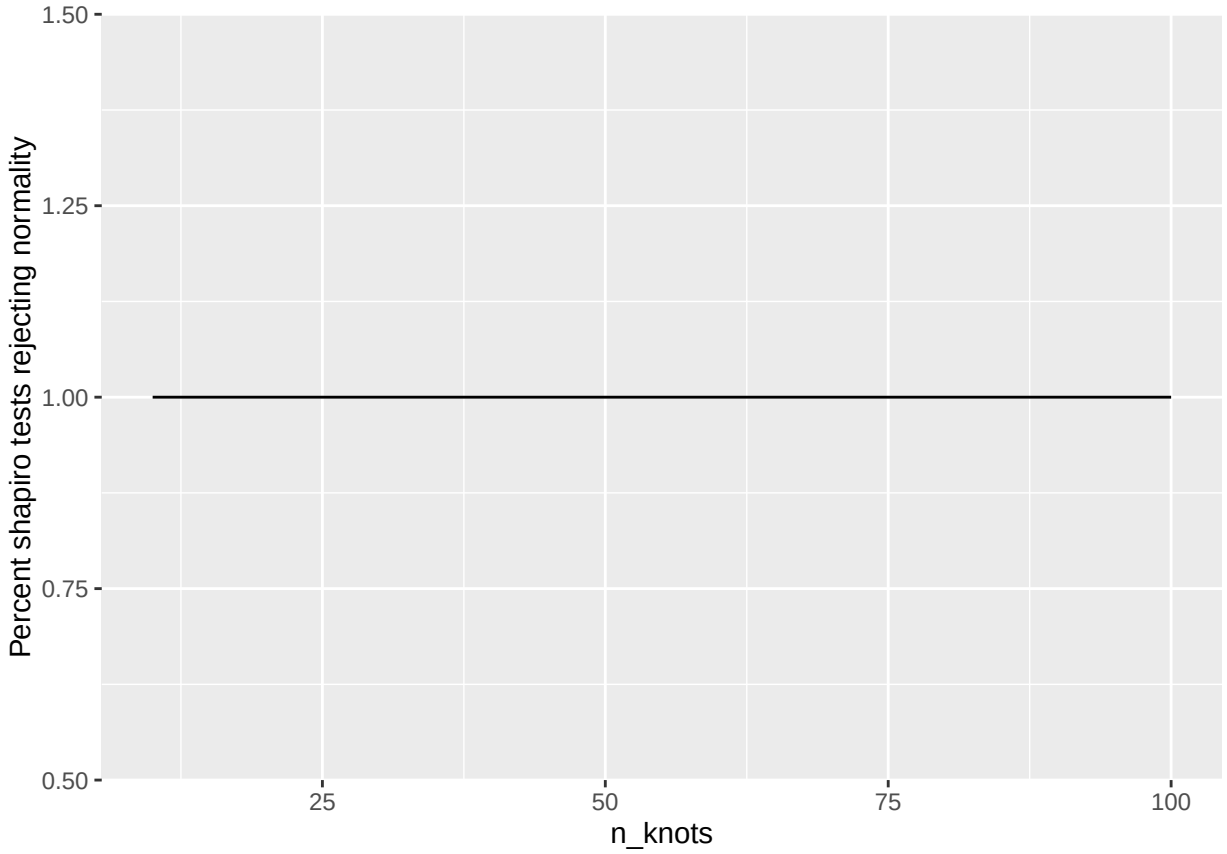
We can visually check to see that the simulation is working ok,



Do we need a minimum number of knots to not destroy skewness?

Part of the ability to recover the t distribution is making sure the number of knots approaches the number of data points. What's an acceptable ratio of knots : data? This isn't something we've been concerned with for the other spatial models to data, where everything is normal. At small values of knots, there's likely more averaging (thus the resolution is lost and the projected random effects become normal). We can explore this by doing a series of tests for normality (shapiro.test) on the projected values.

```
# The above example dataset has 1000 data points, so we can iterate 10:100 knots
df = expand.grid("n_knots" = seq(10, 100, by = 5), "normality" = 0)

for(i in 1:nrow(df)) {
  # simulate 1000 draws of posteriors -- with observation error equal to process error
  dataSim = simulateData(nKnots = df$n_knots[i])
  p.vals = unlist(lapply((apply(dataSim$proj, 2, shapiro.test)), getElement, "p.value"))
  df$normality[i] = length(which(p.vals < 0.05)) / length(p.vals)
}

ggplot(df, aes(n_knots, normality)) + geom_line() +
  ylab("Percent shapiro tests rejecting normality")
```



