# Introduction

This will briefly introduce the steps for taking random samples from a finite population. There are two major types of sampling plans: probabilistic and non-probabilistic sampling plans. The probabilistic sampling plans generate statistically valid data based on which one can perform inferential statistical analysis.

In this project, we consider three probabilistic sampling plans: simple random sampling (SRS), stratified sampling, and systematic sampling. We will use the loan default status as a reference variable to assess the performance of the three sampling plans.

The \*\*Bank loan\*\* data set will be treated as a finite population. After we load the data to R, we will perform some data manipulation and exploratory data analysis to identify a stratification variable for stratified sampling.

```{r}

loan01 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational01.csv", header = TRUE)[, -1]

loan02 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational02.csv", header = TRUE)[, -1]

loan03 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational03.csv", header = TRUE)[, -1]

loan04 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational04.csv", header = TRUE)[, -1]

loan05 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational05.csv", header = TRUE)[, -1]

loan06 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational06.csv", header = TRUE)[, -1]

loan07 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational07.csv", header = TRUE)[, -1]

loan08 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational08.csv", header = TRUE)[, -1]

loan09 = read.csv("https://pengdsci.github.io/datasets/SBAloan/w06-SBAnational09.csv", header = TRUE)[, -1]

loan = rbind(loan01, loan02, loan03, loan04, loan05, loan06, loan07, loan08, loan09)

```

# Identifying A Stratification Variable

We use the 2-digit NAICS code as the stratification variable to define the sub-population.

## 2-Digit NAICS Codes

We will still use NAICS to create a categorical variable that is operational in practice. Recall that in the last note, we summarized the distribution of NAICS in the following table.

```{r data-size}

naics =as.character(loan$NAICS) # make a character vector

N=length(naics) # find the size of the data.

f.table = -sort(-table(naics)) # sort the vector in descending order

n = length(f.table) # find the number of distinct industries

n.0 = sum(f.table < 900) # industry with less than 0.1% of the population size

# A note of length of R variable name: the latest version of R has changed

# the maximum length of variable names from 256 characters to 10,000 characters.

# We should try our best to give meaningful names to R variables.

kable(cbind(Population.size = N, Number.of.Industries=n, Sub.Pop.less.1000 = n.0))

```

The course web page provides two useful tables. One of the tables <https://sta490.s3.amazonaws.com/w03-NAICS-Categories.jpg> in the article listed categories based on the first digits of NAICS code. The other related table gives the loan default rate in the corresponding industries <https://sta490.s3.amazonaws.com/w03-NAICS-Default-Rates.jpg>. You can download these two tables to your local drive and include them in your R Markdown document if you want to practice and reproduce this report.

For the convenience of referring to these tables, I include these two tables in this document.

```{r echo=FALSE, fig.cap="List of all industries using the first two digits of the NAICS code", fig.width=3, fig.length=5, fig.align = "center"}

include\_graphics("img/w08-NAICS-Categories.jpg")

```

```{r echo=FALSE, fig.cap="List of all industries using the first two digits of the NAICS code and the corresponding loan default rates", fig.width=3, fig.length=5, fig.align = "center"}

include\_graphics("img/w08-NAICS-Default-Rates.jpg")

```

Next, we explore the frequency distribution of the 2-digit NAICS codes and decide the potential combinations of categories with a small size.

```{r}

NAICS.2.digits = substr(loan$NAICS, 1, 2) # extract the first two digits of the NAICS code

loan$NAICS2Digit = NAICS.2.digits # add the above two-digit variable the loan data

freq.table = table(NAICS.2.digits) # create a regular frequency table

kable(freq.table) # convert the regular frequency table to kable

```

Several patterns you observe from the above table:

\* 201948 businesses do not have a NAICS code. Since I will use the 2-digit NAICS code to stratify the population. This variable will be included in the study population that will be defined soon.

\* Several categories (21, 22, 49, 55, 92) have relatively small sizes. Since categories 48 and 49 are both transportation and warehouse industries, we will combine the two as indicated in the above two tables.

\* As we can see from the above two tables, several industries have different codes. We will combine these codes. In other words, we need to modify the 2-digit code to define the final stratification for stratified sampling.

