Stats and Probability Information

Rob Hayward

October 27, 2014

Continuous and marginal distributions

The marginal distribution of x in a two-variable distribution is equal to the sum of the joint distribution over y.

$$Pr(X = x) = \sum_y Pr(X = x, Y = y) = \sum_y Pr(X = x|Y = y)Pr(Y = y) \tag{1}$$

From Wikipedia
For the continuous case

$$p_X(x) = \int_y p_{X,Y}(x, y)dy = \int_y p_{X|Y}(x|y)p_Y(y)dy \tag{2}$$

There are three related distributions: the marginal, the joint and the conditional.

0.1 Markov Chain Monte Carlo Methods

This comes from Dave Miles. There are four sections.

- Introduction
- Showing the MCMC works
- Example to extract the marginal posterior distribution
- Use R to implement MCMC

This is an update of the code by Economics by Simulation
The Gibbs sampler exploits the characteristics of the Markov chain. With two parameters θ_1 and θ_2, $p(\theta_1, \theta_2)$ is the prior pdf and $L(\theta_1, \theta_2|y) = p(y|\theta_1, \theta_2$

is the likelihood function. Using Bayes theory, the posterior pdf for the parameters is

$$p(\theta_1, \theta_2|y) \propto p(\theta_1, \theta_2)L(\theta_1, \theta_2|y) \tag{3}$$

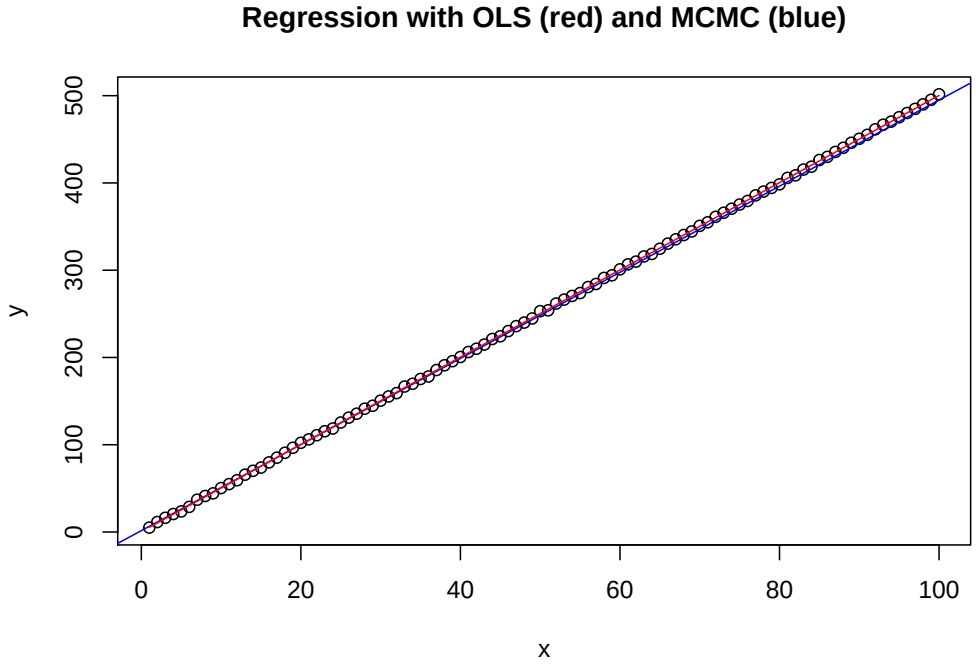
There are a numbrer of steps.

1. Assign initial values to $\theta_1^{(0)}$ and $\theta_2^{(0)}$
2. Draw a random value $\theta_1^{(1)}$ from $p(\theta_1|\theta_2^{(0)}, y)$
3. The draw a random value $\theta_2^{(1)}$ from $p(\theta_2|\theta_1^{(1)}, y)$
4. Draw a random value $\theta_1^{(2)}$ from $p(\theta_1|\theta_2^{(1)}, y)$
5. Repeat items 3 and 4

We end up a with two series that are Markov chains so the initial values do not matter and after many replications the chains start to behave as if they were random draws from the *marginal* posterior distibution $p(\theta_1|y)$ and $p(\theta_2|y)$ rather than the *conditional* posterior distribution $p(\theta_1|\theta_2, y)$ $p(\theta_1|\theta_2, y)$. The early values are part of the *burn in* period and should be discarded.

Here is a very simple version of the sampler applied to linear regression. A draw for the slope coefficient is made conditional upon the given intercept and then a new intercept is drawn conditional on the slope. This is repeated. I think that they should be random draws from the data.

```
# Create a simple MCMC sampler

x <- seq(1, 100, by = 1)
e <- rnorm(length(x))
y <- 0.5 + 5 * x + e
# prior
a <- rep(NA, times = length(x))
b <- rep(NA, times = length(x))
a[1] = 1
b[1] = 1
for(i in 2:100){
b[i] <- (y[i] - a[i-1] - rnorm(1))/x[i]
a[i] <-  y[i] - b[i]*x[i] - rnorm(1)
}
plot(y ~ x, type = 'p', main = "Regression with OLS (red) and MCMC (blue)")
abline(a = mean(a), b = mean(b), col = 'blue')
lines(fitted(lm(y ~x)), col = 'red')
```



Once there is a large sample of $p(\theta_1|y)$ and the burn-in have been discarded, the mean, variance, median or mode of the marginal posterior can be calculated. The process can be expanded to more parameters.

0.1.1 Gibbs Sampler

The Gibbs sampler is a special case of the Metropolis-Hasting algorithm. The R code from Dave Giles in the file DaveGilesMCMCcoes.R has better version than this. I am not sure how we got from $\rho = 0.5$ to $sd = 1 - \rho^2$.

This example is based on two random variables Y_1 and Y_2, with a mean vector of (μ_1, μ_2) a correlation of ρ, the variances of Y_1 and Y_2 are σ_1^2 and σ_2^2 respectively and the covariance between Y_1 and Y_2 is $\rho\sigma_1\sigma_2$. [1]

The conditional distribution of Y_1 given Y_2 is

$$p(Y_1|Y_2) \sim N\left(\mu_1 + \frac{\rho\sigma_1(Y_2 - \mu_2)}{\sigma_2}, \sigma_1^2(1-\rho^2)\right) \tag{4}$$

and the conditional distribution of Y_2 given Y_1 is

$$p(Y_2|Y_1) \sim N\left(\mu_2 + \frac{\rho\sigma_2(Y_1 - \mu_1)}{\sigma_2}, \sigma_1^2(1-\rho^2)\right) \tag{5}$$

[1] As $\rho = \frac{\sigma_{Y_1,Y_2}}{\sigma_1\sigma_2}$. Some additional reading on covariance http://www.cogsci.ucsd.edu/ desa/109/trieschmarksslides.pdf

This can be used to find the marginal distribution. It is known that the marginal distribution for Y_1 is

$$p(Y_1) \sim N(\mu_1, \sigma_1^2) \tag{6}$$

and for Y_2 is

$$p(Y_2) \sim N(\mu_2, \sigma_2^2) \tag{7}$$

However, to test the MCMC draw from the conditional to find the marginal (that we already know).

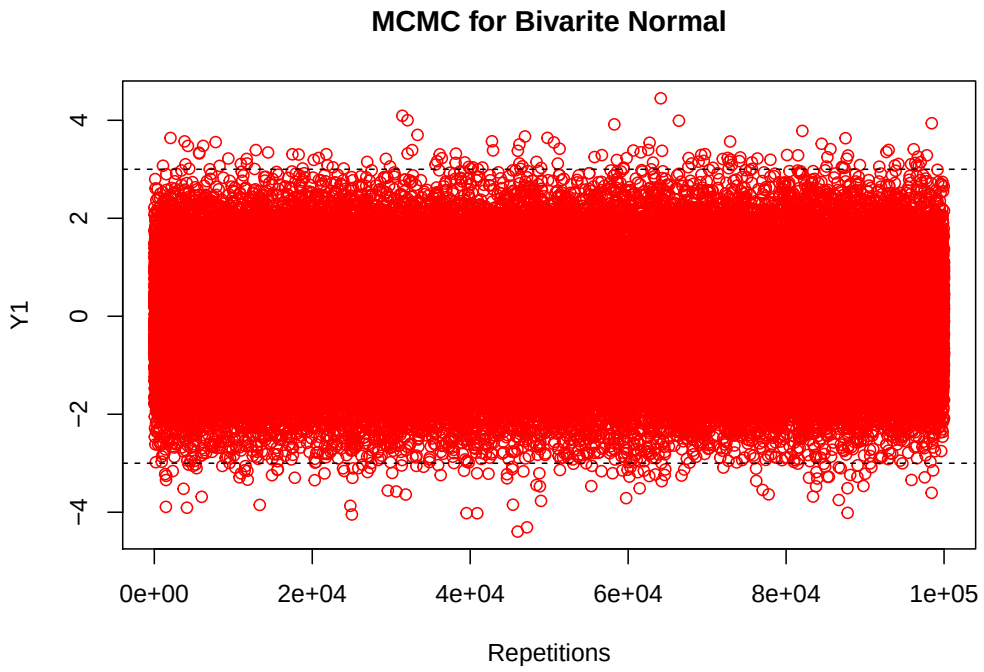
Set $\mu_1 = 0$ and $\mu_2 = 0$ and $\sigma_1 = 1$ and $\sigma_2 = 1$

