Identify outliers using Tukey's Method and Hampel's Method.

https://towardsdatascience.com/outlier-detection-with-hampel-filter-85ddf523c73d

outlier detection for categorical variable https://www.kaggle.com/dharssinikarthi/exercise-5/edit

```
data <- read.csv("/content/dataset.csv")
head(data)
```

```
str(data)
```

```
'data.frame':   47453 obs. of  10 variables:
 $ X          : int  0 1 2 3 4 5 6 7 8 9 ...
 $ open       : num  112.9 3.49 115.98 3.59 112.25 ...
 $ high       : num  118.8 3.69 124.66 3.78 113.44 ...
 $ low        : num  107.14 3.35 106.64 3.12 97.7 ...
 $ close      : num  115.91 3.59 112.3 3.37 111.5 ...
 $ volume     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ marketCap  : num  1.29e+09 6.23e+07 1.25e+09 5.86e+07 1.24e+09 ...
 $ timestamp  : chr  "2013-05-05T23:59:59.999Z" "2013-05-05T23:59:59.999Z" "2013-05-06T23:59:59.999Z" "2013-05-06T23:59:59.999Z" ...
 $ crypto_name: chr  "Bitcoin" "Litecoin" "Bitcoin" "Litecoin" ...
 $ date       : chr  "2013-05-05" "2013-05-05" "2013-05-06" "2013-05-06" ...
```

Tukey's Method

```
library(dplyr)

# Calculate the lower and upper bounds using Tukey's Method
variables <- list('open','high','low','close','volume')
for (price in variables){
  q1 <- quantile(data$price, 0.25)
  q3 <- quantile(data$price, 0.75)
  iqr <- q3 - q1
  lower_bound <- q1 - (1.5 * iqr)
  upper_bound <- q3 + (1.5 * iqr)

# Identify outliers
outliers_tukey <- data%>%filter(price < lower_bound | price > upper_bound)}
```

