Homework 2 Solutions

BEE 4850/5850

Due Date

Friday, 2/20/26, 9:00pm

💡 Tip

To do this assignment in Julia, you can find a Jupyter notebook with an appropriate environment in the homework's Github repository. Otherwise, you will be responsible for setting up an appropriate package environment in the language of your choosing. Make sure to include your name and NetID on your solution.

Load Environment

The following code loads the environment and makes sure all needed packages are installed. This should be at the start of most Julia scripts.

```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

The following packages are included in the environment (to help you find other similar packages in other languages). The code below loads these packages for use in the subsequent notebook (the desired functionality for each package is commented next to the package).

```
using Random # random number generation and seed-setting
using DataFrames # tabular data structure
using CSV # reads/writes .csv files
using Distributions # interface to work with probability distributions
using Plots # plotting library
using StatsBase # statistical quantities like mean, median, etc
using StatsPlots # some additional statistical plotting tools
using Optim # optimization tools
using LaTeXStrings # latex formatting for plot strings
```

Problems

Problem 1

Let's revisit the `chicago` dataset from HW1 (found in `data/chicago.csv`). We will look at the relationship of the potential predictor variables `pm25median` (the median density anomaly of smaller pollutant particles, in particles/m^3), `o3median` (the median concentration anomaly of O_3, in ppb), `so2median` (the median concentration anomaly of SO_2, in ppb), and `tmpd` (mean daily temperature, in degrees Fahrenheit) with the variable we would like to predict, `death` (the number of non-accidental deaths on that day).

Problem 1.1

As always, first, load the data:

```
chicago_dat = CSV.read("data/chicago.csv", DataFrame) # load data into
↪  DataFrame
```

	Column1	death	pm10median	pm25median	o3median	so2median	time	tmpd
	Int64	Int64	String15	String15	Float64	String15	Float64	Float64
1	1	130	-7.433544304	NA	-19.5923	1.9280425967	-2556.5	31.5
2	2	150	NA	NA	-19.0386	-0.985563116	-2555.5	33.0
3	3	101	-0.826530612	NA	-20.2173	-1.891416086	-2554.5	33.0
4	4	135	5.5664556962	NA	-19.6757	6.1393412739	-2553.5	29.0
5	5	126	NA	NA	-19.2173	2.2784648713	-2552.5	32.0
6	6	130	6.5664556962	NA	-17.634	9.8585839137	-2551.5	40.0
7	7	129	-0.433544304	NA	-15.3744	-5.818992059	-2550.5	34.5
8	8	109	-5.433544304	NA	-12.1705	-5.107941432	-2549.5	29.0
9	9	125	-0.571428571	NA	-20.0923	0.1822373332	-2548.5	26.5
10	10	153	NA	NA	-18.5803	-2.046929333	-2547.5	32.5
11	11	124	-19.4335443	NA	-5.71219	-1.60099878	-2546.5	29.5
12	12	111	-15.4335443	NA	-15.6289	2.937930625	-2545.5	34.5
13	13	104	11.566455696	NA	-17.0455	3.6418000592	-2544.5	34.0
14	14	118	1.5664556962	NA	-18.6289	6.1834667259	-2543.5	37.5
15	15	109	-7.256018217	NA	-18.7122	-1.141416086	-2542.5	32.5
16	16	125	-22.4335443	NA	-15.0039	-3.153813391	-2541.5	25.0
17	17	128	NA	NA	-17.3789	1.8072373332	-2540.5	27.0
18	18	141	-2.433544304	NA	-17.0039	0.5626919865	-2539.5	17.5
19	19	130	-9.433544304	NA	-7.634	-0.718613289	-2538.5	23.0
20	20	133	-3.433544304	NA	-18.4136	0.6816042276	-2537.5	20.5
21	21	115	-4.172413793	NA	-16.9933	3.6534648713	-2536.5	22.0
22	22	121	10.566455696	NA	-12.6289	-1.268914807	-2535.5	19.5
23	23	107	13.566455696	NA	-7.92567	-2.268914807	-2534.5	2.5
24	24	123	-3.433544304	NA	-12.259	-1.72474942	-2533.5	2.0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

Now we make the scatterplots. I'll make these into a single plot with different panels, but making these as separate plots is also fine. Putting these all on a single plot with different symbols might be a bit dense (and you have to worry about scales for the predictors).

```
p1 = @df chicago_dat scatter(:pm25median, :death, title="PM 2.5 Daily Median
 ↪  Anomaly", ylabel="Non-Accidental Daily Deaths",
 ↪  xlabel=L"particles/m$^3$", markersize=3, titlefontsize=8,
 ↪  guidefontsize=7, tickfontsize=5, legend=false) # <1>
p2 = @df chicago_dat scatter(:o3median, :death, title=L"O$_3$ Daily Median
 ↪  Anomaly", ylabel="Non-Accidental Daily Deaths", xlabel="ppb",
 ↪  markersize=3, titlefontsize=8, guidefontsize=7, tickfontsize=5,
 ↪  legend=false)                                                      ①
p3 = @df chicago_dat scatter(:so2median, :death, title=L"SO$_2$ Daily Median
 ↪  Anomaly", ylabel="Non-Accidental Daily Deaths", xlabel="ppb",
 ↪  markersize=3, titlefontsize=8, guidefontsize=7, tickfontsize=5,
 ↪  legend=false)
```

