Package 'MaskJointDensity'

May 22, 2018

Type Package

Title Masking, Unmasking and Restoring Confidential Data

Version 1.0

Date 2018-05-05

Author Yan-Xia Lin [aut, cre], Luke Mazur [aut, cre], Jordan Morris [ctb]

Maintainer Luke Mazur <lm810@uowmail.edu.au>

Description Three key functionalities are present. It is able to mask confidential data using multiplicative noise. It is able to unmask this data while still preserving confidentiality. It is able to calculate the numerical joint density function of the original data from the unmasked data, as well as obtaining a sample from the marginal density functions of the unmasked data. The final results are a reasonable approximation to the original data for the purposes of analysis (Lin et al. (2018) <http://www.tdp.cat/issues16/abs.a271a17.php>).

License GPL-2

Imports ks,np,plyr,parallel,MASS

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-22 12:13:25 UTC

R topics documented:

`MaskJointDensity-package`
Masking, unmasking and restoring confidential data.

Description

This package has 3 key functionalities. It is able to mask confidential data using multiplicative noise. It is able to unmask this data while still preserving confidentiality. It is able to calculate the numerical joint density function of the original data from the unmasked data, as well as obtaining a sample from the marginal density functions of the unmasked data. The final results are a reasonable approximation to the original data for the purposes of analysis.

Details

Package:	MaskJointDensity
Type:	Package
Version:	1
Date:	2018-05-05
License:	GPL-2

The Data Provider obtains confidential data, then masks it and writes it. The End-User unmasks this data then can obtain the joint density function of the unmasked variables and samples from the marginal density of them in order to mimic having the confidential data for the purpose of statistical

analysis.

Author(s)

Yan-Xia Lin Luke Mazur Jordan Morris

Maintainer: Luke Mazur <lm810@uowmail.edu.au>

References

Yan-Xia Lin, Luke Mazur, Rathin Sarathy, Krishnamurty Muralidhar Statistical Information Recovery from Multivariate Noise-Multiplied Data, a Computational Approach Transactions on Data Privacy 11:1 (2018) 23 - 45 http://www.tdp.cat/issues16/abs.a271a17.php

Examples

```
# Data Provider
set.seed(100)
data(FPI2002Data)
OriginalVar01 <- log(FPI2002Data$FPIflow2)
OriginalVar02 <- log(FPI2002Data$gdp_o)
OriginalVar03 <- log(FPI2002Data$gdp_d)
OriginalVar04 <- log(FPI2002Data$distw)

testN <- nrow(FPI2002Data)

noise<-rmulti(n=testN, mean=c(80, 100), sd=c(5,3), p=c(0.6, 0.4))
xmask<-mask(vectorToBeMasked=OriginalVar01, noisefile=file.path(tempdir(),"noise1.bin"),
noise=noise, lowerBoundAsGivenByProvider=min(OriginalVar01),
upperBoundAsGivenByProvider=max(OriginalVar01))
write(xmask$ystar, file.path(tempdir(),"xstar1.dat"))

noise<-rmulti(n=testN, mean=c(60, 90), sd=c(5,3), p=c(0.4, 0.6))
xmask<-mask(vectorToBeMasked=OriginalVar02, noisefile=file.path(tempdir(),"noise2.bin"),
noise=noise, lowerBoundAsGivenByProvider=min(OriginalVar02),
upperBoundAsGivenByProvider=max(OriginalVar02))
write(xmask$ystar, file.path(tempdir(),"xstar2.dat"))

noise<-rmulti(n=testN, mean=c(40, 110), sd=c(5,3), p=c(0.6, 0.4))
xmask<-mask(vectorToBeMasked=OriginalVar03, noisefile=file.path(tempdir(),"noise3.bin"),
noise=noise, lowerBoundAsGivenByProvider=min(OriginalVar03),
upperBoundAsGivenByProvider=max(OriginalVar03))
write(xmask$ystar, file.path(tempdir(),"xstar3.dat"))

noise<-rmulti(n=testN, mean=c(70, 100), sd=c(5,3), p=c(0.4, 0.6))
xmask<-mask(vectorToBeMasked=OriginalVar04, noisefile=file.path(tempdir(),"noise4.bin"),
noise=noise, lowerBoundAsGivenByProvider=min(OriginalVar04),
upperBoundAsGivenByProvider=max(OriginalVar04))
write(xmask$ystar, file.path(tempdir(),"xstar4.dat"))

# End-User
```

```
xstar1 <- scan(file.path(tempdir(),"xstar1.dat"))
Unmasked01 <- unmask(maskedVectorToBeUnmasked=xstar1,
noisefile=file.path(tempdir(),"noise1.bin"))

xstar2 <- scan(file.path(tempdir(),"xstar2.dat"))
Unmasked02 <- unmask(maskedVectorToBeUnmasked=xstar2,
noisefile=file.path(tempdir(),"noise2.bin"))

xstar3 <- scan(file.path(tempdir(),"xstar3.dat"))
Unmasked03 <- unmask(maskedVectorToBeUnmasked=xstar3,
noisefile=file.path(tempdir(),"noise3.bin"))

xstar4 <- scan(file.path(tempdir(),"xstar4.dat"))
Unmasked04 <- unmask(maskedVectorToBeUnmasked=xstar4,
noisefile=file.path(tempdir(),"noise4.bin"))

# If the Data Provider has given the End-User the mean, standard deviation,
# or correlation matrix of the original variables

listOfMeansOfOriginalVariables<-list(mean(OriginalVar01), mean(OriginalVar02),
mean(OriginalVar03), mean(OriginalVar04))
listOfStandardDeviationsOfOriginalVariables<-list(sd(OriginalVar01), sd(OriginalVar02),
sd(OriginalVar03), sd(OriginalVar04))
matrixOfCorrelationsOfOriginalVariables<-cor(cbind(OriginalVar01, OriginalVar02,
OriginalVar03, OriginalVar04))

test1 <- getSampleBasedOnUnmaskedData(maskedVectors =
list(xstar1, xstar2, xstar3, xstar4),
unmaskedVectors = list(Unmasked01$unmaskedVariable,
Unmasked02$unmaskedVariable, Unmasked03$unmaskedVariable,
Unmasked04$unmaskedVariable),
mu = listOfMeansOfOriginalVariables,
s = listOfStandardDeviationsOfOriginalVariables,
rho_X = matrixOfCorrelationsOfOriginalVariables,
verbose = 2,
size = 1000)

# If the Data Provider has not, then these are estimated

test2 <- getSampleBasedOnUnmaskedData(meansOfNoises =
list(Unmasked01$meanOfNoise, Unmasked02$meanOfNoise,
Unmasked03$meanOfNoise, Unmasked04$meanOfNoise),
meansOfSquaredNoises = list(Unmasked01$meanOfSquaredNoise,
Unmasked02$meanOfSquaredNoise, Unmasked03$meanOfSquaredNoise,
Unmasked04$meanOfSquaredNoise),
maskedVectors = list(xstar1, xstar2, xstar3, xstar4),
unmaskedVectors = list(Unmasked01$unmaskedVariable,
Unmasked02$unmaskedVariable, Unmasked03$unmaskedVariable,
Unmasked04$unmaskedVariable),
verbose = 2,
size = 1000)
```

```
## alternatively - using batch versions

# Data Provider
set.seed(100)
data(FPI2002Data)
OriginalVar01 <- log(FPI2002Data$FPIflow2)
OriginalVar02 <- log(FPI2002Data$gdp_o)
OriginalVar03 <- log(FPI2002Data$gdp_d)
OriginalVar04 <- log(FPI2002Data$distw)

testN <- nrow(FPI2002Data)

noise1<-rmulti(n=testN, mean=c(80, 100), sd=c(5,3), p=c(0.6, 0.4))
noise2<-rmulti(n=testN, mean=c(60, 90), sd=c(5,3), p=c(0.4, 0.6))
noise3<-rmulti(n=testN, mean=c(40, 110), sd=c(5,3), p=c(0.6, 0.4))
noise4<-rmulti(n=testN, mean=c(70, 100), sd=c(5,3), p=c(0.4, 0.6))

maskBatchOut <- maskBatch(listOfVectorsToBeMasked=
list(OriginalVar01,OriginalVar02,OriginalVar03,OriginalVar04),
listOfNoisefiles=list(file.path(tempdir(),"noise1.bin"),file.path(tempdir(),"noise2.bin"),
file.path(tempdir(),"noise3.bin"),file.path(tempdir(),"noise4.bin")),
listOfNoises=list(noise1,noise2,noise3,noise4),
listOfLowerBoundsAsGivenByProvider=
list(min(OriginalVar01),min(OriginalVar02),min(OriginalVar03),min(OriginalVar04)),
listofUpperBoundsAsGivenByProvider=
list(max(OriginalVar01),max(OriginalVar02),max(OriginalVar03),max(OriginalVar04)),
maxorder = 100, EPS = 1e-06)

dataFileNames <- list(file.path(tempdir(),"xstar1.dat"),file.path(tempdir(),"xstar2.dat"),
file.path(tempdir(),"xstar3.dat"),file.path(tempdir(),"xstar4.dat"))

for(i in 1:length(maskBatchOut)) {
write((maskBatchOut[[i]])$ystar, dataFileNames[[i]])
}


# End-User

dataFileNames <- list(file.path(tempdir(),"xstar1.dat"),file.path(tempdir(),"xstar2.dat"),
file.path(tempdir(),"xstar3.dat"),file.path(tempdir(),"xstar4.dat"))
numberOfFiles <- length(dataFileNames)

maskedVectors <- list()
for(i in 1:numberOfFiles) {
maskedVectors[[i]] <- scan(dataFileNames[[i]])
}

# If the Data Provider has given the End-User the mean, standard deviation,
# or correlation matrix of the original variables
listOfMeansOfOriginalVariables<-list(mean(OriginalVar01), mean(OriginalVar02),
mean(OriginalVar03), mean(OriginalVar04))
listOfStandardDeviationsOfOriginalVariables<-list(sd(OriginalVar01), sd(OriginalVar02),
sd(OriginalVar03), sd(OriginalVar04))
```

```
matrixOfCorrelationsOfOriginalVariables<-cor(cbind(OriginalVar01, OriginalVar02,
OriginalVar03, OriginalVar04))

unmaskAndGetSampleOut1 <- unmaskAndGetSampleBatch(listOfMaskedVectorsToBeUnmasked=maskedVectors,
                                    listOfNoisefiles=
                                    list(file.path(tempdir(),"noise1.bin"),
                                    file.path(tempdir(),"noise2.bin"),
                                    file.path(tempdir(),"noise3.bin"),
                                    file.path(tempdir(),"noise4.bin")),
                                    mu=listOfMeansOfOriginalVariables,
                                    s=listOfStandardDeviationsOfOriginalVariables,
                                    rho_X=matrixOfCorrelationsOfOriginalVariables,
                                    cores = 1, size=1000,
                                    verbose = 2,
                                    onlyUnmasked = FALSE)


# If the Data Provider has not, then these are estimated

unmaskAndGetSampleOut2 <- unmaskAndGetSampleBatch(listOfMaskedVectorsToBeUnmasked=maskedVectors,
                                    listOfNoisefiles=
                                    list(file.path(tempdir(),"noise1.bin"),
                                    file.path(tempdir(),"noise2.bin"),
                                    file.path(tempdir(),"noise3.bin"),
                                    file.path(tempdir(),"noise4.bin")),
                                    cores = 1, size=1000,
                                    verbose = 2,
                                    onlyUnmasked = FALSE)
```

`actualPosition` *Getting a Sample from Marginal Density Functions*

Description

Obtains samples from the marginal density functions of the unmasked variables.

Usage

```
actualPosition(vectorL, prob, boundaryVec, size = 1)
```

Arguments

`vectorL`	Should be the dimension of the Joint Density Function
`prob`	The Joint Density Function
`boundaryVec`	Boundary of each element, min, max, min, max
`size`	The size of the sample

Details

Used by getSampleFromMarginalDistributionOfUnmaskedData

Value

An n*k matrix where n is the sample size and k is the number of vectors. Each column represents the sample from the marginal density of the kth variable.

Author(s)

Jordan Morris

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (vectorL, prob, boundaryVec, size = 1)
{
    len <- length(vectorL)
    n <- c()
    for (i in 1:len) {
        n[i] <- vectorL[i]
    }
    maxSize = 1
    for (i in 1:len) {
        maxSize <- maxSize * n[i]
    }
    w <- c(0:(maxSize - 1))
    k <- sample(w, size = size, replace = TRUE, prob = prob)
    barredPoints <- matrix(nrow = size, ncol = len)
    for (i in 1:size) {
        maxSize <- 1
        for (l in 1:len) {
            maxSize <- maxSize * n[l]
        }
        for (j in 1:(len - 1)) {
            maxSize <- maxSize/n[len + 1 - j]
            if (k[i] > maxSize) {
                barredPoints[i, (len + 1 - j)] <- floor(k[i]/maxSize) +
                  1
                k[i] <- k[i]%%maxSize + 1
            }
            else {
                barredPoints[i, (len + 1 - j)] <- 1
            }
```

```
        }
        barredPoints[i, 1] <- k[i] + 1
    }
    return(barredToActual(vectorL, boundaryVec, barredPoints))
}
```

---

anyNA *Checks for NA in a vector*

Description

Some users have this as an internal function, others do not

Usage

```
anyNA(x)
```

Arguments

x	A vector that one checks for any NA values.

Value

TRUE if any NA values are detected in x. Otherwise FALSE.

Author(s)

Yan-Xia Lin

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
{
    any(is.na(x))
  }
```

barredToActual *Purely used in actualPosition*

Description

See above

Usage

```
barredToActual(vectorL, boundaryVec, barred)
```

Arguments

vectorL

boundaryVec

barred

Author(s)

Jordan Morris

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (vectorL, boundaryVec, barred)
{
    size <- nrow(barred)
    dim <- ncol(barred)
    actual <- matrix(nrow = size, ncol = dim)
    for (i in 1:size) {
        for (j in 1:dim) {
            actual[i, j] <- boundaryVec[2 * j - 1] + (boundaryVec[2 *
                j] - boundaryVec[2 * j - 1])/(vectorL[j] - 1) *
                (barred[i, j] - 1)
        }
    }
    return(actual)
  }
```

calc_muX *Calculate the moments*

Description

Calculate mu_X(j) in equation (9) in Provost paper

Usage

```
calc_muX(muY, a, b)
```

Arguments

muY	moment of Y
a	lower boundary
b	upper boundary

Details

there are no more details required

Value

A vector

Note

no further notes

Author(s)

Mark and Yan-Xia

References

Moment-Based Density Approximants by S.B. Provost

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

a<-2
b<-5
muY<-c(3,3.5,5)
print(calc_muX(muY, a,b))
```

CheckRho *An old artefact from testing, purely for backwards compatibility*

Description

Purely for testing of the rho_0 function.