## Combining Categories

We now combine categories suggested in the above NAICS tables in a meaningful way. Before we combine the NAICS codes, we present an example to illustrate how to combine categories using R.

```{}

cate.vec0=c(1,4,3,6,7,3,6,5,4,6,4,5,8,9,4,3,4,7,3) # vector of category labels

cate.vec=c(1,4,3,6,7,3,6,5,4,6,4,5,8,9,4,3,4,7,3) # a copy of the vector of category labels

labs.2.collapse = c(1,6,7) # define a vector to store categories {1,6,7}

logic.vec=cate.vec %in% labs.2.collapse # TRUE/FALSE ==> match not no-match

cate.vec[logic.vec] = 99 # if matches (i.e., 1, 5, 7), the value

# will be replaced by 99

matx=rbind(cate.vec0=cate.vec0, cate.vec=cate.vec) # check the results

colnames(matx) = 1:length(cate.vec) # next kable() function requires a column names

kable(matx)

```

We now combine the following sets of 2-digit NAICS codes: (31, 32, 33) will be relabeled as \*\*313\*\*, (48, 49) as \*\*489\*\*, and (44, 45) as \*\*445\*\*. We use strNACIS to denote the resulting stratification variable.

```{r}

cate.31.33=c("31","32","33") # combining categories 31, 32, and 33

cate.48.49 = c("48", "49")

cate.44.45 = c("44", "45")

NAICS2Digit0 = loan$NAICS2Digit # extract the 2-digit NAICS

NAICS2Digit = loan$NAICS2Digit # extract the 2-digit NAICS-copy

## combining 31,32, and 33

logic.31.33=NAICS2Digit %in% cate.31.33 # Identify the three categories: 31, 32, 33.

NAICS2Digit[logic.31.33] = 313 # replace 31, 32, 33 with 313

## combining 44 and 45

logic.44.45=NAICS2Digit %in% cate.44.45 # Identify the three categories: 44 and 45.

NAICS2Digit[logic.44.45] = 445

## combining 48 and 49

logic.48.49=NAICS2Digit %in% cate.48.49 # Identify the three categories: 48 and 49.

NAICS2Digit[logic.48.49] = 489

# check the result

# list(tab.naics0=table(NAICS2Digit0),tab.naics=table(NAICS2Digit)) # check the result

### final step

loan$strNAICS = NAICS2Digit

kable(table(loan$strNAICS))

```

## Loan Default Rates by Industry

We now find the loan default rates by industry defined by the stratification variable strNAICS. The loan default status can be defined by the variable MIS\_Status. The following table gives the default rates by industry.

```{r}

x.table = table(loan$strNAICS, loan$MIS\_Status)

no.lab = x.table[,1] # first column consists of unknown default label

default = x.table[,2]

no.default = x.table[,3]

default.rate = round(100\*default/(default+no.default),1)

default.status.rate = cbind(no.lab = no.lab,

default = default,

no.default = no.default,

default.rate=default.rate)

kable(default.status.rate)

```

## Study Population

Based on the above frequency distribution of the modified 2-digit NAICS codes (the 3-digit codes are combined categories). We use the following inclusion rule to define the study population: excluding small size categories 20, 21, 55, 92, and unclassified businesses with NAICS code 0.

```{r}

del.categories = c("0", "21", "22", "55", "92") # categories to be deleted in

# the original population

del.obs.status = !(loan$strNAICS %in% del.categories) # deletion status. ! negation operator

study.pop = loan[del.obs.status,] # excluding the categories

kable(t(table(study.pop$strNAICS))) # Checking correctness operation

```

So we have defined our study population! The new population size is 694216.

# Sampling Plans

In this section, we are implementing three sampling plans. In each sampling plan, we select 3000 observations in the corresponding samples. The sample size is slightly less than 5% of the study population size. We use without-replacement sampling plans for all three probabilistic sample plans.

## Simple Random Sampling

We define a sampling list and add it to the study population.

```{r}

study.pop$sampling.frame = 1:length(study.pop$GrAppv) # sampling list

# names(study.pop) # checking the sampling list variable

sampled.list = sample(1:length(study.pop$GrAppv), 3000) # sampling the list

SRS.sample = study.pop[sampled.list,] # extract the sampling units (observations)

## dimension check

dim(SRS.sample ) # checking the sample size

```

## Systematic sampling

We can choose a random integer that is less than the jump size (231 in our case) to guarantee to obtain enough samples.

```{r}

jump.size = dim(study.pop)[1]%/%3000 # find the jump size in the systematic sampling

# jump.size

rand.starting.pt=sample(1:jump.size,1) # find the random starting value

sampling.id = seq(rand.starting.pt, dim(study.pop)[1], jump.size) # sampling ID

#length(sampling.id)

sys.sample=study.pop[sampling.id,] # extract the sampling units of systematic samples

dim(sys.sample)

```

Because the jump size involves rounding error and the population is large, the actual systematic sample size is slightly different from the target size. In this report, I used the integral part of the actual jump size. The actual systematic sampling size is slightly bigger than the target size. We can take away some records from the systematic sample to make the size to be equal to the target size.

