GR5065 Homework 2

Ben Goodrich

Due February 9, 2021 at 8PM New York Time

```
# call the set.seed function once here to make the knitting conditionally deterministic
```

1 Equilibrium Climate Sensitivity

Read the short, fairly introductory, basically Bayesian article entitled "Prospects for narrowing bounds on Earth's equilibrium climate sensitivity" and the appendices particularly S5. The equilibrium climate sensitivity (ECS) is the long-run increase in temperature due to a "forced" doubling the amount of carbon dioxide in the atmosphere. There are many aspects of climate change that are relevant to the social sciences, but unfortunately what happens at the top of the atmosphere is more physics and chemistry. However, it is not easy to write scientifically relevant homework problems that have only one unknown, which this one does. Thus, immediately ask on CampusWire (it can be public) if you do not understand what any of the relevant scientific words and phrases in the article mean. Also, degrees is expressed in Celsius or Kelvin units rather than Farenheit.

1.1 Drawing from the prior

In equation S3 on page 13 of the appendices, the authors state

> The uncertainty in the forcing, F, and feedback, λ, used to generate the prior [for the Equilibrium Climate Sensitivity] are large, resulting in a prior having the form
>
> $$\text{ECS} = -\frac{F + \sigma_f}{\lambda + \sigma_\lambda}$$
>
> where σ_f is a gaussian distributed random variable with standard deviation $0.2F$. Likewise, σ_λ is also a gaussian distributed random variable with standard deviation 0.5λ. For F and λ we adopt 3.7 and -1.6 … respectively.

Note that λ is negative so the authors meant to say that their beliefs about σ_λ are distributed Gaussian with expectation zero and standard deviation $|0.5\lambda| > 0$. Also, note that using σ for a random variable rather than a standard deviation is weird notation, although it is common in physics, and that σ_f is independent of σ_λ.

Utilize the `rnorm` function in R to draw one million realizations of the numerator and the denominator whose ratio is this implied prior on the ECS over $\Theta = \mathbb{R}$.

1.2 Truncating the prior

In the next sentence, the authors state (albeit after the first version of the article was originally published) that their *actual* prior was only over the parameter space $\Theta = [0, 10]$ as negative values for ECS are impossible and very large values are implausible. Select the realizations in the previous subproblem that satisfy this condition. What proportion of the original one million realizations satisfy this condition?

1.3 Describing the truncated prior

Use the realizations from the $[0, 10]$ interval to reproduce in R each column of the line marked "Prior" in table S3 on page 30 of the appendices, where χ is the ECS.

1.4 PDF of a Ratio of Normals

Let the continuous random variable X be defined on a parameter (or sample) space $\Theta_X = \mathbb{R}$ and is normally distributed with expectation μ_X and standard deviation σ_X. Similarly, let the continuous random variable Y be defined on a parameter (or sample) spaces that is also $\Theta_Y = \mathbb{R}$ and is also normally distributed with expectation μ_Y and standard deviation σ_Y. Then, if X and Y are uncorrelated, the Probability Density Function (PDF) of the random variable $Z = \frac{X}{Y}$ over $\Theta_Z = \mathbb{R}$ is given in Wikipedia where $\Phi(\cdot)$ is the standard normal CDF, which is the `pnorm` function in R, and $\pi \approx 3.14159\ldots$, which is given to the maximum number of decimal places by the symbol `pi` in R. This PDF for Z would apply to the (untruncated) distribution of the ratio

$$\chi = -\frac{F + \sigma_f}{\lambda + \sigma_\lambda}$$

Create a R function with this signature

```
dratio <- function(z, mu_X = -3.7, mu_Y = -1.6,
                   sigma_X = 0.2 * 3.7, sigma_Y = 0.5 * 1.6) {
  # implement
}
```

that implements the PDF for the ratio of two uncorrelated Gaussian random variables and verify that your `dratio` function is a valid PDF. Your `dratio` function should return a vector if `z` is a vector, but that should not require any special effort because all the necessary subfunctions return vectors if given vector inputs.

