Solutions to Exam 1

Rongfei Jin

March 12, 2025

Problem 1

(a) See (Appendix 1)

(b) See (Appendix 1)
Some values are NA in the summary because of the collinearity which means that the design matrix is not full rank

(c) (1) Gender
  (i) Faster multiplication on 0
  (ii) gender encoded in 0,1 is easiser to interpret because the coefficients represents the effect of being male, or no effect if being female

 (2) Income and travel
 inc25p: 1 if income is greater than 25k, 0 otherwise
 inc55p: 1 if income is greater than 55k, 0 otherwise
 inc95p: 1 if income is greater than 95k, 0 otherwise
 tra025p: 1 if travel is greater than .25h, 0 otherwise
 tra400p: 1 if travel is greater than 4h, 0 otherwise

  (i) Design matrix achieves full rank. These transformations solve the collinearity problem since if the model has all the condition except the last one, then the last one is determined, so numerically it is more stable.
  (ii) Easier to interpret as the coefficients are simply the effect of income greater than a certain threshold

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)  4.09673    0.02789  146.90   <2e-16 ***
gen          0.35334    0.02121   16.66   <2e-16 ***
inc25p      -0.01424    0.02669   -0.53     0.59    
inc55p      -0.54173    0.03158  -17.15   <2e-16 ***
inc95p      -0.00799    0.03471   -0.23     0.82    
tra025p     -0.61883    0.02174  -28.47   <2e-16 ***
tra400p     -1.82061    0.04988  -36.50   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 18630  on 341  degrees of freedom
Residual deviance: 11325  on 335  degrees of freedom
AIC: 12661

