# Lab 1

Jaime Lin

11:59PM February 18, 2021

You should have RStudio installed to edit this file. You will write code in places marked "TO-DO" to complete the problems. Some of this will be a pure programming assignment. The tools for the solutions to these problems can be found in the class practice lectures. I want you to use the methods I taught you, not for you to google and come up with whatever works. You won't learn that way.

To "hand in" the homework, you should compile or publish this file into a PDF that includes output of your code. Once it's done, push by the deadline to your repository in a directory called "labs".

- Print out the numerical constant pi with ten digits after the decimal point using the internal constant `pi`.

```
options(digit=11)
pi
```

```
## [1] 3.141593
```

- Sum up the first 103 terms of the series 1 + 1/2 + 1/4 + 1/8 + ...

```
sum(1/(2^(0:102)))
```

```
## [1] 2
```

- Find the product of the first 37 terms in the sequence 1/3, 1/6, 1/9 ...

```
prod(1/(3*(1:37)))
```

```
## [1] 1.613529e-61
```

- Find the product of the first 387 terms of `1 * 1/2 * 1/4 * 1/8 *` ...

```
prod(1/(2^(0:386)))
```

```
## [1] 0
```

Is this answer *exactly* correct?

#No exactly correct because we express a numerical overflow

- Figure out a means to express the answer more exactly. Not compute exactly, but express more exactly.

```
sum(log(1/2^(0:386)))
```

```
## [1] -51771.86
```

- Create the sequence `x = [Inf, 20, 18, ..., -20]`.

```
x = c(Inf, seq(from = 20, to = -20, by=-2))
x
```

```
##  [1] Inf  20  18  16  14  12  10   8   6   4   2   0  -2  -4  -6  -8 -10 -
12 -14
## [20] -16 -18 -20
```

Create the sequence `x = [log_3(Inf), log_3(100), log_3(98), ... log_3(-20)]`.

```
x= c(Inf, seq(from=100, to=-20, by=-2))
x =log(x, base=3)
```

```
## Warning: NaNs produced
```

Comment on the appropriateness of the non-numeric values.

#TO-DO inf, -inf, nans talk about it

- Create a vector of booleans where the entry is true if `x[i]` is positive and finite.

```
y= !is.nan(x) & is.finite(x) & x>0
y
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE
## [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE
## [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE
## [49]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## [61] FALSE FALSE
```

- Locate the indices of the non-real numbers in this vector. Hint: use the `which` function. Don't hesitate to use the documentation via `?which`.

```
?which
```

```
## starting httpd help server ... done
```

```
which(y==FALSE)
```

```
##  [1]  1 52 53 54 55 56 57 58 59 60 61 62
```

- Locate the indices of the infinite quantities in this vector.

```
which(is.infinite(x))
```

```
## [1]  1 52
```

- Locate the indices of the min and max in this vector. Hint: use the `which.min` and `which.max` functions.

```
which.min(x)
```

```
## [1] 52
```

```
which.max(x)
```

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

- Count the number of unique values in x.

```
length(unique(x))
```

```
## [1] 53
```

- Cast x to a factor. Do the number of levels make sense?

```
as.factor(x)
```

```
##  [1] Inf               4.19180654857877  4.1734172518943
4.15464876785729
##  [5] 4.13548512895119  4.11590933734319  4.09590327428938
4.07544759935851
##  [9] 4.05452163806914  4.03310325630434  4.01116871959141
3.98869253500376
## [13] 3.96564727304425  3.94200336638929  3.91772888178973
3.89278926071437
## [17] 3.86714702345081  3.84076143030548  3.81358809221559
3.78557852142874
## [21] 3.75667961082847  3.72683302786084  3.69597450568212
3.66403300987579
## [25] 3.63092975357146  3.59657702661571  3.56087679500731
3.52371901428583
## [29] 3.48497958377173  3.44451784578705  3.40217350273288  3.3577627814323
## [33] 3.31107361281783  3.26185950714291  3.20983167673402
3.15464876785729
## [37] 3.09590327428938  3.03310325630434  2.96564727304425
2.89278926071437
## [41] 2.8135880922156   2.72683302786084  2.63092975357146
2.52371901428583
## [45] 2.40217350273288  2.26185950714291  2.09590327428938
1.89278926071437
## [49] 1.63092975357146  1.26185950714291  0.630929753571457 -Inf
## [53] NaN               NaN               NaN               NaN
## [57] NaN               NaN               NaN               NaN
## [61] NaN               NaN
## 53 Levels: -Inf 0.630929753571457 1.26185950714291 ... NaN
```

- Cast x to integers. What do we learn about R's infinity representation in the integer data type?

```
as.integer((x))
```

```
## Warning: NAs introduced by coercion to integer range
```

```
##  [1] NA  4  4  4  4  4  4  4  4  4  4  3  3  3  3  3  3  3  3  3  3  3
3  3
```

```
## [26]  3  3  3  3  3  3  3  3  3  3  3  3  3  2  2  2  2  2  2  2  2  2  1
1  1
## [51]  0 NA NA NA NA NA NA NA NA NA NA
```

- Use x to create a new vector y containing only the real numbers in x.

```
y= x[(!is.nan(x) & is.finite(x) & x>0)]
y
```

```
##  [1] 4.1918065 4.1734173 4.1546488 4.1354851 4.1159093 4.0959033 4.0754476
##  [8] 4.0545216 4.0331033 4.0111687 3.9886925 3.9656473 3.9420034 3.9177289
## [15] 3.8927893 3.8671470 3.8407614 3.8135881 3.7855785 3.7566796 3.7268330
## [22] 3.6959745 3.6640330 3.6309298 3.5965770 3.5608768 3.5237190 3.4849796
## [29] 3.4445178 3.4021735 3.3577628 3.3110736 3.2618595 3.2098317 3.1546488
## [36] 3.0959033 3.0331033 2.9656473 2.8927893 2.8135881 2.7268330 2.6309298
## [43] 2.5237190 2.4021735 2.2618595 2.0959033 1.8927893 1.6309298 1.2618595
## [50] 0.6309298
```