1.5 Describing the truncated prior, part II

Utilize the `dratio` function somehow to reproduce the probability that $0 \leq \chi \leq 10$ if it were untruncated.

1.6 The likelihood function components

The function that the authors use for the j-th component of the likelihood is given in equation S4 on page 13 of the appendices as

$$P\left(\epsilon_j \mid \chi\right) = \frac{\left(1 - 2\epsilon_j\right) \operatorname{erf}\left(2\chi - 2\chi_j\right) + 1}{2}$$

with $0 \leq \epsilon_j \leq \frac{1}{2}$ but this is written in a very confusing fashion. Using the notation from the lectures, pretend that it (equivalently) said

$$f\left(e_j \mid \chi, c_j\right) = \frac{\left(1 - 2e_j\right)\left(2\Phi\left(2\left(\chi - c_j\right)\sqrt{2}\right) - 1\right) + 1}{2}$$

where I have used e_j instead of ϵ_j for the probability of observing evidence against the proposition that the ECS is less than the hypothesized value, c_j (which I have instead of χ_j).

It is atypical for a likelihood function to be constructed based on the authors' assessment of a probability, but this does not pose a mechanical problem for Bayes' Rule, although it does open up the possibility that other researchers would assess the evidence differently, which Fisher would say debases the entire scientific enterprise. In any event, the necessary values for the likelihood function are provided in Table S2 on page 28 of the appendices. Write these as

```
e <- c(Low_ii = .25, Low_iii = .35, Low_iv = .2, High_i = .75, High_ii = .65, High_iii = 0.6)
c <- c(Low_ii = 1.5, Low_iii = 1.5, Low_iv =  2, High_i = 4.0, High_ii = 4.5, High_iii = 4.5)
```

Then in R, write a function whose signature is

```
likelihood <- function(chi, e_j, c_j) {
  # implement
}
```

that returns a vector whose size is the same as χ that returns the probability of "observing" evidence e_j under the hypothesized value of the ECS given by c_j.

Then call

```
curve(likelihood(chi, e[1], c[1]), from = 0, to = 7,
      col = "black", ylim = 0:1, xname = "chi", lty = "solid")
```

to start to recreate Figure 2a on page 518 of the main text of the paper. You can then call the `curve` function again with the additional argument `add = TRUE` in order to add more lines to an existing plot and reproduce the essence of the entire Figure 2a.

1.7 Posterior PDF

Combine the results of the previous subproblems to reproduce the blue curve in Figure 2b on page 518 of the main text of the paper. You will first need to write a R function that inputs a vector `chi` and outputs a vector of the same size with the numerator of Bayes Rule (assuming conditionally independent likelihood contributions), whose signature is