```julia
p4 = @df chicago_dat scatter(:tmpd, :death, title="Daily Mean Temperature",
 ↪  ylabel="Non-Accidental Daily Deaths", xlabel="°F", markersize=3,
 ↪  titlefontsize=8, guidefontsize=7, tickfontsize=5, legend=false)
plot(p1, p2, p3, p4, layout=(2, 2))                                          ②
```

① This uses the `DataFrame` plotting macro from `StatsPlots.jl` which reduces having to write `chicago_dat` for every variable.

② This arranges all of the plots as panels of a single plot with the specified layout.

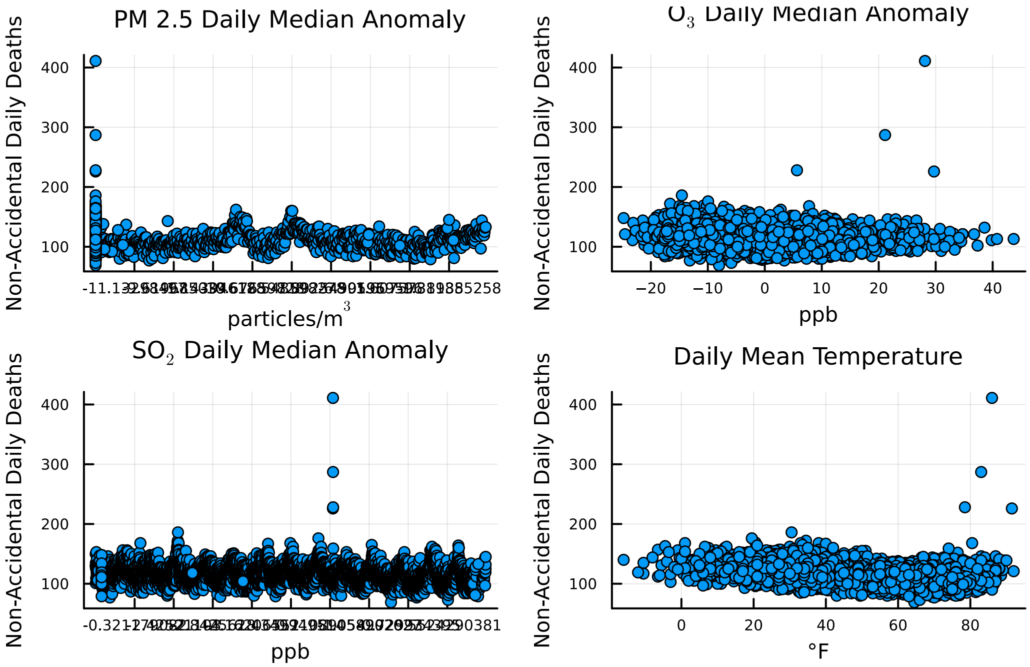

Figure 1: Scatterplots of bivariate relationships between non-accidental daily deaths in Chicago and environmental predictors.

We can see from Figure 1 that there are some large outliers which might be limiting our ability to see if there is an overall linear trend between any of these variables. Let's change the y-axis limits to let us see the bulk of the data more clearly.

```julia
yaxis!(p1, (50, 200))
yaxis!(p2, (50, 200))
yaxis!(p3, (50, 200))
yaxis!(p4, (50, 200))
plot(p1, p2, p3, p4, layout=(2, 2))
```

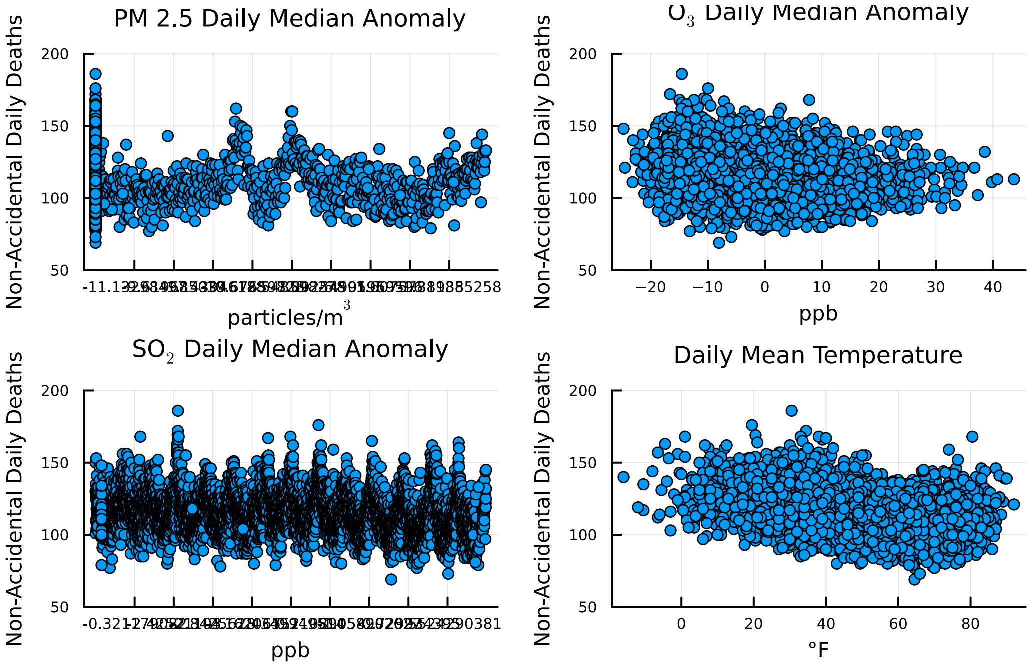

Figure 2: Rescaled scatterplots of bivariate relationships between non-accidental daily deaths in Chicago and environmental predictors.

It does not look like from either Figure 1 or Figure 2 that there is much of a linear relationship between the air pollution predictors and the number of deaths. However, temperature appears like a reasonable linear predictor, and visually it could be that the data has roughly similar variance until we get to the very large outliers at high temperatures.

Problem 1.2

Letting y be the deaths and x the mean temperature, this linear model is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2).$$

As a result, the likelihood is based on the probability model for the errors,

$$\varepsilon_i = y_i - (\beta_0 - \beta_1 x_i) \sim N(0, \sigma^2).$$

Equivalently, we could set this up as

$$y_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2);$$

both will result in the same result but the code might look a little different.

Now, let's code the log-likelihood function and optimize it.

```julia
function gauss_loglik(p, y, x)                                                    ①
    b0, b1, s = p
    pred = b0 .+ b1 * x # calculate predicted y
    ll = sum(logpdf.(Normal.(pred, s), y)) # compute log likelihood
    return ll
end

lb = [-1000.0, -100.0, 0.1]                                                       ②
ub = [1000.0, 100.0, 50.0]
p0 = [0.0, 0.0, 10.0]
optim_out = Optim.optimize(p -> -gauss_loglik(p, chicago_dat.death, chicago_dat.tmpd), lb, ub, p0)
p_mle = round.(optim_out.minimizer; digits=1) # round to a reasonable number of sig figs
```

① `p` is a vector of the model parameters `b0`, `b1`, `s`; often optimization packages want to optimize over a vector so here I unpack this vector within the function. This pattern also keeps the function definition a bit cleaner, particularly as the number of parameters increases. However, you can do the opposite, writing your function with individual parameters and "unpacking" into the optimization function.

② The lower bound is set to 0.1 as the standard deviation cannot be 0 or negative.