The steps for the Gibbs sampler are

1. Chose an initial value for Y_1 called $Y_1^{(0)}$
2. Next, generate a random Y_2 value ($Y_2^{(0)}$) from $p(Y_2|Y_1^{(0)})$
3. Then, generte a new random Y_1 value (say $Y_1^{(1)}$) from $p(Y_1|Y_2^{(0)})$
4. Repeat steps 2 and 3 many times saving the strings of Y_1 and Y_2 values.
5. Throw away the first thousand or so values as they are draws from the *conditional distribution*
6. Now the values from the marginal distribution of interest

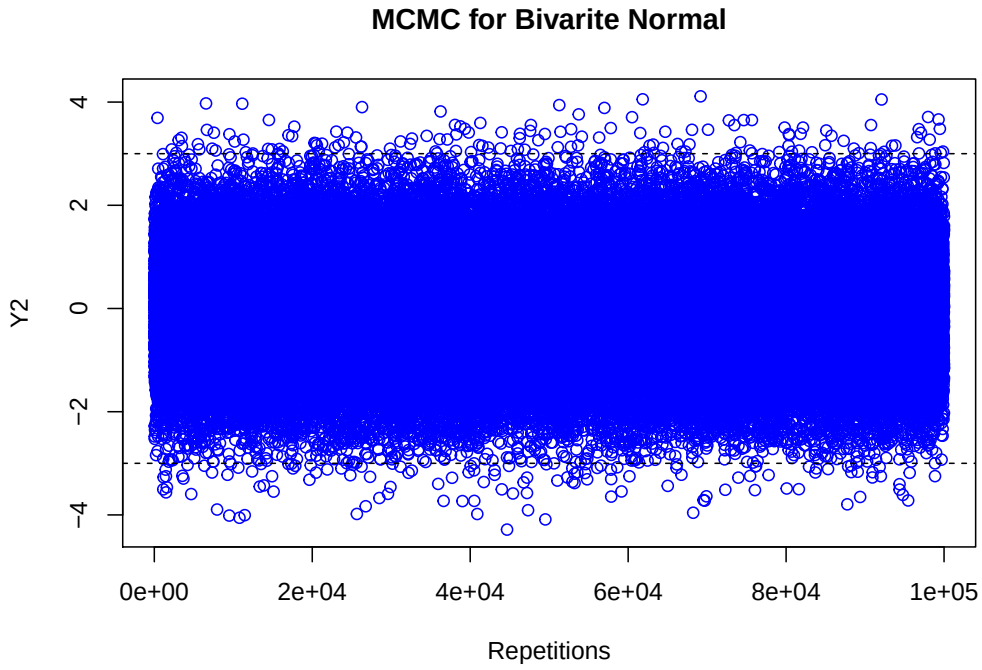
```
set.seed(123)
nreps <- 105000
nb <- 5000
yy1 <- array(, nreps)
yy2 <- array(, nreps)
rho <- 0.5
sd <- sqrt(1 - rho^2)
y1 <- rnorm(1, 0, sd)
for(i in 1:nreps){
  y2 <- rnorm(1, 0, sd) + rho*y1
  y1 <- rnorm(1, 0, sd) + rho*y2
  yy1[i] <- y1
  yy2[i] <- y2
}
nb1 <- nb + 1
yy1b <- yy1[nb1:nreps]
yy2b <- yy2[nb1:nreps]
```

```
plot(yy1b, col = 2, main = "MCMC for Bivarite Normal", xlab = "Repetitions",
     ylab = "Y1")
abline(h = 3, lty = 2)
abline(h = -3, lty = 2)
```



The values are centered around zero as they should be.

```
plot(yy2b, col = 4, main = "MCMC for Bivarite Normal", xlab = "Repetitions",
     ylab = "Y2")
abline(h = 3, lty = 2)
abline(h = -3, lty = 2)
```

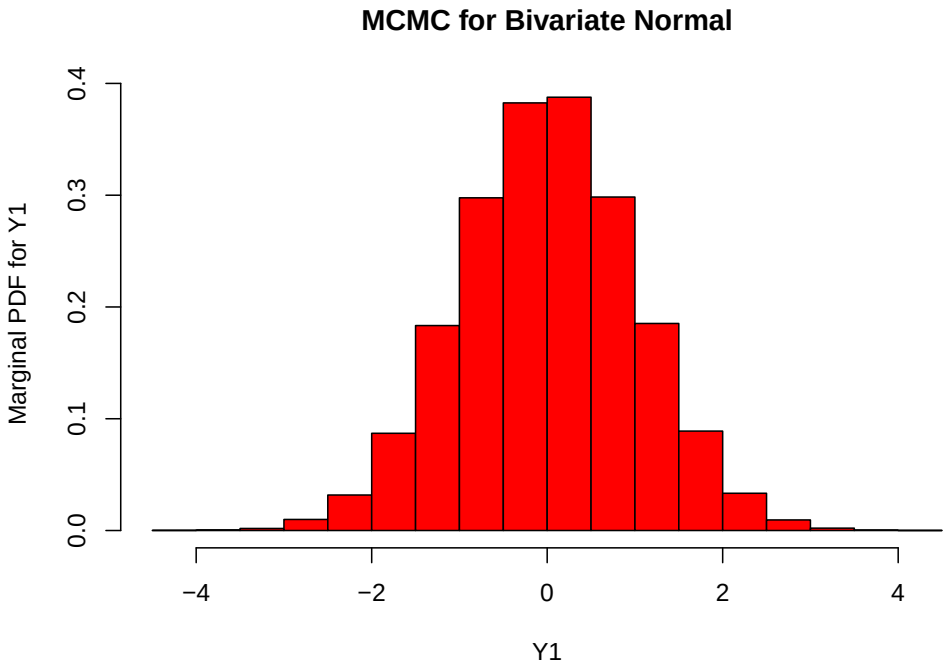


The summary statistics are below.

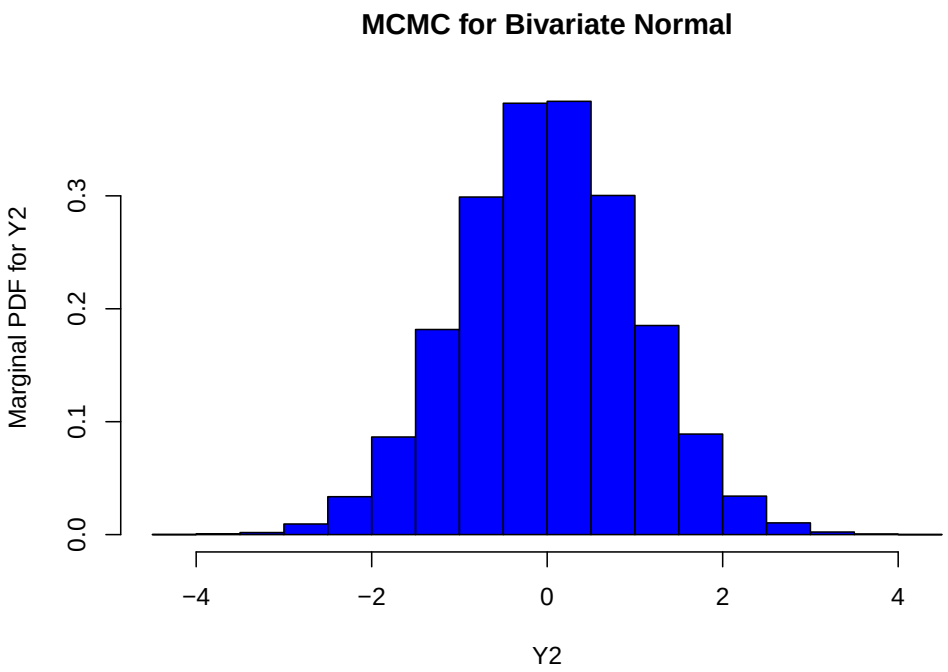
```
summary(yy1b)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -4.396000 -0.671400  0.007824  0.005148  0.678300  4.448000

var(yy1b)

## [1] 0.9954324

summary(yy2b)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -4.285000 -0.667900  0.007165  0.007131  0.680900  4.111000

var(yy2b)

## [1] 1.003804
```

The means are zero and the variances are close to unity.