- Use the left rectangle method to numerically integrate x^2 from 0 to 1 with rectangle width size 1e-6.

```
sum(seq(from=0, to=1-1e-6, by= 1e-6)^2 * 1e-6)
```

```
## [1] 0.3333328
```

- Calculate the average of 100 realizations of standard Bernoullis in one line using the `sample` function.

```
sample(c(0,1), size=100, replace=TRUE)
```

```
##   [1] 0 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1
1 0 1
##  [38] 1 1 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 1
0 1 0
##  [75] 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 0 1 1 0 1
```

- Calculate the average of 500 realizations of Bernoullis with p = 0.9 in one line using the `sample` and `mean` functions.

```
#TO-DO
mean(sample(c(0,1), size=500, replace=TRUE, prob = c(0.1,0.9)))
```

```
## [1] 0.906
```

- Calculate the average of 1000 realizations of Bernoullis with p = 0.9 in one line using `rbinom`.

```
#TO-DO
?rbinom
mean(rbinom(n=1000, size=1, prob=0.9))
```

```
## [1] 0.889
```

- In class we considered a variable `x_3` which measured "criminality". We imagined L = 4 levels "none", "infraction", "misdimeanor" and "felony". Create a variable `x_3` here

with 100 random elements (equally probable). Create it as a nominal (i.e. unordered) factor.

```
#TO-DO
x_3=as.factor(sample(c("none", "infraction", "misdimeanor", "felony"),
size=100, replace=TRUE))
x_3
```

```
##   [1] felony      infraction  infraction  felony      none        none
##   [7] felony      infraction  none        misdimeanor infraction  felony
##  [13] infraction  infraction  felony      infraction  felony
misdimeanor
##  [19] none        none        felony      infraction  none
infraction
##  [25] infraction  infraction  misdimeanor felony      infraction  none
##  [31] felony      misdimeanor misdimeanor infraction  misdimeanor none
##  [37] felony      felony      none        felony      none
misdimeanor
##  [43] none        misdimeanor none        none        none        none
##  [49] none        felony      felony      misdimeanor infraction
misdimeanor
##  [55] none        none        infraction  felony      infraction  none
##  [61] infraction  misdimeanor none        infraction  none
misdimeanor
##  [67] none        misdimeanor misdimeanor none        misdimeanor felony
##  [73] infraction  infraction  felony      misdimeanor misdimeanor
infraction
##  [79] felony      none        infraction  felony      infraction  felony
##  [85] infraction  misdimeanor infraction  none        none        felony
##  [91] infraction  none        misdimeanor infraction  misdimeanor
infraction
##  [97] felony      felony      misdimeanor none
## Levels: felony infraction misdimeanor none
```

- Use `x_3` to create `x_3_bin`, a binary feature where 0 is no crime and 1 is any crime.

```
#TO-DO
x_3_bin=x_3!="none"
as.integer(x_3_bin)
```

```
##   [1] 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1
1 0 1
##  [38] 1 0 1 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 1 1 0 1
1 1 1
##  [75] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0
```

- Use `x_3` to create `x_3_ord`, an ordered factor variable. Ensure the proper ordinal ordering.

```
#TO-DO
factor(x_3, level=c("none", "infraction", "misdimeanor", "felony"),
order=TRUE)
```

```
##   [1] felony      infraction  infraction  felony      none        none
##   [7] felony      infraction  none        misdimeanor infraction  felony
##  [13] infraction  infraction  felony      infraction  felony
misdimeanor
##  [19] none        none        felony      infraction  none
infraction
##  [25] infraction  infraction  misdimeanor felony      infraction  none
##  [31] felony      misdimeanor misdimeanor infraction  misdimeanor none
##  [37] felony      felony      none        felony      none
misdimeanor
##  [43] none        misdimeanor none        none        none        none
##  [49] none        felony      felony      misdimeanor infraction
misdimeanor
##  [55] none        none        infraction  felony      infraction  none
##  [61] infraction  misdimeanor none        infraction  none
misdimeanor
##  [67] none        misdimeanor misdimeanor none        misdimeanor felony
##  [73] infraction  infraction  felony      misdimeanor misdimeanor
infraction
##  [79] felony      none        infraction  felony      infraction  felony
##  [85] infraction  misdimeanor infraction  none        none        felony
##  [91] infraction  none        misdimeanor infraction  misdimeanor
infraction
##  [97] felony      felony      misdimeanor none
## Levels: none < infraction < misdimeanor < felony
```

- Convert this variable into three binary variables without any information loss and put them into a data matrix.

```
#TO-DO
x_3_infraction = as.integer(x_3 == "infraction")
x_3_misdimeanor = as.integer(x_3 == "misdimeanor")
x_3_felony = as.integer(x_3 == "felony")

x_1 = cbind(x_3_infraction,x_3_misdimeanor, x_3_felony)
head(x_1)
```

```
##      x_3_infraction x_3_misdimeanor x_3_felony
## [1,]              0               0          1
## [2,]              1               0          0
## [3,]              1               0          0
## [4,]              0               0          1
## [5,]              0               0          0
## [6,]              0               0          0
```

- What should the sum of each row be (in English)?

  #TO-DO row = sum of each column r_n = c_1 + c_2 + ... + c_n

  Verify that.

```
#TO-DO
rowSums(x_1)
```

```
##   [1] 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1
1 0 1
##  [38] 1 0 1 0 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 1 1 0 1
1 1 1
##  [75] 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0
```

- How should the column sum look (in English)?