```
3-element Vector{Float64}:
 130.0
  -0.3
  14.2
```

How do we interpret the coefficients?

- $\hat{\beta}_0 = 130$ deaths suggests that, at a temperature of 0° F, we would expect 130 non-accidental deaths.
- $\hat{\beta}_1 = -0.3$ suggests that, on two days with a difference in mean temperature of 1° F, we would expect to see 0.3 fewer deaths on the warmer day. Note that this **does not mean** an increase of 1° F "causes" this reduction.

Problem 1.3

Add your fitted regression line and a 90% prediction interval to your scatterplot of `tmpd` and `death`. Using this and any other diagnostics, do you agree with the Gaussian-error linear model assumption? Why or why not?

```
# get regression line
b0, b1, s = p_mle
x_pred = -20:100
y_pred = b0 .+ b1 * x_pred
# simulate conditional distribution to plot prediction interval
err = rand(Normal(0, s), 10_000)                                              ①
err_quantile = quantile(err, [0.05, 0.95])

# make plot
p = @df chicago_dat scatter(:tmpd, :death, title="Daily Mean Temperature",
 ↪   xlabel="Non-Accidental Daily Deaths", ylabel="°F", alpha=0.4,
 ↪   label="Observations") # reduce alpha to make rest of plot easier to see
plot!(p, x_pred, y_pred, ribbon=(abs(err_quantile[1]), err_quantile[2]),
 ↪   fillalpha=0.3, color=:red, linewidth=2, label="Regression")              ②
```

① Since the error distribution is the same around the regression estimate, regardless of x, we can just simulate from a single distribution and add the quantiles back to the regression line to get the prediction interval. For some other cases (such as heteroskedastic errors, or other distributions), we might have to simulate at every prediction point.

② The `ribbon` keyword in `Plots.plot` can either take an array of values or a tuple of (positive) values if the same distance is to be used from the main plotted line.

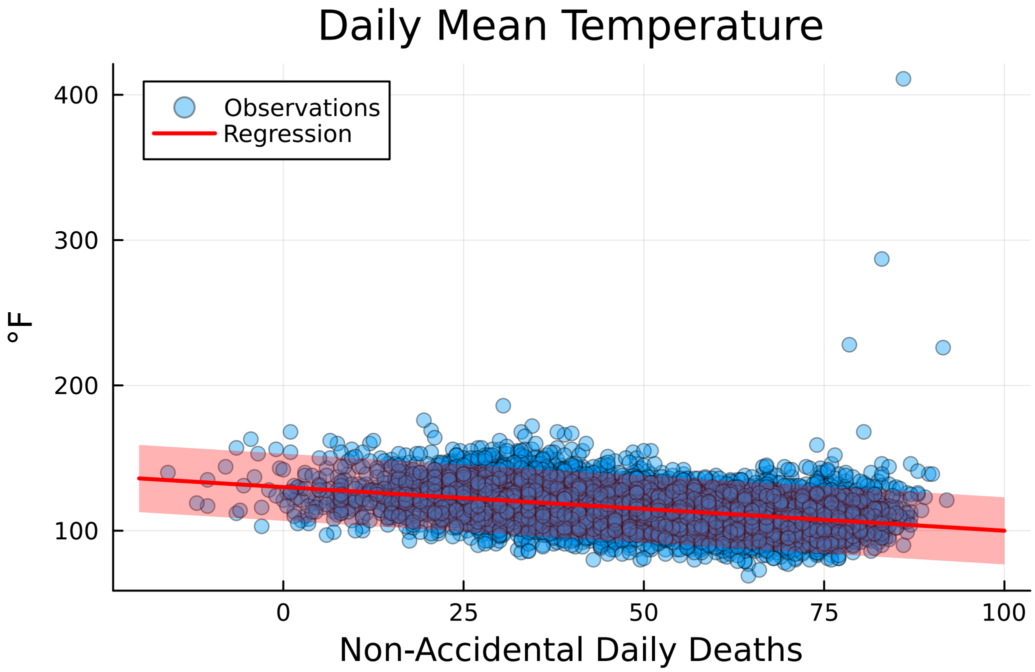

Figure 3: Fitted relationship between mean temperature and non-accidental deaths in Chicago. The shaded region is the 90% prediction interval from the linear regression model.

The general trend in Figure 3 looks reasonable. We can calculate the surprise index by finding the fraction of observations which are outside of the 90% prediction interval.

```
resids = chicago_dat.death - (b0 .+ b1 * chicago_dat.tmpd)
si = sum((resids .< err_quantile[1]) + (resids .> err_quantile[2])) /
 ↪  nrow(chicago_dat)                                                       ①
round(si; digits=2)
```

① The `sum` term here will add together two vectors, one of which is 1 when the residuals are below the lower bound of the interval, and the second which is 1 when they're above the upper bound.

```
0.08
```

The surprise index is about 8%, which is not ideal, but also isn't so far from the expected 10% to be indicative of a major problem. Let's look at the distribution of residuals and a Q-Q plot to see if the residuals appear normal.

