Lab 1

Shoara Chowdhury

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. Most of this will be a pure programming assignment but there are some questions that instead ask you to "write a few sentences". This is a W class! The tools for the solutions to these problems can be found in the class practice lectures. I prefer you to use the methods I taught you. If you google and find esoteric code you don't understand, this doesn't do you too much good.

To "hand in" the homework, you should first download this file. The best way to do this is by cloning the class repository then copying this file from the folder of that clone into the folder that is your personal class repository. Then do the assignment by filling in the TO-DO's. After you're done, compile this file into a PDF (use the "knit to PDF" button on the submenu above). This PDF will include output of your code. Then push the PDF and this Rmd file by the deadline to your github repository in a directory called "labs".

Basic R Skills

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

```
options(digits=11)
x <- pi
x
```

```
## [1] 3.1415926536
```

- 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] 1.613528728e-61
```

```
prod(1/seq(from=3, by=3, length.out=37))
```

```
## [1] 1.613528728e-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?

#TO-DO

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

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

```
## [1] -51771.856063
```

```
-log(2)*sum(0:386)
```

```
## [1] -51771.856063
```

- 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
```

```
log(100, 3)
```

```
## [1] 4.1918065486
```

Comment on the appropriateness of the non-numeric values.

NAN occurs because you cannot take the log of a negative number. -Inf occurs when you take the log of 0.

- 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
which(!y)
```

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

```
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  2  2  2  2  2  2  2  2  2  1  1  1
## [51]  0 NA 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)]
y
```

```
##  [1] 4.19180654858 4.17341725189 4.15464876786 4.13548512895 4.11590933734
##  [6] 4.09590327429 4.07544759936 4.05452163807 4.03310325630 4.01116871959
## [11] 3.98869253500 3.96564727304 3.94200336639 3.91772888179 3.89278926071
## [16] 3.86714702345 3.84076143031 3.81358809222 3.78557852143 3.75667961083
## [21] 3.72683302786 3.69597450568 3.66403300988 3.63092975357 3.59657702662
## [26] 3.56087679501 3.52371901429 3.48497958377 3.44451784579 3.40217350273
## [31] 3.35776278143 3.31107361282 3.26185950714 3.20983167673 3.15464876786
## [36] 3.09590327429 3.03310325630 2.96564727304 2.89278926071 2.81358809222
## [41] 2.72683302786 2.63092975357 2.52371901429 2.40217350273 2.26185950714
```

```
## [46] 2.09590327429 1.89278926071 1.63092975357 1.26185950714 0.63092975357
```

- 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] 0.33333283333
```

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

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

```
## [1] 0.45
```

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

```
sum(sample(c(0,1), size=500, replace=TRUE, prob=c(0.1, 0.9)))/500
```

```
## [1] 0.888
```

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

```
?rbinom
rbinom(n=1000, size=1, p=0.9)
```

```
##    [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1
##   [38] 1 1 1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1
##   [75] 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
##  [112] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0
##  [149] 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
##  [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
##  [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
##  [260] 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [297] 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 1
##  [334] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
##  [371] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1
##  [408] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
##  [445] 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [482] 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1
##  [519] 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1
##  [556] 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1
##  [593] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [630] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1
##  [667] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [704] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1
##  [741] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0
##  [778] 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
##  [815] 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1
##  [852] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
##  [889] 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
##  [926] 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
##  [963] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [1000] 0
```

- 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.

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

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

```
x_3_bin = x_3 != "none"
x_3_bin
```

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

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

```
x_3_ord = factor(x_3, levels = c("none",  "infraction", "misdimeanor", "felony"), order=TRUE)
x_3_ord
```

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

```
##  [85] none        felony      felony      felony      none        misdimeanor
##  [91] felony      felony      infraction  felony      none        misdimeanor
##  [97] none        felony      misdimeanor infraction
## Levels: none < infraction < misdimeanor < felony
```

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

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

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

```
p_level_names = levels(x_3)[-1]
X = matrix(NA, nrow = length(x_3), ncol = length(p_level_names))
colnames(X) = p_level_names
for (j in 1 : length(p_level_names)){
  X[, j] = as.integer(x_3 == p_level_names[j])
}
head(X)
```

```
##      infraction misdimeanor none
## [1,]          0           0    1
## [2,]          0           0    0
## [3,]          0           0    1
## [4,]          0           0    0
## [5,]          0           1    0
## [6,]          0           1    0
```

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

#TO-DO

Verify that.

