Experiments in Imputation

Mean substitution

```
################################################################################
# Imputation of missing values can inflate Type I error probability.
# Imputation is a form of measurement error. If missing values are imputed
# for a variable for which one is "controlling" in a regression, the result
# should be the usual disaster. The values will be missing completely at random.
# Model is   Y  = beta0 + beta1 X1 + beta2 X2 + epsilon
# Seek to test H0: beta2=0
# Values of X1 are missing with probability p1.
# W1 is X1 with the imputed values.
# Values of X2 are missing with probability p2.
# W2 is X2 with the imputed values.
# We can play with different types of imputation.
# Fit the model Y  = beta0 + beta1 W1 + beta2 W2 + epsilon
# Inputs are
#
#   beta0, beta1 beta2      True regression coefficients
#   sigmasq                 Var(epsilon)
#   mu1                     E(X1)
#   mu2                     E(X2)
#   phi11                   Var(X1)
#   phi22                   Var(X2)
#   phi12                   Cov(X1,X2)
#   p1                      Probability X1 is missing
#   p2                      Probability X2 is missing
#   n                       Sample size

> rm(list=ls())
> options(scipen=999) # To avoid scientific notation!
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
>
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.10; p2=0; n=200
>
> # Simulate a data set
> set.seed(9999)
>
> # Try one simulation
>
> epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
> X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
> X1 <- X[,1]; X2 <- X[,2]
> Y = beta0 + beta1*X1 + beta2*X2 + epsilon
> # Generate and impute missing values
> m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
> W1 = X1; W1[m1==1] = NA
> imp1 = mean(W1,na.rm=TRUE) # Or median, same syntax
> W1[is.na(W1)] = imp1 # Omit this line to leave missing values in place
> W2 = X2; W2[m2==1] = NA
> imp2 = mean(W2,na.rm=TRUE) # Or median, same syntax
> W2[is.na(W2)] = imp2 # Omit this line to leave missing values in place
> # Test H0: beta2=0
> summary(lm(Y~W1+W2))
```

```
Call:
lm(formula = Y ~ W1 + W2)

Residuals:
     Min       1Q   Median       3Q      Max
-2.21553 -0.40132 -0.08324  0.43008  2.09375

Coefficients:
            Estimate Std. Error t value            Pr(>|t|)
(Intercept)  1.01854    0.05038  20.216 <0.0000000000000002 ***
W1           0.94227    0.08584  10.977 <0.0000000000000002 ***
W2           0.09200    0.08296   1.109                0.269
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7091 on 197 degrees of freedom
Multiple R-squared:  0.6637,  Adjusted R-squared:  0.6603
F-statistic: 194.4 on 2 and 197 DF,  p-value: < 0.00000000000000022

> ttable = summary(lm(Y~W1+W2))$coefficients; pval = ttable[3,4]
> pval
[1] 0.268826

> ############################################################
>
> # Put simulation code in a loop
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation!
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.10; p2=0; n=200
> M = 10000 # Monte Carlo sample size
>
> set.seed(9999) # Default will be NULL
> simp = numeric(M) # Will hold simulated p-values
>
> for(sim in 1:M)
+ {
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
+ X1 <- X[,1]; X2 <- X[,2]
+ Y = beta0 + beta1*X1 + beta2*X2 + epsilon
+ # Generate and impute missing values
+ m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
+ W1 = X1; W1[m1==1] = NA
+ imp1 = mean(W1,na.rm=TRUE) # Or median, same syntax
+ W1[is.na(W1)] = imp1 # Omit this line to leave missing values in place
+ W2 = X2; W2[m2==1] = NA
+ imp2 = mean(W2,na.rm=TRUE) # Or median, same syntax
+ W2[is.na(W2)] = imp2 # Omit this line to leave missing values in place
+ # Test H0: beta2=0
+ ttable = summary(lm(Y~W1+W2))$coefficients; pval = ttable[3,4]
+ simp[sim] = pval
+ } # Next sim
> mean(simp<0.05)  # T and F will be converted to 1 and 0
[1] 0.5881
```

Results were similar with various settings, and using median substitution instead of mean.