  #TO-DO column = sum of each row c_n = r_1 + r_2 + … + r_n Verify that.

```
#TO-DO
colSums(x_1)
```

```
##  x_3_infraction x_3_misdimeanor      x_3_felony
##              28              21              23
```

- Generate a matrix with 100 rows where the first column is realization from a normal with mean 17 and variance 38, the second column is uniform between -10 and 10, the third column is poisson with mean 6, the fourth column in exponential with lambda of 9, the fifth column is binomial with n = 20 and p = 0.12 and the sixth column is a binary variable with exactly 24% 1's dispersed randomly. Name the rows the entries of the `fake_first_names` vector.

```
fake_first_names = c(
  "Sophia", "Emma", "Olivia", "Ava", "Mia", "Isabella", "Riley",
  "Aria", "Zoe", "Charlotte", "Lily", "Layla", "Amelia", "Emily",
  "Madelyn", "Aubrey", "Adalyn", "Madison", "Chloe", "Harper",
  "Abigail", "Aaliyah", "Avery", "Evelyn", "Kaylee", "Ella", "Ellie",
  "Scarlett", "Arianna", "Hailey", "Nora", "Addison", "Brooklyn",
  "Hannah", "Mila", "Leah", "Elizabeth", "Sarah", "Eliana", "Mackenzie",
  "Peyton", "Maria", "Grace", "Adeline", "Elena", "Anna", "Victoria",
  "Camilla", "Lillian", "Natalie", "Jackson", "Aiden", "Lucas",
  "Liam", "Noah", "Ethan", "Mason", "Caden", "Oliver", "Elijah",
  "Grayson", "Jacob", "Michael", "Benjamin", "Carter", "James",
  "Jayden", "Logan", "Alexander", "Caleb", "Ryan", "Luke", "Daniel",
  "Jack", "William", "Owen", "Gabriel", "Matthew", "Connor", "Jayce",
  "Isaac", "Sebastian", "Henry", "Muhammad", "Cameron", "Wyatt",
  "Dylan", "Nathan", "Nicholas", "Julian", "Eli", "Levi", "Isaiah",
  "Landon", "David", "Christian", "Andrew", "Brayden", "John",
  "Lincoln"
)
#TO-DO
c1 = array(rnorm(100, mean = 17, sd =1))
c2 = array(runif(100, min = -10, max = 10))
c3 = array(rpois(100, 6))
c4 = array(rexp(100, rate = 9))
c5 = array(rbinom(20, 100, 0.12))
c6 = as.integer(array(runif(100, min = 0, max = 2)))
```

```
matrix(cbind( c1, c2, c3, c4, c5, c6), nrow = 100, ncol = 6, byrow = FALSE,
               dimnames = list(c(fake_first_names)))
```

```
##                [,1]        [,2] [,3]         [,4] [,5] [,6]
## Sophia    16.12361 -8.74025072    5 0.0762568411   13    1
## Emma      17.83516  5.77415116    4 0.0936848912   15    0
## Olivia    17.48396  9.68880859    5 0.0590601829   15    0
## Ava       17.82395 -9.69184048    5 0.5577194965    9    0
## Mia       16.01077  1.55219307    6 0.0264056455   16    1
## Isabella  17.39127  3.68296113   11 0.0286468505   11    0
## Riley     16.04047 -4.66795492    5 0.2853221589    7    1
## Aria      17.84073  6.59712660    9 0.1348112108   10    1
## Zoe       15.47106 -1.78166745    4 0.0564198722   12    0
## Charlotte 15.43276 -6.33327097    8 0.0125838031   12    1
## Lily      17.83559 -1.44573286    4 0.0702650206   12    0
## Layla     17.63684 -8.81768078    8 0.1310266470   19    0
## Amelia    16.56390 -6.64186231    4 0.0326414157   12    0
## Emily     16.59789  9.22865061    2 0.0567445340   11    0
## Madelyn   16.84311 -5.91104363    6 0.0781918970   11    1
## Aubrey    18.66345 -8.18212644   10 0.1354201241   13    0
## Adalyn    18.81702 -0.96363829    8 0.1282133915    9    1
## Madison   16.11808  3.99374623    7 0.0233321552   13    0
## Chloe     16.80301 -8.28420297    4 0.0991520952   12    1
## Harper    16.14262 -1.68772390    5 0.0235916287   13    1
## Abigail   15.63024  5.96682265    8 0.0665016229   13    1
## Aaliyah   17.79693  5.78798761    9 0.2329490020   15    0
## Avery     17.99255  8.66313612    9 0.0316453310   15    1
## Evelyn    17.50992 -5.30124666   10 0.0458677567    9    0
## Kaylee    16.77537  0.13119581    6 0.0311416689   16    1
## Ella      17.04158 -2.28440166    8 0.0054706588   11    1
## Ellie     18.54062  4.42587610    9 0.1185029991    7    0
## Scarlett  15.74817  6.25403825    5 0.0489806839   10    0
## Arianna   17.63029 -4.71677228    6 0.1431969423   12    1
## Hailey    16.71991 -1.06751177    7 0.1298251104   12    1
## Nora      17.19299 -9.43505317   10 0.1481553612   12    1
## Addison   18.77396  9.33887162    3 0.1587274004   19    0
## Brooklyn  16.37517  2.04324080    7 0.0059412989   