```
rowSums(X)
```

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

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

#TO-DO

Verify that.

```
colSums(X)
```

```
##  infraction misdimeanor        none
##          17          27          27
```

- 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"
)
rownames(X)= fake_first_names
X
```

```
##           infraction misdimeanor none
## Sophia             0           0    1
## Emma               0           0    0
## Olivia             0           0    1
## Ava                0           0    0
## Mia                0           1    0
## Isabella           0           1    0
## Riley              1           0    0
## Aria               0           1    0
## Zoe                0           0    0
## Charlotte          0           0    1
## Lily               0           0    1
## Layla              0           0    0
## Amelia             0           1    0
## Emily              1           0    0
## Madelyn            0           0    0
## Aubrey             0           0    1
## Adalyn             0           1    0
## Madison            1           0    0
## Chloe              0           0    1
## Harper             0           1    0
## Abigail            1           0    0
## Aaliyah            0           0    1
## Avery              0           0    1
## Evelyn             0           1    0
## Kaylee             0           0    1
## Ella               1           0    0
## Ellie              0           0    1
```

```
## Scarlett          0          0    0
## Arianna           1          0    0
## Hailey            0          0    0
## Nora              1          0    0
## Addison           0          0    0
## Brooklyn          1          0    0
## Hannah            0          1    0
## Mila              1          0    0
## Leah              0          1    0
## Elizabeth         0          1    0
## Sarah             1          0    0
## Eliana            0          1    0
## Mackenzie         0          1    0
## Peyton            1          0    0
## Maria             0          0    1
## Grace             0          0    1
## Adeline           0          0    1
## Elena             0          1    0
## Anna              0          0    0
## Victoria          0          0    0
## Camilla           0          1    0
## Lillian           0          0    0
## Natalie           0          0    0
## Jackson           0          0    1
## Aiden             0          1    0
## Lucas             0          1    0
## Liam              0          0    0
## Noah              0          0    0
## Ethan             0          0    1
## Mason             0          0    1
## Caden             0          0    0
## Oliver            0          0    1
## Elijah            0          0    1
## Grayson           0          1    0
## Jacob             1          0    0
## Michael           0          0    1
## Benjamin          0          1    0
## Carter            0          1    0
## James             0          0    1
## Jayden            0          0    1
## Logan             0          0    0
## Alexander         0          0    0
## Caleb             0          0    0
## Ryan              1          0    0
## Luke              0          1    0
## Daniel            1          0    0
## Jack              0          0    1
## William           0          1    0
## Owen              0          0    0
## Gabriel           0          1    0
## Matthew           0          0    0
## Connor            1          0    0
## Jayce             0          0    0
## Isaac             0          0    0
```

```
## Sebastian          0          1    0
## Henry              0          1    0
## Muhammad           0          0    1
## Cameron            0          0    1
## Wyatt              0          0    0
## Dylan              0          0    0
## Nathan             0          0    0
## Nicholas           0          0    1
## Julian             0          1    0
## Eli                0          0    0
## Levi               0          0    0
## Isaiah             1          0    0
## Landon             0          0    0
## David              0          0    1
## Christian          0          1    0
## Andrew             0          0    1
## Brayden            0          0    0
## John               0          1    0
## Lincoln            1          0    0
```

- 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.

```
#levels = c(0,1)
#fact = c('DOMESTIC', 'FOREIGN')
#s_frame = data.frame(fake_first_names, fact, levels)
#View(name_frame)
```

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

```
#print.table(s_frame)
```

Print out a summary of the whole dataframe.

```
#summary(s_frame)
```

- 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.

```
#R = data.matrix(sample(c(rep(0, n^2**.5), rep(1, n^2**.25), rep(2, n^2*.25))), nrow=50, ncol=50)
```