Including observation error

What ratios of observation : process error are appropriate without destroying the multivariate-t distribution? ### It seems likke as long as observation error standard deviation is < 2x process standard deviation. Increasing the number of knots also helps a little bit (hence the increasing trend for most cases)

```
# The above example dataset has 1000 data points, so we can iterate 10:100 knots
df = expand.grid("n_knots" = seq(10, 100, by = 10),
  "ratio_obs2pro" = c(1, 2, 3, 4, 5), "normality" = 0, "pro_sd" = 0)

for(i in 1:nrow(df)) {
  print(i)
  # simulate 1000 draws of posteriors -- with observation error equal to process error
  dataSim = simulateData(nKnots = df$n_knots[i])
  N = dim(dataSim$proj)[1] * dim(dataSim$proj)[2] # total number of random numbers to gen.
  df$pro_sd[i] = median(apply(dataSim$proj, 2, sd))
  SD = df$pro_sd[i] * df$ratio_obs2pro[i]
  obsError = matrix(rnorm(N, 0, SD), dim(dataSim$proj)[1], dim(dataSim$proj)[2])
  dataSim$proj = dataSim$proj + obsError
  p.vals = unlist(lapply((apply(dataSim$proj, 2, shapiro.test)), getElement, "p.value"))
  df$normality[i] = length(which(p.vals < 0.05)) / length(p.vals)
}

ggplot(df, aes(n_knots, normality, group = ratio_obs2pro, colour = pro_sd)) + geom_line() + facet_wrap(
```

Converting Multivariate-normal to Multivariate-t

The code for implementing the Multivariate-t distribution in JAGS and STAN relies on converting the Multivariate-normal distribution, rather than using built in functions.

The assumption is that two independent random variables are

$$z \sim MVN(0, \Sigma)$$
$$s \sim \chi^2_\nu(df)$$

Then the derived variable,

$$y = z \cdot \sqrt{\nu/s}$$

is Multivariate-t with parameters Σ and ν. Further details can be found in
https://en.wikipedia.org/wiki/Multivariate_t-distribution
https://journal.r-project.org/archive/2013-2/hofert.pdf

Problems of using the built in Multivariate-t distributions in JAGS and STAN can be found on respective forums,
JAGS: http://sourceforge.net/p/mcmc-jags/discussion/610037/thread/491d9ccc/?limit=25
STAN: https://groups.google.com/forum/#!topic/stan-users/0F0O4hfHA8g

Treatment of random effects

In all of the applications, we're assuming that a number of knots k is chosen to approximate the spatial field of the data, and the dimension of the knots is less than the number of data points n. The vector of random effects at the knots is denoted as x_*.

The random effects at the knot locations are multivariate normal,

$$x_* \sim MVN(0, \Sigma)$$

where the covariance matrix Σ_* is a square matrix ($kbyk$) representing the spatial covariance between knots. Elements of Σ_* are modeled by specifying a covariance function (exponential, Gaussian, Matern, etc) and using the distance between locations. For example, with a Gaussian covariance function,

$$\Sigma_{*i,j} = \sigma^2 \cdot exp\left(-\tau \cdot d^2_{i,j}\right)$$

where σ^2 is a scaling parameter, and τ is a shape parameter, relating the distance between knots $d_{i,j}$ to covariance.

The covariance matrix of knots, Σ_* can be viewed as the block matrix of a larger matrix, relating the covariances between the knots to the covariances between the locations where data are observed.

$$\Sigma = \begin{bmatrix} \Sigma_{obs} & \Sigma_{(obs,*)} \\ \Sigma_{(*,\quad obs)} & \Sigma_* \end{bmatrix}$$

Like above, each element of the covariance matrices are only dependent on the distance between locations, and any estimated parameters.

Projecting random effects to observed locations

To project the spatial random effects to the locations where we have observed data (similar to interpolation) We can use the approach of Latimer (2009, Ecol Lett) and others to only use Σ_* and $\Sigma_{(obs,*)}$

Given a vector of random effects at the knot locations, x_*, we can project to the data locations using the equation

$$x_{obs} = \Sigma'_{(obs,*)}\Sigma_*^{-1}x^*$$

And then we can combined those projected random effects x_{obs} with any covariate data at the observed locations to get the total predicted values.