

Homework 2

This homework is to give a brief reminder of R, RStudio, and statistical topics covered in IEMS 303.

Note: The homework is scored out of 100 points. The problems add up to 90 points, while the remaining ten points will be graded according to a writing rubric, given at the end of the assignment.

R/RStudio installation If you have not installed R and RStudio, follow the installation instructions outlined in <https://posit.co/download/rstudio-desktop/>. You are strongly encouraged to use R Markdown to integrate text, code, images and mathematics or you can use the latex code we provide.

Question 1. Linear Regression with Missing Data We consider the standard regression model

$$(1) \quad Y = \beta_0 + \beta_1 X + \epsilon, \quad \epsilon \sim N(0, \sigma^2),$$

with $X \in \mathbb{R}$. In this exercise, 10% of the generated Y values are randomly set to zero. That is, if we denote the observed response by \tilde{Y} , then

$$(2) \quad \tilde{Y} = \begin{cases} Y, & \text{with probability 0.9,} \\ 0, & \text{with probability 0.1.} \end{cases}$$

Exercise 1: Bias of the OLS Estimator. When we run OLS on the observed data \tilde{Y} , Can $E[\tilde{Y} | X]$ be written down as a linear function X ? Prove that the bias in the estimators is then given by

$$\text{Bias}(\hat{\beta}_0) = \beta_0 - 0.9\beta_0 = \frac{\beta_0}{10}, \quad \text{Bias}(\hat{\beta}_1) = \beta_1 - 0.9\beta_1 = \frac{\beta_1}{10}.$$

This means both estimators are biased downward by 10% of the true parameter value.

Write down your answer in the sol environment.

Exercise 2: Log-Likelihood Formulation. To account for the zero-inflation, we use a likelihood that reflects the two different ways an observation can occur. For each observation i , show that the likelihood is given by:

$$L_i(\beta_0, \beta_1, \sigma) = \begin{cases} 0.1, & \text{if } y_i = 0, \\ 0.9 \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}\right), & \text{if } y_i \neq 0. \end{cases}$$

What is the full log-likelihood for the dataset? What is the new objective function you written down for the problem? What is the algorithm looks like?

You only need to complete one of the Question 2 or Question 3.

(Completing Question 3 will have 10 extra credit.)

Question 2. Bias and Variance Trade-off for Newton Method In this question, we use logistic regression as an example. We assume a logistic regression model with one predictor:

$$p_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}}, \quad \text{with } z_i = \beta_0 + \beta_1 x_i,$$

where p_i is the probability that $y_i = 1$ given x_i . The log-likelihood for n independent observations is

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^n \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right].$$

Exercise 1: Compute the Gradient and Hessian. To derive the gradient, differentiate the log-likelihood with respect to β_j (with $j = 0, 1$). First note that:

$$\frac{\partial p_i}{\partial z_i} = p_i(1 - p_i), \quad \text{and} \quad \frac{\partial z_i}{\partial \beta_j} = \begin{cases} 1, & j = 0, \\ x_i, & j = 1. \end{cases}$$

Thus, using the chain rule, $\frac{\partial p_i}{\partial \beta_j} = p_i(1 - p_i) x_{ij}$, where we define $x_{i0} = 1$ and $x_{i1} = x_i$.

Differentiating the log-likelihood yields $\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^n \left[\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i} \right] \frac{\partial p_i}{\partial \beta_j}$. Substituting the derivative of p_i and simplifying gives

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^n [y_i - p_i] x_{ij}.$$

Now you will do the computation for the Hessian, we differentiate the gradient with respect to β_k : $\frac{\partial^2 \ell}{\partial \beta_j \partial \beta_k} = \sum_{i=1}^n \frac{\partial}{\partial \beta_k} \{(y_i - p_i) x_{ij}\}$. Show that if X is the $n \times 2$ design matrix with rows $x_i = (1, x_i)$ and $p = (p_1, \dots, p_n)^\top$, then

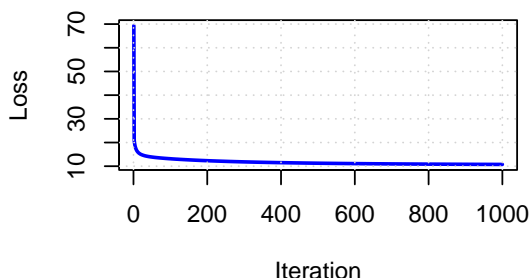
- the gradient is $\nabla \ell(\beta) = X^\top (y - p)$,
- the hessian is $H(\beta) = -X^\top W X$, where W is an $n \times n$ diagonal matrix with $W_{ii} = p_i(1 - p_i)$.