12    0
## Hannah    18.55940  6.80750509    3 0.0873378601   11    1
## Mila      15.19625 -9.61471350    7 0.1425452740   11    1
## Leah      15.66855  7.59202674    6 0.0067958645   13    1
## Elizabeth 16.25514 -5.94852164    9 0.1579353695    9    1
## Sarah     15.58681  4.21359124    8 0.0133161804   13    1
## Eliana    17.74770  6.19714737    2 0.0143005590   12    0
## Mackenzie 15.83333  9.52572500    3 0.0844640256   13    1
## Peyton    16.64757  1.17935279    4 0.0302875429   13    1
## Maria     15.19856 -5.39656905    6 0.0840618241   15    0
## Grace     16.07068 -9.97620929    3 0.0412463840   15    0
## Adeline   17.17374  6.37652265    2 0.0019893711    9    0
## Elena     16.90403  1.79944555    7 0.0726736313   16    1
```

```
## Anna      17.26034 -9.20637979    6 0.0196379576   11    0
## Victoria  17.48073 -9.98504871    1 0.0968650314    7    1
## Camilla   17.22019 -6.97139095    6 0.0095822896   10    1
## Lillian   17.25578 -3.94239814    8 0.0072260350   12    1
## Natalie   18.82133  3.83021162    4 0.0430498167   12    0
## Jackson   17.95039 -1.62387440    1 0.0409666053   12    0
## Aiden     17.02345  2.99899399    5 0.0397281223   19    1
## Lucas     16.17361  6.60676893    2 0.1355936808   12    1
## Liam      17.43387 -5.19216275    6 0.0870119264   11    1
## Noah      18.13995 -5.42501913   11 0.1053539922   11    1
## Ethan     18.03418  1.19216203    8 0.0874727645   13    0
## Mason     15.91975  5.10468131   11 0.0118644439    9    0
## Caden     17.81754  4.02424421    2 0.3493241196   13    0
## Oliver    16.65905 -0.61405737    4 0.0347145824   12    1
## Elijah    16.04828 -3.50183816    5 0.0768697639   13    0
## Grayson   17.59281  2.34314903    6 0.1163676459   13    1
## Jacob     15.86508  3.45999213    4 0.0726146150   15    0
## Michael   17.12571  5.31906025    5 0.0726647678   15    1
## Benjamin  17.77608 -3.95781695    6 0.0401111450    9    0
## Carter    16.21830  6.43266378    3 0.0440442834   16    0
## James     15.47093  1.84555532    9 0.0913321944   11    1
## Jayden    16.13502  3.37165303    4 0.1227545160    7    1
## Logan     18.22695  4.65371799    5 0.1862429707   10    1
## Alexander 16.82767  5.66659398    5 0.2329891188   12    1
## Caleb     17.07208  6.53972517    9 0.0336843657   12    1
## Ryan      17.13335  0.48073907    6 0.1270214765   12    1
## Luke      17.68441 -6.54666059    6 0.0511892980   19    1
## Daniel    17.17239 -9.79936886    6 0.1148626576   12    1
## Jack      18.27388  7.14593718    8 0.0440154689   11    0
## William   17.73572  0.40086662    3 0.0258676673   11    0
## Owen      17.08802 -0.12719825    3 0.1288589368   13    1
## Gabriel   16.88729  7.18351573    6 0.0948930223    9    0
## Matthew   16.69265  1.81269559    7 0.1714466777   13    0
## Connor    16.10685  4.36256766    6 0.1197370046   12    0
## Jayce     17.75890  6.77362106    4 0.1806850611   13    0
## Isaac     16.32447 -0.77620344    6 0.0920256421   13    1
## Sebastian 16.03286  8.45392088   12 0.1589538465   15    1
## Henry     15.16575  6.94444241    0 0.0822696192   15    1
## Muhammad  17.85751 -3.21890803    4 0.1933743925    9    0
## Cameron   17.91636  9.64150066    5 0.2678073791   16    1
## Wyatt     17.09973 -7.63934949    9 0.0845928943   11    1
## Dylan     16.49604  9.23511063    6 0.1070154886    7    1
## Nathan    18.86272  3.06956161    7 0.0587962424   10    1
## Nicholas  16.90841 -8.13216809    7 0.3317673049   12    1
## Julian    17.88337  0.01403837    4 0.1460125690   12    0
## Eli       16.97801 -3.17237061    6 0.0124675828   12    1
## Levi      16.35534  1.79676476    8 0.1113295463   19    0
## Isaiah    17.87684  7.69653420    7 0.1615965095   12    0
## Landon    17.13017 -4.64955956    7 0.0073232461   11    1
## David     17.07462 -6.28374964    4 0.0184754493   11    0
```

```
## Christian 15.31391  8.92506078    7 0.0006058172   13    0
## Andrew    16.22410 -6.07958504    5 0.0314784738    9    1
## Brayden   17.01517  1.40115873    9 0.2241621636   13    1
## John      18.20900  7.27152971   10 0.0917966189   12    1
## Lincoln   17.47020 -7.96394372    3 0.1933096738   13    1
```