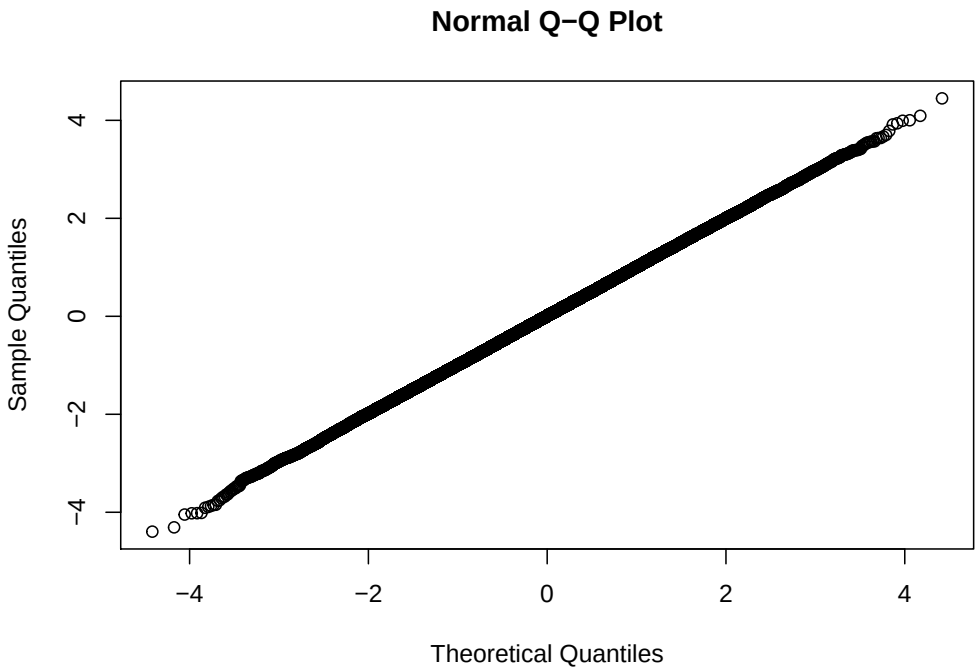
```
hist(yy1b, prob = T, col = 2, main = "MCMC for Bivariate Normal", xlab = "Y1",
     ylab = "Marginal PDF for Y1")
```



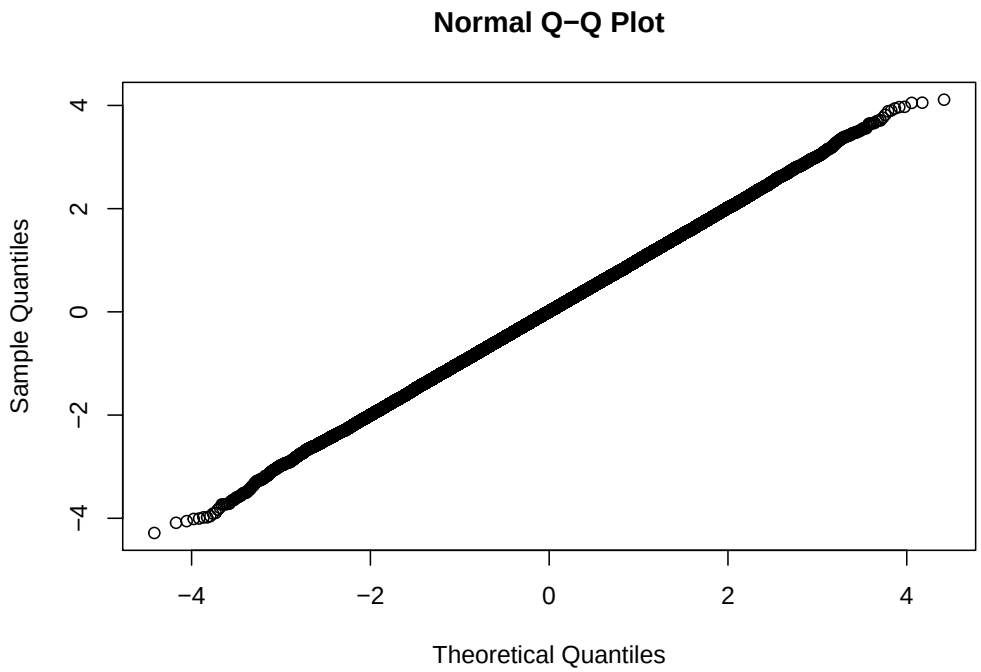

```
hist(yy2b, prob = T, col = 4, main = "MCMC for Bivariate Normal", xlab = "Y2",
     ylab = "Marginal PDF for Y2")
```


MCMC for Bivariate Normal
Marginal PDF for Y2
0.0
0.1
0.2
0.3
−4
−2
0
2
4
Y2

```
qqnorm(yy1b)
```


Normal Q−Q Plot
Sample Quantiles
−4
−2
0
2
4
−4
−2
0
2
4
Theoretical Quantiles

```r
qqnorm(yy2b)
```




```r
library(tseries)
jarque.bera.test(yy1b)

##
##  Jarque Bera Test
##
## data:  yy1b
## X-squared = 1.0699, df = 2, p-value = 0.5857

jarque.bera.test(yy1b)

##
##  Jarque Bera Test
##
## data:  yy1b
## X-squared = 1.0699, df = 2, p-value = 0.5857
```

0.1.2 Gibbs Sampler for Marginal Posterior

Extract the *marginal posterior distribution* from the *joint posterior distribution*. Assume that there is a population with an unknown mean μ and an unknown precistion parameter τ where $\tau = 1/\sigma^2$. Before looking at the data we are *a priori* ignorant about the possible values of the parameters, we assign the following joint pior density:

$$p(\mu, \tau) = p(\mu)p(\tau) \propto 1/\tau; \quad -\inf < \mu < \inf; \quad 0 < \tau < \inf \tag{8}$$

Now take a sample of n independent values from the population. This will give a likelihood function of

$$L(\mu, \tau|y) = p(y|\mu, \tau) \propto \tau^{n/2} exp[-(\tau/2)\sum(y_i - \mu)^2] \tag{9}$$

I do not understand the next step.

Dave says that by Bayes law the *joint* posterior density for the two parameters is

$$p(\mu, \tau|y) = [p(\mu, \tau)L(\mu, \tau|y)/p(y)] \propto p(\mu, \tau)L(\mu, \tau|y) \tag{10}$$

$$\propto \tau^{n/2-1} exp[-(\tau/2)\sum(y_i - \mu)^2] \tag{11}$$

For the marginal densities it is necessary to input the conditional posterior densities into the Gibbs sampler. From Equation 10, treat τ as a constant (so condition on τ),

$$p(\mu|\tau, y) \propto exp[-(\tau/2)\sum(y_i - \mu)^2] \tag{12}$$

$$\propto exp[-(\tau/2)(vs^2 + \sum(\mu - y^*)^2)] \tag{13}$$

$$\propto exp[-\tau/2)(\mu - y^*)^2] \tag{14}$$

where y^* is the sample mean of y; $v = (n - 1)$; and s^2 is the sample variance. Therefore, the conditional posterior for μ is a normal distribution with a mean of y^* and a variance of $(n\tau)^{-1}$.

Alternatively, if the conditioning is on μ

$$p(\tau|\mu, y) \propto \tau^{n/2-1} exp[-\tau(0.5\sum(y_i - \mu)^2)] \tag{15}$$

So the conditional posterior for τ is a Gammer distribution with a shape parameter, $r = (n/2)$ and a scale parameter, $\lambda = [0.5\sum(y_i - \mu)^2]$.

I do not understand the move to the conditional posterior and the distribution.

