Lab 1

Tziporah Horowitz

Due: 11:59PM February 8, 2020

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(digits = 10)
pi
```

```
## [1] 3.141592654
```

- Sum up the first 100 terms of the series $1 + 1/2 + 1/4 + 1/8 + \ldots$

```
options(scipen = 999)
sum(1/(2^(0:99)))
```

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

- Find the product of the first 20 terms of `1/3 * 1/6 * 1/9 * ...`

```
# faster
prod(1/seq(3, 60, 3))
```

```
## [1] 0.00000000000000000000000000001178827582
```

```
# slower
prod(1/(3*(1:20)))
```

```
## [1] 0.00000000000000000000000000001178827582
```

- Find the product of the first 500 terms of `1 * 1/2 * 1/4 * 1/8 * ...`

```
prod(1/(2^(0:499)))
```

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

Is this answer *exactly* correct?

No, because of overflow error.

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

```
sum(-499:0*log(2))
```

```
## [1] -86470.11077
```

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

```
x <- c(Inf, seq(20, -20, -2))
x
```

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

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

```
x <- c(logb(Inf, 3), logb(seq(100, -20, -2), 3))
```

```
## Warning in logb(seq(100, -20, -2), 3): NaNs produced
```

```
x
```

```
##  [1]          Inf 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536         -Inf          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

Comment on the appropriateness of the non-numeric values. You can't take log() of a negative number, log(Inf) = Inf, and log(0) = -Inf.

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

```
pos_real <- (x > 0) & (x != Inf) & (!is.nan(x))
```

- Locate the indices of the non-numbers in this vector. Hint: use the `which` function.

```
which(!pos_real)
```

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

- Locate the indices of the infinite quantities in this vector. Hint: use the `which` function.

```
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.

```
y <- x
y[is.infinite(y)] <- NA
c(which.min(y), which.max(y))
```

```
## [1] 51  2
```

- 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?

```
factor(x)
```

```
##  [1] Inf               4.19180654857877  4.1734172518943
##  [4] 4.15464876785729  4.13548512895119  4.11590933734319
##  [7] 4.09590327428938  4.07544759935851  4.05452163806914
## [10] 4.03310325630434  4.01116871959141  3.98869253500376
## [13] 3.96564727304425  3.94200336638929  3.91772888178973
## [16] 3.89278926071437  3.86714702345081  3.84076143030548
## [19] 3.81358809221559  3.78557852142874  3.75667961082847
## [22] 3.72683302786084  3.69597450568212  3.66403300987579
## [25] 3.63092975357146  3.59657702661571  3.56087679500731
## [28] 3.52371901428583  3.48497958377173  3.44451784578705
## [31] 3.40217350273288  3.3577627814323   3.31107361281783
## [34] 3.26185950714291  3.20983167673402  3.15464876785729
## [37] 3.09590327428938  3.03310325630434  2.96564727304425
## [40] 2.89278926071437  2.8135880922156   2.72683302786084
## [43] 2.63092975357146  2.52371901428583  2.40217350273288
## [46] 2.26185950714291  2.09590327428938  1.89278926071437
## [49] 1.63092975357146  1.26185950714291  0.630929753571457
## [52] -Inf              NaN               NaN
## [55] NaN               NaN               NaN
## [58] 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?

```r
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
## [24]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  2  2  2  2  2  2  2  2
## [47]  2  1  1  1  0 NA NA NA NA NA NA NA NA NA NA NA
```

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

```r
y <- x[is.finite(x)]
na.omit(y)
```

```
##  [1] 4.1918065486 4.1734172519 4.1546487679 4.1354851290 4.1159093373
##  [6] 4.0959032743 4.0754475994 4.0545216381 4.0331032563 4.0111687196
## [11] 3.9886925350 3.9656472730 3.9420033664 3.9177288818 3.8927892607
## [16] 3.8671470235 3.8407614303 3.8135880922 3.7855785214 3.7566796108
## [21] 3.7268330279 3.6959745057 3.6640330099 3.6309297536 3.5965770266
## [26] 3.5608767950 3.5237190143 3.4849795838 3.4445178458 3.4021735027
## [31] 3.3577627814 3.3110736128 3.2618595071 3.2098316767 3.1546487679
## [36] 3.0959032743 3.0331032563 2.9656472730 2.8927892607 2.8135880922
## [41] 2.7268330279 2.6309297536 2.5237190143 2.4021735027 2.2618595071
## [46] 2.0959032743 1.8927892607 1.6309297536 1.2618595071 0.6309297536
```