Usage

```
CheckRho(x1, x2, mu1, mu2, s1, s2, Srho12, G_Point7, GH_Quadrature)
```

Arguments

x1	See rho_0
x2	See rho_0
mu1	See rho_0
mu2	See rho_0
s1	See rho_0
s2	See rho_0
Srho12	See rho_0
G_Point7	See rho_0
GH_Quadrature	See rho_0

Value

See rho_0

Author(s)

Yan-Xia Lin

References

No references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x1, x2, mu1, mu2, s1, s2, Srho12, G_Point7, GH_Quadrature)
{
    fhat1 <- kde(x = x1, binned = TRUE)
    fhat2 <- kde(x = x2, binned = TRUE)
```

```
    g <- 0
    m <- 7
    for (l in 1:m) {
        for (k in 1:m) {
            g <- g + GH_Quadrature[l] * GH_Quadrature[k] * ((qkde(pnorm(G_Point7[l]),
                fhat1) - mu1)/s1) * ((qkde(pnorm(Srho12 * G_Point7[l] +
                sqrt(1 - Srho12^2) * G_Point7[k]), fhat2) - mu2)/s2)
        }
    }
    return(g)
}
```

---

createNoise *Create a noise*

Description

This function is used to create a 5-modal noise

Usage

```
createNoise(n, mean = rep(NA, 5),
sd = rep(1, length(mean)), prob = rep(1, length(mean))/length(mean))
```

Arguments

n	the size of noise
mean	a vector of the positions of the 5-modal distribution
sd	a vector of the standard error of the 5-modal noise
prob	a vector of the propotion of the sample from the 5-modal distribution

Details

no detalis required

Value

a real vector

Note

no further notes

Author(s)

Mark Yan-Xia

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

y<-rnorm(100)
noise<- createNoise(length(y))
```

`density_Rmask` *Density function*

Description

estimate density function based on masked data

Usage

```
density_Rmask(moments, a, b, n = 512)
```

Arguments

`moments`	moments information
`a`	lower boundary
`b`	upper boundary
`n`	scale on x-axis

Details

no details needed

Value

a vector

Note

no further notes

Author(s)

Yan-Xia

References

no reference

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

y<-rnorm(100, 0,1)
val <- 1
for(k in 1:8){
   val<- c(val, mean(y^k))
   }
a<- -2
b<- 2
Provost<- density_Rmask(val, a,b)
```

Dg *Purely used in rho_0*

Description

See above

Usage

```
Dg(x1, x2, mu1, mu2, s1, s2, rho12, star_rho12, fhat1, fhat2, G_Point7, GH_Quadrature)
```

Arguments

x1	See rho_0
x2	See rho_0
mu1	See rho_0
mu2	See rho_0
s1	See rho_0
s2	See rho_0
rho12	See rho_0
star_rho12	See rho_0
fhat1	See rho_0
fhat2	See rho_0
G_Point7	See rho_0
GH_Quadrature	See rho_0

Value

Numeric Scalar

Author(s)

Yan-Xia Lin

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
```

`encriptNoise` *encript noise*

Description

to create a binary file for noise

Usage

```
encriptNoise(noise, a, b, maxorder, levels, EPS, noisefile)
```

Arguments

`noise`	noise used to mask data
`a`	lower boundary
`b`	upper boundary
`maxorder`	determine the maximum order of the moment in Provost
`levels`	levels for categorical variable
`EPS`	criterion used to stop the number of moments in provost
`noisefile`	a binary noise file

Details

no details required

Value

binary code

Note

no further notes

Author(s)

Yan-Xia Lin

References

no reference

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

c2<-rnorm(100,0,1)
c2<-abs(c2)
a<-0
b<-2
maxorder<-10
lvls<-NULL
EPS<-1e-06
encriptNoise(c2,a,b,maxorder,lvls,EPS,file.path(tempdir(),"noisefile"))
```

EQsampleDensity *Samples from a bunch of nodes on a grid.*

Description

EQsampleDensity is used to simulate the nodes from a grid, where the increment of the position of the nodes on each marginal space of the grid is the same.

Usage

```
EQsampleDensity(sx, boundaryVec, NoNote = 215, size = 100)
```

Arguments

sx	Matrix where each column corresponds to a vector
boundaryVec	Vector of boundaries of the columns of sx in the order from_1, to_1, from_2, to_2 etc
NoNote	Number of nodes
size	Size of the sample

Value

See actualPosition

Author(s)

Jordan Morris

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (sx, boundaryVec, NoNote = 215, size = 100)
{
    n <- NoNote
    SXdim <- length(sx[1, ])
    space <- NULL
    for (i in 1:SXdim) {
        a <- boundaryVec[2 * i - 1]
        b <- boundaryVec[2 * i]
        d <- (b - a)/(n - 1)
        space <- c(space, d)
    }
    d <- min(space)
    XP <- NULL
    vectorL <- NULL
    for (i in 1:SXdim) {
        a <- boundaryVec[2 * i - 1]
        b <- boundaryVec[2 * i]
        xposition <- seq(from = a, to = b, by = d)
        L <- length(xposition)
        XP <- c(XP, xposition)
        vectorL <- c(vectorL, L)
    }
    NotePositions <- positions(XP, vectorL)
    H <- Hpi(x = sx)
    fhat <- kde(x = sx, H = H)
    prob <- predict(fhat, x = NotePositions)
    outSample <- actualPosition(vectorL, prob, boundaryVec, size)
    return(outSample)
  }
```

`findOrder_Rmask` *order determination*

Description

To determine the upper order of moment in the Legendre polynomial function

Usage

```
findOrder_Rmask(ystar, noise, a, b, maxorder = 100, EPS)
```

Arguments

`ystar`	masked data
`noise`	noise information
`a`	lower boundary
`b`	upper boundary
`maxorder`	the maximum order of moment used in the analysis
`EPS`	criterion for stopping

Details

no details needed

Value

integer

Note

no further notes

Author(s)

Yan-Xia

References

the idea is described in paper

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

y<-rnorm(100,0,1)
noise<-abs(rnorm(100,2,4))
ystar<- noise*y
a<- -2
b<- 2
order<-findOrder_Rmask(ystar,noise,a,b,maxorder=100, EPS=1e-06)
print(order)
```

FPI2002Data — *A real life dataset consisting of the bilateral investment between many different countries in 2002, as well as the corresponding covariates from 2001. This was to eliminate any dual causality element in data analysis.*

Description

The dataset is a merging of CEPII CPIS data, the CEPII "Network Trade" dataset and the "Life During Growth" dataset from the world bank.

Usage