More generally, if X is the $n \times 2$ design matrix with rows $x_i = (1, x_i)$ and $p = (p_1, \dots, p_n)^\top$, and

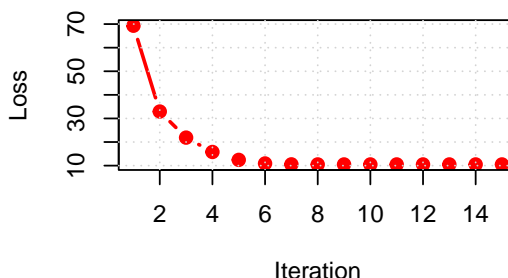
Hint: Since y_i does not depend on β_k , only p_i does. Thus using the chain rule, we know that $\frac{\partial^2 \ell}{\partial \beta_j \partial \beta_k} = -\sum_{i=1}^n \frac{\partial p_i}{\partial \beta_k} x_{ij}$. Then using the earlier derivative, of $\frac{\partial p_i}{\partial \beta_k}$.

Exercise 2: Complete the code. Using the previous computation of gradient and hessian to complete the Newton Method and gradient descent code for solving logistic regression. If you successfully complete the code, you are able to generate the following figures. Write down your findings.

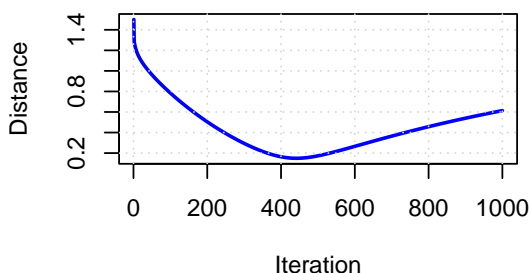
Gradient Descent: Loss



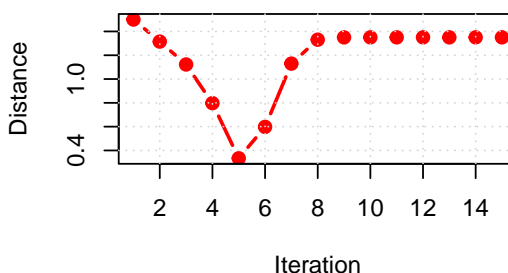
Newton's Method: Loss



Gradient Descent: ||beta - true_beta||



Newton's Method: ||beta - true_beta||



```

1 # Set seed for reproducibility
2 set.seed(123)
3
4 # -----
5 # 1. Create a toy dataset
6 # -----
7 n <- 100                                # number of observations
8 x1 <- rnorm(n)*10
9 x2 <- rnorm(n)
10 X <- cbind(1, x1, x2)                  # include intercept
11

```

```

12 # True coefficients for our logistic model
13 true_beta <- c(0.5, 1, -1)
14
15 # Logistic function
16 logistic <- function(z) 1 / (1 + exp(-z))
17
18 # Compute the probabilities using the logistic function
19 p <- logistic(X %*% true_beta)
20
21 # Generate binary response variable from a Bernoulli distribution
22 y <- rbinom(n, size = 1, prob = p)
23
24 # -----
25 # 2. Define the negative log-likelihood and its derivatives
26 # -----
27 nll <- function(beta, X, y) {
28   p <- logistic(X %*% beta)
29   # Negative log-likelihood (adding a small constant to avoid log(0))
30   -sum(y * log(p + 1e-8) + (1 - y) * log(1 - p + 1e-8))
31 }
32
33 # Gradient of the negative log-likelihood
34 grad_nll <- function(beta, X, y) {
35   # -----
36   # !!!!!!!
37   # PUT Your Code here
38   # !!!!!!!
39   # -----
40 }
41
42 # Hessian of the negative log-likelihood
43 hess_nll <- function(beta, X, y) {
44   # -----
45   # !!!!!!!
46   # PUT Your Code here
47   # !!!!!!!
48   # -----
49 }
50
51 # -----
52 # 3. Solve using Gradient Descent
53 # -----
54 beta_gd <- rep(0, ncol(X))      # Initialize coefficients
55 lr <- 0.001                      # Learning rate
56 num_iter <- 1000                # Number of iterations
57
58
59 loss_gd <- numeric(num_iter)    # Store loss values
60 dist_gd <- numeric(num_iter)    # Store Euclidean distance to true_beta
61
62 for (i in 1:num_iter) {