## Stratified Sampling

We take an SRS from each individual stratum. The sample size should be approximately proportional to the size of the corresponding stratum.

First, we calculate the SRS size for each stratum and then take the SRS from the corresponding stratum. The following table shows the sub-sample sizes.

```{r}

freq.table = table(study.pop$strNAICS) # frequency table of strNAICS

rel.freq = freq.table/sum(freq.table) # relative frequency

strata.size = round(rel.freq\*3000) # strata size allocation

strata.names=names(strata.size) # extract strNAICS names for accuracy checking

#list(strata.names=names(strata.size),

#strata.size = strata.size,

#sample.size = sum(strata.size))

kable(t(strata.size))

```

When coding, we use a loop to take simple random samples from each stratum and then combine them to get the stratified sample.

```{r }

strata.sample = study.pop[1,] # create a reference data frame

strata.sample$add.id = 1 # add a temporary ID to because in the loop

# i =2 # testing a single iteration

for (i in 1:length(strata.names)){

ith.strata.names = strata.names[i] # extract data frame names

ith.strata.size = strata.size[i] # allocated stratum size

# The following code identifies observations to be selected

ith.sampling.id = which(study.pop$strNAICS==ith.strata.names)

ith.strata = study.pop[ith.sampling.id,] # i-th stratified population

ith.strata$add.id = 1:dim(ith.strata)[1] # add sampling list/frame

# The following code generates a subset of random ID

ith.sampling.id = sample(1:dim(ith.strata)[1], ith.strata.size)

## Create a selection status -- pay attention to the operator: %in%

ith.sample =ith.strata[ith.strata$add.id %in%ith.sampling.id,]

## dim(ith.sample) $ check the sample

strata.sample = rbind(strata.sample, ith.sample) # stack all dataframe!

}

# dim(strata.sample)

strat.sample.final = strata.sample[-1,] # drop the temporary stratum ID

# kable(head(strat.sample.final[1:5,5])) # accuracy check! only show the first 5 variables

```

# Performance Analysis of Random Samples

In this section, we perform a comparative analysis of the three random samples. One metric we can use is the default rate in each industry defined by the first two digits of NAICS classification code. That was also used as the stratification variable in the stratified sampling plan.

## Population-level Default Rates

We have calculated the default rate across the industries in the previous section. That table includes the category with no NAICS classification code. We will use this population-level industry-specific rate as a reference and compare it with the sample-level industry-specific default rates.

```{r caption="Population-level default rates"}

x.table = table(loan$strNAICS, loan$MIS\_Status)

no.lab = x.table[,1] # first column consists of unknown default label

default = x.table[,2]

no.default = x.table[,3]

default.rate = round(100\*default/(default+no.default),1)

default.status.rate = cbind(no.lab = no.lab,

default = default,

no.default = no.default,

default.rate=default.rate)

kable(default.status.rate, caption = "Population size, default counts,

and population default rates")

```

## Industry-Specific Default Rates based on SRS

For comparison, we construct the following table that includes the industry-specific default rates.

```{r}

# names(SRS.sample)

x.table = table(SRS.sample$strNAICS, SRS.sample$MIS\_Status)

no.lab.srs = x.table[,1] # first column consists of unknown default label

default.srs = x.table[,2]

no.default.srs = x.table[,3]

default.rate.srs = round(100\*default.srs/(default.srs+no.default.srs),1)

##

industry.code = names(default.rate.srs) # extract NSICS code

industry.name=c("Agri-forest-fish-hunt","Construction",

"Manufacturing", "Wholesale-trade", "Retail-trade",

"Transport-warehousing","Information", "Finance-insurance",

"Real-estate-rental","Prof-sci-tech-ser",

"Admin-support-waste-mgnt-remed", "Edu-serv",

"Healthcare-social-assist","Arts-entertain-rec",

"Accommodation-food-ser", "Other-ser(no-public-admin)") # actual industry names!

default.rate.pop = default.rate[industry.code]

# cbind(industry.code,industry.name)

SRS.pop.rates = cbind(default.rate.pop,default.rate.srs)

rownames(SRS.pop.rates) = industry.name

kable(SRS.pop.rates, caption="Comparison of industry-specific default rates

between population and the SRS.")