```
data("FPI2002Data")
```

Format

A data frame with 1868 observations on the following 46 variables.

`iso_d` a character vector

`iso_o` a character vector

`year` a numeric vector

`contig` a numeric vector

`comlang_off` a numeric vector

`distw` a numeric vector

`pop_o` a numeric vector

`gdp_o` a numeric vector

`gdpcap_o` a numeric vector

`iso2_o` a character vector

`pop_d` a numeric vector

`gdp_d` a numeric vector

gdpcap_d a numeric vector

iso2_d a character vector

heg_d a numeric vector

conflict a numeric vector

indepdate a numeric vector

heg_o a numeric vector

col_to a numeric vector

col_fr a numeric vector

col_hist a numeric vector

col_cur a numeric vector

sever a numeric vector

sib_conflict a numeric vector

gatt_o a numeric vector

gatt_d a numeric vector

rta a numeric vector

comleg a numeric vector

comcur a numeric vector

acp_to_eu a numeric vector

gsp a numeric vector

eu_to_acp a numeric vector

gsp_rec a numeric vector

flow a numeric vector

validmirror a numeric vector

family a character vector

FPIflow a factor with levels - -15 -2 -26 -49 -9 ... 0 1 1,001 1,002 1,006 1,012 1,018 1,028 1,031 1,038 1,039 1,046 1,051 1,060 1,064 1,070 1,081 1,083 1,099 1,100 1,103 1,111 1,114 1,116 1,120 1,124 1,128 1,130 1,138 1,143 1,145 1,147 1,152 1,154 1,156 1,161 1,167 1,167,313 1,169 1,172 1,179 1,185 1,195 1,196 1,203 1,205 1,206 1,208 1,214 1,234 1,238 1,239 1,241 1,251 1,253 1,257 1,265 1,267 1,281,806 1,282 1,283 1,284 1,286 1,288 1,289,749 1,289,876 1,297 1,298 1,300 1,304 1,304,044 1,307 1,313 1,314 1,315 1,321 1,322 1,325 1,331 1,333 1,347 1,359 1,364 1,367 1,371 1,372 1,376 1,377 1,385 1,389 1,400 1,403 1,406 1,407 1,411 1,415 1,416 1,421 1,433 1,435 1,436 1,437 1,441 1,444 1,445 1,446 1,463 1,473 1,482 1,505 1,526 1,528 1,529 1,530 1,534 1,537 1,543 1,555 1,560 1,582 1,586 1,591 1,592 1,605 1,607 1,615 1,619 1,622 1,628 1,629 1,630 1,632 1,642 1,643 1,646 1,647 1,652 1,667 1,669 1,673 1,695 1,697 1,706 1,707 1,716 1,726 1,737 1,749 1,752 1,763 1,773 1,778 1,789 1,814 1,820 1,829 1,836 1,841 1,844 1,846 1,852 1,858 1,867 1,868 1,874 1,878 1,879 1,884 1,887 1,897 1,901 1,903 1,906 1,920 1,934 1,957 1,980 10 10,208 10,335 10,389 10,399 10,428 10,542 10,637 10,694 10,705 10,754 10,785 10,802 10,969 10,992 10,993 100 100,757 101 102 103 103,235 103,660 104 105 106 106,052 107 108 108,168 109 11 11,054 11,123 11,143 11,368 11,508

11,534 11,543 11,645 11,663 11,853 11,881 11,883 11,961 110 110,356 111 111,307 112 112,285 112,457 113 114 115 116 116,530 117 117,085 118 119 119,217 12 12,062 12,115 12,117 12,121 12,206 12,289 12,307 12,319 12,360 12,422 12,672 12,719,412 12,728 12,916 12,933 12,947 12,955 12,987 120 122 123 124 125 125,478 126 127 128 129 13 13,035 13,182 13,489 13,554 13,702 13,897 130 130,030 130,577 131 132 132,785 133 134 134,142 135 136 138 14 14,066 14,140 14,456 14,512 14,572 14,576 14,641 14,872 140 141 142 142,411 143 143,572 144 145 145,557 145,558 145,827 146 146,431 147 148 149 149,082 15 15,013 15,132 15,184 15,280 15,330 15,521 15,623 15,662 15,924 150 150,101 151 152 152,701 153 153,568 154 155 155,710 156 157 158 159 16 16,026 16,046 16,363 16,397 16,484 16,585 16,676 16,679 16,774 16,820 16,861 16,901 16,999 160 161 162 162,433 163 164 165 166 167 168 168,501 169 17 17,133 17,273 17,608 17,776 17,779 170 170,828 171 173 173,143 174 174,551 175 176 177 177,597 177,910 178 179 18 18,125 18,336 18,455 18,528 18,575 18,639 18,647 18,804 180 182 183 184 185 186 187 189 19 19,180 19,246 19,352 19,705 19,813 19,860 19,895 191 192 193 194 196 197,839 198 199 2 2,001 2,011 2,016 2,026 2,041 2,052 2,057 2,067 2,071 2,074 2,075 2,078 2,094 2,095 2,104 2,114 2,118 2,123 2,132 2,133 2,135 2,140 2,146 2,168 2,178 2,184 2,185 2,189 2,196 2,206 2,219 2,230 2,232 2,234 2,235 2,247 2,267 2,277 2,279 2,281 2,287 2,290 2,297 2,303 2,303,603 2,309 2,318 2,331 2,341 2,360 2,364 2,369 2,378 2,389 2,400 2,403 2,407 2,415 2,421 2,423 2,425 2,433 2,435 2,438 2,440 2,448 2,450 2,466 2,469 2,471 2,485 2,492 2,531 2,532 2,533 2,538 2,610 2,625 2,626 2,645 2,646 2,663 2,685 2,688 2,690 2,693 2,696 2,698 2,721 2,723 2,724 2,737 2,739 2,741 2,760 2,762 2,765 2,769 2,770 2,816 2,822 2,842 2,843 2,847 2,862 2,893 2,903 2,907 2,919 2,929 2,933 2,975 20 20,230 20,417 20,421 20,887 201 202 203 204 205 205,600 206 207 208 209 21 21,117 21,233 21,489 21,905 21,910 210 211 211,444 212 213 214 215 217 218 218,587 219 22 22,039 22,551 22,709 22,755 22,818 22,972 220 221 222 223 224 225 228 23 23,466 23,752 23,763 230 231 232 233 234 235 236 238 24 24,013 24,255 24,711 240 241 242 243 245 246 247 248 249 25 25,008 25,154 25,247 25,395 250 251 252 254 257 258 259 26 26,137 26,254 26,657 26,821 26,850 260 261 261,240 263 264 265 266 267 268 269 27 27,342 270 271 273 274 275 277 278 279 279,013 28 28,374 28,401 28,712 28,947 280 280,426 281 283 285 285,067 286 287 288 289 29 29,195 29,411 29,741 29,983 290 291 292 293 295 296 298 299 3 3,002 3,004 3,022 3,048 3,052 3,055 3,071 3,085 3,094 3,099 3,105,841 3,110 3,128 3,146 3,147 3,205 3,208 3,217 3,228 3,251 3,268 3,280 3,286 3,295 3,309 3,316 3,325 3,330 3,348 3,445 3,447 3,460 3,533 3,536 3,539 3,544 3,547 3,564 3,594 3,616 3,643 3,646 3,655 3,656 3,660 3,674 3,677 3,690 3,728 3,742 3,756 3,764 3,766 3,798 3,808 3,815 3,854 3,869 3,875 3,891 3,961 3,963 30 30,260 30,311 30,472 30,519 30,643 300 301 304 306 307 308 308,986 309 31 31,070 310 312 313 314 315 317 318 319 319,795 32 32,047 32,304 32,431 32,783 32,974 320 321 322 325 326 327 328 329 33 33,333 33,731 33,986 330 332 333 334 335 336 338 34 34,130 34,475 342 343 345 348 35 350 352 353 354 355 356 357 358 359 36 36,830 36,932 361 362 364 368 369 37 37,787 370 372 374 375 376 377 379 38 382 384 385 387 388 389 39 39,228 39,253 39,874 39,882 39,922 390 391 393 395 396 397 4 4,015 4,022 4,056 4,063 4,077 4,089 4,129 4,139 4,147 4,161 4,170 4,189 4,258 4,304 4,336 4,346 4,358 4,385 4,398 4,410 4,506 4,509 4,513 4,522 4,524 4,533 4,563 4,607 4,615 4,625 4,645 4,649 4,673 4,676 4,713 4,730 4,735 4,768 4,791 4,821 4,885 4,887 40 40,659 40,787 401 402 403 405 406 407 408 41 41,207 41,632 410 411 412 414 415 416,538 417 419 42 42,547 420 421 422 423 424 425 426 427 429 43 43,658 43,911 430 432 432,839 434 435 44 44,045 44,446 44,741 440 441 442 447

45 45,191 45,864 45,965 452 454 455 456 457 46 46,139 46,999 461 464 469 47 471 472 474 476 48 48,208 48,904 480 481 485 485,669 486 489 49 490,200 490,589 492 493 495 499 5 5,074 5,083 5,100 5,106 5,121 5,190 5,211 5,246 5,268 5,272 5,306 5,350 5,466 5,533 5,657 5,674 5,736 5,763 5,829 5,835 5,844 5,850 5,870 5,879 5,948 5,987 50 50,693 50,817 503 505 509 51 51,144 51,193 510 512,975 514 515 516 518 52 522 523 525 525,324 526 527 529 53 53,154 53,589 530 532 533 536 537 54 54,174 540,800 546 547 548 549 55 55,139 55,183 55,692 55,910 552 552,022 553 555 556 56 561 564 565 567 568 57 57,435 571 572 575 577 579 579,551 58 584 585 587 59 59,187 59,566 592 594 595 597 599 6 6,072 6,082 6,116 6,143 6,162 6,180 6,222 6,242 6,270 6,308 6,337 6,380 6,389 6,394 6,457 6,474 6,487 6,488 6,509 6,514 6,551 6,604 6,697 6,747 6,748 6,757 6,830 6,850 6,872 6,891 6,937 6,969 6,999 60 60,309 601 603 608 61 61,036 61,745 610 614 615 62 620 621 623 626 627 629 63 631 633 635 636 638 639 64 64,356 64,422 641 649 65 65,324 650 653 659 66 661 668 669 67 67,486 673 679 68 68,182 68,221 68,603 682 684 686 689 69 690 699 7 7,005 7,008 7,086 7,150 7,166 7,194 7,198 7,332 7,350 7,400 7,620 7,667 7,722 7,861 7,904 7,921 70 702 704 705 705,536 71 71,039 710,330 714 717 72 72,359 72,961 724 725 73 73,557 73,949 74 74,001 74,191 74,645 74,903 746 747 748,051 75 757 759 76 76,627 762 763 77 77,340 774 776,583 778 779 78 780 781 783 785 786 787 789 79 79,352 791,616 792 794 796 799 8 8,025 8,034 8,084 8,148 8,257 8,275 8,280 8,297 8,328 8,349 8,386 8,416 8,481 8,508 8,540 8,597 8,742 8,746 8,789 8,962 8,963 80 80,991 81 81,399 810 814 82 82,744 820,614 822 825 826 827 83 833 836 837 84 84,186 84,493 842 846 85 85,643 85,838 852 856 857 859 86 86,353 86,571 861 867 87 877 88 881 888 89 891 896 9 9,060 9,093 9,248 9,328 9,447 9,457 9,532 9,539 9,645 9,761 9,867 9,956 90 902 91 91,069 91,384 92 924 925 928 929 93 931 937 94 94,090 95 954 955 96 96,699 960 963 964 966 969 97 978 98 987 99 c −5 −6 1,015 1,017 1,022 1,024 1,032 1,043 1,047 1,052 1,054 1,056 1,061 1,082 1,089 1,090 1,093 1,094 1,105 1,106 1,109 1,113 1,121 1,127 1,136 1,159 1,165 1,174 1,178 1,202 1,210 1,219 1,220 1,228 1,235 1,237 1,242 1,248 1,249 1,254 1,258 1,261 1,271 1,275 1,276 1,279 1,291 1,292 1,301 1,302 1,326 1,353 1,354 1,355 1,360,380 1,360,647 1,362 1,366 1,368,810 1,375 1,378 1,391 1,394,520 1,396 1,399 1,427 1,429,101 1,431 1,438 1,458 1,459 1,460 1,474 1,484 1,490 1,492 1,503 1,517 1,523 1,527 1,535 1,545 1,546 1,558 1,562 1,575 1,589 1,595 1,597 1,604 1,608 1,613 1,623 1,627 1,641 1,678 1,680 1,681 1,699 1,702 1,705 1,722 1,757 1,770 1,785 1,808 1,821 1,827 1,830 1,839 1,855 1,863 1,875 1,876 1,880 1,890 1,896 1,898 1,899 1,900 1,902 1,915 1,922 1,926 1,927 1,932 1,947 1,948 1,952 1,953 1,977 1,978 1,983 1,984 1,988 1,990 1,996 10,115 10,225 10,258 10,453 10,807 10,913 10,989 103,211 104,411 106,095 106,134 11,094 11,228 11,285 11,309 11,374 11,435 11,492 11,578 11,597 11,656 11,729 11,733 11,774 115,441 116,055 119,196 12,054 12,168 12,358 12,359 12,386 12,416 12,510 12,591 12,834 12,880 12,885 121 121,007 123,368 124,146 124,198 127,097 127,297 13,002 13,045 13,122 13,301 13,361 13,403 13,519 13,578 13,615 13,638 13,645 13,689 13,928 130,954 134,989 137 137,976 138,711 138,989 139 14,147,693 14,207 14,243 14,457 14,565 14,621 14,663 14,792 14,838 14,846 14,895 142,260 142,618 142,940 146,750 147,061 148,028 148,440 15,461 15,733 15,827 152,485 153,003 16,008 16,039 16,065 16,168 16,268 16,270 16,401 16,410 16,417 16,442 16,451 16,569 16,644 16,723 16,732 16,972 17,008 17,128 17,317 17,459 17,522 17,540 17,605 17,958 171,262 171,421 171,713 172 175,166 176,180 179,420 18,023 18,046 18,143 18,187 18,803 180,988 181 187,671 188 19,211 19,227 19,228 19,346 19,458 19,569 19,885 194,476 195 195,439 197 2,008 2,019 2,032 2,047 2,049 2,059 2,062

2,063 2,073 2,076 2,080 2,083 2,086 2,127 2,152 2,172 2,174 2,192 2,197 2,207
2,209 2,216 2,233 2,246,034 2,248 2,261 2,273 2,285 2,301 2,306 2,312 2,321 2,325
2,330 2,333 2,344 2,350 2,353 2,359 2,373 2,381 2,409 2,419 2,420 2,429 2,430
2,467 2,486 2,505 2,508 2,514 2,524 2,527 2,534 2,537 2,543 2,551 2,558 2,580
2,584 2,592 2,606 2,617 2,637 2,654 2,661 2,664 2,665 2,674 2,683 2,686 2,699
2,701 2,708 2,711 2,715 2,729 2,759 2,799 2,805 2,812 2,836 2,844 2,846 2,859
2,866 2,922 2,935 2,939 2,942 2,987 2,989 2,993 20,109 20,420 20,474 20,728 20,873
20,900 20,996 200 201,777 21,579 21,631 21,694 21,807 21,817 216 22,190 22,218
22,435 22,628 22,826 22,832 226 227 229 23,007 23,032 23,166 23,290 23,490 23,657
23,939 23,944 230,841 231,526 237 239,129 24,115 24,419 24,447 24,566 24,865
240,475 244,068 25,273 25,282 25,416 25,700 25,815 25,845 255 256 26,095 26,286
26,682 26,920 269,823 27,016 28,463 28,845 282 284 29,762 294 297 3,006 3,053
3,073 3,074 3,075 3,080 3,103 3,122 3,144 3,173 3,182 3,226 3,234 3,236 3,239
3,241 3,244 3,255 3,260 3,263 3,270 3,297,765 3,303 3,310 3,346 3,369 3,383 3,436
3,441 3,449 3,466 3,476 3,482 3,500 3,553 3,596 3,615 3,625 3,641 3,688 3,691
3,734 3,797 3,806 3,811 3,844 3,861 3,866 3,871 3,892 3,930 3,981 3,991 30,068
30,929 302 305 31,353 31,567 311 316 318,766 32,925 323 33,031 33,071 33,261 33,372
33,378 33,832 331 333,183 336,609 337 337,847 34,582 34,859 340 344 346 347 349
35,450 35,685 359,464 36,281 36,419 36,574 36,931 360 363 37,020 37,570 37,651
373 38,379 38,432 38,469 38,865 380 381 383 39,138 39,573 39,985 394 4,012 4,074
4,086 4,111 4,151 4,164 4,179 4,203 4,277 4,282 4,312 4,334 4,356 4,363 4,368
4,374 4,383 4,394 4,395 4,404 4,445 4,447 4,490 4,542 4,555 4,613 4,685 4,695
4,699 4,761 4,775 4,777 4,941 4,958 4,963 4,976 4,998 40,256 404 409 41,277 41,656
41,760 41,959 418 42,273 428 43,128 43,594 431 438 439 443 445 446 449 45,155
45,214 451 453 46,872 460 463 466 467 468 47,544 47,600 477 479 48,786 482 483 484
487 49,029 49,758 49,892 490 491 494,639 499,048 5,004 5,032 5,089 5,124 5,158
5,159 5,227 5,270 5,286 5,302 5,330 5,384 5,385 5,397 5,469 5,477 5,556 5,588
5,604 5,625 5,697 5,704 5,719 5,750 5,761 5,796 5,800 5,864 5,884 5,913 5,915
5,991 50,649 50,736 500 501 502 504 506 507 508 51,513 51,814 512,498 52,303 52,751
524 53,120 53,536 53,984 530,383 535,257 54,150 541 542 543 55,194 55,799 557
56,461 562 563 566 569 570 570,003 573,967 576 578 58,616 581 589 590 596 596,058
6,010 6,015 6,097 6,100 6,102 6,176 6,181 6,202 6,206 6,299 6,306 6,328 6,347
6,355 6,370 6,374 6,452 6,476 6,483 6,505 6,649 6,679 6,724 6,782 6,841 6,862
6,864 6,965 6,973 604 605 61,403 61,513 611 616 619 62,041 62,096 62,918 624 628
63,305 630 632 644 645 648 649,527 65,533 655 657 66,383 66,552 660 662 664 667
67,165 677 680 681 685 687 688 69,241 69,450 69,971 693 7,025 7,026 7,045 7,068
7,073 7,103 7,138 7,139 7,213 7,263 7,309 7,321 7,333 7,375 7,406 7,420 7,449
7,513 7,558 7,586 7,637 7,745 7,760 7,787 7,803 7,937 70,609 700 701 709 71,494
715 716 718 720 721 727 73,186 736 738,323 739 750 751 76,020 761 764 767 768 770 773
78,166 782 79,184 79,331 79,524 8,013 8,096 8,107 8,160 8,170 8,194 8,263 8,282
8,473 8,514 8,531 8,551 8,637 8,695 8,720 8,727 8,754 8,778 8,877 8,956 8,975
80,001 80,135 80,253 802 804 809 81,398 815 819 82,631 82,632 820 824 825,517
827,418 828 83,427 831 834 845 847,951 85,754 850 854 855 858 86,568 862 87,488
87,524 883 885 897 897,832 899 9,050 9,102 9,255 9,294 9,314 9,325 9,435 9,473
9,511 9,568 9,594 9,807 9,897 9,951 9,972 90,453 901 906 907 91,051 911 913 916
923,399 927 93,931 931,876 932 935 936 939 94,836 94,846 941 943 952 959 962 97,036
98,232 981 982 991 997 999 −1 −17 −23 1,006,275 1,014 1,016 1,021 1,034 1,035
1,036 1,049 1,055 1,062 1,065 1,067 1,075 1,078 1,080 1,092 1,104 1,112 1,115

1,117 1,117,614 1,118 1,129 1,132 1,133 1,142 1,150 1,157 1,158 1,160 1,168 1,170
1,184 1,204 1,205,127 1,209 1,216 1,229 1,256 1,280 1,293 1,311 1,313,815 1,324
1,333,204 1,343 1,344 1,350 1,356 1,369,577 1,370 1,373 1,379 1,381 1,393 1,395
1,402 1,410 1,424 1,439 1,443 1,449 1,453 1,457 1,475 1,477 1,483 1,485 1,488
1,493 1,497 1,501 1,508 1,513 1,519 1,521 1,531 1,550 1,554 1,570 1,578 1,579
1,581 1,584 1,590 1,596 1,617 1,620 1,631 1,635 1,639 1,645 1,659 1,670,204 1,684
1,690 1,691 1,713 1,717 1,721 1,721,307 1,740 1,742 1,747 1,754 1,764 1,768 1,780
1,788 1,792 1,793 1,810 1,817 1,843 1,844,047 1,849,783 1,850 1,851 1,855,404
1,865 1,881 1,883 1,886 1,891 1,910 1,931 1,944 1,956 1,961 1,972 1,985 1,992
1,997 10,000 10,040 10,166 10,185 10,296 10,312 10,319 10,344 10,358 10,372 10,443
10,456 10,558 10,643 10,690 10,710 10,773 10,784 10,793 10,808 10,832 10,903
10,958 106,007 106,191 106,984 107,018 107,395 107,412 108,277 109,018 109,190
109,505 109,876 11,149 11,205 11,373 11,378 11,480 11,595 11,683 11,711 11,742
11,786 11,834 11,963 11,980 113,694 116,295 117,333 118,673 119,125 119,172 119,715
12,003 12,038 12,096 12,153 12,225 12,226 12,366 12,491 12,492 12,772 12,828
12,886 12,909 12,966 12,985 122,169 123,220 124,972 126,178 127,034 128,139 128,375
