```
set.seed(123456)
knitr::opts_chunk$set(fig.pos = 'H')
```

Interim Compliance Tests and the Management Letter: Discovery Sampling

Discovery sampling for interim tests: *Discovery sampling* for interim tests sets an estimation sample size for *transaction-unit* samples, so that we are likely to discover at least one error in the sample if the actual transaction error rate exceeds the *minimum acceptable error-rate* (alternatively called the *out-of-control* rate of error). Discovery tests helps the auditor decide whether the systems processing a particular transaction stream are in or out of control.

```
confidence <- seq(.99,.7,-.01)
n <- (log(1-confidence))/log(1-.05)
plot(confidence,n, type="l")
```

So for a 5% intolerable error rate at 95% confidence we have:

```
confidence <- .95
n <- (log(1-confidence))/log(1-.05)
cat("\n Discovery sample size = ", ceiling(n))
```

Attribute Sampling on Occurrences

Attribute sampling for interim tests: sets estimation sample size for *transaction-unit* samples to estimate the error rate in the entire transaction population with some confidence (e.g., 95%) that the estimate is within the *out-of-control* error-rate cutoff for that transaction stream. If it is found that a particular transaction stream is out of control, then attribute estimation will help us decide on the actual error rate of the systems that process this transaction stream. Errors estimates from attribute samples are *rates* in this example.

```
library(pwr)
Client_sales_journal <- read.csv("~/Desktop/523
Midterm/files/sales_journal.csv")

size <- as.numeric(nrow(Client_sales_journal))  # data set size as count of
```

```
transactions

variability <- size/3                       # when you don't know the
dispersion, guess .333 of size

Delta <- .05*size                   # detect 5% occurrence error -- 5% is
the 'out-of-control' cut-off
sigma <- variability
effect <- Delta/sigma



sample <- pwr.t.test(
  d=effect,
  sig.level = 0.05,
  power = .8,
  type="one.sample",
  alternative="greater")                        ## look for overstatement of
earnings

cat("\n Attribute sample size for occurence of error = ",
ceiling(sample$n))
```

Attribute Sampling on Amounts

Attribute sampling for interim tests: sets estimation sample size for *transaction-unit* samples to estimate the error rate in the entire transaction population with some confidence (e.g., 95%) that the estimate is within the *out-of-control* error-rate cutoff for that transaction stream. If it is found that a particular transaction stream is out of control, then attribute estimation will help us decide on the actual error rate of the systems that process this transaction stream. Errors estimates from attribute samples are *amounts* in this example

```
library(pwr)
Client_sales_journal <- read.csv("~/Desktop/523
Midterm/files/sales_journal.csv")

size <- as.numeric(nrow(Client_sales_journal))  # data set size as count of
transactions

mu <- mean(Client_sales_journal$sales_extended)   # average value of
transaction

Delta <- .05*mu                                  # detect 5% amount
intolerable error

sigma <- sd(Client_sales_journal$sales_extended)    # variability

effect <- Delta/sigma
```

```
sample <- pwr.t.test(
  d=effect,
  sig.level = 0.05,
  power = .8,
  type="one.sample",
  alternative="greater")   ## look for sales value too large

cat("\n Attribute sample size for amount of error = ", ceiling(sample$n))
```

Acceptance Sampling

Acceptance sampling for substantive tests: Discovery sample analysis results in a decision of whether we can accept that the account is fairly presented in the financial statements. Errors estimates from acceptance are monetary balances.

```
library(pwr)


Client_sales_journal <- read.csv("~/Desktop/523
Midterm/files/sales_journal.csv")

size <- as.numeric(nrow(Client_sales_journal))  # data set size as count of
transactions

mu <- mean(Client_sales_journal$sales_extended)   # average value of
transaction

Delta <- .1*mu                                  # detect 10% (material)
error in sample

sigma <- sd(Client_sales_journal$sales_extended)    # variability

effect <- Delta/sigma




sample <- pwr.t.test(
  d=effect,
  sig.level = 0.1,
  power = .8,
  type="one.sample",
  alternative="greater")    # look for sales value too large -- principle
of 'Conservatism'

cat("\n Acceptance sample size = ", ceiling(sample$n))
```

The Application of Sampling in Interim Tests

Auditors use a sequence of: (1) discovery sampling to discover an *out-of-control* system; and if needed (2) attribute sampling to estimate the error rate in a system. Discovery sampling sets an estimation sample size for *transaction-unit* samples, so that we are likely to discover at least one error in the sample if the actual transaction error rate exceeds the *minimum acceptable error-rate* (alternatively called the out-of-control rate of error). Discovery tests helps the auditor decide whether the systems processing a particular transaction stream are in or out of control.

To limit the cost of an audit, discovery samples are selected first to determine whether the client's accounting system is in or out of control. If the system is determined to be out of control, additional evidence is obtained until the auditor has a sample size sufficient for attribute sampling. Attribute sampling sets estimation sample size for *transaction-unit* samples to estimate the error rate in the entire transaction population with some confidence (e.g., 95%) that the estimate is within the out-of-control error-rate cutoff for that transaction stream. If it is found that a particular transaction stream is out of control, then attribute estimation will help us decide on the actual error rate of the systems that process this transaction stream.

The profession has never been clear on significance and power (type I and II error levels) of statistical decisions concerning whether a system is in or out of control. In the 1970s, Statement on Auditing Procedures 54 suggested that significance should probably be less than 0.3 (i.e., confidence limits should be set greater than 0.7) but otherwise has left the choice of type I and II error levels has been left to the auditor's judgment. Fisher suggested the .95 confidence level that we commonly used in scientific studies, and generally levels above .9 are safe.