Random substitution

```
>
> ###############################################################
> # Now impute randomly selected x values -- should not affect
> # variance of x variables.
> ###############################################################

> rm(list=ls())
> options(scipen=999) # To avoid scientific notation!
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.10; p2=0; n=200
> M = 10000 # Monte Carlo sample size
>
> set.seed(9999) # Default will be NULL
> simp = numeric(M) # Will hold simulated p-values
>
> for(sim in 1:M)
+ {
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
+ X1 <- X[,1]; X2 <- X[,2]
+ Y = beta0 + beta1*X1 + beta2*X2 + epsilon
+ # Generate and impute missing values
+ m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
+ nmiss1 = sum(m1); nmiss2 = sum(m2)
+ W1 = X1; W1[m1==1] = NA
+ W2 = X2; W2[m2==1] = NA
+ urn1 = subset(W1,is.na(W1)==F); urn2 = subset(W2,is.na(W2)==F)
+ imp1 = sample(urn1,nmiss1,replace=T) # These values will replace NAs
+ imp2 = sample(urn2,nmiss2,replace=T) # These values will replace NAs
+ W1[is.na(W1)] = imp1 # Omit this line to leave missing values in place
+ W2[is.na(W2)] = imp2 # Omit this line to leave missing values in place
+ # Test H0: beta2=0
+ ttable = summary(lm(Y~W1+W2))$coefficients; pval = ttable[3,4]
+ simp[sim] = pval
+ } # Next sim
> mean(simp<0.05) # Proportion significant
[1] 0.8921
```

Single Nearest Neighbour

```
> ######################################################################
>
> # Imputation based on prediction
> # knn is k nearest neighbour
> # One nearest neighbour
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation!
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.25; p2=0; n=200
> M = 10000 # Monte Carlo sample size
>
>
> # Simulate a data set
> set.seed(9999)
>
> simp = numeric(M) # Will hold simulated p-values
>
> for(sim in 1:M)
+ {
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
+ X1 <- X[,1]; X2 <- X[,2]
+ Y = beta0 + beta1*X1 + beta2*X2 + epsilon
+ # Generate missing values
+ m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
+ W1 = X1; W1[m1==1] = NA
+ W2 = X2; W2[m2==1] = NA
+
+ # Impute: Missing w1 gets w1 value of case with closest w2
+ id = 1:n
+ lost = id[is.na(W1)]
+ for(j in lost)
+     {
+     diff = abs(W2-W2[j])
+     diff[lost] = 99 # These differences will not be small
+     closeid = id[diff==min(diff)] # id of obs with closest
+     W1[j] = W1[closeid]
+     }
+ # Test H0: beta2=0
+ ttable = summary(lm(Y~W1+W2))$coefficients; pval = ttable[3,4]
+ simp[sim] = pval
+ } # Next sim
> mean(simp<0.05) # Proportion significant

[1] 0.539
```

Regression imputation

```
> ######################################################################
>
> # Try "x-hat"
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation!
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
>
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.25; p2=0; n=200
> M = 1000 # Monte Carlo sample size
>
> # Simulate a data set
> set.seed(9999)
> simp = numeric(M) # Will hold simulated p-values
>
> for(sim in 1:M)
+ {
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
+ X1 <- X[,1]; X2 <- X[,2]
+ Y = beta0 + beta1*X1 + beta2*X2 + epsilon
+ # Generate missing values
+ m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
+ W1 = X1; W1[m1==1] = NA
+ W2 = X2; W2[m2==1] = NA
+
+ # Impute: Missing w1 gets w-hat based on w2
+ hatmod = lm(W1~W2)
+ id = 1:n
+ lost = id[is.na(W1)]
+ w2 = data.frame(W2[lost]) # W2 values for which W1 is missing
+ colnames(w2) = 'W2'; w2
+ what = predict.lm(hatmod,newdata=w2); # what
+ W1[lost] = what
+ # Test H0: beta2=0
+ ttable = summary(lm(Y~W1+W2))$coefficients; pval = ttable[3,4]
+ simp[sim] = pval
+ } # Next sim
> mean(simp<0.05) # Proportion significant
[1] 0.034
```

Wow, that is promising! Around alpha = 0.03 or 0.04 for rho = 0.80Maybe 0.06 for rho = 0.25.