Number of Fisher Scoring iterations: 6
```

Figure 1: Model 1d

(d) The new model has 7 coefficients (without intercept) and NO coefficients are NA. Gender, income greater than 55k, travel time greater than 0.25h and travel time greater than 4h are significant.

Based on the significant coefficients, we can make the following interpretation.

(i) Males are associated with 0.36 more visit than female

(ii) Income greater than 55k is associated with 0.5 less visit than income less than 55k

(iii) Travel time greater than 0.25h is associated with 0.6 less visit than travel time less than 0.25h

(iv) Travel time greater than 4h is associated with 1.8 less visit than travel time less than 4h

	[,1]	[,2]
(Intercept)	4.04207	4.15138
gen	0.31177	0.39491
inc25p	-0.06654	0.03807
inc55p	-0.60363	-0.47984
inc95p	-0.07603	0.06004
tra025p	-0.66143	-0.57622
tra400p	-1.91836	-1.72285

Figure 2: Model CI

(e)

(f) The possion regression has the following probability density function

$$p(y|\eta) = \frac{\eta^y e^{-\eta}}{y!}$$

where $\eta = \exp(\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p)$ is the mean of the possion distribution. Therefore, we have

$$\mathrm{E}(y|x_1, \ldots, x_p) = \eta = \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)$$

Given female, earning \$65,000 annually, and living two miles from the park. We have the following data gen = 0, inc25p = 1 inc55p = 1, inc95p = 0,tra025p = 0, tra400p = 0.
Therefore, we have

$$\mathrm{E}(y|x_1, \ldots, x_p) = \exp(\beta_0 + \beta_1 0 + \beta_2 1 + \beta_3 1 + \beta_4 0 + \beta_5 0 + \beta_6 0) = \exp(\beta_0 + \beta_2 + \beta_3) = 34.49$$

Given male, earning \$65,000 annually, and living two miles from the park. We have the following data
gen = 1, inc25p = 1 inc55p = 1, inc95p = 0,tra025p = 0, tra400p = 0.

$$\mathrm{E}(y|x_1, \ldots, x_p) = \exp(\beta_0 + \beta_1 1 + \beta_2 1 + \beta_3 1 + \beta_4 0 + \beta_5 0) = \exp(\beta_0 + \beta_1 + \beta_2 + \beta_3) = 49.11$$

$$\frac{E(y|\text{male with given conditions})}{E(y|\text{female with given conditions})} = \frac{49.11}{34.49} \approx 1.42$$

$$\frac{E(y|\text{female with given conditions})}{E(y|\text{male with given conditions})} = \frac{34.49}{49.11} \approx 0.703$$

Problem 2

(a) See (Appendix 2)

	bias	ci_lower	ci_upper	ci_width	ci_with_nb
Intercept	-0.013830	3.664707	4.494433	0.8297	0.10932
gen	-0.010294	0.005338	0.701351	0.6960	0.08314
inc25p	0.003618	-0.464708	0.440823	0.9055	0.10461
inc55p	-0.002206	-1.061014	0.003987	1.0650	0.12380
inc95p	-0.005427	-0.531057	0.449845	0.9809	0.13607
tra025p	-0.006011	-0.946538	-0.292945	0.6536	0.08521
tra400p	-0.091385	-2.723039	-1.026566	1.6965	0.19551

Figure 3: Bootstrap CI

(b) (i) the center from both approaches are very close

(ii) the bootstrapped CI width is wider than the normal CI, which is expected since the normal CI assumes the distribution is normal, but the bootstrapped CI does not make this assumption

Problem 3

(a) Since $Y \sim \text{Poisson}(\lambda)$ we have $P(Y=y;\lambda) = \frac{\lambda^y e^{-\lambda}}{y!}$
We compute the Momement Generating Function

$$\begin{aligned}
M_Y(t) &= \mathrm{E}(e^{tY}) \\
&= \sum_{y=0}^{\infty} e^{ty} P(Y=y;\lambda) \\
&= \sum_{y=0}^{\infty} e^{ty} \frac{\lambda^y e^{-\lambda}}{y!} \\
&= e^{-\lambda} \sum_{y=0}^{\infty} \frac{(\lambda e^t)^y}{y!} \\
&= e^{-\lambda} e^{\lambda e^t} && \text{Taylor expansion} \\
&= e^{\lambda(e^t-1)}
\end{aligned}$$

Now we compute mean by deriving the first moment

$$\begin{aligned}
\mathrm{E}(Y) &= M_Y'(0) \\
&= \frac{d}{dt} e^{\lambda(e^t-1)} \Big|_{t=0} \\
&= \lambda e^{\lambda(e^0-1)} \\
&= \lambda
\end{aligned}$$

Now we compute the variance by first deriving the second moment