If expected error rate is s, intolerable error rate is r and the error rate in the control system is ϵ then restate the hypotheses as:

$H_0: \epsilon=s$ where $\epsilon \in [0,1]$

$H_a: \epsilon \ge r$ where $r, \epsilon \in [0,1]$

This is depicted in the figure 2.

```
x <- seq(.001, .2, length=100)
hx <- dnorm(x,mean=.05, sd=.02)
plot(x, hx, type="l", lwd=2,  col="red", xlim = c(0,.2), xlab="Error rate
in the accounting system", ylim = c(0,22),
  ylab="Density")
curve(dnorm(x, x,mean=.1, sd=.02),
      lwd=2, col="darkblue", add=TRUE, yaxt="n")
abline(v=.08, col="gray80", lwd=3, lty=2)
text(.11,10, "Power", col = "gray60", adj = c(.8,.1))
text(.115,-.5, "Significance", col = "gray60", adj = c(.8,.1))
text(.08,21, "Tolerable error rate", col = "gray60", adj = c(.4, -.1))
```

```
text(.05,5, "In-Control", col = "gray20", adj = c(.8,.1))
text(.1,5, "Out-of-Control", col = "gray20", adj = c(.4, -.1))
```

Discovery sampling chooses a sample to determine whether an error rate does not exceed a designated percentage. If the sample does not contain errors, then the actual error rate is assumed to be lower than the minimum unacceptable rate. The sampling calculation requires the auditor to specify the confidence level of the test and the minimum unacceptable error rate using an "urn model" where models are typical stated as draws of colored balls from an urn. Consider, for illustration, the audit of two sets of transactions: sales and subsequent collections for sales on credit. The decision is couched in terms of two assumptions of error distributions in a transaction stream: (1) errors are 'in-control' and (2) errors are 'out-of-control'.

Assume that the auditor determines that the minimum acceptable error rates are as follows:

- the sales accounting system is 'in-control' when less than 0.05 of transactions are in error, and
- the AR collections system is 'in-control' when less than 0.1 of transactions are in error

For confidence level c we want to choose $n = \frac{log(1-c)}{log(1-p)}$

```
library(tidyverse)

c <- seq(.9,.99, length.out = 100)
p <- .05
n <- log(1-c)/log(1-p)

samp <- data.frame(c,n)
ggplot(samp,aes(c,n))+
    geom_line()+
    labs(title="\n \n Discovery sample size for sales system") +
    xlab("confidence limit (1 - significance)") +
    ylab("sample size")

p <- .1
n <- log(1-c)/log(1-p)
samp <- data.frame(c,n)
ggplot(samp,aes(c,n))+
    geom_line()+
    labs(title="\n Discovery sample size for collections system") +
    xlab("confidence limit (1 - significance)") +
    ylab("sample size")
```

From the graphs, the sales system needs a sample size of 58, and the collections system needs 28 for .95 confidence. Start with the sales system to ascertain whether the system is in or out of control. The selection is from the client's unaudited file and audit evidence is obtained from the real world, which are reflected in "real_world_" prefixed files generated from the code presented in chapter 13. Note that unless you set a random seed, the values will change each time they are generated.

```
library(tidyverse)
library(compareDF)
library(arsenal)
library(compare)

sample_size = 58  # set the sample size based on the discovery sample
algorithm

## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"

if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}

real_world_sales <-
  read.csv("real_world_credit_sales.csv",
           na.strings="0", stringsAsFactors=FALSE) %>%
  select(invoice_no,
         sales_unit_price,
         sales_count)

sales_journal <-
  read_csv("sales_journal.csv",
   col_types = cols(X1 = col_skip()))

sales_temp <- split(sales_journal,sales_journal$cash_not_ar)
sales_journal <- sales_temp$`0`   ## select only credit sales

discovery_sample_sales <-
  sales_journal[runif(sample_size,1,nrow(sales_journal)),] %>%
  select(invoice_no,
         sales_unit_price,
         sales_count)

audited_sample <-
  left_join(discovery_sample_sales,
            real_world_sales,
            by="invoice_no") %>%
  select(invoice_no,
         sales_unit_price.x,
         sales_count.x,
         sales_unit_price.y,
         sales_count.y)

exceptions <-
   audited_sample %>%
   filter(sales_unit_price.x != sales_unit_price.y |
          sales_count.x != sales_count.y
```

```
          )

error_rate_sales <- nrow(exceptions) / nrow(audited_sample)


cat("\n \n \n Discovery: error rate in sales:", error_rate_sales)


error_sales_amt <-
      100 *  sum(exceptions$sales_count.y * exceptions$sales_count.y ) /
      sum(audited_sample$sales_unit_price.x * audited_sample$sales_count.x)


cat("\n \n Discovery: amount for sales sample is overstated (i.e., is in
error):",
    prettyNum(error_sales_amt, big.mark=","),
    " %")
```

If the error rate is above zero (i.e., if the auditor has found more than one exception) additional evidence needs to be collected for an attribute test, to ascertain the true rate of errors. First the auditor needs to consider prior years audit outcomes (if they exist) in determining whether to proceed to attribute samples. This uses the 'rule of threes' to calculate 95% upper confidence bounds. Assume that a review of the audit files shows that the sales transactions have been assessed to be 'in-control' over the past 20 audits; then a $95%$ upper bound on the rate of occurrence is $\frac{3}{n}$. By the law of rare errors, the observed events Y (i.e., no errors in the audit) follow a $Poisson(\lambda)$ distribution using n samples. The sum of Poisson random variables is Poisson, so the question of at least one Y not equal to zero is the probability that the sum $\sum{Y_i}$ is greater than zero. Let this probability be, for example, 0.95 so that $P[\sum{Y_i} = 0] = e^{-n\lambda}=0.05$. Taking logarithms gives $n\lambda =-ln(0.05) \approx 3$. Thus the rate $\lambda \approx \frac{3}{n}$.