```

```

63 loss_gd[i] <- nll(beta_gd, X, y)
64 dist_gd[i] <- sqrt(sum((beta_gd - true_beta)^2))
65 # -----
66 # !!!!!!!
67 # PUT Your Code here
68 # !!!!!!!
69 # The code would look like beta_gd <- ?
70 # -----
71 }
72
73 # -----
74 # 4. Solve using Newton's Method
75 # -----
76 beta_newton <- rep(0, ncol(X))          # Initialize coefficients
77 num_iter_newton <- 15                   # Newton's method converges
    quickly
78
79 loss_newton <- numeric(num_iter_newton)
80 dist_newton <- numeric(num_iter_newton)
81
82 for (i in 1:num_iter_newton) {
83   loss_newton[i] <- nll(beta_newton, X, y)
84   dist_newton[i] <- sqrt(sum((beta_newton - true_beta)^2))
85   grad <- grad_nll(beta_newton, X, y)
86   H <- hess_nll(beta_newton, X, y)
87   # -----
88   # !!!!!!!
89   # PUT Your Code here
90   # !!!!!!!
91   # The code would look like beta_newton <- ?
92   # -----
93 }
94
95 # -----
96 # 5. Plot the Loss and Distance Curves
97 # -----
98 # Set up a 2x2 plotting area
99 par(mfrow = c(2, 2))
100
101 # Gradient Descent Loss
102 plot(loss_gd, type = "l", col = "blue", lwd = 2,
103      main = "Gradient Descent: Loss", xlab = "Iteration", ylab = "Loss"
104      )
105 grid()
106
107 # Newton's Method Loss
108 plot(loss_newton, type = "b", col = "red", lwd = 2, pch = 19,
109      main = "Newton's Method: Loss", xlab = "Iteration", ylab = "Loss")
110 grid()
111
112 # Gradient Descent Distance

```

```

112 plot(dist_gd, type = "l", col = "blue", lwd = 2,
113       main = "Gradient Descent: ||beta - true_beta||", xlab = "Iteration",
114       ylab = "Distance")
115
116 # Newton's Method Distance
117 plot(dist_newton, type = "b", col = "red", lwd = 2, pch = 19,
118       main = "Newton's Method: ||beta - true_beta||", xlab = "Iteration",
119       ylab = "Distance")

```

LISTING 1. R code: Data generation, Newton's method, and plotting

Question 3. Linear Regression with Censored Data Suppose the true (latent) model is linear:

$$(3) \quad Y^* = \beta_0 + \beta_1 X + \epsilon, \quad \epsilon \sim N(0, \sigma^2).$$

However, instead of observing Y^* , we observe

$$(4) \quad Y = \max\{Y^*, 0\}.$$

This is a censored (or Tobit) model where values below 0 are censored to 0.

Exercise 1: Bias of the OLS Estimator. Compute the Expectation. Even though the latent relationship is linear, show that the observed conditional expectation becomes

$$E[Y | X] = (\beta_0 + \beta_1 X) \Phi\left(\frac{\beta_0 + \beta_1 X}{\sigma}\right) + \sigma \varphi\left(\frac{\beta_0 + \beta_1 X}{\sigma}\right),$$

which is not linear in X . Here φ and Φ denote the standard normal PDF and CDF, respectively. Explain why if one naively applies OLS to Y without accounting for the censoring, the estimated coefficients will be biased.