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

```r
sum(seq(0, 1 - 1e-6, 1e-6)^2) * 1e-6
```

```
## [1] 0.3333328333
```

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

```r
zero_one <- c(0, 1)
mean(sample(zero_one, 100, replace = TRUE))
```

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

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

```r
ones9 <- rep(1, 9)
mean(sample(c(0, ones9) , 500, replace = TRUE))
```

```
## [1] 0.898
```

```r
# better way
mean(sample(zero_one , 500, replace = TRUE, prob = c(.1, .9)))
```

```
## [1] 0.916
```

- 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.

```
lvls <- c("none", "infraction", "misdimeanor", "felony")
x_3 <- factor(sample(lvls, 100, replace = TRUE),
              levels = c("none", "infraction", "misdimeanor", "felony"),
              ordered = FALSE)
x_3
```

```
##   [1] felony      felony      none        none        none
##   [6] misdimeanor infraction  none        none        felony
##  [11] infraction  none        none        none        infraction
##  [16] infraction  felony      felony      misdimeanor infraction
##  [21] felony      infraction  misdimeanor misdimeanor misdimeanor
##  [26] none        misdimeanor infraction  none        felony
##  [31] misdimeanor none        infraction  none        misdimeanor
##  [36] felony      misdimeanor felony      misdimeanor infraction
##  [41] felony      misdimeanor misdimeanor infraction  felony
##  [46] misdimeanor felony      felony      none        misdimeanor
##  [51] felony      none        misdimeanor misdimeanor felony
##  [56] none        misdimeanor felony      infraction  infraction
##  [61] infraction  misdimeanor felony      none        misdimeanor
##  [66] infraction  felony      infraction  none        misdimeanor
##  [71] misdimeanor felony      felony      infraction  none
##  [76] infraction  infraction  none        misdimeanor infraction
##  [81] infraction  misdimeanor infraction  felony      felony
##  [86] infraction  misdimeanor infraction  infraction  none
##  [91] infraction  infraction  misdimeanor felony      misdimeanor
##  [96] infraction  infraction  felony      none        felony
## Levels: none infraction misdimeanor felony
```

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

```
x_3_bin <- ifelse(x_3 == "none", 0, 1)
x_3_bin
```

```
##   [1] 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 1
##  [71] 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1
```

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

```
x_3_ord <- factor(x_3, ordered = TRUE)
x_3_ord
```

```
##   [1] felony      felony      none        none        none
##   [6] misdimeanor infraction  none        none        felony
##  [11] infraction  none        none        none        infraction
```

```
##  [16] infraction  felony      felony      misdimeanor infraction
##  [21] felony      infraction  misdimeanor misdimeanor misdimeanor
##  [26] none        misdimeanor infraction  none        felony
##  [31] misdimeanor none        infraction  none        misdimeanor
##  [36] felony      misdimeanor felony      misdimeanor infraction
##  [41] felony      misdimeanor misdimeanor infraction  felony
##  [46] misdimeanor felony      felony      none        misdimeanor
##  [51] felony      none        misdimeanor misdimeanor felony
##  [56] none        misdimeanor felony      infraction  infraction
##  [61] infraction  misdimeanor felony      none        misdimeanor
##  [66] infraction  felony      infraction  none        misdimeanor
##  [71] misdimeanor felony      felony      infraction  none
##  [76] infraction  infraction  none        misdimeanor infraction
##  [81] infraction  misdimeanor infraction  felony      felony
##  [86] infraction  misdimeanor infraction  infraction  none
##  [91] infraction  infraction  misdimeanor felony      misdimeanor
##  [96] infraction  infraction  felony      none        felony
## Levels: none < infraction < misdimeanor < felony
```