Hampel Method

```
# Check for missing values in the dataset
missing_data <- is.na(data)

# Summarize the number of missing values in each column
colSums(missing_data)

# Handle missing values

#  Impute missing values with mean/median/mode
# Impute missing values in the open column with mean
data$open[is.na(data$open)] <- mean(data$open, na.rm = TRUE)
# Impute missing values in the high column with median
data$high[is.na(data$high)] <- mean(data$high, na.rm = TRUE)
data$close[is.na(data$close)] <- mean(data$close, na.rm = TRUE)
data$volume[is.na(data$volume)] <- mean(data$volume, na.rm = TRUE)
data$low[is.na(data$low)] <- mean(data$low, na.rm = TRUE)

# Impute missing values in the low column with mode
mode_low <- names(which.max(table(data$low)))
data$low[is.na(data$low)] <- mode_low
```

```
install.packages('robustbase')
```

```
     Installing package into ‘/usr/local/lib/R/site-library’
     (as ‘lib’ is unspecified)

     also installing the dependency ‘DEoptimR’
```

```
# Load the pracma package
library(pracma)

# Generate a random dataset with outliers
set.seed(123)
data <- c(rnorm(50), c(10, -10, 12, -8))

# Apply the Hampel filter with a window size of 3
outliers <- hampel(data, k = 3)

# Identify the indices of the outliers
outlier_indices <- which(!is.na(outliers))

# Print the indices of the outliers
cat("Outlier indices:", outlier_indices, "\n")
```

```
data_numeric <- data[c('open','high','close','volume')]
```

```
# Function to calculate Tukey's Method lower and upper bounds
tukey_bounds <- function(x) {
  Q1 <- quantile(x, 0.25)
```

```
  Q3 <- quantile(x, 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  return(c(lower_bound, upper_bound))
}


# Function to calculate Hampel's Method lower and upper bounds
hampel_bounds <- function(x) {
  median <- median(x)
  MAD <- median(abs(x - median))
  lower_bound <- median - 3 * MAD
  upper_bound <- median + 3 * MAD
  return(c(lower_bound, upper_bound))
}

outliers_tukey <- lapply(data_numeric, function(x) {
  bounds <- tukey_bounds(x)
  which(x < bounds[1] | x > bounds[2])
})
print(head(outliers_tukey))
# Identify outliers using Hampel's Method
outliers_hampel <- sapply(data_numeric, function(x) {
  if(is.numeric(x)) {
    bounds <- hampel_bounds(x)
    which(x < bounds[1] | x > bounds[2])
  } else {
    numeric()
  }
})
print(outliers_hampel)
```

```
     Streaming output truncated to the last 5000 lines.
      [5773] 16055 16058 16059 16064 16065 16066 16069 16072 16073 16074 16076 16077
      [5785] 16080 16082 16085 16086 16088 16090 16091 16092 16094 16099 16100 16101
      [5797] 16104 16107 16108 16109 16111 16112 16113 16114 16115 16116 16118 16121
      [5809] 16125 16127 16128 16134 16135 16137 16139 16140 16146 16147 16149 16151
      [5821] 16152 16154 16155 16159 16160 16162 16165 16167 16170 16171 16172 16174
      [5833] 16176 16177 16178 16179 16182 16183 16185 16186 16191 16192 16194 16198
      [5845] 16203 16205 16206 16207 16210 16212 16213 16214 16215 16218 16220 16221
      [5857] 16222 16224 16227 16229 16230 16231 16233 16234 16236 16238 16243 16244
      [5869] 16248 16249 16252 16255 16257 16259 16261 16262 16264 16265 16266 16268
      [5881] 16274 16275 16276 16278 16283 16284 16285 16286 16288 16290 16293 16297
      [5893] 16298 16300 16301 16302 16303 16305 16307 16310 16313 16315 16316 16318
      [5905] 16319 16320 16323 16328 16329 16330 16331 16335 16340 16341 16344 16345
      [5917] 16346 16347 16350 16351 16352 16358 16362 16364 16365 16366 16367 16368
      [5929] 16370 16371 16372 16373 16375 16379 16383 16385 16386 16387 16389 16390
      [5941] 16391 16394 16396 16397 16398 16399 16401 16403 16404 16405 16406 16410
      [5953] 16414 16415 16419 16420 16421 16422 16423 16424 16428 16432 16433 16434
      [5965] 16437 16441 16443 16444 16445 16446 16448 16449 16454 16456 16460 16466
      [5977] 16467 16470 16471 16472 16473 16475 16479 16482 16483 16485 16486 16488
      [5989] 16489 16491 16492 16494 16497 16499 16502 16509 16510 16513 16514 16515
      [6001] 16517 16520 16521 16527 16528 16531 16535 16536 16539 16542 16543 16544
      [6013] 16545 16546 16550 16551 16553 16554 16557 16559 16560 16563 16564 16566
      [6025] 16572 16574 16580 16581 16582 16584 16586 16589 16590 16591 16592 16595
      [6037] 16596 16603 16608 16609 16610 16611 16612 16613 16616 16617 16623 16626
      [6049] 16629 16630 16632 16634 16636 16637 16638 16639 16642 16644 16646 16647
      [6061] 16648 16651 16652 16655 16656 16660 16675 16676 16677 16680 16681 16682
      [6073] 16683 16684 16685 16686 16688 16690 16691 16693 16695 16697 16698 16700
      [6085] 16702 16705 16716 16718 16720 16722 16723 16726 16727 16730 16732 16733
      [6097] 16734 16738 16740 16741 16743 16748 16751 16753 16754 16756 16757 16759
      [6109] 16760 16762 16772 16773 16774 16775 16776 16778 16780 16782 16784 16785
      [6121] 16786 16790 16795 16797 16799 16802 16803 16804 16809 16811 16815 16816
      [6133] 16821 16822 16824 16825 16826 16828 16831 16835 16836 16837 16840 16841
      [6145] 16842 16847 16850 16851 16852 16856 16860 16862 16864 16867 16868 16871
      [6157] 16874 16877 16879 16880 16881 16883 16884 16885 16887 16888 16897 16900
      [6169] 16902 16903 16904 16905 16910 16914 16915 16916 16920 16921 16922 16924
      [6181] 16926 16933 16935 16936 16937 16938 16941 16943 16945 16951 16952 16954
      [6193] 16956 16959 16961 16963 16964 16966 16968 16971 16972 16978 16983 16984
      [6205] 16985 16986 16989 16990 16992 16993 16997 16999 17004 17005 17006 17008
      [6217] 17012 17016 17017 17021 17022 17023 17024 17029 17030 17031 17033 17034
      [6229] 17035 17037 17038 17041 17052 17053 17054 17055 17056 17057 17058 17059
      [6241] 17062 17065 17068 17072 17074 17076 17079 17083 17084 17088 17092 17094
      [6253] 17096 17097 17099 17101 17102 17103 17106 17109 17115 17116 17118 17121
      [6265] 17122 17123 17127 17129 17130 17131 17135 17141 17142 17144 17145 17146
      [6277] 17149 17151 17152 17154 17155 17157 17158 17159 17161 17162 17172 17173
      [6289] 17174 17178 17181 17184 17188 17190 17192 17193 17194 17195 17200 17201
      [6301] 17203 17205 17207 17209 17214 17216 17217 17220 17223 17224 17226 17232
      [6313] 17234 17235 17236 17238 17243 17245 17247 17250 17252 17254 17255 17257
      [6325] 17259 17261 17263 17270 17271 17274 17275 17276 17277 17278 17281 17283
      [6337] 17287 17288 17289 17290 17291 17297 17298 17301 17304 17305 17309 17312
```

```
[6349] 17314 17315 17318 17320 17322 17323 17325 17329 17332 17333 17338 17341
[6361] 17344 17346 17348 17351 17353 17354 17363 17365 17367 17369 17371 17373
[6373] 17374 17376 17377 17378 17380 17381 17382 17385 17386 17388 17393 17394
[6385] 17401 17403 17406 17408 17409 17410 17418 17420 17422 17423 17426 17427
[6397] 17428 17431 17432 17436 17439 17443 17448 17449 17450 17451 17452 17455
[6409] 17456 17457 17462 17463 17464 17466 17468 17469 17479 17480 17481 17482
[6421] 17488 17494 17495 17496 17497 17499 17503 17507 17508 17509 17513 17514
[6433] 17515 17517 17523 17524 17526 17527 17529 17530 17531 17533 17541 17544
[6445] 17545 17547 17551 17555 17557 17558 17560 17562 17568 17569 17572 17574
```

In both methods, the value of k determines the number of standard deviations used to calculate the bounds. In Tukey's Method, a value of 1.5 is commonly used, while in Hampel's Method, a value of 3 is used by default.