It could be less appealing if there are lots of variables and a fair amount of missing data, and so just a few complete cases on which to base the regression.

K Nearest Neighbours (Knn)

```
>
> ######################################################################
>
> # Knn with the  DMwR package
> # First just show how it works
>
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # install.packages("DMwR")
> library(DMwR)        #   knnImputation
Loading required package: lattice
Loading required package: grid
>
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.30; p2=0; n=20
>
> # Simulate a small data set
> set.seed(8888)
>
> epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
> X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
> X1 <- X[,1]; X2 <- X[,2]
> Y = beta0 + beta1*X1 + beta2*X2 + epsilon
> # Generate missing values
> m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
> W1 = X1; W1[m1==1] = NA
> W2 = X2; W2[m2==1] = NA
>
> cbind(W1,W2,Y)
               W1         W2           Y
 [1,]          NA -0.3269726 -0.04905072
 [2,] -1.1125958 -0.6190641  0.57754888
 [3,]          NA  0.9995644  2.33532087
 [4,] -1.5265171 -1.5426527 -0.54374267
 [5,] -1.6303883 -1.6740215 -0.61475296
 [6,]  0.8593282  0.6772969  1.35828855
 [7,]          NA  0.3741558  0.81342576
 [8,]  0.6286941  1.2760061  2.65615537
 [9,]  1.0842937  1.7939975  1.97335077
[10,]  0.1537810  1.0594522  0.84474271
[11,] -1.9346400 -1.5432669 -1.30047738
[12,]          NA -0.4899029  0.92245559
[13,]  1.4282238  0.4574312  1.97570334
[14,]          NA  0.2173583  2.26130444
[15,] -0.9976633 -1.1753153  0.06504238
[16,]  1.5679188  0.4748617  3.31988415
[17,]  1.4540424  1.6824024  2.70961959
[18,]          NA  0.9306230  1.76600850
[19,]  0.6204834  0.6737340  2.24264618
[20,] -1.3428314 -1.5652113 -0.15896703
```

```
>
> # Try imputing W1 based on W2 only
> testr = data.frame(cbind(W1,W2)); head(testr)
          W1         W2
1         NA -0.3269726
2 -1.1125958 -0.6190641
3         NA  0.9995644
4 -1.5265171 -1.5426527
5 -1.6303883 -1.6740215
6  0.8593282  0.6772969
> knnImputation(testr)
Error in rep(1, ncol(dist)) : invalid 'times' argument
>
> # Impute based on Y as well, which seems criminal
> datta = data.frame(cbind(W1,W2,Y)); # head(datta)
> datta = knnImputation(datta); datta
           W1         W2           Y
1  -0.7818284 -0.3269726 -0.04905072
2  -1.1125958 -0.6190641  0.57754888
3   0.8712994  0.9995644  2.33532087
4  -1.5265171 -1.5426527 -0.54374267
5  -1.6303883 -1.6740215 -0.61475296
6   0.8593282  0.6772969  1.35828855
7   0.3349904  0.3741558  0.81342576
8   0.6286941  1.2760061  2.65615537
9   1.0842937  1.7939975  1.97335077
10  0.1537810  1.0594522  0.84474271
11 -1.9346400 -1.5432669 -1.30047738
12 -0.5271302 -0.4899029  0.92245559
13  1.4282238  0.4574312  1.97570334
14  0.8138628  0.2173583  2.26130444
15 -0.9976633 -1.1753153  0.06504238
16  1.5679188  0.4748617  3.31988415
17  1.4540424  1.6824024  2.70961959
18  0.7929561  0.9306230  1.76600850
19  0.6204834  0.6737340  2.24264618
20 -1.3428314 -1.5652113 -0.15896703
>
> # With the default settings, I believe knnImputation standardizes the variables,
> # and uses a weighted sum of the 10 nearest neighbours, with the weight inversely
> # proportional to Euclidian distance.
```

```
> ######################################################################
> # Simulation with knnImputation from the DMwR package
> # Impute W1 based on W2 -- and Y!
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # install.packages("DMwR")