```

Some of the industry-specific default rates seem to be significantly different. More visual comparisons will be given in the next section.

## Industry-specific Rates- Systematics Sample

We will use the sample stratification variable to find the industry-specific rates based on the systematic sample. The following table will include rates of population, SRS, and systematic random samples.

```{r}

x.table = table(sys.sample$strNAICS, sys.sample$MIS\_Status)

no.lab.sys = x.table[,1] # The first column consists of an unknown default label

default.sys = x.table[,2]

no.default.sys = x.table[,3]

default.rate.sys = round(100\*default.sys/(default.sys+no.default.sys),1)

sys.SRS.pop.rates = cbind(default.rate.pop, default.rate.srs, default.rate.sys)

rownames(SRS.pop.rates) = industry.name

kable(sys.SRS.pop.rates, caption="Comparison of industry-specific default rates

between population, SRS, and Systematic Sample.")

```

It seems that the systematic sample performs better than the SRS sample.

## Industry-specific Default Rates- Stratified Sample

In this section, we put all the information in the following table.

```{r}

#strat.sample.final

x.table = table(strat.sample.final$strNAICS, strat.sample.final$MIS\_Status)

no.lab.str = x.table[,1] # The first column consists of an unknown default label

default.str = x.table[,2]

no.default.str = x.table[,3]

default.rate.str = round(100\*default.str/(default.str+no.default.str),1)

str.SRS.pop.rates = cbind(default.rate.pop, default.rate.srs, default.rate.sys, default.rate.str)

rownames(str.SRS.pop.rates) = industry.name

kable(str.SRS.pop.rates, caption="Comparison of industry-specific default rates

between population, SRS, Systematic Sample,

and Stratified Samples.")

```

# Visualization - Visual Comparison

In the previous section, we calculated the industry-specific default rates for population, SRS, systematic, and stratified samples. We now create a statistical graphic to compare the default rates among the samples.

```{r caption="Graphical comparison of industry-specific default rates among random samples and population. Caution: This chart is based on one-time samples. ", fig.width=8, fig.height=4, }

n=length(default.rate.pop)

#par(bg = 'gray')

# empty plot

plot(NULL, xlim=c(0,n), ylim=c(0, 50), xlab="", ylab ="", axes=FALSE)

title("Comparison of Industry-specific Default Rates")

points(1:n, as.vector(default.rate.pop), pch=16, col=2, cex = 0.8)

lines(1:n, as.vector(default.rate.pop), lty=1, col=2, cex = 0.8)

#

points(1:n, as.vector(default.rate.srs), pch=17, col=3, cex = 0.8)

lines(1:n, as.vector(default.rate.srs), lty=2, col=3, cex = 0.8)

#

points(1:n, as.vector(default.rate.sys), pch=19, col=4, cex = 0.8)

lines(1:n, as.vector(default.rate.sys), lty=3, col=4, cex = 0.8)

#

points(1:n, as.vector(default.rate.str), pch=20, col=5, cex = 0.8)

lines(1:n, as.vector(default.rate.str), lty=4, col=5, cex = 0.8)

#

axis(1,at=1:n, label=industry.code, las = 2)

axis(2)

#

rowMax=apply(str.SRS.pop.rates, 1, max) # max default rate in each industry

segments(1:n, rep(0,n), 1:n, rowMax, lty=2, col="lightgray", lwd = 0.5)

legend("topright", c("Pop", "SRS", "Sys", "Str"), lty=1:4, col=2:5, pch=c(16,17,19,20), cex=0.6, bty="n")

```

For ease of interpretation, I make the following table to map the NAICS codes and the corresponding industries.

```{r}

kable(cbind(industry.name, industry.code),

caption="Industry NAICS codes and the corresponding names")

```

Since the above graph is based on one-time samples, the variation is not included. Therefore, we need more information to do a meaningful comparison. For example, we can package the code in this Markdown document and take, say 1000, samples based on each sampling plan. We then can find the mean industry-specific default rates and the corresponding variation.

## Critiques of Visual Representation

Visual representation is a key component in effective storytelling. As an example, we critique the figure of performance comparison of the three sampling plans in the previous section and seek improvements for effective representation.