- Create a data frame of the same data as above except make the binary variable a factor "DOMESTIC" vs "FOREIGN" for 0 and 1 respectively. Use RStudio's `View` function to ensure this worked as desired.

```
#TO-DO
m_1 <- matrix(as.integer(array(runif(100, min = 0, max = 2))), nrow=100,
ncol=6)


View(m_1, "DOMESTIC vs FOREIGN")
```

- Print out a table of the binary variable. Then print out the proportions of "DOMESTIC" vs "FOREIGN".

```
#TO-DO
print(m_1)
```

```
##        [,1] [,2] [,3] [,4] [,5] [,6]
##   [1,]    1    1    1    1    1    1
##   [2,]    1    1    1    1    1    1
##   [3,]    1    1    1    1    1    1
##   [4,]    1    1    1    1    1    1
##   [5,]    1    1    1    1    1    1
##   [6,]    0    0    0    0    0    0
##   [7,]    1    1    1    1    1    1
##   [8,]    1    1    1    1    1    1
##   [9,]    1    1    1    1    1    1
##  [10,]    1    1    1    1    1    1
##  [11,]    0    0    0    0    0    0
##  [12,]    1    1    1    1    1    1
##  [13,]    1    1    1    1    1    1
##  [14,]    0    0    0    0    0    0
##  [15,]    0    0    0    0    0    0
##  [16,]    0    0    0    0    0    0
##  [17,]    0    0    0    0    0    0
##  [18,]    0    0    0    0    0    0
##  [19,]    0    0    0    0    0    0
##  [20,]    0    0    0    0    0    0
##  [21,]    0    0    0    0    0    0
##  [22,]    0    0    0    0    0    0
##  [23,]    0    0    0    0    0    0
##  [24,]    0    0    0    0    0    0
##  [25,]    0    0    0    0    0    0
##  [26,]    1    1    1    1    1    1
##  [27,]    0    0    0    0    0    0
##  [28,]    1    1    1    1    1    1
```

```
##  [29,]    1    1    1    1    1    1
##  [30,]    1    1    1    1    1    1
##  [31,]    1    1    1    1    1    1
##  [32,]    0    0    0    0    0    0
##  [33,]    1    1    1    1    1    1
##  [34,]    1    1    1    1    1    1
##  [35,]    0    0    0    0    0    0
##  [36,]    1    1    1    1    1    1
##  [37,]    0    0    0    0    0    0
##  [38,]    1    1    1    1    1    1
##  [39,]    0    0    0    0    0    0
##  [40,]    1    1    1    1    1    1
##  [41,]    1    1    1    1    1    1
##  [42,]    0    0    0    0    0    0
##  [43,]    0    0    0    0    0    0
##  [44,]    0    0    0    0    0    0
##  [45,]    1    1    1    1    1    1
##  [46,]    1    1    1    1    1    1
##  [47,]    1    1    1    1    1    1
##  [48,]    0    0    0    0    0    0
##  [49,]    1    1    1    1    1    1
##  [50,]    1    1    1    1    1    1
##  [51,]    1    1    1    1    1    1
##  [52,]    1    1    1    1    1    1
##  [53,]    0    0    0    0    0    0
##  [54,]    0    0    0    0    0    0
##  [55,]    0    0    0    0    0    0
##  [56,]    1    1    1    1    1    1
##  [57,]    0    0    0    0    0    0
##  [58,]    1    1    1    1    1    1
##  [59,]    0    0    0    0    0    0
##  [60,]    1    1    1    1    1    1
##  [61,]    1    1    1    1    1    1
##  [62,]    0    0    0    0    0    0
##  [63,]    1    1    1    1    1    1
##  [64,]    0    0    0    0    0    0
##  [65,]    0    0    0    0    0    0
##  [66,]    1    1    1    1    1    1
##  [67,]    1    1    1    1    1    1
##  [68,]    1    1    1    1    1    1
##  [69,]    0    0    0    0    0    0
##  [70,]    1    1    1    1    1    1
##  [71,]    0    0    0    0    0    0
##  [72,]    1    1    1    1    1    1
##  [73,]    1    1    1    1    1    1
##  [74,]    1    1    1    1    1    1
##  [75,]    0    0    0    0    0    0
##  [76,]    0    0    0    0    0    0
##  [77,]    0    0    0    0    0    0
##  [78,]    0    0    0    0    0    0
```

```
##  [79,]    1    1    1    1    1    1
##  [80,]    0    0    0    0    0    0
##  [81,]    1    1    1    1    1    1
##  [82,]    0    0    0    0    0    0
##  [83,]    1    1    1    1    1    1
##  [84,]    0    0    0    0    0    0
##  [85,]    0    0    0    0    0    0
##  [86,]    1    1    1    1    1    1
##  [87,]    0    0    0    0    0    0
##  [88,]    1    1    1    1    1    1
##  [89,]    1    1    1    1    1    1
##  [90,]    0    0    0    0    0    0
##  [91,]    1    1    1    1    1    1
##  [92,]    0    0    0    0    0    0
##  [93,]    1    1    1    1    1    1
##  [94,]    1    1    1    1    1    1
##  [95,]    1    1    1    1    1    1
##  [96,]    1    1    1    1    1    1
##  [97,]    1    1    1    1    1    1
##  [98,]    1    1    1    1    1    1
##  [99,]    1    1    1    1    1    1
## [100,]    1    1    1    1    1    1
```

Print out a summary of the whole dataframe.

```
#TO-DO
?summary
summary(m_1)
```

```
##        V1              V2              V3              V4              V5
##  Min.   :0.00   Min.   :0.00   Min.   :0.00   Min.   :0.00   Min.   :0.00
##  1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.00
##  Median :1.00   Median :1.00   Median :1.00   Median :1.00   Median :1.00
##  Mean   :0.56   Mean   :0.56   Mean   :0.56   Mean   :0.56   Mean   :0.56
##  3rd Qu.:1.00   3rd Qu.:1.00   3rd Qu.:1.00   3rd Qu.:1.00   3rd Qu.:1.00
##  Max.   :1.00   Max.   :1.00   Max.   :1.00   Max.   :1.00   Max.   :1.00
##        V6
##  Min.   :0.00
##  1st Qu.:0.00
##  Median :1.00
##  Mean   :0.56
##  3rd Qu.:1.00
##  Max.   :1.00
```