```
resid_hist = histogram(resids, xlabel="Residuals", ylabel="Count",
 ↪  legend=false)
resid_qq = qqplot(resids, Normal(), xlabel="Theoretical Quantiles",
 ↪  ylabel="Empirical Quantiles")
p = plot(resid_hist, resid_qq, layout=(2, 1))
```

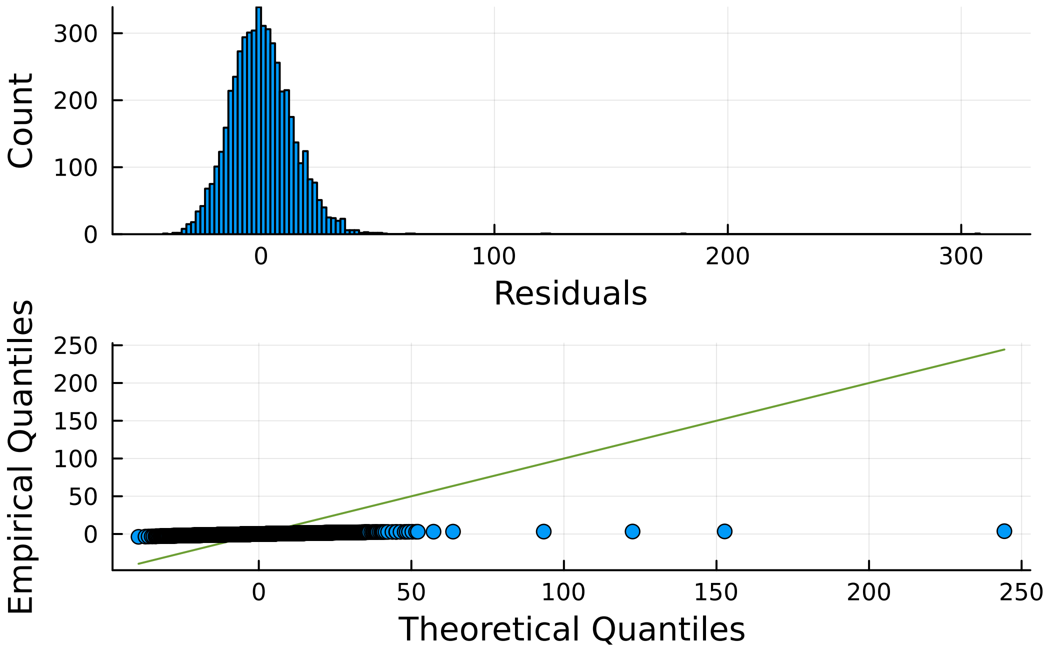

Figure 4: Diagnostics to check Gaussian assumptions for the residuals.

Figure 4 suggests that while the bulk of the residuals appear to follow a Gaussian distribution, there is a long upper tail which throws off the Q-Q plot. From Figure 3, these large outliers occur at very high temperatures; the simple regression model cannot account for this change in trend, which could be due to extreme heat exposure or some other factor which is correlated with high temperatures. We might trust this model for temperatures below 80° F, or we would want to rethink the model completely with a better representation of factors which contribute to deaths.

Problem 2 (6 points)

Problem 2.1

Denoting the occurrence of freezing in year i as y_i, the probability of this occurring as p_i, and the DJF temperature as T, the logistic regression model can be formulated as:

$$
\begin{aligned}
y_i &\sim \mathrm{Binomial}(p_i) \\
\mathrm{logit}(p_i) &= \beta_0 + \beta_1 T.
\end{aligned}
$$

Let's load the data and calculate the winter temperatures. I decided to write a function which would calculate the DJF mean for a given base year, which would allow me to loop over all of the years and apply the function; there are many other possible approaches one could take.

```
temp_dat = CSV.read("data/HadCRUT5.1Analysis_gl.txt", delim=" ",
 ↪  ignorerepeated=true, header=false, silencewarnings=true, DataFrame)      ①
temp_dat = temp_dat[1:2:nrow(temp_dat), :] # only keep even rows

function djf_mean(temps, year)
    idx = findfirst(temps[:, 1] .== year)                                    ②
    return mean([temps[idx - 1, 13], temps[idx, 2], temps[idx, 3]])          ③
end

djf_means = [djf_mean(temp_dat, yr) for yr in 1851:2025]                     ④
```

① We need `silencewarnings=true` because `CSV.read` will complain about the even rows not having the same number of columns.

② `findfirst` (or the equivalent) is the fastest solution since it will terminate after finding any matches, but if you used a different approach which involved searching over the entire `DataFrame`, it wouldn't matter too much in this case since the data is small.

③ `temps[idx - 1, 13]` is the previous December since the first column is the year, and similarly for that year's January and February.

④ This is a straightforward use of a comprehension to apply a function in a loop. Comprehensions are quite memory efficient. There are other strategies, such as mapping the function, that one could use.

```
175-element Vector{Float64}:
 -0.35166666666666674
 -0.3633333333333333
 -0.17966666666666667
 -0.37366666666666665
 -0.31233333333333335
 -0.2793333333333333
 -0.453
 -0.371
 -0.4036666666666667
 -0.393

  0.9269999999999999
```

```
0.7629999999999999
0.8213333333333334
1.0646666666666667
0.6546666666666666
0.7643333333333334
0.8046666666666668
1.2286666666666666
1.167
```

Now we want to create a vector of 1s and 0s for the freezing occurrences and non-occurrences (as a reminder, 1796 and 1816 are outside of the temperature data, so we will not include those). We'll put all of the data together in a `DataFrame` to make it easy to assign the relevant 1s.

```
freezes = [1856, 1875, 1884, 1904, 1912, 1934, 1961, 1979, 2015]
dat = DataFrame(year=1851:2025, temp=djf_means,
 ↪  freeze=zeros(length(djf_means)))
for yr in freezes
    idx = findfirst(dat.year .== yr)
    dat[idx, :freeze] = 1
end
```

Now let's maximize the likelihood of the model. We need to be able to compute the inverse logit of the linear part of the model, which is given by

$$\text{logit}^{-1}(x) = \frac{\exp(x)}{\exp(x) + 1}.$$

```
logit(p) = log(p / (1 - p))
invlogit(x) = exp(x) / (exp(x) + 1)
function freeze_loglik(params, dat)
    b0, b1 = params
    p = invlogit.(b0 .+ b1 * dat.temp)
    ll = sum(logpdf.(Bernoulli.(p), dat.freeze))
    return ll
end

lb = [-20.0, -20.0]
ub = [20.0, 20.0]
p0 = [0.25, 0.25]
optim_out = Optim.optimize(v -> -freeze_loglik(v, dat), lb, ub, p0)
v_mle = round.(optim_out.minimizer; digits=1)
```

```
2-element Vector{Float64}:
```

```
-3.0
-0.7
```

We can interpret these coefficients as follows:

- $\hat{\beta}_0 = -3$ gives us the probability of freezing when the temperature anomaly is the same as the reference period mean, namely $\text{logit}^{-1}(-3) = 4.7\%$.
- $\hat{\beta}_1 = -0.7$ gives the reduction in freezing log-odds associated with a temperature anomaly increase of 1° C. It's not entirely clear what this means, but one important feature is that this is negative, so that increased temperatures decrease the probability of freezing, which is what we would expect.

Problem 2.2

Let's plot how the probability of freezing changes with respect to temperature and time.