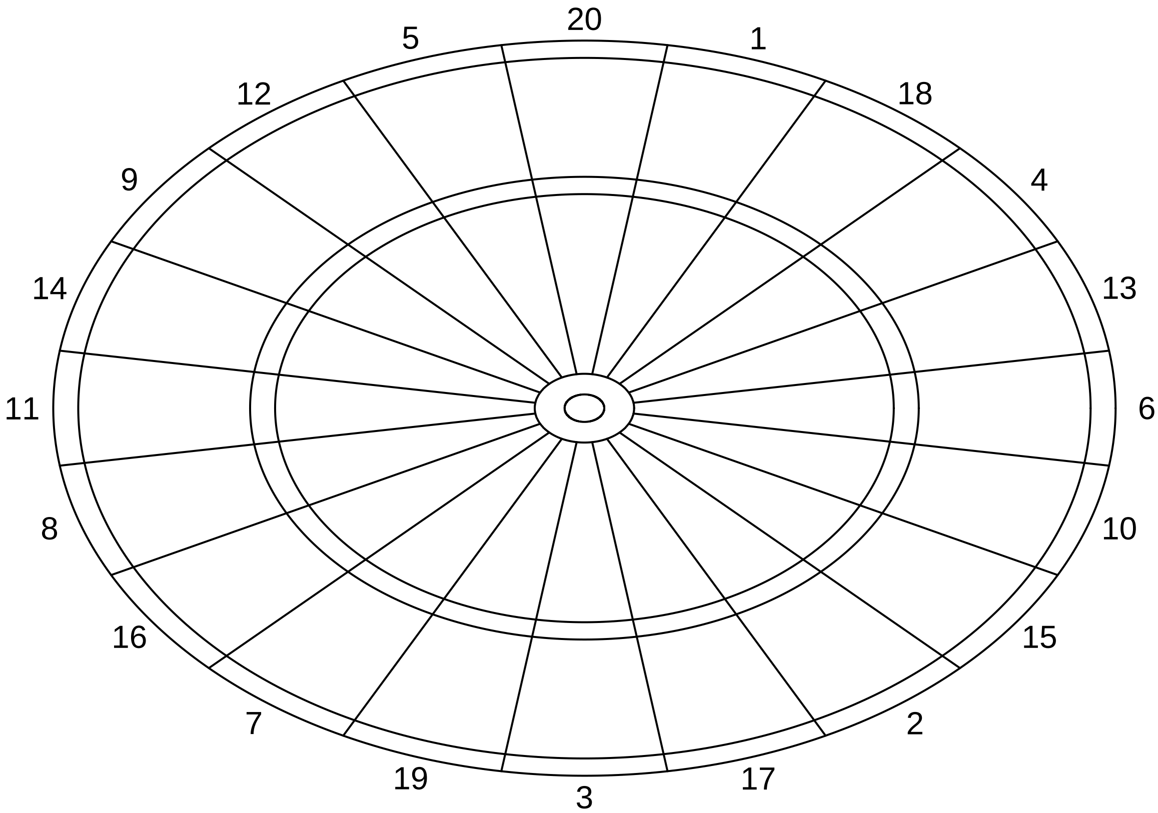
```
numerator <- function(chi, e = e, c = c) {
  # implement
}
```

Remember that the authors' prior on χ is truncated to the $[0, 10]$ interval. Then, reproduce the posterior PDF calculations on the line marked "Posterior P" in table S3 on page 30 of the appendices.

2 Darts

Read this article by Ryan (not Rob) Tibshirani on the optimal strategy for playing darts. Then install (once) Tibshirani's darts R package from CRAN outside your .Rmd file, which can produce the dartboard in the figure on the next page.

```
darts::drawBoard(new = TRUE)
```



You could watch a short YouTube video about throwing darts, although it is not at all necessary to answer these problems.

2.1 Drawing from a bivariate normal distribution

Tibshirani's simplest model for darts presumes that the player is aiming for the exact center of the dartboard, i.e. at $x = 0$ and $y = 0$, but the realization of the dart's location is bivariate normal with standard deviations σ_X, σ_Y, and perhaps correlation ρ.

Suppose that for Tibshirani $\sigma_X = 42.67$, $\sigma_Y = 68.67$, and $\rho = -0.16$ while Tibshirani's roommate (Andy Price) has a $\sigma_X = 17.90$, $\sigma_Y = 39.13$, and $\rho = -0.22$.

In R, draw from the marginal normal distribution of the dart's x-coordinate 100 times for Tibshirani and then draw from the conditional normal distribution of the y-coordinate given the x-coordinate. Do the same for Price. Then use the `points` function to add these values to a previous `darts::drawBoard(new = TRUE)` using red for Tibshirani and green for Price.

2.2 Normal Prior Distributions

Now, think about yourself playing darts and aiming at the exact center of the dartboard. If your beliefs about your own σ_X, σ_Y, and ρ were each described by (appropriately truncated) univariate normal distributions, what would you use for their respective expectations and standard deviations?