- Let `n = 50`. Create a n x n matrix `R` of exactly 50% entries 0's, 25% 1's 25% 2's. These values should be in random locations.

```
#TO-DO
m_2 <-matrix(c(0:2), nrow=50, ncol=50, byrow = FALSE)
```

```
## Warning in matrix(c(0:2), nrow = 50, ncol = 50, byrow = FALSE): data
length [3]
## is not a sub-multiple or multiple of the number of rows [50]

m_2

##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    0    2    1    0    2    1    0    2    1     0     2     1     0
##  [2,]    1    0    2    1    0    2    1    0    2     1     0     2     1
##  [3,]    2    1    0    2    1    0    2    1    0     2     1     0     2
##  [4,]    0    2    1    0    2    1    0    2    1     0     2     1     0
##  [5,]    1    0    2    1    0    2    1    0    2     1     0     2     1
##  [6,]    2    1    0    2    1    0    2    1    0     2     1     0     2
##  [7,]    0    2    1    0    2    1    0    2    1     0     2     1     0
##  [8,]    1    0    2    1    0    2    1    0    2     1     0     2     1
##  [9,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [10,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [11,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [12,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [13,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [14,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [15,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [16,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [17,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [18,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [19,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [20,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [21,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [22,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [23,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [24,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [25,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [26,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [27,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [28,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [29,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [30,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [31,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [32,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [33,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [34,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [35,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [36,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [37,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [38,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [39,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [40,]    0    2    1    0    2    1    0    2    1     0     2     1     0
## [41,]    1    0    2    1    0    2    1    0    2     1     0     2     1
## [42,]    2    1    0    2    1    0    2    1    0     2     1     0     2
## [43,]    0    2    1    0    2    1    0    2    1     0     2     1     0
```

```
## [44,]    1    0    2    1    0    2    1    0    2    1    0    2    1
## [45,]    2    1    0    2    1    0    2    1    0    2    1    0    2
## [46,]    0    2    1    0    2    1    0    2    1    0    2    1    0
## [47,]    1    0    2    1    0    2    1    0    2    1    0    2    1
## [48,]    2    1    0    2    1    0    2    1    0    2    1    0    2
## [49,]    0    2    1    0    2    1    0    2    1    0    2    1    0
## [50,]    1    0    2    1    0    2    1    0    2    1    0    2    1
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[,25]
##  [1,]     2     1     0     2     1     0     2     1     0     2     1
0
##  [2,]     0     2     1     0     2     1     0     2     1     0     2
1
##  [3,]     1     0     2     1     0     2     1     0     2     1     0
2
##  [4,]     2     1     0     2     1     0     2     1     0     2     1
0
##  [5,]     0     2     1     0     2     1     0     2     1     0     2
1
##  [6,]     1     0     2     1     0     2     1     0     2     1     0
2
##  [7,]     2     1     0     2     1     0     2     1     0     2     1
0
##  [8,]     0     2     1     0     2     1     0     2     1     0     2
1
##  [9,]     1     0     2     1     0     2     1     0     2     1     0
2
## [10,]     2     1     0     2     1     0     2     1     0     2     1
0
## [11,]     0     2     1     0     2     1     0     2     1     0     2
1
## [12,]     1     0     2     1     0     2     1     0     2     1     0
2
## [13,]     2     1     0     2     1     0     2     1     0     2     1
0
## [14,]     0     2     1     0     2     1     0     2     1     0     2
1
## [15,]     1     0     2     1     0     2     1     0     2     1     0
2
## [16,]     2     1     0     2     1     0     2     1     0     2     1
0
## [17,]     0     2     1     0     2     1     0     2     1     0     2
1
## [18,]     1     0     2     1     0     2     1     0     2     1     0
2
## [19,]     2     1     0     2     1     0     2     1     0     2     1
0
## [20,]     0     2     1     0     2     1     0     2     1     0     2
1
## [21,]     1     0     2     1     0     2     1     0     2     1     0
```

```
2
## [22,]    2    1    0    2    1    0    2    1    0    2    1
0
## [23,]    0    2    1    0    2    1    0    2    1    0    2
1
## [24,]    1    0    2    1    0    2    1    0    2    1    0
2
## [25,]    2    1    0    2    1    0    2    1    0    2    1
0
## [26,]    0    2    1    0    2    1    0    2    1    0    2
1
## [27,]    1    0    2    1    0    2    1    0    2    1    0
2
## [28,]    2    1    0    2    1    0    2    1    0    2    1
0
## [29,]    0    2    1    0    2    1    0    2    1    0    2
1
## [30,]    1    0    2    1    0    2    1    0    2    1    0
2
## [31,]    2    1    0    2    1    0    2    1    0    2    1
0
## [32,]    0    2    1    0    2    1    0    2    1    0    2
1
## [33,]    1    0    2    1    0    2    1    0    2    1    0
2
## [34,]    2    1    0    2    1    0    2    1    0    2    1
0
## [35,]    0    2    1    0    2    1    0    2    1    0    2
1
## [36,]    1    0    2    1    0    2    1    0    2    1    0
2
## [37,]    2    1    0    2    1    0    2    1    0    2    1
0
## [38,]    0    2    1    0    2    1    0    2    1    0    2
1
## [39,]    1    0    2    1    0    2    1    0    2    1    0
2
## [40,]    2    1    0    2    1    0    2    1    0    2    1
0
## [41,]    0    2    1    0    2    1    0    2    1    0    2
1
## [42,]    1    0    2    1    0    2    1    0    2    1    0
2
## [43,]    2    1    0    2    1    0    2    1    0    2    1
0
## [44,]    0    2    1    0    2    1    0    2    1    0    2
1
## [45,]    1    0    2    1    0    2    1    0    2    1    0
2
## [46,]    2    1    0    2    1    0    2    1    0    2    1
```

```
## [47,]     0     2     1     0     2     1     0     2     1     0     2
1
## [48,]     1     0     2     1     0     2     1     0     2     1     0
2
## [49,]     2     1     0     2     1     0     2     1     0     2     1
0
## [50,]     0     2     1     0     2     1     0     2     1     0     2
1
##       [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36]
[,37]
##  [1,]     2     1     0     2     1     0     2     1     0     2     1
0
##  [2,]     0     2     1     0     2     1     0     2     1     0     2
1