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

```
n = 100
X = matrix(rnorm(n^2), nrow = n, ncol = n)
for (i in 1 : n){
  for (j in 1 : n){
    if (runif(1) < 0.3){
      X[i,j] = NA
    }
  }
}
```

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

```
#TO DO
```

- 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.

```
#apply(B, nrow=B, sd, na.rm = TRUE)
#apply(B, ncol=B, sd, na.rm = TRUE)
```

- 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.

```
#apply(B>0, MARGIN=2, na.rm=TRUE, sum)
```

- 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`.

```
#splt(B, col(B))
#?split
```

- 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
```

- 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.

```
set.seed(3)
i = rnorm(n = 1000, mean=-10, sd=10)
```

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

```
num=(1:10)
for(number in num)
  set.seed(n)
  i = rnorm(n=1000, mean=-10, sd=10)
  print(i)
```

```
##    [1] -15.021923505315  -8.684688346727 -10.789170898189  -1.132151905822
##    [5]  -8.830287294892  -6.813699123830 -15.817906847159  -2.854672891084
##    [9] -18.252594258628 -13.598621313955  -9.101138562225  -9.037255397149
##   [13] -12.016339521834  -2.601595001216  -8.766204989111 -10.293167092293
##   [17] -13.888542469035  -4.891437426301 -19.138141853692  13.102968227791
##   [21] -14.380899811273  -2.359393835896  -7.380387086071  -2.265954032216
##   [25] -18.143791248755 -14.384505690746 -17.202215502161  -7.690554677474
##   [29] -21.577294623998  -7.529240072712 -10.911135614728   7.573756220280
##   [33] -11.379296117311 -11.111934952824 -16.900143205694 -12.217942300197
##   [37]  -8.170923164053  -5.826767137733   0.654023270455  -0.297979826595
##   [41] -11.016292384616   4.032034885582 -27.767756322946  -3.771326090090
##   [45] -15.222833512770   3.222309556847 -13.634403268447   3.190657425688
##   [49]  -9.562209323557 -28.786558820062 -14.470621820196 -27.385979472079
##   [53]  -8.211351513728   8.974657001462 -32.719254860193  -0.195358612654
##   [57] -23.988256162924   8.248724229436   3.812987299288 -18.388518750344
##   [61] -12.619957745140 -10.688440280170 -13.788835565301  15.819589277243
##   [65]  -8.701658626544 -17.130249802164  -3.620057570805  -7.983084083841
##   [69] -10.699169482402 -10.924898754027  -5.510967268886 -20.643556706874
##   [73] -21.624193221422   6.485217467045 -30.620960193399  -9.872502790350
##   [77] -20.875283493104  -7.294605067957   0.084518732727 -30.744047542564
```