Attribute Sampling with t-Tests

For sake of discussion, we will assume that we have determined an 'out-of-control' sales system situation in discovery sampling. Attribute sampling size is determined using Cohen's power analysis which is implemented in R's `pwr` package, If discovery sampling suggests that a particular transaction stream is out of control, then attribute estimation will help us decide on the actual error rate of the systems that process this transaction stream. Since the auditor will have already collected evidence for the discovery sample, the cost of attribute sampling is only that of auditing the increment in sample size recommended by Cohen's `pwr` formulas. Transaction records already selected for discovery sampling should be combined with additional records drawn for attribute sampling. Error estimates from attribute samples may either be *rates of erroneous transactions* or from a monetary unit sampling perspective, can be *rates of monetary error in the transaction stream*. I compute both in the following example. As in the prior example, the post audit information is drawn from the `real_world_credit_sales` file, which is merely a device in this example; such information would not truly be available to the auditor, rather it is the information gained by analyzing

audit evidence. Thus I use an `inner_join` of the `real_world)credit_sales` and `sales_sample` files to mimic the post-audit results for the sample of sales transactions.

```
library(readr)
library(pwr)
library(tidyverse)


## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"

if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}


real_world_sales <-
  read.csv("real_world_credit_sales.csv", na.strings="0",
stringsAsFactors=FALSE)


sales_journal <-
  read_csv("sales_journal.csv",
   col_types = cols(X1 = col_skip()))

sales_temp <- split(sales_journal,sales_journal$cash_not_ar)
sales_journal <- sales_temp$`0`


size <- as.numeric(nrow(sales_journal))  ## data set size
Delta <- .05*size                                  ## detect 5% occurrence
error
sigma <- .3*size                                   ## variability (guess
~1/3 rd)
effect <- Delta/sigma

sample <- pwr.t.test(
  d=effect,
  sig.level = 0.05,
  power = .8,
  type="one.sample",
  alternative="greater")                      ## look for overstatement of
earnings


cat("\n \n Attribute sample size for occurrence of error = ",
ceiling(sample$n))
```

```
size <-
as.numeric(sum(sales_journal$sales_unit_price*sales_journal$sales_count))
## data set size
mu <- mean(sales_journal$sales_unit_price*sales_journal$sales_count)   ##
average value of transaction
Delta <- .05*mu                    ## detect 5% amount intolerable error
sigma <- sd(sales_journal$sales_unit_price*sales_journal$sales_count)
## variability
effect <- Delta/sigma


sample <- pwr.t.test(
  d=effect,
  sig.level = 0.05,
  power = .8,
  type="one.sample",
  alternative="greater")   ## look for sales value too large

cat("\n\n Attribute sample size for amount of error = ", ceiling(sample$n))
```

Since the discovery sample size was 58 that auditor needs to obtain an additional $224- 58 = 166 $ records for attribute tests of occurrence rate, and $636 - 58 = 578$ records for attribute tests of amount rate (what monetary unit sampling field would call 'taintings' rate).

```


discovery_sample_sales_add <-
  sales_journal[runif(166,1,nrow(sales_journal)),] %>%
  select(invoice_no,
         sales_unit_price,
         sales_count)

discovery_sample_sales_occ <- rbind(
  data.frame(discovery_sample_sales_add),
  data.frame(discovery_sample_sales))

audited_sample <-
  left_join(discovery_sample_sales_occ,
            real_world_sales,
            by="invoice_no") %>%
  select(invoice_no,
         sales_unit_price.x,
         sales_count.x,
         sales_unit_price.y,
         sales_count.y)

exceptions <-
   audited_sample %>%
   filter(sales_unit_price.x != sales_unit_price.y |
          sales_count.x != sales_count.y
```

```
          )

error_rate_sales <- nrow(exceptions) / nrow(audited_sample)

cat("\n \n \n Attribute: error rate in sales: ", error_rate_sales)



discovery_sample_sales_add <-
  sales_journal[runif(578,1,nrow(sales_journal)),] %>%
  select(invoice_no,
         sales_unit_price,
         sales_count)

discovery_sample_sales_amt <- rbind(
  data.frame(discovery_sample_sales_add),
  data.frame(discovery_sample_sales))

audited_sample <-
  left_join(discovery_sample_sales_amt
            ,real_world_sales,
            by="invoice_no") %>%
  select(invoice_no,
         sales_unit_price.x,
         sales_count.x,
         sales_unit_price.y,
         sales_count.y)

exceptions <-
   audited_sample %>%
   filter(sales_unit_price.x != sales_unit_price.y |
          sales_count.x != sales_count.y
          )

error_sales_amt <-
      100 *  sum(exceptions$sales_count.y * exceptions$sales_count.y ) /
      sum(audited_sample$sales_unit_price.x * audited_sample$sales_count.x)


cat("\n \n \n Attribute: amount sales sample is overstated (i.e., is in
error)  ",
    prettyNum(error_sales_amt, big.mark=","),
    " %")
```

The 'out-of-control' critical value is 0.05 error rate, thus both in occurrence and in amount, the auditor determines the sales system to be 'in-control'. Note that we added samples to the original discovery set in order to keep down audit costs, as investigating evidence is the major cost of an audit. The final error rate is based on a count of any differences in values in the clients records versus the real world. If this exceeds the critical 'intolerable' error rate, then the system is judged 'out-of-control' I repeat the same tests for the accounts receivable collection transactions below.