##  [3,]     1     0     2     1     0     2     1     0     2     1     0
2
##  [4,]     2     1     0     2     1     0     2     1     0     2     1
0
##  [5,]     0     2     1     0     2     1     0     2     1     0     2
1
##  [6,]     1     0     2     1     0     2     1     0     2     1     0
2
##  [7,]     2     1     0     2     1     0     2     1     0     2     1
0
##  [8,]     0     2     1     0     2     1     0     2     1     0     2
1
##  [9,]     1     0     2     1     0     2     1     0     2     1     0
2
## [10,]     2     1     0     2     1     0     2     1     0     2     1
0
## [11,]     0     2     1     0     2     1     0     2     1     0     2
1
## [12,]     1     0     2     1     0     2     1     0     2     1     0
2
## [13,]     2     1     0     2     1     0     2     1     0     2     1
0
## [14,]     0     2     1     0     2     1     0     2     1     0     2
1
## [15,]     1     0     2     1     0     2     1     0     2     1     0
2
## [16,]     2     1     0     2     1     0     2     1     0     2     1
0
## [17,]     0     2     1     0     2     1     0     2     1     0     2
1
## [18,]     1     0     2     1     0     2     1     0     2     1     0
2
## [19,]     2     1     0     2     1     0     2     1     0     2     1
0
## [20,]     0     2     1     0     2     1     0     2     1     0     2
```

```
1
## [21,]    1    0    2    1    0    2    1    0    2    1    0
2
## [22,]    2    1    0    2    1    0    2    1    0    2    1
0
## [23,]    0    2    1    0    2    1    0    2    1    0    2
1
## [24,]    1    0    2    1    0    2    1    0    2    1    0
2
## [25,]    2    1    0    2    1    0    2    1    0    2    1
0
## [26,]    0    2    1    0    2    1    0    2    1    0    2
1
## [27,]    1    0    2    1    0    2    1    0    2    1    0
2
## [28,]    2    1    0    2    1    0    2    1    0    2    1
0
## [29,]    0    2    1    0    2    1    0    2    1    0    2
1
## [30,]    1    0    2    1    0    2    1    0    2    1    0
2
## [31,]    2    1    0    2    1    0    2    1    0    2    1
0
## [32,]    0    2    1    0    2    1    0    2    1    0    2
1
## [33,]    1    0    2    1    0    2    1    0    2    1    0
2
## [34,]    2    1    0    2    1    0    2    1    0    2    1
0
## [35,]    0    2    1    0    2    1    0    2    1    0    2
1
## [36,]    1    0    2    1    0    2    1    0    2    1    0
2
## [37,]    2    1    0    2    1    0    2    1    0    2    1
0
## [38,]    0    2    1    0    2    1    0    2    1    0    2
1
## [39,]    1    0    2    1    0    2    1    0    2    1    0
2
## [40,]    2    1    0    2    1    0    2    1    0    2    1
0
## [41,]    0    2    1    0    2    1    0    2    1    0    2
1
## [42,]    1    0    2    1    0    2    1    0    2    1    0
2
## [43,]    2    1    0    2    1    0    2    1    0    2    1
0
## [44,]    0    2    1    0    2    1    0    2    1    0    2
1
## [45,]    1    0    2    1    0    2    1    0    2    1    0
```

```
2
## [46,]    2    1    0    2    1    0    2    1    0    2    1
0
## [47,]    0    2    1    0    2    1    0    2    1    0    2
1
## [48,]    1    0    2    1    0    2    1    0    2    1    0
2
## [49,]    2    1    0    2    1    0    2    1    0    2    1
0
## [50,]    0    2    1    0    2    1    0    2    1    0    2
1
##       [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
[,49]
##  [1,]    2    1    0    2    1    0    2    1    0    2    1
0
##  [2,]    0    2    1    0    2    1    0    2    1    0    2
1
##  [3,]    1    0    2    1    0    2    1    0    2    1    0
2
##  [4,]    2    1    0    2    1    0    2    1    0    2    1
0
##  [5,]    0    2    1    0    2    1    0    2    1    0    2
1
##  [6,]    1    0    2    1    0    2    1    0    2    1    0
2
##  [7,]    2    1    0    2    1    0    2    1    0    2    1
0
##  [8,]    0    2    1    0    2    1    0    2    1    0    2
1
##  [9,]    1    0    2    1    0    2    1    0    2    1    0
2
## [10,]    2    1    0    2    1    0    2    1    0    2    1
0
## [11,]    0    2    1    0    2    1    0    2    1    0    2
1
## [12,]    1    0    2    1    0    2    1    0    2    1    0
2
## [13,]    2    1    0    2    1    0    2    1    0    2    1
0
## [14,]    0    2    1    0    2    1    0    2    1    0    2
1
## [15,]    1    0    2    1    0    2    1    0    2    1    0
2
## [16,]    2    1    0    2    1    0    2    1    0    2    1
0
## [17,]    0    2    1    0    2    1    0    2    1    0    2
1
## [18,]    1    0    2    1    0    2    1    0    2    1    0
2
## [19,]    2    1    0    2    1    0    2    1    0    2    1
```

```
## [20,]    0    2    1    0    2    1    0    2    1    0    2
1
## [21,]    1    0    2    1    0    2    1    0    2    1    0
2
## [22,]    2    1    0    2    1    0    2    1    0    2    1
0
## [23,]    0    2    1    0    2    1    0    2    1    0    2
1
## [24,]    1    0    2    1    0    2    1    0    2    1    0
2
## [25,]    2    1    0    2    1    0    2    1    0    2    1
0
## [26,]    0    2    1    0    2    1    0    2    1    0    2
1
## [27,]    1    0    2    1    0    2    1    0    2    1    0
2
## [28,]    2    1    0    2    1    0    2    1    0    2    1
0
## [29,]    0    2    1    0    2    1    0    2    1    0    2
1
## [30,]    1    0    2    1    0    2    1    0    2    1    0
2
## [31,]    2    1    0    2    1    0    2    1    0    2    1
0
## [32,]    0    2    1    0    2    1    0    2    1    0    2
1
## [33,]    1    0    2    1    0    2    1    0    2    1    0
2
## [34,]    2    1    0    2    1    0    2    1    0    2    1
0
## [35,]    0    2    1    0    2    1    0    2    1    0    2
1
## [36,]    1    0    2    1    0    2    1    0    2    1    0
2
## [37,]    2    1    0    2    1    0    2    1    0    2    1
0
## [38,]    0    2    1    0    2    1    0    2    1    0    2
1
## [39,]    1    0    2    1    0    2    1    0    2    1    0
2
## [40,]    2    1    0    2    1    0    2    1    0    2    1
0
## [41,]    0    2    1    0    2    1    0    2    1    0    2
1
## [42,]    1    0    2    1    0    2    1    0    2    1    0
2
## [43,]    2    1    0    2    1    0    2    1    0    2    1
0
## [44,]    0    2    1    0    2    1    0    2    1    0    2
```

```
1
## [45,]    1    0    2    1    0    2    1    0    2    1    0
2
## [46,]    2    1    0    2    1    0    2    1    0    2    1
0
## [47,]    0    2    1    0    2    1    0    2    1    0    2
1
## [48,]    1    0    2    1    0    2    1    0    2    1    0
2
## [49,]    2    1    0    2    1    0    2    1    0    2    1
0
## [50,]    0    2    1    0    2    1    0    2    1    0    2
1
##       [,50]
##  [1,]    2
##  [2,]    0
##  [3,]    1
##  [4,]    2
##  [5,]    0
##  [6,]    1
##  [7,]    2
##  [8,]    0
##  [9,]    1
## [10,]    2
## [11,]    0
## [12,]    1
## [13,]    2
## [14,]    0
## [15,]    1
## [16,]    2
## [17,]    0
## [18,]    1
## [19,]    2
## [20,]    0
## [21,]    1
## [22,]    2
## [23,]    0
## [24,]    1
## [25,]    2
## [26,]    0
## [27,]    1
## [28,]    2
## [29,]    0
## [30,]    1
## [31,]    2
## [32,]    0
## [33,]    1
## [34,]    2
## [35,]    0
## [36,]    1
```

```
## [37,]     2
## [38,]     0
## [39,]     1
## [40,]     2
## [41,]     0
## [42,]     1
## [43,]     2
## [44,]     0
## [45,]     1
## [46,]     2
## [47,]     0
## [48,]     1
## [49,]     2
## [50,]     0
```