```
temp_pred = -1:0.1:2
p1 = plot(-1:0.1:2, invlogit.(v_mle[1] .+ v_mle[2] * temp_pred), xlabel="DJF
 ↪  Temperature Anomaly (°C)", ylabel="Probability Cayuga Lake Freezes",
 ↪  legend=false)
p2 = plot(dat.year, invlogit.(v_mle[1] .+ v_mle[2] * dat.temp),
 ↪  xlabel="Year", ylabel="Probability Cayuga Lake Freezes")
plot(p1, p2, layout=(1, 2), legend=false)
```

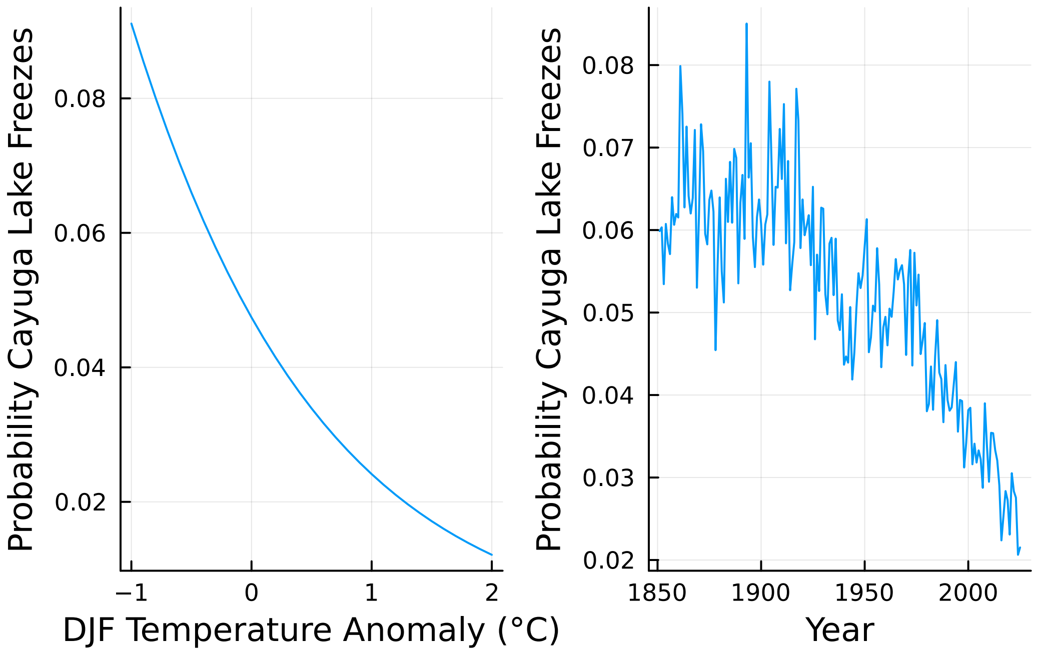

Figure 5: Modeled probability of Cayuga Lake freezing versus the DJF temperature anomaly and over time.

We can see from Figure 5 that the probability has declined precipitously over time as the temperature anomaly has increased, from around a 6-8% probability between 1851 and 1925 down to around 2% today.

How can we determine if this model is reasonable? Since the probabilities of occurrence are so low, we can't try to look at a raw mis-classification rate. However, we can try to assess how well calibrated the model is, *e.g.*, does it generally predict the right number of freezes. We have observed 9 years in which Cayuga Lake has frozen (that align with our temperature data); let's see if the expected number of freezes is similar to this by adding up the modeled probabilities.

```
p_fit = invlogit.(v_mle[1] .+ v_mle[2] * dat.temp)
sum(p_fit)
```

```
9.116926466107916
```

This is almost exactly right, though it is a bit generous to the model because it's an in-sample estimate. We can also look at how well the model captures probabilities of freezing at different ranges of predicted probabilities. Let's look at what happens between probabilities over 5% and under 5%.

```
sum(p_fit[p_fit .> 0.05])
```

```
6.511824596882027
```

```
sum(dat.freeze[p_fit .> 0.05])
```

```
6.0
```

That's off by a little, but not much.

Similarly, for probabilities below 5%:

```
sum(p_fit[p_fit .< 0.05])
```

```
2.605101869225891
```

```
sum(dat.freeze[p_fit .< 0.05])
```

```
3.0
```

Try out other ranges to see where the model performs well and where it doesn't: if you see ranges where the model substantially over or under predicts, that's a signal of mis-specification.

Beyond this, diagnosing logistic regression *post facto* can be difficult because it doesn't make distributional assumptions; you need to convince yourself that the log-odds ought to have a linear relationship with the predictor(s), which can be difficult.

Problem 2.3

To find the temperature required for less than a 1% probability of Cayuga Lake freezing, we need to find logit(0.01) and solve for the temperature based on the logistic regression.