Now implement the Gibbs Sampler

```
set.seed(123)
# MC replications
nrep <- 105000
# burn-in
nb <- 5000
# Sample size
n <- 10
# set up vectors
tau <- array(, nrep)
mu <- array(, nrep)
# Create a sample of data.  true values are 1.
y <- rnorm(n, mean = 1, sd = 1)
ybar <- mean(y)
yy <- sum(y^2)
lambda <- 1/(0.5 * n * var(y))
```

Now create the loop for the Gibbs sampler

```
ttau <- rgamma(1, shape = n/2, rate = lambda)
for(i in 1:nrep){
  mmu <- rnorm(1, mean = ybar, sd = 1/sqrt(n*ttau))
  scale <- (0.5 * (yy + n * mmu^2 - 2 * n * mmu * ybar))
  ttau <- rgamma(1, shape = n/2, scale)
  tau[i] <- ttau
  mu[i] <- mmu
}
```

Now drop the first 5000 values of μ and τ for the burn in.

```
nb1 <- nb +1
taub <- tau[nb1:nrep]
mub <- mu[nb1:nrep]
```

Take a look at the distributions for μ and τ. They should be student-t (n-1) and Gamma respectively.

```
require(moments)

## Loading required package:  moments
##
```

```
## Attaching package:  'moments'
##
## The following objects are masked from 'package:GLDEX':
##
##    kurtosis, skewness
```

```
summary(mub); var(mub)
```

```
ybar # THe mean of the posterior for my should be ybar (1.0746)
```

```
skewness(mub) # The skewness of student-t is zero
```

```
kurtosis(mub) # THe excess kurtosis for student-t is 6/(n-5) = 1.2
```

```
summary(taub); var(taub)
```

```
skewness(taub) # the skewness of gamma is 2/sqrt(shape) = 0.8944
```

```
kurtosis(taub) # excess kkurtosis of gamma is 6/(n/2) = 1.2
```

The last couple of comments from Dave are:

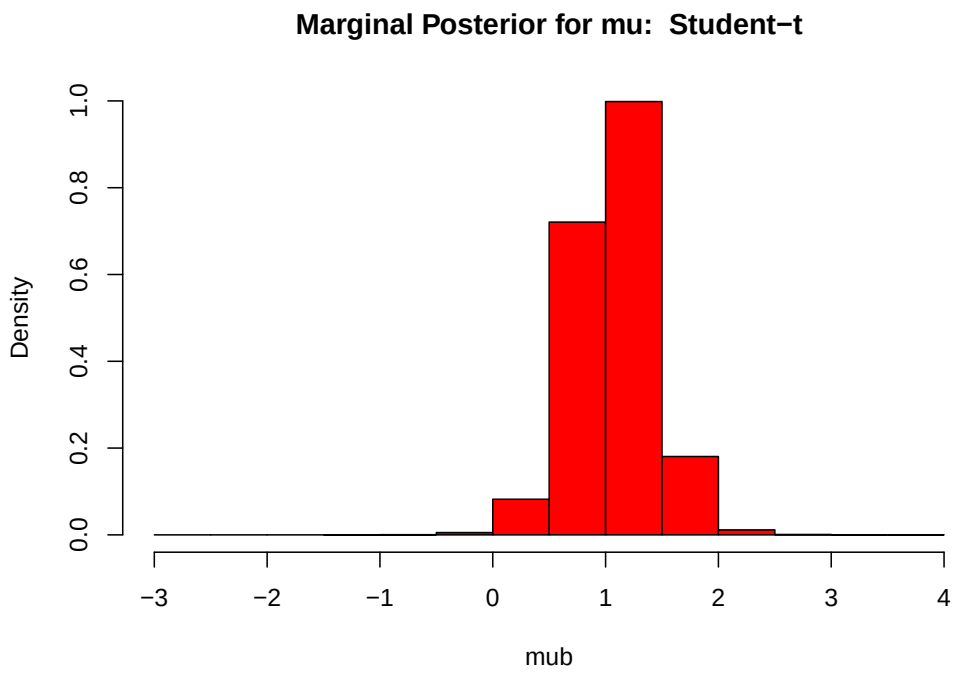
- The sample were drawn from a population with $\mu = \tau = 1$.
- With a quadratic loss function, the Bayes estimators of the parameters

would be the marginal posterior means (see above).

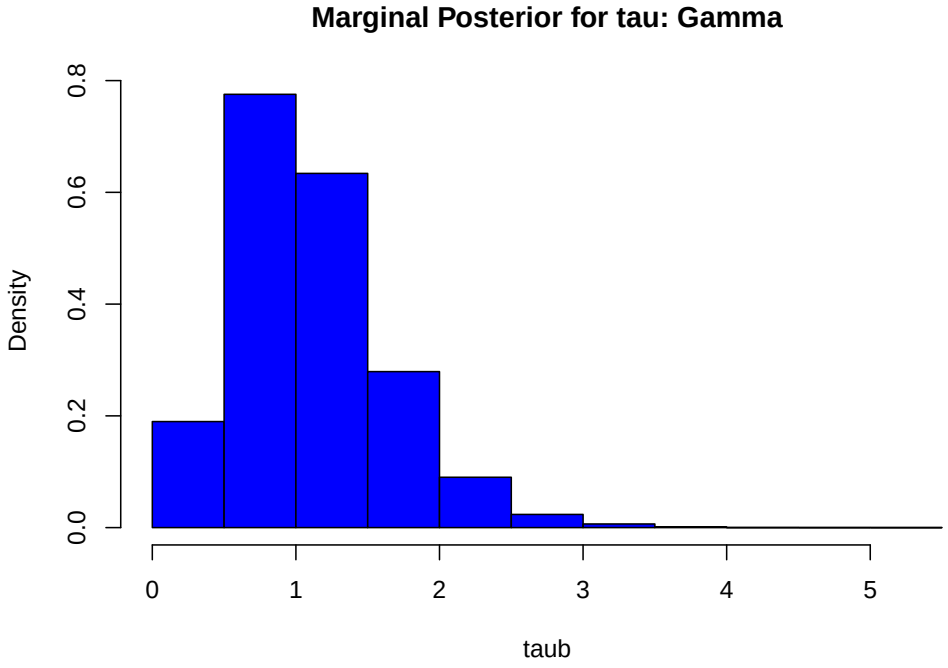
- With an absolute error loss function, the Bayes estimators would be the marginal posterior medians.

The marginal posterior densities.

```
hist(mub, main = "Marginal Posterior for mu:  Student-t", prob = TRUE, col = "re
```




```
hist(taub, main = "Marginal Posterior for tau: Gamma", prob = TRUE, col = "blue"
```



0.2 Mixture Model

This is a probabilistic model that relates some random variables to some other variables. The model has sub-populations. The properties of the sub-population are different from those of the parent. The sub-populations may not be observable. For example, the distribution of returns may be different in different sub-population or regime.

A *mixture distribution* is the probability distribution of a random variable whose values are derived from an underlying set of random variables. The *mixture components* are individual distributions with *mixture weights*. Even in cases where the mixture comonents have a normal distribution, the mixture distribution is likely to be non-normal. Mixture models are used to understand the sub-population when there is only access to the information about the pooled population.

The mixture model will be comprised of N random varibles distributed according to K components, with each component belonging to the same distribution. The k mixture weights sum to one. Each component will have parameters (mean and variance in the case of normal distribution).

The method will try to estimate the all the parameters of the model from the data. The underlying data is known (x_i); the number of mixture

components is set (K); the parameters of the distribution of each mixture component $(\theta_{i=1\ldots K})$; mixture weight $(\Phi_{i=1\ldots K})$; $\mathbf{\Phi}$ K-dimensional vector summing to 1; $F(x|\theta)$ probability distribution of observations parameterised on θ; α shared hyperparameter for component weights; β shared hyperparameter for mixture weights; $H(\theta|\alpha)$ prior probability distribution of component parameters;

0.3 Regression

This is a series of sessions that introduce regression with R. The first page is here. Linear Regression: Step-by-step. This is essentially a version of the Andrew Ng Machine Learning Course.

If Y is the dependent variable, $X_1, X_2, \ldots X_n$ are the explanatory variables and Θ are unknown parameters, regression takes can be carried out in a number of ways.