The following figure is modified based on the comparison line plot given in the previous section.

```{r caption="Graphical comparison of industry-specific default rates among random samples and population. Caution: This chart is based on one-time samples. "}

n=length(default.rate.pop)

#par(bg = 'gray')

# empty plot

plot(NULL, xlim=c(0,n), ylim=c(0, 50), xlab="", ylab ="", axes=FALSE)

# Light gray background

rect(par("usr")[1], par("usr")[3],

par("usr")[2], par("usr")[4],

col = "gray")

# Add white grid

grid(nx = NULL, ny = NULL,

col = "white", lwd = 1)

title("Comparison of Industry-specific Default Rates")

points(1:n, as.vector(default.rate.pop), pch=16, col=2, cex = 0.8)

lines(1:n, as.vector(default.rate.pop), lty=1, col=2, cex = 0.8)

#

points(1:n, as.vector(default.rate.srs), pch=17, col=3, cex = 0.8)

lines(1:n, as.vector(default.rate.srs), lty=2, col=3, cex = 0.8)

#

points(1:n, as.vector(default.rate.sys), pch=19, col=4, cex = 0.8)

lines(1:n, as.vector(default.rate.sys), lty=3, col=4, cex = 0.8)

#

points(1:n, as.vector(default.rate.str), pch=20, col=5, cex = 0.8)

lines(1:n, as.vector(default.rate.str), lty=4, col=5, cex = 0.8)

#

axis(1,at=1:n, label=industry.code, las = 2)

axis(2)

#

rowMax=apply(str.SRS.pop.rates, 1, max) # max default rate in each industry

segments(1:n, rep(0,n), 1:n, rowMax, lty=2, col="lightgray", lwd = 0.5)

legend("topright", c("Pop", "SRS", "Sys", "Str"), lty=1:4, col=2:5, pch=c(16,17,19,20), cex=0.6, bty="n")

```

There's a lot of ink here that doesn't convey information relevant to the main point we're trying to make.

\* \*\*Grey background\*\*: not only does it provide absolutely no information. it's also unsightly. After we remove it, we will likely have to darken some of the lines.

\* \*\*Grid lines\*\*: it's very unlikely that our audience cares about the exact values at each data point - it's the pattern that matters. The grid lines compete with the pattern we're trying to show.

\* \*\*Legend\*\*: It takes time to guess the abbreviation. It also confuses readers.

\* \*\*Tick marks\*\*: Why make the reader tilt his or her head to read?

\* \*\*Labels of X- and Y- Axis\*\*: labels are missing.

\* \*\*Title\*\*: The title should be more specific.

\* \*\*Line types\*\*: Use color and line type to differentiate - this will help people who have a color-impaired vision, and also any grey-scale copies of the poster you make (as for handouts).

\* \*\*Color coding\*\*: Avoid using use green and red colors in the same graphic to represent different objects. More than 99% of all color-blind people are suffering from a red-green color vision deficiency.

It's easy to make a graph that looks cleaner and has a higher ratio of information-to-total ink:

```{r fig.width=10, fig.height=4, caption="Improved graphics"}

n=length(default.rate.pop)

plot(NULL, xlim=c(0,n), ylim=c(0, 50),

xlab="Industry Classification Code",

ylab ="Default Rates (Percentage)", axes=FALSE) # empty plot

# Light gray background

rect(par("usr")[1], par("usr")[3],

par("usr")[2], par("usr")[4],

col = "gray")

# Add white grid

grid(nx = NULL, ny = NULL,

col = "white", lwd = 1)

title("Comparison of Industry-specific Default Rates Based on Random Samples")

points(1:n, as.vector(default.rate.pop), pch=16, col="darkmagenta", cex = 0.8)

lines(1:n, as.vector(default.rate.pop), lty=1, col="darkmagenta", cex = 0.8)

#

points(1:n, as.vector(default.rate.srs), pch=17, col="chartreuse4", cex = 0.8)

lines(1:n, as.vector(default.rate.srs), lty=1, col="chartreuse4", cex = 0.8)

#

points(1:n, as.vector(default.rate.sys), pch=19, col="darkblue", cex = 0.8)

lines(1:n, as.vector(default.rate.sys), lty=1, col="darkblue", cex = 0.8)

#

points(1:n, as.vector(default.rate.str), pch=20, col="darkcyan", cex = 0.8)

lines(1:n, as.vector(default.rate.str), lty=1, col="darkcyan", cex = 0.8)

#

axis(1,at=1:n, label=industry.code)

axis(2, las = 2)

#

clr = c("darkmagenta","chartreuse4","darkblue","darkcyan")

rowMax=apply(str.SRS.pop.rates, 1, max) # max default rate in each industry

#segments(1:n, rep(0,n), 1:n, rowMax, lty=2, col="lightgray", lwd = 0.5)

legend(2, 45, c("Population", "Simple Random Sampling", "Systematic Sampling", "Stratified Sampling"), lty=rep(1,4), col=clr, pch=c(16,17,19,20), cex=0.6, bty="n")