```
freeze_thresh_prob = logit(0.01)
freeze_thresh_temp = (freeze_thresh_prob - v_mle[1]) / v_mle[2]
```

```
2.278742643049414
```

So if the winter temperature anomaly rises above 2.2° C, we would expect the probability of Cayuga Lake freezing to drop below 1%.

Problem 3 (6 points)

The file `data/salamanders.csv` contains counts of salamanders from 47 different plots of the same area in California, as well as the percentage of ground cover and age of the forest in the plot. You would like to see if you can use these data to predict the salamander counts with a Poisson regression.

Problem 3.1

Loading the data (note the delimiter is now a semi-colon):

```
dat = CSV.read(joinpath("data", "salamanders.csv"), DataFrame, delim=";")
```

	SITE	SALAMAN	PCTCOVER	FORESTAGE
	Int64	Int64	Int64	Int64
1	1	13	85	316
2	2	11	86	88
3	3	11	90	548
4	4	9	88	64
5	5	8	89	43
6	6	7	83	368
7	7	6	83	200
8	8	6	91	71
9	9	5	88	42
10	10	5	90	551
11	11	4	87	675
12	12	3	83	217
13	13	3	87	212
14	14	3	89	398
15	15	3	92	357
16	16	3	93	478
17	17	2	2	5
18	18	2	87	30
19	19	2	93	551
20	20	1	7	3
21	21	1	16	15
22	22	1	19	31
23	23	1	29	10
24	24	1	34	49
...	...	...	...	...

The first model, using `PCTCOVER` as a predictor, can be specified as the following (using the standard log link function for Poisson regression):

$$S \sim \text{Poisson}(\lambda)$$
$$\log(\lambda) = aPC + b$$

As noted in the problem, we will need to standardize the `PCTCOVER` variable, as its range is much larger than the salamander counts.

```
function salamander_pcover(p, counts, pctcover)
    a, b = p
      = exp.(a * pctcover .+ b)
    ll = sum(logpdf.(Poisson.( ), counts))
    return ll
end

lb = [-10.0, -50.0]
ub = [10.0, 50.0]
p0 = [0.0, 0.0]

# function to make this more convenient
stdz(x) = (x .- mean(x)) / std(x)                                                  ①

result = optimize(p -> -salamander_pcover(p, dat.SALAMAN,
 ↪  stdz(dat.PCTCOVER)), lb, ub, p0)
pcover_mle = round.(result.minimizer; digits=1)
```

```
2-element Vector{Float64}:
 1.2
 0.4
```

Problem 3.2

To translate these estimates to expected values and uncertainty intervals, we now simulate observations of salamanders based on this model.

```
function sim_salamanders_pcover(params, pctcover, n)
    a, b = params
      = exp.(a * pctcover .+ b)
    sal_pred = zeros(n, length(pctcover))
    for i = 1:length(pctcover)
        sal_pred[:, i] = rand(Poisson( [i]), n)
    end
    return sal_pred
```

```
end

sim_pcover = sim_salamanders_pcover(pcover_mle, -2:0.01:2, 10_000)        ①

pcover_q = mapslices(col -> quantile(col, [0.05, 0.5, 0.95]), sim_pcover;
  ↪  dims=1)
plot(-2:0.01:2, pcover_q[2, :], linewidth=3, ribbon=(pcover_q[2, :] -
  ↪  pcover_q[1, :], pcover_q[3, :] - pcover_q[2, :]), fillalpha=0.5,
  ↪  xlabel="Standardized Percent Ground Cover", ylabel="Salamander Count",
  ↪  label="Simulations")
scatter!(stdz(dat.PCTCOVER), dat.SALAMAN, label="Observations")
```

① Note that our predictors are along the **standardized** predictor range, not the *raw* `PCTCOVER` range, since we standardized the inputs when we fit the regression.

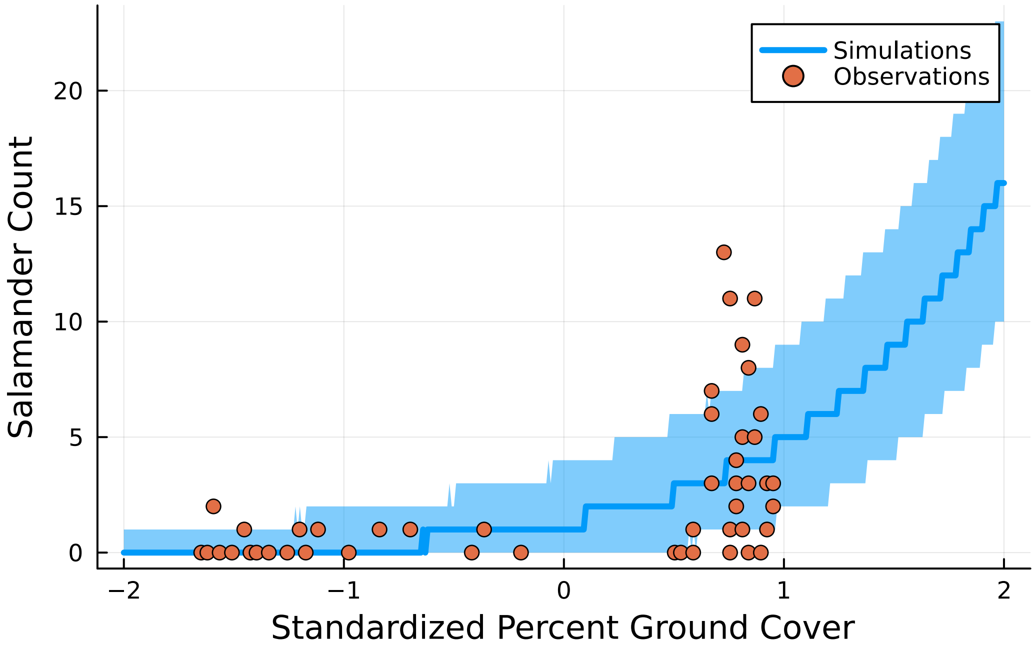

Figure 6: Predictive interval of salamander counts using the percent cover model.

The model seems to do ok for lower levels of percent cover, but does not account for the level of dispersion at higher levels; this might suggest that a Poisson model is not ideal, but we could try a negative binomial regression instead.

Problem 3.3

Let's see if we can improve the model using the FORESTAGE predictor. We'll try to add this in as a linear predictor.