$$\begin{aligned}
\mathrm{E}(Y^2) &= M_Y''(0) \\
&= \frac{d^2}{dt^2} e^{\lambda(e^t-1)} \Big|_{t=0} \\
&= \lambda e^{\lambda(e^0-1)} + \lambda^2 e^{\lambda(e^0-1)} \\
&= \lambda + \lambda^2
\end{aligned}$$

Now we compute the variance

$$\begin{aligned}
\mathrm{Var}(Y) &= \mathrm{E}(Y^2) - \mathrm{E}(Y)^2 \\
&= (\lambda + \lambda^2) - \lambda^2 \\
&= \lambda
\end{aligned}$$

(b)

$$\begin{aligned}
p(y|\lambda) &= \frac{\lambda^y e^{-\lambda}}{y!} \\
&= e^{\log(\frac{\lambda^y e^{-\lambda}}{y!})} \\
&= e^{y\log(\lambda) - \lambda - \log(y!)}
\end{aligned}$$

(c) since $\log(\lambda) = \beta_0 + \boldsymbol{\beta} \cdot \mathbf{x} = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}$, we have

$$p(y|\mathbf{x}, \beta_0, \boldsymbol{\beta}) = \exp\left\{[\beta_0 + \boldsymbol{\beta}^T\mathbf{x}]y - \exp\left\{\beta_0 + \boldsymbol{\beta}^T\mathbf{x}\right\} - \log(y!)\right\}$$

$$E(y|\mathbf{x}, \beta_0, \boldsymbol{\beta}) = \lambda = \exp\left\{\beta_0 + \boldsymbol{\beta}^T\mathbf{x}\right\}$$

(d)

$$\begin{aligned}
\ell(\beta_0, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X}) &= \log \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \beta_0, \boldsymbol{\beta}) \\
&= \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \beta_0, \boldsymbol{\beta}) \\
&= \sum_{i=1}^{n} \left\{ [\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i] y_i - \exp\left\{ \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i \right\} - \log(y_i!) \right\}
\end{aligned}$$

Problem 4

(a)

$$\frac{\partial}{\partial \beta_0}[-\ell(\beta_0, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X})] = -\sum_{i=1}^{n} y_i + \sum_{i=1}^{n} \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)$$

$$\frac{\partial}{\partial \boldsymbol{\beta}}[-\ell(\beta_0, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X})] = -\sum_{i=1}^{n} y_i \mathbf{x}_i + \sum_{i=1}^{n} \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)\mathbf{x}_i$$

(b)

$$\frac{\partial^2}{\partial \beta_0^2}[-\ell(\beta_0, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X})] = \sum_{i=1}^{n} \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)$$

$$\frac{\partial^2}{\partial \boldsymbol{\beta}^2}[-\ell(\beta_0, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X})] = \sum_{i=1}^{n} \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)\mathbf{x}_i \mathbf{x}_i^T$$

$$\frac{\partial^2}{\partial \beta_0 \partial \boldsymbol{\beta}}[-\ell(\beta_0, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X})] = \sum_{i=1}^{n} \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)\mathbf{x}_i$$

$$\mathrm{H} = \begin{bmatrix} \frac{\partial}{\partial \beta_0^2} & \frac{\partial}{\partial \beta_0 \partial \boldsymbol{\beta}} \\ \frac{\partial}{\partial \beta_0 \partial \boldsymbol{\beta}} & \frac{\partial}{\partial \boldsymbol{\beta}^2} \end{bmatrix}$$

where $\frac{\partial}{\partial \boldsymbol{\beta}^2}$ is a matrix with $\frac{\partial}{\partial \beta_i \partial \beta_j}$ where$i = 1, \ldots, p, j = 1, \ldots p$

(c) for the function to be convex, the Hessian matrix must be positive definite.

Problem 5

See (Appendix 5)

(a) for the ease of computation, we will express the gradient and Hessian in matrix form and let $\boldsymbol{\beta}' = \begin{bmatrix}\beta_0 & \beta_1 & \dots & \beta_p\end{bmatrix}^T$,

$$\frac{\partial}{\partial \boldsymbol{\beta}'} = -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T \exp(\mathbf{X}\boldsymbol{\beta}')$$

$$\frac{\partial^2}{\partial \boldsymbol{\beta}' \partial \boldsymbol{\beta}'^T} = \mathbf{X}^T \operatorname{diag}(\exp\{\mathbf{X}\boldsymbol{\beta}'\})\mathbf{X}$$