```
##   [81]  -1.031777282144 -10.499957668760 -23.453493104434 -29.312115343497
##   [85]  -2.904184165229 -11.579050319163  -7.836321272915  -1.826379242247
##   [89]   7.271757545032 -11.037702924633 -15.571222908243   4.283014298765
##   [93] -18.929574021001 -21.575712400727 -15.302964549291  14.456827576668
##   [97] -18.324957979974  -5.864801511701 -21.786831407246 -21.740347584692
##  [101] -13.329233509745   3.631137069225 -14.691473395764  -1.571243678634
##  [105] -24.579937225421 -14.003059200489 -17.764172853363 -13.692965113010
##  [109]   2.401014586168 -11.074338083400  -8.274064937758  -7.453987317514
##  [113] -16.145338289944 -24.292150960010 -13.309754346160  -8.716139366239
##  [117]   0.181199924631 -12.555736915638 -13.025410106927   6.151906825923
##  [121] -17.737133545473  -5.759975984373 -15.839469813294  -5.849643211009
##  [125] -25.452616566240 -15.187495047481 -12.797915543527   0.074573820758
##  [129] -14.695699536150  -7.021029616226 -14.177944330726 -18.503807762912
##  [133]  -3.109538056276 -14.601961947620   3.481843775449  -5.569286157365
##  [137] -11.509261883608  -5.444511431852 -10.401546812987  -5.438789564536
##  [141] -14.084250294878 -31.364938556101  -8.431780836134  -3.399510988267
##  [145] -19.818344130736 -21.136437042034 -14.373476767294 -15.161112466734
##  [149]  -5.810040094290  -8.658445632844   0.346864545560   6.535032260147
##  [153] -10.179468173969 -10.242033206186  -7.497530995251 -13.371245360173
##  [157] -11.133537049771 -10.988829149127  -7.359131771839  -8.610163141087
##  [161] -12.422694988384  -9.409686182769 -11.772718683839  -2.053197318399
##  [165]  -9.932622132941 -16.297902929938 -12.524897825636 -16.904221634974
##  [169]  -7.974578547521  -1.536185621876  -3.679259381939  -7.985864753967
##  [173] -10.910706437284  -7.105158749034 -10.546849392054 -30.418498536218
##  [177]  -6.416307585131 -13.726008516161   2.683088405183  11.686003171661
##  [181] -22.397228421705  -4.101261117665  -8.759807114251 -15.237077864447
##  [185]  -3.797719969803  -2.917784155116 -10.931983511756 -12.951967009138
##  [189] -20.858152308948 -16.248150564397 -12.330065431369 -12.508168637578
##  [193]  -0.461046637761 -12.659725066733   8.952759466334 -14.299908272486
##  [197]   5.755469944189  -8.380588002767 -20.854529069457  -4.230626994755
##  [201]  -9.718282337831 -13.567034057779  -1.473736245908  -4.866347517756
##  [205]   0.182029971727 -20.214790847453 -15.616682709396 -20.125560782345
##  [209] -40.208142990154  -6.676497322833   2.405115662888  -3.286504007226
##  [213] -23.300341111215 -18.505803137182 -27.888307424182   2.315219140500
##  [217] -13.343996735927 -19.673221017859  -1.206296516188 -12.535786898251
##  [221] -25.178757889443 -10.238832551874  -9.733405266690  -8.363187999390
##  [225]  -5.899920605489 -16.273625999191  -3.630819961473 -22.019347812166
##  [229]   3.465424125393 -15.987673103425  -5.577955622311  -3.864735912455
##  [233] -12.990963727338 -26.005880952048 -13.394669553132 -15.813077459912
##  [237] -13.903620100123  -1.544856449512  -3.152727949971  -7.622869164850
##  [241] -17.106218863921  16.133189575741 -26.266473514250 -26.073062930030
##  [245]  -6.596825541943  17.278877071192 -13.271900395189 -22.568967482633
##  [249] -14.313981059368  15.488532476093 -17.714080019969 -15.073423537235
##  [253] -12.700225837864  -2.518834365738  -3.314061895394 -14.552986600815
##  [257] -29.278452242947   8.819770669939 -22.497111299343 -14.893449689024
##  [261]  -7.846087148120  -5.778913263960 -22.211482788687  -6.877721793777
##  [265]  -1.135378659074  -5.251893072729  -1.301586452638 -18.865237959718
##  [269] -11.286555314610   0.965104949590  -4.819704533882  -6.019668248955
##  [273]   0.666594104453 -14.495400812962  -3.738980157994 -18.188359507145
##  [277] -12.635162084606  -8.454247139744  -4.315544504868 -36.601596774607
##  [281]   1.365325417305  -5.782272240708   3.500825971574   1.037568850857
##  [285]  -3.529538902374  -8.243641897930  -3.464422132813 -10.654903037694
##  [289]  -3.544778273328  -6.357059633258  -1.565584318637 -16.764788656649
##  [293] -13.914329030771  -5.950593559294 -21.068586953188 -13.833004080089
```