Draw *one* realization each for your σ_X, σ_Y, and ρ and then use those realizations of the three parameters to then draw the x and y coordinates of 100 of your dart throws and add them in black to your plotted dartboard.

2.3 Scoring Function

When players throw darts at a circular dartboard, the score for each dart is a non-monotonic, discretized, and otherwise weird function of where the dart lands on the dartboard, whose measurements are as follows:

- Radius of innermost circle for the "double bullseye": 6.35 mm. If the dart lands in this region, it is worth 50 points.
- Radius of next circle for the "single bullseye": 15.9 mm. If the dart lands in this region, it is worth 25 points.
- Radius of the next circle that starts the triple-score region: 99mm.
- Radius of the next circle that ends the triple-score region: 107mm If the dart lands in this region, it is worth triple the number of points indicated on the outside of the dartboard.
- Radius of the next circle that starts the double-score region: 162mm If the dart lands in this region, it is worth double the number of points indicated on the outside of the dartboard.
- Radius of outermost circle: 170 mm If the dart lands beyond this line, it is worth zero points.

If the dart lands in any other bed of the dartboard, it is worth the number of points indicated on the outside of the dartboard. A function to evaluate the score of a dart is given below, which first converts the x and y coordinates to a radius and an angle in polar coordinates and then assigns it a score based on the radius and angle.

```
score <- function(x, y) {
  stopifnot(is.numeric(x), length(x) == 1, is.numeric(y), length(y) == 1)

  # convert x and y in Cartesian coordinates to a radius and angle in polar coordinates
  # https://en.wikipedia.org/wiki/Polar_coordinate_system
  radius <- sqrt(x ^ 2 + y ^ 2)
  angle  <- atan2(y, x)

  if (radius > 170)   return(0)  # misses dartboard
  if (radius <= 6.35) return(50) # double bullseye
  if (radius <= 15.9) return(25) # single bullseye

  margin <- pi / 20
  interval <- margin * 2
  small <- pi / 2 - margin - 0:19 * interval
  large <- pi / 2 + margin - 0:19 * interval
  bed <- which(angle > small & angle <= large)
  if (length(bed) == 0) {
    angle <- angle - 2 * pi
    bed <- which(angle > small & angle <= large)
  }
  S <- darts:::getConstants()$S  # 20, 1, ..., 5
  score <- S[bed]
  if (dplyr::between(radius, 99, 107))       score <- 3 * score # in triple ring
  else if (dplyr::between(radius, 162, 170)) score <- 2 * score # in double ring
  return(score)
}
```

Use `mapply` or similar to calculate the scores of your 100 dart throws in the previous subproblem and make a barplot of the distribution of the scores.

2.4 Estimating the parameters

Tibshirani's darts package includes a function called `generalEM` that produces point estimates of the *variances* σ_X^2 and σ_Y^2, as well as the *covariance* $\rho\sigma_X\sigma_Y$, in its `$Sig.final` list element. Call the `darts::generalEM` function on your simulated scores from the previous subproblem. Are the implied point estimates similar to or dissimilar to the realized values of σ_X^2, σ_Y^2, and $\rho\sigma_X\sigma_Y$ that you drew from your prior distributions?

2.5 Expected scores

Tibshirani's darts package includes a function called `generalExpScores` that takes a vector of (estimates of) σ_X^2, σ_Y^2, and $\rho\sigma_X\sigma_Y$ and produces a 341×341 matrix of the *expected* score of a dart if you were to aim at each point on a 341×341 grid. The output of `generalExpScores` can then be passed to the `drawAimSpot` function to mark the place on your plotted dartboard where you should aim (because it maximizes the expected score).

Compute and plot these aim spots both for the *realizations* of σ_X^2, σ_Y^2, and $\rho\sigma_X\sigma_Y$ that you drew from your prior and then for the *estimates* of σ_X^2, σ_Y^2, and $\rho\sigma_X\sigma_Y$ that were produced by `darts::generalEM`. Are those two aim spots similar or dissimilar to each other?