```
               [,1]
            4.11149
gen         0.20623
inc25p      0.05751
inc55p     -0.52299
inc95p      0.01758
tra025p    -0.73525
tra400p    -1.70133
```

Figure 4: Coefficients from Newton's method

```
Coefficients:
        Estimate Std..Error  z.value   Pr...z..
         4.11149    0.03182 129.2064  0.000e+00
gen      0.20623    0.02308   8.9360  4.034e-19
inc25p   0.05751    0.03074   1.8712  6.131e-02
inc55p  -0.52299    0.03275 -15.9683  2.126e-57
inc95p   0.01758    0.03625   0.4849  6.277e-01
tra025p -0.73525    0.02391 -30.7549 1.050e-207
tra400p -1.70133    0.05111 -33.2874 5.875e-243

Null deviance:     15442  on  307  degrees of freedom
Residual deviance: 9408   on  302  degrees of freedom
AIC: 10603
Number of iterations: 30
Converged: TRUE
```

Figure 5: Summary of Newton's method

(b) (a) Use of standard error assumes we are randomly and independently sampling from the population.
 (b) Use of z value and p-value assumes the distribution is normal, and we know the population variance.
 (c) Use of Null and residual deviance assumes the distribution is exponential and sample is independent. [1]
 (d) Use of AIC assumes the sample size is large and sample is independent and model estimates are from MLE

Problem 6

To get the loss function, We first remove the constant term $\log(y!)$ from negative log-likelihood since it does not affect the optimization problem.

(a)

$$\mathcal{L}(\beta_0\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) = \sum_{i=1}^{n}\left\{[\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i]y_i - \exp\left\{\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i\right\}\right\}$$

Then we add the L2 regularization term to the loss function

$$\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) = \sum_{i=1}^{n}\left\{[\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i]y_i - \exp\left\{\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i\right\}\right\} + \lambda||\boldsymbol{\beta}||_2^2$$

We then derive the gradient and Hessian of the loss function

$$\begin{aligned}
\frac{\partial}{\partial\beta_0}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) &= -\sum_{i=1}^{n} y_i + \sum_{i=1}^{n}\exp(\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i)\\
\frac{\partial}{\partial\boldsymbol{\beta}}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) &= -\sum_{i=1}^{n} y_i\mathbf{x}_i + \sum_{i=1}^{n}\exp(\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i)\mathbf{x}_i - 2\lambda\boldsymbol{\beta}\\
\frac{\partial^2}{\partial\beta_0^2}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) &= \sum_{i=1}^{n}\exp(\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i)\\
\frac{\partial^2}{\partial\boldsymbol{\beta}^2}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) &= \sum_{i=1}^{n}\exp(\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^T - 2\lambda\mathbf{I}\\
\frac{\partial^2}{\partial\beta_0\partial\boldsymbol{\beta}}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) &= \sum_{i=1}^{n}\exp(\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i)\mathbf{x}_i
\end{aligned}$$

we can write the gradient and Hessian in matrix form as

$$\begin{aligned}
\frac{\partial}{\partial\boldsymbol{\beta}'}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta},\boldsymbol{\beta}'|\mathbf{y},\mathbf{X}) &= -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\exp(\mathbf{X}\boldsymbol{\beta}') - 2\lambda[0;\boldsymbol{\beta}]\\
\frac{\partial^2}{\partial\boldsymbol{\beta}'\partial\boldsymbol{\beta}'^T}\mathcal{L}_\lambda(\beta_0,\boldsymbol{\beta}\boldsymbol{\beta}'|\mathbf{y},\mathbf{X}) &= \mathbf{X}^T\operatorname{diag}(\exp\{\mathbf{X}\boldsymbol{\beta}'\})\mathbf{X} - 2\lambda(\mathbf{I}-e_1e_1^T)
\end{aligned}$$

(b) It is required that the Hessian matrix is positive definite for the loss function to be convex.

(c)

Coefficients:

	Estimate	Std..Error	z.value	Pr...z..
	4.09703	0.02785	147.1235	0.000e+00
gen	0.35287	0.02118	16.6575	2.669e-62
inc25p	-0.01497	0.02664	-0.5622	5.740e-01
inc55p	-0.54068	0.03144	-17.1973	2.783e-66
inc95p	-0.01015	0.03455	-0.2938	7.689e-01
tra025p	-0.61982	0.02171	-28.5446	3.278e-179
tra400p	-1.79975	0.04917	-36.6034	2.523e-293

Figure 6: Coefficients from Regularized Newton's method

	s1
(Intercept)	4.08646
gen	0.29228
inc25p	-0.08803
inc55p	-0.41363
inc95p	-0.13441
tra025p	-0.61705
tra400p	-0.86173

Figure 7: Coefficients from Regularized GLM

(d) See (Appendix 6)

The coefficients from regularized Newton's method and regularized GLM are very close. With $\lambda = 10$, the tra400p coefficient is greatly increased, while the gen coefficient is slightly decreased.

Problem 7

(a) See (Appendix 7)

(b) See (Appendix 7)

```
> cat("10-fold cross validation error: ", k_fold_glmnet(X, y, lambda=min_lambd$
10-fold cross validation error:  3.526
> cat("10-fold cross validation error: ", k_fold(X, y, lambda=min_lambda, k=10$
10-fold cross validation error:  3.427
```

Figure 8: Compare Newton's method and GLM, both regularized with min lambda, first error is from GLM, second is from Newton's method

(c) The regularized Newton's method has a lower error than the regularized GLM when the data is not randomized. When the data is randomized, the difference is undermined by the randomness.

Problem 8

(a) See (Appendix 8)

(b) See (Appendix 8)

Problem 9

See (Appendix 9)

(a) Let $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}$ be the coefficients of the model obtained at the minimum of the cross-validation error.

$$\mathrm{E}(y|\mathbf{x}_i) = \exp(\hat{\beta}_0 + \hat{\boldsymbol{\beta}}^T \mathbf{x}_i)$$

(b) Yes, the regularization set the coefficients of inc25p and inc95p to 0 while keep the inc55p to be non-zero. This corresponds to the coefficients significance in 1a model where inc25p and inc95p are not significant.

Problem 10

See (Appendix 10)

	newton_mse	ridge_newton_mse	ridge_mse	lasso_mse
MSE	1885.96	1885.93	2900.57	2900.13
RMSE	43.43	43.43	53.86	53.85
MAE	23.02	23.02	23.29	23.27

Figure 9: MSE of different models

If we compare the Newton vs Ridge-Newton method, we can see that these two methods show remarkably similar performance with MSEs of 1885.96 and 1885.93 respectively, and identical RMSE and MAE values (43.43 and 23.02). This suggests that the ridge penalty in the Ridge-Newton method is having minimal impact on your model, possibly because

1. The optimal regularization parameter is very small
2. The features might not have high multicollinearity
3. The dataset may be large enough relative to the number of features that regularization provides little benefit

If we compare the Ridge-Newton vs GLMNet, we can see that both the ridge and lasso implementations via GLMNet show nearly identical performance (MSE 2900, RMSE 53.86, MAE 23.28). This is considerably worse than the Newton-based methods in terms of MSE/RMSE, but interestingly, the MAE values remain close. The large gap in MSE/RMSE between the Newton methods and GLMNet implementations, coupled with similar MAE values, suggests that the GLMNet models are producing more outlier predictions with large errors, while maintaining similar average error magnitudes.

One possible reason for the difference in MSE/RMSE performance is that the Newton method directly optimizes by using the Hessian matrix, which can converge faster and more accurately for well-behaved problems. While the glmnet uses coordinate descent, which may not converge as fast and may not be as accurate for well-behaved problems. [2]

References

[1] Eduardo García-Portugués. 5.5 Deviance — Notes for Predictive Modeling — bookdown.org. [Accessed 12-03-2025].

[2] J. Kenneth Tay, Balasubramanian Narasimhan, and Trevor Hastie. Elastic net regularization paths for all generalized linear models. *Journal of Statistical Software*, 106(1), 2023.

Appendix 1