Exercise 2: Log-Likelihood Formulation. The proper likelihood accounts for whether an observation is censored or not. For each observation i , define $\mu_i = \beta_0 + \beta_1 X_i$. Then the likelihood is given by

$$L_i(\beta_0, \beta_1, \sigma) = \begin{cases} \Phi\left(\frac{-\mu_i}{\sigma}\right), & \text{if } Y_i = 0, \\ \frac{1}{\sigma} \varphi\left(\frac{Y_i - \mu_i}{\sigma}\right), & \text{if } Y_i > 0. \end{cases}$$

What is the full log-likelihood $\ell(\beta_0, \beta_1, \sigma)$ and the new loss function you have?

Exercise 3: Newton Method. We aim to use Newton method to minimize the negative log-likelihood $\text{NLL}(\beta_0, \beta_1, \sigma) = -\ell(\beta_0, \beta_1, \sigma)$, we need first to compute the hessian and gradient.

3 a) Gradient Derivation For non-censored observations ($Y_i > 0$), the log-likelihood is $\ell_i = -\log \sigma - \frac{1}{2} \log(2\pi) - \frac{(Y_i - \mu_i)^2}{2\sigma^2}$. (why?) Differentiating with respect to β_j (with $j = 0, 1$):

$$\frac{\partial \ell_i}{\partial \beta_j} = \frac{Y_i - \mu_i}{\sigma^2} \frac{\partial \mu_i}{\partial \beta_j}.$$

Since $\mu_i = \beta_0 + \beta_1 X_i$, $\frac{\partial \mu_i}{\partial \beta_0} = 1$, $\frac{\partial \mu_i}{\partial \beta_1} = X_i$, the contribution to the gradient of the NLL (remember we take the negative derivative) is

$$-\frac{\partial \ell_i}{\partial \beta_j} = -\frac{Y_i - \mu_i}{\sigma^2} \frac{\partial \mu_i}{\partial \beta_j}.$$

Similarly, for censored observations ($Y_i = 0$), the log-likelihood is $\ell_i = \log \Phi\left(-\frac{\mu_i}{\sigma}\right)$. (why?) Let $z_i = -\frac{\mu_i}{\sigma}$. Then, using the same steps as before show that $\frac{\partial \ell_i}{\partial \beta_j} = -\frac{1}{\sigma} \frac{\varphi(z_i)}{\Phi(z_i)} \frac{\partial \mu_i}{\partial \beta_j}$ (**provide your calculation to check**)

my result) and the overall gradient of the NLL is

$$\nabla \text{NLL}(\beta_0, \beta_1) = - \sum_{i: Y_i > 0} \frac{Y_i - \mu_i}{\sigma^2} \begin{pmatrix} 1 \\ X_i \end{pmatrix} + \sum_{i: Y_i = 0} \frac{1}{\sigma} \frac{\varphi(z_i)}{\Phi(z_i)} \begin{pmatrix} 1 \\ X_i \end{pmatrix}.$$

3 b) Hessian Derivation for Non-censored observations ($Y_i > 0$) For non-censored observations ($Y_i > 0$), show that the hessian is

$$H_i^{\text{non-cens}} = \frac{1}{\sigma^2} \begin{pmatrix} 1 \\ X_i \end{pmatrix} \begin{pmatrix} 1 & X_i \end{pmatrix}.$$

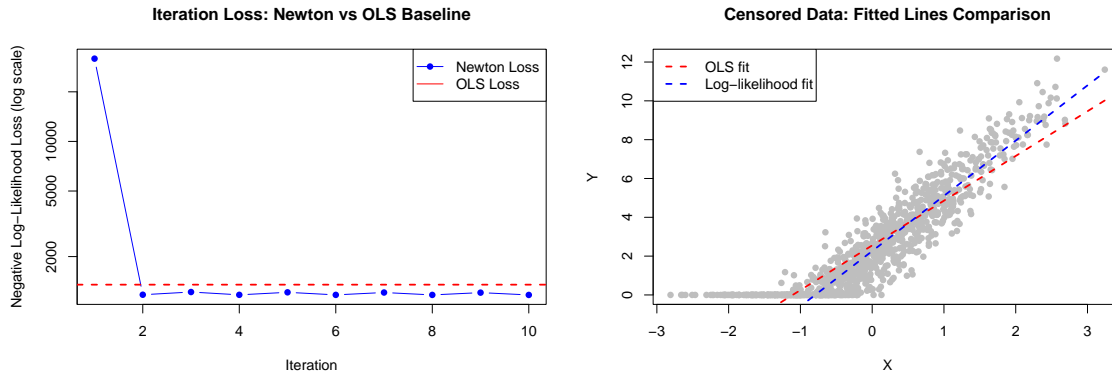
3 c) Hessian Derivation for Censored observations ($Y_i = 0$) For censored observations, we've showed the gradient is $\frac{\partial \ell_i}{\partial \beta_j} = -\frac{1}{\sigma} \frac{\varphi(z_i)}{\Phi(z_i)} \frac{\partial \mu_i}{\partial \beta_j}$. Show that the Hessian contribution is