Audit of Collection Transactions

For sake of discussion, we will assume that we have determined an 'out-of-control' collections system and discovery sample size is determined using the 'urn model' formula, and attribute sampling size is determined using Cohen's power analysis implemented in R's *pwr* package, If discovery sampling suggests that the collections transaction stream is out of control, then attribute estimation will help us decide on the actual error rate of the collection systems that process this transaction stream. Errors estimates from attribute samples may either be *rates of erroneous transactions* or from a monetary unit sampling perspective, can be *rates of monetary error in the transaction stream*.

```
library(tidyverse)

## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"

if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}


real_world_collections <-
  read.csv("real_world_collections.csv", na.strings="0",
stringsAsFactors=FALSE)

ar_collections_journal <-
  read.csv("collections_journal.csv", na.strings="0",
stringsAsFactors=FALSE)

c <- .95
p <- .05
sample_size <- ceiling(log(1-c)/log(1-p))
cat('\n \n \n Collections discovery sample size is ', n)


discovery_sample_collections <-
  ar_collections_journal[
    ceiling(
      runif(
        sample_size,1,nrow(ar_collections_journal))),]

audited_sample <-
  left_join(discovery_sample_collections,
            real_world_sales,
            by="invoice_no") %>%
  select(invoice_no,
         collection_amount.x,
         collection_amount.y)
```

```
audited_sample[is.na(audited_sample)] <- 0

exceptions <-
   audited_sample %>%
   filter(collection_amount.x != collection_amount.y)

error_rate_collections <- nrow(exceptions) / nrow(audited_sample)

cat("\n \n \n Discovery: error rate in collections: ",
error_rate_collections)


error_collections_amt <-
      sum(exceptions$collection_amount.x - exceptions$collection_amount.y)


cat("\n \n \n Discovery: amount collections sample is overstated (i.e., is
in error)  $",
    prettyNum(error_collections_amt, big.mark=","))
```

This error rate is above zero (i.e., the auditor has found more than one exception) so we need to attribute sample. Next set attribute sampling size using Cohen's power analysis which is implemented in R's *pwr* package. Since discovery sampling suggests that the collections transaction processing is out of control, then attribute estimation will help us decide on the actual error rate of the systems that process this transaction stream. Errors estimates from attribute samples may either be *rates of erroneous transactions* or from a monetary unit sampling perspective, can be *rates of monetary error in the transaction stream*. We compute both in the following example.

```
library(readr)
library(pwr)
library(tidyverse)


## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"

if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}


real_world_collections <-
  read.csv("real_world_collections.csv", na.strings="0",
```

```
stringsAsFactors=FALSE)

ar_collections_journal <-
  read.csv("collections_journal.csv", na.strings="0",
stringsAsFactors=FALSE)


size <- as.numeric(nrow(ar_collections_journal))  ## data set size
Delta <- .05*size                                 ## detect 5% occurrence
error
sigma <- .3*size                                  ## variability (guess
~1/3 rd)
effect <- Delta/sigma



sample <- pwr.t.test(
  d=effect,
  sig.level = 0.05,
  power = .8,
  type="one.sample",
  alternative="greater")                      ## look for overstatement of
earnings


cat("\n \n Attribute sample size for occurrence of error = ",
ceiling(sample$n))



size <- as.numeric(sum(
  ar_collections_journal$sales_extended))   ## data set size


mu <- mean(ar_collections_journal$sales_extended)   ## average value of
transaction
Delta <- .05*mu                                     ## detect 5% amount
intolerable error
sigma <- sd(ar_collections_journal$sales_extended)/3       ## variability
effect <- Delta/sigma


sample <- pwr.t.test(
  d=effect,
  sig.level = 0.05,
  power = .8,
  type="one.sample",
  alternative="greater")   ## look for sales value too large

cat("\n Attribute sample size for amount of error = ", ceiling(sample$n))
```

I mentioned earlier that accounting systems are highly multicolinear. This example illustrates this, as means and standard deviations (as well as the resulting samples) are very close to those of the sales transaction stream. Collections are just sales transactions that are delayed (assuming that customers eventually pay their bills). Since the discovery sample size was 59 that auditor needs to obtain an additional $224- 59 = 165 $ records for attribute tests of occurrence rate, and $122 - 59 = 63 $ for records for attribute tests of amount rate (what the monetary unit sampling field would call 'taintings' rate).

```
library(tidyverse)


discovery_sample_collections_add <-
  ar_collections_journal[
    runif(165,1,nrow(ar_collections_journal)),]

discovery_sample_collections_occ <- rbind(
  data.frame(discovery_sample_collections_add),
  data.frame(discovery_sample_collections))

audited_sample <-
  left_join(discovery_sample_collections_occ,
            real_world_sales,
            by="invoice_no") %>%
  select(invoice_no,
         collection_amount.x,
         collection_amount.y)

audited_sample[is.na(audited_sample)] <- 0

exceptions <-
   audited_sample %>%
   filter(collection_amount.x != collection_amount.y)

error_rate_collections <- nrow(exceptions) / nrow(audited_sample)

cat("\n \n \n Attribute: error rate in collections: ",
error_rate_collections)


discovery_sample_sales_add <-
  sales_journal[runif(63,1,nrow(sales_journal)),]

discovery_sample_collections_amt <- rbind(
  data.frame(discovery_sample_collections_add),
  data.frame(discovery_sample_collections))

audited_sample <-
  left_join(discovery_sample_collections,
            real_world_sales,
            by="invoice_no") %>%
  select(invoice_no,
         collection_amount.x,
```

```
          collection_amount.y)

audited_sample[is.na(audited_sample)] <- 0

exceptions <-
   audited_sample %>%
   filter(collection_amount.x != collection_amount.y)


error_collections_amt <-
      sum(exceptions$collection_amount.x - exceptions$collection_amount.y)

cat("\n \n \n Attribute: error amount in collections: $",
    prettyNum(error_collections_amt, big.mark=","))
```

The 'out-of-control' critical value is 0.10 error rate for the collections transaction stream, and this can be compared to the occurrence and amount to determine the collections systems 'out' or 'in-control' status. Note that we added samples to the original discovery set in order to keep down audit costs, as investigating evidence is the major cost of interim testing.