```
# 1a
#install.packages("COUNT")
# load loomis.rda
load("ex1/loomis.rda")
library(glmnet)

# drop income and travel
loomis$income <- NULL
loomis$travel <- NULL

# remove rows with NAs
loomis_preped <- na.omit(loomis)

str(loomis_preped)
# count NAs

# 1b
# fit glm possion model
glm_1b <- glm(anvisits ~ ., data = loomis_preped, family = poisson)

summary(glm_1b)

# Some values are NA in the summary because of the collinearity
# which means that the design matrix is not full rank
cat("Rank of the model matrix:", qr(glm_1b$model)$rank, "\n")
cat("Coefficient number (with intercept):", length(coef(glm_1b)), "\n")

# Rank of the model matrix: 8
# Coefficient number: 9


# copy data
dat <- loomis_preped
y <- dat[, 1] # dat is my data frame after completing 1 a
# create transformed dummy variables
# gender :
gen <- dat$gender - 1
# income :
inc25p <- as.numeric(apply(dat[, c(4, 5, 6)], 1, function(x) 1 * (sum(x) > 0)))
inc55p <- as.numeric(apply(dat[, c(5, 6)], 1, function(x) 1 * (sum(x) > 0)))
inc95p <- dat[, 6]
# travel
tra025p <- as.numeric(apply(dat[, c(8, 9)], 1, function(x) 1 * (sum(x) > 0)))
tra400p <- dat[, 9]
# create data matrix

X <- cbind( gen , inc25p , inc55p , inc95p , tra025p , tra400p )

yX <- data.frame(cbind(y, X))
# 1d
# fit glm model
glm_1d <- glm(y ~ ., family = poisson, data = yX)

summary(glm_1d)
```

```r
# 1e CI of the coefficients

coef <- glm_1d$coefficients
coef
se <- summary(glm_1d)$coefficients[, 2]

# 95% CI
ci <- cbind(coef - 1.96 * se, coef + 1.96 * se)
ci

# 1f


cat(exp(coef[1] + coef[3] + coef[4]))
cat(exp(coef[1] + coef[2] + coef[3] + coef[4]))

# row c(0, 1, 1, 0, 0, 0))
newdata <- data.frame(
  gen = 0,
  inc25p = 1,
  inc55p = 1,
  inc95p = 0,
  tra025p = 0,
  tra400p = 0
)
# copy data


pred_2 <- predict(glm_1d, newdata = newdata, type = "response")

pred_2

ci
```

Appendix 2

```r
source("ex1/q1.R")

boot.fn <- function(data, index) {
  d <- data[index, ]
  model <- glm(y ~ ., family = poisson, data = d)

  return (coef(model))
}

bootstrap <- function(data, boot.fn, R = 1000) {
  initial_stat <- boot.fn(data, 1:nrow(data))

  stat_length <- length(initial_stat)

  boot_results <- matrix(NA, nrow = R, ncol = stat_length)

  # Perform bootstrap
  for(i in 1:R) {
    indices <- sample(nrow(data), replace = TRUE)

    boot_stat <- boot.fn(data, indices)

    boot_results[i, ] <- boot_stat
  }

  boot_means <- colMeans(boot_results, na.rm = TRUE)
  boot_se <- apply(boot_results, 2, sd, na.rm = TRUE)

  ci_lower <- apply(boot_results, 2, quantile, probs = 0.025, na.rm = TRUE)
  ci_upper <- apply(boot_results, 2, quantile, probs = 0.975, na.rm = TRUE)

  ci_width <- ci_upper - ci_lower

  bias <- boot_means - initial_stat

  # Prepare results
  result <- data.frame(
    bias = bias,                          # Bootstrap bias estimates
    ci_lower=ci_lower,
    ci_upper= ci_upper,
    ci_width = ci_width
    )