```
##  [297]  -7.059797517489   7.930606680725  -1.859678289291 -23.200243959750
##  [301] -28.554452269695 -20.185647149432 -20.451110864690 -16.512876533623
##  [305]  -6.173173466631   0.318997103495  -8.689250264039  -6.228082661617
##  [309] -28.876950820172  -4.797805969758  -1.408468385927 -18.024886550899
##  [313] -14.919605165922 -12.109543179260 -14.636766088522  -7.005204327938
##  [317] -16.054738266747 -14.390121348194 -17.207536301341  -2.191949696479
##  [321] -22.222842757760  -1.088059635813  -7.460771571145 -10.658164266599
##  [325]  -7.985339676420  14.777005138454  -5.282472052887   3.261980827443
##  [329]  -3.314015149168 -11.373944133316 -24.111716962865   2.003631048163
##  [333]  -7.009098921408  -5.437731174767  -5.619215558295 -12.150211380980
##  [337] -13.121224760561 -16.926304190749 -33.962296833130 -25.842749336479
##  [341]   7.587482333584 -33.628867886535  -6.068279323935  -4.340621353874
##  [345] -20.439903705227   0.231132926629 -15.126640140205  -9.522633251936
##  [349]   3.928769971462 -14.324991259086 -11.351678361129   2.977351500845
##  [353] -10.707264837276 -14.208281940043   1.413403573723 -14.271264842364
##  [357]   3.956696064749 -12.961871694582  -4.173403894577 -36.762293512110
##  [361] -24.006790094345  23.041511108123  -1.432224586783   1.610164452066
##  [365]  -7.210631142070 -10.135484957350 -36.763788868569  -8.786855455726
##  [369] -15.005432420935   3.796779109849 -14.059866580059   3.959319445350
##  [373]  -2.614038445377  -2.633858472310 -17.330879536366 -28.983960498859
##  [377]  -5.541206986133  -5.437391378403 -25.261972081597 -10.805826170075
##  [381] -13.400106064521  -9.463192012457 -16.916348490356 -13.034948953615
##  [385] -21.950140483833   0.197359156169 -14.784442219738 -16.594920494864
##  [389]  -3.211203472065 -11.053553274145 -14.537176724463   3.414386316784
##  [393]  -0.140427991845 -21.954647751412 -15.748585732445  -0.180086378075
##  [397] -26.573832250238 -21.196469323143 -21.708244066000 -21.390516111211
##  [401] -22.848286453580 -11.417817333519   4.387847713793 -20.400575406747
##  [405] -20.238379514517   0.947737875709 -30.139268834093 -20.465767979997
##  [409]  -1.665900486554  -3.913975481701   0.840573311601  -8.272895816793
##  [413]   5.161919912294 -12.172084561912 -11.082507409247 -13.577165476729
##  [417]  -6.729556184252 -30.129442188482 -11.496945497389 -19.321912284228
##  [421]  -7.215869393463  -1.831637154738 -14.302602642177  -4.276699610898
##  [425]  -8.230062465124   1.135717353924 -15.871049111602 -27.382333228142
##  [429]  -3.463285714810 -11.866564166261  -3.606432172174 -10.318413309541
##  [433]   0.022358498836 -16.410666255411  -7.489629933536  -4.249645592234
##  [437] -16.350221968473  -1.427164997009   1.048824071010 -34.052252801066
##  [441] -17.009649823338  -9.715506478390   7.174920332269 -21.282948595037
##  [445]   0.862461279025   3.058849542507 -19.882646093504 -12.274179105851
##  [449] -11.421033329448  -6.039870401523 -29.394393756475 -43.207822050814
##  [453]  -0.579181084237  -9.628486052012 -12.585550133381  -6.556058591061
##  [457]  -8.506447698257  -0.595362664533 -32.179019192310 -27.935587562962
##  [461]   6.921068899601 -26.724961467969 -27.371928392182 -10.266348570697
##  [465]  -9.341955777309 -24.623234361304 -14.194047005734  -4.231529866930
##  [469] -16.069223553350 -10.692195362732   3.339438409048 -17.677420622503
##  [473]  -4.468053316625 -27.086578997415 -15.267281026119  10.637642335909
##  [477]  -5.762717404801   0.085557278430   3.611417957445 -11.059548633205
##  [481]  -9.584993047709   1.198977184881  -7.608371302783 -20.390839362081
##  [485] -14.917557087827  -6.158103751256 -11.474722653617 -16.504751789175
##  [489]  -9.436961060204 -12.466012235181   2.362215635282 -11.783268966881
##  [493]   2.364695793744 -20.821668313353 -14.941902007095 -27.111130344264
##  [497]  -9.599419516029 -15.611434836846 -35.573620576207 -16.967788106413
##  [501] -24.462874550306  -6.841442382267 -13.427475139157 -29.313530994357
##  [505]  -7.571789957358 -13.627679488290  14.327289003174  -4.079088092275
##  [509] -15.762007682275  -5.933717598314 -10.452546344912  -5.590999754979
```