> library(DMwR)        #   knnImputation
Loading required package: lattice
Loading required package: grid
>
> # Set inputs
> beta0=1; beta1=1; beta2=0; sigmasq = 0.5
> mu1=0; mu2=0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12),
+             c(phi12,phi22))
> p1=0.30; p2=0; n=200
> M = 1000 # Monte Carlo sample size
>
> # Simulation
> set.seed(9999)
> simp = numeric(M) # Will hold simulated p-values
>
> for(sim in 1:M)
+ {
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X <- rmvn(n, mu=c(mu1,mu2), sigma=Phi) # nx2 matrix
+ X1 <- X[,1]; X2 <- X[,2]
+ Y = beta0 + beta1*X1 + beta2*X2 + epsilon
+ # Generate missing values
+ m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2) # Missing value indicators
+ W1 = X1; W1[m1==1] = NA
+ W2 = X2; W2[m2==1] = NA
+
+ datta = data.frame(cbind(W1,W2,Y)); # head(datta)
+ datta = knnImputation(datta)
+
+ # Test H0: beta2=0
+ ttable = summary(lm(Y~W1+W2, data=datta))$coefficients; pval = ttable[3,4]
+ simp[sim] = pval
+ } # Next sim
> mean(simp<0.05) # Proportion significant = 0.273
[1] 0.273
>
>
```

```
> ######################################################################
> # Three explanatory variables, X3 uncorrelated with anything
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # install.packages("DMwR")
> library(DMwR)        #   knnImputation
Loading required package: lattice
Loading required package: grid
>
> # Set inputs
> beta0=1; beta1=1; beta2=0; beta3=0;  sigmasq = 0.5
> mu1=0; mu2=0; mu3 = 0
> phi11=1; phi22=1; phi12 = 0.80
> Phi = rbind(c(phi11,phi12,0),
+             c(phi12,phi22,0),
+             c(   0 ,  0  ,1)  )
> n=200
> p1=0.30; p2=0; p3=0
>
> # Simulation
> M = 1000 # Monte Carlo sample size
> set.seed(9999)
> simp = numeric(M) # Will hold simulated p-values
>
> for(sim in 1:M)
+ {
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X <- rmvn(n, mu=c(mu1,mu2,mu3), sigma=Phi) # nx3 matrix
+ X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]
+ Y = beta0 + beta1*X1 + beta2*X2 + beta3*X3 + epsilon
+ # Generate missing values
+ m1 = rbinom(n,1,p1); m2 = rbinom(n,1,p2); m3 = rbinom(n,1,p3)
+ W1 = X1; W1[m1==1] = NA
+ W2 = X2; W2[m2==1] = NA
+ W3 = X3; W3[m3==1] = NA
+
+ W = cbind(W1,W2,W3); head(W)
+ W = knnImputation(W,k=10,meth='median')
+ datta = data.frame(cbind(W,Y)); # head(datta)
+ # Test H0: beta2=0
+ ttable = summary(lm(Y~W1+W2+W3, data=datta))$coefficients; pval = ttable[3,4]
+ simp[sim] = pval
+ } # Next sim
> mean(simp<0.05) # Proportion significant
[1] 0.196


# Ran it again with n = 500, got
> mean(simp<0.05) # Proportion significant
[1] 0.234

# With beta3 = 1, Corr(X1,X3) = Corr(X2,X3) = 0.5, still n=500
> mean(simp<0.05) # Proportion significant
[1] 0.186
```

```
> ######################################################################
>
> # A bigger example with 5 explanatory variables.
> # Constant correlation between explanatory variables
> # Same probability of missing for all variables.
> # Same nonzero value of beta1-beta4
> # Test beta5 when true beta5 value = 0
>
> # First produce a single small data set and take a look
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # install.packages("DMwR")
> library(DMwR)        #   knnImputation
Loading required package: lattice
Loading required package: grid
>
> # Set inputs
> n = 20
> beta0=5; beta1=beta2=beta3=beta4 = 0.5
> beta5 = 0
> sigmasq = 1;
> rho = .75 # Correlation between explanatory variables
> mu1=mu2=mu3=mu4=mu5 = 3
> Sigma = matrix(rho,5,5); diag(Sigma) = 1; Sigma