```

```{r fig.width=10, fig.height=4, caption="Improved graphics"}

n=length(default.rate.pop)

plot(NULL, xlim=c(0,n), ylim=c(0, 50),

xlab="Industry Classification Code",

ylab ="Default Rates (Percentage)", axes=FALSE) # empty plot

title("Comparison of Industry-specific Default Rates Based on Random Samples")

points(1:n, as.vector(default.rate.pop), pch=16, col="darkmagenta", cex = 0.8)

lines(1:n, as.vector(default.rate.pop), lty=1, col="darkmagenta", cex = 0.8)

#

points(1:n, as.vector(default.rate.srs), pch=17, col="chartreuse4", cex = 0.8)

lines(1:n, as.vector(default.rate.srs), lty=1, col="chartreuse4", cex = 0.8)

#

points(1:n, as.vector(default.rate.sys), pch=19, col="darkblue", cex = 0.8)

lines(1:n, as.vector(default.rate.sys), lty=1, col="darkblue", cex = 0.8)

#

points(1:n, as.vector(default.rate.str), pch=20, col="darkcyan", cex = 0.8)

lines(1:n, as.vector(default.rate.str), lty=1, col="darkcyan", cex = 0.8)

#

axis(1,at=1:n, label=industry.code)

axis(2, las = 2)

#

clr = c("darkmagenta","chartreuse4","darkblue","darkcyan")

rowMax=apply(str.SRS.pop.rates, 1, max) # max default rate in each industry

#segments(1:n, rep(0,n), 1:n, rowMax, lty=2, col="lightgray", lwd = 0.5)

legend(2, 45, c("Population", "Simple Random Sampling", "Systematic Sampling", "Stratified Sampling"), lty=rep(1,4), col=clr, pch=c(16,17,19,20), cex=0.6, bty="n")

```

```{r fig.width=10, fig.height=4, caption="Improved graphics"}

n=length(default.rate.pop)

plot(NULL, xlim=c(0,n), ylim=c(0, 50),

xlab="Industry Classification Code",

ylab ="Default Rates (Percentage)", axes=FALSE) # empty plot

# Light gray background

rect(par("usr")[1], par("usr")[3],

par("usr")[2], par("usr")[4],

col = "white")

# Add white grid

grid(nx = NULL, ny = NULL,

col = "white", lwd = 1)

title("Comparison of Industry-specific Default Rates Based on Random Samples")

points(1:n, as.vector(default.rate.pop), pch=16, col="darkmagenta", cex = 0.8)

lines(1:n, as.vector(default.rate.pop), lty=1, col="darkmagenta", cex = 0.8)

#

points(1:n, as.vector(default.rate.srs), pch=17, col="chartreuse4", cex = 0.8)

lines(1:n, as.vector(default.rate.srs), lty=1, col="chartreuse4", cex = 0.8)

#

points(1:n, as.vector(default.rate.sys), pch=19, col="darkblue", cex = 0.8)

lines(1:n, as.vector(default.rate.sys), lty=1, col="darkblue", cex = 0.8)

#

points(1:n, as.vector(default.rate.str), pch=20, col="darkcyan", cex = 0.8)

lines(1:n, as.vector(default.rate.str), lty=1, col="darkcyan", cex = 0.8)

#

axis(1,at=1:n, label=industry.code)

axis(2, las = 2)

#

clr = c("darkmagenta","chartreuse4","darkblue","darkcyan")

rowMax=apply(str.SRS.pop.rates, 1, max) # max default rate in each industry

#segments(1:n, rep(0,n), 1:n, rowMax, lty=2, col="lightgray", lwd = 0.5)

legend(2, 45, c("Population", "Simple Random Sampling", "Systematic Sampling", "Stratified Sampling"), lty=rep(1,4), col=clr, pch=c(16,17,19,20), cex=0.6, bty="n")

```