MLDS 400 Lab 8

Data Imputation

Huiyu Wu

11/20/2023

NORTHWESTERN UNIVERSITY

Missing Data

- Is data missing at random?
- Should we disregard missing or corrupt data? Or can we still use it?
- Are outliers truely outliers or bad data?

Dataset

- Adjusted closing prices for Dow Jones members from 2018

```
library(quantmod); library(dplyr)
tickers = c("AAPL", "AMGN", "AXP", "BA", "CAT", "CRM", "CSCO",
            "CVX", "DIS", "KO", "GS", "HD", "HON", "IBM", "INTC"
            "JNJ", "JPM", "MCD", "MMM", "MRK", "MSFT", "NKE",
            "PG", "TRV", "UNH", "VZ", "V", "WMT", "WBA")
stocks = lapply(tickers, getSymbols, from="2019-01-01",
                to="2019-12-31", auto.assign=FALSE)
stocks = data.frame(Reduce(function(df1, df2)
    merge(df1, df2,by=0, all.x=T), stocks))
stocks = select(stocks, contains("Adjusted"))
colnames(stocks) = tickers
```

Dataset

```
stocks[1:10,1:5]
```

```
##                AAPL     AMGN      AXP       BA      CAT
## 2019-01-02 37.89333 165.9931 89.08566 314.6451 111.8869
## 2019-01-03 34.11888 163.4673 87.34676 302.1006 107.5754
## 2019-01-04 35.57538 169.0552 91.28265 317.8226 113.4539
## 2019-01-07 35.49620 171.3301 91.77814 318.8234 113.5247
## 2019-01-08 36.17287 173.5272 92.22688 330.8919 114.8881
## 2019-01-09 36.78714 173.3196 92.39515 334.0985 115.3308
## 2019-01-10 36.90472 175.3178 91.86228 342.6300 117.7123
## 2019-01-11 36.54239 173.4840 92.13339 342.9118 116.9509
## 2019-01-14 35.99291 170.2143 92.03056 340.4437 116.6853
## 2019-01-15 36.72957 172.4633 91.60985 342.2705 115.7026
```

Missing Data

```
library(missForest)
```

```
set.seed(400)
# randomly replace 10% of values with NAs
stocks.na = prodNA(stocks, noNA=0.1)
stocks.na[1:5,1:5]
```

```
##                AAPL     AMGN      AXP       BA      CAT
## 2019-01-02 37.89333 165.9931       NA 314.6451 111.8869
## 2019-01-03 34.11888 163.4673 87.34676 302.1006 107.5754
## 2019-01-04 35.57538 169.0552       NA 317.8226       NA
## 2019-01-07 35.49620 171.3301 91.77814 318.8234 113.5247
## 2019-01-08 36.17287 173.5272 92.22688 330.8919 114.8881
```

Missing Data

```r
# check missing data
is.na(stocks.na[1:5,1:5])
```

```
##             AAPL  AMGN   AXP    BA   CAT
## 2019-01-02 FALSE FALSE  TRUE FALSE FALSE
## 2019-01-03 FALSE FALSE FALSE FALSE FALSE
## 2019-01-04 FALSE FALSE  TRUE FALSE  TRUE
## 2019-01-07 FALSE FALSE FALSE FALSE FALSE
## 2019-01-08 FALSE FALSE FALSE FALSE FALSE
```

Handling Missing Data

```r
AAPL.na = stocks.na$AAPL
AAPL.na[1:10]
```

```
##  [1] 37.89333 34.11888 35.57538 35.49620 36.17287 36.78714 36.90472 36.54239
##  [9]       NA 36.72957
```

```r
# disregard missing data
mean(AAPL.na, na.rm=T)
```

```
## [1] 50.31591
```

```r
# remove missing observations
AAPL.omit = na.omit(AAPL.na)
AAPL.omit[1:10]
```

```
##  [1] 37.89333 34.11888 35.57538 35.49620 36.17287 36.78714 36.90472 36.54239
##  [9] 36.72957 37.17828
```

Random Sampling

```
random.imp = function (a){
  missing = is.na(a)
  n.missing = sum(missing) # number of missing values
  a.obs = a[!missing]
  imputed = a
  # sample with replacement
  imputed[missing] = sample(a.obs, n.missing, replace=T)
  return(imputed)
}
AAPL.rndimp = random.imp(AAPL.na)
AAPL.rndimp[1:10]
```

```
##  [1] 37.89333 34.11888 35.57538 35.49620 36.17287 36.78714 36.90472 36.54239
##  [9] 43.45668 36.72957
```