Machine Learning Models for Audit of Controls

Machine learning extends many of the models and concepts pioneered in mathematical statistics over the past two centuries, It's functionality has leapfrogged statistical capabilities in two areas:

1. Data representations are much richer than the $2 \times 2: \ observation \times measurement$ matrix representation that is standard in statistics. Data can be represented as multi-dimensional tensors (arrays in R terminology), and this allows complex datasets such as video and geographic information to be readily processed in machine learning.

2. Solution concepts and loss functions are much richer in machine learning, because computationally intensive methods allow us to compute optima for response surfaces that may not have closed-form representations, and for which differential calculus has no first-order method of finding optima.

TensorFlow requires installation prior to using any of its functions in R. This and other aspects of TensorFlow use are beyond the scope of this text. The reader is directed towards the ample documentation at https://www.tensorflow.org/ and https://tensorflow.rstudio.com/ on installation and use.

```
library(readr)
library(tidyverse)
library(compare)


## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"
```

```
if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}


real_world_cash_sales <-
  read.csv("real_world_cash_sales.csv", na.strings="0",
stringsAsFactors=FALSE)

real_world_credit_sales  <-
  read.csv("real_world_credit_sales.csv", na.strings="0",
stringsAsFactors=FALSE)

sales_journal <-
  read.csv("sales_journal.csv", na.strings="0", stringsAsFactors=FALSE)

sales_journal[is.na(sales_journal)] <- 0

sales_journal <- split(sales_journal, sales_journal$cash_not_ar)

credit_sales_journal <-  sales_journal$`0`
cash_sales_journal <- sales_journal$`1`


real_world_credit_sales$sales_extended <-
real_world_credit_sales$sales_unit_price *
  real_world_credit_sales$sales_count

credit_sales_journal$cost_extended <-
  credit_sales_journal$unit_cost *
  credit_sales_journal$sales_count

p1 <- hist(credit_sales_journal$sales_extended, breaks=20)
p2 <-hist(credit_sales_journal$cost_extended)

plot( p1, col=rgb(0,0,1,1/4), ylim=c(0,250),
      main = "Cost (orange) vs. Revenue (purple) of Sales")
plot( p2, col=rgb(1,0,0,1/4), add=T, ylim=c(0,250))
```

For an autoencoder to work well we need a strong initial assumption: that the distribution of variables for normal transactions is different from the distribution for ones that are in error. A simple graphical example provides some insight into how the autoencoder can reveal control weaknesses at a systemic level. In the following code chunk, I create a ridgeplot of column values for the `left_joined` files for sales on the client's books and actual sales, with a factor field for exceptions. The ridgeplots highlight the errors in the client's sales records for `collection_date`, and `sales_unit_price`. Ridgeplots are not particularly refined, but they can provide a quick method for 'eyeballing' systemic weaknesses.

```
library(tidyverse)
library(ggridges)


## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"

if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}

real_world_sales <-
  read.csv("real_world_credit_sales.csv",
           na.strings="0", stringsAsFactors=FALSE) %>%
  select(invoice_no,
         sales_unit_price,
         customer_no,
         sales_count,
         collection_amount,
         collection_date
         )

real_world_sales$sales_unit_price =
  real_world_sales$sales_unit_price*
  rbinom(nrow(real_world_sales),1,.7)

sales_journal <-
  read_csv("sales_journal.csv",
   col_types = cols(X1 = col_skip()))%>%
  filter(cash_not_ar == 0) %>%
  select(invoice_no,
         sales_unit_price,
         invoice_no,
         customer_no,
         sales_count,
         collection_amount,
         collection_date
         )

sales_except <-
   left_join(sales_journal,
             real_world_sales,
             by="invoice_no")

if(sales_except$sales_unit_price.x != sales_except$sales_unit_price.y) {
  sales_except$exception = 1
} else {
  sales_except$exception = 0
}
```

```
sales_except %>%
   select(exception,
    price_in_real_world = sales_unit_price.y,
    price_on_books = sales_unit_price.x,
    invoice_no,
    customer_no.y,
    sales_count.y,
    collection_amount.y,
    collection_date.y,
    customer_no.x,
    sales_count.x,
    collection_amount.x,
    collection_date.x
          ) %>%
  gather(variable, value, -exception) %>%
  ggplot(aes(y = as.factor(variable),
             fill = as.factor(exception),
             x = percent_rank(value))) +
  geom_density_ridges()+
  theme(legend.position = "none")
```

The plots reveals the simple strategy used to alter `sales_unit_price` values, while other variables had no change in their distribution. Real world situations will generally be more complex, but the basic concepts still apply, and this should be sufficient to demonstrate the autoencoder.