  names = colnames(data)[2:ncol(data)]
  names = c("Intercept", names)
  rownames(result) <- names

  return(list(
    result = result,
    boot_results = boot_results
  ))
}

bootstrap_results <- bootstrap(yX, boot.fn, R = 1000)$result
```

```r
options(digits=4)

ci_width_1 = data.frame(ci_with_nb = ci[,2] - ci[,1])

bootstrap_results <- cbind(bootstrap_results, ci_width_1)

print(bootstrap_results)
```

Appendix 5

```
# Newton's method
source("ex1/q1.R")
newton_method <- function(X, y, max_iter = 100, tol = 1e-8) {
  n <- nrow(X)
  p <- ncol(X)

  X <- cbind(1, X)

  beta <- matrix(rep(0, p + 1))
  beta[1,1] <- mean(y)

  mu <- NULL
  hess <- NULL

  count <- 0
  for (iter in 1:max_iter) {
    eta <- X %*% beta
    mu <- exp(eta)
    grad <- t(X) %*% y - t(X) %*% mu

    W <- diag(as.vector(mu))
    hess <- - t(X) %*% W %*% X
    delta <- solve(hess, grad)

    beta_new <- beta - delta

    count <- count + 1
    if (max(abs(delta)) < tol) {
      break
    }


    beta <- beta_new
  }

  return(list(beta = beta,
  hess = hess,
  mu = mu, n = n, p = p, X = X, y = y
  , iter = count, converged = max(abs(delta)) < tol))
}
newton_model <- newton_method(X,y)

newton_model$beta

summary_newton <- function(model) {
  vcov <- solve(-model$hess)
  se <- sqrt(diag(vcov))
  z <- model$beta / se
  p <- 2 * pnorm(-abs(z))

  null_dev <- 2 * sum(model$y * log(model$y / mean(model$y))) -
  2 * sum(model$y - mean(model$y))
resid_dev <- 2 * sum(model$y * log(model$y / model$mu)) -
  2 * sum(model$y - model$mu)
```

```r
  log_likelihood <- sum(model$y * log(model$mu)) -
    sum(model$mu) -
    sum(lgamma(model$y + 1))

  aic <- 2 * length(model$beta) - 2 * log_likelihood

  coef_table <- data.frame("Estimate" = model$beta, "Std. Error" = se,"z value" = z, 'Pr(>|z|)' = p)
    # Format output
  cat("\nCall: Poisson Newton-Raphson Regression\n\n")
  cat("Coefficients:\n")
  print(coef_table)
  cat("\n")
  cat("Null deviance:   ", null_dev, " on ", model$n - 1, " degrees of freedom\n")
  cat("Residual deviance:", resid_dev, " on ", model$n - model$p, " degrees of freedom\n")
  cat("AIC:", aic, "\n")
  cat("Number of iterations:", model$iter, "\n")
  cat("Converged:", model$converged, "\n")
}

summary_newton(newton_model)
```

Appendix 6

```r
source("ex1/q5.R")
newton_method_l2 <- function(X, y, lambda=1, max_iter = 100, tol = 1e-8) {
  n <- nrow(X)
  p <- ncol(X)

  X <- cbind(1, X)

  beta <- matrix(rep(0, p + 1))
  beta[1,1] <- mean(y)

  mu <- NULL
  hess <- NULL

  count <- 0
  for (iter in 1:max_iter) {
    eta <- X %*% beta
    mu <- exp(eta)
    # 0.5 is used to match the regularization term in the glmnet
    grad <- t(X) %*% y - t(X) %*% mu - 0.5 * lambda * c(0, beta[2:length(beta)])

    W <- diag(as.vector(mu))
    hess <- - t(X) %*% W %*% X - 0.5 * lambda * diag(c(0, rep(1, p)))
    delta <- solve(hess, grad)

    beta_new <- beta - delta

    count <- count + 1
    if (max(abs(delta)) < tol) {
      break
    }


    beta <- beta_new
  }

  return(list(beta = beta,
  hess = hess,
  mu = mu, n = n, p = p, X = X, y = y
  , iter = count, converged = max(abs(delta)) < tol))
}


newton_model_l2 <- newton_method_l2(X,y, lambda=10)
summary_newton(newton_model_l2)

library(glmnet)
glmnet_model <- glmnet(X, y, family = "poisson", alpha=0, lambda=10)
coef(glmnet_model)
```

Appendix 7