```
function salamander_fage(p, counts, pctcover, forestage)
    a1, a2, b = p
      = exp.(a1 * pctcover + a2 * forestage .+ b)
    ll = sum(logpdf.(Poisson.( ), counts))
    return ll
end

lb = [-10.0, -10.0, -50.0]
ub = [10.0, 10.0, 50.0]
p0 = [0.0, 0.0, 0.0]

result = optimize(p -> -salamander_fage(p, dat.SALAMAN, stdz(dat.PCTCOVER),
 ↪  stdz(dat.FORESTAGE)), lb, ub, p0)
fage_mle = round.(result.minimizer; digits=1)
```

```
3-element Vector{Float64}:
  1.2
 -0.0
  0.4
```

We can see that the FORESTAGE predictor variable is effectively zero, which means it does not add much to the prediction. Why is this? The correlation between PCTCOVER and FORESTAGE is quite high: 0.63, which means most of the information in FORESTAGE has already been included in the PCTCOVER model. It makes sense that PCTCOVER is valuable: salamanders are likely to seek areas with high levels of ground cover, while the age of the forest is only meaningful to the extent that older forests may have more ground cover (hence the high correlation).

Problem 4 (7 points)

Problem 4.1

Loading the data (once again, the file is delimited by semi-colons):

```
dat = CSV.read(joinpath("data", "Hurricanes.csv"), DataFrame, delim=";")
```

	name	year	deaths	category	min_pressure	damage_norm	female	femininity
	String15	Int64	Int64	Int64	Int64	Int64	Int64	Float64
1	Easy	1950	2	3	960	1590	1	6.77778
2	King	1950	4	3	955	5350	0	1.38889
3	Able	1952	3	1	985	150	0	3.83333
4	Barbara	1953	1	1	987	58	1	9.83333
5	Florence	1953	0	1	985	15	1	8.33333
6	Carol	1954	60	3	960	19321	1	8.11111
7	Edna	1954	20	3	954	3230	1	8.55556
8	Hazel	1954	20	4	938	24260	1	9.44444
9	Connie	1955	0	3	962	2030	1	8.5
10	Diane	1955	200	1	987	14730	1	9.88889
11	Ione	1955	7	3	960	6200	0	5.94444
12	Flossy	1956	15	2	975	1540	1	7.0
13	Helene	1958	1	3	946	540	1	9.88889
14	Debra	1959	0	1	984	430	1	9.88889
15	Gracie	1959	22	3	950	510	1	9.77778
16	Donna	1960	50	4	930	53270	1	9.27778
17	Ethel	1960	0	1	981	35	1	8.72222
18	Carla	1961	46	4	931	15850	1	9.5
19	Cindy	1963	3	1	996	300	1	9.94444
20	Cleo	1964	3	2	968	6450	1	7.94444
21	Dora	1964	5	2	966	16260	1	9.33333
22	Hilda	1964	37	3	950	2770	1	8.83333
23	Isbell	1964	3	2	974	800	1	9.44444
24	Betsy	1965	75	3	948	20000	1	8.33333
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

Our interpretation of the hypothesis results in the following model specification:

$$D \sim \text{Poisson}(\lambda)$$
$$\log(\lambda) = a_1 FN + a_2 MP + b$$

We need to standardized the pressure predictor.

```julia
function hurricane_pressure(p, counts, fem, pressure)
    a1, a2, b = p
    m = exp.(a1 * fem + a2 * pressure .+ b)
    ll = sum(logpdf.(Poisson.(m), counts))
    return ll
end

lb = [-10.0, -10.0, -50.0]
```

```
ub = [10.0, 10.0, 50.0]
p0 = [5.0, 5.0, 0.0]

# function to make this more convenient
stdz(x) = (x .- mean(x)) / std(x)                                                          ①

result = optimize(p -> -hurricane_pressure(p, dat.deaths, dat.female,
 ↪  stdz(dat.min_pressure)), lb, ub, p0)
p_pressure = result.minimizer
```

```
3-element Vector{Float64}:
  0.47211173924356337
 -0.7195324699559839
  2.414667236694508
```

Problem 4.2

```
function sim_hurricanes(p, fem, damage,  n)
    a1, a2, b = p
     = exp.(a1 * fem + a2 * damage .+ b)
    hur_pred = zeros(n, length(fem))
    for i = 1:length(fem)
        hur_pred[:, i] = rand(Poisson( [i]), n)
    end
    return hur_pred
end

press_range = -3:0.01:2 # these are normalized pressures
sim_m = sim_hurricanes(p_pressure, repeat([-1], length(press_range)),
 ↪  press_range, 10_000)                                                                   ①
sim_f = sim_hurricanes(p_pressure, repeat([1], length(press_range)),
 ↪  press_range, 10_000)

simm_q = mapslices(col -> quantile(col, [0.05, 0.5, 0.95]), sim_m; dims=1)
simf_q = mapslices(col -> quantile(col, [0.05, 0.5, 0.95]), sim_f; dims=1)

std_pressure = stdz(dat.min_pressure)
plot(-3:0.01:2, simm_q[2, :], linewidth=3, ribbon=(simm_q[2, :] - simm_q[1,
 ↪  :], simm_q[3, :] - simm_q[2, :]), alpha=0.5, label="Masculine Storms
 ↪  (Simulation)", color=:blue, xlabel="Standardized Minimum Pressure",
 ↪  ylabel="Deaths")
plot!(-3:0.01:2, simf_q[2, :], linewidth=3, ribbon=(simf_q[2, :] - simf_q[1,
 ↪  :], simf_q[3, :] - simf_q[2, :]), alpha=0.5, label="Feminine Storms
 ↪  (Simulation)", color=:red)
```