I trained the model using Keras' `fit()` function, and used `callback_model_checkpoint()` in order to save the model after each epoch. By passing the argument `save_best_only = TRUE` I save only the epoch (i.e., pass of the dataset) with smallest loss value on the test set. I also use callback_early_stopping() to stop training if the validation loss stops decreasing for 15 epochs (i.e., passes of the dataset).

```
## load Keras, etc.  You need to have installed Tensorflow for this to work
library(tidyverse)
library(keras)
install_keras()
```

R will restart after the installation of keras. We can then proceed to build our deep-learning model.

```
library(tidyverse)
library(lubridate)
library(stringr)
```

```
library(keras)


use_session_with_seed(123) ## make sure results are reproducible
## The use_session_with_seed() function
## establishes a common random seed for R, Python, NumPy, and TensorFlow.
## It furthermore disables hash randomization, GPU computations, and CPU
parallelization,
## which can be additional sources of non-reproducibility.


## Set the directory for the new files (modify the path as needed)
default_dir <- "/home/westland/audit_analytics_book/audit_simulated_files/"

if (file.exists(default_dir)){
    setwd(default_dir)
} else {
    dir.create(default_dir)
    setwd(default_dir)
}

real_world_sales <-
  read.csv("real_world_credit_sales.csv",
           na.strings="0", stringsAsFactors=FALSE) %>%
  select(invoice_no,
         sales_unit_price,
         customer_no,
         sales_count,
         collection_amount,
         collection_date
         )

real_world_sales$sales_unit_price  <-
  real_world_sales$sales_unit_price*
  rbinom(nrow(real_world_sales),1,.9)

sales <-
  read_csv("sales_journal.csv",
   col_types = cols(X1 = col_skip()))%>%
  filter(cash_not_ar == 0) %>%
  select(invoice_no,
         sales_unit_price,
         invoice_no,
         customer_no,
         sales_count,
         collection_amount,
         collection_date
         )


sales_w_error <-
   left_join(sales,
             real_world_sales,
             by="invoice_no")
```

```
if(sales_except$sales_unit_price.x !=
   sales_except$sales_unit_price.y) {
  sales_w_error$error = TRUE
} else {
  sales_w_error$error = FALSE
}

sales_w_error[is.na(sales_w_error)] <- 0

sales_w_error$collection_date.x <-
  as.numeric(
    decimal_date(as.POSIXct(sales_w_error$collection_date.x)))

sales_w_error$collection_date.y <-
  as.numeric(decimal_date(
    as.POSIXct(sales_w_error$collection_date.y)))


y <- sales_w_error %>%  select (error)

x <- sales_w_error %>%
  select (price = sales_unit_price.x,
          count = sales_count.x,
          amt_coll = collection_amount.x,
          date_coll = collection_date.x,
          real_price = sales_unit_price.y,
          real_count = sales_count.y,
          real_amt_coll = collection_amount.y,
          real_date_coll =collection_date.y,
          error)

c <- x$error

freq <- 100 * nrow(x[x$error!=0,])/nrow(x[x$error==0,])
cat("Error frequency = ", freq, "% ")

bucket <- sample(2, nrow(sales), replace=T, prob = c(0.75, 0.25))  # 3:1
split

y_train <- y[bucket==1,]
y_test <- y[bucket==2,]

# don't use the error for ML
# the auditor doesn't know the errors,
# these are something to be found during the audit)
x_train <- x[bucket==1,1:8]
x_test <- x[bucket==2,1:8]
x_train_w_err <- x[bucket==1,9]  # save error for later
x_test_w_err <- x[bucket==2,9]

col_means_train <-as.numeric(apply(x_train,2,mean))   ## invoice is an ID
col_stddevs_train <- as.numeric(apply(x_train,2,sd))
```

```
# Normalize training data
x_train <-  as.matrix(scale(x_train, center = col_means_train, scale =
col_stddevs_train))


# Test data is *not* used when calculating the mean and std.
x_test <-  as.matrix(scale(x_test, center = col_means_train, scale =
col_stddevs_train))



## The autoencoder model

x_train_model <- as.matrix(x_train)  ## don't use the error to train

model <- keras_model_sequential()
model %>%
  layer_dense(units = 100, activation = "tanh", input_shape =
ncol(x_train)) %>%
  layer_dense(units = 10, activation = "tanh") %>%
  layer_dense(units = 100, activation = "tanh") %>%
  layer_dense(units = ncol(x_train))

summary(model)


# Compile and fit the model to the data
## Use callback_model_checkpoint()
## in order to save the model after each epoch.
## By passing the argument save_best_only = TRUE
## I save only the epoch (i.e., pass of the dataset)
## with smallest loss value on the test set.


model %>% compile(
   loss = "binary_crossentropy",
   optimizer = "adam",
   metrics= 'mse'
)

checkpoint <- callback_model_checkpoint(
  filepath = "model.hdf5",      ## Keras model weights are saved to HDF5
format.
  save_best_only = TRUE,
  period = 10,
  verbose = 1
)

early_stopping <- callback_early_stopping(patience = 10)

history <-
  model %>%
  fit(
  x = x_train[y_train == 0,],
```

```
  y = x_train[y_train == 0,],
  epochs = 50,        ## potentially complex model so train for a long time
  batch_size = 8192,   ## adjust for your memory size
  validation_data = list(x_test[y_test == 0,], x_test[y_test == 0,]),
  callbacks = list(checkpoint, early_stopping)
)

plot(history)

# After training we can get the final loss for the test set by using the
evaluate() function.


loss <- evaluate(model, x = x_test[y_test == 0,], y = x_test[y_test == 0,])
loss
```

Once the model is trained and tuned the autoencoder can be used to generate predictions. We assumed that the the mean-squared-error (MSE) of error transactions will be higher, and this can be used to predict errors in unaudited populations.

