Problem Set 3 - Solutions

Applied Statistics and Econometrics II, Spring 2018

Ercan Karadas

February 22, 2018

Contents

Problem 1

```
set.seed(50)
xVec <- sample(0:999, 250, replace=TRUE)
yVec <- sample(0:999, 250, replace=TRUE)
# (a)
aVec <- yVec[-1] - xVec[-length(xVec)]
# (b)
bVec <- sin(yVec[-length(yVec)]) / cos(xVec[-1])
# (c)
cVec <- xVec[-c(249,250)] + 2*xVec[-c(1,250)] - xVec[-c(1,2)] # (d)
dVec <- sum(exp(-xVec[-1])/(xVec[-length(xVec)] + 10))
```

Problem 2

```
# (a)
yVec[yVec>600]
```

```
##   [1] 709 871 621 930 948 783 878 671 860 768 698 974 855 813 776 721 917
##  [18] 985 705 884 840 687 957 955 786 938 930 641 615 988 881 881 997 823
```

```
##  [35] 791 643 779 693 845 815 752 766 635 993 919 686 635 613 660 800 743
##  [52] 965 743 615 615 803 948 760 604 800 772 863 902 689 881 941 924 693
##  [69] 835 632 872 876 850 961 681 791 947 915 712 665 921 798 866 828 942
##  [86] 841 645 681 827 884 890 970 632 717 846 952 609 824 695 675 777 813
## [103] 792 783 611 853 738 668 791
```

```
# (b)
which(yVec > 600)
```

```
##   [1]   1   2   5   6   8  10  11  13  16  18  27  28  32  33  34  36  42
##  [18]  43  45  48  50  55  58  59  60  61  63  66  67  68  72  79  80  86
##  [35]  88  94  95  96  97 101 102 105 107 109 111 114 118 119 120 123 125
##  [52] 127 131 132 134 136 137 138 139 142 143 150 151 154 157 158 159 161
##  [69] 163 164 167 168 172 173 174 175 176 178 180 181 182 183 187 189 190
##  [86] 203 204 205 206 211 213 214 219 220 224 226 227 230 232 237 238 239
## [103] 241 243 245 246 247 249 250
```

```
# (c)
xVec[which(yVec > 600)]
```

```
##   [1] 708 437 513  44 646 107 390 640 676 364 577 257 408 437 618 627 836
##  [18] 278  55 458 803 358 525 511 266 578 197  38 724  61 995 652 956  19
##  [35] 680 760  48 294  69 505 964  24  10 840 878 113 789 444 986 537 515
##  [52] 263 359 189 457 274 543 324 176 160 260 407 216 977 148 293 660 137
##  [69] 852 743 353 371 768 339 203 478  49 880 996 894 357 900 972 467 324
##  [86] 517 446 533 190 501 124  14   5 863 399 256 678 188 258 110 957 285
## [103]  34 631 179 545 123 238 178
```

```
# (d) sqrt(abs(xVec-mean(xVec)))

# (e)
sum(yVec > max(yVec)-200) # (f)
```

```
## [1] 57
```

```
sum(xVec %% 2 == 0)
```

```
## [1] 124
```

```
# (g)
xVec[order(yVec)]
```

```
##   [1] 405 842 308 572 461   8 256 507 373 639  42 616  29 645 376 669 688
##  [18] 197  63 638 862  77 996  93  59 585 661  72 339  20 206 537 174 322
##  [35]  42 603 425  48 707 452 477  99 224 811 715 358 963 222 395 543 480
##  [52] 193 683 710 691 954 700 614 787 835 275 435 309 368 224 460 497 944
##  [69] 530 765 523 171 870 807 469 828 624 200 713 365 781  74 129  76 701
##  [86] 760 193 866 353 168 967 545 920 541 650 148 277  18 667 865 987 120
## [103] 655   1 554 699 311 458 632  84 269  82 280 544  17 621 807 113 136
## [120] 457 702  91 625 767 828 109 860 363 121 657 668 324 382 956 299 403
## [137]  74 928 415  38 127 176 678 179 444 724 189 457 513 743   5  10 789
## [154]  38 760 446 986 894 238 640 110 203 533 113 358 977 294 137 258 577
## [171]  55 708 996 863 627 123 515 359 964 324  24 364 260 618 957  48 107
## [188] 631 266 680 478 178  34 900 537 160 274 437 285 505  19 188 190 467
## [205] 852 803 517  69 399 768 545 408 676 407 972 437 353 371 390 995 652
## [222] 148 458 501 124 216 880 836 878 357 660  44 197 578 293 324  49 646
## [239] 543 256 511 525 339 263  14 257 278  61 840 956
```

```
# (h)
yVec[c(T,F,F)]
```

```
##  [1] 709 517 437 783 671 860 581 347 279 974 216 776 538 460 985 248 317
## [18] 288 687 957 938 101 615 285 106 414 881 488 484 791 246 643 845 553
## [35] 465  87 993 116 473 635 310 428 965  19 489 803 604 800 175 516 902
## [52] 689 881 593 835 398 358 850 791 915 665 167 866 942 320 482 216 488
## [69] 681 273 884 970 469 717 127 952 284 695 325 777 792  72 738 791
```