128,483 13,082 13,115 13,163 13,181 13,453 13,587 13,656 13,738 13,810 13,812
13,817 133,346 14,296 14,359 14,530 14,650 14,812 142,948 143,312 15,296 15,463
15,475 15,699 15,750 15,913 152,900 154,751 159,575 159,934 16,032 16,081 16,148
16,198 16,253 16,438 16,783 16,905 16,987 167,894 17,155 17,343 17,365 17,420
17,564 17,625 17,761 17,849 17,985 170,667 171,140 172,170 178,304 18,150 18,218
18,263 18,433 18,515 18,640 18,644 18,648 18,698 18,699 18,748 18,845 181,419
182,193 183,425 184,358 184,871 186,611 188,203 189,594 19,033 19,052 19,090
19,163 19,217,763 19,325 19,676 19,702 19,816 19,914 190 194,939 2,000 2,007
2,009 2,013 2,021 2,035 2,043 2,056 2,070 2,090 2,093 2,099 2,107 2,117 2,120
2,124 2,129 2,131 2,138 2,154 2,175 2,177 2,179 2,180 2,191 2,199 2,214 2,223
2,238 2,251 2,263 2,271 2,272 2,282 2,299 2,308 2,326 2,332 2,335 2,347 2,352
2,354 2,358 2,367 2,385 2,395 2,398 2,412 2,422 2,427 2,432 2,476 2,512 2,513
2,518 2,523 2,529 2,530 2,535 2,539 2,556 2,567 2,613 2,627 2,638 2,671 2,675
2,694 2,707 2,710 2,713 2,735 2,751 2,752 2,756 2,764 2,815 2,834 2,875 2,888
2,889 2,905 2,912 2,921 2,940 2,965 2,968 20,396 20,488 20,492 20,496 20,588
205,569 206,733 207,035 208,490 21,243 21,335 21,348 21,446 21,555 21,594 21,598
21,760 21,840 212,775 213,956 214,612 22,141 22,253 22,555 221,507 228,604 23,383
23,495 23,518 23,551 23,748 231,493 232,664 234,295 24,022 24,207 24,445 24,678
24,737 24,811 24,852 244 245,985 249,435 25,001 25,717 25,764 25,976 253 253,978
26,149 26,575 26,612 26,654 26,701 26,877 27,230 27,274 27,661 272 28,281 28,284
28,582 28,664 28,824 28,835 29,004 29,547 29,882 291,850 294,833 3,000 3,013
3,058 3,091 3,102 3,107 3,120 3,134 3,134,244 3,135 3,141 3,143 3,149 3,165 3,168
3,211 3,219 3,221 3,231 3,242 3,296 3,308 3,315 3,320 3,365 3,375 3,380 3,393
3,408 3,416 3,417 3,455 3,481 3,487 3,494 3,519 3,526 3,554 3,567 3,568 3,579
3,599 3,621 3,622 3,626 3,647 3,650 3,664 3,681 3,687 3,700 3,701 3,767 3,770
3,790 3,793 3,817 3,822 3,852 3,864 3,878 3,914 3,953 3,972 3,999 30,122 30,175
30,234 30,281 30,886 30,985 300,978 303 311,349 32,226 32,472 323,142 33,226
33,342 33,470 33,586 33,599 33,684 33,713 33,841 33,859 334,912 339 34,262 34,484
34,896 34,966 341 35,098 35,419 35,770 350,592 36,013 36,102 36,563 36,680 365
37,145 37,661 371 378 38,158 38,495 38,641 38,693 38,830 381,921 39,521 392 4,014
4,024 4,029 4,041 4,058 4,073 4,119 4,125 4,143 4,153 4,215,192 4,228 4,292 4,314
4,315 4,328 4,331 4,376 4,382 4,424 4,464 4,471 4,499 4,514 4,526 4,566 4,578

4,590 4,593 4,606 4,650 4,658 4,700 4,710 4,745 4,746 4,790 4,812 4,817 4,850
4,859 4,920 4,965 4,979 40,280 40,752 400 41,091 41,126 41,964 413 414,802 415,562
416 42,147 42,254 42,275 424,395 43,003 43,298 433 436 44,106 45,257 450 455,082
458 46,670 462 465 47,327 470 473 478 48,142 48,574 48,777 48,779 481,570 488
49,587 496 498 5,005 5,024 5,046 5,057 5,061 5,071 5,072 5,079 5,091 5,123 5,265
5,276 5,283 5,285 5,289 5,321 5,328 5,354 5,357 5,370 5,409 5,483 5,491 5,505
5,540 5,559 5,613 5,632 5,644 5,716 5,718 5,737 5,793 5,798 5,817 5,841 5,851
5,852 5,865 5,889 5,897 5,906 5,920 5,935 5,939 5,970 50,229 50,489 51,007 51,547
51,692 511 512 517 52,716 52,991 520 521 528 53,429 53,843 534 535 538 54,395
54,916 540 545 55,286 55,511 55,999 550 551 554 559 56,011 56,240 56,441 57,442
574 58,380 59,149 59,488 591 593 6,008 6,009 6,043 6,057 6,163 6,217 6,248 6,263
6,284 6,334 6,352 6,398 6,429 6,479 6,498 6,513 6,519 6,522 6,555 6,608 6,639
6,650 6,696 6,743 6,751 6,803 6,881 6,915 6,935 6,955 60,897 60,932 600 606 607
609 61,329 612 618 62,366 62,841 62,886 620,208 625 64,288 642 646 65,621 652 658
66,706 66,931 663 663,120 666 672 672,614 675 676 678 68,512 68,602 691 695 698
7,125 7,134 7,137 7,144 7,181 7,205 7,249 7,258 7,260 7,261 7,262 7,292 7,329
7,339 7,427 7,428 7,565 7,635 7,708 7,734 7,770 7,819 7,893 7,895 7,925 7,953
7,961 703 705,596 712 726 728 729 73,902 733 734 737 738,029 749 758 76,524 760
765 766 769 77,394 78,382 79,207 791 791,064 8,002 8,012 8,065 8,132 8,165 8,218
8,261 8,271 8,306 8,363 8,372 8,397 8,438 8,504 8,533 8,583 8,596 8,671 8,737
8,764 8,806 8,858 8,879 8,912 8,915 8,959 8,976 80,232 800 81,638 811 812 813
82,203 82,965 82,969 821 832 834,912 841 843 85,644 851 86,349 860 861,265 87,344
870,535 878 886 89,726 9,006 9,139 9,263 9,296 9,305 9,331 9,346 9,375 9,581 9,591
9,599 9,606 9,697 9,823 9,838 9,941 9,955 90,335 90,502 904 905 918 920 94,843 942
944 945 946 95,825 96,970 968 97,290 98,845 980 988 988,526 989 99,549 99,910 998
−18 −3 1,000 1,004 1,009 1,020 1,029 1,033 1,037 1,040 1,042 1,053 1,066 1,068
1,071 1,072 1,072,312 1,079 1,098 1,104,430 1,114,549 1,137 1,139 1,148 1,162
1,164,799 1,166 1,173 1,175 1,190 1,192 1,207 1,211 1,217,861 1,225 1,247 1,250
1,281 1,297,152 1,303 1,327 1,329 1,334 1,338 1,345 1,348 1,382 1,392 1,394 1,412
1,413 1,414 1,417 1,428 1,454 1,472 1,489 1,494 1,510 1,515,477 1,532 1,538 1,552
1,553 1,556 1,559 1,570,955 1,572 1,580 1,585 1,588 1,601 1,609 1,616,314 1,625
1,633 1,638 1,648 1,654 1,655 1,658 1,675 1,694 1,703 1,714 1,724 1,725 1,738
1,745 1,750,706 1,766 1,769 1,771 1,777 1,779 1,787 1,790 1,812 1,815 1,835 1,840
1,849 1,853 1,870 1,893 1,921 1,925 1,929 1,930 1,942 1,967 1,982 1,999 10,220
10,454 10,492 10,498 10,524 10,631 10,703 10,760 10,873 10,874 10,887 100,111
100,676 103,412 109,380 11,008 11,055 11,058 11,096 11,141 11,171 11,220 11,354
11,420 11,433 11,465 11,570 11,598 11,604 11,655 11,700 11,798 11,803 11,816
11,828 11,910 110,276 110,364 115,549 117,532 118,093 118,616 12,034 12,071 12,154
12,198 12,211 12,323 12,352 12,459 12,462 12,487 12,504 12,636 12,706 12,723
12,733 12,754 12,764 120,253 120,514 124,121 124,822 124,885 124,949 125,062
128,637 13,036 13,049 13,236 13,245 13,349 13,426 13,469 13,471 13,715 13,752
13,882 13,922 13,948 13,991 130,302 133,135 133,252 133,857 134,270 138,043 139,355
139,644 14,097 14,149 14,305 14,610 14,719 14,741 14,756 14,858 14,867 14,878
14,891 14,970 142,013 145,939 149,628 15,019 15,198 15,238 15,249 15,299 15,477
15,630 15,631 15,694 15,874 15,877 15,964 153,187 153,198 153,706 154,493 154,957
154,959 155,680 157,632 16,405 16,735 16,829 16,898 16,927 164,074 167,036 169,831
17,058 17,094 17,103 17,186 17,455 17,555 17,689 17,726 17,919 174,048 179,350
18,059 18,307 18,375 18,629 18,666 18,689 18,743 18,867 18,927 18,936 19,286

19,383 19,443 19,625 19,783 19,817 19,973 193,573 195,241 195,677 197,106 197,487
2,009,672 2,014 2,015 2,033 2,045 2,060 2,061 2,079 2,084 2,085 2,092 2,109,701
2,126,689 2,136 2,137 2,145,013 2,156 2,164 2,173 2,193 2,222 2,226 2,241 2,243
2,244 2,257 2,262 2,264 2,265,854 2,284 2,298 2,313 2,322 2,324 2,327 2,342 2,382
2,397 2,416 2,439 2,447 2,453 2,464 2,465 2,468 2,472 2,488 2,493 2,504 2,568
2,574 2,607 2,621 2,652 2,656 2,667 2,668 2,679 2,717 2,725 2,732 2,736 2,746
2,757 2,771 2,773 2,777 2,780 2,784 2,785 2,786 2,798 2,819 2,835 2,839 2,872
2,874 2,896 2,899 2,900 2,909 2,913 2,931 2,950 2,971 2,976 2,982 20,276 20,339
20,406 20,670 20,694 200,677 201,799 202,361 204,938 206,145 206,700 21,258 21,314
21,325 21,340 21,519 21,856 211,023 212,505 214,950 22,264 22,307 22,421 22,547
22,577 22,600 22,974 225,009 23,247 23,484,559 23,514 23,525 23,618 23,652 23,706
23,882 23,973 230,414 233,231 233,574 239 24,037 24,144 24,147 24,208 24,303
24,346 24,390 24,426 24,448 24,647 24,982 246,106 25,210 256,334 257,737 26,289
26,687 26,746 26,842 260,588 262 264,538 268,262 268,959 27,546 27,743 272,848
274,253 274,739 276 279,210 28,128 28,351 28,368 284,266 29,090 29,091 29,110
29,446 29,551 3,028 3,040 3,047 3,114 3,153 3,157 3,159 3,167 3,174 3,176 3,179
3,207 3,212 3,233 3,237 3,240 3,248 3,258 3,274 3,290 3,306 3,318 3,329 3,342
3,357 3,385 3,386 3,398 3,423 3,450 3,451 3,454 3,457 3,483 3,523 3,538 3,563
3,565 3,629 3,640 3,663 3,665 3,694 3,710 3,721 3,764,346 3,778 3,783 3,802 3,824
3,826 3,853 3,857 3,916 3,935 3,941 3,951 3,976 3,978 30,277 30,379 30,477 307,344
309,658 31,062 31,165 31,371 31,677 31,949 310,693 32,527 32,866 324 33,161 33,344
33,452 34,153 34,315 34,369 34,462 34,651 34,679 34,856 34,887 344,669 35,435
35,691 351 36,227 36,507 36,743 36,781 360,178 366 367 369,806 37,350 37,566
37,922 38,050 38,076 38,090 38,150 38,346 38,616 38,896 380,933 383,871 39,030
39,509 399 4,047 4,126 4,184 4,185 4,201 4,204 4,212 4,238 4,243 4,249 4,256 4,266
4,299 4,309 4,322 4,325 4,366 4,367 4,415 4,436 4,455 4,465 4,467 4,475 4,519
4,560 4,604 4,646 4,680 4,689 4,692 4,723 4,727 4,731 4,736 4,778 4,827 4,832
4,834 4,857 4,910,853 4,916 4,932 4,955 4,962 400,889 41,622 41,716 42,024 42,041
43,023 43,062 43,140 43,248 43,304 43,319 44,413 444 448 45,596 45,610 454,478
47,091 47,099 47,175 47,279 47,304 47,355 475 48,345 48,578 49,199 49,616 49,901
49,966 494 5,012 5,045 5,055 5,076 5,087 5,104 5,105 5,111 5,143 5,194 5,196 5,231
5,267 5,274 5,277 5,290 5,293 5,352 5,356 5,379 5,443 5,525 5,529 5,538 5,609
5,610 5,638 5,690 5,699 5,708 5,820 5,849 5,903 5,911 5,921 5,941 5,943 5,957
5,984 50,431 507,578 513 517,762 519 519,920 52,283 52,655 52,666 52,715 521,430
53,051 53,658 531 534,075 539 54,272 55,264 55,324 558 560 57,790 57,986 58,213
58,378 58,463 58,836 583 586 588 59,651 59,847 6,063 6,064 6,067 6,080 6,084 6,093
6,113 6,170 6,185 6,199 6,216 6,266 6,341 6,383 6,386 6,396 6,397 6,424 6,570
6,628 6,646 6,700 6,712 6,718 6,793 6,806 6,859 6,863 6,902 6,909 6,921 6,941
6,987 6,988 60,252 60,485 602 617 62,398 62,999 63,564 63,715 637 64,620 643 65,276
65,397 651 654 66,069 66,171 66,317 67,702 67,895 670 674 68,021 68,130 68,591
68,613 68,905 683 683,229 694 694,382 7,058 7,090 7,113 7,130 7,172 7,211 7,227
7,231 7,233 7,299 7,399 7,457 7,556 7,571 7,604 7,625 7,689 7,697 7,736 7,764
7,772 7,788 7,799 7,829 7,926 7,940 7,981 706 707 71,275 71,368 71,396 71,828
71,863 710 713 73,613 737,754 74,712 74,930 740 743 744 745 75,339 756 76,496
76,527 76,528 76,758 77,709 77,710 77,849 772 775,523 78,801 795 8,224 8,243
8,259 8,273 8,290 8,324 8,335 8,352 8,364 8,396 8,398 8,457 8,468 8,480 8,552
8,616 8,623 8,666 8,744 8,819 8,823 8,825 8,856 8,935 8,978 80,032 803 805 806
817 823 83,302 83,303 830 835 847 849 853 86,537 86,590 869 87,097 87,450 872 880

887 89,187 892 894 895 898 9,000 9,016 9,039 9,070 9,088 9,246 9,269 9,336 9,368
9,396 9,488 9,512 9,544 9,569 9,612 9,616 9,620 9,622 9,628 9,660 9,661 9,695
9,745 9,809 9,833 9,852 9,853 9,878 9,960 90,281 900 909 914 914,014 921 922 923
930 934 934,790 94,699 951 957,505 965 975 98,654 98,803 98,807 99,291 990 992 −31
−4 −73 1,003 1,008 1,023 1,025 1,027 1,045 1,058 1,063 1,064,478 1,073 1,074 1,076
1,086 1,087 1,101 1,119 1,122 1,134 1,161,463 1,176 1,177 1,180 1,182,212 1,189
1,193 1,197 1,197,069 1,199 1,217 1,218 1,232 1,236 1,244,227 1,248,358 1,262
1,263 1,266 1,285 1,295 1,295,878 1,308 1,314,920 1,317 1,323 1,335 1,360 1,369
1,374 1,388 1,404 1,440 1,448 1,464 1,466 1,476 1,479 1,506 1,511 1,512 1,516
1,522 1,533 1,552,803 1,563 1,574 1,611 1,612 1,616 1,621,245 1,634 1,644 1,676
1,682 1,700 1,712 1,741 1,760 1,772 1,776 1,818 1,822 1,823 1,840,921 1,873,259
1,877 1,889 1,895 1,918 1,939 1,966 1,969 1,976 1,991 1,995 10,016 10,028 10,029
10,066 10,125 10,180 10,210 10,240 10,377 10,408 10,458 10,521 10,537 10,554
10,639 10,683 10,728 10,815 10,836 10,850 10,879 10,886 10,926 10,944 10,962
10,968 100,423 100,693 100,979 101,681 103,075 103,664 104,740 106,398 108,309
109,482 11,116 11,136 11,195 11,253 11,265 11,282 11,340 11,361 11,379 11,388
11,546 11,556 11,716 11,719 11,745 11,799 11,861 11,935 11,956 11,973 111,555
113,991 114,015 116,213 116,791 117,210 117,504 118,500 118,507 118,792 12,059
12,099 12,163 12,277 12,451 12,480 12,531 12,552 12,579 12,661 12,700 12,865
12,938 12,979 124,908 125,184 125,937 128,202 13,168 13,176 13,269 13,308 13,315
13,569 13,662 13,677 13,788 13,805 13,952 13,969 131,754 134,738 135,238 135,241
136,085 137,222 139,456 14,017 14,033 14,091 14,163 14,200 14,375 14,446 14,473
14,593 14,697 14,736 14,749 14,804 14,813 14,870 141,409 143,018 143,806 149,266
15,273 15,314 15,441 15,468 15,644 15,651 15,652 15,723 15,725 15,846 15,951
150,490 152,237 154,183 155,667 158,382 159,906 16,050 16,154 16,193 16,202 16,440
16,450 16,531 16,684 16,747 16,757 16,826 16,833 16,855 16,931 161,236 167,660
17,025 17,063 17,150 17,240 17,281 17,289 17,315 17,367 17,373 17,410 17,447
17,647 17,771 17,863 17,976 171,404 18,014 18,111 18,117 18,196 18,414 18,489
18,547 18,619 18,921 180,242 183,234 183,239 184,443 186,357 186,662 186,807
187,952 19,026 19,057 19,668 19,874 19,920 190,293 191,882 193,259 195,936 196,138
196,730 197,965 2,004 2,005 2,012 2,036 2,051 2,055 2,058 2,069 2,087 2,091 2,101
2,106 2,114,888 2,116 2,139 2,140,568 2,150 2,170 2,181 2,203 2,221 2,221,136
2,239 2,254 2,259 2,276 2,295 2,307 2,311 2,323 2,349 2,366 2,373,924 2,401 2,404
2,405 2,456,985 2,481 2,501 2,517 2,525 2,536 2,552 2,557 2,562 2,573 2,577 2,596
2,600 2,648 2,677 2,716 2,744 2,750 2,758 2,766 2,787 2,818 2,820 2,831 2,867
2,876 2,878 2,891 2,902 2,916 2,918 2,945 2,953 2,958 2,962 2,963 2,970 2,972
2,977 2,992 20,024 20,039 20,056 20,150 20,286 20,503 20,724 20,725 20,761 20,808
20,927 203,112 207,530 21,151 21,329 21,421 21,698 21,962 213,479 22,076 22,387
22,845 229,974 23,139 23,216 23,234 23,371 23,484 23,546 236,122 24,106 24,181
24,664 243,557 247,728 248,770 25,136 25,202 25,275 25,974 250,776 250,958 253,564
258,234 26,045,385 26,151 26,202 26,538 26,632 26,640 26,790 26,916 263,417 269,945
27,540 27,670 27,778 27,841 27,850 277,789 28,433 28,443 28,764 28,916 28,994
280,637 283,149 284,012 284,165 29,023 29,079 29,302 291,749 297,696 3,021 3,026
3,049 3,056 3,066 3,119 3,121 3,136 3,164 3,189 3,199 3,201 3,213 3,229 3,266
3,354 3,360 3,401 3,409 3,430 3,437 3,438 3,439 3,444 3,446 3,469 3,490 3,510
3,527 3,528 3,575 3,589 3,591 3,602 3,605 3,613 3,658 3,683 3,684 3,695 3,699
3,727 3,732 3,739 3,746 3,752 3,755 3,772 3,773 3,781 3,786 3,809 3,823 3,874
3,888 3,900 3,917 3,925 3,926 3,986 3,990 30,017 30,192 30,548 300,213 302,576

303,833 307,173 315,243 32,162 32,228 32,436 32,698 33,196 33,383 33,681 33,689
33,967 34,212 34,388 34,494 34,656 343,592 35,057 35,225 35,523 35,547 35,568
36,095 36,167 36,334 36,361 36,756 36,775 38,284 38,475 38,518 39,136 39,346
392,133 398 4,091 4,097 4,098 4,102 4,105 4,155 4,156 4,186 4,223 4,225 4,227
4,236 4,240 4,275 4,289 4,301 4,311 4,330 4,340 4,403 4,433 4,463 4,477 4,489
4,532 4,539 4,567 4,576 4,588 4,591,122 4,601 4,652 4,666 4,681 4,686 4,697 4,718
4,779 4,795 4,813 4,843 4,898 4,908 4,957 4,959 40,418 40,997 41,302 41,484 41,525
418,925 42,008 43,076 43,127 43,665 43,870 436,599 436,759 437 44,207 44,313
44,422 44,478 44,703 44,901 44,953 45,068 45,511 450,512 459 46,107 46,225 46,254
46,287 46,302 46,426 46,775 469,616 47,222 47,601 47,933 48,778 48,917 49,334
49,349 495,102 497 5,019 5,026 5,043 5,060 5,096 5,102 5,129 5,178 5,200 5,213
5,234 5,331 5,342,979 5,351 5,378 5,434 5,438 5,510 5,545 5,546 5,555 5,564 5,565
5,570 5,577 5,635 5,640 5,709 5,778 5,802 5,834 5,836 5,843 5,890 5,894 5,904
5,917 5,938 5,975 50,163 50,302 50,873 50,999 51,551 52,588 52,930 520,084 53,398
53,515 53,844 54,815 54,819 54,834 544 55,851 550,775 56,838 56,927 57,445 57,877