0.3.1 OLS

$$h_\Theta(x) = \Theta_0 + \Theta_1 x \tag{16}$$

The aim is to find values of Θ so that the model will fit the data. The *cost function* will assess the difference between the predicted and the actual values. One version of the cost function is

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(X^{(i)} - y^{(i)}))^2 \tag{17}$$

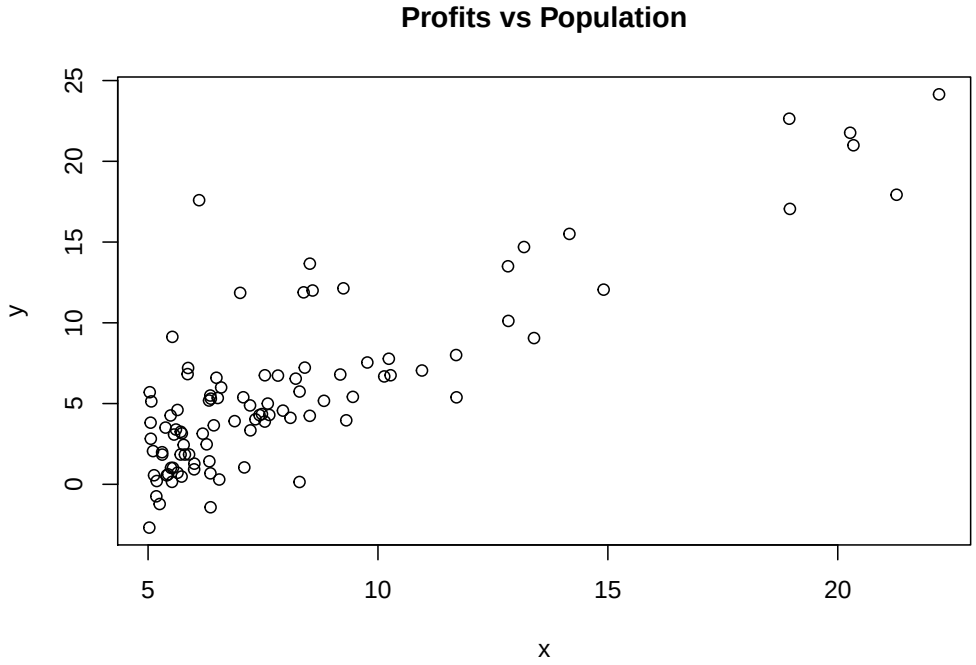
Change the values of Θ to minimise the cost. The method used is gradient decent.

The partital derivative of the cost function with respect to the Θ are

$$\Theta_0 = \Theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)}) \tag{18a}$$

$$\Theta_1 = \Theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})x^{(i)} \tag{18b}$$

```
data <- read.csv("../Data/Reg.csv")
y <- data$profit
x <- data$population
plot(y ~ x, main = "Profits vs Population")
```



The objective of the regression is to minimise the cost function.

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(X^{(i)} - y^{(i)}))^2 \tag{19}$$

Where the hypothesis is given by

$$h_\Theta(x) = \Theta^T x = \Theta_0 + \Theta_1 x_1 \tag{20}$$

To take account of the intercept Θ_0 an additional column of constants is added to x.

```
# Add ones to x
x <- cbind(1,x)
# Initialise theta vector
theta <- c(0,0)
# Number of observations
m <- nrow(x)
# Calculate the cost
cost <- sum(((x%*%theta) - y)^2)/(2*m)
cost

## [1] 32.07273
```

The initial value is 32.07 (how do I take this from the chunk?). Now

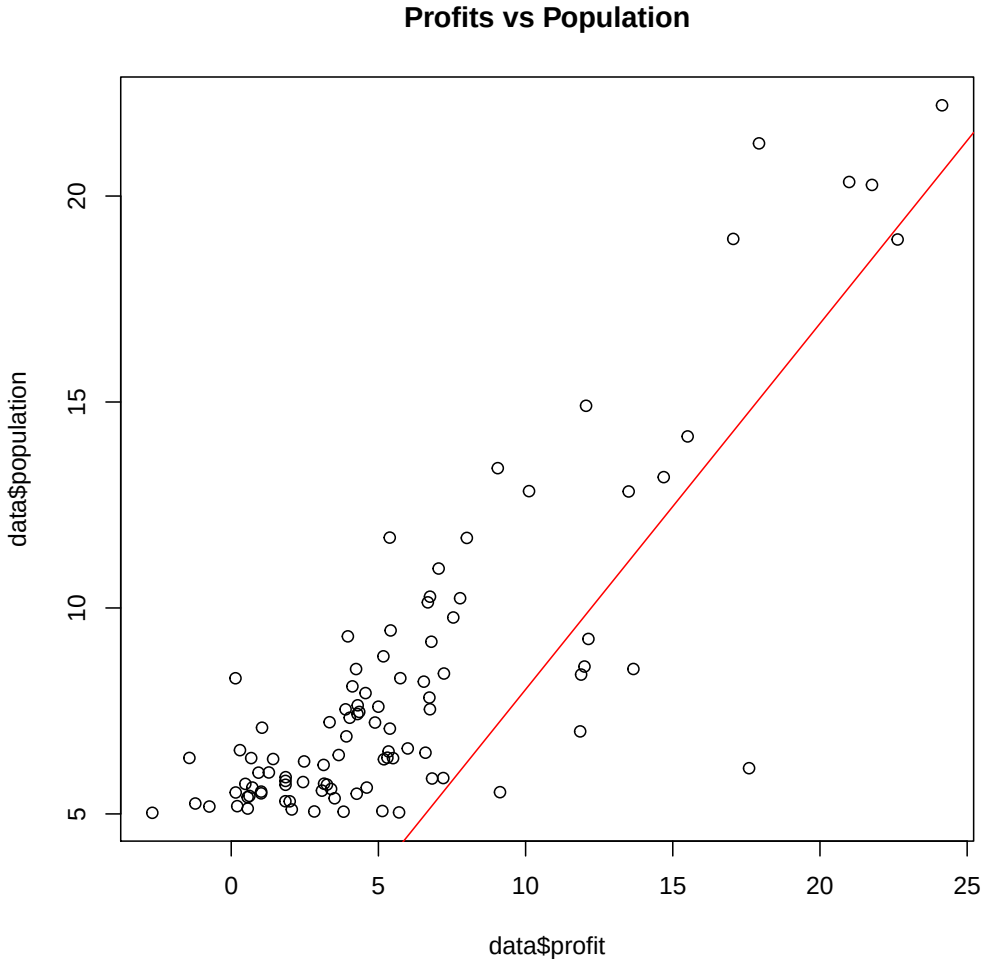
```
# Set learning parameter
alpha <- 0.001
#Number of iterations
iterations <- 1500
# updating thetas using gradient update
for(i in 1:iterations)
{
theta[1] <- theta[1] - alpha * (1/m) * sum(((x%*%theta)- y))
theta[2] <- theta[2] - alpha * (1/m) * sum(((x%*%theta)- y)*x[,2])
}
#Predict for areas of the 35,000 and 70,000 people
predict1 <- c(1,3.5) %*% theta
predict2 <- c(1,7) %*% theta
predict1
```

```
##          [,1]
## [1,] 2.246558
```

```
predict2
```

```
##          [,1]
## [1,] 5.355939
```

```
plot(data$population ~ data$profit, main = "Profits vs Population")
abline(theta, col = 'red')
```



I think it is right, but....Needs to be looked at.

0.4 Adjusted R squared

Adjusted R squared applied a penalty to the basic R squard to account for additional variables. The equartion is

$$R_A^2 = 1 - \left[\frac{(n-1)}{(n-k)}\right][1 - R^2] \tag{21}$$

Adding a regressor to the equation will increase (reduce)) the $R_A^"$ when the absolute value of the t-statistic is greater (less) than one. Adding a group of regressors to the model will reduce (increase) the $R_A^"$ when the absolute value of the F-statistic is greater than one.