A good measure of model performance in highly unbalanced datasets is the Area Under the ROC Curve (AUC). AUC has a straightforward interpretation for the audit task; it is the probability that an error transaction will have higher MSE then a non-error transaction. Calculate this using R's *Metrics* package, which implements a wide variety of common machine learning model performance metrics.

```
pred_train <- predict(model, x_train)
mse_train <- apply((x_train - pred_train)^2, 1, sum)

pred_test <- predict(model, x_test)
mse_test <- apply((x_test - pred_test)^2, 1, sum)


library(Metrics)
train_acc <- auc(y_train, mse_train)
test_acc <- auc(y_test, mse_test)
train_acc
test_acc
```

The values returned by the `Metrics::auc` function give the probability that an error transaction will have higher MSE than a non-error transaction. From the above analysis, the probability that errors have higher MSE is quite high, and the autoencoder is able to generate very informative predictors for the sales transaction data. Making predictions involves finding a threshold k such that if $MSE > k$ the auditor decides that the transaction is in error. To define this value it is useful to look at the information science measures of precision and recall while varying the threshold k. The following code creates two plots that

do just that, using an abbreviated form of R's `ggplot` called `qplot`, mapping precision and recall against the decision threshold for deciding whether or not to audit a transaction.

```
possible_k <- seq(0, 0.5, length.out = 100)      # range of threshold
values k
precision <- sapply(possible_k, function(k) {    # create a vector of
precision x k
  predicted_class <- as.numeric(mse_test > k)
  sum(predicted_class == 1 & y_test == 1)/sum(predicted_class)
})

qplot(possible_k, precision, geom = "line") + labs(x = "Decision threshold
k", y = "Precision")


recall <- sapply(possible_k, function(k) {     # create a vector of recall
x k
  predicted_class <- as.numeric(mse_test > k)
  sum(predicted_class == 1 & y_test == 1)/sum(y_test)
})
qplot(possible_k, recall, geom = "line")  + labs(x = "Decision threshold
k", y = "Recall")
```

These two graphs provide insight into the trade-offs we make in threshold choice, but they are not directly interpretable into specific choices in the audit. For that, it is necessary to map the trade-off between:

1. audit cost, which will depend on sample size n, and
2. audit risk, i.e., the probability of accepting an account value α (in this case sales) containing an intolerable error. This in turn depends on the auditor's choice of intolerable error level M

The auditor starts with the assumption that the transaction system is 'out-of-control' $X%$ of the years, based on prior years' workpapers. In the current year the system is by default assumed to be 'in-control'. Use a one-sample proportion test to see if the probability of an 'out-of-control' transaction system is significantly different from zero. Significance is measured by p-value; if this falls below the significance threshold, say 0.05, we can conclude that the transaction processing is 'out-of-control'. This can be determined using R's `pwr.p.test` function in the `pwr` package.

Note that the thresholds for the following tests are for MSE of the model predictions, and will be different from those of precision and recall (which are simply ratios of counts). The prediction process identifies errors train_acc = ~87% of time thus sampling from these predictions can be thought of as a 'distilled' sample of errors that are $\frac{1}{1-train_acc}$ as concentrated as in a sample from all of the population. The estimate of population mean from the prediction sample multiplies the sample estimate of error by $\frac {(population-size) * (1-train_acc)}{sample-size}$ for a 95% confidence (5% significance) this will be $z_{.95} \approx 1.96$

```
library(tidyverse)


# range of sample sizes
r <- seq(10,100,10)
nr <- length(r)

# thresholds
k <-  seq(.5,20,.5)
nk <- length(k)

pred_mean <- array(numeric(nr*nk), dim=c(nr,nk))
full_mean <- array(numeric(nr*nk), dim=c(nr,nk))
pred_sd <- array(numeric(nr*nk), dim=c(nr,nk))
full_sd <- array(numeric(nr*nk), dim=c(nr,nk))

pop_error_pred <- array(numeric(nr*nk), dim=c(nr,nk))
pop_error_full <- array(numeric(nr*nk), dim=c(nr,nk))




## recover the error in trainiing and test data to do sampling
x_train <- cbind(x_train,x_train_w_err)
x_test <- cbind(x_test,x_test_w_err)
x_test <- cbind(x_test,mse_test)



for(l in 1:nk) {

  predicted <- x_test[mse_test > k[l],]

  for (j in 1:nr){


pred_sample <- dplyr::sample_n(as_tibble(predicted), r[j], replace=T)

## sample_n is part of tidyverse
## a tibble is tidyverse's version of a data.frame
## (a cute variation on 'table')


pred_mean[j,l] <- mean(pred_sample$x_test_w_err)
pred_sd[j,l] <- sd(pred_sample$x_test_w_err)

## for a 95% confidence (5% significance) this will be $z_{.95} \approx
1.96$

pop_error_pred[j,l] <-
  (pred_mean[j,l] + pred_sd[j,l] * 1.96) *   ## one-sided 95% CL
  nrow(sales)*(1-train_acc)/r[j]
```

```
## sample from all rows and compute sample mean-sd

full_sample <- dplyr::sample_n(as_tibble(x_test), r[j])
full_mean[j,l] <- mean(full_sample$x_test_w_err)
full_sd[j,l] <- sd(full_sample$x_test_w_err)
pop_error_full[j,l] <-
  (pred_mean[j,l] + pred_sd[j,l] * 1.96) *
  nrow(sales) / r[j]

  }
}

## save all of our computed data for the sales transaction audit;
# reload with load("audit_data.RData")
save(
  pop_error_full,
  pop_error_pred,
  pred_mean,
  pred_sd,
  full_mean,
  full_sd,
  file="audit_data.RData")
```

```
library(dplyr)
library(plotly)
library(reshape)
library(ggplot2)
library(scales)   # needed for formatting y-axis labels to non-scientific
type

# webshot::install_phantomjs()  # phantomjs is required to run plotly,
install webshot first

r <- seq(10,100,10)
nr <- length(r)

k <- seq(.5,20,.5)
nk <- length(k)


pred_scatter <- melt(pop_error_pred)
colnames(pred_scatter)=c("sample_size", "threshold", "error_est")
full_scatter <- melt(pop_error_full)
colnames(full_scatter)=c("sample_size", "threshold", "error_est")
```

```
full_scatter$sample_size <- full_scatter$sample_size * 20
full_scatter$threshold <- full_scatter$threshold * .5

pred_scatter$sample_size <- pred_scatter$sample_size * 20
pred_scatter$threshold <- pred_scatter$threshold * .5

pred_scatter$pred_or_full <- "predicted"   ## indicator for plotting
full_scatter$pred_or_full <- "full"

full_pred_scatter <- rbind(full_scatter, pred_scatter)
grouped <- full_pred_scatter %>% dplyr::group_by(sample_size,pred_or_full)
full_pred_2D_plot <- summarize(grouped, mean=mean(error_est))


p <- full_pred_2D_plot %>%
  ggplot(aes(sample_size, mean, color=pred_or_full)) +
  geom_smooth(se=F) +
  xlab("Sample Size") +
  ylab("Estimated Population Error") +
  scale_y_continuous(labels = comma)

p

q <-
  pred_scatter %>%
  plot_ly(
    x = ~sample_size,
    y = ~threshold,
    z = ~error_est) %>%
  add_markers(size=10) %>%
  layout(scene = list(xaxis = list(title = 'sample size'),
                      yaxis = list(title = 'MSE decision threshold'),
                      zaxis = list(title = 'estimated error in sales
account')))

#  type "q" if you would like to generate
# an interactive 3D graph of
# the predicted population errors
```