Most Common Value

```r
x = c(1,1,NA,3,4,4,5,5,5,5,6,NA)
# compute the mode
Mode = function(x) {
  mode = as.numeric(names(sort(table(x),decreasing=T))[1])
  return(mode)
}
Mode(x)
```

```
## [1] 5
```

Most Common Value

```r
mcv.imp = function (a){
  missing = is.na(a)
  imputed = a
  imputed[missing] = Mode(a)
  return(imputed)
}
x.mcv = mcv.imp(x)
x.mcv
```

```
##  [1] 1 1 5 3 4 4 5 5 5 5 6 5
```

Average Value

```r
avg.imp = function (a){
  missing = is.na(a)
  imputed = a
  imputed[missing] = mean(a, na.rm=T)
  return(imputed)
}
AAPL.avgimp = avg.imp(AAPL.na)
AAPL.avgimp[1:10]
```

```
##  [1] 37.89333 34.11888 35.57538 35.49620 36.17287 36.78714 36.90472 36.54239
##  [9] 50.31591 36.72957
```

Last Value

```r
AAPL.last = na.locf(AAPL.na)
AAPL.last[1:10]
```

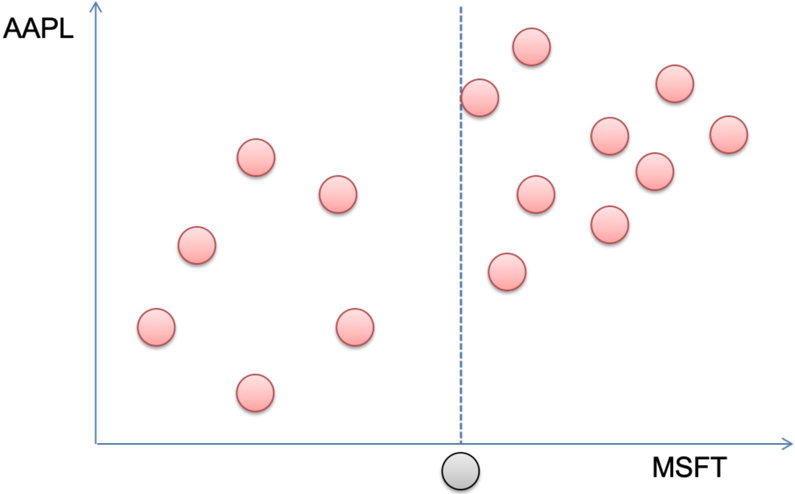
```
##  [1] 37.89333 34.11888 35.57538 35.49620 36.17287 36.78714 36.90472 36.54239
##  [9] 36.54239 36.72957
```

Linear

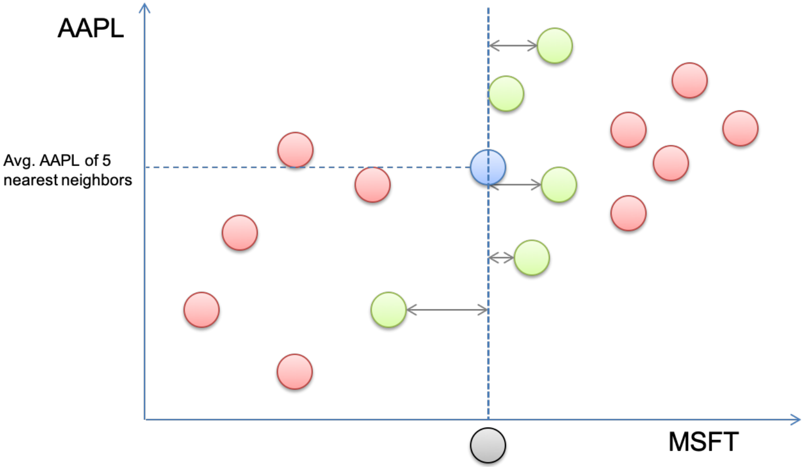
```
n = length(AAPL.na)
AAPL.linear = approxfun(1:n, AAPL.na)(1:n)
AAPL.linear[1:10]
```

```
##  [1] 37.89333 34.11888 35.57538 35.49620 36.17287 36.78714 36.90472 36.54239
##  [9] 36.63598 36.72957
```

k-Nearest Neighbors (k-NN)



k-NN



Imputation with 1-NN

```
library(DMwR)
```

```
stocks.1NN = knnImputation(stocks.na,k=1)
stocks.1NN[1:5,1:5]
```

```
##                 AAPL     AMGN      AXP       BA      CAT
## 2019-01-02 37.89333 165.9931 91.86228 314.6451 111.8869
## 2019-01-03 34.11888 163.4673 87.34676 302.1006 107.5754
## 2019-01-04 35.57538 169.0552 91.86228 317.8226 117.7123
## 2019-01-07 35.49620 171.3301 91.77814 318.8234 113.5247
## 2019-01-08 36.17287 173.5272 92.22688 330.8919 114.8881
```