Proof http://davegiles.blogspot.com/2014/04/proof-of-result-about-adjusted.html

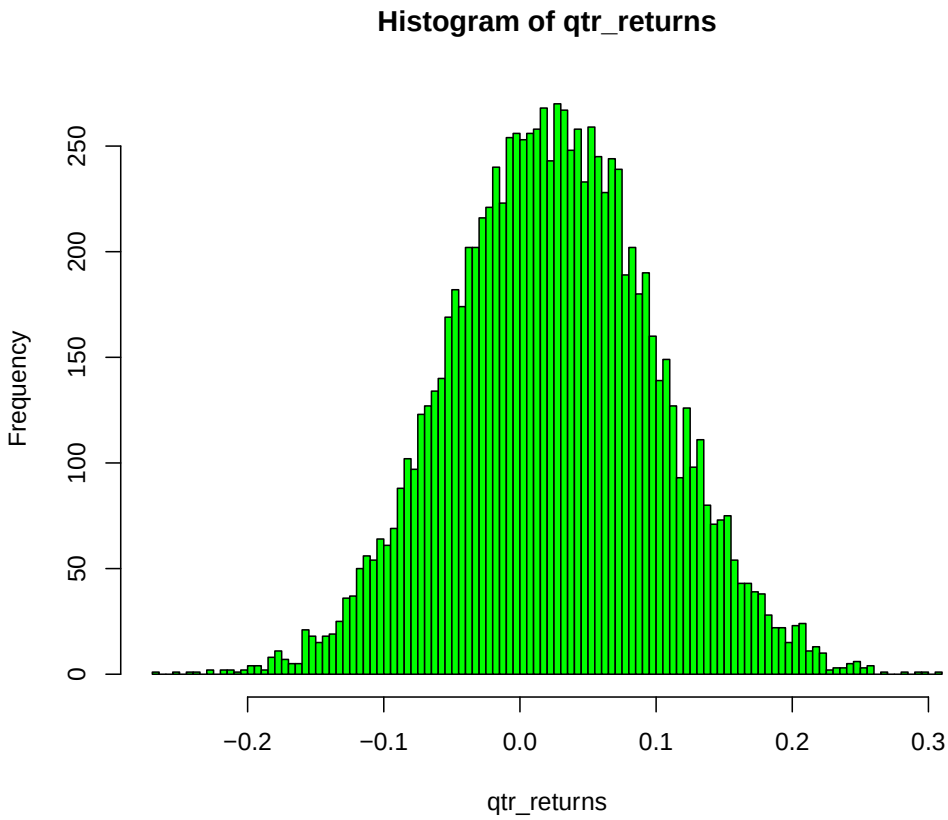
0.5 Monte Carlo Simulation

This comes from Revoluitionary Analytics. The analysis is in annual terms.

$$\mu\Delta t + \sigma Z\sqrt{\Delta t} \tag{22}$$

where μ is the drift or average annual return, Z is a standard Normal random variable, t is measured in years so for monthly returns Δt equals $\frac{1}{12}$.

```
n <- 10000
# Fixing the seed gives us a consistent set of simulated returns
set.seed(106)
z <- rnorm(n)         # mean = 0 and sd = 1 are defaults
mu <- 0.10
sd <- 0.15
delta_t <- 0.25
# apply to expression (*) above
qtr_returns <- mu*delta_t + sd*z*sqrt(delta_t)
hist(qtr_returns, breaks = 100, col = "green")
```



Now the descriptive statistics can be uncovered from the simulated results.

```
stats <- c(mean(qtr_returns) * 4, sd(qtr_returns) * 2)    # sqrt(4)
names(stats) <- c("mean", "volatility")
stats

##       mean volatility
## 0.09901252 0.14975805
```

This is the basic model. It would also be possible to simulate two variables and to include some relationship between the two in the analysis. It would also be possible to simulate an asset in two different regimes. A Monte-Carlo Markov Model (MCMM) would require another set of μ and σ inputs as well as a transition matrix of the probabilities that there is a switch from one regime to another.

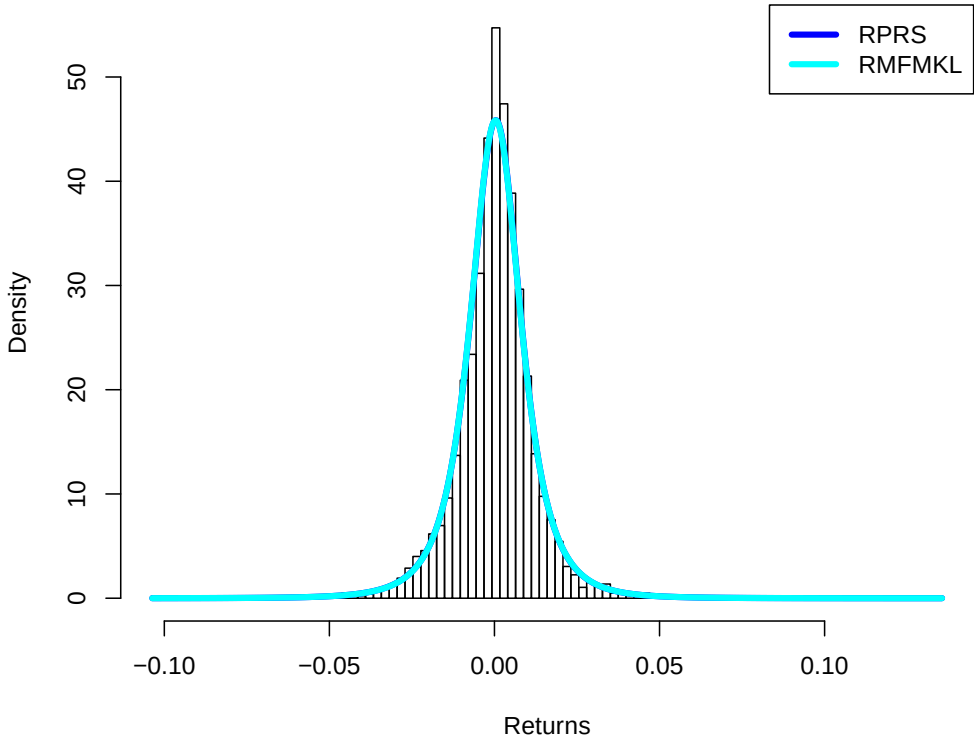
0.6 Generalised Lambda Distribution

This is from Revolutionary Analytics. The four parameters λ_1, λ_2, λ_3 and λ_4 indicate the location, scale, skew and kurtosis of the distribution.

```
require(GLDEX)
require(quantmod)

getSymbols("SPY", from = "1994-02-01")

## [1] "SPY"

SPY.Close <- SPY[,4] # Closing prices

SPY.vector <- as.vector(SPY.Close)

# Calculate log returns
sp500 <- diff(log(SPY.vector), lag = 1)
sp500 <- sp500[-1] # Remove the NA in the first position
# Set normalise="Y" so that kurtosis is calculated with
# reference to kurtosis = 0 under Normal distribution
fun.moments.r(sp500, normalise = "Y")

##          mean      variance      skewness      kurtosis
##  0.0002688808  0.0001504232 -0.0992342708  9.6630402623
```

Now fit the GLD with the function fun.data.fit.mm. There are warnings but these can be ignored.

```
spLambdaDist = fun.data.fit.mm(sp500)
spLambdaDist

##             RPRS     RMFMKL
## [1,]  4.213e-04  3.258e-04
## [2,] -3.432e+01  2.058e+02
## [3,] -1.685e-01 -1.706e-01
## [4,] -1.648e-01 -1.623e-01

fun.plot.fit(fit.obj = spLambdaDist, data = sp500, nclass = 100,
             param = c("rs", "fmkl"), xlab = "Returns")
```



Now it is possible to generrate simualted results using the function rgl(). Lambdas need to be identified.

```
lambda_params_rs <- spLambdaDist[, 1]
lambda1_rs <- lambda_params_rs[1]
lambda2_rs <- lambda_params_rs[2]
lambda3_rs <- lambda_params_rs[3]
lambda4_rs <- lambda_params_rs[4]

lambda_params_fmkl <- spLambdaDist[, 2]
lambda1_fmkl <- lambda_params_fmkl[1]
lambda2_fmkl <- lambda_params_fmkl[2]
lambda3_fmkl <- lambda_params_fmkl[3]
lambda4_fmkl <- lambda_params_fmkl[4]
```

Now generate simulations of each variety.

There are problems with the rgl function. I am not sure what this does. It is 10 million simulations. I think that the rgl just uses extra hardware to

make the change. It may be useful to re-do this last section using a different method.

```
require(gld)

## Loading required package:  gld
##
## Attaching package:  'gld'
##
## The following objects are masked from 'package:GLDEX':
##
##    dgl, pgl, qdgl, qgl, rgl, starship, starship.adaptivegrid,
##    starship.obj

require(GLDEX)
# RS version:
set.seed(100)     # Set seed to obtain a reproducible set
rs_sample <- rgl(n = 10000000, lambda1=lambda1_rs, lambda2 = lambda2_rs,
                 lambda3 = lambda3_rs,
                 lambda4 = lambda4_rs,param = "rs")

# Moments of simulated returns using RS method:
fun.moments.r(rs_sample, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002675628  0.0001506395 -0.1073842455 10.1150396924

# Moments calculated from market data:
fun.moments.r(sp500, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002688808  0.0001504232 -0.0992342708  9.6630402623

# FKML version:
set.seed(100)     # Set seed to obtain a reproducible set
fmkl_sample <- rgl(n = 100000, lambda1=lambda1_fmkl, lambda2 =
                   lambda4 = lambda4_fmkl,param = "fmkl")

# Moments of simulated returns using FMKL method:
fun.moments.r(fmkl_sample, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002446417  0.0001518191 -0.0890330751  8.6926187636
```

```
# Moments calculated from market data:
fun.moments.r(sp500, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002688808  0.0001504232 -0.0992342708  9.6630402623
```

Compare the moments to the S&P500 market data

```
fun.moments.r(rs_sample, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002675628  0.0001506395 -0.1073842455 10.1150396924

fun.moments.r(sp500, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002688808  0.0001504232 -0.0992342708  9.6630402623

fun.moments.r(fmkl_sample, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002446417  0.0001518191 -0.0890330751  8.6926187636

fun.moments.r(sp500, normalise="Y")

##          mean      variance      skewness      kurtosis
##  0.0002688808  0.0001504232 -0.0992342708  9.6630402623
```

0.7 Lasso method

Rob TibshiraniCancer example that requires identification of appropriate cell. There are 20 cases that are being used as a training set. Train a classifier to identify whether the cells are cencerous or not. There are 11,000 features. It would be useful to use as few of the features as possible. Therefore, also want to know which features are important for the classification.