```
##  [513]  12.353762019620 -15.861548439781  -4.006397506810   2.742331465481
##  [517] -19.117711542019 -23.754590566489 -12.877268633888  -9.185165893956
##  [521]  -6.621046860610  -4.338624610813 -19.342229343527   7.267698138739
##  [525]   5.883149264745 -20.374466078584  -6.304925802636  -5.613418799144
##  [529] -18.779960424210  -2.390823716551  -9.357180154628  12.984790421730
##  [533] -22.999779810670 -20.796372313229   4.883234655508 -12.307636320571
##  [537]  -3.349261493007 -21.603170872473   0.815408563127  -5.757881041441
##  [541]  -2.239436739579 -17.227887179474  -4.909245895558  -8.453850253616
##  [545]  -6.907527606549  -6.834863069678  -2.923201040374 -14.624353257849
##  [549]   6.983027470566 -18.619855613729 -20.518296436501  -4.684813811389
##  [553]  -9.227160065449 -20.248410877937  -8.558407109712 -12.208962301642
##  [557] -19.773314955226  -6.647197689767   5.873666216841   2.698258739956
##  [561]  18.873330102868  -3.167610858302 -16.769760875148 -11.638704332462
##  [565]  -2.903883874604 -20.536691385134 -19.562285020279 -19.451873742158
##  [569]  -7.752115526547  -6.405448256204 -16.637104246803 -11.265081803606
##  [573]   6.423742433085 -15.070984815119 -15.184870443385   3.436530046102
##  [577]  -9.125129840263  -4.623272630145   4.921863761876  -1.232073129451
##  [581]  -5.041147620941 -12.473539199190 -17.287776086754  -1.258064369647
##  [585] -17.606030396582  -5.970804609329 -15.317419807333  -9.999880665156
##  [589]  -1.818216185050  -7.308807898548 -13.880011417867 -13.247585096546
##  [593]   6.800087290503 -38.807355584066  -3.778618889344 -32.035523322016
##  [597] -12.726917703308 -19.221195390492 -13.062826697435  15.365592399809
##  [601]   9.342381375881 -10.125836361505 -22.518850965154  -2.252038371216
##  [605]  -1.961100367552 -24.987623524588  -4.141445090060 -18.406564623798
##  [609] -17.739843458122 -22.085508893000  -9.811756183891 -23.673322054288
##  [613] -29.901089219404 -31.561951318866  -9.328244198698  -1.397464518880
##  [617] -17.832884038661 -15.561804700282  -2.930492785951  -8.084320862666
##  [621]   6.396780603972 -11.755369812915 -13.624680981557  11.615820433097
##  [625] -11.300728538318 -23.129710914188  -3.402428132942  -9.081967859475
##  [629]  -5.519276972401  -4.559164103856   6.207774131267   3.212910272466
##  [633] -10.978417240900  -6.157664612307  -2.249581247460  -5.935518707308
##  [637] -10.267831338845   3.410596535409  -2.550907341645 -14.466688857099
##  [641] -17.368303461872 -18.505734093534 -23.400741081962  -3.417221474111
##  [645] -13.563549363721  -9.505582636261 -20.853411417150  -9.633471502453
##  [649] -36.236019786881 -31.473128478753   0.924836262883 -26.843423260054
##  [653] -15.064454176816 -22.900015547688 -16.603616970083  -8.888435603191
##  [657]   6.881817556572   4.419494931736  -7.921843134816 -15.353693679673
##  [661]  -7.477960337196 -19.684963057325  -9.480479978703   2.725829761188
##  [665] -14.675589113678 -15.359253624664  -1.684781425640  -6.449808166612
##  [669]  -5.479610333109  -1.204866223773 -11.084012068111 -34.381210115423
##  [673]  -2.644851019089 -24.684430194808  -6.228050568955  -3.354346158280
##  [677] -10.550644975736  -8.303793844082   1.991088429030  -1.703923546141
##  [681]   2.999316422263 -22.808785807808 -19.355994435346  -6.326223354293
##  [685] -10.103076310611 -15.902139962610  -0.747818608283 -10.939943133807
##  [689]  -0.210516524519 -18.986775172653  -7.626935773863 -17.078980926410
##  [693] -23.049974407800  -1.726232953787  -4.178560901289 -23.573058582004
##  [697]  -6.811324527688 -12.627420921639 -21.840093700084  -8.127472968077
##  [701]  -4.928578187675  -9.443201314665   3.796851660683  -1.093845335388
##  [705] -24.772999321683  10.090770078509  -0.948256087223  -3.702834201113
##  [709]   2.805868917696 -23.964175095700  -9.221216840106 -27.274428910290
##  [713]  -2.285950160965  -8.436253323170 -15.123788635184  -7.401477430515
##  [717]   2.016580026661  -5.646289558177  -6.762045350354   1.124682540694
##  [721]  13.875360211792  -2.617130166151 -19.008642971940  -2.420592127925
##  [725]   3.072572713432  -5.531365088870 -18.512294469875 -20.479472595309
```