573 58,160 58,257 580 581,153 582 589,974 6,018 6,053 6,105 6,114 6,120 6,128
6,249 6,292 6,305 6,360 6,416 6,431 6,471 6,506 6,567 6,716 6,745 6,769 6,770
6,774 6,805 6,807 6,810 6,852 6,873 6,884 6,888 6,906 60,412 61,276 613 62,580
62,945 623,382 634 64,686 640 65,232 656 657,441 66,773 67,128 67,674 68,646
69,079 69,219 69,382 69,519 69,821 69,979 692 697 7,097 7,131 7,140 7,148 7,179
7,210 7,244 7,254 7,316 7,323 7,435 7,456 7,461 7,496 7,537 7,546 7,570 7,622
7,692 7,693 7,728 7,739 7,744 7,765 7,766 7,790 7,820 7,823 7,832 7,852 7,863
7,924 7,983 7,992 72,302 73,076 73,871 732 735 74,619 74,660 74,675 74,683 743,426
747,747 75,200 75,368 75,579 75,979 753 754 76,287 76,574 77,540 77,594 77,856
775 777 788 79,059 79,393 797,570 8,054 8,069 8,100 8,146 8,216 8,291 8,318 8,346
8,377 8,382 8,493 8,575 8,598 8,663 8,706 8,749 8,772 8,783 8,801 8,863 8,924
8,947 80,858 801 807 808 814,784 829 838 839 84,326 840 844 86,108 86,430 866 868
87,394 870 873 875 88,740 884 89,201 89,733 893 9,001 9,025 9,030 9,051 9,109
9,111 9,120 9,121 9,164 9,186 9,203 9,270 9,395 9,440 9,444 9,514 9,548 9,647
9,708 9,718 9,740 9,747 9,919 90,285 903 908 91,072 91,446 92,998 933 940 949
95,211 950 96,676 967 97,664 976 977 982,065 983 99,349 993 −12 −21 −28 1,005
1,007 1,009,619 1,026 1,030 1,041 1,044 1,050 1,069 1,075,579 1,077 1,095 1,096
1,097 1,107 1,121,070 1,123 1,125 1,126 1,140,617 1,163 1,188 1,193,438 1,200
1,226 1,227 1,244 1,268 1,270 1,272 1,278 1,290 1,294 1,305 1,306 1,309 1,318
1,320 1,327,585 1,336 1,349 1,351 1,358 1,387 1,397 1,418 1,420 1,423 1,423,135
1,442 1,447 1,452,132 1,467 1,468,947 1,469 1,481 1,507 1,520 1,534,760 1,540
1,551 1,561 1,564 1,567 1,602 1,610 1,618 1,620,218 1,624 1,626 1,664 1,674 1,685
1,698 1,711 1,715 1,718 1,731 1,732 1,735,411 1,743 1,775 1,784 1,791 1,794 1,795
1,796 1,798 1,799 1,806 1,819 1,833 1,837 1,847 1,862 1,869 1,905 1,912 1,951
1,958 1,960 1,964 1,974 10,036 10,054 10,104 10,162 10,197 10,230 10,267 10,304
10,347 10,388 10,403 10,455 10,494 10,497 10,519 10,522 10,525 10,610 10,619
10,746 10,848 10,892 10,901 10,902 10,966 10,996 101,974 102,067 102,096 102,633
103,631 103,636 103,888 105,893 108,071 108,450 11,018 11,157 11,176 11,215 11,259
11,372 11,376 11,506 11,521 11,557 11,615 11,676 11,677 11,723 11,812 11,877
11,913 11,982 11,998 110,294 110,956 112,895 113,107 115,675 116,582 119,843
12,037 12,170 12,201 12,250 12,267 12,411 12,456 12,493 12,524 12,538 12,539
12,565 12,628 12,732 12,789 12,876 12,986 120,513 122,769 123,650 123,875 125,833
128,153 128,654 129,275 129,639 13,042 13,046 13,052 13,076 13,090 13,257 13,273

13,277 13,288 13,577 13,602 13,821 13,825 13,921 13,970 13,997 130,974 132,255
132,546 135,549 14,072 14,083 14,159 14,233 14,262 14,309 14,338 14,371 14,729
14,750 14,803 14,826 14,886 14,931 14,932 14,968 140,883 142,364 145,272 15,002
15,151 15,205 15,266 15,284 15,369 15,404 15,500 15,562 15,793 15,870 151,056
152,252 157,381 157,832 16,013 16,023 16,098 16,149 16,152 16,208 16,242 16,456
16,563 16,657 16,827 16,915 16,925 160,337 160,558 160,864 162,273 163,392 164,896
167,892 17,108 17,127 17,212 17,306 17,330 17,404 17,423 17,494 17,534 17,683
17,836 17,890 171,227 172,361 173,153 173,502 174,937 18,265 18,354 18,441 18,482
18,504 18,530 18,638 18,668 18,721 18,730 18,817 18,844 182,082 183,115 183,731
19,053 19,110 19,158 19,307 19,316 19,592 19,654 19,678 19,686 19,831 19,849
19,915 19,979 193,453 194,603 195,898 2,010 2,030 2,042 2,053 2,072 2,096 2,108
2,112 2,155 2,160 2,176,999 2,205 2,208 2,211 2,213 2,225 2,253 2,255 2,258 2,266,371
2,315 2,316 2,329 2,343,482 2,376 2,383 2,393 2,410 2,413 2,430,920 2,434 2,461
2,463,960 2,521 2,526 2,540 2,550 2,558,358 2,576 2,585 2,595 2,598 2,603 2,623,552
2,644 2,650 2,669 2,697 2,700 2,702 2,709 2,712 2,722 2,728 2,788 2,790 2,795
2,796 2,797 2,821 2,824 2,849 2,853 2,855 2,861 2,871 2,884 2,914 2,951 2,954
2,957 2,985 2,986 2,996 2,998 20,060 20,106 20,130 20,185 20,364 20,416 20,441
20,485 20,551 20,567 20,844 200,252 202,184 206,929 208,050 208,102 21,057 21,166
21,239 21,316 21,353 21,450 21,650 21,805 21,870 21,911 216,014 22,128 22,376
22,560 22,846 220,461 220,681 220,724 220,753 226,380 226,760 23,097 23,157 23,178
23,246 23,318 23,376 23,405 23,445 23,479 23,672 23,689 23,776 23,938 234,065
235,906 237,419 238,616 24,252 24,371 24,495 24,499 24,524 24,683 24,709 24,764
24,921 24,935 243,553 244,902 248,625 25,111 25,214 25,240 25,278 25,459 25,490
25,492 25,828 250,527 251,884 252,263 26,069 26,081 26,290 26,411 26,561 26,569
26,674 26,681 26,726 26,987 264,243 268,689 27,033 27,100 27,527 27,987 270,331
278,810 28,124 28,460 28,598 28,754 280,141 281,061 281,594 282,094 287,359 288,324
29,130 29,598 29,725 3,018 3,054 3,061 3,086 3,111 3,112 3,127 3,137 3,140,509
3,194 3,200 3,210 3,218 3,230,085 3,249 3,256 3,261 3,312 3,314 3,333 3,344 3,352
3,362 3,388 3,456 3,495 3,516 3,555 3,560 3,562 3,587 3,600 3,638 3,644 3,649
3,692 3,713 3,718 3,757 3,763 3,789 3,820 3,836 3,838 3,862 3,884 3,897 3,924
3,928 3,931 3,948 3,983 3,996 30,676 304,036 308,536 31,010 316,113 32,015 32,199
32,891 32,971 324,365 324,382 324,791 329,852 33,145 33,161,440 33,308 33,429
33,978 34,633 34,705 34,926 348,241 35,094 35,451 35,715 35,914 35,943 35,968
352,400 357,692 359,758 36,015 36,694 36,702 37,050 37,109 37,303 37,999 371,666
375,542 377,822 377,949 38,176 38,566 38,571 38,719 38,817 38,893 38,920 386
39,572 39,843 39,857 39,864 394,558 395,711 397,295 4,003 4,013 4,025 4,044 4,067
4,079 4,087 4,108 4,118 4,124 4,136 4,167 4,180 4,206 4,218 4,219 4,233 4,245
4,252 4,271 4,274 4,310 4,352 4,354 4,388 4,397 4,402 4,419 4,443 4,448 4,482
4,496 4,520 4,581 4,599 4,611 4,620 4,677 4,684 4,704 4,729 4,743 4,748 4,754
4,782 4,822 4,823 4,836 4,875 4,879 4,890 4,891 4,917 4,927 4,928 4,931 4,946
4,953 4,969 4,970 4,980 4,986 40,367 40,616 41,047 41,517 417,048 42,321 42,687
42,757 42,874 425,560 434,670 448,661 45,454 46,040 46,504 46,815 47,164 47,590
47,764 47,787 47,908 477,889 48,143 48,441 49,231 49,459 5,001 5,007 5,027 5,031
5,059 5,075 5,094 5,109 5,115 5,202 5,228 5,237 5,253 5,269 5,296 5,319 5,347
5,349 5,394 5,436 5,468 5,486 5,487 5,517 5,541 5,571 5,579 5,599 5,630 5,647
5,694 5,734 5,747 5,786 5,806 5,807 5,828 5,929 5,972,356 50,573 51,486 52,164
52,440 52,630 52,731 52,917 53,178 54,236 54,878 55,535 558,632 56,294 56,773
56,984 57,033 58,173 58,288 58,315 58,602 580,552 584,297 585,567 59,094 59,371

59,934 6,006 6,055 6,095 6,096 6,132 6,145 6,148 6,189 6,190 6,201 6,215 6,233
6,372 6,385 6,387 6,442,416 6,461 6,485 6,496 6,529 6,561 6,601 6,629 6,648 6,686
6,722 6,733 6,781 6,785 6,787 6,800 6,877 6,882 6,896 6,899 6,938 6,956 6,960
60,102 60,258 60,749 603,080 61,266 61,779 61,895 622 63,236 63,336 63,511 63,578
631,565 64,047 64,118 64,410 64,749 64,808 64,837 64,871 65,902 65,952 66,520
66,711 66,893 66,946 665 666,658 67,964 671 676,551 68,911 69,356 69,757 696
7,054 7,070 7,094 7,105 7,111 7,121 7,133 7,151 7,156 7,164 7,188 7,190 7,297
7,344 7,365 7,451 7,498 7,534 7,552 7,562 7,582 7,699 7,725 7,805 7,828 7,859
7,914 7,915 7,927 70,637 70,790 708 71,254 71,975 711 712,884 719 73,549 730 738
74,448 75,187 75,314 75,850 752 76,676 76,870 76,981 77,145 77,360 77,970 78,456
79,487 793 797,608 8,043 8,092 8,101 8,104 8,113 8,181 8,207 8,254 8,405 8,410
8,472 8,505 8,529 8,545 8,683 8,704 8,792 8,816 8,900 80,606 81,550 818 82,194
834,510 84,905 848 85,069 85,652 863 87,518 87,851 87,936 871 876 88,407 88,496
89,031 89,199 890 896,372 9,005 9,041 9,046 9,057 9,113 9,159 9,194 9,237 9,362
9,366 9,460 9,465 9,481 9,489 9,493 9,494 9,509 9,710 9,743 9,746 9,775 9,781 910
915 917 919 926 93,465 938 96,223 96,237 961 97,766 97,850 973 98,653 99,775 994
-20 -36 -8 1,010 1,011 1,082,402 1,102 1,141 1,142,380 1,144 1,146 1,149 1,151
1,182 1,183 1,187 1,191 1,198 1,210,673 1,212 1,222 1,223 1,230 1,242,179 1,243
1,245 1,252 1,255 1,287 1,296 1,330 1,339 1,341 1,342 1,360,296 1,363 1,380 1,386
1,390 1,398 1,405 1,408 1,409 1,425 1,462 1,465 1,468 1,471 1,493,231 1,509 1,514
1,518 1,536 1,539 1,541 1,552,154 1,564,798 1,566 1,578,568 1,593 1,637 1,657
1,671 1,672 1,687 1,688,807 1,692 1,701 1,704 1,720 1,739 1,753 1,804 1,809 1,813
1,825 1,834,473 1,911 1,914 1,917 1,923 1,955 1,959 1,970,468 1,971 10,004 10,010
10,035 10,101 10,106 10,128 10,158 10,196 10,211 10,343 10,581 10,601 10,759
10,764 10,777 10,929 10,934 100,205 103,286 103,801 104,288 104,332 104,390 106,411
109,731 11,089 11,092 11,119 11,152 11,230 11,234 11,263 11,264 11,295 11,306
11,314 11,393 11,394 11,424 11,463 11,482 11,579 11,585 11,705 11,735 11,788
11,796 11,844 11,868 11,878 11,978 11,988 11,992 110,277 110,686 110,814 111,091
111,368 111,799 112,057 113,910 117,927 118,054 118,154 119,035 119,963 12,006
12,009 12,078 12,205 12,380 12,400 12,402 12,415 12,432 12,526 12,592 12,595
12,633 12,697 12,774 12,795 12,852 12,934 12,935 121,276 122,126 127,849 13,184
13,194 13,408 13,473 13,586 13,617 13,763 13,776 13,793 13,803 13,826 13,860
13,906 13,996 130,093 131,459 131,854 132,660 133,447 135,019 135,032 137,362
137,567 139,555 14,004 14,025 14,146 14,162 14,210 14,240 14,292 14,300 14,340
14,353 14,461 14,492 14,564 14,695 14,700 14,784 14,843 14,917 14,996 142,311
142,493 143,340 145,953 146,413 15,046 15,048 15,108 15,187 15,354 15,438 15,501
15,610 15,687 15,742 15,753 15,830 15,888 15,895 15,911 153,563 154,627 154,645
156,708 158,606 16,071 16,072 16,125 16,165 16,194 16,240 16,262 16,407 16,479
16,595 16,823 16,842 16,879 16,958 160,989 161,297 162,825 165,721 17,070 17,100
17,138 17,244 17,504 17,524 17,527 17,553 17,576 17,681 17,696 17,773 17,886
17,935 170,165 171,019 171,058 173,759 173,908 176,004 177,138 178,417 18,075
18,349 18,351 18,366 18,775 180,613 183,388 188,807 189,317 19,135 19,242 19,373
19,546 19,648 19,657 19,893 19,969 190,592 192,530 199,503 199,564 2,006 2,017
2,018 2,020 2,024 2,025 2,054 2,066 2,077 2,089 2,097 2,100 2,110 2,119 2,122
2,125 2,134 2,142,022 2,143 2,158 2,167 2,183 2,194 2,198 2,227 2,236 2,314 2,348
2,370 2,384 2,386 2,408 2,411 2,417 2,417,571 2,424 2,426 2,442 2,455 2,477 2,487
2,497 2,499 2,516 2,523,566 2,528 2,570 2,578 2,591 2,608 2,611 2,624,809 2,631
2,632 2,680 2,681 2,682 2,703 2,704 2,740 2,743 2,754 2,774 2,779 2,791 2,800

2,807 2,809 2,882,607 2,928 2,930 2,936 2,964,995 20,098 20,134 20,234 20,256
20,274 20,295 20,298 20,434 20,447 20,533 20,702 20,751 20,793 20,827 20,977
20,978 20,990 204,595 208,283 21,004 21,321 21,338 21,385 21,551 21,600 21,635
21,756 21,815 21,847 21,943 21,998 211,774 213,659 216,337 22,006 22,125 22,138
22,164 22,230 22,343 22,363 22,644 22,851 22,953 221,926 222,570 224,535 23,010
23,268 23,286 23,458 23,480 23,578 23,641 23,845 235,059 238,917 24,016 24,018
24,162 24,263 24,266 24,660 24,740 24,812 24,816 24,836 240,170 247,809 248,785
25,086 25,209 25,305 25,484 251,098 252,198 255,115 256,140 256,393 26,065 26,104
26,166 26,510 26,605 26,619 26,634 26,685 26,697 26,762 26,808 264,325 265,025
268,468 268,792 27,153 27,391 27,425 27,653 27,682 27,773 27,837 27,958 272,662
272,968 28,388 28,633 28,731 28,736 28,761 28,997 287,675 29,426 29,435 29,438
29,975 293,688 3,005 3,012 3,016 3,020 3,109,369 3,177 3,178 3,196 3,209 3,214
3,215 3,223 3,227 3,228,198 3,243 3,247 3,278 3,281 3,298 3,299 3,319 3,335 3,347
3,350 3,358 3,367 3,377 3,389 3,393,411 3,399 3,429 3,435 3,468 3,470 3,501 3,507
3,509 3,520 3,549 3,620 3,633 3,686 3,695,184 3,719 3,729 3,736 3,751 3,777 3,847
3,889 3,910 3,923 3,933 3,988 30,037 30,167 30,231 30,663 30,802 301,287 31,125
31,145 31,409 31,451 31,922 31,998 311,073 32,061 32,074 32,130 32,315 32,399
32,477 32,568 32,872 33,136 33,323 33,455 33,837 33,871 332,381 335,430 337,600
34,220 34,221 35,108 35,460 35,572 35,602 35,662 35,822 35,903 357,292 359,418
36,046 36,103 36,170 36,672 365,848 369,047 37,117 37,417 37,578 37,774 37,874
37,942 372,887 373,153 373,476 373,606 374,022 38,264 38,341 38,845 38,912 383,824
385,547 39,307,344 39,350 39,625 39,678 39,750 39,809 395,806 4,026 4,053 4,131
4,159 4,174 4,176 4,191 4,193 4,198 4,200 4,231 4,262 4,268 4,270 4,300 4,329
4,338 4,365 4,384 4,386 4,411 4,413 4,430 4,451 4,453 4,456 4,472 4,494 4,502
4,535 4,570 4,636 4,665 4,678 4,705 4,747 4,755 4,769 4,826 4,835 4,838 4,858
4,878 4,883 4,903 4,924 4,937 4,954 4,974 4,985 4,991 40,105 40,238 40,289 40,356
40,547 404,423 404,823 406,965 41,768 41,968 417,493 42,045 42,378 42,439 421,683
424,166 43,365 43,652 43,755 44,185 44,972 440,474 443,513 45,340 45,528 45,593
455,516 459,054 46,658 46,964 47,084 47,157 47,822 478,004 481,146 484,361 49,287
491,138 5,022 5,065 5,078 5,127 5,135 5,142 5,152 5,169 5,172 5,184 5,233 5,261
5,292 5,327 5,366 5,372 5,390 5,416 5,423 5,450 5,454 5,471 5,485 5,513 5,532
5,539 5,566 5,587 5,655 5,663 5,698 5,730 5,735 5,769 5,794 5,801 5,810 5,818
5,832 5,875 5,922 5,949 5,966 5,998 50,127 50,310 508,768 51,235 51,891 515,857
52,030 53,141 53,535 53,650 53,771 53,793 53,802 54,124 54,250 54,518 54,820
543,887 544,458 55,163 55,687 56,114 56,943 57,737 573,708 58,370 58,469 58,589
582,342 586,454 59,044 59,858 6,035 6,054 6,086 6,110 6,126 6,168 6,169 6,220
6,271 6,278 6,302 6,323 6,327 6,350 6,373 6,408 6,410 6,414 6,441 6,489 6,499
6,520 6,539 6,541 6,546 6,560 6,577 6,584 6,594 6,698 6,730 6,731 6,756 6,763
6,783 6,795 6,840 6,949 6,951 6,953 6,964 60,077 60,150 60,491 60,673 62,298
62,431 625,224 64,170 64,239 64,705 64,710 64,737 647 65,086 65,726 65,931 66,714
66,984 669,482 67,184 67,864 67,909 68,300 69,830 7,007 7,072 7,096 7,160 7,191,779
7,234 7,252 7,268 7,273 7,304 7,308 7,334 7,357 7,362 7,373 7,379,911 7,425 7,460
7,642 7,690 7,779 7,871 7,903 7,909 7,919 7,952 70,391 71,276 71,277 71,577 72,838
720,008 722 723 73,857 742,704 748 755 76,041 76,134 76,187 771 773,381 778,580
78,780 784 79,029 79,521 79,947 79,995 797 798 8,033 8,067 8,167 8,193 8,199 8,215
8,296 8,422 8,424 8,428 8,437 8,461 8,495 8,515 8,517 8,536 8,548 8,556 8,618
8,711 8,769 8,774 8,926 8,958 8,984 80,641 80,919 81,192 813,311 82,583 82,930
828,691 83,449 85,149 86,022 86,456 86,660 86,749 86,854 863,632 867,903 87,205

87,645 882 889 89,289 89,923 9,072 9,082 9,162 9,184 9,225 9,317 9,391 9,394 9,400 9,409 9,413 9,432 9,461 9,499 9,584 9,607 9,634 9,639 9,676 9,737 9,798 9,805 9,811 9,813 9,826 9,856 9,862 9,872 9,902 9,938 9,984 90,008 90,540 92,173 92,854 94,173 94,649 94,993 947 948 95,121 95,295 957 97,240 97,575 979 98,149 98,599 985 986 995 996

`time_o` a numeric vector

`time_d` a numeric vector

`timediff` a numeric vector

`FPIflow2` a numeric vector

`iszero` a logical vector

`correla` a numeric vector

`logtradebyGDP` a numeric vector

`gravity` a numeric vector

`morethangravity` a logical vector

Details

o is origin and d is destination. Correla is the correlation between the GDP growth rates of the origin and destination over the last 10 years. Comlang_off is TRUE if the two share a common official language. Comleg refers to a common legal origin between the two. Flow is trade flow, FPIflow2 is poorly named, it is actually FPI holdings.

Source

http://cpis.imf.org/ http://www.cepii.fr/CEPII/en/bdd_modele/presentation.asp?id=27 http://econ.worldbank.org/WBSITE/E

Examples