Sparcity means that the features are reduced by only using those that pass a particular level of significance. More here.

0.8 Standard Error of the estimated mean

The standard error of the estimate of the mean is

$$SD_x = \frac{\sigma}{\sqrt{n}} \tag{23}$$

This can be derived from the variance of the sum of independent random variables

- If $X_1, X_2, \ldots, X_n$ are independent observations from a population with a mean μ and a standard deviation σ,
- Variance of the Total $= T = (X_1, X_2, \ldots, X_n)$ is $n\sigma^2$
- T/n is the mean $\bar{x}$
- the variance of T/n is $\frac{1}{n^2} n\sigma^2 = \frac{\sigma^2}{n}$
- Explained
- the standard deviation of T/n must be $\frac{n}{\sqrt{n}}$

0.9 Logistic Regression

This comes from Wim-Vector - logistic regression.

```
CarData <- read.table(url('http://archive.ics.uci.edu/ml/machine-learning-databa
logisticModel <- glm(rating!='unacc' ~ buying + maintenance + doors + persons +

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(logisticModel)

##
## Call:
## glm(formula = rating != "unacc" ~ buying + maintenance + doors +
##     persons + lug_boot + safety, family = binomial(link = "logit"),
##     data = CarData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2160   0.0000   0.0000   0.0257   2.4336
##
## Coefficients:
```

```
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -28.4255  1257.5255  -0.023    0.982
## buyinglow           5.0481     0.5670   8.904  < 2e-16 ***
## buyingmed           3.9218     0.4842   8.100 5.49e-16 ***
## buyingvhigh        -2.0662     0.3747  -5.515 3.49e-08 ***
## maintenancelow      3.4064     0.4692   7.261 3.86e-13 ***
## maintenancemed      3.4064     0.4692   7.261 3.86e-13 ***
## maintenancevhigh   -2.8254     0.4145  -6.816 9.36e-12 ***
## doors3              1.8556     0.4042   4.591 4.41e-06 ***
## doors4              2.4816     0.4278   5.800 6.62e-09 ***
## doors5more          2.4816     0.4278   5.800 6.62e-09 ***
## persons4           29.9652  1257.5256   0.024    0.981
## personsmore        29.5843  1257.5255   0.024    0.981
## lug_bootmed        -1.5172     0.3758  -4.037 5.40e-05 ***
## lug_bootsmall      -4.4476     0.4750  -9.363  < 2e-16 ***
## safetylow         -30.5045  1300.3428  -0.023    0.981
## safetymed          -3.0044     0.3577  -8.400  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2110.47  on 1727  degrees of freedom
## Residual deviance:  339.36  on 1712  degrees of freedom
## AIC: 371.36
##
## Number of Fisher Scoring iterations: 21
```

The variable levels and values are joined together. The model will provide an estimate of the effect of each category on the rating. The values for each category are adde together to get the overall rating score.

The error results of the model can be examined.

```
table(CarData$rating,predict(logisticModel,type='response')>=0.5)
```

```
##
##          FALSE TRUE
##   acc       32  352
##   good       0   69
##   unacc   1166   44
##   vgood      0   65
```

To be completed.

0.10 F-Tests

This is an example that is based on testing for fabricated evidence. The full documentation is Deborah Mayo. As the paper says

> "In each experiment, participants (undergraduate students) were randomly assigned to three groups, and each group was given a different intervention. All participants were then tested on some outcome measure."

The accusation is that the results are **too** linear. The relationships are too perfect compared to other studies. The method used to asses this is called *delta-F*. The odds of seeing such linear trends are calculated based on the assumption that the trend is linear.

Unless there is a huge sample, the probability of obtaining a linear trend is very low becuase there is noise. THe amount of noise is evident in the *within group variance*. For a given sample size and a given level of within group variance, the odds of obtaining a linear trend are calculated as the sum of squares accounted for by a linear model and a none-linear model (one-way ANOVA) divided by the mean-square error (within group variance).

$$\Delta F = \frac{SS_{REG} - SS_B}{MS_W} \tag{24}$$

There is one degree of freedom in the numerator and $3(n-1)$ degrees of freedom in the denominator.

If the difference between the two models (linear and non-linear) is small, it means that there is an underlying linear relationship. Assuming that the relationship is linear, this delta-F metric should follow an F distribution.

This is more or less the same method that is used to test whether a simple model fits the data as well as a complex model. In that case, the null hypothesis is that the simple model is the correct version and the objective is to determine if the difference between the two is unlikely given the null.

From the paper

> "'But here the whole thing is turned on its head. Random noise means that a complex model will sometimes fit the data better than a simple one, even if the simple model describes reality. In a conventional use of F-tests, that would be regarded as a false positive. But in this case its the absence of those false positives thats unusual.'"

0.11 Wilcoxon rank Sum

This comes from SAS and R. Using the Wilcoxon Rank-sum test is often used as a test of medias. However, it should really be a test of whether a random number drawn from one distibution would be above a random number drawn from the other. For this to be a test of medians, there is an additional assumption that the distrubtions are the same. This is rarely true or acknowledged.

For example,

```
y1 <- rexp(1000)
y2 <- rnorm(1000) + log(2)
# two sets of random numbers with the same median
wilcox.test(y1, y2)

##
##  Wilcoxon rank sum test with continuity correction
##
## data:  y1 and y2
## W = 549208, p-value = 0.0001386
## alternative hypothesis: true location shift is not equal to 0
```

The null is rejected.

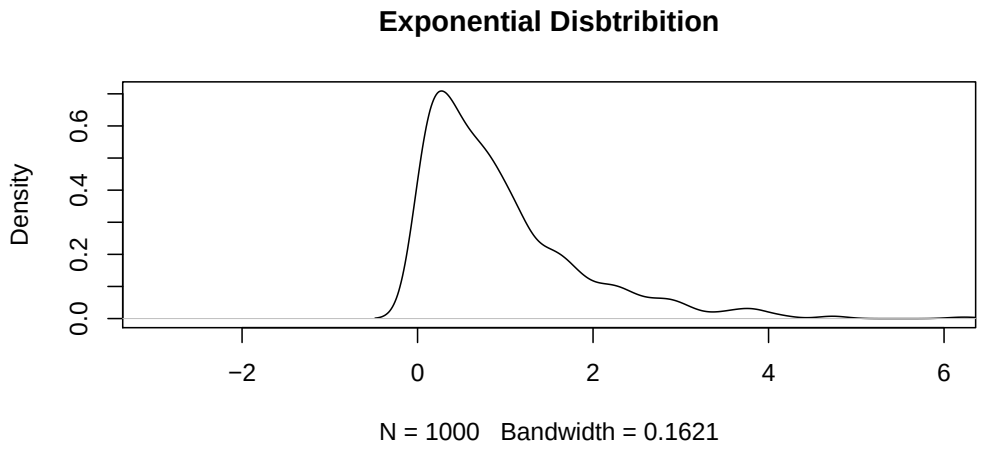
Can use a quantile regression where there are two series with the same median but different distributions.