```
##  [729]  -9.960062433201 -24.195842160941  -5.045079381439  -0.357051547192
##  [733]  -4.852648745943 -21.107099697507 -12.924673359992   9.240198814611
##  [737] -33.607136988653 -14.254397403426 -11.998128633356   0.812771578021
##  [741] -15.808502467252 -12.117392390328 -14.123577097369  -6.045372026096
##  [745] -23.371766688664  13.113178876591 -13.504910285134 -13.211033238054
##  [749] -14.289615923361 -12.296112600239 -10.559180156589 -13.720454408811
##  [753]  -5.257000303962 -10.617046325517 -31.276890675566 -20.022370580777
##  [757] -15.178045164381  12.102097472075  -6.750155634516 -22.768088614645
##  [761]  10.786263322692  -4.490811537766  10.104804684371 -12.875759015229
##  [765]  -5.184942352091 -23.785225113445  -0.770894625020 -18.196908184170
##  [769] -16.494398653286   4.891043753646 -19.412148440568 -21.740049454022
##  [773]  -2.068880680762 -17.960834823162  -0.105521861594  11.072029366323
##  [777]   8.273070109509  -6.072221032353 -10.956555212343  -2.596718394000
##  [781]   1.353106402527  -2.065664851924  -2.431226896817 -13.131748237440
##  [785] -34.927885950986 -10.750525965471 -19.273169540169   2.348211910283
##  [789]  -8.166864235781  -1.616340119403  -3.293714357919 -23.640604629644
##  [793]  -3.347265888798  -2.737773703685  -7.062575079029 -19.347789318479
##  [797]   2.484769605738 -26.897063574418   4.823772912475   0.912432707122
##  [801] -13.155180490017   8.707183890599  -7.232121408118 -16.323034444171
##  [805]  -0.816124648353  -7.893657175678 -29.142914011425 -12.921799523888
##  [809] -26.363686577091 -16.111953628886 -14.534528497332   3.575165898054
##  [813] -24.730808139676  -8.970823112848  -8.018063289910 -14.396025376181
##  [817] -17.442426335809  -2.580923562562  -5.247630361286  -7.567437537481
##  [821]  -7.352437985169   6.433485333713  -0.129884239945 -13.469055863666
##  [825] -11.479659486855 -12.742107532961   2.324012795348 -14.074934581559
##  [829] -20.916352011478  -4.142744103894  -0.288827523482  -3.162767909657
##  [833]   0.455373307975  -8.356610369199 -27.373205806200  -6.404672635681
##  [837]   1.040810896147  -2.849161451864 -16.828943776700  -2.079241589581
##  [841]   3.225729887339  10.055220312362   2.855116446469  -8.829425750487
##  [845]  -0.650341688082   3.330158446770  -0.184658577951  -3.438488364179
##  [849]   0.069589729773 -26.108297371440 -12.558933107310   9.081753260480
##  [853]  -2.984500504117 -16.856952369003 -13.376275871731 -20.213916299765
##  [857]   6.506287438423 -27.518968504512   8.034927560555 -23.976587650587
##  [861]   5.246792479618  12.449576484435  -5.502731968981  -8.518859695582
##  [865] -13.074763036183   3.758135174853   5.522821704782 -10.416805189630
##  [869] -19.989634192901   1.739191909605  -8.706205777127 -10.804995381767
##  [873]  -7.550064441722 -14.672756474754  -9.502262637938  20.922498527840
##  [877]   0.017852257745 -14.657394796131  -4.992519328497 -21.339497048390
##  [881] -22.971982446816 -18.914614119942 -16.575677281798 -12.684425788434
##  [885] -28.294749201132  -7.716197571993  -3.004702758495   2.866863109608
##  [889] -23.576061011541  -9.335186162349  -4.378440055093  -8.196009271002
##  [893] -28.842102405057  13.426585304189  -1.876222989011 -10.351342766012
##  [897] -18.868691963919  -5.309768035340   7.008740335064   6.712866195953
##  [901]  -8.225007433935 -15.037254206835  -8.949689592260 -21.791536707698
##  [905]  -1.631024812692  -7.388238034354 -19.809081661673  12.326762889991
##  [909] -15.851966436160 -19.064920698506 -18.964359415602 -15.289195276919
##  [913]  -7.408057563467 -12.203092254263   5.404225244923 -22.452496476916
##  [917]   2.535620565517 -12.433293230296 -16.227403021459 -20.141699367382
##  [921] -10.519150843030 -18.977110476583 -19.308737994439   2.517658922154
##  [925]   7.999633897174 -16.284043044961 -11.912881229706 -22.428087190895
##  [929] -11.937271390340  -5.705226492975  -8.565712740119 -12.549452666641
##  [933] -17.205138331166  -8.544916382586 -29.283177074879 -17.864687875171
##  [937]  -2.097864452843 -17.450033336766 -35.026115993680 -25.604590523748
##  [941]   5.268796341894   8.658880547211 -28.244468172721  -4.359755578285
```