In this example, using the deep-learning autoencoder to create error rich samples provided the auditor the ability, for 95% confidence (5% significance) level of audit risk, to estimate much tighter bounds for the population error for any sample size. This is possible, because the deep-learning 'distillation' process solves the problem of highly unbalanced error data $-$ i.e., that there are many more correct transactions than errors.

```
knitr::include_graphics(
```

```
"/home/westland/audit_analytics_book/aaa_chapters/pictures/plotly_just_pred
ictions.png",
    dpi = 300)
```

From the 3D `plotly` graph, it is possible to see the effect of applying different MSE decision thresholds. Values from $5 \ to \ 10$ generate the most stable estimates of population error at all sample sizes. Very high values perform badly for small samples, whereas small values tend not to provide enough information to correct the unbalanced dataset problem.

The `plotly` package can create attractive, interactive 3D plots that are useful for exploring more complex relationships between variables. I am not enthused about widespread application of `plotly`, though, because `phantomjs` (required for it to generate interactive HTML) can significantly slow the response of your computer; nor does `plotly` reliably render its graphics into image formats such as .png, that can be embedded in PDF files.

The 3D plot created by the `plotly` package shows that estimation based on the predicted errors provides much tighter bounds on the sample error (by an order of magnitude at least). Particularly for small sample sizes, estimation based on deep-learning choices for the sampled items is much less variable than estimation from taking a (uniform) random sample over the entire population. Using deep-learning predictive models offers the potential for cheaper and more effective audit risk-cost-scope trade-offs.

The R package `plotly` provides a rich set of 3D plotting tools that allow plots to be rotated, and individual points to be queried by placing the cursor over them. This can be very useful in planning audits, offering the potential for fine tuning of the audit risk-cost trade-offs.

The threshold k is an audit planning decision parameter and in order to optimize audit planning, we want to find the value of k that minimizes audit cost (i.e., sample size) for a given audit risk of incorrectly accepting an account balance that is intolerably (materially) in error, where the intolerable error rate is ρ. Let $\rho = $100,000$ be the intolerable error for the the population. We want k for the minimum sample size such that $population \ error \ge \rho$

```
library(ggplot2)

audit_cost <- size_index <- 0
for (l in 1:nk) {
  size_index[l] <- which.min(pop_error_pred[,l] >= 100000)
  audit_cost[l] <- r[size_index[l]]
#  cat("min size is = ",audit_cost[l] ,"\n\n")

  }

qplot(k, audit_cost, geom = "line") +
  labs(x = "Decision threshold k",
       y = "Audit cost (sample size)")
```

```
# qplot is a simplified form for ggplot,
## which is handy for quick plots
```

Because we are computing decision thresholds using samples (as we would in a real audit) rather than having access to the full population statistics (the real state of the world) the curve is not a smooth monotonic curve, and the guidance provided by by this graph is only approximate. The plot suggests that $k \approx 5 \ to \ 10$ will keep sample sizes under 50 transactions, while providing the desired level of resolution (i.e., such that we can detect any level of $population \ error \ge \rho = $100,000$ with a confidence of greater than 95%).

The choice of suitable thresholds is clearly important in controlling audit costs and quality. More detailed exploration of this audit choice can be accomplished with the `PresenceAbsence` package for R. It provides a set of functions useful when evaluating the results of presence-absence analysis, for example, models of species distribution or the analysis of diagnostic tests. The package provides a toolkit for selecting the optimal threshold for translating a probability surface into presence-absence maps specifically tailored to their intended use. The package includes functions for calculating threshold dependent measures such as confusion matrices, percent correctly classified (PCC), sensitivity, specificity, and Kappa, and produces plots of each measure as the threshold is varied. It also includes functions to plot the Receiver Operator Characteristic (ROC) curve and calculates the associated area under the curve (AUC), a threshold independent measure of model quality. Finally, the package computes optimal thresholds by multiple criteria, and plots these optimized thresholds on the graphs.