```
data(FPI2002Data)
## maybe str(FPI2002Data) ; plot(FPI2002Data) ...
```

---

fY_Rmask *density of Y*

Description

evaluate the density function for the general case

Usage

```
fY_Rmask(y, muY, a, b)
```

Arguments

y	values of Y
muY	a vector with entries of moments of Y
a	lower boundary of Y
b	upper boundary of Y

Details

no details needed

Value

real number

Note

no further notes

Author(s)

Yan-Xia

References

Equation (12) in Provost's paper

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

y<-4
muY<-c(3,3,5,5)
a<-2
b<-5
print(fY_Rmask(y,muY,a,b))
```

generalizedJointF *Purely used in getSampleFromMarginalDistributionOfUnmaskedData*

Description

See above

Usage

```
generalizedJointF(x, Vx, mu, s, rho_X, G_Point7,
GH_Quadrature, maxSize, choleskySpeed = TRUE, cores = 1, verbose = -1)
```

Arguments

x	List of vectors of points to estimate the fitted numerical density functions at.
Vx	List of samples from each vector
mu	List of means of each vector
s	List of standard deviations of each vector
rho_X	Correlation matrix of the vectors
G_Point7	Seven Points of Gaussian Hermite Quadrature
GH_Quadrature	Seven Weights of Gaussian Hermite Quadrature
maxSize	Used for a speed improvement, the larger this is the faster the function will work as it iterates over each bin, but within each bin vectorized operations take place. The restriction is memory. Typically floor(sqrt(1450000)) performs well on 32 Bit machines, but it is up to user discretion to find the optimal value for their machine.
choleskySpeed	TRUE to apply a slight speed improvement via a cholesky decomposition. Occasionally this will not work if the matrices are not able to be decomposed in such a manner, theoretically impossible for a correlation matrix but can occur with matrices that are almost singular.
cores	Number of cores to use in parallel processing, only works with 1 core on windows machines.
verbose	If greater than 1 it provides additional information to the console.

Value

An array of dimension corresponding to the dimension of each vector in x.

Author(s)

Luke Mazur

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, Vx, mu, s, rho_X, G_Point7, GH_Quadrature, maxSize,
    choleskySpeed = TRUE, cores = 1, verbose = -1)
{
    numberOfVectors <- length(x)
    if (length(Vx) != numberOfVectors) {
        stop("x and Vx must be the same length")
    }
    numberOfVectors <- length(x)
    if (length(mu) != numberOfVectors) {
        stop("x and mu must be the same length")
    }
    numberOfVectors <- length(x)
    if (length(s) != numberOfVectors) {
        stop("x and s must be the same length")
    }
    n <- unlist(lapply(1:numberOfVectors, FUN = function(i) {
        return(length(x[[i]]))
    }))
    fhat <- lapply(1:numberOfVectors, FUN = function(i) {
        return(kde(x = Vx[[i]], binned = TRUE))
    })
    if (verbose > 1) {
        print("calculating Nataf_rho matrix")
    }
    Nataf_rho <- matrix(rep(1, (numberOfVectors^2)), nrow = numberOfVectors,
        ncol = numberOfVectors)
    Nataf_rho[upper.tri(Nataf_rho, diag = TRUE)] <- NA
    Nataf_rho <- mclapply(1:numberOfVectors, mc.cores = cores,
        FUN = function(j) {
            return(lapply(1:numberOfVectors, FUN = function(i) {
                if (verbose > 1) {
                  print("row and column")
                  print(i)
                  print(j)
                }
                if (!is.na(Nataf_rho[i, j])) {
                  return(rho_0(Vx[[j]], Vx[[i]], mu[[j]], mu[[i]],
                    s[[j]], s[[i]], rho_X[i, j], G_Point7, GH_Quadrature,
                    verbose))
                } else {
                  return(NA)
                }
            }))
        })
    Nataf_rho <- unlist(Nataf_rho)
    Nataf_rho <- matrix(Nataf_rho, nrow = numberOfVectors, ncol = numberOfVectors)
```

```
diag(Nataf_rho) <- 1
Nataf_rho[upper.tri(Nataf_rho, diag = FALSE)] <- Nataf_rho[lower.tri(Nataf_rho,
    diag = FALSE)]
if (verbose > 1) {
    print("calculating y and phiy")
}
y <- lapply(1:numberOfVectors, FUN = function(i) {
    return(qnorm(pkde(x[[i]], fhat[[i]])))
})
phiy <- lapply(1:numberOfVectors, FUN = function(i) {
    return(exp(-((y[[i]])^2)/2)/sqrt(2 * pi))
})
if (verbose > 1) {
    print("calculating inverse of Nataf_rho")
}
A <- solve(Nataf_rho)
if (verbose > 1) {
    print("calculating all possible combinations")
}
allPossibleCombinationsX <- expand.grid(x)
allPossibleCombinationsY <- expand.grid(y)
allPossibleCombinationsPhiy <- expand.grid(phiy)
yIndependentPartOfPhi <- (1/(sqrt((2 * pi)^numberOfVectors *
    det(Nataf_rho))))
nrowAllPossibleCombinationsY <- nrow(allPossibleCombinationsY)
numberOfBins <- ceiling(nrowAllPossibleCombinationsY/maxSize)
binFactors <- cut(1:nrowAllPossibleCombinationsY, breaks = numberOfBins)
bins <- split(allPossibleCombinationsY, f = binFactors)
if (verbose > 1) {
    print(paste("There are ", numberOfBins, " bins."))
}
if (choleskySpeed) {
    cholA <- chol(A)
    yDependentPartOfPhi <- unlist(mclapply(1:numberOfBins,
        mc.cores = cores, FUN = function(i) {
            if (verbose > 1) {
              print(i)
            }
            temp <- as.matrix(bins[[i]])
            temp <- crossprod(cholA %*% t(temp))
            return(exp(-diag(temp)))
        }))
}
else {
    yDependentPartOfPhi <- unlist(mclapply(1:numberOfBins,
        mc.cores = cores, FUN = function(i) {
            if (verbose > 1) {
              print(i)
            }
            temp <- as.matrix(bins[[i]])
            temp <- temp %*% A %*% t(temp)
            return(exp(-diag(temp)))
        }))
```

```
    }
    if (verbose > 1) {
        print("Moving on to xAndPhiy")
    }
    xAndPhiy <- mclapply(1:numberOfVectors, mc.cores = cores,
        FUN = function(j) {
          return(dkde(allPossibleCombinationsX[, j], fhat[[j]])/allPossibleCombinationsPhiy[,
                j])
        })
    if (verbose > 1) {
        print("Taking products of xAndPhiy")
    }
    xAndPhiy <- unlist(apply(as.data.frame(xAndPhiy), 1, FUN = prod))
    f <- yIndependentPartOfPhi * yDependentPartOfPhi * xAndPhiy
    f <- array(f, dim = n)
    return(f)
  }
```

`getSampleBasedOnUnmaskedData` *Second core function the End-User uses to obtain the samples from the marginal distributions of the unmasked data*

Description

This function is used after the unmasked data have been obtained from the unmask function. If the Data Provider supplies all the means, standard deviations and the correlation matrix of the original data then these can be used as arguments. Otherwise, these are calculated using the mean of noise sample and the mean of the vector created by squaring each element of the noise sample.

A sample of the chosen size is then simulated from the estimated joint density function.

Usage