     [,1] [,2] [,3] [,4] [,5]
[1,] 1.00 0.75 0.75 0.75 0.75
[2,] 0.75 1.00 0.75 0.75 0.75
[3,] 0.75 0.75 1.00 0.75 0.75
[4,] 0.75 0.75 0.75 1.00 0.75
[5,] 0.75 0.75 0.75 0.75 1.00
> probmiss = 0.20
> cat("\nWith probmiss = ", probmiss, " the probability of a complete case is",
+ (1-probmiss)^5, "\n" )

With probmiss =  0.2  the probability of a complete case is 0.32768
>
> # Simulate a data set
> set.seed(9999)
>
> epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
> X = rmvn(n, mu=c(mu1,mu2,mu3,mu4,mu5), sigma=Sigma) # nx5 matrix
> X1 = X[,1]; X2 = X[,2]; X3 = X[,3]; X4 = X[,4]; X5 = X[,5]
> Y = beta0 + beta1*X1 + beta2*X2 + beta3*X3 + beta4*X4 + beta5*X5 + epsilon
>
> cbind(X,Y)
                                                          Y
 [1,] 2.966357 2.456798 2.441560 2.399692 2.536024 11.216302
 [2,] 2.784790 3.319396 3.598142 3.385462 4.054478 12.387004
 [3,] 2.705898 3.444172 4.187505 4.317215 3.521014 12.821784
 [4,] 2.624332 3.413221 2.536280 2.243120 2.634343  9.635460
 [5,] 2.603306 3.125400 2.232340 1.947468 2.798160 12.858073
 [6,] 2.551025 3.366282 2.591804 2.465453 2.926967 11.396165
 [7,] 2.027664 1.481639 3.020892 1.968750 2.431269  8.286396
 [8,] 5.202803 4.951620 5.336306 5.164585 4.585880 15.139368
 [9,] 2.751356 2.430483 2.094493 2.421976 1.147022 11.269529
[10,] 2.020488 3.767879 3.616579 3.001200 2.895737 12.189511
[11,] 2.857869 2.679230 3.475215 4.190114 3.144460 11.151274
[12,] 2.861586 2.925745 3.039758 3.095637 4.268098 12.033029
[13,] 2.365571 2.596419 2.049507 1.549590 2.241077  8.863972
[14,] 3.425100 3.142778 3.630749 3.319110 2.745293 11.329729
[15,] 1.391107 2.769090 2.209901 1.766838 1.729544  8.951809
[16,] 5.043442 3.253465 3.646349 4.287308 4.436354 12.816744
```

```
[17,] 4.445520 3.578295 4.309919 3.745911 3.930245 12.860584
[18,] 3.389424 2.968791 3.020457 3.246019 4.085396 13.645248
[19,] 3.068546 2.478116 3.037484 3.276084 3.814461 10.631170
[20,] 3.696176 4.538447 4.123893 3.869397 3.991984 14.848158
>
> # Generate missing values
> miss = rbinom(5*n,1,probmiss); dim(miss) = c(n,5); miss
      [,1] [,2] [,3] [,4] [,5]
 [1,]    0    0    0    0    1
 [2,]    1    0    0    0    0
 [3,]    0    0    0    0    1
 [4,]    0    0    0    0    0
 [5,]    0    0    0    0    0
 [6,]    1    0    0    1    0
 [7,]    1    1    1    1    0
 [8,]    0    0    0    0    0
 [9,]    0    0    0    0    0
[10,]    0    1    0    0    1
[11,]    0    0    0    0    1
[12,]    0    0    1    0    0
[13,]    0    1    0    0    0
[14,]    1    0    0    1    0
[15,]    1    0    0    0    0
[16,]    0    0    1    1    0
[17,]    0    0    0    0    0
[18,]    0    0    0    0    0
[19,]    0    0    0    0    0
[20,]    0    0    0    0    0
> W = X
> colnames(W) = c("W1","W2","W3","W4","W5")
> W[miss==1] = NA; W
           W1       W2       W3       W4       W5
 [1,] 2.966357 2.456798 2.441560 2.399692       NA
 [2,]       NA 3.319396 3.598142 3.385462 4.054478
 [3,] 2.705898 3.444172 4.187505 4.317215       NA
 [4,] 2.624332 3.413221 2.536280 2.243120 2.634343
 [5,] 2.603306 3.125400 2.232340 1.947468 2.798160
 [6,]       NA 3.366282 2.591804       NA 2.926967
 [7,]       NA       NA       NA       NA 2.431269
 [8,] 5.202803 4.951620 5.336306 5.164585 4.585880
 [9,] 2.751356 2.430483 2.094493 2.421976 1.147022
[10,] 2.020488       NA 3.616579 3.001200       NA
[11,] 2.857869 2.679230 3.475215 4.190114       NA
[12,] 2.861586 2.925745       NA 3.095637 4.268098
[13,] 2.365571       NA 2.049507 1.549590 2.241077