```
source("ex1/q6.R")

# 10-fold cross validation to estimate the error of lambda=10

predict_newton <- function(model, newdata) {
  # add intercept to newdata
  newdata <- cbind(1, newdata)
  eta <- as.vector(newdata %*% model$beta)
  fit <- eta
  fit <- exp(fit)
  return(fit)
}

k_fold <- function(X, y, lambda, k) {
  index <- sample(nrow(X))
  X <- X[index, ]
  y <- y[index]
  # add intercept to X
  n <- nrow(X)
  fold_size <- n / k
  errors <- numeric(k)
  for (i in 1:k) {
    test_index <- ((i - 1) * fold_size + 1):(i * fold_size)
    test_X <- X[test_index, ]
    test_y <- y[test_index]
    train_X <- X[-test_index, ]
    train_y <- y[-test_index]
  }
  model <- newton_method_l2(train_X, train_y, lambda)
  pred <- predict_newton(model, test_X)
  errors[i] <- mean(abs(pred - test_y))

  return(mean(errors))
}

cat("10-fold cross validation error: ", k_fold(X, y, lambda=10, k=10), "\n")

# use a specific lambda for glmnet
k_fold_result <- cv.glmnet(X, y, family="poisson", alpha=0)
cat("10-fold cross validation best lambda: ", k_fold_result$lambda.min, "\n")

min_lambda <- k_fold_result$lambda.min


k_fold_glmnet <- function(X, y, lambda, k) {
  # randomize X and y
  index <- sample(nrow(X))
  X <- X[index, ]
  y <- y[index]

  # add intercept to X
  n <- nrow(X)
  fold_size <- n / k
  errors <- numeric(k)
  for (i in 1:k) {
```

```
    test_index <- ((i - 1) * fold_size + 1):(i * fold_size)
    test_X <- X[test_index, ]
    test_y <- y[test_index]
    train_X <- X[-test_index, ]
    train_y <- y[-test_index]
  }
  model <- glmnet(train_X, train_y, family="poisson", alpha=0, lambda=lambda)
  pred <- predict(model, test_X, s=lambda)
  errors[i] <- mean(abs(pred - test_y))

  return(mean(errors))
}

cat("10-fold cross validation error: ", k_fold_glmnet(X, y, lambda=min_lambda, k=10), "\n")
cat("10-fold cross validation error: ", k_fold(X, y, lambda=min_lambda, k=10), "\n")
```

Appendix 8

```
source("ex1/q5.R")

n <- nrow(X)
XtX <- t(X)%*%X/n

pm <- function(A,v1=rnorm(ncol(A)),eps=1e-6){
        v1 <- v1/max(abs(v1))
        v0 <- v1+1
        while(max(abs(v1-v0))>eps){
                v0 <- v1
                v1 <- A%*%v1
                v1 <- sign(v1[1])*v1/max(abs(v1))
        }
        v1 <- v1/sqrt(sum(v1*v1))
        list(lam=t(v1)%*%(A%*%v1),v=v1)
}

e <- pm(XtX)
p <- X%*%e$v

pmall <- function(A,eps=1e-6){
        n <- ncol(A)
        V <- matrix(0,n,n)
        lam = rep(0,n)
        for(k in 1:n){
                e <- pm(A,eps=eps)
                lam[k] <- e$lam
                V[,k] <- e$v
                A <- A-lam[k]*outer(V[,k],V[,k])
        }
        list(lam=lam,V=V)
}

E <- pmall(XtX)

principal <- function(X,eps=1e-6){
        A <- t(X)%*%X
        E <- pmall(A,eps=eps)
        X%*%E$V
}

P <- principal(X)


# model 1, y ~ PC1
pc_model1 <- newton_method(as.matrix(P[,1]), y)
summary_newton(pc_model1)

# model 2, y ~ PC1 + PC2
pc_model2 <- newton_method(as.matrix(P[,1:2]), y)
summary_newton(pc_model2)

# model 3, y ~ PC1 + PC2 + PC3
pc_model3 <- newton_method(as.matrix(P[,1:3]), y)
summary_newton(pc_model3)
```

Appendix 9