```
getSampleBasedOnUnmaskedData(meansOfNoises, meansOfSquaredNoises,
maskedVectors, unmaskedVectors, mu,
s, rho_X, cores = 1, size, verbose = -1)
```

Arguments

`meansOfNoises`	Used to calculate mu, s and rho_X if any of them are not supplied
`meansOfSquaredNoises`	Used to calculate mu, s and rho_X if any of them are not supplied
`maskedVectors`	List of masked vectors
`unmaskedVectors`	List of unmasked vectors, note that they are the unmasked vectors of the respective marginal variables. This is not to be confused with what the function returns, which is a sample based on the unmasked joint distribution

mu	List of means of unmasked vectors - if not supplied will be calculated using meansOfNoises and meansOfSquaredNoises
s	List of standard deviations of unmasked vectors - if not supplied will be calculated using meansOfNoises and meansOfSquaredNoises
rho_X	Correlation matrix of unmasked vectors - if not supplied will be calculated using meansOfNoises and meansOfSquaredNoises
cores	Passed to mclapply
size	Passed to actualPosition
verbose	If greater than 0 output is printed to tell the user at what stage the function is in, is also passed to many internal functions and will give more detailed output from them if it is greater than 1

Value

List of length corresponding to the number of original variables with each element of the list being a vector of length size of samples of synthetic data corresponding to the original variable. I.e. first element is a sample of synthetic data corresponding to the first original variable, second element is a sample of synthetic data corresponding to the second original variable etc.

Author(s)

Luke Mazur

References

no references

Examples

```
##outputNL1 <- getSampleBasedOnUnmaskedData(
##  meansOfNoises = list(Vx1$meanOfNoise, Vx2$meanOfNoise),
##  meansOfSquaredNoises = list(Vx1$meanOfSquaredNoise,
##                              Vx2$meanOfSquaredNoise),
##  maskedVectors = list(xstar1, xstar2),
##  unmaskedVectors = list(Vx1$unmaskedVariable, Vx2$unmaskedVariable),
##  verbose = 2, size = 1000)
## not a real example because ultimately in order to demonstrate this
## function the entire package functionality must be demonstrated
## this is demonstrated in the package example
```

---

getSampleFromMarginalDistributionOfUnmaskedData

Second function the End-User uses to obtain the samples from the marginal distributions of the unmasked data

---

Description

This function is used after the unmasked data have been obtained from the unmask function. If the Data Provider supplies all the means, standard deviations and the correlation matrix of the original data then these can be used as arguments. Otherwise, these are calculated using the mean of noise sample from and the mean of the vector created by squaring each element of the noise sample.

Usage

```
getSampleFromMarginalDistributionOfUnmaskedData(xLengths, meansOfNoises,
meansOfSquaredNoises, xStars, Vx, mu, s, rho_X, maxSize,
choleskySpeed = TRUE, cores = 1, size,
returnJointDensity = FALSE, verbose = -1)
```

Arguments

xLengths	List of integer number of points from which to sample the fitted density functions of each respective variable
meansOfNoises	Used to calculate mu, s and rho_X if any of them are not supplied
meansOfSquaredNoises	Used to calculate mu, s and rho_X if any of them are not supplied
xStars	List of masked vectors
Vx	List of unmasked vectors
mu	List of means of unmasked vectors - if not supplied will be calculated using meansOfNoises and meansOfSquaredNoises
s	List of standard deviations of unmasked vectors - if not supplied will be calculated using meansOfNoises and meansOfSquaredNoises
rho_X	Correlation matrix of unmasked vectors - if not supplied will be calculated using meansOfNoises and meansOfSquaredNoises
maxSize	Passed to generalizedJointF
choleskySpeed	Passed to generalizedJointF
cores	Passed to generalizedJointF
size	Passed to actualPosition
returnJointDensity	If TRUE, then the joint density function calculated in an intermediate step by generalizedJointF is returned as the second element in the list. If FALSE, then the returned list has only one element.
verbose	If greater than 0 output is printed to tell the user at what stage the function is in, is also passed to many internal functions and will give more detailed output from them if it is greater than 1

Value

List with first element equal to the samples from the marginal densities of the unmasked variables. If returnJointDensity is TRUE then returns a second element that is the jointDensityFunction of the unmasked variables.

Author(s)

Luke Mazur

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (xLengths, meansOfNoises, meansOfSquaredNoises, xStars,
    Vx, mu, s, rho_X, maxSize, choleskySpeed = TRUE, cores = 1,
    size, returnJointDensity = FALSE, verbose = -1)
{
    n <- length(xLengths)
    if (missing(mu)) {
        mu <- lapply(1:n, FUN = function(i) {
            mean(xStars[[i]])/meansOfNoises[[i]]
        })
    }
    if (missing(s)) {
        s <- lapply(1:n, FUN = function(i) {
            sqrt((mean(xStars[[i]]^2) - (meansOfSquaredNoises[[i]]) *
                mean(xStars[[i]])^2/(meansOfNoises[[i]])^2)/(meansOfSquaredNoises[[i]]))
        })
    }
    if (missing(rho_X)) {
        rho_X <- matrix(1, n, n)
        for (i in 1:n) {
            for (j in 1:n) {
                if (i != j) {
                  rho_X[i, j] <- (cov(xStars[[i]], xStars[[j]])/((meansOfNoises[[i]]) *
                    (meansOfNoises[[j]])))/(s[[i]] * s[[j]])
                  rho_X[j, i] <- rho_X[i, j]
                }
            }
        }
    }
    if (verbose > 1) {
        print(mu)
        print(s)
        print(rho_X)
    }
    if (verbose > 0) {
        print("finished estimating mu, s and rho_X if missing")
    }
    testBoundary <- lapply(1:n, FUN = function(i) {
        return(c(min(Vx[[i]]), max(Vx[[i]])))
```

```
    })
    testX <- lapply(1:n, FUN = function(i) {
        return(seq(from = (testBoundary[[i]])[1], to = (testBoundary[[i]])[2],
            by = ((testBoundary[[i]])[2] - (testBoundary[[i]])[1])/xLengths[[i]]))
    })
    G_Point7 <- c(-3.75043971768, -2.36675941078, -1.1544053948,
        0, 1.1544053948, 2.36675941078, 3.75043971768)
    GH_Quadrature <- c(0.000548268858737, 0.0307571239681, 0.240123178599,
        0.457142857143, 0.240123178599, 0.0307571239681, 0.000548268858737)
    if (verbose > 0) {
        print("calculating jointDensityFunction")
    }
    jointDensityFunction <- generalizedJointF(testX, Vx, mu,
        s, rho_X, G_Point7, GH_Quadrature, maxSize = floor(sqrt(1450000)),
        choleskySpeed, cores, verbose)
    if (verbose > 0) {
        print("finished calculation of jointDensityFunction")
    }
    boundaryVec <- unlist(testBoundary)
    finalOutput <- actualPosition(dim(jointDensityFunction),
        jointDensityFunction, boundaryVec, size = size)
    if (returnJointDensity) {
        return(list(sample = finalOutput, jointDensityFunction = jointDensityFunction))
    }
    return(list(sample = finalOutput))
}
```

GH_Quadrature *The seven weights used for Gaussian Hermite Quadrature*

Description

This is purely for user's curiousity, modifying this will not affect the workings of the package as it is either hardcoded in or taken as a parameter.

Usage

```
data("GH_Quadrature")
```

Format

The format is: num [1:7] 0.000548 0.030757 0.240123 0.457143 0.240123 ...

Examples

```
data(GH_Quadrature)
```

gRho *Purely used in rho_0*

Description

See above

Usage

```
gRho(x1, x2, mu1, mu2, s1, s2, rho12, star_rho12, fhat1, fhat2, G_Point7, GH_Quadrature)
```

Arguments

x1	See rho_0
x2	See rho_0
mu1	See rho_0
mu2	See rho_0
s1	See rho_0
s2	See rho_0
rho12	See rho_0
star_rho12	See rho_0
fhat1	See rho_0
fhat2	See rho_0
G_Point7	See rho_0
GH_Quadrature	See rho_0

Value

Numeric Scalar

Author(s)

Yan-Xia Lin

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
```

G_Point7 *The seven points used for Gaussian Hermite Quadrature*

Description

This is purely for user's curiousity, modifying this will not affect the workings of the package as it is either hardcoded in or taken as a parameter.

Usage

```
data("G_Point7")
```

Format

The format is: num [1:7] -3.75 -2.37 -1.15 0 1.15 ...

Examples

```
data(G_Point7)
```

lambda_Rmask *expectation of Legendre polynomial*

Description

calculate lambda function given by Equation (3) in Provost's paper

Usage

```
lambda_Rmask(muX, k)
```

Arguments

muX	a vector of moment X
k	the upper order of moment used in the Legendre polynomial

Details

no details needed

Value

real number

Note

no further notes

Author(s)

Yan-Xia

References

Equation (3) in provost's paper

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.


muX<-c(1,2,3,4,5)
k<-4
print(lambda_Rmask(muX,k))
```

`mask` *Core function the Data Provider uses to mask the confidential original data to give to the user*

Description

Use the multiplicative noise method to mask micro data and creates a vector of masked data, and creates a noise file as well.

Usage

```
mask(vectorToBeMasked, noisefile, noise,
lowerBoundAsGivenByProvider=min(vectorToBeMasked),
upperBoundAsGivenByProvider=max(vectorToBeMasked),
maxorder=100,EPS=1e-06)
```

Arguments

`vectorToBeMasked`	data vector. All entries must be numeric or categorical. Missing values (NAs) are not allowed. If vectorToBeMasked are categorical, the argument is given by factor(vectorToBeMasked).
`noisefile`	a binary output file. The file name is ended by .bin. This file contains the information of lowerBoundAsGivenByProvider, upperBoundAsGivenByProvider, maxorder, the levels of the categorical data if vectorToBeMasked is categorical and EPS.
`noise`	noise data used to mask vectorToBeMasked. The size of the noise data is the same as the size of vectorToBeMasked.

`lowerBoundAsGivenByProvider`
: The lower boundary used in evaluating the estimated density approximant. The default value is min(vectorToBeMasked). To protect min(vectorToBeMasked) and achieve an accurate estimate of the density function of vectorToBeMasked, the value of lowerBoundAsGivenByProvider is critical. In any case, the value of lowerBoundAsGivenByProvider should be less than min(vectorToBeMasked).

`upperBoundAsGivenByProvider`
: The upper boundary used in evaluating the estimated density approximant. The default value is max(vectorToBeMasked). To protect max(vectorToBeMasked) and achieve an accurate estimate of the density function of vectorToBeMasked, the value of upperBoundAsGivenByProvider is critical. In any case, the value of upperBoundAsGivenByProvider should be greater than max(vectorToBeMasked).

`maxorder`
: the maximum order of the moments in the sample-moment-based density approximate to be tested. The default value is 100

`EPS`
: a threshold value. The default value is 1e-06.

Details

This R function is used to mask micro data by the multiplicative noise method and to produce a binary noise file for R function unmask. This file contains a sample of noise and other relevant information required by R function unmask. The size of the sample of noise stored in the file is ten times the size of vectorToBeMasked.

It is up to the user of mask to write the masked data to a file to provide to the end user.

Value

Returns a list with two elements.

`ystar`
: masked data

`noisefile`
: the title of a file containing the information of noise

Author(s)

Yan-Xia Lin

References

Lin, Yan-Xia and Fielding, Mark James (2015). MaskDensity14: An R Package for the Density Approximant of a Univariate Based on Noise Multiplied Data, SoftwareX (accepted)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

set.seed(123)
n=10000
y <- rmulti(n=10000, mean=c(30, 50), sd=c(4,2), p=c(0.3, 0.7))
```

```
      # y is a sample drawn from Y.
noise<-rmulti(n=10000, mean=c(80, 100), sd=c(5,3), p=c(0.6, 0.4))
      # noise is a sample drawn from C.
a1<-runif(1, min=min(y)-2,max=min(y))
b1<-runif(1, min=max(y), max=max(y)+2)
ymask<-mask(vectorToBeMasked = y, noisefile=file.path(tempdir(),"noise.bin"), noise,
lowerBoundAsGivenByProvider=a1, upperBoundAsGivenByProvider=b1)
write(ymask$ystar, file.path(tempdir(),"ystar.dat"))
```

`maskBatch` *A batch function that the Data Provider can use to mask the confidential original data to give to the user, all in one batch*

Description

Use the multiplicative noise method to mask micro data and creates a number of masked data vectors and a number of matching noisefiles.

Usage

```
maskBatch(listOfVectorsToBeMasked,
                      listOfNoisefiles,
                      listOfNoises,
                      listOfLowerBoundsAsGivenByProvider,
                      listofUpperBoundsAsGivenByProvider,
                      maxorder = 100, EPS = 1e-06)
```

Arguments

`listOfVectorsToBeMasked`
: data vectors. All entries in each vector must be numeric or categorical. Missing values (NAs) are not allowed. If any of the vectors are categorical, they should be made factors.

`listOfNoisefiles`
: binary output files. The file names are ended by .bin. This file contains the corresponding information from listOfLowerBoundsAsGivenByProvider, listofUpperBoundsAsGivenByProvider, maxorder, the levels of the categorical data if the corresponding vector from listOfVectorsToBeMasked is categorical, and EPS.

`listOfNoises`
: noise data vectors used to mask the corresponding elements of listOfVectorsToBeMasked. The size of the noise data is the same as the size of the corresponding element of listOfVectorsToBeMasked. Unlike in the function mask no list elements may be omitted, instead if one wants to supply the default value as for the function mask one should place the string "nullString" in the relevant position.

`listOfLowerBoundsAsGivenByProvider`
: The lower boundaries used in evaluating the estimated density approximant. The default value is the minimum of the corresponding element of listOfVectorsToBeMasked, and for this to be used one should place the string "nullString"

in the corresponding place. To protect the minimum of the corresponding element of listOfVectorsToBeMasked and achieve an accurate estimate of the density function of that element, the value of the element of listOfLowerBoundsAsGivenByProvider is critical. In any case, the value of the corresponding element of listOfLowerBoundsAsGivenByProvider should be less than the minimum of the corresponding element of listOfVectorsToBeMasked.

`listofUpperBoundsAsGivenByProvider`
: The upper boundaries used in evaluating the estimated density approximant. The default value is the maximum of the corresponding element of listOfVectorsToBeMasked, and for this to be used one should place the string "nullString" in the corresponding place. To protect the maximum of the corresponding element of listOfVectorsToBeMasked and achieve an accurate estimate of the density function of that element, the value of the element of listofUpperBoundsAsGivenByProvider is critical. In any case, the value of the corresponding element of listofUpperBoundsAsGivenByProvider should be greater than the maximum of the corresponding element of listOfVectorsToBeMasked.

`maxorder`
: the maximum order of the moments in the sample-moment-based density approximate to be tested. The default value is 100

`EPS`
: a threshold value. The default value is 1e-06.

Details

This R function is used to mask micro data by the multiplicative noise method and to produce binary noise files for R function unmask. Each file contains a sample of noise and other relevant information required by R function unmask. The size of the sample of noise stored in each file is ten times the size of the corresponding element of listOfVectorsToBeMasked.

It is up to the user of maskBatch to write the masked data vectors to a set of files to provide to the end user.

Calling maskBatch is the equivalent of calling mask once for each variable.

Value

Returns a list of lists, each list contains two elements.

`ystar`
: masked data

`noisefile`
: the title of a file containing the information of noise

Author(s)

Luke Mazur

References

Lin, Yan-Xia and Fielding, Mark James (2015). MaskDensity14: An R Package for the Density Approximant of a Univariate Based on Noise Multiplied Data, SoftwareX (accepted)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

set.seed(123)
y1 <- rmulti(n=10000, mean=c(30, 50), sd=c(4,2), p=c(0.3, 0.7))
      # y1 is a sample drawn from Y.
noise1 <-rmulti(n=10000, mean=c(80, 100), sd=c(5,3), p=c(0.6, 0.4))
      # noise is a sample drawn from C.
a1<-runif(1, min=min(y1)-2,max=min(y1))
b1<-runif(1, min=max(y1), max=max(y1)+2)

set.seed(123)
y2 <- rmulti(n=10000, mean=c(30, 50), sd=c(4,2), p=c(0.4, 0.6))
      # y2 is a sample drawn from Y.
noise2<-rmulti(n=10000, mean=c(80, 100), sd=c(5,3), p=c(0.5, 0.5))
      # noise2 is a sample drawn from C.
a2<-runif(1, min=min(y2)-2,max=min(y2))
b2<-runif(1, min=max(y2), max=max(y2)+2)


ymaskBatch <- maskBatch(listOfVectorsToBeMasked = list(y1,y2),
listOfNoisefiles=list(file.path(tempdir(),"noise1.bin"),file.path(tempdir(),"noise2.bin")),
listOfNoises=list(noise1,noise2),
listOfLowerBoundsAsGivenByProvider=list(a1,a2),
listofUpperBoundsAsGivenByProvider=list(b1,b2))

fileNamesToWrite <- list(file.path(tempdir(),"ystar1.dat"), file.path(tempdir(),"ystar2.dat"))
for(i in 1:2) {
write((ymaskBatch[[i]])$ystar, fileNamesToWrite[[i]])
}
```