[14,]       NA 3.142778 3.630749       NA 2.745293
[15,]       NA 2.769090 2.209901 1.766838 1.729544
[16,] 5.043442 3.253465       NA       NA 4.436354
[17,] 4.445520 3.578295 4.309919 3.745911 3.930245
[18,] 3.389424 2.968791 3.020457 3.246019 4.085396
[19,] 3.068546 2.478116 3.037484 3.276084 3.814461
[20,] 3.696176 4.538447 4.123893 3.869397 3.991984
>
> # Impute missing values
> W = knnImputation(W)
Error in knnImputation(W) :
  Not sufficient complete cases for computing neighbors.
> W = knnImputation(W,k=1,meth='median')
Error in rep(1, ncol(dist)) : invalid 'times' argument
>
```

```
>
> ########################################################################
>
> # Simulate a bigger example with 5 explanatory variables.
> # Constant correlation between explanatory variables
> # Same probability of missing for all variables.
> # Same nonzero value of beta1-beta4
> # Test beta5 when true beta5 value = 0
>
> rm(list=ls())
> options(scipen=999) # To avoid scientific notation
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/rmvn.txt")
> # install.packages("DMwR")
> library(DMwR)        #   knnImputation
Loading required package: lattice
Loading required package: grid
>
> # Set inputs
> n = 200
> beta0=5; beta1=beta2=beta3=beta4 = 0.5
> beta5 = 0
> sigmasq = 1;
> rho = .75 # Correlation between explanatory variables
> mu1=mu2=mu3=mu4=mu5 = 3
> Sigma = matrix(rho,5,5); diag(Sigma) = 1; Sigma
     [,1] [,2] [,3] [,4] [,5]
[1,] 1.00 0.75 0.75 0.75 0.75
[2,] 0.75 1.00 0.75 0.75 0.75
[3,] 0.75 0.75 1.00 0.75 0.75
[4,] 0.75 0.75 0.75 1.00 0.75
[5,] 0.75 0.75 0.75 0.75 1.00
> probmiss = 0.20
> cat("\nWith probmiss = ", probmiss, " the probability of a complete case is",
+ (1-probmiss)^5, "\n" )

With probmiss =  0.2  the probability of a complete case is 0.32768
>
>
> # Simulation
> M = 1000 # Monte Carlo sample size
> set.seed(9999)
> simp = numeric(M) # Will hold simulated p-values
>
> # With varying probability, the knnImputation function crashes. This
> # can happen even with k=1 when there clearly is a nearest neighbour
> # to all the missing values. It appears to depend on the pattern of missing
> # values. It never seems to crash when there is a variable with no
> # missing values. To get around the problem, trap the errors and keep trying
> # until the desired number of simulations are completed.
>
```

```
> simtry = goodsim=0
>
> while(goodsim < M)
+ {
+ simtry = simtry+1
+ epsilon <- rnorm(n,mean=0,sd=sqrt(sigmasq))
+ X = rmvn(n, mu=c(mu1,mu2,mu3,mu4,mu5), sigma=Sigma) # nx5 matrix
+ X1 = X[,1]; X2 = X[,2]; X3 = X[,3]; X4 = X[,4]; X5 = X[,5]
+ Y = beta0 + beta1*X1 + beta2*X2 + beta3*X3 + beta4*X4 + beta5*X5 + epsilon
+ # cbind(X,Y)
+ # Generate missing values
+ miss = rbinom(5*n,1,probmiss); dim(miss) = c(n,5); # miss
+ W = X
+ colnames(W) = c("W1","W2","W3","W4","W5")
+ W[miss==1] = NA; # W
+ # sum(complete.cases(W))
+ # Impute missing values
+ W = try(knnImputation(W),silent=TRUE)
+ if(is.matrix(W)) # If W is not a matrix, there was an error
+     {
+     datta = data.frame(cbind(W,Y))
+     # Fit the regression model
+     model = lm(Y ~ W1+W2+W3+W4+W5, data = datta); summary(model)
+     # Test H0: beta5=0
+     pval = summary(model)$coefficients[6,4]
+     goodsim = goodsim+1
+     simp[goodsim] = pval
+     } # End if it worked
+ } # Next simulation
> mean(simp<0.05) # Proportion significant
[1] 0.061
> cat("It took", simtry, "tries to reach", M, "successful simulations. \n")
It took 3612 tries to reach 1000 successful simulations.
>
```