$$H_i^{\text{cens}} = \frac{1}{\sigma^2} \begin{pmatrix} 1 \\ X_i \end{pmatrix} \begin{pmatrix} 1 & X_i \end{pmatrix} \frac{\varphi(z_i)}{\Phi(z_i)^2} \left(z_i \Phi(z_i) + \varphi(z_i) \right)$$

based on the following Fact.

Fact. If we define $r(z_i) = \frac{\varphi(z_i)}{\Phi(z_i)}$. Then its derivative is $r'(z_i) = -\frac{\varphi(z_i)}{\Phi(z_i)^2} \left(z_i \Phi(z_i) + \varphi(z_i) \right)$. (The code for computing $r'(z_i)$ is provided in the following R code.)

3 d) Complete the Newton Method's Code. Complete the following code using Newton method for NLL to do linear regression with censored data. If you successfully complete the code, you are able to generate the following figures. Write down your findings. (You can try different choices of n or other hyperparameter in the code to see what you find.)



```

1 # -----
2 # Data Generation
3 # -----
4 set.seed(123)
5 n <- 1000
6 beta0_true <- 2; beta1_true <- 3; sigma <- 1
7 X <- rnorm(n)
8 epsilon <- rnorm(n, 0, sigma)
9 Y_star <- beta0_true + beta1_true * X + epsilon
10 # Censor at 0
11 Y <- pmax(Y_star, 0)
12
13 # -----
14 # Negative Log-Likelihood Function (NLL)
15 # -----
16 nll <- function(params, X, Y, sigma) {
17   beta0 <- params[1]

```

```

18  beta1 <- params[2]
19  mu <- beta0 + beta1 * X
20  # For censored observations: Y == 0
21  cens <- (Y == 0)
22  ll_cens <- sum( log(pnorm(-mu[cens] / sigma)) )
23  # For non-censored observations: Y > 0
24  noncens <- (Y > 0)
25  ll_noncens <- sum( -log(sigma) - 0.5 * log(2*pi) - ((Y[noncens] - mu[
    noncens])^2) / (2*sigma^2) )
26  return( - (ll_cens + ll_noncens) )
27 }
28
29 # -----
30 # Gradient of NLL
31 # -----
32 grad_nll <- function(params, X, Y, sigma) {
33   beta0 <- params[1]
34   beta1 <- params[2]
35   mu <- beta0 + beta1 * X
36
37   noncens <- (Y > 0)
38   cens <- (Y == 0)
39
40   # For non-censored observations:
41   grad_non_cens_0 <- - sum((Y[noncens] - mu[noncens]) / sigma^2)
42   grad_non_cens_1 <- - sum((Y[noncens] - mu[noncens]) * X[noncens] /
    sigma^2)
43
44   # For censored observations:
45   z <- -mu[cens] / sigma
46   Phi_z <- pnorm(z) + 1e-10 # avoid division by zero
47   factor <- dnorm(z) / (sigma * Phi_z)
48   grad_cens_0 <- sum(factor)
49   grad_cens_1 <- sum(factor * X[cens])
50
51   grad0 <- grad_non_cens_0 + grad_cens_0
52   grad1 <- grad_non_cens_1 + grad_cens_1
53
54   return(c(grad0, grad1))
55 }
56
57 # -----
58 # Hessian of NLL
59 # -----
60 hessian_nll <- function(params, X, Y, sigma) {
61   beta0 <- params[1]
62   beta1 <- params[2]
63   mu <- beta0 + beta1 * X
64
65   H11 <- 0; H12 <- 0; H22 <- 0
66