```
# glmnet cv lasso
source("ex1/q2.R")


cv_lasso <- cv.glmnet(X, y, family="poisson", alpha=1)
cv_lasso$lambda.min

# print the coefficients
coef(cv_lasso, s="lambda.min")

# q2
ridge_cv <- cv.glmnet(X, y, family="poisson", alpha=0)
lasso_cv <- cv.glmnet(X, y, family="poisson", alpha=1)


# get the lambda
lambda_ridge <- ridge_cv$lambda.min
lambda_lasso <- lasso_cv$lambda.min

boot.fn.mle <- function(data, index) {
  d <- data[index, ]
  x <- d[, -1]
  y <- d[, 1]
  model <- glmnet(x, y, family="poisson", alpha=0, lambda=0)
  return(as.vector(coef(model)))
}

boot.fn.ridge <- function(data, index) {
  d <- data[index, ]
  x <- d[, -1]
  y <- d[, 1]
  model <- glmnet(x, y, family="poisson", alpha=0, lambda=lambda_ridge)
  return(as.vector(coef(model)))
}

boot.fn.lasso <- function(data, index) {
  d <- data[index, ]
  x <- d[, -1]
  y <- d[, 1]
  model <- glmnet(x, y, family="poisson", alpha=1, lambda=lambda_lasso)
  return(as.vector(coef(model)))
}


bootstrap_results_mle <- bootstrap(yX, boot.fn.mle, R = 1000)
bootstrap_results_ridge <- bootstrap(yX, boot.fn.ridge, R = 1000)
bootstrap_results_lasso <- bootstrap(yX, boot.fn.lasso, R = 1000)


newton_variance <- apply(bootstrap_results_mle$boot_results, 2, var, na.rm = TRUE)
ridge_variance <- apply(bootstrap_results_ridge$boot_results, 2, var, na.rm = TRUE)
lasso_variance <- apply(bootstrap_results_lasso$boot_results, 2, var, na.rm = TRUE)

newton_variance
ridge_variance
```

`lasso_variance`

Appendix 10

```r
source("ex1/q7.R")

# create validation set randomly
n <- nrow(X)

mse_results <- data.frame(
  newton_mse = numeric(10),
  ridge_newton_mse = numeric(10),
  ridge_mse = numeric(10),
  lasso_mse = numeric(10)
)

mae_results <- data.frame(
  newton_mae = numeric(10),
  ridge_newton_mae = numeric(10),
  ridge_mae = numeric(10),
  lasso_mae = numeric(10)
)

for (i in 1:10) {
  validation_index <- sample(1:n, size = 0.1 * n)
  validation_X <- X[validation_index, ]
  validation_y <- y[validation_index]

  # create training set
  train_X <- X[-validation_index, ]
  train_y <- y[-validation_index]


  # fit all models
  newton_model <- newton_method(train_X, train_y)

  newton_pred <- predict_newton(newton_model, validation_X)
  newton_mse <- mean((newton_pred - validation_y)^2)
  newton_mae <- mean(abs(newton_pred - validation_y))

  ridge_cv_lambda <- cv.glmnet(train_X, train_y, family="poisson", alpha=0)$lambda.min

  ridge_newton_model <- newton_method_l2(train_X, train_y, lambda=ridge_cv_lambda)
  ridge_newton_pred <- predict_newton(ridge_newton_model, validation_X)
  ridge_newton_mse <- mean((ridge_newton_pred - validation_y)^2)
  ridge_newton_mae <- mean(abs(ridge_newton_pred - validation_y))


  ridge_model <- glmnet(train_X, train_y, family="poisson", alpha=0, lambda=ridge_cv_lambda)
  ridge_pred <- predict(ridge_model, validation_X)
  ridge_mse <- mean((ridge_pred - validation_y)^2)
  ridge_mae <- mean(abs(ridge_pred - validation_y))


  lasso_cv_lambda <- cv.glmnet(train_X, train_y, family="poisson", alpha=1)$lambda.min

  lasso_model <- glmnet(train_X, train_y, family="poisson", alpha=1, lambda=lasso_cv_lambda)
  lasso_pred <- predict(lasso_model, validation_X)
  lasso_mse <- mean((lasso_pred - validation_y)^2)
```

```
  lasso_mae <- mean(abs(lasso_pred - validation_y))

  mse_results[i, ] <- c(newton_mse, ridge_newton_mse, ridge_mse, lasso_mse)
  mae_results[i, ] <- c(newton_mae, ridge_newton_mae, ridge_mae, lasso_mae)
}

# calculate the mean of the mse results for each column
mse_results_mean <- colMeans(mse_results)
mse_results_mean

rmse_results_mean <- sqrt(mse_results_mean)


mae_results_mean <- colMeans(mae_results)
mae_results_mean

results <- data.frame(rbind(mse_results_mean, rmse_results_mean, mae_results_mean))
row.names(results) <- c("MSE", "RMSE", "MAE")
results
```