`moments` *moment*

Description

calculate the value of the moments of Y

Usage

```
moments(y, order = 8)
```

Arguments

`y`	a sample from Y
`order`	the maximum of order of the moments used

Details

no details needed

Value

a vector

Note

no further notes

Author(s)

Yan-Xia Lin

References

no reference needed

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

y<-rnorm(100,0,1)
print(moments(y, order=8))
```

`positions` *Function for finding the positions of the node representing the points at which to sample from the kernel density estimate.*

Description

Purely used by EQsampleDensity and sampleDensity

Usage

```
positions(x, vectorL)
```

Arguments

`x`	Placeholder
`vectorL`	Vector of lengths of the number of nodes for each dimension

Author(s)

Jordan Morris

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, vectorL)
{
    backwardNumber <- 1
    forwardNumber <- 1
    for (i in 1:length(vectorL)) {
        backwardNumber <- backwardNumber * vectorL[i]
    }
    backwardNumber <- backwardNumber/vectorL[1]
    y1 <- rep(x[1:vectorL[1]], each = backwardNumber)
    y <- y1
    forwardNumber <- forwardNumber * vectorL[1]
    if (length(vectorL) - 2 <= 0) {
        y1 <- rep(x[vectorL[length(vectorL) - 1] + 1:vectorL[length(vectorL)]],
            forwardNumber)
        y <- cbind(y, y1)
    }
    else {
        a1 <- 0
        b1 <- 0
        for (i in 1:(length(vectorL) - 2)) {
            backwardNumber <- backwardNumber/vectorL[i + 1]
            a1 <- a1 + vectorL[i]
            b1 <- a1 + vectorL[i + 1]
            a <- a1 + 1
            b <- b1
            y1 <- rep(x[a:b], each = backwardNumber)
            y1 <- rep(y1, forwardNumber)
            forwardNumber <- forwardNumber * vectorL[i + 1]
            y <- cbind(y, y1)
        }
        a1 <- a1 + vectorL[length(vectorL) - 1]
        b1 <- a1 + vectorL[length(vectorL)]
        a <- a1 + 1
        b <- b1
        y1 <- rep(x[a:b], forwardNumber)
        y <- cbind(y, y1)
    }
    return(y)
  }
```

P_Rmask *Legendre polynomial*

Description

An R code for a Legendre polynormal of degree k in x

Usage

```
P_Rmask(x, k)
```

Arguments

x	variable in the polynomial
k	the order in the polynormal

Details

no details needed

Value

real number

Note

no further notes

Author(s)

Yan-Xia Lin

References

Equation (2) in Provost paper

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.

## The function is currently defined as

x<-5
k<-4
print(P_Rmask(x,k))
```

qkdeSorted *A slightly altered version of qkde from package ks*

Description

Qkde complains if the output is not sorted. Theoretically it is impossible for this to happen, but in practise floating point error can allow this to occur. This function is a cheap work-around to this problem.

Usage

```
qkdeSorted(p, fhat)
```

Arguments

p	See qkde in package ks
fhat	See qkde in package ks

Details

See qkde in package ks

Value

See qkde in package ks

Author(s)

Luke Mazur

References

see package ks

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (p, fhat)
{
    if (any(p > 1) | any(p < 0))
        stop("p must be <= 1 and >= 0")
    cumul.prob <- pkde(q = fhat$eval.points, fhat = fhat)
    ind <- findIntervalSorted(x = p, vec = cumul.prob)
    quant <- rep(0, length(ind))
    for (j in 1:length(ind)) {
```

```
        i <- ind[j]
        if (i == 0)
            quant[j] <- fhat$eval.points[1]
        else if (i >= length(fhat$eval.points))
            quant[j] <- fhat$eval.points[length(fhat$eval.points)]
        else {
            quant1 <- fhat$eval.points[i]
            quant2 <- fhat$eval.points[i + 1]
            prob1 <- cumul.prob[i]
            prob2 <- cumul.prob[i + 1]
            alpha <- (p[j] - prob2)/(prob1 - prob2)
            quant[j] <- quant1 * alpha + quant2 * (1 - alpha)
        }
    }
    return(quant)
}
```

`rho_0` *Purely used in generalizedJointF*

Description

See above

Usage

```
rho_0(x1, x2, mu1, mu2, s1, s2, rho12, fhat1, fhat2,
G_Point7, GH_Quadrature, verbose = -1)
```

Arguments

`x1`	Sample from vector 1
`x2`	Sample from vector 2
`mu1`	Mean of vector 1
`mu2`	Mean of vector 2
`s1`	Standard deviation of vector 1
`s2`	Standard deviation of vector 2
`rho12`	Correlation between vector 1 and vector 2
`fhat1`	kernel density estimated function for vector 1
`fhat2`	kernel density estimated function for vector 2
`G_Point7`	Seven Points of Gaussian Hermite Quadrature
`GH_Quadrature`	Seven Weights of Gaussian Hermite Quadrature
`verbose`	Provides output if greater than 1

Details

Calculates the Nataf_Rho matrix required in the nataf transformation.

Value

Returns the Nataf_Rho correlation matrix

Author(s)

Yan-Xia Lin

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
```

`rmulti` *Simple way to generate noise*

Description

Generates noise by selecting from a mixture distribution of normal densities with means, standard deviations and probabilities provided

Usage

```
rmulti(n, mean, sd, p)
```

Arguments

`n`	Size of the noise vector to be outputted
`mean`	Vector of means corresponding to each component density
`sd`	Vector of standard deviations corresponding to each component density
`p`	Probability of selecting from each component distribution

Value

Vector of random numbers

Author(s)

Yan-Xia Lin

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (n, mean, sd, p)
{
    x <- rnorm(n)
    u <- sample(1:length(mean), n, prob = p, replace = T)
    for (i in 1:length(mean)) x[u == i] = mean[i] + sd[i] * x[u ==
        i]
    return(x)
  }
```

---

sampleDensity *A function used to simulate a sample from a kernel function, where the kernel function is determined by a given sample.*

Description

Very similar to EQsampleDensity without the restriction of having the nodes equally spaced.

Usage

```
sampleDensity(sx, boundaryVec, NoNote = 151, size = 100)
```

Arguments

sx	Matrix where each column corresponds to a vector
boundaryVec	Vector of boundaries of the columns of sx in the order from_1, to_1, from_2, to_2 etc
NoNote	Number of nodes
size	Size of the sample

Value

See actualPosition

Author(s)

Jordan Morris

References

no references

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (sx, boundaryVec, NoNote = 151, size = 100)
{
    n <- NoNote
    SXdim <- length(sx[1, ])
    vectorL <- rep(n, SXdim)
    XP <- NULL
    for (i in 1:SXdim) {
        a <- boundaryVec[2 * i - 1]
        b <- boundaryVec[2 * i]
        xposition <- seq(from = a, to = b, by = (b - a)/(n -
            1))
        XP <- c(XP, xposition)
    }
    NotePositions <- positions(XP, vectorL)
    H <- Hpi(x = sx)
    fhat <- kde(x = sx, H = H)
    prob <- predict(fhat, x = NotePositions)
    outSample <- actualPosition(vectorL, prob, boundaryVec, size)
    return(outSample)
  }
```

---

unmask *First core function used by End-User*

Description

Use the sample-moment-based density approximant method to estimate the density function of univariate distributions based noise multiplied data.

Usage

```
unmask(maskedVectorToBeUnmasked, noisefile)
```

Arguments

maskedVectorToBeUnmasked	masked data. The masked data were generated by R Function mask.
noisefile	Noise file containing a sample of the noise used to mask maskedVectorToBeUnmasked from R function mask

Details

unmask is fully described in Lin and Fielding (2015). The theory used to support unmask can be found in Lin (2014). unmask implements the sample-moment-based density approximate method the estimated the smoothed density function of the original data based on their make data maskedVectorToBeUnmasked. The output of the function unmask is a set of sample data from the estimated mouthed density function. The size of the output is the same as that of the original data that were masked by the multiplicative noise and yielded maskedVectorToBeUnmasked.

Value

Returns a list with four elements.

unmaskedVariable	vector of unmasked data
outMeanOfNoise	sample mean of the noise
outMeanOfSquaredNoise	sample mean of the squared noise
prob	vector mass function returned if the original data are categorical

Author(s)

Yan-Xia Lin

References

Lin, Yan-Xia (2014). Density approximant based on noise multiplied data. In J. Domingo-Ferrer (Eds.), Privacy in Statistical Databases 2014, LNCS 8744, Springer International Publishing Switzerland, 2014, pp. 89-104. Lin, Yan-Xia and Fielding, Mark James (2015). MaskDensity14: An R Package for the Density Approximant of a Univariate Based on Noise Multiplied Data, SoftwareX 34, 3743, doi:10.1016/j.softx.2015.11.002

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

#Example 1:
set.seed(123)
n=10000

y <- rmulti(n=10000, mean=c(30, 50), sd=c(4,2), p=c(0.3, 0.7))
```

```
      # y is a sample drawn from Y.
noise<-rmulti(n=10000, mean=c(80, 100), sd=c(5,3), p=c(0.6, 0.4))
      # noise is a sample drawn from C.


a1<-runif(1, min=min(y)-2,max=min(y))
b1<-runif(1, min=max(y), max=max(y)+2)
ymask<-mask(vectorToBeMasked=y, noisefile=file.path(tempdir(),"noise.bin"), noise,
lowerBoundAsGivenByProvider=a1, upperBoundAsGivenByProvider=b1)
write(ymask$ystar, file.path(tempdir(),"ystar.dat")) # Create masked data and noise.bin.
         # The two files can be issued to the public.


      # After received the two files "ystar.dat" and
      # noise.bin, the data user can use the following code to
      # obtain the synthetic data of the original data.

ystar <- scan(file.path(tempdir(),"ystar.dat"))
y1 <- unmask(maskedVectorToBeUnmasked=ystar, noisefile=file.path(tempdir(),"noise.bin"))
sample<-y1$unmaskedVariable
   # y1$unmaskedVariable gives the  synthetic data of the
   # original data y.  The size of the synthetic data is  the
   # same as that of y
plot(density(y1$unmaskedVariable), main="density(ymask)", xlab="y")
   # the plot of the approximant of $f_Y$

#Example 2:

set.seed(124)
n<-2000
a<-170
b<-80
y<-rbinom(n, 1, 0.1)+1
noise<-(a+b)/2+ sqrt(1+(a-b)^2/4)*rnorm(n, 0,1)
noise[noise<0]<- - noise[noise<0]

ymask<-mask(vectorToBeMasked=factor(y), noisefile=file.path(tempdir(),"noise.bin"), noise,
lowerBoundAsGivenByProvider=0,upperBoundAsGivenByProvider=3)
      # using factor(y) because y is a categorical variable
write(ymask$ystar, file.path(tempdir(),"ystar.dat"))

ystar<-scan(file.path(tempdir(),"ystar.dat"))
y1 <- unmask(maskedVectorToBeUnmasked=ystar, noisefile=file.path(tempdir(),"noise.bin"))
unmaskY<-y1$unmaskedVariable  # synthetic data
mass_function<-y1$prob  # estimated mass function
```

`unmaskAndGetSampleBatch`

Batch function that allows the user to unmask a number of variables as well as to sample from the estimated joint density function. Serves as a wrapper for the first and second functions the End-User calls to obtain the samples from the marginal distributions of the unmasked data.

Description

Use the sample-moment-based density approximant method to estimate the density function of univariate distributions based noise multiplied data. Afterwards if the Data Provider supplies all the means, standard deviations and the correlation matrix of the original data then these can be used as arguments. Otherwise, these are calculated using the mean of noise sample and the mean of the vector created by squaring each element of the noise sample.

A sample of the chosen size is then simulated from the estimated joint density function.

Usage

```
unmaskAndGetSampleBatch(listOfMaskedVectorsToBeUnmasked,
                                    listOfNoisefiles,
                                    mu, s, rho_X,
                                    cores = 1, size,
                                    verbose = -1,
                                    onlyUnmasked = FALSE)
```

Arguments

`listOfMaskedVectorsToBeUnmasked`	list of masked vectors. The masked data were generated by R Function mask, or maskBatch.
`listOfNoisefiles`	Noise files containing a sample of the noise used to mask the vectors in listOfMaskedVectorsToBeUnmasked from R function mask, or maskBatch
`mu`	List of means of unmasked vectors - if not supplied will be estimated
`s`	List of standard deviations of unmasked vectors - if not supplied will be estimated
`rho_X`	Correlation matrix of unmasked vectors - if not supplied will be estimated
`cores`	Passed to mclapply
`size`	Passed to actualPosition
`verbose`	If greater than 0 output is printed to tell the user at what stage the function is in, is also passed to many internal functions and will give more detailed output from them if it is greater than 1
`onlyUnmasked`	If true then only the output from unmask is returned. Effectively this makes the function "unmaskBatch" - so to speak. False by default.

Details

unmask is fully described in Lin and Fielding (2015). The theory used to support unmask can be found in Lin (2014). unmask implements the sample-moment-based density approximate method the estimated the smoothed density function of the original data based on their make data masked-VectorToBeUnmasked. The output of the function unmask is a set of sample data from the estimated mouthed density function.

Using this function is the equivalent of calling unmask once for each variable and then calling getSampleBasedOnUnmaskedData

Value

If onlyUnmasked is false then returns a list with two elements, both of which are lists.

`unmaskedOutputs`
: list containing the three (four if categorical) outputs from the function unmask namely: unmaskedVariable, meanOfNoise, meanOfSquaredNoise, prob

`getSampleOutputs`
: list containing the output from getSampleBasedOnUnmaskedData which is a list containing vectors corresponding to samples for each variable

If onlyUnmasked is true then returns a list with only one element:

`unmaskedOutputs`
: list containing the three (four if categorical) outputs from the function unmask namely: unmaskedVariable, meanOfNoise, meanOfSquaredNoise, prob

Author(s)

Luke Mazur

References

Lin, Yan-Xia (2014). Density approximant based on noise multiplied data. In J. Domingo-Ferrer (Eds.), Privacy in Statistical Databases 2014, LNCS 8744, Springer International Publishing Switzerland, 2014, pp. 89-104. Lin, Yan-Xia and Fielding, Mark James (2015). MaskDensity14: An R Package for the Density Approximant of a Univariate Based on Noise Multiplied Data, SoftwareX 34, 3743, doi:10.1016/j.softx.2015.11.002

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

##outputNL1 <- unmaskAndGetSampleBatch(listOfMaskedVectorsToBeUnmasked = ##list(ystar1, ystar2),
##                                     listOfNoisefiles =
## (file.path(tempdir(),"noise1.bin"),file.path(tempdir(),"noise2.bin")),
##                                     cores = 1, size = 1000,
##                                     verbose = 2)
## not a real example because ultimately in order to demonstrate this
```

```
## function the entire package functionality must be demonstrated
## this is demonstrated in the package example
```

Index