Problem 3

```
set.seed(75)
aMat <- matrix(sample(10, size=60, replace=T), nrow=6)
```

(a) `apply` returns a vector or array or list of values obtained by applying a function to margins of an array or matrix

```
apply(aMat, 1, function(x) sum(x>4)) # (b)
```

```
## [1] 4 7 6 2 6 7
```

```
which(apply(aMat, 1, function(x) {sum(x==7)==2} ))
```

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

(c1) repetition permitted

```
set.seed(75)
aMat <- matrix(sample(10, size=60, replace=T), nrow=6)
aMatColSums <- colSums(aMat)
which(outer(aMatColSums,aMatColSums,"+")>75, arr.ind = T)
```

```
##      row col
## [1,]   2   2
## [2,]   6   2
## [3,]   8   2
## [4,]   2   6
## [5,]   8   6
## [6,]   2   8
## [7,]   6   8
## [8,]   8   8
```

(c2) repetition not permitted

```
aMatColSums <- colSums(aMat)
logicalMat <- outer(aMatColSums,aMatColSums,"+")>75
```

When we run the above statement, we can see 10*10 T/F matrix, indicating that True means sum of the columns is bigger than 75, and False means those less than 75.

```
logicalMat[lower.tri(logicalMat,diag=T)] <- NA
```

There is no need to repeat the columns, and so remove the lower triangle.

```
which(logicalMat, arr.ind=T)
```

```
##      row col
```

```
## [1,]   2   6
## [2,]   2   8
## [3,]   6   8
```

Then, only for True components, R reads the columns.

Problem 4

```
# (a)
tmpFn1 <- function(x){ x^(1:length(x))
}
tmpFn2 <- function(x){
x^(1:length(x))/(1:length(x)) }

# (b)
tmpFn3 <- function(x,n){
n <- length(x)
sum((x^(1:n))/(1:n)) + 1 }
```

Problem 5

```
tmpFn <- function(x,n){
n <- length(x)
(x[-c(n-1,n)] + x[-c(n,1)] + x[-c(1,2)])/3 }
tmpFn(c(1:5, 6:1))
```

```
## [1] 2.000000 3.000000 4.000000 5.000000 5.333333 5.000000 4.000000 3.000000
## [9] 2.000000
```

Problem 6

```
tmpFn <- function(y){
if(y < 0) {y^2 + 2*y + 3} else if (y < 2) {y + 3}
else {y^2 + 4*y - 7}
}
```

Problem 7

```
# (a)
tmpFn1 <- function(x){ a <- x - mean(x)
b <- sum(a^2)
n <- length(x)
r1 <- sum( a[-c(1)]*a[-c(n)] ) / b
list(r1) }
x <- seq(2, 56, by=3) tmpFn1(x)
```

```
# (b)
tmpFn2 <- function(x){
a <- x - mean(x)
b <- sum(a^2)
n <- length(x)
r1 <- sum( a[-c(1)]*a[-c(n)] )/ b
r2 <- sum( a[-c(1,2)]*a[-c(n, n-1)] )/ b*list(r1, r2)
}
x <- seq(2, 56, by=3) tmpFn2(x)
```

Problem 8

```
# lecture notes - "for"
primeFinder1 <- function(n){ if(n >=2) {
x <- seq(2, n) primes <- NULL for(i in seq(2,n)){
if(any(x==i)){
primes <- c(primes, i)
x <- c(x[(x %% i) != 0], i)
} }
return(primes) }
}

# Solution - "while" primeFinder2 <- function(n){
if(n >= 2) {
x <- seq(2, n)
primes <- NULL
while (length(x) > 0){
j <- x[1]
primes <- c(primes, j) x <- x[(x %% j) != 0] }
return(primes) }
}
```

Problem 9

```
primeFinder1 <- function(n){
if(n >=2) {
x <- seq(2, n) primes <- NULL for(i in seq(2,n)){
if(any(x==i)){
primes <- c(primes, i)
x <- c(x[(x %% i) != 0], i)
} }
return(primes)
} else {
stop("Input value of n should be at least 2.") }
}
twinPrimesFinder <- function(n) { primes <- primeFinder1(n)
lg <- length(primes)
twinwhere <- diff(primes)==2 twin1 <- primes[-m]
```

```
twin2 <- primes[-1] cbind(twin1[twinwhere], twin2[twinwhere])
}
```

Problem 10

```
# (a)
x <- 1
fNewton <- cos(x) - exp(x) tolerance <- 0.00000001
while (abs(fNewton) > tolerance) {
fNewtonPrime <- -sin(x) - exp(x) x <- x - fNewton/fNewtonPrime fNewton <- cos(x) - exp(x)
} x

# (b)
fNewtonFun <- function(x){
fNewton <- cos(x) - exp(x) tolerance <- 0.00000001
if (abs(fNewton) < tolerance) {
fNewtonPrime <- -sin(x) - exp(x) x <- x - fNewton/fNewtonPrime fNewton <- cos(x) - exp(x)
x
} else {print(NA)} }
```