========== More results

With probmiss = 0.10,

```
> mean(simp<0.05) # Proportion significant
[1] 0.061
> cat("It took", simtry, "tries to reach", M, "successful simulations. \n")
It took 1273 tries to reach 1000 successful simulations.
```

With probmiss = 0.20,

```
> mean(simp<0.05) # Proportion significant
[1] 0.062
> cat("It took", simtry, "tries to reach", M, "successful simulations. \n")
It took 22810 tries to reach 1000 successful simulations.
```

With probmiss = 0.10, meth = 'median'

```
> mean(simp<0.05) # Proportion significant
[1] 0.072
> cat("It took", simtry, "tries to reach", M, "successful simulations. \n")
It took 3600 tries to reach 1000 successful simulations.
```

With probmiss = 0.10, k=10, n=2,000
Waited quite a while and then killed it.

Another Approach for linear regression

$$\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\Sigma}}_x^{-1} \widehat{\boldsymbol{\Sigma}}_{xy}$$

$$\frac{1}{n} SSE \stackrel{a.s.}{\to} Var(y_i) - \boldsymbol{\Sigma}_{xy}^\top \boldsymbol{\Sigma}_x^{-1} \boldsymbol{\Sigma}_{xy}$$

$$\widehat{\sigma}^2 = S_y^2 - \widehat{\boldsymbol{\Sigma}}_{xy}^\top \widehat{\boldsymbol{\Sigma}}_x^{-1} \widehat{\boldsymbol{\Sigma}}_{xy}$$

$$\text{Estimate } cov(\widehat{\boldsymbol{\beta}}) \text{ with } \widehat{\sigma}^2 \, \tfrac{1}{n} \, \widehat{\boldsymbol{\Sigma}}_x^{-1}$$

Base on pairwise complete observations.

The only question is, what should we use for n?