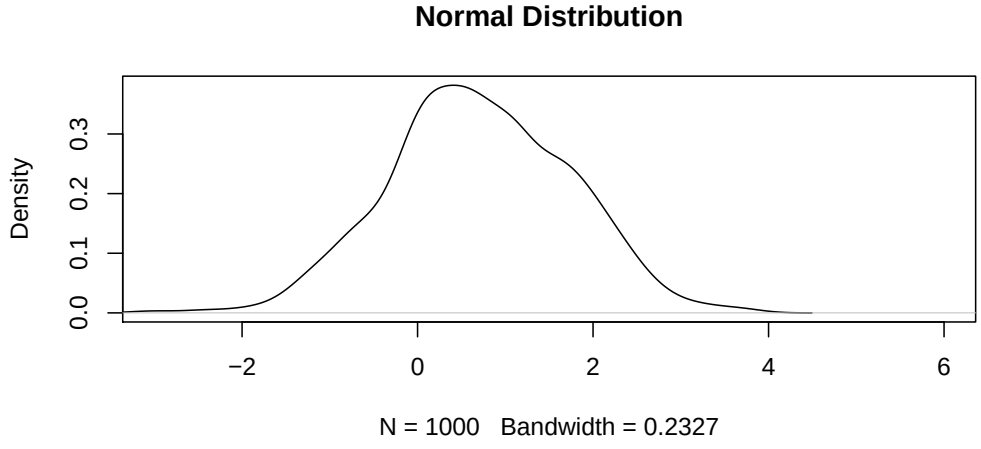
```
library(quantreg)

## Loading required package:  SparseM
##
## Attaching package:  'SparseM'
##
## The following object is masked from 'package:base':
##
##    backsolve

y = c(y1, y2)
c = rep(1:2, each = 1000)
summary(rq(y ~ as.factor(c)))

##
## Call: rq(formula = y ~ as.factor(c))
```

```
##
## tau: [1] 0.5
##
## Coefficients:
##                Value    Std. Error t value  Pr(>|t|)
## (Intercept)     0.70697  0.02823   25.04214  0.00000
## as.factor(c)2 -0.04460  0.05118   -0.87141  0.38364
```

Fails to reject the null of equal medians. See text.

```
par(mfrow = c(2,1))
plot(density(y1), xlim = c(-3, 6), main = "Exponential Disbtribition")
plot(density(y2), xlim = c(-3, 6), main = "Normal Distribution")
```





1 Probability Distributions

This comes from Are you cereal?. The range of fucntions are s follows.

- probbility density function
- cumulative function
- quntile function
- random number genrator

It starts with the equation for the probability density function. The result can be deived (analytically or numerically) from there.

This is an example with an *exponential function*.

1.1 probability density function

- probbility density is relative density
- at a given dat point, pdf is likelihood
- pdf can be greter than one.

$$p(x) = \lambda e^{-\lambda x}, \quad x \in [0, \text{inf}] \tag{25}$$

1.2 Chisquared test of independence

This comes from Arthur Charpentier.

Given the following table

```
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    68   20    15     5
##   Brown   119   84    54    29
##   Red      26   17    14    14
##   Blond     7   94    10    16
```

To test for independence between eye colour and hair colour

$$P_{i,j} = ¶(X = i, Y = j) = \frac{N_{i,j}}{n} \tag{26}$$

must be equal to

$$P^{\perp}_{i,j,} = ¶(X = i, Y = j) = \frac{N_{i.}}{n}$$

N$_j n$(27)For each pair, under the assumption independence, there should be the following joint probabilities.

```
n=sum(N)
NHair=apply(N,1,sum)
NEye =apply(N,2,sum)
Nind=NHair%*%t(NEye)/n
Nind/n
```

```
##           Brown       Blue      Hazel      Green
## [1,] 0.06779584 0.06625502 0.02865915 0.01972243
## [2,] 0.17953342 0.17545311 0.07589367 0.05222790
## [3,] 0.04456949 0.04355654 0.01884074 0.01296567
## [4,] 0.07972288 0.07791100 0.03370104 0.02319211
```

This gives the following counts

```
Nind
```

```
##          Brown      Blue    Hazel     Green
## [1,]  40.13514  39.22297 16.96622 11.675676
## [2,] 106.28378 103.86824 44.92905 30.918919
## [3,]  26.38514  25.78547 11.15372  7.675676
## [4,]  47.19595  46.12331 19.95101 13.729730
```

The distance is the Chi-squared distance in the two contingency tables.

$$\sum_{i,j} \frac{(N_{i,j} - N^{\perp}_{i,j})^2}{N^{\perp}_{i,j}} \tag{28}$$

```
X = (N-Nind) ^ 2 / Nind
Qobs=sum(X)
Qobs
```

```
## [1] 138.2898
```

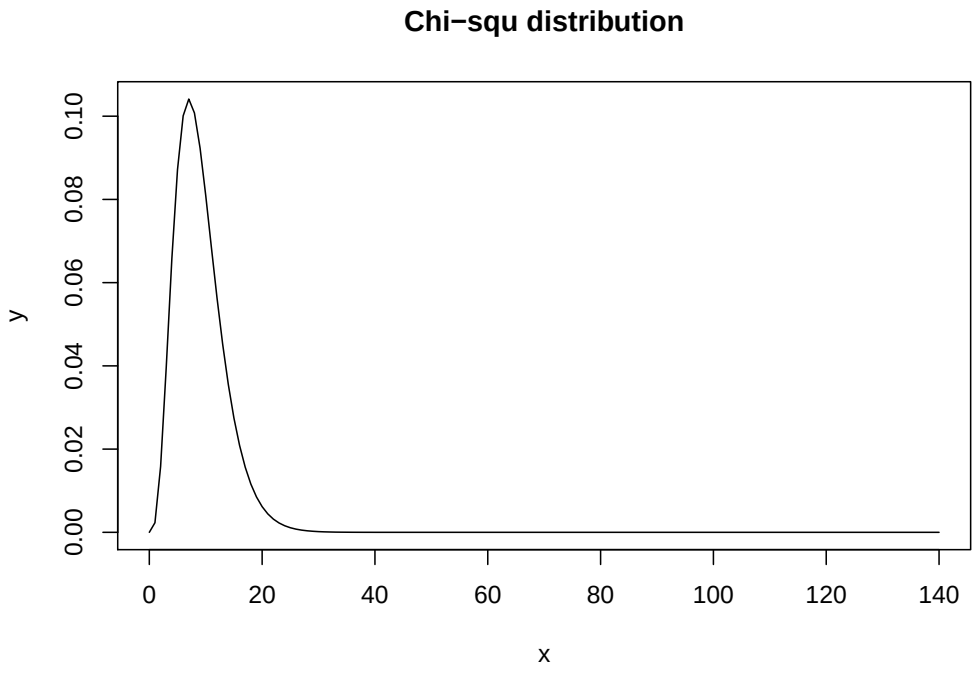
```
X
```

```
##        Eye
## Hair           Brown         Blue        Hazel        Green
##   Black 19.345909546  9.421078055  0.227864962  3.816879379
##   Brown  1.521418876  3.800459864  1.831377537  0.119093744
##   Red    0.005621691  2.993334093  0.726334723  5.210886943
##   Blond 34.234170713 49.696722274  4.963290205  0.375399021
```

Under the assumption of idependence

$$\sum_{i,j} \frac{(N_{i,j} - N_{i,j}^{\perp})^2}{N_{i,j}^{\perp}} \chi^2((I-1)(J-1)) \tag{29}$$

```
x = seq(0,140)
y = dchisq(x,df=(ncol(N)-1)*(nrow(N)-1))
plot(x, y, type = 'l', main = "Chi-squ distribution")
```



1.3 Statistical testing

This is from Arthur CHarpentier. The *modus tollens* says that if A implies B, not A impies not B. In words, if smoke implies fire, no smoke implies no fire.

Given the dataset

```
X = c(0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1)
```

where X_i follow a $B(p_*)$ law, test

$$H_0 : P_* = 0.5, \text{against} \quad H_1 : P_* \neq 0.5 \tag{30}$$

If H_0 is true (and there is enough data for asymptotic distribution),

$$T = \sqrt{n}[2\bar{X} - 1] \tag{31}$$

must follow a $N(0, 1)$. Here A is the H_0 and B is the standard normal distribution of the test statistic T. If the T is in the *rejection region*, we reject $T \sim N(0, 1)$. There is a one in twenty chanvce of doing that if it is true. If we reject B, we reject A. However, if we are not in the rejection region, we accept B and this means that we cannot reject A. There is no way that B can imply A.

```
sqrt(20)*(2*mean(X)-1)
```

```
## [1] 0.4472136
```

```
prop.test(sum(X), 20)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  sum(X) out of 20, null probability 0.5
## X-squared = 0.05, df = 1, p-value = 0.8231
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.3204804 0.7617145
## sample estimates:
##    p
## 0.55
```