```
scatter!(std_pressure[dat.female .== 0.0], dat.deaths[dat.female .== 0.0],
 ↪  color=:blue, label="Masculine Storms (Observations)")
scatter!(std_pressure[dat.female .== 1.0], dat.deaths[dat.female .== 1.0],
 ↪  color=:red, label="Feminine Storms (Observations)")
```

① Here we want to fix one of the predictors (`female`) to generate the relevant conditional simulations. The `repeat()` function lets us repeat the same values for the female predictor.

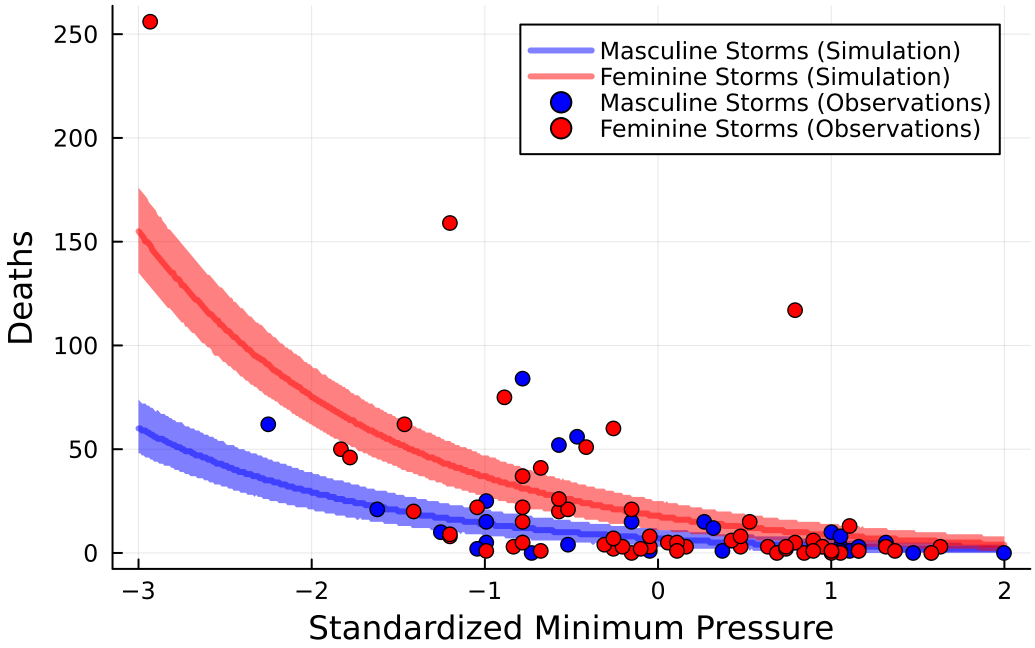

Figure 7: Simulation of hurricane deaths based on femininity of name and minimum hurricane pressure. Included are projections assuming the most masculine and feminine names in the dataset.

The model fails to predict the storms with the most and least deaths for both genders; this isn't entirely shocking since most of the data consists of storms with low deaths, and a Poisson model has equal variance to its mean. We would need a model with more dispersion (such as a negative binomial, which was used in the original paper) to potentially predict these. However, the model does clearly predict an effect from the gender of the storm name, as we can see in Figure 7.

Problem 4.3

The effect size shown in Figure 7 is quite strong, and does not in general seem supported by the data. While there are female-named storms with more deaths than the male-named storms with similar minimum pressures, we can also see many cases where they are quite similar. This effect seems to be driven by the large outlying storms, which tend to have female names, but aside from these few events, the data do not appear to suggest a systematic pattern of greater deaths from female storms.

Problem 4.4

We can now stop taking this hypothesis seriously. To what extent do you think the finding could be an artifact of the dataset (e.g. there is no actual effect, but there are coincidental features of the data that produce the result that female-named storms are more deadly than male-named storms)? Justify this conclusion with specific reference to an exploratory analysis of the data.

As noted in the solution to Problem 4.3, the model results seem to be driven entirely by the most deadly four storms, all of which had female names. We can see how likely this is from random variation. Let's look at how many more storms in the dataset have female names than male names.

```
sum(dat.female) / nrow(dat)
```

```
0.6739130434782609
```

So 2/3 of the storms in the dataset have female names. As a result, there is a $(2/3)^4 = 20\%$ probability that the four deadliest storms would happen to have female names just by chance, which is hardly negligible.

Extra (I don't expect you to have looked this up):

In fact, this isn't random: the dataset includes storms between 1953–1978, when all tropical storms were given female names. And in fact, other than Sandy (which is classified as having a female name, but it is generally considered unisex), the four most deadly storms were from this period:

```
sort(dat, :deaths, rev=true)
```

	name	year	deaths	category	min_pressure	damage_norm	female	femininity
	String15	Int64	Int64	Int64	Int64	Int64	Int64	Float64
1	Camille	1969	256	5	909	23040	1	9.05556
2	Diane	1955	200	1	987	14730	1	9.88889
3	Sandy	2012	159	2	942	75000	1	9.0
4	Agnes	1972	117	1	980	20430	1	8.66667
5	Ike	2008	84	2	950	20370	0	1.88889
6	Betsy	1965	75	3	948	20000	1	8.33333
7	Andrew	1992	62	5	922	66730	0	2.22222
8	Rita	2005	62	3	937	10690	1	9.5
9	Carol	1954	60	3	960	19321	1	8.11111
10	Floyd	1999	56	2	956	8130	0	1.83333
11	Gustav	2008	52	2	954	4360	0	1.72222
12	Isabel	2003	51	2	957	4980	1	9.38889
13	Donna	1960	50	4	930	53270	1	9.27778
14	Carla	1961	46	4	931	15850	1	9.5
15	Irene	2011	41	1	952	7110	1	9.27778
16	Hilda	1964	37	3	950	2770	1	8.83333
17	Fran	1996	26	3	954	8260	1	7.16667
18	Ivan	2004	25	3	946	18590	0	1.05556
19	Gracie	1959	22	3	950	510	1	9.77778
20	Celia	1970	22	3	945	6870	1	9.44444
21	Eloise	1975	21	3	955	6190	1	8.94444
22	Alicia	1983	21	3	962	10400	1	9.83333
23	Hugo	1989	21	4	934	20020	0	2.88889
24	Edna	1954	20	3	954	3230	1	8.55556
...	...	...	...	...	...	...	...	...

As noted, while Katrina and Audrey would have also been among the top storms (but were left out of the dataset), Hurricane Audrey was in 1957, which is also in the female-only period. Neglecting the storms from that period makes the deadliest storms Katrina, Sandy, and Ike: hardly signs of a clear female bias. Moreover, we know that many of the deaths from Katrina resulted from engineering failures rather than residents "not taking storms with female names seriously."

For a thorough analysis of this paper's statistical failings (which go beyond what we have considered here), see this paper by Gary Smith. As a sign of how much of a laughingstock this paper is among statisticians, you can see how often it's mockingly referenced at Andrew Gelman's blog. This is further evidence that just because you can do statistics, it doesn't mean your results make sense, even if you fit your model and get "statistically significant" results. You should always think carefully about what your data-generating mechanism actually implies, and how fit for purpose your dataset is to examine your hypothesis (in this case, the inclusion of data between 1953 and 1978 clearly biases the results in favor of the authors' hypothesis).

References