```

```

67 # Non-censored part:
68 noncens <- (Y > 0)
69 H11 <- H11 + sum( rep(1, sum(noncens)) / sigma^2 )
70 H12 <- H12 + sum( X[noncens] / sigma^2 )
71 H22 <- H22 + sum( (X[noncens]^2) / sigma^2 )
72
73 # Censored part:
74 cens <- (Y == 0)
75 if(sum(cens) > 0){
76 # -----
77 # !!!!!!!
78 # PUT Your Code here
79 # !!!!!!!
80 # -----
81 # derivative of r(z)=phi(z)/Phi(z)
82 rprime <- - (phi_z / (Phi_z^2)) * ( z * Phi_z + phi_z )
83
84 }
85
86 H <- matrix(c(H11, H12, H12, H22), nrow = 2)
87 return(H)
88 }
89
90 # -----
91 # Newton's Method with Loss History
92 # -----
93 params_newton <- c(0, 0) # initial guess for (beta0, beta1)
94 num_iter_newton <- 10
95 loss_history_newton <- numeric(num_iter_newton)
96
97 for(i in 1:num_iter_newton) {
98   grad <- grad_nll(params_newton, X, Y, sigma)
99   H <- hessian_nll(params_newton, X, Y, sigma)
100   # -----
101   # !!!!!!!
102   # PUT Your Code here
103   # !!!!!!!
104   # The code would look like params_newton <- ?
105   # -----
106
107   loss_history_newton[i] <- nll(params_newton, X, Y, sigma)
108 }
109 newton_est <- params_newton
110 cat("Newton's Method Estimates (Beta0, Beta1):\n")
111 print(newton_est)
112
113
114 # -----
115 # OLS Estimation (ignoring censoring) as Baseline
116 # -----
117 ols_model <- lm(Y ~ X)

```



```

118 ols_est <- coef(ols_model)
119 cat("OLS Estimates (Beta0, Beta1):\n")
120 print(ols_est)
121 loss_ols <- nll(ols_est, X, Y, sigma)
122 cat("Negative Log-Likelihood Loss (OLS):\n")
123 print(loss_ols)
124
125 # -----
126 # Plot: Newton Iteration Loss vs OLS Baseline
127 # -----
128 # Combined Loss Plot with Log-Scale on Y-Axis: Newton, Gradient Descent
129 # , and OLS Baseline
130 plot(1:num_iter_newton, loss_history_newton, type='b', pch=16, col='
    blue', log="y",
131       xlab='Iteration', ylab='Negative Log-Likelihood Loss (log scale)',
132       main='Iteration Loss: Newton vs OLS Baseline')
133 abline(h=loss_ols, col='red', lwd=2, lty=2)
134 legend("topright", legend=c("Newton Loss", "OLS Loss"),
135       col=c("blue", "red"), lty=c(1,1,2), pch=c(16, NA, NA))
136
137 # -----
138 # Plot: Fitted Lines Comparison
139 # -----
140 plot(X, Y, pch=16, col='grey',
141       main='Censored Data: Fitted Lines Comparison',
142       xlab='X', ylab='Y')
143 curve(ols_est[1] + ols_est[2]*x, add=TRUE, col='red', lwd=2, lty=2)
144 curve(newton_est[1] + newton_est[2]*x, add=TRUE, col='blue', lwd=2, lty
    =2)
145 legend("topleft", legend=c("OLS fit", "Log-likelihood fit"),
146       col=c("red", "blue", "purple"), lty=2, lwd=2)

```

LISTING 2. R code: Data generation, Newton's method, and plotting

Rubric (10)

- The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow.
- Questions which ask for a plot or table are answered with both the figure itself and the command (or commands) use to make the plot. Plots are carefully labeled, with informative and legible titles, axis labels.
- All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision.
- Code is either properly integrated with a tool like R Markdown or included as a separate R file. In the former case, both the knitted and the source file are included. In the latter case, the code is clearly divided into sections referring to particular problems. In either case, the code is indented, commented, and uses meaningful names.
- All parts of all problems are answered with actual coherent sentences, and never with raw computer code or its output. For full credit, all code runs, and the Markdown file knits (if applicable).