```
##  [945] -13.616177827989 -15.843079583096  -3.909900791438  -4.722101718252
##  [949] -13.448838473814 -16.742192484055 -13.014487387267 -15.675218492798
##  [953] -25.128015926462   1.873220324264 -22.758556273965 -16.749254222418
##  [957] -14.222786007467 -22.569656151057 -17.988782295215   1.377032622586
##  [961]  -2.634736689856   5.852416662837   1.028381336585 -19.825917624721
##  [965]  -1.674698789021 -25.691685002019   3.176814875670   5.341506547902
##  [969] -10.241321442807 -16.096518698408  -8.883639047669  -6.173186060105
##  [973] -19.507196965928   2.875467340177  -1.419430742749  -0.172746378397
##  [977] -13.153690515729  -9.675387086743  -2.089631947452  -6.605033688577
##  [981]  -0.553953229636  -4.312072453225  -7.415832076048 -12.280701590417
##  [985]  -6.896104877402 -26.078787727753 -18.974102470302  -1.443043727644
##  [989] -29.620625688288  -0.512790349649  -8.959460034445 -20.442572814595
##  [993]  -2.904799096818  -8.247066734808 -20.584697205160 -16.806536472064
##  [997] -20.146769499865  13.366152539103 -21.714500261255 -31.414276596459
```

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

```
mean_i = mean(i,sqrt(1000))
```

- 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?

```
#qnorm(i, mean=mean_i, sd=.05, lower.tail = TRUE, log.i = TRUE )
```

- 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
```