- Randomly punch holes (i.e. `NA`) values in this matrix so that an each entry is missing with probability 30%.

```
#TO-DO
```

- Sort the rows in matrix `R` by the largest row sum to lowest. Be careful about the NA's!

```
#TO-DO
r_Sum = rowSums(m_2)
r_Sum
```

```
##  [1] 50 49 51 50 49 51 50 49 51 50 49 51 50 49 51 50 49 51 50 49 51 50 49
51 50
## [26] 49 51 50 49 51 50 49 51 50 49 51 50 49 51 50 49 51 50 49 51 50 49 51
50 49
```

```
sort(r_Sum, decreasing = TRUE)
```

```
##  [1] 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 50 50 50 50 50 50 50
50 50
## [26] 50 50 50 50 50 50 50 50 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49
49 49
```

- We will now learn the `apply` function. This is a handy function that saves writing for loops which should be eschewed in R. Use the apply function to compute a vector whose entries are the standard deviation of each row. Use the apply function to compute a vector whose entries are the standard deviation of each column. Be careful about the NA's! This should be one line.

```
#TO-DO
?apply
print("Row")
```

```
## [1] "Row"
```

```
row_Means = apply(m_2, 1, sd)
row_Means
```

```
##  [1] 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931
##  [8] 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031
## [15] 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031
## [22] 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931
## [29] 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031
## [36] 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031
## [43] 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931
## [50] 0.8204031
```

```
print("columns")
```

```
## [1] "columns"
```

```
col_Means = apply(m_2, 2, sd)
col_Means
```

```
##  [1] 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031
##  [8] 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931
## [15] 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031
## [22] 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031
## [29] 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931
## [36] 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031
## [43] 0.8204031 0.8329931 0.8204031 0.8204031 0.8329931 0.8204031 0.8204031
## [50] 0.8329931
```

- Use the `apply` function to compute a vector whose entries are the count of entries that are 1 or 2 in each column. This should be one line.

```
#TO-DO

count = function(m_2){
  count_0 = 0
  for(i in 1:length(m_2)){
    if(m_2[i]== 1 || m_2[i]== 2)
      count_0 = count_0 +1
  }
  count_0
}
apply(m_2, 2, count)
```

```
##  [1] 33 33 34 33 33 34 33 33 34 33 33 34 33 33 34 33 33 34 33 33 34 33 33
34 33
## [26] 33 34 33 33 34 33 33 34 33 33 34 33 33 34 33 33 34 33 33 34 33 33 34
33 33
```

- Use the `split` function to create a list whose keys are the column number and values are the vector of the columns. Look at the last example in the documentation `?split`.

```
#TO-DO
?split
g_0 <- cbind(a= c("a", "b", "c"), b= c(1:3))
split(g_0, col(g_0))
```

```
## $`1`
## [1] "a" "b" "c"
##
## $`2`
## [1] "1" "2" "3"
```

- In one statement, use the `lapply` function to create a list whose keys are the column number and values are themselves a list with keys: "min" whose value is the minimum of the column, "max" whose value is the maximum of the column, "pct_missing" is the proportion of missingness in the column and "first_NA" whose value is the row number of the first time the NA appears.

```
#TO-DO
?lapply
l_0 <- list(a=1:10, b= c("a","b","c","d", "e", "f", "g", "h", "i", "j"))
lapply(l_0, min)

## $a
## [1] 1
##
## $b
## [1] "a"

lapply(l_0, max)

## $a
## [1] 10
##
## $b
## [1] "j"
```

- Set a seed and then create a vector v consisting of a sample of 1,000 iid normal realizations with mean -10 and variance 100.

```
#TO-DO
set.seed(1000)
v <- sample(c(-31:10), size = 1000, replace = TRUE)
mean(v)

## [1] -10.108

var(v)

## [1] 137.1655
```

- Repeat this exercise by resetting the seed to ensure you obtain the same results.

```
#TO-DO
rm(v)
set.seed(1000)
v <- sample(c(-31:10), size = 1000, replace = TRUE)
mean(v)

## [1] -10.108
```

```
var(v)

## [1] 137.1655
```

- Find the average of v and the standard error of v.

```
#TO-DO
mean(v)

## [1] -10.108

standard_error = sd(v) / 1000
standard_error

## [1] 0.01171177
```

- Find the 5%ile of v and use the qnorm function to compute what it theoretically should be. Is the estimate about what is expected by theory?

```
#TO-DO
?qnorm
qnorm(0.05, mean = -10)

## [1] -11.64485
```

- What is the percentile of v that corresponds to the value 0? What should it be theoretically? Is the estimate about what is expected by theory?

```
#TO-DO
j=0
for (i in v==0){
  j = j+i
}
j

## [1] 19

percent = (j/1000)*100
print(percent)

## [1] 1.9
```