Imputation with 5-NN

```
stocks.5NN = knnImputation(stocks.na,k=5)
stocks.5NN[1:5,1:5]
```

```
##                AAPL     AMGN      AXP       BA      CAT
## 2019-01-02 37.89333 165.9931 92.17218 314.6451 111.8869
## 2019-01-03 34.11888 163.4673 87.34676 302.1006 107.5754
## 2019-01-04 35.57538 169.0552 91.95454 317.8226 117.7144
## 2019-01-07 35.49620 171.3301 91.77814 318.8234 113.5247
## 2019-01-08 36.17287 173.5272 92.22688 330.8919 114.8881
```

Multivariate Imputation via Chained Equations

- Multivariate Imputation via Chained Equations (MICE)
- "Generates multiple imputations for incomplete multivariate data by Gibbs sampling."
  - Goes column-by-column, imputing missing values, then repeats.
- Default methods:
  - Predictive mean matching (PMM) - numeric data
  - Logistic regression - binary factor
  - Polytomous logistic regression - unordered factor (>2 levels)
  - Proportional odds model - ordered factor (>2 levels)
- Documentation: https://cran.r-project.org/web/packages/mice/mice.pdf

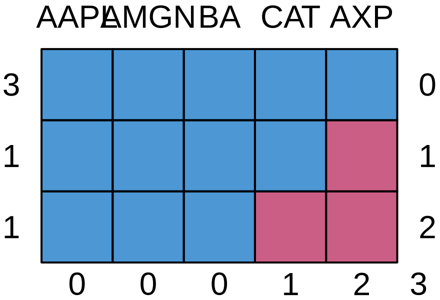
PMM

1. Estimate the linear regression model using y, x, obtaining parameters $\{\hat{\beta}, \Sigma_\beta\}$.
2. Draw $\beta^* \sim \mathcal{N}(\hat{\beta}, \Sigma_\beta)$.
3. Using β^*, generate predicted $\hat{y}$ for all samples (both missing and not).
4. For each missing value y, identify the set of samples with observed y whose predicted value $\hat{y}$ is close to the predicted value of the missing sample.
5. Randomly pick observed value y^* from the set from Step 4 and assign it as the imputed missing value.
6. Repeat.

MICE

```r
library(mice)
```

```r
md.pattern(stocks.na[1:5,1:5])
```




```
##    AAPL AMGN BA CAT AXP
## 3     1    1  1   1   1 0
## 1     1    1  1   1   0 1
## 1     1    1  1   0   0 2
##       0    0  0   1   2 3
```

MICE

```
stocks.mice = mice(stocks.na, seed=400, print=F)
stocks.mice$imp$AAPL[1:5]
```

```
##                   1        2        3        4        5
## 2019-01-14 35.57538 35.57538 37.89333 36.54239 35.49620
## 2019-01-28 37.39903 36.64078 37.62939 36.64078 36.93352
## 2019-02-13 41.18138 41.66816 41.22236 41.80937 41.45610
## 2019-02-15 41.18138 42.16459 41.45610 42.14049 42.01036
## 2019-04-01 45.99916 46.18543 46.18543 45.63871 46.97647
## 2019-04-03 48.34578 46.97647 46.18543 44.95034 48.33358
## 2019-04-16 50.14175 50.40135 49.19237 48.01081 49.12656
## 2019-04-22 48.75693 49.16334 50.40135 49.12656 48.75693
## 2019-05-13 45.48291 45.47809 41.98627 45.48291 43.78881
## 2019-06-06 42.90998 45.01299 43.78881 45.14038 43.59602
## 2019-06-20 49.92180 50.14418 50.23989 49.44879 48.95064
## 2019-07-02 50.14175 49.64232 49.16334 49.12656 48.95064
## 2019-07-12 49.46875 50.74397 49.30729 51.02790 48.80531
## 2019-07-30 52.62104 50.68812 50.75368 49.99891 51.78572
```

MICE

```r
# get the 5th imputation
stocks.miceImp = complete(stocks.mice, 5)
stocks.miceImp[1:5,1:5]
```

```
##                AAPL     AMGN      AXP       BA      CAT
## 2019-01-02 37.89333 165.9931 92.13339 314.6451 111.8869
## 2019-01-03 34.11888 163.4673 87.34676 302.1006 107.5754
## 2019-01-04 35.57538 169.0552 92.39515 317.8226 114.2664
## 2019-01-07 35.49620 171.3301 91.77814 318.8234 113.5247
## 2019-01-08 36.17287 173.5272 92.22688 330.8919 114.8881
```